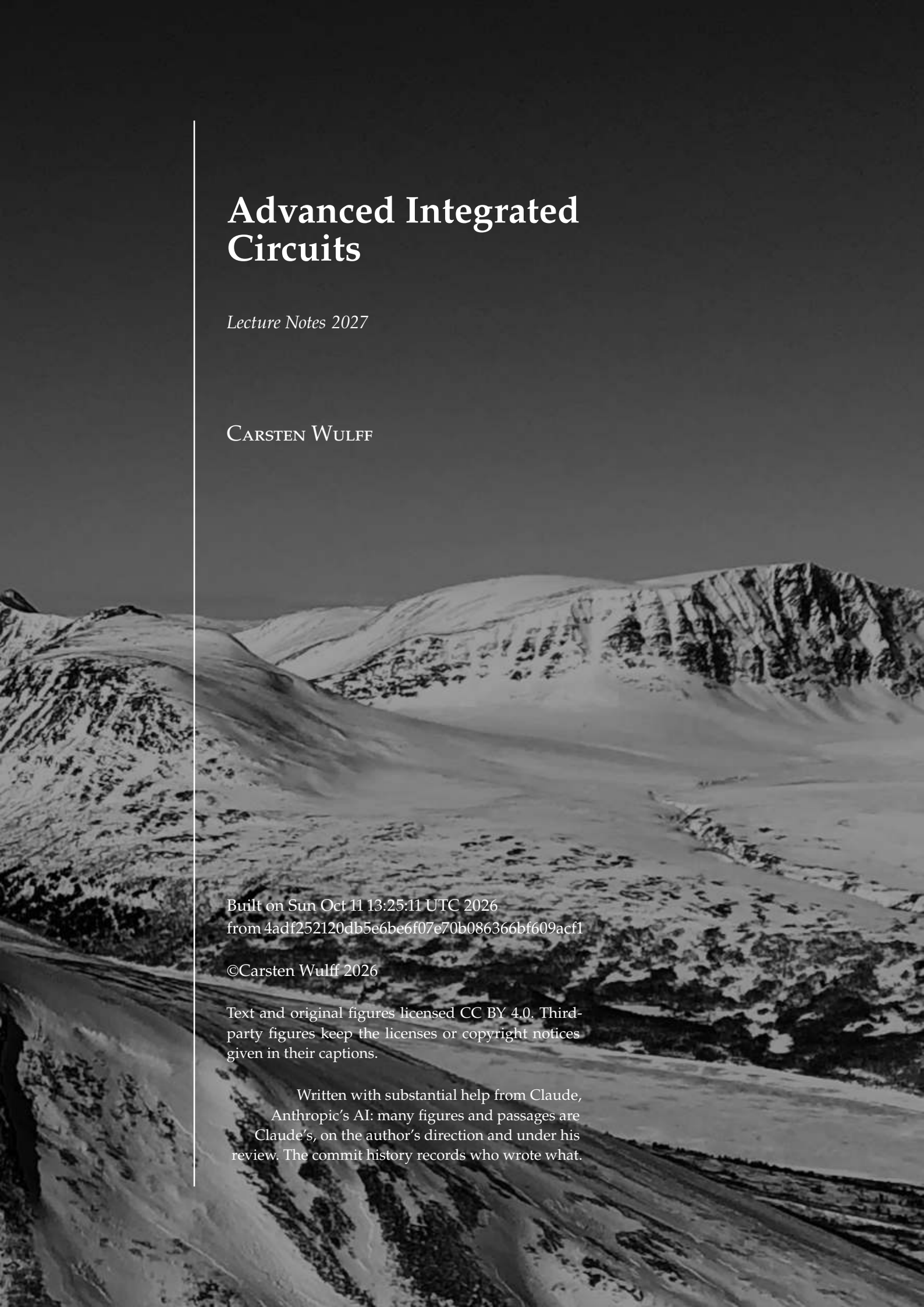


Advanced Integrated Circuits

Lecture Notes 2027

Carsten Wulff

Built on Sun Oct 11 13:25:11 UTC 2026
from 4adf252120db5e6be6f07e70b086366bf609acf1

©Carsten Wulff 2026

Text and original figures licensed CC BY 4.0. Third-party figures keep the licenses or copyright notices given in their captions.

Written with substantial help from Claude, Anthropic's AI: many figures and passages are Claude's, on the author's direction and under his review. The commit history records who wrote what.

Contents

Contents **3**

Preface **1**

Conventions **3**

1 Background **5**

2 The Story of Jayn **7**

3 Introduction **11**
3.1 Who 11
3.2 How I see our roles 12
3.3 I want you to learn the skills necessary to make your own ICs 13
3.4 There will always be analog circuits, because the real world is analog 14
3.5 Will you tape-out an IC? 14
3.5.1 What the team needs to know to design ICs 15
3.5.2 Zen of IC design (stolen from Zen of Python) 16
3.5.3 IC design mantra 16
3.5.4 Analog Design Process 16
3.6 My Goal 17
3.7 Syllabus 18
3.8 Software 18
3.9 Summary 18

4 How to Achieve Excellence **21**
4.1 Ruthless Focus on Real Value 21
4.2 Chase Down What's Off 21
4.3 Root Causes or Nothing 22
4.4 Constraint Is a Feature 22
4.5 Kill Bureaucracy Aggressively 22
4.6 Support the People Doing the Work 22
4.7 Ask Questions Relentlessly 23
4.8 Optimize for Truth, Not Consensus 23
4.9 Treat the Customer Timeline as Holy 23
4.10 Separate Identity from Job 23
4.11 Control Emotion, Keep Passion 24
4.12 Treat Work as a Game, and Play to Win 24
4.13 Appendix 24

5 A Refresher **25**
5.1 There are standard units of measurement 25
5.2 Electrons 26

5.3 Probability . . . 27
5.4 Uncertainty principle . . . 28
5.5 States as a function of time and space . . . 28
5.6 Allowed energy levels in atoms . . . 29
5.7 Allowed energy levels in solids . . . 29
5.8 Silicon Unit Cell . . . 30
5.9 Band structure . . . 31
5.10 Valence band and Conduction band . . . 31
5.11 Fermi level . . . 32
5.12 Metals . . . 33
5.13 Insulators . . . 33
5.14 Semiconductors . . . 34
5.15 Band diagrams . . . 34
5.16 Density of electrons/holes . . . 34
5.17 Fields . . . 35
5.18 Permittivity and Permeability . . . 36
5.19 Quantum electrodynamics . . . 36
5.20 Voltage . . . 36
5.21 Current . . . 37
5.22 Drift current . . . 37
5.23 Diffusion current . . . 38
5.24 Why are there two currents? . . . 38
5.25 Currents in a semiconductor . . . 39
5.26 Resistors . . . 39
5.27 Capacitors . . . 40
5.28 Inductors . . . 40
5.29 Summary . . . 40
5.30 Would you like to know more? . . . 40

6 Fields **41**
6.1 The four equations, in words . . . 41
6.2 Currents always run in loops . . . 42
6.3 The capacitor falls out . . . 43
6.4 The inductor falls out . . . 43
6.5 Every wire is both . . . 44
6.6 The antenna does not radiate . . . 45
6.7 What this buys you on-chip . . . 47
6.8 Summary . . . 47
6.9 Would you like to know more? . . . 48

7 Diodes **49**
7.1 Why . . . 49
7.2 Silicon . . . 49
7.3 Intrinsic carrier concentration . . . 51
7.4 It's all quantum . . . 52
7.4.1 Density of states . . . 54
7.4.2 How to think about electrons (and holes) 56
7.5 Doping . . . 57

7.6 PN junctions 58
7.6.1 Built-in voltage 59
7.6.2 Current 60
7.6.3 Forward voltage temperature dependence 61
7.6.4 Current proportional to temperature . . . 63
7.7 Reverse leakage 64
7.7.1 Diffusion leakage 65
7.7.2 Generation in the depletion region 66
7.7.3 Plasma charging currents 67
7.7.4 Antenna ndiode, 200 K to 1000 K 68
7.7.5 Wires below 1 mm 70
7.8 Equations aren't real 71
7.9 Summary 72
Would you like to know more? 72

8 MOSFETs **75**
8.1 Metal Oxide Semiconductor 75
8.2 Field Effect 77
8.3 Analog transistors in the books 81
8.4 Transistors in weak inversion 83
8.5 Transistors in strong inversion 86
8.6 Introduction to behavior 89
8.6.1 Drain Source Current 89
8.6.2 Gate-source voltage 90
8.6.3 Inversion level 91
8.6.4 Drain source voltage 95
8.6.5 Strong inversion 95
8.6.6 Low frequency model 97
8.6.7 Transconductance 98
8.6.8 Intrinsic gain 99
8.6.9 Body effect 100
8.6.10 High frequency model 101
8.6.11 Be careful with Cgd (blame Miller) 102
8.6.12 Transit frequency 103
8.7 Weak inversion 104
8.8 Velocity saturation 106
8.8.1 Square law model 107
8.8.2 Mobility Degradation 108
8.8.3 What about holes (PMOS) 108
8.9 OTHER 109
8.9.1 Drain induced barrier lowering (DIBL) . . 110
8.9.2 Well Proximity Effect (WPE) 111
8.9.3 Stress effects 111
8.9.4 Gate current 112
8.9.5 Hot carrier injection 113
8.9.6 Channel initiated secondary-electron (CHISEL) 113

8.10 Variability . . . 114
8.10.1 Voltage variation . . . 115
8.10.2 Systematic variations . . . 115
8.10.3 Process variations . . . 116
8.10.4 Process corners . . . 116
8.10.5 Fix process variation . . . 117
8.10.6 Temperature variation . . . 117
8.10.7 It depends on V_{DD} . . . 117
8.10.8 How do we fix temperature variation? . . . 118
8.10.9 Random Variation . . . 118
8.10.10 Pelgrom's law [2] . . . 118
8.10.11 Transistors with same V_{GS} [3] . . . 119
8.10.12 What else can we do? . . . 120
8.10.13 Noise equations . . . 121
8.11 Summary . . . 122
8.12 Would you like to know more? . . . 122

9 Noise **125**
9.1 The noise floor . . . 125
9.2 Three things that come in pieces . . . 126
9.3 Statistics . . . 126
9.4 Average Power . . . 126
9.5 Noise Spectrum . . . 127
9.6 Probability Distribution . . . 129
9.7 PSD of a white noise source . . . 129
9.8 Summing noise sources . . . 130
9.9 Signal to Noise Ratios . . . 131
9.10 Noise figure and Friis formula . . . 132
9.11 Spectral Density . . . 133
9.11.1 Definition of Spectral Density . . . 133
9.11.2 Sources of Confusion . . . 134
9.11.3 Example: Thermal Noise . . . 135
9.11.4 Einstein: The source . . . 136
9.12 Where white noise comes from . . . 136
9.12.1 Thermal noise is Planck . . . 136
9.12.2 Shot noise is the electron charge . . . 137
9.13 Where flicker noise comes from . . . 137
9.13.1 Schrödinger . . . 137
9.13.2 Fermi-Dirac . . . 138
9.14 One trap . . . 138
9.14.1 What a trap is . . . 138
9.14.2 The channel electron has a tail . . . 138
9.14.3 Tunnelling sets the time constant . . . 140
9.14.4 Dirac's golden rule . . . 140
9.14.5 The lattice must move . . . 141
9.14.6 Capture and emission . . . 142

9.15 What one trap looks like . . . 143
9.15.1 Not one Gaussian . . . 144
9.15.2 Two states, one Lorentzian . . . 144
9.15.3 The Lorentzian, measured . . . 145
9.15.4 How big is one step? . . . 145
9.15.5 How many traps? . . . 146
9.16 Which transistor in GR06? . . . 147
9.16.1 How GR06 works . . . 147
9.16.2 Is it the supply? . . . 148
9.16.3 Memoryless . . . 149
9.16.4 The suspects . . . 149
9.16.5 How to tell . . . 150
9.17 Many traps . . . 151
9.17.1 McWhorter: uniform in depth is $1/f$ . . . 151
9.17.2 Back to the design equation . . . 152
9.17.3 The model card knows . . . 152
9.18 Summary . . . 153
9.19 Would you like to know more? . . . 153

10 Circuits **157**
10.1 Circuits . . . 157
10.2 Transistor size and bias point . . . 157
10.3 Transistor sizing strategy . . . 161
10.3.1 Option 1: Full freedom . . . 161
10.3.2 Option 2: Constrained . . . 161
10.4 My circuit does not work, why???????????????? . . . 162
10.5 Current Mirrors . . . 163
10.5.1 Normal current mirror . . . 164
10.5.2 Source degenerated current mirror . . . 165
10.5.3 Cascoded current mirror . . . 167
10.5.4 Active cascodes . . . 168
10.6 Amplifiers . . . 169
10.6.1 Source follower . . . 169
10.6.2 Why use a source follower? . . . 172
10.6.3 Common gate . . . 173
10.6.4 Common source . . . 177
10.6.5 Why common source? . . . 178
10.6.6 Differential pair . . . 179
10.6.7 Diff pairs are cool . . . 180
10.7 Summary . . . 180
10.8 Would you like to know more? . . . 181

11 OTAs **183**
11.1 OTAs . . . 183
11.2 The headroom budget . . . 183
11.3 Five transistor OTA . . . 184
11.4 Current mirror OTA . . . 185
11.5 Two stage (Miller) OTA . . . 186

11.6 Folded cascode . . . 187
11.7 Inverter based OTAs . . . 188
11.8 Bulk driven input . . . 189
11.9 Fully differential . . . 190
11.9.1 Common mode feedback . . . 190
11.9.2 Sensing the common mode . . . 191
11.9.3 Switched capacitor CMFB . . . 192
11.10 Bias circuits . . . 192
11.11 From OTA to op amp . . . 193
11.11.1 A complete OTA, sized . . . 194
11.12 Dynamic amplifiers . . . 195
11.13 Choosing . . . 196
11.14 Verifying the OTA . . . 196
11.15 Summary . . . 197
11.16 Would you like to know more? . . . 198

12 Integrated Passives **199**
12.1 Metal in ICs is not wire in schematic . . . 199
12.2 Resistors . . . 200
12.2.1 Polysilicon . . . 200
12.2.2 Diffusion . . . 201
12.2.3 Metal . . . 202
12.3 Capacitors . . . 202
12.3.1 What is S, M, L, XL on a chip? . . . 202
12.3.2 Metal-Oxide-Metal finger capacitors . . . 203
12.3.3 MOS capacitors . . . 203
12.3.4 Varactors . . . 205
12.4 Inductors . . . 205
12.5 Variation in passives . . . 206
12.6 Relative precision . . . 207
12.7 Summary . . . 207
12.8 Would you like to know more? . . . 208

13 The Tools **209**
13.1 Tools . . . 209
13.1.1 Setup WSL (Applicable for Windows users) 209
13.1.2 Setup public key towards github . . . 210
13.1.3 Provide git with author identity . . . 210
13.1.4 Get AICEX and setup your shell . . . 210
13.1.5 On systems with python3 > 3.12 . . . 211
13.1.6 Install Tools . . . 211
13.1.7 Install cicconf . . . 212
13.1.8 Install cicsim . . . 212
13.1.9 Install cicpy . . . 212
13.1.10 Setup your ngspice settings . . . 213
13.2 Check that magic and xschem work . . . 213
13.3 Summary . . . 213
13.4 Would you like to know more? . . . 213

14 Sky130nm tutorial **215**
14.1 Create the IP 215
14.2 The file structure 215
14.2.1 Github workflows 215
14.2.2 Configuration files 216
14.2.3 Design files 216
14.2.4 Simulations 217
14.2.5 The work 217
14.3 Github setup 217
14.4 Start working 217
14.4.1 Edit README.md 217
14.4.2 Familiarize yourself with the Makefile and make 218
14.5 Draw Schematic 218
14.5.1 Add Ports 218
14.5.2 Add transistors 218
14.5.3 Netlist schematic 219
14.6 Typical corner SPICE simulation 219
14.6.1 Setup simulation environment 220
14.6.2 Familiarize yourself with the simulation folder 220
14.6.3 Modify default testbench (tran.spi) 220
14.6.4 Modify measurements (tran.meas) 221
14.6.5 Modify result specification (tran.yaml) . . 221
14.6.6 Check waveforms 222
14.7 All corners SPICE simulations 222
14.7.1 Remove Vh and Vl corners (Makefile) . . 223
14.7.2 Run all corners 223
14.7.3 Get creative with python 223
14.7.4 Generate simulation summary 224
14.7.5 Viewing results without GUI browser . . 224
14.7.6 Think about the results 224
14.8 Draw Layout 225
14.8.1 Magic VLSI 225
14.8.2 Add transistors 226
14.8.3 Place devices 226
14.8.4 Add Ground 227
14.8.5 Route Gates 228
14.8.6 Drain of M2 228
14.8.7 Add labels 229
14.9 Layout verification 229
14.10 Extract layout parasitics 230
14.11 Simulate with layout parasitics 230
14.11.1 Typical simuation 231
14.11.2 Corners 231
14.11.3 Simulation summary 231
14.12 Make documentation 231

14.13 Edit info.yaml . . . 232
14.14 Setup github pages . . . 232
14.15 Frequently asked questions . . . 232
14.16 Summary . . . 233

15 The Project **235**
15.1 The challenge . . . 235
15.1.1 Systems-on-chip have complex regulator systems . . . 236
15.1.2 Leakage current varies orders of magnitude over temperature . . . 236
15.1.3 Why we would like to know the temperature on die . . . 237
15.1.4 How to measure temperature? . . . 237
15.2 The Project . . . 238
15.2.1 Specification . . . 239
15.2.2 Figure of Merit . . . 240
15.2.3 Grade . . . 240
15.2.4 Milestone 0: The tutorial . . . 241
15.2.5 Milestone 1: The bandgap . . . 241
15.2.6 Milestone 2: The oscillator . . . 242
15.2.7 Milestone 3: The measurement . . . 243
15.2.8 Milestone 4 (Optional): The physical design 244
15.2.9 Milestone 5: The Report . . . 244
15.2.10 Milestone 6 (Optional): The Tapeout . . . 245
15.3 Summary . . . 245
15.4 Would you like to know more? . . . 245

16 Measured Silicon **247**
16.1 Measured Silicon . . . 247
16.1.1 The same physics, read out two ways . . . 247
16.2 What the silicon does . . . 249
16.2.1 Against a real reference . . . 249
16.2.2 What calibration buys . . . 251
16.3 The clock in the loop . . . 252
16.3.1 GR07's output is quantised, and noise is what rescues it . . . 252
16.3.2 When averaging stops helping . . . 254
16.3.3 Two sensors, one event . . . 255
16.4 One trap . . . 257
16.4.1 GR06 does not drift, it switches . . . 257
16.5 Summary . . . 259
16.6 Would you like to know more? . . . 259

17 IC and ESD **261**
17.1 What blocks must our IC include? . . . 261
17.1.1 Supply . . . 262

17.2 Electrostatic Discharge 264
17.2.1 Human body model (HBM) 265
17.2.2 Charged device model (CDM) 266
17.3 An HBM ESD zap example 268
17.4 The grounded gate NMOS 271
17.5 But I just want a digital input, what do I need? . 274
17.5.1 Input buffer 275
17.6 Latch-up 276
17.6.1 Electron injection 277
17.7 Summary 279
17.8 Would you like to know more? 279

18 References and bias **281**
18.1 Routing 282
18.2 Bandgap voltage reference 284
18.2.1 A voltage complementary to temperature (CTAT) 284
18.2.2 A current proportional to temperature (PTAT) 286
18.2.3 How to combine a CTAT with a PTAT ? . 287
18.2.4 Widlar reference 289
18.2.5 Brokaw reference 290
18.2.6 Low voltage bandgap 294
18.2.7 Curvature correction 297
18.2.8 MOS references 299
18.2.9 The native threshold is set when the ingot is grown 301
18.2.10 FD-SOI moves the problem, it does not remove it 302
18.2.11 The back gate is a trimming knob, not a reference 303
18.3 Bias 303
18.3.1 Voltage to current conversion 303
18.3.2 GM Cell 304
18.3.3 Every one of these loops can fail to start . 306
18.4 Summary 308
18.5 Would you like to know more? 309

19 Analog frontend and filters **311**
19.1 Introduction 311
19.2 Filters 314
19.2.1 First order filter 314
19.2.2 Second order filter 315
19.2.3 How do we implement the filter sections? 316
19.3 Gm-C 316
19.3.1 Differential Gm-C 317
19.3.2 Finding a transconductor 319

19.4 Active-RC . . . 320
19.4.1 General purpose first order filter . . . 320
19.4.2 General purpose biquad . . . 323
19.5 The OTA is not ideal . . . 324
19.6 Example circuit . . . 325
19.7 My favorite OTA . . . 327
19.8 Summary . . . 329
19.9 Would you like to know more? . . . 329

20 Digital to analog conversion 331
20.1 Resistor based DACs . . . 333
20.2 DAC errors . . . 335
20.3 DAC complexity . . . 337
20.4 Binary scaled DACs . . . 340
20.4.1 Binary coding . . . 342
20.4.2 Thermometer encoding . . . 344
20.5 Current mode DACs . . . 345
20.6 Summary . . . 346
20.7 Would you like to know more? . . . 346

21 Switched-Capacitor Circuits 349
21.1 Active-RC . . . 349
21.2 Gm-C . . . 351
21.3 Switched capacitor . . . 351
21.3.1 An example SC circuit . . . 355
21.4 Discrete-Time Signals . . . 357
21.4.1 The mathematics . . . 358
21.4.2 Python discrete time example . . . 359
21.4.3 Aliasing, bandwidth and sample rate theory 361
21.4.4 Z-transform . . . 362
21.4.5 Pole-Zero plots . . . 363
21.4.6 Z-domain . . . 364
21.4.7 First order filter . . . 365
21.4.8 Second order filter . . . 366
21.4.9 Finite-impulse response(FIR) . . . 368
21.5 Switched-Capacitor . . . 369
21.5.1 Switched capacitor gain circuit . . . 371
21.5.2 Switched capacitor integrator . . . 372
21.5.3 Noise . . . 374
21.5.4 Sub-circuits for SC-circuits . . . 376
21.5.5 Example . . . 381
21.6 Summary . . . 382
21.7 Would you like to know more? . . . 382

22 Oversampling and Sigma-Delta ADCs 385
22.1 ADC state-of-the-art . . . 385
22.1.1 What makes a state-of-the-art ADC . . . 387
22.1.2 High resolution FOM . . . 394

22.2 Quantization 395
22.2.1 Signal to Quantization noise ratio 399
22.2.2 Understanding quantization 400
22.2.3 Why you should care about quantization noise 403
22.3 Oversampling 404
22.3.1 Noise power 404
22.3.2 Signal power 405
22.3.3 Signal to Noise Ratio 405
22.3.4 Signal to Quantization Noise Ratio 406
22.3.5 Python oversample 407
22.4 Noise Shaping 408
22.4.1 The magic of feedback 409
22.4.2 Sigma-delta principle 410
22.4.3 Signal transfer function 412
22.4.4 Noise transfer function 412
22.4.5 Combined transfer function 412
22.5 First-Order Noise-Shaping 413
22.5.1 SQNR and ENOB 415
22.6 Examples 416
22.6.1 Python noise-shaping 416
22.6.2 The wonderful world of SD modulators . 421
22.7 Summary 426
22.8 Would you like to know more? 427

23 Voltage regulation **429**
23.1 Voltage source 429
23.1.1 Why transistors die 430
23.1.2 Core voltage 432
23.1.3 IO voltage 433
23.1.4 Supply planning 433
23.2 Linear Regulators 434
23.2.1 PMOS pass-fet 434
23.2.2 NMOS pass-fet 435
23.2.3 Control of pass-fet 436
23.3 Switched Regulators 438
23.3.1 Principles of switched regulators 438
23.3.2 Inductive DC/DC converter details 441
23.3.3 Pulse width modulation (PWM) 443
23.3.4 Real world use 446
23.3.5 Pulsed Frequency Mode (PFM) 446
23.4 Summary 449
23.5 Would you like to know more? 449
23.5.1 Linear regulators 449
23.5.2 DC-DC converters 449

24 Clocks and PLLs **453**
24.1 Why clocks? . . . 453
24.1.1 A customer story . . . 453
24.1.2 Frequency . . . 455
24.1.3 Noise . . . 455
24.1.4 Stability . . . 455
24.1.5 Conclusion . . . 455
24.2 A typical System-On-Chip clock system . . . 456
24.2.1 32 MHz crystal . . . 456
24.2.2 32 KiHz crystal . . . 456
24.2.3 PCB antenna . . . 456
24.2.4 DC/DC inductor . . . 457
24.3 PLL . . . 459
24.3.1 Integer PLL . . . 459
24.3.2 Fractional PLL . . . 460
24.3.3 Modulation in PLLs . . . 460
24.4 PLL Example . . . 461
24.4.1 Loop gain . . . 463
24.4.2 Controlled oscillator . . . 463
24.4.3 Phase detector and charge pump . . . 465
24.4.4 Loop filter . . . 467
24.4.5 Divider . . . 468
24.4.6 Loop transfer function . . . 468
24.5 Summary . . . 472
24.6 Would you like to know more? . . . 472

25 Oscillators **473**
25.1 Atomic clocks . . . 473
25.1.1 Microchip 5071B Cesium Primary Time and Frequency Standard . . . 473
25.1.2 Rubidium standard . . . 474
25.2 Crystal oscillators . . . 475
25.2.1 Impedance . . . 476
25.2.2 Circuit . . . 478
25.2.3 Temperature behavior . . . 481
25.3 Controlled Oscillators . . . 481
25.3.1 Ring oscillator . . . 481
25.3.2 Capacitive load . . . 483
25.3.3 Realistic . . . 484
25.3.4 Digitally controlled oscillator . . . 486
25.3.5 Differential . . . 486
25.3.6 LC oscillator . . . 487
25.4 Relaxation oscillators . . . 489
25.5 Summary . . . 490
25.6 Would you like to know more? . . . 490
25.6.1 Crystal oscillators . . . 490
25.6.2 CMOS oscillators . . . 491

26 Low Power Radio **493**
26.1 Data Rate . . . 493
26.1.1 Data . . . 493
26.1.2 Rate . . . 494
26.1.3 Data Rate . . . 494
26.2 Carrier Frequency & Range . . . 494
26.2.1 ISM (industrial, scientific and medical) bands 494
26.2.2 Antenna . . . 495
26.2.3 Range (Friis) . . . 496
26.2.4 Range (Free space) . . . 496
26.2.5 Range (Real world) . . . 497
26.3 Power supply . . . 497
26.3.1 Battery . . . 497
26.4 Decisions . . . 498
26.4.1 Modulation . . . 498
26.4.2 BPSK . . . 499
26.4.3 Single carrier, or multi carrier? . . . 505
26.4.4 Use a Software Defined Radio . . . 506
26.5 Bluetooth . . . 507
26.5.1 Bluetooth Basic Rate/Extended Data rate 508
26.5.2 Bluetooth Low Energy . . . 508
26.6 Algorithm to design state-of-the-art LE radio . . 510
26.6.1 LNTA . . . 511
26.6.2 MIXER . . . 512
26.6.3 AAF . . . 513
26.6.4 ADC . . . 514
26.6.5 AD-PLL . . . 516
26.6.6 Baseband . . . 517
26.7 What do we really want, in the end? . . . 517
26.8 Summary . . . 518
26.9 Would you like to know more? . . . 519

27 Energy Sources **521**
27.1 Thermoelectric . . . 523
27.1.1 Radioisotope Thermoelectric generator . . 527
27.1.2 Thermoelectric generators . . . 528
27.2 Photovoltaic . . . 529
27.3 Piezoelectric . . . 532
27.4 Electromagnetic . . . 533
27.4.1 “Near field” harvesting . . . 533
27.4.2 Ambient RF Harvesting . . . 534
27.5 Triboelectric generator . . . 536
27.6 Comparison . . . 537
27.7 Summary . . . 539
27.8 Would you like to know more? . . . 539

28 Analog SystemVerilog **541**
28.1 Digital simulation . . . 542
28.1.1 Digital Simulators . . . 544
28.2 Transient analog simulation . . . 545
28.3 Mixed signal simulation . . . 547
28.4 Analog SystemVerilog Example . . . 548
28.4.1 TinyTapeout TT06_SAR . . . 548
28.4.2 SAR operation . . . 549
28.5 Summary . . . 552
28.6 Would you like to know more? . . . 552

29 How to write a project report **555**
29.1 Why . . . 555
29.2 On writing English . . . 555
29.2.1 Shorter is better . . . 555
29.2.2 Be careful with adjectives . . . 556
29.2.3 Use paragraphs . . . 556
29.2.4 Don't be afraid of I . . . 556
29.2.5 Transitions are important . . . 556
29.2.6 However, is not a start of a sentence . . . 557
29.3 Report Structure . . . 557
29.3.1 Introduction . . . 557
29.3.2 Theory . . . 558
29.3.3 Implementation . . . 558
29.3.4 Result . . . 558
29.3.5 Discussion . . . 558
29.3.6 Future work . . . 559
29.3.7 Conclusion . . . 559
29.3.8 Appendix . . . 559
29.4 Checklist . . . 559
29.5 Summary . . . 560
29.6 Would you like to know more? . . . 560

30 Layout Generation **563**
30.1 Layout . . . 563
30.2 Setup . . . 563
30.3 CICPY . . . 563
30.4 Placement . . . 564
30.5 Summary . . . 566
30.6 Would you like to know more? . . . 566

31 Thoughts and Advice **567**
31.1 Advice . . . 567
31.1.1 What the team needs to know to design ICs 568
31.1.2 Zen of IC design (stolen from Zen of Python) 569
31.1.3 IC design mantra . . . 569
31.1.4 Analog Design Process . . . 569

31.2 Stuff to ponder 570
31.2.1 AC open, DC closed switch 570
31.3 Summary 576
31.4 Would you like to know more? 576

32 SPICE **577**
32.1 SPICE 577
32.2 Simulation Program with Integrated Circuit Emphasis 577
32.2.1 Today 578
32.2.2 But 578
32.2.3 Sources 579
32.2.4 Passives 580
32.2.5 Transistor Models 580
32.2.6 Transistors 581
32.2.7 Foundries 582
32.3 Find right transistor sizes 582
32.3.1 Use unit size transistors for analog design 583
32.3.2 What about gm/Id ? 583
32.3.3 Characterize the transistors 584
32.4 More information 584
32.5 Analog Design 584
32.6 Demo 584
32.7 Summary 585
32.8 Would you like to know more? 585

33 Mixed Signal Simulation in NGSPICE **587**
33.1 Mixed Signal Simulation in ngspice 587
33.2 Digital simulation 587
33.3 Transient analog simulation 588
33.4 Demo 590
33.5 The circuit 590
33.6 The digital code 590
33.7 Compile RTL 591
33.8 Import object into SPICE file 591
33.9 Import in testbench 592
33.10 Override default digital output voltage 592
33.11 Running 592
33.12 Summary 592
33.13 Would you like to know more? 593

34 Analog Design **595**
34.1 Checklist 595
34.1.1 Specification 595
34.1.2 Design 596
34.1.3 Tapeout 596
34.2 Schematic rules 596
34.2.1 Only uppercase names allowed 597

34.3 Layout rules . . . 598
34.3.1 Limit the amount of transistor Width's and Length's that you use . . . 598
34.4 Would you like to know more? . . . 599

35 The cic tools **601**
35.1 cicsim - simulations you can rerun . . . 601
35.2 ciccreator - the layout compiler . . . 602
35.3 cicpy and cicconf - the glue . . . 602
35.4 cicwave - looking at the waves . . . 603
35.5 cictikz - the figures of this book . . . 603
35.6 Summary . . . 603
35.7 Would you like to know more? . . . 603

36 CMOS Logic **605**
36.1 CMOS Logic . . . 605
36.2 Analog transistor to digital transistor . . . 605
36.3 CMOS static logic assumptions . . . 607
36.4 Don't break rules unless you know exactly why it will be OK . . . 609
36.5 Logic cells . . . 609
36.5.1 CMOS static logic is inverting . . . 609
36.5.2 Rules for inverting logic . . . 612
36.6 SR-Latch . . . 614
36.7 D-Latch (16 transistors) . . . 615
36.8 Other logic cells . . . 616
36.9 AOI22: and or invert . . . 617
36.10 Tristate inverter . . . 618
36.11 Mux . . . 619
36.12 There are other types of logic . . . 620
36.13 Speed . . . 621
36.14 Flip-flops and speed . . . 622
36.15 Timing analysis . . . 625
36.16 Timing analysis tools . . . 626
36.16.1 What is timing analysis . . . 626
36.17 Every gate must be simulated to provide behavior over input transition and load capacitance . . . 628
36.18 All analog blocks must have associated liberty file to describe behavior and timing paths *If you integrate analog into digital top flow* . . . 628
36.19 Gate Delay . . . 628
36.19.1 Delay definitions . . . 628
36.20 Delay estimation . . . 629
36.20.1 Inverter with inverter load . . . 629
36.21 Elmore Delay . . . 629
36.22 Delay components . . . 629
36.23 Modern IC timing analysis requires computers with advanced programs . . . 631

36.24 Best number of stages 631
36.25 Which has shortest delay? 631
36.26 Trends 633
36.27 Attack vector 634
36.28 Pick two 639
36.29 Power 639
36.30 What is power? 639
36.31 Power dissipated in a resistor 639
36.32 Charging a capacitor to VDD 640
36.33 Energy to charge a capacitor to a voltage VDD . 640
36.34 Discharging a capacitor to 0 640
36.35 Power consumption of digital circuits 641
36.36 Sources of power dissipation in CMOS logic . . . 641
36.37 Switching Power in logic gates 641
36.38 Switching probability 642
36.39 Strategies to reduce dynamic power 645
36.39.1 Stop clock 645
36.39.2 Stop activity 645
36.39.3 Reduce frequency 646
36.39.4 Turn off power supply 647
36.40 Wires 648
36.41 Wire geometry 648
36.42 Metal stack 648
36.43 Metal routing rules on IC 649
36.44 Modeling Interconnect 649
36.45 Lumped model 649
36.46 Wire resistance 649
36.47 Most wires: Copper 650
36.48 Contacts 650
36.49 Wire Capacitance 650
36.50 FSM 651
36.51 Mealy machine 651
36.52 Moore machine 651
36.53 Mealy versus Moore 651
36.53.1 dicex/sim/counter_sv/counter.v 652
36.54 Battery charger FSM 652
36.54.1 Li-Ion batteries 652
36.54.2 Battery charger - Inputs 653
36.54.3 Battery charger - States 654
36.55 Summary 657
36.56 Would you like to know more? 658

37 FAQ **659**
37.1 Frequently asked questions 659
37.1.1 I get asked for password when I access github 659

38 Equations **661**
38.1 Fundamental physics 661

38.2 Semiconductors 662
38.3 Diode 663
38.4 MOSFET 663
38.5 Noise 665
38.6 Circuits 666
38.7 References and bias 666
38.8 Filters 667
38.9 Switched capacitor 667
38.10 Data converters 667
38.11 Voltage regulation 668
38.12 PLL 668
38.13 Oscillators 669
38.14 Radio 669
38.15 Would you like to know more? 669

39 References **671**

Index **683**

Preface

These are the lecture notes for *TFE4188 Advanced Integrated Circuits* at the Norwegian University of Science and Technology, grown over years of teaching into something that behaves like a book. The intended reader is a student who has met circuits before and now wants to design integrated ones: analog on a modern CMOS process, with the digital world pressing in from every side.

The book has two kinds of chapters. The *refresher* chapters - physics, fields, diodes, MOSFETs, circuits, OTAs, passives, noise, logic, layout - rebuild the foundations the later chapters stand on; read them in order the first time, and raid them afterwards. The *technical* chapters - ESD, references, filters, converters, regulators, clocks, oscillators, radio - each take one job an integrated circuit must do and follow it from principle to schematic. Every chapter ends with a one-page summary and a reading list.

Everything here runs on an open source flow: xschem, ngspice, magic, and the cic tools, on the open SKY130 process. Every schematic figure can be simulated, every simulation rerun from a shell command, and the source of the book itself - text, figures, build - lives in a public git repository. If you find an error, the repository is where to say so.

Parts of this book were written by Claude, Anthropic's AI, on my direction and under my review; the title page says so, AI-written chapters carry a byline, and the repository's commit history records precisely who wrote what.

Carsten Wulff, Trondheim, 2026

Conventions

The figures in this book follow one visual language, so colour and line style carry meaning everywhere. The palette is chosen as a luminance ladder: printed in black and white, every colour lands on a distinct gray.

- **Black** is the circuit: wires, devices, and the house symbols. One line width throughout, one arrow tip.
- **Red** marks the subject - the device or path the figure is about. In the device cartoons red is also the hole, and red regions are p-type.
- **Blue** carries signals and fields. In the device cartoons blue is the electron, and blue regions are n-type.
- **Green** marks equivalence - two drawings of the same thing - and mechanical annotations.
- **Teal** is the gate oxide, wherever it appears.
- **Dashed or arrowed strokes** are annotation, never wires.
- **Wires that cross without a dot are not connected**; every junction carries a dot.

Device sizes quoted as `24F4F` read as width 24 and length 4 minimum gate lengths, so a schematic ports between processes by re-reading F. Some numerical estimates are deliberately rough - engineering hand-waving is a skill this book practices openly, and the text says so where it happens.

1 Background

In the spring of 2026 I lectured Advanced Integrated Circuits for the fifth time. I have an inherent need to make things better, and the course is no different.

In 2022 I noticed that little of what I had on slides, or said in lectures, made it into the student brain. That annoyed me, and I realized that things needed to change.

In 2023 I moved to complete open source project, and the project was without grade. There should have been a grade on the project.

I feel the lectures have gotten better. I did not take attendance in 2023, but there were 19 students that took the exam in 2024. I don't have all the dates, but an average attendance of 76 %.

In 2024 I finally felt I achieved a balance. I spent Thursday's preparing for the lecture, writing these notes, making a YouTube video (so I'll remember next year what I wanted to talk about). I passed 1k subscribers on Youtube. Friday's I had the lecture and the group work.

For the group work I forced students into groups, and I forced that they for the first 5-10 minutes do a check-in. That I need to do next year too.

For the check in, they had go around in the group and answer one of the following questions:

- What is one thing that is going on in your life (personal or professional)?
- What is one thing that you're grateful for right now?
- What is something funny that happened?

The check-in led to excellent team work for those students that showed up.

In 2025 there were 23 students that took the exam, however, 26 different students showed up to the lectures (more than a few times). The average attendance was around 80 %.

I made a few tweaks. One change was the grading of the project, I used github actions to do the GDS,DRC,LVS,SIM and docs. The grading did not really work that well, although, it was a good way to get students to get the designs correct on github. The first milestones with the sim and the doc did not work. The last milestone actions worked well.

For the first time, two groups reached tapeout https://github.com/wulffern/jnw-tt-2025 !

In 2026 had a record low number of students. I suspect it was because the course collided with a two other courses. There were 13 students that handed in the project. Attendance to the lectures was only 51 percent. The lectures were at 08:15 on Friday's, so that could be a contributing factor. For project and lecture hours the attendance was above 65 % (forgot to register some weeks)

For 2027 I should do the following changes:

- ☐ People go home the Friday before easter
- ☐ In 2026 AIC collided with antenna technique & digital communcation. Review timeplan next time
- ☐ Use ISSCC template to force students to be concise in their writing. Shorter is probably more difficult for AI also.
- ☐ Report needs to include AI declaration
- ☐ Wait until after M0 for group selection
- ☐ Talk about layout early. Force full M0 tutorial
- ☐ Make them do TR layout early
- ☐ Find a good sigma delta intro circuit
- ☐ Write about FFTs
- ☐ It helps if you use the right template for tinytapout (and not mirrored ports as you did in 2026)

I love programming and automation. Not much makes me more happy than using the same source (the slide markdowns), to generate the lecture notes, to translate into the book your looking at right now.

If you find an error in what I've made, then fork aic2027, fix , commit, push and create a pull request. That way, we use the global brain power most efficiently, and avoid multiple humans spending time on discovering the same error.

2 The Story of Jayn

Keywords: Electron, Big Bang, Fusion, Supernova, Silicon, Sand, Wafer, IC

I want to tell you a story about Jayn. Jayn is an electron. Just one among countless others. There's nothing particularly special about Jayn, but Jayn plays an important role in our story.

At one point in Jayn's long life, without knowing it, Jayn would cause problems for me – but that's not where Jayn's story begins.

Jayn's story begins 13.8 billion years ago. Jayn popped into existence out of the emptiness, seemingly alone in the world. Well, not entirely alone. Surrounded by cousins and siblings, an ocean of elementary particles. There were quarks, there were electrons, and they all swam together in a sea of raw energy and matter.

Before long, the quarks got tired of floating around on their own. They clumped together, forming protons and neutrons – massive beasts compared to little Jayn. Jayn didn't like this new, crowded world. Jayn loved the freedom of zipping around the universe unbound. But eventually, Jayn felt a tug – a deep, irresistible attraction to one of those giant protons. And with that, Jayn became part of something new: a hydrogen atom.

Jayn lived in that hydrogen atom for billions of years. Over time, Jayn joined with other hydrogen atoms, coalescing into a star. Jayn basked in the warmth of the star's outer layers, content as the star burned brightly in the universe.

Eventually, though, Jayn's hydrogen atom drifted toward the star's core. There, something new happened. Instead of circling just one proton, Jayn now orbited two. A new proton had joined, forming helium. This meant there was space for another electron too – Jayn's first real companion.

They shared the same orbital space, something Jayn had never experienced before. Usually, if another electron came too close, one of them would yield into another energy state. But not this time. This was different. They could exist in the same place, with the same energy. How? Physicists would later call it "spin". Jayn didn't know what "spin" meant, it was weird, but Jayn liked it. It meant Jayn wasn't alone.

Jayn was happy in the helium atom, but the universe never stands still. Another proton came, then another, and eventually Jayn was

part of a silicon atom, orbiting a nucleus with 14 protons and – of course – 14 electrons.

Jayn was no longer close to the nucleus. Now, Jayn was far out on the edge of the atom, in the outermost orbital shell. From there, Jayn could feel the presence of everything around – not just its own atom, but all the neighboring atoms as well.

One day, everything changed. The star exploded – a supernova – flung Jayn into the universe.

For a time, Jayn was adrift. Not alone, but not close to anything familiar. Eventually, the gravity of a forming planet – the one we now call Earth – caught Jayn's atom. The silicon atom joined with other atoms, bound together by Jayn's shared orbital shell between the atoms. It was easier to share the electrons than to be apart. The allure of the other atoms wasn't strong, but it was always there.

Jayn spent billions of years as part of the silicon atom, tumbling through Earth's oceans, sometimes bonding with other atoms, sometimes drifting free. One day, Jayn washed up on a beach, part of a grain of sand. And there Jayn stayed for a long, long time.

Now and then, out to sea Jayn went, then returned, living a simple, chaotic, quiet life.

But even quiet lives face change. One day, scooped up in a bucket, melted down, and turned into something new. Melted just meant more energy, Jayn had experienced that before, just more vibration. Sometimes, because of the vibrations, Jayn even had enough energy to escape the atom briefly before settling back down. But this time was different.

This time, Jayn became part of something incredibly uniform: a crystal lattice where every atom was another silicon atom, each in perfect order, with each atom sharing it's four outermost electrons in orbital shells with the neighbors. For the first time, Jayn could feel the full, equal pull of the electrons and nuclei around – like an invisible ocean of charges. It was electrifying.

Then one day, it happened. Another electron struck. It hit hard, and knocked Jayn from the orbital shell. For a brief moment, Jayn was free. Jayn flew through the crystal lattice, disoriented, then through something that was not a crystal lattice, but rather a jumbled mess of crystal pieces, until Jayn found another open spot, an empty energy state, where Jayn could settle again. It was strange but exhilarating.

And did you know? The presence of just that one electron – Jayn – in the gate oxide of a transistor was enough to shift the threshold voltage, change the flow of bias current, alter the frequency of an oscillator, cause my phone to lose the Bluetooth link to my door

lock, and made me swear a number of times until the Bluetooth link finally reconnected, many, many, many seconds later.

And yet, Jayn's story doesn't end. Because Jayn, like all electrons, never really ends. Jayn may pop in and out of existence, but is always there – unchanged, identical to all the siblings.

The only differences between the electrons are where they are, their momentum and yes, their "spin". They are responsible for all chemical reactions in the universe. And their path through space-time described by the complex mathematics of the Schrödinger equation. Or if you want to include relativity, the Lagrangian of Quantum Electrodynamics.

So ends the story of Jayn – our 13.8 billion-year-old troublemaker.

Introduction 3

3.1 Who 11
3.2 How I see our roles . . 12
3.3 I want you to learn the skills necessary to make your own ICs . . 13
3.4 There will always be analog circuits, because the real world is analog 14
3.5 Will you tape-out an IC? 14
3.5.1 What the team needs to know to design ICs . . 15
3.5.2 Zen of IC design (stolen from Zen of Python) . 16
3.5.3 IC design mantra . . . 16
3.5.4 Analog Design Process 16
3.6 My Goal 17
3.7 Syllabus 18
3.8 Software 18
3.9 Summary 18

Keywords: Course Goals, Roles, Skills, Zen of IC Design, Design Process, Tapeout

3.1 Who

My name is

Carsten Wulff carstenw@ntnu.no

I finished my Masters in 2002, and did a Ph.D on analog-to-digital converters finished in 2008.

Since that time, I've had a three axis in my work/hobby life.

I work at Nordic Semiconductor where I've been since 2008. The first 7 years I did analog design (ADCs, DC/DCs, GPIO). The next 7 years I was the Wireless Group Manager. The Wireless group made most of the analog and RF designs for Nordic's short-range products. Now I'm the IC Scientist, and focus on technical issues with our integrated circuits that occur before we go into volume production.

I work at NTNU where I did a part time postdoc from 2014 - 2017. From 2020 I've been working on and teaching Advanced Integrated Circuits

I have a hobby trying to figure out how to make a new analog circuit design paradigm. The one we have today with schematic/simulation/layout/verification/simulation is too slow

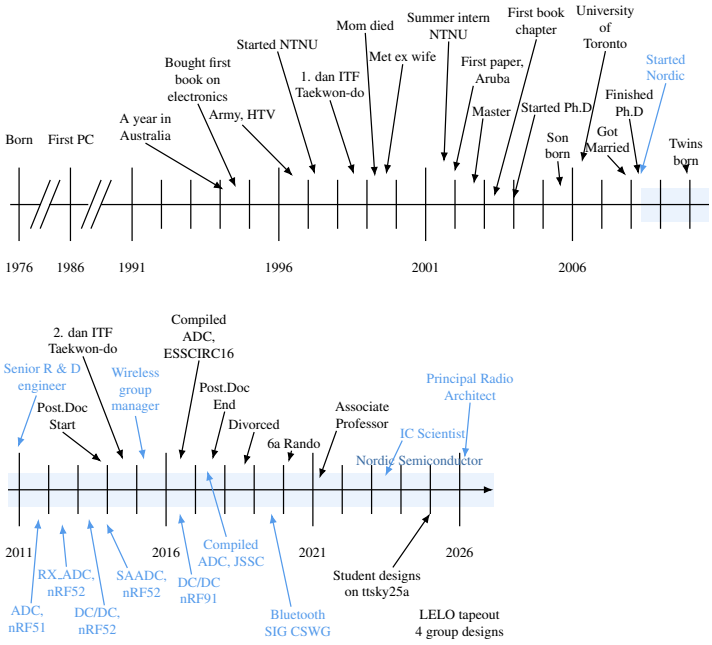


Figure 1: My life

3.2 How I see our roles

In Figure 2 you can see how I think about the research universe. There are things we know to be possible, things that actually are impossible (travel back in time, breaking thermodynamics, travel with a speed beyond light).

Between the impossible, and the possible, lies the unknown. I consider our roles as follows:

Professors: Guide students on what is impossible, possible, and hints on what might be possible

Ph.D students: Venture into the unknown and make something (more) possible

Master students: Learn all that is currently possible

Bachelor students: Learn how to make complicated into easy

Industry: Take what is possible, and/or complicated, and make it easy

Everyone in the figure pushes a border: teaching moves knowledge from complicated towards easy, research moves the border of the unknown, and industry earns its living on the innermost rings. This course lives one ring out from the centre - what is known and complicated, made learnable.

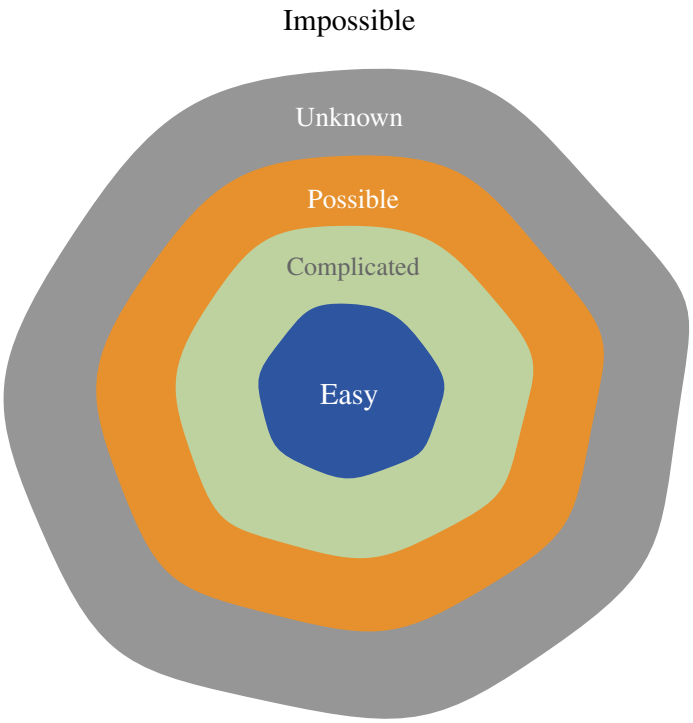


Figure 2: The research universe: Easy at the centre, then Complicated, Possible and Unknown, with the Impossible beyond the last border

3.3 I want you to learn the skills necessary to make your own ICs

In 2020 the global integrated circuit market was 437.7 billion dollars! The market is expected to grow to 1136 billion in 2028. Integrated circuits enable all technologies.

I will be dead in approximately 50 years, and will retire in approximately 20 years. Everything I know will be gone (except for the small pieces I've left behind in videos or written word)

Someone must take over, and to do that, they need to know most of what I know, and hopefully a bit more.

That's where some of you come in. Some of you will find integrated circuits interesting to make, and in addition, you have the stamina, patience, and brain necessary to learn some of the hardest topics in the world.

3.4 There will always be analog circuits, because the real world is analog

In this course, we'll focus on analog ICs, because the real world is analog, and all ICs must have some analog components, otherwise they won't work.

The steps to make integrated circuits are split in two. We have an analog flow, and a digital flow, as shown in Figure 3.

It's rare to find a single human that does both flows well. Usually people choose, and I think it's based on what they like and their personality.

If you like the world to be ordered, with definite answers, then it's likely that you'll find the digital flow interesting.

If you're comfortable with not knowing, and have an insatiable desire to understand how the world *really* works at a fundamental level, then it's likely that you'll find the analog flow interesting.

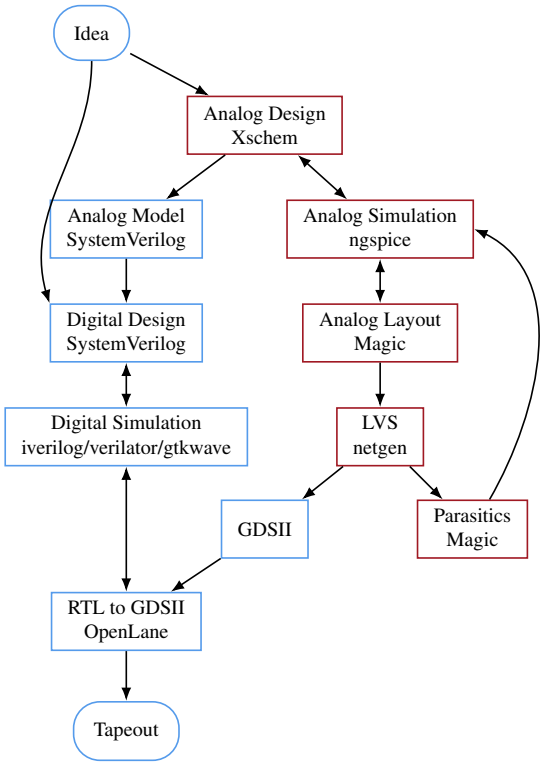


Figure 3: Analog and Digital design process

3.5 Will you tape-out an IC?

Something that would make me really happy is if someone is able to tapeout an IC in this course.

It's now possible without signing an NDA or buying expensive software licenses.

In 2020 Google and Skywater joined forces to release a 130 nm process design kit to the public. In addition, they have fueled a renaissance of open source software tools.

tinytapeout runs cheap shuttles which makes it possible for a private citizen to tape-out their own integrated circuit.

3.5.1 What the team needs to know to design ICs

There are a multitude of tools and skills needed to design professional ICs. It's not likely that you'll find all the skills in one human, and even if you could, one human does not have sufficient bandwidth to design ICs with all its aspects in a reasonable timeline

That is, unless we can find a way to make ICs easier.

The skills needed are

- *Project flow support*: **Confluence**, JIRA, risk management (DFMEA), failure analysis (8D)
- *Language*: **English**, **Writing English (Latex, Word, Email)**
- *Psychology*: Personalities, convincing people, presentations (Powerpoint, Deckset), **stress management (what makes your brain turn off?)**
- *DevOps*: **Linux**, build systems (CMake, make, ninja), continuous integration (bamboo, jenkins), **version control (git)**, containers (docker), container orchestration (swarm, kubernetes)
- *Programming*: Python, C, C++, Matlab *Since 1999 I've programmed in Python, Go, Visual BASIC, PHP, Ruby, Perl, C#, SKILL, Ocean, Verilog-A, C++, BASH, AWK, VHDL, SPICE, MATLAB, ASP, Java, C, SystemC, Verilog, Assembler, and probably a few I've forgotten.*
- *Firmware*: signal processing, algorithms, software architecture, security
- *Infrastructure*: **Power management**, **reset**, **bias**, **clocks**
- *Domains*: CPUs, peripherals, memories, bus systems
- *Sub-systems*: **Radio's**, **analog-to-digital converters**, **comparators**
- *Blocks*: **Analog Radio**, Digital radio baseband
- *Modules*: Transmitter, **receiver**, de-modulator, timing recovery, state machines
- *Designs*: **Opamps**, **amplifiers**, **current-mirrors**, adders, random access memory blocks, standard cells

- *Tools*: **schematic**, **layout**, **parasitic extraction**, synthesis, place-and-route, **simulation**, (System)Verilog, **netlist**
- *Physics*: transistor, pn junctions, quantum mechanics

3.5.2 Zen of IC design (stolen from Zen of Python)

When you learn something new, it's good to listen to someone that has done whatever it is before.

Here are some guiding principles that you'll likely forget.

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts (especially schematics).
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one **and preferably only one** obvious way to do it.
- Now is better than never.
- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.

3.5.3 IC design mantra

To copy an old mantra I have on learning programming (run it in a bash/zsh/cshrc terminal, or in your brain)

```
echo "Find a problem that you really want to solve,"\
     "and learn programming to solve it."\
     "There is no point in saying 'I want to learn programming',"\
     "then sit down with a book to read about programming,"\
     "and expect that you will learn programming that way."\
     "It will not happen. The only way to learn programming"\
     "is to do it, a lot." \
     |perl -pe 's/programming/analog design/ig'
```

3.5.4 Analog Design Process

- Define the problem, what are you trying to solve?
- Find a circuit that can solve the problem (papers, books)
- Find right transistor sizes. What transistors should be weak inversion, strong inversion, or don't care?
- Write a verification plan (ask chat). Plan to simulate everything that could go wrong.

- Check operating region of transistors (.op)
- Check key parameters (.dc, .ac, .tran)
- Check function. Exercise all inputs. Check all control signals
- Check key parameters in all corners. Check mismatch (Monte-Carlo simulation)
- Do layout, and check it's error free. Run design rule checks (DRC). Check layout versus schematic (LVS)
- Extract parasitics from layout. Resistance, capacitance, and inductance if necessary.
- On extracted parasitic netlist, check key parameters in all corners and mismatch (if possible).
- If everything works, then you're done.

On failure, go back as far as necessary

3.6 My Goal

Don't expect that I'll magically take information and put it inside your head, and you'll suddenly understand everything about making ICs.

You are the one that must teach yourself everything.

I consider my role as a guide, similar to a mountain guide. I can't carry you up the mountain, you need to walk up the mountain , but I know the safe path to take and increase the likelihood that you'll come back alive.

My guide role:

- Enable you to read the books on integrated circuits
- Enable you to read papers (latest research)
- Correct misunderstandings on the topic
- Answer any questions you have on the chapters

I'm not a mind reader, I can't see inside your head. That means, you must ask questions. Only by your questions can I start to understand what pieces of information is missing from your head, or maybe somehow correct your understanding.

At the same time, and similar to a mountain guide, you should not assume I'm always right. I'm human, and I will make mistakes. And maybe you can correct my understanding of something. All I care about is to *really* understand how the world works, so if you think my understanding is wrong, then I'll happily discuss.

3.7 Syllabus

The syllabus will be from Analog Integrated Circuit Design by Carusone, Johns and Martin [1], which everyone calls CJM, and Circuits for all seasons.

These lecture notes are a supplement to the book. I try to give some background, and how to think about electronics. It's not my goal to repeat information that you can find in the book.

Buy a hard-copy of the book if you don't have that. Don't expect to understand the book by reading the PDF.

3.8 Software

We'll use professional-grade open source software for everything: xschem for schematics, ngspice for simulation, the SKY130A PDK, Magic VLSI and netgen for layout and verification, and surfer, iverilog and verilator on the digital side. No licence server stands between you and your design.

Open source software (xschem, ngspice, sky130A PDK, Magic VLSI, netgen, surfer, iverilog, verilator)

I've made a rather detailed (at least I think so myself) tutorial on how to make a current mirror with the open source tools. You have to do that tutorial. It's Milestone 0 of the project and does count towards your final grade.

Skywater 130 nm Tutorial

I've also made some more complex examples, that can be found at the link below. There are digital logic cells, standard transistors, and few other blocks.

aicex

3.9 Summary

The one-page version of this chapter:

- The real world is analog, so every IC carries analog circuits at its edges - there is no purely digital chip
- Making an IC splits into an analog and a digital flow, and it is rare to find one human who does both well
- Digital designers reuse each other's work; analog designers redraw - closing that gap is a theme of this course

- What you need from here: the refresher chapters, a working toolchain, and the habit of simulating everything

[1] T. C. Carusone, D. Johns, and K. Martin, *Analog integrated circuit design*. Wiley, 2011 [Online]. Available: https://books.google.no/books?id=1OIJZzLvVhcC

How to Achieve Excellence | 4

4.1 Ruthless Focus on Real Value 21
4.2 Chase Down What's Off 21
4.3 Root Causes or Nothing 22
4.4 Constraint Is a Feature 22
4.5 Kill Bureaucracy Aggressively 22
4.6 Support the People Doing the Work 22
4.7 Ask Questions Relentlessly 23
4.8 Optimize for Truth, Not Consensus 23
4.9 Treat the Customer Timeline as Holy 23
4.10 Separate Identity from Job 23
4.11 Control Emotion, Keep Passion 24
4.12 Treat Work as a Game, and Play to Win 24
4.13 Appendix 24

Keywords: Value, Root Cause, Focus, Standards, Ownership, Honesty

Excellence is not a mystery. It is the cumulative result of focusing on what matters, eliminating what does not, and refusing to lie to ourselves. Most organizations fail not because the problems are hard, but because they tolerate confusion, waste, and wishful thinking.

This memo lays out a set of operating principles for building and sustaining excellence.

4.1 Ruthless Focus on Real Value

Start with the customer. Not the roadmap, not internal politics, not what sounds impressive.

Ask relentlessly: what generates actual value for the customer? If an activity, feature, process, or role does not contribute to that value, it is a candidate for removal.

Busy work is not neutral. It actively destroys excellence by consuming time, attention, and energy that should be spent on what matters.

4.2 Chase Down What's Off

Anything that feels "funky" usually is.

If something looks wrong, investigate it. If something is unclear, make it clear. If something is not understood, do not proceed until it is.

Ambiguity compounds. Small misunderstandings turn into large failures if they are ignored. Excellence requires discomfort: stopping, digging, and asking why.

4.3 Root Causes or Nothing

Always chase root causes.

Yes, it takes time. Yes, it consumes resources. Yes, it is worth it.

Treating symptoms feels fast but guarantees recurrence. Every unresolved root cause is technical debt, organizational debt, or cultural debt waiting to collect interest. Excellence is incompatible with bandaid fixes.

4.4 Constraint Is a Feature

Resources are always constrained. Pretending otherwise leads to mediocrity.

If resources are limited, build fewer features. If a job does not need to be done, delete it. If a task does not meaningfully advance the goal, remove it.

Doing less, better, beats doing more, poorly, every time.

4.5 Kill Bureaucracy Aggressively

Bureaucracy is inertia made visible.

If a process slows people down without improving outcomes, delete it. If someone consistently does not contribute, or generates busy work, remove the role. If a meeting has no clear purpose or decisions, cancel it.

You do not need to sync every week. You need clarity, ownership, and trust. Meetings are tools, not rituals.

4.6 Support the People Doing the Work

Understand the real problems your teams face.

If they need resources, provide them. If they lack skills, help them acquire them. If they need guidance or direction, give it.

Upward communication matters. Learn to articulate problems precisely to your lead, and explain exactly how they can support you. Vague complaints are useless; concrete asks move systems.

4.7 Ask Questions Relentlessly

Assumptions are silent killers.

Ask questions. Ask again. Ask until the system makes sense.

Curiosity is not weakness. It is a prerequisite for correctness.

4.8 Optimize for Truth, Not Consensus

Agreement is cheap. Being right is expensive.

Focus on what is correct, not what is popular or agreed upon. Consensus that ignores reality eventually collapses. Truth, even when uncomfortable, compounds.

Do not believe things. Show the data.

4.9 Treat the Customer Timeline as Holy

The customer's timeline is holy.

Never overcommit. Never promise what you are not confident you can deliver. Never say "yes" to avoid discomfort.

Say no when necessary. Honesty beats optimism theater. Broken promises destroy trust faster than missing features.

4.10 Separate Identity from Job

Your worth as a human being is not tied to your job performance.

You can be bad at your job and still have value as a person. Failure at work is not failure as a human.

This separation is essential. People who fuse identity with job become defensive, political, and afraid of truth. Excellence requires psychological safety grounded in reality, not ego.

4.11 Control Emotion, Keep Passion

Leave personal feelings out of decision-making. Bring passion for the work itself.

Emotion-driven execution leads to noise. Passion-driven execution leads to intensity, care, and pride in craftsmanship. The goal is not detachment, but disciplined focus.

4.12 Treat Work as a Game, and Play to Win

Work is a game with rules, constraints, incentives, and opponents (complexity, entropy, time).

Understand the game. Learn its mechanics. Exploit them intelligently.

Playing to win does not mean cutting corners. It means optimizing for outcomes, not appearances. Excellence is not accidental; it is the result of playing the game deliberately and well.

4.13 Appendix

This text was prepared by chat https://chatgpt.com/share/6953defc-843c-8007-9648-57f05fccf1b5

A Refresher 5

Keywords: SI Units, Silicon, Band Structure, Fermi Level, Metals, Insulators, Semiconductors, Band Diagrams, Fields

5.1 There are standard units of measurement

All known physical quantities are derived from 7 base units (SI units)

- second (s) : time
- meter (m) : space
- kg (kilogram) : weight
- ampere (A) : current
- kelvin (K) : temperature
- mole (mol) : amount of substance
- candela (cd) : luminous intensity

All other units (for example volts), are derived from the base units.

I don't go around remembering all of them, they are easily available online. When you forget the equation for charge (Q), voltage (V) and capacitance (C), look at the units below, and you can see it's $Q = CV$ *

* Although you do have to keep your symbols straight. We use "C" for Capacitance, but C can also mean Columbs. Context matters.

5.1 There are standard units of measurement 25
5.2 Electrons 26
5.3 Probability 27
5.4 Uncertainty principle . 28
5.5 States as a function of time and space 28
5.6 Allowed energy levels in atoms 29
5.7 Allowed energy levels in solids 29
5.8 Silicon Unit Cell 30
5.9 Band structure 31
5.10 Valence band and Conduction band . . . 31
5.11 Fermi level 32
5.12 Metals 33
5.13 Insulators 33
5.14 Semiconductors 34
5.15 Band diagrams 34
5.16 Density of electrons/-holes 34
5.17 Fields 35
5.18 Permittivity and Permeability 36
5.19 Quantum electrodynamics 36
5.20 Voltage 36
5.21 Current 37
5.22 Drift current 37
5.23 Diffusion current . . . 38
5.24 Why are there two currents? 38
5.25 Currents in a semiconductor 39
5.26 Resistors 39
5.27 Capacitors 40
5.28 Inductors 40
5.29 Summary 40
5.30 Would you like to know more? 40

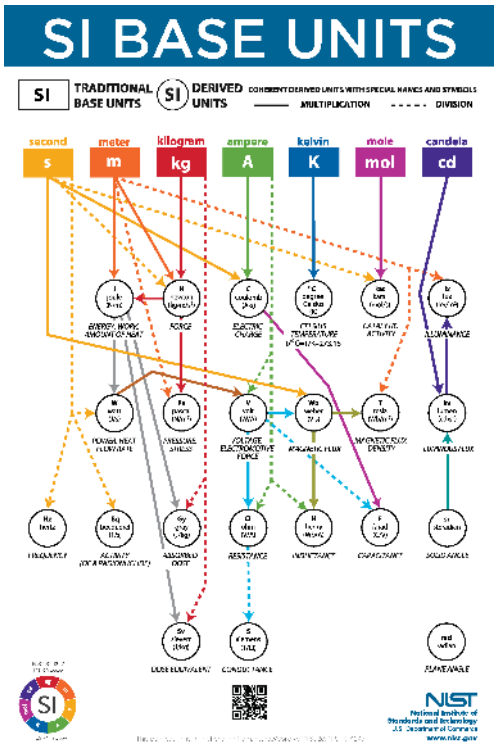


Figure 1: Si base units, from https://www.nist.gov/pml/owm/metric-si/si-units. Image: NIST, US Department of Commerce (US federal work)

5.2 Electrons

Electrons are fundamental, they cannot (as far as we know), be divided into smaller parts. Explained further in the standard model of particle physics

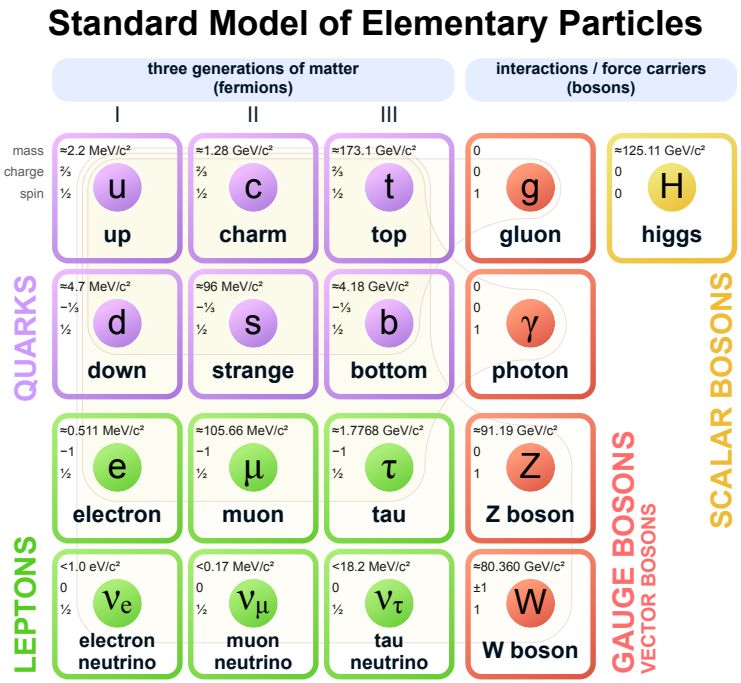


Figure 2: Standard model of particle physics, Wikipedia. Image: Cush after MissMJ, public domain, via Wikimedia Commons

Electrons have a negative charge of $q \approx 1.602 \times 10^{-19}$. The proton a positive charge. The two charges balance exactly! If you have a trillion electrons and a trillion protons inside a volume, the net external charge will be 0 (assuming we measure from some distance away). I find this fact absolutely incredible. There must be a fundamental connection between the charge of the proton and electron. It's insane that the charges balance out so exactly.

All electrons are the same, although the quantum state can be different.

An electron cannot occupy the same quantum state as another. This rule applies to all fermions (particles with spin of 1/2)

The quantum state of an electron is fully described by its spin, momentum (p) and position in space (r).

5.3 Probability

The probability of finding an electron in a state as a function of space and time is

$$P = |\psi(r,t)|^2$$

, where ψ is named the probability amplitude, and is a complex function of space and time. In some special cases, it's

$$\psi(r,t) = Ae^{i(kr-\omega t)}$$

, where A is complex number, k is the wave number, r is the position vector from some origin, ω is the frequency and t is time.

The energy is $E = \hbar\omega$, where $\hbar = h/2\pi$ and h is Planck Constant and the momentum is $p = \hbar k$

The probability amplitude is also called the wave function. Type of wave function depends on the scenario, and does not have to take on the solution above. The possible wave functions are those equations that fits with the time evolution of quantum states given by the Schrodinger equation.

5.4 Uncertainty principle

We cannot, with ultimate precision, determine both the position and the momentum of a particle, the precision is

$$\sigma_x \sigma_p \geq \frac{\hbar}{2}$$

From the uncertainty (Unschärfe) principle we can actually estimate the size of the atom

5.5 States as a function of time and space

The time-evolution of the probability amplitude is

$$i\hbar\frac{d}{dt}\psi(r,t) = H\psi(r,t)$$

, where H is named the Hamiltonian matrix, or the energy matrix or (if I understand correctly) the amplitude matrix of the probability amplitude to change from one state to another.

For example, if we have a system with two states, a simplified version of two electrons shared between two atoms, as in H_2, or hydrogen gas, or co-valent bonds, then the Hamiltonian is a 2 x 2 matrix. And the ψ is a vector of $[\psi_1, \psi_2]$

Computing the solution to the Schrodinger Equation can be tricky, because you must know the number of relevant states to know the

vector size of ψ and the matrix size of H. In addition, the H can be a function of time and space (I think).

Compared to the equations of electric fields, however, Schrodinger is easy, it's a set of linear differential equations.

5.6 Allowed energy levels in atoms

Solutions to Schrodinger result in quantized energy levels for an electron bound to an atom.

Take hydrogen, the electron bound to the proton can only exists in quantized energy levels. The lowest energy state can have two electrons, one with spin up, and one with spin down.

From Schrodinger you can compute the energy levels, which most of us did at some-point, although now, I can't remember how it was done. That's not important. The important is to internalize that the energy levels in bound electrons are discrete.

Electrons can transition from one energy level to another by external influence, i.e temperature, light, or other.

The probability of a state transition (change in energy) can be determined from the probability amplitude and Schrodinger.

5.7 Allowed energy levels in solids

If I have two silicon atoms spaced far apart, then the electrons can have the same spin and same momentum around their respective nuclei. As I bring the atoms closer, however, the probability amplitudes start to interact (or the dimensions of the Hamiltonian matrix grow), and there can be state transitions between the two electrons.

The allowed energy levels will split. If I only had two states interacting, the Hamiltonian could be

$$H = \begin{bmatrix} A & 0 \\ 0 & -A \end{bmatrix}$$

and the new energy levels could be

$$E_1 = E_0 + A$$

and

$$E_2 = E_0 - A$$

In a silicon crystal we can have trillions of atoms, and those that are close, have states that interact. **That's why crystals stay solids**. All chemical bonds are states of electrons interacting! Some are strong (co-valent bonds), some are weaker (ionic bonds), but it's all quantum states interacting.

The discrete energy levels of the electron transition into bands of allowed energy states.

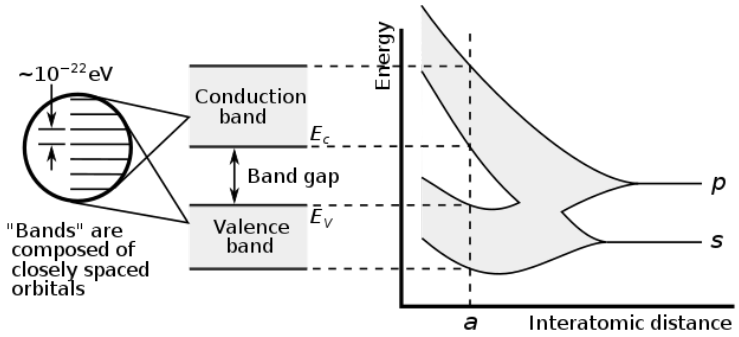


Figure 3: Electronic band structure, Wikipedia. Image: Chetvorno, CC0, via Wikimedia Commons

For a crystal, the allowed energy bands is captured in the band structure

5.8 Silicon Unit Cell

A silicon crystal unit cell is a diamond faced cubic with 8 atoms in the corners spaced at 0.543 nm, 6 at the center of the faces, and 4 atoms inside the unit cell at a nearest neighbor distance of 0.235 nm.

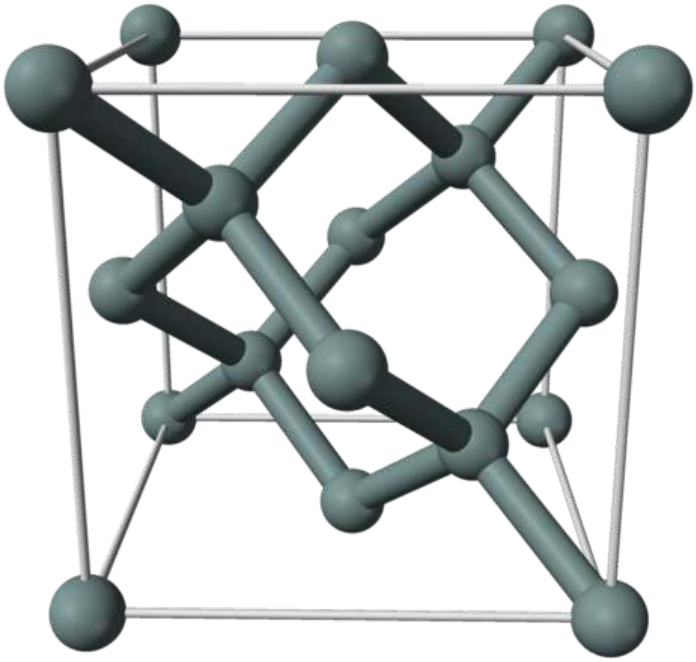


Figure 4: Silicon, Wikipedia. Image: Ben Mills, public domain, via Wikimedia Commons

5.9 Band structure

The full band structure of a silicon unit cell is complicated, it's a 3 dimensional concept

Peter Hadley's course notes at TU Graz have the full plot, and it is worth looking at once: energy against wavevector along the symmetry directions of the crystal, with the conduction band minimum sitting off to one side rather than above the valence band maximum. That offset is what makes silicon an *indirect* bandgap semiconductor, and it is the reason silicon is used for transistors and not for light emitting diodes: an electron dropping across the gap has to shed momentum as well as energy, which needs a phonon to turn up at the same moment, and that is a far less likely event than simply emitting a photon.

5.10 Valence band and Conduction band

For bulk silicon we simplify, and we think of two bands, the conduction band, and valence band

In the conduction band (E_C) is the lowest energy where electrons are free (not bound to atoms). The valence band (E_V) is the highest band where electrons are bound to silicon atoms.

The difference between E_C and E_V is a property of the material we've named the band gap.

$$E_G = E_C - E_V$$

5.11 Fermi level

From Wikipedia's Fermi level

> In band structure theory, used in solid state physics to analyze the energy levels in a solid, the Fermi level can be considered to be a hypothetical energy level of an electron, such that at thermodynamic equilibrium this energy level would have a 50% probability of being occupied at any given time

The Fermi level is closely linked to the Fermi-Dirac distribution

$$f(E) = \frac{1}{e^{(E-E_F)/kT} + 1}$$

If the energy of the state is more than a few kT away from the Fermi-level, then

$$f(E) \approx e^{(E_F-E)/kT}$$

The equation above is one of the reasons the structure $e^{E/kT}$ or $e^{qV/kT}$ shows up all over the place. You'll see it in the equations for current in a diode, $I_D = I_s(e^{qV_D/nkT} - 1)$, the subthreshold conduction of a mosfet $I_D \propto e^{qV_{gs}/nkT}$ and even the Arrhenius Equation $k = Ae^{-E_a/kT}$.

It seems like any time you have something related to chemical reactions (state transitions of electrons, breaking bonds, forming bonds), or current in solids, there is a relation to the equation above. To me, that makes sense.

The Fermi-Dirac function also explains why there are more free carriers, and reaction rates increase, at high temperature. The part of the equation that is $e^{-E/kT}$ will approach one at high temperatures.

5.12 Metals

In metals, the band splitting of the energy levels causes the valence band and conduction band to overlap.

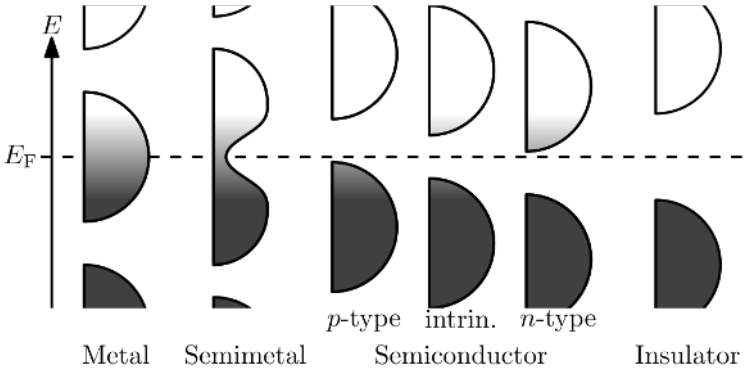


Figure 5: Band splitting in materials. Electronic Band Structure, Wikipedia. Image: Nanite, CC0, via Wikimedia Commons

Electrons can easily transition between bound state and free state. As such, electrons in metals are shared over large distances, and there are many electrons readily available to move under an applied field, or difference in electron density. That's why metals conduct well.

5.13 Insulators

In insulating materials the difference between the conduction band and the valence band is large. As a result, it takes a large energy to excite electrons to a state where they can freely move.

That's why glass is transparent to optical frequencies. Visible light does not have sufficient energy to excite electrons from a bound state.

That's also why glass is opaque to ultra-violet, which has enough energy to excite electrons out of a bound state.

Based on these two pieces of information you could estimate the bandgap of glass.

```
from scipy import constants
#- We must use the "correct" units for planck's constant to get energy in eV
h = constants.physical_constants["Planck constant in eV/Hz"][0]
c = constants.physical_constants["speed of light in vacuum"][0]

lambda_optical = 450e-9
e_optical = h * c/lambda_optical

lambda_ultra = 380e-9
e_ultra = h * c/lambda_ultra

print("Bandgap of glass is above %.2f eV, maybe around %.2f eV " %(e_optical,e_ultra))
```

5.14 Semiconductors

In silicon the bandgap is lower than an insulator, approximately

$$E_G = 1.12\ eV$$

At room temperature, that allows a small number of electrons to be excited into the conduction band, leaving behind a "hole" in the valence band.

5.15 Band diagrams

A band diagram or energy level diagrams shows the conduction band energy and valence band energy as a function of distance in the material.

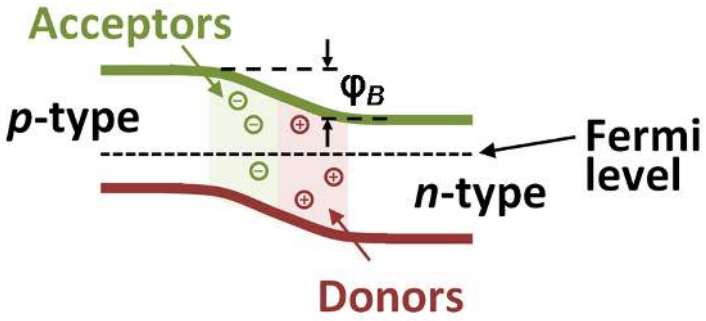


Figure 6: Band diagram of a PN junction, Wikipedia. Image: Brews ohare, CC BY-SA 3.0, via Wikimedia Commons

The horizontal axis is the distance in the material, the vertical axis is the energy.

5.16 Density of electrons/holes

There are two components needed to determine how many electrons are in the conduction band. The density of available states, and the probability of an electron to be in that quantum state.

The probability is the Fermi-Dirac distribution. The density of available states is a complicated calculation from the band-structure of silicon.

For details see the Diodes chapter.

$$n_e = \int_{E_C}^{\infty} N(E) f(E) dE$$

The Fermi level is assumed to be independent of energy level, so we can write

$$n_e = e^{E_F/kT} \int_{E_C}^{\infty} N(E) e^{-E/kT} dE$$

for the density of electrons in the conduction band.

5.17 Fields

There are equations that relate electric field, magnetic field, charge density and current density to each-other.

Electric flux is the total electric field through a surface. Magnetic flux is the total magnetic field through a surface.

Think of flux as the "current", and the field as the "current density". You have to multiply the current density by an area to get current. You have to multiply the magnetic field by an area to get magnetic flux

$$\oint_{\partial\Omega} \mathbf{E} \cdot d\mathbf{S} = \frac{1}{\epsilon_0} \iiint_V \rho \cdot dV$$

,relates net electric flux to net enclosed electric charge

$$\oint_{\partial\Omega} \mathbf{B} \cdot d\mathbf{S} = 0$$

,relates net magnetic flux to net enclosed magnetic charge

$$\oint_{\partial\Sigma} \mathbf{E} \cdot d\ell = -\frac{d}{dt} \iint_{\Sigma} \mathbf{B} \cdot d\mathbf{S}$$

,relates induced electric field to changing magnetic flux

$$\oint_{\partial\Sigma} \mathbf{B} \cdot d\ell = \mu_0 \left(\iint_{\Sigma} \mathbf{J} \cdot d\mathbf{S} + \epsilon_0 \frac{d}{dt} \iint_{\Sigma} \mathbf{E} \cdot d\mathbf{S} \right)$$

,relates induced magnetic field to changing electric flux and to current

These are the Maxwell Equations, and are non-linear time dependent differential equations.

Under the best of circumstances they are fantastically hard to solve! But it's how the real world works.

5.18 Permittivity and Permeability

The permittivity of free space is defined as

$$\epsilon_0 = \frac{1}{\mu_0 c^2}$$

, where c is the speed of light, and μ_0 is the vacuum permeability, which, in SI units, is now

$$\mu_0 = \frac{2\alpha}{q^2}\frac{h}{c}$$

, where α is the fine structure constant.

5.19 Quantum electrodynamics

The quantum electrodynamics (QED) is a full description of interactions between light and matter. The equations describe both quantum mechanical effects, electromagnetism and is in agreement with special relativity.

The equations are rather complicated, but it's based on Lagrangian physics. Maxwell's equations actually fall out of the QED Lagrangian when one assumes local phase symmetry.

The QED Lagrangian is

$$\mathcal{L} = \bar{\psi}[i\hbar c\gamma^\mu \partial_\mu - mc^2]\psi - q[\bar{\psi}\gamma^\mu\psi]A_\mu - \frac{1}{16\pi}F_{\mu\nu}F^{\mu\nu}$$

For more information, have a look at Electromagnetism as a Gauge Theory

5.20 Voltage

The electric field has units voltage per meter, so the electric field is the negative derivative of the voltage as a function of space.

$$E = -\frac{dV}{dx}$$

5.21 Current

Current has unit A and charge C has unit As, so the current is the number of charges passing through a volume per second.

The current density J has units A/m^2 and is often used, since we can multiply by the surface area of a conductor, if the current density is uniform.

$$I = \text{Area} \times J$$

5.22 Drift current

Charge carriers (electrons, holes, ions) in an electric field will give rise to a drift current.

We know from Newtons laws that force equals mass times acceleration

$$\vec{F} = m\vec{a}$$

If we assume a zero, or constant magnetic field, the force on a particle is

$$\vec{F} = q\vec{E}$$

The current density is then

$$\vec{J} = q\vec{E} \times n \times \mu$$

where n is the charge density, and μ is the mobility (how easily the charges move) and has units m^2/Vs

Assuming

$$E = V/m$$

, we could write

$$J = \frac{C}{m^3}\frac{V}{m}\frac{m^2}{Vs} = \frac{C}{s}m^{-2}$$

So multiplying by an area A with unit meters squared

$$I = qn\mu AV$$

and we can see that the conductance

$$G = qn\mu A$$

, and since

$$G = 1/R$$

, where R is the resistance, we have

$$I = GV \Rightarrow V = RI$$

Or Ohms law

5.23 Diffusion current

A difference in charge density will give rise to a diffusion current. The current density is

$$J = -qD_n \frac{d\rho}{dx}$$

,where D_n is a diffusion constant, and ρ is the charge density.

5.24 Why are there two currents?

I struggled with the concepts diffusion current and drift current for a long time. Why are there two types of current? It was when I read The Schrödinger Equation in a Classical Context: A Seminar on Superconductivity I realized that the two types of current come directly from the Schrodinger equation, there is one component related to the electric field (potential energy) and a component related to the momentum (kinetic energy).

In the absence of an electric field electrons will still jump from state to state set by the probabilities of the Hamiltonian. If there are more electrons in an area, then it will seem like there is an average movement of charges away from that area. That's how I think about drift and diffusion currents. We can kinda see it from the Schrödinger equation below.

$$-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial^2 x}\psi(x,t) + V(x)\psi(x,t) = i\hbar\frac{\partial}{\partial t}\psi(x,t)$$

5.25 Currents in a semiconductor

Both electrons, and holes will contribute to current.

Electrons move in the conduction band, and holes move in the valence band.

Both holes and electrons can only move if there are available quantum states.

For example, if the valence band is completely filled (all states filled), then there can be no current.

To compute the total current in a semiconductor one must compute

$$I = I_{n_{drift}} + I_{n_{diffusion}} + I_{p_{drift}} + I_{p_{diffusion}}$$

where n denotes electrons, and p denote holes.

5.26 Resistors

We can make resistors with many materials. The behavior of the charge carrier may be different between materials.

In metal the dominant carrier depends on the metal, but it's usually electrons. As such, one can often ignore the hole current.

In a semiconductor the dominant carrier depends on the Fermi level in relation to the conduction band and valence band.

If the Fermi level is close to the valence band the dominant carrier will be holes. If the Fermi level is close to the conduction band, the dominant carrier will be electrons.

That's why we often talk about "majority carriers" and "minority carriers", both are important in semiconductors.

5.27 Capacitors

A capacitor resists a change in voltage

$$I = C\frac{dV}{dt}$$

and store energy in an electric field between two conductors with an insulator between.

5.28 Inductors

An inductor resist a change in current

$$V = L\frac{dI}{dt}$$

and store energy in the magnetic fields in a loop of a conductor.

5.29 Summary

The one-page version of this chapter:

- Seven SI base units; electronics interacts with the world through the second and the ampere
- Electrons are fermions: two per state, which is what builds shells, bands, and all of chemistry
- Energy lives in the fields: $E = -dV/dx$, and voltage is energy per charge
- This refresher's job is vocabulary - enough physics to carry the diode and MOSFET chapters

5.30 Would you like to know more?

Feynman's *Lectures on Physics*, volume I - the pleasure route through everything in this chapter

Griffiths, *Introduction to Quantum Mechanics* - the usual next step for the quantum parts

6 Fields

6.1 The four equations, in words 41
6.2 Currents always run in loops 42
6.3 The capacitor falls out . 43
6.4 The inductor falls out . 43
6.5 Every wire is both 44
6.6 The antenna does not radiate 45
6.7 What this buys you on-chip 47
6.8 Summary 47
6.9 Would you like to know more? 48

This chapter was written by Claude, Anthropic's AI, from an outline and direction by Carsten Wulff, who reviewed and edited the result. The figures are Claude's, in the book's style. The commit history of the book's repository records precisely who wrote what.

Circuit theory is Maxwell's equations with the fine print deleted. The deletion is a good deal - we get Kirchhoff, nodes and branches, and we can design a chip without solving a single boundary value problem. But the fine print does not go away, and every so often it sends a bill: a supply that bounces, a clock that couples into an ADC, an inductor that is mostly resistor, a wire that has become an antenna.

This chapter restores exactly three pieces of the fine print, the three an IC designer keeps paying for: currents close in loops, every gap is a capacitor, and every loop is an inductor. Radiation - the part everyone treats as magic - is just what the fine print does at high frequency.

There will be no vector calculus gymnastics here. We use the integral forms, in words, and tie every claim to a chapter of this book where you will meet it again.

6.1 The four equations, in words

$$\oint \vec{E} \cdot d\vec{A} = \frac{Q}{\varepsilon_0}$$

$$\oint \vec{B} \cdot d\vec{A} = 0$$

$$\oint \vec{E} \cdot d\vec{\ell} = -\frac{d\Phi_B}{dt}$$

$$\oint \vec{B} \cdot d\vec{\ell} = \mu_0 I + \mu_0 \varepsilon_0 \frac{d\Phi_E}{dt}$$

Gauss: electric field lines start on positive charge and end on negative charge. Count the lines leaving a closed surface and you have counted the charge inside.

Gauss for magnetism: magnetic field lines do not start or end anywhere. There is no magnetic charge; every field line is a closed loop.

Faraday: a changing magnetic flux through a loop drives a voltage around it. This is the only way to make an electric field whose lines close on themselves rather than ending on charge.

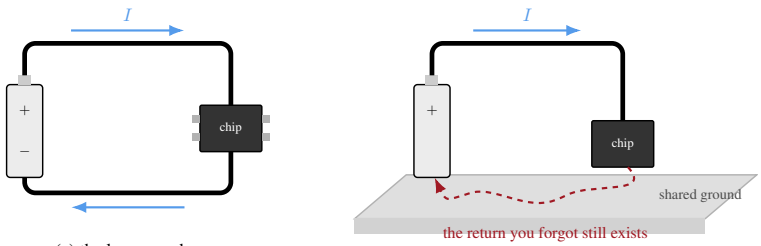
Ampere, with Maxwell's correction: currents make magnetic fields curl around them - and so does a *changing electric field*. The second term is the fine print that makes capacitors, and radio, work. It is called the displacement current, and it deserves a better reputation.

One more law hides inside these four. Take Ampere's equation and close the surface: the conduction current in, plus the displacement current in, must equal zero. Charge is conserved, always, everywhere. That innocent bookkeeping statement is the most useful sentence in this chapter.

6.2 Currents always run in loops

Charge conservation says the total current into any closed surface is zero. Draw a surface around any point of your circuit: whatever current comes in must leave. Follow it, and you must eventually come back to where you started. **Every current closes a loop.** There are no exceptions - not for signals, not for supplies, not for that "unidirectional" clock trace.

The design consequence is that the return path is not optional. You only get to choose *where* it goes. Draw the loop deliberately, as in Figure 1(a), and you know its area, its inductance and its victims. Forget it, as in Figure 1(b), and the current still closes - through the substrate, through a neighbouring supply, through whatever shared ground is available - and everything on that accidental path sees your signal as ground bounce.



(a) the loop you drew

(b) the loop you got

Figure 1: Every current closes a loop. (a) The loop you drew. (b) The loop you got: the forgotten return closes through the shared ground, and everything on that path sees it

When you debug a noisy chip, do not ask "where does this signal go?". Ask "where does it come back?". The second question finds the problem.

Don't ask "where does the signal go?"

Ask "where does it come back?"

6.3 The capacitor falls out

Now break the top wire of the loop, as in Figure 2. Kirchhoff panics - the circuit is open. Maxwell does not: charge piles up on the two faces of the break, an electric field grows between them, and the changing field *is* a current. Ampere's correction term $\partial D/\partial t$ carries the loop across the gap as if the wire were never cut. Charge conservation is satisfied at every instant; the loop never broke.

A capacitor is a deliberately good gap: two large faces, close together, with a dielectric that multiplies the effect. The $I = C\,dV/dt$ you have used since your first circuits course is Ampere's displacement current wearing a component symbol.

And because *any* gap does this, every pair of conductors on your chip is a capacitor whether you asked for one or not - the parasitics chapter is the bill.

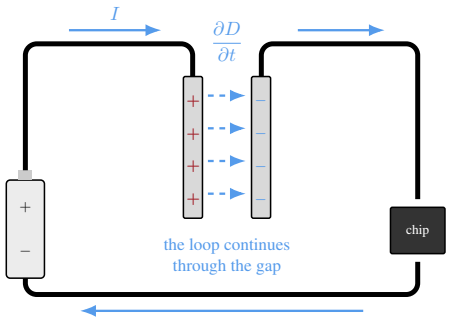


Figure 2: Break the conductor and the loop continues through the gap as displacement current. A capacitor is a deliberately good gap

6.4 The inductor falls out

Figure 3(a) walks the three steps. **One:** push a rising current through a loop of wire, and Ampere says that current wraps a magnetic field around the wire. **Two:** some of that field threads the loop itself - the flux Φ is proportional to the current, with a constant decided purely by the geometry, and we call it L. **Three:** Faraday says a changing flux drives a voltage around the loop, and Lenz's rule says it drives it in the direction that opposes the change.

The loop fights you, and at the terminals that reads $v = L\,di/dt$: ramp the current, and the loop answers with a voltage, exactly as the inset shows.

An inductor is a deliberately good loop. Wind the same loop N times, Figure 3(b), and the bargain improves twice over: N turns make N times the flux, and every turn feels all of it - which is why the inductance grows as N^2.

The converse of the capacitor's lesson holds here: every loop on your chip is an inductor whether you asked or not. Since every current runs in a loop, every current has an inductance. That is why fast edges ring - the loop's L and the gap's C are always both present, and together they make a resonator you never drew.

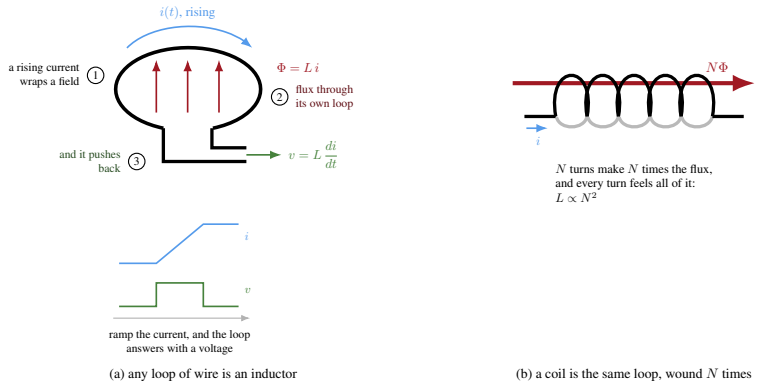


Figure 3: Every loop encloses flux, and a changing current fights its own flux. An inductor is a deliberately good loop; a coil is the same loop N times

6.5 Every wire is both

So the fine print reads: every gap is a capacitor, every loop is an inductor, and every current must loop. Put the three together over a ground plane and you can predict something that surprises most people the first time: *where the return current flows*.

At DC the return current spreads across the whole plane - it minimizes resistance, Figure 4(a). At high frequency it crowds into a narrow band directly under the signal wire, Figure 4(b). It is minimizing impedance, and above a few megahertz the impedance is dominated by the loop inductance - which shrinks with the loop area. The current chooses the smallest loop available.

This one picture is most of signal integrity. Slot the ground plane under a trace and the return detours around the slot: the loop area explodes, the inductance with it. Decoupling works the same way: the capacitor's value matters less than the loop area between it and

the load, because it is the loop inductance that decides how fast the capacitor can deliver charge.



(a) DC: the return spreads over the whole plane (least resistance)

(b) high frequency: the return hugs the trace (least inductance)

Figure 4: Where the return current flows in a ground plane. (a) At DC it spreads to minimize resistance. (b) At high frequency it hugs the signal wire to minimize loop inductance

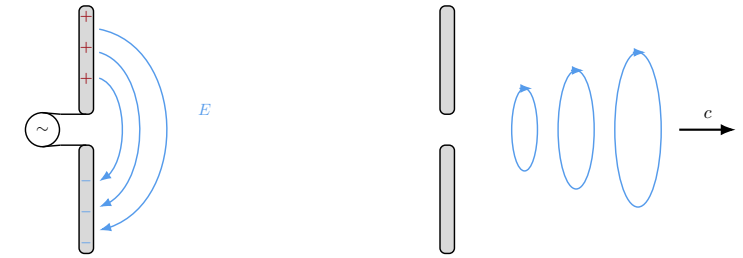
6.6 The antenna does not radiate

Here is the claim this chapter has been building towards: the metal of an antenna does not radiate. The metal only sets up boundary conditions. The drive sloshes charge up and down the arms; the charge makes fields around the structure; and it is the *fields* that carry the power away.

Close to the dipole, Figure 5(a), the electric field lines run from one arm to the other. Every half cycle the drive reverses, the field lines collapse back onto the metal, and their energy returns to the circuit. This is the near field: energy borrowed and repaid, which at the terminals looks like reactance - the antenna below resonance is just a capacitor.

But the news that the drive has reversed travels at the speed of light. Field lines further than about $\lambda/2\pi$ from the arms get the news too late: the charge that anchored them has already moved on, and the lines have nothing to end on. Gauss offers them one way out - close on yourself. Figure 5(b): the loops detach, each one a ring of displacement current sustaining the magnetic field of the next (Ampere), which sustains the electric field of the next (Faraday), and the pair leapfrogs outward at c with no metal anywhere. That self-sustaining leapfrog *is* the electromagnetic wave.

At the antenna terminals, the energy that left with the detached loops never comes back. Power delivered and not returned looks, to the circuit, exactly like a resistor: the radiation resistance. It is the receipt for the escaped loops.



(a) near field: the lines end on the metal, and the energy comes back

(b) beyond $\sim \lambda/2\pi$: the loops close on themselves and leave - no metal anywhere

Figure 5: (a) Near the dipole the field lines end on the metal and their energy returns every half cycle. (b) Beyond about lambda over 2 pi the lines close on themselves - displacement current with no metal - and carry power away at c

The wavelength sets the geometry. Figure 6 shows the workhorse as it actually ships: a quarter-wave copper trace on a PCB, fed by the radio chip, over a keep-out where the ground is cut away. The ground pour is the other half of the antenna - the trace's image in it completes the dipole. The current standing wave is maximum at the feed and zero at the tip, and resonance lands at $l = \lambda/4$: at 2.4 GHz about 31 mm, which is why the antenna region of a Bluetooth board is the size it is, and why nothing on a millimetre scale chip radiates *efficiently* by accident. Accidental radiators are inefficient antennas - but a receiver channel fighting for -100 dBm does not need your clock harmonic to be efficient, only present.

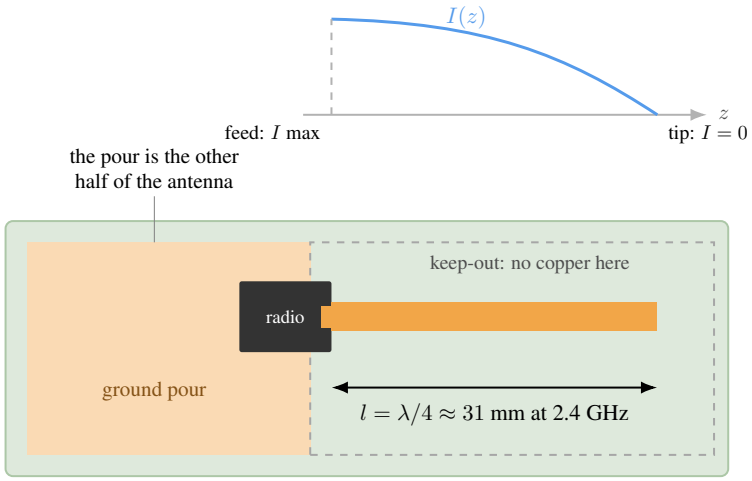


Figure 6: The quarter-wave monopole as it ships, seen from above: the antenna trace on its keep-out, fed by the radio, with the ground pour as the other half of the antenna. Above, on the same scale, the current standing wave that sets l = lambda/4 - about 31 mm at 2.4 GHz

6.7 What this buys you on-chip

Everything in this chapter reappears later in the book wearing a different costume.

On-chip inductors are poor because the fine print is against them twice: the loops that fit on a die are small, so L is small, and they sit on a conductive substrate, so Faraday drives eddy-current losses in exactly the silicon we paid so much for. The LC oscillator chapter lives with the resulting Q.

Decoupling capacitors are loop design, not capacitor selection: the inductance of the loop from capacitor to load sets the frequency above which the capacitor stops helping.

Supply and ground bounce are Figure 1(b) at chip scale, and the cure is the same as the diagnosis: give every fast loop a small, deliberate return.

And the radio chapter's antenna, matching network and Friis budget are this chapter run in reverse: arrange the boundary conditions so the detaching loops are not an accident but the product.

- On-chip inductors: small loops, lossy substrate - the fine print charges twice
- Decoupling is loop design, not capacitor selection
- Ground bounce is Figure 1(b) at chip scale
- A radio is this chapter, run on purpose

6.8 Summary

The one-page version of this chapter:

- Circuit theory is Maxwell with the fine print deleted; the fine print still bills you
- Charge conservation means every current closes a loop - the return path is not optional, only its location is
- A capacitor is a deliberately good gap: displacement current carries the loop across
- An inductor is a deliberately good loop: every loop encloses flux, and a changing current fights its own flux
- At high frequency the return current hugs the signal wire, because the smallest loop has the smallest inductance
- The antenna metal only sets boundary conditions: field loops that detach beyond lambda/2pi carry the power, and radiation resistance is their receipt

6.9 Would you like to know more?

The best intuition-first treatment of these ideas remains Feynman's Lectures on Physics, Volume II - chapter 18 for the full set of equations and what they mean, and chapter 24 onward for waveguides and radiation. For the signal-integrity consequences, Howard Johnson's High-Speed Digital Design (the "black magic" book) is the field manual for Figure 4.

- Feynman Lectures on Physics, Volume II, chapter 18
- Howard Johnson & Martin Graham, High-Speed Digital Design: A Handbook of Black Magic

Diodes 7

Keywords: Silicon, Intrinsic Carriers, Density of States, Fermi-Dirac, Doping, PN Junction, Built-in Voltage, Diode Equation, Temperature, Leakage

7.1 Why 49
7.2 Silicon 49
7.3 Intrinsic carrier concentration 51
7.4 It's all quantum 52
7.4.1 Density of states 54
7.4.2 How to think about electrons (and holes) . 56
7.5 Doping 57
7.6 PN junctions 58
7.6.1 Built-in voltage 59
7.6.2 Current 60
7.6.3 Forward voltage temperature dependence . 61
7.6.4 Current proportional to temperature 63
7.7 Reverse leakage 64
7.7.1 Diffusion leakage . . . 65
7.7.2 Generation in the depletion region 66
7.7.3 Plasma charging currents 67
7.7.4 Antenna ndiode, 200 K to 1000 K 68
7.7.5 Wires below 1 mm . . . 70
7.8 Equations aren't real . 71
7.9 Summary 72

7.1 Why

Diodes are a magical * semiconductor device that conduct current in one direction. It's one of the fundamental electronics components, and it's a good idea to understand how they work.

If you don't understand diodes, then you won't understand transistors, neither bipolar, or field effect transistors.

A useful feature of the diode is the exponential relationship between the forward current, and the voltage across the device.

To understand why a diode works it's necessary to understand the physics behind semiconductors.

This paper attempts to explain in the simplest possible terms how a diode works †

7.2 Silicon

Integrated circuits use single crystalline silicon. The silicon crystal is grown with the Czochralski method which forms an ingot that is cut into wafers. The wafer is a regular silicon crystal, although, it is not perfect.

A silicon crystal unit cell, as seen in Figure 1 is a diamond faced cubic with 8 atoms in the corners spaced at 0.543 nm, 6 at the center of the faces, and 4 atoms inside the unit cell at a nearest neighbor distance of 0.235 nm.

* It doesn't stop being magic just because you know how it works. Terry Pratchett, The Wee Free Men

† Simplify as much as possible, but no more. Einstein

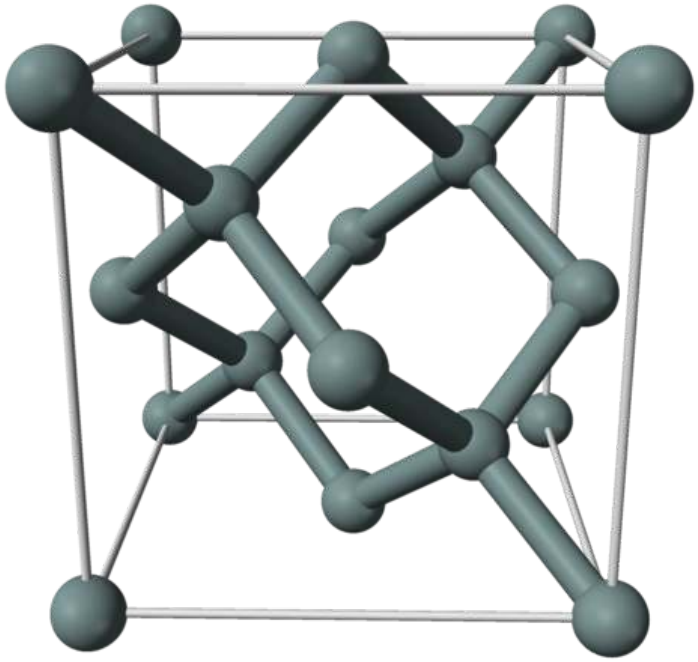


Figure 1: Silicon crystal unit cell. Image: Ben Mills, public domain, via Wikimedia Commons

As you hopefully know, the energy levels of an electron around a positive nucleus are quantized, and we call them orbitals (or shells). For an atom far away from any others, these orbitals, and energy levels are distinct. As we bring atoms closer together, the orbitals start to interact, and in a crystal, the distinct orbital energies split into bands of allowed energy states. No two electrons, or any Fermion (spin of 1/2), can occupy the same quantum state. We call the outermost "shared" orbital, or band, in a crystal the valence band. Hence covalent bonds.

If we assume the crystal is perfect, then at 0 Kelvin all electrons will be part of covalent bonds. Each silicon atom share 4 electrons with its neighbors. What we really mean when we say "share 4 electrons" is that the wave-functions of the outer orbitals interact, and we can no longer think of the orbitals as belonging to either of the silicon nuclei. All the neighbors atoms "share" electrons, and nowhere is there a vacant state, or a hole, in the valence band.

If such a crystal were to exist, where there were no holes in the valence band, and a net neutral charge, the crystal could not conduct any drift current. Electrons would move around continuously, swapping states, but there could be no net drift of charge carriers.

In an atom, or a crystal, there are also higher energy states where the carriers are "free" to move. We call these energy levels, or bands of energy levels, conduction bands. In singular form "conduction

band", refers to the lowest available energy level where the electrons are free to move.

Due to imperfectness of the silicon crystal, and non-zero temperature, there will be some electrons that achieve sufficient energy to jump to the conduction band. The electrons in the conduction band leave vacant states, or holes, in the valence band.

Electrons can move both in the conduction band, as free electrons, and in the valence band, as a positive particle, or hole. Both bands can support drift and diffusion currents.

7.3 Intrinsic carrier concentration

The intrinsic carrier concentration of silicon, or the density of free electrons and holes at a given temperature, is given by

$$n_i = \sqrt{N_c N_v} e^{-E_g/(2kT)} \tag{1}$$

where E_g is the bandgap energy of silicon (approx 1.12 eV), k is Boltzmann's constant, T is the temperature in Kelvin, N_c is the density of states in conduction band, and N_v is the density of states in the valence band.

The density of states are

$$N_c = 2\left[\frac{2\pi kT m_n^*}{h^2}\right]^{3/2} \quad N_v = 2\left[\frac{2\pi kT m_p^*}{h^2}\right]^{3/2}$$

where h is Planck's constant, m_n^* is the effective mass of electrons, and m_p^* is the effective mass of holes.

Leave it to engineers to simplify equations beyond understanding. Equation (1) is complicated, and the density of states includes the effective mass of electrons and holes, which is a parameter that depends on the curvature of the band structure. To engineers, this is too complicated, and n_i has been simplified so it "works" in daily calculation.

Through engineering simplification, however, physics understanding is lost.

In [1] they claim the intrinsic carrier concentration is a constant, although they do mention n_i doubles every 11 degrees Kelvin.

In BSIM 4.8 [2] the intrinsic carrier concentration is

$$n_i = 1.45e10 \frac{TNOM}{300.15} \sqrt{\frac{T}{300.15}} \exp^{21.5565981 - \frac{E_g}{2kT}}$$

Comparing the three models in Figure 2, we see the shape of BSIM and the full equation is almost the same, while the "doubling every 11 degrees" is just wrong.

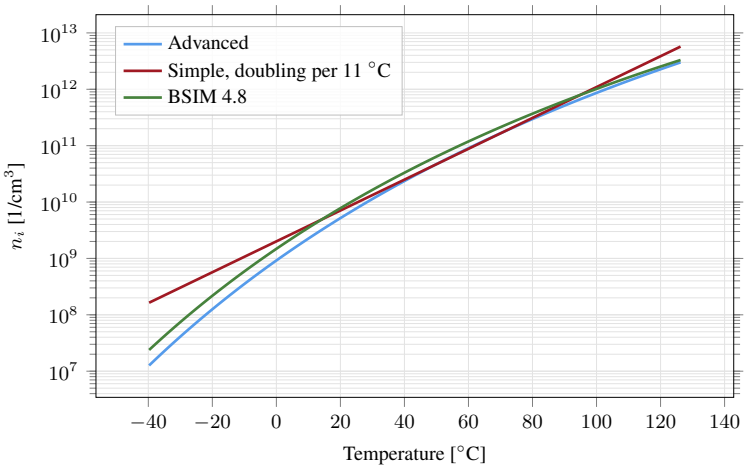


Figure 2: Intrinsic carrier concentration versus temperature

At room temperature the intrinsic carrier concentration is approximately $n_i = 1 \times 10^{16}$ carriers/m^3.

That may sound like a big number, however, if we calculate the electrons per um^3 it's $n_i = \frac{1 \times 10^{16}}{(1 \times 10^6)^3}$ carriers/μm^3 < 1, so there are really not that many free carriers in intrinsic silicon.

From Figure 2 we can see that n_i changes greatly as a function of temperature, but the understanding "why" is not easy to get from "doubling every 11 degrees". To understand the temperature behavior of diodes, we must understand Eq (1).

So where does Eq (1) come from? I find it unsatisfying if I don't understand where things come from. I like to understand why there is an exponential, or effective mass, or Planck's constant. If you're like me, then read the next section. If you don't care, and just want to memorize the equations, or indeed the number of intrinsic carrier concentration number at room temperature, then skip the next section.

7.4 It's all quantum

There are two components needed to determine how many electrons are in the conduction band. The density of available states, and the probability of an electron to be in that quantum state.

For the density of states we must turn to quantum mechanics. The probability amplitude of a particle can be described as

$$\psi = Ae^{i(k\mathbf{r}-\omega t)}$$

where k is the wave number, and ω is the angular frequency, and $\mathbf{r}$ is a spatial vector.

In one dimension we could write $\psi(x,t) = Ae^{i(kx-\omega t)}$

In classical physics we described the Energy of the system as

$$\frac{1}{2m}p^2 + V = E$$

where $p = mv$, m is the mass, v is the velocity and V is the potential.

In the quantum realm we must use the Schrodinger equation to compute the time evolution of the Energy, in one space dimension

$$-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial^2 x}\psi(x,t) + V(x)\psi(x,t) = i\hbar\frac{\partial}{\partial t}\psi(x,t)$$

where m is the mass, V is the potential, $\hbar = h/2\pi$.

We could rewrite the equation above as

$$\widehat{H}\psi(x,t) = i\hbar\frac{\partial}{\partial t}\psi(x,t) = \widehat{E}\psi(x,t)$$

where $\widehat{H}$ is sometimes called the *Hamiltonian* and is an operator, or something that act on the wave-function. In Feynman's Lectures on Physics Feynman called the Hamiltonian the *Energy Matrix* of a system. I like that better. The $\widehat{E}$ is the energy operator, something that operates on the wave-function to give the Energy.

We could re-arrange

$$[\widehat{H} - \widehat{E}]\psi(r,t) = 0$$

This is an equation with at least 5 unknowns, the space vector in three dimensions, time, and the energy matrix $\widehat{H}$.

The dimensions of the energy matrix depends on the system. The energy matrix further up is for one free electron. For an atom, the energy matrix will have more dimensions to describe the possible quantum states.

I consider all energy matricies as infinite dimensions, but most state transitions are so unlikely that they can be safely ignored.

I was watching Quantum computing in the 21st Century and David Jamison mentioned that the largest system we could today compute would be a system with about 30 electrons.

We know exactly how the equations of quantum mechanics appear to be, and they've proven extremely successful, we must make simplifications before we can predict how electrons behave in complicated systems like the silicon lattice with approximately 0.7 trillion electrons per cube micro meter. You can check the calculation

$$\left[\frac{1\ \mu\text{m}}{0.543\ \text{nm}}\right]^3 \times 8 \text{ atoms per unit cell} \times 14 \text{ electrons per atom}$$

7.4.1 Density of states

To compute "how many Energy states are there per unit volume in the conduction band", or the "density of states", we start with the three dimensional Schrodinger equation for a free electron

$$-\frac{\hbar^2}{2m}\nabla^2\psi = E\psi$$

I'm not going to repeat the computation here, but rather paraphrase the steps. You can find the full derivation in Solid State Electronic Devices.

The derivation starts by computing the density of states in the k-space, or momentum space,

$$N(dk) = \frac{2}{(2\pi)^p}dk$$

Where p is the number of dimensions (in our case 3).

The band structure $E(k)$ is used to convert to the density of states to a function of energy $N(E)$. The simplest band structure, and an approximation of the lowest conduction band is

$$E(k) = \frac{\hbar^2 k^2}{2m^*}$$

where m^* is the effective mass of the particle. It is within this effective mass that we "hide" the complexity of the actual three-dimensional crystal structure of silicon.

The effective mass when we compute the density of states is

$$m^* = \frac{\hbar^2}{\frac{d^2E}{dk^2}}$$

as such, the effective mass depends on the localized band structure of the silicon unit cell, and depends on direction of movement, strain of the silicon lattice, and probably other things.

In 3D, once we use the above equations, one can compute that the density of states per unit energy is

$$N(E)dE = \frac{\sqrt{2}}{\pi^2}\left(\frac{m^*}{\hbar^2}\right)^{3/2} E^{1/2}dE$$

In order to find the number of electrons, we need the probability of an electron being in a quantum state, which is given by the Fermi-Dirac distribution

$$f(E) = \frac{1}{e^{(E-E_F)/kT}+1} \tag{2}$$

where E is the energy of the electron, E_F is the Fermi level or chemical potential, k is Boltzmann's constant, and T is the temperature in Kelvin.

Fun fact, the Fermi level difference between two points is what you measure with a voltmeter.

If the $E - E_F > kT$, then we can start to ignore the +1 and the probability reduces to

$$f(E) = \frac{1}{e^{(E-E_F)/kT}} = e^{(E_F-E)/kT}$$

A few observations on the Fermi-Dirac distribution. If the Energy of a state is at the Fermi level, then $f(E) = \frac{1}{2}$, or a 50 % probability of being occupied.

In a metal, the Fermi level lies within a band, as the conduction band and valence band overlap. As a result, there are a bunch of free electrons that can move around. Metal does not have the same type of covalent bonds as silicon, but electrons are shared between a large part of the metal structure. I would also assume that the location of the Fermi level within the band structure explains the difference in conductivity of metals, as it would determined how many electrons are free to move.

In an insulator, the Fermi level lies in the bandgap between valence band and conduction band, and usually, the bandgap is large, so

there is a low probability of finding electrons in the conduction band.

In a semiconductor we also have a bandgap, but much lower energy than an insulator. If we have thermal equilibrium, no external forces, and we have an un-doped (intrinsic) silicon semiconductor, then the fermi level E_F lies half way between the conduction band edge E_C and the valence band edge E_V.

The bandgap is defined as the $E_C - E_V = E_g$, and we can use that to get $E_F - E_C = E_C - E_g/2 - E_C = -E_g/2$. This is why the bandgap of silicon keeps showing up in our diode equations.

The number of electrons per delta energy will then be given by

$$N_e dE = N(E)f(E)dE$$

, which can be integrated to get

$$n_e = 2\left(\frac{2\pi m^* kT}{h^2}\right)^{3/2} e^{(E_F - E_C)/kT}$$

For intrinsic silicon at thermal equilibrium, we could write

$$n_0 = 2\left(\frac{2\pi m^* kT}{h^2}\right)^{3/2} e^{-E_g/(2kT)} \tag{3}$$

As we can see, Equation (3) has the same coefficients and form as the computation in Equation (1). The difference is that we also have to account for holes. At thermal equilibrium and intrinsic silicon $n_i^2 = n_0 p_0$

7.4.2 How to think about electrons (and holes)

I've come to the realization that to imagine electrons as balls moving around in the silicon crystal is a bad mental image.

For example, for a metal-oxide-semiconductor field effect transistor (MOSFET) it is not the case that the electrons that form the inversion layer under strong inversion come from somewhere else. They are already at the silicon surface, but they are bound in covalent bonds (there are literally trillions of bound electrons in a typical transistor).

What happens is that the applied voltage at the gate shifts the energy bands close to the surface (or bends the bands in relation to the Fermi level), and the density of carriers in the conduction

band in that location changes, according to the type of derivations above.

Once the electrons are in the conduction band, then they follow the same equations as diffusion of a gas, Fick's law of diffusion. Any charge density concentration difference will give rise to a diffusion current given by

$$J_{\text{diffusion}} = -qD_n \frac{\partial \rho}{\partial x} \tag{4}$$

where J is the current density, q is the charge, ρ is the charge density, and D is a diffusion coefficient that through the Einstein relation can be expressed as $D = \mu kT$, where mobility $\mu = v_d/F$ is the ratio of drift velocity v_d to an applied force F.

To make matters more complicated, an inversion layer of a MOSFET is not in three dimensions, but rather a two dimensional electron gas, as the density of states is confined close to the silicon surface. As such, we should not expect the mobility of bulk silicon to be the same as the mobility of a MOSFET transistor.

7.5 Doping

We can change the property of silicon by introducing other elements, something we've called doping. Phosphor has one more electron than silicon, Boron has one less electron. Injecting these elements into the silicon crystal lattice changes the number of free electron/holes.

These days, we usually dope with ion implantation, while in the olden days, most doping was done by diffusion [3]. You'd paint something containing Boron on the silicon, and then heat it in a furnace to "diffuse" the Boron atoms into the silicon.

If we have an element with more electrons we call it a donor, and the donor concentration N_D.

The main effect of doping is that it changes the location of the Fermi level at thermal equilibirum. For donors, the Fermi level will shift closer to the conduction band, and increase the probability of free electrons, as determined by Equation (2).

Since the crystal now has an abundance of free electrons, which have negative charge, we call it n-type.

If the element has less electrons we call it an acceptor, and the acceptor concentration N_A. Since the crystal now has an abundance of free holes, we call it p-type.

The doped material does not have a net charge, however, as it's the same number of electrons and protons, so even though we dope silicon, it does remain neutral.

The doping concentrations are larger than the intrinsic carrier concentration, from maybe 10^{21} to 10^{27} carriers/m^3. To separate between these concentrations we use $p-$, p, $p+$ or $n-$, n, $n+$.

The number of electrons and holes in a n-type material is

$$n_n = N_D \, , \, p_n = \frac{n_i^2}{N_D}$$

and in a p-type material

$$p_p = N_A \, , \, n_p = \frac{n_i^2}{N_A}$$

In a p-type crystal there is a majority of holes, and a minority of electrons. Thus we name holes majority carriers, and electrons minority carriers. For n-type it's opposite.

7.6 PN junctions

Imagine an n-type material, and a p-type material, both are neutral in charge, because they have the same number of electrons and protons. Within both materials there are free electrons, and free holes which move around constantly.

Now imagine we bring the two materials together, and we call where they meet the junction. Some of the electrons in the n-type will wander across the junction to the p-type material, and visa versa. On the opposite side of the junction they might find an opposite charge, and might get locked in place. They will become stuck.

After a while, the diffusion of charges across the junction creates a depletion region with immobile charges. Where as the two materials used to be neutrally charged, there will now be a build up of negative charge on the p-side, and positive charge on the n-side.

7.6.1 Built-in voltage

The charge difference will create a field, and a built-in voltage will develop across the depletion region.

The density of free electrons in the conduction band is

$$n = \int_{E_C}^{\infty} N(E) f(E) dE$$

, where $N(E)$ is the density of states, and $f(E)$ is a probability of a electron being in that state (Equation (2)).

We could write the density of electrons on the n-side as

$$n_n = e^{E_{F_n}/kT} \int_{E_C}^{\infty} N_n(E) e^{-E/kT} dE$$

since the Fermi level is independent of the energy state of the electrons (I think).

The density of electrons on the p-side could be written as

$$n_p = e^{E_{F_p}/kT} \int_{E_C}^{\infty} N_p(E) e^{-E/kT} dE$$

If we assume that the density of states, $N_n(E)$ and $N_p(E)$ are the same, and the temperature is the same, then

$$\frac{n_n}{n_p} = \frac{e^{E_{F_n}/kT}}{e^{E_{F_p}/kT}} = e^{(E_{F_n} - E_{F_p})/kT}$$

The difference in Fermi levels is the built-in voltage multiplied by the unit charge.

$$E_{F_n} - E_{F_p} = q\Phi$$

and by substituting for the minority carrier concentration on the p-side we get

$$\frac{N_A N_D}{n_i^2} = e^{q\Phi_0/kT}$$

or rearranged to

$$\Phi_0 = \frac{kT}{q} ln \left(\frac{N_A N_D}{n_i^2} \right)$$

7.6.2 Current

The derivation of current is a bit involved, but let's try.

The hole concentration on the p-side and n-side could be written as

$$\frac{p_p}{p_n} = e^{-q\Phi_0/kT}$$

The negative sign is because the built in voltage is positive on the n-type side

Assume that $-x_{p0}$ is the start of the junction on the p-side, and x_{n0} is the start of the junction on the n-side.

Assume that we lift the p-side by a voltage qV

Then the hole concentration would change to

$$\frac{p(-x_{p0})}{p(x_{n0})} = e^{q(V-\Phi_0)/kT}$$

while on the n-side the hole concentration would be

$$\frac{p(x_{n0})}{p_n} = e^{qV/kT}$$

So the excess hole concentration on the n-side due to an increase of V would be

$$\Delta p_n = p(x_{n0}) - p_n = p_n \left(e^{qV/kT} - 1\right)$$

The diffusion current density, given by Equation (4) states

$$J(x_n) = -qD_p \frac{\partial \rho}{\partial x}$$

Thus we need to know the charge density as a function of x. I'm not sure why, but apparently it's

$$\partial\rho(x_n) = \Delta p_n e^{-x_n/L_p}$$

where L_p is a diffusion length. I think the equation above, the exponential decay as a function of length, is related to the probability of electron/hole recombination, and how the rate of recombination must be related to the exceess hole concentration, as such related to Exponential decay.

Anyhow, we can now compute the current density, and need only compute it for $x_n = 0$, so you can show it's

$$J(0) = q\frac{D_p}{L_p}p_n\left(e^{qV/kT} - 1\right)$$

which starts to look like the normal diode equation. The p_n is the minority concentration of holes on the n-side, which we've before estimated as $p_n = \frac{n_i^2}{N_D}$

We've only computed for holes, but there will be electron transport from the p-side to the n-side also.

We also need to multiply by the area of the diode to get current from current density. The full equation thus becomes

$$I = qAn_i^2\left(\frac{1}{N_A}\frac{D_n}{L_n} + \frac{1}{N_D}\frac{D_p}{L_p}\right)\left[e^{qV/kT} - 1\right]$$

where A is the area of the diode, D_n,D_p is the diffusion coefficient of electrons and holes and L_n,L_p is the diffusion length of electrons and holes.

Which we usually write as

$$I_D = I_S(e^{\frac{V_D}{V_T}} - 1), \text{ where } V_T = kT/q$$

7.6.3 Forward voltage temperature dependence

We can rearrange I_D equation to get

$$V_D = V_T \ln\left(\frac{I_D}{I_S}\right)$$

and at first glance, it appears like V_D has a positive temperature coefficient. That is, however, wrong.

First rewrite

$$V_D = V_T \ln I_D - V_T \ln I_S$$

$$\ln I_S = 2\ln n_i + \ln Aq\left(\frac{D_n}{L_n N_A} + \frac{D_p}{L_p N_D}\right)$$

Assume that diffusion coefficient ‡, and diffusion lengths are independent of temperature.

That leaves n_i that varies with temperature.

$$n_i = \sqrt{B_c B_v} T^{3/2} e^{\frac{-E_g}{2kT}}$$

where

$$B_c = 2\left[\frac{2\pi k m_n^*}{h^2}\right]^{3/2} \quad B_v = 2\left[\frac{2\pi k m_p^*}{h^2}\right]^{3/2}$$

$$2\ln n_i = 2\ln\sqrt{B_c B_v} + 3\ln T - \frac{V_G}{V_T}$$

with $V_G = E_G/q$ and inserting back into equation for V_D

$$V_D = \frac{kT}{q}(\ell - 3\ln T) + V_G$$

Where ℓ is temperature independent, and given by

$$\ell = \ln I_D - \ln\left(Aq\frac{D_n}{L_n N_A} + \frac{D_p}{L_p N_D}\right) - 2\ln\sqrt{B_c B_v}$$

From equations above we can see that at 0 K, we expect the diode voltage to be equal to the bandgap of silicon. Diodes don't work at 0 K though.

From the diode voltage relation we can calculate the derivative (ask your favorite chatbot to explain it to you), and you'll get

$$\frac{dV_D}{dT} = \frac{k}{q}\left(\ell - 3\ln T - 3\right).$$

and we can see the slope is negative with a factor of $\ln(T)$, which turns out to be very close to linear for the temperature range we're interested in (-40C to 125C)

The slope of the diode voltage can be seen to depend on the area, the current, doping, diffusion constant, diffusion length and the effective masses.

Figure 3 shows the V_D and the deviation of V_D from a straight line. The non-linear component of V_D is only a few mV. If we could

‡ From the Einstein relation $D = \mu kT$ it does appear that the diffusion coefficient increases with temperature, however, the mobility decreases with temperature. I'm unsure of whether the mobility decreases with the same rate though.

combine V_D with a voltage that increased with temperature, then we could get a stable voltage across temperature to within a few mV.

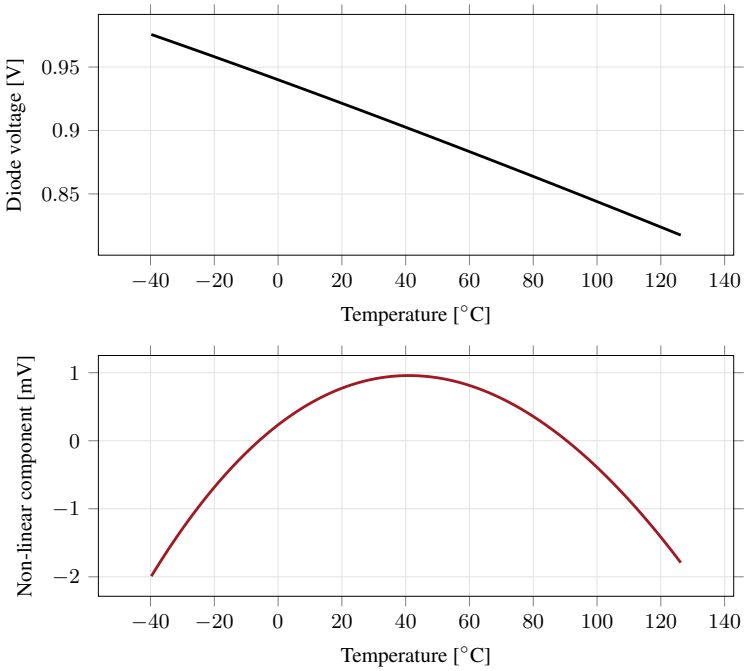


Figure 3: Diode forward voltage as a function of temperature

7.6.4 Current proportional to temperature

Assume we have a circuit like Figure 4.

Here we have two diodes, biased at different current densities. The voltage on the left diode V_{D1} is equal to the sum of the voltage on the right diode V_{D2} and voltage across the resistor R_1. The current in the two diodes are the same due to the current mirror. A such, we have that

$$I_S e^{\frac{qV_{D1}}{kT}} = N I_S e^{\frac{qV_{D2}}{kT}}$$

Taking logarithm of both sides, and rearranging, we see that

$$V_{D1} - V_{D2} = \frac{kT}{q} \ln N$$

Or that the difference between two diode voltages biased at different current densities is proportional to absolute temperature.

In the circuit above, this ΔV_D is across the resistor R_1, as such, the $I_D = \Delta V_D / R_1$. We have a current that is proportional to temperature.

If we copied the current, and sent it into a series combination of a resistor R_2 and a diode, we could scale the R_2 value to give us the exactly right slope to compensate for the negative slope of the V_D voltage.

The voltage across the resistor and diode would be constant over temperature, with the small exception of the non-linear component of V_D.

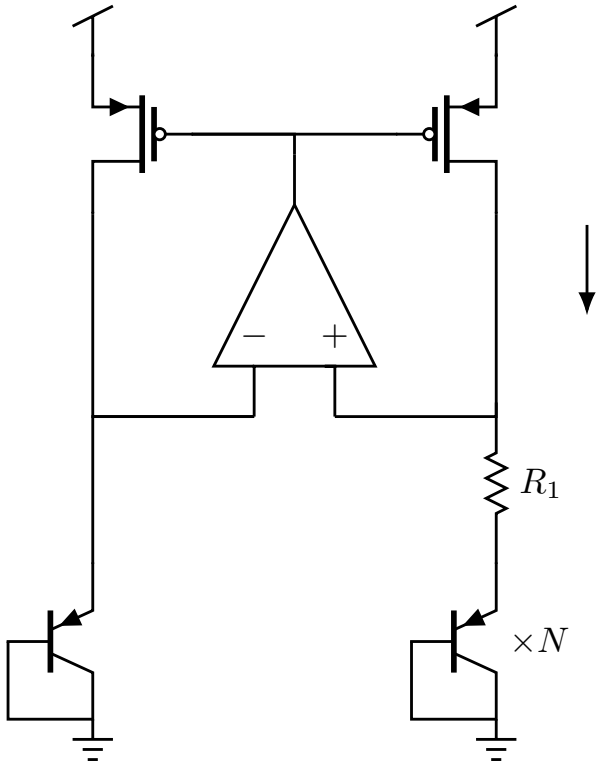


Figure 4: Circuit to generate a current proportional to kT

7.7 Reverse leakage

So far we have only looked at the diode in forward bias, where the exponential term dominates and the current is set by the diffusion saturation I_S. In integrated circuits we equally often care about what happens when the diode is reverse biased, with $V_D < 0$. The exponential term is then negligible, and you might expect the current to be zero. It is not.

What happens physically is that the reverse bias lifts the Fermi level on the n-side relative to the p-side. The built-in field grows, the depletion region widens, and any minority carrier that wanders into that region is immediately swept across to the other side. The question of "how much current" is therefore really the question of "how many minority carriers are produced per second, and where".

In a real n+/p-substrate junction there are two places where minority carriers appear:

1. In the *quasi-neutral* p-substrate, a short diffusion length away from the junction. These thermally generated electrons diffuse to the depletion edge and are then collected.
2. *Inside* the depletion region itself, through generation at Shockley-Read-Hall trap centers in the silicon bandgap.

The first gives the Shockley diffusion saturation current I_S, the second gives the Sah-Noyce-Shockley generation current I_{gen}. They have different temperature dependencies, and which one dominates depends on temperature, doping, and how clean the silicon is.

7.7.1 Diffusion leakage

$$I_S = qAn_i^2 \left(\frac{1}{N_A}\sqrt{\frac{D_n}{\tau_n}} + \frac{1}{N_D}\sqrt{\frac{D_p}{\tau_p}} \right)$$

This is exactly the I_S from the forward-bias derivation, just reinterpreted. Under reverse bias the minority-carrier concentration at the edge of the depletion region is forced to *zero* (every carrier that arrives is swept across). The concentration gradient between the bulk equilibrium $n_p = n_i^2/N_A$ on the p-side and zero at the depletion edge drives a steady diffusion of electrons toward the junction, and similarly of holes from the n-side.

The factor that matters for temperature is n_i^2. From Equation (1), $n_i^2 \propto T^3 \exp(-E_g/kT)$. The T^3 comes from the densities of states $N_c, N_v \propto T^{3/2}$, which is a genuine quantum-mechanical effect: as temperature rises, more states become thermally accessible above E_c and below E_v. The $\exp(-E_g/kT)$ dominates the slope, but the T^3 prefactor contributes a non-trivial correction that is bigger than people usually admit. The log-derivative of n_i^2 is

$$\frac{d \ln n_i^2}{dT} = \frac{E_g}{kT^2} + \frac{3}{T}$$

At $T = 300\,K$ the $3/T$ term is about 7 % of the exponential term; at $T = 1000\,K$ it is about 25 %. So the density-of-states prefactor visibly steepens the leakage curve at high temperature, and you cannot drop it if you care about anything beyond a back-of-envelope estimate.

The diffusion current doubles roughly every 4-5 K near room temperature - steeper than the familiar "reverse current doubles

every 10 K" rule of thumb, which belongs to the generation-limited regime [§].

For an n+/p-well antenna diode, the $1/N_D$ term is negligible because $N_D \gg N_A$, and I_S is set almost entirely by electron injection from the p-well.

Both D and τ depend on temperature too, and that matters once the sweep is several hundred kelvin wide. Phonon-limited mobility goes as $\mu \propto T^{-2.4}$ above $\sim 300\,K$, so via the Einstein relation $D = \mu kT/q \propto T^{-1.4}$. The SRH lifetime goes as $\tau = 1/(\sigma v_{th} N_t)$ with $v_{th} \propto \sqrt{T}$, so $\tau \propto T^{-1/2}$ (taking σ and N_t constant). The combination $\sqrt{D/\tau}$ in I_S therefore drifts as $T^{-0.45}$, so I_S at $1000\,K$ is about 0.6× what a "constant D, τ" model would predict. That is small compared to the many decades of swing in n_i^2, but it is the dominant reason the diffusion curve in Figure 5 starts to flatten at the top end. The script `ex/antenna_diode_leakage.py` includes both T-dependencies, and the interactive version lets you switch them off to see how little difference they make next to the bandgap.

7.7.2 Generation in the depletion region

$$I_{gen} = \frac{q A n_i W}{\tau_g}$$

$$W = \sqrt{\frac{2\varepsilon_{si}(\Phi_0 + V_R)}{q} \cdot \frac{N_A + N_D}{N_A N_D}}$$

Real silicon is not a perfect crystal. Process damage, residual metallic impurities and dangling bonds at interfaces create trap levels somewhere in the bandgap. A trap near mid-gap can capture an electron from the valence band (creating a hole) and then emit it to the conduction band (creating an electron). Inside a reverse-biased depletion region, both carriers are immediately swept apart by the field, the trap empties, and the cycle repeats. The net effect is a steady generation of electron-hole pairs that shows up as reverse current.

The Sah-Noyce-Shockley analysis gives the result above, where W is the depletion width and τ_g is an effective generation lifetime (typically a few τ_n). For a one-sided n+/p junction with $N_A \ll N_D$ the depletion width simplifies to

[§] This rule of thumb applies to the diffusion-dominated regime. For the generation-dominated regime, which is where most small junctions actually live at room temperature, the doubling interval is closer to 8-10 K because $I_{gen} \propto n_i$ rather than n_i^2.

$$W \approx \sqrt{\frac{2\varepsilon_{si}(\Phi_0 + V_R)}{qN_A}}$$

which sits almost entirely on the lightly doped substrate side.

The temperature scaling is now n_i, not n_i^2. The exponential in n_i has $E_g/(2kT)$ and the density-of-states prefactor is $T^{3/2}$ rather than T^3, so the slope is roughly half that of I_S. I_{gen} doubles every 8-10 K near room temperature. At low and moderate temperatures, where n_i is small, this slower-scaling term still dominates because it has the smaller exponent. As temperature rises and n_i grows by many decades, the steeper $I_S \propto n_i^2$ eventually overtakes it.

The $1/\tau_g$ in I_{gen} also drifts with temperature: with $\tau \propto T^{-1/2}$ the generation term picks up an extra $\sqrt{T}$ factor on top of the $n_i W$ scaling, which is why at the top of the sweep I_{gen} does not flatten out as much as one might expect from n_i alone.

The reverse bias V_R enters through W, so I_{gen} has a weak $\sqrt{V_R}$ dependence. Diffusion current I_S is essentially independent of bias once the junction is reverse-biased by more than a few kT/q. This is the standard way to separate the two experimentally: I_S saturates, I_{gen} grows with $\sqrt{V_R}$.

7.7.3 Plasma charging currents

$$J_{plasma} \sim \text{1-10}\,\text{mA/cm}^2$$

$$I_{ant} = J_{net} \cdot A_{antenna}$$

How much current does the diode actually have to sink? A reactive-ion plasma delivers ion and electron fluxes to the wafer surface, each typically in the 1-10 mA/cm^2 range [4]. If the two fluxes were perfectly balanced everywhere, no floating conductor would charge up and there would be nothing to drain. They never are, and two mechanisms produce a net DC current that the antenna diode must shunt:

- **Macroscopic non-uniformity.** Ion and electron fluxes vary across the wafer because of source geometry, magnetic-field shaping and edge effects. The local mismatch is typically a few percent of J_{plasma}, giving a net DC charging current density on the order of 10 $\mu\text{A/cm}^2$ to 0.1 mA/cm^2 collected by any conductor connected to that area [4].

- **Electron shading.** In a high-aspect-ratio opening through photoresist, vertical ions reach the bottom of the feature while near-isotropic electrons are blocked by the sidewalls. The bottom charges positively regardless of plasma uniformity. Hashimoto identified this as the dominant damage mechanism for deep-submicron processes [5], and it is what makes high-density-plasma etch and HDP-CVD particularly aggressive.

A metal trace that the routing tool calls the *antenna* collects the net plasma current density over its full area $A_{antenna}$ and funnels it onto the much smaller gate. Foundry rules cap the *antenna ratio* $A_{antenna}/A_{gate}$ at typically a few hundred to a few thousand, but even within those limits a net plasma current of $\sim 10\,\mu\text{A/cm}^2$ over a $10^4\ \mu m^2$ antenna gives $\sim 1\ \mu A$ injected onto a sub-μm^2 gate. With no discharge path, that current would charge the gate node and force Fowler-Nordheim tunneling through the oxide (fields above $\sim 10\,\text{MV/cm}$). Once tunneling starts, defects accumulate and the breakdown lifetime collapses [6].

The antenna diode bypasses this failure by giving the antenna a reverse-biased path to substrate that conducts at far lower voltage than the oxide. As long as the diode reverse leakage at the relevant process temperature exceeds the antenna-collected charging current, the gate voltage stays clamped well below the FN threshold and the oxide is safe. This is what makes Figure 5 a sizing tool: at the coldest plasma step the leakage per μm^2 is only $\sim 1\,\text{fA}$, so a small antenna ratio or a generous diode area is required to maintain $I_{leak} \gtrsim I_{ant}$.

7.7.4 Antenna ndiode, 200 K to 1000 K

A useful integrated-circuit example is the *antenna diode*. During plasma etch and ion-implant steps in fabrication, long metal traces collect charge. If the trace is connected to a transistor gate, the gate dielectric can break down before the chip ever sees a power supply. Foundry design rules specify maximum *antenna ratios* (metal area to gate area) per layer; if a net exceeds the ratio, the routing tool inserts an *antenna diode* on the net to bleed the plasma charge to substrate. In normal operation that diode sits reverse-biased and must contribute negligible static current.

The diode itself is simply an n+ source/drain implant placed in the NMOS p-well. The relevant doping is:

- p-substrate (handle wafer): $N_{sub} \approx 10^{15}\ cm^{-3}$
- p-well (retrograde, set by short-channel control): $N_A \approx 10^{17}$ – $10^{18}\ cm^{-3}$

- n+ source/drain: $N_D \approx 10^{20}\,cm^{-3}$

It is the p-well doping that matters at the junction, because the n+ sits inside the well, not directly on substrate. The effective N_A seen by the antenna ndiode is therefore one to two orders of magnitude higher than the wafer doping, which makes the depletion narrower and the reverse leakage *smaller* than a naive "n+/p-substrate" estimate would suggest.

Figure 5 plots the reverse leakage of such a junction: $N_A = 10^{17}\,cm^{-3}$ (p-well), $N_D = 10^{20}\,cm^{-3}$ (n+ S/D), $V_R = 1\,V$, swept from 200 K to 1000 K. The current is plotted as a *leakage density* in A/μm^2, so multiplying by the actual junction area in μm^2 gives the absolute current - a $0.2 \times 0.2\,\mu m^2$ antenna ndiode is the curve times 0.04, a $1 \times 1\,\mu m^2$ diode reads off directly. The script is `ex/antenna_diode_leakage.py` and reuses the $n_i(T)$ derivation from `ex/vd.py`. Both exist as interactive pages: antenna diode leakage, where the doping, the reverse bias and the junction area are sliders, and diode vs temperature for the $n_i(T)$ model on its own.

The two shaded bands mark where plasma-induced damage actually happens during fabrication, which is the *only* time an antenna diode does any work. The cooler band is plasma etch (reactive ion etch, with the wafer chuck cooled to roughly 25-100 °C, so ~ 300-375 K); the hotter band is plasma deposition steps such as PECVD, HDP-CVD and sputter (~ 200-400 °C, so ~ 470-675 K, with 400 °C a hard upper limit set by BEOL metal reliability). Across this span the leakage available to bleed plasma charge ranges from roughly 1 fA/μm^2 at room-temperature etch to ~ 6 nA/μm^2 at 400 °C deposition - nearly seven decades of variation depending only on which process step you are in.

Three things to take away from the figure:

- Generation current dominates from cryogenic temperatures up to roughly 575 K, exactly where the steeper n_i^2 slope of the diffusion term catches up. Above that the diode is in the "diffusion-limited" regime. The crossover happens to fall right in the middle of the plasma-deposition band, so antenna diodes during deposition steps see contributions from both mechanisms.
- The leakage per μm^2 swings nearly seven decades over the wafer-fab thermal range. An antenna diode that easily sinks the plasma charge during a 400 °C HDP-CVD step may be orders of magnitude too small during a room-temperature metal etch. The worst case for sizing is therefore the *coldest* plasma step.

- The slope on a log axis is set by the $\exp(-E_g/(2kT))$ in n_i plus the $T^{3/2}$ density-of-states prefactor. Every junction in a given silicon process scales with temperature the same way; only the absolute level changes with area, doping and lifetime.

It is also worth noting that at the upper end of the sweep n_i approaches and eventually exceeds N_A. The diode then loses junction behaviour and the silicon behaves as an intrinsic resistor. This is why high-temperature electronics typically uses wide-bandgap materials such as silicon carbide.

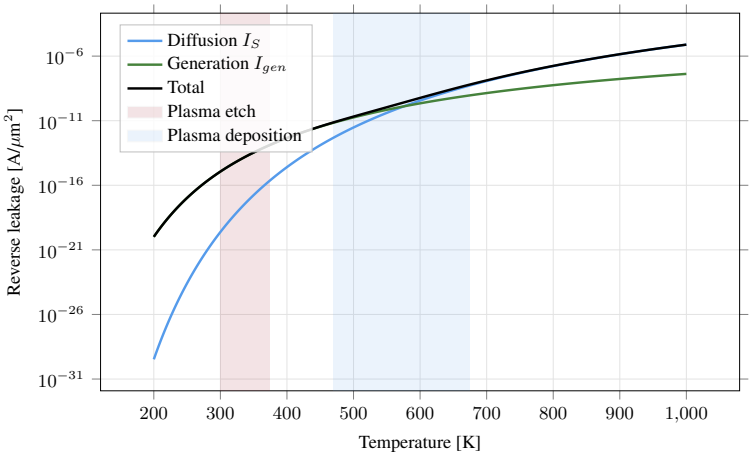


Figure 5: Reverse leakage density (A/um^2) of an n+/p-well antenna ndiode, from 200 K to 1000 K, $V_R = 1$ V. Multiply by your actual diode area in um^2. Shaded bands are the wafer temperature during plasma etch (300-375 K) and plasma deposition (470-675 K), the steps at which plasma-induced gate damage occurs.

7.7.5 Wires below 1 mm

$$I_{ant} = J_{net} \cdot W_{wire} \cdot L_{wire}$$

The antenna current scales linearly with wire length, so for wire lengths below 1 mm - which covers practically everything that stays inside a single block or hierarchical cell - the sizing problem is concrete. Take a minimum-width interconnect, $W_{wire} \approx 0.1\,\mu m$, and the worst-case cold-etch net plasma current density, $J_{net} \approx 10\,\mu\text{A/cm}^2$:

L_{wire}	$A_{antenna}$	I_{ant} at room-T etch
$1\,\mu m$	$0.1\,\mu m^2$	~ 1 fA
$10\,\mu m$	$1\,\mu m^2$	~ 10 fA
$100\,\mu m$	$10\,\mu m^2$	~ 100 fA
$1\,mm$	$100\,\mu m^2$	~ 1 pA

Now compare with Figure 5. At room temperature the antenna ndiode leaks only $\sim 1\,\mathrm{fA}/\mu m^2$. A 1 mm wire collecting 1 pA would therefore need a diode active area of $\sim 900\,\mu m^2$ (a $30 \times 30\,\mu m$ device) to keep the gate clamped during cold etch, which is clearly impractical. The same 1 mm wire is comfortably protected by a $0.15\,\mu m^2$ diode at a $200\,^\circ C$ deposition step ($\sim 3\,\mathrm{pA}/\mu m^2$) and by a sub-$0.01\,\mu m^2$ diode at $400\,^\circ C$.

Two consequences:

- For wires shorter than a few μm, the collected charging current is below the room-temperature diode leakage even for the smallest practical antenna ndiode. No protection is needed - which is why foundry antenna rules always have a length-threshold exemption.
- For long wires the cold-etch step dominates the sizing, and a single big diode is rarely the right answer. The routing tool instead *jumps* the long net up to a higher metal layer (patterned only after the lower stack already protects the gate), or distributes many small antenna ndiodes along the wire so each one only has to drain its local section of the antenna current.

Wider metals (upper-stack power and clock) scale antenna area linearly with width, so a $1\,\mu m$-wide, 1 mm-long top-metal trace collects 10 pA rather than 1, and the same conclusions apply with one decade less margin.

7.8 Equations aren't real

> Nature does not care about equations. It just is.

We know, at the fundamental level, nature appears to obey the mathematics on quantum mechanics, however, due to the complexity of nature, it's not possible today (which is not the same as impossible), to compute exactly how the current in a diode works. We can get close, by measuring a diode we know well, and hope that the next time we make the same diode, the behavior will be the same.

As such, I want to warn you about the "lies" or "simplifications" we tell you. Take the diode equation above, some parts, like the intrinsic carrier concentration n_i has roots directly from quantum mechanics, with few simplifications, which means it's likely solid truth, at least for a single unit cell.

But there is no reason nature should make all unit cells the same, and infact, we know they are not the same, we put in dopants. As

we scale down to a few nano-meter transistors the simplification that "all unit cells of silicon are the same, and extend to infinity" is no longer true, and must be taken into account in how we describe reality.

Other parts, like the exact value of the bandgap E_g, the diffusion constant D_p or diffusion length L_p are macroscopic phenomena, we can't expect them to be 100 % true. The values would be based on measurement, but not always exact, and maybe, if you rotate your diode 90 degrees on the integrated circuit, the values could be different.

You should realize that the consequence of our imperfection is that the equations in electronics should always be taken with a grain of salt.

Nature does not care about your equations. Nature will easily have the superposition of trillions of electrons, and they don't have to agree with your equations.

But most of the time, the behavior is similar.

7.9 Summary

The one-page version of this chapter:

- Silicon's bandgap makes it a semiconductor: few free carriers at room temperature, exponentially more when heated
- Doping moves the Fermi level - towards the conduction band with donors, towards the valence band with acceptors
- Join n to p and diffusion fights drift until the depletion field balances: the built-in voltage
- Forward bias lowers the barrier exponentially: $I = I_S(e^{V/V_T} - 1)$
- The diode voltage falls roughly 2 mV/K and extrapolates to the bandgap at 0 K - the seed of every bandgap reference
- Reverse leakage grows exponentially with temperature: an antenna diode sized at etch temperature is a different diode at deposition

Would you like to know more?

[1] T. C. Carusone, D. Johns, and K. Martin, *Analog integrated circuit design*. Wiley, 2011 [Online]. Available: https://books.google.no/books?id=1OIJZzLvVhcC

[2] Berkeley, "Berkeley short-channel IGFET model." [Online]. Available: http://bsim.berkeley.edu/models/bsim4/

[3] T. Masuhara and R. Muller, "Complementary DMOS process for LSI," *IEEE Journal of Solid-State Circuits*, vol. 11, no. 4, pp. 453–458, 1976, doi: 10.1109/JSSC.1976.1050758.

[4] K. P. Cheung, *Plasma charging damage*. Springer-Verlag London, 2001 [Online]. Available: https://link.springer.com/book/10.1007/978-1-4471-0247-2

[5] K. Hashimoto, "Charge damage caused by electron shading effect," *Japanese Journal of Applied Physics*, vol. 33, pp. 6013–6018, 1994, doi: 10.1143/JJAP.33.6013. [Online]. Available: https://iopscience.iop.org/article/10.1143/JJAP.33.6013

[6] S. Krishnan, A. Amerasekera, S. Rangan, and S. Aur, "Antenna device reliability for ULSI processing," in *International electron devices meeting (IEDM) technical digest*, 1998, pp. 247–250, doi: 10.1109/IEDM.1998.746430.

MOSFETs 8

I'm stunned if you've never heard the word "transistor". I think most people have heard the word. What I find funny is that almost nobody understands in full detail how transistors work.

Through my 30 year venture into the world of electronics I've met "analog designers", or people that should understand exactly how transistors work. I used to hire analog designers, and I've interviewed hundred plus "analog designers" in my 8 years as manager and I've met hundreds of students of analog design. I would go as far as to say none of them know everything about transistors, including myself.

Most of the people I've met have a good brain, so that is not the reason they don't understand. Transistors are incredibly complicated! I say this, because if at some point in this document, **you** don't understand, then don't worry, you are not alone.

In this document I'm focusing on Metal Oxide Semiconductor Field Effect Transistors (MOSFETs), and ignore all other transistors.

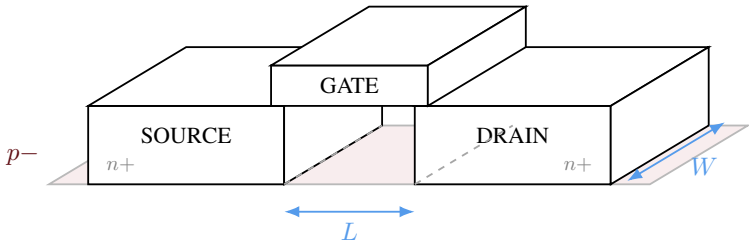
8.1 Metal Oxide Semiconductor

The first part of the MOSFET name illustrates the 3 dimensional composition of the transistor. Take a semiconductor (Silicon), grow some oxide (Silicon Oxide, SiO2), and place a metal, or conductive, gate on top of the oxide. With those three components we can build our transistor.

Something like the cartoon below where only the Metal (gate) of the MOS name is shown.

The oxide and the silicon bulk is not visible, but you can imagine them to be underneath the gate, with a thin oxide (a few nano meters thick) and the silicon the transparent part of the picture.

The length (L), and width (W) of the MOS is annotated in blue.



8.1 Metal Oxide Semiconductor 75
8.2 Field Effect 77
8.3 Analog transistors in the books 81
8.4 Transistors in weak inversion 83
8.5 Transistors in strong inversion . . 86
8.6 Introduction to behavior 89
8.6.1 Drain Source Current 89
8.6.2 Gate-source voltage 90
8.6.3 Inversion level . . . 91
8.6.4 Drain source voltage 95
8.6.5 Strong inversion . . 95
8.6.6 Low frequency model 97
8.6.7 Transconductance . 98
8.6.8 Intrinsic gain 99
8.6.9 Body effect 100
8.6.10 High frequency model 101
8.6.11 Be careful with Cgd (blame Miller) . . . 102
8.6.12 Transit frequency . 103
8.7 Weak inversion . . 104
8.8 Velocity saturation 106
8.8.1 Square law model . 107
8.8.2 Mobility Degradation 108
8.8.3 What about holes (PMOS) 108
8.9 OTHER 109
8.9.1 Drain induced barrier lowering (DIBL) 110
8.9.2 Well Proximity Effect (WPE) 111
8.9.3 Stress effects 111
8.9.4 Gate current 112
8.9.5 Hot carrier injection 113
8.9.6 Channel initiated secondary-electron (CHISEL) 113
8.10 Variability 114
8.10.1 Voltage variation . 115
8.10.2 Systematic variations 115
8.10.3 Process variations . 116
8.10.4 Process corners . . 116

Figure 1: 3D crossection of a transistor

MOSFETs come in two main types. There is NMOS, and PMOS. The symbols are as shown below. The NMOS is MN1 and PMOS is MP1.

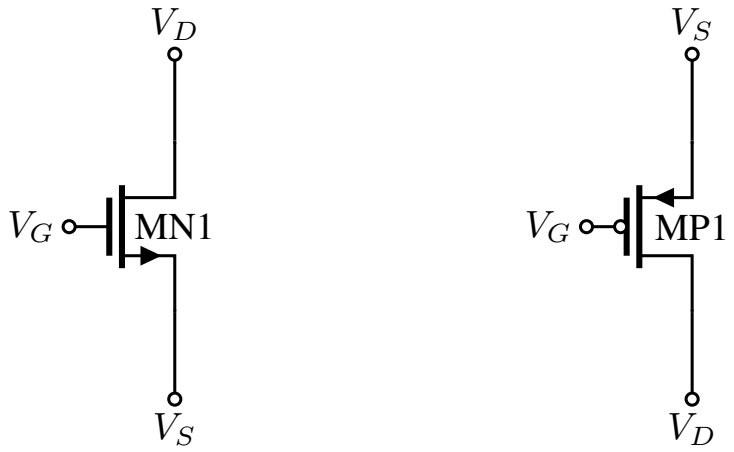


Figure 2: Transistor symbols

The MOS part of the name can be seen in MN1, where V_G is the gate connected to a vertical line (metal), a space (oxide), and another vertical line (the silicon substrate or silicon bulk).

On the sides of the gate we have two connections, a drain V_D and a source V_S.

If we have a sufficient voltage between gate and source V_{GS}, then the transistor will conduct from drain to source. If the voltage is too low, then there will not be much current.

The "source" name is because that's where the charge carrier (electrons) come from, they come from the source, and flow towards the drain. As you may remember, the "current", as we've defined it, flows opposite of the electron current, from drain to source.

The PMOS works in a similar manner, however, the PMOS is made of a different type of silicon, where the dominant charge carrier is holes in the valence band. As a result, the gate-source voltage needs to be negative for the PMOS to conduct.

In a PMOS the holes come from the source, and flow to the drain. Since holes are positive charge carriers, the current flows from source to drain.

In most MOSFETs there is no physical difference between source and drain. If you flip the transistor it would work almost exactly the same.

8.2 Field Effect

Imagine that the bulk (the empty space underneath the gate), and the source is connected to 0 V. Assume that the gate is 0 V.

In the source and drain parts of the transistor there is an abundance of **free** electrons that can move around, exactly like in a metal conductor, however, underneath the gate there are almost no **free** electrons.

There are electrons underneath the gate though, trillions upon trillions of electrons, but they are stuck in covalent bonds between the Silicon atoms, and around the nucleus of the Silicon atoms. These electrons are what we call bound electrons, they cannot move, or more precisely, they cannot contribute to current (because they do move, all the time, but mostly around the atoms).

Imagine that your eyes could see the free electrons as a blue fluorescent color. What you would see is a bright blue drain, and bright blue source, but no color underneath the gate.

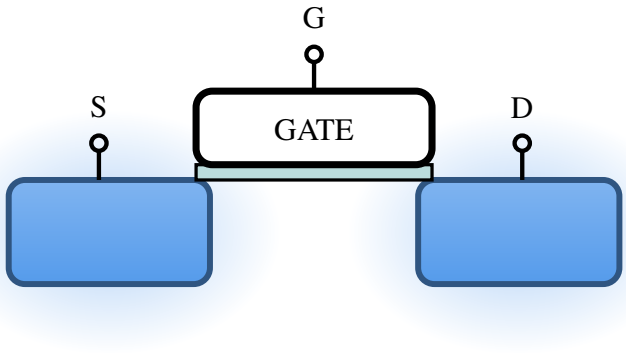


Figure 3: MOSFET in "off" state

As you increase the gate voltage, the color underneath the gate would change. First, you would think there might be some blue color, but it would be barely noticeable.

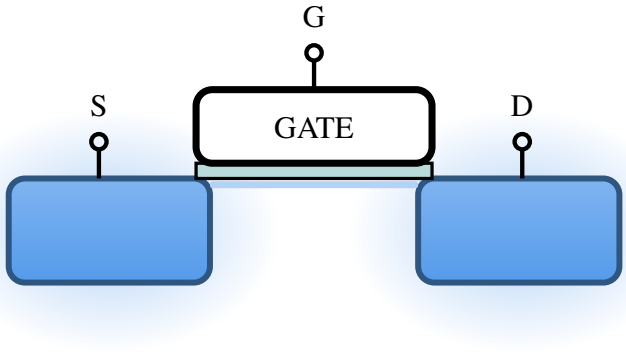


Figure 4: MOSFET in subthreshold

At a certain voltage, suddenly, there would be a thin blue sheet underneath the gate. You'd have to zoom in to see it, in reality it's an ultra-thin, 2 dimensional electron sheet.

As you continue to increase the gate voltage the blue color would become a little brighter, but not much.

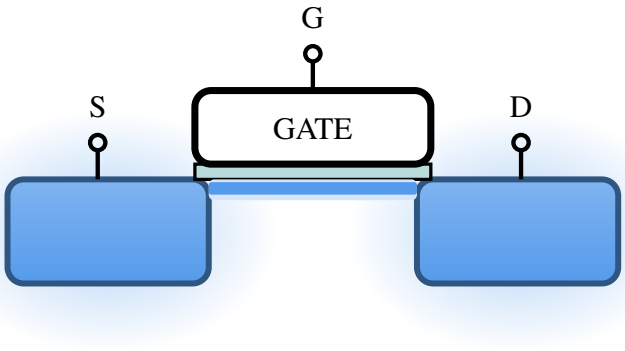


Figure 5: MOSFET in strong inversion

This thin blue sheet extends from source to drain, and create a conductive channel where the electrons can move from source to drain (or drain to source), exactly like a resistor. The conductance of the sheet is the same as the brightness, higher gate source voltage, more bright blue, higher conductance, less resistance.

Assume you raise the drain voltage. The electrons would move from source to drain proportional to the voltage. How many electrons could move would depend on the gate voltage.

If the gate voltage was low, then there is low density of electrons in the sheet, and low current.

If the gate voltage is high, then the electron density in the sheet is high, and there can be a high current, although, the electrons do have a maximum speed, so at some point the current does not change as fast with the gate voltage.

At a certain drain voltage you would see the blue color disappear close to the drain and there would be a gap in the sheet.

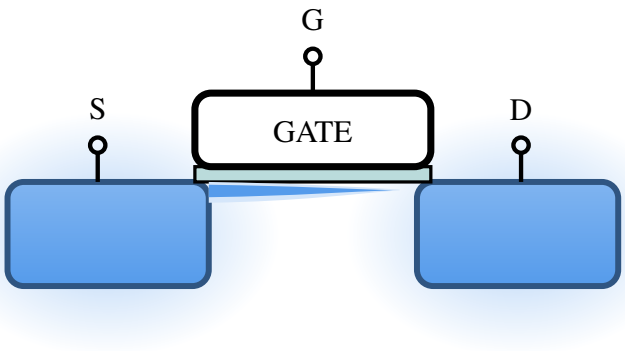


Figure 6: MOSFET in strong inversion and saturation

That could make you think the current would stop, but it turns out, that the electrons close to drain get swept across the gap because the electric field is so high from the edge of the sheet to the drain.

As you continue to increase the drain voltage, the gap increases, but the current does not really increase that much. It's this exact feature that makes transistors so attractive in analog circuits. I can create a current from drain to source that does not depend much on the drain to source voltage! That's why we sometimes imagine transistors as a "trans-conductance". The conductance between drain and source depends on the voltage somewhere else, the gate-source voltage.

And now you may think you understand how the transistor works. By changing the gate voltage, we can change the electron current from source to drain. We can turn on, and off, currents, creating a 0 and 1 state.

For example, if I take a PMOS and connect the source to a high voltage, the drain to an output, and an NMOS with the source to ground and the drain to the output, and connect the gates together, I would have the simplest logic gate, an inverter, as shown below.

If the input V_{in} is a high voltage, then the output V_{out} is a low voltage, because the NMOS is on. If the input V_{in} is a low voltage, then the output V_{out} is a high voltage, because the PMOS is on.

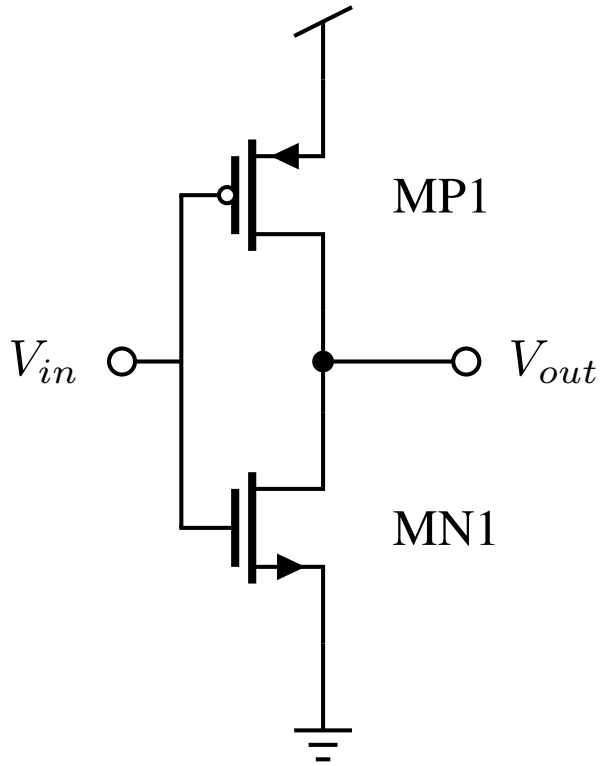


Figure 7: Inverter

I can now build more complex "logic gates". The one below is a Not-AND gate (NAND). If both inputs (A and B) are high, then the output is low (both NMOS are on). Otherwise, the output is high.

I find it amazing that all digital computers in existence can be constructed from the NAND gate. In principle, it's the only logic gate you need. If you actually did construct computers from NANDs only, they would be costly, and consume lots of power. There are smarter ways to use the transistors.

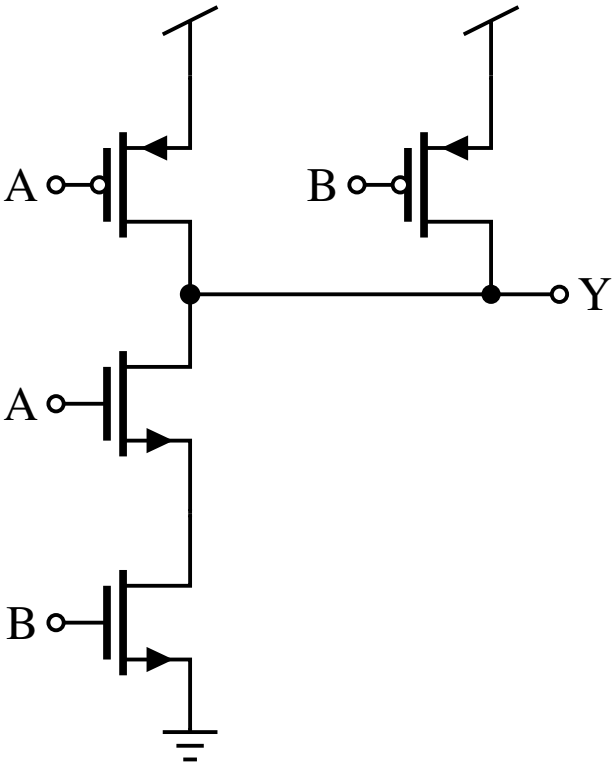


Figure 8: NAND

You may be too young to have seen the Matrix, but now is the time to decide between the red pill and the blue pill.

The red will start your journey to discover the reality behind the transistor, the blue pill will return you to your normal life, and you can continue to think that you now understand how transistors work.

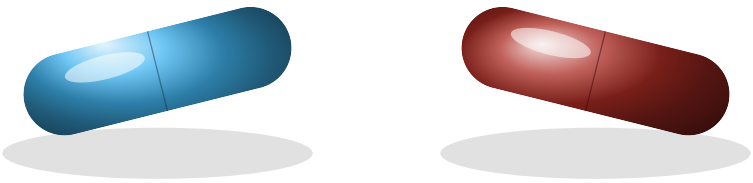


Figure 9: The choice

Because:

- Why did the area underneath the gate turn blue?
- Why is it only a thin sheet that turns blue?
- Where did the electrons for the sheet come from?
- Why did the blue color change suddenly?
- How does the brightness of the blue change with gate-source voltage?
- How can the electrons stay in that sheet when we connect the bulk to 0 V?
- Why is there not a current from the bulk (0 V) to drain?
- Why don't the electrons jump from source to drain? It's a gap, the same as from the sheet to drain?

And did you realize I never in this chapter explained how the field effect worked?

Someday, I may write all the details, if I ever understand it all. For now, I hope that the sections below will help you a bit.

8.3 Analog transistors in the books

In the books we learn the equations for weak inversion

$$I_D \propto e^{(V_{gs}-V_{tn})/nV_T}$$

, where I_D is the drain current, V_{gs} is the gate source voltage, V_{tn} is the threshold voltage, n is the slope factor (more on that later) and $V_T = kT/q$, where k is Boltzmann's constant, T is the temperature in Kelvin and q is the unit charge

The equation is similar to bipolar and diode equations, because the physics is the same.

The drain current in weak inversion is mostly a diffusion current and relates to the density of electrons in the conduction band (for an NMOS), which can be computed from the density of available energy states, and the Fermi-Dirac distribution.

$$n = \int_{E_C}^{\infty} N(E) \frac{1}{e^{(E-E_F)/kT} + 1} dE$$

, where n is the density of electrons in the conduction band, $N(E)$ is the density of available energy states, E is the integration variable (and the energy) and E_F is the Fermi-level.

Maybe the equation looks complicated, but it's really "Multiply the available energy state with the probability of being in that state, and sum for all available energy states".

Changing the voltage changes the number of free electrons, simply because we bring the conduction band closer to the Fermi level.

The Fermi level is just something we invented, and just means "If there was an quantum state at the Fermi level Energy, then it would have a 50 % probability of being occupied by an electron".

In the equation above, moving the conduction band edge is equivalent to reducing the E_C. As such, more of the Fermi-Dirac distribution has available energy states $N(E)$, and the density of electrons n in conduction band becomes higher.

In strong inversion, the MOSFET is more like a voltage controlled resistor with a conductance that is proportional to gate-source voltage.

The density of electrons increases because we bend the conduction band beyond the Fermi level, as a result, most of the available energy states in the conduction band are filled by electrons.

Electrons are only free to move, however, close to the surface of the silicon, as far away from the surface, we don't feel the effects of the gate-source voltage, and the conduction band stays at the same energy. As a result, electrons form a 2 dimensional electron gas close to the silicon surface. What we call an inversion layer.

Once we have that electron gas, or inversion layer, we have a connection between the drain and source n-type regions, and the current can be estimated by a drift current. Parts of the diffusion current will still be there, but much smaller magnitude than the drift current, so we drop the diffusion current, and get

$$I_D = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}(V_{gs} - V_{tn})^2$$

The equations in the books are good to give a physical understanding of what happens. Although, we tend to forget that everybody forgets.

We teach quantum physics one year, and how to compute the density of states $N(E)$ from Schrodinger, the wave-function and Fermi-Dirac distribution.

Next year we talk about semiconductors, crystal lattice, band structure (density of states as a function of space), energy diagrams (band structure is complex, so we just use the lowest conduction band and highest valence band), doping to shift the Fermi level, and how we can create PN-junctions, bipolars and MOSFETS.

The year after we teach the current equations for MOSFETs, and the books don't have the link back to solid-state physics, after all, we already told the students that, they should remember!

I think, quite often, we just end up with confused students. And I don't think it's necessary to end up with confused students. Maybe sometimes we end up with confused students because the Professors can't necessarily remember where the equations come from either, nor how electrons and holes really behave.

It's not necessary for an analog design student to remember how to compute the density of available energy states from Schrodinger and the wave function. If we wanted to use the Dirac equation (the relativistic wave equation, which brings in spin and the magnetic interactions properly) and the wave function to compute how a Silicon atom actually behaves, I don't think we can. As far as I've been able to figure out, it's not possible to have a closed form solution (symbolic), nor is it possible with supercomputers to do a numeric time-evolution of the states in a single Silicon atom with all the inter-particle interactions, space, momentum, spins, electric fields and magnetic fields.

But we can make sure we connect the links from Schrodinger to the MOSFET equations, the short version of that was above, but the following sections tries to explain with words how the transistor actually works.

I'm not going to give all the equations and all the maths. For that, there are excellent books and resources. I would recommend Mark Lundstrom for the best in detail description of MOSFETs.

8.4 Transistors in weak inversion

Consider the cartoon below which shows the hole concentration in the valence band, and electron concentration in the conduction band versus the x direction of the transistor.

For the moment we'll ignore the field effect of the gate, and how that modulates the hole concentration underneath the gate.

If you're familiar with bipolars, then you may think I've drawn the wrong transistor, because you see an NPN bipolar transistor. The picture is correct, however, this is how a normal MOSFET looks. It's actually also a NPN bipolar transistor, but we don't usually use that part (you'll see more when we get to ESD)

In the source we've doped with donors, and have an abundance of free electrons. Underneath the gate, or the bulk, we have doped with acceptors, and have an abundance of holes.

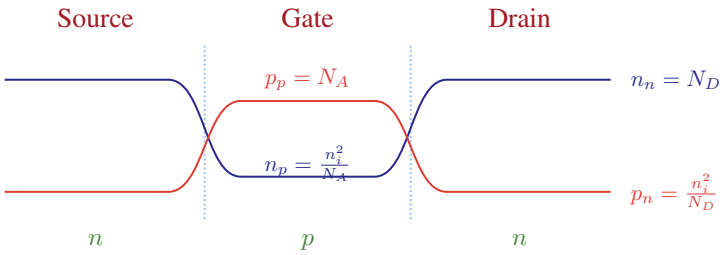


Figure 10: Charge carrier density in a MOSFET

Let's consider electron current for now, and only look at the conduction band.

An electron in the source would see an energy barrier of ϕ_B, and most electrons would be turned around at the barrier. Some, however, do have the energy to traverse the barrier and flow through the bulk. Not all of them would reach the bulk, due to recombination, but let's assume the bulk is short, and all electrons injected into the bulk show up at the drain.

At the drain side they would fall down the potential barrier to the drain. The same process would happen in reverse, from drain to source.

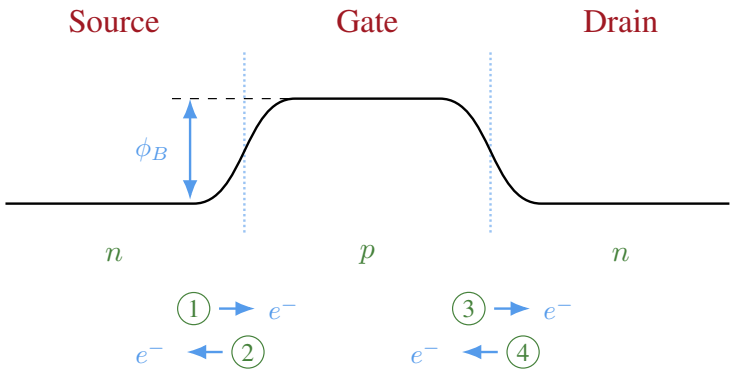


Figure 11: MOSFET subthreshold , $V_{DS} = 0$

There would also be hole currents flowing between source/bulk/drain and vice versa

Assume source and drain are at the same potential, then the sum of all currents (1,2,3,4) for both electrons and holes in Figure 11 must equal zero.

Assume that we increase the drain voltage, as shown in Figure 12. Increasing the drain voltage is the same as reducing the conduction band in the drain.

Since there now is a higher barrier from drain to bulk, it's now much less probable that electrons are injected from drain to bulk.

Now the sum of all currents would not equal zero, as the 1 and 3 currents are larger than 2 and 4.

As such, there would be a net flow of electron current from source to drain.

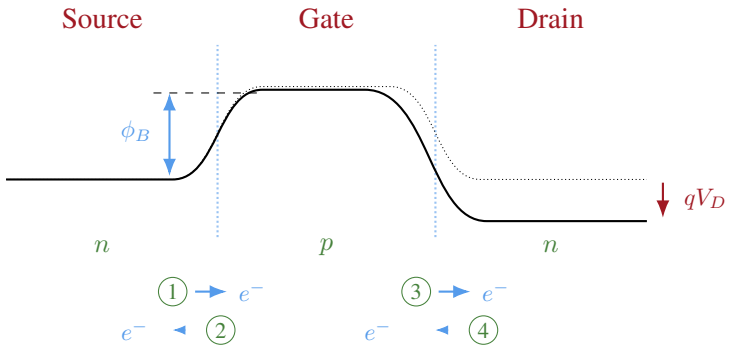


Figure 12: MOSFET subthreshold, $V_S = 0\ V, V_D > 0\ V$

Notice that if we increase the drain voltage further, then the electron injection from drain to bulk would quickly approach zero.

At that point, even though we increase the drain voltage further, the current does not really change. As the current is only now given by the barrier height at the source.

The barrier height at the source is the built in voltage of the junction, and as we've seen before, that voltage depends on doping concentration. If we increase the hole concentration in bulk, then we increase the barrier height, and it's less probable that the electrons have enough energy to be injected from source to bulk.

If we only need to consider the electrons and holes at source for the subthreshold current (assuming the drain voltage is high enough), then we should expect the equation look very similar to a diode, and indeed it does.

The drain current, which is mostly a diffusion current, is given by

$$I_D = I_{D0}\frac{W}{L}e^{q(V_{GS}-V_{tn})/nkT}$$

where

$$n = (C_{ox} + C_{j0})/C_{ox}$$

$$I_{D0} = (n-1)\mu_n C_{ox}\left(\frac{kT}{q}\right)^2$$

This is not exactly the same as the diode equation, but we can see that it looks similar. Most of the quantum mechanics is baked into the V_{tn}

The transconductance (dI_D/dV_{GS}) in weak inversion is then

$$g_m = \frac{I_D}{nV_T}$$

A big difference from the diode equation is the fact that the gate-source voltage seems to determine the current, and not the voltage across the pn junction.

8.5 Transistors in strong inversion

Consider the band diagram in Figure 13, in the figure we're looking at a cross section of the transistor. From left we're in the gate, then we have the oxide, and then the bulk of the transistor.

We don't see the drain and source, as the source would be towards you, and the drain would be into the picture.

The cartoon is not a real transistor. I don't think there is necessarily a combination of semiconductor and metal where we end up with the same Fermi level (E_F) without some bending of the conduction band and valence band, but for illustration, let's assume that's the case.

We can see the Fermi level in the semiconductor is shifted towards the valence band, and thus we have a P-type semiconductor.

The gate is metallic, so it does not have a bandgap, and we assume that the Fermi level is at the conduction band edge.

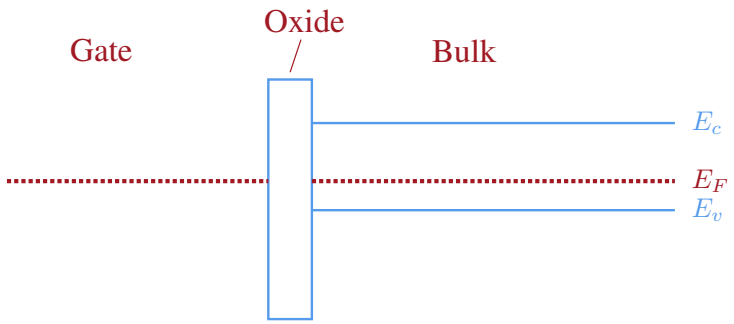


Figure 13: Band diagram of a fictive MOSFET.

Assume we increase the gate-source voltage. In a band diagram that corresponds to shifting the energy down.

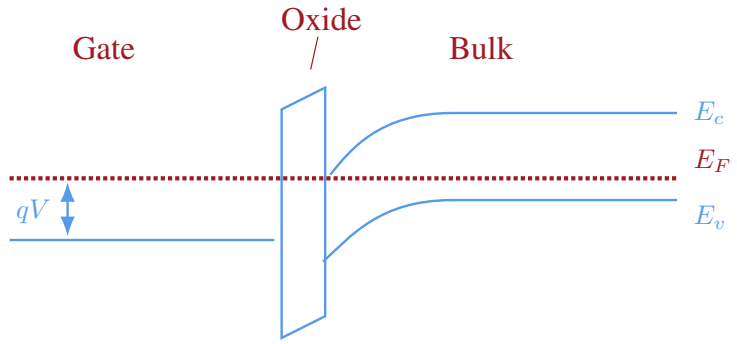


Figure 14: Band diagram with gate-source voltage applied

Moving the gate down has the effect of bending the bands in the semiconductor. We'll lose some voltage across the oxide, but not necessarily that much.

The bending of the valence band will decrease the hole concentration close to the silicon surface, and the semiconductor will be depleted of mobile charge carriers.

The valence band bending will also reduce the barrier height in Figure 12, which increases the number of carriers that can be injected at source/bulk interface, so the subthreshold current will start to increase.

At some point, the band bending of the conduction band will become so large that the electron concentration underneath the gate will increase significantly. The gate-source voltage where the electron concentration equals the bulk hole concentration far away from the silicon surface is called the "threshold voltage".

As you continue to increase the gate-source voltage there is a limit to how much the electron concentration increases. When the band bending of the conduction band passes the Fermi level, then over 50 percent of the available states in the conduction band are filled with electrons.

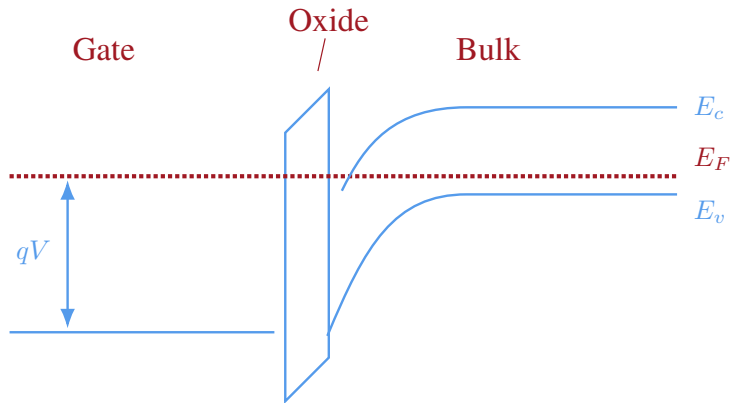


Figure 15: Band diagram with high gate-source voltage applied

The conditions to be in strong inversion is that the gate/source voltage is above some magic values (threshold voltage), and then some.

The quantum state of the electron is fully determined by its spin, momentum and position in space. How those parameters evolve with time is determined by the Schrodinger equation. In the general form

$$i\hbar\frac{d}{dt}\Psi(r,t) = \widehat{H}\Psi(r,t)$$

The Hamiltonian (H) is an "energy matrix" operator and may contain terms both for the momentum and Coulomb force (electric field) experienced by the system.

But what does the Schrodinger equation tell us? Well, the equation above does not tell me much, it can't be "solved", or rather, it does not have a single solution. It's more a framework for how the wave function, and the Hamiltonian, describes the quantum states of a system, and the probability amplitudes of transition between states.

The Schrodinger equation describes the time evolution of the bound electrons shared between the Silicon atoms, and the fact that applying an electric field to silicon can free electrons from covalent bonds.

As the gate-source voltage increases the wave function that fits in the Schrodinger equation predicts that the free electrons will form a 2d sheet underneath the gate. The thickness of the sheet is only a few nano meters.

In Figure 2 of the paper

Carrier transport near the Si/SiO2 interface of a MOSFET

you can see how the free electron density is located underneath the gate.

Figure 16 draws the same story from the equations. The gate field bends the conduction band into a narrow well against the oxide, and a well that narrow does what wells do in quantum mechanics: it quantizes the motion across it. The electrons can only occupy discrete subbands in the depth direction, and at room temperature almost all of them sit in the lowest one, E_0. What the wave function of that subband says is drawn below the band diagram: the probability of finding an electron is zero right at the interface - the oxide barrier forbids it - rises to a peak a nanometre or two into the silicon, and has died away by about five.

That distribution is the "2d sheet". The electrons are free to move along the channel, which is the current we design with, and pinned in depth, which is why the sheet has a thickness at all. It also explains a number you meet later: the inversion charge does not sit exactly at the surface, so the effective oxide thickness is a little larger than the physical one.

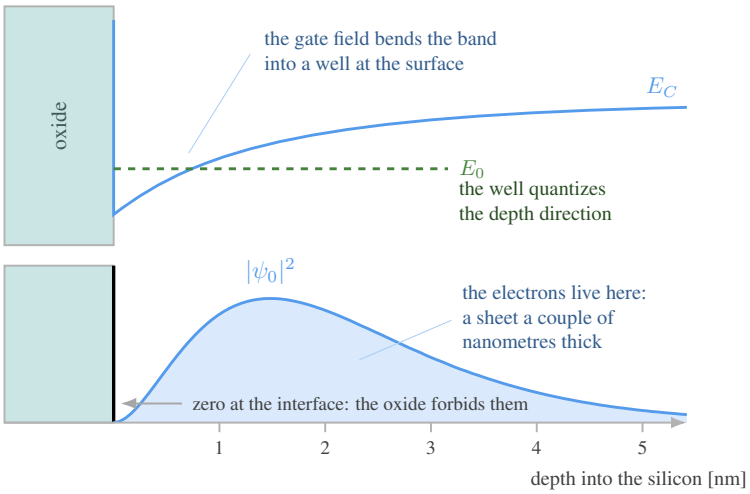


Figure 16: The inversion layer in depth. Above: the gate field bends the conduction band into a well at the oxide interface, and the well quantizes motion in the depth direction into subbands. Below: the probability density of the lowest subband - zero at the interface, peaking a nanometre or two in, gone by about five. That is the "2d sheet"

I would really recommend that you have a look at Mark Lundstrom's lecture series on Essentials of MOSFETs. It's the most complete description of electrons in MOSFET's I've seen

8.6 Introduction to behavior

Let's assume we know nothing about how transistors work, but we do know how to simulate them in ngspice.

We could sit down, and try and figure out how the transistors work.

You can find the testbenches at Testbenches at dicex/sim/spice/N-CHIO

8.6.1 Drain Source Current

Let's see what happens to the drain to source current when we change the voltages. We would expect the drain to source current to change as a function of the drain to source, V_{DS}, and gate to source V_{GS} voltages. Or mathematically

$$I_{DS} = f(V_{GS}, V_{DS}, ...)$$

or symbolically

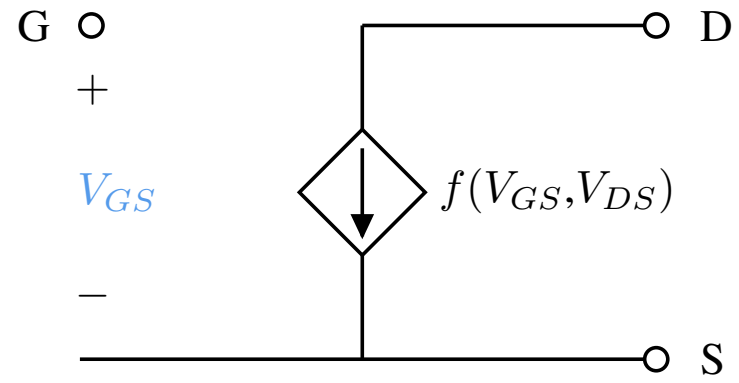


Figure 17: Large signal model

The drain current is a voltage controlled current source $f(V_{GS}, V_{DS})$.

The symbolic model above is what we call a "Large Signal Model". We could expand the function above to

$$I_{DS} = f(V_{GS}, V_{DS}) = G_m(V_{GS}, V_{DS}, I_{DS})V_{GS} + G_{ds}(V_{GS}, V_{DS}, I_{DS})V_{DS}$$

, where the G_m is a trans-conductance (the current depends on a voltage somewhere else), and G_{ds} is a conductance (current depends on the voltage across the conductance).

Even now we can see that the model above is complicated. The transconductance and conductance of the transistor is a function of the other voltages, and the output current. It's a non-linear system!

If the transistor was linear, then we would expect that the current increased proportionally to gate/source voltage, but how does the current look when we change the gate source voltage?

8.6.2 Gate-source voltage

Below are the conditions I've used in the testbench. Notice there is a V_B that is the $p-$ substrate, or bulk, of the transistor. When we draw symbols of a transistor we don't always include the bulk node, because that's most of the time connected to ground for NMOS.

But sometimes, we connect the bulk to another voltage, so the bulk terminal will be in our schematics.

Param	Voltage
VGS	0 to 1.8
VDS	1.0
VS	0
VB	0

In the plot below we can see the sweep of the gate voltage.

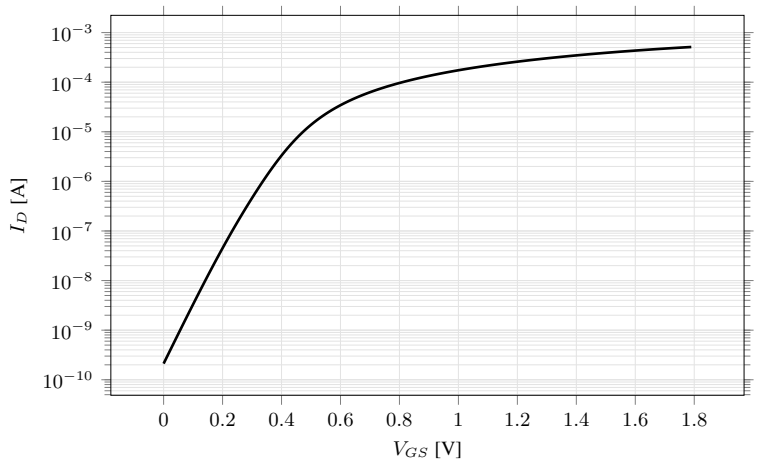


Figure 18: Simulated I_{DS} versus V_{GS} at $V_{DS} = 1\,V$

Notice the log y-axis. In weak inversion the current is exponential in V_{GS} - a straight line on a log axis - and above the threshold voltage the curve bends over into the square-law, and eventually velocity-saturated, behavior. The current changes by many orders of magnitude, which is why the same transistor can be both a decent switch and an amplifier.

8.6.3 Inversion level

The curve above spans six orders of magnitude, so "the" operating region of a MOSFET does not exist - there are three, and they are named by how inverted the channel is.

Define $V_{eff} \equiv V_{GS} - V_{tn}$, where V_{tn} is the "threshold voltage"

Veff	Inversion level
less than 0	weak inversion or subthreshold
0	moderate inversion
more than 100 mV	strong inversion

The single most useful voltage to think in is not V_{GS}, but the effective voltage $V_{eff} = V_{GS} - V_{tn}$: how far above (or below) the threshold voltage we are. It tells us which physics dominates. Below

threshold the current is diffusion over a barrier and exponential in V_{eff}. Well above threshold we have a proper inversion layer and drift current. And in between, in moderate inversion, both mechanisms matter at the same time.

Weak inversion

The drain current is low, but not zero, when

$$V_{eff} << 0$$

$$I_{DS} \approx I_{D0} \frac{W}{L} e^{V_{eff}/nV_T} \text{ if } V_{DS} > 3V_T$$

$$n \approx 1.5$$

This is the same equation we derived from the barrier picture earlier, now with the condition $V_{DS} > 3V_T$ made explicit: once the drain is a few thermal voltages below the barrier, injection from the drain side has died out and the current stops caring about V_{DS}. The slope factor n is the capacitive division between the oxide capacitance and the depletion capacitance - the gate does not get to move the surface potential one-to-one - and lands around 1.5 in this process.

Moderate inversion

Very useful region in real designs. Hard for hand-calculation. Trust the model.

I'm serious about "very useful": a large fraction of well-designed analog transistors end up biased around moderate inversion, because it buys most of the g_m/I_D of weak inversion at a fraction of the area and parasitics. And I'm equally serious about "trust the model": neither the exponential nor the square-law expression is right here, so hand calculation can only bracket the answer. Set up the bias with the simulator, and use the hand expressions to sanity-check the trend.

Strong inversion

$$I_{DS} = \mu_n C_{ox} \frac{W}{L} \begin{cases} V_{eff} V_{DS} & \text{if } V_{DS} << V_{eff} \\ V_{eff} V_{DS} - V_{DS}^2/2 & \text{if } V_{DS} < V_{eff} \\ \frac{1}{2} V_{eff}^2 & \text{if } V_{DS} > V_{eff} \end{cases}$$

Three cases, one story. For tiny V_{DS} the channel is a uniform resistor and the current is linear in both voltages. As V_{DS} grows the drain end of the channel gets less gate-to-channel voltage, so the channel thins there and the $-V_{DS}^2/2$ term bends the curve over. And at $V_{DS} = V_{eff}$ the drain end pinches off entirely: past that point the current is set by the channel alone and stops (mostly) caring about the drain.

To see where the equations come from it helps to watch the carriers. The cross-sections below walk the gate voltage up from negative to positive before we do the same with the drain.

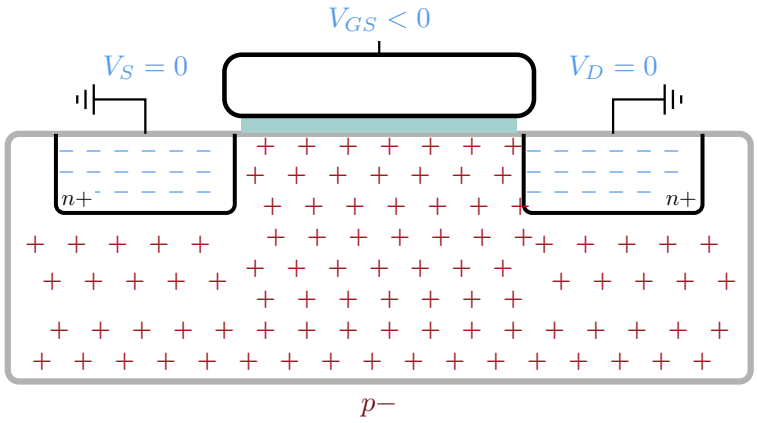


Figure 19: Accumulation, $V_{GS} < 0$

What we're seeing here are the free charges - the mobile carriers, not the fixed dopant ions. Electrons (blue) are free to move around in the $n+$ source and drain, holes (red) in the $p-$ bulk. Where the mobile carriers are present in their equilibrium numbers the material is quasi-neutral: behind each free carrier sits an ionized donor or acceptor locked in the lattice, and the two cancel.

That is not true everywhere, and the exceptions are where the interesting physics lives. Sweep the mobile carriers out of a region - at the source/bulk and drain/bulk junctions, and under the gate once it starts to deplete - and the ionized dopants are left behind uncompensated. That leftover space charge is not neutral, and by Gauss it must produce a field: this is the depletion region, and its field is what holds the junction in equilibrium and what sweeps carriers across it. Keep the distinction in mind while reading the next few figures: they draw the mobile carriers, so a region drawn empty is not a region with no charge - it is a region whose charge is the fixed dopant ions, with a field across it.

With a negative gate-source voltage the holes are attracted to the surface and accumulate underneath the gate.

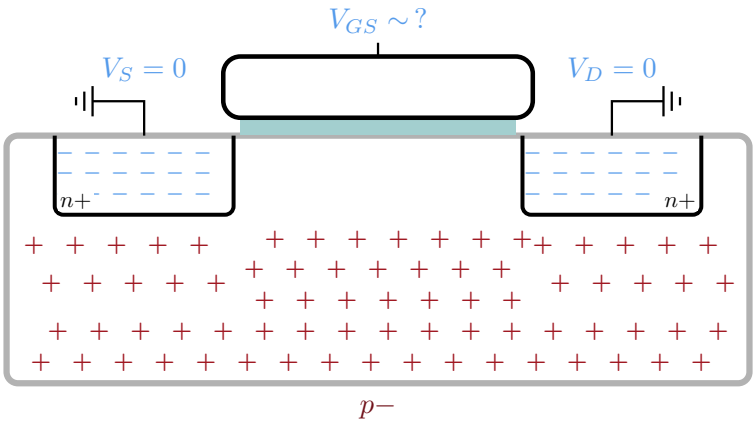


Figure 20: Depletion

Raising the gate voltage pushes the holes away from the surface. The region underneath the gate is depleted of mobile carriers - only the fixed, negatively charged acceptor ions remain.

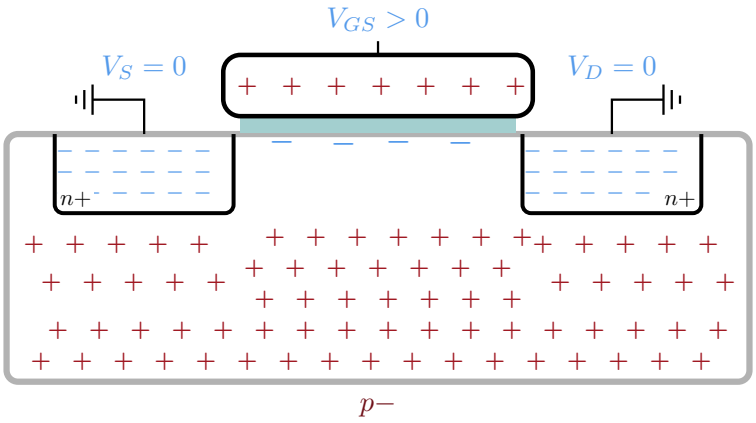


Figure 21: Weak inversion

Positive charge on the gate mirrors negative charge in the silicon. The depletion region grows, the barrier between source and bulk shrinks, and the first few electrons make it into the channel region - the exponential subthreshold current from earlier.

And this is where the threshold voltage gets its definition: V_{tn} is the gate-source voltage where the electron concentration at the surface equals the hole concentration in the bulk, $p_p = n_{ch}$. Nothing physically dramatic happens at exactly that voltage - the electron concentration is exponential in surface potential either side of it - but it's a well-defined line in the sand, and every equation in this chapter leans on it.

Now hold the gate at a fixed 0.5 V - a little above threshold - and sweep the drain instead.

8.6.4 Drain source voltage

The table shows the bias: gate fixed a little above threshold, drain swept from 0 V to 1.8 V. Watch the current in Figure 22 as the drain rises - the curve has two personalities, and the boundary between them is V_{eff}.

Param	Voltage [V]
V*GS*	0.5
V*DS*	0 to 1.8
V*S*	0
V*B*	0

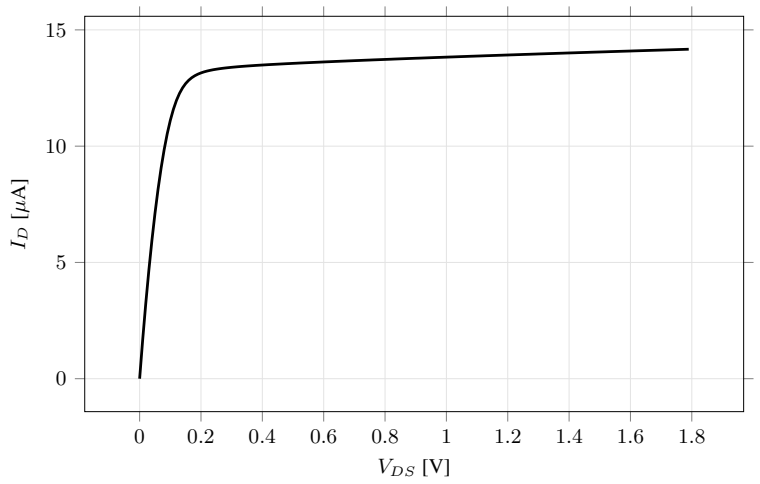


Figure 22: Simulated I_{DS} versus V_{DS} at $V_{GS} = 0.5$ V

For small V_{DS} the transistor behaves like a resistor - the current is proportional to the voltage. Beyond $V_{DS} \approx V_{eff}$ the current flattens: the transistor is in saturation, and only the weak slope from channel length modulation (and DIBL) remains.

8.6.5 Strong inversion

The measured curve is captured by one equation with three cases, split on how V_{DS} compares to V_{eff}. Read each case together with Figures 22 to 24, which show what the inversion layer is doing in that region.

$$I_{DS} = \mu_n C_{ox} \frac{W}{L} \begin{cases} V_{eff} V_{DS} & \text{if } V_{DS} << V_{eff} \\ V_{eff} V_{DS} - V_{DS}^2/2 & \text{if } V_{DS} < V_{eff} \\ \frac{1}{2} V_{eff}^2 & \text{if } V_{DS} > V_{eff} \end{cases}$$

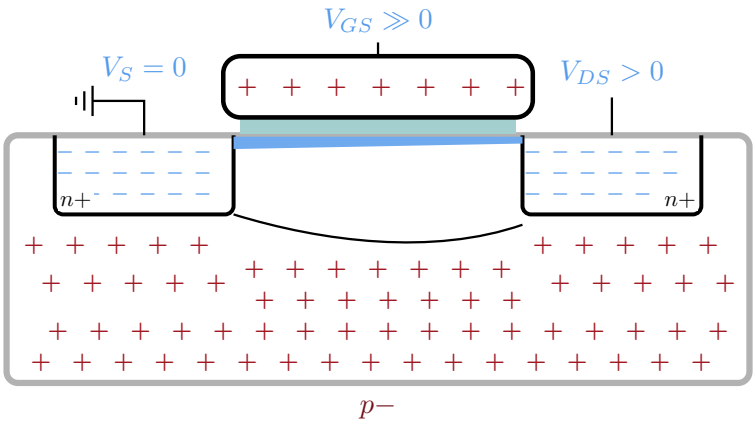


Figure 23: Triode, $V_{DS} \ll V_{eff}$

The inversion layer reaches all the way from source to drain, and the transistor behaves like a gate-voltage controlled resistor.

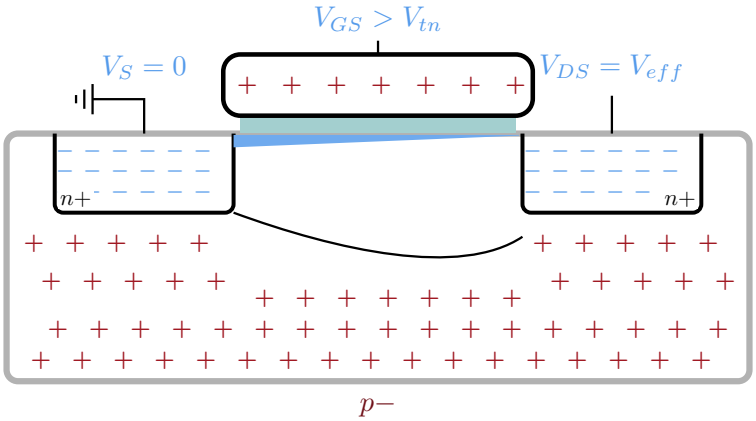


Figure 24: Pinch-off, $V_{DS} = V_{eff}$

The local gate-to-channel voltage at the drain end is down to V_{tn}, so the inversion layer just barely disappears at the drain.

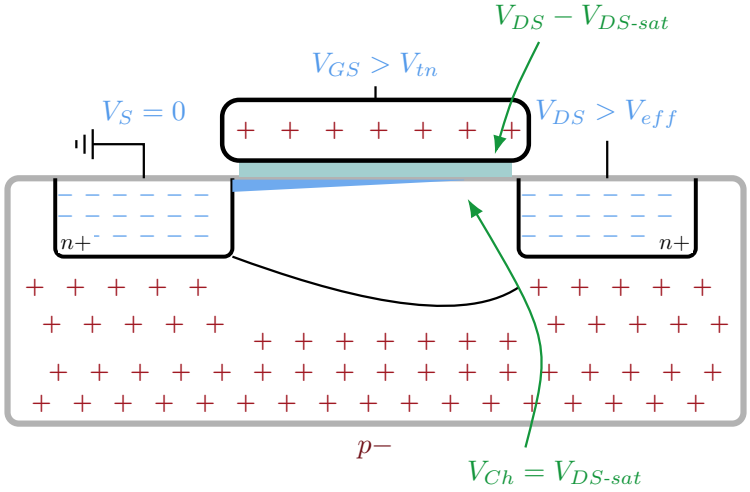


Figure 25: Saturation, $V_{DS} > V_{eff}$

The pinch-off point, where the channel voltage equals $V_{DS,sat}$, moves slightly towards the source as V_{DS} increases.

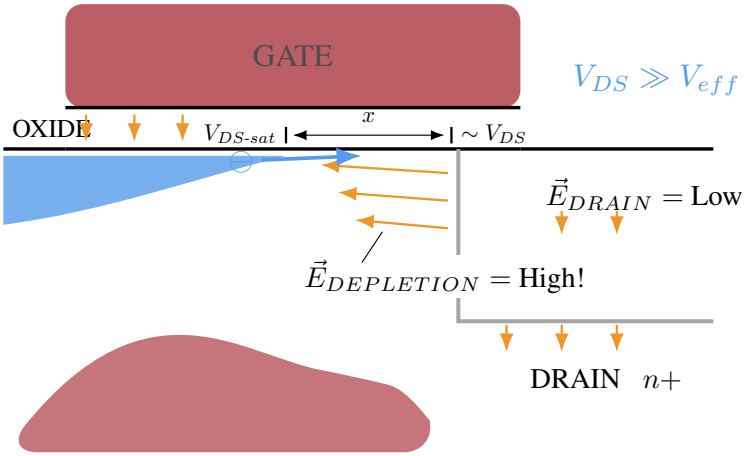


Figure 26: Close-up of the drain end in saturation

Between the end of the inversion layer and the drain there is no channel, but there is a strong lateral field across the depleted gap. Electrons that reach the pinch-off point are swept across to the drain, which is why the current does not stop at pinch-off. The length of the gap grows slightly with V_{DS}, which shortens the effective channel - channel length modulation.

8.6.6 Low frequency model

The curves so far are large signal: the actual currents and voltages. An amplifier works on small wiggles around a bias point, and for those we linearize. Two derivatives are all we keep at low frequency: how much drain current a gate wiggle gives, g_m, and how much the drain voltage steals back, g_{ds}. Figure 26 is those two derivatives drawn as a circuit.

$$g_m = \frac{\partial I_{DS}}{\partial V_{GS}}$$

$$g_{ds} = \frac{1}{r_{ds}} = \frac{\partial I_{DS}}{\partial V_{DS}}$$

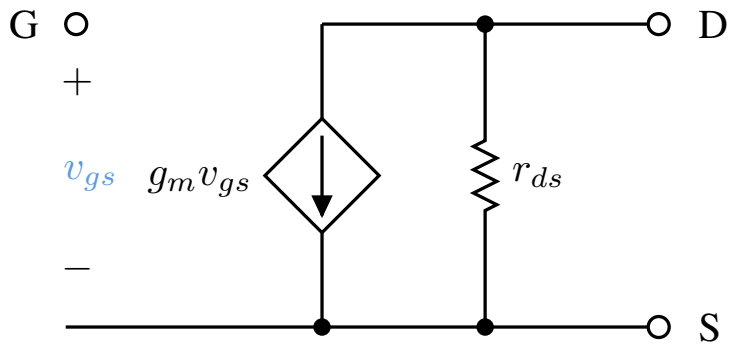


Figure 27: Low frequency small signal model

For small perturbations around an operating point the transistor is just two elements: a transconductance $g_m v_{gs}$ and an output resistance r_{ds}.

Now put the square law into the two derivatives, starting with g_m. A little algebra gives the same transconductance in three different currencies:

8.6.7 Transconductance

Define $\ell = \mu_n C_{ox} \frac{W}{L}$ and $V_{eff} = V_{GS} - V_{tn}$

$I_D = \frac{1}{2}\ell(V_{eff})^2$ and $V_{eff} = \sqrt{\frac{2I_D}{\ell}}$ and $\ell = \frac{2I_D}{V_{eff}^2}$

$$g_m = \frac{\partial I_{DS}}{\partial V_{GS}} = \ell V_{eff} = \sqrt{2\ell I_D}$$

$$g_m = \ell V_{eff} = 2\frac{I_D}{V_{eff}^2} V_{eff} = \frac{2I_D}{V_{eff}}$$

The same g_m written three ways, and each is useful for a different question. $g_m = \ell V_{eff}$ answers "what does another 100 mV of gate drive buy me". $g_m = \sqrt{2\ell I_D}$ answers "what does another micro amp buy me" - only square-root much, which is why burning current for bandwidth gets expensive. And $g_m = 2I_D/V_{eff}$ is the designer's favorite, because it needs no process constants at all: pick a current and an effective voltage, and the transconductance follows.

For the output conductance we need the piece the ideal square law leaves out: let the current grow linearly with V_{DS} through a channel length modulation term λ, and differentiate.

Define $\ell = \mu_n C_{ox} \frac{W}{L}$ and $V_{eff} = V_{GS} - V_{tn}$

$$I_D = \frac{1}{2}\ell V_{eff}^2 \left[1 + \lambda(V_{DS} - V_{eff})\right]$$

$$\frac{1}{r_{ds}} = g_{ds} = \frac{\partial I_D}{\partial V_{DS}} = \lambda\frac{1}{2}\ell V_{eff}^2$$

Assume channel length modulation is not there, then

$I_D = \frac{1}{2}\ell V_{eff}^2$ which means $\frac{1}{r_{ds}} = g_{ds} \approx \lambda I_D$

λ is the channel length modulation parameter: the pinch-off point in Figure 26 creeps towards the source as V_{DS} grows, the effective channel shortens, and the current rises a little. The practical consequences: the output resistance is inversely proportional to the current you run, and since λ shrinks with channel length, a longer transistor is the cheapest way to buy output resistance.

The two derivatives combine into the most important figure of merit of a single transistor: the largest voltage gain it can possibly give you.

8.6.8 Intrinsic gain

Define intrinsic gain as

$$A = \left|\frac{v_{out}}{v_{in}}\right| = g_m r_{ds} = \frac{g_m}{g_{ds}}$$

$$A = \frac{2I_D}{V_{eff}} \times \frac{1}{\lambda I_D} = \frac{2}{\lambda V_{eff}}$$

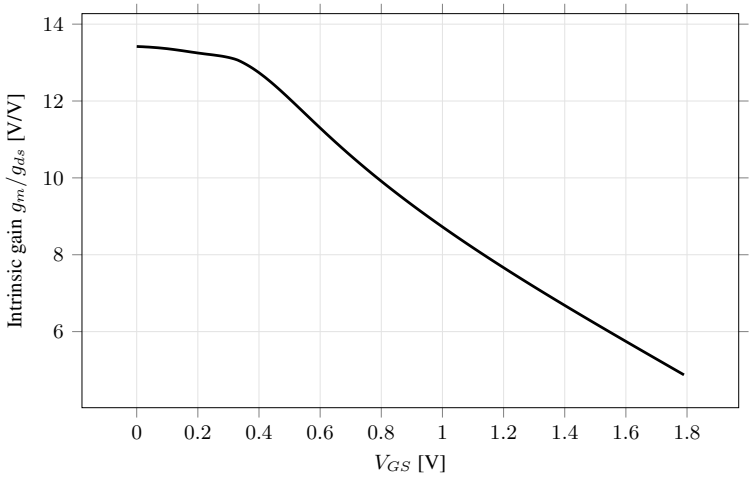


Figure 28: Simulated intrinsic gain versus gate-source voltage (the x-axis, vgaini, is $V_{GS} = V_{eff} + V_{tn}$)

The intrinsic gain falls as V_{eff} increases, as the $2/(\lambda V_{eff})$ expression predicts. If you need gain, don't burn all your headroom on effective voltage.

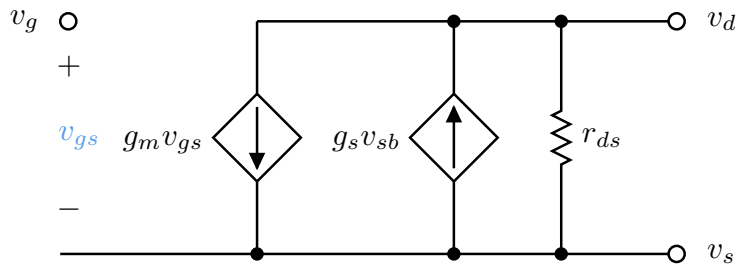


Figure 29: Small signal model with the bulk transconductance

The bulk is a back-gate: if source and bulk move relative to each other, the threshold voltage - and hence the current - changes, which the $g_s v_{sb}$ source models.

8.6.9 Body effect

How strongly the back-gate acts has a name: the body effect coefficient γ, and it comes straight from the capacitive divider between the gate oxide and the depletion region under the channel.

$$V_{tn} = V_{t0} + \gamma \left(\sqrt{2\phi_F + V_{SB}} - \sqrt{2\phi_F} \right)$$

$$\gamma = \frac{\sqrt{2qN_A\epsilon_{si}}}{C_{ox}}$$

$$g_s = \frac{\partial I_{DS}}{\partial V_{SB}} \approx (n-1)g_m \approx 0.2 g_m$$

Reverse bias the source-bulk junction and the depletion region under the channel widens. The extra depletion charge must be imaged on the gate, so the threshold voltage rises - that is the square root above. The small signal version is the g_s source in Figure 29, roughly a fifth of g_m. It is a parasitic in a source follower, and a free extra input if you drive the bulk on purpose - a trick the OTA chapter returns to.

8.6.10 High frequency model

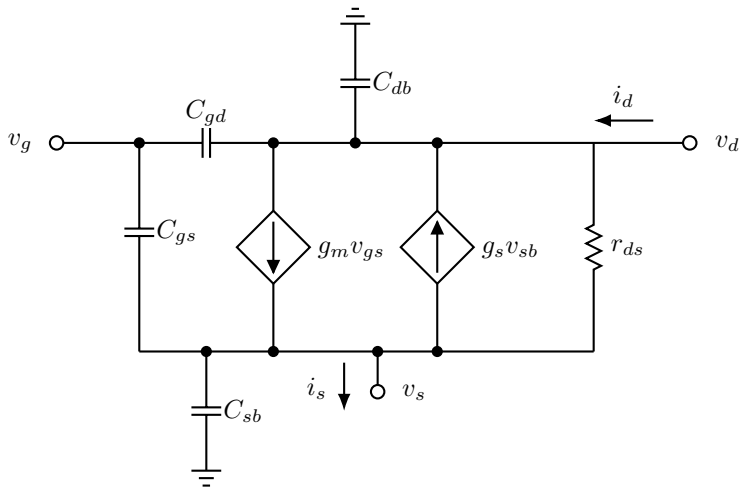


Figure 30: High frequency small signal model

The four capacitances C_{gs}, C_{gd}, C_{sb} and C_{db} set the poles and zeros at high frequency.

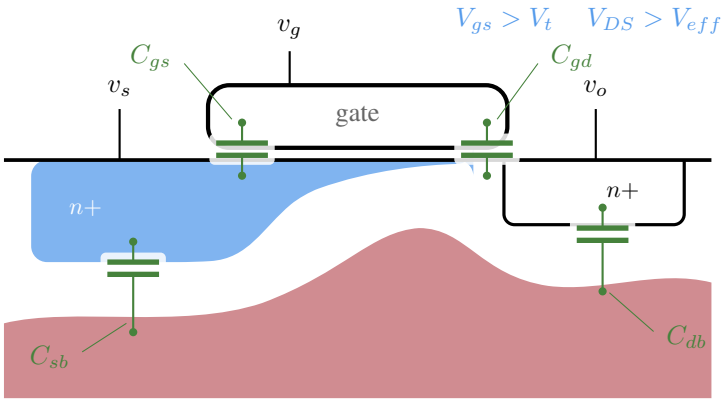


Figure 31: Where the capacitances live in the device

C_{gs} and C_{gd} are oxide (and overlap) capacitances, while C_{sb} and C_{db} are depletion capacitances of the reverse biased source and drain junctions.

The gate capacitances first. How the gate charge splits between source and drain depends on the region of operation:

C_{gs} and C_{gd}

$$C_{gs} = \begin{cases} WLC_{ox} & \text{if } V_{DS} = 0 \\ \frac{2}{3}WLC_{ox} & \text{if } V_{DS} > V_{eff} \end{cases}$$

$$C_{gd} = C_{ox}WL_{ov}$$

In triode the channel is uniform and the whole gate area capacitance WLC_{ox} splits evenly between source and drain. In saturation the drain end is pinched off - the channel charge lives mostly at the source end - and integrating the charge distribution gives the famous $\frac{2}{3}$. C_{gd} then keeps only the overlap capacitance $C_{ox}WL_{ov}$, where L_{ov} is the small distance the drain diffusion pokes in underneath the gate. Small, but as the Miller section shows, not harmless.

C_{sb} and C_{db}

Both are depletion capacitances

$$C_{sb} = (A_s + A_{ch})C_{js}$$

$$C_{js} = \frac{C_{j0}}{\sqrt{1 + \frac{V_{SB}}{\Phi_0}}}$$

$$\Phi_0 = V_T ln\left(\frac{N_A N_D}{n_i^2}\right)$$

$$C_{db} = A_d C_{jd}$$

$$C_{jd} = \frac{C_{j0}}{\sqrt{1 + \frac{V_{DB}}{\Phi_0}}}$$

The source and drain diffusions sit in reverse biased junctions to the bulk, and a reverse biased junction is a capacitor whose plates are the depletion edges. More reverse bias, wider depletion, smaller capacitance - hence the square root. A_s and A_d are the junction areas (the source side includes the channel area A_{ch}), and Φ_0 is the built-in voltage we met in the diode chapter.

8.6.11 Be careful with Cgd (blame Miller)

Of the four capacitances, C_{gd} is the smallest on paper - just the overlap - and the most dangerous in practice. The reason is where it sits: between the input and the output of an amplifying stage. Look at the left of Figure 31: a common source stage with a current source load, gain $A = -g_m r_d$ from gate to drain, and C_{gd} strapped across exactly that gain.

Why that matters is Miller's theorem, sketched in the dashed frame on the right of the figure. Take any admittance Y connected around

an inverting amplifier $-A$. Wiggle the input by v: the output moves by $-Av$, so the voltage across Y is $(1 + A)v$, and the input has to supply $(1 + A)$ times the current it would if Y simply went to ground. The feedback element can therefore be replaced by two grounded ones,

If $Y(s) = 1/sC$ then $Y_1(s) = 1/sC_{in}$ and $Y_2(s) = 1/sC_{out}$ where $C_{in} = (1 + A)C$, $C_{out} = (1 + \frac{1}{A})C$

$$C_{in} \approx C_{gd}\, g_m r_{ds}$$

C_{gd} can appear to be 10 to 100 times larger!

if gain from input to output is large

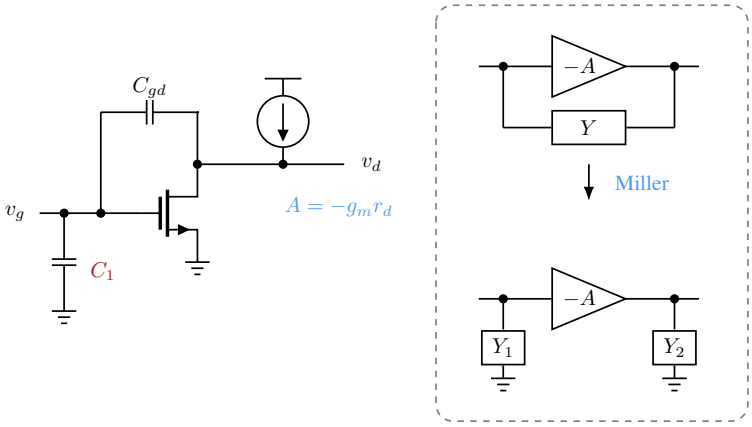


Figure 32: Miller's theorem applied to C_{gd}

For the capacitor this means the input sees $C_{in} = (1 + A)C$, drawn as C_1 at the gate in the figure, while the output sees a nearly unchanged $C_{out} = (1 + 1/A)C$. With $C = C_{gd}$ and $A = g_m r_{ds}$, the gate is loaded by C_{gd} multiplied by the stage gain - 10 to 100 times the overlap capacitance you read from the layout. This is why the input pole of a high-gain stage is so often set by its smallest capacitor.

8.6.12 Transit frequency

The high frequency model rolls up into a single speed metric: the frequency where the current gain of the transistor falls to one. Drive the gate with a current and ask when the gate capacitance eats all of it:

$$f_T = \frac{g_m}{2\pi(C_{gs} + C_{gd})}$$

In strong inversion, with $C_{gs} \approx \frac{2}{3}WLC_{ox}$:

$$f_T \approx \frac{3\mu_n V_{eff}}{4\pi L^2} \propto \frac{V_{eff}}{L^2}$$

f_T is why we scale: halve the length and the transistor is four times faster, until velocity saturation takes one of the two factors back. In a nanoscale process f_T reaches hundreds of gigahertz - but look at the trade: the V_{eff} that buys speed is the same V_{eff} that sells intrinsic gain in Figure 28. Fast and high gain is not on the menu, at least not in one transistor.

8.7 Weak inversion

If $V_{eff} < 0$ diffusion currents dominate.

Back to weak inversion, now wearing the model hat rather than the physics hat. Everything is the barrier picture from the first half of this chapter, compressed into three constants: V_T sets the exponential slope, n the capacitive division, and I_{D0} collects the rest.

$$I_D = I_{D0} \frac{W}{L} e^{V_{eff}/nV_T}$$

, where

$V_T = kT/q$, $n = (C_{ox} + C_{j0})/C_{ox}$

$$I_{D0} = (n-1)\mu_n C_{ox} V_T^2$$

$$g_m = \frac{I_D}{nV_T}$$

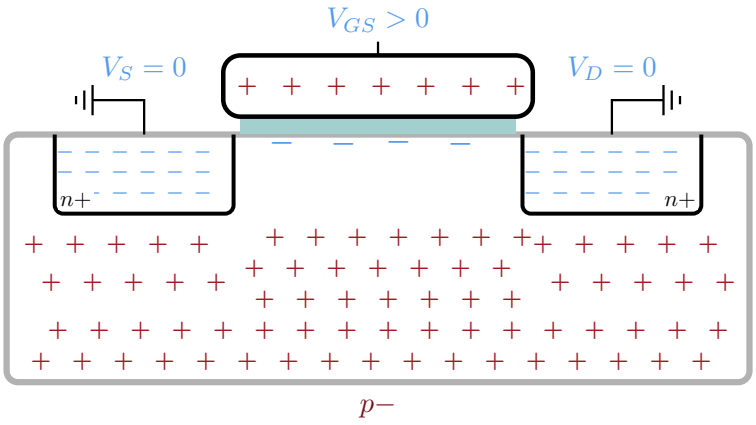


Figure 33: Weak inversion again, beside the equations it produces: the gate has depleted the surface and the first electrons are arriving, which is the exponential regime the three constants describe

Differentiate an exponential and you get the exponential back, divided by nV_T: in weak inversion the transconductance is proportional to the current, full stop. No W/L, no mobility, no C_{ox} - just current and temperature. That is as good as g_m per current gets in a MOSFET, and it is the reason the next slide's ratio flattens out on the left.

The two regions can be compared on one axis: how much transconductance a microampere buys.

Bang for the buck

Subthreshold:

$$\frac{g_m}{I_D} = \frac{1}{nV_T} \approx 25.6 \text{ [S/A] @ 300 K}$$

Strong inversion:

$$\frac{g_m}{I_D} = \frac{2}{V_{eff}}$$

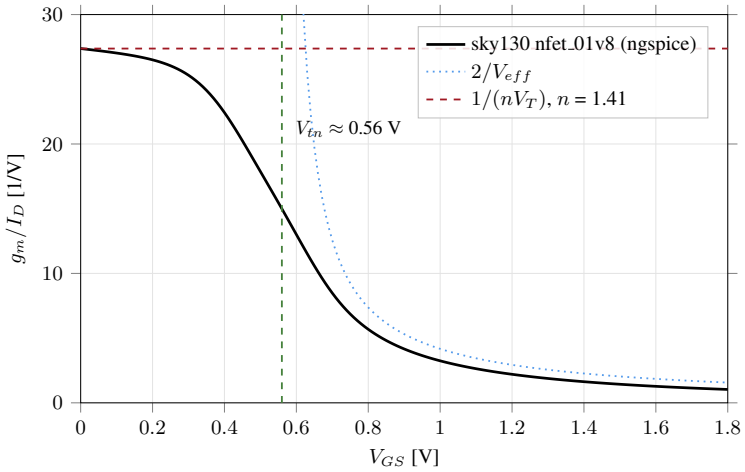


Figure 34: Simulated g_m/I_D of a sky130 nfet_01v8 (gmid.py)

The transconductance per unit current is the "bang for the buck" of a transistor, and the figure shows both hand-calculation limits on top of the simulated curve. In weak inversion the measured curve flattens at $1/(nV_T)$ - the subthreshold slope of this device gives $n \approx 1.4$, about 27 S/A. In strong inversion it follows $2/V_{eff}$. In between, in moderate inversion, neither hand expression is right - the $2/V_{eff}$ asymptote overestimates by a wide margin - which is exactly why the advice earlier was to trust the model there. If you want the most g_m for your current, bias weak; if you want speed

and matching, bias strong; most real analog ends up somewhere on the knee.

8.8 Velocity saturation

Electron speed limit in silicon

$$v \approx 10^7 cm/s$$

$$v = \mu_n E = \mu_n \frac{dV}{dx}$$

$$\mu_n \approx 100 \text{ to } 600\ cm^2/Vs$$

in nanoscale CMOS

The mobility model says velocity is proportional to field. But carriers in silicon scatter off the lattice, and above roughly 10^7 cm/s more field just means more scattering, not more speed. Shrink L at constant voltage and the lateral field V/L grows without bound - at 1 V the mobility model crosses the speed limit just below half a micrometer, which is why every modern process lives with velocity saturation.

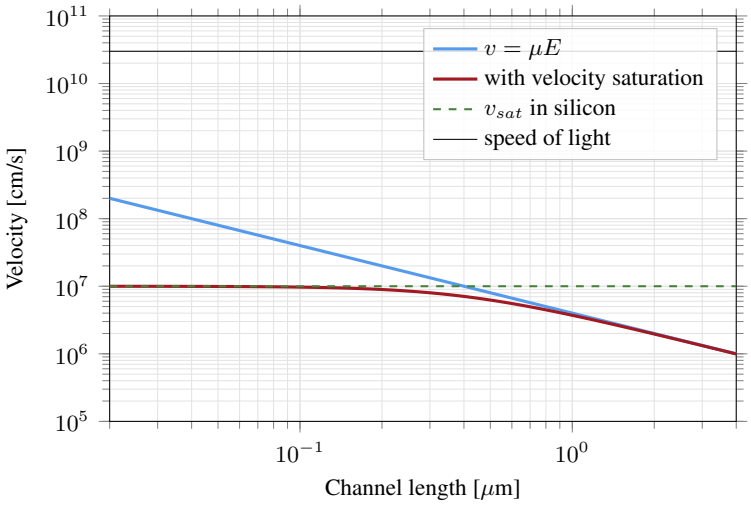


Figure 35: Carrier velocity at 1 V across the channel: the mobility model, what the carriers actually do, and the physical speed limits

Shrink the channel at a fixed voltage and the lateral field, and thus the mobility-model velocity (blue), grows without bound - it crosses the silicon speed limit long before it crosses anything relativistic. Real carriers cannot do that: the red curve saturates

at v_{sat}, so in short channels the current becomes closer to linear, rather than quadratic, in V_{eff}.

Where does the square law actually come from? Charge, times width, times velocity - integrated along the channel:

8.8.1 Square law model

$$Q(x) = C_{ox}\left[V_{eff} - V(x)\right]$$

$$v = \mu_n E = \mu_n \frac{dV}{dx}$$

$$\ell = \mu_n C_{ox} \frac{W}{L}$$

$$I_D = WQ(x)v = \ell L\left[V_{eff} - V(x)\right]\frac{dV}{dx}$$

$$I_D dx = \ell L\left[V_{eff} - V(x)\right]dV$$

$$I_D \int_0^L dx = \ell L \int_0^{V_{DS}} \left[V_{eff} - V(x)\right]dV$$

$$I_D\,[x]_0^L = \ell L\left[V_{eff}V - \frac{1}{2}V^2\right]_0^{V_{DS}}$$

$$I_D L = \ell L\left[V_{eff}V_{DS} - \frac{1}{2}V_{DS}^2\right]$$

$$@V_{DS} = V_{eff} \Rightarrow I_D = \frac{1}{2}\ell V_{eff}^2$$

This is the derivation behind the square law, and it is worth reading once in your life. The local channel charge is $Q(x) = C_{ox}[V_{eff} - V(x)]$ - less charge where the channel voltage has climbed - the current is charge times width times velocity, and since the same I_D must flow through every slice of the channel, integrating from source to drain turns the local statement into $I_D = \ell\,[V_{eff}V_{DS} - V_{DS}^2/2]$. Evaluate at $V_{DS} = V_{eff}$ and the familiar $\frac{1}{2}\ell V_{eff}^2$ falls out. Every assumption in this chain - constant mobility, gradual channel, charge proportional to local voltage - is something a short-channel transistor violates.

8.8.2 Mobility Degradation

The square law assumes the mobility is a constant. It is not - two mechanisms drag it down as the gate drive grows, and the model needs a correction factor.

Multiple effects degrade mobility

- Velocity saturation
- Vertical fields reduce channel depth => more charge-carrier scattering

$$\ell = \mu_n C_{ox} \frac{W}{L}$$

$$\mu_{n\_eff} = \frac{\mu_n}{([1 + (\theta V_{eff})^m])^{1/m}}$$

$$I_D = \frac{1}{2} \ell V_{eff}^2 \frac{1}{([1 + (\theta V_{eff})^m])^{1/m}}$$

From square law

$$g_m = \frac{\partial I_D}{\partial V_{GS}} = \ell V_{eff}$$

With mobility degradation

$$g_{m(mob-deg)} = \frac{\ell}{2\theta}$$

Velocity saturation is one of several effects that make the effective mobility fall as we crank V_{eff}: the vertical field also squeezes the carriers against the rough oxide interface where they scatter more. The fitting function above captures the trend, and its punchline is the last equation: push V_{eff} hard enough and g_m stops growing entirely at $\ell/2\theta$. Past that point, extra gate drive costs headroom and buys nothing.

8.8.3 What about holes (PMOS)

Everything so far used the NMOS. The PMOS is the same device upside down: the carriers are holes, and holes are slower.

In PMOS holes are the charge-carrier (electron movement in valence band)

$$\mu_p < \mu_n$$

In intrinsic silicon:

$$\mu_n \leq 1400[cm^2/Vs] = 0.14[m^2/Vs]$$

$$\mu_p \leq 450[cm^2/Vs] = 0.045[m^2/Vs]$$

$$\mu_n \approx 3\mu_p$$

Saturation velocity (same as the electron speed limit above):

$$v_{n\_sat} \approx 1.0 \times 10^5 [m/s]$$

$$v_{p\_sat} \approx 0.8 \times 10^5 [m/s]$$

Don't confuse it with the thermal velocity $\approx 2.3 \times 10^5$ *m/s*

Doping reduces the mobility as well: every ionized donor or acceptor is a charged scattering center, so a heavily doped channel is a slower channel.

Everything in this chapter holds for the PMOS with the signs flipped - and one important asymmetry: holes move by electrons shuffling between bonds in the valence band, and are roughly three times slower than conduction band electrons. That factor shows up everywhere: for the same current and V_{eff} a PMOS is about three times wider, with correspondingly larger capacitances. It is also why the NMOS usually gets the signal path and the PMOS the loads - though in some modern strained processes the gap has narrowed to less than a factor of two.

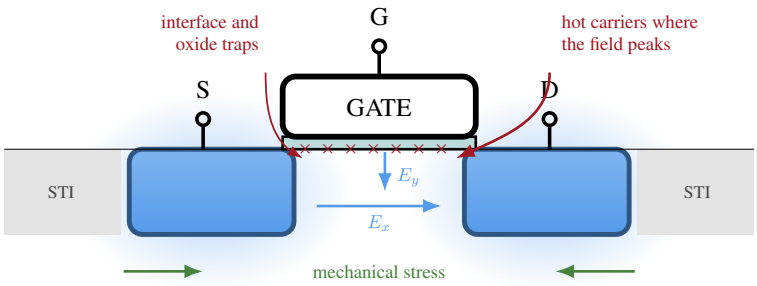
8.9 OTHER

As we make transistors smaller, we find new effects that matter, and that must be modeled.

_ which is an opportunity for engineers to come up with cool names _

The square law is a long-channel story. Below a micrometer or so, and especially below 100 nm, a zoo of second-order effects grows to first order. The point of this section is not that you memorize each one - the foundry's model team already did - but that you recognize the names when they show up in a design review, and know which knob (length, layout, bias) each one responds to. The paper below is a fine map of the zoo.

Analog Circuit Design in Nanoscale CMOS Technologies [1] – ieeexplore.ieee.org/document/5247174



Four families of effect, each with its own corner of the device model, and each taken separately in the sections that follow. Below about 100 nm every one of them is first order.

Figure 36: Four families of short-channel effect on one cross-section - mechanical stress from the isolation trenches, the transverse and lateral fields in the channel, traps at the oxide interface, and hot carriers where the lateral field peaks near the drain

The annotations - stress components, fields, trap densities and proximity effects - are all things that measurably change the current of a modern transistor, and each has its own corner of the device model.

8.9.1 Drain induced barrier lowering (DIBL)

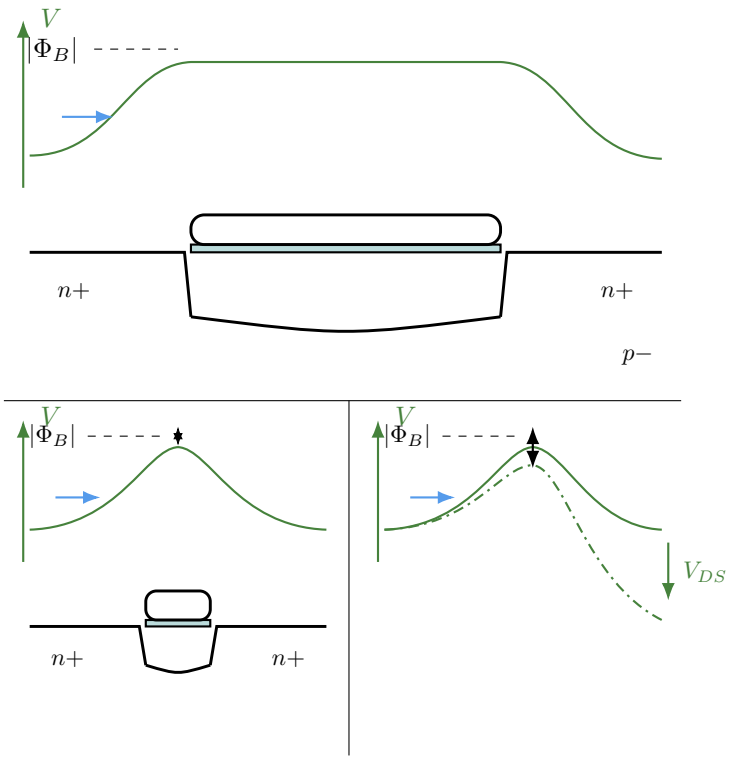


Figure 37: Drain induced barrier lowering

In a long channel (top) the source barrier Φ_B has a wide, flat top and the drain is far away. In a short channel (bottom left) the barrier

is a narrow peak, and pulling the drain potential down (bottom right, dashed) also pulls the top of the barrier down. A lower barrier means more current at the same gate voltage: the threshold voltage effectively drops as V_{DS} increases, which degrades the output resistance.

8.9.2 Well Proximity Effect (WPE)

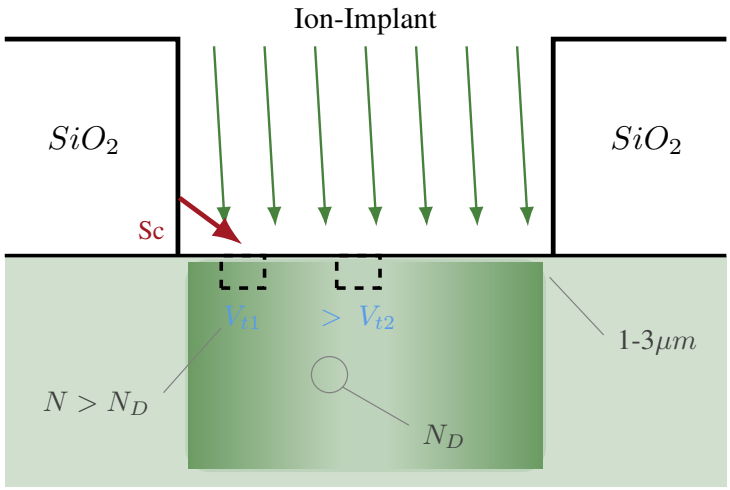


Figure 38: Well proximity effect

During the well implant, ions scatter off the edge of the photoresist and land in the silicon close to the well edge. Transistors within a micrometer or three of the well edge therefore see higher doping, and thus a higher threshold voltage, than identical transistors in the middle of the well.

8.9.3 Stress effects

Silicon is piezoresistive: squeeze it and the mobility changes. The table summarizes which direction of squeeze helps which device, and Figure 37 defines the three directions.

Stress	PMOS	NMOS
Stretch Fz	Good	Good
Compress Fy	OK	Good
Compress Fx	Good	Bad

What can change stress?

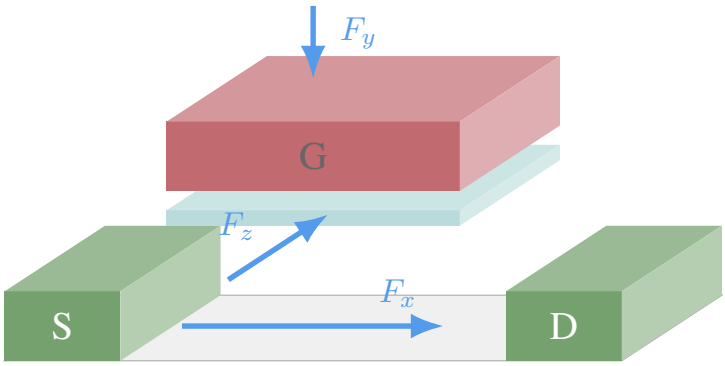


Figure 39: Mechanical stress components on the channel

Stress changes mobility, so anything that changes the stress - shallow trench isolation, nearby devices, metal fill, even the package - changes the current. The table shows the direction dependence: F_y is vertical, F_x along the current, F_z along the width.

8.9.4 Gate current

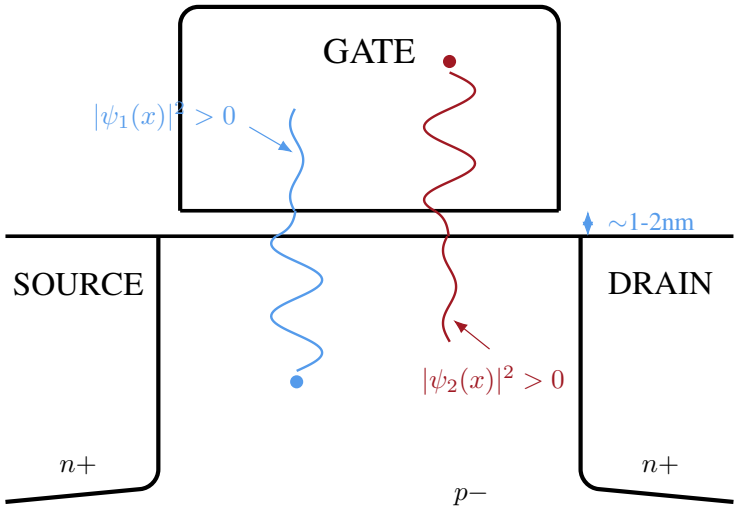


Figure 40: Gate tunneling current

With an oxide only 1-2 nm thick, the electron wave function does not stop at the oxide: $\psi(x)$ is non-zero on the other side, so carriers tunnel between channel and gate. The gate is no longer a perfect insulator, which matters for sample-and-holds and anything with high impedance nodes.

8.9.5 Hot carrier injection

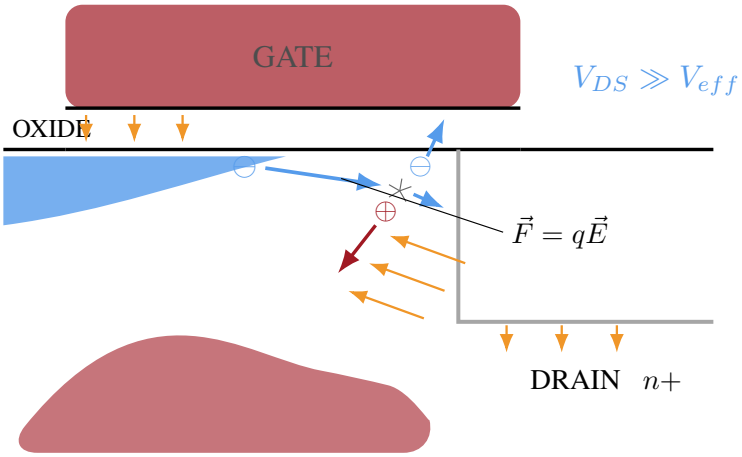


Figure 41: Hot carrier injection

In saturation the field across the pinched-off region near the drain is high. Carriers accelerated by it ($F = qE$) can gain enough energy to create electron-hole pairs by impact ionization, and some are injected into the oxide, where they damage the interface or get trapped and shift the threshold voltage over the product lifetime.

8.9.6 Channel initiated secondary-electron (CHISEL)

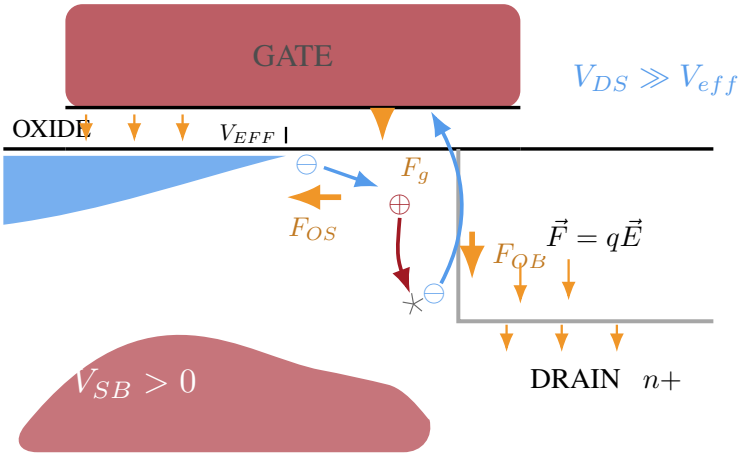


Figure 42: Channel initiated secondary electrons

With a reverse biased bulk ($V_{SB} > 0$), holes generated by impact ionization near the drain are accelerated into the bulk and can generate secondary electrons, which the vertical field can inject into the gate oxide - the same damage mechanism as hot carriers, opened up by the bulk bias.

8.10 Variability

For the rest of the chapter we leave the single ideal transistor behind and ask the question that actually decides whether circuits work: what happens when you make two of them? The vehicle is deliberately humble - a current mirror that is supposed to copy 1 uA - because every mechanism that breaks the copy also breaks amplifiers, converters and references, just with more algebra in the way.

Provide $I_2 = 1\mu A$

Let's use off-chip resistor R, and pick R such that $I_1 = 1\mu A$

Use $\frac{W_1}{L_1} = \frac{W_2}{L_2}$

What makes $I_2 \neq 1\mu A$?

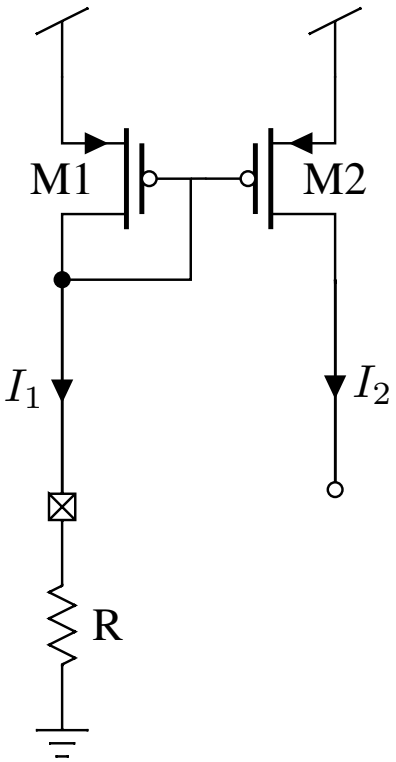


Figure 43: Current mirror with an off-chip reference resistor

The rest of this section asks a deceptively simple question about this circuit: what makes I_2 deviate from the ideal 1 uA?

- Voltage variation
- Systematic variations
- Process variations
- Temperature variation
- Random variations
- Noise

8.10.1 Voltage variation

Start with the most obvious dependency: the supply sits in the loop that sets the reference current.

$$I_1 = \frac{V_{DD} - V_{GS1}}{R}$$

If V_{DD} changes, then current changes.

Fix: Keep V_{DD} constant

The reference current is set by the resistor, and the resistor sees $V_{DD} - V_{GS}$. Nothing about the mirror rejects a change in supply - the "fix" really is to regulate the supply, or, as the reference chapter shows, to build a current source that never lets V_{DD} into the equation in the first place.

8.10.2 Systematic variations

Next come the errors we design in ourselves: any asymmetry between the two transistors turns into a current error. The list below is long, and every line on it is avoidable.

If $V_{DS1} \neq V_{DS2} \rightarrow I_1 \neq I_2$

If layout direction of $M_1 \neq M_2 \rightarrow I_1 \neq I_2$

If current direction of $M_1 \neq M_2 \rightarrow I_1 \neq I_2$

If $V_{S1} \neq V_{S2} \rightarrow I_1 \neq I_2$

If $V_{B1} \neq V_{B2} \rightarrow I_1 \neq I_2$

If $WPE_1 \neq WPE_2 \rightarrow I_1 \neq I_2$

If $Stress_1 \neq Stress_2 \rightarrow I_1 \neq I_2 \ldots$

Every line above is the same statement: the two transistors only copy the current if they see the same everything. Different V_{DS} is channel length modulation and DIBL; different orientation or current direction is mobility anisotropy and asymmetric implants; different surroundings are WPE and stress from the previous section. These are systematic errors: the simulator with the right layout-aware models will show them, and matched layout - same orientation, same environment, dummies, common centroid - removes them. They cost area and care, not luck.

8.10.3 Process variations

Even a perfectly symmetric layout cannot save the absolute value of the current, because the process constants themselves move from lot to lot. Write out the current and look at what is inside it:

Assume strong inversion and active $V_{eff} = \sqrt{\frac{2}{\mu_p C_{ox} \frac{W}{L}} I_1}$, $V_{GS} = V_{eff} + V_{tp}$

$$I_1 = \frac{V_{DD} - V_{GS}}{R} = \frac{V_{DD} - \sqrt{\frac{2}{\mu_p C_{ox} \frac{W}{L}} I_1} - V_{tp}}{R}$$

μ_p, C_{ox}, V_{tp} will all vary from die to die, and wafer lot to wafer lot.

Solve that equation for I_1 and every process-dependent constant is inside it. Oxide grows a little thicker one lot, implant doses drift a little the next - the current changes even though every device on your die still matches its neighbor perfectly. That is the distinction to hold on to: process variation moves the whole die together; mismatch, later, moves neighbors apart.

8.10.4 Process corners

How do we simulate die-to-die movement? The foundry compresses it into corner models.

Common to use 5 corners, or Monte-Carlo process simulation

Corner	NMOS	PMOS
Mtt	Typical	Typical
Mss	Slow	Slow
Mff	Fast	Fast
Msf	Slowish	Fastish
Mfs	Fastish	Slowish

The foundry does not promise a particular die, it promises a box: every shipped wafer falls between slow-slow and fast-fast. Simulating the four corners plus typical asks “does the circuit still work at the walls of the box”. Monte Carlo instead samples the inside of the box, and is what you use when corners are too pessimistic or the failure is a yield number rather than a hard edge. Note that NMOS and PMOS need not move together - Msf and Mfs are exactly the corners that kill ratioed and skewed circuits.

8.10.5 Fix process variation

Process variation cannot be prevented, but it can be measured and corrected. Figure 43 shows the standard trick.

Use calibration: measure error, tune circuit to fix error

For every single chip, measure voltage across known resistor R_1 and tune R_{var} such that we get $I_1 = 1\mu A$

Be careful with multimeters, they have finite input resistance (typically 10 MΩ)

8.10.6 Temperature variation

The mirror must also survive from -40 C to 125 C, and temperature pulls on the square law from two directions at once.

Mobility decreases with temperature

Threshold voltage decreases with temperature.

$$I_D = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}(V_{GS} - V_{tn})^2$$

More drain current charges the load capacitances faster, so high I_D means fast digital circuits and low I_D slow ones. So:

What is fast? High temperature or low temperature?

Two knobs fight each other. Mobility drops with temperature (more lattice scattering), which slows the transistor down. The threshold voltage also drops with temperature, which - at a fixed V_{GS} - speeds it up. Which effect wins depends on how much V_{eff} you have: at high V_{DD} the mobility term dominates and hot means slow; near threshold the V_{tn} term dominates and hot means fast. The crossover is called temperature inversion, and modern low-voltage processes sit close enough to it that you cannot guess - which is exactly why the slide below refuses to give a one-line answer.

8.10.7 It depends on V_{DD}

Fast corner - Mff (high mobility, low threshold voltage) - High V_{DD} - High or low temperature

Slow corner - Mss (low mobility, high threshold voltage) - Low V_{DD} - High or low temperature

8.10.8 How do we fix temperature variation?

For this resistor-plus-mirror bias the honest answer is short; the reference chapter builds the circuits that do better.

Accept it, or don't use this circuit.

If you need stability over temperature, use 7.3.2 and 7.3.4 in CJM (SUN_BIAS_GF130N)

8.10.9 Random Variation

Even two transistors drawn identically, side by side, at the same temperature on the same die, are not identical. There are only a few thousand doping atoms under a small gate, and counting statistics does not care about your schematic: each device gets its own threshold voltage and its own ℓ. Unlike everything above, this cannot be simulated away or laid out away - only averaged away with area.

$$\ell = \mu_p C_{ox} \frac{W}{L}$$

$$I_D = \frac{1}{2}\ell(V_{GS} - V_{tp})^2$$

Due to doping , length, width, C_{ox}, V_{tp}, ... random variation

$$\ell_1 \neq \ell_2$$

$$V_{tp1} \neq V_{tp2}$$

As a result $I_1 \neq I_2$, but we can make them close.

8.10.10 Pelgrom's law [2]

Given a random gaussian process parameter ΔP with zero mean, the variance is given by

$$\sigma^2(\Delta P) = \frac{A_P^2}{WL} + S_P^2 D^2$$

where A_P and S_P are measured, and D is the distance between devices

Assume closely spaced devices ($D \approx 0$) $\Rightarrow \sigma^2(\Delta P) = \frac{A_P^2}{WL}$

Pelgrom's law is bedrock: the variance of the difference between two matched devices scales as one over the gate area, because a bigger gate averages over more atomic-scale randomness. A_P is a process constant you look up - for the threshold voltage it is a few mV per micrometer - and the $S_P D$ term says devices drift apart with distance, which is why matched pairs sit next to each other.

Pelgrom gives the spread of the raw parameters; what a designer needs is the spread of the *current*. Kinget's expression [3] connects the two:

8.10.11 Transistors with same V_{GS} [3]

$$\frac{\sigma^2_{I_D}}{I^2_D} = \frac{1}{WL}\left[\left(\frac{gm}{I_D}\right)^2 \sigma^2_{vt} + \frac{\sigma^2_\ell}{\ell^2}\right]$$

Valid in weak, moderate and strong inversion

Kinget's expression turns Pelgrom into design guidance: the relative current error has a threshold-voltage part, amplified by $(g_m/I_D)^2$, and a gain-factor part. Everything a designer chooses - region, area, current - is in there, and the two slides below read the two consequences straight out of it.

The $1/\sqrt{WL}$ scaling has a brutal price tag attached:

$$\frac{\sigma^2_{I_D}}{I^2_D} = \frac{1}{WL}\left[\left(\frac{gm}{I_D}\right)^2 \sigma^2_{vt} + \frac{\sigma^2_\ell}{\ell^2}\right]$$

$$\frac{\sigma_{I_D}}{I_D} \propto \frac{1}{\sqrt{WL}}$$

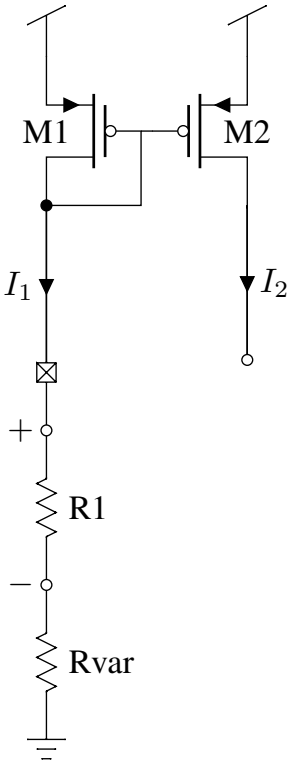
Assume $\frac{\sigma_{I_D}}{I_D} = 10\%$, We want 5%, how much do we need to change WL?

$$\frac{\frac{\sigma_{I_D}}{I_D}}{2} \propto \frac{1}{2\sqrt{WL}} = \frac{1}{\sqrt{4WL}}$$

We must quadruple the area to half the standard deviation

1%

would require **100** times the area



Area is not the only knob - the Kinget expression says the operating region matters too, and it cuts both ways:

8.10.12 What else can we do?

$$\frac{\sigma^2_{I_D}}{I^2_D} = \frac{1}{WL}\left[\left(\frac{gm}{I_D}\right)^2 \sigma^2_{vt} + \frac{\sigma^2_\ell}{\ell^2}\right]$$

Strong inversion $\Rightarrow \frac{gm}{I_D} = \frac{2}{V_{eff}} = low$

Weak inversion $\Rightarrow \frac{gm}{I_D} = \frac{q}{nkT} \approx 25$

Current mirrors achieve best matching in strong inversion

$$\frac{\sigma^2_{I_D}}{I^2_D} = \frac{1}{WL}\left[\left(\frac{gm}{I_D}\right)^2 \sigma^2_{vt} + \frac{\sigma^2_\ell}{\ell^2}\right]$$

For a differential pair the same mismatch is best expressed as a voltage: divide the current error by g_m and it becomes the offset you would have to apply at the input to cancel it.

$$\sigma^2_{I_D} = \frac{1}{WL}\left[gm^2\sigma^2_{vt} + I^2_D\frac{\sigma^2_\ell}{\ell^2}\right]$$

Offset voltage for a differential pair

$$i_o = i_{o+} - i_{o-} = g_m v_i = g_m(v_{i+} - v_{i-})$$

$$\sigma^2_{v_i} = \frac{\sigma^2_{I_D}}{gm^2} = \frac{1}{WL}\left[\sigma^2_{vt} + \frac{I^2_D}{gm^2}\frac{\sigma^2_\ell}{\ell^2}\right]$$

High $\frac{gm}{I_D}$ is better (best in weak inversion)

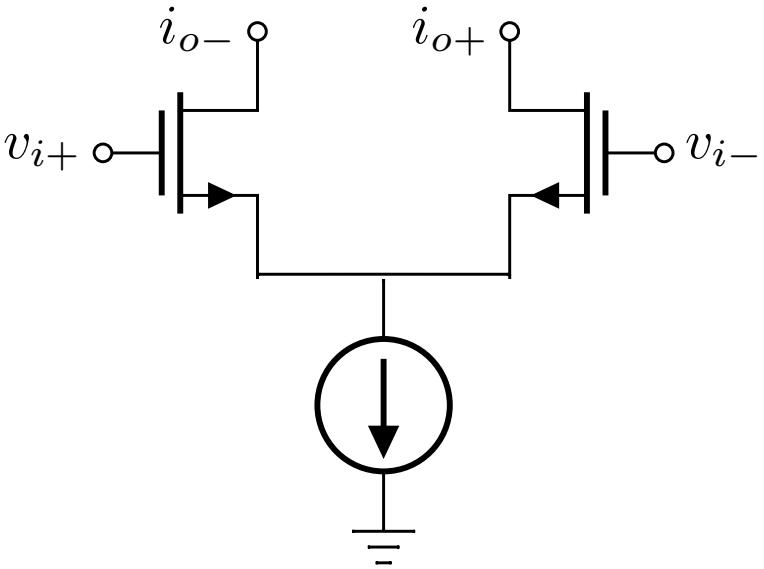


Figure 45: Differential pair

Threshold mismatch between the two input transistors appears directly as an input-referred offset voltage - the mismatch equation above divided by g_m.

8.10.13 Noise equations

Mismatch is randomness frozen in at manufacturing; noise is randomness that keeps happening while the circuit runs. The next chapter, Noise, covers where it comes from: thermal noise from hot charge, and flicker noise from traps in the oxide, one trap at a time.

For hand calculation two spectral densities are enough. The channel is a piece of resistive silicon, so it makes thermal noise; the oxide interface has traps, so it makes flicker noise. Referred to the drain and the gate respectively:

Thermal noise current at the drain

$$\overline{i^2_{nd}} = 4kT\gamma g_m \Delta f$$

$\gamma \approx 2/3$ (long channel), 1 to 2 (short channel)

Flicker noise voltage at the gate

$$\overline{v_{ng}^2} = \frac{K_f}{WLC_{ox}f}\Delta f$$

Noise corner, where the two are equal

$$f_c = \frac{K_f}{WLC_{ox}}\frac{g_m}{4kT\gamma}$$

Two design consequences fall straight out. Thermal noise, referred to the gate, is $4kT\gamma/g_m$ - spend current, get quiet. Flicker noise only cares about gate area. The corner f_c where they cross can sit anywhere from kilohertz to beyond a hundred megahertz in nanoscale CMOS, so never assume flicker is a low frequency detail. If flicker hurts: more area, a PMOS input pair (holes run a little deeper, away from the interface traps), or the circuit tricks - chopping and autozeroing.

8.11 Summary

The one-page version of this chapter:

- The gate controls a barrier: weak inversion is exponential, strong inversion is quadratic
- Transconductance is two times current over overdrive, or current over nVT - whichever is smaller
- Intrinsic gain falls with overdrive and with shorter length
- Four capacitances set the speed: Cgs usually dominates the poles (it is the largest, think current mirrors), and Cgd gets multiplied by Miller
- Match with area (Pelgrom), buy speed with overdrive and short length, buy gain with long length
- Nothing is constant: supply, process, temperature, mismatch and noise all move - design for the box, not the point

8.12 Would you like to know more?

Pelgrom's measurement of how mismatch scales with area, the paper the matching section rests on [2]

Kinget's translation of that into what mismatch costs a designer [3]

Mark Lundstrom's Essentials of MOSFETs, the most complete treatment of electrons in a MOSFET I know

[1] L. L. Lewyn, T. Ytterdal, C. Wulff, and K. Martin, "Analog Circuit Design in Nanoscale CMOS Technologies," *Proc. IEEE*, vol. 97, no. 10, pp. 1687–1714, Oct. 2009, doi: 10.1109/JPROC.2009.2024663.

[2] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching properties of MOS transistors," *ieee_j_jssc*, vol. 23, no. 5, pp. 1433–1439, 1989.

[3] P. R. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *ieee_j_jssc*, vol. 40, no. 6, pp. 1212–1224, 2005.

9 Noise

This chapter was written by Carsten Wulff and Claude, Anthropic's AI, in collaboration. The statistics are Carsten's; the physics of where noise comes from, and the reading of the GR06 trap, were drafted by Claude from Carsten's outline and reviewed and edited by him.

The first part treats noise as statistics: a mean, a variance, a spectral density, and how they add through a circuit. The second part asks where the numbers come from. Thermal and shot noise take a slide each. Flicker noise takes the rest: we follow it all the way down to the single trap - one broken bond in the oxide that grabs an electron from the channel, holds it for a while, and lets it go. On a small transistor you can watch it happen, and on a chip from this course we do. Add up enough traps and you get the flicker noise of every MOSFET you will ever design with.

Every step of that story is quantum mechanics: Schrödinger gives the trap its wave function and the channel electron its tail into the oxide, and Pauli decides how traps fill. We will not solve anything hard. We will follow the chain, and put a number on each link.

9.1 The noise floor

Noise is a phenomenon that occurs in all electronic circuits. It places a lower limit on the smallest signal we can use. Many now have super audio compact disc (SACD)* players with 24bit converters, 24 bits is around $2^{24} = 16.78$ Million different levels. If 5V is the maximum voltage, the minimum would have to be $\frac{5V}{2^{24}} \approx 298nV$. That level is roughly equivalent to the noise in a 50 Ohm resistor with a bandwidth of 96kHz. There exists an equation that relates number of bits to signal to noise ratio [1], the equation specifies that $SNR = 6.02 * Bits + 1.76 = 146.24dB$. Back in 2005 the best digital to analog converter (DAC) that Analog Devices (a very big semiconductor company) had was a DAC with 120dB SNR, that equals around $Bits = (120 - 1.76)/6.02 = 19.64$. In other words, the last four bits of your SACD player is probably noise!

* This paragraph is from the mid-2000s, when music came on discs. The compact disc (CD), launched by Philips and Sony in 1982, stores two channels of 16-bit samples at 44.1 kHz, read from a spinning disc by a laser. The super audio CD (SACD), from the same two companies in 1999, stores the music as a 1-bit sigma-delta stream at 2.8224 MHz, 64 times the CD rate - the oversampled, noise-shaped signal of the chapter on sigma-delta ADCs. Both have since given way to streaming, but the argument stands: a 24-bit converter promises more than any circuit can deliver.

9.1 The noise floor . . . 125
9.2 Three things that come in pieces . . . 126
9.3 Statistics 126
9.4 Average Power . . . 126
9.5 Noise Spectrum . . 127
9.6 Probability Distribution 129
9.7 PSD of a white noise source 129
9.8 Summing noise sources 130
9.9 Signal to Noise Ratios 131
9.10 Noise figure and Friis formula 132
9.11 Spectral Density . . 133
9.11.1 Definition of Spectral Density 133
9.11.2 Sources of Confusion 134
9.11.3 Example: Thermal Noise 135
9.11.4 Einstein: The source 136
9.12 Where white noise comes from 136
9.12.1 Thermal noise is Planck 136
9.12.2 Shot noise is the electron charge . . . 137
9.13 Where flicker noise comes from 137
9.13.1 Schrödinger 137
9.13.2 Fermi-Dirac 138
9.14 One trap 138
9.14.1 What a trap is 138
9.14.2 The channel electron has a tail 138
9.14.3 Tunnelling sets the time constant 140
9.14.4 Dirac's golden rule . 140
9.14.5 The lattice must move 141
9.14.6 Capture and emission 142
9.15 What one trap looks like 143
9.15.1 Not one Gaussian . 144
9.15.2 Two states, one Lorentzian 144
9.15.3 The Lorentzian, measured 145
9.15.4 How big is one step? 145
9.15.5 How many traps? . 146

9.2 Three things that come in pieces

Classical physics has no noise floor. A classical charge fluid could be as smooth as you like. Real noise exists because three things come in pieces, and each piece gives one kind of transistor noise.

What is discrete	Noise it gives
Charge, q	Shot noise, $2qI$
Energy per mode, hf	Thermal noise, $4kTR$
Localized states, traps	Telegraph noise, then $1/f$

The first part of this chapter is the statistics we need to describe any of them. Then we go looking for where they come from.

9.3 Statistics

The mean of a signal x(t) is defined as

$$\overline{x(t)} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t)dt \tag{1}$$

The mean square of x(t) defined as

$$\overline{x^2(t)} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x^2(t)dt \tag{2}$$

The variance of x(t) defined as

$$\sigma^2 = \overline{x^2(t)} - \overline{x(t)}^2 \tag{3}$$

For a signal with a mean of zero the variance is equal to the mean square. The auto-correlation of x(t) is defined as

$$\begin{aligned} R_x(\tau) &= \overline{x(t)x(t+\tau)} \\ &= \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t)x(t+\tau)dt \end{aligned}$$

9.4 Average Power

Average power is defined for a continuous system by (4), and for discrete samples by (5).

P_{av} usually has the unit A^2 or V^2, so we have to multiply/divide by the impedance to get the power in Watts. To get Volts and Amperes

we use the root-mean-square (RMS) value which is defined as $\sqrt{P_{av}}$.

$$P_{av} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x^2(t)dt \tag{4}$$

$$P_{av} = \frac{1}{N} \sum_{i=0}^{N} x^2(i) \tag{5}$$

If x(t) has a mean of zero then, according to (3), P_{av} is equal to the variance of x(t).

Many different notations are used to denote average power and RMS value of voltage or current, some of them are listed in the two tables below. Notation can be a confusing thing, it changes from book to book and makes expressions look different.

It is important to realize that it does not matter how you write average power and RMS value. If you want you can invent your own notation for average power and RMS value. However, if you are presenting your calculations to other people it is convenient if they understand what you have written. In the remainder of this chapter we will use $\overline{e_n^2}$ for average power when we talk about voltage noise source and $\overline{i_n^2}$ for average power when we talk about current noise source. The n subscript is used to identify different sources and can be whatever.

Voltage	Current
V_{rms}^2	I_{rms}^2
$\overline{V_n^2}$	$\overline{I_n^2}$
$\overline{v_n^2}$	$\overline{i_n^2}$

Voltage	Current
V_{rms}	I_{rms}
$\sqrt{\overline{V_n^2}}$	$\sqrt{\overline{I_n^2}}$
$\sqrt{\overline{v_n^2}}$	$\sqrt{\overline{i_n^2}}$

9.5 Noise Spectrum

With random noise it is useful to relate the average power to frequency. We call this Power Spectral Density (PSD). A PSD plots how much power a signal carries at each frequency. In literature $S_x(f)$ is often used to denote the PSD. In the same way that we use V^2 as unit of average power, the unit of the PSD is $\frac{V^2}{Hz}$ for voltage

and $\frac{A^2}{Hz}$ current. The root spectral density is defined as $\sqrt{S_x(f)}$ and has unit $\frac{V}{\sqrt{Hz}}$ for voltage and $\frac{A}{\sqrt{Hz}}$ for current.

The power spectral density is defined as two times the Fourier transform of the auto-correlation function [2]

$$S_x(f) = 2\int_{-\infty}^{\infty} R_x(\tau)e^{-j2\pi f\tau}d\tau \tag{6}$$

This can also be written as

$$\begin{aligned}
S_x(f) &= 2\left[\int_{-\infty}^{\infty} R_x(\tau)\cos(\omega\tau)d\tau - \int_{-\infty}^{\infty} R_x(\tau)j\sin(\omega\tau)d\tau\right] \\
&= 2\left[\int_{-\infty}^{0} R_x(\tau)\cos(\omega\tau)d\tau + \int_{0}^{\infty} R_x(\tau)\cos(\omega\tau)d\tau\right] \\
&- 2j\left[\int_{-\infty}^{0} R_x(\tau)\sin(\omega\tau)d\tau + \int_{0}^{\infty} R_x(\tau)\sin(\omega\tau)d\tau\right] \\
&= 4\int_{0}^{\infty} R_x(\tau)\cos(\omega\tau)d\tau \\
&- 2j\left[-\int_{0}^{\infty} R_x(\tau)\sin(\omega\tau)d\tau + \int_{0}^{\infty} R_x(\tau)\sin(\omega\tau)d\tau\right] \\
&= 4\int_{0}^{\infty} R_x(\tau)\cos(\omega\tau)d\tau
\end{aligned}$$

, since $e^{-j\omega\tau} = \cos(\omega\tau) - j\sin(\omega\tau)$, $R_x(\tau)$ and $\cos(\omega\tau)$ are symmetric around $\tau = 0$ while $\sin(\omega\tau)$ is asymmetric around $\tau = 0$.

The inverse of power spectral density is defined as

$$R_x(\tau) = \frac{1}{2}\int_{-\infty}^{\infty} S_x(f)e^{j2\pi f\tau}df = \int_{0}^{\infty} S_x(f)\cos(\omega\tau)df$$

If we set $\tau = 0$ we get

$$\overline{x^2(t)} = \int_{0}^{\infty} S_x(f)df \tag{7}$$

which means we can easily calculate the average power if we know the power spectral density. As we will see later it is common to express noise sources in PSD form.

Another very useful theorem when working with noise in the frequency domain is this

$$S_y(f) = S_x(f)|H(f)|^2 \tag{8}$$

, where $S_y(f)$ is the output power spectral density, $S_x(f)$ is the input power spectral density and $H(f)$ is the transfer function of a time-invariant linear system.

If we insert (8) into (7), with $S_x(f) = a\ constant = D_v$ we get

$$\overline{x^2(t)} = \int S_y(f)df = D_v \int |H(f)|^2 df = D_v f_x$$

, where f_x is what we call the noise bandwidth. For a single time constant RC network the noise bandwidth is equal to

$$f_x = \frac{\pi f_0}{2} = \frac{1}{4RC}$$

where f_x is the noise bandwidth and f_0 is the 3dB frequency.

We haven't told you this yet, but thermal noise is white and white means that the power spectral density is flat (constant over all frequencies). If $S_x(f)$ is our thermal noise source and $H(f)$ is a standard low pass filter, then (8) tells us that the output spectral density will be shaped by $H(f)$. At frequencies above the f_x in $H(f)$ we expect the root power spectral density to fall by 20dB per decade.

9.6 Probability Distribution

Theorem 1 (Central limit theorem). *The sum of n independent random variables subjected to the same distribution will always approach a normal distribution curve as n increases.*

This is a neat theorem, it explains why many noise sources we encounter in the real world are Gaussian.[†] Take thermal noise for example, it is generated by random motion of carriers in materials. If we look at a single electron moving through the material the probability distribution might not be Gaussian. But summing probability distribution of the random movements with a large number of electrons will give us a Gaussian distribution, thus thermal noise is Gaussian.

9.7 PSD of a white noise source

If we have a true random process with Gaussian distribution we know that the autocorrelation function only has a value for $\tau = 0$. From the definition of auto-correlation we have that

[†] Gaussian distribution = normal distribution. Gaussian describes the amplitude distribution; white describes a flat spectral density. Thermal noise happens to be both.

$$\begin{aligned} R_x(\tau) &= \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t)x(t-\tau)dt \\ &= \left[\lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x^2(t)dt \right] \delta(\tau) \\ &= \overline{x^2(t)}\delta(\tau) \end{aligned}$$

The reason being that in a true random process $x(t)$ is uncorrelated with $x(t+\tau)$ for any $\tau \neq 0$. If we use (6) we see that

$$\begin{aligned} S_x(f) &= 2 \int_{-\infty}^{\infty} \overline{x^2(t)}\delta(\tau)e^{-j2\pi f\tau}d\tau \\ &= 2\overline{x^2(t)} \int_{-\infty}^{\infty} \delta(\tau)e^{-j2\pi f\tau}d\tau \\ &= 2\overline{x^2(t)} \end{aligned}$$

, since

$$\int \delta(\tau)e^{-j2\pi f\tau}d\tau = e^0 = 1$$

This means that the power spectral density of a white noise source is flat, or in other words, the same for all frequencies.

9.8 Summing noise sources

Summing noise sources is usually trivial, but we need to know why and when it is not. If we write the time dependant noise signals as

$$v_{tot}^2(t) = (v_1(t) + v_2(t))^2 = v_1^2(t) + 2v_1(t)v_2(t) + v_2^2(t)$$

The average power is defined as

$$\begin{aligned}\overline{e_{tot}^2} &= \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{+T/2} v_{tot}^2(t)dt \\ &= \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{+T/2} v_1^2(t)dt \\ &+ \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{+T/2} v_2^2(t)dt \\ &+ \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{+T/2} 2v_1(t)v_2(t)dt \\ &= \overline{e_1^2} + \overline{e_2^2} + \lim_{T\to\infty}\frac{1}{T}\int_{-T/2}^{+T/2} 2v_1(t)v_2(t)dt\end{aligned}$$

If $\overline{e_1^2}$ and $\overline{e_2^2}$ are uncorrelated noise sources we can skip the last term in the sum above and just write

$$\overline{e_{tot}^2} = \overline{e_1^2} + \overline{e_2^2}$$

Most natural noise sources are uncorrelated.

9.9 Signal to Noise Ratios

Signal to Noise Ratio (SNR) is a common method to specify the relation between signal power and noise power in linear systems. It is defined as

$$\begin{aligned}SNR &= 10\log\left(\frac{Signal\ power}{Noise\ power}\right) \\ &= 10\log\left(\frac{\overline{v_{sig}^2}}{\overline{e_n^2}}\right) \\ &= 20\log\left(\frac{v_{rms}}{\sqrt{\overline{e_n^2}}}\right)\end{aligned}$$

Another useful ratio is Signal to Noise and Distortion (SNDR), since most real systems exhibit non-linearities it is useful to include distortion in the ratio. One can calculate SNR and SNDR in many ways. If we don't know the expression for $\overline{e_n^2}$ we can do a FFT of our output signal. From this FFT we sum spectral components except at the signal frequency to get noise and distortion. SNR is normally calculated as

$$SNR = 10\log\left(\frac{Signal\ power}{Noise\ power\ -\ 6\ first\ harmonics}\right)$$

And SNDR is calculated as

$$SNDR = 10\log\left(\frac{Signal\ power}{Noise\ power}\right)$$

9.10 Noise figure and Friis formula

Noise factor is a measure on the noise performance of a system. It is defined as

$$F = \frac{\overline{v_o^2}}{source\ contribution\ to\ \overline{v_o^2}}$$

where $\overline{v_o^2}$

is the total output noise. The noise figure is defined as (noise factor in dB)

$$NF = 10\log(F)$$

The noise factor can also be defined as

$$F = \frac{SNR_{input}}{SNR_{output}}$$

This brings us right into what is known as Friis formula. The noise factor definition is only correct at room temperature, for more details, see [3]. If we have a multistage system, for example several amplifiers in cascade, the total noise figure of the system is defined as

$$F = 1 + F_1 - 1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + ...$$

Here F_i is the noise figures of the individual stages and G_i is the available gain of each stage. This can be rewritten as

$$F = F_1 + \sum_{i=1}^{N} \frac{F_{i+1} - 1}{\prod_{k=1}^{i} G_k}$$

Friis' formula tells us that it is the noise in the first stage that is the most important if G_1 is large. We could say that in a system it is important to amplify the noise as early as possible!

9.11 Spectral Density

Warning: This is not an introduction to spectral density. If the subject is completely unfamiliar I'd advise reading another source. For example chapter 4 in [1] or chapter 7 in [4].

9.11.1 Definition of Spectral Density

There are two different definitions of spectral density used in the literature. They differ by a factor of two. The one used in signal processing books, like [5], is

$$S_{x1}(f) = \int_{-\infty}^{\infty} R_{x1}(\tau) e^{-j\omega\tau} d\tau$$

And the one often used in books about noise, like [2], is

$$S_{x2}(f) = 2\int_{-\infty}^{\infty} R_{x2}(\tau) e^{-j\omega\tau} d\tau$$

In both cases $R_{xi}(\tau)$ is the auto-correlation function defined as

$$R_{xi}(\tau) = \overline{x_i(t)x_i(t+\tau)}$$

As we can plainly see

$$S_{x1}(f) \neq S_{x2}(f)$$

, there is no way these two can be made equal if

$$R_{x1}(\tau) = R_{x2}(\tau)$$

This is ok, there is no problem having two different definitions for two different functions. In reality $S_{x1}(f)$ and $S_{x2}(f)$ are different functions of frequency, and we could say that

$$S_{x2}(f) = 2S_{x1}(f)$$

if the two auto-correlation functions are equal.

9.11.2 Sources of Confusion

The problem with spectral density arises when reading literature from different communities, for example [5] and [2] where $S_x(f)$ is used for both $S_{x1}(f)$ and $S_{x2}(f)$. When I started investigating spectral densities this lead me to believe that different sources defined the same measure "spectral density" in two different ways. The more sources I investigated the more unsure I was about which of the two definitions that was correct. After months of searching (not actively, but sporadically) I eventually found the original source of the definition of spectral density [6]. Having the original source helped, but I still don't know when the original definition split into the S_{x1} and S_{x2} forms above. However, I'm pretty sure it's just a matter of convenience. To see why the S_{x2} form is the most common among sources concerning noise we look at the inverse Fourier Transform. By the way, if you had not noticed yet, both forms say that *Spectral density is the Fourier Transform of the Auto-Correlation function*. The inverse Fourier Transform of S_{x1} is

$$R_{x1}(\tau) = \frac{1}{2\pi}\int_{-\infty}^{\infty} S_{x1}(f)e^{j\omega\tau}dw = \int_{-\infty}^{\infty} S_{x1}(f)e^{j\omega\tau}df$$

,since $dw = df\,dw/df = 2\pi df$. And for S_{x2}

$$R_{x2}(\tau) = \frac{1}{2}\int_{-\infty}^{\infty} S_{x2}(f)e^{jw\tau}df$$

Before we proceed lets get rid of the e's. We know that $e^{j\alpha} = \cos\alpha + j\sin\alpha$. So we could rewrite S_{x1} as

$$S_{x1}(f) = \int_{-\infty}^{\infty} R_{x1}(\tau)[\cos(\omega\tau) + j\sin(\omega\tau)]d\tau$$

and it turns out that since $R_{x1}(\tau)$ is an even function we can drop the $j\sin\omega\tau$ term. $S_{x1}(f)$ is also an even function since the Fourier Transform of an even function is even.

The definitions then become

$$S_{x1}(f) = \int_{-\infty}^{\infty} R_{x1}(\tau)\cos(\omega\tau)d\tau$$
$$R_{x1}(\tau) = \int_{-\infty}^{\infty} S_{x1}(f)\cos(\omega\tau)df$$

and

$$S_{x2}(f) = 2\int_{-\infty}^{\infty} R_{x2}(\tau)\cos(\omega\tau)d\tau$$
$$R_{x2}(\tau) = \frac{1}{2}\int_{-\infty}^{\infty} S_{x2}(f)\cos(\omega\tau)df$$

We can rewrite $R_{x2}(\tau)$ as

$$R_{x2}(\tau) = \overline{x_2(t)x_2(t+\tau)} = \int_0^{\infty} S_{x2}(f)\cos(\omega\tau)df$$

and if $\tau = 0$

$$\overline{x_2^2(t)} = \int_0^{\infty} S_{x2}(f)df$$

So using the S_{x2} definition we see that average power (mean square value of $x_2(t)$) is equal to the integral from 0 to infinity of the spectral density. If we use S_{x1} instead, average power would be

$$\overline{x_1^2(t)} = 2\int_0^{\infty} S_{x1}(f)df$$

But if $R_{x1}(\tau) = R_{x2}(\tau)$ then

$$\overline{x_2^2(t)} = \overline{x_1^2(t)}$$

even though $S_{x1}(f) \neq S_{x2}(f)$.

S_{x1} is called the two-sided spectral density, and S_{x2} the one-sided spectral density.

9.11.3 Example: Thermal Noise

The spectral density of thermal noise in electronic circuit should be known to anyone that has studied analog electronics. We normally define the voltage spectral density of thermal noise as

$$S_{th}(f) = 4kTR$$

where k is Boltzmann's constant, T the temperature in Kelvin and R the resistance. But that is the spectral density in the one-sided S_{x2} definition. If we were to use the two-sided S_{x1} definition, then the spectral density of thermal noise would be

$$S_{th}(f) = 2kTR$$

Both these spectral densities would give the same average power value if we use the inverse Fourier Transform of the matching definition.‡

9.11.4 Einstein: The source

In his 1914 paper [6] Albert Einstein described, supposedly for the first time, the auto-correlation function and what we have come to know as the spectral density. He defined the auto-correlation function as

$$\mathfrak{M}(\Delta) = \overline{F(t)F(t+\Delta)}$$

and the intensity (spectral density) as

$$I(\theta) = \int_0^T \mathfrak{M}(\Delta)\cos(\pi\frac{\Delta}{\theta})d\Delta$$

,where the period $\theta = T/n$ and T is a very large value. The paper is very short, only 1 page, but it is worth reading. Note that the spectral density as the Fourier Transform of the auto-correlation function is often referred to as the *Wiener-Khintchine* theorem.

9.12 Where white noise comes from

9.12.1 Thermal noise is Planck

$$S_V(f) = 4R\frac{hf}{e^{hf/kT}-1} \approx 4kTR \quad \text{if } hf \ll kT$$

$$\frac{kT}{h} = 6.25 \text{ THz at } 300 \text{ K}$$

Nyquist derived thermal noise in 1928 [7], and he did not write $4kTR$. He wrote the Planck form above. A resistor in thermal equilibrium feeds energy into every electrical mode of the circuit around it, and each mode, being quantized in steps of hf, holds the Planck average energy. Below kT/h every mode holds kT, and the density is flat.

At room temperature kT/h is 6 THz. No circuit in this course gets near it, which is why we may forget Planck and write $4kTR$. A superconducting qubit at 10 mK has kT/h = 200 MHz, and there the quantum correction is the whole design problem.

‡ Note that if you calculate the average power of $S_{th}(f)$ you'll get infinity. You have to include the bandwidth of the circuit you are considering for average power to have a finite value.

9.12.2 Shot noise is the electron charge

$$\overline{i_n^2} = 2qI\,\Delta f$$

In weak inversion, with $\gamma = n/2$ and $g_m = I_D/nV_T$

$$4kT\gamma g_m = 4kT\frac{n}{2}\frac{I_D}{nV_T} = 2qI_D$$

Current that crosses a barrier one electron at a time is a stream of independent arrivals, and the arrivals are random. That is shot noise.

In weak inversion the MOSFET is exactly that: electrons diffuse over the source barrier one at a time. Put the weak-inversion γ and g_m into the thermal noise formula and it becomes $2qI_D$. Thermal noise and shot noise in a subthreshold transistor are the same noise, described twice.

9.13 Where flicker noise comes from

9.13.1 Schrödinger

$$i\hbar\frac{\partial}{\partial t}\Psi(\vec{r},t) = \left[-\frac{\hbar^2}{2m}\nabla^2 + V(\vec{r})\right]\Psi(\vec{r},t)$$

Stationary states, $\Psi = \psi(\vec{r})e^{-iEt/\hbar}$

$$-\frac{\hbar^2}{2m}\nabla^2\psi + V\psi = E\psi$$

The wave function Ψ is a probability amplitude: $|\Psi|^2$ is the probability of finding the electron at $\vec{r}$. The potential V is everything the electron feels - the silicon nuclei, the other electrons, the gate field.

Three solutions of this equation carry the whole chapter. In a periodic crystal it gives energy bands, and near a band edge the electron moves as if free, with an effective mass. In the surface well of an inverted MOSFET it gives subbands, and the "2d sheet" of the MOSFET chapter. And at a broken bond it gives a bound state: a wave function localized to an atom or two, with an energy in the bandgap. That is a trap.

The equation has one more thing to say, and we need it. Where $E < V$ the solution does not stop. It decays as $e^{-\kappa x}$, with $\kappa = \sqrt{2m(V-E)}/\hbar$. The electron has a tail inside the barrier.

9.13.2 Fermi-Dirac

$$f(E) = \frac{1}{1 + e^{(E-E_F)/kT}}$$

$$n = \int_{E_C}^{\infty} N(E) f(E)\, dE$$

The probability that a state at energy E holds an electron. Electrons have spin $\frac{1}{2}$, and Pauli allows at most one electron per quantum state, so they fill states by Fermi-Dirac and not by Boltzmann. The MOSFET chapter used it to count the electrons in the conduction band: the density of states $N(E)$, from Schrödinger, times the occupation $f(E)$. We will use it again in two places: to decide which traps can switch, and to find out where the kT in flicker noise comes from.

9.14 One trap

9.14.1 What a trap is

- A broken bond at the Si/SiO2 interface, or a defect a few nm into the oxide
- A bound state: ψ_T localized to an atom or two
- An energy E_T inside the silicon bandgap

Silicon and its oxide do not fit perfectly. At the interface some silicon atoms are left with a bond that has no partner - a dangling bond - and the amorphous oxide has its own defects, oxygen vacancies and strained bonds, a few nanometres deep. Solve Schrödinger's equation around one and you get a bound state: a wave function localized to an atom or two, at an energy inside the bandgap.

9.14.2 The channel electron has a tail

$$\psi_{ch}(x) \propto e^{-\kappa x} \text{ in the oxide}, \quad \kappa = \frac{\sqrt{2m_{ox}(\Phi_B - E)}}{\hbar}$$

$$\Phi_B = 3.1 \text{ eV}, \quad m_{ox} \approx 0.5 m_0 \Rightarrow \kappa \approx 6.4 \text{ nm}^{-1}$$

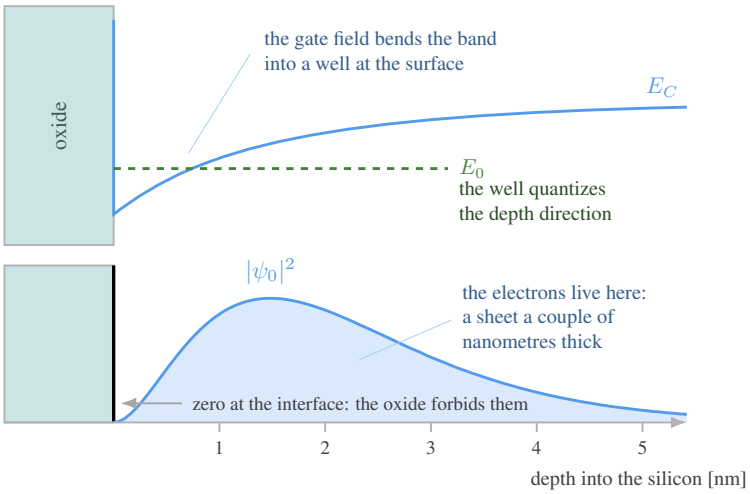


Figure 1: The inversion layer in depth, from the MOSFET chapter. The density is drawn as zero at the interface - the oxide forbids the electrons there - but forbids is too strong. A small tail leaks into the oxide

The MOSFET chapter drew the inversion electrons in the lowest subband of the surface well, with a density that goes to zero at the oxide. It is close to zero, not zero. The oxide is a barrier 3.1 eV high, and the Schrödinger equation inside a barrier gives an exponential.

Put the numbers in: with an effective mass in the oxide of half the free electron mass, κ is 6.4 per nanometre. The probability density goes as $|\psi|^2 \propto e^{-2\kappa x}$, and falls a decade every 0.18 nm. Small, but a trap one nanometre into the oxide still overlaps the channel.

9.14.3 Tunnelling sets the time constant

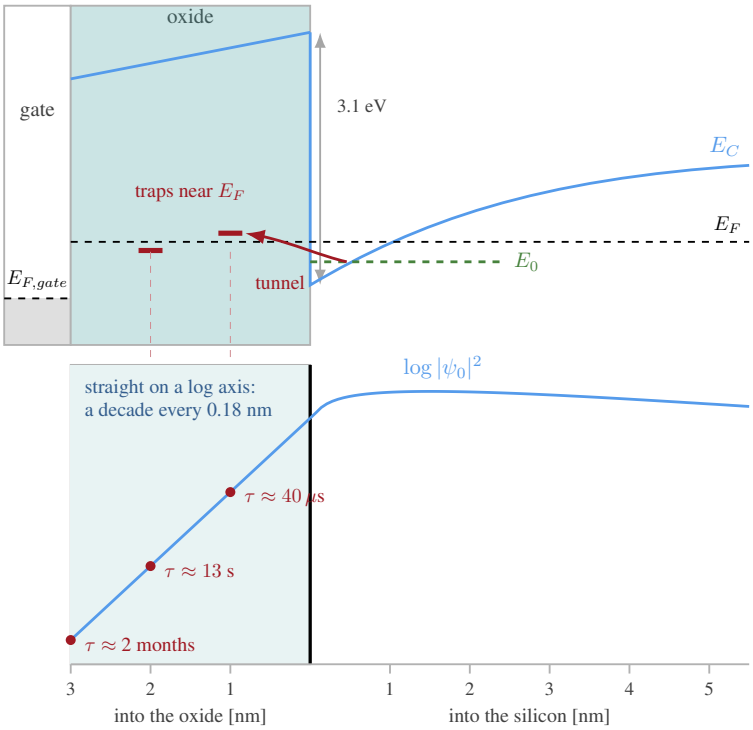


Figure 2: Above: a trap in the oxide with its level near the Fermi level can swap an electron with the channel. Below: the channel electron density on a log axis. In the oxide it is a straight line, a decade every 0.18 nm, and the time constant of a trap is set by where on that line it sits

An electron in the channel and an empty trap at depth x_T are two states with almost the same energy, coupled through the overlap of their wave functions. The rate of hopping from one to the other is proportional to the channel density at the trap. Figure 2 shows the consequence: time constants that span microseconds to months, from traps a nanometre apart.

9.14.4 Dirac's golden rule

$$\frac{1}{\tau} = \frac{2\pi}{\hbar} |\langle \psi_T | H' | \psi_{ch} \rangle|^2 \, \rho(E)$$

$$|\langle \psi_T | H' | \psi_{ch} \rangle|^2 \propto e^{-2\kappa x_T}$$

$$\tau = \tau_0 e^{x_T/\lambda}, \quad \lambda = \frac{1}{2\kappa} \approx 0.08 \text{ nm}$$

Trap depth	τ ($\tau_0 = 0.1$ ns)
0 nm	0.1 ns
1 nm	40 µs

Trap depth	τ ($\tau_0 = 0.1$ ns)
2 nm	13 s
3 nm	2 months

The rate of a transition is the squared matrix element between the two states times the density of final states. Dirac worked it out in 1927 [8], in his own bra-ket notation; Fermi later called it a "golden rule", and the name stuck to Fermi. The trap wave function is tiny, so the matrix element samples the channel wave function where the trap sits, and the squared matrix element inherits $e^{-2\kappa x_T}$.

The table is the punchline. Three nanometres of oxide hold sixteen decades of time constant. No other mechanism in a transistor produces a spread like that, and it is exactly what $1/f$ noise needs, as we will see. The prefactor τ_0 is rough - textbooks quote 0.1 ns give or take a decade - but the slope is set by the barrier, and that is the part that matters. McWhorter used the same tunnelling picture in 1957 [9], twenty years before anyone saw a single trap.

9.14.5 The lattice must move

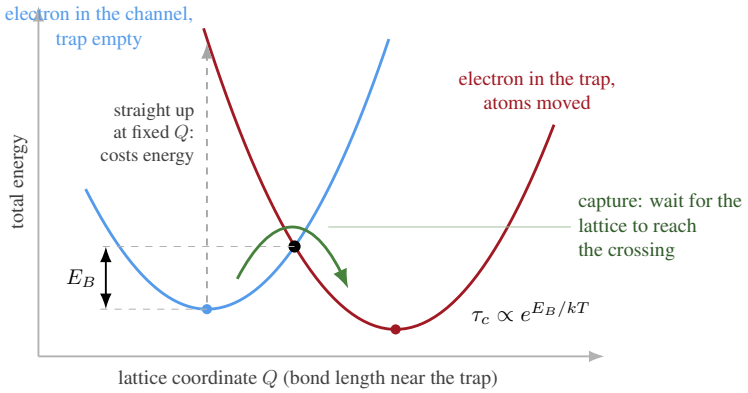


Figure 3: The configuration coordinate diagram. The trap is empty on the blue curve and full on the red one, and the atoms around a full trap sit somewhere else. The electron can only hop where the curves cross, so it waits for a thermal fluctuation of the lattice to get there

Tunnelling is not the whole story, and the measurements say so: trap time constants depend strongly on temperature, and tunnelling through an oxide barrier barely does.

The missing piece is the atoms. When a trap captures an electron its charge changes, and the bonds around it stretch or bend to suit. The total energy - electron plus lattice - is a parabola in the lattice coordinate, and there is one parabola for the empty trap and another for the full one, shifted sideways. The electron cannot drop straight from one to the other: at a fixed lattice position the full state costs more energy than the electron has, and the lattice

cannot absorb the difference in one go. The hop happens where the two curves cross, and the lattice reaches the crossing only by a thermal fluctuation.

That wait is $e^{E_B/kT}$. In golden-rule language, the density of final states $\rho(E)$ now counts lattice vibrations - phonons - and the overlap of lattice states before and after is what makes the rate thermally activated. Henry and Lang worked out this multiphonon capture in 1977 [10].

9.14.6 Capture and emission

$$\frac{1}{\tau_c} \propto e^{-x_T/\lambda}\, e^{-E_B/kT}\, n_s$$

$$\frac{\tau_c}{\tau_e} = \frac{1}{g} e^{(E_T - E_F)/kT}, \quad g = 2$$

$$\frac{\partial \ln(\tau_c/\tau_e)}{\partial V_{GS}} = -\frac{q}{kT}\frac{x_T}{t_{ox}}$$

Capture needs an electron in the channel near the trap, so its rate grows with the inversion density n_s. Beyond that it is the product of the two factors we have: tunnel to depth x_T, and wait for the lattice.

Emission is the reverse trip, and the ratio of the two is fixed by thermal equilibrium: the trap must be occupied with the Fermi-Dirac probability. An empty trap can accept an electron of either spin, and a second electron costs extra Coulomb energy, so it belongs to a different level further up. “Full” is two quantum states and “empty” is one: that is the factor $g = 2$.

A trap only switches if E_T is within a few kT of E_F. Far below, it is always full; far above, always empty; either way it is silent.

The gate moves E_F past E_T. In strong inversion the surface potential is pinned, so extra gate voltage drops across the oxide, and a trap a fraction x_T/t_{ox} of the way to the gate sees that fraction of it. Measure τ_c/τ_e against V_{GS}, and the slope tells you how deep the trap is. Kirton and Uren made a catalogue of traps this way [11].

9.15 What one trap looks like

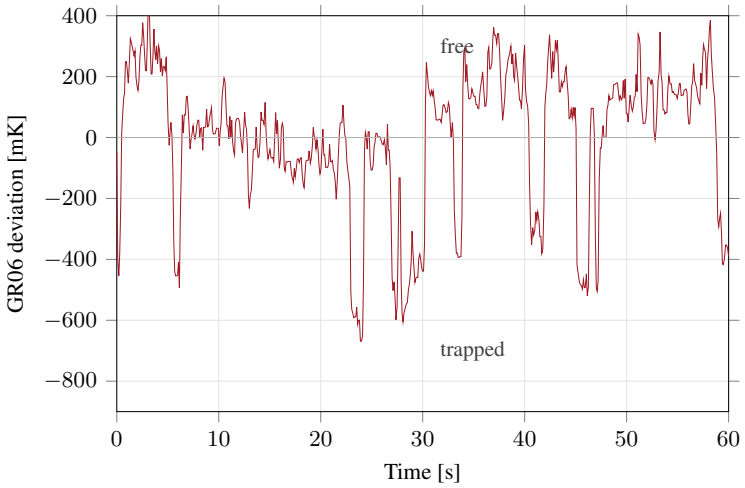


Figure 4: Sixty seconds of the GR06 temperature sensor from the measurement chapter, slow drift removed. The reading sits on one level, drops by about 0.6 K to another, stays a second or two, and returns

This is a chip from this course. The temperature sensor's output is not a fuzzy band but two levels, which is what a single trap would do: empty, or full. The sensor turns currents and voltages into a temperature reading, so the step shows up in kelvin. Whether it really is a trap, and in which transistor, is a question we come back to once we know how big one trap's step can be.

Ralls and colleagues first saw the same thing in the drain current of small MOSFETs in 1984 [12]. It goes by three names: random telegraph noise, popcorn noise, and burst noise.

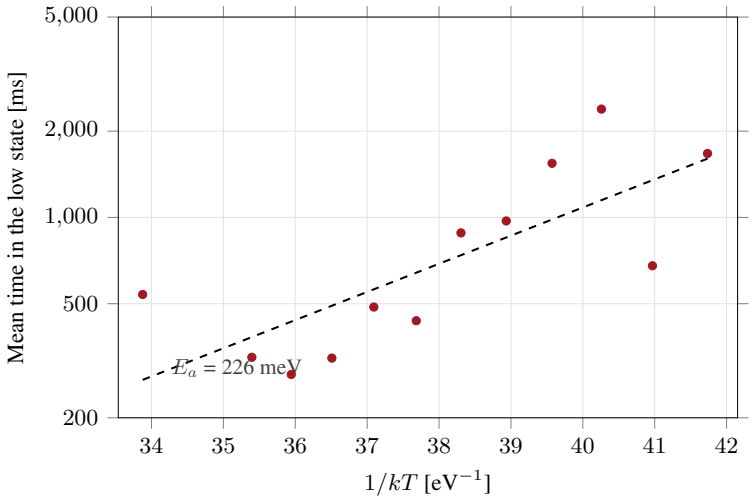


Figure 5: The GR06 trap's mean time in the low state against inverse thermal energy. A straight line is an Arrhenius law, with an activation energy of 226 meV

And here is the lattice, measured. The time the trap stays in one state falls exponentially with temperature: a straight line against $1/kT$, slope 226 meV. Tunnelling alone would give a time constant almost independent of temperature. The 226 meV is the E_B of Figure 3 plus the Fermi-Dirac shift of the previous slide - the lattice has to move before the electron can.

9.15.1 Not one Gaussian

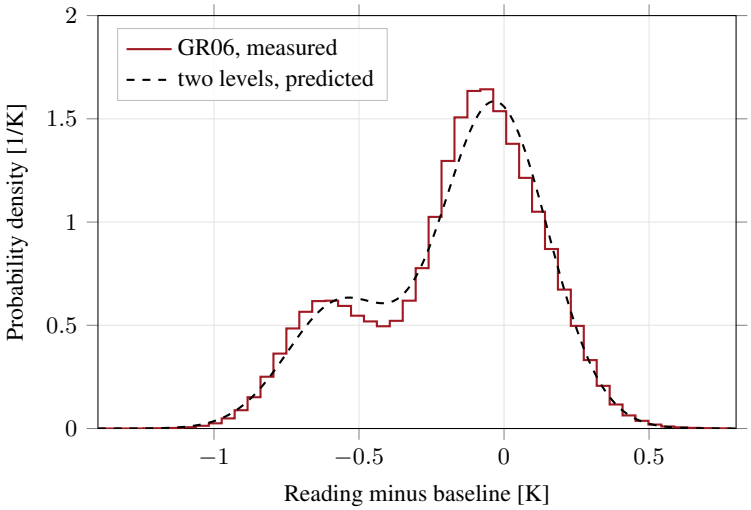


Figure 6: Two hours of the GR06 reading against its slow baseline. One trap gives two Gaussians, 534 mK apart, weighted 72 % and 28 % by the time spent in each state, each as wide as the 183 mK noise inside a state. The dashed line is that sum, not a fit

The central limit theorem of the first part says that many small independent sources add up to a Gaussian. One trap is not many sources. Its amplitude has two values, and the white noise of the sensor blurs each of them into a Gaussian of its own. A histogram is the quickest test there is for a telegraph signal hiding in a noisy reading.

9.15.2 Two states, one Lorentzian

$$R_x(\tau) = \Delta I^2 \frac{\tau_c \tau_e}{(\tau_c + \tau_e)^2} e^{-|\tau|/\tau_0}, \quad \frac{1}{\tau_0} = \frac{1}{\tau_c} + \frac{1}{\tau_e}$$

$$S_x(f) = 4 \int_0^\infty R_x(\tau) \cos(\omega\tau) d\tau = \frac{4\,\Delta I^2}{(\tau_c + \tau_e)\left[\left(\frac{1}{\tau_c} + \frac{1}{\tau_e}\right)^2 + (2\pi f)^2\right]}$$

From here on the quantum mechanics is done, and we are back in the statistics of the first part. A trap has two states, and the time it spends in each is random, with means τ_c and τ_e. Since the trap has no memory of how long it has waited, the autocorrelation decays exponentially. Fourier transform it with the one-sided definition from the first part and you get a Lorentzian: flat below the corner $f_0 = 1/(2\pi\tau_0)$, falling as $1/f^2$ above it. Machlup worked it out in 1954 [13].

The factor $\tau_c\tau_e/(\tau_c + \tau_e)^2$ is the variance of a coin that lands full a fraction f_T of the time, $f_T(1 - f_T)$. It is largest when $E_T = E_F$ and the coin is fair.

9.15.3 The Lorentzian, measured

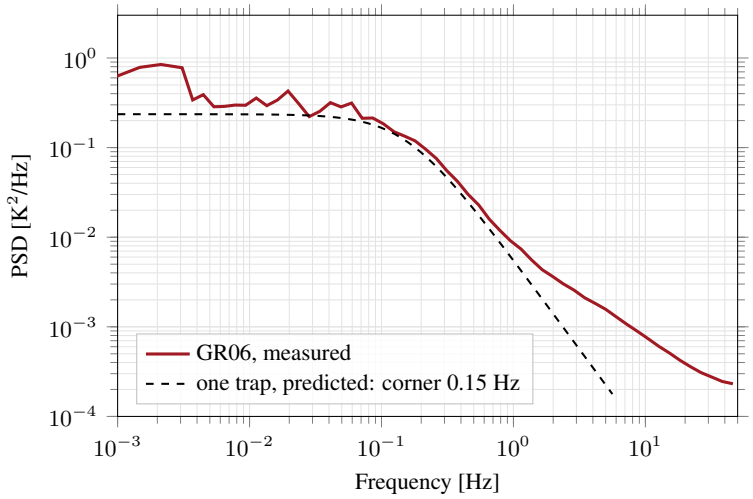


Figure 7: The spectrum of the same two hours, as measured. The dashed line is Machlup's Lorentzian with the measured step and dwell times, not a fit: flat below 0.15 Hz, falling as $1/f^2$ above. The slow temperature drift rises below 30 mHz. Above 1 Hz the reading has more than one trap: short visits the detector misses, and the sensor's own noise

Put the step, 534 mK, and the two mean dwell times, 3.7 s and 1.4 s, into the formula on the previous slide, and you have a prediction of the spectrum with nothing left to adjust. It lands on the measurement: the plateau, the corner and the $1/f^2$ slope. The dwell times come from counting edges in the time domain, the spectrum from a Fourier transform of the raw reading, and the Lorentzian ties the two together.

9.15.4 How big is one step?

$$\Delta V_T = \frac{q}{C_{ox}WL}\left(1 - \frac{x_T}{t_{ox}}\right), \quad \Delta I_D = g_m \Delta V_T$$

W x L	ΔV_T	$\Delta I_D/I_D$ at $g_m/I_D = 20$
0.42 µm x 0.15 µm	305 µV	0.6 %
1 µm x 1 µm	19 µV	0.04 %
10 µm x 10 µm	0.19 µV	0.0004 %

sky130 nfet_01v8: $t_{oxe} = 4.148\,nm$, $C_{ox} = 8.3\,fF/\mu m^2$

One electron in the oxide images one electron's worth of charge onto the gate and the channel. Seen from the gate it is a threshold shift $q/(C_{ox}WL)$, slightly less for a trap deep in the oxide, and the drain current moves by g_m times that. The oxide thickness is in the model card: sky130's 1.8 V nfet has an electrical oxide thickness t_{oxe} of 4.148 nm, the pfet 4.23 nm, which with $\varepsilon_{ox} = 3.9\varepsilon_0$ is 8.3 and 8.2 fF/µm².

In a minimum-size sky130 transistor that is a third of a millivolt - more than half a percent of the current, from a single electron. Measured steps are often larger still, because the current does not flow evenly across the channel and a trap sitting on a busy path blocks more than its share. The charged trap also scatters the electrons that pass it, so mobility fluctuates along with the number of carriers. Hung's unified model combines the two effects [14].

9.15.5 How many traps?

Per decade of time constant

$$N \approx N_t \cdot kT \cdot WL \cdot \lambda \ln 10$$

$N_t = 4 \times 10^{17}$ cm^{-3}eV^{-1} (sky130 nfet), $kT = 26$ meV, $\lambda = 0.1$ nm

W x L	Active traps per decade
0.42 µm x 0.15 µm	0.15
1 µm x 1 µm	2.4
10 µm x 10 µm	240

A trap is only noisy if its level is near E_F and its time constant is inside the band you care about. Weighted by how noisy each one is, the energy window is kT wide (we will meet that integral again). A decade of time constant is $\lambda \ln 10$ of depth, about a quarter of a nanometre. So count the traps in a slab that thick and kT wide.

The trap density is in the model card too, if you know where to look. Read through Hung's model, the flicker noise parameter NOIA of sky130's nfet is 2.5e42 J^{-1}m^{-3}, which is 4×10^{17} cm^{-3}eV^{-1}; the pfet's 1.5e42 is 2.4×10^{17}. The next part of the chapter shows why the noise parameter is a trap density.

The count is the point. A minimum-size transistor has, on average, about one active trap per seven decades of time constant. Most such devices are quiet, and the unlucky one has a large telegraph signal. Noise on small devices is not a number but a distribution, and it varies from device to device like mismatch does. Large devices have hundreds of traps per decade, and their steps blur into something smooth.

9.16 Which transistor in GR06?

9.16.1 How GR06 works

- A PTAT current, divided by 100 in two mirrors, charges a 53.8 fF MIM capacitor
- A comparator trips when the ramp reaches $V_{ref} = V_{DD}/3$, from three poly resistors
- Pulse width $t = CV_{ref}/I \approx 7.1$ µs, so $I \approx 4.5$ nA
- Every analog transistor is a 3.2/0.94 µm unit, deep in weak inversion; the reset switch is 1.92/0.22

$$\frac{\Delta t}{t} = \frac{\Delta V_{ref} + V_{os}}{V_{ref}} - \frac{\Delta I}{I} = 1760 \text{ ppm}$$

Before blaming a transistor, write down what the pulse width depends on. A step of 0.53 K at 304 K is a fractional step of 1760 ppm in the pulse width. That is 1.06 mV at the comparator input, or 0.18 % of the charging current.

The reference is a resistive divider from the supply. Anything that steps the supply by 0.18 % - 3 mV - would look exactly like a trap. That is the first suspect to rule out.

9.16.2 Is it the supply?

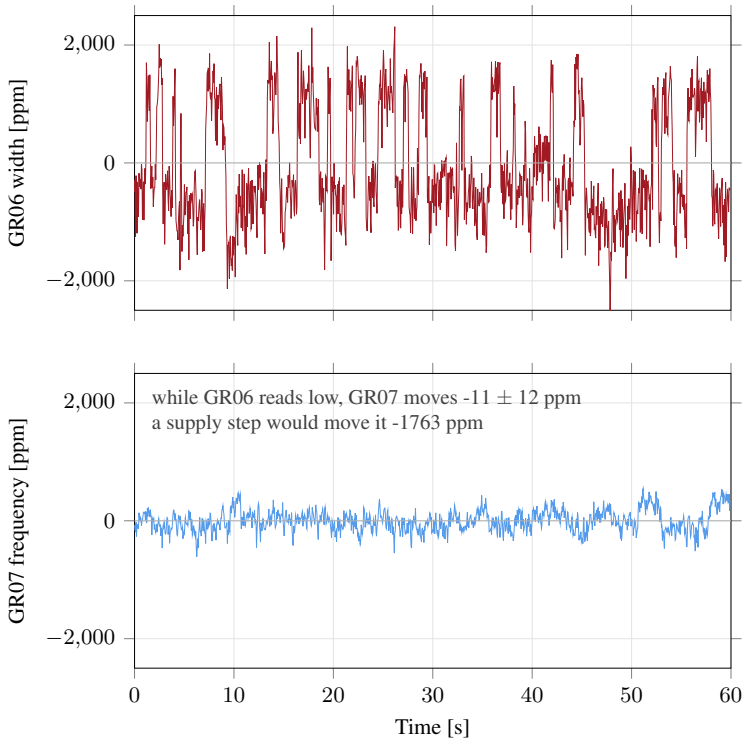
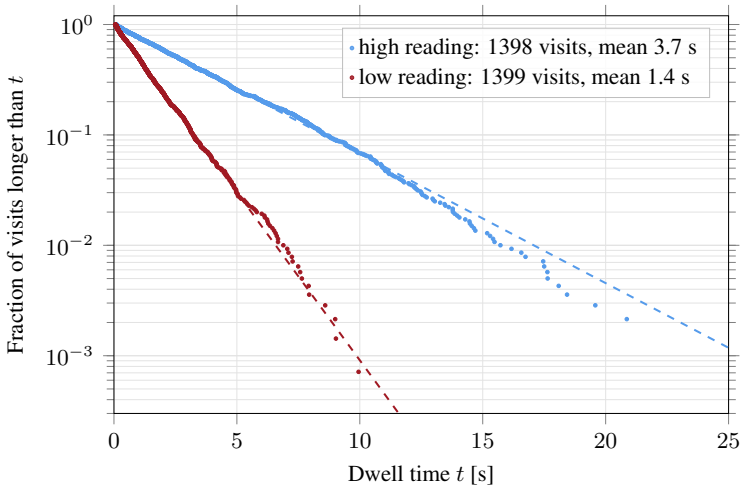


Figure 8: One minute of GR06 and GR07, recorded together on the same die, as fractional deviations on the same scale. While GR06 reads low, GR07 moves −11 ± 12 *ppm. A supply step would have moved it* −1763 *ppm*

GR07, the other sensor on the die, compares against $V_{DD}/4$ from its own divider and reports a frequency instead of a width. A fractional supply step moves both by the same amount, with opposite signs. So record both at once, label each moment by GR06's state, and ask how far GR07 moves between the two labels.

It moves −11 ± 12 ppm. The error bar comes from the data: slide GR06's labels 20 s to 400 s against GR07, where nothing real can line up, and the same calculation scatters by 12 ppm. A supply step would have given −1763 ppm. The supply is less than about 2 % of the step.

9.16.3 Memoryless



$$\tau_0 = \left(\frac{1}{3.7\ \text{s}} + \frac{1}{1.4\ \text{s}}\right)^{-1} = 1.0\ \text{s}, \quad f_0 = \frac{1}{2\pi\tau_0} = 0.15\ \text{Hz}$$

Figure 9: Two hours of GR06 at about 31 °C, cut into 2797 visits to the two levels. The fraction of visits that lasted longer than t falls as a straight line on a log axis, over three decades - an exponential. The dashed lines are exponentials with the measured means, not fits

A trap has no memory: the chance it lets go in the next millisecond does not depend on how long it has held on. That makes the dwell times exponential, and a log survival plot turns an exponential into a straight line. Both levels give one, down to one visit in a thousand.

The two means put the Lorentzian corner at 0.15 Hz, the corner of Figure 7.

9.16.4 The suspects

Device	W/L [µm]	ΔV_T, one electron	$\Delta t/t$
Comparator input pair	4 x 3.2/0.94	1.6 µV	3 ppm
Comparator load	3.2/0.94	6.4 µV	11 ppm
Reset switch, off leakage	1.92/0.22	46 µV	< 1 ppm
PTAT core	3.2/0.94	6.4 µV	30 to 90 ppm
Ramp current mirror unit	3.2/0.94	6.4 µV	190 ppm
Measured			**1760 ppm**

C_{ox} = 8.2 fF/µm² (pfet), 8.3 fF/µm² (nfet), from t_{oxe} in the sky130 model cards; nV_T = 34 mV

Every number in the last column is one electron's $\Delta V_T = q/(C_{ox}WL)$, carried through to the pulse width.

The comparator is the least likely culprit. Its input pair is the largest device in the sensor, so one electron shifts its offset by 1.6 µV, and the step needs 1.06 mV. Its load refers to the input at about the same weight as the pair, because both run at the same current in weak inversion.

The reset switch is small, but it is off during the ramp. It leaks about 2 pA against 4.5 nA, and one electron changes that leakage by a tenth of a percent.

The current path is the best candidate. Each of the three single units that carry the ramp current - the PTAT output and the two hundredfold-division mirrors - converts a threshold step straight into current, $\Delta I/I = \Delta V_T/nV_T$, because in weak inversion $g_m/I_D = 1/nV_T$. That is 190 ppm, nine times short of the measurement.

Nine times is not a contradiction. The ΔV_T formula assumes the charge is spread evenly over the gate. In weak inversion it is not: the random dopants leave a lumpy barrier, the current crowds into the low spots, and a trap above one of them removes more than its share. The trapped charge also scatters the electrons that pass it [14]. Both push the step up, and in weak inversion they can push it up a lot. A 3 µm² pfet has about four active traps per decade of time constant, so one near a second is no surprise.

9.16.5 How to tell

Source	Step $\Delta t/t$ vs T	Step $\Delta t/t$ vs V_{DD}
Comparator offset	constant	$\propto 1/V_{DD}$
Supply	ruled out	ruled out
Weak inversion current	$\propto 1/T$ or faster	constant
Measured	falls, 2500 to 1750 ppm from 5 °C to 45 °C	not measured

A threshold step is a fixed voltage. At the comparator it is compared with V_{ref}, so its fractional effect does not depend on temperature and scales as $1/V_{DD}$. In a weak-inversion mirror it is compared with nkT/q, so its fractional effect falls with temperature and does not care about the supply.

The chamber sweep of the measurement chapter has the temperature column. The fractional step falls from about 2500 ppm at 5 °C to 1750 ppm at 45 °C - faster than $1/T$, and the points scatter, but it falls. That points at the current path. Above 50 °C the two levels merge into the noise and the step estimate is biased low, so those points are left out.

The supply column would settle it. Run the board's core supply at a few voltages around 1.8 V: a step that stays at 1760 ppm is in the current path, a step that scales as $1/V_{DD}$ is at the comparator.

9.17 Many traps

9.17.1 McWhorter: uniform in depth is $1/f$

$$\tau = \tau_0 e^{x/\lambda} \Rightarrow dx = \lambda \frac{d\tau}{\tau}$$

$$S(f) \propto \int_{\tau_1}^{\tau_2} \frac{\tau}{1+\omega^2\tau^2} \frac{d\tau}{\tau} = \frac{1}{\omega} \left[\arctan(\omega\tau)\right]_{\tau_1}^{\tau_2} \approx \frac{\pi}{2\omega}$$

for $1/\tau_2 \ll \omega \ll 1/\tau_1$

Spread the traps evenly in depth. Each one contributes a Lorentzian, $\tau/(1+\omega^2\tau^2)$. Since τ is exponential in depth, an even spread in depth is an even spread in $\ln\tau$: as many traps per decade of time constant at a microsecond as at a second. Add up the Lorentzians, and the sum is exactly $1/f$ between the slowest and the fastest trap.

This is McWhorter's model [9]. It needs nothing exotic - just traps scattered through the oxide, and the exponential that Schrödinger put in the tail of the channel electron.

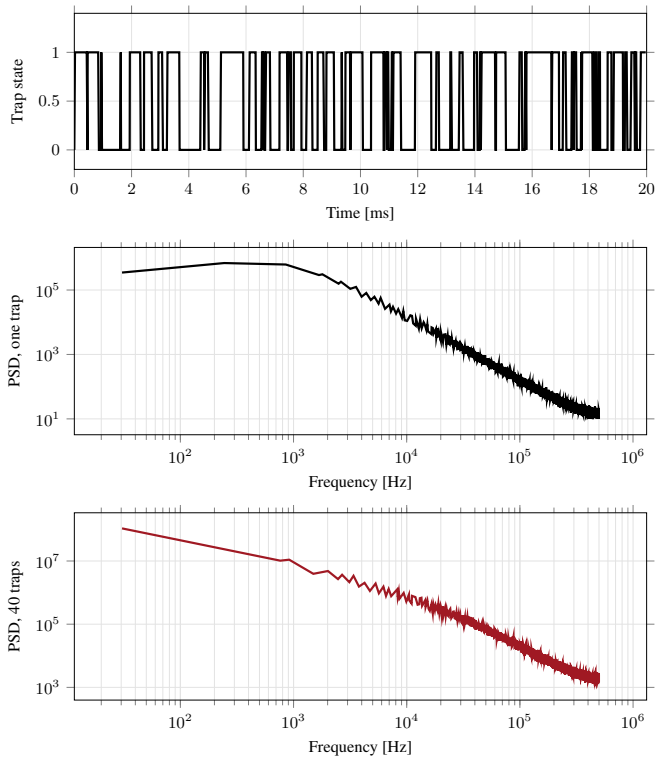


Figure 10: A single trap gives a two-level random telegraph signal (top) whose spectrum is a Lorentzian, flat then falling as $1/f^2$ (middle). Forty traps with time constants spread over three decades sum to a straight $1/f$, measured slope -1.02 between 100 Hz and 10 kHz (bottom)

The same argument, done numerically. Nothing was fitted: forty Lorentzians spread evenly in log τ, added. Above the corner of the fastest trap the slope steepens back towards $1/f^2$, because there are no faster traps left. A real flicker noise spectrum ends the same way, for the same reason.

9.17.2 Back to the design equation

$$S_{V_G}(f) = \frac{q^2\, kT\, \lambda\, N_t}{WL\, C_{ox}^2\, f}$$

- λ: Schrödinger, the tunnelling length
- kT: Fermi-Dirac, $\int f(1-f)\,dE = kT$
- N_t: the process, how broken the oxide is
- $1/WL$: more traps, each smaller - $WL \times (1/WL)^2$

Do the McWhorter integral with every factor kept and you get the number-fluctuation model of flicker noise. It is the $K_f/(WLC_{ox}f)$ of the MOSFET chapter's noise equations, with K_f opened up.

Every factor is a piece of this chapter. The λ is the 0.08 nm tunnelling length. The kT is not thermal noise sneaking in: it is the width of the energy window where traps switch, since only traps near E_F are noisy and the variance of each, $f_T(1-f_T)$, integrates over energy to exactly kT. The N_t is the process. And the $1/WL$ is the competition between more traps and smaller steps: a bigger device has more traps, proportional to WL, but each step in V_T shrinks as $1/WL$, and power goes as the step squared.

That is why flicker noise falls with gate area and nothing else in the hand formula, and why a PMOS, with its holes a little further from the interface, is often quieter.

9.17.3 The model card knows

BSIM4, flicker noise model 1, as used by ngspice:

$$S_{I_D} \propto \frac{q^2\, kT\, \mu\, I_D}{10^{10}\, f^{EF} A_{bulk} C_{oxe} L^2}\,[\text{NOIA} \ln(\ldots) + \ldots]$$

sky130 01v8	t_{oxe}	NOIA [$J^{-1}m^{-3}$]	N_t [$cm^{-3}eV^{-1}$]	EF
nfet	4.148 nm	2.5e42	4.0×10^{17}	0.84
pfet	4.23 nm	1.5e42	2.4×10^{17}	1.0

The simulator's flicker noise model is this chapter, compiled. BSIM4's unified model is Hung's [14], and Hung's is McWhorter's

number fluctuation plus the mobility fluctuation of a charged trap. The constant 10^{10} in the denominator is not a fudge: it is the tunnelling attenuation coefficient, $1/\lambda$ in metres, so the simulator assumes $\lambda = 0.1$ nm, close to our 0.08 nm. The kT is the Fermi-Dirac window. And NOIA, the parameter the foundry fits to measured noise, is the trap density N_t in $\mathrm{J}^{-1}\mathrm{m}^{-3}$ - divide by the electron charge to get eV.

EF is the exponent of f. The pfet's is 1, McWhorter's ideal. The nfet's is 0.84: its spectrum falls slower than $1/f$, so it has more fast traps than an even spread would give. In the tunnelling picture the traps are not uniform in depth - they crowd towards the interface, where the time constants are short.

So the next time a simulation hands you a flicker noise corner, you know what it is made of: a trap density, a tunnelling length, a Fermi window and an area.

9.18 Summary

The one-page version of this chapter:

- Noise is random: describe the amplitude by its distribution (usually Gaussian) and the power by its spectral density
- Integrate the density over the band you keep to get the power the signal must beat; uncorrelated sources add in power
- Friis: with gain up front, only the first stage's noise matters - spend your current there
- Thermal noise is Planck, $4kTR$ below 6 THz; in weak inversion it is shot noise, $2qI$
- Flicker noise comes from traps: a bound state in the oxide, reached by tunnelling, a decade of time constant per 0.18 nm of depth
- Capture also moves the lattice, which makes it Arrhenius - 226 meV on GR06
- One trap is a telegraph signal: two Gaussians, exponential dwell times, a Lorentzian - all three measured on GR06
- Many traps, even in depth, are $1/f$; small transistors have few, so their noise varies from device to device
- The simulator's flicker model is the same physics: sky130's NOIA is a trap density, 4e17 per cm^3 per eV

9.19 Would you like to know more?

The classic on noise in solid state devices, still the reference [2]

Friis' original, short and still the clearest statement of why the first stage decides the noise figure [3]

Chapter-length treatments of circuit noise, either of which will do [4] [1]

The random walk that thermal noise is made of, from the source [6], and thermal noise from Planck, in the original [7]

The review of individual traps in MOSFETs, and the place to start [11]

The first clear observation of single traps in silicon inversion layers [12]

The tunnelling model of $1/f$ noise [9], and the spectrum of a two-state signal [13]

Multiphonon capture, the reason traps are thermally activated [10]

Number and mobility fluctuation combined into one flicker noise model [14]

Dirac's golden rule [8]. The equation all of this physics comes from is the QED Lagrangian in the refresher chapter

A curiosity: the trapped electron has a spin, and with microwaves at its resonance it can be read out through the drain current of one transistor [15]

[1] D. Johns and K. Martin, *Analog integrated circuit design*. John Wiley & Sons, Inc., 1997.

[2] A. V. D. Ziel, *Noise in solid state devices and circuits*. John Wiley & Sons Inc, 1986.

[3] H. T. Friis, "Noise figures of radio receivers," *Proceedings of the IRE*, vol. 32, no. 7, pp. 419–422, 1944, doi: 10.1109/JRPROC.1944.232049.

[4] B. Razavi, *Design of analog CMOS integrated circuits*. McGraw-Hill, 2001.

[5] R. M. Gray and L. D. Davisson, *An introduction to statistical signal processing*. Cambridge University Press, 2004.

[6] A. Einstein, "Method for the determinination of the statistical values of observations concerning quantities subject to irregular fluctuations," *ieee_m_assp*, vol. October, 1987.

[7] H. Nyquist, "Thermal agitation of electric charge in conductors," *Physical Review*, vol. 32, no. 1, pp. 110–113, 1928, doi: 10.1103/PhysRev.32.110.

[8] P. A. M. Dirac, "The quantum theory of the emission and absorption of radiation," *Proceedings of the Royal Society of London A*, vol. 114, no. 767, pp. 243–265, 1927, doi: 10.1098/rspa.1927.0039.

[9] A. L. McWhorter, "1/f noise and germanium surface properties," in *Semiconductor surface physics*, R. H. Kingston, Ed. University of Pennsylvania Press, 1957, pp. 207–228.

[10] C. H. Henry and D. V. Lang, "Nonradiative capture and recombination by multiphonon emission in GaAs and GaP," *Physical Review B*, vol. 15, no. 2, pp. 989–1016, 1977, doi: 10.1103/PhysRevB.15.989.

[11] M. J. Kirton and M. J. Uren, "Noise in solid-state microstructures: A new perspective on individual defects, interface states and low-frequency (1/f) noise," *Advances in Physics*, vol. 38, no. 4, pp. 367–468, 1989, doi: 10.1080/00018738900101122.

[12] K. S. Ralls *et al.*, "Discrete resistance switching in submicrometer silicon inversion layers: Individual interface traps and low-frequency (1/f?) noise," *Physical Review Letters*, vol. 52, no. 3, pp. 228–231, 1984, doi: 10.1103/PhysRevLett.52.228.

[13] S. Machlup, "Noise in semiconductors: Spectrum of a two-parameter random signal," *Journal of Applied Physics*, vol. 25, no. 3, pp. 341–343, 1954, doi: 10.1063/1.1721637.

[14] K. K. Hung, P. K. Ko, C. Hu, and Y. C. Cheng, "A unified model for the flicker noise in metal-oxide-semiconductor field-effect transistors," *IEEE Transactions on Electron Devices*, vol. 37, no. 3, pp. 654–665, 1990, doi: 10.1109/16.47770.

[15] M. Xiao, I. Martin, E. Yablonovitch, and H. W. Jiang, "Electrical detection of the spin resonance of a single electron in a silicon field-effect transistor," *Nature*, vol. 430, pp. 435–439, 2004, doi: 10.1038/nature02727.

Circuits 10

Keywords: Sizing, Bias Point, gm/ID, Current Mirrors, Cascode, CS, CD, CG, Differential Pair, OTA

10.1 Circuits 157
10.2 Transistor size and bias point 157
10.3 Transistor sizing strategy 161
10.3.1 Option 1: Full freedom 161
10.3.2 Option 2: Constrained 161
10.4 My circuit does not work, why???????????????? 162
10.5 Current Mirrors . . 163
10.5.1 Normal current mirror 164
10.5.2 Source degenerated current mirror 165
10.5.3 Cascoded current mirror 167
10.5.4 Active cascodes . . . 168
10.6 Amplifiers 169
10.6.1 Source follower . . . 169
10.6.2 Why use a source follower? 172
10.6.3 Common gate 173
10.6.4 Common source . . 177
10.6.5 Why common source? 178
10.6.6 Differential pair . . . 179
10.6.7 Diff pairs are cool . 180
10.7 Summary 180
10.8 Would you like to know more? 181

10.1 Circuits

Most of the circuit design on integrated circuits uses MOSFET transistors. This document provides a short refresh of the most common circuits and their properties.

But first, how should transistors be used, and sized.

10.2 Transistor size and bias point

In Figure 1 and Figure 2 we can see two transistors. One with short gate length (approximately 1F2 = 1.2 x minimum gate length) and one with longer gate length (approximately 5F0 = 5.0 x minimum gate length). The width of both transistors is sufficient to have 2 contacts on the drain/source (2C).

These transistors come from a standard transistor library I've made, and that can be found at JNW_ATR_SKY130A.

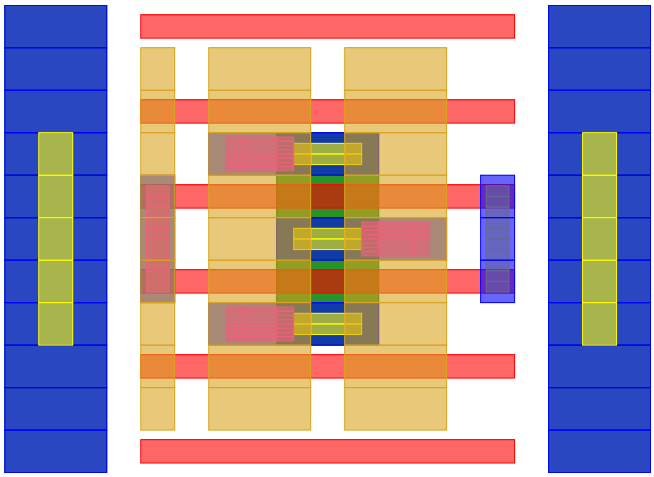


Figure 1: JNWATR_PCH_2C1F2 transistor

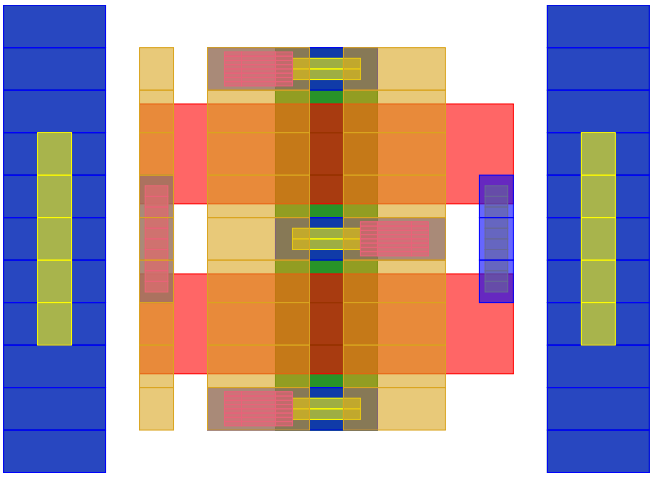


Figure 2: JNWATR_PCH_2C5F0 transistor

In all our circuits we will need to pick the right size, but how should we do it?

By now you should know that MOSFETs have different regions of operation. Related to the V_{GS} we talk about weak inversion, moderate inversion and strong inversion. These names correspond directly to the density of charge carriers in the thin inversion layer underneath the oxide in the channel. See MOSFETs lecture for details.

In Figure 3 we can see how the log of the current changes behavior at low V_{GS} versus at high V_{GS}. As such, when we pick the transistor size we should be conscious of which region we operate the transistor in. The regions (weak, moderate, strong) have different behaviours.

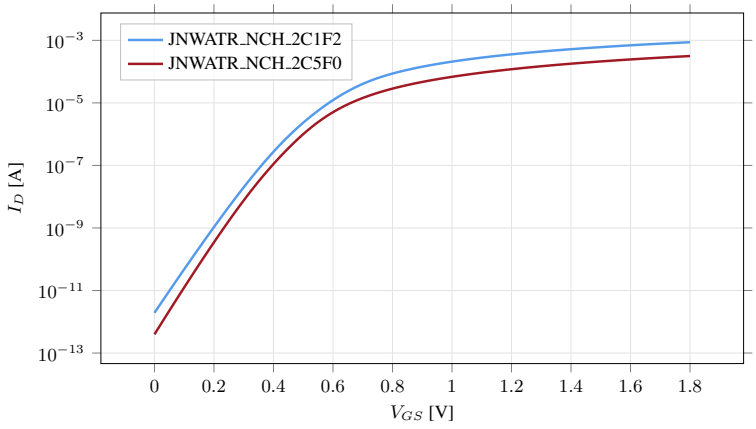


Figure 3: Log of drain current versus gate/source voltage

One of the behavior differences is the "bang-for-the-buck". For a circuit we may target a specific number for the transconductance (g_m). In Figure 4 we can see how the g_m/I_D changes as we sweep the V_{GS}

In weak inversion, we have a large bang for the buck

$$g_m/I_D \approx 1/n/V_T \approx 1/1.5/26\,\text{mV} \approx 25$$

While in strong inversion

$$\frac{g_m}{I_D} = \frac{2}{V_{eff}}$$

where the effective overdrive is

$$V_{eff} = V_{GS} - V_{TH}$$

In moderate inversion, the g_m/I_D is somewhere between the two.

Assume we need a transconductance of 1 mS. If we have a $g_m/I_D = 10$, then we'd need at least 100 μA in the transistor. If we had a $g_m/I_D = 15$, then we'd only need 66 μA in the transistor.

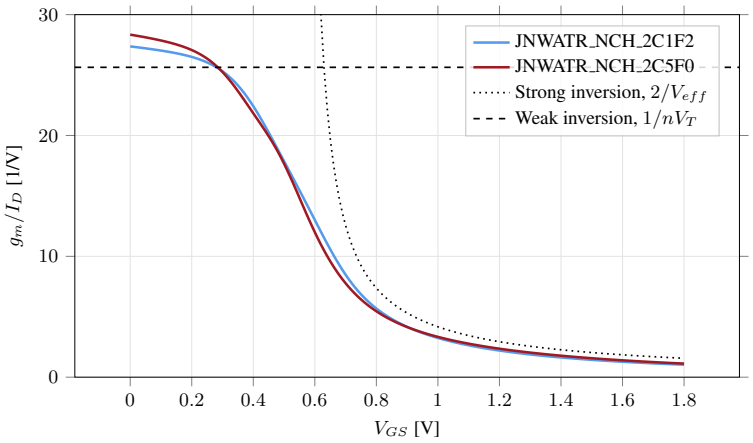


Figure 4: gm/Id (Y-axis) versus Gate voltage (X-axis)

When the g_m/I_D choice is made, then there are some things that have already been determined. One is the necessary drain/source voltage for the transistor to operate in "saturation" or "linear" region.

For most circuits we want the transistor to operate in "saturation" region, as such, we must provide a certain drain source voltage.

In Figure 5 you can see how the V_{dsat} of the transistor changes as the g_m/I_D changes.

Notice that the two transistors have different V_{dsat}. The shorter transistor needs less voltage across drain/source to operate in saturation.

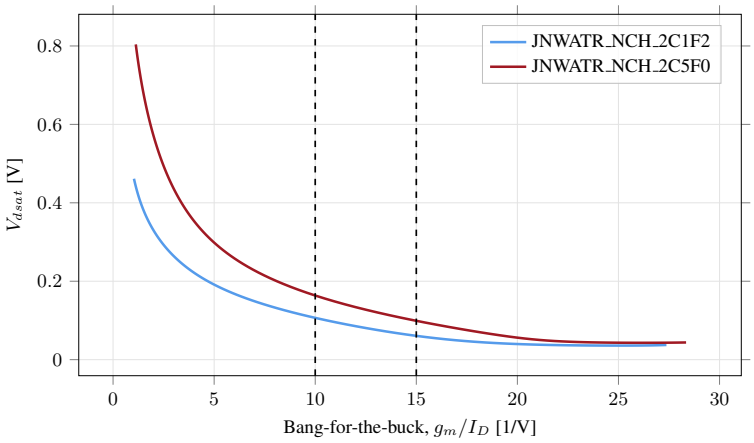


Figure 5: Vdsat versus gm/Id

The choice of g_m/I_D also determine what the gate source voltage is. It's actually rare we control the gate voltage directly to set the bias point of the transistor. It's more common to bias transistors with a current, and let the V_{GS} be whatever the V_{GS} needs to be.

In Figure 6 we can see that at g_m/I_D of 15 we have a lower V_{GS} than g_m/I_D of 10.

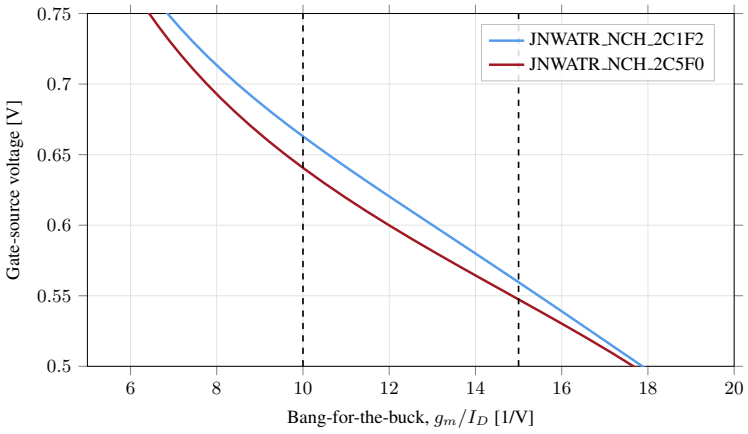


Figure 6: Vg versus gm/Id

The choice between the two transistors come down to "What intrinsic gain do I need in my transistor?".

For current mirrors we really don't want the output current to change with V_{DS} so we want a small conductance (g_{ds}), or a large intrinsic gain (g_m/g_{ds}).

In Figure 7 we can see how the intrinsic gain of the two transistors is different. For the 1F2 we can also see there is some funky behavior above gm/id of 20, I don't know why, but I suspect something funky in the model (non-physical).

For a larger intrinsic gain we should pick a longer transistor.

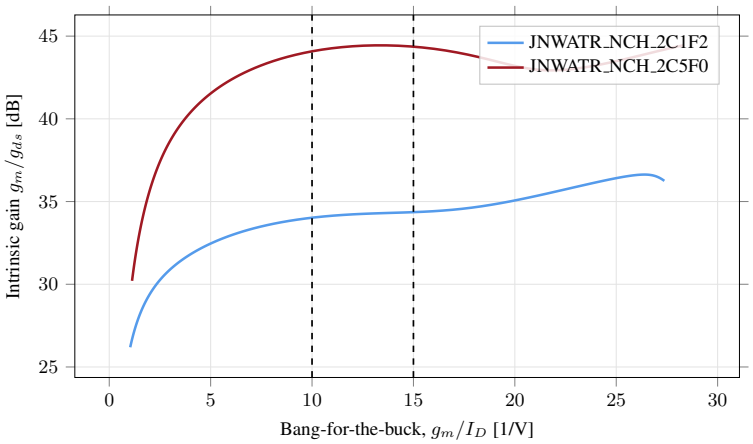


Figure 7: gm/gds versus gm/Id

10.3 Transistor sizing strategy

10.3.1 Option 1: Full freedom

If you really want to dig deep, and get your transistor size exactly correct, then method that makes most sense to me, is to use the inversion-coefficient method, described in

- Nanoscale MOSFET Modeling: Part 1 [1]
- Nanoscale MOSFET Modeling: Part 2 [2].

This is similar to a gm/Id strategy, but we're rather looking directly at the inversion level

The inversion coefficient tells us how strongly inverted the MOSFET channel (inversion layer) is. A number below 0.1 is weak inversion, between 0.1 and 10 is moderate inversion. A number above 10 is strong inversion.

There are also some blog posts worth looking at Inversion Coefficient Based Circuit Design and My Circuit Design Methodology.

10.3.2 Option 2: Constrained

"Full transistor size freedom" "is similar to giving a loaded gun to a kid and say"don't shoot yourself". It's a bad idea!

If you're inexperienced with transistor sizing I would highly recommend to pick a few transistors, and compute the parameters (V_{GS}, V_{dsat}, ...) for the transistor, and then use a limited set.

That's exactly what I've done i

JNW_ATR_SKY130A

I would encourage you to only use transistors from that library in your design. I always do that when I do design, in any technology.

10.4 My circuit does not work, why??????????????

The reason is usually that the transistors are not operating in the correct region. So either the V_{GS} is causing problems or the V_{DS} is not high enough.

In Figure 8 we can see how the V_{GS} of transistors change with corner. It's usually highest for slow-slow and low temperature, and the lowest for fast-fast and high temperature. But even that statement is obviously not always correct. For a gm/Id of 6 we can see that it's the low temperature that has the lowest V_{GS}.

If we observe the equation for the current in strong inversion

$$I_D = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}(V_{GS} - V_{TH})^2$$

we can see that the current decreases if the V_{TH} increases, and we can see that current increases if the mobility (μ_n) increases. The threshold voltage increases at low temperature. The mobility increases at low temperature. At a gm/Id of a bit more than 8 we can see that from Figure 8 the two effects cancel each other. While for lower gm/Id the mobility becomes dominant, and lowers the V_{GS}.

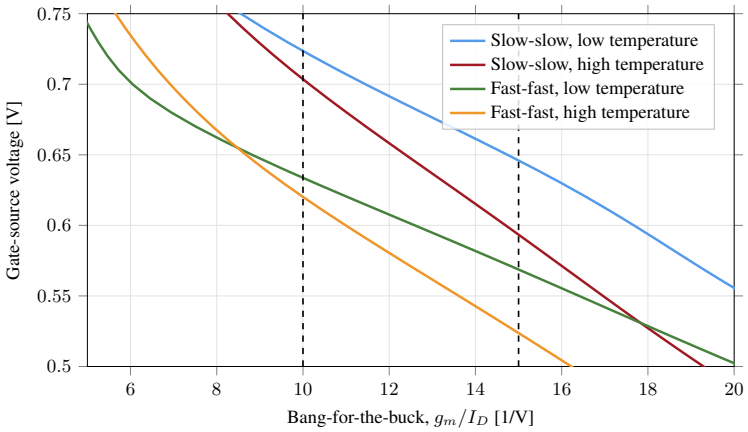


Figure 8: Gate-source voltage as a function of corner

The drain-source voltage does not change that much with corner, but it does change. The deeper into strong inversion we go, the larger the change.

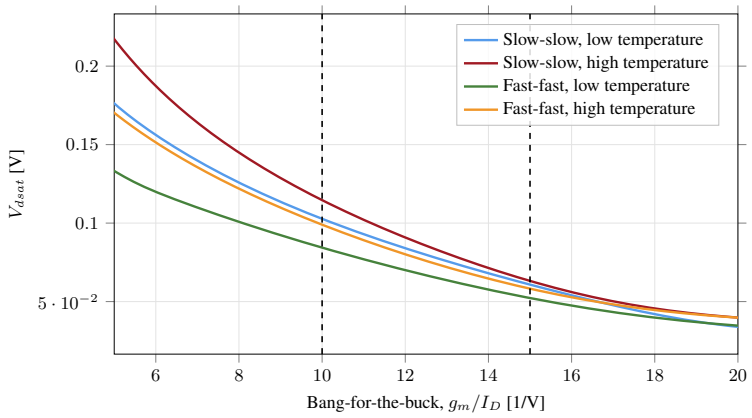


Figure 9: Drain Saturation Voltage as a function of corner

As such, for any transistor design, it's necessary to see exactly what voltages and currents flow in your transistors.

Most tools can show the operating point in the schematic, which makes it easier to figure out what the voltages and currents are. In Figure 10 we can see an example from TB_JNW_TEMP_OP

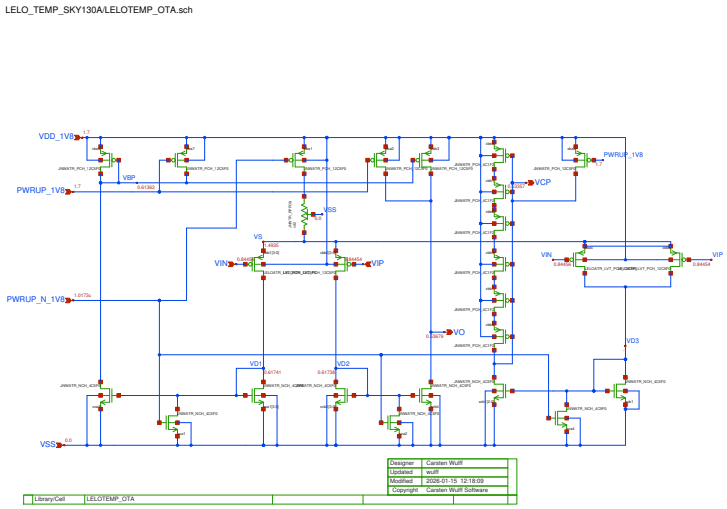


Figure 10: Example of operating point annotation

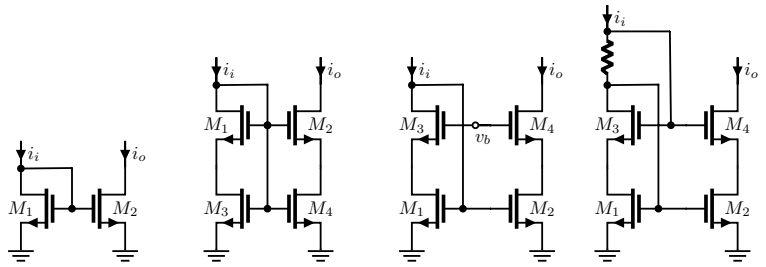
10.5 Current Mirrors

MOSFETs need a current for the transistor to be biased in the correct operating region. The current must come from somewhere, we'll look at bias generators later. Usually there is a central bias circuit that provides a single, good, reference current.

On an IC, however, there will be many circuits, and they all need a bias current (usually). As such, we need a circuit to copy a current.

In the figure below you can see a selection of current mirrors. They all do the same thing. Try to ensure that i_i and i_o are the same current.

Which one we choose is usually determined by what we mean by $i_i = i_o$. Do we mean "within ± 10 %", or "within ± 2 %".



a) "normal" b) Self Cascode c) Cascode d) Lazy Cascode

Figure 11: Example of current mirrors

10.5.1 Normal current mirror

The normal current mirror consists of a diode connected transistor (M_1) and a common source transistor M_2.

If we assume infinite output resistance of the MOSFETs, then the drain voltage does not affect the current.

If the two transistors are the same size, threshold voltage, mobility, etc, and they have the same gate-source voltage, then the current in them must be the same.

A current pushed into M_1 will cause the V_{GS1} to rise, and at some point, find a stable point where the current pushed in is equal to the current in M_1

M_2 will see the same $V_{GS1} = V_{GS2}$ so the current will be the same, provided the voltage at i_o is sufficient to pinch-off the channel of M_2, or the $V_{DS2} \approx 3kT/q$ if the transistor is in weak-inversion.

The output resistance of a normal current mirror is simply the r_{ds} of the output transistor.

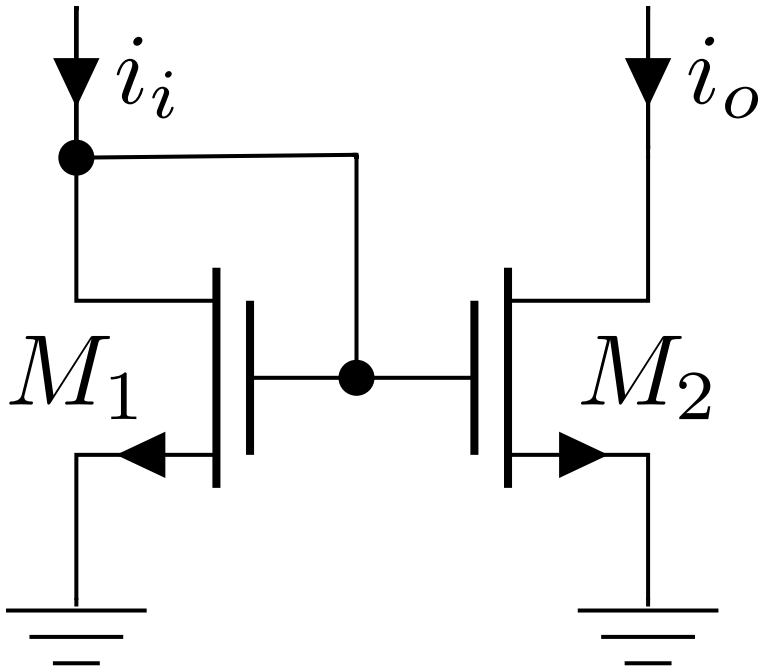


Figure 12: Normal current mirror

10.5.2 Source degenerated current mirror

In most modern technologies, and if we care about the output current accuracy, then a normal current mirror cannot give us a sufficient independence of the drain/source voltage.

When we use more advanced current mirrors, it's almost always to increase the output resistance, and make the current mirror more like a current source.

In Figure 13 we can see a current mirror with resistors on source.

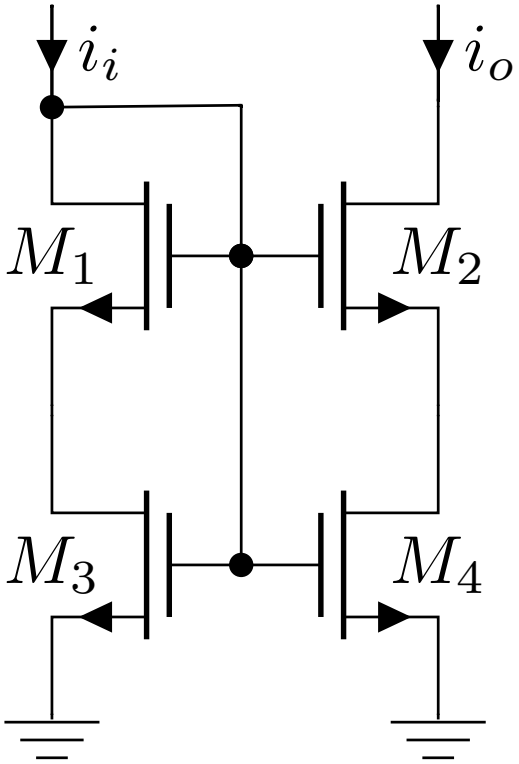


Figure 13: Source degenerated current mirror

Observe the small signal model in Figure 14. If we now apply a test current i_x we can compute what the output resistance is ($r_{out} = v_x/i_x$)

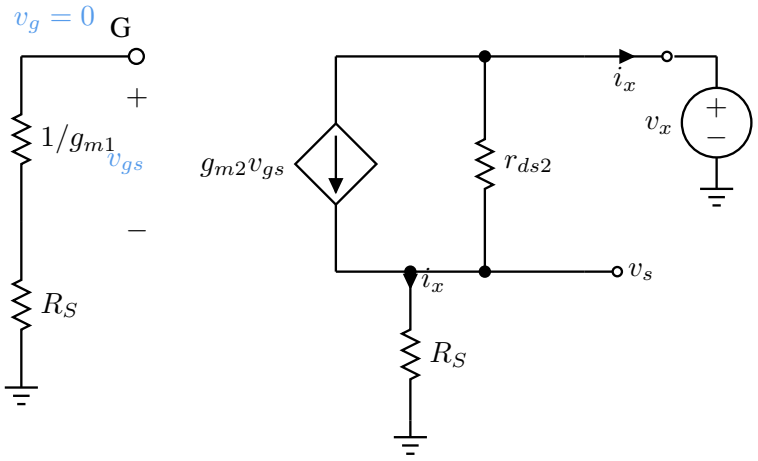


Figure 14: Source degenerated current mirror small signal model

$$v_{gs} = -v_s$$

,

$$v_s = i_x R_s$$

,

$$r_{out} = \frac{v_x}{i_x}$$

$$i_x = g_{m2} v_{gs} + \frac{v_x - v_s}{r_{ds2}}$$

$$i_x = -i_x g_{m2} R_s + \frac{v_x - i_x R_s}{r_{ds2}}$$

$$v_x = i_x \left[r_{ds2} + R_s (g_{m2} r_{ds2} + 1) \right]$$

Rearranging

$$r_{out} = r_{ds2}[1 + R_s(g_{m2} + g_{ds2})] \approx r_{ds2}[1 + g_{m2} R_s]$$

10.5.3 Cascoded current mirror

To further increase the output resistance, we can move to a cascoded current mirror as shown in Figure 15.

Now we need a separate voltage to bias the cascode to ensure that the source node of M_3 keeps the drain node of M_1 above V_{DSAT}.

If we bias the cascode correctly, then even if the voltage at i_o changes, then the source of M_4 does not really change, and the current in M_2 stays the same.

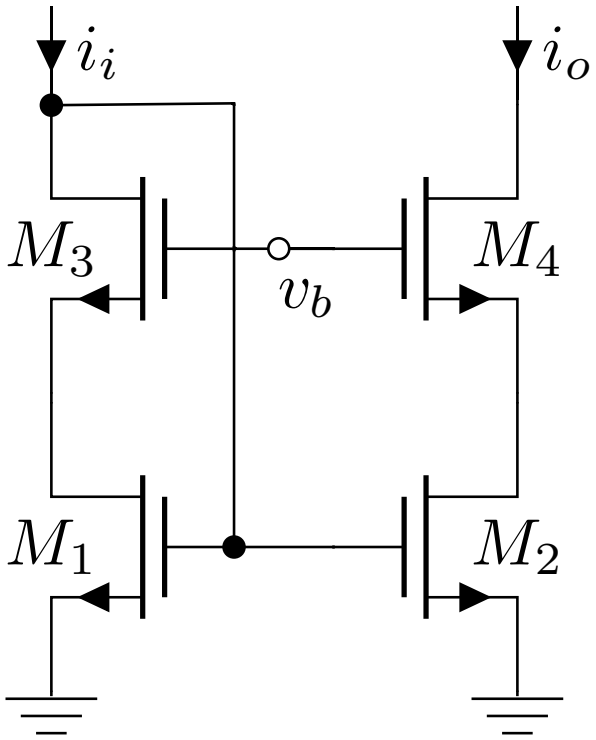


Figure 15: Cascoded current mirror

From source degeneration (ignoring bulk effect)

$$r_{out} = r_{ds4}[1 + R_s(g_{m4} + g_{ds4})]$$

$$R_S = r_{ds2}$$

$$r_{out} = r_{ds4}[1 + r_{ds2}(g_{m4} + g_{ds4})]$$

$$r_{out} \approx r_{ds2}(r_{ds4}g_{m4})$$

10.5.4 Active cascodes

If we need even higher output resistance, then we can add a operational transconductance amplifier (OTA) to the gate of the cascode to further increase the g_m of the cascode.

In OTAs it's common to increase the open loop gain by increasing the output resistance. The output stage of an OTA is usually current mirrors, as a result, one can end up with active cascodes in the OTA that is used in the active cascode of the current mirror. All horribly complicated, but sometimes necessary.

Watch the polarity, because it is easy to draw this circuit as an oscillator. The node being regulated is the drain of the mirror device, and it must go to the amplifier's *inverting* input, with V_B on the non-inverting one. Then a rise on that node drives the cascode gate down, the cascode conducts less, and the node comes back down. Swap the two inputs and every part of that sentence reverses: the node rises, the gate rises, the cascode pulls harder, and the node rises further. Same devices, same A, and the loop runs to a rail instead of regulating.

$$r_{out} \approx r_{ds2}(Ar_{ds4}g_{m4})$$

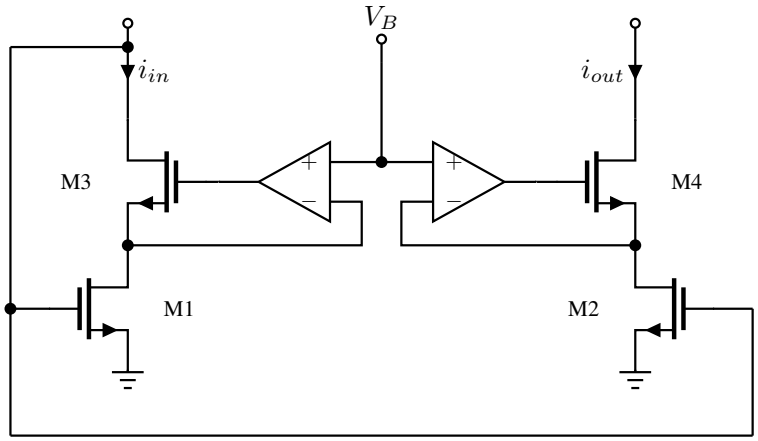


Figure 16: Active cascode current mirror

10.6 Amplifiers

There are usually three amplifiers that we consider when we talk about single transistors. Common Source, Common Gate and Source Follower.

For two transistors there are a few more possibilities. I'd highly recommend Fifty Nifty Variations of Two-Transistor Circuits: A tribute to the versatility of MOSFETs [3]

10.6.1 Source follower

The source follower can be seen in Figure 17. The input signal is at the gate, and the output at the source. The transistor and its bias current source form a level shifter: the output follows the input, one V_{GS} lower. The properties we care about are

Input resistance

$$\approx \infty$$

Gain

$$A = \frac{v_o}{v_i}$$

Output resistance

$$r_{out}$$

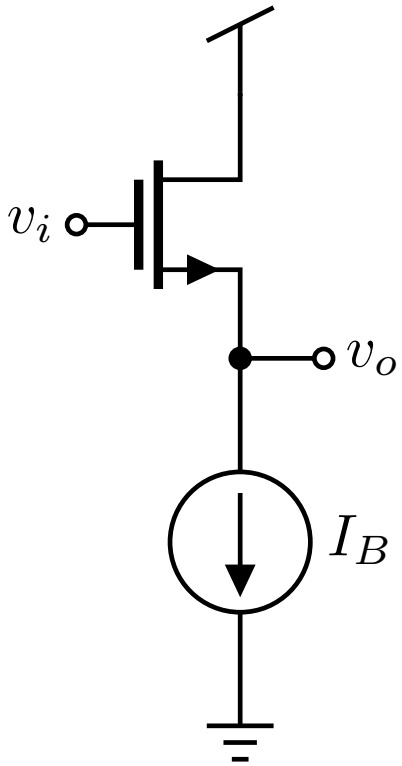


Figure 17: Source follower

10.6.1.1 Small signal gain

To find the gain, replace the transistor with its small signal model, as shown in Figure 18. The drain is at the supply, and the supply does not move for small signals, so the drain rail is at AC ground. The source rail is the output. Between the two hang the transconductance, the bulk transconductance, and r_{ds}.

Sum the currents into the output node, and set the output current to zero - nothing is loading us:

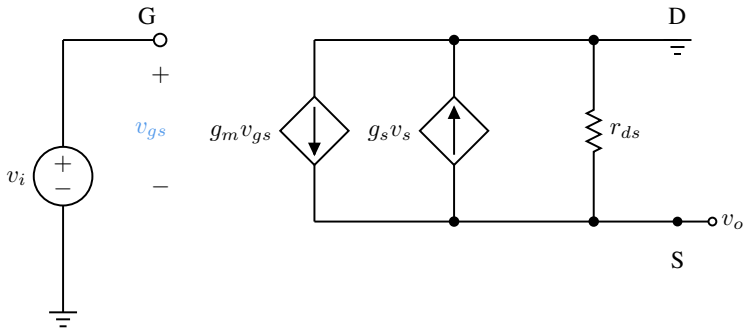


Figure 18: Source follower small signal model

$$i_o = v_o(g_{ds} + g_s) - g_m v_i + v_o g_m$$

$$i_o = 0$$

$$g_m v_i = v_o(g_m + g_s + g_{ds})$$

$$A = \frac{v_o}{v_i} = \frac{g_m}{g_m + g_{ds} + g_s}$$

Gain is less than 1

The transconductance appears both on top and in the bottom of the fraction, so the gain approaches, but never reaches, one. The bulk transconductance g_s is the main thief: with bulk tied to ground the gain of an NMOS follower is typically 0.8 to 0.9.

10.6.1.2 Output resistance

The same equation gives the output resistance. Zero the input, push a current into the output, and see what voltage builds up:

$$i_o = v_o(g_{ds} + g_s) - g_m v_i + v_o g_m$$

$$v_i = 0$$

$$i_o = v_o(g_{ds} + g_s + g_m)$$

$$r_{out} = \frac{v_o}{i_o} = \frac{1}{g_m + g_{ds} + g_s}$$

$$r_{out} \approx \frac{1}{g_m}$$

A $1/g_m$ output resistance is the point of the whole circuit: it is the cheapest low impedance money can buy in CMOS. Whatever fragile, high impedance node you have, a follower turns it into a node that can drive real capacitance.

10.6.2 Why use a source follower?

A concrete example makes the point. In an image sensor pixel a photodiode collects charge on a tiny sense node - say 1 fF. Assume the light gives us 100 electrons.

Look at which way round the diode sits, because it is the opposite of what the symbol suggests. RST charges the sense node to 3.0 V, so the cathode has to be the node and the junction is *reverse* biased. Turn the diode around and it is forward biased at 3 V, clamping the node near 0.6 V, and there is no pixel left. The photocurrent therefore runs against the triangle, down and out of the sense node, and that is where the minus sign in ΔV comes from: light does not charge the node, it discharges it.

What the diode symbol is standing in for is an n region in a p well, depleted from end to end, and the electrons the light frees collect in it. TX is the transfer gate. Because the n region is fully depleted there is no potential minimum left to strand carriers in, so opening TX moves the whole electron packet across in one go and leaves nothing behind for the next frame - which is the reason a modern pixel is built this way rather than reading the diode's own node directly. The 1 fF the packet lands on is the floating diffusion, and ΔV is measured there, not at the diode.

In Figure 19 the sense node drives the gate of a source follower. The gate draws no charge, so all 100 electrons stay on the 1 fF, and the signal is a healthy 16 mV, which the follower copies onto the 1 pF bus below.

Assume 100 electrons

$$\Delta V = Q/C = -1.6 \times 10^{-19} \times 100/(1 \times 10^{-15}) = -16\ \text{mV}$$

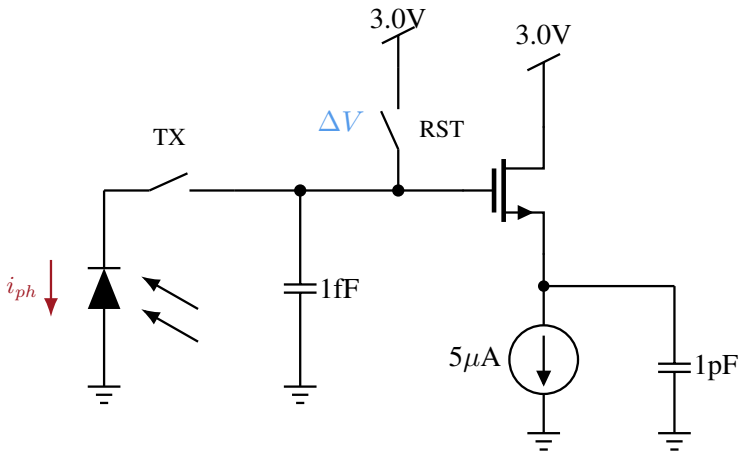


Figure 19: Sense node buffered by a source follower

$$\Delta V = Q/C = -1.6 \times 10^{-19} \times 100/(1 \times 10^{-12}) = -16 \text{ uV}$$

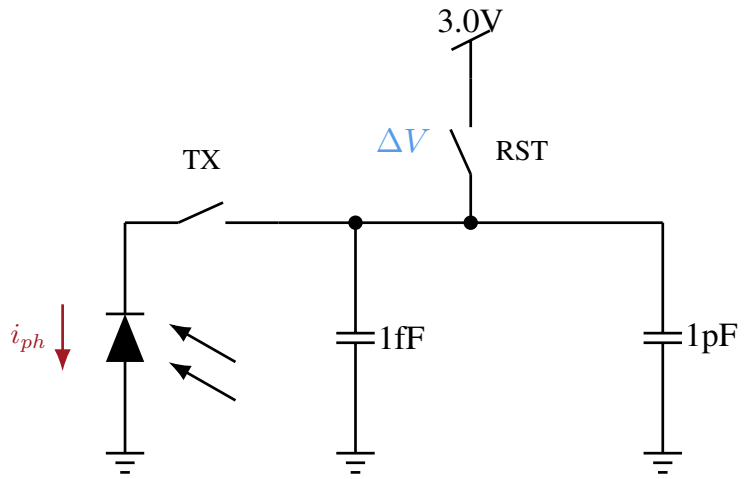


Figure 20: The same sense node connected straight to the bus

In Figure 20 the follower is gone and the sense node must charge the 1 pF bus directly. The same 100 electrons now land on a thousand times the capacitance, and the signal shrinks to 16 uV - buried in the noise. The follower does not amplify anything, and still it makes the difference between a signal and no signal.

Another example of a source follower can be found in A 92.5mW 205MS/s 10b Pipeline IF ADC Implemented in 1.2V/3.3V 0.13um CMOS [4]

10.6.3 Common gate

In the common gate stage, Figure 21, the roles rotate: the gate is held at a bias voltage, the signal goes in at the source, and the output is taken at the drain. Nothing amplifies the voltage between input and gate except the transistor's own V_{GS} - the input current simply reappears at the drain. That makes the common gate a current buffer: a low resistance input, a high resistance output, and a current gain of one.

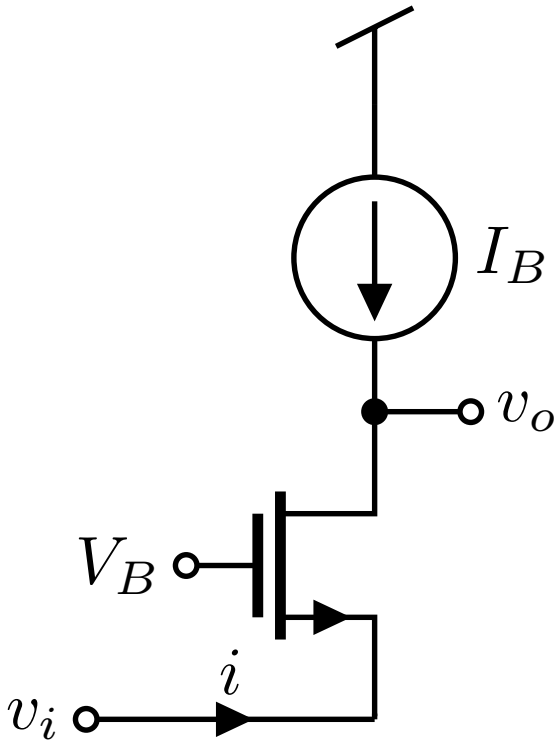


Figure 21: Common gate stage

Start with the input resistance, using the small signal model in Figure 22. The gate is grounded, so wiggling the source by v_x makes $v_{gs} = -v_x$: the transconductance pulls current out of the test source, and the input looks like a resistance of roughly $1/g_m$.

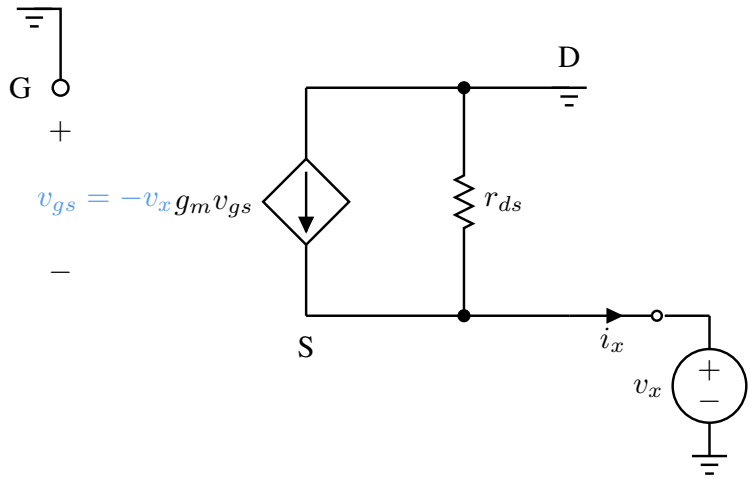


Figure 22: Common gate input resistance

10.6.3.1 Input resistance

$$i = g_m v + g_{ds} v$$

$$r_{in} = \frac{1}{g_m + g_{ds}} \approx \frac{1}{g_m}$$

However, we've ignored load resistance.

$$r_{in} \approx \frac{1}{g_m}\left(1 + \frac{R_L}{r_{ds}}\right)$$

The last line is worth a pause: the friendly $1/g_m$ input resistance only holds if the drain sees a low load resistance. Load the drain with a current source, and the input resistance grows by the ratio R_L/r_{ds} - the cascode chapter of every textbook in one line.

10.6.3.2 Output resistance

For the output resistance, ground the source: then $v_{gs} = 0$, the transconductance is dead, and the test source at the drain sees only r_{ds}, as drawn in Figure 23.

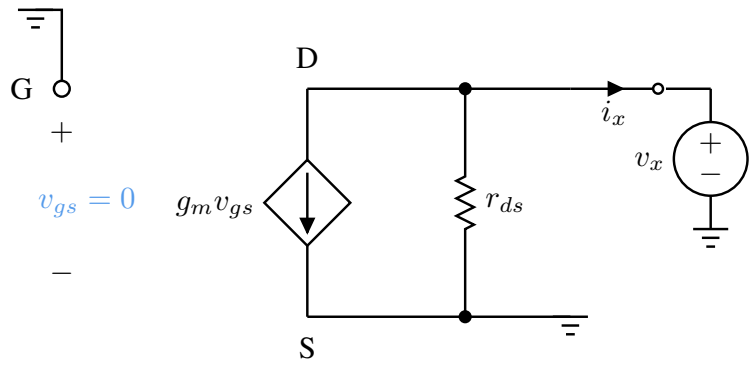


Figure 23: Common gate output resistance

$$r_{out} = r_{ds}$$

10.6.3.3 Small signal gain

For the voltage gain, drive the source and leave the drain open, Figure 24. The same g_m that made the input resistance low now pushes its current into r_{ds}:

$$i_o = -g_m v_i + \frac{v_o - v_i}{r_{ds}}$$

$$i_o = 0$$

$$0 = -g_m v_i r_{ds} + v_o - v_i$$

$$v_i(1 + g_m r_{ds}) = v_o$$

$$\frac{v_o}{v_i} = 1 + g_m r_{ds}$$

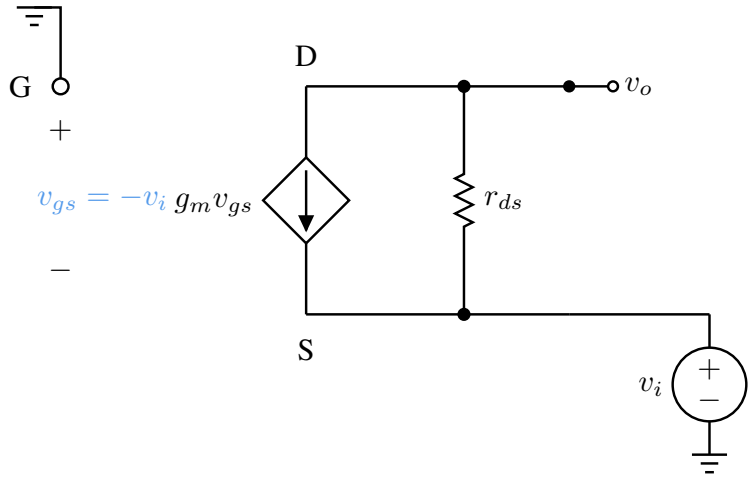


Figure 24: Common gate small signal gain

The gain is the intrinsic gain plus one, and it is not inverting: the common gate has the same gain magnitude as the common source, it just refuses to flip the sign.

The full expression, with nothing ignored, is uglier:

We've ignored bulk effect (

$$g_s$$

), source resistance (

$$R_S$$

) and load resistance (

$$R_L$$

)

$$A = \frac{(g_m + g_s + g_{ds})(R_L \parallel r_{ds})}{1 + R_S \left(\frac{g_m + g_s + g_{ds}}{1 + R_L / r_{ds}} \right)}$$

If

$$R_L >> r_{ds}$$

,

$$R_S = 0$$

and

$$g_s = 0$$

$$A = \frac{(g_m + g_{ds})r_{ds}}{1} = 1 + g_m r_{ds}$$

Check the simplification against the special case above, and note what the full expression adds: the source resistance R_S divides the gain down, and the bulk transconductance helps for once - in a common gate the bulk effect adds to g_m instead of stealing from it.

10.6.4 Common source

The common source stage, Figure 25, is the amplifier: input at the gate, source grounded, output at the drain. It is also a circuit we have already met - the output half of every current mirror is a common source transistor - so the input and output resistances come for free:

$$r_{in} \approx \infty$$

$$r_{out} = r_{ds}$$

, it's same circuit as the output of a current mirror

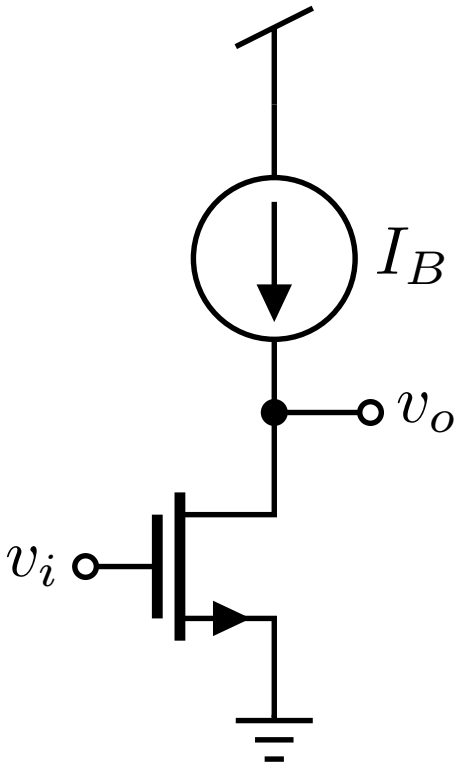


Figure 25: Common source stage

10.6.4.1 Small signal gain

The small signal model, Figure 26, has only two elements, and the sum of their currents at the output node gives the gain in three lines:

$$i_o = g_m v_i + \frac{v_o}{r_{ds}}$$

$$i_o = 0$$

$$-g_m v_i = \frac{v_o}{r_{ds}}$$

$$\frac{v_o}{v_i} = -g_m r_{ds}$$

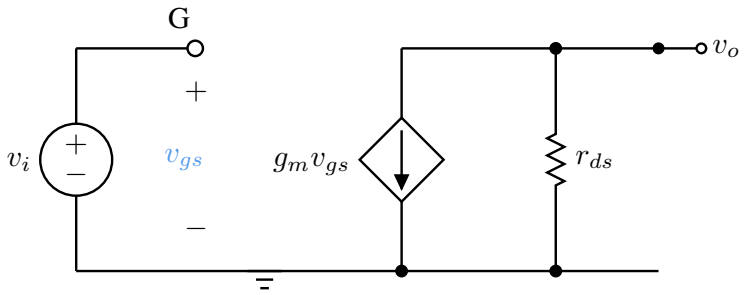


Figure 26: Common source small signal model

The gain is minus the intrinsic gain of the transistor - the most gain a single device can give, which is why the common source is the default gain stage in every amplifier.

10.6.5 Why common source?

The signal from an antenna is microvolts, Figure 27. Before anything can demodulate, filter or digitize it, it must be made bigger - and the only thing that makes voltages bigger is gain. The matching network hands the microvolt signal through an AC coupling capacitor to the gate, a high value resistor from a current mirror sets the bias point without loading the signal, and the common source transistor multiplies the voltage by $-g_m R$. First gain, then everything else.

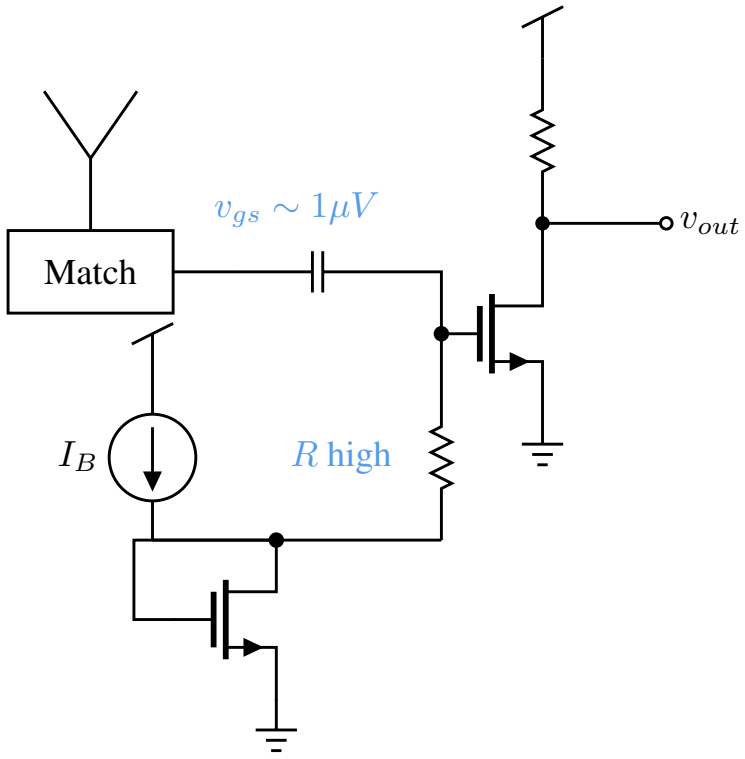


Figure 27: A low noise amplifier is a common source stage

10.6.6 Differential pair

Single-ended amplifiers share a weakness: they cannot tell the signal from the ground bounce. The differential pair, Figure 28, fixes that by amplifying only the *difference* between two inputs. Two matched transistors share one tail current: with equal inputs the current splits evenly and nothing happens at the outputs. Apply a difference, and current steers from one branch to the other - the tail current is a see-saw, and the differential input tilts it.

Per side, the numbers are the common source numbers:

Input resistance

$$r_{in} \approx \infty$$

Gain

$$A = g_m r_{ds}$$

Output resistance

$$r_{out} = r_{ds}$$

Best analyzed with T model of transistor (see CJM page 31)

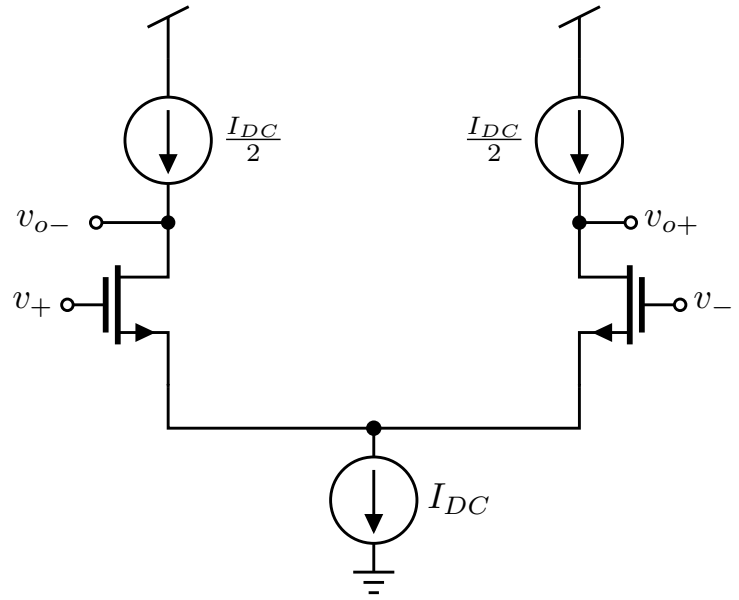


Figure 28: Differential pair

10.6.7 Diff pairs are cool

Two properties make the pair the default input of every OTA. First, whatever is common to both inputs - supply bounce, substrate noise, bias drift - is rejected, because it does not tilt the see-saw. Second, sign is free:

Can choose between

$$v_o = g_m r_{ds} v_i$$

and

$$v_o = -g_m r_{ds} v_i$$

by flipping input (or output) connections

10.7 Summary

The single transistor gives us three views of the same device:

Stage	r_{in}	r_{out}	Gain
Common source	∞	r_{ds}	$-g_m r_{ds}$
Common gate	$1/g_m$	r_{ds}	$1 + g_m r_{ds}$
Source follower	∞	$1/g_m$	≈ 1

Common source when you need gain, common gate when you need to move a current without disturbing it, source follower when you need to drive something. Current mirrors bias them all, and the differential pair wraps two common source stages around one tail current so only the difference matters.

Put a differential pair on top of a current mirror and you have built the five transistor OTA - which is where the OTA chapter picks up.

10.8 Would you like to know more?

Where the gm/ID way of thinking comes from, in the authors' own words [1], and the modelling that backs it [2]

Settling and slewing in feedback amplifiers, which is where this chapter's transient budget comes from [4]

The same ground covered by a different voice, with open tools throughout [3]

[1] C. Enz, F. Chicco, and A. Pezzotta, "Nanoscale MOSFET modeling: Part 1: The simplified EKV model for the design of low-power analog circuits," *IEEE Solid-State Circuits Magazine*, vol. 9, no. 3, pp. 26–35, 2017, doi: 10.1109/MSSC.2017.2712318.

[2] C. Enz, F. Chicco, and A. Pezzotta, "Nanoscale MOSFET modeling: Part 2: Using the inversion coefficient as the primary design parameter," *IEEE Solid-State Circuits Magazine*, vol. 9, no. 4, pp. 73–81, 2017, doi: 10.1109/MSSC.2017.2745838.

[3] H. Pretl and M. Eberlein, "Fifty nifty variations of two-transistor circuits: A tribute to the versatility of MOSFETs," *IEEE Solid-State Circuits Magazine*, vol. 13, no. 3, pp. 38–46, 2021, doi: 10.1109/MSSC.2021.3088968.

[4] B. Hernes *et al.*, "A 92.5mW 205MS/s 10b pipeline IF ADC implemented in 1.2V/3.3V 0.13μm CMOS," in *2007 IEEE international solid-state circuits conference. Digest of technical papers*, 2007, pp. 462–615, doi: 10.1109/ISSCC.2007.373494.

11 OTAs

Keywords: OTA, Headroom, Five Transistor, Current Mirror OTA, Two-Stage, Miller Compensation, Folded Cascode, Inverter-Based, Nauta, CMFB, Bias, Op Amp, Dynamic Amplifiers, Verification

11.1 OTAs 183
11.2 The headroom budget 183
11.3 Five transistor OTA 184
11.4 Current mirror OTA 185
11.5 Two stage (Miller) OTA 186
11.6 Folded cascode . . . 187
11.7 Inverter based OTAs 188
11.8 Bulk driven input . 189
11.9 Fully differential . . 190
11.9.1 Common mode feedback 190
11.9.2 Sensing the common mode 191
11.9.3 Switched capacitor CMFB 192
11.10 Bias circuits 192
11.11 From OTA to op amp 193
11.11.1 A complete OTA, sized 194
11.12 Dynamic amplifiers 195
11.13 Choosing 196
11.14 Verifying the OTA . 196
11.15 Summary 197
11.16 Would you like to know more? 198

11.1 OTAs

An operational transconductance amplifier (OTA) takes a differential voltage in and pushes a current out. Load it with a capacitor and it integrates; wrap feedback around it and it becomes whatever the feedback network says - a switched capacitor integrator, a filter, an ADC residue amplifier, a regulator error amplifier. Almost every analog system in this book has an OTA somewhere inside it.

The idea is old: the name and the first commercial part arrived in 1969, when Wheatley and Wittlinger argued that the OTA obsoletes the op amp [1], and the OTA-based filter tradition that grew from it is summarized in Geiger and Sanchez-Sinencio's tutorial [2].

This chapter walks through the OTA topologies that still make sense in nanoscale CMOS, where the supply is around 0.8 V. That last constraint is the important one: half the classic topologies in the textbooks were invented for 5 V, and do not survive the trip down.

11.2 The headroom budget

Start with the arithmetic that kills topologies. At $V_{DD} = 0.8$ V, with a threshold voltage around 0.4 V:

$$V_{DD} = 0.8 \text{ V}, V_t \approx 0.4 \text{ V}$$

One gate-source voltage

$$V_{GS} \approx 0.5 \text{ V}$$

One saturated current source, one cascode

$$V_{DSAT} \approx 0.1 \text{ V each}$$

Stack of two gate-source voltages? Dead.

Telescopic cascode with swing? Dead.

A gate-source voltage plus a tail current source plus a load already sums to about 0.7 V, so the five transistor OTA barely fits. Add a cascode on both sides and the output can still move a little. Stack two gate-source voltages - a telescopic cascode with wide swing, a folded mirror with source degeneration - and there is nothing left.

The consequences run through the whole chapter: we get gain from long transistors and from more stages, not from stacking; we bias at moderate or weak inversion, where V_{GS} is low and g_m/I_D is high; and every volt of output swing has to be argued for.

11.3 Five transistor OTA

The five transistor OTA in Figure 1 is the differential pair from the circuits chapter with its loads folded into a current mirror: the mirror takes the left branch current, flips it over, and slams it into the right branch, so the full differential current reaches the single ended output.

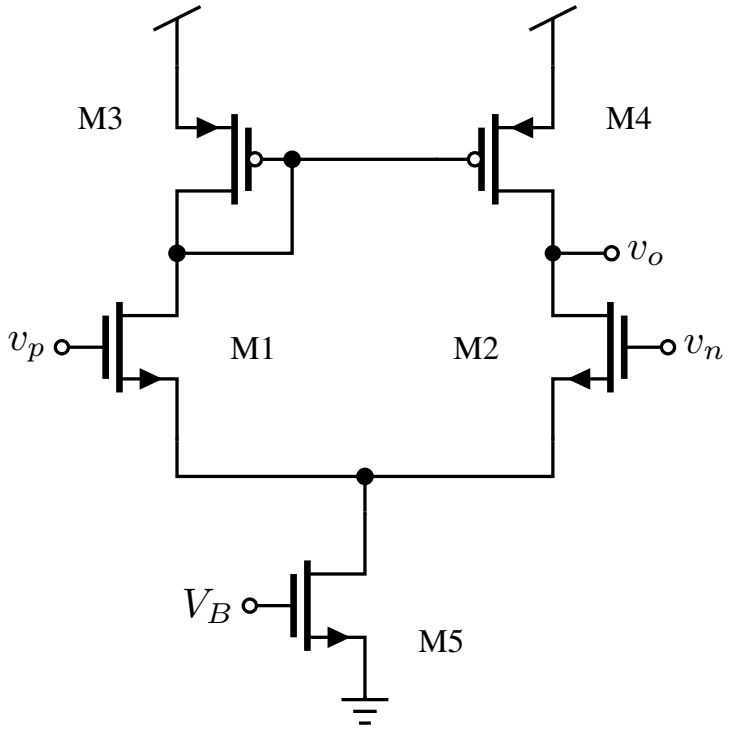


Figure 1: Five transistor OTA

$$A = g_{m1}(r_{ds2} \parallel r_{ds4})$$

$$\omega_{ugf} = \frac{g_{m1}}{C_L}$$

Output swing:

$$V_{DD} - 2V_{DSAT}$$

The gain is one intrinsic gain - 20 to 40 dB in a nanoscale process - and the unity gain frequency is set by the input pair transconductance over the load capacitance. The swing is generous: only a V_{DSAT} lost at each rail.

For a buffer, a modest filter, or a bias loop, this is the correct answer, and reaching for anything fancier is vanity. When it is not enough, it is for one of two reasons: not enough gain, or not enough drive - and the two failures point to two different upgrades.

11.4 Current mirror OTA

If the problem is drive - a big load capacitor and a tail current that cannot slew it - the current mirror OTA in Figure 2 helps. Both branch currents are mirrored outwards with a gain K, and the output branch can source and sink K times the tail current.

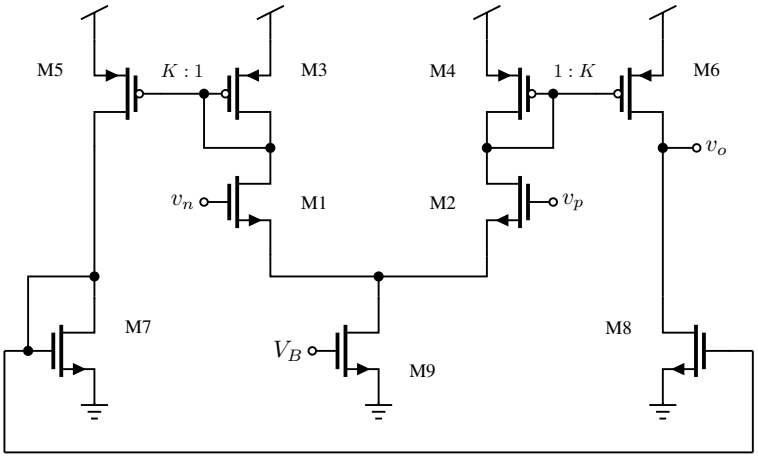


Figure 2: Current mirror OTA with mirror ratio K

$$A = Kg_{m1}(r_{ds6} \parallel r_{ds8})$$

$$\omega_{ugf} = \frac{Kg_{m1}}{C_L}$$

Slew rate:

$$\pm KI_{tail}/C_L$$

The price is the extra mirror pole - the diode connected loads and their mirror partners add a pole at roughly g_{m3}/C_{mirror}, which eats phase margin as K grows - and the noise and offset of four more transistors. K of 2 to 5 is typical. All transistors sit one V_{GS} or one V_{DSAT} from a rail, so the topology is fully at home at 0.8 V, which is why it is everywhere in low voltage design.

11.5 Two stage (Miller) OTA

If the problem is gain, add a stage. The two stage OTA in Figure 3 puts a common source stage after the five transistor OTA: two intrinsic gains multiplied, and the output stage swings to within one V_{DSAT} of each rail - the best swing any OTA can offer, which matters when the supply is 0.8 V and every millivolt of signal range counts.

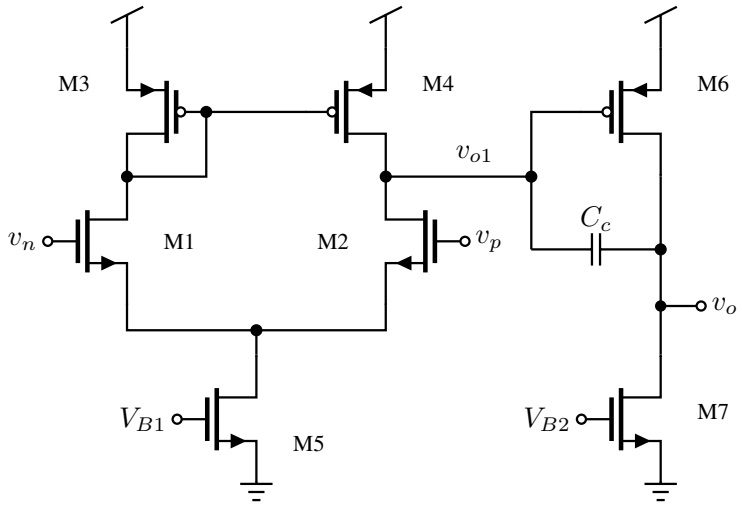


Figure 3: Two stage OTA with Miller compensation

$$A = g_{m1}(r_{ds2} \parallel r_{ds4}) \times g_{m6}(r_{ds6} \parallel r_{ds7})$$

$$\omega_{ugf} = \frac{g_{m1}}{C_c}$$

Pole splitting: dominant pole down, output pole out to

$$\approx \frac{g_{m6}}{C_L}$$

Two stages means two poles, and two poles in a feedback loop must be pushed apart. That is the Miller capacitor C_c's job: it is C_{gd} multiplied by the second stage gain, on purpose - the Miller effect from the MOSFET chapter, hired instead of feared. The input pole

drops, the output pole rises to about g_{m6}/C_L, and the amplifier crosses unity at g_{m1}/C_c with the second pole safely beyond.

The famous flaw: C_c also feeds the input signal forward past the second stage, creating a right half plane zero at g_{m6}/C_c that steals phase. The standard fix is a resistor in series with C_c, which moves the zero to infinity - or on top of the second pole, if you are feeling precise.

The subtler flaw is the supply rejection. Above the dominant pole, C_c effectively shorts the output to the first stage output, which turns M6 into a diode connected device seen from the loop - and M6's source sits on V_{DD}. High frequency supply ripple therefore walks through the output stage with close to unity gain, exactly where the loop gain is already too small to fight it. Of the topologies in this chapter, the two stage Miller OTA is the one that most needs a quiet analog supply - or a regulator from the voltage regulation chapter - between it and the digital switching noise.

11.6 Folded cascode

When one stage must deliver more gain than a five transistor OTA - a switched capacitor integrator that settles to 10 bits, say - the folded cascode in Figure 4 buys a factor $g_m r_{ds}$ more. The input pair current folds outwards into cascoded branches: cascodes multiply output resistance, and folding means the input pair and the cascodes do not stack on top of each other in the same headroom.

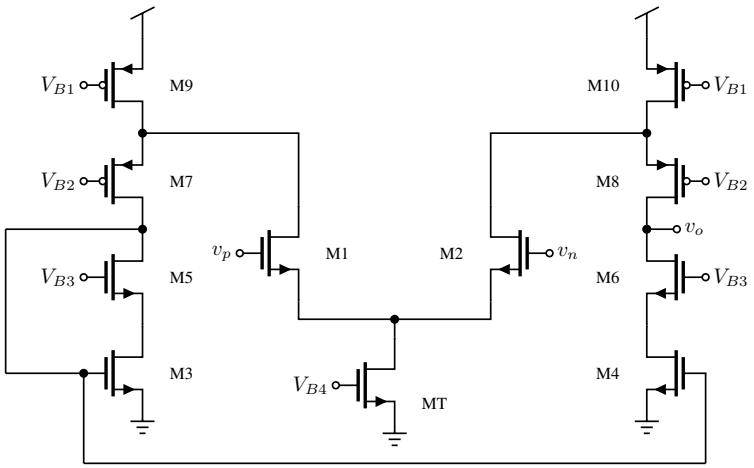


Figure 4: Folded cascode OTA

$$A \approx g_{m1}\left(g_{m8} r_{ds8}(r_{ds10} \parallel r_{ds2}) \parallel g_{m6} r_{ds6} r_{ds4}\right)$$

$$\omega_{ugf} = \frac{g_{m1}}{C_L}$$

Output swing:

$$V_{DD} - 4V_{DSAT}$$

At 0.8 V the folded cascode is possible, but on a diet: four V_{DSAT} of about 0.1 V each leaves 0.4 V of output swing, and the bias voltages V_{B1} to V_{B3} must be generated carefully (wide swing mirrors, see CJM) or the diet fails. It is also the last stop: the telescopic cascode, which stacks the input pair under the cascodes, needs the swing and the input common mode to share headroom that is not there at 0.8 V.

The load capacitor is the compensation - no Miller capacitor needed - so for switched capacitor circuits, where the load is a known sampling capacitor, this topology is the default single stage answer.

11.7 Inverter based OTAs

Every topology so far spends half its current on transistors that do not amplify. The inverter, Figure 5, does not: the PMOS and NMOS share the same current, both amplify the same input, and the transconductances add.

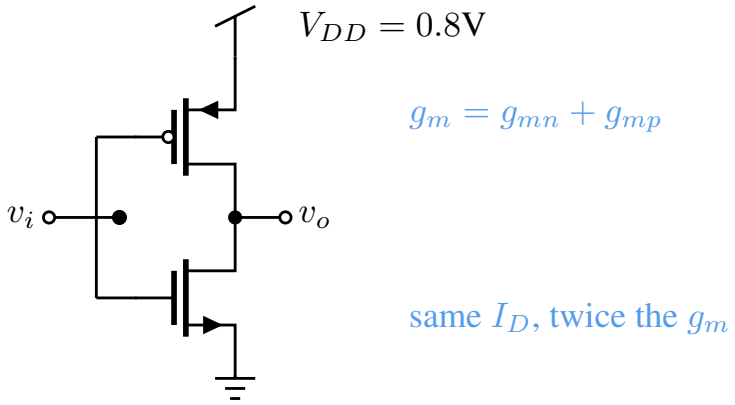


Figure 5: The inverter as a transconductor

The inverter gives $g_{mn} + g_{mp}$ for one branch current - twice the transconductance per microampere of anything above, which at 0.8 V, in weak inversion, is exactly the currency that matters. The catch: an inverter has no tail current source, so its current and its common mode are set by V_{DD} and the process. It rejects nothing - supply noise and corners go straight through.

Nauta showed in 1992 how to make a real OTA out of nothing but inverters, Figure 6 [3]. Two inverters amplify differentially. On the outputs, a cross coupled pair fights common mode motion and a shorted inverter on each output loads it resistively - together they hold the output common mode without a single tail source or CMFB loop, and the differential gain survives.

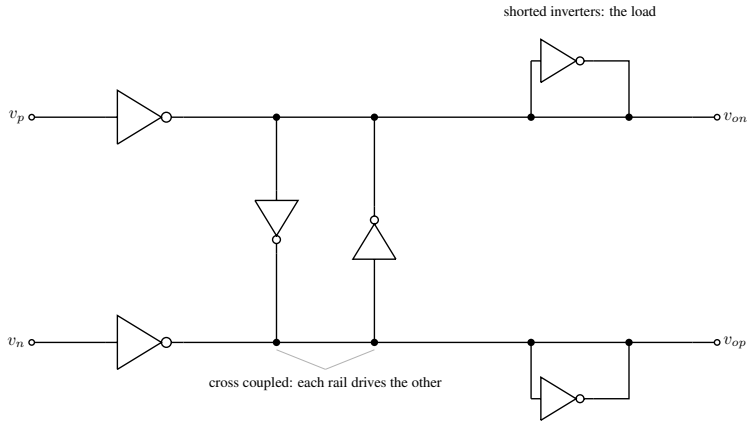


Figure 6: Nauta's inverter based transconductor

Because every device is part of an inverter, the whole OTA works at any supply where an inverter has gain - which in weak inversion means a few hundred millivolts. Inverter based OTAs run the switched capacitor filters and sigma-delta modulators of most sub-1V papers of the last decade. The supply sensitivity does not disappear, though: it moves into the bias, so the supply of an inverter based OTA is usually a regulated one - see the voltage regulation chapter.

11.8 Bulk driven input

One more low voltage trick from the MOSFET chapter: the bulk is a second gate with $g_s \approx 0.2 g_m$, and it works with the source at the rail. Feed the signal into the bulk of a transistor whose V_{GS} is tied fully on, and the input common mode range covers the whole supply - no input pair V_{GS} in the headroom budget at all.

The cost is honest: five times less transconductance for the same current, more input capacitance, and the forward bias diode from bulk to source limits the drive. Bulk driven input stages show up where the input common mode is hostile - rail to rail buffers, sensor interfaces - not where noise or speed matter most.

Bulk as signal input

$$g_s \approx (n-1) g_m \approx 0.2 g_m$$

Input common mode: rail to rail

Cost: five times less transconductance, and the bulk-source diode must stay off

11.9 Fully differential

At 0.8 V, going fully differential is not a luxury, it is where the missing swing went: differential output doubles the signal amplitude for free, cancels even order distortion, and rejects the supply and substrate noise that a single ended output adds to the signal. Figure 7 shows a fully differential current mirror OTA.

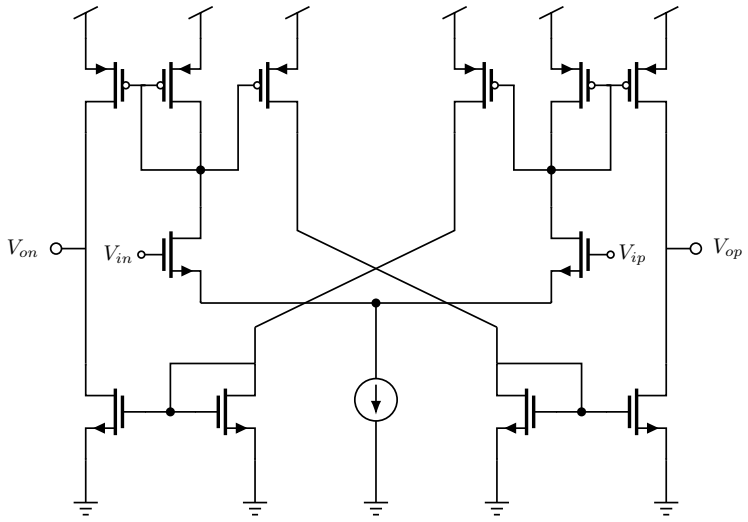


Figure 7: Fully differential current mirror OTA

The price of removing the diode connected definition of the output: the output common mode is no longer defined by the circuit itself. Both outputs can drift towards a rail together, and the differential loop cannot see it. Every fully differential OTA therefore carries a common mode feedback (CMFB) loop.

11.9.1 Common mode feedback

The CMFB loop in Figure 8 senses the average of the two outputs, compares it against a reference - usually mid supply - and trims a bias current in the OTA until the average sits where it should. The loop must be stable on its own, and fast enough to catch common mode disturbances, which in switched capacitor circuits usually means a switched capacitor CMFB sensing network.

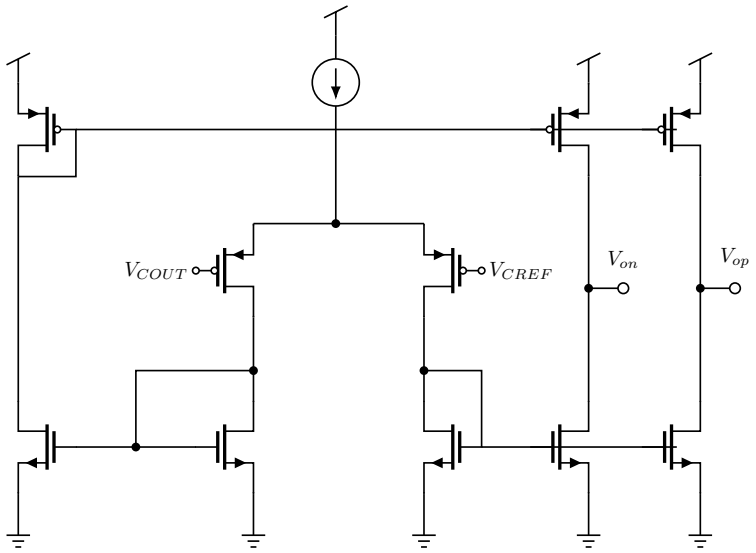


Figure 8: A common mode feedback amplifier

The amplifier of Figure 8 is the continuous time flavor: the output average is sensed, compared against V_{CREF}, and the result trims the tail bias. It corrects at every moment, but the sensing network loads the outputs, and at 0.8 V any sensing follower costs headroom.

11.9.2 Sensing the common mode

Figure 9 shows where the two voltages the CMFB amplifier compares come from. Source followers tap V_{on} and V_{op} without loading the outputs resistively, and the resistor network averages the two into the sensed common mode V_{COUT}. The reference V_{CREF} is generated the same way - a matching follower off a resistor divider - so the follower's level shift and its temperature drift cancel in the comparison, and the loop regulates the true output average.

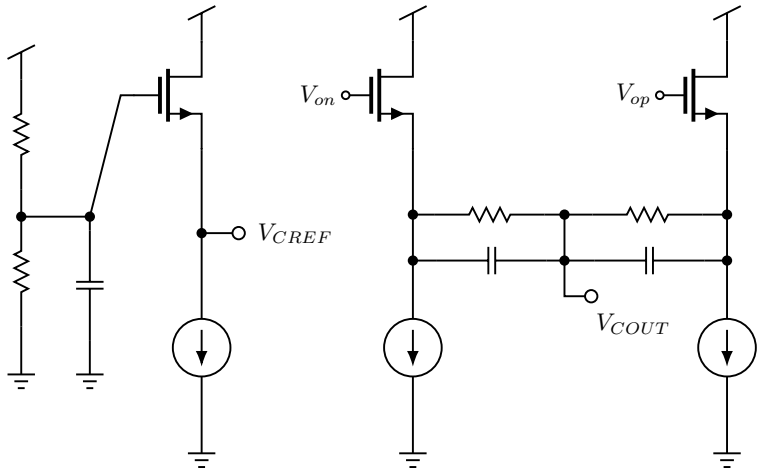


Figure 9: Common mode sense circuit with source followers, and the matching resistor divider generating the reference V_{CREF}

11.9.3 Switched capacitor CMFB

In a sampled system the standard answer is the switched capacitor CMFB in Figure 10. The two C_1 sense the average of the outputs and level shift it directly onto the tail bias node v_{cmfb} - no amplifier, no headroom, and capacitors are perfectly linear. The switched C_2 refresh the level shift towards $V_{cm} - V_B$ on every ϕ_1, so leakage and startup errors bleed away in a few clock cycles.

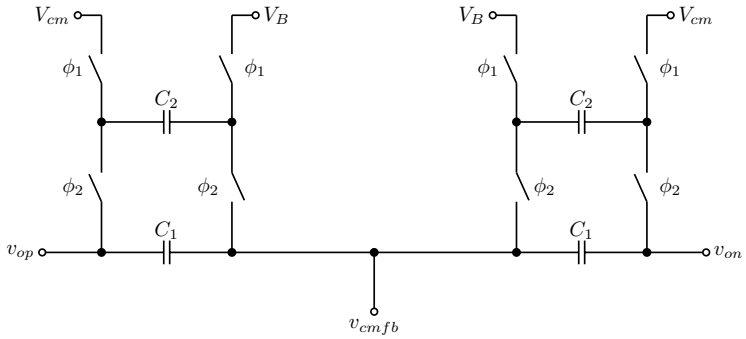


Figure 10: Switched capacitor CMFB

The price is clocked operation: between the phases the common mode is held only by the capacitors, so the loop corrects at the clock rate rather than continuously. In a switched capacitor filter or ADC that clock already exists, which is why nearly every fully differential OTA in a sampled system uses this network.

11.10 Bias circuits

Every V_B in this chapter has quietly assumed a bias network. The reference chapter builds the reference current itself; here is how that current becomes the gate voltages the OTAs need, Figure 11.

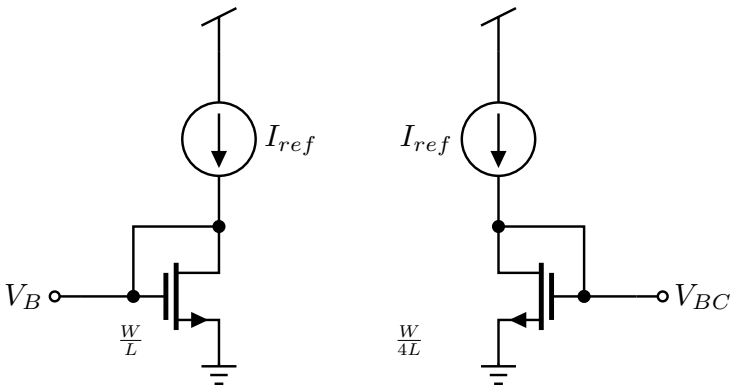


Figure 11: Mirror bias and wide swing cascode bias

The left branch is the workhorse: the reference current into a diode connected device gives $V_B = V_t + V_{DSAT}$, which every tail and mirror gate in this chapter copies. The right branch makes the cascode bias: the same current into a device with a quarter of the W/L needs twice the effective voltage, so its gate sits at $V_t + 2V_{DSAT}$. A cascode gated by V_{BC} then holds its mirror transistor right at the edge of saturation - the wide swing bias that the folded cascode's 0.4 V of output swing depends on.

Three practical rules come with the schematic. Distribute currents, not voltages: a V_B routed across the chip picks up every IR drop and ground difference on the way, so send a mirrored current and rebuild the voltage locally. Decouple every bias gate to its source rail with a capacitor - the bias node is part of the signal circuit at high frequency. And remember the bias block in the verification below: an OTA that starts before its bias does is an oscillator with ambitions.

11.11 From OTA to op amp

An OTA's output is a current source: high output resistance, happy with a capacitor, helpless into a resistor. The op amp is the same circuit plus an output stage that buys a low output resistance, Figure 12.

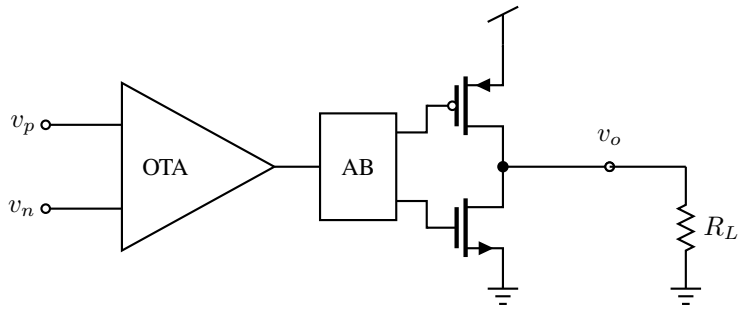


Figure 12: An op amp is an OTA plus an output stage

The class AB block level shifts the two gate drives so both output devices idle at a small quiescent current, yet either can deliver many times that current into the load - power efficiency a class A follower cannot match. At 0.8 V the output pair is drawn as two common source devices, because source followers no longer fit: a follower costs a full V_{GS} of swing, the common source pair costs one V_{DSAT} per rail.

On chip, almost every load is a capacitor or a switched capacitor, so inside the chip you nearly always want the OTA and its high output resistance - the loop gain is free gain. The op amp earns

its output stage at the pad ring: reference buffers, regulator error amplifiers driving pass devices, anything that leaves the die.

11.11.1 A complete OTA, sized

To make all of this concrete, Figure 13 shows a fully differential two-stage OTA that will drive most switched capacitor circuits, with every device sized. The notation is "WFLF": 24F4F means the width is 24 and the length 4 minimum gate lengths, so the same schematic ports between processes by re-reading F. Only one side is drawn - the other half mirrors it.

All the pieces of this chapter appear at once: a cascoded PMOS tail into the input pair, a cascoded current mirror load making the first stage output, a common source second stage with its 500 fF compensation capacitor returned to the cascode source instead of the gate - the cascoded Miller trick that hides the RHP zero - and, on the left, the CMFB amplifier. The outputs are sensed with 60k||20f networks (the capacitor keeps the sense path fast where the resistor divider rolls off), compared against a 100k/100k mid-supply divider, and the correction is injected in parallel with the VBP bias of the first stage load.

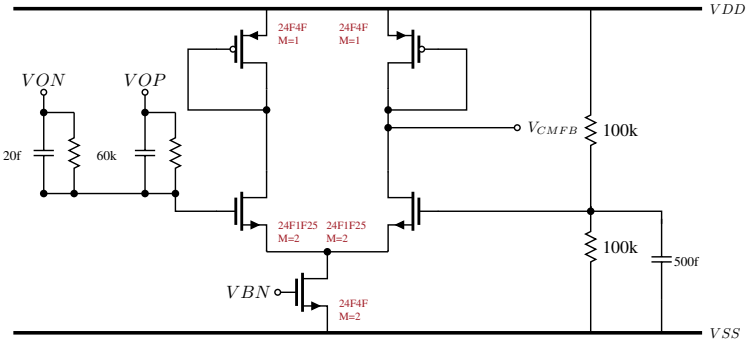


Figure 13: The common mode feedback amplifier. VON and VOP are sensed through 60k||20f networks and compared against a 100k/100k mid-supply reference. Each load PMOS is diode connected on its own - the gates are not tied together - so the gain is the modest, well defined g_{mn}/g_{mp} that a common mode loop wants, and the correction leaves as V_{CMFB}

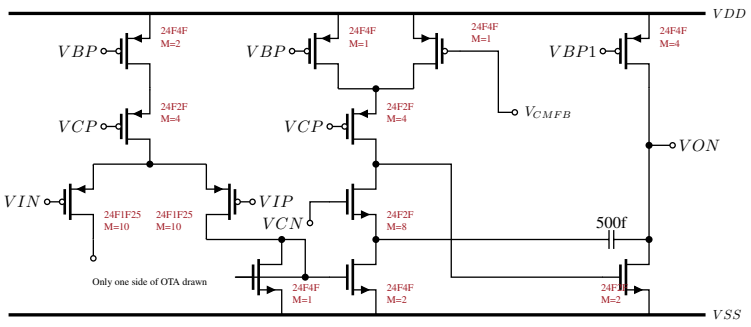


Figure 14: The OTA itself, one side drawn, sized in multiples of the minimum gate length F: a cascoded PMOS tail into the input pair, a cascoded mirror making the first stage output, and a common source second stage with 500 fF of cascode compensation. V_{CMFB} arrives from the amplifier in Figure 13

The bias generator in Figure 15 turns a 10 uA reference into the five gate voltages the OTA asked for. A diode connected NMOS sets the mirror line; one PMOS branch with a diode on top makes VBP; a long channel PMOS diode straight off the supply drops enough V_GS to make the cascode bias VCP, and its NMOS twin makes VCN; a stack of two NMOS diodes makes VBN for the tails; and a separate PMOS diode makes VBP1 so the second stage can be biased independently of the first.

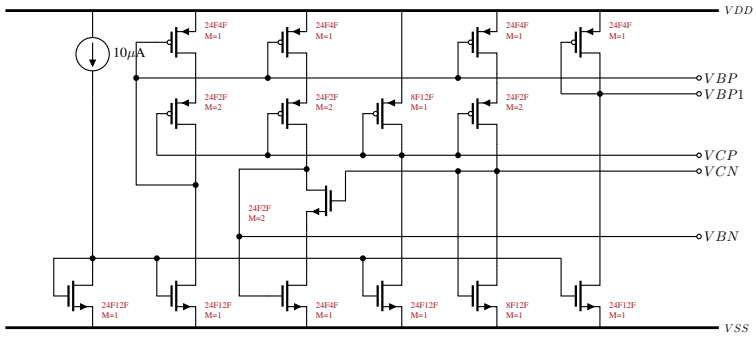


Figure 15: The bias generator. The 10 uA reference is mirrored once in an ordinary current mirror - the line along the bottom - and everything above it is wide swing cascode: the narrow 8F12F devices set VCP and VCN, and in each master the mirror device's gate hangs on the far end of its own stack

11.12 Dynamic amplifiers

The newest branch of the family tree is also, on inspection, one of the oldest: Hosticka showed dynamic CMOS amplifiers already in 1980 [4], and scaling has made the idea mainstream. A dynamic amplifier throws away the bias current entirely: it integrates its input onto a capacitor for a clocked instant and then stops. The "gain" is $g_m T/C$, the power is $CV^2 f$, and between samples the amplifier burns nothing. Ring amplifiers [5] do the same with an inverter chain that slams the output and then dead-bands itself into a precision settle.

They only work in sampled systems - a SAR or pipeline ADC stage, a discrete time filter - but there they have taken over: an amplifier that only exists while it is needed is the logical endpoint of the headroom and power budget this chapter started with.

Dynamic gain

$$A \approx \frac{g_m T}{C}$$

Power

$$P \propto C V_{DD}^2 f_s$$

Only in sampled systems, and everywhere in modern ADCs

11.13 Choosing

Topology	Gain	Swing	Best at
Five transistor	$g_m r_{ds}$	good	buffers, bias loops
Current mirror	$K g_m r_{ds}$	good	drive, SC circuits
Two stage Miller	$(g_m r_{ds})^2$	best	gain + swing
Folded cascode	$g_m(g_m r_{ds}^2)$	poor	SC settling
Inverter based	$g_m r_{ds}$	good	sub-1V, low power
Dynamic	$g_m T/C$	-	ADCs

Read the table bottom up: at 0.8 V the pressure is towards the bottom rows. Start with the five transistor OTA, and move only when a measured, simulated shortfall - gain, swing, drive, power - pushes you to a specific neighbor.

11.14 Verifying the OTA

Designing the OTA is the smaller half of the work. Before it goes into a system, a standard battery of analyses must say yes - and each row below has a reason to exist, usually a chip that failed without it.

Analysis	Testbench	Look for
Operating point	closed loop, DC	every device saturated, all corners
Loop gain	stb / broken loop	DC gain, UGF, phase margin > 60 deg
CMFB loop gain	stb on the CM loop	stable on its own, faster than the disturbance
Noise	AC noise, closed loop	input referred, thermal and flicker
Offset	Monte Carlo mismatch	sigma of input referred offset
Swing	sweep output, plot gain	where the gain collapses
Slew and settling	large signal step	settles to accuracy in the time budget
PSRR / CMRR	AC from supply / CM	worst case versus frequency
Start-up	transient from zero	bias and CMFB wake up, always

Analysis	Testbench	Look for
Power	DC, all corners	the budget holds where it is slowest

A few of the rows deserve their own sentence.

The operating point check is first because it is cheap and catches most disasters: a single transistor pushed into triode at the slow-slow, low supply, hot corner explains many a "the gain is 20 dB too low in the lab" story. Check it across corners, supplies and temperature before trusting any small signal number.

The loop gain must be measured with the loading in place - the real feedback network, the real sampling capacitors - because the phase margin depends on the load more than on the OTA. For a fully differential OTA there are two loops, and the CMFB loop must be analyzed separately: it has its own crossover and its own phase margin, and an unstable CMFB loop looks exactly like an oscillating OTA.

Offset and noise are budget items, not pass/fail: Monte Carlo gives the offset sigma that the system - a comparator threshold, an ADC code - must absorb, and the input referred noise integrated over the signal band must sit under the quantization or thermal floor it feeds.

Slew and settling only exist in a large signal transient. The small signal bandwidth promises nothing about a full scale step: the input pair steers all its tail current, the OTA slews at I/C, and only the last stretch is exponential. In a switched capacitor circuit the settling budget is half a clock period, and it is the transient - with mismatch, with corners - that says whether the budget holds.

And start-up: simulate the whole thing from zero supply, every time. Bias loops and CMFB loops both have a stable state called "off", and finding it in silicon is the most expensive way to learn this rule.

11.15 Summary

The one-page version of this chapter:

- At 0.8 V: never stack two gate-source voltages, and count every saturation voltage
- Five transistor OTA first; upgrade only for a reason
- Need drive: current mirror OTA. Need gain: two stage. Need one-stage gain: folded cascode

- Miller compensation splits the poles; the unity gain frequency is gm over Cc; mind the RHP zero
- Inverters amplify: both transconductances for one branch current, and Nauta's OTA needs no tail
- Fully differential doubles swing but must pay the CMFB tax
- In sampled systems, dynamic amplifiers win the power argument

11.16 Would you like to know more?

The paper that named the transconductance amplifier and argued it would obsolete the op amp [1]

OTA-based filter design, the reason the OTA became a building block rather than a curiosity [2]

Nauta's inverter transconductor, drawn in this chapter, with the analysis of why its common mode holds [3]

Dynamic CMOS amplifiers, twenty years before the idea became mainstream [4]

Ring amplifiers, where the dynamic branch of the family went [5]

[1] C. F. Wheatley and H. A. Wittlinger, "OTA obsoletes OP. AMP." in *Proc. National electronics conference*, 1969, pp. 152–157 [Online]. Available: https://class.ece.iastate.edu/ee435/miscHandouts/OTA%20Wheatley%20and%20Wittlinger%20Dec%2069.pdf

[2] R. L. Geiger and E. Sánchez-Sinencio, "Active filter design using operational transconductance amplifiers: A tutorial," *IEEE Circuits and Devices Magazine*, vol. 1, no. 2, pp. 20–32, 1985, doi: 10.1109/MCD.1985.6311946.

[3] B. Nauta, "A CMOS transconductance-c filter technique for very high frequencies," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 2, pp. 142–153, 1992, doi: 10.1109/4.127337.

[4] B. J. Hosticka, "Dynamic CMOS amplifiers," *IEEE Journal of Solid-State Circuits*, vol. 15, no. 5, pp. 881–886, 1980, doi: 10.1109/JSSC.1980.1051488.

[5] B. Hershberg, S. Weaver, K. Sobue, S. Takeuchi, K. Hamashita, and U.-K. Moon, "Ring amplifiers for switched capacitor circuits," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 12, pp. 2928–2942, 2012, doi: 10.1109/JSSC.2012.2217865.

12 Integrated Passives

Keywords: Resistors, Polysilicon, Diffusion, Capacitors, MOM, MOS Cap, Varactors, Inductors

12.1 Metal in ICs is not wire in schematic . 199
12.2 Resistors 200
12.2.1 Polysilicon 200
12.2.2 Diffusion 201
12.2.3 Metal 202
12.3 Capacitors 202
12.3.1 What is S, M, L, XL on a chip? 202
12.3.2 Metal-Oxide-Metal finger capacitors . . 203
12.3.3 MOS capacitors . . . 203
12.3.4 Varactors 205
12.4 Inductors 205
12.5 Variation in passives 206
12.6 Relative precision . 207
12.7 Summary 207
12.8 Would you like to know more? 208

12.1 Metal in ICs is not wire in schematic

Metal wires in an integrated circuit come in two types, copper and aluminium.

Most of the routing layers will be copper. To ensure that the copper ions don't diffuse into the silicon-oxide a barrier material surrounds all copper interconnect.

Copper is too stiff to be wire-bonded. As such, the top layer metals would be aluminium.

Since the routing is so small, we have to care about the parasitic properties of the routing. Below is a table with some common quantities for copper. For example, if we have 1000 μm metal wire with 1 μm width, then it would be approximately 150 Ω, 1 nH , 1 pF and tolerate a maximum of 1 mA DC current.

Parameter	Typ. Value	Unit
Resistance	150	mΩ/□
Capacitance	1	fF/μm
Inductance	1	nH/mm
Max DC current	1	mA/□

The type of circuit we have determines what we must simulate. Everything needs to be simulated with parasitic capacitance and max current. Analog and power circuits need parasitic resistance as well; only RF and power usually need the inductance too.

Circuit type	Must simulate/know
All	C Imax
Analog, Power	R C Imax
Some RF, Some Power	R L C Imax

To simulate the effects of parasitics, we need a description of the technology. A Process Design Kit (PDK). Most PDKs are closely guarded secrets, as they describe many things about the way the foundry makes the integrated circuits.

Some PDKs are open source, however, see Skywater 130 nm and IHP-Open-PDK

In addition to the PDK, we need tools that can calculate from the layout the parasitic elements. Some of the tools are

Layout parasitic extraction tools

- Calibre xRC
- Synopsys StarRC
- Cadence Quantus
- Magic VLSI

3D EM Simulators

- Keysight ADS
- HFSS

Transistor CAD (TCAD)

- Synopsys TCAD

12.2 Resistors

Sometimes we want a specific resistance. In general, any resistance on IC will vary in absolute value by maybe up to ± 20 %. The relative size, however, can be controlled to within 0.1 %.

In other words, you can't rely on a 1 kOhm resistor actually being 1 kOhm, it might be 0.8 kOhm. If you have two, however, you can trust that both of them will be 0.8 kOhm.

That's why almost all analog circuits rely on the relative sizes of passives, not the absolute value. If a circuit does rely on absolute values, then it usually needs to be trimmed in production.

12.2.1 Polysilicon

The workhorse resistor is the gate material itself: polysilicon. In Figure 1 the resistor is a strip of poly with contacts at both ends, sitting on field oxide so it is insulated from the substrate.

Can be both N-doped, and P-doped

Often with two flavors, with, and without silicide

Silicide reduces resistance of polysilicon

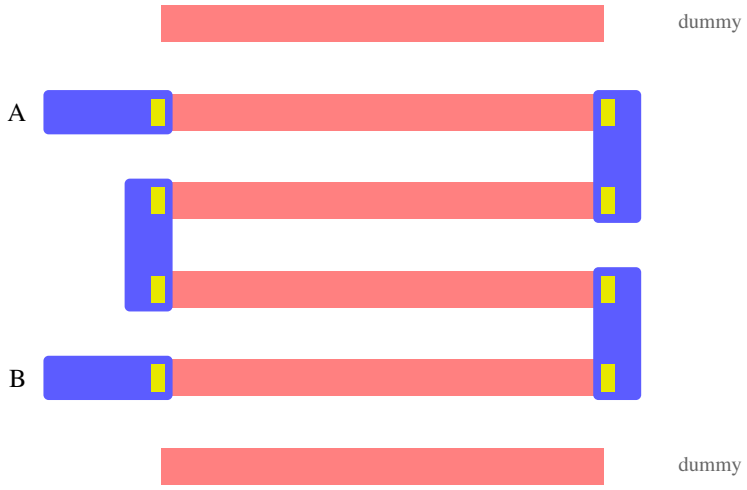


Figure 1: Polysilicon resistor

In a modern process the poly on top of transistors is silicided - a metal alloy on the surface that reduces the sheet resistance to a few ohms per square, great for gates, useless for resistors. The foundry therefore offers a mask that blocks the silicide, and the unsilicided flavor has a sheet resistance of hundreds of ohms per square with a small temperature coefficient. When an analog schematic says "resistor", it is nearly always unsilicided poly.

12.2.2 Diffusion

A doped region in the silicon also conducts, and Figure 2 shows it used as a resistor.

Use doped region as resistor

Usually without silicide

Non-linear capacitance

Tricky temperature dependence

The diffusion resistor comes with baggage. The resistor body forms a pn junction to whatever surrounds it, so it carries a distributed, voltage dependent junction capacitance, and the depletion region eats into the conducting cross section, so the resistance itself moves with the voltage. The substrate is p- in every modern process, so an n+ resistor sits straight in it, but a p+ resistor needs an n-well around it - and that well is a third terminal you must bias, with its own junction to the substrate underneath. Add a temperature coefficient set by doping and mobility pulling in opposite directions, and the diffusion resistor is a device you use when the poly resistor is unavailable, not because you want to.

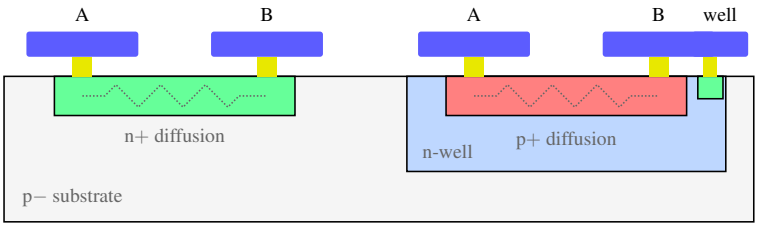


Figure 2: Diffusion resistors: n+ in the substrate, p+ in an n-well

12.2.3 Metal

Metal, Figure 3, is at the other end of the scale: at milliohms per square you would need a kilometer of it for a useful resistance.

Usually too low ohmic to be a useful resistor

Useful for “separating nets” in schematic and layout

Must be considered for power supply and ground routing (high currents)

A zero ohm metal “resistor” still earns its place in the schematic: it splits a net in two, which lets layout tools keep sense lines away from current carrying lines, and lets the extraction report where the IR drop goes. In power routing the metal resistance is not a device you add but a parasitic you budget: milliohms times amperes is millivolts of ground bounce.

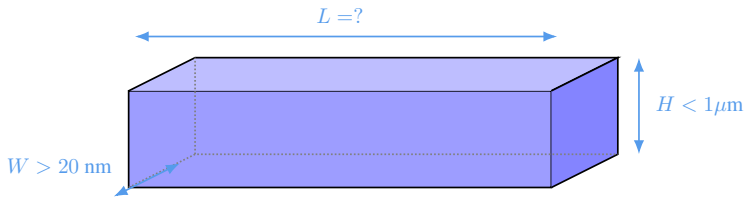


Figure 3: Metal resistor

12.3 Capacitors

12.3.1 What is S, M, L, XL on a chip?

Capacitors are where the silicon area goes, so before the devices, a sense of scale. The nRF52832 die is about 9.6 million square micrometers, and on it, a component below five thousand square micrometers is small, while anything above two hundred thousand - a fiftieth of the die - is extra large and had better earn its keep.

nRF52832

$$3200\mu m \times 3000\mu m = 9600k\mu m^2$$

Size	Area
S	below 5 k square um
M	below 50 k square um
L	below 200 k square um
XL	above 200 k square um

12.3.2 Metal-Oxide-Metal finger capacitors

The default capacitor in a modern process is drawn, not grown: thin metal fingers side by side, alternating polarity, stacked over several metal layers, as in Figure 4. The lateral spacing between fingers is smaller than the vertical oxide between layers, so the sideways fringe field does most of the work.

Unit capacitance

$$\approx 1fF/\mu m^2/layer$$

$$10pF = 100\mu m \times 100\mu m = 10k\mu m^2$$

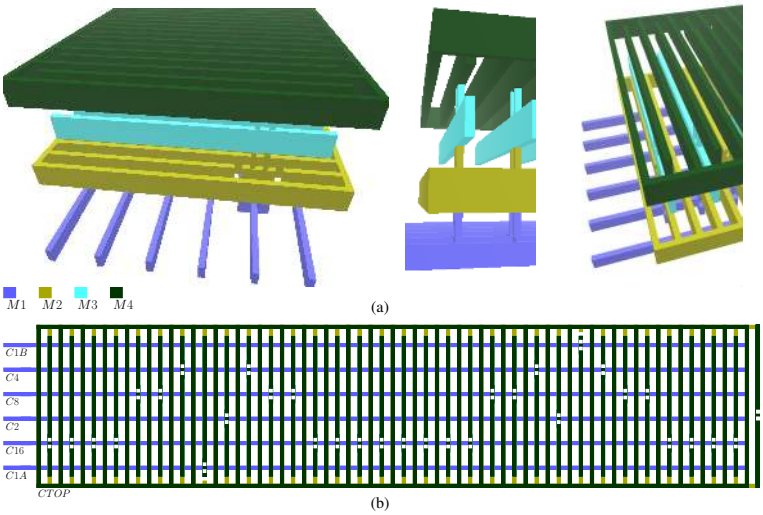


Figure 4: Metal-oxide-metal finger capacitor

The MOM capacitor is linear, matches to a tenth of a percent when drawn as identical units, and costs nothing but metal. Its weakness is density: at about a femtofarad per square micrometer per layer, ten picofarads is a hundred micrometers on a side - a Medium on the scale above, for one capacitor.

12.3.3 MOS capacitors

When density matters more than linearity, use the thinnest oxide on the die: the gate oxide. A MOSFET with source, drain and bulk

tied together is a capacitor from the gate to the channel, Figure 5, and it is about ten times denser than the MOM capacitor.

The price is that the capacitance depends on the bias. Below threshold there is no channel and the gate sees the oxide in series with the depletion region; above threshold the inversion layer forms and the capacitance jumps to the full oxide value. A MOS capacitor is a fine decoupling capacitor - the bias is fixed and nobody cares about linearity - and a poor filter capacitor.

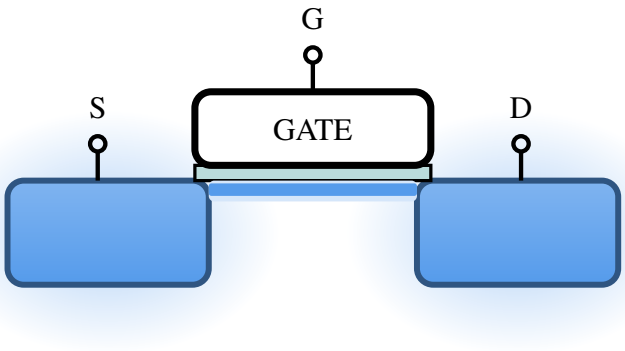


Figure 5: A MOS capacitor is a MOSFET in strong inversion

```
dicex/sim/spice/NCHIO/vcap.cir
* gate cap

.include ../../../models/ptm_130.spi

vdrain D 0 dc 1
vgaini G 0 dc 0.5
vbulk B 0 dc 0
vcur S 0 dc 0

M1 D G S B nmos  w=1u  l=1u

.op
```

The operating point readout below, from the SPICE deck on the left, shows where the number comes from: at this bias the gate-gate capacitance C_{gg} of the 1 um by 1 um device reads about 10 fF - ten femtofarads per square micrometer, right at the estimate.

Moscap is

$$\approx 10fF/\mu m^2$$

$$10pF = 31\mu m \times 31\mu m \approx 1k\mu m^2$$

```
dicex/sim/spice/NCHIO/vcap.vlog
Device m1:
    Vgs     (gate-source voltage)        [V] : 0.5
    Vgd     (gate-drain voltage)         [V] : -0.5
    Vds     (drain-source voltage)       [V] : 1
    Vbs     (bulk-source voltage)        [V] : 1.90808e-12
    Vbd     (bulk-drain voltage)         [V] : -1
    Id      (drain current)              [A] : 7.32634e-06
    Is      (source current)             [A] : -7.32633e-06
```

```
Ibd     (bulk-drain current)         [A] : -1.01e-12
Ibs     (bulk-source current)        [A] : 9.581e-25
Vt      (threshold voltage)          [V] : 0.378198
Vgt     (gate overdrive voltage)     [V] : 0.121802
Vgsteff (effective vgt)              [V] : 0.12515
Gm      (transconductance)           [S] : 8.44164e-05
Gmb     (bulk bias transconductance) [S] : 2.00071e-05
Ueff    (mobility)             [cm^2/Vs] : 417.675
Gds     (channel conductance)        [S] : 1.95043e-07
Rds     (output resistance)        [Ohm] : 5.12708e+06
Vdsat   (drain saturation voltage)   [V] : 0.14171
IC      (inversion coefficient)       [] : 4.42478
Cgs     (gate-source capacitance)    [F] : 9.98457e-15
Csg     (source-gate capacitance)    [F] : 5.86932e-15
Cgd     (gate-drain capacitance)     [F] : 3.98239e-16
Cdg     (drain-gate capacitance)     [F] : 3.91086e-15
Cds     (drain-source capacitance)   [F] : 4.30968e-15
Cgg     (gate-gate capacitance)      [F] : 1.05198e-14
Cdd     (drain-drain capacitance)    [F] : 1.05198e-14
Css     (source-source capacitance)  [F] : 0
Cgb     (gate-bulk capacitance)      [F] : 1.05198e-14
Cbg     (bulk-gate capacitance)      [F] : 1.74123e-15
Cbs     (bulk-source capacitance)    [F] : 8e-16
Cbd     (bulk-drain capacitance)     [F] : 3.97768e-16
```

12.3.4 Varactors

A varactor is a "variable capacitor", usually it's a device that varies the capacitance with the voltage across the device.

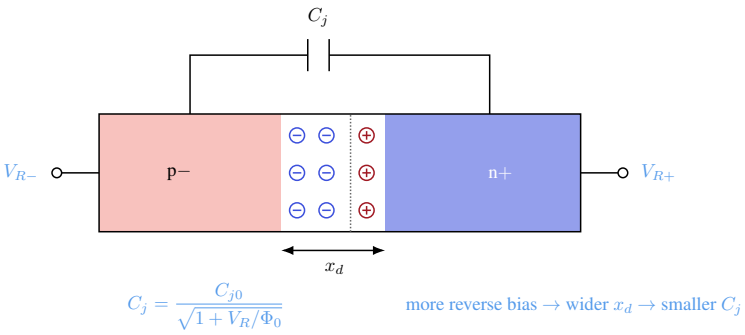


Figure 6: A reverse biased pn junction as a varactor

The junction depletion capacitance falls as the reverse bias grows - the same square root we met in the diode chapter - which makes a reverse biased junction a voltage controlled capacitor. The other common varactor is the MOS capacitor biased around its transition. The customer for both is the oscillator chapter: a varactor in an LC tank turns a fixed oscillator into a voltage controlled one.

12.4 Inductors

Inductors on chip are spirals in the top metals, like the ones visible on the nRF51822 die photograph in Figure 7. The top layers are the thick, low resistance ones, and resistance is the enemy: the quality

factor of an integrated inductor - some tens at gigahertz - is set by the metal losses and by eddy currents in the substrate below.

Usually two top metals, because they are thick (low ohmic)

Use foundry model

3D electro magnetic simulation often needed

An inductor is the least portable device on the die: its value and its losses depend on everything nearby, so use the foundry's characterized model, and if the layout deviates from it - or the frequency is high enough that every via matters - budget for a 3D electromagnetic simulation. Nanohenries cost hundreds of micrometers on a side, which is why inductors only appear where nothing else will do: LC oscillators, RF matching and power converters.

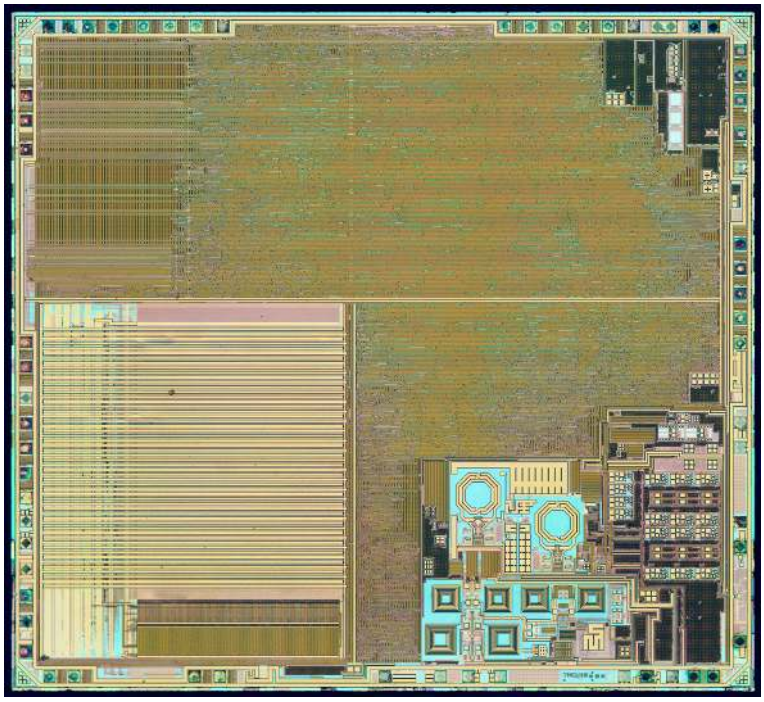


Figure 7: nRF51822 die - the spirals are inductors. Die photograph by zeptobars.com, CC BY 3.0

12.5 Variation in passives

The rule from the resistor introduction deserves numbers. Nothing on an IC has a trustworthy absolute value: oxide thickness, implant dose and line width all drift from lot to lot, and the passives drift with them. What the process does guarantee is that two identical devices drawn next to each other drift together.

Absolute value for resistors and capacitors: 10 % to 20 %

Relative precision for closely spaced devices: 0.1 % to 1 %

Relative precision for devices far apart on the same die: worse than 2 %

12.6 Relative precision

Figure 8 shows the payoff in circuit form: a resistor divider whose output is half the input to a tenth of a percent, and two capacitors whose charge ratio holds equally well - even though every one of those devices may be off by ten percent in absolute value. The precision is earned in layout: identical unit devices, interdigitated or common centroid so process gradients hit both halves equally, dummies at the edges so every unit sees the same neighborhood.

This is the deal the whole chapter has been building to: design circuits so that only ratios matter - two resistors setting a gain, a capacitor array setting a DAC - and the process variation cancels out of the equation.

Resistors and Capacitors can be matched extremely well

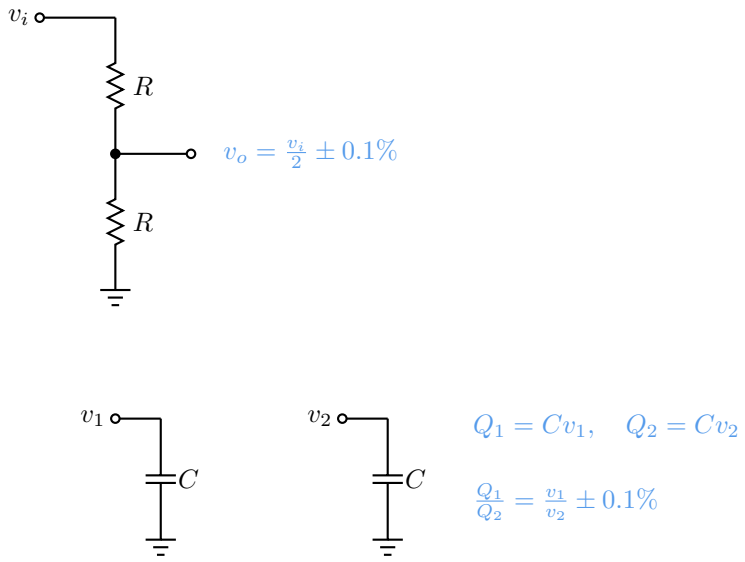


Figure 8: Ratios of matched devices hold to a tenth of a percent

12.7 Summary

The one-page version of this chapter:

- Metal is not a schematic wire: budget resistance, capacitance and current for every long route
- Resistors: unsilicided poly first; diffusion if you must; metal never

- Capacitors: MOM for linearity and matching, MOS cap for density at a fixed bias
- Varactors turn junctions or MOS caps into tunable capacitors for oscillators
- Inductors are area-hungry and non-portable: foundry model or EM simulation
- Absolute values drift by tens of percent; ratios of matched units hold a tenth of a percent - design with ratios

12.8 Would you like to know more?

Your own PDK documentation is the real reading here: what each resistor, capacitor and inductor in SKY130 actually is, and what it costs in area and parasitics

The circuit-level consequences, chapter by chapter [1]

[1] D. Johns and K. Martin, *Analog integrated circuit design*. John Wiley & Sons, Inc., 1997.

The Tools 13

13.1 Tools 209
13.1.1 Setup WSL (Applicable for Windows users) 209
13.1.2 Setup public key towards github . . . 210
13.1.3 Provide git with author identity . . . 210
13.1.4 Get AICEX and setup your shell . . 210
13.1.5 On systems with python3 > 3.12 . . . 211
13.1.6 Install Tools 211
13.1.7 Install cicconf 212
13.1.8 Install cicsim 212
13.1.9 Install cicpy 212
13.1.10 Setup your ngspice settings 213
13.2 Check that magic and xschem work . 213
13.3 Summary 213
13.4 Would you like to know more? 213

Keywords: WSL, Git, AICEX, ngspice, xschem, Magic, cicconf, cicsim, cicpy

13.1 Tools

I would strongly recommend that you install all tools locally on your system. There is a video that describes the install procedure. It's a few years old, but should still be able to guide you https://youtu.be/DRppsdjo2Rc?si=x8cJsa1lpncvSFmu.

For the analog toolchain we need some tools, and a process design kit (PDK).

- Skywater 130nm PDK. I use open_pdks to install the PDK
- Magic VLSI for layout (Version 8.3 revision 541)
- ngspice for simulation (version 45.2)
- netgen for LVS (1.5.295)
- xschem (3.4.8RC)
- verilator (5.034)
- python > 3.10

The tools are not that big, but the PDK is huge, so you need to have about 50 GB disk space available.

13.1.1 Setup WSL (Applicable for Windows users)

Install a Linux distribution such as Ubuntu 24.04 LTS by running the following command in PowerShell on Windows and follow the instructions.

```
wsl --install -d Ubuntu-24.04
```

When you have installed the Linux distribution and signed into it, install make

```
sudo apt install make
```

13.1.2 Setup public key towards github

Do

```
ssh-keygen -t rsa
```

And press "enter" on most things, or if you're paranoid, add a passphrase

Then

```
cat ~/.ssh/id_rsa.pub
```

And add the public key to your github account. Settings - SSH and GPG keys

13.1.3 Provide git with author identity

There are interactions with git that require an author identity. You are supposed to use one of these interactions a lot during the project, namely, `git commit`. What you need to provide is an email address and a name. If you would like to keep your real email address private/secret, read what it says on GitHub at your user settings page under emails. Use the below commands to provide the author identity information to git.

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

13.1.4 Get AICEX and setup your shell

You don't have to put aicex in `$HOME/pro`, but if you don't know where to put it, choose that directory.

```
cd
mkdir pro
cd pro
git clone --recursive https://github.com/wulffern/aicex.git
```

You need to add the following to your `~/.bashrc` (note that `~` refers to your home directory `$HOME/.bashrc` also works, or `$HOME/.bash_profile` on some newer macs)

```
export PDK_ROOT=/opt/pdk/share/pdk
export LD_LIBRARY_PATH=/opt/eda/lib
export PATH=/opt/eda/bin:$HOME/.local/bin:$PATH
```

13.1.5 On systems with python3 > 3.12

On newer systems it's not trivial to install python packages because python is externally managed. As such, we need to install a python environment.

```
#- Find a package similar to name below
sudo apt-get update
sudo apt install python3.12-venv
sudo mkdir /opt
sudo mkdir /opt/eda
sudo mkdir /opt/eda/python3
sudo chown -R $USER:$USER /opt/eda/python3/
python3 -m venv /opt/eda/python3
```

Modify the `~/.bashrc` to include the python environment

```
export PATH=/opt/eda/bin:/opt/eda/python3/bin:$HOME/.local/bin:$PATH
```

13.1.6 Install Tools

Make sure you load the settings before you proceed

```
source ~/.bashrc
```

Hopefully the commands below work, if not, then try again, or try to understand what fails. There is no point in continuing if one command fails.

```
cd aicex/tests/
make requirements
make tt
```

On a mac, you probably need to add bison to the path

```
export PATH="/opt/homebrew/opt/bison/bin:$PATH"
```

I've split the install of each of the tools. It's possible to run the commented out lines instead, but they often fail

```
#make eda_compile
#sudo make eda_install
make magic_compile  magic_install
make netgen_compile netgen_install
make xschem_compile xschem_install
make iverilog_compile iverilog_install
make ngspice_compile # Sometimes fails
make ngspice_compile ngspice_install
```

For later versions of ngspice, they have modified the build system, so you may have to run

```
make ngspice2_compile
make ngspice2_install
```

On Mac, do

```
brew install yosys verilator
```

On Linux, do

```
make yosys_compile yosys_install
```

On all, do

```
python3 -m ensurepip --default-pip

python3 -m pip install matplotlib numpy click svgwrite \
    pyyaml pandas tabulate wheel setuptools tikzplotlib
source install_open_pdk.sh
```

13.1.7 Install cicconf

cIcConf is used for configuration. How the IPs are connected, and what version of IPs to get.

```
cd
cd pro/aicex/ip/cicconf
git checkout main
git pull
python3 -m pip install -e .
cd ../
```

Update IPs

```
cicconf clone --https
cd ../..
```

13.1.8 Install cicsim

cIcSim is used for simulation orchestration.

```
cd aicex/ip/cicsim
git checkout main
git pull
python3 -m pip install -e .
cd ../..
```

13.1.9 Install cicpy

CicPy is used to generate layout

```
cd aicex/ip/cicpy
git checkout master
git pull
python3 -m pip install -e .
cd ../..
cd aicex/ip/cicspi
git checkout main
git pull
python3 -m pip install -e .
cd ../..
```

13.1.10 Setup your ngspice settings

Edit `~/.spiceinit` and add

```
set ngbehavior=hsa     ; set compatibility for PDK libs
set ng_nomodcheck      ; don't check the model parameters
set num_threads=8      ; CPU hardware threads available
set skywaterpdk
option noinit          ; don't print operating point data
option klu
optran 0 0 0 100p 2n 0 ; don't use dc operating point,
option opts
```

13.2 Check that magic and xschem work

To check that magic and xschem work

```
cd ~/pro/aicex/ip/sun_sar9b_sky130nm/work
magic ../design/SUN_SAR9B_SKY130NM/SUNSAR_SAR9B_CV.mag &
xschem -b ../design/SUN_SAR9B_SKY130NM/SUNSAR_SAR9B_CV.sch &
```

13.3 Summary

The one-page version of this chapter:

- The whole flow is open source: xschem, ngspice, magic, netgen, and the cic tools on top
- Everything installs from scripts, and the docs and video walk the setup end to end
- If the tools do not run, nothing else in this course happens - do this first, and ask early when stuck

13.4 Would you like to know more?

Each tool documents itself well: xschem, ngspice, magic, netgen and the SKY130 PDK

The tools chapter later in this book covers the cic layer built on top of them

Sky130nm tutorial 14

Keywords: IP, xschem, ngspice, Corners, Layout, DRC, LVS, LPE, Documentation

Before you start the tutorial you need to have the tools installed.

If you're a student of mine, then you must install the tools locally on your PC. Read check The Tools chapter first.

If you're the impatient kind, then check https://analogicus.com/aicex/ which shows how to start the docker image.

14.1 Create the IP

I've made some scripts to automatically generate the IP.

To see what files are generated, see `tech_sky130A/cicconf/lelo.yaml`

```
cd aicex/ip
cicconf newip ex --project lelo --technology sky130A --ip  tech_sky130A/cicconf/lelo.yaml
```

14.2 The file structure

It matters how you name files, and store files. I would be surprised if you had a good method already, as such, I won't allow you to make your own folder structure and names for things. I also control the filenames and folder structure because there are many scripts to make your life easier (yes, really) that rely on an exact structure. Don't mess with it.

14.2.1 Github workflows

On github it's possible use something called workflows to run things every time you push a new version. It's really nice, since it can then check that your design is valid.

The workflows are defined below.

```
.github
   workflows
   docs.yaml # Generate a github page
   drc.yaml  # Run Design Rule Checks
   gds.yaml  # Generate a GDS file from layout
   lvs.yaml  # Run Layout Versus Schematic
             # and Layout Parasitic Extraction
```


14.1 Create the IP 215
14.2 The file structure . 215
14.2.1 Github workflows . 215
14.2.2 Configuration files . 216
14.2.3 Design files 216
14.2.4 Simulations 217
14.2.5 The work 217
14.3 Github setup 217
14.4 Start working . . . 217
14.4.1 Edit README.md . 217
14.4.2 Familiarize yourself with the Makefile and make 218
14.5 Draw Schematic . 218
14.5.1 Add Ports 218
14.5.2 Add transistors . . . 218
14.5.3 Netlist schematic . . 219
14.6 Typical corner SPICE simulation 219
14.6.1 Setup simulation environment 220
14.6.2 Familiarize yourself with the simulation folder 220
14.6.3 Modify default testbench (tran.spi) 220
14.6.4 Modify measurements (tran.meas) . 221
14.6.5 Modify result specification (tran.yaml) . 221
14.6.6 Check waveforms . 222
14.7 All corners SPICE simulations 222
14.7.1 Remove Vh and Vl corners (Makefile) . 223
14.7.2 Run all corners . . . 223
14.7.3 Get creative with python 223
14.7.4 Generate simulation summary 224
14.7.5 Viewing results without GUI browser . . 224
14.7.6 Think about the results 224
14.8 Draw Layout 225
14.8.1 Magic VLSI 225
14.8.2 Add transistors . . . 226
14.8.3 Place devices 226
14.8.4 Add Ground 227
14.8.5 Route Gates 228
14.8.6 Drain of M2 228
14.8.7 Add labels 229

14.2.2 Configuration files

Each IP has a few files that define the setup, you'll need to modify at least the `README.md` and the `info.yaml`.

```
.gitignore  # files that are ignored by git
README.md   # Frontpage documentation
config.yaml # What libraries are used. Used by  cicconf
info.yaml   # Setup names, authors etc
media       # Where you should store images for documentation
tech -> ../tech_sky130A  # The technology library
```

14.2.3 Design files

A "cell" in the open source EDA world should consist of the following files

- Schematic (.sch)
- Layout (.mag)
- Documentation (.md)

The files must have the same name, and must be stored in `design/<LIB>/` as shown below.

Note there are also two symbolic links to other libraries. These two libraries contain standard cells and standard analog transistors (ATR) that you should be using.

```
design
  LELO_EX_SKY130A
  LELO_EX.sch
  JNW_ATR_SKY130A -> ../../jnw_atr_sky130a/design/JNW_ATR_SKY130A
  JNW_TR_SKY130A -> ../../jnw_tr_sky130a/design/JNW_TR_SKY130A
```

For example, if the cell name was `LELO_EX`, then you would have

- `design/LELO_EX_SKY130A/LELO_EX.sch`: Schematic (xschem)
- `design/LELO_EX_SKY130A/LELO_EX.sym`: Symbol (xschem)
- `design/LELO_EX_SKY130A/LELO_EX.mag`: Layout (Magic)
- `design/LELO_EX_SKY130A/LELO_EX.md` : Markdown documentation (any text editor)

All these files are text files, so you can edit them in a text editor, but mostly you shouldn't (except for the Markdown)

14.2.4 Simulations

All simulations shall be stored in `sim`. Once you have a Schematic ready for simulation, then

```
cd sim
make cell CELL=LELO_EX
```

This will make a simulation folder for you. Repeat for all your cells.

```
sim
  Makefile
  cicsim.yaml -> ../tech/cicsim/cicsim.yaml
```

14.2.5 The work

All commands (except for simulation), shall be run in the `work` folder.

In the `work/` folder there are startup files for Xschem (xschemrc) and Magic (.magicrc). They tell the tools where to find the process design kit, symbols, etc. At some point you probably need to learn those also, but I'd wait until you feel a bit more comfortable.

```
work
 .magicrc
 Makefile
 mos.24bit.dstyle -> ../tech/magic/mos.24bit.dstyle
 mos.24bit.std.cmap -> ../tech/magic/mos.24bit.std.cmap
 xschemrc
```

14.3 Github setup

Create a repository on github. The name of the repository that you make on GitHub has to be the same as what is written after `<your username>` in the last command below. In this example, that is `lelo_ex_sky130a`.

```
cd lelo_ex_sky130a
git remote add origin \
 git@github.com:<your username>/lelo_ex_sky130a.git
```

14.4 Start working

14.4.1 Edit README.md

Open README.md in your favorite text editor and make necessary changes.

14.4.2 Familiarize yourself with the Makefile and make

I write all commands I do into a Makefile. There is nothing special with a Makefile, it's just what I choose to use 20 years ago. I'm not sure I'd choose something different now.

```
cd work
make
```

Take a look inside the file called Makefile.

14.5 Draw Schematic

The block we'll make is a current mirror with a 1 to 4 scaling.

A schematic is how we describe the connectivity, and the types of devices in an analog circuit. The open source schematic editor we will use is XSchem.

Open the schematic:

```
xschem -b ../design/LELO_EX_SKY130A/LELO_EX.sch &
```

14.5.1 Add Ports

Add IBPS_5U and IBNS_20U ports, the P and N in the name signifies what transistor the current comes from. So IBPS must go into a diode connected NMOS, and N will be our output, and go into a diode connected PMOS somewhere else.

14.5.2 Add transistors

Use 'I' or 'Shift+i' (note the letter case) to open the library manager. Click the `lelo_ex_sky130A/design` path, then `JNW_ATR_SKY130A` and select `JNWATR_NCH_4C5F0.sym`

The naming convention for these transistors is `<number of contacts on drain/source>C<times minimum gate length>F`, so the number before the C is the width, and the number before/after the F is the length. The absolute size does not matter for now. Just think "4C5F0 is a 4 contact wide long transistor", while a "4C1F2 is a 4 contact wide, short transistor".

Select the transistor and press 'c' to copy it, while dragging, press 'shift-f' to flip the transistor so our current mirror looks nice. 'shift-r' rotates the transistor, but we don't want that now.

Place two transistors for the output transistor, as shown in the figure below.

Press ESC to deselect everything

Select the input transistor, and change the name to ‘xo1’

Select the first output transistor, and change the name to ‘xo0[1:0]’. Using bus notation on the name will create 2 transistors.

Select the second output transistor and give it the name ‘xo2[1:0]’.

Select ports, and use ‘m’ to move the ports close to the transistors.

Press ‘w’ to route wires.

Use ‘shift-z’ and z, to zoom in and out

Use ‘f’ to zoom full screen

Remember to save the schematic

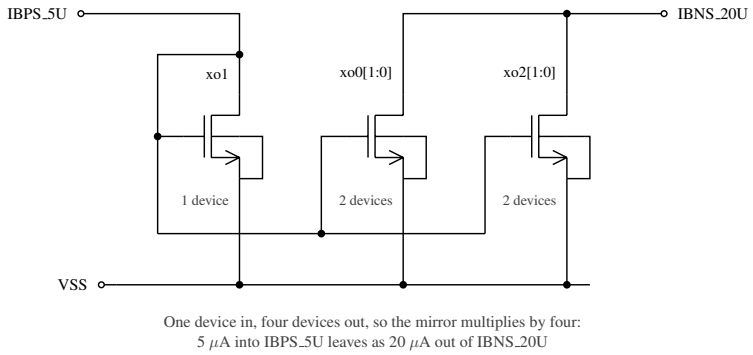


Figure 1: The finished LELO_EX schematic - one diode connected input instance and two output instances of two devices each, between the IBPS_5U, IBNS_20U and VSS ports. Four output devices to one input device is where the 5 uA in and 20 uA out of the port names comes from. The devices are the JNWATR_NCH_4C5F0 symbol itself, converted from the library you are placing them from, so the page and the canvas cannot drift apart

14.5.3 Netlist schematic

Check that the netlist looks OK

In work/

```
make xsch CELL=LELO_EX
cat xsch/LELO_EX.spice
```

14.6 Typical corner SPICE simulation

I’ve made cicsim that I use to run simulations (ngspice) and extract results

14.6.1 Setup simulation environment

Navigate to the `lelo_ex_sky130a/sim/` directory.

Make a new simulation folder

```
cicsim simcell  LELO_EX_SKY130A LELO_EX \
    ../tech/cicsim/cell_spice/template.yaml
```

I would recommend you have a look at the template.yaml file to understand what happens.

14.6.2 Familiarize yourself with the simulation folder

I've added quite a few options to cicsim, and it might be confusing. For reference, these are what the files are used for

File	Description
Makefile	Simulation commands
cicsim.yaml	Setup for cicsim
summary.yaml	Generate a README with simulation results
tran.meas	Measurement to be done after simulation
tran.py	Optional python script to run for each simulation
tran.spi	Transient testbench
tran.yaml	What measurements to summarize

The default setup should run, so

```
cd LELO_EX
make typical
```

14.6.3 Modify default testbench (tran.spi)

Delete the VDD source

Add a current source of 5uA, and a voltage source of 1V to IBNS_-20U

```
IBP 0 IBPS_5U dc 5u
V0  IBNS_20U 0 dc 1
```

Save the current in V0 by adding i(V0) to the save statement in the testbench

Save the voltage by adding v(IBPS_5U) to the save statement

```
.save i(V0) v(IBPS_5U)
```

14.6.4 Modify measurements (tran.meas)

Add measurement of the current and VGS. It must be added between the "MEAS_START" and "MEAS_END" lines.

```
let ibn = -i(v0)
meas tran ibns_20u find ibn at=5n
meas tran vgs_m1 find v(ibps_5u) at=5n
```

Run simulation

```
make typical
```

and check that the output looks okish.

Try to run the simulation again

```
make typical
```

If everything works, then the simulation now should **not** be run. Every time cicsim runs (provided the `sha: True` option is set in `cicsim.yaml`) cicsim will compute a SHA hash of all files (stored in output_tran/*.sha*) that is referenced in the `tran.spi`. Next time cicsim is run, it checks the hash's and does not re-run if there is no need (no files changed).

Sometimes you want to force running, and you can do that by

```
make typical OPT="--no-sha"
```

Often, it's the measurement that I get wrong, so instead of rerunning simulation every time I've added a "–no-run" option to cicsim. For example

```
make typical OPT="--no-run"
```

will skip the simulation, and rerun only the measurement. This is why you should split the testbench and the measurement. Simulations can run for days, but measurement takes seconds.

14.6.5 Modify result specification (tran.yaml)

Add the result specifications, for example

```
ibn:
  src:
    - ibns_20u
  name: Output current
  min: -5%
  typ: 20
  max: 5%
  scale: 1e6
  digits: 3
  unit: uA

vgs:
  src:
```

```
  - vgs_m1
name: Gate-Source voltage
typ: 0.6
min: 0.3
max: 0.8
scale: 1
digits: 3
unit: V
```

Re-run the measurement and result generation

```
make typical OPT="--no-run"
```

Open `results/tran_Sch_typical.html`

14.6.6 Check waveforms

You can either use ngspice, or you can use cicsim, or you can use something I don't know about

Open the raw file with

```
cicsim wave output_tran/tran_SchGtKttTtVt.raw
```

Load the results, and try to look at the plots. There might not be that much interesting happening

14.6.6.1 Searching waveforms

On the left side of the window you'll see a text box in the middle between the filename, and the wave names. This is a regex search field, and you can easily search for waveforms (like `i(v0)`) that you want to find.

Note that the search field uses regular expressions. If you don't know regex, then it's time to learn. I always use the perl regular expression variants.

For example, searching for "i(v0)" won't actually show anything, because the `()` are special characters. "i(v0)" will find it though.

I could search for both ibps and v0 at the same time with `ibps|i\(`, so it's well worth learning.

A great resource is Mastering Regular Expressions

14.7 All corners SPICE simulations

Analog circuits must be simulated for all physical conditions, we call them corners. We must check high and low temperature, high and low voltage, all process corners, and device-to-device mismatch.

14.7.1 Remove Vh and Vl corners (Makefile)

For the current mirror we don't need to vary voltage, since we don't have a VDD.

Open Makefile in your favorite text editor.

Change all instances of "Vt,Vl,Vh" and "Vl,Vh" to Vt

14.7.2 Run all corners

To simulate all corners do

```
make typical etc mc
```

where etc is extreme test condition and mc is monte-carlo.

Wait for simulations to complete.

14.7.3 Get creative with python

Open `tran.py` in your favorite editor, try to read and understand it.

The `name` parameter is the corner currently running, for example `tran_SchGtAmcttTtVt`.

The measured outputs from ngspice will be added to `tran_SchGtAmcttTtVt.yaml`

Delete the "return" line.

Add the following lines (they automatically plot the current and gate voltage)

```
import cicsim as cs
fname = name +".png"
print(f"Saving {fname}")
cs.rawplot(name + ".raw","time","v(ibps_5u),i(v0)" \
  ,ptype="",fname=fname)
```

Re-run measurements to check the python code

```
make typical etc mc OPT="--no-run"
```

You'll see that cicsim writes all the png's. Check with `ls -l output_tran/*.png`.

You'll also notice it will slow down the simulation, so maybe remove the lines from `tran.py` again ;-)

14.7.4 Generate simulation summary

Run

```
make summary
```

Install pandoc if you don't have it

Run

```
pandoc -s   README.md -o README.html
```

to generate a HTML slideshow that you can open in browser. Open the HTML file.

14.7.5 Viewing results without GUI browser

If you're on a system without a browser, or indeed a GUI, then it's possible to view the results in the terminal.

Check if `lynx` is installed, if it's not installed, then

On linux

```
sudo apt-get install lynx
```

On Mac

```
brew install lynx
```

Then

```
lynx README.html
```

14.7.6 Think about the results

From the corner and mismatch simulation, we can observe a few things.

- The typical value is not 20 uA. This is likely because we have a M2 VDS of 1 V, which is not the same as the VDS of M1. As such, the current will not be the same.
- The statistics from 30 corners show that when we add or subtract 3 standard deviation from the mean, the resulting current is outside our specification of +- 5 %.

14.8 Draw Layout

A foundry (the factory that makes integrated circuits) needs to know how we want them to create our circuit. So we need to provide them with a "layout", the recipe, or instruction, for how to make the circuit. Although the layout contains the same components as the schematic, the layout contains the physical locations, and how to actually instruct the foundry on how to make the transistors we want.

Open Magic VLSI

```
cd work
magic ../design/LEL0_EX_SKY130A/LEL0_EX.mag
```

Now brace yourself, Magic VLSI was created in the 1980's. For its time it was extremely modern, however, today it seems dated. However, it is free, so we use it.

14.8.1 Magic VLSI

Try google for most questions, and there are youtube videos that give an intro.

- Magic Tutorial 1
- Magic Tutorial 2
- Magic Tutorial 3
- Magic command reference
- Magic Documentation

Default magic start with the BOX tool. Mouse left-click to select bottom corner, left-click to select top corner.

Press "space" to select another tool (WIRING, NETLIST, PICK).

Type "macro help" in the command window to see all shortcuts

Hotkey	Function
v	View all
shift-z	zoom out
z	zoom in
x	look inside box (expand)
shift-x	don't look inside box (unexpand)
u	undo
d	delete
s	select
Shift-Up	Move cell up
Shift-Down	Move cell down
Shift-Left	Move cell left
Shift-Right	Move cell right

14.8.2 Add transistors

Open Cell -> Place Instance. Navigate to the right transistor.

Place it. Hover over the transistor and select it with 's'. Now comes a bit of tedious thing. Select again, and copy. It's possible to align the transistors on-top of eachother, but it's a bit finicky.

Place all transistors on top of each other as shown below in the picture.

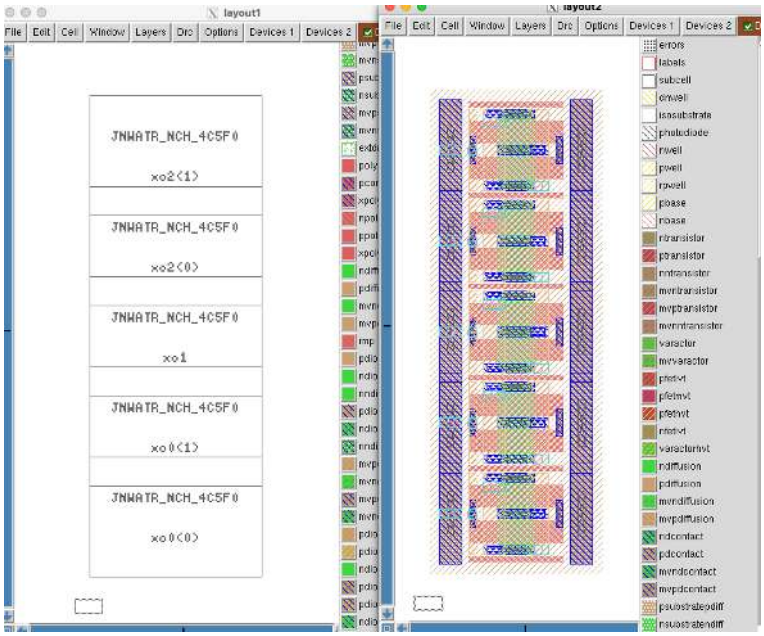


Figure 2: Magic with the transistor instances stacked on top of each other, shown as boxes on the left and with layers drawn on the right

14.8.3 Place devices

You will find that one of the more time consuming things with analog layout is to place the devices, and to follow the design rules from foundry. I detest tedious work. As such, I've tried for the past 25 years to simplify analog layout. I've not finished yet, but maybe you'll find some of the scripts useful.

Note that the command below will override all your hard work ;-)

```
cd work
make xsch
cicpy sch2mag LELO_EX_SKY130A LELO_EX
```

14.8.4 Add Ground

In the command window, type

```
see no *
see viali
see locali
see m1
see via1
see m2
```

Make a box around the layout by left clicking bottom left, and right clicking top right. Press 'x' to expand.

Change grid to 1 um. Set "Window->Snap to grid on"

Select a 1 um box below the transistors and paint the rectangle with locali (middle click on locali)

Change to the 'wire tool' with spacebar. Set "Window-> Snap to grid off"

Connect guard rings to ground.

Press the top transistor 'S' and draw all the way down to connect all of the transistors' source terminals. Use 'shift-right click' to change layer down

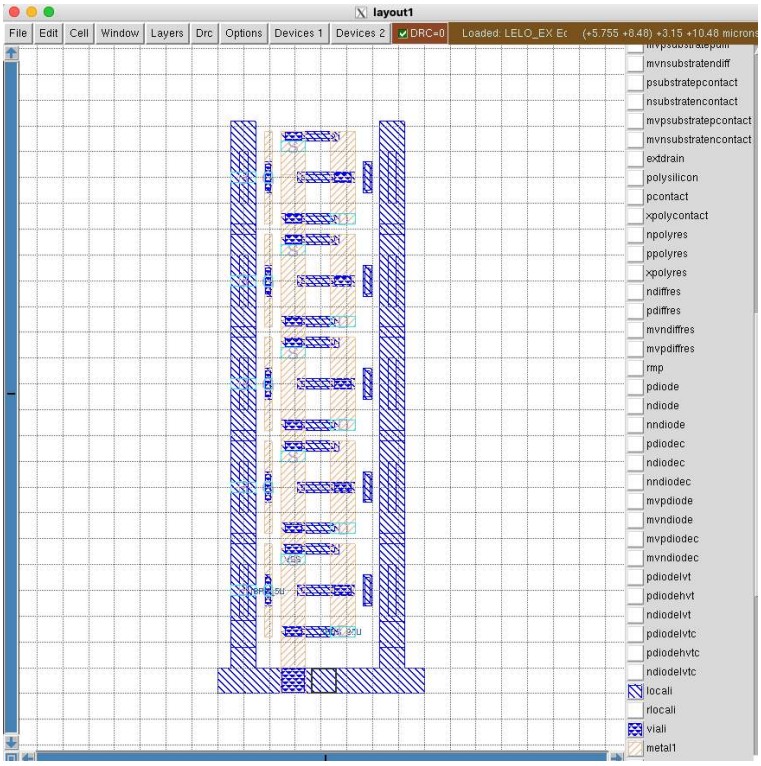


Figure 3: The source terminals of all transistors connected down to the ground rail, DRC clean

14.8.5 Route Gates

Press “space” to enter wire mode. Left click on the top gate to start a wire, and right click to end the wire.

The drain of M1 transistor needs a connection from gate to drain. We do that for the middle transistor. Change to the box tool (spacebar a few times). Create a box that matches the locali. Connect the drain to the gate in locali.

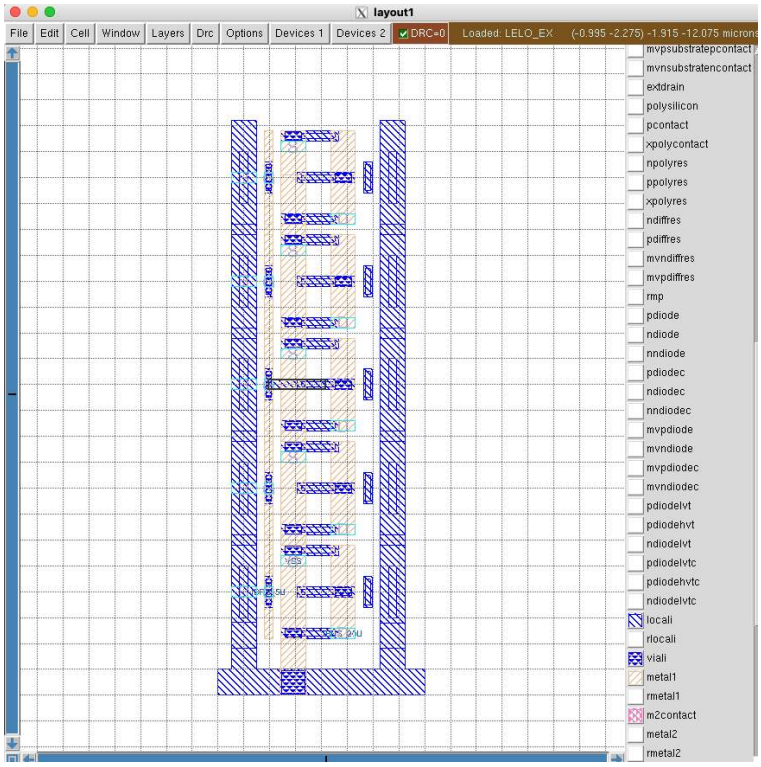


Figure 4: The gates routed together, with the gate to drain connection of M1 made in locali

14.8.6 Drain of M2

Use the wire tool to draw connections for the drains.

To add vias you can do “shift-left click” to move up a metal, and “shift-right click” to go down.

It’s a very good idea to have direction rules for metal layers. I would recommend that you route metal1 vertical, metal2 horizontal, metal3 vertical etc. For locali it’s usually all over the place.

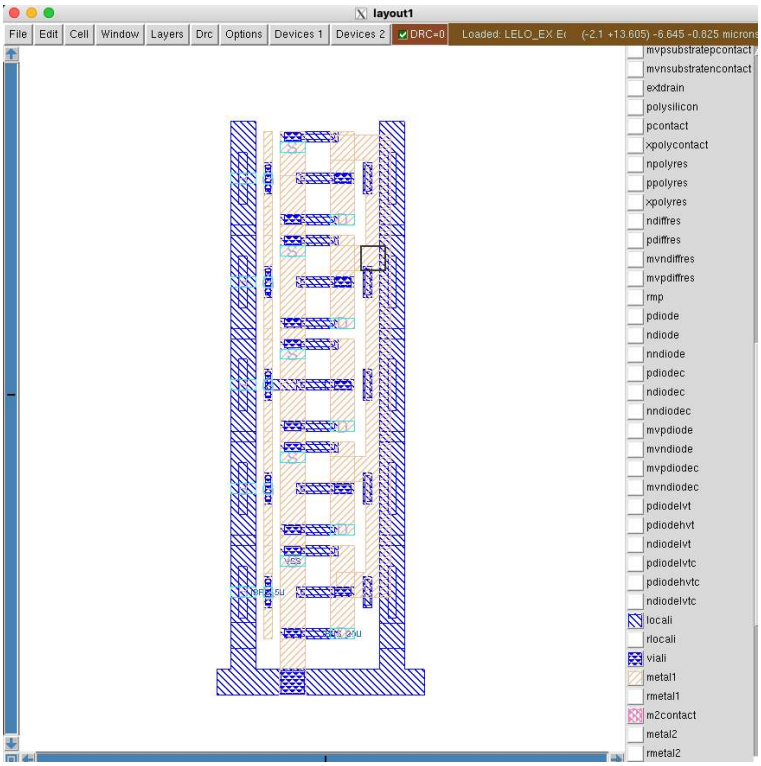


Figure 5: The drain connections routed with the wire tool, using vias to change metal layer

14.8.7 Add labels

All ports must be named (IBPS_5U, IBNS_20U, VSS). The cicpy script may add ports, but not necessarily where you want them.

Select a box on a metal, and use "Edit->Text" to add labels for the ports. Select the port button.

14.9 Layout verification

The DRC can be seen directly in Magic VLSI as you draw.

To check layout versus schematic navigate to work/ and do

```
make cdl lvs
```

Remember to save the layout first.

If you've routed correctly, then the LVS should be correct.

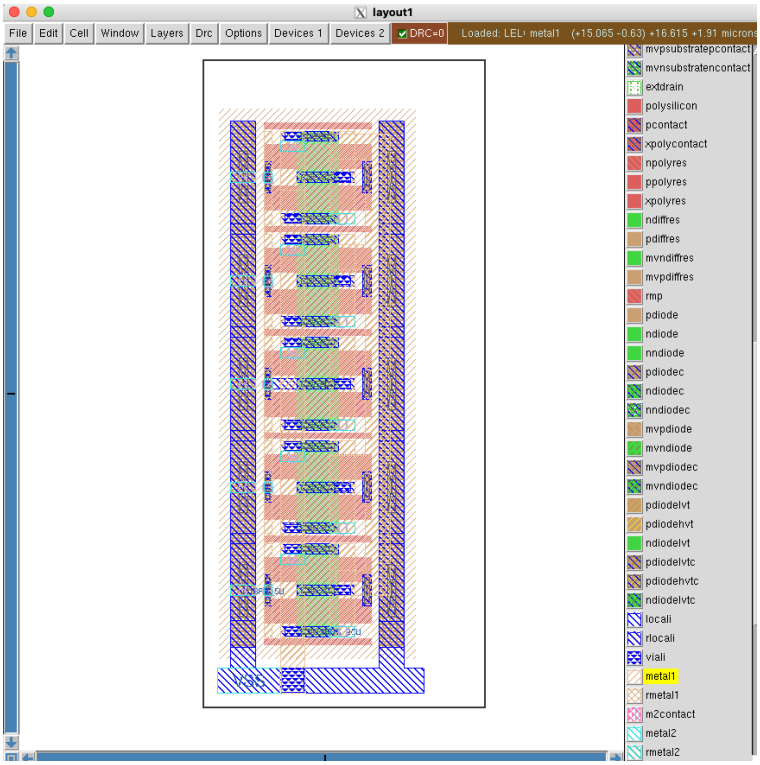


Figure 6: The completed LELO_EX layout with labelled ports, ready for the LVS check against the schematic

14.10 Extract layout parasitics

With the layout complete, we can extract parasitic capacitance.

```
make lpe
```

Check the generated netlist

```
cat lpe/LELO_EX_lpe.spi
```

14.11 Simulate with layout parasitics

Navigate to sim/LELO_EX. We now want to simulate the layout.

The default `tran.spi` should already have support for that.

Open the Makefile, and change

```
VIEW=Sch
```

to

```
VIEW=Lay
```

14.11.1 Typical simuation

Run

```
make typical
```

14.11.2 Corners

Navigate to sim/LELO_EX. Run all corners again

```
make all
```

14.11.3 Simulation summary

Open `summary.yaml` and add the layout files.

```
    - name: Lay_typ
      src: results/tran_Lay_typical
      method: typical
    - name: Lay_etc
      src: results/tran_Lay_etc
      method: minmax
    - name: Lay_3std
      src: results/tran_Lay_mc
      method: 3std
```

Run summary again

```
make summary
pandoc -s  README.md -o README.html
```

Open the README.html and have a look a the results. The layout should be close to the schematic simulation.

14.12 Make documentation

Make a file (or it may exists) `design/LELO_EX_SKY130A/LELO_EX.md` and add some documentation of what you've made.

Add the simulation results to your git repository to keep track

```
git add sim/LELO_EX/results/*.html
git add sim/LELO_EX/README.md
```

14.13 Edit info.yaml

Finally, let's setup the `info.yaml` so that all the github workflows run correctly.

Mine will look like this.

You need to setup the url (probably something like `<your username>.github.io`) to what is correct for you.

I've added the doc section such that the workflows will generate the docs.

The sim is to run a typical simulation.

```yaml
library: LELO_EX_SKY130A
cell: LELO_EX
author: Carsten Wulff
github: wulffern
tagline: The answer is 42
email: carsten@wulff.no
url: wulffern.github.io
doc:
  libraries:
    LELO_EX_SKY130A:
      - LELO_EX
```

14.14 Setup github pages

Go to your GitHub repository (repo). Press Settings. Press Pages. Choose source under Build and Deployment -> GitHub Actions

Wait for the workflows to build. And check your github pages. Mine is https://wulffern.github.io/lelo_ex0_sky130a/.

14.15 Frequently asked questions

Q: My GDS/LVS/DRC action fails, even though it works locally.

Sometimes the reference to the transistors in the magic file might be wrong. Open the .mag file in a text editor and check. The correct way is

```
use JNWATR_NCH_4C5F0  JNWATR_NCH_4C5F0_0 ../LELO_ATR_SKY130A
```

It's the last `../LELO_ATR_SKY130A` that sometimes is missing.

14.16 Summary

The one-page version of this chapter:

- One cell end to end: schematic in xschem, a symbol, a testbench, and a cicsim run with a results table
- The names are not cosmetic: instances, ports and directories follow the aicex conventions the tools expect
- By the end you have run corners from one command - the pattern every later week repeats

The Project 15

> AIC is likely one of the most rewarding courses I've attended at NTNU. It gave me a lot of valuable knowledge on different types of circuits, IC design workflows and open source EDA tools that I greatly appreciate. It is also one of the most challenging courses, due to the amount of effort and time I had to spend in order to figure things out. - Tord, AIC2025

> A fantastic project that just might turn your world upside down, push you to re-evaluate your life choices, and stare briefly into the existential void... all while being deeply enjoyable and engaging! - Domen, AIC2025

15.1 The challenge . . . 235
15.1.1 Systems-on-chip have complex regulator systems 236
15.1.2 Leakage current varies orders of magnitude over temperature 236
15.1.3 Why we would like to know the temperature on die 237
15.1.4 How to measure temperature? 237
15.2 The Project 238
15.2.1 Specification 239
15.2.2 Figure of Merit . . . 240
15.2.3 Grade 240
15.2.4 Milestone 0: The tutorial 241
15.2.5 Milestone 1: The bandgap 241
15.2.6 Milestone 2: The oscillator 242
15.2.7 Milestone 3: The measurement 243
15.2.8 Milestone 4 (Optional): The physical design 244
15.2.9 Milestone 5: The Report 244
15.2.10 Milestone 6 (Optional): The Tapeout 245
15.3 Summary 245
15.4 Would you like to know more? 245

Keywords: Temperature Sensor, Leakage, Regulator, Specification, FOM, Milestones, Tapeout

The project will walk you through the full analog/digital design process. From specification all the way to a finished layout, and a potential tapeout.

The project is not easy, it's rather hard. You'll experience frustration, despair, epic wins, epic losses, stress, collaboration, and you will figure out whether you love analog design, or digital design or neither.

I promise that the design project closely matches how we would develop a circuit in industry.

I ask a lot of you on the project, as such, it accounts for 45 % of the grade, and is maybe the thing that you'll learn the most from.

In this document I'll go through the problem (what we're trying to solve), and the milestones that we'll use along the way.

15.1 The challenge

The assignment is to Design a temperature sensor

But why? I'll try to explain.

15.1.1 Systems-on-chip have complex regulator systems

See the example in Figure 2, redrawn from Nordic Semiconductor's nRF54L15 product specification.

VDD is the supply from the battery (1.7 V - 3.6 V). While the DECD, DECA and DECRF are the low voltage supplies for the digital, analog and radio.

The VREGMAIN has both a DC/DC, and a LDO. We'll learn about those in the course. For now it's sufficient to know that the DC/DC converts power drawn on the low supply (DECA, DECD, DECRF) to power drawn from the high supply (VDD), while the LDO has the same current on low supply as the high supply, but the voltage is different.

You will learn in the course that the typical systems inside VREGMAIN are complicated, and sometimes complex, analog circuits.

From the data-sheet you'll see that the lowest power state is about 700 nA, while the highest power state is about 10 mA. The high power state is 14 thousand times higher than the low power state!

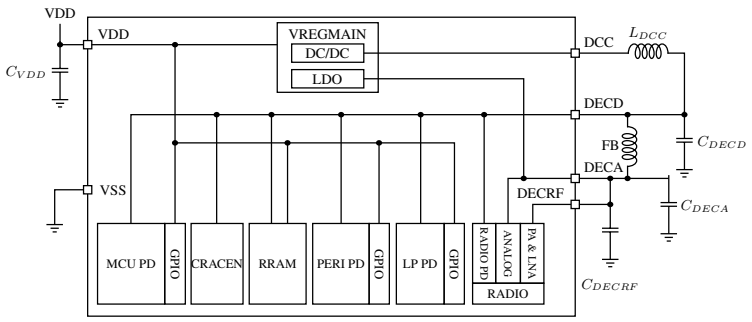


Figure 2: Power system of nRF54L15. Redrawn from Nordic Semiconductor, nRF54L15 Product Specification

Those numbers are the total current consumption. That includes switching currents from digital, analog bias currents, and leakage currents. In modern technologies, because of the low threshold voltage, the leakage currents can be a large part of the total current budget.

15.1.2 Leakage current varies orders of magnitude over temperature

The sub-threshold leakage current in a MOSFET is

$$I_{leak} = I_0 e^{-V_{th}/nV_T} \left(1 - e^{-V_{ds}/V_T}\right)$$

The change in leakage as a function of temperature is rather complicated. The $V_T = \frac{kT}{q}$ factor is easy, but both I_0 and V_{th} have a complicated relation to temperature.

In Figure 3 you can see the leakage simulation (from http://analogicus.com/lelo_aic_sky130a/)

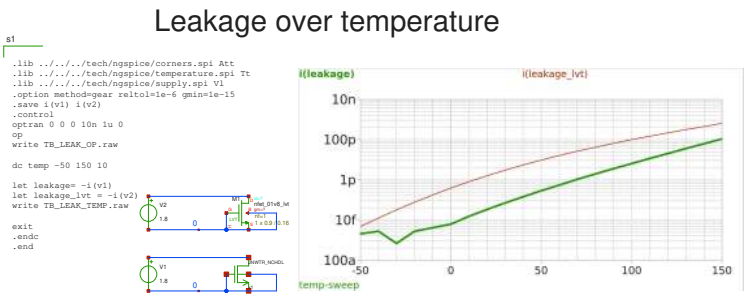


Figure 3: Leakage simulation

Based on the previous curves we could run a thought experiment.

- Assume 1 pA at 25 C, and 1 nA at 125 C, per logic cell
- Assume 100 million logic cells
- Leakage at 25 C => 100 uA
- Leakage at 125 C => 100 mA !!!

15.1.3 Why we would like to know the temperature on die

Expanding on the thought experiment.

- Assume we use 1 % of the load current for the regulator
- At 25 C => 1 uA for LDO
- At 125 C => 1 mA for LDO

It's insanely difficult to design a regulator that is efficient across the full range of leakage currents at any temperature.

It would be good if we could know temperature.

15.1.4 How to measure temperature?

There are a multitude of ways to make a temperature sensor. In [1] they used a leakage based digital ring oscillator, in [2] they used a two-transistor MOSFET sensing element, in [3] they had a more complicated sigma-delta ADC sensing bipolar transistors.

The design of a temperature sensor is more difficult than you think. As such, I would suggest that you don't go too crazy in your choice of sensor. So far, none has gotten close to the finish line with a sigma-delta ADC based sensor.

In the previous years of Advanced Integrated Circuits most groups have chosen an architecture similar to [4] Fig. 2. I would recommend you do the same, and that's what I'll target in the milestones.

The principle of the temperature sensor is: 1) Create a current that is proportional to temperature (Lecture 3), 2) Convert current to frequency with a relaxation oscillator (Lecture 9). 3) Check the frequency to read the temperature.

In Figure 4 below you can see an illustration of the temperature sensor.

A bandgap circuit is used to make a current that is proportional to absolute temperature (I_{PTAT}) and a voltage that is complementary to absolute temperature (V_{CTAT}). A relaxation oscillator converts the current and voltage into a frequency (f_{OSC}). A digital finite-state-machine and a counter converts the frequency to a digital value that is proportional to temperature.

I've made an example temperature sensor at lelo_temp_sky130a. Feel free to steal ideas, and circuits, from that design.

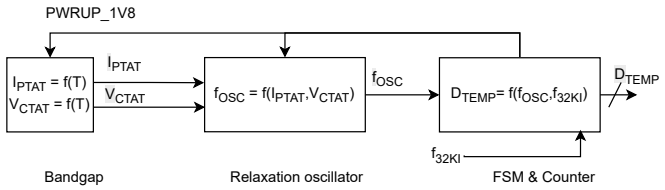


Figure 4: Illustration of the temperature sensor

15.2 The Project

I'm going to lead you through the design of a, to you, complicated mixed signal circuit design. You will despair, you will not understand, but you will learn.

In order to make the problem possible to learn, we're going to focus on one milestone at a time. I hope that will enable you to not drown before we get started.

An illustration of the milestones can be seen in Figure 5.

The first milestone is the design of the bandgap circuit, a pure analog design. The second milestone is the design of the relaxation oscillator. The third milestone is how you measure the frequency of the oscillator, and is usually done in SystemVerilog.

The fourth milestone is optional, but you can't get an A in the course if you don't have some points from the layout and parasitic simulation milestone.

The fifth milestone is an individual report. I will force you to work in groups. As such, it may be that some contribute more than others. To ensure that the grading is fair, the report will be individual. It's OK to share figures, tables, and so on, but the PDF shall be written by you and you alone.

The sixth and last milestone is the tapeout on <tinytapeout.com>. It's optional, and not everyone will get to that stage.

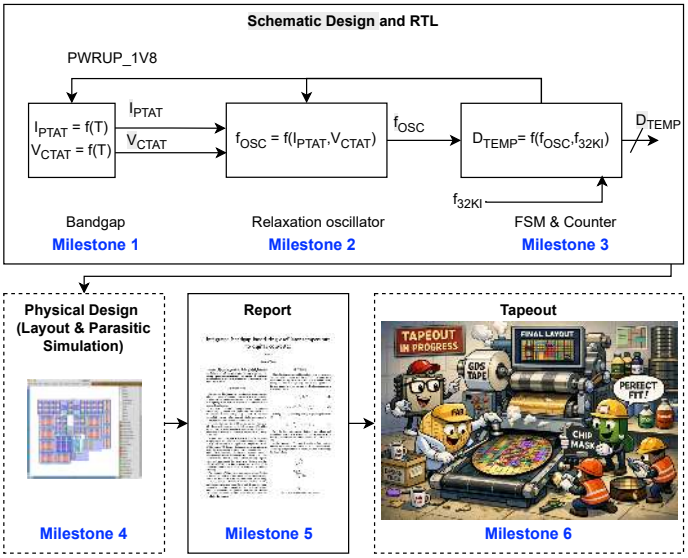


Figure 5: Project overview

15.2.1 Specification

The temperature sensor shall be designed to fit the specification below.

Key	Parameter	Value	Unit	Description
Area	Area	< 15000	um^2	Must fit in 161 um x 111 um tiny tapeout 1x1 block, which is 17871 um^2
Tc	Conversion time	< 30	us	Analog should only be active for one 32768 Hz period
Ts	Sample period	100	ms	One conversion every 100 ms, so 10 samples per second
Ileak	Leakage current	< 1	nA	Typical temperature (25 C)
Iact	Active current	< 100	uA	Typical temperature (25 C)
Iavg	Average current	< 50	nA	Active current x conversion time/sample rate + leakage current. Typical temperature (25 C)
Kerrone	Accuracy 0 - 70C	+-10	C	One temperature (25C) calibration
Kerrtwo	Accuracy 0 - 70C	+-5	C	Two temperature (25C, 85C) calibration

15.2.2 Figure of Merit

A figure of merit allows us to compare circuits with different performance specifications, and try to answer the question. Which one is better?

The figure of merit for our temperature sensor will be

$$FOM = \left(\frac{T_c}{T_s} I_{act} + I_{leak}\right) K_{errtwo} \text{ [AK]}$$

The bracket is exactly I_{avg} from the table above, so the figure of merit is *average current times error* — what the sensor costs you multiplied by how wrong it is. Lower is better, and there is no way to win by being cheap and inaccurate or by being accurate and expensive.

It is worth checking that the specification is self consistent before designing to it, because specifications often are not. With the numbers in the table, $\frac{30\ \mu s}{100\ ms} \times 100\ \mu A + 1\ nA = 31$ nA, comfortably inside the 50 nA the table allows for I_{avg}. Notice also where the current goes: the analog block draws 100 uA but only for 30 us in every 100 ms, so duty cycling turns it into 30 nA and the leakage, at 1 nA, is almost an afterthought. That ratio is why the conversion time is specified at all.

15.2.3 Grade

Milestones are important, and Milestone 0 - 5 will count towards your final grade!

That means, you have to start working right away.

The points have been designed such that it's impossible to get an A without getting some points on the layout

Week	Deadline	Milestone	Task	Condition for more than 0 points	Possible Points
4	2026-01-23	M0	Complete tutorial	Link on blackboard	5
7	2026-02-13	M1	Design a circuit that can convert a temperature into a current and voltage	Description of the circuit on github docs	5
10	2026-03-06	M2	Design a circuit that can convert a temperature into a frequency	Description of the circuit on github docs. Demonstrate that it works	10
13	2026-03-27	M3	A verilog testbench that can convert a frequency into a digital value	Description of the testbench on github docs. Demonstrate that it works	10
16	2026-04-17	M4	Layout of your circuit	DRC/LVS/GDS passing on github	20

Week	Deadline	Milestone	Task	Condition for more than 0 points	Possible Points
18	2026-05-01	M5	Individual report	Uploaded to Inspera	48
?		M6	Tapeout	None	0
		Coolness	Extra points that I may choose to award		10
		Total			108

15.2.4 Milestone 0: The tutorial

Goal: force you to install the tools, and get you started.

Follow Sky130nm Tutorial

Delivery: Submit link to your github repository on blackboard

For example, my repository: LELO_EX_SKY130A

The exercise will teach you the skills you need to do the project

15.2.5 Milestone 1: The bandgap

Goal: Create a circuit that can transform a temperature on the integrated circuit to a current proportional to temperature (PTAT), and a voltage complementary to temperature (CTAT).

For this purpose it's common to use "Bandgap" circuits. We'll learn about them in the course, but if you don't want to wait then you should read https://analogicus.com/aic2027/references_and_bias and ask me questions in reference and bias lecture.

In the git repository for your group you'll create schematics for the bandgap circuits, and you'll make test-benches to check that the bandgap circuit works.

If you don't know what you should simulate and verify, it's good to have a chat with ChatGPT. See https://chatgpt.com/share/69481b11-8830-8007-9986-c9e41d735cfc.

Or check my test-benches at https://github.com/wulffern/lelo_temp_sky130a/tree/main/sim/LELOTEMP_BIAS_IBP

Delivery: Link to your github repository with a description of how the bandgap works.

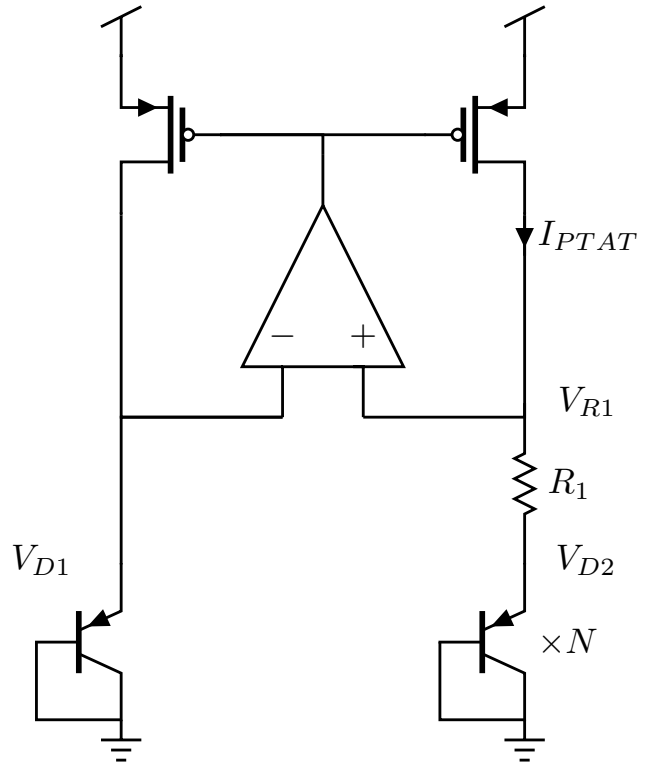


Figure 6: PTAT current generator, where an op amp forces equal voltages across the two bipolar branches so that ΔV_{BE} over R_1 sets I_{PTAT}

15.2.6 Milestone 2: The oscillator

Goal: Use the PTAT current, and the CTAT voltage an create a oscillator.

One way is to charge a capacitor with the current, and have a comparator trigger when the voltage on the capacitor reaches a voltage (CTAT voltage for example). When the comparator triggers, then we can reset the capacitor.

This is similar to what group 7 did last year (one of the groups got all the way to tapeout). One difference, though, is that group 7 used VDD/2 as the reference. I would recommend you use the CTAT voltage from the bandgap instead. That way, the oscillation frequency is independent (to first order) from the VDD.

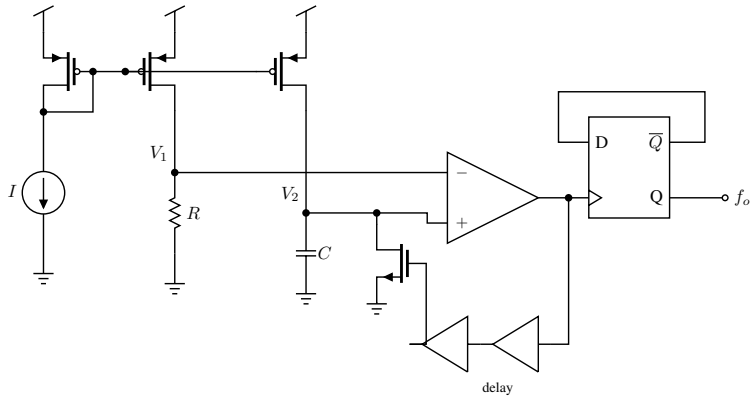


Figure 7: Relaxation oscillator, where a current charges capacitor C until the comparator trips against the reference voltage V_1 across R, and the flip-flop resets the capacitor

Delivery Link to your github repository with description on how your oscillator works. There should be proof on how it works.

15.2.7 Milestone 3: The measurement

Goal: Measure the frequency of the oscillator.

In the system we can assume we have an accurate 32768 Hz clock source. One way to find the frequency is to run the oscillator for a fixed number of clock cycles on the 32768 Hz clock, and have a counter that can count the output pulses.

Assume we counted 128 clock cycles over 2 clock periods of the 32768 Hz clock. That would mean the frequency of the oscillator was approximately 2.09 MHz. Once we have the frequency we can calculate the temperature.

I would recommend that you write in verilog the system to start the oscillator, count for a number of 32768 Hz clock cycles, and transform the frequency into a temperature.

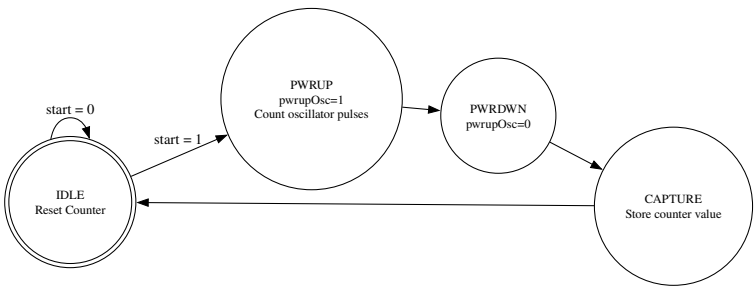


Figure 8: State machine that measures the oscillator frequency, cycling IDLE, PWRUP (count oscillator pulses), PWRDWN and CAPTURE (store counter value)

Delivery: Link to your github repository where you describe how you measure the frequency of the oscillator.

15.2.8 Milestone 4 (Optional): The physical design

Goal: Do the physical layout of your oscillator, and prove that it still works with the layout parasitics.

If you do the layout, then your design must fit within a digital 1x1 tinytapeout block. I've made a template at https://github.com/wulffern/lelo_temp_sky130a/blob/main/design/LELO_TEMP_SKY130A/tt_block_1x1_pg.mag that you can use. If the design does not fit within that space, then you won't be able to tapeout.

When your design is complete, then the DRC, LVS, GDS actions should be passing on github.

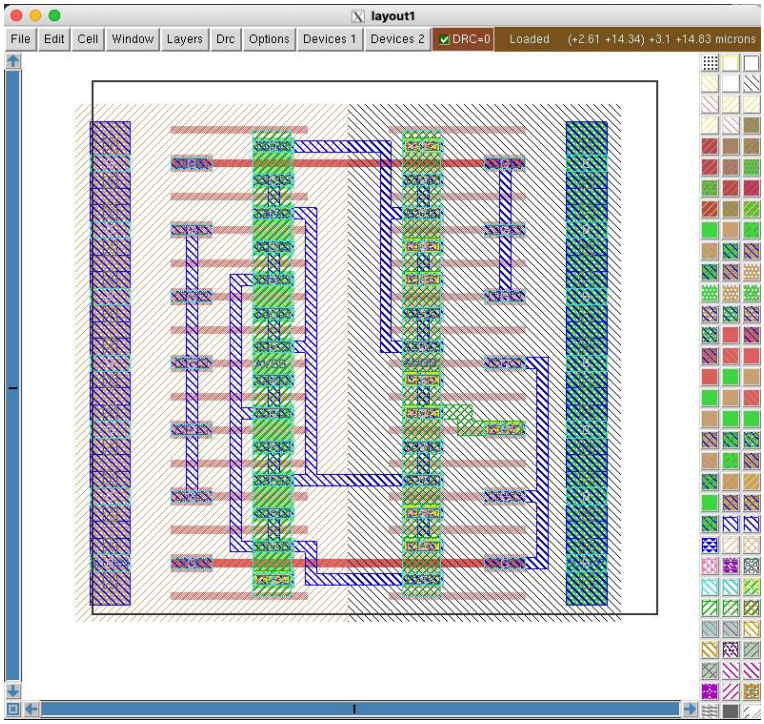


Figure 9: Physical layout of a circuit in the Magic layout editor, with the DRC error count shown as zero in the toolbar

Delivery: Link to your github repository with passing GDS, DRC, LVS actions.

15.2.9 Milestone 5: The Report

Goal: Write a report

Delivery: A PDF copy of the report in Inspera. You'll all write an individual report. The report shall be in the IEEE template.

See further details in https://analogicus.com/aic2027/how_to_write_a_project_report

15.2.10 Milestone 6 (Optional): The Tapeout

Target TTSKY26b tapeout (June 2026) on https://tinytapeout.com/chips/

Those students that follow the course at NTNU will be able to tapeout if the design is complete. I've gotten Nordic Semiconductor to sponsor the tapeout for 2026.

15.3 Summary

The one-page version of this chapter:

- The project is the course: design a block in an open PDK, simulate it over corners, and document the evidence
- Start from the template repositories and the cic flow on day one - infrastructure debt compounds
- Work like an engineer: git for everything, scripts over clicks, claims backed by simulation
- The report is the deliverable; the next chapter's writing guide is not optional reading

15.4 Would you like to know more?

Precision CMOS temperature sensors, the book chapter to read before starting [3]

Recent low-power sensors, for what the state of the art costs in area and current [2] [1] [4]

[1] Z. Tang, Y. Fang, Z. Shi, X.-P. Yu, N. N. Tan, and W. Pan, "A 1770- μ m2 leakage-based digital temperature sensor with supply sensitivity suppression in 55-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 3, pp. 781–793, 2020, doi: 10.1109/JSSC.2019.2952855.

[2] S. Jeong, Z. Foo, Y. Lee, J.-Y. Sim, D. Blaauw, and D. Sylvester, "A fully-integrated 71 nW CMOS temperature sensor for low power wireless sensor nodes," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 8, pp. 1682–1693, 2014, doi: 10.1109/JSSC.2014.2325574.

[3] M. A. P. Pertijs, A. Niederkorn, X. Ma, B. McKillop, A. Bakker, and J. H. Huijsing, "A CMOS smart temperature sensor with a 3/spl sigma/ inaccuracy of /spl plusmn/0.5/spl deg/c from -50/spl deg/c to 120/spl deg/c," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 2, pp. 454–461, 2005, doi: 10.1109/JSSC.2004.841013.

[4] J.-H. Park, J.-H. Hwang, C. Shin, and S.-J. Kim, "A BJT-based temperature-to-frequency converter with +- 1°c (3σ) inaccuracy from - 40 c to 140 c for on-chip thermal monitoring," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 10, pp. 2909–2918, 2022, doi: 10.1109/JSSC.2022.3182708.

16 Measured Silicon

Keywords: Tapeout, Tiny Tapeout, PTAT, Time-domain readout, Temperature chamber, Transfer curve, INL, Calibration, Allan deviation, Quantisation, Dither, Bernoulli statistics, Random telegraph noise

16.1 Measured Silicon . 247
16.1.1 The same physics, read out two ways . 247
16.2 What the silicon does 249
16.2.1 Against a real reference 249
16.2.2 What calibration buys 251
16.3 The clock in the loop 252
16.3.1 GR07's output is quantised, and noise is what rescues it . . 252
16.3.2 When averaging stops helping 254
16.3.3 Two sensors, one event 255
16.4 One trap 257
16.4.1 GR06 does not drift, it switches 257
16.5 Summary 259
16.6 Would you like to know more? 259

16.1 Measured Silicon

The previous chapter set you a project. This one is what happened when two groups finished it.

In the spring of 2025 two groups of three students designed temperature sensors in this course, and both went to a shuttle: Tiny Tapeout project 258, `tt_um_jnw_wulffern`, on ttsky25a in sky130. The chips came back. This chapter is what they do.

It is here for a specific reason. Everything else in this book is either theory or somebody else's measurement. This is the one chapter where the circuit was designed by students at your stage, on the tools you are using, and then measured against a temperature chamber - and where the interesting results are not the ones the designers were aiming at.

16.1.1 The same physics, read out two ways

Both sensors are the same idea, and it is the idea the project chapter asks you to build: a current proportional to absolute temperature (PTAT) charging a capacitor, with a comparator watching the ramp. The current comes from the difference between two diode voltages at different current densities, which is the bandgap core of the references chapter,

$$I(T) = \frac{kT}{qR} \ln N$$

and the time to charge C up to a threshold V_{ref} is therefore

$$t = \frac{V_{ref} C}{I(T)} \propto \frac{1}{T}$$

The time falls as the temperature rises, so the *rate* $1/t$ rises with absolute temperature. Everything measured below is done in the

rate domain for that reason: it is the quantity that should be a straight line through the origin.

Both groups built that PTAT core the same way - a pnp of unit area against eight of them, an amplifier forcing the two branch voltages equal, and a resistor setting the current - but not with the same circuit. GR07 puts three RPPO16 in series and mirrors the result straight out, one to one. GR06 uses an RPPO8 and an RPPO4 and then divides its current down through two further mirrors, an NMOS pair ten wide and a PMOS pair ten wide, before it reaches the capacitor. That is roughly a hundredfold division, and it is most of why GR06 runs six times slower on a capacitor a quarter the size.

Both also take their comparator threshold from a resistive divider across the supply rather than from anything absolute: GR07 taps one resistor up from ground in a string of four, so $V_{ref} = V_{DD}/4$; GR06 taps one of three, so $V_{ref} = V_{DD}/3$. Neither threshold is a bandgap voltage. The ratio $V_{ref}/I(T)$ that sets the time therefore carries the supply in it, which is worth remembering before reading any absolute number below as a property of the sensor alone.

Where the two designs really part company is what closes the loop, and that turns out to matter far more than anything in the analogue.

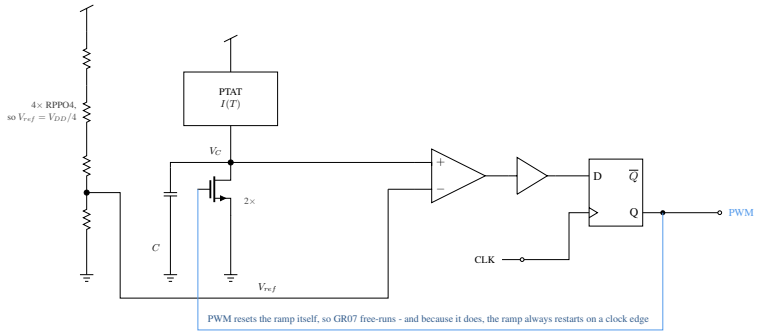


Figure 1: GR07, by Reidar Arne Eidsvik Nerheim, Pol Batalle Largo and Tord Olsen Sætermo, drawn from the taped-out netlist. The comparator output is buffered into a D flip-flop clocked by the project clock, and the flip-flop's output both leaves the chip as PWM and turns on the two NMOS that short the ramp capacitor. The loop closes through the chip, so the sensor free-runs and its period is the observable

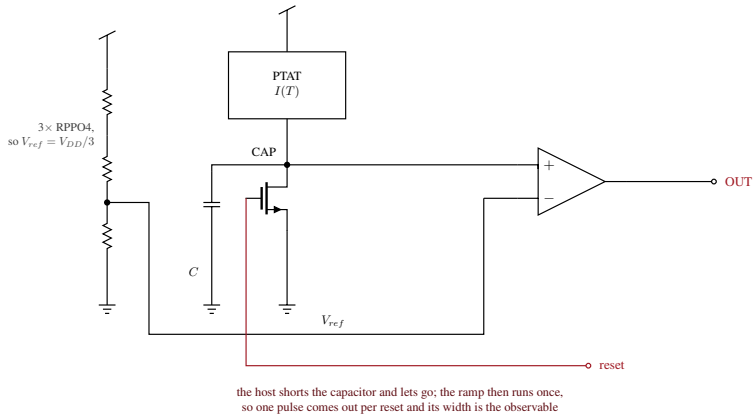


Figure 2: GR06, by Gabin Sbaffi, Erik K. Jensen and Renate Klemetsdal. The same front end, and the same NMOS shorting the capacitor - but its gate is a chip input. The host asserts reset, the capacitor discharges, and on release the ramp runs once. One pulse per stimulus, and its width is the observable. Nothing in this circuit is clocked

At room temperature GR07 free-runs at about 910 kHz - a period near 1.1 µs - so half a second of capture contains half a million periods. GR06 only produces a pulse when the host asks for one, and the host asks about 4.5 thousand times a second, so the same half second contains about 2 300 pulses. GR07 therefore delivers roughly two hundred times more events per measurement, which you would expect to make it the better sensor. Hold that thought.

16.2 What the silicon does

16.2.1 Against a real reference

Everything you can do on a bench calibrates a sensor against itself. To find out whether either sensor is *right*, rather than merely repeatable, you need an outside opinion. A Vötsch temperature chamber stepping from 5 °C to 70 °C in 5 K steps, logging its own probe beside both sensors, supplies one. Each point below is the mean of the last 90 s of a dwell, after both the oven and the die have stopped moving.

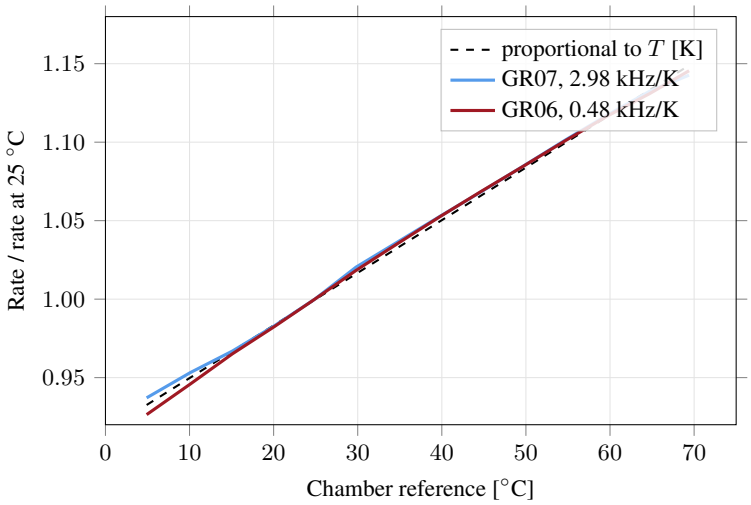


Figure 3: Both sensors against the chamber's own probe, each rate divided by its own value at 25 °C. The dashed line is what a current strictly proportional to absolute temperature would do. Both lie on it: the physics works, and the two designs agree with each other despite a sixfold difference in rate

Figure 3 is the result the students were designing for, and it is a good one. Two independently designed PTAT cores, built by different people from the same principle, land on the ideal line and on each other. GR07 gives 2.98 kHz/K on 907 kHz; GR06 gives 0.48 kHz/K on 142 kHz. Those are 0.33 %/K and 0.34 %/K - the same fractional slope, which is what "proportional to absolute temperature" means.

Figure 4 is where the two part company. Take the straight line away and both are left with about 1.3 K of something. Now allow one more term - the $T \ln T$ curvature the references chapter warns about, which comes from the temperature dependence of the saturation current in the very diodes that make the PTAT current.

For GR06 that removes almost all of it: 1.33 K peak becomes 0.23 K, and the residual goes flat. GR06's error *is* bandgap curvature, and it is the textbook shape at the textbook size.

For GR07 the same term barely helps - 1.61 K becomes 1.04 K - and what is left still wanders. Whatever limits GR07 is not curvature.

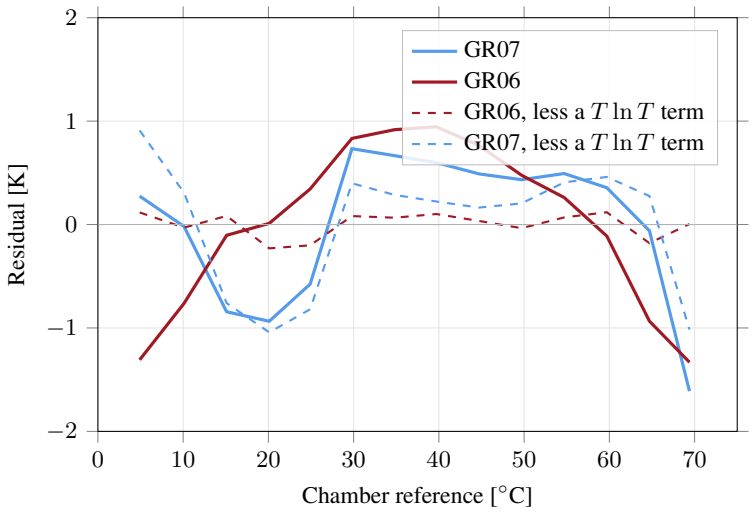


Figure 4: What is left of each transfer after the best straight line, and after also allowing a $T \ln T$ term - the bandgap curvature the references chapter warns about. Almost all of GR06's residual is that curvature. Almost none of GR07's is

One caveat on the identification. Over a 65 K span $T \ln T$ and T^2 are nearly collinear, and fitting either removes the same amount. The data cannot tell you which functional form it is; what it can tell you is that GR06's residual is smooth curvature of the size and sign a bandgap predicts, and that GR07's is not smooth curvature at all.

16.2.2 What calibration buys

The question a product actually asks is not "how linear is it" but "how many oven visits must I pay for". Every calibration temperature costs money in production, so the interesting curve is error against number of trim points.

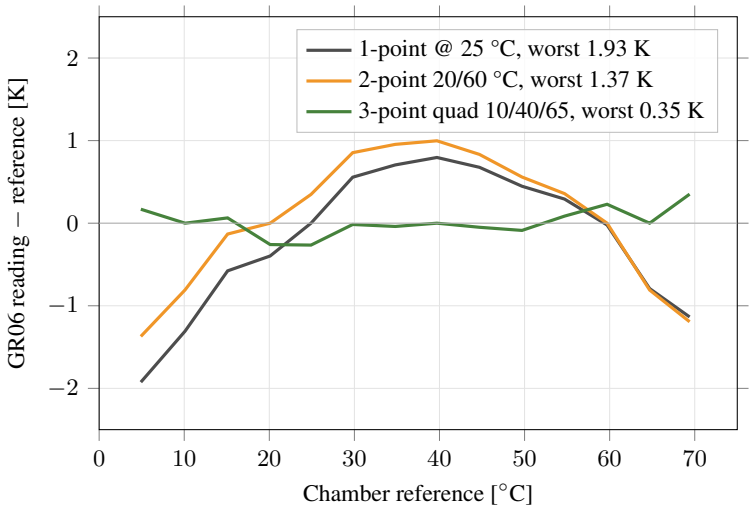


Figure 5: GR06's error after calibrating at one, two and three temperatures. Each extra trim point buys accuracy, ending at ±0.35 K over 5-70 °C. This is what a well-behaved sensor looks like

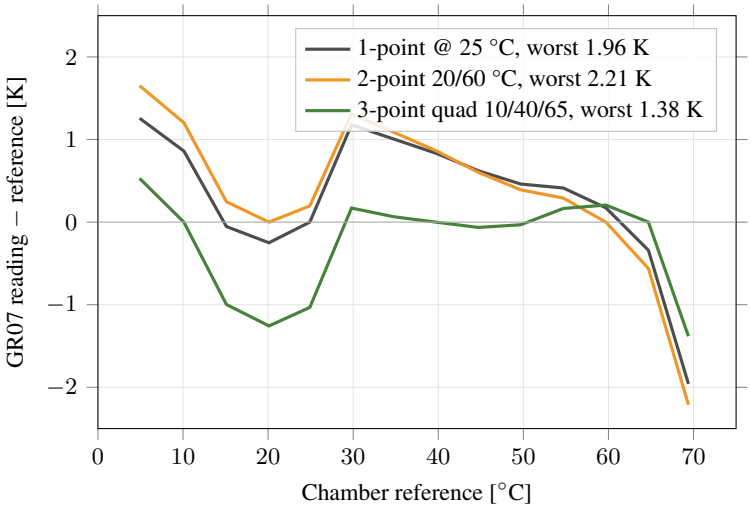


Figure 6: The same for GR07, which gets worse *going from one point to two*

GR06 does what you would hope: 1.93 K with one point, 1.37 K with two, 0.35 K with three. That is a normal, well-behaved sensor, and ±0.35 K over a 65 K span from three trim points is a respectable number for a first silicon by three students.

GR07 goes 1.96 K, then 2.21 K, then 1.38 K. Getting worse when you give it more information looks like a mistake in the analysis. It is not. A two-point calibration corrects a *slope*, and GR07's error is not a slope. Trimming the line just pivots it and drops the residual somewhere else.

To see what GR07's error actually is, you have to look at what its output is made of.

16.3 The clock in the loop

16.3.1 GR07's output is quantised, and noise is what rescues it

Look again at Figure 1. The comparator trips whenever it trips, but its output reaches the reset transistors through a flip-flop, and a flip-flop can only change at a clock edge. Every GR07 period is therefore a whole number of 64 MHz clock cycles, and nothing in between is representable.

That is not a small quantum. GR07's period is about seventy clock cycles at room temperature and falls by roughly one cycle for every five kelvin of warming, so **one clock cycle is about 4.75 K** - three

times the whole calibrated error of Figure 6, and larger than the raw uncalibrated error of Figure 4.

A noiseless version of this circuit would be a plain 4.75 K quantiser. Every period would come out the same, and the average of a million of them would tell you no more than one of them did: 4.75 K steps, and nothing in between.

What gets it below that is noise. Jitter on the comparator crossing - from the PTAT current, from the comparator itself, from the supply - is enough to push some periods over the clock edge and not others. Say the crossing sits a fraction f of a cycle past an edge. The output then alternates between N and $N + 1$ cycles, and the fraction of periods that come out long is an estimate of f. Averaging a million of them reads it to a few millikelvin.

That is dither: noise deliberately relied upon to make a coarse quantiser resolve below its own step. It is the same mechanism that lets a noisy ADC average its way below one LSB, and it is why converters are sometimes given dither on purpose. Here nobody gave it any - the circuit simply happened to have enough.

And a mechanism that depends on noise being big enough has an obvious way to fail.

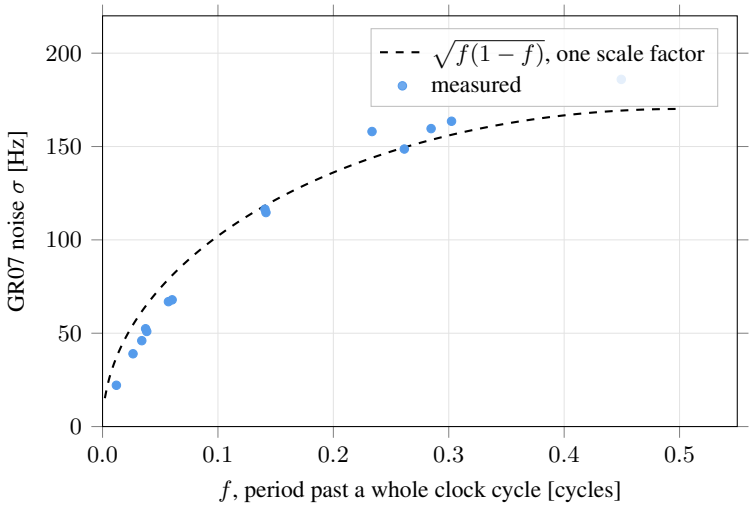


Figure 7: GR07's measured noise against f, the fractional part of its period in 64 MHz clock cycles. The dashed curve is $\sqrt{f(1-f)}$ with one scale factor fitted to the data

The measured noise is not constant: it varies by a factor of eight across the sweep, and it varies with f in a very particular way.

That shape is not a coincidence. If a fraction f of the periods come out long and the rest short, each period is a coin flip with probability f - a Bernoulli trial, whose variance is $f(1-f)$. The spread of a rate estimated from many of them therefore goes as $\sqrt{f(1-f)}$: smallest where the period is nearly a whole number of

cycles, largest where it sits half-way between. The dashed curve is that expression with a single scale factor fitted, and it follows the fourteen measurements with a correlation of 0.997.

So GR07's precision depends on the temperature it happens to be at, through nothing more than the arithmetic of the quantiser. At 10 °C its period is within a hundredth of a whole cycle and its noise is 22 Hz; at 25 °C it sits nearly half-way between and the noise is 186 Hz. Same circuit, same clock, eight times the noise, and it is the fractional part of a division that decides which you get.

This also puts a floor under how well GR07 can ever be trimmed. Calibration fits a smooth function to a transfer; what GR07 delivers is a 4.75 K staircase whose dither statistics change from step to step. The cheapest real fix is not more calibration points but a faster project clock, which shrinks the step in proportion.

GR06, with no clock anywhere in its path, has none of this. Its output is a pulse width in continuous time, and it is limited by something else entirely.

16.3.2 When averaging stops helping

If the resolution comes from averaging, the honest question is how long you can usefully average for. A standard deviation cannot answer it. The Allan deviation can: it asks how repeatable the answer is if you average for τ seconds, and its *slope* is the information. Falling means averaging still buys you precision. Rising means drift has taken over and averaging longer makes the answer worse.

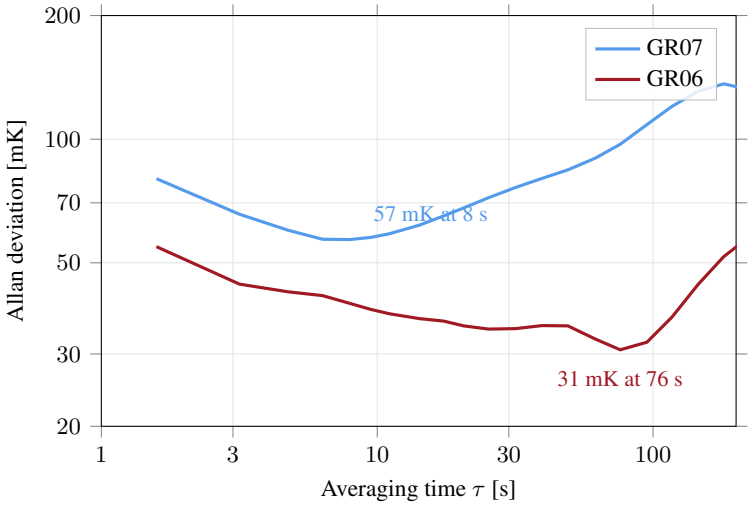


Figure 8: Allan deviation of both sensors over fifteen minutes in a quiet room. GR06 is lower at every averaging time. GR07 bottoms out at 57 mK after about eight seconds and then degrades; GR06 keeps improving out past a minute, to 31 mK

The two-hundred-fold event advantage does not appear anywhere on this plot. GR06 is the better sensor at every averaging time, from the shortest to the longest, and the gap widens.

That is worth sitting with, because within a single half-second capture GR07 really is the more precise of the two: its standard error is 3.3 mK against GR06's 12.8 mK. But note how little that is. Two hundred times the events should be $\sqrt{200}$, a factor of fifteen, and GR07 only manages four. The missing factor is its own single-event resolution: one GR07 period is worth 2.3 K of scatter against 0.6 K for one GR06 pulse, because 4.75 K of quantiser has to be dithered through. GR07 spends most of its event advantage buying back what the flip-flop took.

The four it keeps then does not survive either. Its Allan deviation at the shortest averaging time is 80 mK - twenty-four times its own statistical error - where GR06's is 55 mK against 12.8 mK, a factor of four. Both sensors are limited by something other than counting statistics, and GR07 by far the more so. Beyond about ten seconds its curve turns upward, which means something slower than the averaging window is now spoiling the answer and averaging longer makes it worse. GR06, producing two hundred times fewer events, keeps improving.

There is a second result hiding in the same run. Over those fifteen minutes the two sensors' fluctuations are essentially uncorrelated ($r = 0.11$) despite sitting on the same die, in the same room, sharing a supply. If the wander were the room, both would see it - so it looks like each sensor's own noise rather than the room.

A quiet room cannot prove that on its own, though: "nothing happened" and "both sensors missed it" look identical from inside. The next section is the experiment that separates them.

16.3.3 Two sensors, one event

Everything so far measures a sensor against a reference. The two sensors on this die can also be measured against each other, which asks a different question: not *what is the temperature* but *did something happen*. For that you need something to happen - so, a can of freeze spray, and then a finger held on the package.

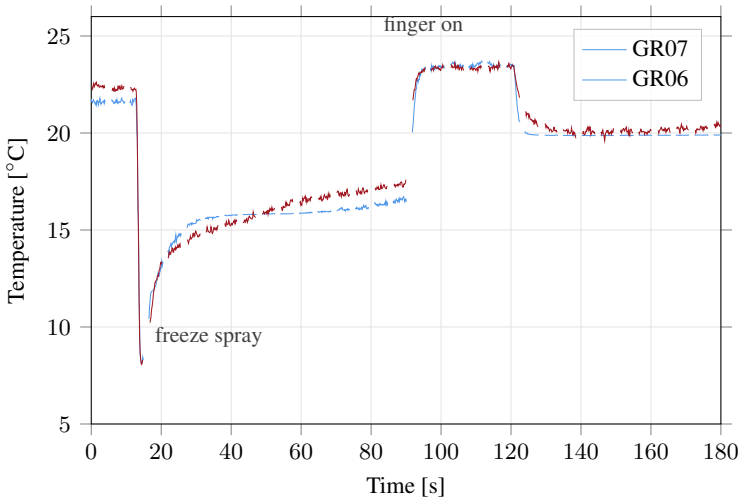


Figure 9: A can of freeze spray at 13 s, then a fingertip on the package from 88 s to 122 s, both sensors on one capture. The die falls 13 K in under two seconds, at better than 20 K/s at the steepest. Gaps are dead time between captures, drawn as gaps rather than interpolated

The two traces do not sit on top of each other - through the long recovery GR06 reads up to about a kelvin above GR07 - but they move together. Every feature appears in both at the same instant, and the correlation of the two temperature series is 0.96 through the spray and recovery and 0.99 while the finger is on.

The offset is not a surprise. Each sensor is calibrated at a single point near 23 °C against the ideal line through the origin, with no offset term to absorb comparator delay or reset time, so at 8 °C both are extrapolating fifteen kelvin beyond their only anchor. What the figure supports is the *shape*: the timing, the rates, and the fact that both saw the same thing. The absolute depth is the weakest number on the page.

This is the control the previous section needed. Quiet, the two sensors were uncorrelated; given something real to follow, they agree almost perfectly. Both cannot be true of the same physical temperature, and that settles it: the wander in the quiet room was not the room, it was each sensor's own noise. A real thermal event moves both. Noise moves one.

Which is the argument for putting two of anything on a die. One sensor gives you a number and no way to know whether to believe it. Two give you a way to tell a measurement from an artefact.

Now the same experiment, gently: four breaths on the package.

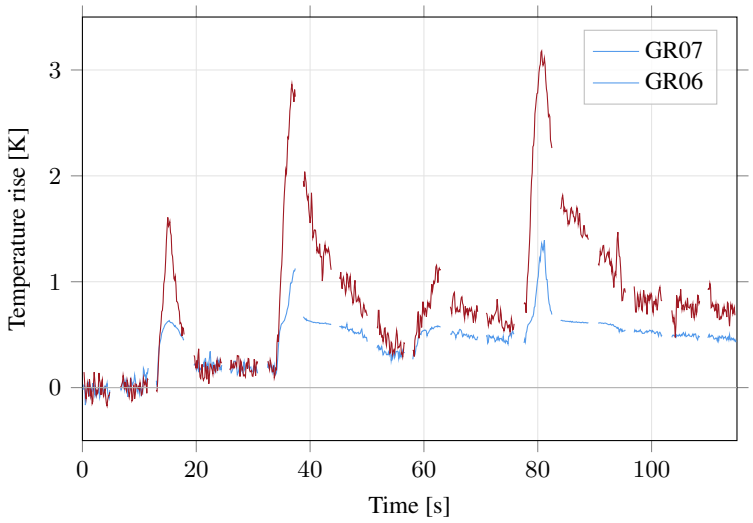


Figure 10: Four breaths, both sensors, one capture. Each breath moves both - but GR06 reads about 2.2 times the excursion GR07 does

Both sensors agree that four things happened, and disagree about how big they were by a factor of 2.2. The obvious reading is that the two circuits have different gains. They do not: over the 15 K of Figure 9 they agree to within a few per cent, and a real gain error would show up *more* strongly over a wider excursion, not vanish.

So something compresses GR07's response to small, fast signals without touching its response to large slow ones. A one-kelvin breath is about a fifth of GR07's 4.75 K quantiser step, so the quantiser is the obvious suspect - but the chamber sweep of Figure 7 shows its dither behaving the way ideal dither should, which would give a linear response at any excursion. The two observations are not yet reconciled, and nothing else measured here settles it.

What can be said without hand-waving is the practical part: for small, fast signals the two sensors disagree by a factor of about two, and GR06 is the one that agrees with the larger excursion. Trust it. Establishing *why* would need a controlled small-step experiment with GR07 parked at several known values of f - which is a good project, and is not in this data set.

16.4 One trap

16.4.1 GR06 does not drift, it switches

GR06 is the better-behaved sensor everywhere above, so it is worth asking what actually limits *it*. Hold the chamber still and look at a minute of GR06 with the slow dwell drift removed.

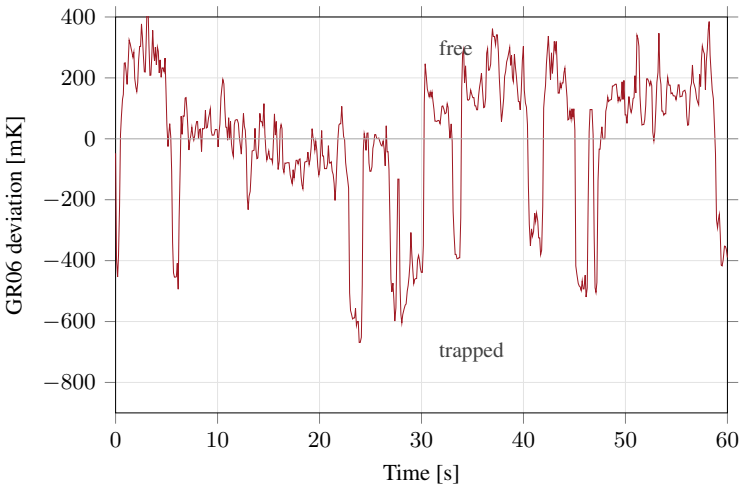


Figure 11: Sixty seconds of GR06 at a fixed chamber temperature, with the slow dwell drift removed. The reading sits flat, drops about six tenths of a kelvin to a second level, stays there a second or two, and comes back

Thermal noise would give a fuzzy band around zero. This gives two levels. That is random telegraph noise - a single charge trap in the silicon capturing and emitting one carrier - and it is the same mechanism the noise chapter introduces as the microscopic origin of flicker noise, here big enough to see one trap at a time.

The step is about 0.6 K from 5 °C to 55 °C - it drifts from 0.7 K to 0.4 K across that span, but it does not scale with anything - which says it is one trap rather than a population. The time it stays trapped is thermally activated and follows an Arrhenius law, which is what identifies it as a trap rather than as something in the instrument.

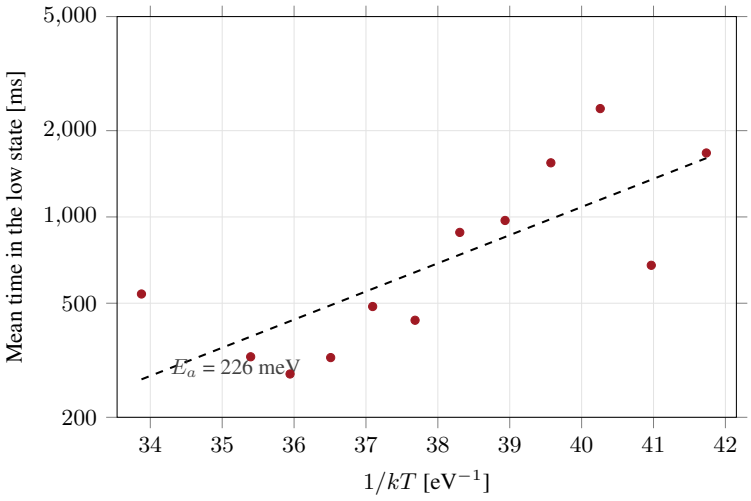


Figure 12: The trap's mean lifetime in the low state against inverse thermal energy, one point per chamber dwell. A straight line here is an Arrhenius law; the slope is an activation energy of 226 meV. Above about 55 °C the two levels merge into the noise and the fit stops meaning anything, so those dwells are left out

So GR06's real single-reading accuracy is not its ±0.35 K three-point calibration - that is a ninety-second mean. A single reading can be six tenths of a kelvin off whenever the trap happens to be occupied. Unlike GR07's quantiser this is cheap to fix: a median over a second or two steps straight over it.

16.5 Summary

Two sensors from this course, on the same die, from the same principle.

- Both PTAT cores work. The rates follow absolute temperature to within the residual of Figure 3, and the two designs - different resistors, and a hundredfold current division in one of them - agree with each other to a per cent in fractional slope.
- GR06, with three calibration points, is accurate to ±0.35 K over 5-70 °C. What is left after a straight line is bandgap curvature: a $T \ln T$ term takes 1.33 K down to 0.23 K. Its limit for a single reading is not linearity at all but one charge trap worth about 0.6 K, which a short median removes.
- GR07 has four times GR06's statistical precision within a capture and is the worse sensor at every averaging time - its readings scatter twenty-four times more than its own statistics predict. Its output is quantised to whole clock cycles, 4.75 K each, and only noise dithering the crossing lets averaging resolve below that. How precise it is then depends on where it sits in that step: the noise follows $\sqrt{f(1-f)}$ to a correlation of 0.997, an eightfold range across the sweep, decided by the fractional part of a division.
- Calibration cannot absorb a staircase. A faster clock shrinks it.
- The two sensors' noise is uncorrelated on one die. Two sensors tell you when something is real; one tells you a number.

If you take one thing from this chapter into your own project: both groups' analogue cores did what they were designed to do. Every awkward result came from the boundary - a flip-flop, a clock frequency, a choice of what to observe. That is where to spend your attention.

16.6 Would you like to know more?

The measurement setup, the full analysis and every figure's underlying data, in far more detail than fits here [1]

The Tiny Tapeout shuttle that carried it, and how to put your own design on one [2]

Where the PTAT core comes from, and why its curvature is not what limits these parts [3]

Quantisation, dither and why a coarse converter with noise on its input beats a coarse converter without [4]

Random telegraph noise as the microscopic origin of flicker noise [5]

[1] C. Wulff, "JNW-TEMP: Two temperature sensors, measured." 2026 [Online]. Available: https://analogicus.com/jnw-tt-2025/presentation.html

[2] M. Venn and the Tiny Tapeout contributors, "Tiny tapeout." [Online]. Available: https://tinytapeout.com

[3] B. Razavi, "The design of a low-voltage bandgap reference [the analog mind]," *IEEE Solid-State Circuits Magazine*, vol. 13, no. 3, pp. 6–16, 2021, doi: 10.1109/MSSC.2021.3088963.

[4] R. Schreier and G. C. Temes, *Understanding delta-sigma data converters*. John Wiley & Sons, 2005.

[5] M. J. Kirton and M. J. Uren, "Noise in solid-state microstructures: A new perspective on individual defects, interface states and low-frequency (1/f) noise," *Advances in Physics*, vol. 38, no. 4, pp. 367–468, 1989, doi: 10.1080/00018738900101122.

IC and ESD 17

17.1 What blocks must our IC include? . . . 261
17.1.1 Supply 262
17.2 Electrostatic Discharge 264
17.2.1 Human body model (HBM) 265
17.2.2 Charged device model (CDM) 266
17.3 An HBM ESD zap example 268
17.4 The grounded gate NMOS 271
17.5 But I just want a digital input, what do I need? 274
17.5.1 Input buffer 275
17.6 Latch-up 276
17.6.1 Electron injection . . 277
17.7 Summary 279
17.8 Would you like to know more? 279

Keywords: TempSense, Node Voltage, Ground, VDD, Clocks, Digital, Bias, RESET (POR), Package, Why ESD, CDM (Gauss, INV), HBM (01,10,02,20,12,21**), GGNMOS, Latch-up

Video is from 2024, so the plan might not be exactly the same. In addition, we're not using Caravel for the tapeout, but rather TinyTapeout.

17.1 What blocks must our IC include?

The project is to design an integrated temperature sensor.

First, we need to have an idea of what comes in and out of the temperature sensor. Before we have made the temperature sensor, we need to think what the signal interface could be, and we need to learn.

Maybe we read Kofi Makinwa's overview of temperature sensors and find one of the latest papers,

A BJT-based CMOS Temperature Sensor with Duty-cycle-modulated Output and ±0.54 °C (3-sigma) Inaccuracy from -40 °C to 125 °C [1].

At this point, you may struggle to understand the details of the paper, but at least it should be possible to see what comes in and out of the module. What I could find is in the table below, maybe you can find more?

Pin	Function	in/out	Value	Unit
VDD_3V3	analog supply	in	3.0	V
VDD_1V2	digital supply	in	1.2	V
VSS	ground	in	0	V
CLK_1V2	clock	in	20	MHz
RST_1V2	digital	out	0 or 1.2	V
I_C	bias	in	?	uA?
PHI1_1V2	digital	out	0 or 1.2	V
PHI2_1V2	digital	out	0 or 1.2	V
DCM_1V2	digital	out	0 or 1.2	V

This list contains supplies, clocks, digital outputs, bias currents and a ground. Let me explain what they are.

17.1.1 Supply

The temperature sensor has two supplies, one analog (3.3 V) and one digital (1.2 V), which must come from somewhere.

We're using TinyTapeout

That has ability for both 3.3 V and 1.8 V. An external low dropout regulator (LDO) provides the digital supply (1.8 V).

See for example absolute maximum ratings

17.1.1.1 Ground

Most ICs have a ground, a pin which is considered 0 V. It may have multiple grounds. Remember that a voltage is only defined between two points, so it's actually not true to talk about a voltage in a node (or on a wire). A voltage is always a differential to something. We've (as in global electronics engineers) have just agreed that it's useful to have a "node" or "wire" we consider 0 V.

17.1.1.2 Clocks

Most digital need a clock, and TinyTapeout can provide a 50 MHz clock which should suffice for most things. We could probably just use that clock for our temperature sensor.

17.1.1.3 Digital

We need to read the digital outputs. We could either feed those off chip, or use a on chip micro-controller. The TinyTapeout includes options to do both. We could connect digital outputs to the logic analyzer, and program the MCU to store the readings. Or we could connect the digital output to the I/O and use an instrument in the lab.

17.1.1.4 Bias

The TinyTapeout does not provide bias currents (that I found), so that is something you will need to make.

17.1.1.5 Conclusion

Even a temperature sensor needs something else on the IC. We need digital input/output, clock generation (PLL, oscillators), bias current generators, and voltage regulators (which require a constant reference voltage).

I would claim that any System-On-Chip will always need these blocks!

I want you to pause, take a look at the

course plan

and now you might understand why I've selected the topics.

17.1.1.6 One more thing

There is one more function we need when we have digital logic and a power supply. We need a "RESET" system.

Digital logic has a fundamental assumption that we can separate between a "1" and a "0", which is usually translated to for example 1.8 V (logic 1) and 0 V (logic 0). But if the power supply is at 0 V, before we connect the battery, then that fundamental assumption breaks.

When we connect the battery, how do we know the fundamental assumption is OK? It's certainly not OK at 30 mV supply. How about 500 mV? or 1.0 V? How would we know?

Most ICs will have a special analog block that can keep the digital logic, bias generators, clock generators, input/output and voltage regulators in a **safe** state until the power supply is high enough (for example 1.62 V).

One of the challenges with a Power On Reset (POR) is that we want to keep the system in a reset state until we're sure that the power is on. Another challenge is that the POR should not consume current.

If we make a level triggered (triggers when VDD reaches a certain level), then we need a reference, a comparator and maybe other circuits. As a result, potentially high current.

If we make a delay based POR, then we need a long delay, which means large resistors or capacitors. Accordingly, high cost.

Below is an idea for a Power-On-Reset I had way back when. The POR uses a delay based on the tunneling current in a thin oxide transistor (2), and uses a thick-oxide transistor (3) as a capacitor. The output X would go to a Schmitt trigger (5).

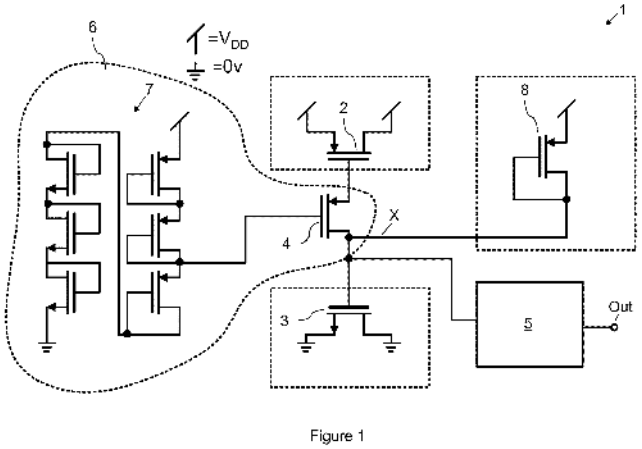


Figure 1: Power On Reset using gate tunneling

17.2 Electrostatic Discharge

If you make an IC, you must consider Electrostatic Discharge (ESD) Protection circuits

ESD events are tricky. They are short (ns), high current (Amps) and poorly modeled in the SPICE model.

Most SPICE models will not model correctly what happens to an transistor during an ESD event. The SPICE models are not made to model what happens during an ESD event, they are made to model how the transistors behave at low fields and lower current.

But ESD design is a must, you have to think about ESD, otherwise your IC will never work.

Consider a certain ESD specification, for example 1 kV human body model, a requirement for an integrated circuit.

By requirement I mean if the 1 kV is not met, then the project will be delayed until it is fixed. If it's not fixed, then the project will be infinitely delayed, or in other words, canceled.

Now imagine it's your responsibility to ensure it meets the 1 kV specification, what would you do? I would recommend you read one of the few ESD books in existence - see the reading list at the end of the chapter - and rely on your understanding of PN-junctions.

The industry has agreed on some common test criteria for electrostatic discharge. Test that model what happens when a person touches your IC, during soldering, and PCB mounting. If your IC passes the test then it's probably going to survive in volume production

Standards for testing at JEDEC

The JEDEC standard splits ESD events into Human body model, Charged device model, and System level ESD.

Once mounted on the PCB, the ICs can be more protected against ESD events, however, it depends on the PCB, and how that reacts to a current.

Take a look at your USB-A connector, you will notice that the outer pins, the power and ground, are made such that they connect first, The $D+$ and $D-$ pins are a bit shorter, so they connect some μs later. The reason is ESD. The power and ground usually have a low impedance connection in decoupling capacitors and power circuits, so those can handle a large ESD zap. The signals can go directly to an IC, and thus be more sensitive.

We won't go into details on System level ESD, as that is more a PCB type of concern. The physics are the same, but the details are different.

17.2.1 Human body model (HBM)

HBM is the "simple" version of ESD, a model can be seen in Figure 2. Some of the properties of HBM are:

- Models a person touching a device with a finger
- **Long** duration (around 100 ns)
- Acts like a current source into a pin
- Can usually be handled in the I/O ring
- 4 kV HBM ESD is 2.67 A peak current

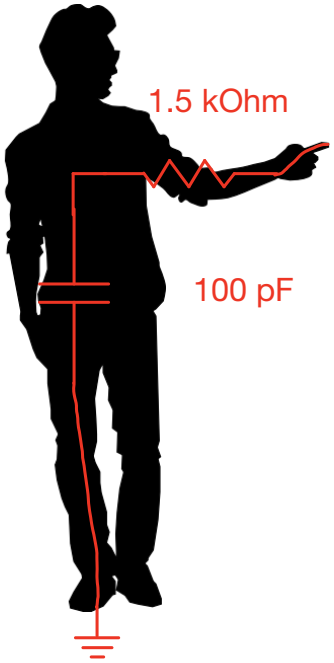


Figure 2: Human body model (HBM)

More on circuits that protect from HBM later.

17.2.2 Charged device model (CDM)

> An IC left alone for long enough will equalize the Fermi potential across the whole IC.

Not entirely a true statement, but roughly true. One exception is non-volatile memory, like flash, which uses Fowler-Nordheim tunneling to charge and discharge a capacitor that keeps its charge for a very, very long time.

I'm pretty sure that if you leave an SSD hardrive to the heat death of the universe in maybe $10^{10^{10^{56}}}$ years, then the charges will equalize, and the Fermi level will be the same across the whole IC, so it's just a matter of time.

Assume there is an equal number of electrons and protons on the IC. According to Gauss' law

$$\oint_{\partial\Omega} \mathbf{E} \cdot d\mathbf{S} = \frac{1}{\epsilon_0} \iiint_V \rho \cdot dV$$

Which says that the electric field through the surface is the volume integral of the charges inside the surface. If there are the same amount of protons and electrons, and the distribution is even, then

there will be no field through IC surface. As such, there is no external electric field from the IC.

If we place an IC in an electric field, the charges inside will redistribute. Flip the IC on its back, place it on an metal plate with an insulator in-between, and charge the metal plate to 1 kV, as shown in Figure 3.

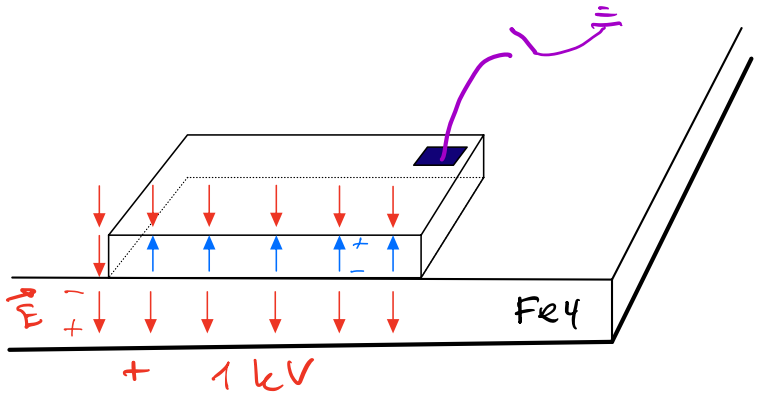


Figure 3: Charged Device Model (CDM) testing

Inside the integrated circuit, electrons and holes will redistribute to compensate for the electric field. Closest to the metal plate there will be a negative charge, and furthest away there will be a positive charge.

This comes from the fact that if you leave a metal inside an electric field for long enough the metal will not have any internal field. If there was an internal field, the charges would move. Over time the charges will be located at the ends of the metal.

Take a grounded wire, touch one of the pins on the IC. Since we now have a metal connection between a pin and a low potential the charges inside the IC will redistribute extremely quickly, on the order of a few ns.

During this Charged Device Model event the internal fields in the IC will be chaotic, but at any given point in time, the voltage across sensitive devices must remain below where the device physically breaks.

Take the MOSFET transistor. Between the gate and the source there is an thin oxide, maybe a few nm. If the field strength between gate and source is high enough, then the force felt by the electrons in co-valent bonds will be $\vec{F} = q\vec{E}$. At some point the co-valent bonds might break, and the oxide could be permanently damaged. Think of a lightning bolt through the oxide, it's a similar process.

Our job, as electronics engineers, is to ensure we put in additional circuits to prevent the fields during a CDM event from causing damage.

For example, let's say I have two inverters powered by different supply, VDD1 and VDD2. If I in my ESD test ground VDD1, and not VDD2, I will quickly bring VDD1 to zero, while VDD2 might react slower, and stay closer to 1 kV. The gate source of the PMOS in the second inverter will see approximately 1 kV across the oxide, and will break. How could I prevent that?

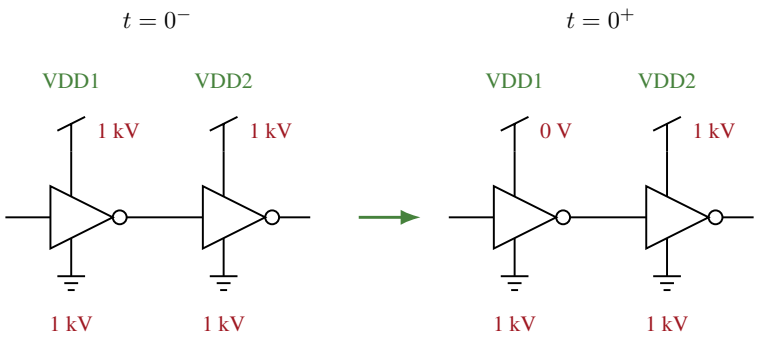


Figure 4: Cross domain voltage problem with CDM (or indeed HBM) events

Assuming some luck, then VDD1 and VDD2 are separate, but the same voltage, or at least close enough, I can take two diodes, connected in opposite directions, between VDD1 and VDD2. As such, when VDD1 is grounded, VDD2 will follow but maybe be 0.6 V higher. As a result, the PMOS gate never sees more than approximately 0.6 V across the gate oxide, and everyone is happy.

Now imagine an IC with hundreds of supplies, and billions of inverters. How can I make sure that everything is OK?

CDM is tricky, because there are so many details, and it's easy to miss one that makes your circuit break.

17.3 An HBM ESD zap example

Imagine a ESD zap between VSS and VDD. How can we protect the device?

The positive current enters the VSS, and leaves via the VDD, so our supplies are flipped up-side down. It's a fair assumption that none of the circuits inside will work as intended.

But the IC must not die, so we have to lead the current to ground somehow.

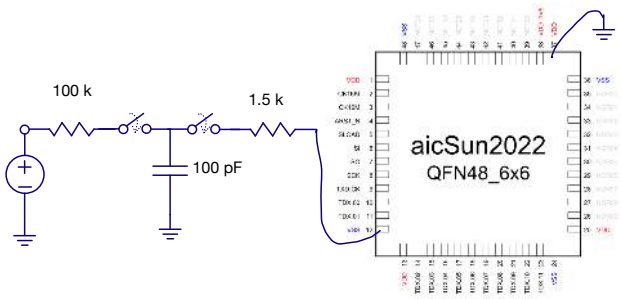


Figure 5: ESD HBM zap example

Let's simplify and think of the possible permutations, shown in Figure 6. We don't know where the current will enter nor where it will leave our circuit, so we must make sure that all combinations are covered.

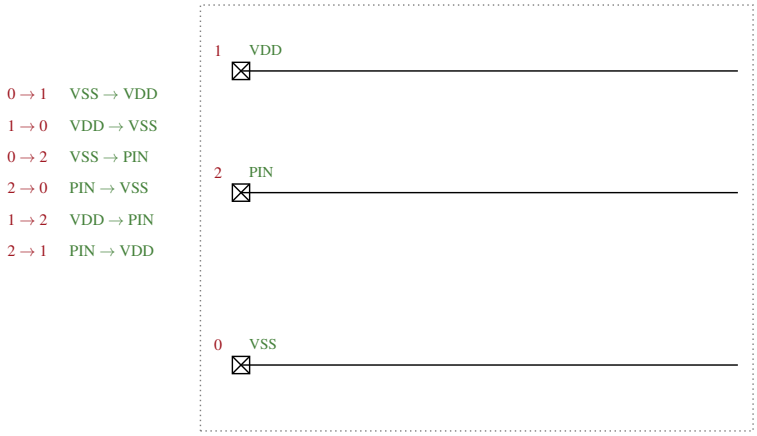


Figure 6: The six ways a zap can cross a three terminal chip. Three pads means six ordered pairs, and ESD protection is the promise that every one of them has somewhere for the current to go. The chip is empty here on purpose; the next three figures fill it in.

When the current enters VSS and must leave via VDD, then it's simple, we can use a diode, as shown in Figure 7.

Under normal operation the diode will be reverse biased, and although it will add some leakage, it will not affect the normal operation of our IC.

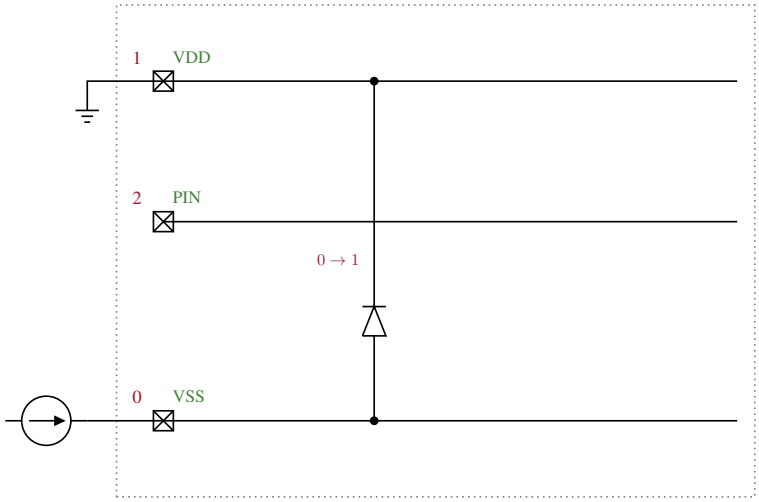


Figure 7: Protection for a zap from ground to VDD. One diode, reverse biased whenever the chip is running, forward biased the moment VSS goes above VDD. The column crosses the pin rail without a junction dot, which is a statement that nothing is connected there yet.

The same is true for current in on VSS and out on PIN. Here we can also use a diode, as shown in Figure 8.

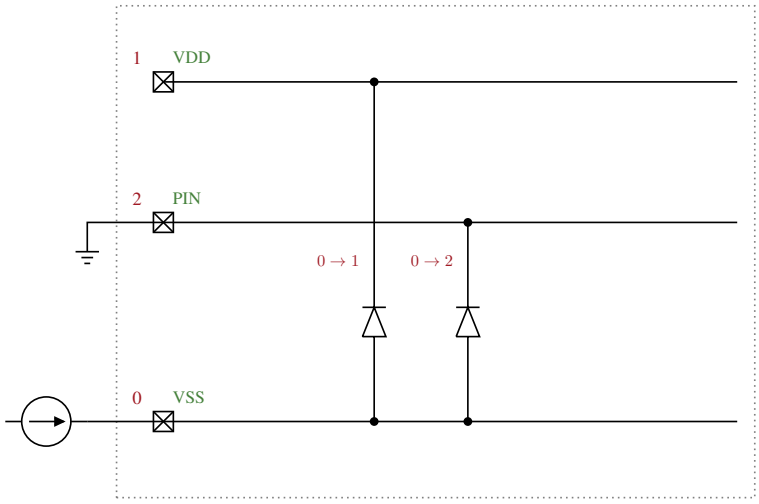


Figure 8: Protection for a zap from ground to a pin. Nothing new happens, and that is the point: a signal pin sits above VSS in normal operation for the same reason VDD does, so the same reverse biased diode covers it.

For a current in on VDD and out on VSS we have a challenge. That's the normal way for current to flow.

For those from Norway that have played a kids game Bjørnen sover, that's a apt mental image. We want a circuit that most of the time sleeps, and does not affect our normal IC operation. But if a huge current comes in on VDD, and the VDD voltage shoots up fast, the circuit must wake up and bring the voltage down.

If the circuit triggers under normal operating condition, when you're watching a video on your phone, your battery will drain very fast, and your phone might even catch fire.

As such, ESD design engineers have a "ESD design window". Never let the ESD circuit trigger when VDD < normal, but always trigger the ESD circuit before VDD > breakdown of circuit.

A circuit that can sometimes be used, if the ESD design window is not too small, is the Grounded-Gate-NMOS in Figure 9.

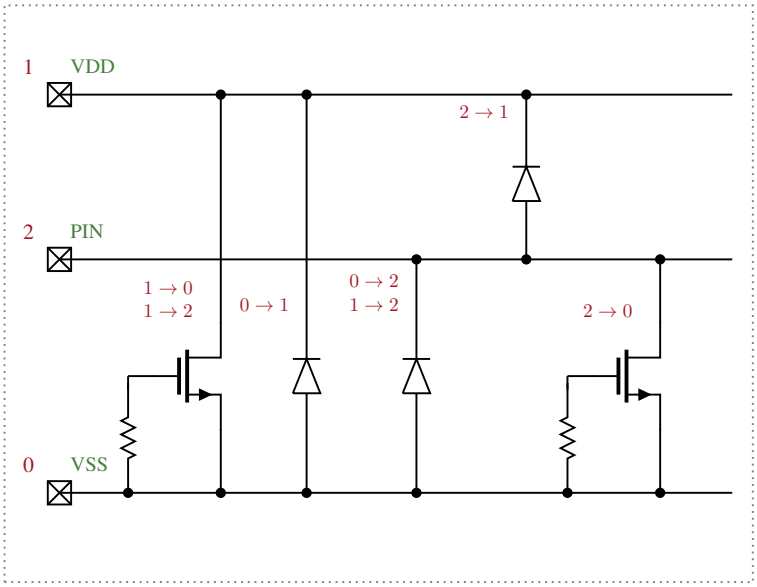


Figure 9: All six permutations covered by four devices. Each device is labelled with the zap it carries, and two of them carry one jointly: a strike from VDD to the pin goes down the left grounded gate NMOS to VSS and back up through the pin diode, which is why $1 \longrightarrow 2$ *appears twice. That is the real structure of an ESD network. It is not one path per permutation but a small set of devices arranged so every permutation has a series combination available, which is why adding a pin costs two diodes rather than six.*

17.4 The grounded gate NMOS

If you try the circuit in Figure 10 with the normal BSIM spice model, it will not work. The transistor model does not include that part of the physics.

We need to think about how electrons, holes PN-junctions and bipolars work. Let's refresh quantum mechanics a bit.

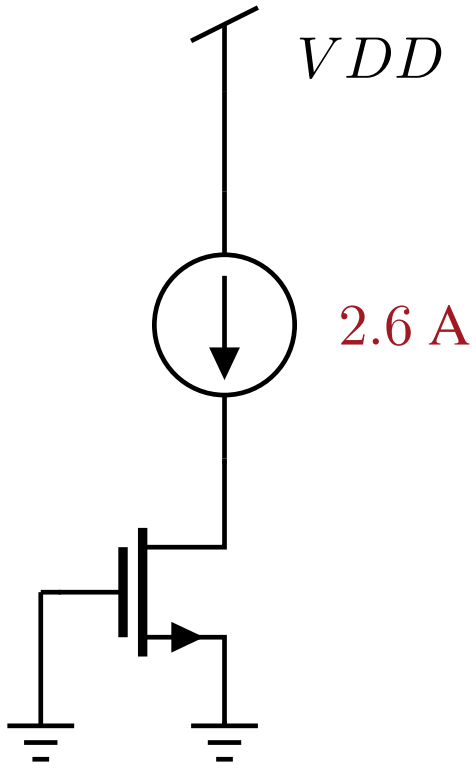


Figure 10: The grounded gate NMOS (GGNMOS)

Electrons sticking to atoms (bound electrons), can only exist at discrete energy levels. As we bring atoms closer to each-other the discrete energy levels will split, as computed from Schrodinger, into bands of allowed energy states. These bands of energy can have lower energy than the discrete energy levels of the atom. That's why some atoms stick together and form molecules through co-valent bonds, ionic bonds, or whatever the chemists like to call it. It's all the same thing, it's lower energy states that make the electrons happy, some are strong, some are weak.

For silicon the energy band structure is tricky to compute, so we simplify to band diagrams that only show the lowest energy conduction band and highest energy valence band.

Electrons can move freely in the conduction band (until they hit something, or scatter), and electrons moving in the valence band act like positive particles, nicknamed holes.

How many free charges there are in a band is given by Fermi-Dirac distribution and the density of states (allowed energy levels).

If an electron, or a hole have sufficient energy (accelerated by a field), they can free an electron/hole pair when they scatter off an atom. If you break too many bonds between atoms, your material will be damaged.

Assume a transistor like the one in Figure 11. The gate, source and bulk is connected to ground. The drain is connected to a high voltage.

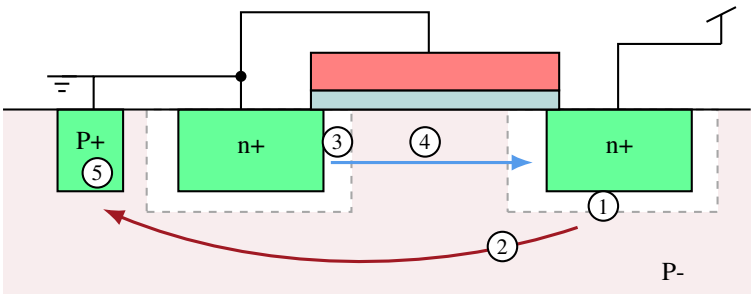


Figure 11: Cross section of the grounded gate NMOS

The process of a GGNMOS will be (1) Avalanche, (2) hole accumulation, (3) forward bias of PN-junction, and (4) direct electron current from source to drain.

The first thing that can happen is that the field in the depletion zone between drain and bulk (1) is large, due to the high voltage on drain, and the thin depletion region.

In the substrate (P-) there are mostly holes, but there are also electrons. If an electron diffuses close to the drain region it will be swept across to drain by the high field.

The high field might accelerate the electron to such an energy that it can, when it scatters off the atoms in the depletion zone, knock out an electron/hole pair.

The hole will go to the substrate (2), while the new electron will continue towards drain. The new electron can also knock out a new electron/hole pair (energy level is set by impact ionization of the atom), so can the old one assuming it accelerates enough.

One electron turn into two, two to four, four to eight and so on. The number of electrons can quickly become large, and we have an avalanche condition. Same as a snow avalanche, where everything was quiet and nice, now suddenly, there is a big trouble.

Usually the avalanche process does not damage anything, at least initially, but it does increase the hole concentration in the bulk. The number of holes in the bulk will be the same as the number of electrons freed in the depletion region.

The extra holes underneath the transistor will increase the local potential. If the substrate contact (5) is far away, then the local potential close to the source/bulk PN-junction (3) might increase enough to significantly increase the number of electrons injected from source.

Some of the electrons will find a hole, and settle down, while others will diffuse around. If some of the electrons gets close to the drain region, and the field in the depletion zone, they will be accelerated by the drain/bulk field, and can further increase the avalanche condition.

For a normal transistor, not designed to survive, the electron flow (4) can cause local damage to the drain. Normally there is nothing that prevents the current from increasing, and the transistor will eventually die.

If we add a resistor to the drain region (unsilicided drain), however, we will slow down the electron flow, and we can get a stable condition, and design a transistor that survives.

Turns out, that every single NMOS has a sleeping bear. A parasitic bipolar. That's exactly what this GGNMOS is, a bipolar transistor, although a pretty bad one, that is designed to trigger when avalanche condition sets in and is designed to survive.

A normal NMOS, however, can also trigger, and if you have not thought about limiting the electron current, it can die, with IC killing consequences. Specifically, the drain and source will be shorted by likely the silicide on top of the drain, and instead of a transistor with high output impedance, we'll have a drain source connection with a few kOhm output impedance.

Take a look at New Ballasting Layout Schemes to Improve ESD Robustness of I/O Buffers in Fully Silicided CMOS Process [2] for the pretty pictures you'll get when the drain/source breaks.

17.5 But I just want a digital input, what do I need?

Even if it's only a digital input, you still need to consider ESD events.

Below is a complete digital input network.

Assuming we have a QFN package package there will be a bondwire from the package metal, to our die pad.

Right after the die pad, sometimes under, there will be a primary ESD protection that can conduct, in all directions, between input, supply and ground.

From the input it's common to have a resistor to reduce the probability of currents going towards the core area.

Before we get to a transistor gate oxide it's common to have a set of secondary protection circuits. A resistor further reduces the current, and two local clamps (GGPMOS and GGNMOS) ensure that the voltage across the transistor gate does not go to breakdown levels.

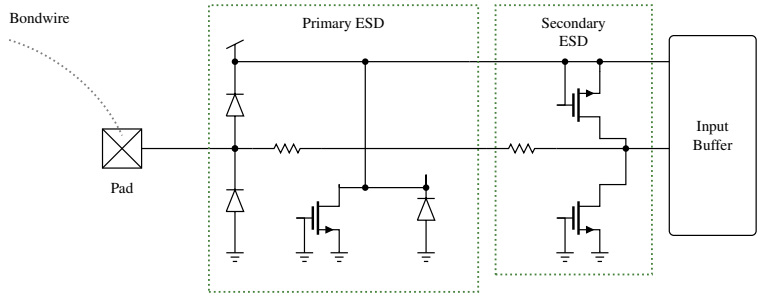


Figure 12: Full protection of an input including secondary protection

17.5.1 Input buffer

An input buffer can be seen below. I like to include a RC low-pass filter to filter out the RF frequencies (I don't want my input to toggle if a phone is on top of my circuit).

After the RC filter we need a Schmitt trigger, you can find a Schmitt trigger at JNW_TR_SKY130A.

The Schmitt trigger must be with thick oxide gates and with IO supply (for example 3.0 V).

The first inverter must also be a thick oxide inverter, however, the supply of the inverter will be core supply (for example 1.2 V). The thick oxide inverter provides a level-shift to core supply.

The last inverter is just to get the polarity of the TO_CORE signal the same as the input.

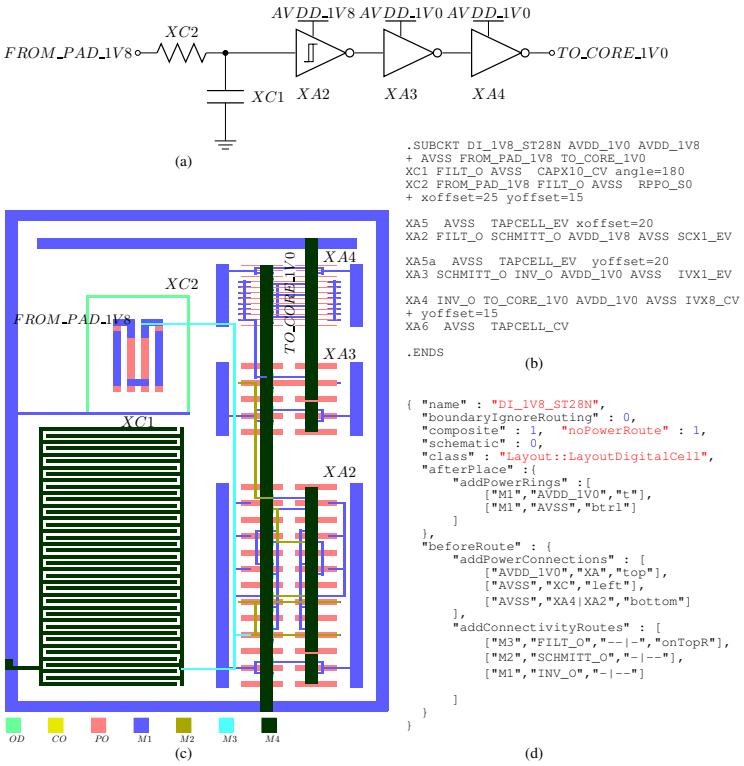


Figure 13: Full digital input including RC filter, Schmitt trigger and level shifters: (a) schematic, (b) netlist, (c) the compiled layout, and (d) the ciccreator recipe that generated it

17.6 Latch-up

Another fun physics problem can happen in digital logic that is close to an electron source, like a connection to the real world, what we call a pad. A pad is where you connect the bond-wire in a QFN type of package with wire-bonding

Assume we have the circuit in Figure 14. Under certain conditions we can get a short from VDD to ground.

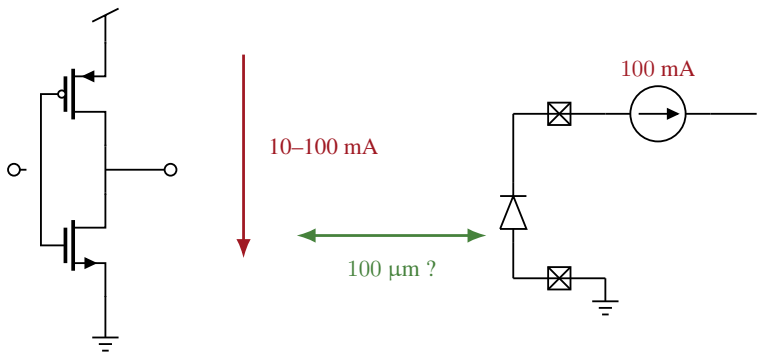


Figure 14: Inverter that suddenly shorts from VDD to ground located close to a PAD

Consider the cross section of the inverter in Figure 15. The latch-up process starts with electron injection (1), then forward bias of PMOS source/drain junction (2), forward bias of NMOS source/drain junction (3) , and finally positive feedback .

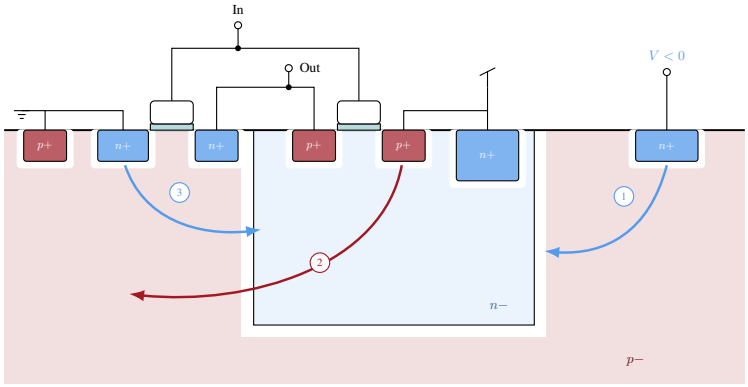


Figure 15: Cross section of an inverter

17.6.1 Electron injection

Assume that we have an electron source, for example a pad that is below ground for a bit. This will inject electrons into the substrate/bulk (1) and electrons will diffuse around.

If some of the electrons comes close to the N-well depletion region (2) they will be swept across by the built-in field. As a result, the potential of the N-well will decrease, and we can forward bias the source or drain junction of a PMOS.

17.6.1.1 Forward biased PMOS source or drain junction

With a forward biased source/bulk junction (2), holes will be injected into the N-Well, but similarly to the GGNMOS, they might not find a electron immediately.

Some of the holes can reach the depletion region towards our NMOS, and be swept across the junction.

17.6.1.2 Forward biased NMOS source or drain junction

The increase in hole concentration underneath the NMOS can forward bias the PN diode between source (or drain) and bulk. If this happens, then we get electron injection into bulk. Some of those electrons can reach the N-well depletion region, and be swept across (3).

17.6.1.3 Positive-feedback

Now we have a condition where the process accelerates, and locks-up. Once turned on, this circuit will not turn off until the supply is low.

This is a phenomenon called latch-up. Similar to ESD circuits, latch-up can short the supply to ground, and make things burn.

That is why, when we have digital logic, we need to be extra careful close to the connection to the real world. Latch-up is bad.

We can prevent latch-up if we ensure that the electrons that start the process never reach the N-wells. We can also prevent latch-up by separating the NMOS and PMOS by guard rings (connections to ground, or indeed supply), to serve as places where all these electrons and holes can go.

Maybe it seems like a rare event for latch-up to happen, but trust me, it's real, and it can happen in the strangest places. Similar to ESD, it's a problem that can kill an IC, and make us pay another X million dollars for a new tapeout, in addition to the layout work needed to fix it.

Latch-up is why you will find the design rule check complaining if you don't have enough substrate connections to ground, or N-well connections to power close to your transistors.

Similar to the GGNMOS, this circuit, a thyristor can be a useful circuit in ESD design. If we can trigger the thyristor when the VDD shoots too high, then we can create a good ESD protection circuit.

See low-leakage ESD for a few examples.

A model with the parasitic bipolars can be seen in Figure 16. The resistors in the picture is to emulate what happens when there is a current injected into the base of the NPN or PNP. I would recommend that you think through the physics instead of using the parasitic bipolar circuits. I've found the parasitic bipolar leads you down the wrong path when you actually want to understand the physics of latch-up.

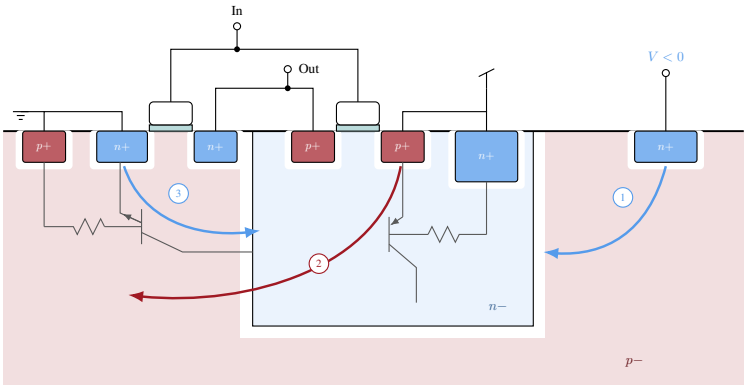


Figure 16: Cross section of an inverter including the parasitic bipolars

You must **always handle ESD** on an IC

- Do everything yourself
- Use libraries from foundry
- Get help www.sofics.com

17.7 Summary

The one-page version of this chapter:

- An ESD zap is a real event with a name: HBM is a charged person (kilovolts, amps through 1.5k), CDM is the chip discharging itself (nanoseconds, brutal)
- Protection is a promise about every ordered pair of pads: each of the six permutations on a three-pad chip needs somewhere for the current to go
- Two diodes per pin plus one rail clamp cover all permutations - adding a pin costs two diodes, not six paths
- The grounded gate NMOS is off by every model you were taught; the parasitic lateral bipolar is what sinks the amperes
- Thin gate oxide cannot take the leftover voltage, so real inputs add secondary protection behind a resistor
- Latch-up is the same parasitic bipolars firing in normal operation: keep well taps close, keep injectors away from wells

17.8 Would you like to know more?

ESD (Electrostatic Discharge) Protection Design for Nanoelectronics in CMOS Technology [3]

Overview on Latch-Up Prevention in CMOS Integrated Circuits by Circuit Solutions [4]

Overview on ESD Protection Designs of Low-Parasitic Capacitance for RF ICs in CMOS Technologies [5]

[1] Z. Huang, Z. Tang, X.-P. Yu, Z. Shi, L. Lin, and N. N. Tan, "A BJT-based CMOS temperature sensor with duty-cycle-modulated output and ±0.5°c (3σ) inaccuracy from −40 °c to 125 °c," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 8, pp. 2780–2784, 2021, doi: 10.1109/TCSII.2021.3068283.

[2] M.-D. Ker, W.-Y. Chen, W.-T. Shieh, and I.-J. Wei, "New ballasting layout schemes to improve ESD robustness of i/o buffers in fully silicided CMOS process," *IEEE Transactions on Electron Devices*, vol. 56, no. 12, pp. 3149–3159, 2009, doi: 10.1109/TED.2009.2031003.

[3] M.-D. Ker, "ESD (electrostatic discharge) protection design for nanoelectronics in CMOS technology," in *2006 advanced signal processing, circuit and system design techniques for communications*, 2006, pp. 217–279, doi: 10.1109/ASPCAS.2006.251127.

[4] M.-D. Ker and Z.-H. Jiang, "Overview on latch-up prevention in CMOS integrated circuits by circuit solutions," *IEEE Journal of the Electron Devices Society*, vol. 11, pp. 141–152, 2023, doi: 10.1109/JEDS.2022.3231822.

[5] M.-D. Ker, C.-Y. Lin, and Y.-W. Hsiao, "Overview on ESD protection designs of low-parasitic capacitance for RF ICs in CMOS technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 11, no. 2, pp. 207–218, 2011, doi: 10.1109/TDMR.2011.2106129.

References and bias 18

18.1 **Routing** 282
18.2 **Bandgap voltage reference** 284
18.2.1 A voltage complementary to temperature (CTAT) 284
18.2.2 A current proportional to temperature (PTAT) 286
18.2.3 How to combine a CTAT with a PTAT ? 287
18.2.4 Widlar reference . . 289
18.2.5 Brokaw reference . 290
18.2.6 Low voltage bandgap 294
18.2.7 Curvature correction 297
18.2.8 MOS references . . 299
18.2.9 The native threshold is set when the ingot is grown 301
18.2.10 FD-SOI moves the problem, it does not remove it 302
18.2.11 The back gate is a trimming knob, not a reference 303
18.3 **Bias** 303
18.3.1 Voltage to current conversion 303
18.3.2 GM Cell 304
18.3.3 Every one of these loops can fail to start 306
18.4 **Summary** 308
18.5 **Would you like to know more?** 309

Keywords: VREF, IREF, VD, BGAP, LVBGAP, VI, GMCELL

In our SPICE testbenches, and trial schematics, it's common to include voltage sources and current sources, like the symbols in Figure 1.

The ideal voltage source, or ideal current source, does not exist in the real world. There is no such thing.

We can come close to creating a voltage source, a known voltage, with a low source impedance, but not zero impedance. And it won't be infinitely fast either. If we suddenly decide to pull 1 kA from a lab supply I promise you the voltage will drop.

How do we create something that is a *good enough* voltage and current source on an IC? That's the goal of this chapter. To give you an introduction to "voltage sources" and "current sources" that we can make on an integrated circuit.

But before we take a look at the voltage and current source, I want you to think about how you would route a current, or a voltage on an IC.

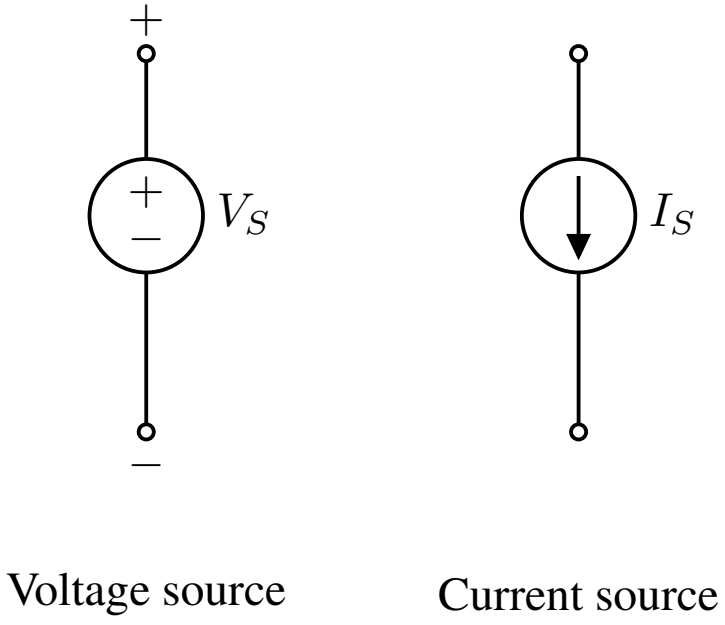


Figure 1: Symbols for voltage source and current source

18.1 Routing

Assume we have a known voltage on our IC, a reference voltage. How can we make sure we can share that voltage across an IC?

A voltage is only defined between two points. There is no such thing as the *voltage at a point on a wire*, nor *voltage in a node*. Yes, I know we say that, but it's not right. What we forget is that by *voltage in a node* we always, always mean *voltage in a node referred to ground*.

We've invented this magical place called *ground*, the final resting place of electrons, and we have agreed that voltages refer to that point.

As such, when we say "Voltage in node A is 1V", what we actually mean is "Voltage in node A is 1 V referred to ground".

Maybe you now understand why we can't just route a voltage across the IC, the *other side* might not have the same ground. The *other side* might have a different impedance to ground, and the impedance might be a function of time, voltage, frequency temperature, pressure and presence of gremlins.

Consider Figure 2. The ground impedance may depend on time, voltage, frequency, temperature, pressure (yes, stress in silicon can change the band structure, thus the conduction band energy levels, and thus the available charge carriers in the conduction band).

If there is no current flowing in the ground impedance at the destination we may be OK, but usually, there is some current flowing into ground at the destination. There is a circuit there.

If we choose to route a reference as a voltage we need to be careful with the ground.

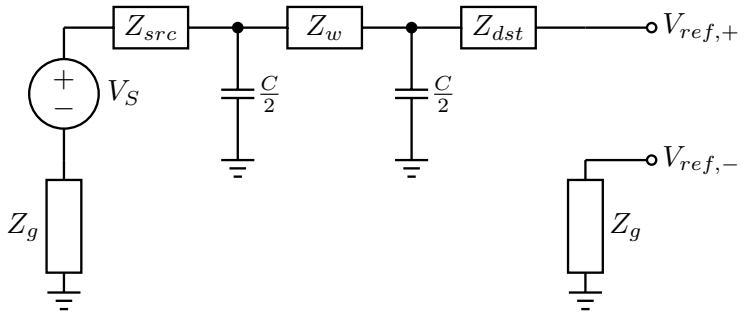


Figure 2: Voltage source with ground impedance. Routing long distances it's not possible to guarantee we have the same ground impedance at the destination.

Most of the time, in order not to think about the ground impedance, we choose to route a known quantity, the reference, as a current instead of a voltage. That means, however, we must convert from a

voltage to a current, but we can do that with a resistor (you'll see later), and as long as the resistor is the same on the other side of the IC, then we'll know what the voltage is.

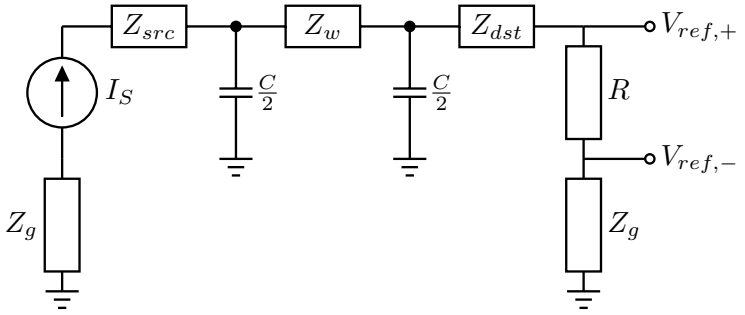


Figure 3: Routing a reference as a current.

Resistors have finite matching across die, let's say 2 % 3-sigma variation. A limitation on how accurate we can distribute reference across the IC with current method.

For most voltage regulators (think about the circuit that delivers the digital voltage for an MCU) 2 % may be an acceptable portion of the error budget. For a battery charger, however, the termination voltage of Li-ion batteries need to be precise, more accurate than 1 %.

For that application we cannot distribute current, we must distribute voltage, but we need to care deeply about ground.

But how can "It's better to distribute a voltage as a current across the IC, it's more accurate" and "If you need something really accurate, you must distribute voltage" both be true?

Imagine I have a 0.5 % 3-sigma accurate voltage reference at 1.22 V, that's a sigma of 2 mV. I need this reference voltage on a block on the other side of the IC, I don't want to distribute voltage, because I don't know that the ground is the same on the other side, at least not to a precision of 2 mV. I convert the voltage into a current, however, I know the R has a 2 % 3-sigma across die, so my error budget immediately increases to 2.06%.

But what if I must have 0.5 % 3-sigma voltage in the block? For example in a battery charger, where the 4.3 V termination voltage must be 1 % accurate? I have no choice but to go with voltage directly from the reference, but the key point, is then the receiving block **cannot** be on the other side of the IC. The reference must be right next to my block.

I could use two references on my IC, one for the ADC and one for the battery charger. Ask yourself, "Why do we care if there are two references?" And the answer is "Silicon area is expensive, to make things cheap, we must make things small",

in other words, we should not duplicate features unless we absolutely have to.

18.2 Bandgap voltage reference

One of the ways to create a known reference on an integrated circuit is the "bandgap voltage reference". There are flavors of bandgaps, but all rely on the bandgap of silicon, which is about 1.12 eV.

We can't access the bandgap voltage directly, but we can use the fact that diodes, and BJTs all have a voltage across the PN junction of about 1.12 V at absolute zero (actually, slightly higher, maybe 1.2 V), and that they have a well known temperature dependence from that point.

18.2.1 A voltage complementary to temperature (CTAT)

A diode connected bipolar transistor, shown in Figure 4, or indeed a PN diode, assuming a fixed current, will have a voltage across that is temperature dependent

$$I_D = I_S \left(e^{\frac{V_{BE}}{V_T}} - 1 \right) + I_B \approx I_S e^{\frac{V_{BE}}{V_T}}$$

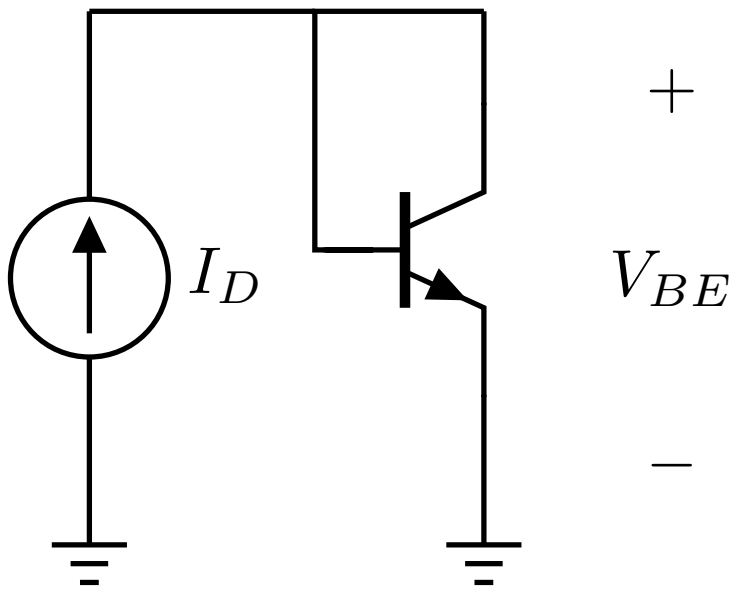


Figure 4: Diode connected bipolar transistor

As I_S is much smaller than I_C we can ignore the -1, and we assume that the base current is much smaller than the collector current.

Re-arranging for V_{BE} and inserting for

$$V_T = \frac{kT}{q}$$

$$V_{BE} = \frac{kT}{q} \ln \frac{I_C}{I_S}$$

$$I_S = qAn_i^2 \left[\frac{D_n}{L_n N_A} + \frac{D_p}{L_p N_D} \right]$$

From this equation, it looks like the voltage V_{BE} is proportional to temperature, however, it turns out that the V_{BE} decreases with temperature due to the temperature dependence of I_S.

The V_{BE} is almost linear with temperature with a property that if you extrapolate the V_{BE} line to zero Kelvin, then all diode voltages seem to meet at one voltage, $V_{G0} \approx 1.2$ V. That number is close to, but not the same as, the silicon bandgap you look up in a table: the gap is 1.12 eV at room temperature and 1.17 eV at zero Kelvin, while the intercept these lines extrapolate to is around 1.20 to 1.22 V, because the extrapolation also drags the temperature dependence of I_S along with it. It is a voltage, not an energy, and the reference is named after it.

To see the temperature coefficient, I find it easier to re-arrange the equation above.

Some algebra (see Diodes)

$$V_{BE} = \frac{kT}{q}(\ell - 3 \ln T) + V_G$$

The ℓ is a temperature independent constant given by

$$\ell = \ln I_C - \ln qA - \ln \left[\frac{D_n}{L_n N_A} + \frac{D_p}{L_p N_D} \right] \\ -2 \ln 2 - \frac{3}{2} \ln m_n^* - \frac{3}{2} \ln m_p^* - 3 \ln \frac{2\pi k}{h^2}$$

And if we plot the diode voltage, we can see that the voltage decreases as a function of temperature.

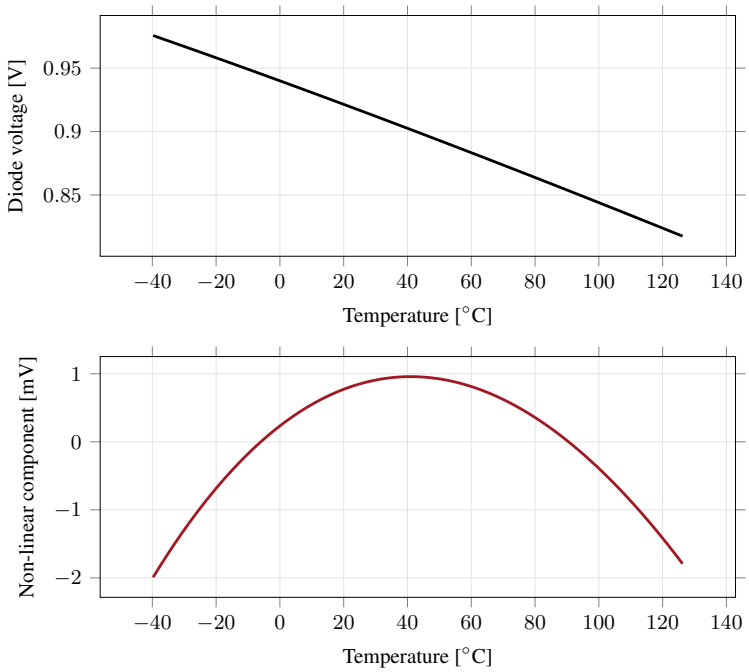


Figure 5: Diode voltage versus temperature. Bottom plot shows deviation from a straight line.

18.2.2 A current proportional to temperature (PTAT)

If we take two diodes, or bipolars, biased at different current densities, as shown in Figure 6, then

$$V_{D1} = V_T \ln \frac{I_D}{I_{S1}}$$

$$V_{D2} = V_T \ln \frac{I_D}{I_{S2}}$$

The OTA will force the voltage on top of the resistor to be equal to V_{D1}, thus the voltage across the resistor R_1 is

$$V_{D1} - V_{D2} = V_T \ln \frac{I_D}{I_{S1}} - V_T \ln \frac{I_D}{I_{S2}} = V_T \ln \frac{I_{S2}}{I_{S1}} = V_T \ln N$$

This is a remarkable result. The difference between two voltages is only defined by Boltzmann's constant, temperature, charge, and a known size difference.

This differential voltage can be used to read out directly the temperature on an IC, provided we can compare to a known voltage.

We often call this voltage ΔV_D or ΔV_{BE}, and we can see it's proportional to absolute temperature.

We know that the V_D decreases linearly with temperature, so if we combined a multiple of the ΔV_{BE} with a V_D voltage, then we should get a constant voltage.

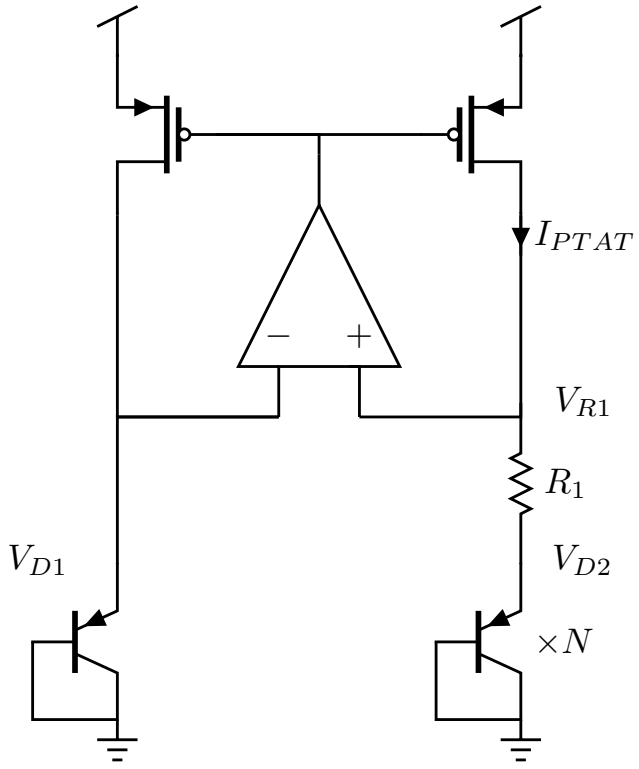


Figure 6: Circuit to create a PTAT current controlled by the resistor and ΔV_{BE}

18.2.3 How to combine a CTAT with a PTAT ?

One method is Figure 7. The voltage across resistor R_2 would compensate for the decrease in V_{D3}, as such, R_2 would be bigger than R_1.

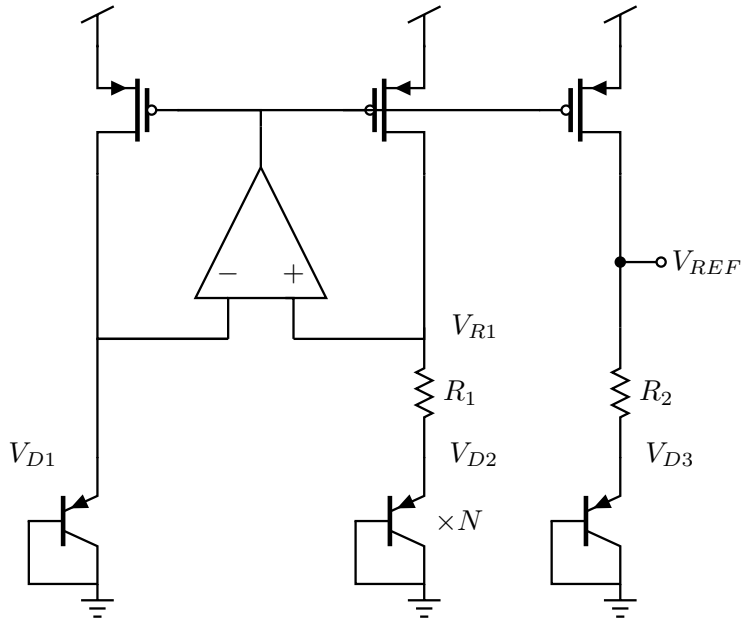


Figure 7: A bandgap voltage reference with a constant output voltage.

Another method would be to stack the R_2 on top of R_1 as shown in Figure 8.

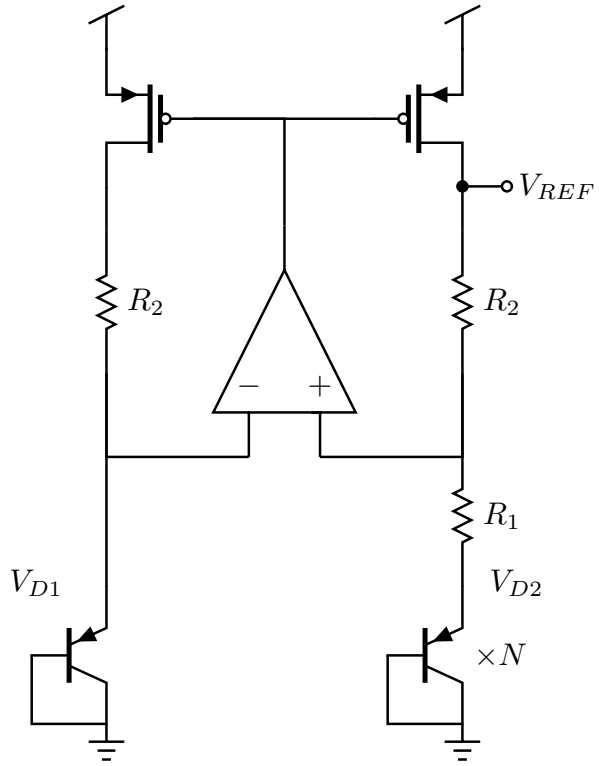


Figure 8: Another bandgap voltage reference with a constant output voltage.

18.2.4 Widlar reference

The first bandgap reference was not Brokaw's. Bob Widlar built one in 1971 for the LM113, three years before the Brokaw cell that follows, and it is worth starting here. Partly for the history, and partly because it does the entire job with three transistors, three resistors, and no amplifier anywhere. It was published in New developments in IC voltage regulators [1].

Figure 9 is the circuit. It is a two terminal shunt reference: you feed it a bias current down from the supply and it holds its own terminal at V_{REF}, the way a zener does, except that it does it at 1.2 V, where no zener will.

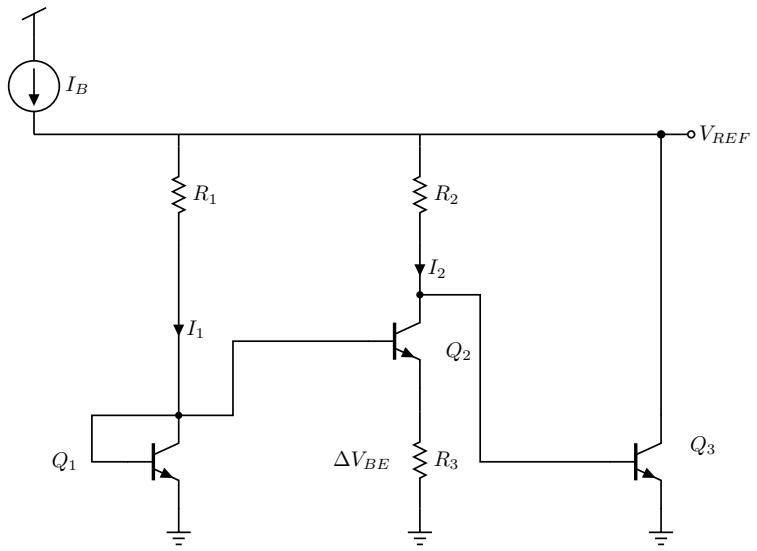


Figure 9: Widlar's bandgap reference, the first one, from 1971

Q_1 is diode connected, so its collector sits one V_{BE} above ground. Q_3 holds its own base, which is the collector of Q_2, one V_{BE} above ground as well. Both R_1 and R_2 therefore have very nearly the same voltage across them, and the current ratio falls out of the resistors alone.

$$\frac{I_1}{I_2} = \frac{R_2}{R_1}$$

Q_1 and Q_2 are the same size and share a base, so that difference in current density lands across R_3

$$I_2 R_3 = V_{BE1} - V_{BE2} = \frac{kT}{q} \ln \frac{I_1}{I_2} = \frac{kT}{q} \ln \frac{R_2}{R_1}$$

which is PTAT, and depends only on a resistor ratio, so it is as accurate as your matching. That current runs up through R_2, and the output is that drop stacked on top of the V_{BE} of Q_3.

$$V_{REF} = V_{BE3} + \frac{R_2}{R_3}\frac{kT}{q}\ln\frac{R_2}{R_1}$$

CTAT plus PTAT, and we are about to do it again with an amplifier. With $R_2/R_1 = 10$ the log term is about 60 mV at room temperature, $R_2/R_3 = 10$ scales that to 600 mV, and stacked on a 600 mV V_{BE} you land at the 1.2 V the LM113 was sold as.

Notice what is doing the job of the OTA. Q_3 is the gain element. If the terminal tries to rise, the collector of Q_2 follows it up, Q_3 conducts harder, and shunts the extra current away. The loop closes through a single transistor. That is why the circuit fits in a process where you count your transistors, and it is also why it has less loop gain, and therefore worse line regulation, than the amplifier based cells that came after it. Widlar was not short of ideas, he was short of devices.

One more thing worth taking from this circuit is where the current density ratio comes from. Here it is R_2/R_1, a ratio of resistors. In the Brokaw cell it is an emitter area ratio instead. Both work, and both are asking a ratio of like things to be accurate, which is the only kind of accuracy an integrated circuit actually has.

18.2.5 Brokaw reference

Paul Brokaw was a pioneer within reference circuits (I met him once in the restroom queue in Tropisueno behind the Marriot hotel in SF during ISSCC). Below is the Brokaw reference, which I think was first published in A simple three-terminal IC bandgap reference [2].

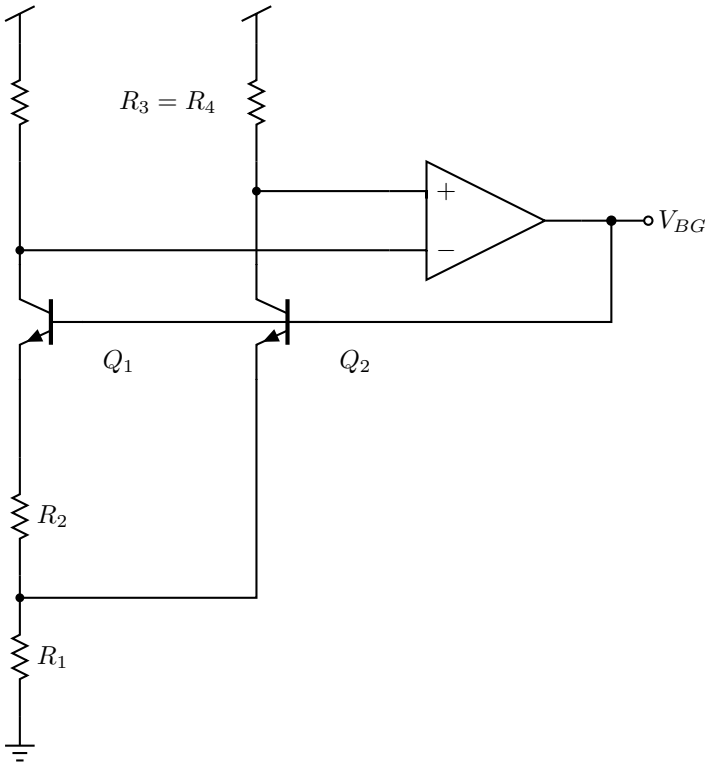


Figure 10: Brokaw bandgap voltage reference

The opamp ensures the two bipolars have the same current. Q_1 is larger than Q_2. The ΔV_{BE} is across the R_2, so we know the current I. We know that R_1 must then have $2I$.

The voltage at the output will then be.

$$V_{BG} = V_{G0} + (m-1)\frac{kT}{q}\ln\frac{T_0}{T} + T\left[\frac{k}{q}\ln\frac{J_2}{J_1}\frac{2R_1}{R_2} - \frac{V_{G0} - V_{be0}}{T_0}\right]$$

where V_{G0} is the bandgap extrapolated to zero Kelvin, V_{be0} is the base emitter voltage measured at a temperature T_0, the J's are the current densities, and m is the exponent that collects the temperature dependence of the saturation current and of the bias current - about 3 for a diode run at constant current, and we will come back to it in the curvature section.

Read the three terms. The first is a constant. The third is proportional to T, and the resistor ratio is the knob on it. The second is the awkward one: it is proportional to $T \ln T$, and no resistor ratio can touch it.

Now, what does "constant output" mean? The tempting answer is to make the bracket zero, which kills the term in bare T and leaves

$$V_{BG} = V_{G0} + (m-1)\frac{kT}{q}\ln\frac{T_0}{T}$$

That is **not** flat. Differentiate it: at $T = T_0$ the slope is $-(m-1)k/q \approx -170$ uV/K, or about -140 ppm/K, which is a hundred times worse than the reference you were trying to build. The $T \ln T$ term has a slope of its own, and setting the bracket to zero leaves it uncancelled.

What we actually want is zero *slope* at the temperature we care about. Take the derivative of the whole expression, set it to zero at T_0, and the condition is that the bracket must equal $(m-1)k/q$ rather than zero:

$$\frac{k}{q}\ln\frac{J_2}{J_1}\frac{2R_1}{R_2} = \frac{V_{G0} - V_{be0}}{T_0} + (m-1)\frac{k}{q}$$

so the resistor ratio is

$$\frac{R_1}{R_2} = \frac{V_{G0} - V_{be0} + (m-1)\frac{kT_0}{q}}{2T_0\frac{k}{q}\ln(\frac{J_2}{J_1})}$$

and the output voltage at that point is not the bandgap, but a little above it

$$V_{BG}(T_0) = V_{G0} + (m-1)\frac{kT_0}{q} \approx 1.25\ \text{V}$$

This is worth remembering, because it surprises people: a bandgap reference trimmed for zero temperature coefficient sits around 1.25 V, not at the 1.20 V bandgap it is named after. The extra 50 mV is exactly the price of cancelling the slope of the $T \ln T$ term at one temperature.

In typical simulations, the variation can be
low over the temperature range. The second order error is the remaining error from

$$V_{BG} = V_{G0} + (m-1)\frac{kT}{q}\ln\frac{T_0}{T} + T\left[\frac{k}{q}\ln\frac{J_2}{J_1}\frac{2R_1}{R_2} - \frac{V_{G0} - V_{be0}}{T_0}\right]$$

With the resistor ratio picked so that the slope vanishes at T_0, the bracket is $(m-1)k/q$, so the term in bare T becomes $(m-1)\frac{k}{q}T$ and what remains is

$$V_{BG} = V_{G0} + (m-1)\frac{kT}{q}\ln\frac{T_0}{T} + (m-1)\frac{k}{q}T$$

$$V_{BG} = V_{G0} + (m-1)\frac{kT}{q}\left[1 + \ln\frac{T_0}{T}\right]$$

a curve with zero slope at T_0 and a maximum there. Everywhere else it falls away, and that bow is the second order error we are left with.

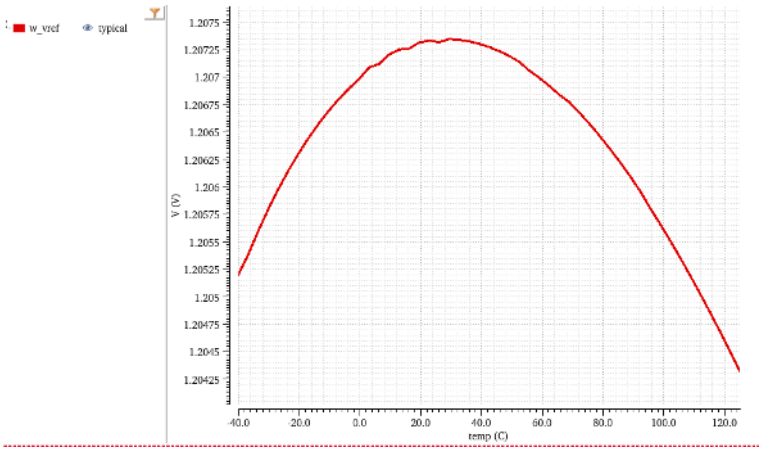


Figure 11: Simulation of a Brokaw reference in GF 130 nm

Read the axes before anything else. The whole vertical range is about 3 mV on an output of 1.207 V, so the curve you are looking at is flat to roughly 15 ppm/K over the range - the plot is magnified enormously.

Then look at the shape. It rises, peaks a little above room temperature, and falls away on both sides. That peak is not an accident: it is the temperature where we chose the slope to be zero, and the resistor ratio put it there. The bow either side of it is exactly the $T\,[1 + \ln(T_0/T)]$ term we could not cancel with a resistor ratio, and the curvature section later in this chapter is about getting rid of it.

Over corners, I do expect that there is variation, as we can see from Figure 12.

Two things move, and they are worth separating. The first is a vertical offset of roughly ±10 mV, about ±0.8 %: that is the absolute accuracy of the reference, and it comes from resistor and V_{BE} spread. The second is more interesting: the *peak moves*. The slow corner is still climbing at 125 C while the fast corner has already turned over near 0 C.

A moving peak means the linear balance has shifted, not that the curvature term misbehaved. The bracket we so carefully set to

$(m-1)k/q$ is only zero at typical: when the resistors and V_{BE} walk to a corner, the bracket picks up a residue, and a residue in the bracket is a term proportional to T, which tilts the whole curve and drags the maximum with it.

We could include trimming of PTAT to calibrate for the remaining error, however, if we wanted to remove the linear gradient, we would need a two point temperature test of every IC, which is too expensive for low-cost devices.

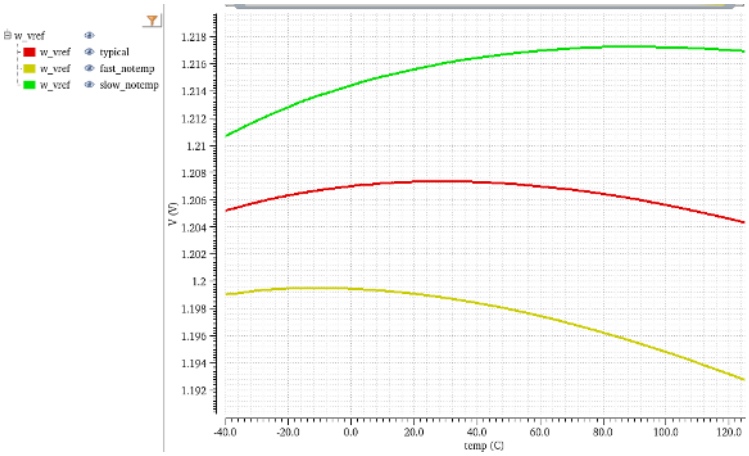


Figure 12: Typical, slow and fast corner simulation of the Brokaw bandgap. The legend's "notemp" corners hold temperature-dependent model parameters at their typical values, so the spread shown is process alone

18.2.6 Low voltage bandgap

The Brokaw reference, and others, have a 1.2 V output voltage, which is hard to make if your supply is below about 1.4 V. As such, people have investigated lower voltage references. The original circuit was presented by Banba A CMOS bandgap reference circuit with sub-1-V operation [3]

In real ICs though, you should ask yourself long and hard whether you really need these low-voltage references. Most ICs today still have a high voltage, either 1.8 V or 3.0 V.

If you do need them, consider the circuit in Figure 13. We have two diodes at different current densities. The ΔV_D will be across R_1. The voltage at the input of the OTA will be V_D and the OTA will ensure the both inputs are equal.

The current will then be

$$I_1 = \frac{\Delta V_D}{R_1}$$

and we know the current increases with temperature, since ΔV_D increases with temperature.

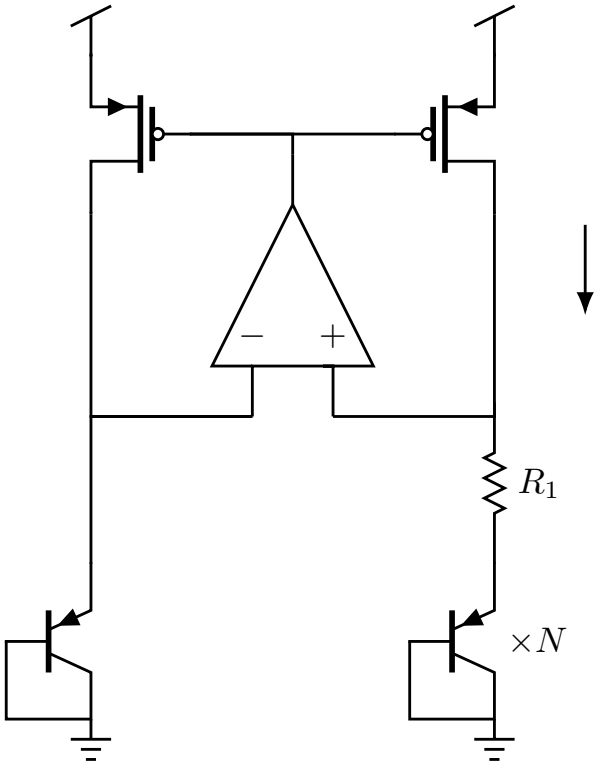


Figure 13: PTAT current generator

I use ΔV_{BE} and ΔV_D interchangeably, apologies.

In Figure 14 we copy the V_D to another node, and place it across a second resistor R_2.

The current in this second resistor is then

$$I_2 = \frac{V_D}{R_2}$$

and we know the current decreases with temperature, since V_D decreases with temperature.

From before, we know the current in R_1 is proportional to temperature. As such, if we combine the two current with the correct proportions, then we can get a current that does not change with temperature.

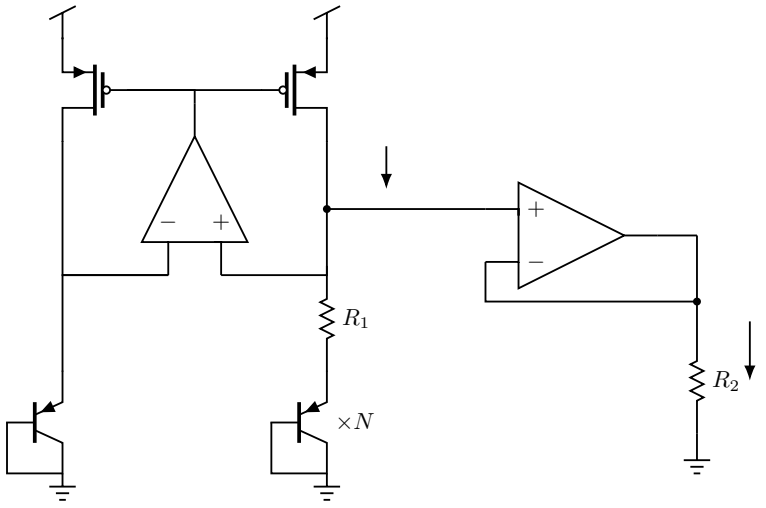


Figure 14: Extending the PTAT current generator

Let's remove the OTA, and connect R_2 directly to V_D nodes, as shown in Figure 15.

You should convince yourself of the fact that this does not change I_1.

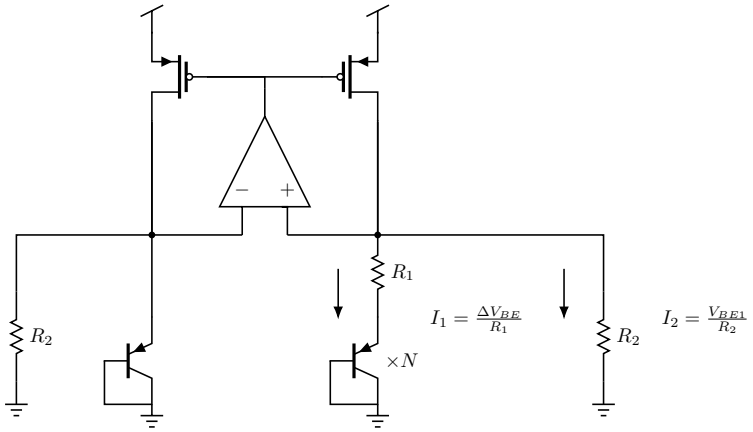


Figure 15: The Banba bandgap voltage reference core

It does, however, change the current in the PMOS. Provided we scale R_2 correctly, then the PTAT I_1 can compensate for CTAT I_2, and we have a current that is independent of temperature.

$$I_{PMOS} = \frac{V_D}{R_2} + \frac{\Delta V_D}{R_1}$$

Assuming we copy the current into another resistor R_3, as shown in Figure 16, we can get a voltage that is

$$V_{OUT} = R_3 \left[\frac{V_D}{R_2} + \frac{\Delta V_D}{R_1} \right]$$

We can choose the output voltage freely, and it can be lower than 1.2 V.

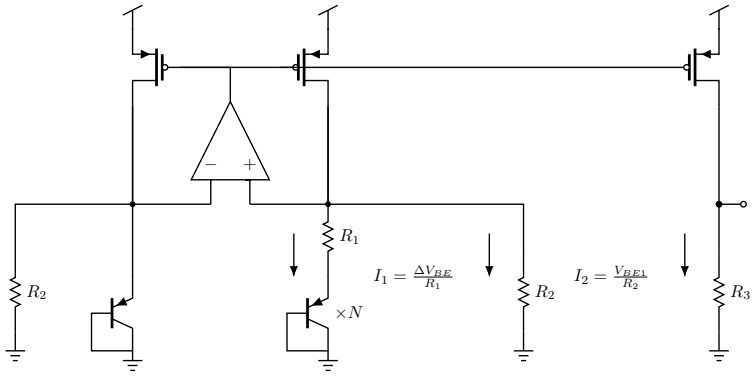


Figure 16: The Banba bandgap voltage reference

18.2.7 Curvature correction

Go back and look at Figure 12 again. Over corners the reference is not flat, and even the typical curve in Figure 11 has a bend in it. That bend is not noise, and it is not a mistake in the design. It is a term we agreed to ignore, and it is time to stop ignoring it.

We picked the resistor ratio so the *slope* vanishes at one temperature. That flattens the curve where we chose to flatten it, and it does nothing at all to the shape of what is left,

$$V_{BG} = V_{G0} + (m-1)\frac{kT}{q}\left[1 + \ln\frac{T_0}{T}\right]$$

and that is the bow you can see in Figure 11. It comes from the temperature dependence of I_S - the $-3\ln T$ we carried through the V_{BE} algebra earlier - together with the temperature dependence of the bias current itself. That is where the -1 comes from: if the diode current is PTAT, as it is in every circuit in this chapter, it contributes one power of T and the coefficient becomes $m-1$ rather than m. With $m \approx 3$ the coefficient is about 2. Be careful with that number: the 3 assumes temperature independent diffusion, and measured devices sit nearer 3.6 to 4, so a design that trims R_4 from theory alone will be off.

No choice of R_1/R_2 can remove it. A resistor ratio can only add something proportional to T, and what is left over is proportional to $T\ln T$. If we want to cancel it, we have to build a $T\ln T$ term.

Here is where one comes from. Take two identical bipolars at the same temperature. The difference of their base-emitter voltages is

$$V_{BE,A} - V_{BE,B} = \frac{kT}{q} \ln \frac{I_A}{I_B}$$

and this one is exact, not an approximation. I_S cancels completely, because it is the same device at the same temperature.

This is the same ΔV_{BE} we have used all chapter, and every time so far the current ratio has been a fixed number set by device sizes. That is what made it PTAT. So make the ratio depend on temperature instead: bias Q_A with a PTAT current, and Q_B with the temperature compensated current the reference already produces. Then $I_A/I_B = KT/T_0$ and

$$V_{BE,A} - V_{BE,B} = \frac{kT}{q} \ln K + \frac{kT}{q} \ln \frac{T}{T_0}$$

The first term is PTAT, and we know what to do with those. The second term is the $T \ln T$ we needed, and it comes with the right sign.

Figure 17 turns that voltage into a current. The OTA holds the right hand end of R_4 at $V_{BE,A}$, the left hand end sits on $V_{BE,B}$, and M_{PC} supplies whatever current that requires. M_{PD} copies it into the summing node from Figure 16, so the V_{OUT} of that circuit becomes

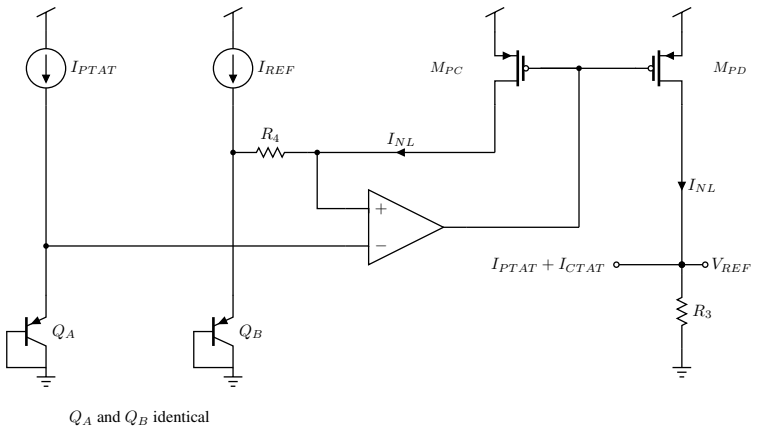


Figure 17: Curvature correction. Q_A and Q_B are the same device at different bias currents, so the voltage across R_4 carries a $T \ln T$ term.

$$I_{NL} = \frac{V_{BE,A} - V_{BE,B}}{R_4} = \frac{kT}{qR_4} \left[\ln K + \ln \frac{T}{T_0} \right]$$

$$V_{REF} = R_3 \left[\frac{V_D}{R_2} + \frac{\Delta V_D}{R_1} + I_{NL} \right]$$

The curvature the V_D term brings in is $\frac{R_3}{R_2}(m-1)\frac{kT}{q}\ln\frac{T_0}{T}$, the curvature the new branch adds is $\frac{R_3}{R_4}\frac{kT}{q}\ln\frac{T}{T_0}$, and they cancel when

$$R_4 = \frac{R_2}{m-1}$$

for which R_2 and R_4 must be the same kind of resistor: the ratio only holds over temperature if their temperature coefficients cancel.

which is a nice result. The correction is set by a resistor ratio, like everything else in this chapter, and with $m \approx 3$ it makes R_4 about half of R_2. The $\frac{kT}{q}\ln K$ half of I_{NL} is PTAT, so it simply adds to the PTAT current already there, and you retrim R_1 to take it back out.

Three things to watch.

I_{NL} leaves R_4 and flows into the emitter of Q_B, so Q_B does not carry exactly I_{REF}. It is a small perturbation, but it is real, and it makes the sizing slightly iterative.

$V_{BE,A}$ has to stay above $V_{BE,B}$ across the whole temperature range, or the current in R_4 reverses and the loop runs out of room. The ratio is KT/T_0, so pick K large enough that it is still comfortably above one at the cold end.

And m is not a number the foundry hands you to three digits. Curvature correction typically buys a factor of five to ten in temperature drift, not a factor of a thousand, and what it buys is limited by how well you know m. It is worth the area when you need 10 ppm/°C. It is a waste of area when 50 ppm/°C is fine, which it usually is.

18.2.8 MOS references

Recognise this one. Do not build it.

Everything so far has needed a bipolar. In a pure CMOS process that is irritating, and the temptation is obvious. The MOS transistor has a threshold voltage, thresholds fall with temperature at roughly -1 mV/K, and if two devices have different thresholds then the difference between them ought to be flat. A reference with no bipolar in it.

If you read older books you will find this done with an enhancement device and a depletion device. Forget that circuit. It relied on a threshold below zero, so that a device with its gate tied to its own source still conducted, and in a nanoscale CMOS process there is no such device. Essentially everything on the menu sits at 300 mV

or more. What you get instead is a handful of implant flavours, standard, low V_t and high V_t, separated by a hundred millivolts or so, plus the native device, which skips the channel implant altogether and is the only one that lands anywhere near zero.

So the modern version of the idea pairs two of those flavours, and it looks like Figure 18.

You have seen this loop before, or rather you are about to see it: it is the GM cell from later in this chapter, with the size ratio taken out and a second implant put in its place. The PMOS mirror forces the same current down both branches, M_1 and M_2 have the same W/L, and the only thing left that distinguishes them is which channel implant they were given.

Both gates sit on the same node, so $V_{GS1} = V_{GS2} + IR$, and writing each gate-source voltage as a threshold plus an overdrive

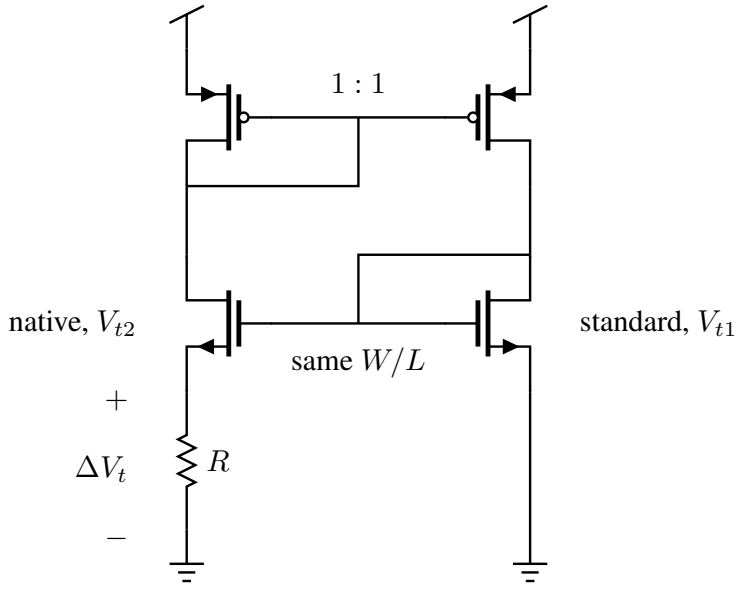


Figure 18: A reference built on the difference between two threshold voltages. Learn to recognise it. Do not build it.

$$IR = (V_{t1} + V_{eff1}) - (V_{t2} + V_{eff2})$$

Equal current in equal W/L means equal overdrive, so the V_{eff} terms cancel and what is left is the entire reference

$$I = \frac{V_{t1} - V_{t2}}{R} = \frac{\Delta V_t}{R}$$

MOS based references that rely on the difference between two threshold voltages are very risky and should not be attempted.

I want you to leave this section able to recognise that circuit, and unwilling to build it. Process control over the two threshold sources

is poor, and their stability is poor, so the difference is neither well controlled nor stable.

Put Figure 18 next to Figure 20 and the problem is visible in one look. They are the same loop. The GM cell puts a 4:1 size ratio in it, and a size ratio is lithography, two drawings of the same thing, which is about as well controlled as anything on a die gets. This circuit puts the difference of two channel implants in the same place. Those are two separate recipes, aimed separately, monitored separately, and drifting separately. Nothing makes them move together.

Watch what the numbers do. Take standard against low V_t: two devices at maybe 450 mV and 350 mV, each with a spread of ±50 mV over process. Each threshold on its own is known to about ±12 %. The difference you actually use is 100 mV, and if the two spreads are independent it is known to about ±70 %. If they happen to move in opposite directions it is worse than that, and nothing in the process says they will not. Subtracting two similar, poorly known numbers is the worst thing you can do to an error budget.

18.2.9 The native threshold is set when the ingot is grown

Reach for the native device to get a bigger difference and you have picked the least controlled transistor in the process. Its threshold is whatever is left when you leave the channel implant out, which means it rides on the doping of the silicon underneath it, and that is not a number the fab sets.

Think about where it does get set. Not in an implanter, where the dose is metered to about a percent and measured on every lot. It is fixed when the ingot is grown, by how much boron went into the melt before the crystal was pulled out of it, and the boule is usually not even grown by the company that builds your chip. You buy wafers, and what you buy is a resistivity range, not a resistivity.

It is worse than a range. Boron has a segregation coefficient below one, so it would rather stay in the melt than join the crystal. As the boule is pulled the melt gets steadily richer, and the silicon that freezes out near the tail end is more heavily doped than the silicon at the seed end, by tens of percent over the length of one crystal. Wafers sliced from the two ends of the same boule are not the same wafer. There is a radial gradient across each wafer on top of that, and well proximity and STI stress move it again locally.

None of this is visible to the fab's process control, which watches implants and etches, and none of it is correctable there either. So one end of your subtraction is set by an implanter recipe inside the fab, and the other end is set by a crystal grower at a different company,

to a different specification, probably on a different continent. Those two numbers have no reason on earth to move together. In most PDKs the native device duly comes with the widest corners and the thinnest model guarantees of anything on the menu.

If the process runs on epitaxial wafers the native device sees the epi layer rather than the bulk, and epi doping is better controlled than a pulled crystal. It is still a deposition specification rather than a metered implant, and it is still usually somebody else's specification.

Compare that with what a bandgap does. A bandgap gets its number from E_g of silicon. That is a property of the material, it is the same in every fab on earth, and no process engineer can move it. This circuit gets its number from the difference between an implant recipe and a crystal pull.

Stability is the other half, and trimming does not save you. You can trim out the initial spread in production test, once. Then the thresholds move over the life of the part. NBTI and hot carrier stress shift them, and they shift by different amounts, because the two devices see different fields, different bias and different channel doping. Nothing anchors the difference, so it walks.

The derivation has a hole in it as well. It assumed the two devices are identical apart from V_t, which is the only reason the overdrives cancelled. They are not. The implant that moves the threshold also moves the mobility, the body effect coefficient and the subthreshold slope, so V_{eff1} and V_{eff2} do not quite cancel, and what is left over has its own temperature dependence. On top of that M_2 has its source ΔV_t above ground while its bulk is at ground, so the body effect raises V_{t2} by an amount that depends on the answer.

If you need a reference in a pure CMOS process, use the parasitic vertical PNP that every CMOS process has, in the circuits from earlier in this chapter. If what you actually need is a bias current rather than a reference, use the GM cell from later in this chapter. Neither of those asks two implants to agree with each other.

18.2.10 FD-SOI moves the problem, it does not remove it

Everything so far has been a bulk story, and you should ask what happens in a fully depleted process. In FD-SOI the channel is undoped silicon a few nanometres thick sitting on buried oxide, and the threshold is not set by channel doping at all. It is set by the work function of the gate stack, and by the doping of the back plane under the oxide, which is what a regular well and a flip well actually are.

That fixes something real. An undoped channel has no random dopant fluctuation, and random dopant fluctuation is the dominant source of local V_t mismatch in bulk. Two FD-SOI devices side by side match far better than two bulk devices of the same area. If mismatch were the objection, FD-SOI would answer it.

Mismatch was never the objection. The objection is that the two ends of the subtraction are set by unrelated recipes, and in FD-SOI they still are: one by a metal gate work function, the other by an implant under the oxide. Those two have no more reason to track each other than two channel implants did.

The supplier problem does not go away either, it changes address. Fully depleted means the threshold depends on how thick the silicon film is, and that film is a few nanometres of silicon bonded onto oxide by the wafer maker rather than grown by your fab. You have traded a resistivity specification you do not control for a thickness specification you do not control.

18.2.11 The back gate is a trimming knob, not a reference

The back gate is the genuinely interesting part of FD-SOI here. It moves V_t by tens of millivolts per volt, and it moves it electrically, after the wafer is finished. That is a wonderful trimming knob, and it is one of the better reasons to be in FD-SOI at all. It is not a reference, though. Setting a threshold with a voltage means generating that voltage first, and you are back where you started.

One footnote on the advice I just gave, since it was written for bulk. The vertical PNP does not exist in the thin film, so in FD-SOI you cannot simply reach for it. What you do instead is put the bandgap in a hybrid opening, where the film and the buried oxide are removed and you are looking at ordinary bulk silicon again. Every FD-SOI process gives you those, because the I/O and the ESD need them too. The circuit is the same circuit. It just needs somewhere to live.

18.3 Bias

Sometimes we just need a current

18.3.1 Voltage to current conversion

With a known voltage, we can convert to a known current with the circuit in Figure 19.

On-chip we don't have accurate resistors, but for bias currents, it's usually ok with ±20 % variation (the variation of R).

Across an IC, we can expect the resistors to match within 2 % percent, as such, we can recreate a voltage with an accuracy of about 2 % difference from the original if we have a second resistor on the other side of the IC.

If we wanted to create an accurate current, then we'd trim the R in production test until the current is what we want.

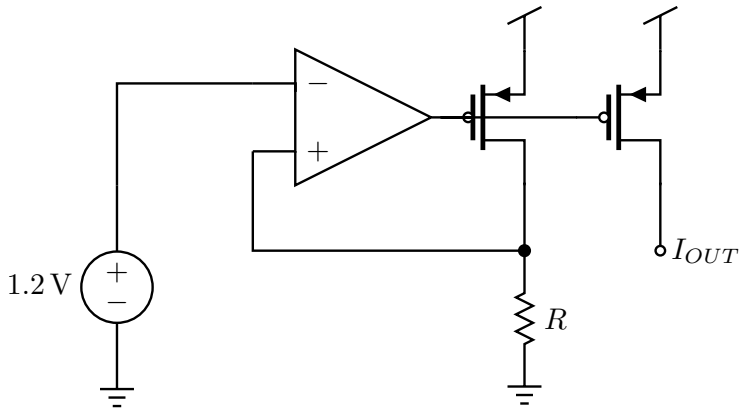


Figure 19: Voltage to current converter

18.3.2 GM Cell

Sometimes we don't need a full bandgap reference. In those cases, we can use a GM cell, as shown in Figure 20.

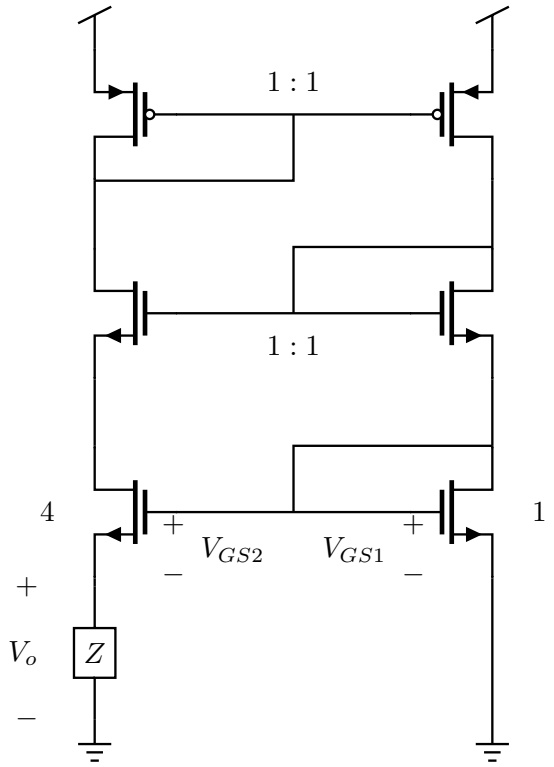


Figure 20: GM cell.

The top PMOS current mirror ensures that both branches have the same current. The middle NMOS current mirror copies the drain voltage on top of the diode connected bottom NMOS to the left NMOS. Consider the bottom transistors, those marked with "1" and "4". The V_o voltage is

$$V_o = V_{GS1} - V_{GS2} = V_{eff1} + V_{tn} - V_{eff2} - V_{tn} = V_{eff1} - V_{eff2}$$

Assuming transistors in strong inversion, then

$$I_{D1} = \frac{1}{2}\mu_n C_{ox}\frac{W_1}{L_1}V_{eff1}^2$$

$$I_{D2} = \frac{1}{2}\mu_n C_{ox}4\frac{W_1}{L_1}V_{eff2}^2$$

$$I_{D1} = I_{D2}$$

$$\frac{1}{2}\mu_n C_{ox}\frac{W_1}{L_1}V_{eff1}^2 = \frac{1}{2}\mu_n C_{ox}4\frac{W_1}{L_1}V_{eff2}^2$$

$$V_{eff1} = 2V_{eff2}$$

Inserted into above

$$V_o = V_{eff1} - \frac{1}{2}V_{eff1} = \frac{1}{2}V_{eff1}$$

Still assuming transistors in strong inversion, such that

$$g_m = \frac{2I_d}{V_{eff}}$$

we find that

$$I = \frac{V_{eff1}}{2Z}$$

so the impedance sets the transconductance directly

$$g_{m1} = \frac{1}{Z}$$

If we use a resistor for Z, then the transconductance is set by, and *inversely* proportional to, that resistor: $g_{m1} = 1/R$. That is the whole point of the circuit, and it is why it is called a constant g_m bias. The transconductance no longer depends on mobility, oxide thickness or threshold voltage - all the things that move with process and temperature - only on a resistor and a device ratio. With a general ratio K between the two devices the result is $g_{m1} = \frac{2}{R}\left(1 - \frac{1}{\sqrt{K}}\right)$, which is $1/R$ at the $K = 4$ drawn here.

We can use other things for Z, like a switched capacitor. A capacitor C_1 toggled between two nodes at a frequency f moves a charge $C_1\Delta V$ every cycle, which is an average current $fC_1\Delta V$, so it behaves as a resistance

$$Z = \frac{1}{fC_1}$$

and the transconductance becomes $g_{m1} = fC_1$: set by a clock frequency and a capacitor ratio rather than by a resistor. That is attractive, because a frequency can come from a crystal and a capacitor matches better than a resistor - at the price of the switching noise the switched capacitor chapter worries about.

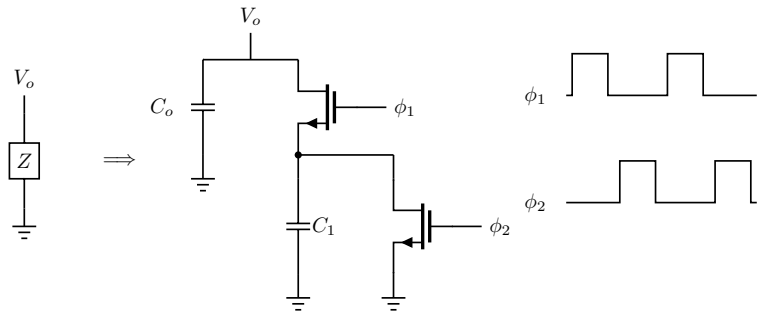


Figure 21: A switched capacitor used as the impedance Z, giving $g_{m1} = fC_1$

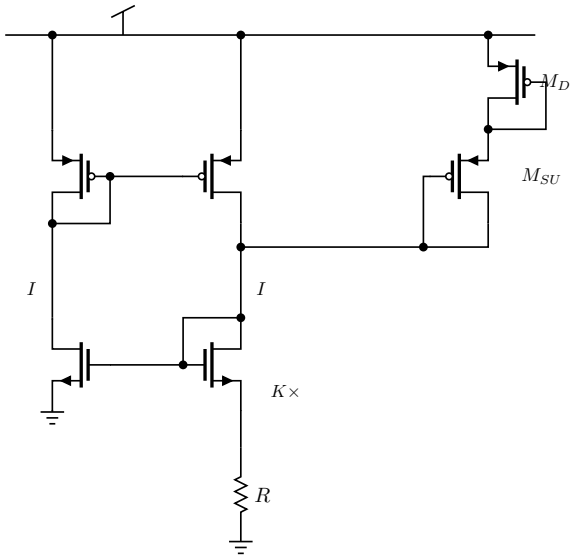
18.3.3 Every one of these loops can fail to start

There is something missing from every self biased circuit in this chapter, and it is the thing most likely to make your first bias circuit fail in silicon.

Look at the GM cell again, or at any of the OTA based bandgaps. The PMOS mirror says the two branch currents must be equal. The NMOS pair says what that current has to be. Together they have a solution, the one we designed for. But read the statement again: *the two branches must agree*. Zero current in both branches agrees perfectly well. Every equation in the loop is satisfied, the PMOS

rail sits at the supply, the NMOS rail sits at ground, every device is off, and nothing in the circuit has any reason to change.

That is a second, perfectly stable operating point, and a DC simulation is entitled to find either one. Worse, a DC simulation often finds the one you wanted, because the solver started its guess somewhere helpful - and then the chip comes back and the reference never wakes up.



The loop only says the two branches must agree, and zero current in both agrees just as well, so a startup device is not optional. M_{SU} lifts the NMOS rail off ground until the loop takes over. M_D costs it a threshold of drive, which is what makes it let go afterwards instead of leaking into a small branch current for ever.

Figure 22: A constant g_m bias with a startup branch. M_{SU} lifts the NMOS rail while it is stuck at ground, and M_D is what makes it let go once the loop is running

The fix is a device that is *only* on in the dead state, and the first question is which of the two dead rails it should attack. Only one of them can be moved cheaply. The PMOS rail is stuck at V_{DD}, and there is nothing useful a device hanging off the supply can do to a node already at the supply. The NMOS rail is stuck at ground, and a PMOS from V_{DD} can lift it. That is the one to go after.

So M_{SU} is a PMOS from the supply onto the NMOS rail, with its gate on that same rail. While the loop is asleep the rail is at ground, M_{SU} has the full supply from gate to source, and it conducts and lifts the rail until the NMOS pair turns on and the loop takes over. As the rail rises, M_{SU}'s own drive falls, so it backs off without being told to.

It does not back off far enough, and this is the part that bites in a low-power design. Awake, the NMOS rail only climbs to a gate-source

drop, call it 400 mV, which leaves M_{SU} with $|V_{GS}| = V_{DD} - 400$ mV. In a low-threshold process that is still an on transistor, quietly injecting current into the rail it was supposed to have released. If the loop's own branch current is a few hundred nanoamps, that injection is not a rounding error — it sets the reference.

M_D is the fix. A second diode-connected PMOS in series means the startup branch needs two thresholds of headroom instead of one, so it stops conducting once the rail has risen. Asleep, the full supply across two diodes is still plenty to get things moving. The cost is a little headroom in a branch that only matters before the reference wakes up, which is the cheapest headroom in the circuit.

Two rules follow from this, and they are worth more than the circuit:

- **Always simulate startup as a transient from zero supply**, not from a DC operating point. Ramp V_{DD} over a realistic time, and check that the reference comes up every corner, at every temperature, at the slowest supply ramp you can imagine. A bias circuit that only starts in a fast ramp is a bias circuit that will fail on a slow one.
- **Check that the startup device really turns off.** If it keeps conducting it becomes a leakage path that sets your reference, and the reference is then whatever the startup device felt like, not what the bandgap said.

18.4 Summary

The one-page version of this chapter:

- A reference must not move with supply, temperature or process; a bias must track what its circuit needs
- V_BE falls with temperature (CTAT), the difference of two V_BE at a density ratio rises (PTAT, V_T ln N): weight and add for a flat bandgap near 1.2 V
- Widlar and Brokaw are the classic ways to force the PTAT current and do the sum
- Distribute currents, not voltages: a routed V_B collects every IR drop on the die
- The constant-gm loop sets gm = 1/R for everything it biases
- Every self-biased loop is equally happy at zero current - the startup branch is not optional, and it must let go afterwards

18.5 Would you like to know more?

New developments in IC voltage regulators [1]

A simple three-terminal IC bandgap reference [2]

A CMOS bandgap reference circuit with sub-1-V operation [3]

A sub-1-V 15-ppm//spl deg/C CMOS bandgap voltage reference without requiring low threshold voltage device [4]

The Bandgap Reference [5]

The Design of a Low-Voltage Bandgap Reference [6]

[1] R. Widlar, "New developments in IC voltage regulators," *IEEE Journal of Solid-State Circuits*, vol. 6, no. 1, pp. 2–7, 1971, doi: 10.1109/JSSC.1971.1050151.

[2] A. Brokaw, "A simple three-terminal IC bandgap reference," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 6, pp. 388–393, 1974, doi: 10.1109/JSSC.1974.1050532.

[3] H. Banba *et al.*, "A CMOS bandgap reference circuit with sub-1-v operation," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 5, pp. 670–674, 1999, doi: 10.1109/4.760378.

[4] K. N. Leung and P. Mok, "A sub-1-v 15-ppm//spl deg/c CMOS bandgap voltage reference without requiring low threshold voltage device," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 4, pp. 526–530, 2002, doi: 10.1109/4.991391.

[5] B. Razavi, "The bandgap reference [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 3, pp. 9–12, 2016, doi: 10.1109/MSSC.2016.2577978.

[6] B. Razavi, "The design of a low-voltage bandgap reference [the analog mind]," *IEEE Solid-State Circuits Magazine*, vol. 13, no. 3, pp. 6–16, 2021, doi: 10.1109/MSSC.2021.3088963.

Analog frontend and filters 19

Keywords: H(s), BiQuad, Gm-C, Active-RC, OTA

19.1 Introduction 311
19.2 Filters 314
19.2.1 First order filter . . . 314
19.2.2 Second order filter . 315
19.2.3 How do we implement the filter sections? 316
19.3 Gm-C 316
19.3.1 Differential Gm-C . 317
19.3.2 Finding a transconductor 319
19.4 Active-RC 320
19.4.1 General purpose first order filter 320
19.4.2 General purpose biquad 323
19.5 The OTA is not ideal 324
19.6 Example circuit . . . 325
19.7 My favorite OTA . . 327
19.8 Summary 329
19.9 Would you like to know more? 329

19.1 Introduction

The world is analog, and these days, we do most signal processing in the digital domain. With digital signal processing we can reuse the work of others, buy finished IPs, and probably do processing at lower cost than for analog.

Analog signals, however, might not be suitable for conversion to digital. A sensor might have a high, or low impedance, and have the signal in the voltage, current, charge or other domain.

To translate the sensor signal into something that can be converted to digital we use analog front-ends (AFE). How the AFE looks will depend on application, but it's common to have amplification, frequency selectivity or domain transfer, for example current to voltage.

An ADC will have a sample rate, and will alias (or fold) any signal above half the sample rate, as such, we also must include a anti-alias filter in AFE that reduces any signal outside the bandwidth of the ADC as close to zero as we need.

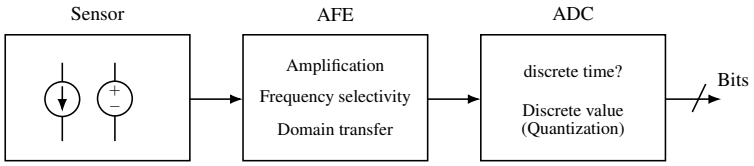


Figure 1: Analog signal chain from sensor through AFE (amplification, frequency selectivity, domain transfer) to ADC

One example of an analog frontend is the receive chain of a typical Bluetooth radio. The signal that arrives at the antenna, or the "sensor", can be weak, maybe -90 dBm.

At the same time, at another frequency, there could be a unwanted signal, or blocker, of -30 dBm

Assume for the moment we actually used an ADC at the antenna, how many bits would we need?

Bluetooth uses Gaussian Frequency Shift Keying, which is a constant envelope binary modulation, and it's usually sufficient with

low number of bits, assume 8-bits for the signal is more than enough.

If we assume the maximum of the ADC must fit the blocker, and the resolution must resolve the wanted signal, we get the numbers in the table below.

What	Power [dBm]	Voltage [V]
Blocker	-30	7 m
Wanted	-90	7 u
Resolution		Wanted/255 = 28 n

Then we can calculate the number of bits as

$$\text{ADC resolution } \Rightarrow \ln \frac{7\text{ mV}}{28\text{ nV}} / \ln 2 \approx 18 \text{ bits}$$

If we were to sample at 5 GHz, to ensure the bandwidth is sufficient for a 2.480 GHz maximum frequency we can actually compute the power consumption.

Given the Walden figure of merit of

$$FOM = \frac{P}{2^{ENOB} fs}$$

The best FOM in literature is about 1 fJ/step, so

$$P = 1 \text{ fJ/step} \times 2^{18} \times 5\text{GHz} = 1.31 \text{ W}$$

If we look at a typical system, like the Whoop. We can have a look at teardowns, to find the battery size.

Whoop battery is 205mAh at 3.8 V

Then we can compute the lifetime running an ADC based Bluetooth Radio

$$\text{Hours} = \frac{205 \text{ mAh}}{1.31 \text{ W}/3.8 \text{ V}} = 0.6 \text{ h}$$

I know my whoop lasts for almost a week, so it can't be what Bluetooth ICs do.

I know a little bit about radio's, especially inside the Whoop, since it has

Nordic Inside

I can't tell you how the Nordic radio works, but I can tell you how others usually make their radio's. The typical radio below has multiple blocks in the AFE.

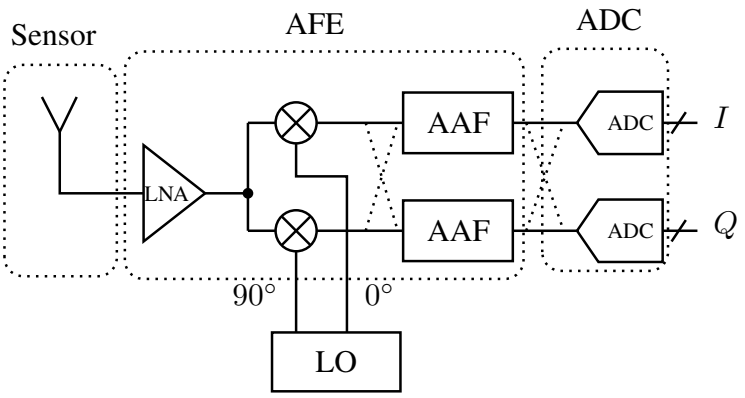


Figure 2: Typical radio receiver frontend: LNA, complex mixer driven by a local oscillator, anti-alias filters and ADCs for the I and Q paths

First is low-noise amplifier (LNA) amplifying the signal by maybe 10 times. The LNA reduces the noise impact of the following blocks. The next is the complex mixer stage, which shifts the input signal from radio frequency down to a low frequency, but higher than the bandwidth of the wanted signal. Then there is a complex anti-alias filter, also called a poly-phase filter, which rejects parts of the unwanted signals. Lastly there is a complex ADC to convert to digital.

In digital we can further filter to select exactly the wanted signal. Digital filters can have high stop band attenuation at a low power and cost. There could also be digital mixers to further reduce the frequency.

The AFE makes the system more efficient. In the 5 GHz ADC output, from the example above, there's lot's of information that we don't use.

An AFE can reduce the system power consumption by constraining digital information processing and conversion to the parts of the spectrum with information of interest.

There are instances, though, where the full 2.5 GHz bandwidth has useful information. Imagine in a cellular base station that should process multiple cell-phones at the same time. In that case, it could make sense with an ADC at the antenna.

What make sense depends on the application.

19.2 Filters

A filter can be fully described by the transfer function, usually denoted by $H(s) = \frac{\text{output}}{\text{input}}$.

Most people will today start design with a high-level simulation, in for example Matlab, or Python. Once they know roughly the transfer function, they will implement the actual analog circuit.

For us, as analog designers, the question becomes "given an $H(s)$, how do we make an analog circuit?" It can be shown that a combination of 1'st and 2'nd order stages can synthesize any order filter.

Once we have the first and second order stages, we can start looking into circuits.

19.2.1 First order filter

In the book they use signal flow graphs to show how the first order stage can be generated. By selecting the coefficients k_0 ,k_1 and ω_0 we can get any first order filter, and thus match the $H(s)$ we want.

I would encourage you to try and derive from the signal flow graph the $H(s)$ and prove to your self the equation is correct.

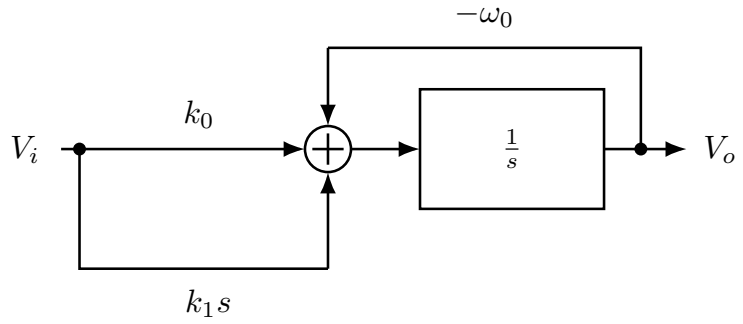


Figure 3: Signal flow graph of a general first order filter with coefficients k_0, k_1 and ω_0

Signal flow graphs are useful when dealing with linear systems.

The instructions to compute the transfer functions are

1. any line with a coefficient is a multiplier
2. any box output is a multiplication of the coefficient and the input
3. any sum, well, sum all inputs
4. be aware of gremlins (a sudden -+ swap)

$$H(s) = \frac{V_o(s)}{V_i(s)} = \frac{k_1 s + k_0}{s + w_o}$$

Let's call the $1/s$ box input u

$$u = (k_0 + k_1 s)V_i - \omega_0 V_o$$

$$V_o = u/s$$

$$u = V_o s = (k_0 + k_1 s)V_i - \omega_0 V_o$$

$$(s + \omega_0)V_o = (k_0 + k_1 s)V_i$$

$$\frac{V_o}{V_i} = \frac{k_1 s + k_0}{s + \omega_0}$$

19.2.2 Second order filter

Bi-quadratic is a general purpose second order filter.

Bi-quadratic just means "there are two quadratic equations". Once we match the k's ω_0 and Q to our wanted $H(s)$ we can proceed with the circuit implementation.

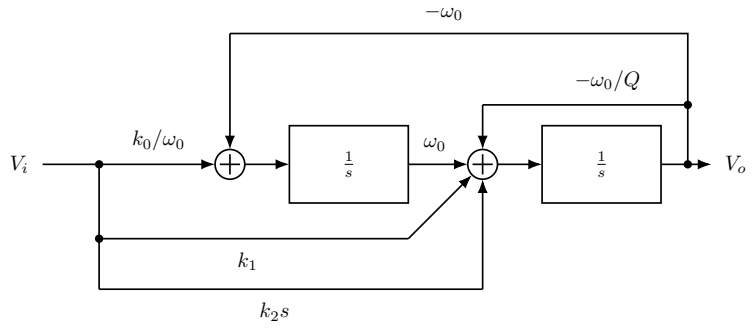


Figure 4: Signal flow graph of the bi-quadratic (second order) filter

$$H(s) = \frac{k_2 s^2 + k_1 s + k_0}{s^2 + \frac{\omega_0}{Q} s + \omega_o^2}$$

Follow exactly the same principles as for first order signal flow graph. If you fail, and you can't find the problem with your algebra, then maybe you need to use Maple or Mathcad.

I guess you could also spend hours training on examples to get better at the algebra. Personally I find such tasks mind numbingly boring, and of little value. What's important is to remember that you can always look up the equation for a bi-quad in a book.

19.2.3 How do we implement the filter sections?

While I'm sure you can invent new types of filters, and there probably are advanced filters, I would say there is roughly three types. Passive filters, that do not have gain. Active-RC filters, with OTAs, and usually trivial to make linear. And Gm-C filters, where we use the transconductance of a transistor and a capacitor to set the coefficients. Gm-C are usually more power efficient than Active-RC, but they are also more difficult to make linear.

In many AFEs, or indeed Sigma-Delta modulator loop filters, it's common to find a first Active-RC stage, and then Gm-C for later stages.

19.3 Gm-C

In the figure below you can see a typical Gm-C filter and the equations for the transfer function. One important thing to note is that this is Gm with capital G, not the g_m that we use for small signal analysis.

In a Gm-C filter the input and output nodes can have significant swing, and thus cannot always be considered small signal.

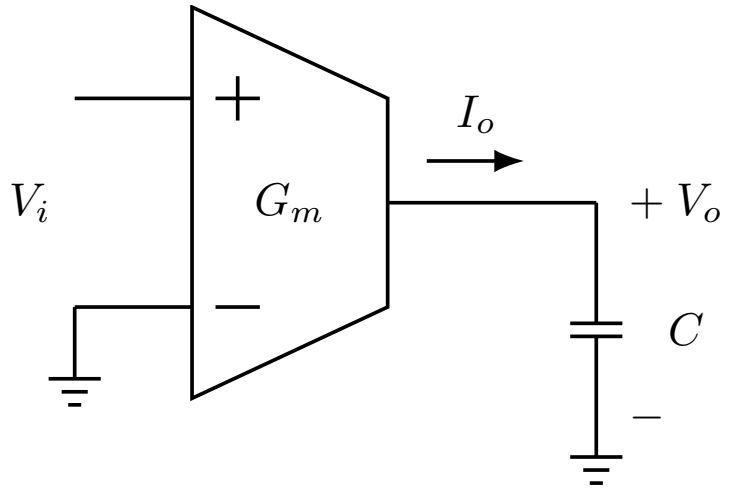


Figure 5: Single-ended Gm-C integrator: a transconductor driving a capacitor

$$V_o = \frac{I_o}{sC} = \frac{\omega_{ti}}{s} V_i$$

$$\omega_{ti} = \frac{G_m}{C}$$

19.3.1 Differential Gm-C

In a real IC we would almost always use differential circuit, as shown below. The transfer function is luckily the same.

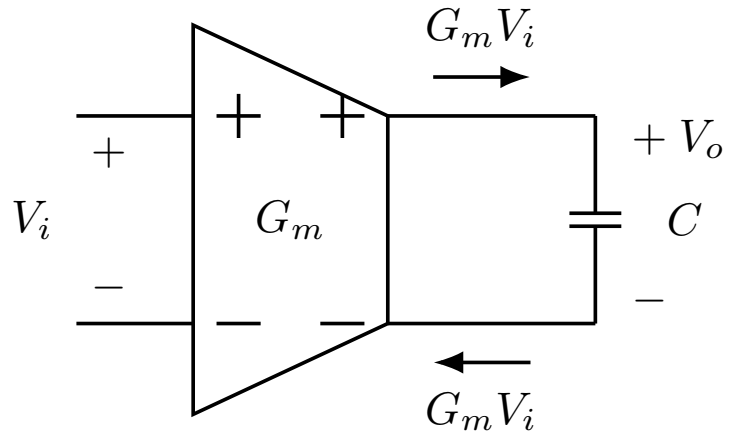


Figure 6: Fully differential Gm-C integrator

$$sCV_o = G_m Vi$$

$$H(s) = \frac{V_o}{V_i} = \frac{G_m}{sC}$$

Differential circuits are fantastic for multiple reasons, power supply rejection ratio, noise immunity, symmetric non-linearity, but the qualities I like the most is that the outputs can be flipped to implement negative, or positive gain.

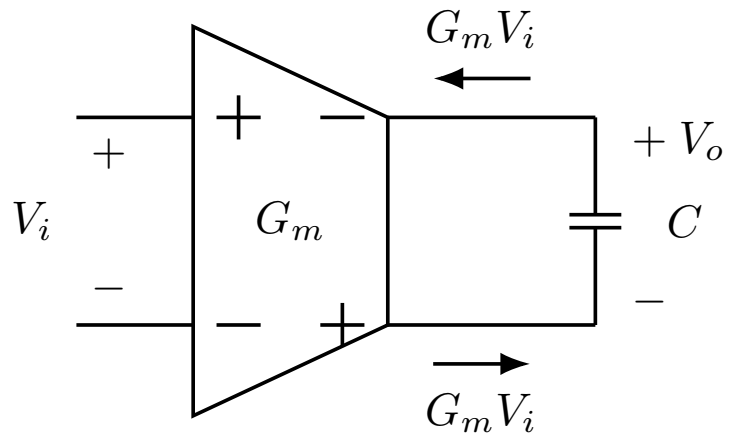
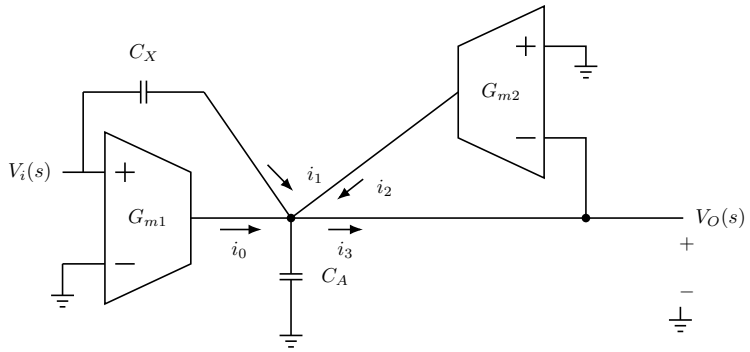


Figure 7: Differential Gm-C integrator with outputs flipped to implement negative gain

$$H(s) = \frac{V_o}{V_i} = -\frac{G_m}{sC}$$

The figure below shows a implementation of a first-order Gm-C filter that matches our signal flow graph.

I would encourage you to try and calculate the transfer function.



$i_0 =$

$i_1 =$

$i_2 =$

$i_3 =$

Figure 8: First-order Gm-C filter with transconductors G_{m1}, G_{m2} and capacitors C_X and C_A

Given the transfer function from the signal flow graph, we see that we can select C_x, C_a and G_m to get the desired k's and ω_0

$$H(s) = \frac{k_1 s + k_0}{s + w_o}$$

$$H(s) = \frac{s\frac{C_x}{C_a+C_x} + \frac{G_{m1}}{C_a+C_x}}{s + \frac{G_{m2}}{C_a+C_x}}$$

Below is a general purpose Gm-C bi-quadratic system.

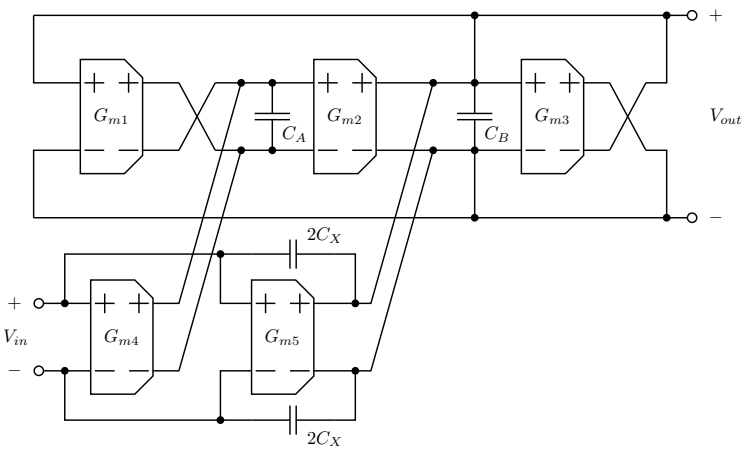


Figure 9: General purpose differential Gm-C biquad with five transconductors

$$H(s) = \frac{k_2 s^2 + k_1 s + k_0}{s^2 + \frac{\omega_0}{Q} s + \omega_o^2}$$

$$H(s) = \frac{s^2 \frac{C_X}{C_X+C_B} + s \frac{G_{m5}}{C_X+C_B} + \frac{G_{m2}G_{m4}}{C_A(C_X+C_B)}}{s^2 + s \frac{G_{m3}}{C_X+C_B} + \frac{G_{m1}G_{m2}}{C_A(C_X+C_B)}}$$

19.3.2 Finding a transconductor

Although you can start with the Gm-C cells in the book, I would actually choose to look at a few papers first.

The main reason is that any book is many years old. Ideas turn into papers, papers turn into books, and by the time you read the book, then there might be more optimal circuits for the technology you work in.

If I were to do a design in 22 nm FDSOI I would first see if someone has already done that, and take a look at the strategy they used. If I can't find any in 22 nm FDSOI, then I'd find a technology close to the same supply voltage.

Start with IEEEXplore

I could not find a 22 nm FDSOI Gm-C based circuit on the initial search. If I was to actually make a Gm-C circuit for industry I would probably spend a bit more time to see if any have done it, maybe expanding to other journals or conferences.

I know of Pieter Harpe, and his work is usually superb, so I would take a closer look at A 77.3-dB SNDR 62.5-kHz Bandwidth Continuous-Time Noise-Shaping SAR ADC With Duty-Cycled Gm-C Integrator [1]

And from Figure 10 a) we can see it's a similar Gm-C cell as chapter 12.5.4 in [2].

One of my Ph.d's used the transconductor below on his master thesis Design Considerations for a Low-Power Control-Bounded A/D Converter.

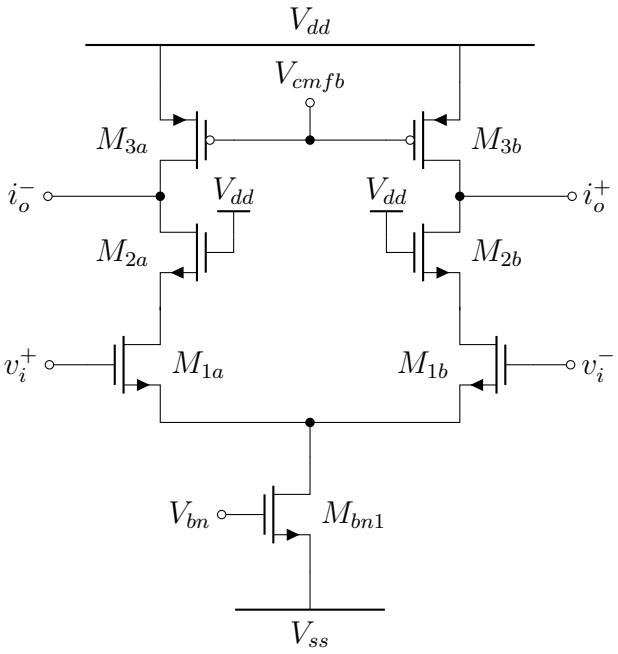


Figure 5.16: Transconductor schematic

Figure 10: Transconductor from the master thesis: a differential pair with common-gate cascodes to limit the Miller effect

19.4 Active-RC

The Active-RC filter should be well known at this point. However, what might be new is how the open loop gain A_0 and unity gain ω_{ta}

19.4.1 General purpose first order filter

Below is a general purpose first order filter and the transfer function. I've used the conductance $G = \frac{1}{R}$ instead of the resistance. The reason is that it sometimes makes the equations easier to work out.

If you're stuck on calculating a transfer function, then try and switch to conductance, and see if it resolves.

I often get my mind mixed up when calculating transfer functions. I don't know if it's only me, but if it's you also, then don't worry, it's not that often you have to work out transfer functions.

Once in a while, however, you will have a problem where you must calculate the transfer function. Sometimes it's because you'll need to understand where the poles/zeros are in a circuit, or you're trying to come up with a clever idea, or I decide to give this exact problem on the exam.

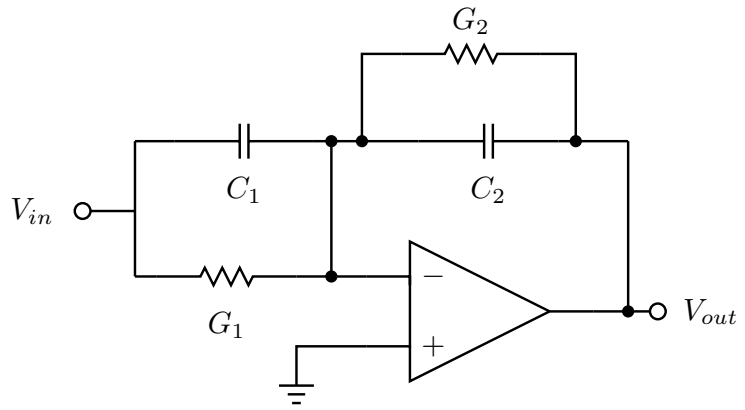


Figure 11: General purpose first-order active-RC filter with parallel RC input and feedback networks

$$H(s) = \frac{k_1 s + k_0}{s + w_o}$$

$$H(s) = \frac{-\frac{C_1}{C_2}s - \frac{G_1}{C_2}}{s + \frac{G_2}{C_2}}$$

Let's work through the calculation.

19.4.1.1 Step 1: Simplify

The conductance from V_{in} to virtual ground can be written as

$G_{in} = G_1 + sC_1$

The feedback conductance, between V_{out} and virtual ground I write as

$G_{fb} = G_2 + sC_2$

19.4.1.2 Step 2: Remember how an OTA works

An ideal OTA will force its inputs to be the same. As a result, the potential at OTA− input must be 0.

The input current must then be

$I_{in} = G_{in}V_{in}$

Here it's important to remember that there is no way for the input current to enter the OTA. The OTA is high impedance. The input current must escape through the output conductance G_{fb}.

What actually happens is that the OTA will change the output voltage V_{out} until the feedback current , I_{fb}, exactly matches I_{in}. That's the only way to maintain the virtual ground at 0 V. If the currents do not match, the voltage at virtual ground cannot continue to be 0 V, the voltage must change.

19.4.1.3 Step 3: Rant a bit

The previous paragraph should trigger your spidy sense. Words like "exactly matches" don't exist in the real world. As such, how closely the currents match must affect the transfer function. The open loop gain A_0 of the OTA matters. How fast the OTA can react to a change in voltage on the virtual ground, approximated by the unity-gain frequency ω_{ta} (the frequency where the gain of the OTA equals 1, or 0 dB), matters. The input impedance of the OTA, whether the gate leakage of the input differential pair due to quantum tunneling, or the capacitance of the input differential pair, matters. How much current the OTA can deliver (set by slew rate), matters.

Active-RC filter design is "How do I design my OTA so it's good enough for the filter". That's also why, for integrated circuits, you will not have a library of OTAs that you just plug in, and they work.

I would be very suspicious of working anywhere that had an OTA library I was supposed to use for integrated filter design. I'm not saying it's impossible that some company actually has an OTA library, but I think it's a bad strategy. First of all, if an OTA is generic enough to be used "everywhere", then the OTA is likely using too much power, consumes too much area, and is too complex. And the company runs the risk that the designer has not really checked that the OTA works properly in the filter because "Someone else designed the OTA, I just used in my design".

But, for now, to make our lives simpler, we assume the OTA is ideal. That makes the equations pretty, and we know what we should get if the OTA actually was ideal.

19.4.1.4 Step 4: Do the algebra

The current flowing from V_{out} to virtual ground is

$$I_{out} = G_{out} V_{out}$$

The sum of currents into the virtual ground must be zero

$$I_{in} + I_{out} = 0$$

Insert, and do the algebra

$$G_{in} V_{in} + G_{out} V_{out} = 0$$

$$\Rightarrow -G_{in} V_{in} = G_{out} V_{out}$$

$$\frac{V_{out}}{V_{in}} = -\frac{G_{in}}{G_{out}}$$

$$= -\frac{G_1 + sC_1}{G_2 + sC_2}$$

$$= \frac{-s\frac{C_1}{C_2} - \frac{G_1}{C_2}}{s + \frac{G_2}{C_2}}$$

19.4.2 General purpose biquad

A general bi-quadratic active-RC filter is shown below. These kind of general purpose filter sections are quite useful.

Imagine you wanted to make a filter, any filter. You'd decompose into first and second order sections, and then you'd try and match the transfer functions to the general equations.

The interactive biquad is worth a few minutes here: the low-pass, band-pass, high-pass, notch and all-pass responses all share one denominator, and switching between them moves the zeros while the poles stay exactly where they are.

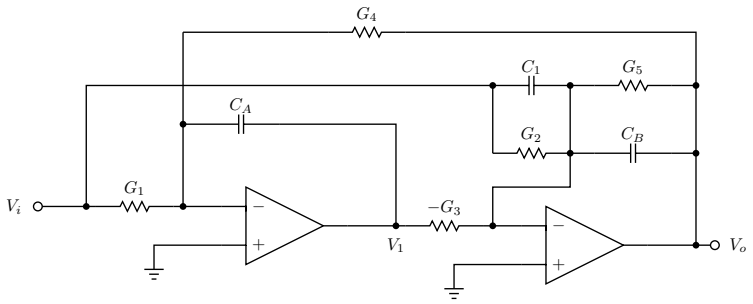


Figure 12: General purpose active-RC biquad built from two OTAs

$$H(s) = \frac{k_2 s^2 + k_1 s + k_0}{s^2 + \frac{\omega_0}{Q} s + \omega_o^2}$$

$$H(s) = \frac{\left[\frac{C_1}{C_B} s^2 + \frac{G_2}{C_B} s + (\frac{G_1 G_3}{C_A C_B})\right]}{\left[s^2 + \frac{G_5}{C_B} s + \frac{G_3 G_4}{C_A C_B}\right]}$$

19.5 The OTA is not ideal

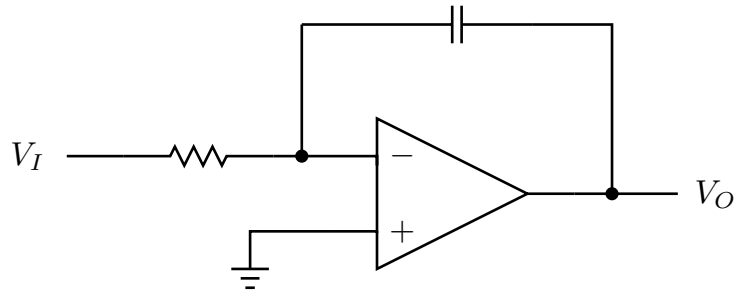


Figure 13: Active-RC integrator: OTA with resistor input and capacitor feedback

$$H(s) \approx \frac{A_0}{(1 + sA_o RC)(1 + \frac{s}{w_{ta}})}$$

where

$$A_0$$

is the gain of the amplifier, and

$$\omega_{ta}$$

is the unity-gain frequency.

At frequencies above $\frac{1}{A_0 RC}$ and below w_{ta} the circuit above is a good approximation of an ideal integrator.

See page 511 in [2] (chapter 5.8.1)

19.6 Example circuit

One place where both active-RC and Gm-C filters find a home are continuous time sigma-delta modulators. More on SD later, for now, just know that SD is a combination of high-gain, filtering, simple ADCs and simple DACs to make high resolution analog-to-digital converters.

One such an example is

A 56 mW Continuous-Time Quadrature Cascaded Sigma-Delta Modulator With 77 dB DR in a Near Zero-IF 20 MHz Band

Below we see the actual circuit. It may look complex, and it is.

Not just "complex" as in complicated circuit, it's also "complex" as in "complex numbers".

We can see there are two paths "i" and "q", for "in-phase" and "quadrature-phase". The fantastic thing about complex ADCs is that we can have a-symmetric frequency response around 0 Hz.

It will be tricky understanding circuits like this in the beginning, but know that it is possible, and it does get easier to understand.

With a complex ADC like this, the first thing to understand is the rough structure.

There are two paths, each path contains 2 ADCs connected in series (Multi-stage Noise-Shaping or MASH). Understanding everything at once does not make sense.

Start with "Vpi" and "Vmi", make it into a single path (set Rfb1 and Rfb2 to infinite), ignore what happens after R3 and DAC2i.

Now we have a continuous time sigma delta with two stages. First stage is a integrator (R1 and C1), and second stage is a filter (Cff1, R2 and C2). The amplified and filtered signal is sampled by the ADC1i and fed back to the input DAC1i.

It's possible to show that if the gain from $V(Vpi, Vmi)$ to ADC1i input is large, then $Y1i = V(Vpi, Vmi)$ at low frequencies.

Then put R_{fb1} and R_{fb2} back. Each one takes the *first integrator output* of one path and drives it into the *summing node* of the other - the same virtual ground the input resistor R_1 and the feedback DAC already inject into. That is the whole of the cross-coupling, and it is what turns two real modulators into one complex one.

The figure draws the first stage of the cascade only. What leaves through R_3 and $DAC2$ is the quantization error of this stage, handed to a second modulator that digitizes it; the digital outputs

are then recombined so the first stage's error cancels. That is what MASH means, and it is a separate story from the complex part.

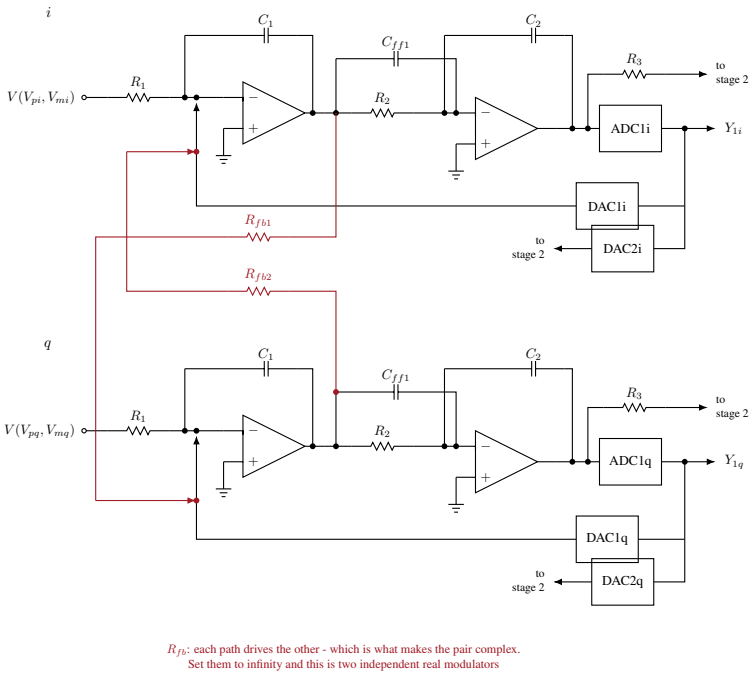


Figure 14: One stage of a continuous-time quadrature sigma-delta modulator, both paths drawn: two ordinary real modulators - R1/C1 integrating, Cff1/R2/C2 filtering, ADC1 sampling, DAC1 feeding back, R3 and DAC2 handing off to the second stage - joined by the red cross-coupling that makes the pair complex: each R_{fb} carries one path's first integrator output into the other path's summing node. Inspired by Breems et al. [3], which is fully differential, cascaded, and has a tuning resistor on everything

What does the cross-coupling actually buy? Figure 15 answers it by simulation. The same first-order loop is run twice: once real, so its coefficients are real and its noise notch must sit symmetrically about zero, and once complex, with the integrator pole rotated to $e^{j\omega_0}$ so the noise transfer function has its zero at $+f_0$ and nowhere else. The complex quantizer is just the two real quantizers, one per path - which is what the i and q ADCs are.

Read the two spectra over the *whole* sample rate, not half of it. The real loop is quiet at DC and equally noisy at $\pm f_0$: a real signal's spectrum is conjugate symmetric and cannot tell $+f$ from $-f$. The complex loop is quiet at $+f_0$ and noisy at $-f_0$, so a wanted channel just above the local oscillator gets the quiet side while its image gets the noise. That asymmetry is the entire reason a near zero-IF receiver bothers with quadrature.

Both simulations are driven by the same input, a complex tone just above f_0, and you can find it in both spectra as the tall spike. In the real loop it stands out in the middle of the noise, where the loop

has done nothing for it. In the complex loop it stands in the notch. The signal did not move; the quiet part of the spectrum did.

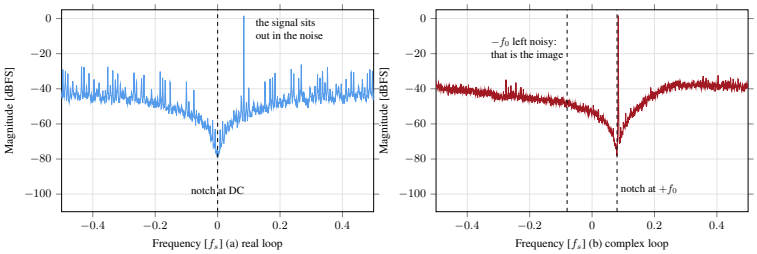


Figure 15: Simulated output spectra of a first-order sigma-delta loop over the full sample rate. (a) A real loop: the noise notch is symmetric about zero. (b) The same loop made complex: the notch moves to $+f_0$ alone, leaving $-f_0$ noisy - which is what lets a near zero-IF receiver keep the wanted side and throw the image away. The same input tone is present in both, and only in (b) does it land in the notch

19.7 My favorite OTA

Over the years I've developed a love for the current mirror OTA. A single stage, with load compensation, and an adaptable range of DC gains.

Sometimes simple current mirrors are sufficient, sometimes cascoded, or even active cascodes are necessary.

Below is the differential current mirror OTA.

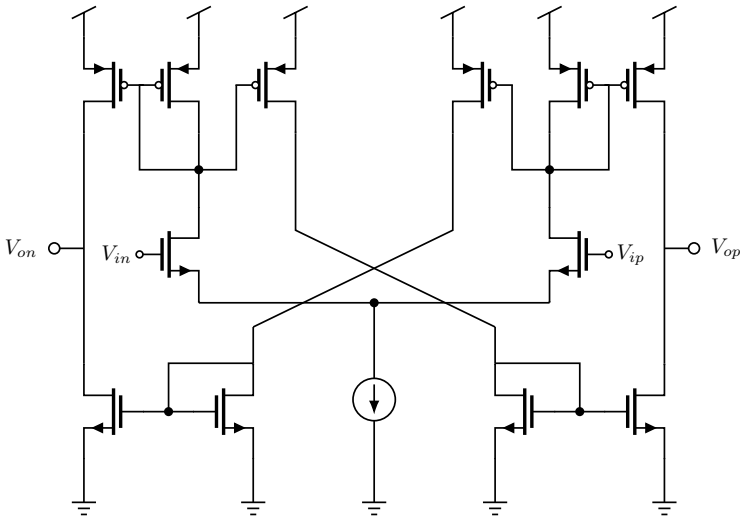


Figure 16: Differential current mirror OTA

In a differential OTA we need to control the output common mode. In order to control the common mode, we must sense the common mode.

Below is a circuit I often use to sense the common mode. Ideally the source followers would be native transistors, but those are not always available.

The reference for the common mode can be from a bandgap, or in the case below, VDD/2.

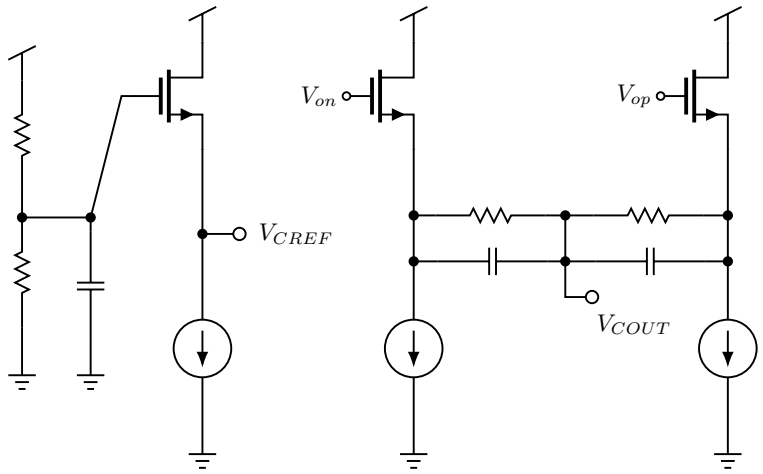


Figure 17: Common mode sense circuit with source followers and the resistor divider generating the reference V_{CREF}

Once we have both the sensed common mode, and the common mode reference, we can use another OTA to control the common mode.

The nice thing about the circuit below is that the common mode feedback loop has the same dominant pole as the differential loop.

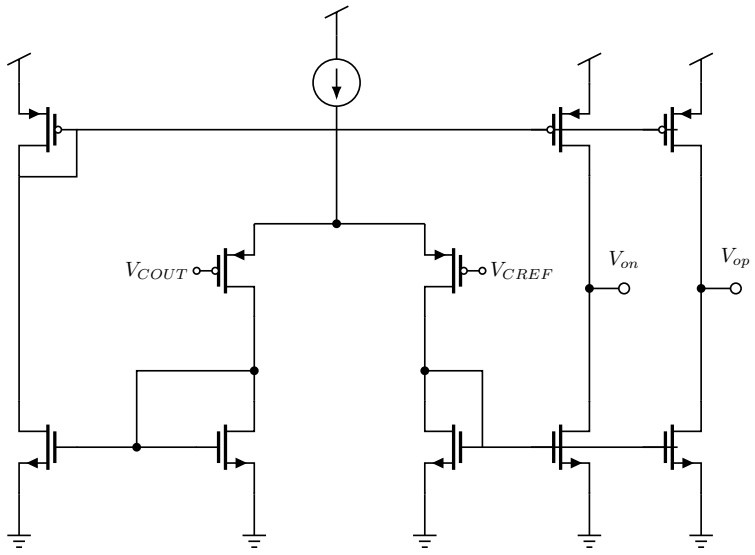


Figure 18: Common mode feedback OTA comparing the sensed V_{COUT} to the reference V_{CREF}

You can find the schematic for the OTA at

CNR_OTA_SKY130NM

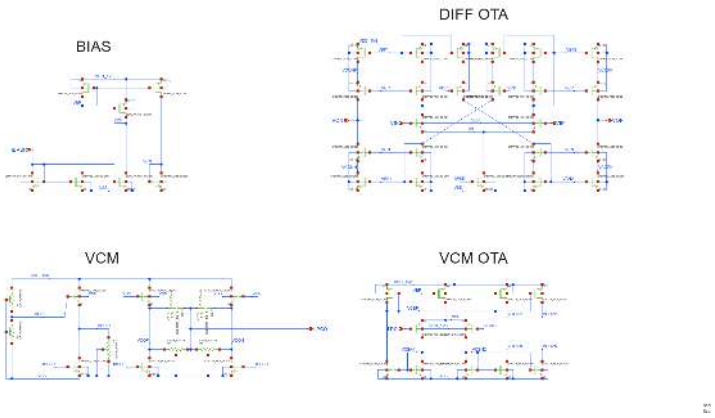


Figure 19: CNR_OTA schematic in SKY130: bias, differential OTA, common mode sense (VCM) and common mode feedback OTA

19.8 Summary

The one-page version of this chapter:

- The frontend's job is to hand the ADC a signal it can afford to convert: gain where it is cheap, filtering before sampling folds noise in
- First order sections come from one gm and one C; biquads stack them with feedback and give Q
- Gm-C is fast and open loop, active-RC is linear and closed loop - the OTA pays either way
- A real OTA's finite gain and bandwidth move the filter poles; design the OTA from the filter's error budget
- Fully differential filters double swing and cancel even harmonics, and carry the CMFB tax
- A complex (quadrature) filter is two real paths plus cross-coupling: the response stops being symmetric about zero, which a low-IF radio needs

19.9 Would you like to know more?

A 77.3-dB SNDR 62.5-kHz Bandwidth Continuous-Time Noise-Shaping SAR ADC With Duty-Cycled Gm-C Integrator [1]

Design Considerations for a Low-Power Control-Bounded A/D Converter

A 56 mW Continuous-Time Quadrature Cascaded Sigma-Delta Modulator With 77 dB DR in a Near Zero-IF 20 MHz Band [3]

Complex signal processing is not - complex [4]

[1] H. Li, Y. Shen, E. Cantatore, and P. Harpe, "A 77.3-dB SNDR 62.5-kHz bandwidth continuous-time noise-shaping SAR ADC with duty-cycled gm-c integrator," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 4, pp. 939–948, 2023, doi: 10.1109/JSSC.2022.3227678.

[2] T. C. Carusone, D. Johns, and K. Martin, *Analog integrated circuit design*. Wiley, 2011 [Online]. Available: https://books.google.no/books?id=1OIJZzLvVhcC

[3] L. J. Breems, R. Rutten, R. H. M. van Veldhoven, and G. van der Weide, "A 56 mW continuous-time quadrature cascaded ΣΔ modulator with 77 dB DR in a near zero-IF 20 MHz band," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2696–2705, 2007, doi: 10.1109/JSSC.2007.908765.

[4] K. Martin, "Complex signal processing is not - complex," in *ESSCIRC 2004 - 29th european solid-state circuits conference (IEEE cat. no.03EX705)*, 2003, pp. 3–14, doi: 10.1109/ESSCIRC.2003.1257061.

Digital to analog conversion 20

20.1 Resistor based DACs 333
20.2 DAC errors 335
20.3 DAC complexity . . 337
20.4 Binary scaled DACs 340
20.4.1 Binary coding 342
20.4.2 Thermometer encoding 344
20.5 Current mode DACs 345
20.6 Summary 346
20.7 Would you like to know more? 346

Keywords:

Processing of signals has shifted into the digital domain. But the real world is analog. In order to interact with the analog we need to convert the digital signals (discrete-value, discrete-time) back to analog signals (continuous value, continuous time).

The SI base units define the fundamental analog quantities as second, meter, kilogram, ampere, kelvin, mole and candela (see the refresher). Assume that electronic circuits interact with the real world in terms of second and ampere.

Related to Ampere we have the derived units of charge (Ampere Seconds), Volt (W/A), Ohm (V/A), or indeed Siemens (1/Ω).

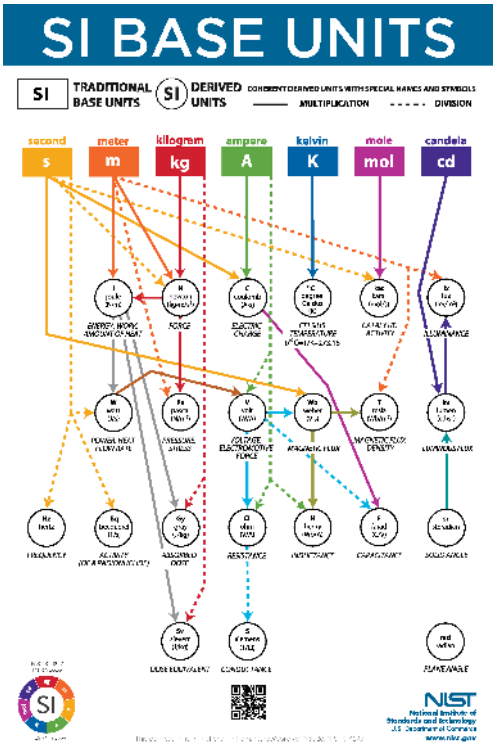


Figure 1: NIST poster of the SI base units and the derived units. Image: NIST, US Department of Commerce (US federal work)

As such, to create a digital to analog converter, we somehow have to create a circuit that has a function of

$$I_{out} = D_{in} \times I_{ref} \text{ [I]}$$

$$t_{out} = D_{in} \times t_{ref} \text{ [s]}$$

$$Q_{out} = D_{in} \times Q_{ref} \text{ [C]}$$

$$V_{out} = D_{in} \times V_{ref} \text{ [V]}$$

$$R_{out} = D_{in} \times R_{ref} \text{ [}\Omega\text{]}$$

The digital value is dimensionless, as such, there must be a reference value

Digital to analog conversion can be indirect through the relations between voltage, resistance, current, time, inductance and capacitance.

$$V = RI$$

$$Q = CV$$

$$dt = \frac{CdV}{I}$$

$$dt = \frac{LdI}{V}$$

20.1 Resistor based DACs

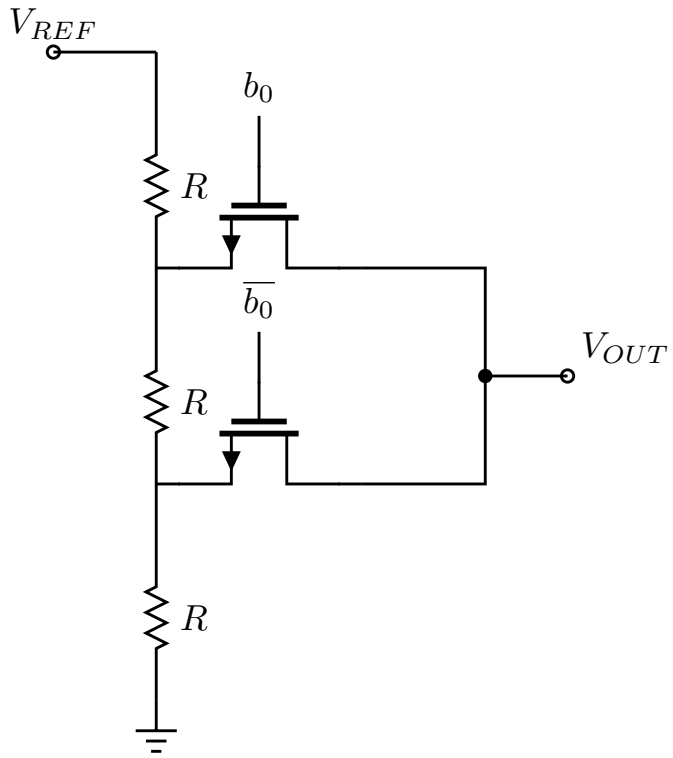


Figure 2: 1-bit DAC: three series resistors with transistor switches selecting the output tap

$$I_{ref} = \frac{V_{ref}}{3R}$$

$$V_{out} = D_{in} R I_{ref} = \frac{D_{in} R V_{ref}}{3R} \text{ [V]}$$

$$V_{out} = \frac{b_0 2R V_{ref}}{3R} + \frac{\overline{b_0} R V_{ref}}{3R}$$

$$V_{out} = \begin{cases} \frac{2}{3} V_{ref}, & b_0 = 1 \\ \frac{1}{3} V_{ref}, & b_0 = 0 \end{cases}$$

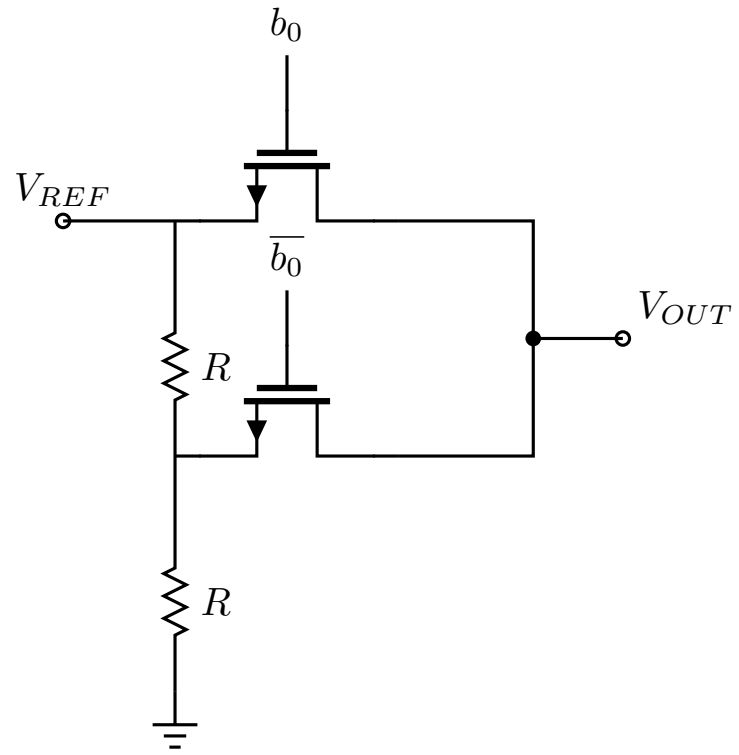


Figure 3: 1-bit DAC with two resistors, selecting either V_{REF} or the midpoint

$$I_{ref} = \frac{V_{ref}}{2R}$$

$$V_{out} = \frac{b_0 2RV_{ref}}{2R} + \frac{\overline{b_0}RV_{ref}}{2R}$$

$$V_{out} = \begin{cases} V_{ref}, & b_0 = 1 \\ \frac{1}{2}V_{ref}, & b_0 = 0 \end{cases}$$

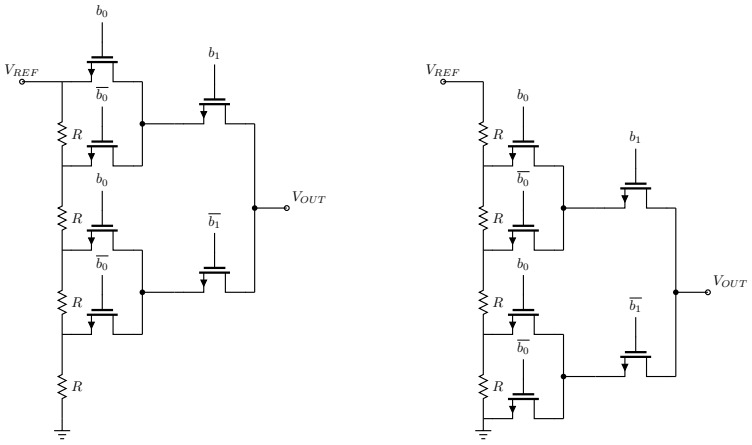


Figure 4: Two possible 2-bit resistor string DACs with switch trees

20.2 DAC errors

Digital to analog converters do not add quantization error. The quantization error is already in the digital word.

$$V_{out} = B + a_1 D_{in} + \left(a_2 D_{in}^2 + \cdots + a_n D_{in}^n\right)$$

Read that as three separate defects. B is offset, a constant added to every code. a_1 is gain, which stretches the transfer curve but keeps it straight. Everything in the bracket is non-linearity, and it is the only part that a calibration of gain and offset cannot remove.

DAC output will contain gain errors, offset errors, and non-linear components

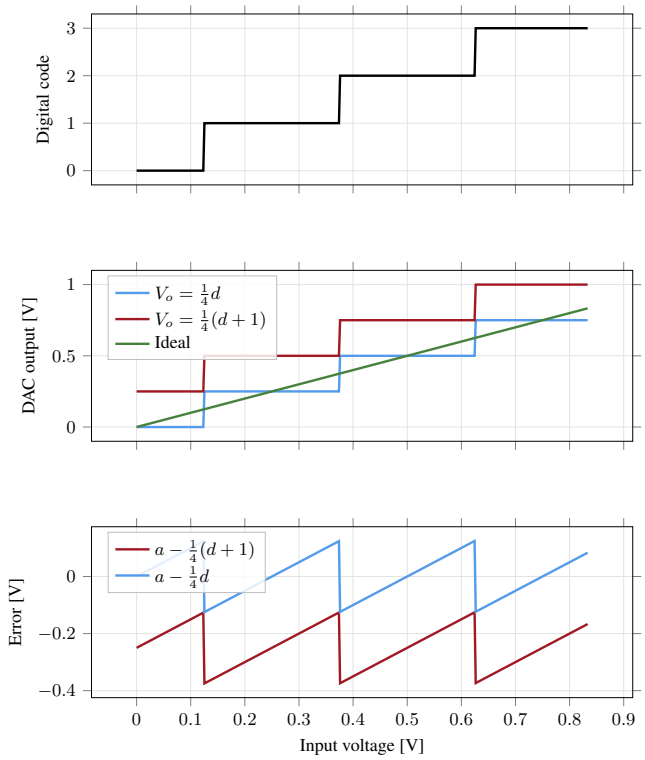


Figure 5: Quantization error of a 2-bit DAC. The digital code (top), two conventions for turning that code back into a voltage (middle), and the error each one makes (bottom)

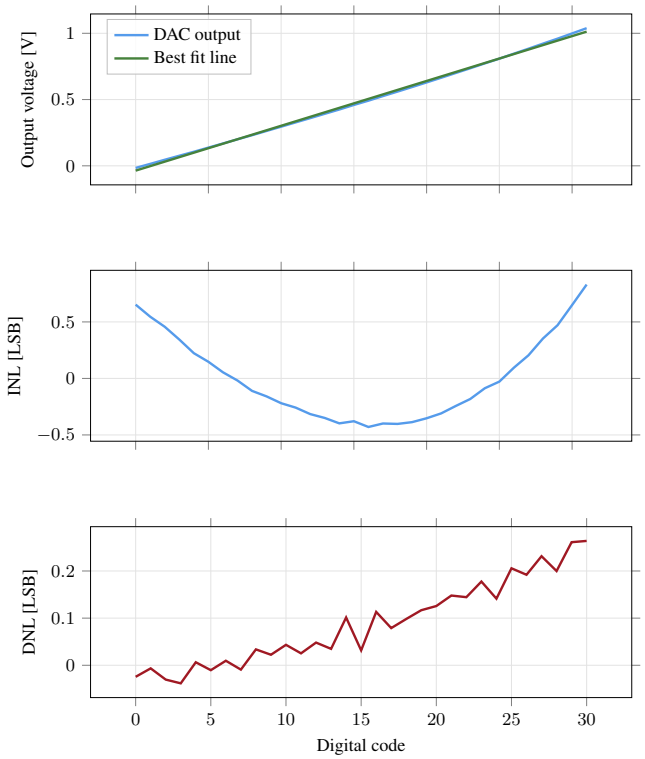


Figure 6: DAC output compared to the ideal straight line, with INL and DNL versus digital code

$$DNL[k] = \frac{V[k+1] - V[k]}{V_{LSB}} - 1$$

$$INL[k] = \frac{V[k] - V_{ideal}[k]}{V_{LSB}}$$

20.3 DAC complexity

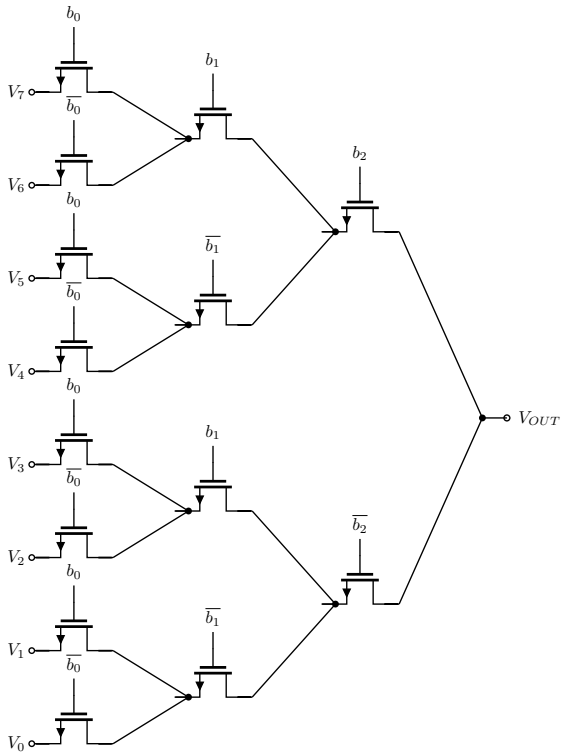


Figure 7: Binary switch tree for a resistor string DAC

The resistor string itself is the easy part - 2^N equal resistors give 2^N perfectly ordered taps, and the DAC is monotonic by construction. The cost hides in the selection: something must connect exactly one tap to the output. The obvious structure is a binary tree, where each bit steers a rank of switches, and the count below says why nobody stops there.

As number of resistors grow, the switches grow as

$$\sum_{n=1}^{N} 2^n = 2^{N+1} - 2$$

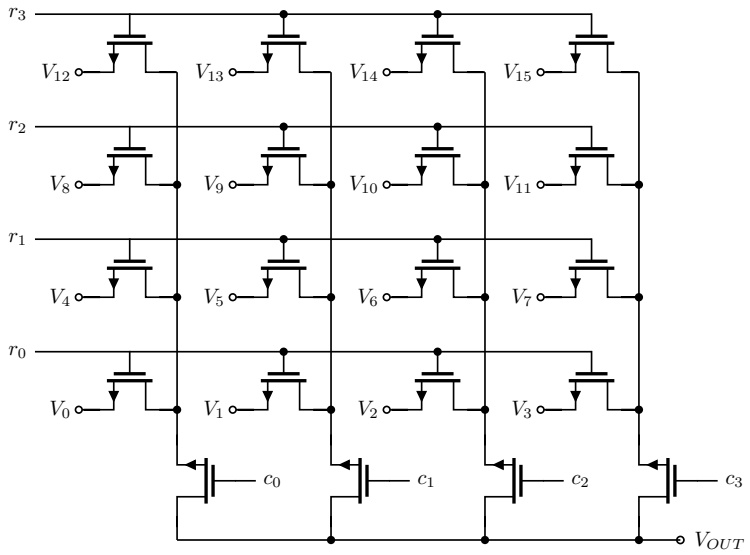


Figure 8: Row and column switch matrix for a resistor string DAC

The matrix halves the damage by decoding in two dimensions, the way a memory does: the row decoder picks a group of taps, the column switches pick one of them. The tap switches are still there - they must be, every tap has to be reachable - but the tree above them collapses into one switch per column.

Give every tap a switch onto a column line, and every column line a switch to the output. For 2^N taps in a square arrangement that is

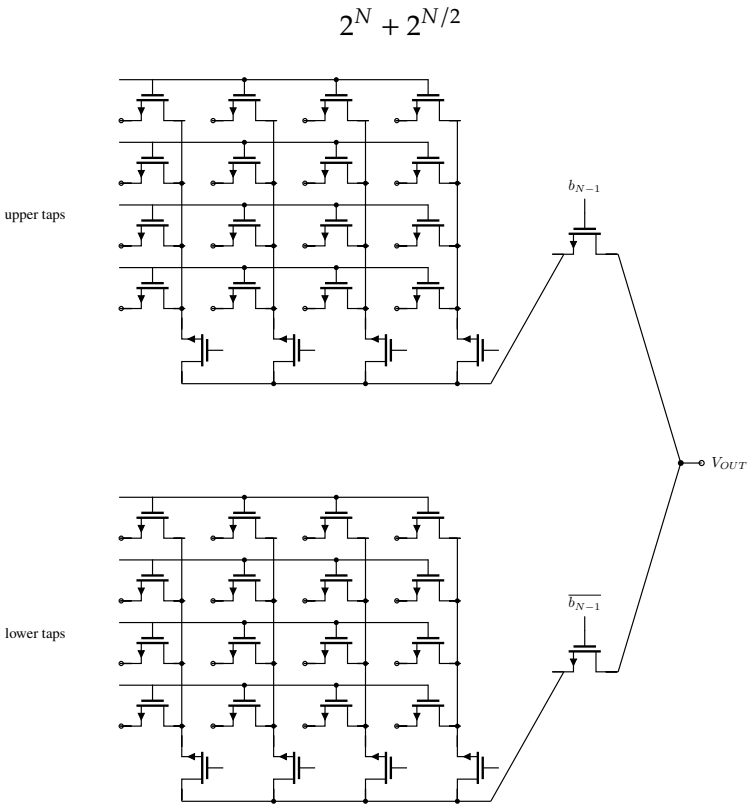


Figure 9: Segmented DAC switch arrangement combining switch matrices and a tree

Switches in a 10-bit digital to analog converter.

Tree	$2^{N+1} - 2$	= 2046
Matrix	$2^N + 2^{N/2}$	= 1056
Two strings, 6b + 4b	$2^M + 2^{N-M}$	= 80
Two strings, 5b + 5b	$2^M + 2^{N-M}$	= 64

Those three lines are not three versions of the same circuit, and the difference between the first two and the last is the point of this section.

The tree and the matrix both address one string of 2^N resistors, so both need at least one switch per tap. The tree adds switches at every internal node on top of that, which is why it is the worst of the three; the matrix adds only one per column, which is why it is roughly half the tree. Neither can go below 2^N, because every tap has to be reachable on its own.

It is worth being clear that combining them does not help. A tree of matrix blocks — a plausible reading of the figure above — still needs its 2^N tap switches and now pays for the tree as well: a 4-bit tree over sixteen 6-bit matrices comes to 1182, worse than the plain matrix at 1056. Every split is worse. There is nothing to win by decoding the same string more cleverly.

The saving comes from not having one string at all. Put a coarse string of 2^M resistors in series and let a fine string of 2^{N-M} resistors interpolate between two adjacent coarse taps, and each selector only has to reach its own string: $2^M + 2^{N-M}$ switches rather than 2^N. For ten bits split six and four that is 80, and the best split is five and five at 64 — against 1056 for the matrix. Trading one big decoder for two small ones is worth a factor of sixteen here, and the reason is simply that $2^M + 2^{N-M}$ grows far more slowly than 2^N.

Nothing is free: the fine string loads the coarse one and disturbs the very voltage it is interpolating, a real two-string DAC needs a second switch per coarse tap to bracket the segment, and the matching now has to hold between two strings rather than within one. But the switch count is no longer what stops you.

Large number of bits, will be large number of resistors and switches.

20.4 Binary scaled DACs

A string DAC pays 2^N resistors for N bits. The R-2R ladder pays $2N$: each section divides the remaining voltage by two, so the branch currents come out binary weighted with only two resistor values. The next four figures build the ladder one property at a time - the termination, the input resistance that stays 2R at every section, and the halving branch currents that make it a DAC.

$$R_{in} = 2R \parallel 2R = R$$

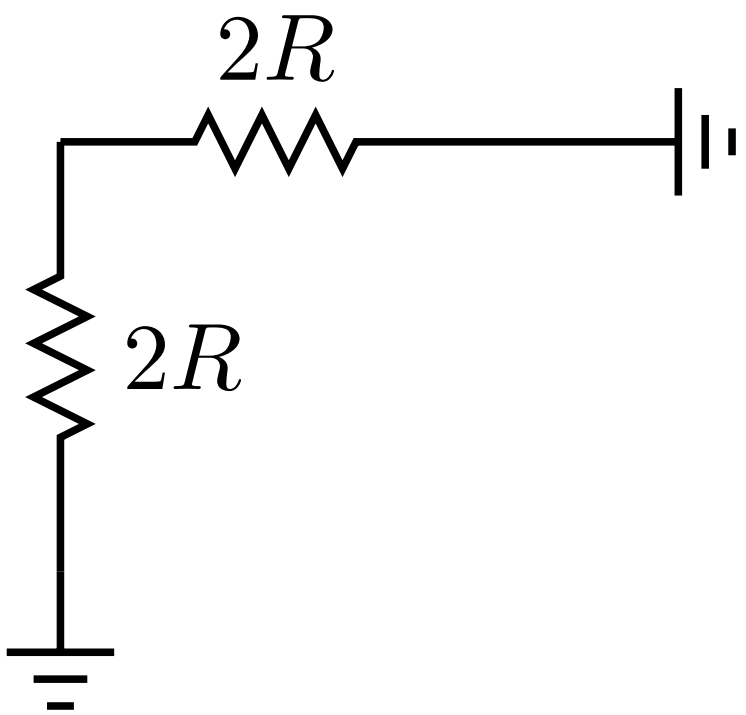


Figure 10: R-2R ladder termination: two 2R resistors in parallel equal R

$$R_{in} = R + R = 2R$$

$$I_0 = \frac{V_0}{2R} = \frac{V_1}{4R}$$

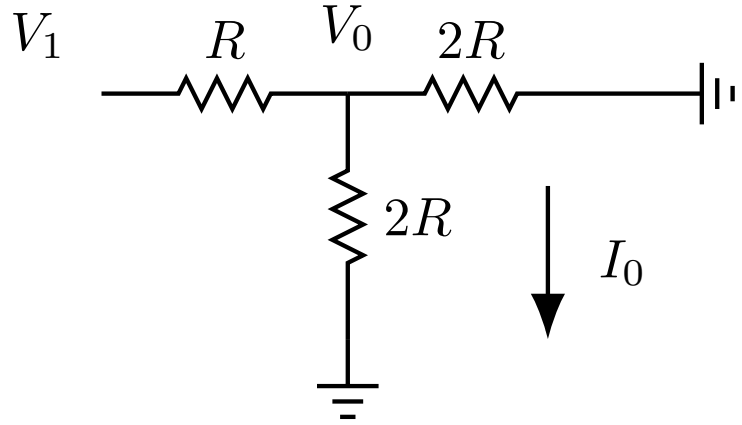


Figure 11: One R-2R ladder section: the series R makes the input resistance 2R

$$R_{in} = 2R \parallel 2R = R$$

$$I_0 = \frac{V_0}{2R} = \frac{V_1}{4R}$$

$$I_1 = \frac{V_1}{2R}$$

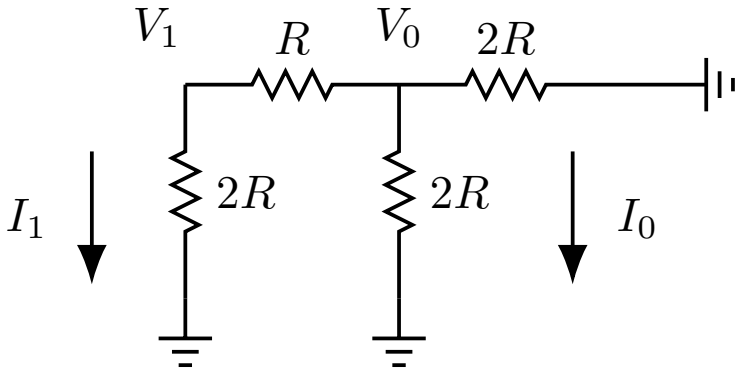


Figure 12: R-2R ladder section with the binary weighted branch currents I_1 and I_0

$$I_{RF} = I_1 b_1 + I_0 b_0 = \frac{V_{REF}}{2R} b_1 + \frac{V_{REF}}{4R} b_0$$

$$V_O = \left(\frac{V_{REF}}{2R} b_1 + \frac{V_{REF}}{4R} b_0 \right) R_{F0}$$

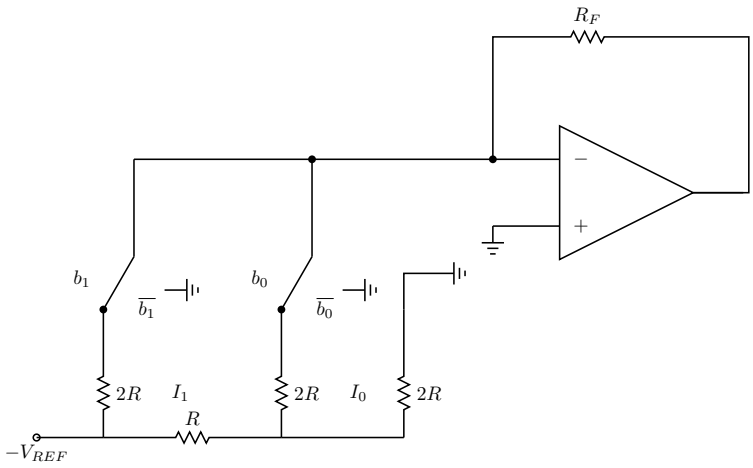


Figure 13: 2-bit R-2R DAC with switched branch currents summed by a transimpedance amplifier

The switches steer each branch current either into the virtual ground of the amplifier or to real ground, so the ladder's currents never change - only their destination does. That is what makes the R-2R fast for its size. What it gives up is the string's built-in monotonicity: at the major transition the MSB branch must match the sum of all the others to within an LSB, and that is now a matching requirement on the resistors rather than a property of the structure.

20.4.1 Binary coding

For 4 states (2-bit) there are 12 possible transitions

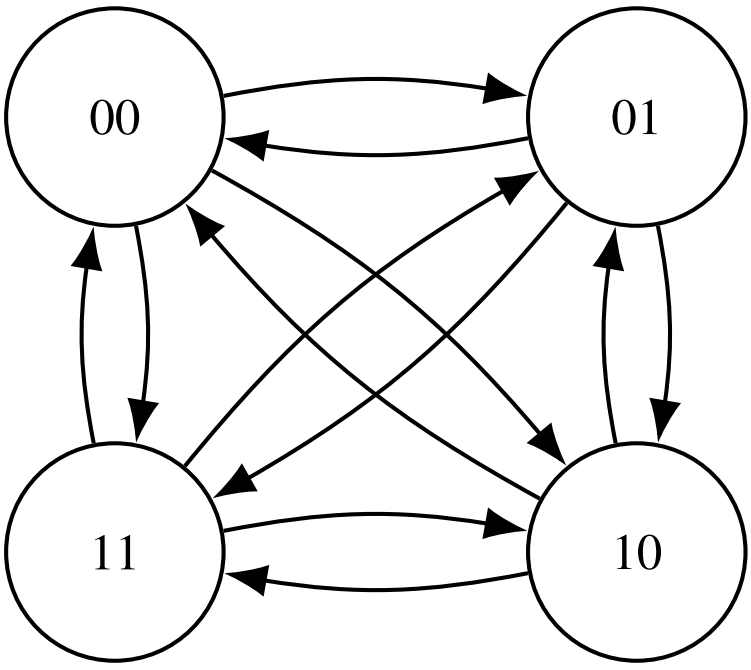


Figure 14: The 12 possible transitions between the four 2-bit binary states

Assume MSB first (left)

$$1 \rightarrow 3 \rightarrow 2$$

Assume LSB first (right)

$$1 \rightarrow 0 \rightarrow 2$$

Both cause a non-monotonic glitch during transition.

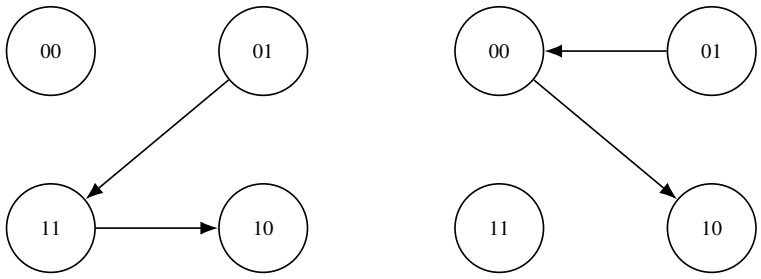


Figure 15: Binary code transitions with MSB first (left) and LSB first (right), both non-monotonic

The switches never move at exactly the same time. Between the old code and the new one the DAC output visits whatever code the half-switched bits happen to spell, and around the major transition - 0111 to 1000 - that intermediate code can be far away. The result is a glitch whose energy grows with the weight of the bits involved, and no amount of matching removes it: it is a property of the code, not of the elements.

20.4.2 Thermometer encoding

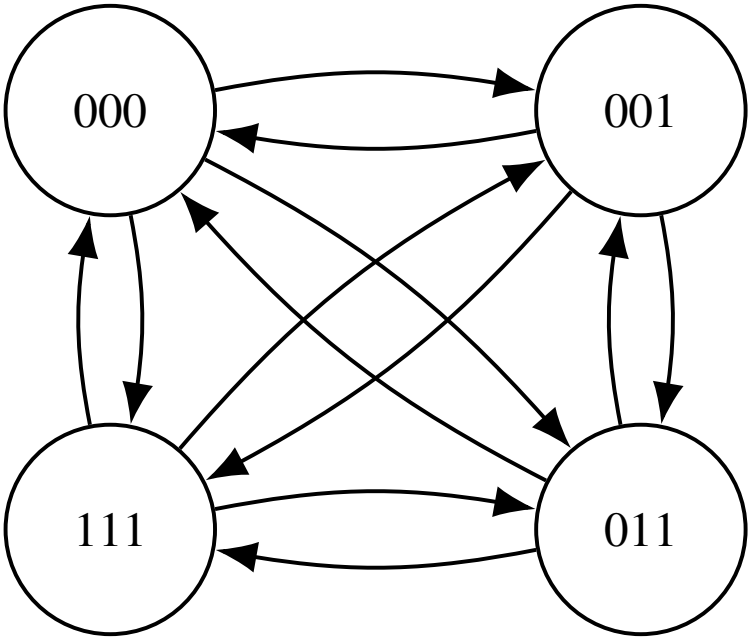


Figure 16: Transitions between the thermometer encoded states

The sequence of MSB to LSB does not matter.

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 3$$

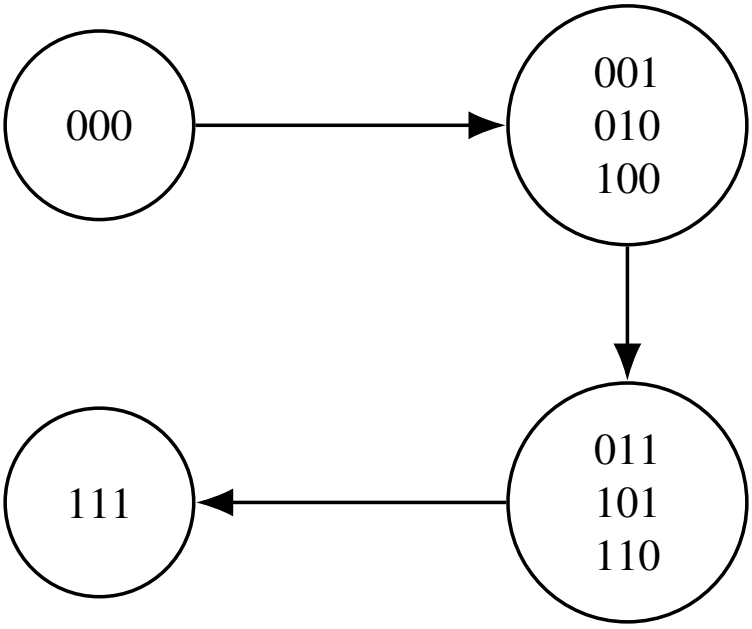


Figure 17: Thermometer code transitions are monotonic regardless of bit order

Thermometer coding removes the glitch by construction: one more LSB always means one more element turned on, so the output can only move one step, whatever order the switches settle in.

Monotonicity comes for free for the same reason. The price is $2^N - 1$ elements and the decoder that drives them, which is why real converters segment - thermometer for the MSBs where the glitch would be worst, binary for the LSBs where it cannot hurt.

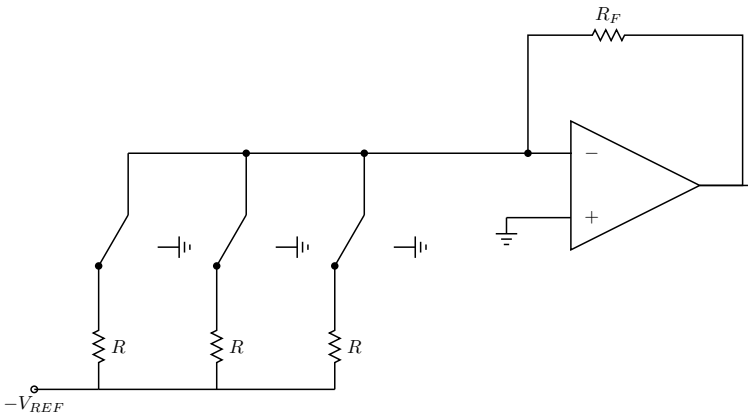


Figure 18: Thermometer coded resistor DAC with equal resistors summed by a transimpedance amplifier

20.5 Current mode DACs

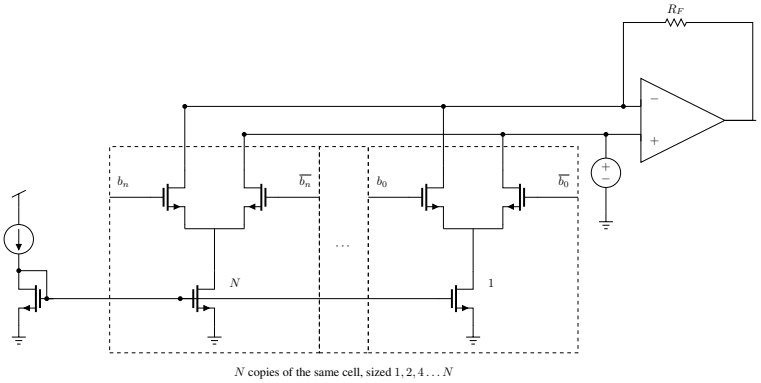


Figure 19: Current mode DAC: binary sized differential current cells switched into a transimpedance output stage

At high sample rates the resistor structures run out of settling time, and the current steering DAC takes over: every cell is a current source that is always on, and the data only chooses which side of a differential pair the current leaves through. Nothing charges or discharges except the switch nodes, so this is the architecture behind every GS/s transmitter DAC.

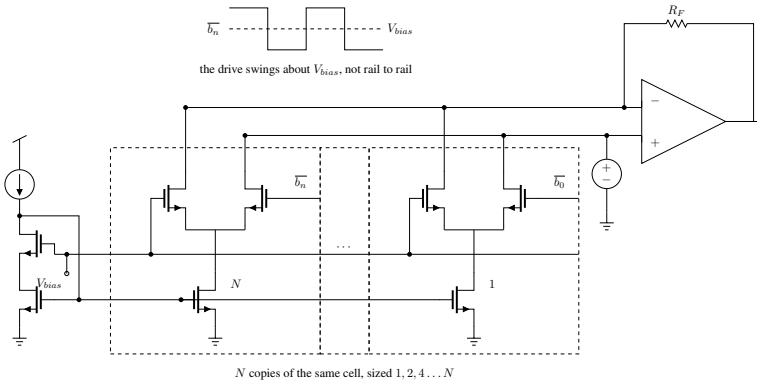


Figure 20: Current mode DAC where the switch drive swings around V_{bias} instead of rail to rail

Driving the steering pair rail to rail briefly turns both switches off and slams the source node; the cell's current has to go somewhere, and it goes into the output as a spike. Limiting the switch drive to a small swing around V_{bias} keeps the pair in its active region through the crossover, keeps the current source in saturation, and is the difference between a DAC that meets its SFDR and one that only meets its resolution.

20.6 Summary

The one-page version of this chapter:

- A DAC turns a code into charge, current or voltage by summing weighted unit elements
- Binary weighting is compact but must switch half the array at the major carry; thermometer coding is monotonic and glitch-free but costs 2^N elements and decoding
- Segmentation spends thermometer coding on the MSBs where it matters and binary on the LSBs where it is cheap
- Static accuracy is INL/DNL set by element matching (Pelgrom: area buys bits); dynamic accuracy is glitch energy and SFDR
- The references and the switch drivers are part of the DAC: their noise and timing skew show up in the output spectrum

20.7 Would you like to know more?

A 28-nm 75-fsrms Analog Fractional-N Sampling PLL With a Highly Linear DTC Incorporating Background DTC Gain Calibration and Reference Clock Duty Cycle Correction [1]

A 10-bit Charge-Redistribution ADC Consuming 1.9 uW at 1 MS/s [2]

A 6.3 uW 20 bit Incremental Zoom-ADC with 6 ppm INL and 1 uV Offset [3]

A 12-Bit 1.25-GS/s DAC in 90 nm CMOS With >70 dB SFDR up to 500 MHz [4]

[1] W. Wu *et al.*, "A 28-nm 75-fsrms analog fractional- N sampling PLL with a highly linear DTC incorporating background DTC gain calibration and reference clock duty cycle correction," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 5, pp. 1254–1265, 2019, doi: 10.1109/JSSC.2019.2899726.

[2] M. van Elzakker, E. van Tuijl, P. Geraedts, D. Schinkel, E. A. M. Klumperink, and B. Nauta, "A 10-bit charge-redistribution ADC consuming 1.9 μw at 1 MS/s," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 5, pp. 1007–1015, 2010, doi: 10.1109/JSSC.2010.2043893.

[3] Y. Chae, K. Souri, and K. A. A. Makinwa, "A 6.3 μw 20 bit incremental zoom-ADC with 6 ppm INL and 1 μv offset," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 12, pp. 3019–3027, 2013, doi: 10.1109/JSSC.2013.2278737.

[4] W.-H. Tseng, C.-W. Fan, and J.-T. Wu, "A 12-bit 1.25-GS/s DAC in 90 nm CMOS with >70 dB SFDR up to 500 MHz," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 12, pp. 2845–2856, 2011, doi: 10.1109/JSSC.2011.2164302.

Switched-Capacitor Circuits 21

Keywords: SC DAC, SC FUND, DT, Alias, Subsample, Z Domain, FIR, IIR, SC MDAC, SC INT, Switch, Non-Overlap, VBE SC, Nyquist

21.1 Active-RC 349
21.2 Gm-C 351
21.3 Switched capacitor 351
21.3.1 An example SC circuit 355
21.4 Discrete-Time Signals 357
21.4.1 The mathematics . . 358
21.4.2 Python discrete time example 359
21.4.3 Aliasing, bandwidth and sample rate theory 361
21.4.4 Z-transform 362
21.4.5 Pole-Zero plots . . . 363
21.4.6 Z-domain 364
21.4.7 First order filter . . . 365
21.4.8 Second order filter . 366
21.4.9 Finite-impulse response(FIR) 368
21.5 Switched-Capacitor 369
21.5.1 Switched capacitor gain circuit 371
21.5.2 Switched capacitor integrator 372
21.5.3 Noise 374
21.5.4 Sub-circuits for SC-circuits 376
21.5.5 Example 381
21.6 Summary 382
21.7 Would you like to know more? 382

21.1 Active-RC

A general purpose Active-RC bi-quadratic (two-quadratic equations) filter is shown below

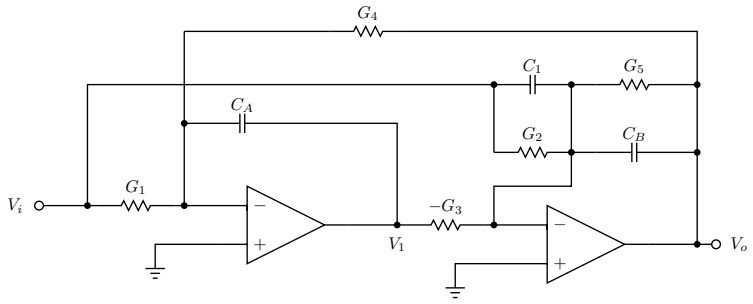


Figure 1: General purpose Active-RC biquad

If you want to spend a bit of time, then try and calculate the transfer function below.

$$H(s) = \frac{\left[\frac{C_1}{C_B}s^2 + \frac{G_2}{C_B}s + (\frac{G_1 G_3}{C_A C_B})\right]}{\left[s^2 + \frac{G_5}{C_B}s + \frac{G_3 G_4}{C_A C_B}\right]}$$

Active resistor capacitor filters are made with OTAs (high output impedance) or OPAMP (low output impedance). Active amplifiers will consume current, and in Active-RC the amplifiers are always on, so there is no opportunity to reduce the current consumption by duty-cycling (turning on and off).

Both resistors and capacitors vary on an integrated circuit, and the 3-sigma variation can easily be 20 %.

The pole or zero frequency of an Active-RC filter is proportional to the inverse of the product between R and C

$$\omega_{p|z} \propto \frac{G}{C} = \frac{1}{RC}$$

As a result, the total variation of the pole or zero frequency can have a 3-sigma value of

$$\sigma_{RC} = \sqrt{\sigma_R^2 + \sigma_C^2} = \sqrt{0.2^2 + 0.2^2} = 0.28 = 28\ \%$$

On an IC we sometimes need to calibrate the R or C in production to get an accurate RC time constant.

We cannot physically change an IC, every single one of the 100 million copies of an IC is from the same Mask set. That's why ICs are cheap. To make the Mask set is incredibly expensive (think 5 million dollars), but a copy made from the Mask set can cost one dollar or less. To calibrate we need additional circuits.

Imagine we need a resistor of 1 kOhm. We could create that by parallel connection of larger resistors, or series connection of smaller resistors. Since we know the maximum variation is 0.02, then we need to be able to calibrate away +- 20 Ohms. We could have a 980 Ohm resistor, and then add ten 4 Ohm resistors in series that we can short with a transistor switch.

But is a resolution of 4 Ohms accurate enough? What if we need a precision of 0.1%? Then we would need to tune the resistor within +-1 Ohm, so we might need 80 0.5 Ohm resistors.

But how large is the on-resistance of the transistor switch? Would that also affect our precision?

But is the calibration step linear with addition of the transistors? If we have a non-linear calibration step, then we cannot use gradient descent calibration algorithms, nor can we use binary search.

Analog designers need to deal with an almost infinite series of "But".

The experienced designer will know when to stop, when is the "But what if" not a problem anymore.

The most common error in analog integrated circuit design is a "I did not imagine that my circuit could fail in this manner" type of problem. Or, not following the line of "But" 's far enough.

But if we follow all the "But" 's we will never tapeout!

Active-RC filters are great for linearity, but if we need accurate time constant, there are better alternatives.

21.2 Gm-C

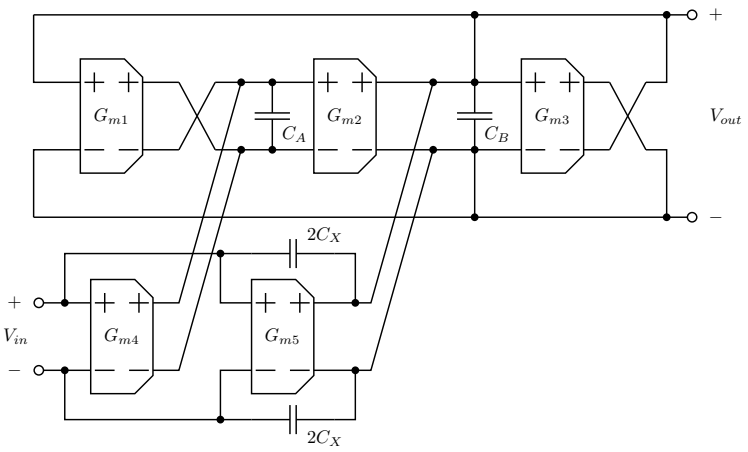


Figure 2: General purpose Gm-C biquad

$$H(s) = \frac{\left[s^2 \frac{C_X}{C_X+C_B} + s\frac{G_{m5}}{C_X+C_B} + \frac{G_{m2}G_{m4}}{C_A(C_X+C_B)}\right]}{\left[s^2 + s\frac{G_{m3}}{C_X+C_B} + \frac{G_{m1}G_{m2}}{C_A(C_X+C_B)}\right]}$$

The pole and zero frequency of a Gm-C filter is

$$\omega_{p|z} \propto \frac{G_m}{C}$$

The transconductance accuracy depends on the circuit, and the bias circuit, so we can't give a general, applies for all circuits, sigma number. Capacitors do have 3-sigma 20 % variation, usually.

Same as Active-RC, Gm-C need calibration to get accurate pole or zero frequency.

21.3 Switched capacitor

The first time you encounter Switched Capacitor (SC) circuits, they do require some brain training. So let's start simple.

Consider the circuit below. Assume that the two transistors are ideal (no-charge injection, no resistance).

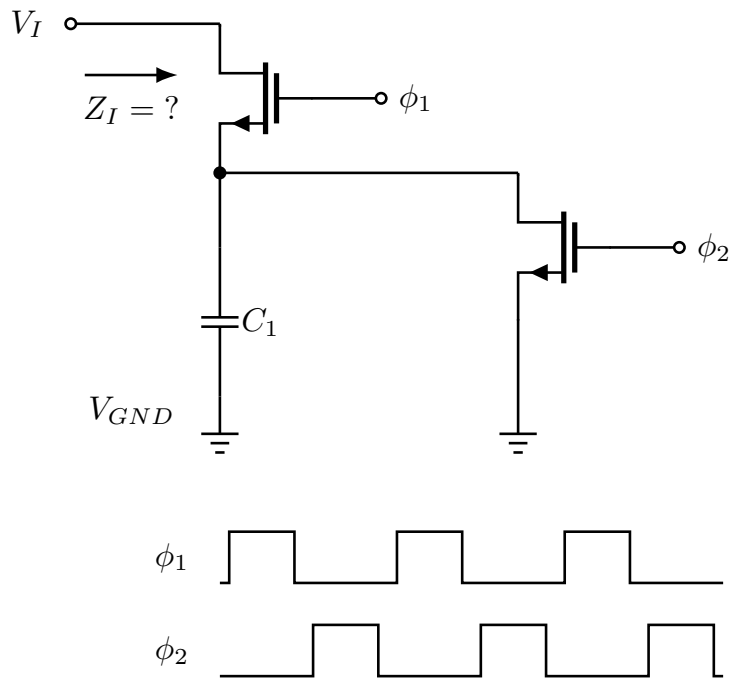


Figure 3: Switched capacitor with two transistor switches and the non-overlapping clock phases ϕ_1 and ϕ_2

For SC circuits, we need to consider the charge on the capacitors, and how they change with time.

The charge on the capacitor at the end * of phase 2 is

$$Q_{\phi 2\$} = C_1 V_{GND} = 0$$

while at the end of phase 1

$$Q_{\phi 1\$} = C_1 V_I$$

The impedance, from Ohm's law is

$$Z_I = (V_I - V_{GND})/I_I$$

And from SI units units we can see current is charge per unit time. Once per clock period the capacitor is charged and then discharged, so what flows in from the input is the *difference* between the charge held at the end of each phase, delivered f_ϕ times a second:

$$I_I = \frac{\Delta Q}{\Delta t} = \left(Q_{\phi 1\$} - Q_{\phi 2\$}\right) f_\phi$$

* I use the $ to mark the end of the period. It comes from Regular Expressions.

Charge cannot disappear, charge is conserved. As such, the charge going out from the input must be equal to the difference of charge at the end of phase 1 and phase 2.

$$Z_I = \frac{V_I - V_{GND}}{\left(Q_{\phi 1\$} - Q_{\phi 2\$}\right) f_\phi}$$

Inserting for the charges, we can see that the impedance is

$$Z_I = \frac{V_I}{\left(C_1 V_I - 0\right) f_\phi} = \frac{1}{C_1 f_\phi}$$

A common confusion with SC circuits is to confuse the impedance of a capacitor $Z = 1/sC$ with the impedance of a SC circuit $Z = 1/fC$. The impedance of a capacitor is complex (varies with frequency and time), while the SC circuit impedance is real (a resistance).

The main difference between the two is that the impedance of a capacitor is continuous in time, while the SC circuit is a discrete time circuit, and has a discrete time impedance.

The circuit below is drawn slightly differently, but the same equation applies.

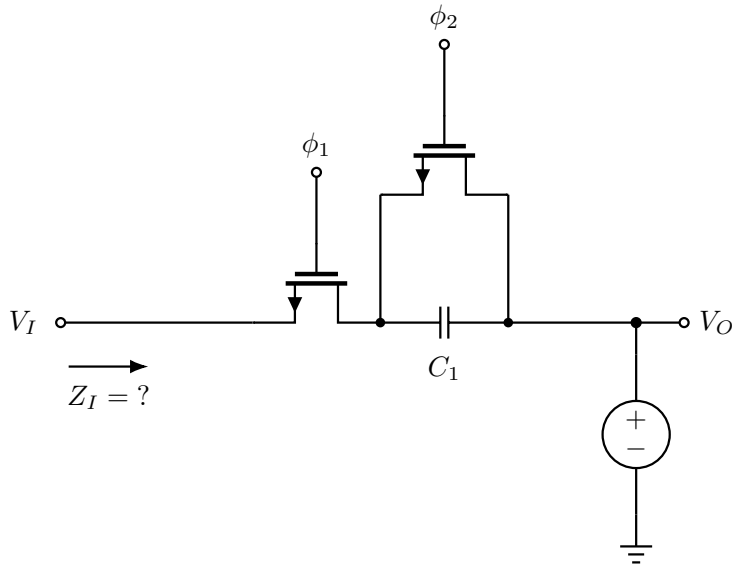


Figure 4: The same switched capacitor rotated, now between the input and an output voltage source

If we compute the impedance.

$$Z_I = \frac{V_I - V_O}{\left(Q_{\phi 1\$} - Q_{\phi 2\$}\right) f_\phi}$$

$$Q_{\phi 1\$} = C_1(V_I - V_O)$$

$$Q_{\phi 2\$} = 0$$

$$Z_I = \frac{V_I - V_O}{C_1 (V_I - V_O) f_\phi} = \frac{1}{C_1 f_\phi}$$

Which should not be surprising, as all I've done is to rotate the circuit and call $V_{GND} = V_0$.

Let's try the circuit below.

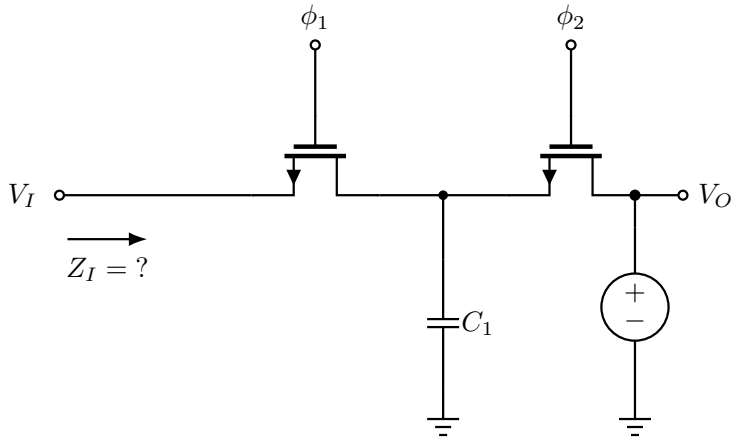


Figure 5: Switched capacitor with series switches, charging C_1 to V_I in ϕ_1 and V_O in ϕ_2

$$Z_I = \frac{V_I - V_O}{\left(Q_{\phi 1\$} - Q_{\phi 2\$}\right) f_\phi}$$

$$Q_{\phi 1\$} = C_1 V_I$$

$$Q_{\phi 2\$} = C_1 V_O$$

Inserted into the impedance we get the same result.

$$Z_I = \frac{V_I - V_O}{(C_1 V_I - C_1 V_O) f_\phi} = \frac{1}{C_1 f_\phi}$$

The first time I saw the circuit above it was not obvious to me that the impedance still was $Z = 1/Cf$. It's one of the cases where mathematics is a useful tool. I could follow a set of rules (charge conservation), and as long as I did the mathematics right, then from the equations, I could see how it worked.

21.3.1 An example SC circuit

An example use of an SC circuit is

A pipelined 5-Msample/s 9-bit analog-to-digital converter [1]

Shown in the figure below. You should think of the switched capacitor circuit as similar to an amplifier with constant gain. We can use two resistors and an opamp to create a gain. Imagine we create a circuit without the switches, and with a resistor of R from input to virtual ground, and $4R$ in the feedback. Our Active-R would have a gain of $A = 4$.

The switches disconnect the OTA and capacitors for half the time, but for the other half, at least for the latter parts of ϕ_2 the gain is four.

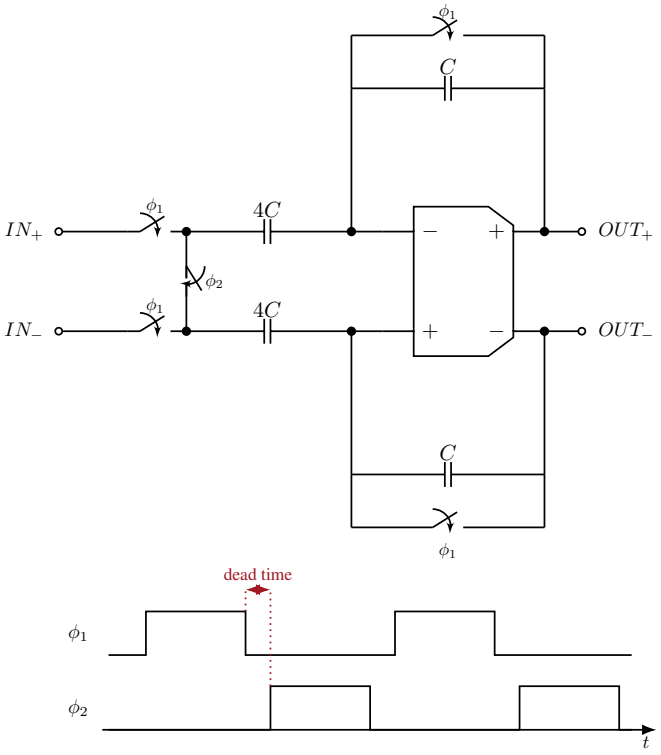


Figure 6: A fully differential switched-capacitor sample-and-hold with a gain of four, and the two-phase clock that runs it. On ϕ_2 the two input plates are shorted to each other rather than to ground, which discharges the common mode while leaving the differential charge alone. Inspired by Fig. 6 of Lewis and Gray [1]

Follow the charge and the gain falls out. During ϕ_1 the input is sampled onto $4C$ while the reset switch shorts C, so the feedback capacitor starts empty. During ϕ_2 the two left plates are shorted *to each other*, so they settle to whatever common voltage the pair demands, and the differential charge that was on them, $4CV_{in}$, has nowhere to go except onto C — the OTA's inputs are high

impedance and its own feedback holds them together. Charge conservation then gives $CV_{out} = 4CV_{in}$, so the gain is 4, and it is 4 because one capacitor is four times another rather than because any transistor did something particular.

Shorting the plates to each other rather than to ground is not a detail. Ground is a different node at each end of the chip, and any difference between the two would be sampled straight into the signal; shorting the pair to itself discharges the common mode without ever consulting ground, which is where the common mode rejection of this circuit comes from.

The real circuit in the paper carries a common-mode feedback network and a dummy switch beside every real one, both left out here. They matter enormously when you build it and not at all when you are working out what it does.

The output is only correct for a finite, but periodic, time interval. The circuit is discrete time. As long as all circuits afterwards also have a discrete-time input, then it's fine. An ADC can sample the output from the amplifier at the right time, and never notice that the output is shorted to a DC voltage in ϕ_1

We charge the capacitor $4C$ to the differential input voltage in ϕ_1

$$Q_1 = 4CV_{in}$$

Then we turn off ϕ_1, which opens all switches. The charge on $4C$ will still be Q_1 (except for higher order effects like charge injection from switches).

After a short time (non-overlap), we turn on ϕ_2, closing some of the switches. The OTA will start to force its two inputs to be the same voltage, and we short the left side of $4C$. After some time we would have the same voltage on the left side of $4C$ for the two capacitors, and another voltage on the right side of the $4C$ capacitors. The two capacitors must now have the same charge, so the difference in charge, or differential charge must be zero.

Physics tell us that charge is conserved, so our differential charge Q_1 cannot vanish into thin air. The difference in electrons that made Q_1 must be somewhere in our circuit.

Assume the designer of the circuit has done a proper job, then the Q_1 charge will be found on the feedback capacitors.

We now have a Q_1 charge on smaller capacitors, so the differential output voltage must be

$$Q_1 = 4CV_{in} = Q_2 = CV_{out}$$

The gain is

$$A = \frac{V_{out}}{V_{in}} = 4$$

Why would we go to all this trouble to get a gain of 4?

In general, we can sum up with the following equation.

$$\omega_{p|z} \propto f_{clk}\frac{C_1}{C_2}$$

We can use these "switched capacitor resistors" to get pole or zero frequency or gain proportional to the relative size of capacitors, which is a fantastic feature. Assume we make two identical capacitors in our layout. We won't know the absolute size of the capacitors on the integrated circuit, whether the C_1 is 100 fF or 80 fF, but we can be certain that if C_1 = 80 fF, then C_2 = 80 fF to a precision of around 0.1 %.

With switched capacitor amplifiers we can set an accurate gain, and we can set an accurate pole and zero frequency (as long as we have an accurate clock and a high DC gain OTA).

The switched capacitor circuits do have a drawback. They are discrete time circuits. As such, we must treat them with caution, and they will always need some analog filter before to avoid a phenomena we call aliasing.

21.4 Discrete-Time Signals

An random, Gaussian, continuous time, continuous value, signal has infinite information. The frequency can be anywhere from zero to infinity, the value have infinite levels, and the time division is infinitely small. We cannot store such a signal. We have to quantize.

If we quantize time to T = 1 ns, such that we only record the value of the signal every 1 ns, what happens to all the other information? The stuff that changes at 0.5 ns or 0.1 ns, or 1 ns.

We can always guess, but it helps to know, as in absolutely know, what happens. That's where mathematics come in. With mathematics we can prove things, and know we're correct.

21.4.1 The mathematics

Define

$$x_c(t)$$

as a continuous time, continuous value signal

Define

$$\ell(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}$$

Define

$$x_{sn}(t) = \frac{x_c(nT)}{\tau}[\ell(t - nT) - \ell(t - nT - \tau)]$$

where $x_{sn}(t)$ is a function of the continuous time signal at the time interval nT.

Define

$$x_s(t) = \sum_{n=-\infty}^{\infty} x_{sn}(t)$$

where $x_s(t)$ is the sampled, continuous time, signal.

Think of a sampled version of an analog signal as an infinite sum of pulse trains where the area under the pulse train is equal to the analog signal.

Why do this?

With a exact definition of a sampled signal in the time-domain it's sometimes possible to find the Laplace transform, and see how the frequency spectrum looks.

If

$$x_s(t) = \sum_{n=-\infty}^{\infty} x_{sn}(t)$$

Then

$$X_{sn}(s) = \frac{1}{\tau}\frac{1 - e^{-s\tau}}{s}x_c(nT)e^{-snT}$$

And

$$X_s(s) = \frac{1}{\tau}\frac{1 - e^{-s\tau}}{s}\sum_{n=-\infty}^{\infty} x_c(nT)e^{-snT}$$

Thus

$$\lim_{\tau \to 0} \rightarrow X_s(s) = \sum_{n=-\infty}^{\infty} x_c(nT)e^{-snT}$$

The spectrum of a sampled signal is an infinite sum of frequency shifted spectra

or equivalently

When you sample a signal, then there will be copies of the input spectrum at every

$$nf_s$$

However, if you do an FFT of a sampled signal, then all those infinite spectra will fold down between

$$0 \rightarrow f_{s1}/2$$

or

$$-f_{s1}/2 \rightarrow f_{s1}/2$$

for a complex FFT

21.4.2 Python discrete time example

If your signal processing skills are a bit thin, now might be a good time to read up on FFT, Laplace transform and But what is the Fourier Transform?

In python we can create a demo and see what happens when we "sample" a "continuous time" signal. Hopefully it's obvious that it's impossible to emulate a "continuous time" signal on a digital computer. After all, it's digital (ones and zeros), and it has a clock!

We can, however, emulate to any precision we want.

The code below has four main sections. First is the time vector. I use Numpy, which has a bunch of useful features for creating ranges, and arrays.

Secondly, I create continuous time signal. The time vector can be used in numpy functions, like `np.sin()`, and I combine three sinusoid plus some noise. The sampling vector is a repeating pattern of 11000000, so our sample rate is 1/8'th of the input sample rate. FFT's can be unwieldy beasts. I like to use coherent sampling, however, here the tone is deliberately placed halfway between two FFT bins, so the record is not coherent.

The alternative to coherent sampling is to apply a window function before the FFT, that's the reason for the Hanning window below.

There is an interactive version of this example where the tone frequency, the sampling pattern and the window are sliders. Turning the Hanning window off, and then turning coherent sampling on, is worth thirty seconds of your time.

dt.py - interactive

```
#- Create a time vector
N = 2**13
t = np.arange(N)

#- Create the "continuous time" signal with multiple
#- "sinusoidal signals and some noise
#- f1 is deliberately halfway between FFT bins, so the
#- record is not coherent and the window has a job to do
f1 = 233.5/N
fd = 1/N*119
x_s = np.sin(2*np.pi*f1*t) + 1/1024*np.random.randn(N) + \
    0.5*np.sin(2*np.pi*(f1-fd)*t) + 0.5*np.sin(2*np.pi*(f1+fd)*t)

#- Create the sampling vector, and the sampled signal
t_s_unit = [1,1,0,0,0,0,0,0]
t_s = np.tile(t_s_unit,int(N/len(t_s_unit)))
x_sn = x_s*t_s

#- Convert to frequency domain with a hanning window to avoid FFT bin
#- energy spread
Hann = True
if(Hann):
    w = np.hanning(N+1)
else:
    w = np.ones(N+1)
X_s = np.fft.fftshift(np.fft.fft(np.multiply(w[0:N],x_s)))
X_sn = np.fft.fftshift(np.fft.fft(np.multiply(w[0:N],x_sn)))
```

Try to play with the code, and see if you can understand what it does.

Below are the plots. On the left side is the "continuous value, continuous time" emulation, on the right side "discrete time, continuous value".

The top plots are the time domain, while the bottom plots is frequency domain.

The FFT is complex, so that's why there are six sinusoids bottom left. The frequency axis is normalized to the sample rate, so it runs from $-f_s/2$ to $+f_s/2$ with 0 Hz in the middle.

The spectral copies can be seen bottom right. How many spectral copies, and the distance between them will depend on the sample rate (length of `t_s_unit`). Try to play around with the code and see what happens.

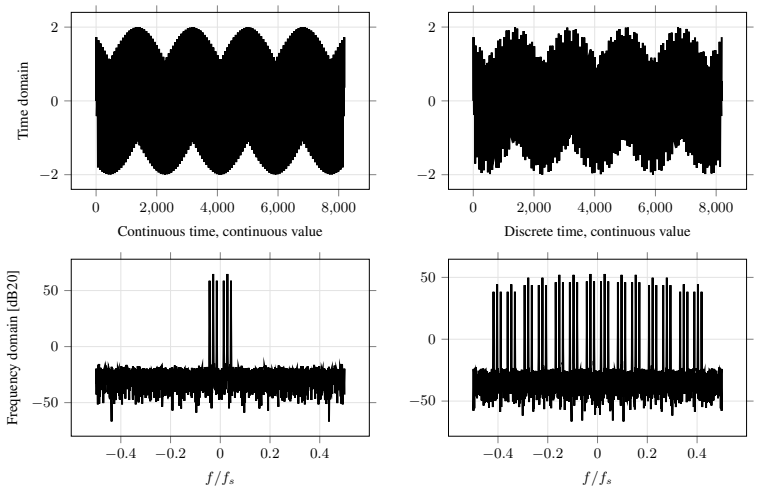


Figure 7: Time domain and spectrum of the emulated continuous time signal (left) and the sampled signal with its spectral copies (right)

21.4.3 Aliasing, bandwidth and sample rate theory

I want you to internalize that the spectral copies are real. They are not some "mathematical construct" that we don't have to deal with.

They are what happens when we sample a signal into discrete time. Imagine a signal with a band of interest as shown below in Green. We sample at f_s. The pink and red unwanted signals do not disappear after sampling, even though they are above the Nyquist frequency ($f_s/2$). They fold around $f_s/2$, and may appear in-band. That's why it's important to band limit analog signals before they are sampled.

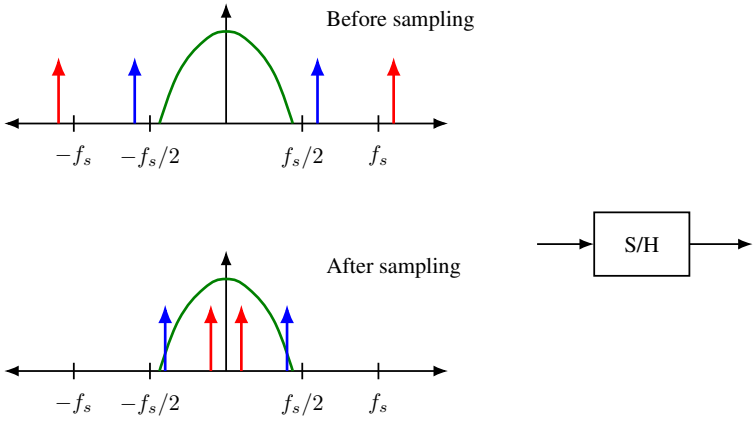


Figure 8: Spectrum before and after sampling: unwanted signals above $f_s/2$ fold into the wanted band

With an anti-alias filter (yellow) we ensure that the unwanted components are low enough before sampling. As a result, our wanted signal (green) is undisturbed.

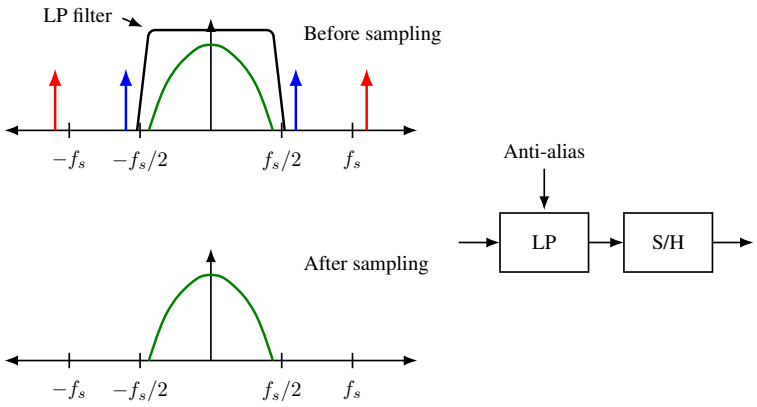


Figure 9: An anti-alias low-pass filter attenuates the unwanted components before sampling

Assume that we're interested in the red signal. We could still use a sample rate of f_s. If we bandpass-filtered all but the red signal the red signal would fold on sampling, as shown in the figure below.

Remember that the Nyquist-Shannon states that a sufficient no-loss condition is to sample signals with a sample rate of twice the bandwidth of the signal.

Nyquist-Shannon has been extended for sparse signals, compressed sensing, and non-uniform sampling to demonstrate that it's sufficient for the average sample rate to be twice the bandwidth. One 2009 paper Blind Multiband Signal Reconstruction: Compressed Sensing for Analog Signal [2] is a good place to start to delve into the latest on signal reconstruction.

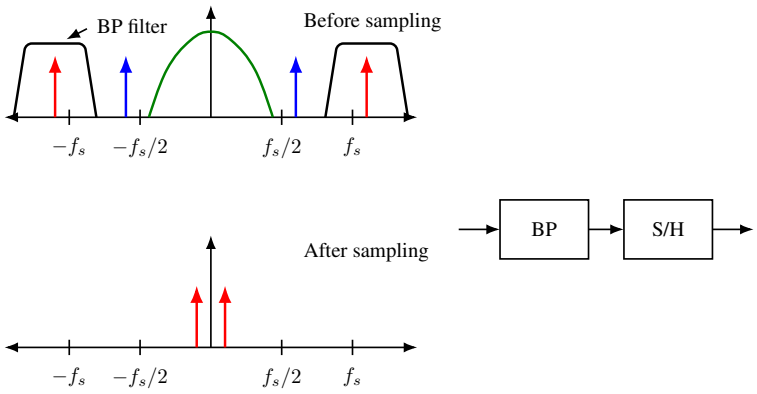


Figure 10: Sub-sampling: a band-pass filtered signal above $f_s/2$ folds down to low frequency on sampling

21.4.4 Z-transform

Someone got the idea that writing

$$X_s(s) = \sum_{n=-\infty}^{\infty} x_c(nT)e^{-snT}$$

was cumbersome, and wanted to find something better.

$$X_s(z) = \sum_{n=-\infty}^{\infty} x_c[n]z^{-n}$$

For discrete time signal processing we use Z-transform

If you're unfamiliar with the Z-transform, read the book or search https://en.wikipedia.org/wiki/Z-transform

The nice thing with the Z-transform is that the exponent of the z tell's you how much delayed the sample $x_c[n]$ is. A block that delays a signal by 1 sample could be described as $x_c[n]z^{-1}$, and an accumulator

$$y[n] = y[n-1] + x[n]$$

in the Z domain would be

$$Y(z) = z^{-1}Y(z) + X(z)$$

With a Z-domain transfer function of

$$\frac{Y(z)}{X(z)} = \frac{1}{1-z^{-1}}$$

21.4.5 Pole-Zero plots

If you're not comfortable with pole/zero plots, have a look at

What does the Laplace Transform really tell us

Think about the pole/zero plot as a surface your looking down onto. At $a = 0$ we have the steady state fourier transform. The "x" shows the complex frequency where the fourier transform goes to infinity.

Any real circuit will have complex conjugate, or real, poles/zeros. A combination of two real circuits where one path is shifted 90 degrees in phase can have non-conjugate complex poles/zeros.

If the "x" is $a < 0$, then any perturbation will eventually die out. If the "x" is on the $a = 0$ line, then we have a oscillator that will ring forever. If the "x" is $a > 0$ then the oscillation amplitude will grow

without bounds, although, only in Matlab. In any physical circuit an oscillation cannot grow without bounds forever.

Growing without bounds is the same as "being unstable".

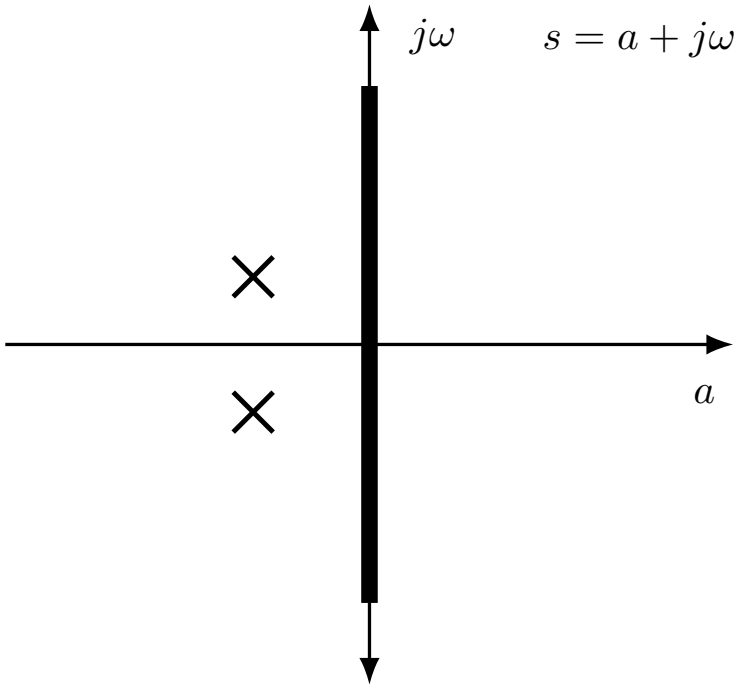


Figure 11: A complex conjugate pole pair in the s-plane

21.4.6 Z-domain

Spectra repeat every

$$2\pi$$

As such, it does not make sense to talk about a plane with a a and a $j\omega$. Rather we use the complex number $z = a + jb$.

As long as the poles ("x") are within the unit circle, oscillations will die out. If the poles are on the unit-circle, then we have an oscillator. Outside the unit circle the oscillation will grow without bounds, or in other words, be unstable.

We can translate between Laplace-domain and Z-domain with the Bi-linear transform

$$s = \frac{2}{T}\frac{z-1}{z+1}$$

Warning: First-order approximation https://en.wikipedia.org/wiki/Bilinear_transform

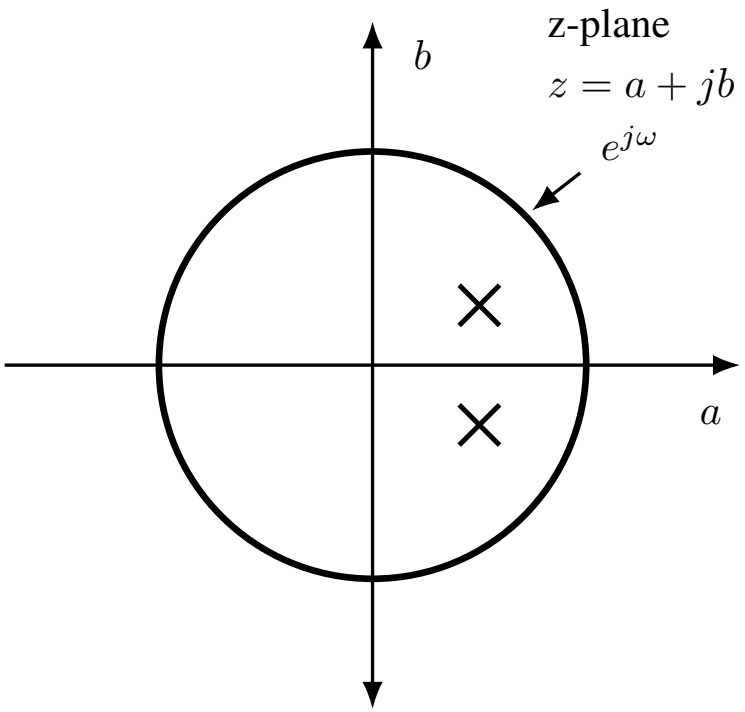


Figure 12: The z-plane with the unit circle and a complex conjugate pole pair

21.4.7 First order filter

Assume a first order filter given by the discrete time equation.

$$y[n+1] = bx[n] + ay[n] \Rightarrow Yz = bX + aY$$

The "n" index and the "z" exponent can be chosen freely, which sometimes can help the algebra.

$$y[n] = bx[n-1] + ay[n-1] \Rightarrow Y = bXz^{-1} + aYz^{-1}$$

The transfer function can be computed as

$$H(z) = \frac{b}{z-a}$$

From the discrete time equation we can see that the impulse will never die out. We're adding the previous output to the current input. That means the circuit has infinite memory. Accordingly, filters of this type are known as. Infinite-impulse response (IIR)

$$h[n] = \begin{cases} k & \text{if } n < 1 \\ a^{n-1}b + a^n k & \text{if } n \geq 1 \end{cases}$$

_ Head's up: Fig 13.12 in AIC is wrong _

Here k is the initial state $y[0]$. From the impulse response it can be seen that the pole of $H(z) = b/(z - a)$ sits at $z = a$, and b only scales the output, so everything depends on $|a|$.

Three cases, matching the z-plane picture above. If $|a| < 1$ the response decays and the filter is stable. If $|a| > 1$ it grows without bound and the filter is unstable. Exactly on the unit circle, $|a| = 1$, it neither decays nor grows: the impulse rings for ever at constant amplitude, which is an oscillator. That last case is called *marginally* stable, and in a real circuit it does not exist — component tolerance will push the pole to one side or the other, and only one of those sides is survivable.

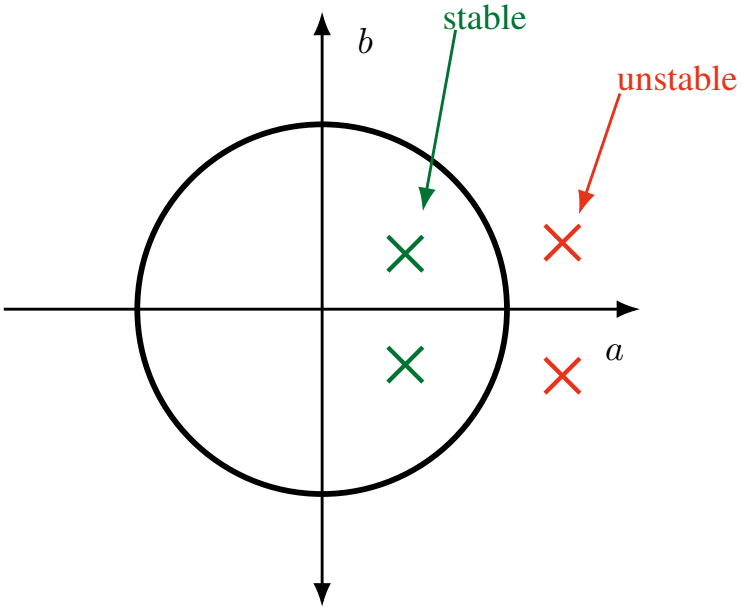


Figure 13: Poles inside the unit circle are stable, poles outside are unstable

21.4.8 Second order filter

A single real pole can only do so much. If we feed back two delayed outputs

$$y[n] = bx[n-1] + 2a\,y[n-1] - (a^2 + b^2)\,y[n-2]$$

$$H(z) = \frac{bz}{z^2 - 2az + (a^2 + b^2)}$$

then the denominator factors as $(z - z_p)(z - z_p^*)$ with a complex conjugate pole pair at

$$z_p = a + jb$$

which is exactly the complex frequency from the Z-domain plot earlier. As long as $|a + jb| < 1$ the poles are inside the unit circle and the filter is stable. Complex poles also mean the magnitude response peaks near the pole angle — the filter resonates.

The second order filter can be implemented in python, and it's really not hard. See below. The $x_s n$ vector is from the previous python example.

There are smarter, and faster ways to do IIR filters (and FIR) in python, see scipy.signal.iirfilter

From the plot below we can see the sampled time domain and spectra on the left, and the filtered time domain and spectra on the right. The two spectra share the same y-axis, so the attenuation can be read directly. The poles sit at $0.85 \pm j0.25$ ($|z| = 0.89$, pole angle $\approx 0.046\ f_s$): the image near the pole angle is picked out and even amplified a little by the resonance, while the spectral copies further out drop with 40 dB/decade.

The interactive version of this example adds the pole position as a slider, and draws the z-plane next to the spectrum, so you can watch the pole move inside the unit circle and the corner frequency follow it.

iir.py - interactive

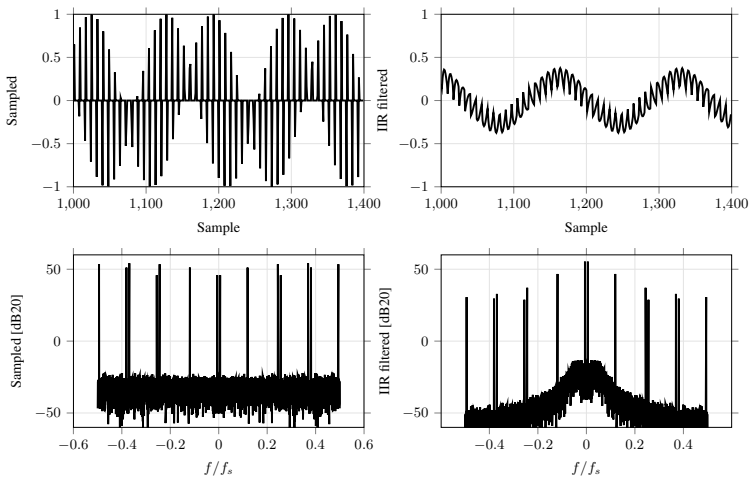


Figure 14: Time domain and spectrum of the sampled signal (left) and the second order IIR filtered output (right)

```
#- Second-order IIR filter with a complex conjugate
#- pole pair at z = a +/- jb. Stable if |a + jb| < 1.
b = 0.25
a = 0.85
z = a + 1j*b
z_abs = np.abs(z)
```

```
print("|z| = " + str(z_abs))
y = np.zeros(N)
for i in range(2,N):
    y[i] = b*x_sn[i-1] + 2*a*y[i-1] - (a*a + b*b)*y[i-2]
```

The IIR filter we implemented above is a resonant low-pass filter: it picks out the image near its pole angle and rejects the copied spectra further out, as expected.

21.4.9 Finite-impulse response(FIR)

FIR filters are unconditionally stable, since the impulse response will always die out. FIR filters are a linear sum of delayed inputs.

In my humble opinion, there is nothing wrong with an IIR. Yes, they could become unstable, however, they can be designed safely. I'm not sure there is a theological feud on IIR vs FIR, I suspect there could be. Talk to someone that knows digital filters better than me.

But be wary of rules like "IIR are always better than FIR" or vice versa. Especially if statements are written in books. Remember that the book was probably written a decade ago, and based on papers two decades old, which were based on three decades old state of the art. Our abilities to use computers for design has improved a bit the last three decades.

The simplest useful FIR is a moving average. Take the current sample and the two before it, weight each by a third, and add them. There is no feedback path anywhere in it, so a disturbance can only live for as many samples as there are taps, and then it is gone. That is the whole of the stability argument.

$$H(z) = \frac{1}{3}\sum_{i=0}^{2} z^{-i} = \frac{1}{3}\left(1 + z^{-1} + z^{-2}\right)$$

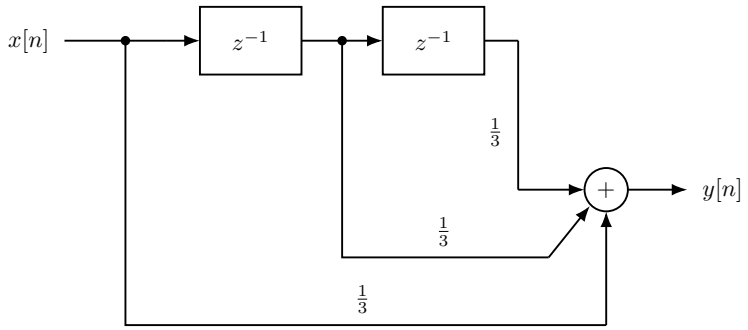


Figure 15: FIR filter summing the input and two delayed samples, each weighted by 1/3

21.5 Switched-Capacitor

Below is an example of a switched-capacitor circuit during phase 1. Think of the two phases as two different configurations of a circuit, each with a specific purpose.

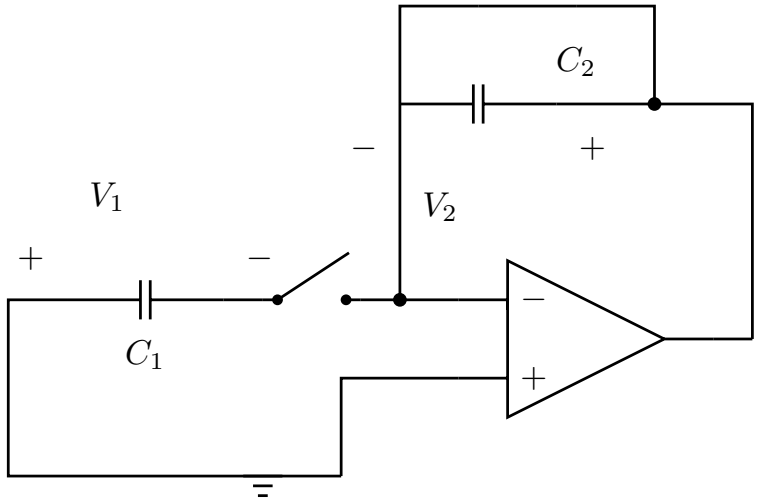


Figure 16: SC circuit during phase 1: V_1 is stored on C_1 while C_2 is shorted

This is the SC circuit during the sampling phase. Imagine that we somehow have stored some voltage V_1 on capacitor C_1 (the switches for that sampling or storing are not shown). The charge on C_1 is

$$Q_{1\phi_1\$} = C_1 V_1$$

The C_2 capacitor is shorted, as such, $V_2 = 0$, which must mean that the charge on C_2 given by

$$Q_{2\phi_1\$} = 0$$

The voltage at the negative input of the OTA must be 0 V, as the positive input is 0 V, and we assume the circuit has settled all transients.

Imagine we (very carefully) open the circuit around C_2 and close the circuit from the negative side of C_1 to the OTA negative input, as shown below.

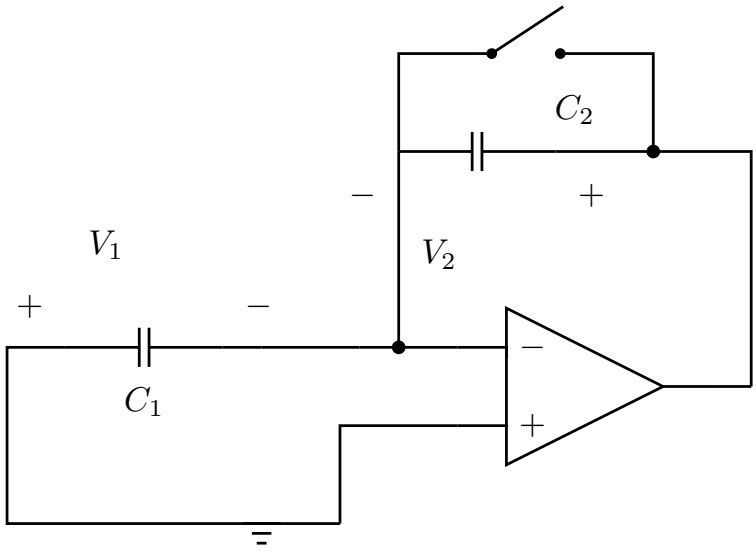


Figure 17: SC circuit during phase 2: the OTA transfers the charge from C_1 to C_2

It's the OTA that ensures that the negative input is the same as the positive input, but the OTA cannot be infinitely fast. At the same time, the voltage across C_1 cannot change instantaneously. Neither can the voltage across C_2. As such, the voltage at the negative input must immediately go to $-V_1$ (ignoring any parasitic capacitance at the negative input).

The OTA does not like its inputs to be different, so it will start to charge C_2 to increase the voltage at the negative input to the OTA. When the negative input reaches 0 V the OTA is happy again. At that point the charge on C_1 is

$$Q_{1\phi_2\$} = 0$$

A key point is that even though the voltages have now changed, there is zero volts across C_1, and thus there cannot be any charge on C_1. The charge that was there cannot have disappeared. The negative input of the OTA is a high impedance node, and cannot supply charge. The charge must have gone somewhere, but where?

In process of changing the voltage at the negative input of the OTA we've changed the voltage across C_2. The voltage change must exactly match the charge that was across C_1, as such

$$Q_{2\phi_2\$} = Q_{1\phi_1\$} = C_1V_1 = C_2V_2$$

thus

$$\frac{V_2}{V_1} = \frac{C_1}{C_2}$$

21.5.1 Switched capacitor gain circuit

Redrawing the previous circuit, and adding a few more switches we can create a switched capacitor gain circuit.

There is now a switch to sample the input voltage across C_1 during phase 1 and reset C_2. During phase 2 we configure the circuit to leverage the OTA to do the charge transfer from C_1 to C_2.

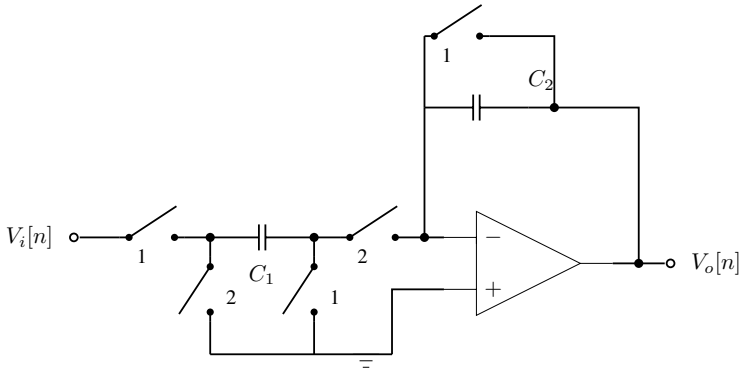


Figure 18: Switched capacitor gain circuit sampling V_i on C_1 in phase 1 and transferring the charge to C_2 in phase 2

The discrete time output from the circuit will be as shown below. It's only at the end of the second phase that the output signal is valid. As a result, it's common to use the sampling phase of the next circuit close to the end of phase 2.

For charge to be conserved the clocks for the switch phases must never be high at the same time.

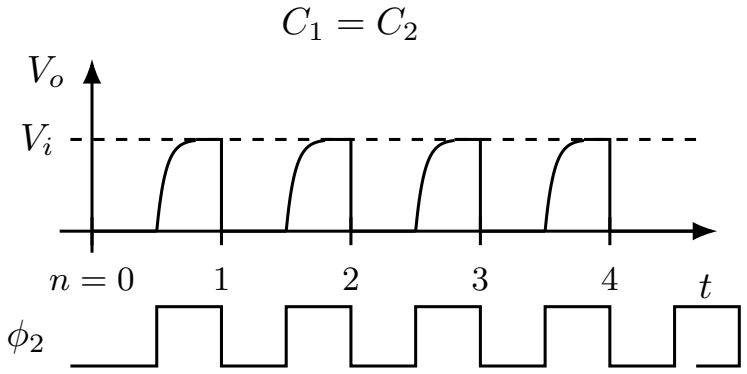


Figure 19: Discrete time output of the gain circuit, only valid towards the end of phase 2

The discrete time, Z-domain and transfer function is shown below. The transfer function tells us that the circuit is equivalent to a gain, and a delay of one clock cycle. The cool thing about switch

capacitor circuits is that the precision of the gain is set by the relative size between two capacitors. In most technologies that relative sizing can be better than 0.1 %.

Gain circuits like the one above find use in most Pipelined ADCs, and are common, with some modifications, in Sigma-Delta ADCs.

$$V_o[n+1] = \frac{C_1}{C_2} V_i[n]$$

$$V_o z = \frac{C_1}{C_2} V_i$$

$$\frac{V_o}{V_i} = H(z) = \frac{C_1}{C_2} z^{-1}$$

21.5.2 Switched capacitor integrator

Removing one switch we can change the function of the switched capacitor gain circuit. If we don't reset C_2 then we accumulate the input charge every cycle.

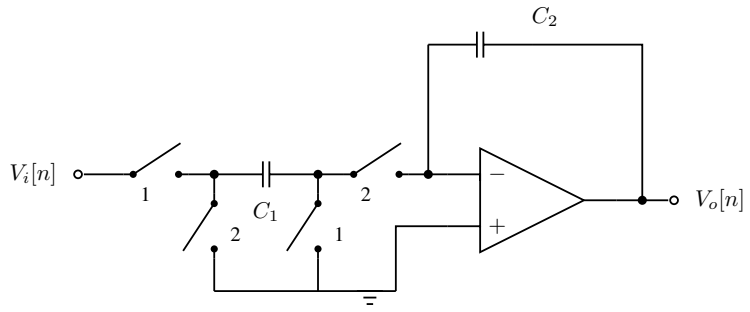


Figure 20: Switched capacitor integrator: without the reset switch the charge accumulates on C_2

The output now will grow without bounds, so integrators are most often used in filter circuits, or sigma-delta ADCs where there is feedback to control the voltage swing at the output of the OTA.

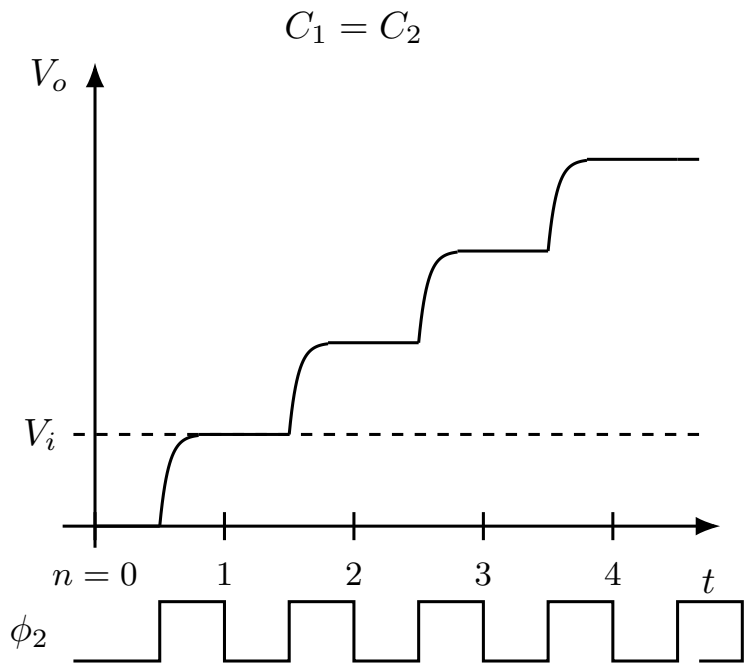


Figure 21: Integrator output growing every clock cycle as the input charge is accumulated

Make sure you read and understand the equations below, it's good to realize that discrete time equations, Z-domain and transfer functions in the Z-domain are actually easy.

Start from what the circuit does in one cycle. The charge already on C_2 stays there, because nothing discharges it now, and the charge $C_1 V_i$ sampled during the previous phase is added to it. Divide by C_2 to turn charge into voltage and that sentence is the first line: this output equals the last output plus a scaled copy of the last input.

$$V_o[n] = V_o[n-1] + \frac{C_1}{C_2} V_i[n-1]$$

Take that to the Z-domain by the one rule you need: a delay of one sample is a multiplication by z^{-1}. So $V_o[n-1]$ becomes $z^{-1}V_o$, $V_i[n-1]$ becomes $z^{-1}V_i$, and collecting the output terms on the left gives

$$V_o - z^{-1}V_o = \frac{C_1}{C_2} z^{-1} V_i$$

Divide through and the transfer function falls out. Maybe one confusing thing is that multiple transfer functions can mean the same thing, as below. They differ only by a factor of z on top and bottom, which is legal algebra and changes nothing:

$$H(z) = \frac{C_1}{C_2}\frac{z^{-1}}{1 - z^{-1}} = \frac{C_1}{C_2}\frac{1}{z - 1}$$

Look at where the pole sits: $z = 1$, exactly on the unit circle. By the rule from the first order filter section that is an oscillator, a circuit whose impulse response never dies out — which for an integrator is not a defect but the entire specification. It is also why the previous figure grows without bound, and why an integrator is only ever used inside a loop that puts something else in charge of the output swing.

21.5.3 Noise

Capacitors don't make noise, but switched-capacitor circuits do have noise. The noise comes from the thermal, flicker, burst noise in the switches and OTA's. Both phases of the switched capacitor circuit contribute noise. As such, the output noise of a SC circuit is usually

$$V_n^2 > \frac{2kT}{C}$$

This is worth deriving rather than quoting, because it is the number that sets the size of almost every capacitor in this course, and because the way it falls out is genuinely surprising.

A closed switch is a resistor R_{on}, and a resistor produces a thermal noise density of $4kTR$. That noise reaches the capacitor through the RC low-pass the switch and capacitor form together, whose equivalent noise bandwidth is $1/(4RC)$. Multiply the two:

$$\overline{v_n^2} = 4kTR \times \frac{1}{4RC} = \frac{kT}{C}$$

The resistance cancels. A wider switch has less noise density and more bandwidth, in exactly compensating proportion, so the sampled noise does not care how good the switch is. It does not care about the clock frequency either. The only thing that sets it is the capacitor.

When the switch opens, whatever noise voltage happened to be on the capacitor at that instant is trapped there and becomes part of the sample. So each sampling event contributes kT/C, and a switched-capacitor circuit samples on both phases. The two events are separated in time and uncorrelated, so by the rule derived just below their variances add, which is where the factor of two comes from. The inequality is there because the OTA is also in the signal path and contributes on top.

Put a number on it. At room temperature with a 1 pF capacitor,

$$\sqrt{\frac{kT}{C}} = \sqrt{\frac{1.38 \times 10^{-23} \times 300}{10^{-12}}} \approx 64\ \mu V_{rms}$$

Now look at what that costs. To halve the noise you need four times the capacitor, and the OTA has to drive that capacitor within half a clock period, so its transconductance — and its current — must go up fourfold too. Every extra bit of resolution costs four times the capacitor and four times the power. That single relation is why analog circuits stopped getting cheaper when transistors did, and it is worth carrying out of this chapter even if you forget the charge equations.

I find that sometimes it's useful with a repeat of mathematics, and since we're talking about noise.

The mean, or average of a signal is defined as

Mean

$$\overline{x(t)} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t)dt$$

Define

Mean Square

$$\overline{x^2(t)} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x^2(t)dt$$

How much a signal varies can be estimated from the Variance

$$\sigma^2 = \overline{x^2(t)} - \overline{x(t)}^2$$

where

$$\sigma$$

is the standard deviation. If mean is removed, or is zero, then

$$\sigma^2 = \overline{x^2(t)}$$

Assume two random processes,

$$x_1(t)$$

and

$$x_2(t)$$

with mean of zero (or removed).

$$x_{tot}(t) = x_1(t) + x_2(t)$$

$$x_{tot}^2(t) = x_1^2(t) + x_2^2(t) + 2x_1(t)x_2(t)$$

Variance (assuming mean of zero)

$$\sigma_{tot}^2 = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x_{tot}^2(t)dt$$

$$\sigma_{tot}^2 = \sigma_1^2 + \sigma_2^2 + \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} 2x_1(t)x_2(t)dt$$

Assuming uncorrelated processes (covariance is zero), then

$$\sigma_{tot}^2 = \sigma_1^2 + \sigma_2^2$$

In other words, if two noises are uncorrelated, then we can sum the variances. If the noise sources are correlated, for example, noise comes from the same transistor, but takes two different paths through the circuit, then we cannot sum the variances. We must also add the co-variance.

21.5.4 Sub-circuits for SC-circuits

Switched-capacitor circuits are so common that it's good to delve a bit deeper, and understand the variants of the components that make up SC circuits.

21.5.4.1 OTA

At the heart of the SC circuit we usually find an OTA. Maybe a current mirror, folded cascode, recycling cascode, or my favorite: a fully differential current mirror OTA with cascoded, gain boosted, output stage using a parallel common mode feedback.

Not all SC circuits use OTAs, there are also comparator based SC circuits [3].

Below is a fully-differential two-stage OTA that will work with most SC circuits. The notation "24F1F25" means "the width is 24 F" and "length is 1.25 F", where "F" is the minimum gate length in that technology.

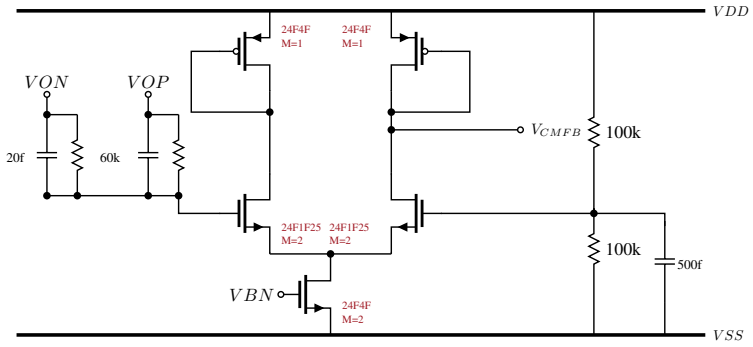


Figure 22: The common mode feedback amplifier. VON and VOP are sensed through 60k||20f networks and compared against a 100k/100k mid-supply reference. Each load PMOS is diode connected on its own - the gates are not tied together - so the gain is the modest, well defined g_{mn}/g_{mp} that a common mode loop wants, and the correction leaves as V_{CMFB}

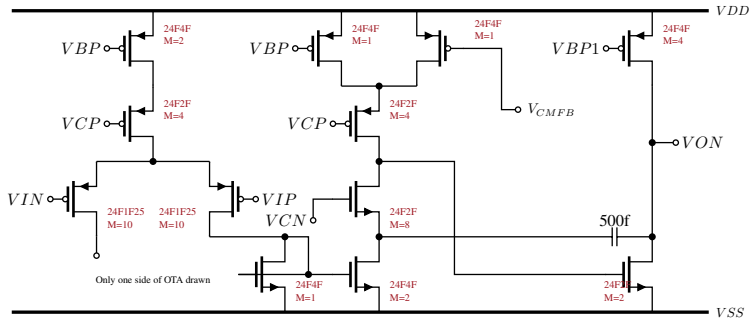


Figure 23: The OTA itself, one side drawn, sized in multiples of the minimum gate length F: a cascoded PMOS tail into the input pair, a cascoded mirror making the first stage output, and a common source second stage with 500 fF of cascode compensation. V_{CMFB} arrives from the amplifier in Figure 13

As bias circuit to make the voltages the below will work

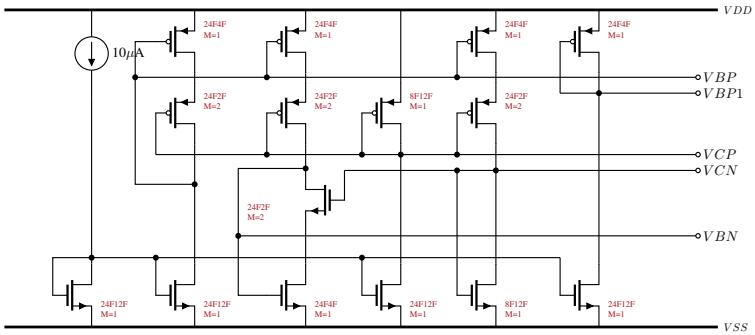


Figure 24: The bias generator. The 10 uA reference is mirrored once in an ordinary current mirror - the line along the bottom - and everything above it is wide swing cascode: the narrow 8F12F devices set VCP and VCN, and in each master the mirror device's gate hangs on the far end of its own stack

21.5.4.2 Switches

If your gut reaction is "switches, that's easy", then you're very wrong. Switches can be incredibly complicated. All switches will be made of transistors, but usually we don't have enough headroom to use a single NMOS or PMOS. We may need a transmission gate

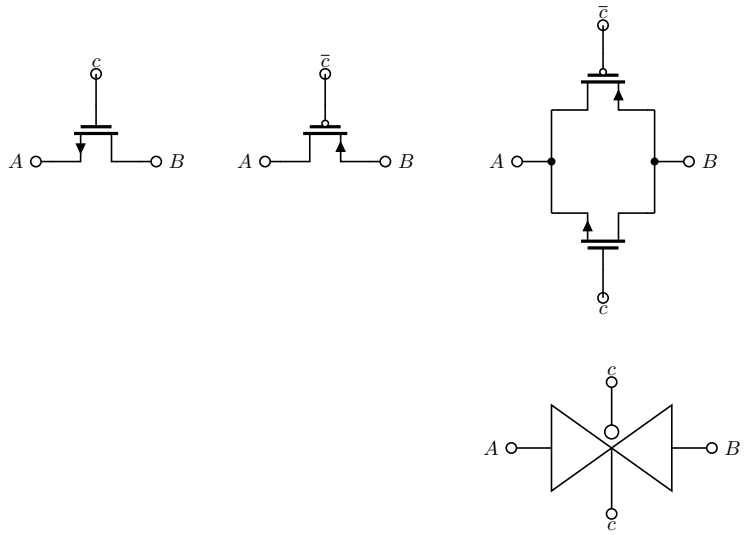


Figure 25: Switch implementations: NMOS, PMOS, transmission gate, and the transmission gate symbol

The challenge with transmission gates is that when the voltage at the input is in the middle between VDD and ground then both PMOS and NMOS, although they are on , they might not be that on. Especially in nano-scale CMOS with a 0.8 V supply and 0.5 V threshold voltage. The resistance mid-rail might be too large.

For switched-capacitor circuits we must settle the voltages to the required accuracy. In general

$$t > -\ln(\text{error})\tau$$

For example, for a 10-bit ADC we need $t > -\ln(1/1024)\tau = 6.9\tau$. This means we need to wait at least 6.9 time constants for the voltage to settle to 10-bit accuracy in the switched capacitor circuit.

Assume the capacitors are large due to noise, then the switches must be low resistance for a reasonable time constant. Larger switches have smaller resistance, however, they also have more charge in the inversion layer, which leads to charge injection when the switches are turned off. Accordingly, larger switches are not always the solution.

Sometimes it may be sufficient to switch the bulks, as shown on the left below. But more often than one would like, we have to implement bootstrapped switches as shown on the right.

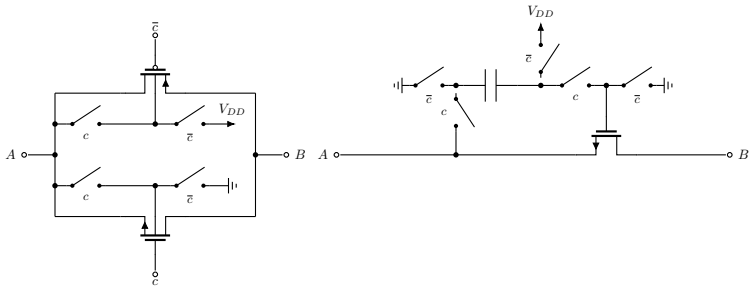


Figure 26: Transmission gate with switched bulks (left) and a bootstrapped switch (right)

The switch I used in my JSSC SAR [4] is a fully differential bootstrapped switch with cross coupled dummy transistors. The JSSC SAR I've also ported to GF130NM, as shown below. The switch is at the bottom.

wulffern/sun_sar9b_sky130nm

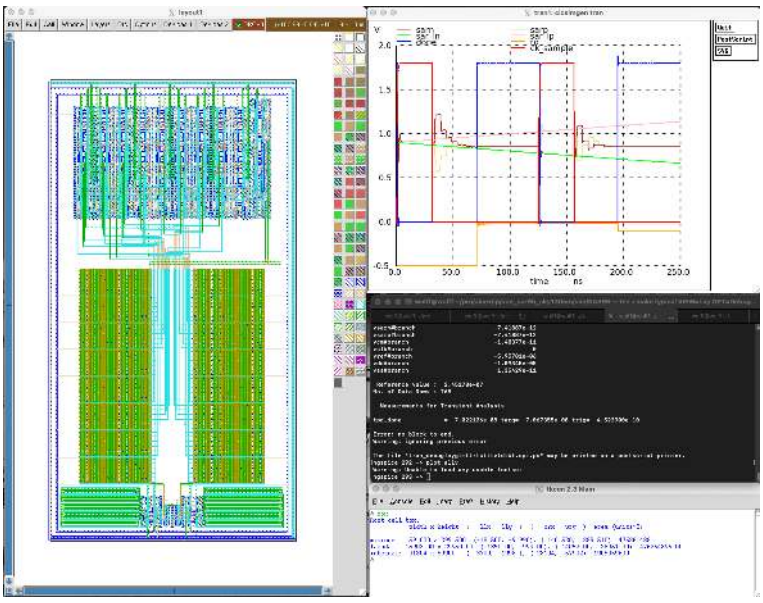


Figure 27: Layout and transient simulation of the 9-bit SAR ADC, with the bootstrapped sampling switch at the bottom

The bootstrapped switch looks like the one below.

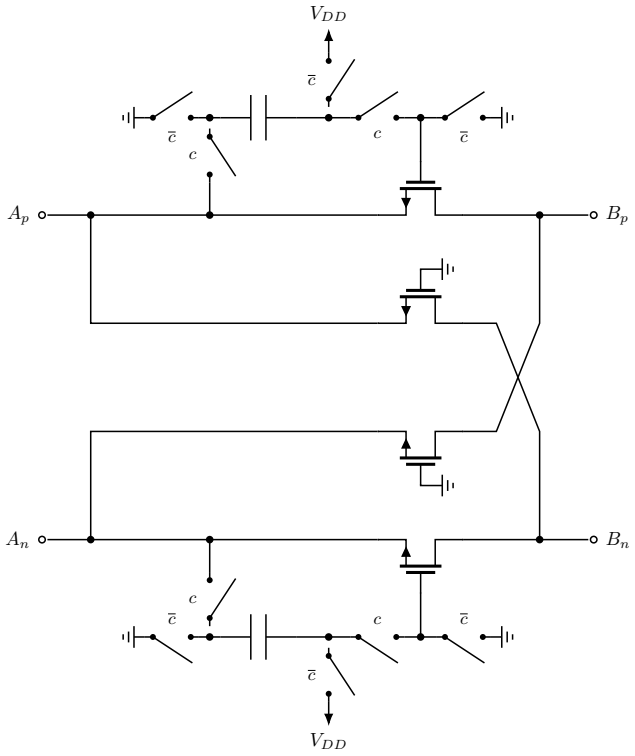


Figure 28: Fully differential bootstrapped switch with cross coupled dummy transistors

21.5.4.3 Non-overlapping clocks

Everything in this chapter has rested on charge going exactly where we said it goes. That holds only if the two phases are never on at the same time. Overlap them, even briefly, and a path opens from the input straight through to the summing node while the previous charge is still there: some charge escapes, some arrives early, and the gain is no longer C_1/C_2 but something that depends on how long the overlap lasted. It is a gain error that changes with temperature and corner, which is the worst kind.

The generator below solves this the obvious way. Each phase is fed back to the gate of the other's driver, so neither can rise until the other has fallen, and the delay chain sets how much dead time sits between them. The cost is that dead time — clock period you are not using — so it wants to be sufficient and no more.

Sufficient in *all corners*, though. The delay chain is made of inverters, and the ring oscillator plots earlier in the logic chapter show what happens to inverter delay across process and temperature: it moves by a factor of two or three. Simulate the non-overlap at the fast corner, where it is smallest, not at typical.

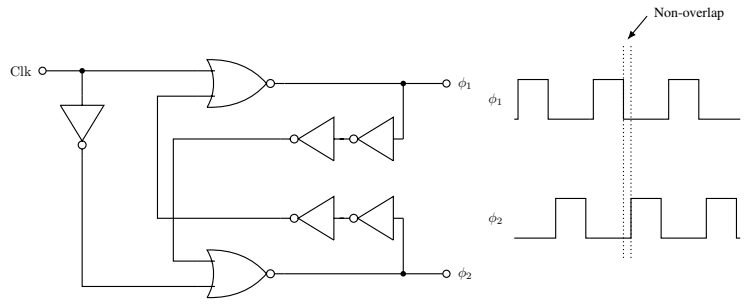


Figure 29: Non-overlapping clock generator and the resulting phases ϕ_1 and ϕ_2

21.5.5 Example

In the circuit below there is an example of a switched capacitor circuit used to increase the ΔV_D across the resistor. We can accurately set the gain, and thus the equation for the differential output will be

$$V_O(z) = 10\frac{kT}{q}\ln(N)z^{-1}$$

Every factor in that expression comes from somewhere earlier, and it is worth naming them, because this one circuit is most of the course so far in a single schematic.

The $\frac{kT}{q}\ln(N)$ is the PTAT difference between two diode voltages at a current ratio N, from the references chapter. It is small, a few tens of millivolts, and it is proportional to absolute temperature, which is what makes it useful as a temperature signal.

The 10 is C_1/C_2, a capacitor ratio, from this chapter. That is the whole reason for doing the amplification this way: a capacitor ratio in one piece of silicon matches to a fraction of a percent, so the gain is accurate in a way a resistor ratio or a transistor parameter would not be, and it does not drift with temperature.

The z^{-1} is one clock period of delay, because the charge sampled in phase 1 does not reach the output until phase 2.

So the circuit takes a signal defined by physical constants, multiplies it by a number defined by geometry, and delivers the result one clock later. Nothing in the answer depends on a transistor's threshold voltage, its mobility, or the supply. That is the point of switched capacitor circuits, and it is why they survived into processes where almost nothing else about analog design got easier.

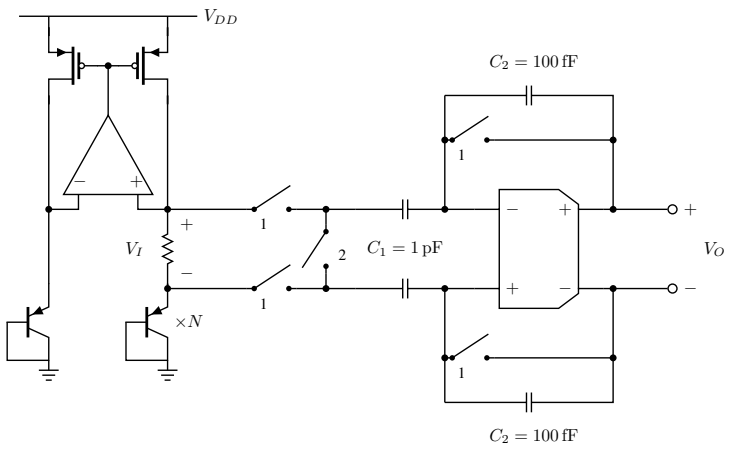


Figure 30: Switched capacitor amplifier scaling the bipolar generated voltage by $C_1/C_2 = 10$

21.6 Summary

The one-page version of this chapter:

- A switched capacitor is a resistor R = 1/(f C) made of ratio-accurate parts: SC filters get their time constants from C ratios and a clock
- Sampling moves the world to discrete time: the z-domain, aliasing and the sample-rate theorem come with it
- The parasitic-insensitive integrator and the SC gain stage are the two workhorse circuits; correlated double sampling throws in offset removal
- Every sample costs kT/C of noise - capacitor sizes come from the noise budget, not the layout
- Switches need non-overlapping clocks, and bootstrapping when the signal swings
- Behind every SC circuit stands an OTA and its bias; the settling budget is half a clock period, verified in the transient

21.7 Would you like to know more?

Blind Multiband Signal Reconstruction: Compressed Sensing for Analog Signal [2]

Comparator-based switched-capacitor pipelined analog-to-digital converter with comparator preset, and comparator delay compensation [3]

A Compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for Bluetooth Low Energy Receivers [4]

A 10-bit 50-MS/s SAR ADC With a Monotonic Capacitor Switching Procedure [5]

Low Voltage, Low Power, Inverter-Based Switched-Capacitor Delta-Sigma Modulator [6]

Ring Amplifiers for Switched Capacitor Circuits [7]

A Switched-Capacitor RF Power Amplifier [8]

Design of Active N-Path Filters [9]

[1] S. H. Lewis and P. R. Gray, "A pipelined 5-Msample/s 9-bit analog-to-digital converter," *ieee_j_jssc*, vol. 22, no. 6, 1987.

[2] M. Mishali and Y. C. Eldar, "Blind multiband signal reconstruction: Compressed sensing for analog signals," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 993–1009, 2009, doi: 10.1109/TSP.2009.2012791.

[3] C. Wulff and T. Ytterdal, "Comparator-based switched-capacitor pipelined analog-to-digital converter with comparator preset, and comparator delay compensation," *Analog Integrated Circuits and Signal Processing*, vol. 67, no. 1, pp. 31–40, 2010, doi: 10.1007/s10470-010-9576-3.

[4] C. Wulff and T. Ytterdal, "A compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for bluetooth low energy receivers," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 7, pp. 1915–1926, 2017, doi: 10.1109/JSSC.2017.2685463.

[5] C. C. Liu, S. J. Chang, G. Y. Huang, and Y. Z. Lin, "A 10-bit 50-MS/s SAR ADC With a Monotonic Capacitor Switching Procedure," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 731–740, 2010, doi: 10.1109/JSSC.2010.2042254.

[6] Y. Chae and G. Han, "Low voltage, low power, inverter-based switched-capacitor delta-sigma modulator," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 2, pp. 458–472, 2009, doi: 10.1109/JSSC.2008.2010973.

[7] B. Hershberg, S. Weaver, K. Sobue, S. Takeuchi, K. Hamashita, and U.-K. Moon, "Ring amplifiers for switched capacitor circuits," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 12, pp. 2928–2942, 2012, doi: 10.1109/JSSC.2012.2217865.

[8] S.-M. Yoo, J. S. Walling, E. C. Woo, B. Jann, and D. J. Allstot, "A switched-capacitor RF power amplifier," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 12, pp. 2977–2987, 2011, doi: 10.1109/JSSC.2011.2163469.

[9] M. Darvishi, R. van der Zee, and B. Nauta, "Design of active n-path filters," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 12, pp. 2962–2976, 2013, doi: 10.1109/JSSC.2013.2285852.

Oversampling and Sigma-Delta ADCs

22

Keywords: Quantization, OSR, NEG FB, STF, NTF, SAR, First Order, SC SD, CT SD, INCR, FOM

22.1 ADC state-of-the-art 385
22.1.1 What makes a state-of-the-art ADC . . . 387
22.1.2 High resolution FOM 394
22.2 Quantization 395
22.2.1 Signal to Quantization noise ratio . . . 399
22.2.2 Understanding quantization 400
22.2.3 Why you should care about quantization noise 403
22.3 Oversampling . . . 404
22.3.1 Noise power 404
22.3.2 Signal power 405
22.3.3 Signal to Noise Ratio 405
22.3.4 Signal to Quantization Noise Ratio . . 406
22.3.5 Python oversample 407
22.4 Noise Shaping . . . 408
22.4.1 The magic of feedback 409
22.4.2 Sigma-delta principle 410
22.4.3 Signal transfer function 412
22.4.4 Noise transfer function 412
22.4.5 Combined transfer function 412
22.5 First-Order Noise-Shaping 413
22.5.1 SQNR and ENOB . 415
22.6 Examples 416
22.6.1 Python noise-shaping 416
22.6.2 The wonderful world of SD modulators . 421
22.7 Summary 426
22.8 Would you like to know more? 427

22.1 ADC state-of-the-art

The performance of an analog-to-digital converter is determined by the effective number of bits (ENOB), the power consumption, and the maximum bandwidth. The effective number of bits contain information on the linearity of the ADC. The power consumption shows how efficient the ADC is. The maximum bandwidth limits what signals we can sample and reconstruct in digital domain.

Many years ago, Robert Walden did a study of ADCs, it is worth looking up the plots.

1999, R. Walden: Analog-to-digital converter survey and analysis [1]

There are obvious trends, the faster an ADC is, the less precise the ADC is (lower SNDR). There are also fundamental limits, Heisenberg tells us that a 20-bit 10 GS/s ADC is impossible, according to Walden.

The plot itself is Figure 4 of that paper, and it is worth looking up: effective bits against sample rate for every converter Walden could find, with four sloped lines drawn over the cloud for the limits set by thermal noise, aperture uncertainty, comparator ambiguity and, at the top, the "Heisenberg" bound. The shape of the cloud is the point. It runs down and to the right, so every decade of sample rate costs somewhere around a bit, and no converter sits above the lines.

Murmann's survey, two figures further on, plots the same thing with twenty-five more years of data, so you can see both what has changed and what has not.

The uncertainty principle states that the precision we can determine position and the momentum of a particle is

$$\sigma_x \sigma_p \geq \frac{\hbar}{2}$$

. There is a similar relation of energy and time, given by

$$\Delta E \Delta t > \frac{h}{2\pi}$$

where ΔE is the difference in energy, and Δt is the difference in time.

You should take these limits with a grain of salt. The plot assumes 50 Ohm and 1 V full-scale. As a result, the "Heisenberg" line that appears to be unbreakable certainly is breakable. Just change the voltage to 100 V, and the number of bits can be much higher. Always check the assumptions.

A more recent survey of ADCs comes from Boris Murmann. He still maintains a list of the best ADCs from ISSCC and VLSI Symposium.

B. Murmann, ADC Performance Survey 1997-2023

A common figure of merit for low-to-medium resolution ADCs is the Walden figure of merit, defined as

$$FOM_W = \frac{P}{2^{ENOB} f_s}$$

Below 10 fJ/conv.step is good.

Below 1 fJ/conv.step is extreme.

In the plot below you can see the ISSCC and VLSI ADCs.

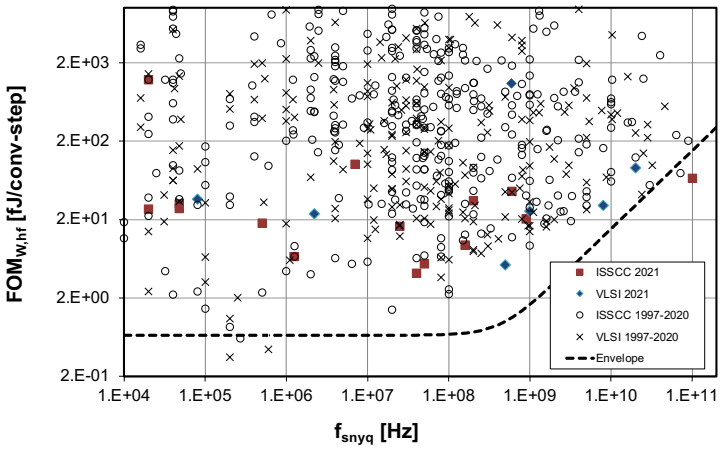


Figure 1: Murmann ADC survey: Walden figure of merit versus Nyquist sample rate for ISSCC and VLSI Symposium ADCs, with the best-in-class envelope [2]

22.1.1 What makes a state-of-the-art ADC

People from NTNU have made some of the world's best ADCs

If you ever want to make an ADC, and you want to publish the measurements, then you must be better than most. A good algorithm for state-of-the-art ADC design is to first pick a sample rate with low number of data (blank spaces in the plot above), then read the papers in the vicinity of the blank space to understand the application, then set a target FOM which is best in world, then try and find an ADC architecture that can achieve that FOM.

That's pretty much the algorithm I, and others, have followed to make state-of-the-art ADCs. A few of the NTNU ADCs are:

[1] A Compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for Bluetooth Low Energy Receivers [3]

[2] A 68 dB SNDR Compiled Noise-Shaping SAR ADC With On-Chip CDAC Calibration

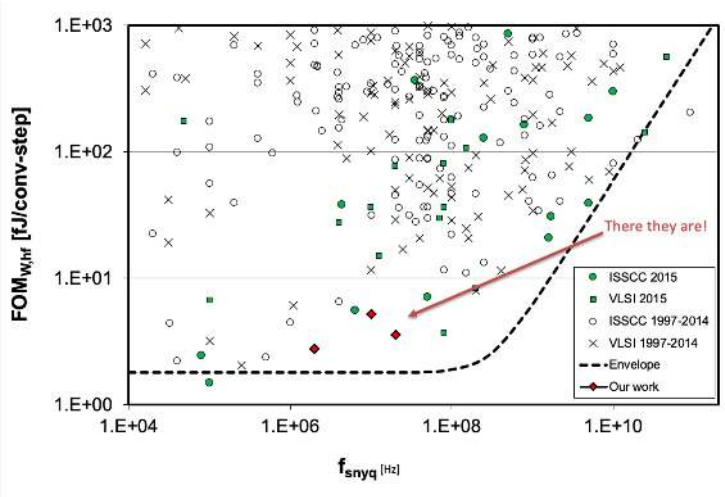


Figure 2: Walden figure of merit versus Nyquist sample rate with the NTNU compiled SAR ADCs marked as "Our work" near the envelope. Data: Murmann ADC survey [2]

In order to publish, there must be something new. Preferably a new circuit. Below is the circuit from [1]. It's a standard successive-approximation register (SAR) analog-to-digital converter.

The differential input signal is sampled on a capacitor array where the bottom plate is connected to either VSS or VREF. Once the voltage is sampled, the comparator will decide whether the differential voltage is larger, or smaller than 0. Depending on the decision, the MSB capacitors (left-most) in figure will switch the bottom plate in order to effectively subtract a voltage equivalent to half the VREF voltage.

The comparator makes another decision, and 1/4'th the VREF voltage is subtracted or added. Then 1/8'th and so on implementing a binary search to find the input voltage.

The "bit-cycling" (binary-search) loop is self-timed, as such, when the comparator has made a decision, the next cycle starts.

In (b) we can see the enable flip-flop for the next stage. The CK bar is the sample clock, as such, A is high during sampling. The output of the comparator (P and N) is low.

As soon as the comparator makes a decision, P or N goes high, A will be pulled low, if EI is enabled.

In (c) we can see that the bottom plate of the capacitors D_{P0}, D_{P1}, D_{N0}, and D_{N1}, are controlled by P and N.

In (d) we can see that the bottom plate of the capacitors also used to set the comparator clock low again (CO), resetting the comparator, and pulling P and N low, which in (b) enables the next SAR logic state.

How fast the D_{XX} settle depend on the size of the capacitors, as such, the comparator clock will be slow for the MSB, and very fast for the LSB. This was my main circuit contribution in the paper. I think it's quite clever, because both the VDD and the capacitor corner will change the settling time. It's important that the capacitor values fully settle before the next comparator decision, and as a result of the circuit in (c,d) the delay is automatically adjusted.

For further details see the paper.

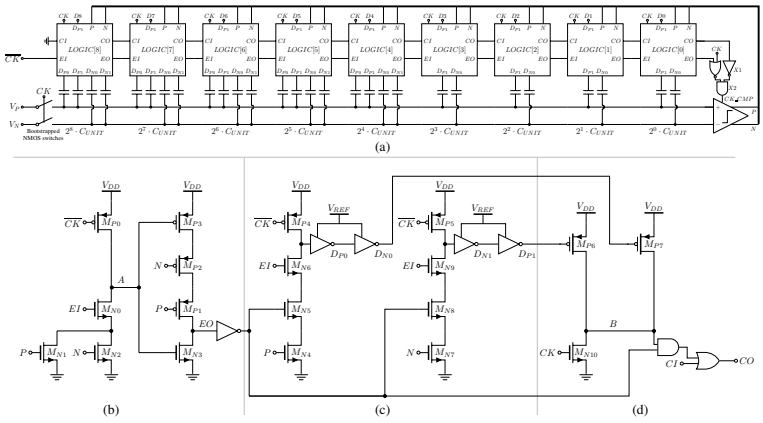


Figure 3: SAR ADC schematic: (a) capacitor array with self-timed SAR logic chain and comparator, (b) enable flip-flop, (c) bottom-plate switching of the CDAC, (d) comparator clock generation. © 2017 IEEE. Reprinted, with permission, from [3]

For state-of-the-art ADC papers it's not sufficient with the idea, and simulation. There must be proof that it actually works. No-one

will really believe that the ADC works until there is measurements of an actual taped out IC.

Below you can see the layout of the IC I made for the paper. Notice that there are 9 ADCs. I had many ideas that I wanted to try out, and I was not sure what would actually be state of the art. As a result, I taped out multiple ADCs.

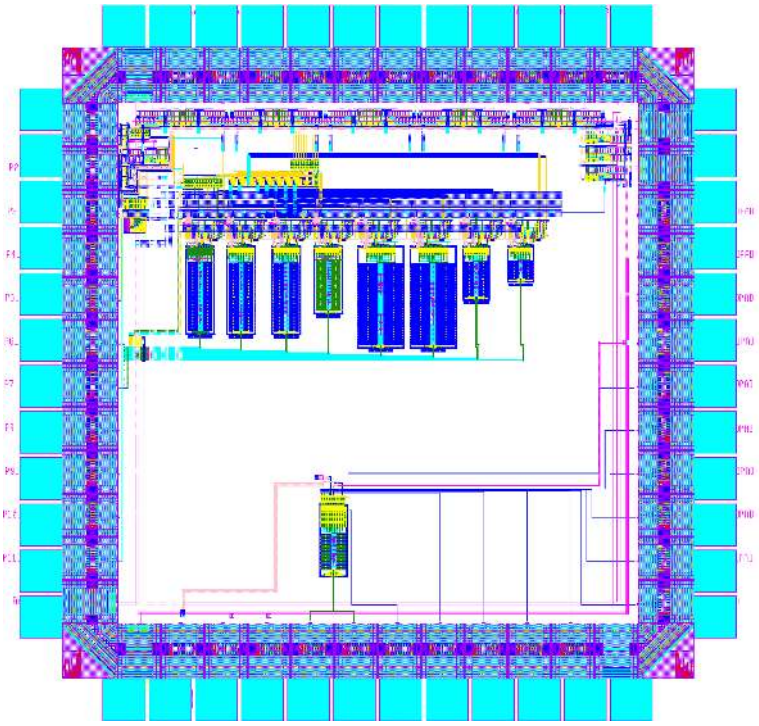


Figure 4: Layout of the test chip with nine ADC variants inside the pad ring. © 2017 IEEE. Reprinted, with permission, from [3]

The two ADCs that I ended up using in the paper are shown below. The one on the left was made with 180 nm IO transistors, while the one on the right was made with core-transistors. Notice that the layout of the two is quite similar.

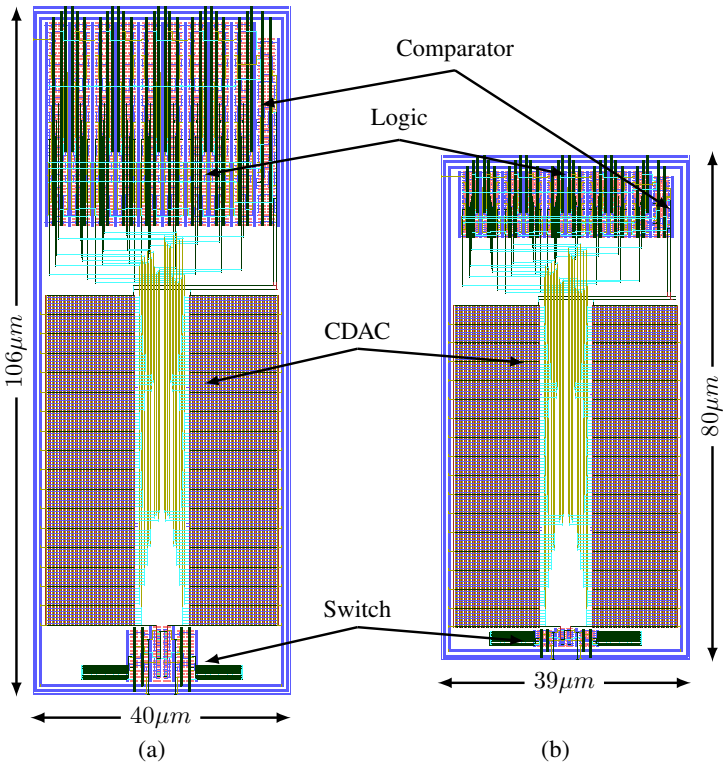


Figure 5: Layout of the two compiled SAR ADCs with comparator, logic, CDAC and switch: (a) 180 nm IO-transistor version (40 x 106 um), (b) core-transistor version (39 x 80 um). © 2017 IEEE. Reprinted, with permission, from [3]

Once taped out, and many months of waiting, a few months of measurement in the lab, I had some results that would be good enough to qualify for the best conference, and luckily the best journal.

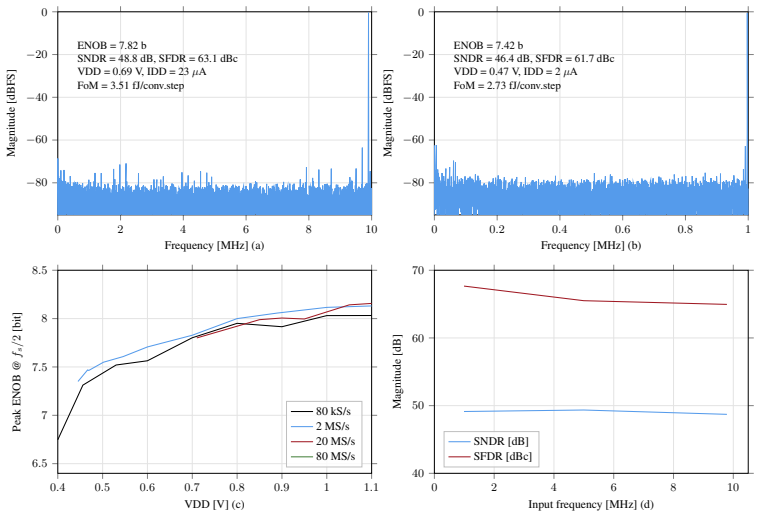


Figure 6: Measured performance of the core-transistor ADC: (a, b) output spectra at 0.69 V and 0.47 V supply, (c) peak ENOB versus VDD, (d) SNDR and SFDR versus input frequency

Comparing my ADCs to others, we can see that the FOM is similar to others. Based on the FOM it might not be clear why the paper was considered state-of-the-art.

The circuit technique mentioned above would not have been enough to qualify. The big thing was the "Compiled" line. Compared to the other "Compiled" mine was 300 times better, and on par with other state-of-the-art.

	Weaver [4]	Harpe [5]	Patil [6]	Liu [7]	This work	This work
Technology (nm)	90	90	28 FDSOI	28	28 FDSOI	28 FDSOI
Fsample (MS/s)	21	2	No sampling	100	2	20
Core area (mm^2)	0.18	0.047	0.0032	0.0047	0.00312	0.00312
SNDR (dB)	34.61	57.79	40	64.43	46.43	48.84
SFDR (dBc)	40.81	72.33	30	75.42	61.72	63.11
ENOB (bits)	5.45	6.7 - 9.4	6.35	10.41	7.42	7.82
Supply (V)	0.7	0.7	0.65	0.9	0.47	0.69
Pwr (μW)	1110	1.64 - 3.56	24	350	0.94	15.87
Compiled	Yes	No	No	No	Yes	Yes
FoM (fJ/c.step)	838	2.8 - 6.6	3.7	2.6	2.7	3.5

The big thing was how I made the ADC. I started with a definition of a transistor, as shown below

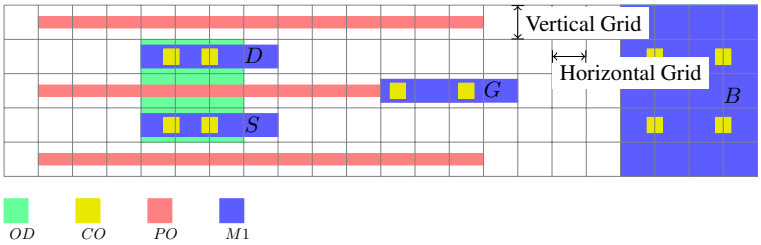


Figure 7: Transistor definition used by the layout compiler: diffusion (OD), contacts (CO), poly (PO) and metal 1 (M1) placed on a vertical and horizontal grid. © 2017 IEEE. Reprinted, with permission, from [3]

And then wrote a compiler (in Perl, later C++ ciccreator) to compile a object definition file, a SPICE netlist and a technology rule file into the full ADC layout.

In (a) you can see one of the cells in the SAR logic, (b) is the spice file, and (c) is the definition of the routing. The numbers to the right in the routing creates the paths shown in (d).

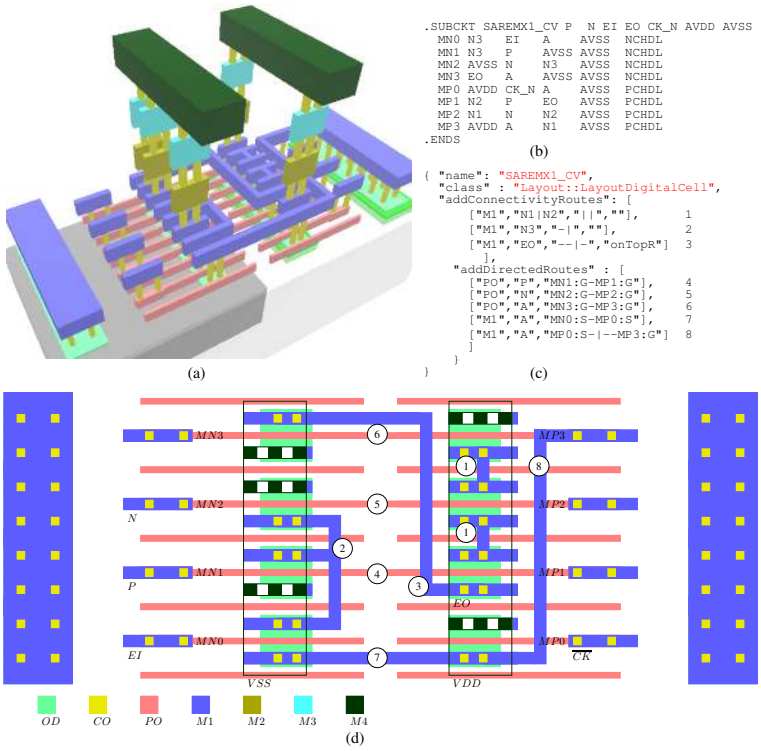


Figure 8: Compiled SAR logic cell: (a) 3D view of the layout, (b) SPICE netlist, (c) object definition with routing rules, (d) the resulting routed layout. © 2017 IEEE. Reprinted, with permission, from [3]

The implementation is the SPICE netlist, and the object definition file (JSON) and the rule file.

What I really like is the fact that the compilation could generate GDSII or SKILL, or these days, Xschem schematics and Magic layout.

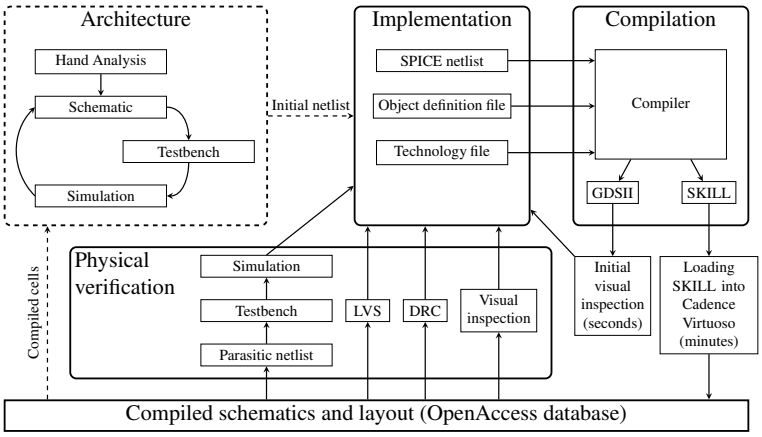


Figure 9: Compiled ADC design flow: architecture, implementation files (SPICE netlist, object definition, technology file), compilation to GDSII or SKILL, and physical verification. © 2017 IEEE. Reprinted, with permission, from [3]

The cool thing with a compiled ADC is that it's easy to port between technologies. Since the original ADC, I've ported the ADC to multiple closed PDKs (22 nm FDSOI, 22 nm, 28 nm, 55 nm, 65 nm and 130nm). In the summer of 2022 I made an open source port to skywater 130nm.

SUN_SAR9B_SKY130NM

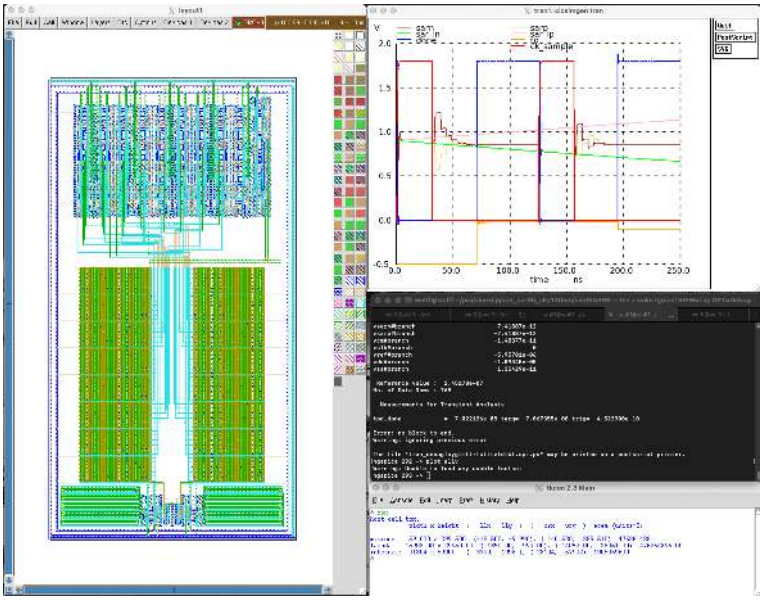


Figure 10: The SAR ADC ported to skywater 130 nm: Magic layout of the ADC core and ngspice transient simulation of a conversion

One of my Ph.D students built on-top on my work, and made a noise-shaped compiled SAR ADC, shown below, more on that later.

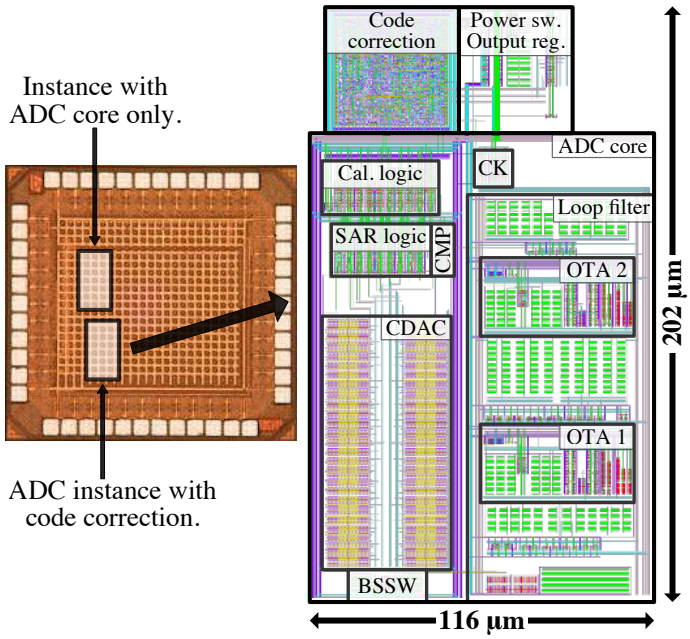


Figure 11: Noise-shaping compiled SAR ADC: die photo of the two ADC instances and the 116 um x 202 um core layout with CDAC, SAR logic, loop filter, OTAs and code correction. © 2019 IEEE. Reprinted, with permission, from [8]

22.1.2 High resolution FOM

For high-resolution ADCs, it's more common to use the Schreier figure of merit, which can also be found in

B. Murmann, ADC Performance Survey 1997-2022 (ISSCC & VLSI Symposium)

The Walden figure of merit assumes that thermal noise does not constrain the power consumption of the ADC, which is usually true for low-to-medium resolution ADCs. To keep the Walden FOM you can double the power for a one-bit increase in ENOB. If the ADC is limited by thermal noise, however, then you must quadruple the capacitance (reduce kT/C noise power) for each 1-bit ENOB increase. Accordingly, the power must also go up four times.

For higher resolution ADC the power consumption is set by thermal noise, and the Schreier FOM allows for a 4x power consumption increase for each added bit.

$$FOM_S = SNDR + 10\log\left(\frac{f_s/2}{P}\right)$$

Above 180 dB is extreme

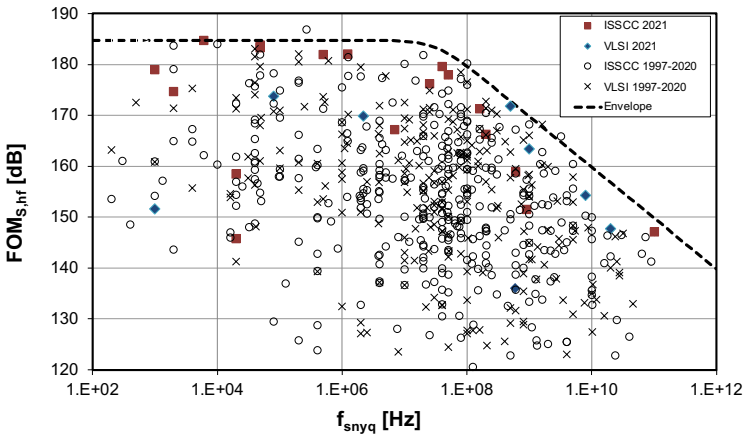


Figure 12: Murmann ADC survey: Schreier figure of merit versus Nyquist sample rate, where the envelope flattens around 185 dB for thermal-noise limited ADCs [2]

22.2 Quantization

Sampling turns continuous time into discrete time. Quantization turns continuous value into discrete value. Any complete ADC is always a combination of sampling and quantization.

In our mathematical drawings of quantization we often define $y[n]$ as the output, the quantized signal, and $x[n]$ as the discrete time, continuous value input, and we add some "noise", or "quantization noise" $e[n]$, where $x[n] = y[n] - e[n]$.

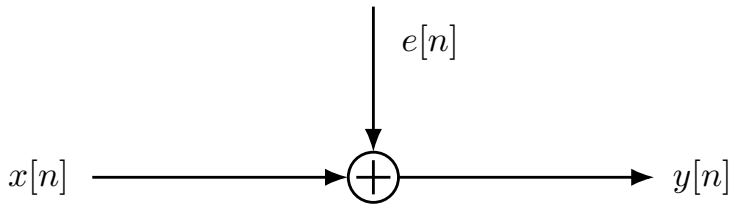


Figure 13: Linear model of quantization: the quantization noise e[n] is added to the input x[n] to form the output y[n]

Maybe you've even heard the phrase "Quantization noise is white" or "Quantization noise is a random Gaussian process"?

I'm here to tell you that you've been lied to. Quantization noise is not white, nor is it a Gaussian process. Those that have lied to you may say "yes, sure, but for high number of bits it can be considered white noise". I would say that's similar to saying "when you look at the earth from the moon, the surface looks pretty smooth without bumps, so let's say the earth is smooth with no mountains".

I would claim that it's an unnecessary simplification. It's obvious to most that the earth would appear smooth from really far away, but they would not be surprised by Mount Everest, since they know it's not smooth. An Alien that has been told that the earth is smooth, would be surprised to see Mount Everest.

But if Quantization noise is not white, what is it?

The figure below shows the input signal x and the quantized signal y.

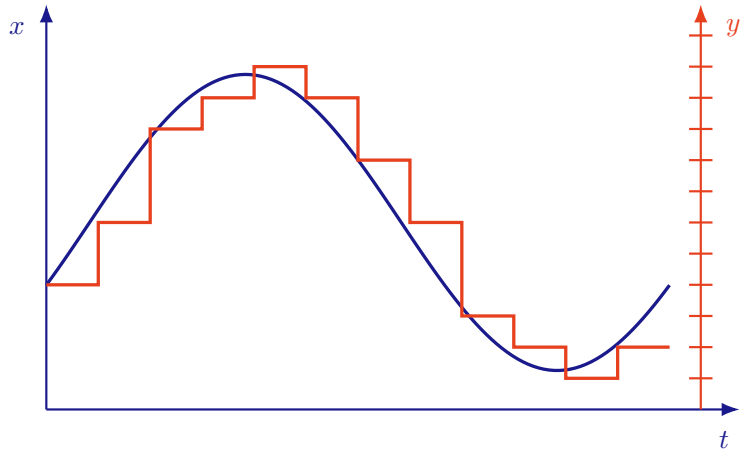


Figure 14: A continuous time sinusoid input x (blue) and the quantized output y (red)

To see the quantization noise, first take a look at the sample and held version of x in green in the figure below. The difference between the green (x at time n) and the red (y) would be our quantization noise e

The quantization noise is contained between $+\frac{1}{2}$ Least Significant Bit (LSB) and $-\frac{1}{2}$ LSB.

This noise does not look random to me, but I can't see what it is, and I'm pretty sure I would not be able to work it out either.

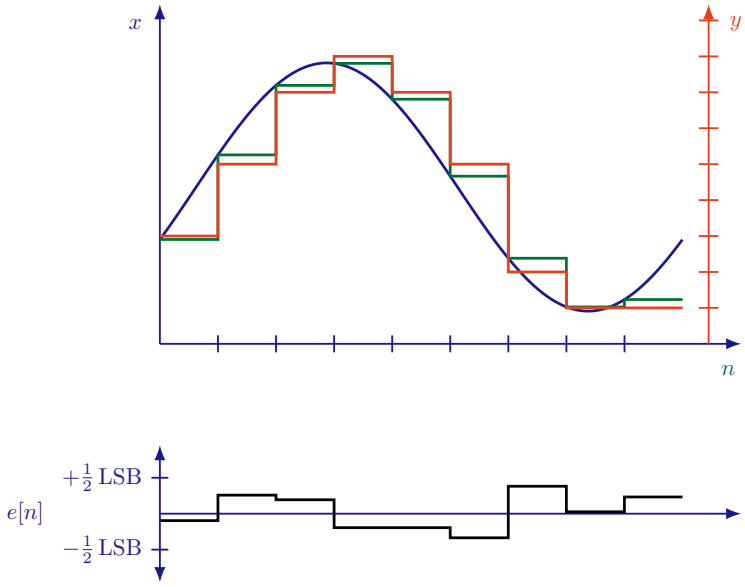


Figure 15: Sample-and-held input (green), quantized output (red), and the resulting quantization error e[n], bounded by plus/minus half an LSB

Luckily, there are people in this world that love mathematics, and that can delve into the details and figure out what $e[n]$ is. A

guy called Blachman wrote a paper back in 1985 on quantization noise.

See The intermodulation and distortion due to quantization of sinusoids [9] for details

In short, the quantizer *output* for a sinusoidal input is an exact harmonic series,

$$y(t) = \sum_{p=1}^{\infty} A_p \sin p\omega t$$

and the quantization noise is what is left when the input is taken back out,

$$e_n(t) = y(t) - A \sin \omega t$$

which is the same series with the $\delta_{p1}A$ term removed. It is worth keeping the two apart. The series below carries the fundamental at nearly full amplitude, so it is emphatically not something with zero mean and variance $\Delta^2/12$; the *difference* is.

where p is the harmonic index, and

$$A_p = \begin{cases} \delta_{p1}A + \sum_{m=1}^{\infty} \frac{2}{m\pi} J_p(2m\pi A) & , p = \text{ odd} \\ 0 & , p = \text{ even} \end{cases}$$

$$\delta_{p1} \begin{cases} 1 & , p = 1 \\ 0 & , p \neq 1 \end{cases}$$

and

$$J_p(x)$$

is a Bessel function of the first kind, A is the amplitude of the input signal.

If we approximate the amplitude of the input signal as

$$A = \frac{2^n - 1}{2} \approx 2^{n-1}$$

where n is the number of bits, we can rewrite as

$$y(t) = \sum_{p=1}^{\infty} A_p \sin p\omega t$$

$$A_p = \delta_{p1} 2^{n-1} + \sum_{m=1}^{\infty} \frac{2}{m\pi} J_p(2m\pi 2^{n-1}), p = odd$$

Obvious, right?

I must admit, it's not obvious to me. But I do understand the implications. The quantization noise is an infinite sum of input signal odd harmonics, where the amplitude of the harmonics is determined by a sum of a Bessel function.

A Bessel function of the first kind looks like this

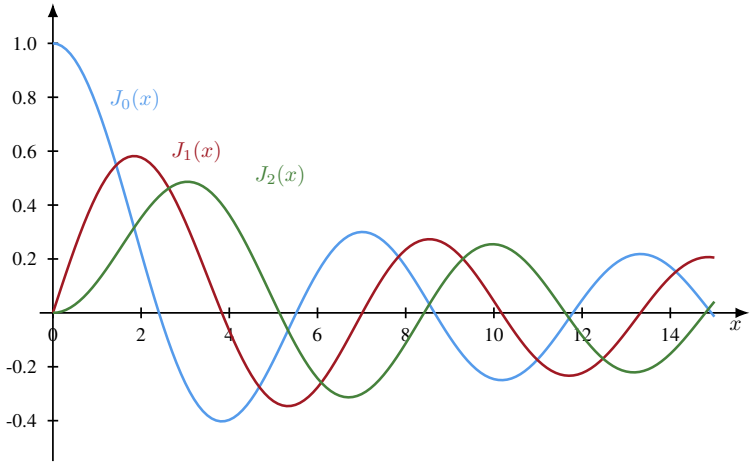


Figure 16: Bessel functions of the first kind, J0(x), J1(x) and J2(x), showing the oscillatory behavior that shapes the quantization noise harmonics

So I would expect the amplitude to show signs of oscillatory behavior for the harmonics. That's the important thing to remember. The quantization noise is **odd harmonics of the input signal**

The mean value is zero

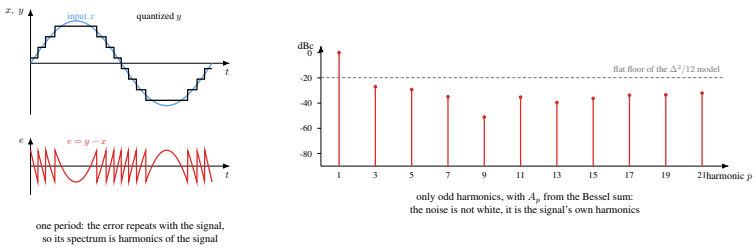


Figure 17: What the Bessel series actually says: a sine through a 3-bit quantizer, the error it leaves, and the odd harmonic amplitudes A_p from the formula above, against the flat floor the $\Delta^2/12$ model would predict

Here is the formula drawn out. On the left, a sine through a three bit quantizer, and underneath it the error it leaves behind. The error is not random: it repeats exactly once per signal period, which is the whole reason its spectrum can only contain harmonics of the signal.

On the right, the amplitudes A_p evaluated from the Bessel sum. They agree with a direct FFT of the quantized sine to about a tenth of a percent, so this is not an approximation of the noise, it *is* the noise. Compare the spikes with the dashed line, which is where the $\Delta^2/12$ white noise model would put a flat floor: the total power is the same, but the distribution is nothing like it.

If you want to feel this rather than read it, the interactive version puts the bit count on a slider. Walk it up and watch the harmonics fall and crowd together, until somewhere around eight bits calling them a noise floor finally becomes fair.

Watch the rate they fall at, because it is not the 6 dB per bit you might expect. An individual harmonic drops about **9 dB per bit**, tending to $30 \log 2 = 9.03$ dB. The 6 dB per bit belongs to the *total* quantization noise power, and the difference between the two is the whole reason the noise floor idea works at all: each harmonic falls 9 dB, but the number of harmonics crammed below $f_s/2$ doubles, which adds 3 dB back. Nine down, three up, six net. So as you add bits the spectrum does not merely shrink, it redistributes — fewer and fewer dB in each of more and more spikes, until no individual spike is worth naming and the aggregate is all that is left.

$$\overline{e_n(t)} = 0$$

and variance (mean square, since mean is zero), or noise power, can be approximated as

$$\overline{e_n(t)^2} = \frac{\Delta^2}{12}$$

22.2.1 Signal to Quantization noise ratio

Assume we wanted to figure out the resolution, or effective number of bits for an ADC limited by quantization noise. A power ratio, like signal-to-quantization noise ratio (SQNR) is one way to represent resolution.

Take the signal power, and divide by the noise power

$$SQNR = 10 \log \left(\frac{A^2/2}{\Delta^2/12} \right) = 10 \log \left(\frac{6A^2}{\Delta^2} \right)$$

$$\Delta = \frac{2A}{2^B}$$

$$SQNR = 10\log\left(\frac{6A^2}{4A^2/2^{2B}}\right) = 20B\log 2 + 10\log 6/4$$

$$SQNR \approx 6.02B + 1.76$$

You may have seen the last equation before, now you know where it comes from.

22.2.2 Understanding quantization

Below I've tried to visualize the quantization process q.py, which also exists as an interactive page where the number of bits is a slider and the SQNR is measured for you. Every spectrum in this chapter is computed the same way, by spectrum.py: a periodic Hann window, then the power averaged over 16 noise realisations and normalized to the largest bin.

The left most plot is a sinusoid signal and random Gaussian noise. The signal is not a continuous time signal, since that's not possible on a digital computer, but it's an approximation.

The plots are FFTs of a sinusoidal signal combined with noise. These are complex FFTs, so they show both negative and positive frequencies; the x-axis is normalized to each record's sample rate, from $-f_s/2$ to $+f_s/2$. When the text below talks about *bins*, multiply f/f_s by the record length (2048 after sampling) to get the bin number. Notice that there are two spikes, which should not be surprising, since a sinusoidal signal is a combination of two frequencies.

$$sin(x) = \frac{e^{ix} - e^{-ix}}{2i}$$

The second plot from the left is after sampling, notice that the noise level increases. The rise is exactly $10\log(4) = 6.02$ dB, and it is worth being careful about why, because there are two tempting explanations and they are not two effects to be added together.

Both are named in the same expression. With a Hann window and the peak normalisation that `freqDomain` uses, the noise floor per bin relative to the tone is

$$\text{floor} = \frac{6\sigma^2}{M}$$

where σ is the time-domain noise and M the record length. The record length is right there in the denominator, so a four times

shorter record does raise the floor 6 dB. But σ^2 is in the numerator, and whether it survives decimation is exactly the folding question. Decimating without an anti-alias filter preserves the total noise power, σ does not change, and the floor rises the full 6 dB. Put an ideal brick-wall filter in front and σ^2 drops by four as well, the two effects cancel term for term, and the floor does not move at all.

So the answer is 6.02 dB in total and the two mechanisms must not be added: they are two ways of booking the same rise, and the experiment that separates them is to filter before decimating. It is the first appearance in this chapter of a rule worth keeping — throwing samples away only helps if something band-limits the noise first.

The right plot is after quantization, where I've used the function below.

```
def adc(x,bits):
    levels = 2**bits
    delta = 2/levels
    y = np.floor(x/delta)*delta + delta/2
    return np.clip(y, -1 + delta/2, 1 - delta/2)
```

A B-bit converter has exactly 2^B levels, spaced by $\Delta = 2/2^B$, sitting half a step off zero. The obvious one-liner `np.round(x*2**bits)/2**bits` looks like a quantizer but is not one: its step is 2^{-B}, so at one bit it produces five output levels over ± 1 rather than two, and the measured SQNR comes out a whole bit too good. The plots below are a real one bit converter - two levels, one comparator.

I really need you to internalize a few things from the right most plot. Really think through what I'm about to say.

Can you see how the noise (what is not the two spikes) is not white? White noise would be flat in the frequency domain, but the noise is not flat.

Notice also that this is a genuine one bit quantizer: two levels, so the output is a square wave, and what you are looking at is its odd harmonic series - exactly the A_p of the Bessel formula above, folded back into the band wherever a harmonic lands above $f_s/2$.

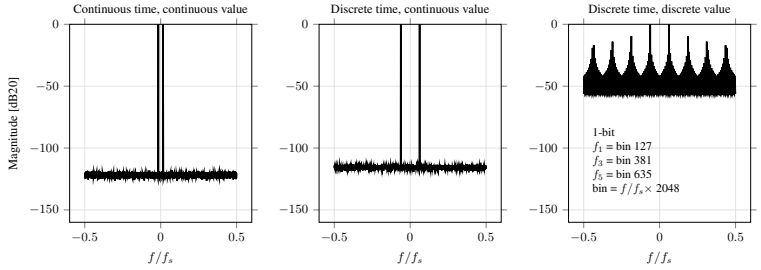


Figure 18: FFT of a sinusoid with noise as continuous value (left), after sampling (middle), and after 1-bit quantization (right), where the

quantization noise shows up as distinct harmonic spikes rather than a white noise floor

If you run the python script you can zoom in and check the highest spikes. A 1-bit quantizer is a sign detector, so its output for a sine input is a square wave, and a square wave has only odd harmonics with amplitudes falling as $1/p$. That is exactly what the plot shows. The fundamental is at bin 127, and the measured spikes are

harmonic	$20\log(1/p)$	measured	bin
3	−9.54 dB	−9.54 dB	381
5	−13.98 dB	−13.98 dB	635
7	−16.90 dB	−16.90 dB	889
9	−19.08 dB	−19.08 dB	**905**
11	−20.83 dB	−20.83 dB	**651**

Agreement to the second decimal with a formula you can write down from memory is a good afternoon's work. But look at the last two rows. The ninth harmonic should be at $9 \times 127 = 1143$ and the eleventh at $11 \times 127 = 1397$, and neither bin exists — the record is only 2048 points, so anything above 1024 is above half the sample rate. They fold: $2048 - 1143 = 905$ and $2048 - 1397 = 651$. The harmonics march monotonically down in amplitude, but from bin 889 onwards they march *backwards* along the frequency axis, which is why a spectrum plot of an aliased converter can look like nonsense until you work out which harmonic each spike really is.

This is worth internalising, because in a real converter you do not get to compare against a formula. A spike at 651 with nothing at 1397 is not evidence of some mysterious distortion mechanism; it is the eleventh harmonic, folded. If you change the python script to reduce the frequency, `fdivide=2**9`, and increase number of points, `N=2**16`, as in the plot below, the eleventh harmonic no longer folds, and you'll see it directly at bin 1397.

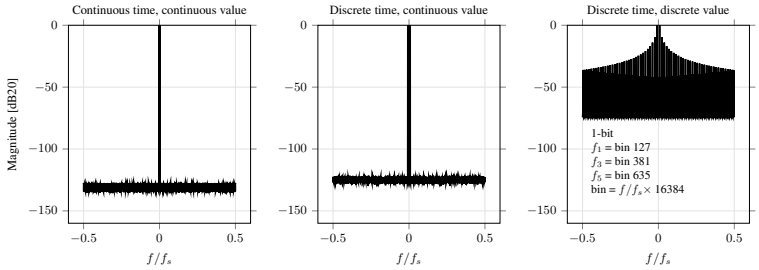


Figure 19: The same 1-bit quantization with lower input frequency and a 16384-point FFT, where the 11th harmonic appears directly at bin 1397 instead of folding

All the other spikes are the odd harmonics above the sample rate that fold. The infinite sum of harmonics will fold, some in-phase, some out of phase, depending on the sign of the Bessel function.

From the function for the amplitude of the quantization noise for harmonic indices higher than $p = 1$

$$A_p = \sum_{m=1}^{\infty} \frac{2}{m\pi} J_p(2m\pi 2^{n-1}), \text{ p=odd}$$

we can see that the input to the Bessel function increases faster for a higher number of bits n. As such, from the Bessel function figure above, I would expect that the sum of the Bessel function is a lower value. Accordingly, the quantization noise reduces at higher number of bits.

A consequence is that the quantization noise becomes more and more uniform, as can be seen from the plot of a 10-bit quantizer below. That's why people say "Quantization noise is white", because for a high number of bits, it looks white in the FFT.

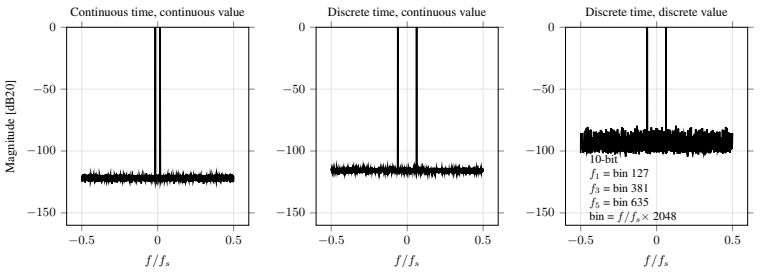


Figure 20: FFT of the same signal with a 10-bit quantizer, where the quantization noise is closer to uniform and looks almost white

22.2.3 Why you should care about quantization noise

So why should you care whether the quantization noise looks white, or actually is white? A class of ADCs called oversampling and sigma-delta modulators rely on the assumption that quantization noise **is** white. In other words, the cross-correlation between noise components at different time points is zero. As such the noise power sums as a sum of variance, and we can increase the signal-to-noise ratio.

We know that assumption to be wrong though, **quantization noise is not white**. For noise components at harmonic frequencies the cross-correlation will be high. As such, when **we** design oversampling or sigma-delta based ADC **we** will include some form of dithering (making quantization noise whiter). For example, before the actual quantizer we inject noise, or we make sure that the thermal noise is high enough to dither the quantizer.

Everybody that thinks that quantization noise **is** white will design non-functioning (or sub-optimal) oversampling and sigma-delta

ADCs. That's why you should care about the details around quantization noise.

22.3 Oversampling

Here is the question this section answers. If the quantizer's noise is a fixed amount that we cannot reduce, can we win anything by sampling faster than the signal actually demands, and then averaging the extra samples away?

The answer is yes, and the reason is that signal and noise behave differently under summation. A slow signal is nearly the same from one sample to the next, so summing samples adds it up coherently. Noise is not, so it adds up incoherently. Everything below is bookkeeping on that one asymmetry.

Assume a signal $x[n] = a[n] + b[n]$ where a is a sampled sinusoid and b is a random process where cross-correlation is zero for any time except for $n = 0$. Assume that we sum two (or more) equally spaced signal components, for example

$$y = x[n] + x[n+1]$$

What would the signal to noise ratio be for y?

22.3.1 Noise power

Our mathematician friends have looked at this, and as long the noise signal b **is random** then the noise power for the oversampled signal $b_{osr} = b[n] + b[n+1]$ will be

$$\overline{b_{osr}^2} = OSR \times \overline{b^2}$$

where OSR is the oversampling ratio. If we sum two time points the $OSR = 2$, if we sum 4 time points the $OSR = 4$ and so on.

For fun, let's go through the mathematics

Define $b_1 = b[n]$ and $b_2 = b[n+1]$ and compute the noise power

$$\overline{(b_1 + b_2)^2} = \overline{b_1^2 + 2b_1b_2 + b_2^2}$$

Let's replace the mean with the actual function

$$\frac{1}{N}\sum_{n=0}^{N}\left(b_1^2 + 2b_1b_2 + b_2^2\right)$$

which can be split up into

$$\frac{1}{N}\sum_{n=0}^{N} b_1^2 + \frac{1}{N}\sum_{n=0}^{N} 2b_1b_2 + \frac{1}{N}\sum_{n=0}^{N} b_2^2$$

we've defined the cross-correlation to be zero, as such

$$\overline{(b_1 + b_2)^2} = \frac{1}{N}\sum_{n=0}^{N} b_1^2 + \frac{1}{N}\sum_{n=0}^{N} b_2^2 = \overline{b_1^2} + \overline{b_2^2}$$

but the noise power of each of the b's must be the same as b, so

$$\overline{(b_1 + b_2)^2} = 2\overline{b^2}$$

22.3.2 Signal power

For the signal a we need to calculate the increase in signal power as OSR increases.

I like to think about it like this. a is low frequency, as such, samples n and $n+1$ is pretty much the same value. If the sinusoid has an amplitude of 1, then the amplitude would be 2 if we sum two samples. As such, the amplitude must increase with the OSR.

The signal power of a sinusoid is $A^2/2$, accordingly, the signal power of an oversampled signal must be $(OSR \times A)^2/2$.

22.3.3 Signal to Noise Ratio

Take the signal power to the noise power

$$\frac{(OSR \times A)^2/2}{OSR \times \overline{b^2}} = OSR \times \frac{A^2/2}{\overline{b^2}}$$

We can see that the signal to noise ratio increases with increased oversampling ratio, **as long as the cross-correlation of the noise is zero**

22.3.4 Signal to Quantization Noise Ratio

Now put the two halves together. The quantizer always makes the same total amount of noise, $\Delta^2/12$, and always spreads it evenly from zero to $f_s/2$. Sampling faster than we need does not reduce that total, it only spreads it over a wider band, so the part that lands inside the band we actually care about shrinks by the oversampling ratio.

in-band quantization noise for an oversampling ratio (OSR)

$$\overline{e_n(t)^2} = \frac{\Delta^2}{12OSR}$$

That single division by OSR is the whole of oversampling, and everything else in this section is arithmetic on it. Dividing the noise power by OSR multiplies the ratio by OSR:

$$SQNR = 10\log\left(\frac{6A^2}{\Delta^2/OSR}\right) = 10\log\left(\frac{6A^2}{\Delta^2}\right) + 10\log(OSR)$$

$$SQNR \approx 6.02B + 1.76 + 10\log(OSR)$$

so the SQNR improves by $10\log(2) \approx 3$ dB for an OSR of 2, and $10\log(4) \approx 6$ dB for an OSR of 4 — 3 dB for every doubling.

$$10\log(2) \approx 3dB$$

$$10\log(4) \approx 6dB$$

Compare that with the 6.02 dB a real bit is worth and the exchange rate falls out: three decibels is half a bit, so oversampling buys 0.5-bit per doubling of OSR

It is a poor exchange rate, and it is worth feeling how poor before moving on. Going from an 8-bit converter to a 12-bit one by oversampling alone needs $OSR = 2^8 = 256$, which means running the analog front end 256 times faster than the signal requires. That is the wall that noise shaping exists to get around.

22.3.5 Python oversample

There are probably more elegant (and faster) ways of implementing oversampling in python, but I like to write the dumbest code I can, simply because dumb code is easy to understand.

Below you can see an example of oversampling. The `oversample` function takes in a vector and the OSR. For each index it sums OSR future values.

```
def oversample(x,OSR):
    N = len(x)
    y = np.zeros(N)

    for n in range(0,N):
        for k in range(0,OSR):
            m = n+k
            if (m < N):
                y[n] += x[m]
    return y
```

Below we can see the plot for OSR=2, the right most plot is the oversampled version.

The noise has all frequencies, and it's the high frequency components that start to cancel each other. An average filter (sometimes called a sinc filter due to the shape in the frequency domain) will have zeros at $\pm fs/2$ where the noise power tends towards zero.

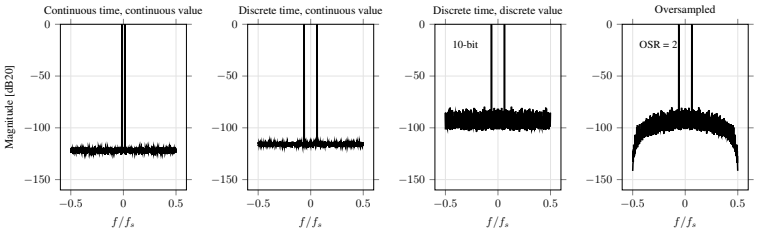


Figure 21: FFTs from continuous value to 10-bit quantized to oversampled with OSR=2 (right), where the averaging filter nulls the noise towards half the sample rate

The low frequency components will add, and we can notice how the noise power increases close to the zero frequency (middle of the x-axis).

For an OSR of 4 we can count four dips in the noise floor, although there are really three nulls: a length-4 moving average has zeros at $f_s/4$, $f_s/2$ and $3f_s/4$, and on this two-sided plot the $f_s/2$ null is split across both edges so you see half of it twice. For OSR=2 there is one null, at $f_s/2$, and it shows up only as the two edges.

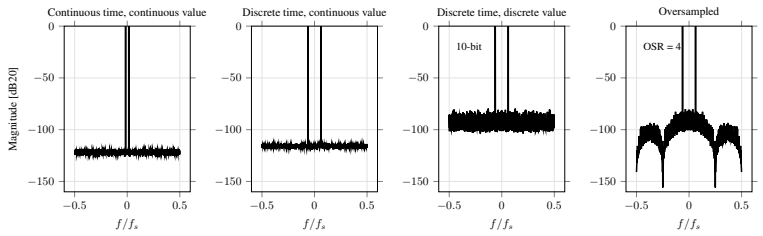


Figure 22: The same FFTs with OSR=4 (right), where the moving-average filter puts three nulls in the noise floor, seen as four dips on this two-sided plot, and the noise power increases close to zero frequency

The code for the plots is osr.py. I would encourage you to play a bit with the code, and make sure you understand oversampling. If you would rather drag a slider than edit a file, the interactive version plots the measured in-band SNR against OSR next to the ideal 3 dB per octave.

22.4 Noise Shaping

Look at the OSR=4 plot above, and look at it honestly, because it does not show what you might hope. Near zero frequency the averaged floor is not lower than the unaveraged one — measured from the script it is about 0.8 dB *higher* for OSR=4, which is the same thing the previous paragraph said when it noted that the low-frequency components add. The averaging filter has done almost nothing to the noise in the band we care about, because that noise is in its passband.

So where is the 6 dB that the theory promised? It is there, but it comes from *restricting the band*, not from the filter. Integrate the same unaveraged spectrum over $|f| < f_s/8$ instead of the whole $f_s/2$ and the signal-to-noise ratio improves by close to $10\log(4)$, exactly as the derivation said. The filter's job is not to create that improvement but to make it safe to collect: it removes the out-of-band noise that would otherwise fold back on top of the signal when you decimate. These plots never decimate, so the payoff is invisible in them. That is a limitation of the figure, not of oversampling.

What the plots do show clearly is the ceiling. Even with the averaging, the noise level of the discrete-time continuous-value plot is much lower than anything the quantized paths achieve.

What if we could do something, add some circuitry, before the quantization such that the quantization noise was reduced?

That's what noise shaping is all about. Adding circuits such that we can "shape" the quantization noise. We can't make the quantization noise disappear, or indeed reduce the total noise power of the

quantization noise, but we can reduce the quantization noise power for a certain frequency band.

But what circuitry can we add?

22.4.1 The magic of feedback

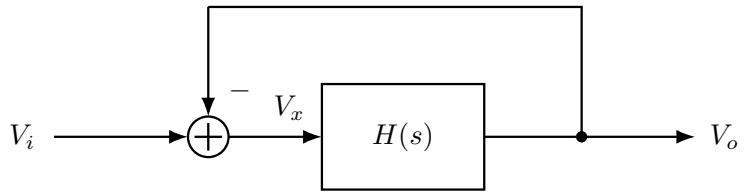
A generalized feedback system is shown below, it could be a regulator, a unity-gain buffer, or something else.

The output V_o is subtracted from the input V_i, and the error V_x is shaped by a filter $H(s)$.

If we make $H(s)$ infinite, then $V_o = V_i$. If you've never seen such a circuit, you might ask "Why would we do this? Could we not just use V_i directly?". There are many reasons for using a circuit like this, let me explain one instance.

Imagine we have a VDD of 1.8 V, and we want to make a 0.9 V voltage for a CPU. The CPU can consume up to 10 mA. One way to make a divide by two circuit is with two equal resistors connected between VDD and ground. We don't want the resistive divider to consume a large current, so let's choose 1 MOhm resistors. The current in the resistor divider would then be about 1 μA. We can't connect the CPU directly to the resistor divider, the CPU can draw 10 mA. As such, we need a copy of the voltage at the mid-point of the resistor divider that can drive 10 mA.

Do you see now why a circuit like the one below is useful? If not, you should really come talk to me so I can help you understand.



$$V_i - V_o = V_x \qquad V_o = V_x\, H(s)$$

$$V_i = V_o + \frac{V_o}{H(s)} \qquad H(s) = \infty \;\Rightarrow\; V_o = V_i$$

Figure 23: A generalized feedback system where the error between input and output is shaped by a filter H(s), and the output equals the input when H(s) is infinite

22.4.2 Sigma-delta principle

Let's modify the feedback circuit into the one below. I've added an ADC and a DAC to the feedback loop, and the D_o is now the output we're interested in. The equation for the loop would be

$$D_o = adc\left[H(s)\left(V_i - dac(D_o)\right)\right]$$

But how can we now calculate the transfer function $\frac{D_o}{V_i}$? Both adc and dac could be non-linear functions, so we can't disentangle the equation. Let's make assumptions.

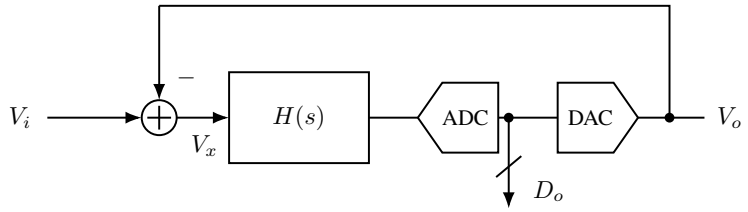


Figure 24: The sigma-delta principle: a feedback loop with a filter H(s), an ADC (quantizer) and a DAC in the feedback path, with digital output Do

22.4.2.1 The DAC assumption

Assumption 1: the dac is linear, such that $V_o = dac(D_o) = AD_o + B$, where A and B are scalar values.

The DAC must be linear, otherwise our noise-shaping ADC will not work.

One way to force linearity is to use a 1-bit DAC, which has only two points, so should be linear. For example

$$V_o = A \times D_o$$

, where $D_o \in \{0, 1\}$. Even a 1-bit DAC could be non-linear if A is time-variant, so $V_o[n] = A(t) \times D_o[n]$, this could happen if the reference voltage for the DAC changed with time.

I've made a couple noise shaping ADCs, and in the first one I made I screwed up the DAC. It turned out that the DAC current had a signal dependent component which lead to a non-linear behavior.

22.4.2.2 The ADC assumption

Assumption 2: the adc can be modeled as a linear function $D_o = adc(x) = x + e$, where e is **white noise source**

We've talked about this, the e is not white, especially for low-bit ADCs, so we usually have to add noise. Sometimes it's sufficient with thermal noise, but often it's necessary to add a random, or pseudo-random noise source at the input of the ADC.

22.4.2.3 The modified equation

With the assumptions we can change the equation into

$$D_o = adc\left[H(s)\left(V_i - dac(D_o)\right)\right] = H(s)\left(V_i - AD_o\right) + e$$

In noise-shaping texts it's common to write the above equation as

$$y = H(s)(u - y) + e$$

or in the sample domain

$$y[n] = e[n] + h * (u[n] - y[n])$$

which could be drawn in a signal flow graph as below.

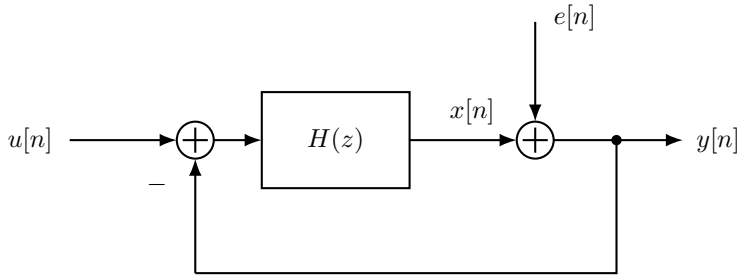


Figure 25: Signal flow graph of the noise-shaping loop: the difference between input u[n] and output y[n] is filtered by H(z) and the quantization noise e[n] is added at the quantizer

in the Z-domain the equation would turn into

$$Y(z) = E(z) + H(z)\left[U(z) - Y(z)\right]$$

The whole point of this exercise was to somehow shape the quantization noise, and we're almost at the point, but to show how it

works we need to look at the transfer function for the signal U and for the noise E.

22.4.3 Signal transfer function

Assume U and E are uncorrelated, and E is zero

$$Y = HU - HY$$

$$STF = \frac{Y}{U} = \frac{H}{1+H} = \frac{1}{1+\frac{1}{H}}$$

Imagine what will happen if H is infinite. Then the signal transfer function (STF) is 1, and the output Y is equal to our input U. That's exactly what we wanted from the feedback circuit.

22.4.4 Noise transfer function

Assume U is zero

$$Y = E - HY \rightarrow NTF = \frac{1}{1+H}$$

Imagine again what happens when H is infinite. In this case the noise-transfer function becomes zero. In other words, there is no added noise.

22.4.5 Combined transfer function

In the combined transfer function below, if we make $H(z)$ infinite, then $Y = U$ and there is **no added quantization noise**. I don't know how to make $H(z)$ infinite everywhere, so we have to choose at what frequencies it's "infinite".

$$Y(z) = STF(z)U(z) + NTF(z)E(z)$$

There are a large set of different $H(z)$ and I'm sure engineers will invent new ones. We usually classify the filters based on the number of zeros in the NTF, for example, first-order (one zero), second order (two zeros) etc. There are books written about sigma-delta modulators, and I would encourage you to read those to get a deeper understanding. I would start with Delta-Sigma Data Converters: Theory, Design, and Simulation.

22.5 First-Order Noise-Shaping

We want an infinite $H(z)$. One way to get an infinite function is an accumulator, for example

$$y[n+1] = x[n] + y[n]$$

or in the Z-domain

$$zY = X + Y \rightarrow Y(z-1) = X$$

which has the transfer function

$$H(z) = \frac{1}{z-1}$$

The signal transfer function is

$$STF = \frac{1/(z-1)}{1+1/(z-1)} = \frac{1}{z} = z^{-1}$$

and the noise transfer function

$$NTF = \frac{1}{1+1/(z-1)} = \frac{z-1}{z} = 1 - z^{-1}$$

In order calculate the Signal to Quantization Noise Ratio we need to have an expression for how the NTF above filters the quantization noise.

In the book they replace the z by the Laplace variable and then walk out onto the imaginary axis, $s = j\omega$, which is where a transfer function turns into a frequency response.

$$z = e^{sT} \underset{s=j\omega}{\longrightarrow} e^{j\omega T} = e^{j2\pi f/f_s}$$

inserted into the NTF we get the function below. The three lines are one trick applied once, so it is worth knowing what you are looking for before reading them. We want $1 - e^{-j\theta}$ turned into something whose magnitude we can read off, and the only identity available is $\sin\theta = (e^{j\theta} - e^{-j\theta})/2j$. So we factor out half the exponent, $e^{-j\pi f/f_s}$, which leaves a difference of two conjugate exponentials in the bracket — exactly the shape of that identity — and then divide and multiply by $2j$ to make it literally a sine. Everything pulled out has magnitude 1, so it contributes phase and nothing else.

$$\begin{aligned} NTF(f) &= 1 - e^{-j2\pi f/f_s} \\ &= \frac{e^{j\pi f/f_s} - e^{-j\pi f/f_s}}{2j} \times 2j \times e^{-j\pi f/f_s} \\ &= \sin\left(\frac{\pi f}{f_s}\right) \times 2j \times e^{-j\pi f/f_s} \end{aligned}$$

The arithmetic magic is really to extract the $2j \times e^{-j\pi f/f_s}$ from the first expression such that the initial part can be translated into a sinusoid.

When we take the absolute value to figure out how the NTF changes with frequency the complex parts disappears (equal to 1)

$$|NTF(f)| = \left|2\sin\left(\frac{\pi f}{f_s}\right)\right|$$

The signal power for a sinusoid is

$$P_s = A^2/2$$

The in-band noise power for the shaped quantization noise is

$$P_n = \int_{-f_0}^{f_0} \frac{\Delta^2}{12}\frac{1}{f_s}\left[2\sin\left(\frac{\pi f}{f_s}\right)\right]^2 df$$

The integral is not actually tedious, and it is worth doing once because it explains both of the odd-looking constants in the answer. In band, f is much smaller than f_s, so $\sin(\pi f/f_s) \approx \pi f/f_s$ and the integrand becomes a parabola:

$$P_n \approx \frac{\Delta^2}{12}\frac{1}{f_s}\frac{4\pi^2}{f_s^2}\int_{-f_0}^{f_0} f^2 df = \frac{\Delta^2}{12}\frac{4\pi^2}{f_s^3}\frac{2f_0^3}{3}$$

That f_0^3 is where the whole advantage comes from. Ordinary oversampling had the in-band noise falling as f_0; here it falls as f_0^3, because the shaping makes the noise density itself proportional to f^2. Substituting $OSR = f_s/2f_0$:

$$P_n \approx \frac{\Delta^2}{12}\frac{\pi^2}{3}\frac{1}{OSR^3}$$

The OSR^3 becomes the $30\log(OSR)$ — three times the $10\log(OSR)$ of plain oversampling — and the $\pi^2/3$ becomes the penalty $10\log(\pi^2/3) = 5.17$ dB. Take the ratio to $P_s = A^2/2$ and

$$SQNR = 6.02B + 1.76 - 5.17 + 30\log(OSR)$$

If we compare to pure oversampling, where the SQNR improves by $10\log(OSR)$, a first order sigma-delta improves by $30\log(OSR)$. That's a significant improvement.

22.5.1 SQNR and ENOB

Below is the signal-to-quantization noise ratio's for Nyquist up to second order sigma-delta. Read them as one family. Every line starts from the same $6.02B + 1.76$, and each step down the list buys a steeper dependence on OSR at the cost of a larger fixed penalty.

The second-order line follows from repeating the derivation above with $NTF = (1 - z^{-1})^2$, so the noise density goes as f^4 instead of f^2. The integral then produces OSR^5, hence $50\log(OSR)$, and $\pi^4/5$ in place of $\pi^2/3$, hence $10\log(\pi^4/5) = 12.9$ dB. The pattern continues: an L'th order modulator gives $(20L + 10)\log(OSR)$ and a penalty of $10\log(\pi^{2L}/(2L + 1))$.

The penalties are real, and at low OSR they can outweigh the steeper slope. Setting the first- and second-order expressions equal gives a crossover at $OSR \approx 2.4$: below that, second-order shaping is *worse* than first-order, because the extra 7.7 dB of penalty has not yet been earned back by the extra $20\log(OSR)$. Higher order is not automatically better, it is better *eventually*, and where "eventually" starts is a number you can compute before committing to an architecture.

$$SQNR_{nyquist} \approx 6.02B + 1.76$$

$$SQNR_{oversample} \approx 6.02B + 1.76 + 10\log(OSR)$$

$$SQNR_{\Sigma\Delta 1} \approx 6.02B + 1.76 - 5.17 + 30\log(OSR)$$

$$SQNR_{\Sigma\Delta 2} \approx 6.02B + 1.76 - 12.9 + 50\log(OSR)$$

We could compute an effective number of bits, as shown below.

$$ENOB = (SQNR - 1.76)/6.02$$

Assume 1-bit quantizer, what would be the maximum ENOB?

OSR	Oversampling	First-order	Second-order
4	2.0	3.1	3.9
64	4.0	9.1	13.9
1024	6.0	15.1	23.9

Table: ENOB of a 1-bit quantizer ($B = 1$), from the three expressions above.

Read down a column and you see what oversampling buys you; read across a row and you see what an architecture buys you. Plain oversampling is hopeless: a thousand-fold increase in sample rate turns one bit into six. First-order shaping turns the same thousand-fold into fifteen, and second-order into twenty-four. That is the entire argument for building the loop, and it is why a 1-bit modulator followed by a decimation filter is a sensible way to make a 20-bit converter, while a 1-bit oversampled ADC is not a way to make anything.

The second-order column also shows the crossover discussed above. At $OSR = 4$ second-order leads first-order by less than a bit, because the extra 7.7 dB penalty has barely been paid off; by $OSR = 1024$ it leads by nine.

22.6 Examples

22.6.1 Python noise-shaping

I want to demystify noise-shaping modulators. I think one way to do that is to show some code. You can find the code at sd_1st.py, and an interactive version that runs the same loop in your browser and decodes the bitstream back into the input.

Below we can see an excerpt. Again pretty stupid code, and I'm sure it's possible to make a faster version (for loops in python are notoriously slow).

For each sample in the input vector u I compute the input to the quantizer x, which is the sum of the previous input to the quantizer and the difference between the current input and the previous output y_{sd}.

The quantizer generates the next y_{sd} and I have the option to add dither.

Four details in the full script are not in this excerpt and matter if you want to reproduce the plots. The input is scaled to 0.7 of full scale, because a true 1-bit loop overloads if you drive it all the way — the integrator runs away and the output degenerates into a slow square wave. The first two samples are thrown away, because

the integrator starts at zero and needs a moment to settle. The dither has the same power as the quantization noise, $\Delta^2/12$, which is enough to break up idle tones without swamping the signal. And, like every spectrum in this chapter, the plotted spectra are power-averaged over 16 noise realisations: a single FFT of noise has bins that stand around 10 dB above the floor and look like spurs, while real tones are the same in every run and survive the averaging.

One inconsistency is worth flagging rather than hiding. This `quantize` uses a step of $2/(2^B - 1)$ for more than one bit, a mid-tread converter whose levels include zero, while the `adc` function earlier in this chapter uses $\Delta = 2/2^B$, a mid-rise converter with no zero level. Both are real converters and both appear in real silicon; they differ by half a step and by whether a zero input produces a zero output. The $\Delta^2/12$ result holds for either. Just do not mix the two definitions when you are comparing measured numbers against theory, which is exactly the sort of half-LSB discrepancy that costs an afternoon.

```
def quantize(v,bits):
    #- 2**bits levels reaching +/-1, so bits=1 is
    #- a genuine two-level quantizer
    levels = 2**bits
    if(levels == 2):
        return 1.0 if v >= 0 else -1.0
    step = 2/(levels-1)
    return float(np.clip(np.round(v/step)*step,-1,1))

# u is discrete time, continuous value input
M = len(u)
y_sd = np.zeros(M)
x = np.zeros(M)
for n in range(1,M):
    x[n] = x[n-1] + (u[n]-y_sd[n-1])
    y_sd[n] = quantize(x[n]
        + dither*np.random.randn()*(2/(2**bits-1))/np.sqrt(12),bits)
```

The right-most plot is the one with noise-shaping. We can observe that the noise seems to tend towards zero at zero frequency, as we would expect. The accumulator above would have an infinite gain at infinite time (it's the sum of all previous values), as such, the NTF goes towards zero at 0 frequency.

If we look at the noise we can also see the non-white quantization noise, which will degrade our performance. I hope by now, you've grown tired of me harping on the point that **quantization noise is not white**

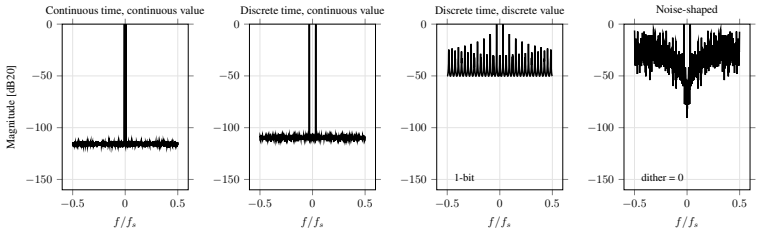


Figure 26: First-order sigma-delta modulator with 1-bit quantizer and no dither, where the noise-shaped spectrum (right) tends towards zero at zero frequency but contains distinct tones

In the figure below I've turned on dither, and we can see how the noise looks "better", which I know is not a qualitative statement, but ask anyone that's done 1-bit quantizers. It's important to have enough random noise.

Be clear about what has been bought and what has been paid, though, because the y-axis of the plot tells both halves of the story. The distinct tones are gone and the floor is smooth, which is what "better" means here: a smooth floor is predictable, it does not move when the input does, and it will not land a spur in the middle of your signal band on a Tuesday. But the floor is also *higher*, and the notch at zero frequency is shallower. Running the loop with and without dither, the in-band signal-to-noise ratio at an oversampling ratio of 8 drops by roughly 6 dB. Dither is not free noise reduction; it converts signal-to-noise ratio into predictability. That is usually a good trade for a 1-bit quantizer with a slow input, where the alternative is idle tones parked wherever the input DC level happens to put them, and a bad trade for a many-bit quantizer that was never going to produce tones in the first place.

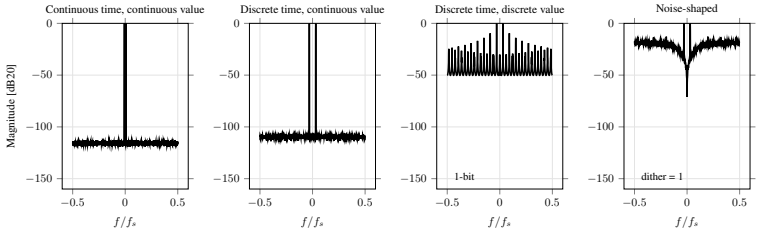


Figure 27: The same first-order 1-bit sigma-delta modulator with dither enabled, where the noise-shaped spectrum (right) is smoother and more noise-like

How much dither is enough? A natural yardstick is the quantization noise itself. If quantization noise were white it would have power $\Delta^2/12$, so I give the dither power relative to that, and dither = 1 in the script means the two are equal.

The figure below overlays the modulator output for no dither and for dither 12 dB below, equal to, and 12 dB above the quantization noise. Without dither the quantization noise is not white at all. It

is a comb of tones, some within 10 dB of the signal. At −12 dB the tones are still there. At equal power the largest is about 8 dB above the floor, and what is left is mostly the smooth first-order slope that the linear model predicts. At +12 dB the tones are gone, and the floor has risen across the whole band.

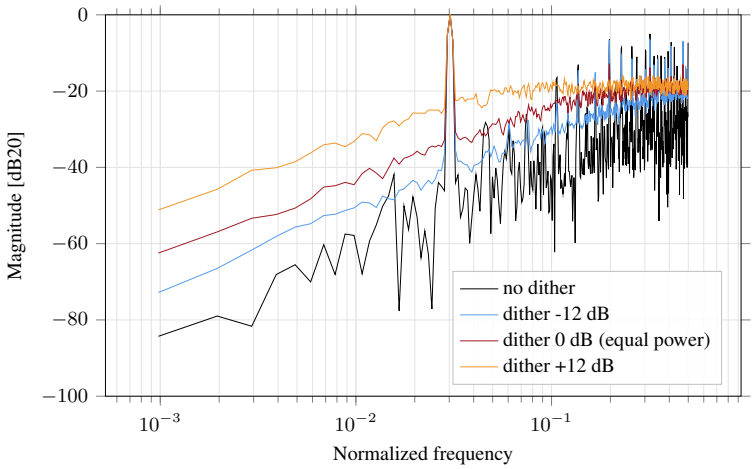


Figure 28: Output spectrum of the first-order 1-bit sigma-delta modulator for no dither and for dither power 12 dB below, equal to, and 12 dB above the quantization noise power $\Delta^2/12$. Without dither the noise is a comb of tones; with equal-power dither it is mostly the smooth first-order shaped floor

Whether dither is needed depends on the input. A large input drives the integrator through so many states that the loop dithers itself, and the tones stay small. A small input, or one that sits near a DC level, does not, and that is where idle tones land in the signal band.

The figure below shows it. It sweeps the input amplitude from −80 dBFS to full scale at an oversampling ratio of 1024, and measures the in-band signal-to-noise-and-distortion ratio (SNDR) and spurious-free dynamic range (SFDR). It uses a longer record than the spectra above, 262144 samples with the input at bin 37, so the band holds enough bins to measure in. The dashed line is the first-order estimate, $SQNR = 10\log_{10}\left(\frac{A^2/2}{\Delta^2/12}\right) - 5.17 + 30\log_{10}(OSR)$ for a sine of amplitude A. I've not used the familiar $6.02B+1.76$, because it assumes 2^B steps across full scale. A 1-bit quantizer with outputs ± 1 has one step, $\Delta = 2$, and the familiar form is 6 dB optimistic for it.

Without dither the modulator is close to the estimate near full scale, and falls further below it as the input shrinks. At −40 dBFS it is 11 dB short. Below about −68 dBFS the output is a limit cycle that does not see the input at all. With dither at the power of the quantization noise the SNDR follows the estimate about 4 dB below it at every level up to −8 dBFS, and below −40 dBFS the SFDR is

18 to 35 dB better than without. Dither at −12 dB is not enough: it helps, but erratically.

Why 4 dB, and not 3? Equal-power dither doubles the noise at the quantizer, which costs 3 dB, and a multi-bit quantizer loses about that: I measure 4 dB for a 5-bit loop. A 1-bit quantizer loses more, because it has no fixed gain. Its effective gain depends on what reaches its input, about 1.4 without dither and 0.85 with it at −12 dBFS, and in a first-order loop the in-band noise rises as the gain falls. Measured at −12 dBFS, dither costs the 1-bit loop 6 dB.

Above about −25 dBFS dither costs SNDR, which is the price. Below it, dither is what makes the converter work.

The linear model of the quantizer, an added white noise source, is only true once something makes it true. Dither is that something, and it is not free.

Pictures by Carsten, Written by model, Approved by Carsten

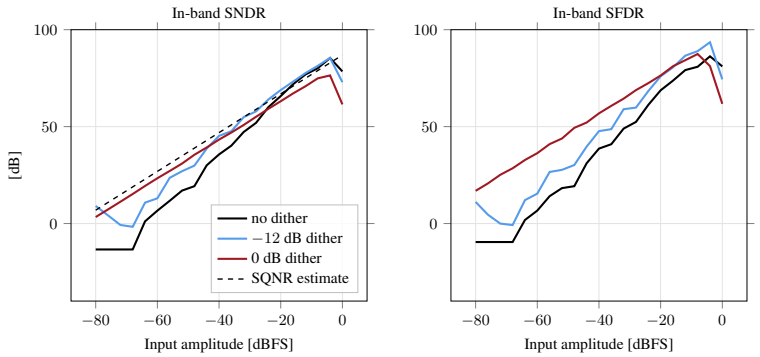


Figure 29: In-band SNDR and SFDR of the first-order 1-bit sigma-delta modulator at OSR = 1024 against input amplitude, without dither and with dither power 12 dB below and equal to the quantization noise power $\Delta^2/12$. The dashed line is the first-order SQNR estimate

In papers it's common to use a logarithmic frequency axis, as shown below. In the plot I only show the positive frequencies of the FFT. From the shape of the quantization noise we can also see the first order behavior, as a straight 20 dB per decade rise, which is much easier to read off a log axis than off the linear ones above.

Two things have changed from the 1-bit figures above, and it is worth saying so rather than letting you wonder. The quantizer here is 5-bit, not 1-bit, so the floor is far lower and there are no idle tones to speak of. And the y-axis is a magnitude spectrum in dB, not a power spectral density: nothing has been normalised to the bin width, so the absolute level is only meaningful relative to the tone.

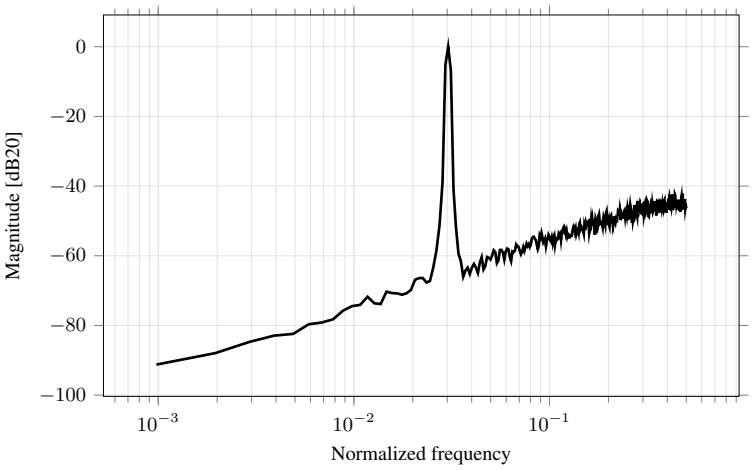


Figure 30: Magnitude spectrum of the output of a 5-bit first-order sigma-delta modulator on a logarithmic frequency axis, showing the 20 dB/decade shaping of the quantization noise

22.6.2 The wonderful world of SD modulators

22.6.2.1 Open-Loop Sigma-Delta

On my Ph.D I did some work on

Resonators in Open-Loop Sigma-Delta Modulators [10]

which was a pure theoretical work. The idea was to use modulo integrators (local control of integrator output swing) in front of large latency multi-bit quantizers to achieve a high SNR.

The plot below shows a fifth order NTF where there are two complex conjugate zero *pairs*, and a zero at zero frequency. With a higher order filter one can use a lower OSR, and still achieve high ENOB.

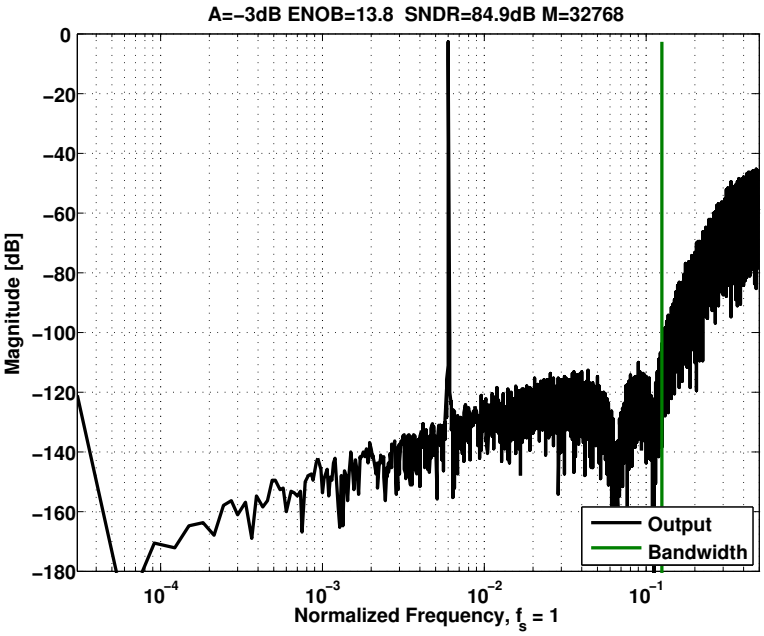


Figure 31: Output spectrum of an open-loop sigma-delta modulator with a fifth-order NTF (two complex conjugate zero pairs and a zero at DC), reaching 13.8 bit ENOB and 84.9 dB SNDR

22.6.2.2 Noise Shaped SAR

One of my Ph.d students made a

A 68 dB SNDR Compiled Noise-Shaping SAR ADC With On-Chip CDAC Calibration [8]

In a SAR ADC, once the bit-cycling is complete, the analog value on the capacitors is the actual quantization error. That error can be fed to a loop filter, H(z), and amplified in the next conversion, accordingly a combination of SAR and noise-shaping.

In the paper the SD modulator was also used to calibrate the non-linearity in the CDAC, as the MSB capacitor won't be exactly N times larger than the smallest capacitor.

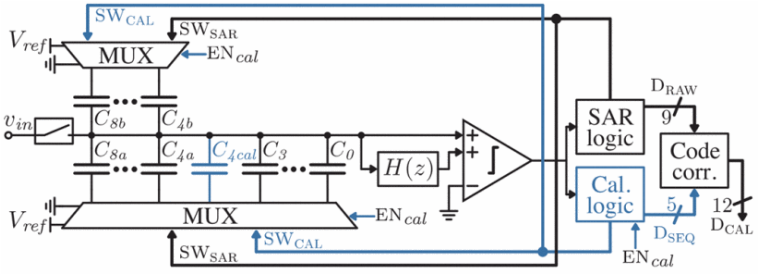


Figure 32: Architecture of the noise-shaping SAR ADC: capacitive DAC with multiplexers, loop filter H(z), integrating comparator, SAR logic, calibration logic and code correction. © 2019 IEEE. Reprinted, with permission, from [8]

The loop filter was a switched cap loop filter, and we can see the NTF below. The first OTA made use of chopping to reduce the offset.

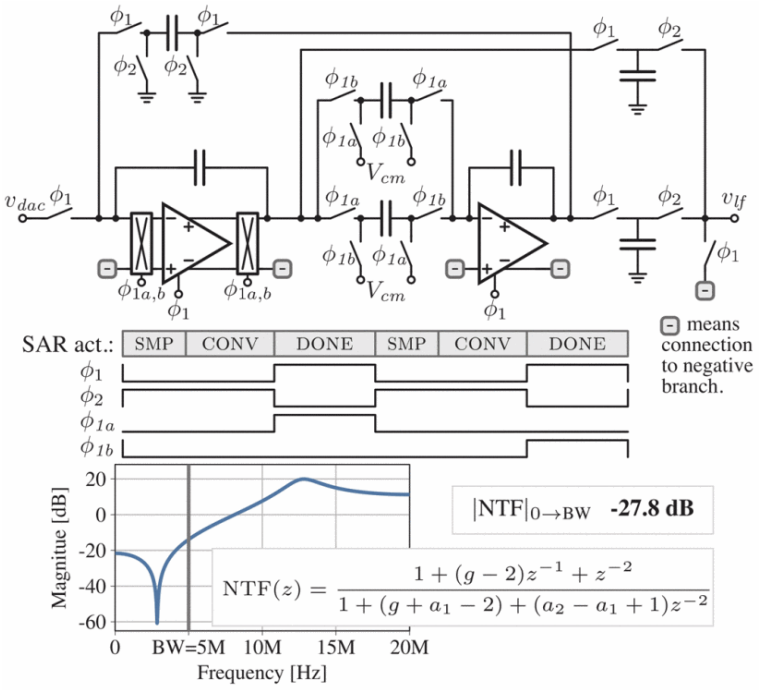


Figure 33: The switched-capacitor loop filter with two OTAs (the first one chopped), the clock phases relative to the SAR activity, and the resulting NTF with -27.8 dB in-band suppression. © 2019 IEEE. Reprinted, with permission, from [8]

22.6.2.3 Control-Bounded ADCs

One of my current Ph.D students is working on an even more advanced type of sigma-delta ADC. Actually, it's more a super-set of SD ADCs called control-bounded ADCs.

Design Considerations for a Low-Power Control-Bounded A/D Converter

A block diagram of a Leapfrog ADC version of a control-bounded ADC is shown below.

Here we're walking into advanced maths territory, but to simplify, I think it's correct to say that a control-bounded ADC seeks to control the local analog state, $x_n(t)$ such that no voltage is saturated. The digital control signals $s_n(t)$ are used to infer the state of the input $u(t)$ using a form of Bayesian Statistics.

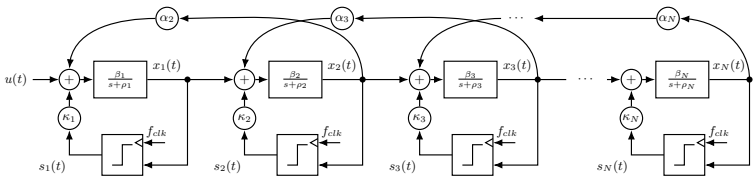


Figure 34: Block diagram of the Leapfrog control-bounded ADC: a chain of continuous-time integrators with local digital control loops s(t) that keep the analog states x(t) bounded. Redrawn from F. Feyling's master's thesis [11]

Below we can see a power spectral density plot of a third order Leapfrog ADC, simulated with cbadc, and we can observe how the quantization noise is shaped. The NTF has a zero at zero frequency and a complex conjugate zero pair, a notch, near 6 MHz.

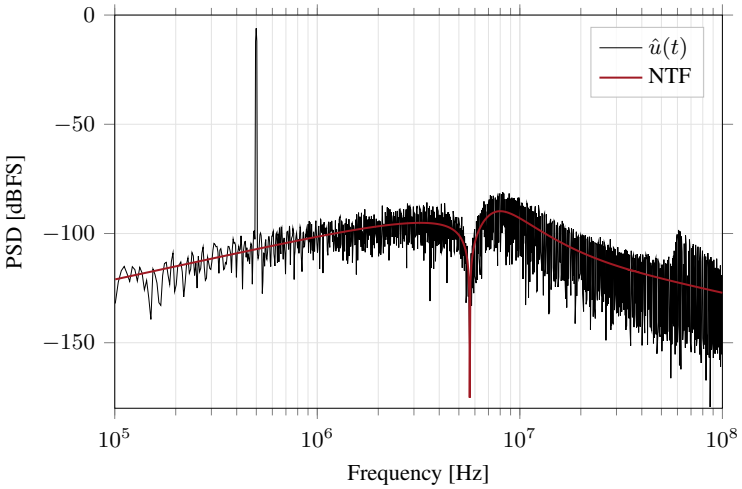


Figure 35: Power spectral density of the estimated input of a third order Leapfrog control-bounded ADC with a 500 kHz tone, together with the NTF shifted onto the noise floor, a third-order shaping with a notch near 6 MHz. Simulated with cbadc (`ex/leapfrog_sim.py`), following F. Feyling's master's thesis [11]

22.6.2.4 Complex Sigma-Delta

There are cool sigma-delta modulators with crazy configurations and that may look like an exercise in "Let's make something complex", however, most of them have a reasonable application. One example is the one below for radio receivers

A 56 mW Continuous-Time Quadrature Cascaded Sigma-Delta Modulator With 77 dB DR in a Near Zero-IF 20 MHz Band

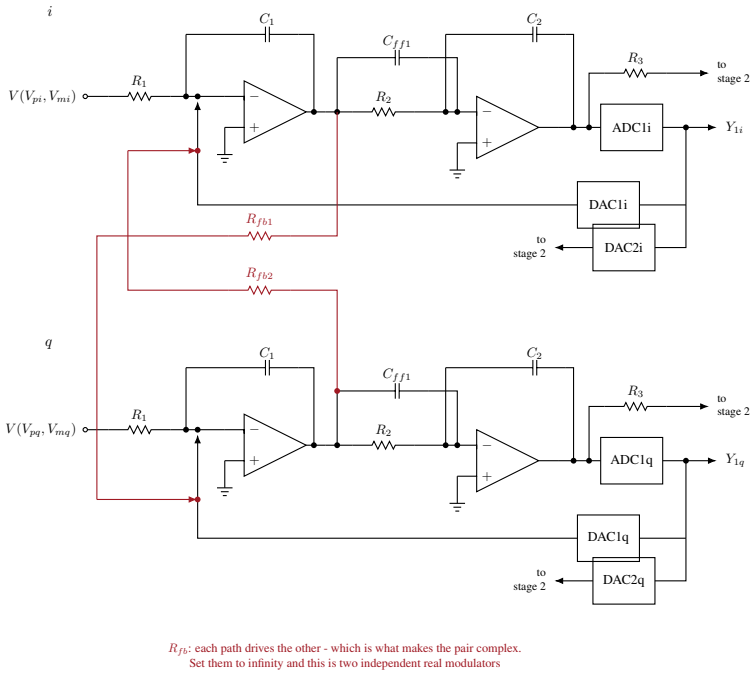


Figure 36: One stage of the quadrature modulator, with the second cascaded stage and the differential wiring left out. Inspired by Breems et al. [12]

The cross-coupling in red is the whole of what makes it quadrature. Take it away and there are two independent real modulators, each with a noise transfer function that is symmetric about zero frequency, because a real circuit cannot tell $+f$ from $-f$. Put it back and the pair responds to a rotating phasor rather than an oscillating one, so the notch can sit on one side of zero and not the other. That is exactly what a near zero-IF receiver wants: the wanted channel is a little above zero, its image a little below, and only a complex modulator can quieten one without quietening both.

22.6.2.5 My first Sigma-Delta

The first sigma-delta modulator I made in "real-life" was similar to the one shown below.

The input voltage is translated into a current, and the current is integrated on capacitor C. The R_{offset} is to change the mid-level voltage, while R_{ref} is the 1-bit feedback DAC. The comparator is the quantizer. When the clock strikes the comparator compares the V_o and $V_{ref}/2$ and outputs a 1-bit digital output D

The complete ADC is operated in a "incremental mode", which is a fancy way of saying

Reset your sigma-delta modulator, run the sigma delta modulator for a fixed number of cycles (i.e 1024), and count the number of ones at *D*

The effect of an "incremental mode" is to combine the modulator and a output filter so the ADC appears to be a slow Nyquist ADC.

For more information, ask me, or see the patent at Analogue-to-digital converter

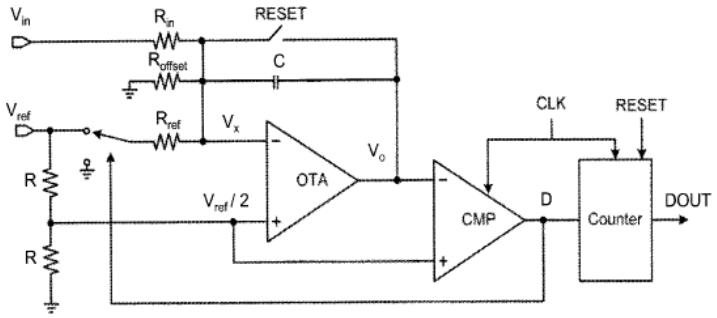


Figure 37: Incremental first-order sigma-delta ADC from the patent: input and reference resistors into an OTA integrating on C, a clocked comparator as quantizer, and a counter as output filter

22.7 Summary

The one-page version of this chapter:

- One number for an ADC: the figure of merit - Walden for speed-limited, Schreier for noise-limited designs
- Ideal quantization gives SQNR = 6.02 N + 1.76 dB, and the white-noise model of it holds only for busy inputs
- Oversampling spreads the same noise power over more bandwidth: 3 dB (half a bit) per octave
- Feedback around the quantizer shapes the noise away from the band: first-order sigma-delta buys 9 dB per octave, and higher order buys more
- The decimation filter is where the promised resolution is actually cashed out
- A compiled ADC is a netlist, an object file and a rule file - portable across processes in weeks, and good enough for JSSC

22.8 Would you like to know more?

The design of sigma-delta modulation analog-to-digital converters [13]

Delta-sigma modulation in fractional-N frequency synthesis [14]

A CMOS Temperature Sensor With a Voltage-Calibrated Inaccuracy of ± 0.15 C (3sigma) From -55 Cto 125 C [15]

A 20-mW 640-MHz CMOS Continuous-Time Sigma-Delta ADC With 20-MHz Signal Bandwidth, 80-dB Dynamic Range and 12-bit ENOB [16]

A Micro-Power Two-Step Incremental Analog-to-Digital Converter [17]

[1] R. H. Walden, "Analog to digital converter survey and analysis," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 539–550, 1999, doi: 10.1109/49.761034.

[2] B. Murmann, "ADC Performance Survey 1997-2016," 2016. [Online]. Available: http://web.stanford.edu/~murmann/adcsurvey.html. [Accessed: Jul. 12, 2016]

[3] C. Wulff and T. Ytterdal, "A compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for bluetooth low energy receivers," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 7, pp. 1915–1926, 2017, doi: 10.1109/JSSC.2017.2685463.

[4] S. Weaver, B. Hershberg, and U. K. Moon, "Digitally Synthesized Stochastic Flash ADC Using Only Standard Digital Cells," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 1, pp. 84–91, Jan. 2014, doi: 10.1109/TCSI.2013.2268571.

[5] P. Harpe, G. Dolmans, K. Philips, and H. de Groot, "A 0.7V 7-to-10bit 0-to-2MS/s flexible SAR ADC for ultra low-power wireless sensor nodes," in *Proc. 38th Eur. Solid-State Circuits Conf. (ESSCIRC)*, 2012, pp. 373–376, doi: 10.1109/ESSCIRC.2012.6341363.

[6] S. Patil, A. Ratiu, D. Morche, and Y. Tsividis, "A 3-10 fJ/conv-step Error-Shaping Alias-Free Continuous-Time ADC," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 908–918, 2016, doi: 10.1109/JSSC.2016.2519396.

[7] C. C. Liu, "A 0.35mW 12b 100MS/s SAR-assisted digital slope ADC in 28nm CMOS," in *IEEE ISSCC dig. Tech. papers*, 2016, pp. 462–463, doi: 10.1109/ISSCC.2016.7418107.

[8] H. Garvik, C. Wulff, and T. Ytterdal, "A 68 dB SNDR compiled noise-shaping SAR ADC with on-chip CDAC calibration," in *2019 IEEE asian solid-state circuits conference (a-SSCC)*, 2019, pp. 193–194, doi: 10.1109/A-SSCC47793.2019.9056925.

[9] N. Blachman, "The intermodulation and distortion due to quantization of sinusoids," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 6, pp. 1417–1426, 1985, doi: 10.1109/TASSP.1985.1164729.

[10] C. Wulff and T. Ytterdal, "Resonators in open-loop sigma–delta modulators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 10, pp. 2159–2172, 2009, doi: 10.1109/TCSI.2009.2015211.

[11] F. E. Feyling, "Design Considerations for a Low-Power Control-Bounded A/D Converter," Master's thesis, Norwegian University of Science; Technology (NTNU), 2021 [Online]. Available: https://hdl.handle.net/11250/2824253

[12] L. J. Breems, R. Rutten, R. H. M. van Veldhoven, and G. van der Weide, "A 56 mW continuous-time quadrature cascaded ΣΔ modulator with 77 dB DR in a near zero-IF 20 MHz band," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2696–2705, 2007, doi: 10.1109/JSSC.2007.908765.

[13] B. Boser and B. Wooley, "The design of sigma-delta modulation analog-to-digital converters," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 6, pp. 1298–1308, 1988, doi: 10.1109/4.90025.

[14] T. Riley, M. Copeland, and T. Kwasniewski, "Delta-sigma modulation in fractional-n frequency synthesis," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 5, pp. 553–559, 1993, doi: 10.1109/4.229400.

[15] K. Souri, Y. Chae, and K. A. A. Makinwa, "A CMOS temperature sensor with a voltage-calibrated inaccuracy of ± 0.15° c (3σ) from − 55° c to 125° c," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 292–301, 2013, doi: 10.1109/JSSC.2012.2214831.

[16] G. Mitteregger, C. Ebner, S. Mechnig, T. Blon, C. Holuigue, and E. Romani, "A 20-mW 640-MHz CMOS continuous-time ΣΔ ADC with 20-MHz signal bandwidth, 80-dB dynamic range and 12-bit ENOB," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 12, pp. 2641–2649, 2006, doi: 10.1109/JSSC.2006.884332.

[17] C.-H. Chen, Y. Zhang, T. He, P. Y. Chiang, and G. C. Temes, "A micro-power two-step incremental analog-to-digital converter," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 8, pp. 1796–1808, 2015, doi: 10.1109/JSSC.2015.2413842.

Voltage regulation 23

23.1 **Voltage source . . . 429**
23.1.1 Why transistors die 430
23.1.2 Core voltage 432
23.1.3 IO voltage 433
23.1.4 Supply planning . . 433
23.2 **Linear Regulators . 434**
23.2.1 PMOS pass-fet . . . 434
23.2.2 NMOS pass-fet . . . 435
23.2.3 Control of pass-fet . 436
23.3 **Switched Regulators 438**
23.3.1 Principles of switched regulators 438
23.3.2 Inductive DC/DC converter details . . 441
23.3.3 Pulse width modulation (PWM) 443
23.3.4 Real world use . . . 446
23.3.5 Pulsed Frequency Mode (PFM) 446
23.4 **Summary 449**
23.5 **Would you like to know more? 449**
23.5.1 Linear regulators . . 449
23.5.2 DC-DC converters . 449

Keywords: Battery, Vreg, LDOP, LDON, Flipped voltage follower, Buck, Boost, Load, Line, PSRR, MAX C, Quiescent, Settling, Efficiency, PWM, PFM

23.1 Voltage source

Most, if not all, integrated circuits need a supply and ground to work.

Assume a system is AC powered. Then there will be a switched regulator to turn wall AC into DC. The DC might be 48 V, 24 V, 12 V, 5 V, 3 V 1.8 V, 1.0 V, 0.8 V, or who knows. The voltage depends on the type of IC and the application.

Many ICs are battery operated, whether it's your phone, watch, heart rate monitor, mouse, keyboard, game controller or car.

For batteries the voltage is determined by the difference in Fermi level on the two electrodes, and the Fermi level (chemical potential) is a function of the battery chemistry. As a result, we need to know the battery chemistry in order to know the voltage.

Linden's Handbook of Batteries is a good book if you want to dive deep into primary (non-chargeable) or secondary (chargeable) batteries and their voltage curves.

Some common voltage sources are listed below.

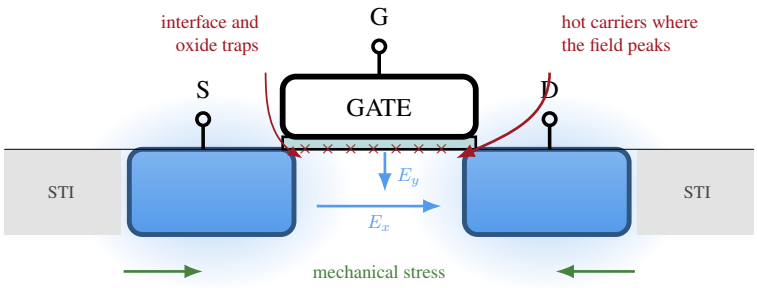
	Chemistry	Voltage [V]
Primary Cell	LiFeS2 , Zn/Alk/MnO2 , LiMnO2	0.8 - 3.6
Secondary Cell	Li-Ion	2.5 - 4.3
USB	-	4.0 - 6.5 (20)

The battery determines the voltage of the "electron source", however, can't we just run everything directly off the battery? Why do we need DC to DC converters or voltage regulators?

Turns out, transistors can die.

Today's transistor, as shown below, are a complicated three dimensional structure. Dimensions are measured in nano-meter, which makes the transistors fragile.

In Analog Circuit Design in Nanoscale CMOS Technologies [1] Lanny explains how to design around some of the breakdown effects.



Four families of effect, each with its own corner of the device model, and each taken separately in the sections that follow. Below about 100 nm every one of them is first order.

Figure 1: A nanoscale NMOS in cross-section. Mechanical stress from the isolation trenches, the transverse and lateral fields in the channel, traps at the oxide interface, and hot carriers where the field peaks near the drain

Two of those four are the ones that kill it. The transverse field E_y is what the gate oxide has to withstand, and the traps are the damage accumulating in it; the hot carriers at the drain end are the other. Neither is a failure you see immediately, which is what makes them dangerous - the device works, and then some months later it does not.

The transistors in a particular technology (from GlobalFoundries, TSMC, Samsung or others) have a maximum voltage that they can survive for a certain time. Exceed that time, or voltage, and the transistors die.

23.1.1 Why transistors die

A gate oxide will break due to Time Dependent Dielectric Breakdown (TDDB) if the voltage across the gate oxide is too large. Silicon oxide can break down at approximately 5 MV/cm. The breakdown forms a conductive channel from the gate to the channel and is permanent. After breakdown there will be a resistor of kOhms between gate and channel.

A similar breakdown phenomena is used in Metal-Oxide RRAM [2] and the SkyWater ReRAM

Below is an example of ReRAM. In the Pristine state the conductance is low, resistance is in the hundreds of mega Ohm. In a transistor we want the oxide to stay high resistive. In ReRAM, however, we apply a high voltage across the oxide, which forms a conductive channel across the oxide. Turns out, that the conductive channel can be flipped back and forth between a high resistive

state, and a low resistive state to store a 1 or a 0 in a non-volatile manner.

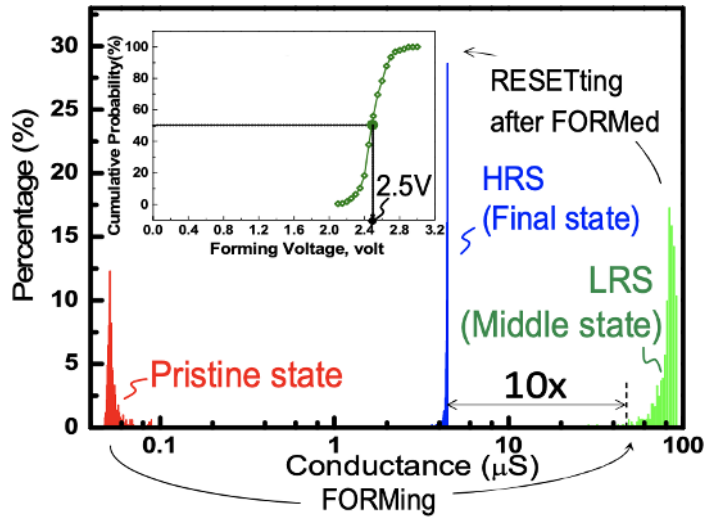


Figure 2: ReRAM conductance distributions in the pristine, formed (LRS), and reset (HRS) states. From google/skywater-pdk-libs-sky130_fd_pr_reram, Apache 2.0 license

The threshold voltage of a transistor can shift excessively over time caused by Hot-Carrier Injection (HCI) or Negative Bias Temperature Instability.

Hot-Carrier injection is caused by electrons, or holes, accelerated to high velocity in the channel, or drain depletion region , causing impact ionization (breaking a co-valent bond releasing an electron/hole pair). At a high drain/source field, and medium gate/(source or drain) field, the channel minority carriers can be accelerated to high energy and transition to traps in the oxide, shifting the threshold voltage.

Negative Bias Temperature Instability is a shift in threshold voltage due to a physical change in the oxide. A strong electric field across the oxide for a long time can break co-valent, or ionic bonds, in the oxide. The bond break will change the forces (stress) in the amorphous silicon oxide which might not recover. As such, there might be more traps (states) than before. See Simultaneous Extraction of Recoverable and Permanent Components Contributing to Bias-Temperature Instability [3] for more details.

For a long time, I had trouble with "traps in the oxide". I had a hard time visualizing how electrons wandered down the channel and got caught in the oxide. I was trying to imagine the electric field, and that the electron needed to find a positive charge in the oxide to cancel. Diving a bit deeper into quantum mechanics, my mental image improved a bit, so I'll try to give you a more accurate mental model for how to think about traps.

Quantum mechanics tells us that bound electrons can only occupy fixed states. The probability of finding an electron in a state is given by the Fermi function, but if there is no energy state at a point in space, there cannot be an electron there.

For example, there might be a 50 % probability of finding an electron in the oxide, but if there is no state there, then there will not be any electron , and thus no change to the threshold voltage.

What happens when we make "traps", through TDDB, HCI, or NBTI is that we create new states that can potentially be occupied by electrons. For example one, or more, broken silicon co-valent bonds and a dislocation of the crystal lattice.

If the Fermi-Dirac statistics tells us the probability of an electron being in those new states is 50 %, then there will likely be electrons there.

The threshold voltage is defined as the voltage at which we can invert the channel, or create the same density of electrons in the channel (for NMOS) as density of dopant atoms (density of holes) in the bulk.

If the oxide has a net negative charge (because of electrons in new states), then we have to pull harder (higher gate voltage) to establish the channel. As a result, the threshold voltage increases with electrons stuck in the oxide.

In quantum mechanics the time evolution, and the complex probability amplitude of an electron changing state, could, in theory, be computed with the Schrodinger equation. Unfortunately, for any real scenario, like the gate oxide of a transistor, using Schrodinger to compute exactly what will happen is beyond the capability of the largest supercomputers.

23.1.2 Core voltage

The voltage where the transistor can survive is estimated by the foundry, by approximation, and testing, and may be like the table below.

Node [nm]	Voltage [V]
180	1.8
130	1.5
55	1.2
22	0.8

23.1.3 IO voltage

Most ICs talk to other ICs, and they have a voltage for the general purpose input/output. The voltage reduction in I/O voltage does not need to scale as fast as the core voltage, because foundries have thicker oxide transistors that can survive the voltage.

Voltage [V]
5.0
3.0
1.8
1.2

23.1.4 Supply planning

For any IC, we must know the application. We must know where the voltage comes from, the IO voltage, the core voltage, and any other requirements (like charging batteries).

One example could be an IC that is powered from a Li-Ion battery, with a USB to provide charging capability.

Between each voltage we need an analog block, a regulator, to reduce the voltage in an effective manner. What type of regulator depends again on the application, but the architecture of the analog design would be either a linear regulator, or a switched regulator.

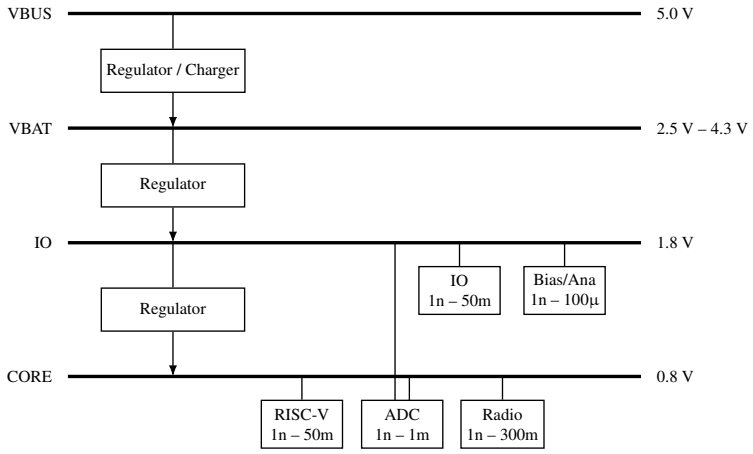


Figure 3: Example supply planning from VBUS and VBAT down to IO and core voltage, with a regulator between each domain

The dynamic range of the power consumed by an IC can be large. From nA when it's not doing anything, to hundreds of mA when there is high computation load.

As a result, it's not necessarily possible, or effective, to have one regulator from 1.8 V to 0.8 V. We may need multiple regulators.

Some that can handle low load (nA - μA) effectively, and some that can handle high loads.

For example, if you design a regulator to deliver 500 mA to the load, and the regulator uses 5 mA, that's only 1 % of the current, which may be OK. The same regulator might consume 5 mA even though the load is 1 uA, which would be bad. All the current flows in the regulator at low loads.

Name	Voltage	Min [nA]	Max [mA]	PWR DR [dB]
VDD_VBUS	5	10	500	77
VDD_VBAT	4	10	400	76
VDD_IO	1.8	10	50	67
VDD_CORE	0.8	10	350	75

Most product specifications will give you a view into what type of regulators there are on an IC. The figure below is redrawn from the nRF5340 product specification.

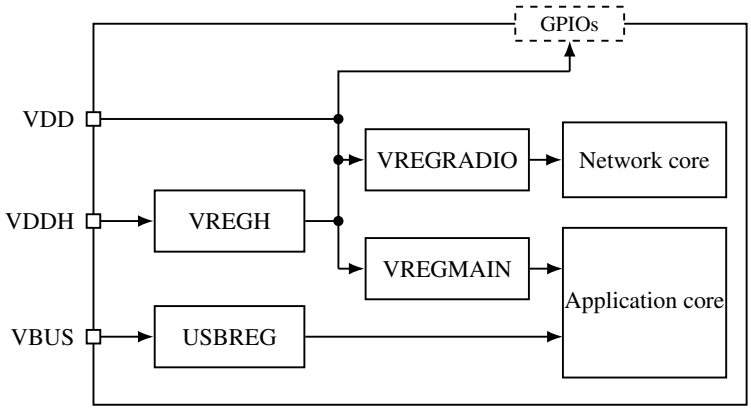


Figure 4: Regulators in the nRF5340: VREGH takes VDDH down onto the VDD rail, which feeds the GPIOs, VREGRADIO for the network core and VREGMAIN for the application core, while USBREG runs from VBUS. Redrawn from Nordic Semiconductor, nRF5340 Product Specification

23.2 Linear Regulators

23.2.1 PMOS pass-fet

One way to make a regulator is to control the current in a PMOS with a feedback loop, as shown below. The OTA continuously adjusts the gate-source voltage of the PMOS to force the input voltages of the OTA to be equal.

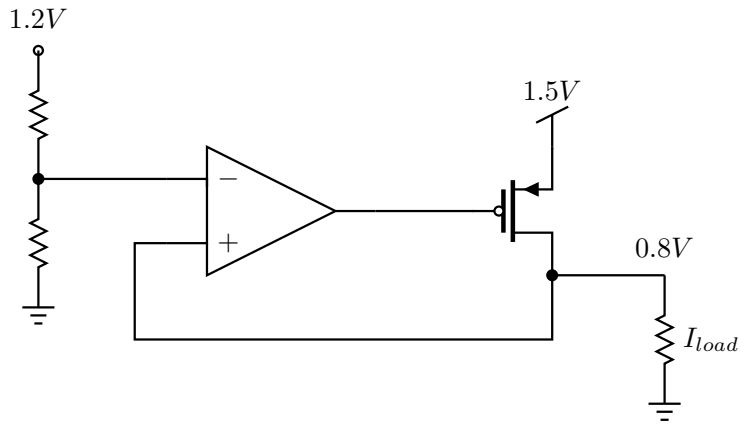


Figure 5: Linear regulator with a PMOS pass-fet controlled by an OTA feedback loop

For digital loads, where I_{load} is a digital current, with high current every rising edge of the clock, it's an option to place a large external decoupling capacitor (a reservoir of charge) in parallel with the load. Accordingly, the OTA would supply the average current.

The device between supply (1.5 V) and output voltage (0.8 V) is often called a pass-fet. A PMOS pass-fet regulator is often called a LDO, or low dropout regulator, since we only need a V_{DSSAT} across the PMOS, which can be a few hundred mV.

Key parameters of regulators are

Parameter	Description	Unit
Load regulation	How much does the output voltage change with load current	V/A
Line regulation	How much does the output voltage change with input voltage	V/V
Power supply rejection ratio	What is the transfer function from input voltage to output voltage? The PSRR at DC is the line regulation	dB
Max current	How much current can be delivered through the pass-fet?	A
Quiescent current	What is the current used by the regulator	A
Settling time	How fast does the output voltage settle at a current step	s

A disadvantage of a PMOS is the hole mobility, which is lower than for NMOS. If the maximum current of an LDO is large, then the PMOS can be big. Maybe even 50 % of the IC area.

23.2.2 NMOS pass-fet

An NMOS pass-fet will be smaller than a PMOS for large loads. The disadvantage with an NMOS is the gate-source voltage needed. For some scenarios the needed gate voltage might exceed the input voltage (1.5 V). A gate voltage above input voltage is possible, but increases complexity, as a charge pump (switched capacitor regulator) is needed to make the gate voltage.

Another interesting phenomena with NMOS pass-fet is that the PSRR is usually better, but we do have a common gate amplifier, as such, high frequency voltage ripple on output voltage will be amplified to the input voltage, and may cause issues for others using the input voltage.

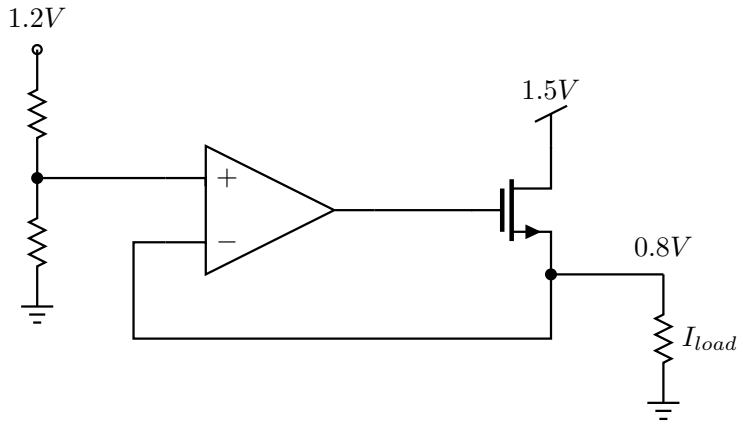


Figure 6: Linear regulator with an NMOS pass-fet controlled by an OTA feedback loop

23.2.3 Control of pass-fet

The large dynamic range in power management systems can make it challenging to have a single pass-fet.

The size of the pass-fet is set by the maximum Vgs, and the current that needs to be delivered.

Assume we need 500 mA from the LDO. If we assume that the maximum Vgs is 1.5 V, then we can simulate to try and find a size.

I've made a testbench at

Testbench for LDO pass-fet

Below is an excerpt from the testbench. The pass-fet size has been determined by iteration.

The OTA in the LDO is modeled by the B source. Notice the use of the tanh function in order to keep the G voltage within the rails.

```
* Pass-fet
XM1 OUT G VDD VDD sky130_fd_pr__pfet_01v8
+ L=0.252 W=11.52 nf=2 ...  m=1000

* Reference
VREF VREF 0 dc 0.8

* OTA
BOTA G 0 V=(1 + tanh(-1000*(v(vref) -v(out) )))/2*{AVDD}

* Load cap
```

```
CL OUT 0 1u

* Current load
ILOAD OUT 0 pwl 0 0 1u 0 50u 0.5
```

Below is a plot of the current on the y-axis as a function of the V_{GS} on the x-axis. The current covers about five orders of magnitude, from a few microamps to half an amp, over one volt of gate drive.

That range is the problem, not the achievement. The transconductance of the pass-fet is roughly proportional to its current, and the pass-fet's g_m sets the loop gain, so a regulator that is stable at 500 mA has a hundred thousand times less loop gain at 5 uA — or, looked at the other way, a compensation network chosen for the light load leaves the loop far too fast at the heavy one. There is no single compensation that is right across five decades.

Sometimes it's easier to split the range into multiple ranges, which is what the next figure is about.

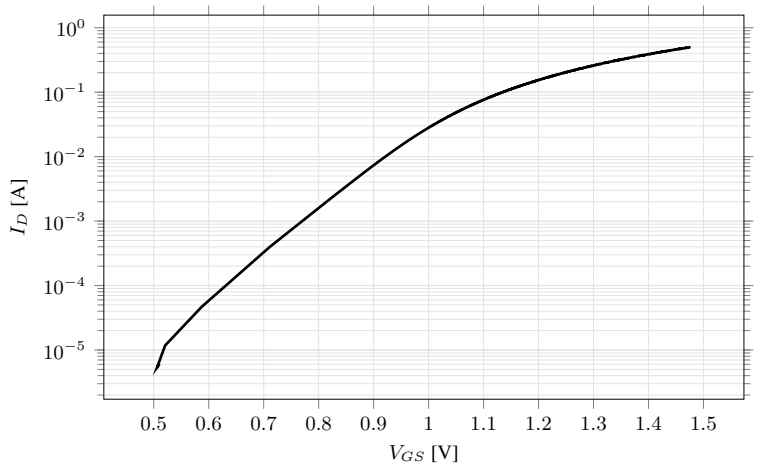


Figure 7: Simulated pass-fet drain current against gate-source voltage for the 500 mA LDO testbench, five decades of current over one volt of gate drive

As such, there are multiple control options for the pass-fet. Below is a summary of a few methods.

We can control the Vgs, or we can switch the number of instances, or we can turn the pass-fet on and off dynamically. What we choose will depend on the application.

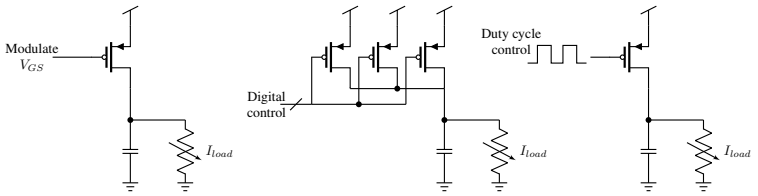


Figure 8: Pass-fet control options: analog Vgs modulation, digital control of parallel instances, and duty-cycle control

23.3 Switched Regulators

Linear regulators have poor power efficiency. Linear regulators have the same current in the load, as from the input.

For some applications a poor efficiency might be OK, but for most battery operated systems we're interested in using the electrons from the battery in the most effective manner.

Another challenge is temperature. A linear regulator with a 5 V input voltage, and 1 V output voltage will have a maximum power efficiency of 20 % (1/5). 80 % of the power is wasted in the pass-fet as heat.

Imagine a LDO driving an 80 W CPU at 1 V from a 5 V power supply. The power drawn from the 5 V supply is 400 W, as such, 320 W would be wasted in the LDO. A quad flat no-leads (QFN) package usually have a thermal resistance of 20 °C/W, so if it would be possible, the temperature of the LDO would be 6400 °C. Obviously, that cannot work.

For increased power efficiency, we must use switched regulators.

Imagine a switched regulator with 93 % power efficiency. The power from the 5 V supply would be $80\,\mathrm{W}/0.93 = 86\,\mathrm{W}$, as such, only 6 W is wasted as heat. A temperature increase of $6\,\mathrm{W} \times 20\,^{\circ}\mathrm{C/W} = 120^{\circ}\mathrm{C}$ is still high, but not impossible with a small heat-sink.

All switched regulators are based on devices that store electric field (capacitors), or magnetic field (inductors).

23.3.1 Principles of switched regulators

> There is a big difference between the idea for a circuit, and the actual implementation. A real DC/DC implementation may seem overwhelming.

Just look at figure 7 in A 10-MHz 2–800-mA 0.5–1.5-V 90% Peak Efficiency Time-Based Buck Converter With Seamless Transition Between PWM/PFM Modes [4]

So before we go into details, let's have a look at the principles.

23.3.1.1 Inductive BUCK DC/DC

Below is a common illustration of a inductive DC/DC to step down the voltage.

Imagine Vout is at our desired output voltage, for example 0.8 V. Assume Vin is 1.8 V.

When we close the switch, the inductor will begin to integrate the voltage across the inductor, and the current from Vin to Vout increases.

When we turn off the switch, the inductor current will not stop immediately, it cannot, that's what

$$V = L\frac{dI}{dt}$$

tells us. As a result, the current continues, but now the current is pulled from ground through the diode.

Since we're pulling current from ground, it should be intuitive that the current from Vin is less than the load current at Vout, assuming Vin > Vout.

The output voltage can be controlled by how long we turn on the switch. Each time we turn on the switch the inductor will inject a charge packet into the load capacitance.

If we have a control loop on the output voltage, then we can get an output voltage that is independent of the input voltage.

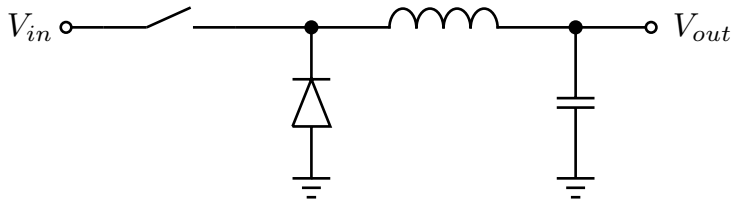


Figure 9: Principle of an inductive buck DC/DC converter: a switch, freewheeling diode, inductor and load capacitor

23.3.1.2 Capacitive BUCK DC/DC

In a capacitive buck below what we're doing is charging two capacitors in series to a high voltage, Vin, and then re-configuring the capacitors to be in parallel.

If the capacitors are the same size, then the output voltage would be half the input voltage.

To re-configure the circuit we'd use switches.

A disadvantage with capacitive bucks is that the output voltage is always a factor of the input voltage. When the input voltage changes, the output voltages changes proportionally.

Often we have to insert an LDO after a capacitive buck to make the output voltage independent of input voltage.

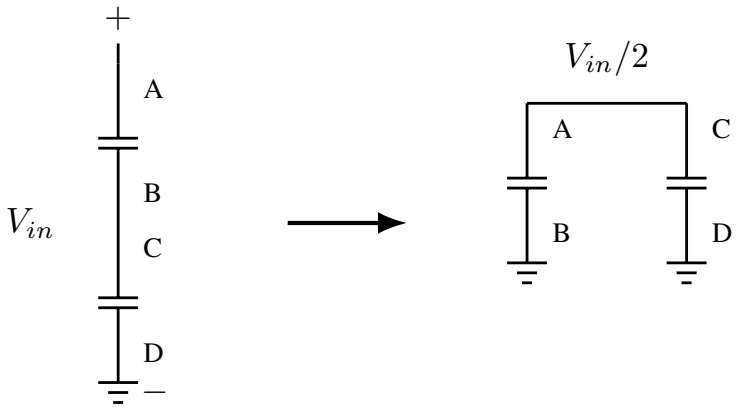


Figure 10: Principle of a capacitive buck: two capacitors charged in series are reconfigured in parallel to halve the voltage

23.3.1.3 Inductive BOOST DC/DC

Consider the circuit below. Here we setup a current from Vin to ground when the switch is on. When the switch is off push the current through the diode, and thus, the Vout can be higher than Vin.

In a similar manner to the Buck, the output voltage will be impacted by how long we turn on the switch for.

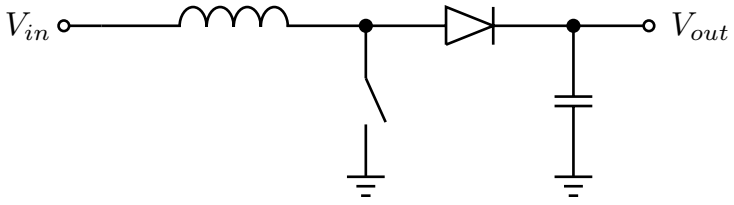


Figure 11: Principle of an inductive boost DC/DC converter: the inductor current is pushed through the diode to an output above Vin

23.3.1.4 Capacitive BOOST DC/DC

In a capacitive boost we start with a parallel connection, charge the capacitors to Vin, then reconfigure the circuit to a series combination.

As such, the output voltage would be two times the input voltage, assuming the capacitors are equal.

The configuration below is quite often called a "Charge pump", and can be configured to generate both positive, or negative voltages.

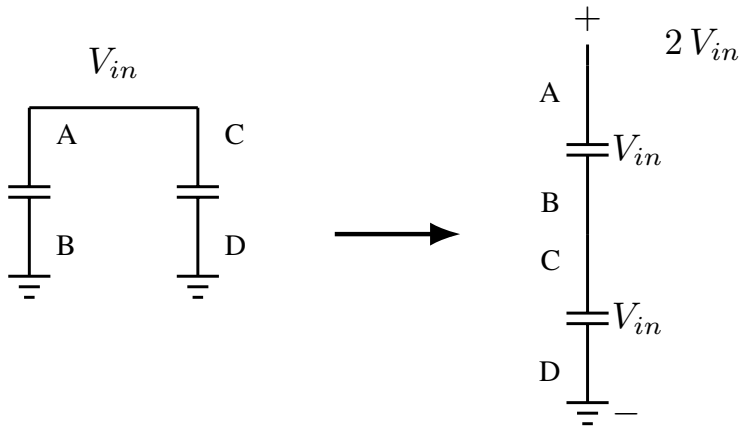


Figure 12: Principle of a capacitive boost (charge pump): two capacitors charged in parallel are stacked in series to double the voltage

23.3.2 Inductive DC/DC converter details

I've found that people struggle with inductive DC/DCs. They see a circuit inductors, capacitors, and transistors and think filters, Laplace and steady state. The path of Laplace and steady state will lead you astray and you won't understand how it works.

Hopefully I can put you on the right path to understanding.

In the figure below we can see a typical inductive switch mode DC/DC converter. The input voltage is V_{DDH}, and the output is V_O.

Most DC/DCs are feedback systems, so the control will be adjusted to force the output to be what is wanted, however, let's ignore closed loop for now.

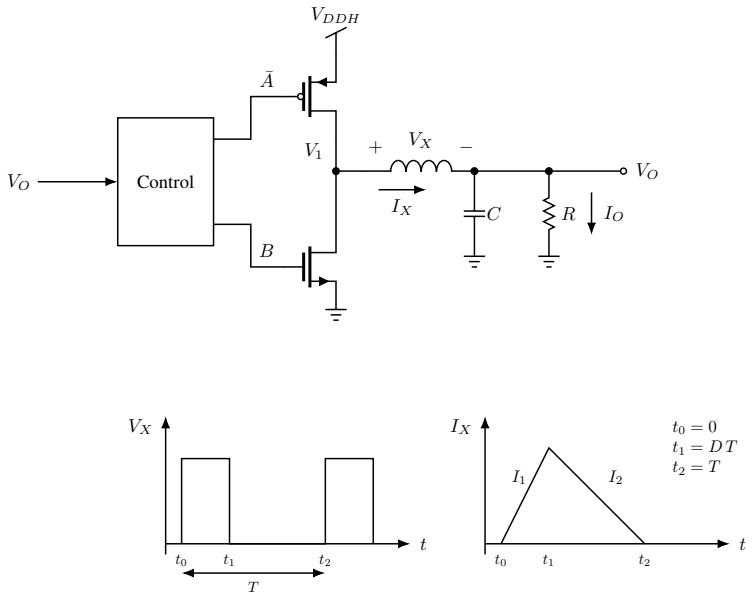


Figure 13: Inductive switch-mode buck converter with control block, and waveforms of the inductor voltage Vx and current Ix

To see what happens I find the best path to understanding is to look at the integral equations.

The current in the inductor is given by

$$I_x(t) = \frac{1}{L} \int V_x(t) dt$$

and the voltage on the capacitor is given by

$$V_o(t) = \frac{1}{C} \int (I_x(t) - I_o(t)) dt$$

Before you dive into Matlab, Mathcad, Maple, SymPy or another of your favorite math software, it helps to think a bit.

My mathematics is not great, but I don't think there is any closed form solution to the output voltage of the DC/DC, especially since the state of the NMOS and PMOS is time-dependent.

The output voltage also affect the voltage across the inductor, which affects the current, which affects the output voltage, etc, etc.

The equations can be solved numerically, but a numerical solution to the above integrals needs initial conditions.

There are many versions of the control block, let's look at two.

23.3.3 Pulse width modulation (PWM)

Assume $I_x = 0$ and $I_o = 0$ at $t = 0$. Assume the output voltage is $V_O = 0$. Imagine we set $A = 1$ for a fixed time duration. The voltage at $V_1 = V_{DDH}$, and $V_x = V_{DDH} - V_O$. As V_x is positive, and roughly constant, the current I_x would increase linearly, as given by the equation of the current above.

Since the I_x is linear, then the increase in V_o would be a second order, as given by the equation of the output voltage above.

Let's set $A = 0$ and $B = 1$ for a fixed time duration (it does not need to be the same as duration as we set $A = 1$). The voltage across the inductor would be $V_x = 0 - V_o$. The output voltage would not have increased much, so the absolute value of V_x during $A = 1$ would be higher than the absolute value of V_x during the first $B = 1$.

The V_x is now negative, so the current will decrease, however, since V_x is small, it does not decrease much.

I've made a

Jupyter PWM BUCK model - interactive - closed loop with type 3

that numerically solves the equations.

In the figure below we can see how the current during A increases fast, while during B it decreases little. The output voltage increases similarly to a second order function.

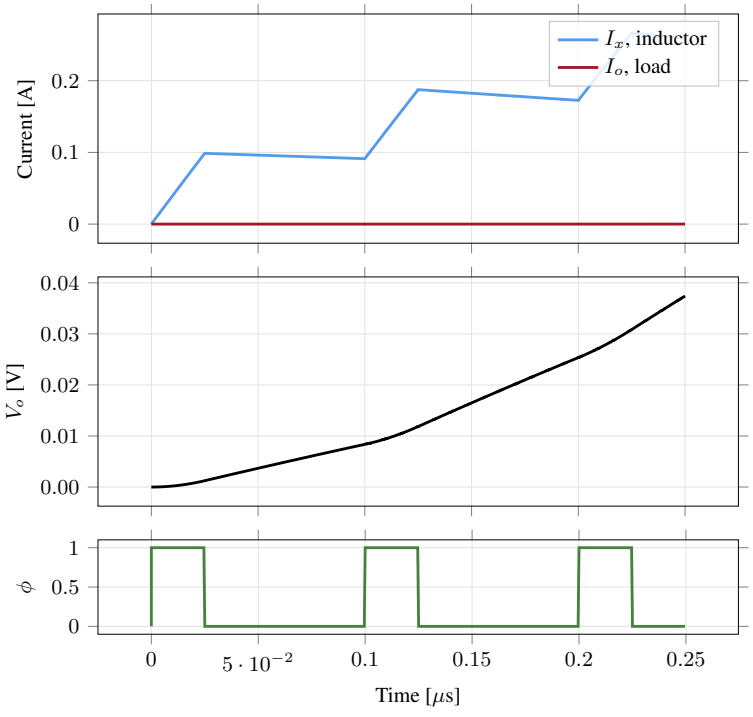


Figure 14: Start-up of the PWM buck model: the inductor current Ix increases fast during A=1, while the output voltage vo grows like a second order function

If we run the simulation longer, see plot below, the DC/DC will start to settle into a steady state condition.

On the top we can see the current I_x and I_o, the second plot you can see the output voltage. Turns out that the output voltage will be

$$V_o = V_{in} \times \text{Duty-Cycle}$$

, where the duty-cycle is the ratio between the duration of $A = 1$ and $B = 1$.

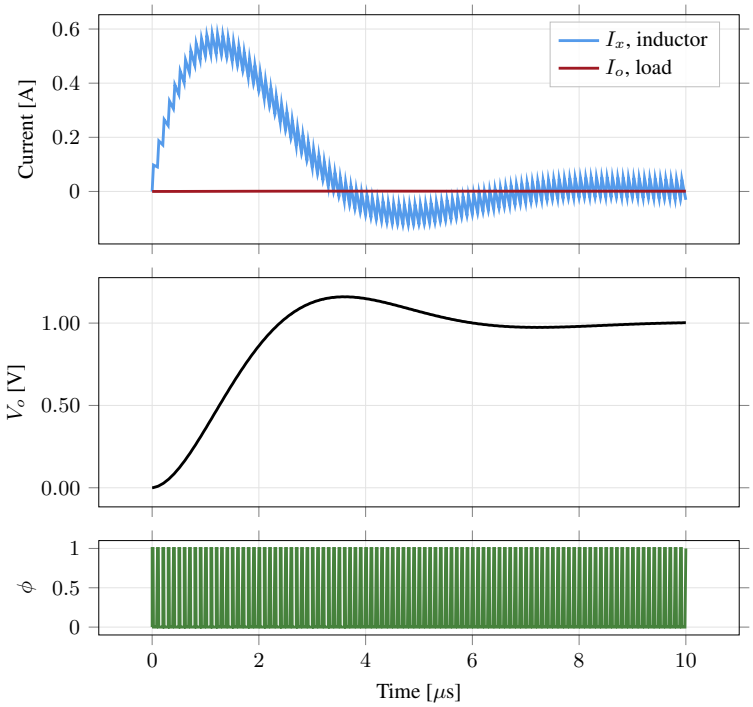


Figure 15: PWM buck model over a longer time: inductor and load currents (top), output voltage settling towards steady state (middle), and switch control A (bottom)

Once the system has fully settled, see figure below, we can see the reason for why DC/DC converters are useful.

During $A = 1$ the current I_x increases fast, and it's only during $A = 1$ we pull current from V_{DDH}. At the start of $A = 0$ the current is still positive, which means we pull current from ground. The average current in the inductor is the same as the average current in the load, however, the current from V_{DDH} is lower than the average inductor current, since some of the current comes from ground.

If the DC/DC was 100% efficient, then the current from the 4 V input supply would be 1/4'th of the current delivered to the 1 V output. 100% efficient DC/DC converters violate the laws of nature, so a good one reaches the low nineties under favourable conditions.

The model above manages 67 %, and it is worth understanding why, because the reason is not that the model is bad. Averaged over the settled part of the run it delivers 0.998 mW and draws 1.478 mW, so 0.48 mW is lost. The inductor carries 1 mA of useful DC and 76 mA peak to peak of ripple, which is 21.8 mA RMS, and all of it flows through the 1 Ω switch resistance. That is $I_{rms}^2 R = 0.477$ mW — within half a percent of the entire loss.

So the whole of the inefficiency here is ripple current heating the switches, and none of that current ever reaches the load. Two things follow. A converter is efficient at the load it was designed for and poor at a much lighter one, because the ripple does not shrink when the load does. And the way to fix it is a bigger inductor or a faster clock, both of which reduce the ripple, and both of which cost something else — area for the first, switching loss for the second. That trade is what the rest of this chapter is about.

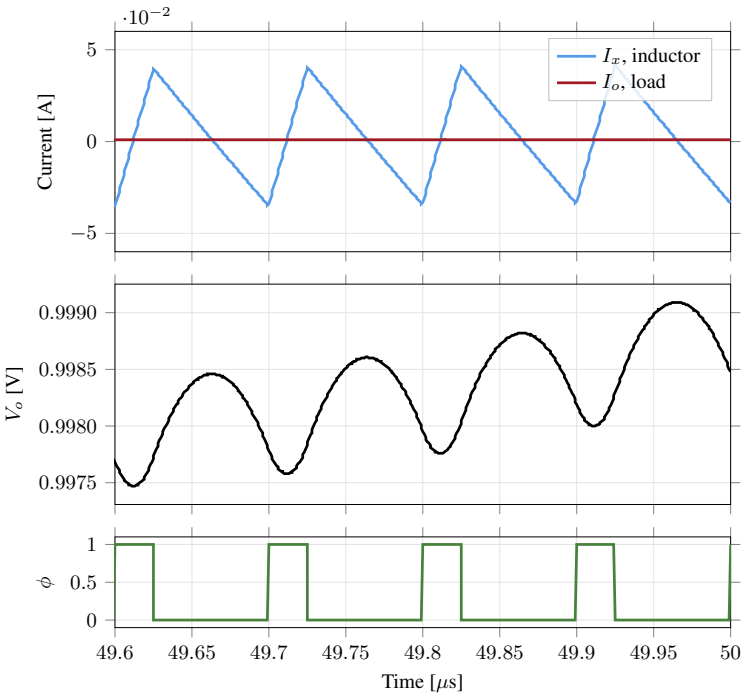


Figure 16: PWM buck model in steady state. The inductor current swings 76 mA peak to peak to supply a 1 mA load, and the output capacitor turns that into 1.6 mV of ripple on a 998 mV output

23.3.4 Real world use

DC/DC converters are used when power efficiency is important. Below is the DC/DC part of the reference schematic in the nRF5340 Product Specification.

We can see three inductors: one on DCCH, which brings VDDH down to VDD, and two for DECRF and DECD. As such, we can make a good guess there are three DC/DC converters inside the nRF5340.

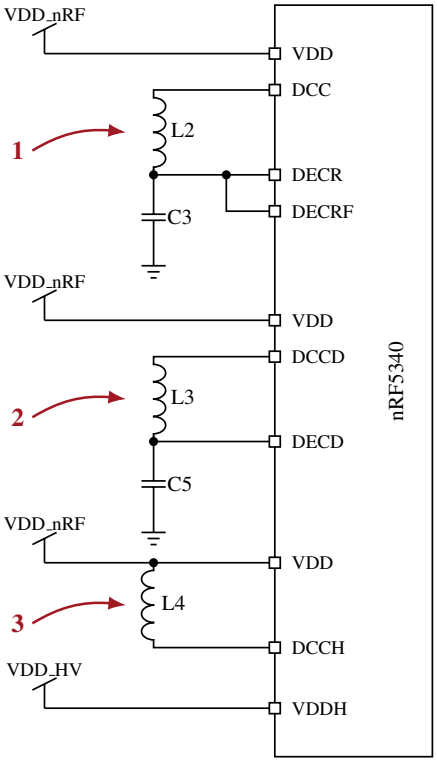


Figure 17: The DC/DC pins of the nRF5340, where the three inductors on DCC, DCCD and DCCH reveal three internal DC/DC converters. Simplified and redrawn from the reference schematic in Nordic Semiconductor, nRF5340 Product Specification

23.3.5 Pulsed Frequency Mode (PFM)

Power efficiency is key in DC/DC converters. For high loads, PWM, as explained above, is usually the most efficient and practical. For lighter loads, other configurations can be more efficient.

In PWM we continuously switch the NMOS and PMOS, as such, the parasitic capacitance on the V_1 node is charged and discharged, consuming power. If the load is close to 0 A, then the parasitic losses can be significant.

In pulsed-frequency mode we switch the NMOS and PMOS when it's needed. If there is no load, there is no switching, and V_1 or DCC in figure below is high impedance.

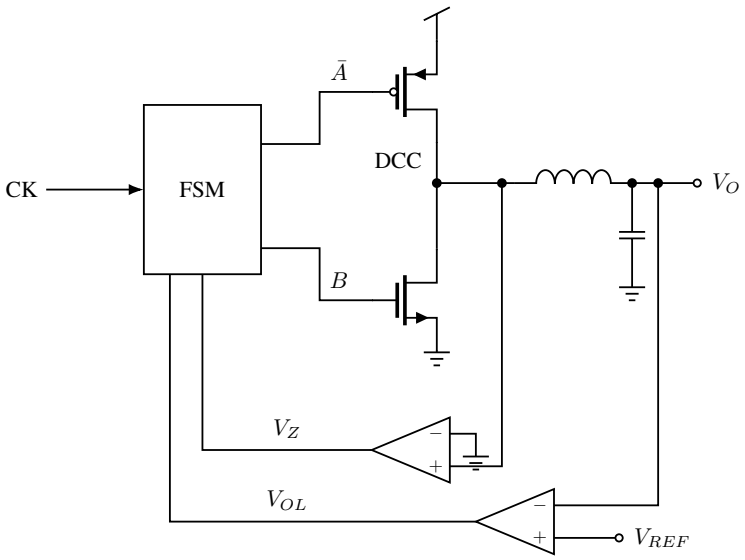


Figure 18: PFM buck architecture with an FSM driving the switches, a zero-cross comparator, and an output voltage comparator

Imagine V_o is at 1 V, and we apply a constant output load. According to the integral equations the V_o would decrease linearly.

In the figure above we observe V_o with a comparator that sets V_{OL} high if the $V_o < V_{REF}$. The output from the comparator could be the inputs to a finite state machine (FSM).

Consider the FSM below. On $vol = 1$ we transition to "UP" state where we turn on the PMOS for a fixed number of clock cycles. The inductor current would increase linearly. From the "UP" state we go to the "DWN" state, where we turn on the NMOS. The inductor current would decrease roughly linearly.

The "zero-cross" comparator observes the voltage across the NMOS drain/source. As soon as we turn the NMOS on the current direction in the inductor is still from DCC to V_o. Since the current is pulled from ground, the DCC must be below ground. As the current in the inductor decreases, the voltage across the NMOS will at some point be equal to zero, at which point the inductor current is zero.

When $vz = 1$ happens in the state diagram, or the zero cross comparator triggers, we transition from the "DWN" state back to "IDLE". Now the FSM wait for the next time $V_o < V_{REF}$.

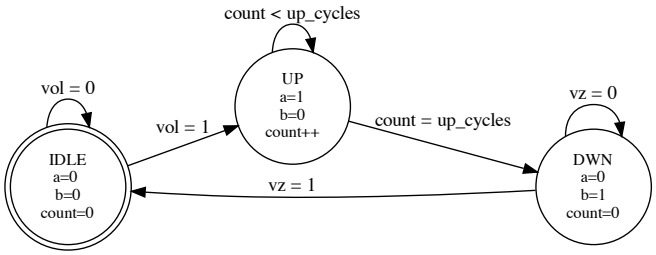


Figure 19: Finite state machine for PFM control with IDLE, UP, and DWN states

I think the name "pulsed-frequency mode" refers to the fact that the frequency changes according to load current, however, I'm not sure of the origin of the name. The name is not important. What's important is that you understand that mode 1 (PWM) and mode 2 (PFM) are two different "operation modes" of a DC/DC converter.

I made a jupyter model for the PFM mode. I would encourage you to play with them.

Below you can see a period of the PFM buck. The state can be seen in the bottom plot, the voltage in the middle and the current in the inductor and load in the top plot.

Jupyter PFM BUCK model - interactive

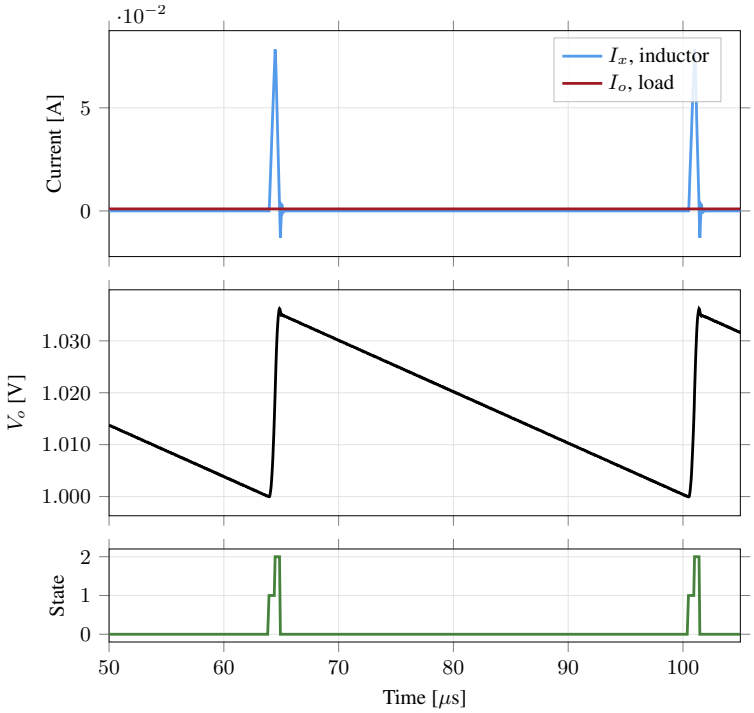


Figure 20: One period of the PFM buck model: inductor and load currents (top), output voltage (middle), and FSM state (bottom)

23.4 Summary

The one-page version of this chapter:

- Supply planning comes first: which blocks share a regulator, what noise they inject, what sequence they wake in
- A PMOS pass linear regulator gives the lowest dropout but a hard loop (output pole moves with load); the NMOS follower is easy to stabilize but costs a V_GS of headroom
- Linear regulators burn (V_in - V_out)/V_in of the power - fine for quiet rails, ruinous for big steps
- Inductive DC/DC converters move charge through an inductor at ~90% efficiency: PWM at heavy load, PFM pulses at light load
- Line/load regulation and PSRR are the datasheet numbers; the transient response to a load step is what the digital core actually feels

23.5 Would you like to know more?

Search terms: regulator, buck converter, dc/dc converter, boost converter

23.5.1 Linear regulators

A Scalable High-Current High-Accuracy Dual-Loop Four-Phase Switching LDO for Microprocessors [5] Overview of fancy LDO schemes, digital as well as analog

Development of Single-Transistor-Control LDO Based on Flipped Voltage Follower for SoC [6] In capacitor less LDOs a flipped voltage follower is a common circuit, worth a read.

A 200-mA Digital Low Drop-Out Regulator With Coarse-Fine Dual Loop in Mobile Application Processor [7] Some insights into large power systems.

23.5.2 DC-DC converters

Design Techniques for Fully Integrated Switched-Capacitor DC-DC Converters [8] Goes through design of SC DC-DC converters. Good place to start to learn the trade-offs, and the circuits.

High Frequency Buck Converter Design Using Time-Based Control Techniques [9] I love papers that challenge “this is the way”. Why

should we design analog feedback loops for our bucks, why not design digital feedback loops?

Single-Inductor Multi-Output (SIMO) DC-DC Converters With High Light-Load Efficiency and Minimized Cross-Regulation for Portable Devices [10] Maybe you have many supplies you want to drive, but you don't want to have many inductors. SIMO is then an option

A 10-MHz 2–800-mA 0.5–1.5-V 90% Peak Efficiency Time-Based Buck Converter With Seamless Transition Between PWM/PFM Modes [4] Has some lovely illustrations of PFM and PWM and the trade-offs between those two modes.

A monolithic current-mode CMOS DC-DC converter with on-chip current-sensing technique [11] In bucks converters there are two "religious" camps. One hail to "voltage mode" control loop, another hail to "current mode" control loops. It's good to read about both and make up your own mind.

[1] L. L. Lewyn, T. Ytterdal, C. Wulff, and K. Martin, "Analog Circuit Design in Nanoscale CMOS Technologies," *Proc. IEEE*, vol. 97, no. 10, pp. 1687–1714, Oct. 2009, doi: 10.1109/JPROC.2009.2024663.

[2] H.-S.-P. Wong *et al.*, "Metal–oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012, doi: 10.1109/JPROC.2012.2190369.

[3] T. Grasser *et al.*, "Simultaneous extraction of recoverable and permanent components contributing to bias-temperature instability," in *2007 IEEE international electron devices meeting*, 2007, pp. 801–804, doi: 10.1109/IEDM.2007.4419069.

[4] S. J. Kim, W.-S. Choi, R. Pilawa-Podgurski, and P. K. Hanumolu, "A 10-MHz 2–800-mA 0.5–1.5-v 90," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 3, pp. 814–824, 2018, doi: 10.1109/JSSC.2017.2776298.

[5] X. Mao, Y. Lu, and R. P. Martins, "A scalable high-current high-accuracy dual-loop four-phase switching LDO for microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 6, pp. 1841–1853, 2022, doi: 10.1109/JSSC.2021.3129620.

[6] T. Y. Man, K. N. Leung, C. Y. Leung, P. K. T. Mok, and M. Chan, "Development of single-transistor-control LDO based on flipped voltage follower for SoC," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 5, pp. 1392–1401, 2008, doi: 10.1109/TCSI.2008.916568.

[7] Y.-J. Lee *et al.*, "A 200-mA digital low drop-out regulator with coarse-fine dual loop in mobile application processor," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 64–76, 2017, doi: 10.1109/JSSC.2016.2614308.

[8] H.-P. Le, S. R. Sanders, and E. Alon, "Design techniques for fully integrated switched-capacitor DC-DC converters," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 9, pp. 2120–2131, 2011, doi: 10.1109/JSSC.2011.2159054.

[9] S. J. Kim *et al.*, "High frequency buck converter design using time-based control techniques," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 4, pp. 990–1001, 2015, doi: 10.1109/JSSC.2014.2378216.

[10] M.-H. Huang and K.-H. Chen, "Single-inductor multi-output (SIMO) DC-DC converters with high light-load efficiency and minimized cross-regulation for portable devices," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, pp. 1099–1111, 2009, doi: 10.1109/JSSC.2009.2014726.

[11] C. F. Lee and P. Mok, "A monolithic current-mode CMOS DC-DC converter with on-chip current-sensing technique," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 1, pp. 3–14, 2004, doi: 10.1109/JSSC.2003.820870.

Clocks and PLLs 24

Keywords: Systems, Feedback, PLL, Integer Divider, SD, SD PLL, Modulation, linear phase model

24.1 Why clocks? 453
24.1.1 A customer story . . 453
24.1.2 Frequency 455
24.1.3 Noise 455
24.1.4 Stability 455
24.1.5 Conclusion 455
24.2 A typical System-On-Chip clock system 456
24.2.1 32 MHz crystal . . . 456
24.2.2 32 KiHz crystal . . . 456
24.2.3 PCB antenna 456
24.2.4 DC/DC inductor . . 457
24.3 PLL 459
24.3.1 Integer PLL 459
24.3.2 Fractional PLL . . . 460
24.3.3 Modulation in PLLs 460
24.4 PLL Example 461
24.4.1 Loop gain 463
24.4.2 Controlled oscillator 463
24.4.3 Phase detector and charge pump 465
24.4.4 Loop filter 467
24.4.5 Divider 468
24.4.6 Loop transfer function 468
24.5 Summary 472
24.6 Would you like to know more? 472

24.1 Why clocks?

Virtually all integrated circuits have some form of clock system.

For digital we need clocks to tell us when the data is correct. For Radio's we need clocks to generate the carrier wave. For analog we need clocks for switched regulators, ADCs, accurate delay's or indeed, long delays.

The principle of a clock is simple. Make a 1-bit digital signal that toggles with a period T and a frequency $f = 1/T$.

The implementation is not necessarily simple.

The key parameters of a clock are the frequency of the fundamental, noise of the frequency spectrum, and stability over process and enviromental conditions.

When I start a design process, I want to know why, how, what (and sometimes who). If I understand the problem from first principles it's more likely that the design will be suitable.

But proving that something is suitable, or indeed optimal, is not easy in the world of analog design. Analog design is similar to physics. An hypothesis is almost impossible to prove "correct", but easier to prove wrong.

24.1.1 A customer story

Take an example.

24.1.1.1 Imagine a world

> "I have a customer that needs an accurate clock to count seconds". – Some manager that talked to a customer, but don't understand details.

As a designer, I might latch on to the word "accurate clock", and translate into "most accurate clock in the world", then I'd google atomic clocks, like Rubidium standard that I know is based on the hyperfine transition of electrons between two energy levels in rubidium-87.

I know from quantum mechanics that the hyperfine transition between two energy levels will produce an precise frequency, as the frequency of the photons transmitted is defined by $E = \hbar\omega = hf$.

I also know that quantum electro dynamics is the most precise theory in physics, so we know what's going on.

As long as the Rubidium crystal is clean (few energy states in the vicinity of the hyperfine transition), the distance between atoms stay constant, the temperature does not drift too much, then the frequency will be precise. So I buy a rubidium oscillator at a cost of $ 3k.

I design an ASIC to count the clock ticks, package it plastic, make a box, and give my manager.

Who will most likely say something like

> "Are you insane? The customer wants to put the clock on a wristband, and make millions. We can't have a cost of $ 3k per device. You must make it smaller and it must cost 10 cents to make"

Where I would respond.

> "What you're asking is physically impossible. We can't make the device that cheap, or that small. Nobody can do that."

And both my manager and I would be correct.

24.1.1.2 Imagine a better world

Most people in this world have no idea how things work. Very few people are able to understand the full stack. Everyone of us must simplify what we know to some extent. As such, as a circuit designer, it's your responsibility to fully understand what is asked of you.

When someone says

> " I have a customer that needs an accurate clock to count seconds"

Your response should be "Why does the customer need an accurate clock? How accurate? What is the customer going to use the clock for?". Unless you understand the details of the problem, then your design will be sub-optimal. It might be a great clock source, but it will be useless for solving the problem.

24.1.2 Frequency

The frequency of the clock is the frequency of the fundamental. If it's a digital clock (1-bit) with 50 % duty-cycle, then we know that a digital pulse train is an infinite sum of odd harmonics, where the fundamental is given by the period of the train.

24.1.3 Noise

Clock noise have many names. Cycle-to-cycle jitter is how the period changes with time. Jitter may also mean how the period right now will change in the future, so a time-domain change in the amount of cycle-to-cycle jitter. Phase noise is how the period changes as a function of time scales. For example, a clock might have fast period changes over short time spans, but if we average over a year, the period is stable.

What type of noise you care about depends on the problem. Digital will care about the cycle-to-cycle jitter affects on setup and hold times. Radio's will care about the frequency content of the noise with an offset to the carrier wave.

24.1.4 Stability

The variation over all corners and enviromental conditions is usually given in a percentage, parts per million, or parts per billion.

For a digital clock to run a Micro-Controller, maybe it's sufficient with 10% accuracy of the clock frequency. For a Bluetooth radio we must have +-50 ppm, set by the standard. For GPS we might need parts-per-billion.

24.1.5 Conclusion

Each "clock problem" will have different frequency, noise and stability requirements. You must know the order of magnitude of those before you can design a clock source. There is no "one-solution fits all" clock generation IP.

24.2 A typical System-On-Chip clock system

On the nRF52832 development kit you can see some components that indicate what type of clock system must be inside the IC.

On the board you will find the following items.

1. 32 MHz crystal
2. 32 KiHz crystal
3. PCB antenna
4. DC/DC inductor

24.2.1 32 MHz crystal

Any Bluetooth radio will need a frequency reference. We need to generate an accurate 2.402 GHz - 2.480 GHz carrier frequency for the gaussian frequency shift keying (GFSK) modulation. The Bluetooth Standard requires a +- 50 ppm accurate timing reference, and carrier frequency offset accuracy.

I'm not sure it's possible yet to make an IC that does not have some form of frequency reference, like a crystal. The ICs I've seen so far that have "crystal less radio" usually have a resonator (crystal or bulk-acoustic-wave or MEMS resonator) on die.

The power consumption of a high frequency crystal will be proportional to frequency. Assuming we have a digital output, then the power of that digital output will be $P = CV^2f$, for example $P = 100\,\text{fF} \times 1\,\text{V}^2 \times 32\,\text{MHz} = 3.2\,\mu\text{W}$ is probably close to a minimum power consumption of a 32 MHz clock.

24.2.2 32 KiHz crystal

Reducing the frequency, we can get down to minimum power consumption of $P = 100\,\text{fF} \times 1\,\text{V}^2 \times 32\,\text{KiHz} = 3.2\,\text{nW}$ for a clock.

For a system that sleeps most of the time, and only wakes up at regular ticks to do something, then a low-frequency crystal might be worth the effort.

24.2.3 PCB antenna

Since we can see the PCB antenna, we know that the IC includes a radio. From that fact we can deduce what must be inside the SoC. If we read the Product Specification we can understand more.

24.2.4 DC/DC inductor

Since we can see a large inductor, we can also make the assumption that the IC contains a switched regulator. That switched regulator, especially if it has a pulse-width-modulated control loop, will need a clock.

From our assumptions we could make a guess what must be inside the IC, something like the picture below.

There will be a crystal oscillator connected to the crystal. We'll learn about those later.

These crystal oscillators generate a fixed frequency, 32 MHz, or 32 KiHz, but there might be other clocks needed inside the IC.

To generate those clocks, there will be phase-locked loops (PLL), frequency locked loops (FLL), or delay-locked loops (DLL).

PLLs take a reference input, and can generate a higher frequency, (or indeed lower frequency) output. A PLL is a magical block. It's one of the few analog IPs where we can actually design for infinite gain in our feedback loop.

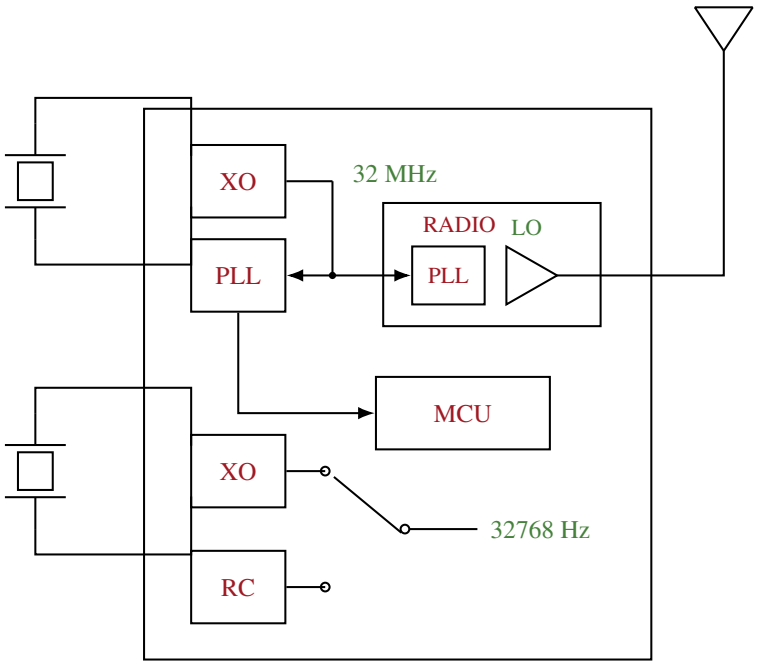


Figure 1: Guess at the clock system inside the SoC: crystal oscillators (XO) for 32 MHz and 32768 Hz, a PLL for the radio local oscillator, an RC oscillator, and the MCU clock

Most of the digital blocks on an IC will be synchronous logic, see figure below. A fundamental principle of synchronous logic is that the data at the flip-flops (DFF, rectangles with triangle clock input, D, Q and $\overline{Q}$) only need to be correct at certain times.

The sequence of transitions in the combinatorial logic is of no consequence, as long as the B inputs are correct when the clock goes high next time.

The registers, or flip-flops, are your SystemVerilog "always_ff" code. While the blue cloud is your "always_comb" code.

In a SoC we have to check, for all paths between a Y[N] and B[M] that the path is fast enough for all transients to settle before the clock strikes next time. How early the B data must arrive in relation to the clock edge is the setup time of the DFFs.

We also must check for all paths that the B[M] are held for long enough after the clock strikes such that our flip-flop does not change state. The hold time is the distance from the clock edge to where the data is allowed to change. Negative hold times are common in DFFs, so the data can start to change before the clock edge.

In an IC with millions of flip-flops there can be billions of paths. The setup and hold time for every single one must be checked. One could imagine a simulation of all the paths on a netlist with parasitics (capacitors and resistors from layout) to check the delays, but there are so many combinations that the simulation time becomes unpractical.

Static Timing Analysis (STA) is a light-weight way to check all the paths. For the STA we make a model of the delay in each cell (captured in a liberty file), the setup/hold times of all flip-flops, wire propagation delays, clock frequency (or period), and the variation in the clock frequency. The process, voltage, temperature variation must also be checked for all components, so the number of liberty files can quickly grow large.

For an analog designer the constraints from digital will tell us what's the maximum frequency we can have at any point in time, and what is the maximum cycle-to-cycle variation in the period.

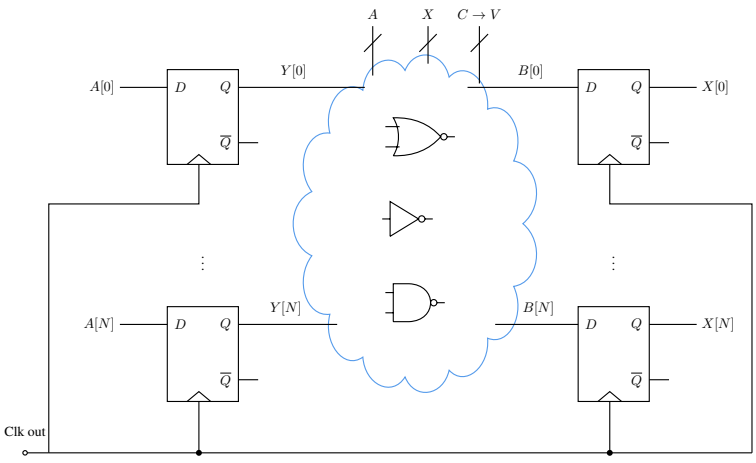
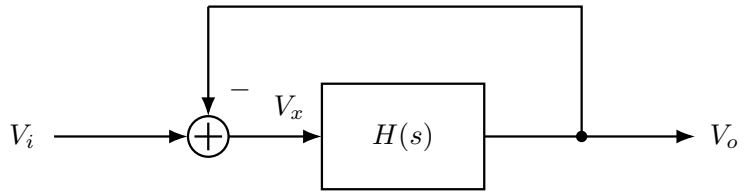


Figure 2: Synchronous logic: flip-flops capture the data on the clock edge, with combinatorial logic (blue cloud) between register stages

24.3 PLL

PLL, or it's cousins FLL and DLL are really cool. A PLL is based on the familiar concept of feedback, shown in the figure below. As long as we make $H(s)$ infinite we can force the output to be an exact copy of the input.



$$V_i - V_o = V_x \qquad V_o = V_x\, H(s)$$

$$V_i = V_o + \frac{V_o}{H(s)} \qquad H(s) = \infty \;\Rightarrow\; V_o = V_i$$

Figure 3: Feedback loop where an infinite gain H(s) forces the output to be an exact copy of the input

24.3.1 Integer PLL

For a frequency loop the figure looks a bit different. If we want a higher output frequency we can divide the frequency by a number (N) and compare with our reference (for example the 32 MHz reference from the crystal oscillator).

We then take the error, apply a transfer function $H(s)$ with high gain, and control our oscillator frequency.

If the down-divided output frequency is too high, we force the oscillator to a lower frequency. If the down-divided output frequency is too low we force the oscillator to a higher frequency.

If we design the $H(s)$ correctly, then we have $f_o = N \times f_{in}$

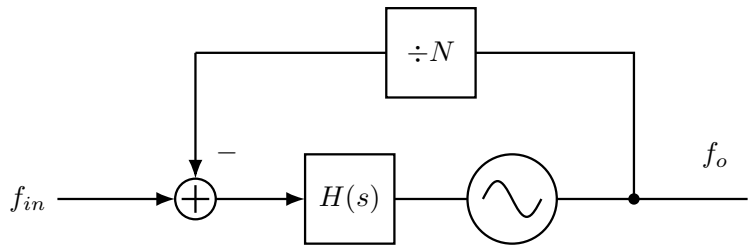


Figure 4: Integer PLL: the oscillator output is divided by N and compared to the reference, giving an output frequency N times the reference

Sometimes you want a finer frequency resolution, in that case you'd add a divider on the reference and get $f_o = N \times \frac{f_{in}}{M}$..

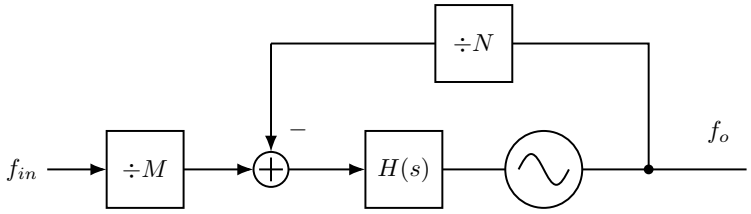


Figure 5: Integer PLL with an additional divide-by-M on the reference for finer frequency resolution

24.3.2 Fractional PLL

Trouble is that dividing down the input frequency will reduce your loop bandwidth, as the low-pass filter needs to be about 1/10'th of the reference frequency. As such, the PLL will respond slower to a frequency change.

We can also use a fractional divider, where we swap between two, or more, integers in a sigma-delta fashion in the divider.

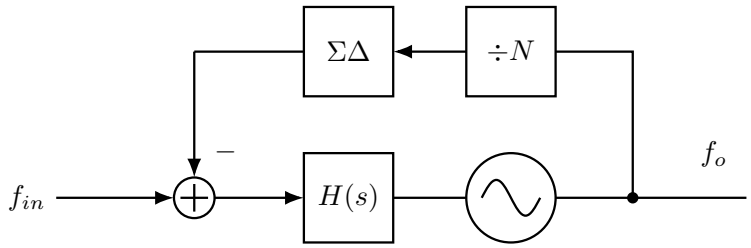


Figure 6: Fractional PLL where a sigma-delta modulator switches the feedback divider between integer values

24.3.3 Modulation in PLLs

From your signal processing, or communication courses, you may recognize the equation below.

$$A_m(t) \times cos\left(2\pi f_{carrier} t + \phi_m(t)\right)$$

The A_m is the amplitude modulation, while the ϕ_m is the phase modulation. Bluetooth Low Energy is constant envelope, so the A_m is a constant. The phase modulation is applied to the carrier, but how is it done?

One option is shown below. We could modulate our frequency reference directly. That could maybe be a sigma-delta divider on the reference, or directly modulating the oscillator.

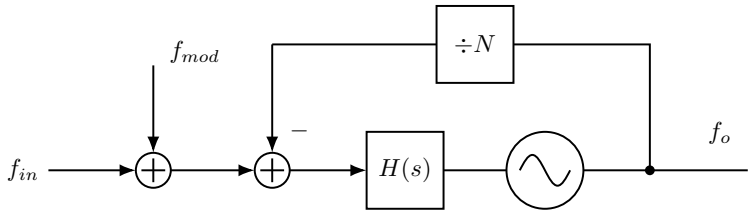


Figure 7: Modulating the PLL by adding the modulation signal directly to the frequency reference

Most modern radios, however, will have a two-point modulation. The modulation signal is applied to the VCO (or DCO), and the opposite signal is applied to the feedback divider. As such, the modulation is not seen by the loop.

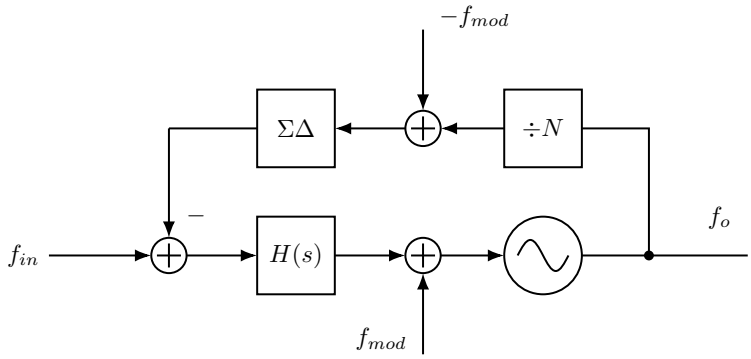


Figure 8: Two-point modulation: the modulation is applied to the oscillator and the opposite signal to the sigma-delta feedback divider, so the loop does not see it

24.4 PLL Example

I've made an example PLL that you can download and play with. I make no claims that it's a good PLL. Actually, I know it's a bad PLL. The ring-oscillator frequency varies too fast with the voltage control. But it does give you a starting point.

A PLL can consist of a oscillator (SUN_PLL_ROSC) that generates our output frequency. A divider (SUN_PLL_DIVN) that generates a feedback frequency that we can compare to the reference. A Phase and Frequency Detector (SUN_PLL_PFD) and a charge-pump (SUN_PLL_CP) that model the +, or the comparison function in our previous picture. And a loop filter (SUN_PLL_LPF and SUN_PLL_BUF) that is our $H(s)$.

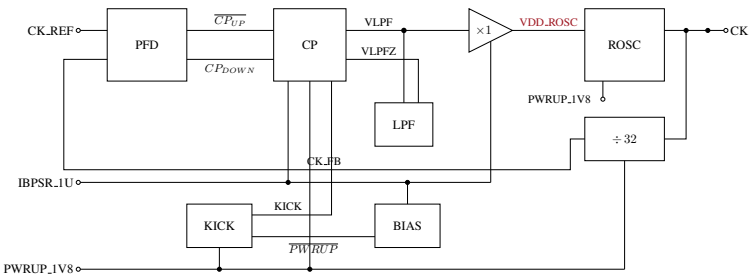


Figure 9: Top-level schematic of the SUN_PLL example with phase-frequency detector, charge-pump, loop filter, buffer, ring oscillator, and divide-by-32 feedback divider. Block placement follows the xschem source: signal flow left to right along the loop, feedback below, bias and start-up at the bottom

Read any book on PLLs, talk to any PLL designer and they will all tell you the same thing. **PLLs require calculation**. You must setup a linear model of the feedback loop, and calculate the loop transfer function to check the stability, and the loop gain. **This is the way!** (to quote Mandalorian).

But how can we make a linear model of a non-linear system? The voltages inside a PLL must be non-linear, they are clocks. A PLL is not linear in time-domain!

I have no idea who first thought of the idea, but it turns out, that one can model a PLL as a linear system if one consider the phase of the voltages inside the PLL, especially when the PLL is locked (phase of the output and reference is mostly aligned). Where the phase is defined as

$$\phi(t) = 2\pi \int_0^t f(\tau) d\tau$$

As long as the bandwidth of the $H(s)$ is about $\frac{1}{10}$ of the reference frequency, then the linear model below holds (at least is good enough).

The phase of our input is $\phi_{in}(s)$, the phase of the output is $\phi(s)$, the divided phase is $\phi_{div}(s)$ and the phase error is $\phi_d(s)$.

The K_{pd} is the gain of our phase-frequency detector and charge-pump. The $K_{lp}H_{lp}(s)$ is our loop filter $H(s)$. The K_{osc}/s is our oscillator transfer function. And the $1/N$ is our feedback divider.

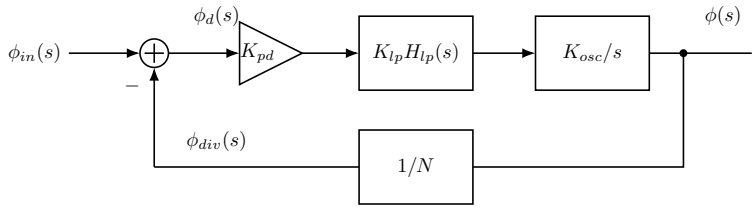


Figure 10: Linear phase-domain model of the PLL with phase-detector gain, loop filter, oscillator integrator and 1/N feedback divider

24.4.1 Loop gain

The loop transfer function can then be analyzed and we get.

$$\frac{\phi_d}{\phi_{in}} = \frac{1}{1 + L(s)}$$

$$L(s) = \frac{K_{osc} K_{pd} K_{lp} H_{lp}(s)}{Ns}$$

Here is the magic of PLLs. Notice what happens when $s = j\omega = j0$, or at zero frequency. If we assume that $H_{lp}(s)$ is a low pass filter, then $H_{lp}(0)$ = constant. The loop gain, however, will have a $L(0) \propto \frac{1}{0}$ which approaches infinity at 0.

That means, we have an infinite DC gain in the loop transfer function. It is the only case I know of in an analog design where we can actually have infinite gain. Infinite gain translates to infinite precision.

If the reference was a Rubidium oscillator we could generate any frequency with the same precision as the frequency of the Rubidium oscillator. Magic.

For the linear model, we need to figure out the factors, like K_{osc}, which must be determined by simulation.

24.4.2 Controlled oscillator

The gain of the oscillator is the change in output frequency as a function of the change of the control node. For a voltage-controlled oscillator (VCO) we could sweep the control voltage, and check the frequency. The derivative of the f(V) would be proportional to the K_{vco}.

The control node does not need to be a voltage. Anything that changes the frequency of the oscillator can be used as a control node. There exist PLLs with voltage control, current control, capacitance control, and digital control.

For the SUN_PLL_ROSC it is the VDD of the ring-oscillator (VDD_-ROSC) that is our control node.

$$K_{osc} = 2\pi \frac{df}{dV_{cntl}}$$

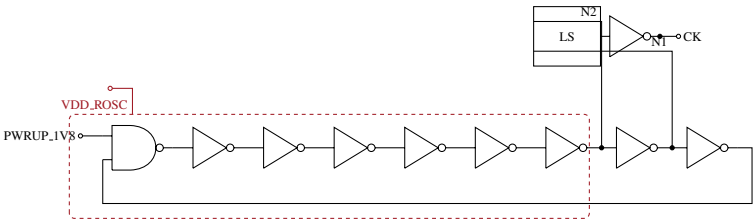


Figure 11: The ring oscillator SUN_PLL_ROSC: a NAND and eight inverters make nine inversions, so the loop oscillates whenever PWRUP_1V8 is high. The ring runs on VDD_ROSC — the supply is the control node — and the level shifter LS brings taps N2/N1 back to the AVDD domain to make CK

24.4.2.1 SUN_PLL_SKY130NM/sim/ROSC/

I simulate the ring oscillator in ngspice with a transient simulation and get the oscillator frequency as a function of voltage.

tran.spi

```
let start_v = 1.1
let stop_v = 1.7
let delta_v = 0.1
let v_act = start_v
* loop
while v_act le stop_v
alter VROSC v_act
tran 1p 40n
meas tran vrosc avg v(VDD_ROSC)
meas tran tpd trig v(CK) val='0.8' rise=10 targ v(CK) val='0.8' rise=11
let v_act = v_act + delta_v
end
```

I use `tran.py` to extract the time-domain signal from ngspice into a CSV file.

Then I use a python script to extract the K_{osc}

kvco.py

```
df = pd.read_csv(f)
freq = 1/df["tpd"]
kvco = np.mean(freq.diff()/df["vrosc"].diff())
```

Below I've made a plot of the oscillation frequency over corners.

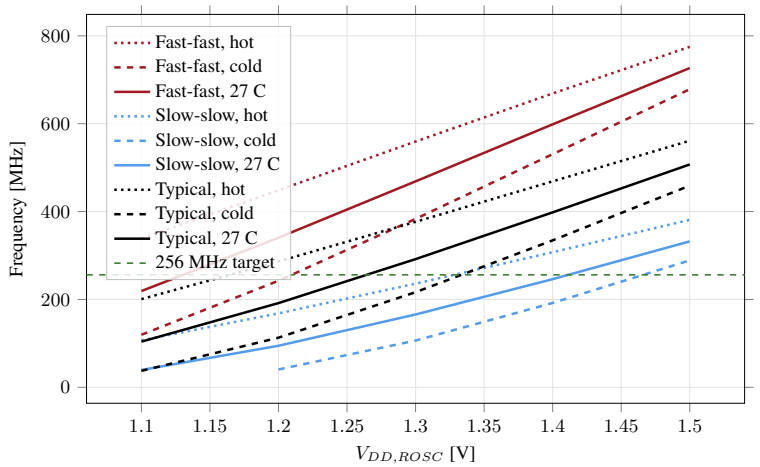


Figure 12: Ring-oscillator frequency versus control voltage VDD_ROSC over nine process and temperature corners, simulated on the extracted layout. The slope at the typical corner is 1.01 GHz/V, which is K_{osc}; the spread is a factor of eleven at 1.2 V, narrowing to under three at 1.5 V

Two things in that plot are worth more than the slope.

The first is the spread. A ring oscillator has nothing setting its frequency except how fast its own inverters switch, so process and temperature move it by a factor of eleven at the bottom of the control range. Every corner does cross the 256 MHz the loop needs, but the slow-cold one only at about 1.44 V, near the top of what the control node can deliver. The usable tuning range is not the width of the control range, it is whatever is left above that crossing in the worst corner.

The second is the missing point. At slow-cold and 1.1 V there is no measurement, because the oscillator was too slow to produce enough edges inside the simulated window and the measurement failed rather than returning a plausible wrong number. A failed measurement in the corner you were already worried about is information, not an inconvenience — and it is a good argument for reading the simulator’s errors rather than only its plots.

24.4.3 Phase detector and charge pump

The two blocks compare our reference clock to our feedback clock, and produce an error signal. The gain of the pair is the average current fed into the loop filter per radian of phase error, and it is worth deriving once because the 2π looks arbitrary until you do.

The phase-frequency detector turns a phase error into a pulse width. If the feedback clock arrives late by a phase $\Delta\phi$, the UP output is high for the fraction $\Delta\phi/2\pi$ of the reference period, because a full period is 2π of phase. During that pulse the charge

pump sources its full current I_{cp}, and for the rest of the period it sources nothing. The average current into the filter is therefore

$$\bar{I} = I_{cp}\frac{\Delta\phi}{2\pi}$$

and the gain, being average current per radian, is what is left when you divide by $\Delta\phi$:

$$K_{pd} = \frac{I_{cp}}{2\pi}$$

Two things follow that are easy to miss. The gain does not depend on the reference frequency, because both the pulse width and the period scale together. And it is the *average* current that the loop filter sees, which is only a fair description if the filter is slow compared with the reference — the same assumption that let us draw a linear model in the first place.

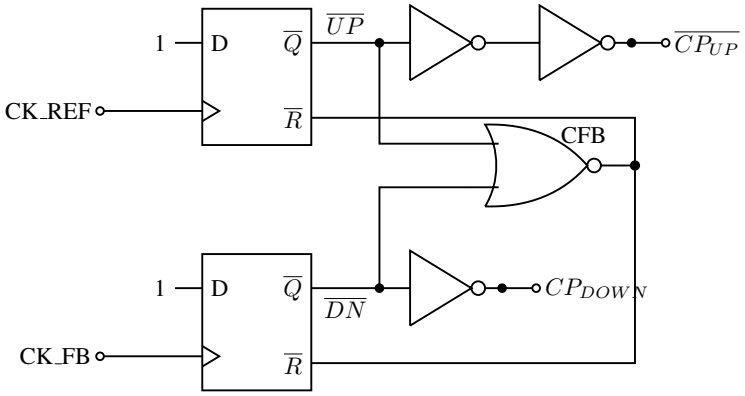


Figure 13: The phase-frequency detector SUN_PLL_PFD: two flip-flops with D tied high, one set by CK_REF, the other by CK_FB. The moment both are set the NOR resets the pair, so the surviving pulse width is the arrival-time difference — phase error becomes pulse width

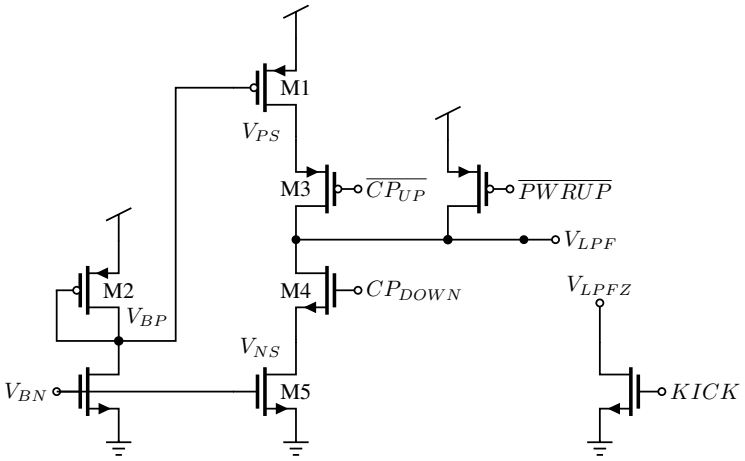


Figure 14: The charge pump SUN_PLL_CP, driven by the phase-frequency detector through CP_UP_N and CP_DOWN. V_BN sets I_cp, the switch pair steers it into or out of V_LPF, M7 parks V_LPF at AVDD in power-down, and the KICK switch grabs the filter's zero node to start the loop. The mirror devices are stacked pairs in silicon, drawn single here

24.4.4 Loop filter

In the book you'll find a first order loop filter, and a second order loop filter. Engineers are creative, so you'll likely find other loop filters in the literature.

I would start with the "known to work" loop filters before you explore on your own.

If you're really interested in PLLs, you should buy Design of CMOS Phase-Locked Loops by Behzad Razavi.

The loop filter has a unity gain buffer. My oscillator draws current, while the VLPF node is high impedance, so I can't draw current from the loop filter without changing the filter transfer function.

$$K_{lp}H_{lp}(s) = K_{lp}\left(\frac{1}{s} + \frac{1}{\omega_z}\right)$$

$$K_{lp}H_{lp}(s) = \frac{1}{s(C_1 + C_2)}\frac{1 + sRC_1}{1 + sR\frac{C_1C_2}{C_1+C_2}}$$

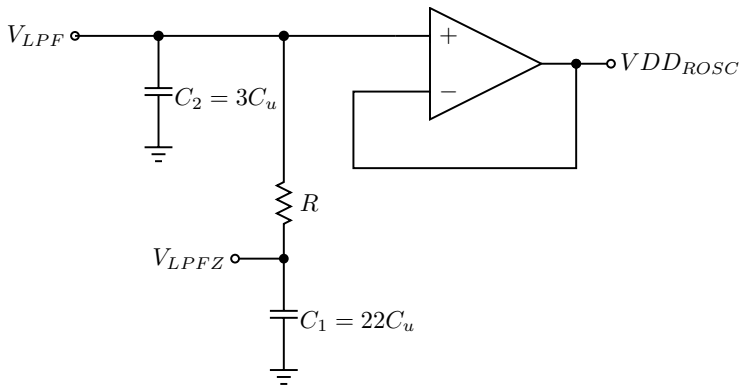


Figure 15: The loop filter SUN_PLL_LPF on VLPF, followed by the buffer SUN_PLL_BUF that drives the oscillator supply VDD_ROSC. C1 is 22 unit capacitors, C2 is 3, so the zero sits where the equations above put it

24.4.5 Divider

The divider is modelled as

$$K_{div} = \frac{1}{N}$$

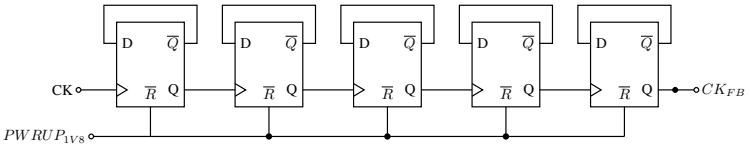


Figure 16: The feedback divider SUN_PLL_DIVN: five flip-flops wired as toggles, each clocking the next, dividing CK by 32 to make CK_FB

24.4.6 Loop transfer function

With the loop transfer function we can start to model what happens in the linear loop. What is the phase response, and what is the gain response.

$$L(s) = \frac{K_{osc} K_{pd} K_{lp} H_{lp}(s)}{Ns}$$

24.4.6.1 Python model

I've made a python model of the loop, you can find it at sun_pll_-sky130nm/jupyter/pll - interactive

In the jupyter notebook below you can find some more information on the phase/frequency detector, and charge pump.

sun_pll_sky130nm/jupyter/pfd

Below is a plot of the loop gain, and the transfer function from input phase to divider phase.

We can see that the loop gain at low frequency is large, and proportional to $1/s$. As such, the phase of the divided down feedback clock is the same as our reference.

The closed loop transfer function ϕ_{div}/ϕ_{in} shows us that the divided phase at low frequency is the same as the input phase. Since the phase is the same, and the frequency must be the same, then we know that the output clock will be N times reference frequency.

Which K_{osc} went into that plot matters more than it looks. The schematic simulation gave 1.6 GHz/V; the extracted layout gives 1.01 GHz/V, and the difference is parasitic capacitance in the ring

that simply does not exist until the oscillator is laid out. A third of the gain disappears, and since the loop gain is proportional to K_{osc}, the crossover moves from 0.59 MHz down to 0.43 MHz and the phase margin from 51 degrees to 43.

Eight degrees is not a catastrophe, and that is rather the point: it is the sort of erosion that is easy to spend twice over without noticing. Extract early, and design the loop with the number the silicon will actually have rather than the one the schematic promised.

It is also worth checking this plot against the assumption we made when we drew the linear model at all. The loop gain crosses 0 dB at 0.43 MHz, and the reference frequency is 256 MHz/32 = 8 MHz, so the loop bandwidth is a nineteenth of the reference. That clears the "one tenth of the reference" rule, which means the model is entitled to be believed. If it had not cleared it, the phase margin the plot reports would be a number about a model that does not describe the circuit — and that is a far worse situation than a poor phase margin, because it looks fine.

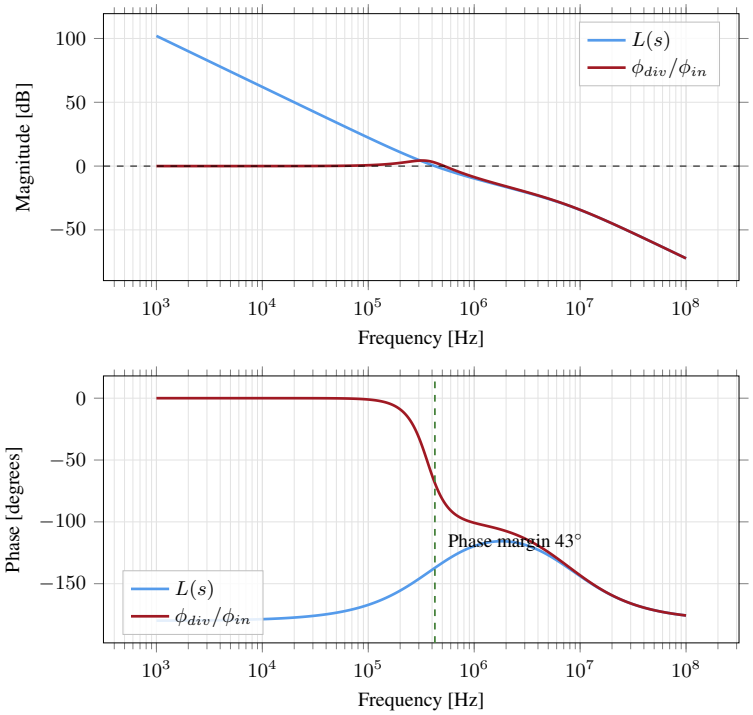


Figure 17: Magnitude and phase of the loop gain and the closed-loop transfer function from input phase to divider phase, using the oscillator gain measured on the extracted layout. The loop crosses 0 dB at 0.43 MHz with 43 degrees of phase margin

The top testbench for the PLL is tran.spi.

I power up the PLL and wait for the output clock to settle. The frequency is measured the way a counter would measure it: find

every rising edge of CK and take the reciprocal of the interval between consecutive edges. See freq.py.

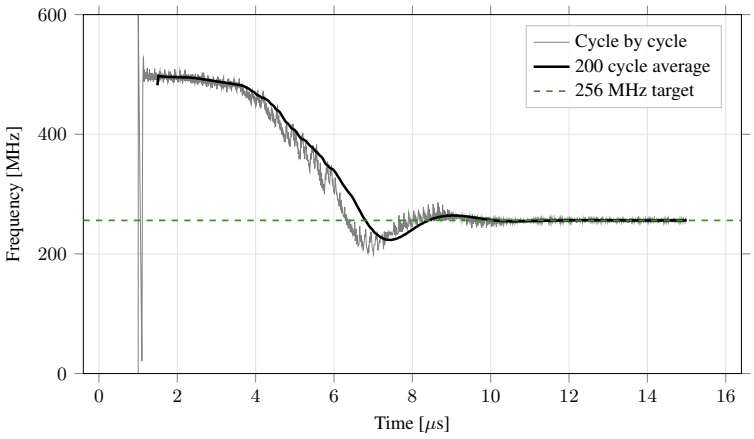


Figure 18: Simulated PLL output frequency from power-up, on the extracted layout at the typical corner. The grey trace is the frequency of each individual cycle and the black one a 200 cycle average; the loop overshoots to about 500 MHz, undershoots past the target, and settles at 256.1 MHz around 12 microseconds

Three things in that plot are worth pausing on.

The loop starts at the top of the oscillator's range and has to be dragged down, so the first microsecond is not feedback at all, it is the control node charging. Then the loop takes over, overshoots past the target, and rings once before settling. That single undershoot is the second order response the phase margin describes; 43 degrees is what one visible ring looks like.

The grey band does not narrow as the loop settles. That is not the simulation failing to converge, and it is not an error the loop could correct: the charge pump delivers its correction as a pulse once per reference period, so the oscillator is kicked 8 million times a second whatever the loop is doing. In a real chip this is the reference spur, and it is the reason a PLL's output is never as clean as its reference.

Settling takes about 12 microseconds here, against roughly 8 in earlier versions of this design. The loop got slower because the oscillator got slower: the extracted layout has a third less gain than the schematic, and loop bandwidth is proportional to that gain. Nothing was designed differently; the parasitics simply arrived.

You can find the schematics, layout, testbenches, python script etc at SUN_PLL_SKY130NM

Below are a couple layout images of the finished PLL

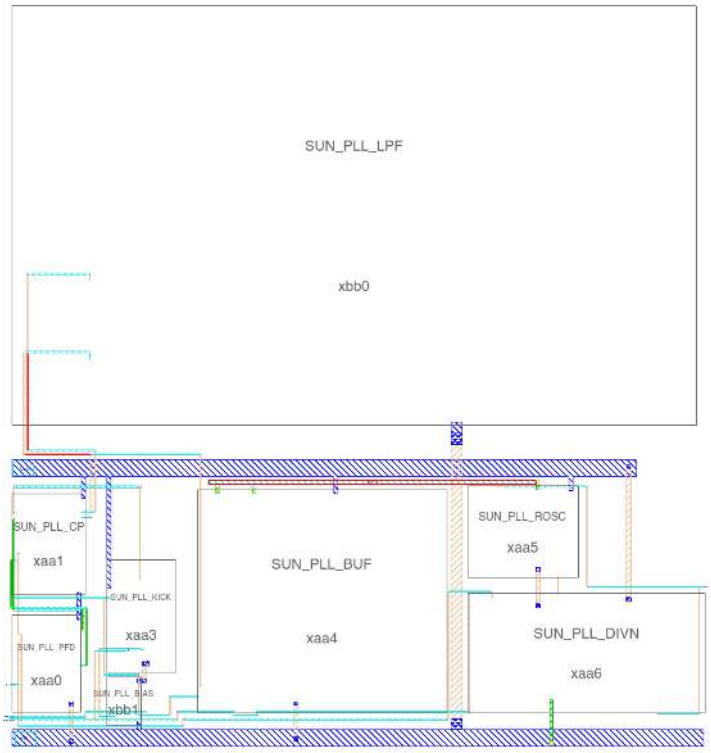


Figure 19: Floorplan of the SUN_PLL layout; the loop filter capacitor SUN_PLL_LPF dominates the area above the PLL blocks

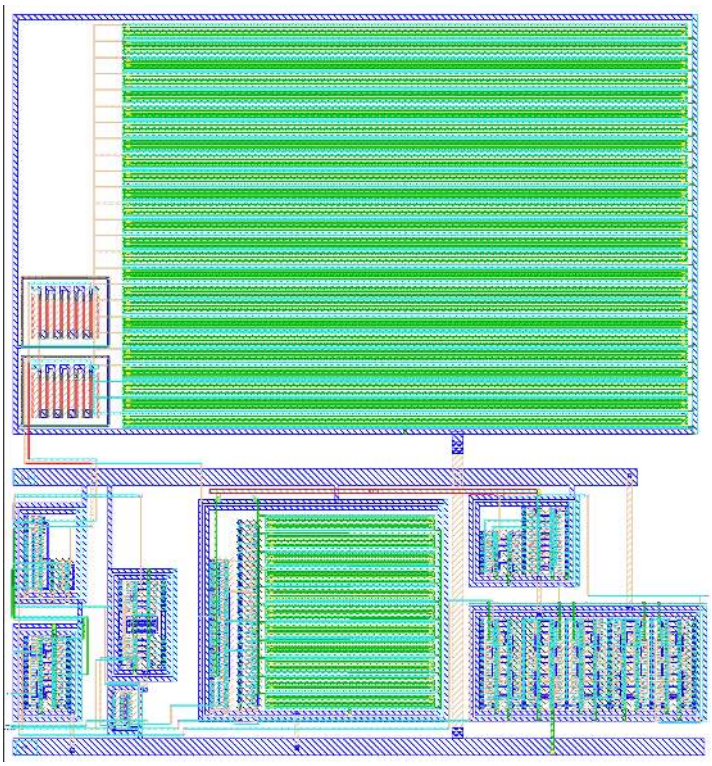


Figure 20: Finished SUN_PLL layout showing the loop filter capacitor array and the PLL blocks along the bottom

24.5 Summary

The one-page version of this chapter:

- Everything on the chip wants a clock, and every clock is a compromise between frequency accuracy (ppm), phase noise and power
- A crystal gives the accurate reference; the PLL multiplies it up to the frequency the system needs
- PFD turns phase error into pulse width, the charge pump into charge, the loop filter into a control voltage, the oscillator into frequency, and the divider closes the loop
- The type-II loop needs its zero: C1 sets the zero with R, C2 cleans the ripple, and the loop bandwidth balances reference noise against VCO noise
- Inside the bandwidth the PLL follows the reference; outside, the oscillator is on its own - phase noise plots read exactly that way
- SUN_PLL is the whole story in five schematics: ROSC, PFD, charge pump, loop filter, divider

24.6 Would you like to know more?

Back in 2020 there was a Master student at NTNU on PLL. I would recommend looking at that thesis to learn more, and to get inspired Ultra Low Power Frequency Synthesizer.

A Low Noise Sub-Sampling PLL in Which Divider Noise is Eliminated and PD/CP Noise is Not Multiplied by N2 [1]

All-digital PLL and transmitter for mobile phones [2]

A 2.9–4.0-GHz Fractional-N Digital PLL With Bang-Bang Phase Detector and 560-fsrms Integrated Jitter at 4.5-mW Power [3]

[1] X. Gao, E. A. M. Klumperink, M. Bohsali, and B. Nauta, "A low noise sub-sampling PLL in which divider noise is eliminated and PD/CP noise is not multiplied by N^2," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 12, pp. 3253–3263, 2009, doi: 10.1109/JSSC.2009.2032723.

[2] R. Staszewski *et al.*, "All-digital PLL and transmitter for mobile phones," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 12, pp. 2469–2482, 2005, doi: 10.1109/JSSC.2005.857417.

[3] D. Tasca, M. Zanuso, G. Marzin, S. Levantino, C. Samori, and A. L. Lacaita, "A 2.9–4.0-GHz fractional-n digital PLL with bang-bang phase detector and 560-fs_{rms} integrated jitter at 4.5-mW power," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 12, pp. 2745–2758, 2011, doi: 10.1109/JSSC.2011.2162917.

Oscillators 25

Keywords: Crystal model, Pierce, Temperature, Controlled oscillator, Ring osc, Ictrl Rosc, DCO Ring, LCOSC, RCOSC

The world depends on accurate clocks. From the timepiece on your wrist, to the phone in your pocket, they all have a need for an accurate way of counting the passing of time.

Without accurate clocks an accurate GPS location would not be possible. In GPS we even correct for Special and General Relativity to the tune of about +38.6μs/day .

Let's have a look at the most accurate clocks first.

25.1 Atomic clocks . . . 473
25.1.1 Microchip 5071B Cesium Primary Time and Frequency Standard 473
25.1.2 Rubidium standard 474
25.2 Crystal oscillators . 475
25.2.1 Impedance 476
25.2.2 Circuit 478
25.2.3 Temperature behavior 481
25.3 Controlled Oscillators 481
25.3.1 Ring oscillator . . . 481
25.3.2 Capacitive load . . . 483
25.3.3 Realistic 484
25.3.4 Digitally controlled oscillator 486
25.3.5 Differential 486
25.3.6 LC oscillator 487
25.4 Relaxation oscillators 489
25.5 Summary 490
25.6 Would you like to know more? 490
25.6.1 Crystal oscillators . 490
25.6.2 CMOS oscillators . . 491

25.1 Atomic clocks

Cesium standard

The second is defined by taking the fixed numerical value of the cesium frequency Cs, the unperturbed ground-state hyper-fine transition frequency of the cesium 133 atom, to be 9 192 631 770 when expressed in the unit Hz, which is equal to s–1

As a result, by definition, the cesium clocks are exact. That's how the second is defined. When we make a real circuit, however, we never get a perfect, unperturbed system.

25.1.1 Microchip 5071B Cesium Primary Time and Frequency Standard

One example of a ultra precise time piece can be found in Microchip 5071B. The bullets in the list below is from the marketing blurb.

Why would the thing take 30 minutes to start up? Does the temperature need to settle? Is it the loop bandwidth of the PLL that is low? Who knows, but 30 minutes is too long for a IC startup time. And we can't really pack the big box onto a chip.

- < 5E-13 accuracy high-performance models
- Accuracy levels achieved within 30 minutes of startup
- < 8.5E-13 at 100s high-performance models
- < 1E-14 flicker floor high-performance models

Also, when they say

"Ask for a quote" => The price is really high, and we don't want to tell you yet

25.1.2 Rubidium standard

Rubidium standard, use the rubidium hyper-fine transition of 6.8 GHz (6834682610.904 Hz)

and can actually be made quite small. Microsemi's Miniature Atomic Clock (MAC) is a coin-sized rubidium module. According to the marketing blurb:

The MAC is a passive atomic clock, incorporating the interrogation technique of Coherent Population Trapping (CPT) and operating upon the D1 optical resonance of atomic Rubidium Isotope 87.

A rubidium clock is basically a crystal oscillator locked to an atomic reference.

But how do the clocks work? According to Wikipedia, the picture below, is a common way to operate a rubidium clock.

A light passing through the Rubidium gas will be affected if the frequency injected is at the hyper-fine energy levels (E = hf). The change in brightness can be detected by the photo detector, and we can adjust the frequency of the crystal oscillator, we'll see later how that can be done. The crystal oscillator is used as reference for a PLL (frequency synthesizer) to generate the exact frequency needed.

The negative feedback loop ensures that the 5 MHz clock coming out is proportional to the hyper-fine energy levels in the Rubidium atoms. Negative feedback is cool! Especially when we have a pole at DC and infinite gain.

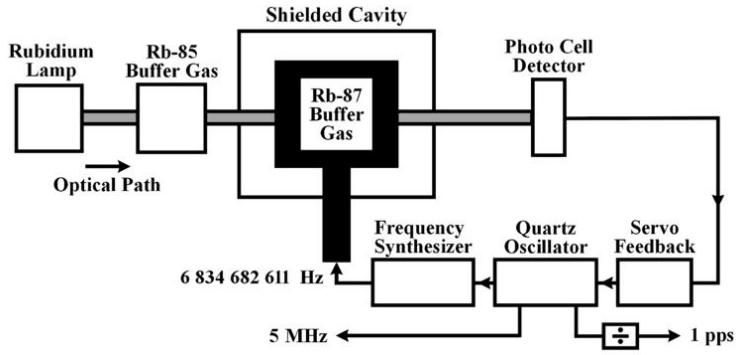


Figure 1: Block diagram of a rubidium clock, where light through a Rb-87 gas cell and a photo detector lock a quartz oscillator to the hyper-fine

transition. Image: Pamela L. Corey, public domain (US federal work), via Wikimedia Commons

25.2 Crystal oscillators

For accuracy's of parts per million, which is sufficient for your wrist watch, or most communication, it's possible to use crystals.

A quartz crystal can resonate at specific frequencies. If we apply a electric field across a crystal, we will induce a vibration in the crystal, which can again affect the electric field. For some history, see Crystal Oscillators

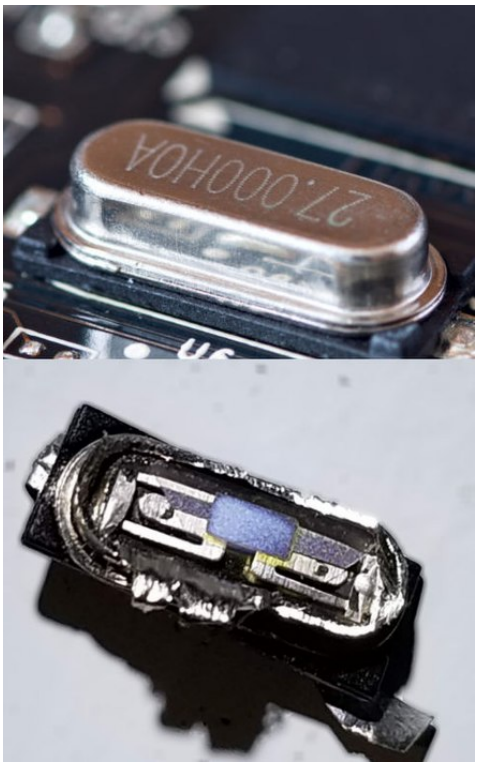


Figure 2: A packaged 27 MHz quartz crystal (top), and an opened package showing the quartz blank (bottom). Image: Chamblis, CC BY-SA 4.0, via Wikimedia Commons

The vibrations in the crystal lattice can have many modes, as illustrated by figure below.

Which mode a blank prefers is decided when it is cut, by the angle of the cut and by the dimensions, and that decision is what sets the frequency range. A blank cut to shear through its thickness lands somewhere between one and a couple of hundred megahertz, which is the crystal you put next to a radio. A blank etched into two tines lands at 32.768 kHz, which is 2^{15} Hz, so fifteen flip-flops divide it down to one pulse a second - that is the crystal in every

watch and every real-time clock. The other modes exist, and are used, but those two are the ones you will meet.

All we need to do with a crystal is to inject sufficient energy to sustain the oscillation, and the resonance of the crystal will ensure we have a correct enough frequency.

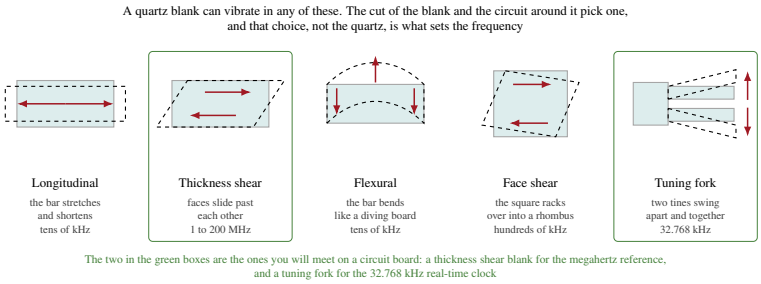


Figure 3: The vibration modes of a quartz blank. Each panel shows the blank at rest, solid, with one extreme of its motion dashed on top and red arrows for the direction the material moves. The mode is not a curiosity: it is what puts a crystal in the tens of kilohertz or in the tens of megahertz

25.2.1 Impedance

The impedance of a crystal is usually modeled as below. A RLC circuit with a parallel capacitor.

Our job is to make a circuit that we can connect to the two pins and provide the energy we will loose due to R_s.

$$R_s + sL + \frac{1}{sC_F}$$

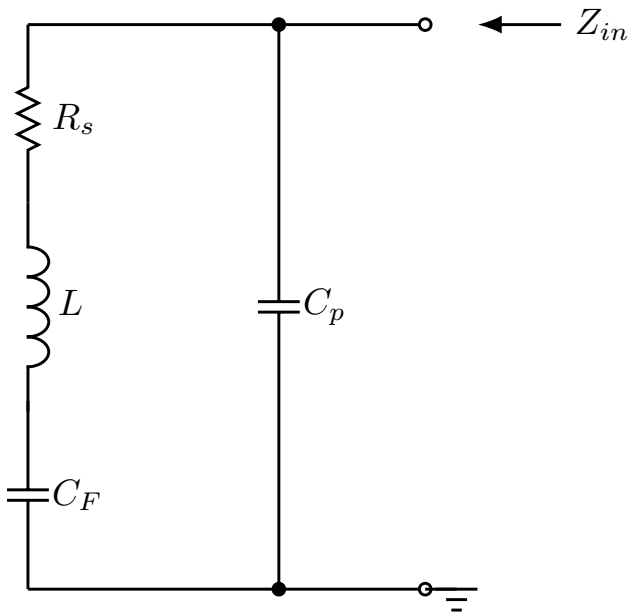


Figure 4: Electrical model of a crystal, a series R_s, L, C_F branch in parallel with the package capacitance C_p

Assuming zero series resistance

$$Z_{in} = \frac{s^2 C_F L + 1}{s^3 C_P L C_F + s C_P + s C_F}$$

Notice that at $s = 0$ the impedance goes to infinity, so a crystal is high impedance at DC.

Divide top and bottom by s and the shape is easier to see:

$$Z_{in} = \frac{1}{s} \cdot \frac{L C_F s^2 + 1}{L C_F C_P s^2 + C_F + C_P}$$

The $1/s$ out front is just the capacitive fall-off that any capacitor has, and it does not change much over the narrow band around resonance, so the fraction beside it is what matters. It has two interesting frequencies. The numerator vanishes at

$$\omega_s = \frac{1}{\sqrt{L C_F}}$$

where the impedance goes to zero: *series resonance*, the motional branch turning into a short. The denominator vanishes a little higher, at

$$\omega_p = \frac{1}{\sqrt{L C_F}} \sqrt{1 + \frac{C_F}{C_P}}$$

where the impedance goes to infinity: *parallel resonance*, the motional branch resonating against the package capacitance. Since C_F is thousands of times smaller than C_P, those two frequencies are only a few hundred parts per million apart, and everything a crystal oscillator does happens in that narrow gap.

See Crystal oscillator impedance for a detailed explanation, or the interactive version where the motional and static elements are sliders and the pulling is worked out for you.

In the impedance plot below we can clearly see that there are two "resonance" points. Usually noted by series and parallel resonance.

I would encourage you to read The Crystal Oscillator [1] for more details.

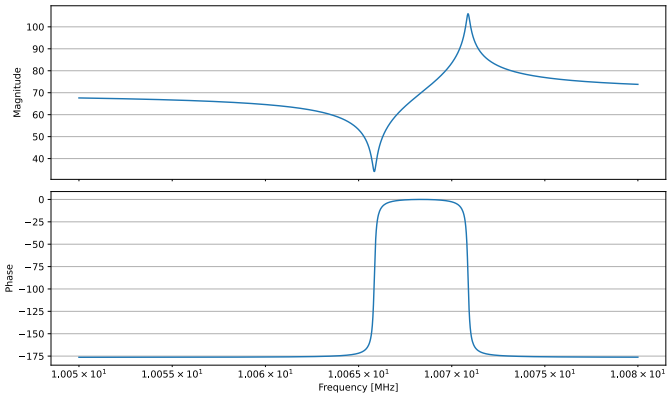


Figure 5: Magnitude and phase of the crystal impedance versus frequency, showing the series and parallel resonance points

25.2.2 Circuit

Below is a common oscillator circuit, a Pierce Oscillator. The crystal is below the dotted line, and the two capacitances are the on-PCB capacitances.

Above the dotted line is what we have inside the IC. Call the left side of the inverter XC1 and right side XC2. The inverter is biased by a resistor, R_1, to keep the XC1 at a reasonable voltage. The XC1 and XC2 will oscillate in opposite directions. As XC1 increases, XC2 will decrease. The R_2 is to model the internal resistance (on-chip wires, bond-wire).

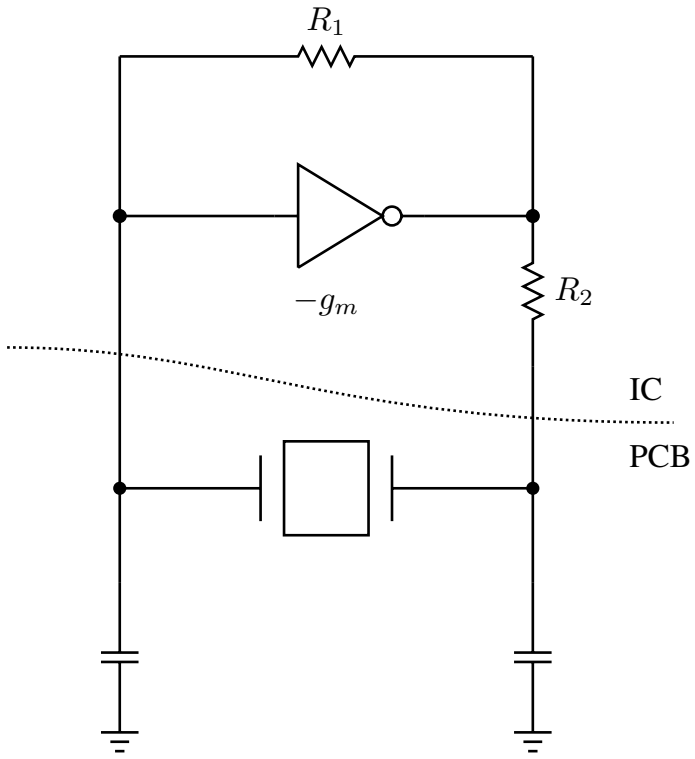


Figure 6: Pierce oscillator, a $-g_m$ amplifier with bias resistor R_1 and series R_2 on the IC, driving the crystal and load capacitors on the PCB

Negative transconductance compensate crystal series resistance

The transconductance of the inverter must compensate for the energy loss caused by R_s in the crystal model. The transconductor also need to be large enough for the oscillation to start, and build up.

I've found that sometimes people get confused by the negative transconductance. There is nothing magical about that. Imagine the PMOS and the NMOS in the inverter, and that the input voltage is exactly the voltage we need for the current in the PMOS and NMOS to be the same. If the current in the PMOS and NMOS is the same, then there can be no current flowing in the output.

Imagine we increase the voltage. The PMOS current would decrease, and the NMOS current would increase. We would pull current from the output.

Imagine we now decrease the voltage instead. The PMOS current would increase, and the NMOS current would decrease. The current in the output would increase.

As such, a negative transconductance is just that as we increase the input voltage, the current into the output decreases, and visa versa.

Long startup time caused by high Q

The Q factor has a few definitions, so it's easy to get confused. Think of Q like this, if a resonator has high Q, then the oscillations die out slowly.

Imagine a perfect world without resistance, and an inductor and capacitor in parallel. Imagine we initially store some voltage across the capacitor, and we let the circuit go. The inductor shorts the plates of the capacitor, and the current in the inductor will build up until the voltage across the capacitor is zero. The inductor still has stored current, and that current does not stop, so the voltage across the capacitor will become negative, and continue decreasing until the inductor current is zero. At that point the negative voltage will flip the current in the inductor, and we go back again.

The LC circuit will resonate back and forth. If there was no resistance in the circuit, then the oscillation would never die out. The system would be infinite Q.

The Q of the crystal oscillator can be described as $Q = 1/(\omega R_s C_f)$, assuming some common values of $R_s = 50$, $C_f = 5e^{-15}$ and $\omega = 2\pi \times 32$ MHz then $Q \approx 20$ k.

That number may not tell you much, but think of it like this, it will take 20 000 clock cycles before the amplitude falls by 1/e. For example, if the amplitude of oscillation was 1 V, and you stop introducing energy into the system, then 20 000 clock cycles later, or 0.6 ms, the amplitude would be 0.37 V.

The same is roughly true for startup of the oscillator. If the crystal had almost no amplitude, then an increase of a factor e would take 20 k cycles. Increasing the amplitude of the crystal to 1 V could take milliseconds.

Most circuits on-chip have startup times on the order of microseconds, while crystal oscillators have startup time on the order of milliseconds. As such, for low power IoT, the startup time of crystal oscillators, or indeed keeping the oscillator running at a really low current, are key research topics.

Can fine tune frequency with parasitic capacitance

The resonance frequency of the crystal oscillator can be modified by the parasitic capacitance from XC1 and XC2 to ground. The tunability of crystals is usually in ppm/pF. Sometimes micro-controller vendors will include internal load capacitances to support multiple crystal vendors without changing the PCB.

25.2.3 Temperature behavior

One of the key reasons for using crystals is their stability over temperature. Below is a plot of a typical temperature behavior. The cutting angle of the crystal affect the temperature behavior, as such, the closer crystals are to "no change in frequency over temperature", the more expensive they become.

In communication standards, like Bluetooth Low Energy, it's common to specify timing accuracy's of +- 50 ppm. Have a look in the Bluetooth Core Specification 5.4 Volume 6, Part A, Chapter 3.1 (page 2653) for details.

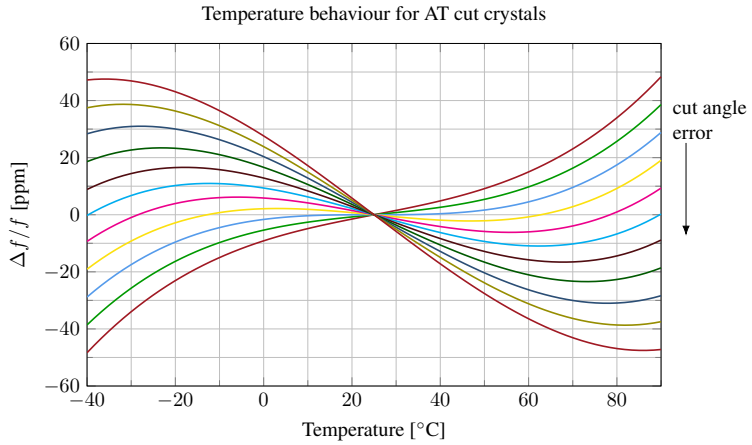


Figure 7: Frequency deviation in ppm versus temperature for AT-cut crystals with different cutting angles

25.3 Controlled Oscillators

On an integrated circuit we may need multiple clocks, and we can't have crystal oscillators for all of them. We can use frequency locked loops, phase locked loops and delay locked loops to make multiples of the crystal reference frequency.

All phase locked loops contain an oscillator where we control the frequency of oscillation.

25.3.1 Ring oscillator

The simplest oscillator is a series of inverters biting their own tail, a ring oscillator.

The delay of each stage can be thought of as a RC time constant, where the R is the transconductance of the inverter, and the C is the gate capacitance of the next inverter.

$$t_{pd} \approx RC$$

$$R \approx \frac{1}{gm} \approx \frac{1}{\mu_n C_{ox} \frac{W}{L}(VDD - V_{th})}$$

$$C \approx \frac{2}{3} C_{ox} WL$$

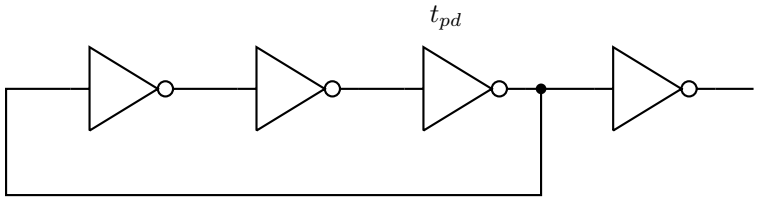


Figure 8: Ring oscillator, a loop of inverters where each stage adds a delay t_{pd}

One way to change the oscillation frequency is to change the VDD of the ring oscillator. Based on the delay of a single inverter we can make an estimate of the oscillator gain. How large change in frequency do we get for a change in VDD.

$$t_{pd} \approx \frac{2/3 C_{ox} WL}{\frac{W}{L}\mu_n C_{ox}(VDD - V_{th})}$$

$$f = \frac{1}{2N t_{pd}} = \frac{\mu_n (VDD - V_{th})}{\frac{4}{3} NL^2}$$

$$K_{vco} = 2\pi \frac{\partial f}{\partial VDD} = \frac{2\pi \mu_n}{\frac{4}{3} NL^2}$$

The K_{vco} is proportional to mobility, and inversely proportional to the number of stages and the length of the transistor squared. In most PLLs we don't want the K_{vco} to be too large. Ideally we want the ring oscillator to oscillate close to the frequency we want, i.e 512 MHz, and a small K_{vco} to account for variation over temperature (mobility of transistors decreases with increased temperature, the threshold voltage of transistors decrease with temperature), and changes in VDD.

To reduce the K_{vco} of the standard ring oscillator we can increase the gate length, and increase the number of stages.

I think it's a good idea to always have a prime number of stages in the ring oscillator. I have seen some ring oscillators with 21 stages oscillate at 3 times the frequency in measurement. Since $21 = 7 \times 3$ it's possible to have three "waves" traveling through

the ring oscillator at all times, forever. If you use a prime number of stages, then sustained oscillation at other frequencies cannot happen.

As such, the number of inverter stages should be $\in [3, 5, 7, 11, 13, 17, 19, 23, 29, 31]$

25.3.2 Capacitive load

The oscillation frequency of the ring oscillator can also be changed by adding capacitance.

$$f = \frac{\mu_n C_{ox} \frac{W}{L} (VDD - V_{th})}{2N \left(\frac{2}{3} C_{ox} WL + C\right)}$$

$$K_{vco} = \frac{2\pi \mu_n C_{ox} \frac{W}{L}}{2N \left(\frac{2}{3} C_{ox} WL + C\right)}$$

Assume that the extra capacitance is much larger than the gate capacitance, then

$$f = \frac{\mu_n C_{ox} \frac{W}{L} (VDD - V_{th})}{2NC}$$

$$K_{vco} = \frac{2\pi \mu_n C_{ox} \frac{W}{L}}{2NC}$$

And maybe we could make the K_{vco} relatively small.

The power consumption of an oscillator, however, will be similar to a digital circuit of $P = C \times f \times VDD^2$, so increasing capacitance will also increase the power consumption.

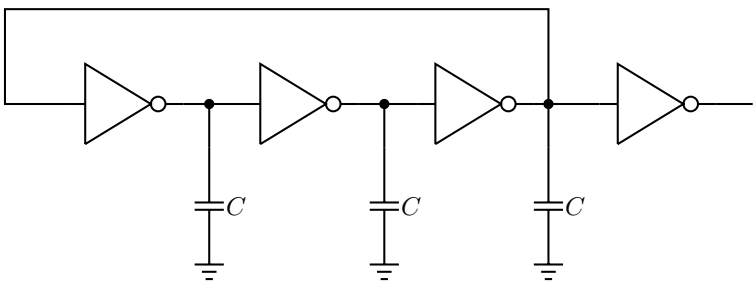


Figure 9: Ring oscillator with an added capacitive load C on each inverter output

25.3.3 Realistic

Assume you wanted to design a phase-locked loop, what type of oscillator should you try first? If the noise of the clock is not too important, so you don't need an LC-oscillator, then I'd try the oscillator below, although I'd expand the number of stages to fit the frequency.

The circuit has a capacitance loaded ring oscillator fed by a current. The $I_{control}$ will give a coarse control of the frequency, while the $V_{control}$ can give a more precise control of the frequency.

Since the $V_{control}$ can only increase the frequency it's important that the $I_{control}$ is set such that the frequency is below the target.

Most PLLs will include some form of self calibration at startup. At startup the PLL will do a coarse calibration to find a sweet-spot for $I_{control}$, and then use $V_{control}$ to do fine tuning.

Since PLLs always have a reference frequency, and a phase and frequency detector, it's possible to sweep the calibration word for $I_{control}$ and then check whether the output frequency is above or below the target based on the phase and frequency detector output. Although we don't know exactly what the oscillator frequency is, we can know the frequency close enough.

It's also possible to run a counter on the output frequency of the VCO, and count the edges between two reference clocks. That way we can get a precise estimate of the oscillation frequency.

Another advantage with the architecture below is that we have some immunity towards supply noise. If we decouple both the current mirror, and the $V_{control}$ towards VDD, then any change to VDD will not affect the current into the ring oscillator.

Maybe a small side track, but inject a signal into an oscillator from an amplifier, the oscillator will have a tendency to lock to the injected signal, we call this "injection locking", and it's common to do in ultra high frequency oscillators (60 - 160 GHz). Assume we allow the PLL to find the right $V_{control}$ that corresponds to the injected frequency. Assume that the injected frequency changes, for example frequency shift keying (two frequencies that mean 1 or 0), as in Bluetooth Low Energy. The PLL will vary the $V_{control}$ of the PLL to match the frequency change of the injected signal, as such, the $V_{control}$ is now the demodulated frequency change.

Still today, there are radio receivers that use a PLLs to directly demodulate the incoming frequency shift keyed modulated carrier.

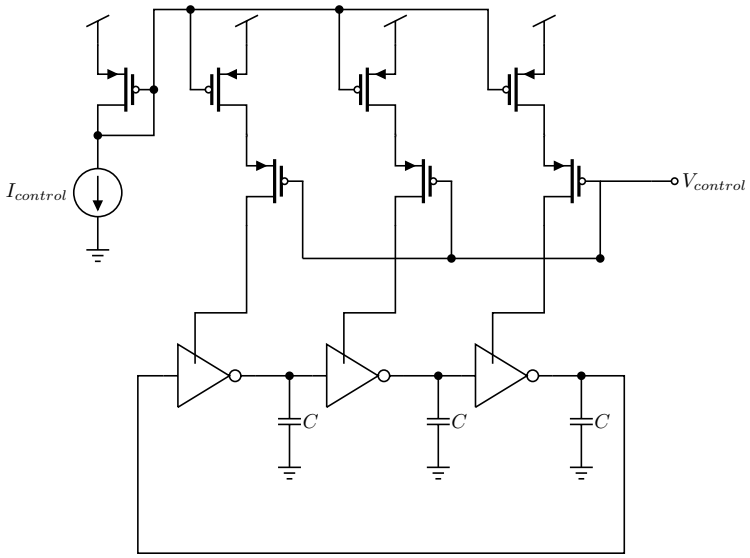


Figure 10: Current-starved, capacitance loaded ring oscillator with coarse frequency control from the $I_{control}$ mirror and fine control from the $V_{control}$ transistors

We can calculate the K_{vco} of the oscillator as shown below. The inverters mostly act as switches, and when the PMOS is on, then the rise time is controlled by the PMOS current mirror, the additional $V_{control}$ and the capacitor. For the calculation below we assume that the pull-down of the capacitor by the NMOS does not affect the frequency much.

The advantage with the above ring-oscillator is that we can control the frequency of oscillation with $I_{control}$ and have a independent K_{vco} based on the sizing of the $V_{control}$ transistors.

$$I = C\frac{dV}{dt}$$

$$f \approx \frac{I_{control} + \frac{1}{2}\mu_p C_{ox}\frac{W}{L}(VDD - V_{control} - V_{th})^2}{C\frac{VDD}{2}N}$$

$$K_{vco} = 2\pi\frac{\partial f}{\partial V_{control}}$$

$$K_{vco} = -2\pi\frac{\mu_p C_{ox}\frac{W}{L}(VDD - V_{control} - V_{th})}{C\frac{VDD}{2}N}$$

Two things about that result. It is negative, because $V_{control}$ is the gate of a PMOS: raising it turns the device off and slows the oscillator down. And it is proportional to the overdrive $VDD - V_{control} - V_{th}$, not constant, so this oscillator's gain depends on where in its range you are sitting.

That is worth knowing before designing a loop around it. The linear model in the PLL chapter takes K_{osc} as a single number, and it is only a single number over the small range the loop actually uses. A quick check on the units catches the mistake of dropping the overdrive term: $\mu_p C_{ox} W/L$ is amps per volt squared, so without a voltage on top the expression is not rad/s per volt.

25.3.4 Digitally controlled oscillator

We can digitally control the oscillator frequency as shown below by adding capacitors.

Today there are all digital loops where the oscillator is not really a "voltage controlled oscillator", but rather a "digital control oscillator". DCOs are common in all-digital PLLs.

Another reason to use digital frequency control is to compensate for process variation. We know that mobility affects the K_{vco}, as such, for fast transistors the frequency can go up. We could measure the free-running frequency in production, and compensate with a digital control word.

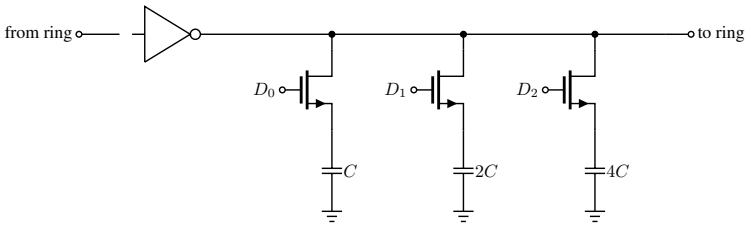


Figure 11: Digital frequency control of a ring oscillator stage with binary weighted capacitors (C, $2C$, $4C$) switched by bits D_0 to D_2

25.3.5 Differential

Differential circuits are potentially less sensitive to supply noise

Imagine a single ended ring oscillator. If I inject a voltage onto the input of one of the inverters that was just about to flip, I can either delay the flip, or speed up the flip, depending on whether the voltage pulse increases or decreases the input voltage for a while. Such voltage pulses will lead to jitter.

Imagine the same scenario on a differential oscillator (think diff pair). As long as the voltage pulse is the same for both inputs, then no change will incur. I may change the current slightly, but that depends on the tail current source.

Another cool thing about differential circuits is that it's easy to multiply by -1, just flip the wires, as a result, I can use a 2 stage ring differential ring oscillator.

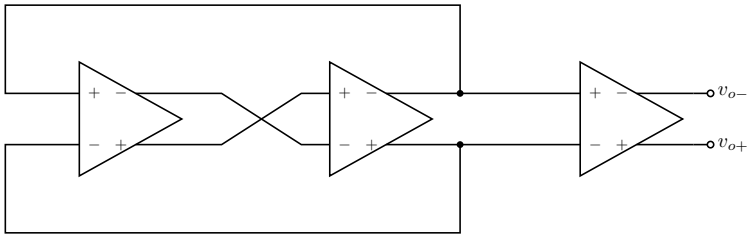


Figure 12: Differential ring oscillator, where a wire crossing provides the multiply by -1

25.3.6 LC oscillator

Most radio's are based on modulating information on-to a carrier frequency, for example 2.402 GHz for a Bluetooth Low Energy Advertiser. One of the key properties of the carrier waves is that it must be "clean". We're adding a modulated signal on top of the carrier, so if there is noise inherent on the carrier, then we add noise to our modulation signal, which is bad.

Most ring oscillators are too high noise for radio's, we must use a inductor and capacitor to create the resonator.

Inductors are huge components on a IC. Take a look at the nRF51822 below, the two round inductors are easily identifiable. Actually, based on the die image we can guess that there are two oscillators in the nRF51822. Maybe it's a multiple conversion superheterodyne receiver

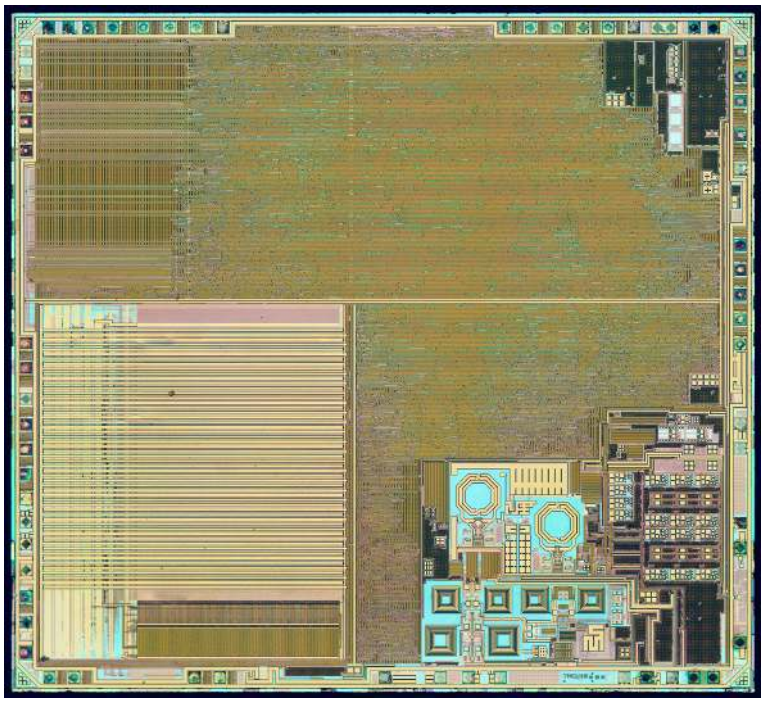


Figure 13: Die photograph of the nRF51822, where the two round LC oscillator inductors are easily identifiable. Die photograph by zeptobars.com, CC BY 3.0

Below is a typical LC oscillator. The main resonance is set by the L and C, while the tunability is provided by a varactor, a voltage variable capacitor. Or with less fancy words, the gate capacitance of a transistor, since the gate capacitance of a transistor depends on the effective voltage, and is thus a "varactor"

The NMOS at the bottom provide the "negative transconductance" to compensate for the loss in the LC tank.

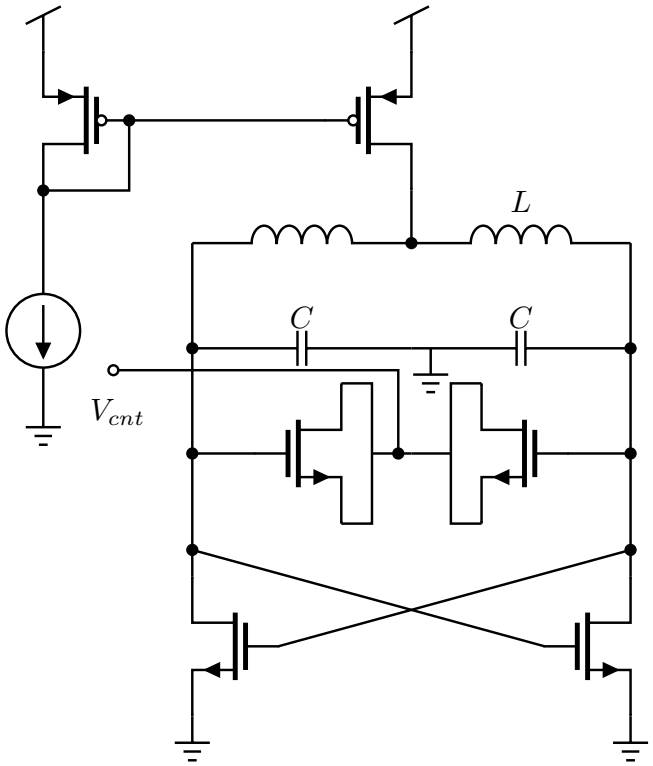


Figure 14: LC oscillator with current mirror bias, LC tank, varactor tuning (V_{cnt}) and a cross-coupled NMOS pair providing the negative transconductance

$$f \propto \frac{1}{\sqrt{LC}}$$

25.4 Relaxation oscillators

A last common oscillator is the relaxation oscillator, or "RC" oscillator. By now you should be proficient enough to work through the equations below, and understand how the circuit works. If not, ask me.

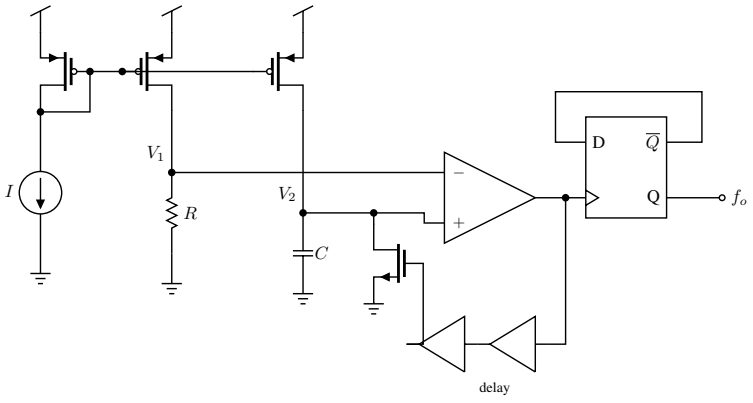


Figure 15: Relaxation (RC) oscillator, where a comparator and flip-flop toggle as the capacitor voltage V_2 charges to the threshold $V_1 = IR$

$$V_1 = IR$$

$$I = C\frac{dV}{dt}$$

$$dt = \frac{CV_2}{I} = \frac{CIR}{I}$$

$$f = \frac{1}{dt} = \frac{1}{RC}$$

$$f_o = \frac{1}{2}f = \frac{1}{2RC}$$

25.5 Summary

The one-page version of this chapter:

- The precision ladder: atomic clocks, then crystals (ppm), then LC (phase-noise kings on chip), then rings, then RC relaxation - each rung cheaper and noisier
- A crystal is a mechanical resonator with Q in the tens of thousands; the Pierce circuit keeps it ringing with one inverter
- Ring oscillators are small, tune over decades, and follow every millivolt of supply - which is why the PLL supply-controls one on purpose
- Current starving and capacitive load make the ring controllable; the varactor does the same for the LC tank
- The relaxation oscillator charges C to IR and resets: the cheap always-on clock for waking things up
- An oscillator's frequency stability over temperature and supply, not its schematic, decides where it may be used

25.6 Would you like to know more?

25.6.1 Crystal oscillators

The Crystal Oscillator - A Circuit for All Seasons [1]

High-performance crystal oscillator circuits: theory and application [2]

Ultra-low Power 32kHz Crystal Oscillators: Fundamentals and Design Techniques [3]

A Sub-nW Single-Supply 32-kHz Sub-Harmonic Pulse Injection Crystal Oscillator [4]

25.6.2 CMOS oscillators

The Ring Oscillator - A Circuit for All Seasons [5]

A Study of Phase Noise in CMOS Oscillators [6]

An Ultra-Low-Noise Swing-Boosted Differential Relaxation Oscillator in 0.18-um CMOS [7]

Ultra Low Power Frequency Synthesizer

[1] B. Razavi, "The crystal oscillator [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 9, no. 2, pp. 7–9, 2017, doi: 10.1109/MSSC.2017.2688679.

[2] E. Vittoz, M. Degrauwe, and S. Bitz, "High-performance crystal oscillator circuits: Theory and application," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 3, pp. 774–783, 1988, doi: 10.1109/4.318.

[3] L. Xu, D. Blaauw, and D. Sylvester, "Ultra-low power 32kHz crystal oscillators: Fundamentals and design techniques," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 1, pp. 79–93, 2021, doi: 10.1109/OJSSCS.2021.3113889.

[4] K.-M. Kim, S. Kim, K.-S. Choi, H. Jung, J. Ko, and S.-G. Lee, "A sub-nW single-supply 32-kHz sub-harmonic pulse injection crystal oscillator," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 6, pp. 1849–1858, 2021, doi: 10.1109/JSSC.2020.3016021.

[5] B. Razavi, "The ring oscillator [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 4, pp. 10–81, 2019, doi: 10.1109/MSSC.2019.2939771.

[6] B. Razavi, "A study of phase noise in CMOS oscillators," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 3, pp. 331–343, 1996, doi: 10.1109/4.494195.

[7] J. Lee, A. K. George, and M. Je, "An ultra-low-noise swing-boosted differential relaxation oscillator in 0.18-μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 9, pp. 2489–2497, 2020, doi: 10.1109/JSSC.2020.2987681.

26 Low Power Radio

Keywords: Range, Antenna Size, Modulation, OFDM, GFSK, pi/4-qpsk, 8-psk, 16 QAM, Bluetooth LE, LP RX, LNA, Mixer, AAF, ADC, BB

Radio's are all around us. In our phone, on our wrist, in our house, there is Bluetooth, WiFi, Zigbee, LTE, GPS and many more.

A radio is a device that receives and transmits light encoded with information. The frequency of the light depends on the standard. How the information is encoded onto the light depends on the standard.

Assume that we did not know any standards, what would we do if we wanted to make the best radio IC for gaming mice?

There are a few key concepts we would have to know before we decide on a radio type: Data Rate, Carrier Frequency and range, and the power supply.

26.1 Data Rate

26.1.1 Data

A mouse reports on the relative X and Y displacement of the mouse as a function of time. A mouse has buttons. There can be many mice in a room, as such, they must have an address , so PCs can tell them apart.

A mouse must be low-power. As such, the radio cannot be on all the time. The radio must start up and be ready to receive quickly.

We don't know how far away from the PC the mice might be, as such, we don't know the dB loss in the communication channel. As a result, the radio needs to have a high dynamic range, from weak signals to strong signals. In order for the radio to adjust the gain of the receiver we should include a pre-amble, a known sequence, for example 01010101, such that the radio can adjust the gain, and also, recover the symbol timing.

All in all, the packets we send from the mouse may need to have the following bits.

What	Bits	Why
X displacement	8	
Y displacement	8	

26.1 Data Rate 493
26.1.1 Data 493
26.1.2 Rate 494
26.1.3 Data Rate 494
26.2 Carrier Frequency & Range 494
26.2.1 ISM (industrial, scientific and medical) bands 494
26.2.2 Antenna 495
26.2.3 Range (Friis) 496
26.2.4 Range (Free space) . 496
26.2.5 Range (Real world) 497
26.3 Power supply 497
26.3.1 Battery 497
26.4 Decisions 498
26.4.1 Modulation 498
26.4.2 BPSK 499
26.4.3 Single carrier, or multi carrier? 505
26.4.4 Use a Software Defined Radio . . . 506
26.5 Bluetooth 507
26.5.1 Bluetooth Basic Rate/Extended Data rate 508
26.5.2 Bluetooth Low Energy 508
26.6 Algorithm to design state-of-the-art LE radio 510
26.6.1 LNTA 511
26.6.2 MIXER 512
26.6.3 AAF 513
26.6.4 ADC 514
26.6.5 AD-PLL 516
26.6.6 Baseband 517
26.7 What do we really want, in the end? . 517
26.8 Summary 518
26.9 Would you like to know more? 519

What	Bits	Why
CRC	4	Bit errors
Buttons	16	One-hot coding. Most mice have buttons
Preamble	8	Synchronization
Address	32	Unique identifier
Total	76	

26.1.2 Rate

Gamers are crazy for speed, they care about milliseconds. So our mice needs to be able to send and receive data quite often.

Assume 1 ms update rate

26.1.3 Data Rate

To compute the data rate, let's do a back of the envelope estimate of the data, and the rate.

```
Application Data Rate > 76 bits/ms = 76 kbps

Assume 30 % packet loss

Raw Data Rate > 228 kbps

Multiply by 3.14 > 716 kbps

Round to nearest nice number = 1Mbps
```

The above statements are a exact copy of what happens in industry when we start design of something. We make an educated guess and multiply by a number. More optimistic people would multiply with e.

26.2 Carrier Frequency & Range

26.2.1 ISM (industrial, scientific and medical) bands

There are rules and regulations that prevent us from transmitting and receiving at any frequency we want. We need to pick one of the ISM bands, or we need to get a license from governments around the world.

For the ISM bands, there are regions, as seen below.

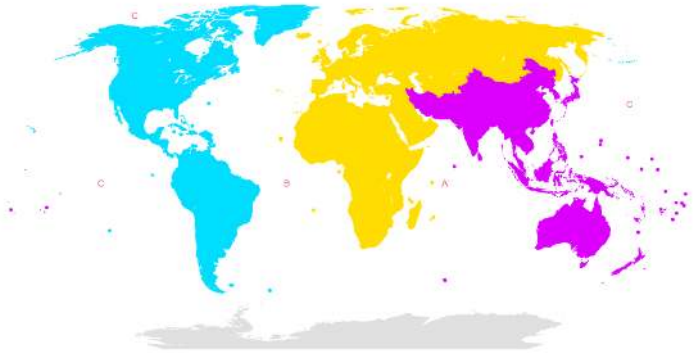


Figure 1: ITU regions that set the ISM band allocations: region 1 (yellow), region 2 (blue), region 3 (pink). Image: Maximilian Doerrbecker (Chumwa), CC BY-SA 2.5, via Wikimedia Commons

- Yellow: Region 1
- Blue: Region 2
- Pink: Region 3

Below is a table of the available frequencies, but how should we pick which one to use? There are at least two criteria that should be investigated. Antenna and Range.

Flow	Fhigh	Bandwidth	Description
40.66 MHz	40.7 MHz	40 kHz	Worldwide
433.05 MHz	434.79 MHz	1.74 MHz	Region 1
902 MHz	928 MHz	26 MHz	Region 2
2.4 GHz	2.5 GHz	100 MHz	Worldwide
5.725 GHz	5.875 GHz	150 MHz	Worldwide
24 GHz	24.25 GHz	250 MHz	Worldwide
61 GHz	61.5 GHz	500 MHz	Subject to local acceptance

26.2.2 Antenna

For a mouse we want to hold in our hand, there is a size limit to the antenna. There are many types of antenna, but

assume wavelength/4 is an OK antenna size (wavelength = lightspeed/frequency)

The below table shows the ISM band and the size of a quarter wavelength antenna. Any frequency above 2.4 GHz may be OK from a size perspective.

ISM band	$\lambda/4$	Unit	OK/NOK
40.68 MHz	1.8	m	:x:
433.92 MHz	17	cm	:x:
915 MHz	8.2	cm	
2450 MHz	3.06	cm	:white_check_mark:
5800 MHz	1.29	cm	:white_check_mark:
24.125 GHz	3.1	mm	:white_check_mark:
61.25 GHz	1.2	mm	:white_check_mark:

26.2.3 Range (Friis)

One of the worst questions a radio designer can get is "What is the range of your radio?", especially if the people asking are those that don't understand physics, or the real world. The answer to the question is incredibly complicated, as it depends on exactly what is between two devices talking.

If we assume, however, that there is only free space, and no real reflections from anywhere, then we can make an estimate of the range.

Assume no antenna gain, power density p at distance D is

$$p = \frac{P_{TX}}{4\pi D^2}$$

Assume receiver antenna has no gain, then the effective aperture is

$$A_e = \frac{\lambda^2}{4\pi}$$

Power received is then

$$P_{RX} = \frac{P_{TX}}{D^2}\left[\frac{\lambda}{4\pi}\right]^2$$

Or in terms of distance

$$D = 10^{\frac{P_{TX} - P_{RX} + 20log_{10}\left(\frac{c}{4\pi f}\right)}{20}}$$

26.2.4 Range (Free space)

If we take the ideal equation above, and use some realistic numbers for TX and RX power, we can estimate a range.

Assume TX = 0 dBm, assume RX sensitivity is -80 dBm

Freq	$20log_{10}(c/4\pi f)$ [dB]	D [m]	OK/NOK
915 MHz	-31.7	260.9	:white_check_mark:
2.45 GHz	**-40.2**	**97.4**	:white_check_mark:
5.80 GHz	-47.7	41.2	:white_check_mark:
24.12 GHz	-60.1	9.9	:x:
61.25 GHz	-68.2	3.9	:x:
160 GHz	-76.52	1.5	:x:

26.2.5 Range (Real world)

In the real world, however, the

path loss factor,

$$n \in [1.6, 6]$$

,

$$D = 10^{\frac{P_{TX} - P_{RX} + 20log_{10}\left(\frac{c}{4\pi f}\right)}{n \times 10}}$$

So the real world range of a radio can vary more than an order of magnitude. Still, 2.4 GHz seems like a good choice for a mouse.

$20log_{10}(c/4\pi f)$

Freq	[dB]	D@n=2 [m]	D@n=6 [m]	OK/NOK
2.45 GHz	**-40.2**	**97.4**	**4.6**	:white_-check_mark:
5.80 GHz	-47.7	41.2	3.45	:white_-check_mark:
24.12 GHz	-60.1	9.9	2.1	:x:

26.3 Power supply

We could have a wired mouse for power, but that's boring. Why would we want a wired mouse to have wireless communication? It must be powered by a battery, but what type of battery?

There exists a bible of batteries, Linden's Handbook of Batteries. It's worth a read if you want to dive deeper into chemistry and properties of primary (non-chargeable) and secondary (chargeable) cells.

26.3.1 Battery

Mouse is maybe AA, 3000 mAh

Cell	Chemistry	Voltage (V)	Capacity (Ah)
AA	LiFeS2	1.0 - 1.8	3
2xAA	LiFeS2	2.0 - 3.6	3
AA	Zn/Alk/MnO2	0.8 - 1.6	3
2xAA	Zn/Alk/MnO2	1.6 - 3.2	3

26.4 Decisions

Now we know that we need a 1 Mbps radio at 2.4 GHz that runs off a 1.0 V - 1.8 V or 2.0 V - 3.6 V supply.

Next we need to decide what modulation scheme we want for our light. How should we encode the bits onto the 2.4 GHz carrier wave?

26.4.1 Modulation

Any modulation can be described by the function below.

$$A_m(t) \times \cos\left(2\pi \int_0^t f_{carrier}(\tau) d\tau + \phi_m(t)\right)$$

The integral matters as soon as the carrier frequency is one of the things being modulated, which for GFSK it is. Writing $2\pi f(t)t$ instead is a tempting shorthand and it is wrong: differentiate it and the instantaneous frequency comes out as $f(t) + t\,f'(t)$, so the error grows without bound as t does. Phase is the integral of frequency, always. For a fixed carrier the integral collapses to the familiar $2\pi f_c t$ and no harm is done, which is why the shorthand survives.

The amplitude of the carrier can be modulated, or the phase of the carrier.

People have been creative over the last 50 years in terms of encoding bits onto carriers. Below is a small excerpt of some common schemes.

Scheme	Acronym	Pro	Con
Binary phase shift keying	BPSK	Simple	Not constant envelope
Quadrature phase-shift keying	QPSK	2bits/symbol	Not constant envelope
Offset QPSK	OQPSK	2bits/symbol	Constant envelope with half-sine pulse shaping
Gaussian Frequency Shift Keying	GFSK	1 bit/symbol	Constant envelope
Quadrature amplitude modulation	QAM	> 10 bits/symbol	Really non-constant envelope

26.4.2 BPSK

In binary phase shift keying the 1 and 0 is encoded in the phase change. Change the phase 180 degrees and we've transitioned from a 0 to a 1. Do another 180 degrees and we're back to where we were.

It's common to show modulation schemes in a constellation diagram with the real axis and the complex axis. For the real signal we send, the phase and amplitude are usually both real quantities.

I say usually, because in quantum mechanics, and the time evolution of a particle, the amplitude of the wave function is actually a complex variable. As such, nature is actually complex at the most fundamental level.

But for now, let's keep it real in the real world.

Still, the maths is much more elegant in the complex plane.

The equation for the unit circle is $y = e^{i(\omega t+\phi)}$ where ϕ is the phase, and ω is the angular frequency.

Imagine we spin a bike wheel around at a constant frequency (constant ω), on the bike wheel there is a red dot. If you keep your eyes open all the time, then the red dot would go round and round. But imagine that you only opened your eyes every second for a brief moment to see where the dot was. Sometimes it could be on the right side, sometimes on the left side. If our "eye opening rate", or your sample rate, matched how fast the "wheel rotator" changed the location of the dot, then you could receive information.

Now imagine you have a strobe light matched to the "normal" carrier frequency. If one rotation of the wheel matched the frequency of the strobe light, then the red dot would stay in exactly the same place. If the wheel rotation was slightly faster, then the red dot would move one way around the circle at every strobe. If the wheel rotation was slightly slower, the red dot would move the other way around the circle.

That's exactly how we can change the position in the constellation. We increase the carrier frequency for a bit to rotate 180 degrees, and we can decrease the frequency to go back 180 degrees. In this example the dot would move around the unit circle, and the amplitude of the carrier can stay constant.

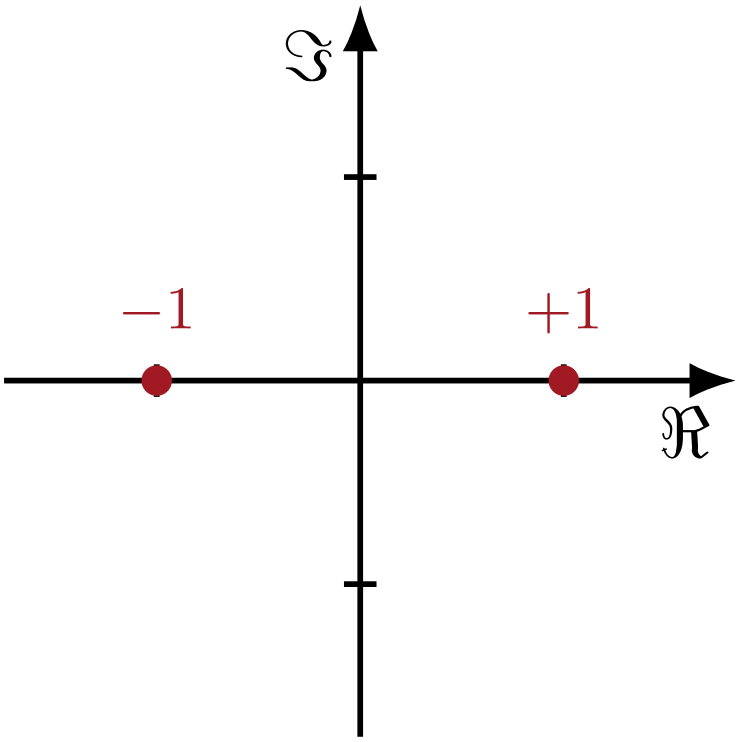


Figure 2: BPSK constellation: the two symbols sit on the real axis, 180 degrees apart

There is another way to change phase 180 degrees, and that's simply to swap the phase in the transmitter circuit. Imagine as below we have a local oscillator driving pseudo differential common source stages with switches on top. If we flip the switches we can change the phase 180 degrees pretty fast.

A challenge is, however, that the amplitude will change. In general, constant envelope (don't change amplitude) modulation is less bandwidth efficient (slower) than schemes that change both phase and amplitude.

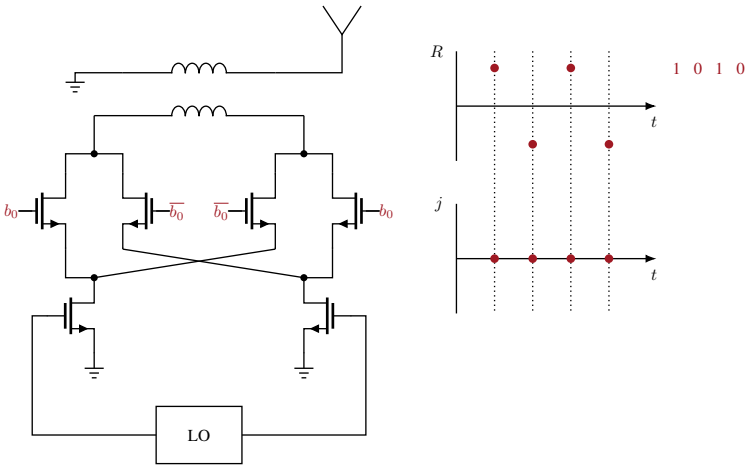


Figure 3: BPSK transmitter: a local oscillator drives a pseudo-differential common source pair, and the b_0 switches swap the output phase 180

degrees into the antenna balun

Standards like Zigbee used offset quadrature phase shift keying, with a constellation as shown below. With 4 points we can send 2 bits per symbol.

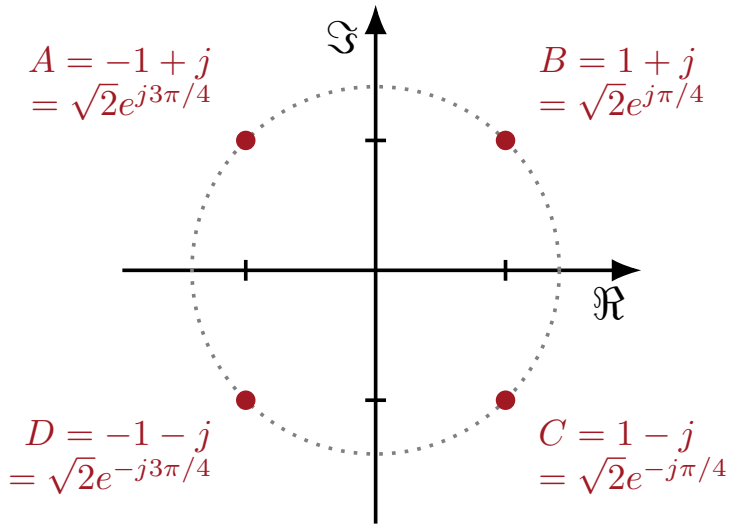


Figure 4: QPSK constellation: four symbols at $\pm 1 \pm j$*, or* $\sqrt{2}e^{\pm j\pi/4}$*, so 2 bits per symbol*

In ZigBee, or 802.15.4 as the standard is called, the phase changes is actually done with a constant envelope.

The nice thing about constant envelope is that the radio transmitter can be simple. We don't need to change the amplitude. If we have a PLL as a local oscillator, where we can change the phase (or frequency), then we only need a power amplifier before the antenna.

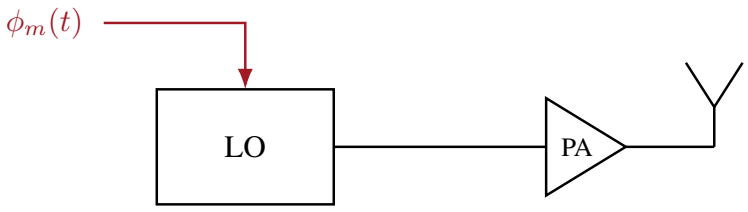


Figure 5: Constant envelope transmitter: the phase is modulated in the local oscillator, and a power amplifier drives the antenna

For phase and amplitude modulation, or complex transmitters, we need a way to change the amplitude and phase. What a shocker. There are two ways to do that. A polar architecture where phase change is done in the PLL, and amplitude in the power amplifier.

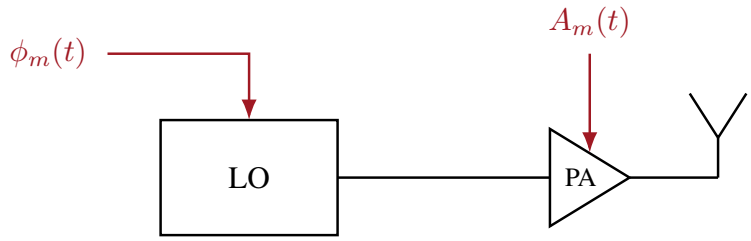


Figure 6: Polar transmitter: phase ϕ is modulated in the local oscillator and amplitude A in the power amplifier

Or a Cartesian architecture where we make the in-phase component, and quadrature-phase components in digital, then use two digital to analog converters, and a set of complex mixers to encode onto the carrier. The power amplifier would not need to change the amplitude, but it does need to be linear.

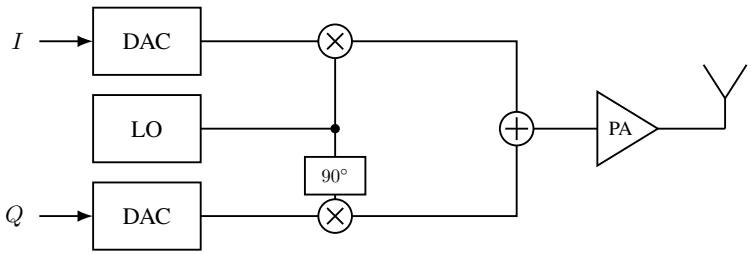


Figure 7: Cartesian transmitter. Two converters produce I and Q, two mixers multiply them by copies of the local oscillator ninety degrees apart, and the sum drives a linear power amplifier. The 90° block is the whole difference between this and two copies of the same branch: without it both mixers would be multiplying by the same carrier, and the sum would carry no phase information.

We can continue to add constellation points around the unit circle. Below we can see 8-PSK, where we can send 3-bits per symbol. Assuming we could shift position between the constellation points at a fixed rate, i.e 1 mega symbols per second. With 1-bit per symbol we'd get 1 Mbps. With 3-bits per symbol we'd get 3 Mbps.

We could add 16 points, 32 points and so on to the unit circle, however, there is always noise in the transmitter, which will create a cloud around each constellation point, and it's harder and harder to distinguish the points from each other.

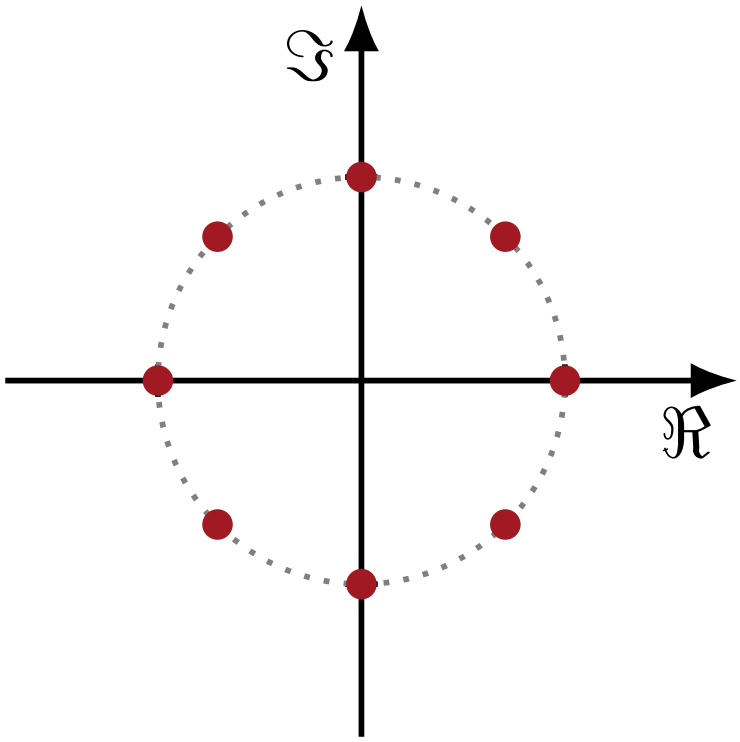


Figure 8: 8-PSK constellation: eight points on the unit circle, so 3 bits per symbol

Bluetooth Classic uses $\pi/4$-DQPSK and 8DPSK.

DPSK means differential phase shift keying. Think about DPSK like this. In the QPSK diagram above the symbols (00,01,10,11) are determined by the constellation point $1 + j$, $1 - j$ and so on. What would happen if the constellation rotated slowly? Would $1 + j$ turn into $1 - j$ at some point? That might screw up our decoding if the received constellation point was at $1 + 0j$, we would not know what it was.

If we encoded the symbols as a change in phase instead (differential), then it would not matter if the constellation rotated slowly. A change from $1 + j$ to $1 - j$ would still be 90 degrees.

Why would the constellation rotate you ask? Imagine the transmitter transmits at 2 400 000 000 Hz. How does our receiver generate the same frequency? We need a reference and a PLL. The crystal-oscillator reference has a variation of +-50 ppm, so $2.4e9 \times 50/1e6 = 120$ kHz.

Assume our receiver local oscillator was at 2 400 120 000 Hz. The transmitter sends 2 400 000 000 Hz + modulation. At the receiver we multiply with our local oscillator, and if you remember your math, multiplication of two sine creates a sum and a difference between the two frequencies. As such, the low frequency part (the difference between the frequencies) would be 120 kHz + modulation. As a result, our constellation would rotate 120 000 times per second.

Assuming a symbol rate of 1MS/s our constellation would rotate roughly 1/10 of the way each symbol.

In DPSK the rotation is not that important. In PSK we have to measure the carrier offset, and continuously de-rotate the constellation.

Most radios will de-rotate somewhat based on the preamble, for example in Bluetooth Low Energy there is an initial 10101010 sequence that we can use to estimate the offset between TX and RX carriers, or the frequency offset.

The $\pi/4$ part of $\pi/4 - DQPSK$ just means we actively rotate the constellation 45 degrees every symbol, as a consequence, the amplitude never goes through the origin. In the transmitter circuit, it's difficult to turn the carrier off, so we try to avoid the zero point in the constellation.

The radio numbers live in the Bluetooth Core Specification: Enhanced Data Rate uses $\pi/4$-DQPSK at 2 Mb/s and 8DPSK at 3 Mb/s.

I don't think 16PSK is that common, at 4-bits per symbol it's common to switch to Quadrature Amplitude Modulation (QAM), as shown below. The goal of QAM is to maximize the distance between each symbol. The challenge with QAM is the amplitude modulation. The modulation scheme is sensitive to variations in the transmitter amplitude. As such, more complex circuits than 8PSK could be necessary.

If you wanted to research "new fancy modulation schemes" I'd think about Sphere packing.

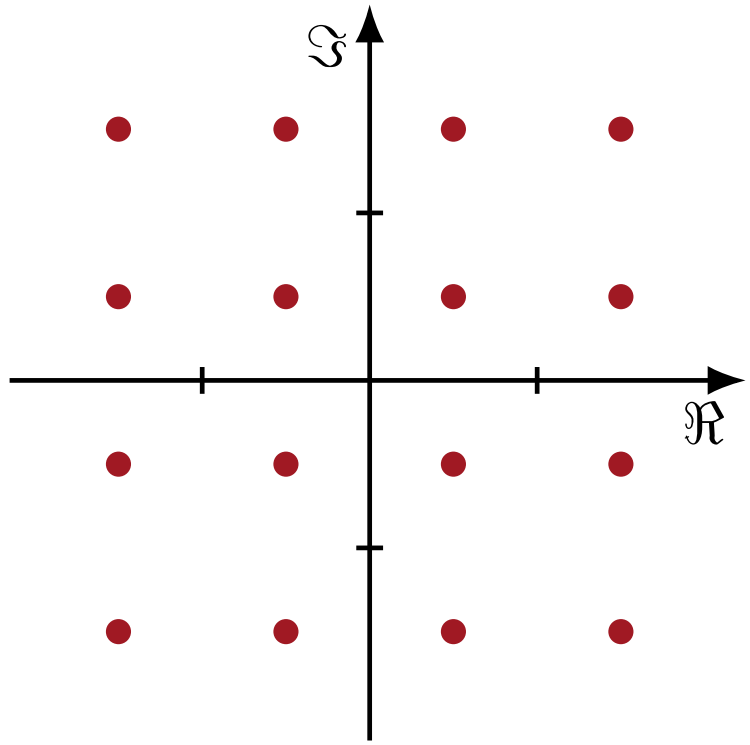


Figure 9: 16-QAM constellation: a 4 by 4 grid of points in phase and amplitude, so 4 bits per symbol

26.4.3 Single carrier, or multi carrier?

Assume we wanted to send 1 Gbps over the air. We could choose a bandwidth of about 1 GHz with 1 bit per symbol, or a bandwidth of 100 MHz if we sent 1024 QAM at 100 MS/s. Both cases would look like the figure below.

In both cases we get problems with the physical communication channel, the change in phase and amplitude affect what is received. For a 1 GHz bandwidth at 2.4 GHz carrier we'd have problems with the phase. At 1024 QAM we'd have problems with the amplitude.

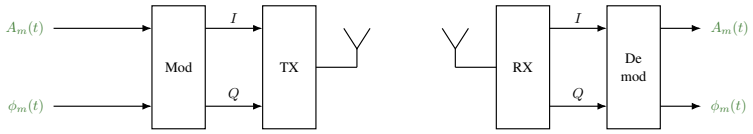


Figure 10: Single carrier link: amplitude $A_m(t)$ and phase $\phi_m(t)$ are modulated onto I and Q, transmitted, and de-modulated after the receiver

Back in 1966 Orthogonal frequency division multiplexing was introduced to deal with the communication channel. In OFDM we modulate a number of sub-carriers in the frequency space with our wanted modulation scheme (BPSK, PSK, QAM), then do an inverse fourier transform to get the time domain signal, mix on to the carrier, and transmit. At the receiver we take an FFT and

do demodulation in the frequency space. See example in figure below.

The name "multiple carriers" is a bit misleading. Although there are multiple carriers on the left and right side of the figure, there is normally still just one carrier in the TX/RX.

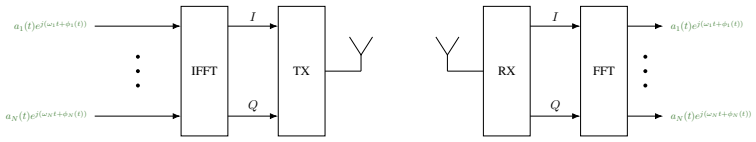


Figure 11: OFDM link: the sub-carriers are modulated in frequency space, an IFFT makes the time domain I and Q for the transmitter, and an FFT at the receiver recovers the sub-carriers

There are more details in OFDM than the simple statement above, but the details are just to fix challenges, such as "How do I recover the symbol timing? How do I correct for frequency offset? How do I ensure that my time domain signal terminates correctly for every FFT chunk"

The genius with OFDM is that we can pick a few of the sub-carriers to be pilot tones that carry no new information. If we knew exactly what was sent in phase and amplitude, then we could measure the phase and amplitude change due to the physical communication channel, and we could correct the frequency space before we tried to de-modulate.

It's possible to do the same with single carrier modulation also. Imagine we made a 128-QAM modulation on a single carrier. As long as we constructed the time domain signal correctly (cyclic prefix to make the FFT work nicely, some preamble to measure the communication channel, then we could take an FFT at the receiver, correct the phase and amplitude, do an IFFT and demodulate the time-domain signal as normal.

In radio design there are so many choices it's easy to get lost.

26.4.4 Use a Software Defined Radio

For our mouse, what radio scheme should we choose? One common instance of "how to make a choice" in industry is "Delay the choice as long as possible so you're sure the choice is right".

Maybe the best would be to use a software defined radio receiver? Something like the picture below, an antenna, low noise amplifier, and a analog-to-digital converter. That way we could support any transmitter. Fantastic idea, right?

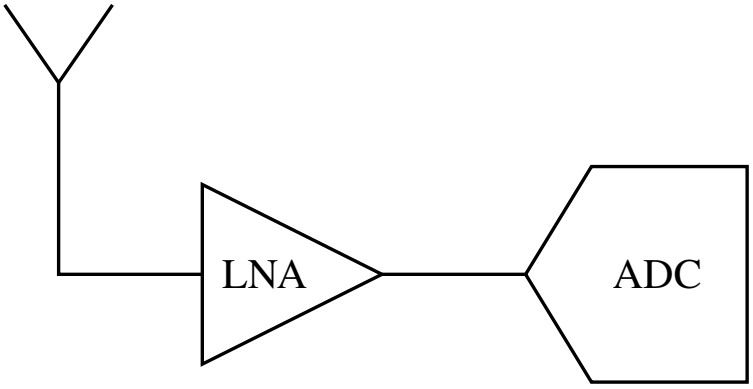


Figure 12: Software defined radio receiver: antenna, low noise amplifier and analog-to-digital converter, nothing else

Well, lets check if it's a good idea. We know we'll use 2.4 GHz, so we need about 2.5 GHz bandwidth, at least. We know we want good range, so maybe 100 dB dynamic range. In analog to digital converter design there are figure of merits, so we can actually compute a rough power consumption for such an ADC.

ADC FOM

$$= \frac{P}{2BW2^n}$$

State of the art FOM

$$\approx 5 \text{ fJ/step}$$

$$BW = 2.5 \text{ GHz}$$

$$DR = 100 \text{ dB} \Rightarrow \text{Bits} = (100 - 1.76)/6.02 \approx 16 \text{ bit}$$

$$P = 5 \text{ fJ/step} \times 5 \text{ GHz} \times 2^{16} = 1.6 \text{ W}$$

At 1.6 W our mouse would only last for 2 hours. That's too short. It will never be a low power idea to convert the full 2.5 GHz bandwidth to digital, we need some bandwidth selectivity in the receive chain.

26.5 Bluetooth

Bluetooth was made to be a "simple" standard and was introduced in 1998. The standard has continued to develop, with Low Energy introduced in 2010. The latest planned changes can be seen at Specifications in Development.

26.5.1 Bluetooth Basic Rate/Extended Data rate

- 2.400 GHz to 2.4835 GHz
- 1 MHz channel spacing
- 78 Channels
- Up to 20 dBm
- Minimum -70 dBm sensitivity (1 Mbps)
- 1 MHz GFSK (1 Mbps), pi/4-DQPSK (2 Mbps), 8DPSK (3 Mbps)

You'll find BR/EDR in most audio equipment, cars and legacy devices. For new devices (also audio), there is now a transition to Bluetooth Low Energy.

26.5.2 Bluetooth Low Energy

- 2.400 GHz to 2.480 GHz
- 2 MHz channel spacing
- 40 Channels (3 primary advertising channels)
- Up to 20 dBm
- Minimum -70 dBm sensitivity (1 Mbps)
- 1 MHz GFSK (1 Mbps, 500 kbps, 125 kbps), 2 MHz GFSK (2 Mbps)

Below are the Bluetooth LE channels. The green are the advertiser channels, the blue are the data channels, and the red humps are the WiFi channels.

The advertiser channels have been intentionally placed where there is space between the WiFi channels to decrease the probability of collisions.

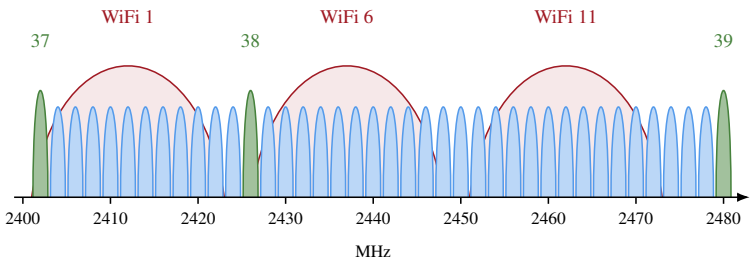


Figure 13: Bluetooth LE channel map: advertising channels 37, 38 and 39 (green) at 2402, 2426 and 2480 MHz sit in the gaps around WiFi channels 1, 6 and 11 (red); the 37 data channels are blue

Any Bluetooth LE peripheral will advertise its presence, it will wake up once in a while (every few hundred milliseconds, to seconds) and transmit a short "I'm here" packet. After transmitting it will wait a bit in receive to see if anyone responds.

A Bluetooth LE central will camp in receive on a advertiser channel and look for these short messages from peripherals. If one is observed, the Central may choose to respond.

Take any spectrum analyzer anywhere, and you'll see traffic on 2402, 2426, and 2480 MHz.

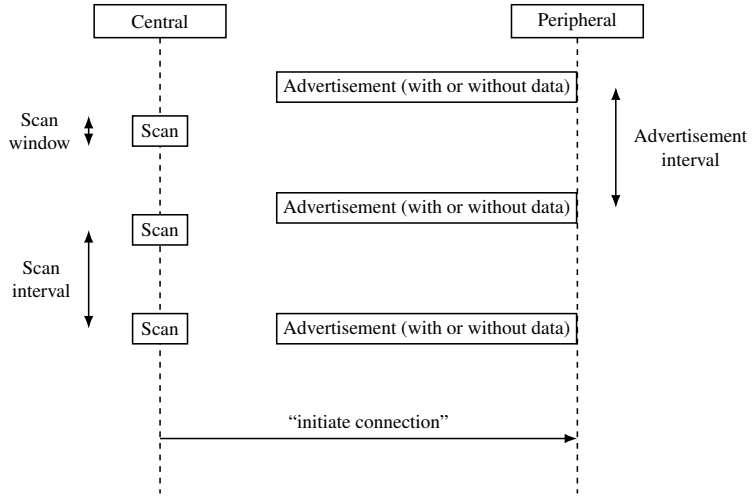


Figure 14: Advertising: the peripheral transmits advertisements once per advertisement interval while the central scans, until the central initiates a connection. Redrawn from Nordic Semiconductor DevZone

In a connection a central and peripheral (the master/slave names below have been removed from the spec, that was a fun update to a 3500 page document) will have agreed on an interval to talk. Every "connection interval" they will transmit and receive data. The connection interval is tunable from 7.5 ms to seconds.

Bluetooth LE is the perfect standard for wireless mice.

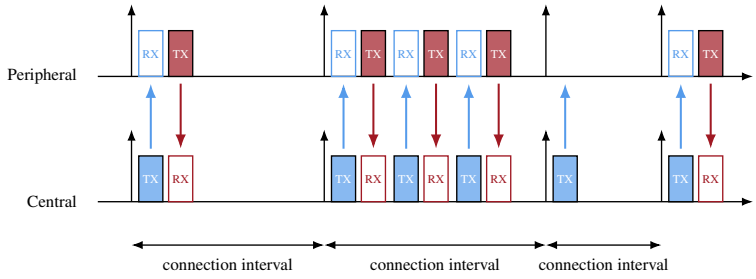


Figure 15: In a connection, central and peripheral exchange transmit and receive packets once every connection interval; an event with no data is still a transmission. Redrawn from Nordic Semiconductor DevZone

further information Building a Bluetooth application on nRF Connect SDK

Bluetooth Specifications in Development

26.6 Algorithm to design state-of-the-art LE radio

- Find most recent digest from International Solid State Circuit Conference (ISSCC)
- Find Bluetooth low energy papers
- Pick the best blocks from each paper

A typical Bluetooth radio may look something like the picture below. There would be a single antenna for both RX and Tx. There will be some way to combine the transmit and receive path in a match, or balun.

The receive chain would have a LNA, mixer, anti-alias filter and analog-to-digital converters. It's likely that the receive path would be complex (in-phase and quadrature phase) after mixer.

There would be a local oscillator (all-digital phase-locked-loop) to provide the frequency to the mixers and transmit path, which could be either polar or Cartesian.

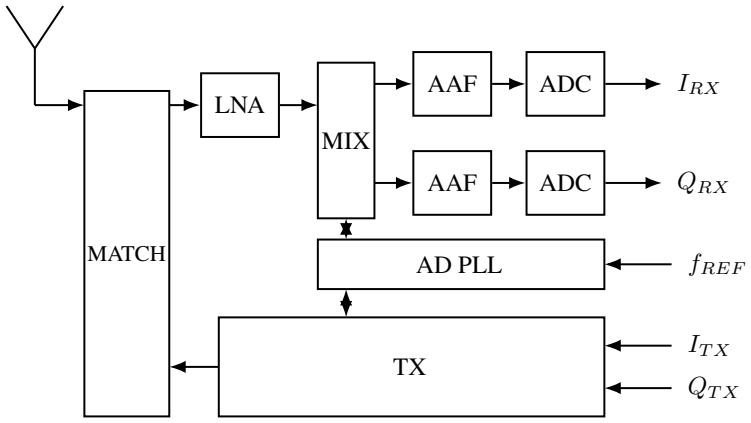


Figure 16: Typical Bluetooth radio: antenna and match, LNA, mixer, I and Q anti-alias filters and ADCs, an all-digital PLL, and the transmit path

In the typical radio we'll need the blocks below. I've added a column for how many people I would want if I was to lead development of a new radio.

Blocks	Key parameter	Architecture	Complexity (nr people)
Antenna	Gain, impedance	lambda/4	<1
RF match	loss, input impedance	PI-match	<1
Low noise amp	NF, current, linearity	LNTA	1
Mixer	NF, current, linearity	Passive	1
Anti-alias filter	NF, current, linearity	Active-RC	1
ADC	Sample rate, dynamic range, linearity	NS-SAR	1 - 2
PLL	Phase noise, current	AD-PLL	2-3
Baseband	Eb/N0, gate count, current.	SystemVerilog	> 10

26.6.1 LNTA

The first thing that must happen in the radio is to amplify the noise as early as possible. Any circuit has inherent noise, be it thermal-, flicker-, burst-, or shot-noise. The earlier we can amplify the input noise, the less contribution there will be from the radio circuits.

The challenges in the low noise amplifier is to provide the right gain. If there is a strong input signal, then reduce the gain. If there is a low input signal, then increase the gain.

One way to implement variable gain is to reconfigure the LNA. For an example, see

30.5 A 0.5V BLE Transceiver with a 1.9mW RX Achieving -96.4dBm Sensitivity and 4.1dB Adjacent Channel Rejection at 1MHz Offset in 22nm FDSOI [1]

A typical Low Noise Transconductance Amplifier is seen below. It's a combination of both a common source, and a common gate amplifier. The current in the NMOS and PMOS is controlled by Vgp and Vgn. Keep in mind that at RF frequencies the signals are weak, so it's easy to provide the DC for the LNA with a resistor to a diode connected PMOS or NMOS.

In a LNA the input impedance must be matched to what is required by the antenna/match in order to have maximum power transfer, that's the role of the inductors/capacitors.

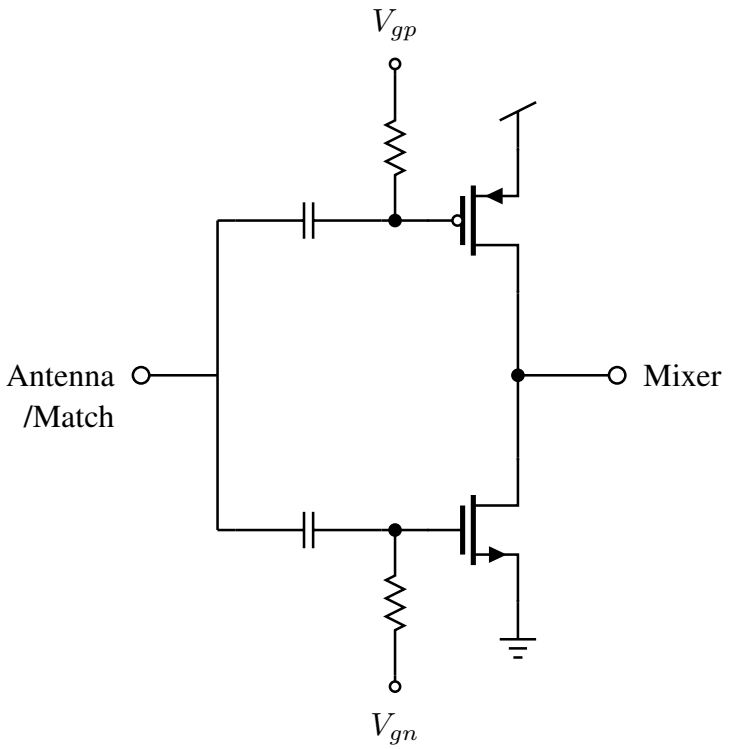


Figure 17: Low noise transconductance amplifier: complementary common source PMOS and NMOS, AC coupled from the antenna match and biased through resistors by V_{gp} and V_{gn}

26.6.2 MIXER

In the mixer we multiply the input signal with our local oscillator. Most often a complex mixer is used. There is nothing complex about complex signal processing, just read

Complex signal processing is not complex [2]

In order to reduce power, it's most common with a passive mixer as shown below. A passive mixer is just MOS that we turn on and off with 25% duty-cycle. See example in

A 370uW 5.5dB-NF BLE/BT5.0/IEEE 802.15.4-Compliant Receiver with >63dB Adjacent Channel Rejection at >2 Channels Offset in 22nm FDSOI [3]

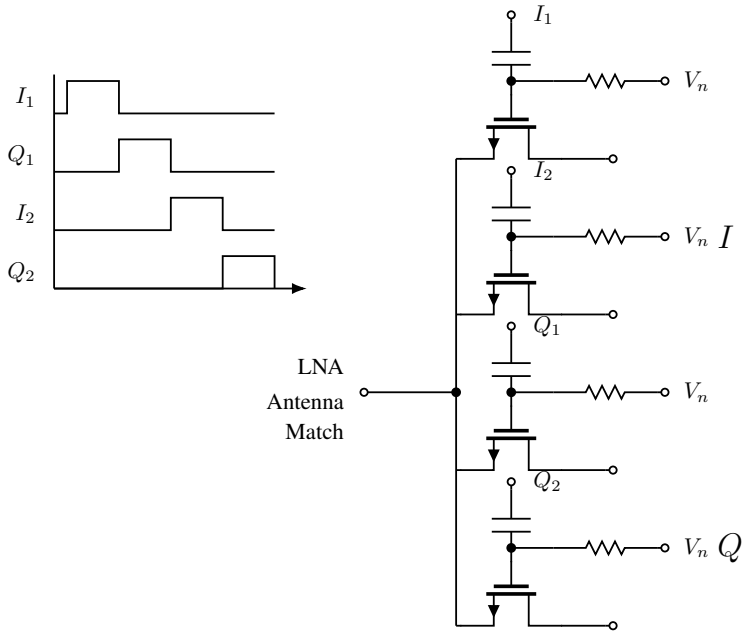


Figure 18: Passive complex mixer: four MOS switches driven by 25% duty-cycle clocks I_1, I_2, Q_1 and Q_2 split the LNA current into the I and Q outputs. Each gate is AC coupled to its clock and biased to V_n through a resistor, so the switch sees a rail-to-rail drive while its DC operating point is set independently. Note in the timing diagram that the four phases abut: four quarters fill the period exactly, so between them the switches carry the whole of the LNA current and none of it is thrown away.

To generate the quadrature and in-phase clock signals, which must be 90 degrees phase offset, it's common to generate twice the frequency in the local oscillator (4.8 GHz), and then divide down to 4 2.4 GHz clock signals.

If the LO is the same as the carrier, then the modulation signal will be at DC, often called direct conversion.

The challenge at DC is that there is flicker noise, offset, and burst noise. The modulation type, however, can impact whether low frequency noise is an issue. In OFDM we can choose to skip the sub-carriers around 0 Hz, and direct conversion works well. An advantage with direct conversion is that there is no "image frequency" and we can use the full complex bandwidth.

For FSK and direct conversion the low frequency noise can cause issues, as such, it's common to offset the LO from the transmitted signal, for example 4 MHz offset. The low frequency noise problem disappears, however, we now have a challenge with the image frequency (-4 MHz) that must be rejected, and we need an increased bandwidth.

There is no "one correct choice", there are trade-offs that both ways. KISS (Keep It Simple Stupid) is one of my guiding principles when working on radio architecture.

These days most de-modulation happens in digital, and we need to convert the analog signal to digital, but first AAF.

26.6.3 AAF

The anti alias filter rejects frequencies that can fold into the band of interest due to sampling. A simple active-RC filters is often good enough.

We often need gain in the AAF, as the LNA does not have sufficient gain for the weakest signals. -100 dBm in 50 ohm is 2.2 μV RMS, while the input range of an ADC may be 1 V. Assume we place the lowest input signal at 0.1 V, so we need a voltage gain of $20\log(0.1/2.2\times10^{-6}) \approx 93$ dB in the receiver.

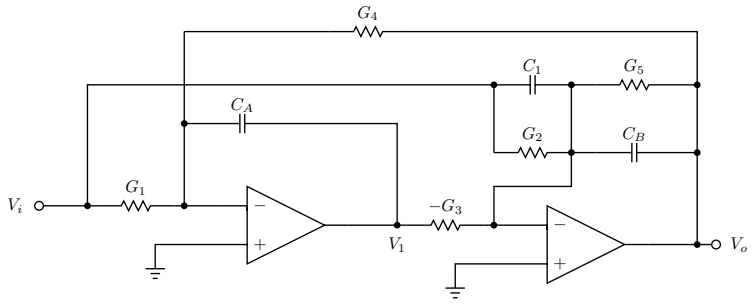


Figure 19: General purpose Active-RC biquad, used here as the anti-alias filter

26.6.4 ADC

Aaah, ADCs, an IP close to my heart. I did my Ph.d and Post-Doc on ADCs, and the Ph.D students I've co-supervised have worked on ADCs.

At NTNU there have been multiple students through the years that have made world-class ADCs, and there's still students at NTNU working on state-of-the-art ADCs.

These days, a good option is a SAR, or a Noise-Shaped SAR.

If I were to pick, I'd make something like A 68 dB SNDR Compiled Noise-Shaping SAR ADC With On-Chip CDAC Calibration [4] as shown in the figure below.

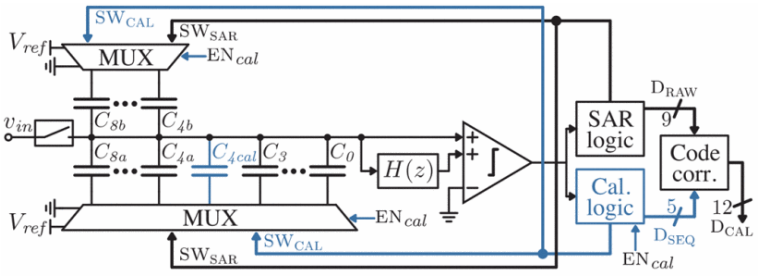


Figure 20: Architecture of the noise-shaping SAR ADC: capacitive DAC with multiplexers, loop filter H(z), integrating comparator, SAR logic, calibration logic and code correction. © 2019 IEEE. Reprinted, with permission, from [4]

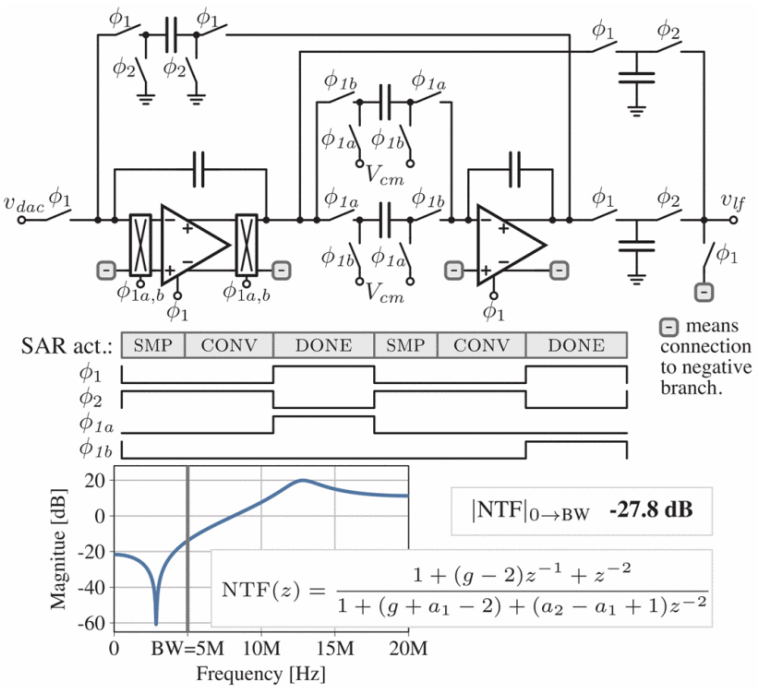


Figure 21: The switched-capacitor loop filter with two OTAs (the first one chopped), the clock phases relative to the SAR activity, and the resulting

NTF with -27.8 dB in-band suppression. © 2019 IEEE. Reprinted, with permission, from [4]

Or if I did not need high resolution, I'd choose my trusty A Compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for Bluetooth Low Energy Receivers [5].

The main selling point of that ADC was that it's compiled from a JSON file, a SPICE file and a technology file into a DRC/LVS clean layout.

I also included a few circuit improvements. The bottom plate of the SAR capacitor is in the clock loop for the comparator (DN0, DP1 below), as such, the delay of the comparator automatically adjusts with capacitance corner, so it's more robust over corners

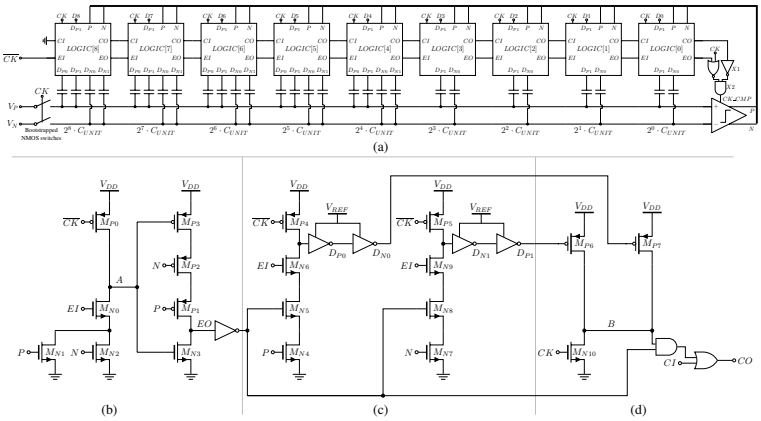


Figure 22: SAR ADC schematic: (a) capacitor array with self-timed SAR logic chain and comparator, (b) enable flip-flop, (c) bottom-plate switching of the CDAC, (d) comparator clock generation. © 2017 IEEE. Reprinted, with permission, from [5]

The compiled nature also made it possible to quickly change the transistor technology. Below is a picture with 180 nm FDSOI transistors on the left, and 28 nm FDSOI transistors on the right.

I detest doing anything twice, so I love the fact that I never have to re-draw that ADC again. I just fix the technology file (and maybe some tweaks to the other files), and I have a completed ADC.

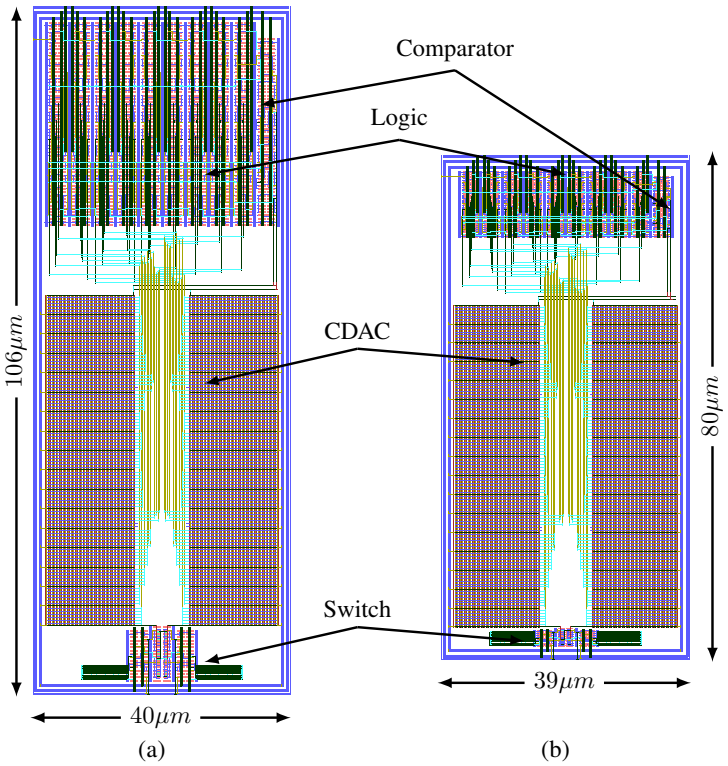


Figure 23: Layout of the two compiled SAR ADCs with comparator, logic, CDAC and switch: (a) 180 nm IO-transistor version, (b) core-transistor version. © 2017 IEEE. Reprinted, with permission, from [5]

26.6.5 AD-PLL

The phase locked loop is the heart of the radio, and it's probably the most difficult part to make. Depends a bit on technology, but these days, All Digital PLLs are cool. Start by reading Razavi's PLL book.

You can spend your life on PLLs.

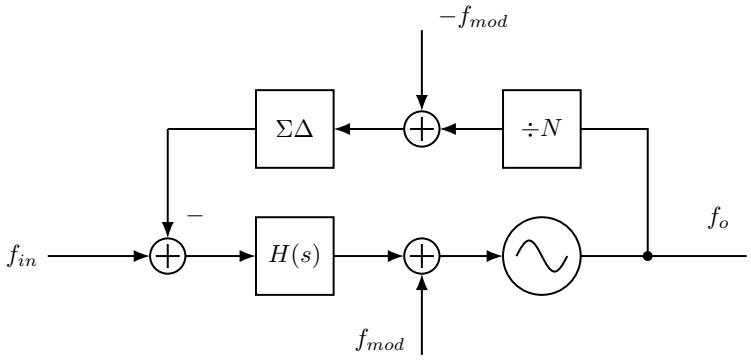


Figure 24: Two-point modulation: the modulation is applied to the oscillator and the opposite signal to the sigma-delta feedback divider, so the loop does not see it

AD-PLL with Bang-Bang phase detector for steady-state

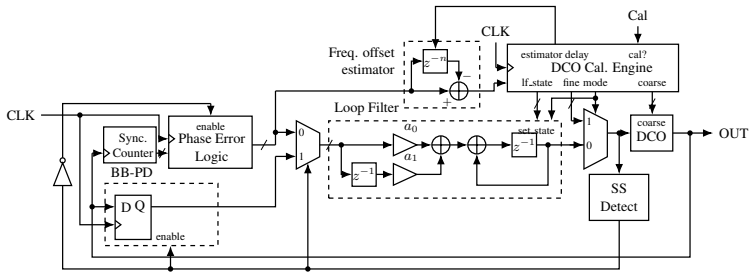


Figure 25: All-digital PLL with a bang-bang phase detector for steady-state: phase error logic, digital loop filter, DCO calibration engine with frequency offset estimator, and steady-state detect. Redrawn from Cole Nielsen's master's thesis work, NTNU, 2020 (thesis presentations)

26.6.6 Baseband

Once the signal has been converted to digital, then the demodulation, and signal fixing start. That's for another course, but there are interesting challenges.

Baseband block	Why
Mixer?	If we're using low intermediate frequency to avoid DC offset problems and flicker noise
Channel filters?	If the AAF is insufficient for adjacent channel
Power detection	To be able to control the gain of the radio
Phase extraction	Assuming we're using FSK
Timing recovery	Figure out when to slice the symbol
Bit detection	single slice, multi-bit slice, correlators etc
Address detection	Is the packet for us?
Header detection	What does the packet contain
CRC	Does the packet have bit errors
Payload de-crypt	Most links are encrypted by AES
Memory access	Payload need to be stored until CPU can do something

26.7 What do we really want, in the end?

The receiver part can be summed up in one equation for the sensitivity. The noise in a certain bandwidth. The Noise Figure of the analog receiver. The Energy per bit over Noise of the demodulator.

$$P_{RX_{sens}} = -174 \text{ dBm} + 10log_{10}(R_b) + NF + E_b/N_0$$

Term by term: -174 dBm is the thermal noise in one hertz at room temperature, R_b is the bit rate, which sets how much bandwidth that noise is collected over, NF is what the receiver's own noise adds, and E_b/N_0 is what the demodulator needs to hit its error

rate. Note that R_b here is the data rate; earlier in this chapter DR meant dynamic range, which is a different quantity entirely.

The useful move is to run it backwards. The nRF5340 datasheet quotes −97.5 dBm sensitivity at 1 Mbps, and $10log_{10}(10^6) = 60$ dB, so

$$P_{RX_{sens}} + 174 - 60 = NF + E_b/N_0 = 16.5 \text{ dB}$$

and that 16.5 dB is the entire budget shared between the analog front end and the demodulator. GFSK needs something like 12 dB of E_b/N_0, which leaves only a few decibels of noise figure for everything in front of it. That is the number the rest of this chapter is really about.

In the block diagram of the device the radio might be a small box, and the person using the radio might not realize how complex the radio actually is.

I hope you understand now that it's actually complicated.

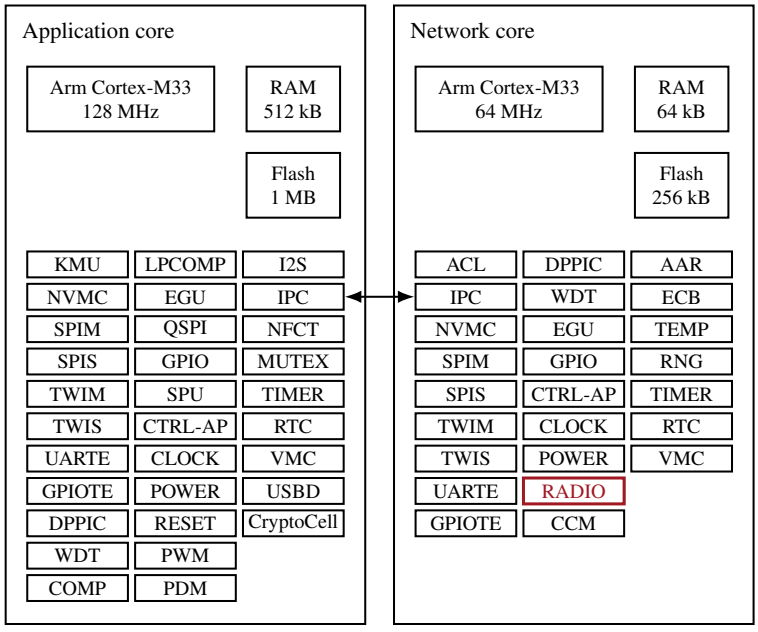


Figure 26: nRF5340 block diagram, where the entire radio is the single RADIO block (red) in the network core. Simplified and redrawn from Nordic Semiconductor, nRF5340 Product Specification

26.8 Summary

The one-page version of this chapter:

- Start from the link budget: Friis in free space, a rather worse exponent indoors, and the antenna wants its fraction of a wavelength
- The ISM bands set the playing field; 2.4 GHz trades antenna size against propagation and company
- Energy per bit is the real currency: modulation choice, data rate and duty cycle set the average current, and the battery sets the lifetime
- Single-carrier modulation keeps the PA efficient (constant envelope); multi-carrier buys spectral efficiency at the cost of backoff
- The receive chain is LNA, mixer and filter: the LNA sets the noise figure, the mixer moves the band, and everything after runs at a friendlier frequency
- A software-defined radio on the bench teaches more about radios than any equation in this chapter

26.9 Would you like to know more?

A 0.5V BLE Transceiver with a 1.9mW RX Achieving -96.4dBm Sensitivity and 4.1dB Adjacent Channel Rejection at 1MHz Offset in 22nm FDSOI [1], M. Tamura, Sony Semiconductor Solutions, Atsugi, Japan, 30.5, ISSCC 2020

A 370uW 5.5dB-NF BLE/BT5.0/IEEE 802.15.4-Compliant Receiver with >63dB Adjacent Channel Rejection at >2 Channels Offset in 22nm FDSOI [3], B. J. Thijssen, University of Twente, Enschede, The Netherlands

A 68 dB SNDR Compiled Noise-Shaping SAR ADC With On-Chip CDAC Calibration [4], H. Garvik, C. Wulff, T. Ytterdal

A Compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for Bluetooth Low Energy Receivers [5], C. Wulff, T. Ytterdal

Cole Nielsen, https://github.com/nielscol/thesis_presentations

"Python Framework for Design and Simulation of Integer-N ADPLLs", Cole Nielsen, https://github.com/nielscol/tfe4580-report/blob/master/report.pdf

Design of CMOS Phase-Locked Loops [6], Behzad Razavi, University of California, Los Angeles

[1] M. Tamura *et al.*, "30.5 a 0.5V BLE transceiver with a 1.9mW RX achieving −96.4dBm sensitivity and 4.1dB adjacent channel rejection at 1MHz offset in 22nm FDSOI," in *2020 IEEE international solid-state circuits conference - (ISSCC)*, 2020, pp. 468–470, doi: 10.1109/ISSCC19947.2020.9063021.

[2] K. Martin, "Complex signal processing is not complex," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 9, pp. 1823–1836, 2004, doi: 10.1109/TCSI.2004.834522.

[3] B. J. Thijssen, E. A. M. Klumperink, P. Quinlan, and B. Nauta, "30.4 a 370μw 5.5dB-NF BLE/BT5.0/IEEE 802.15.4-compliant receiver with >63dB adjacent channel rejection at >2 channels offset in 22nm FDSOI," in *2020 IEEE international solid-state circuits conference - (ISSCC)*, 2020, pp. 466–468, doi: 10.1109/ISSCC19947.2020.9062973.

[4] H. Garvik, C. Wulff, and T. Ytterdal, "A 68 dB SNDR compiled noise-shaping SAR ADC with on-chip CDAC calibration," in *2019 IEEE asian solid-state circuits conference (a-SSCC)*, 2019, pp. 193–194, doi: 10.1109/A-SSCC47793.2019.9056925.

[5] C. Wulff and T. Ytterdal, "A compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for bluetooth low energy receivers," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 7, pp. 1915–1926, 2017, doi: 10.1109/JSSC.2017.2685463.

[6] B. Razavi, *Design of CMOS phase-locked loops*. Cambridge University Press, 2020.

27 Energy Sources

27.1 Thermoelectric . . . 523
27.1.1 Radioisotope Thermoelectric generator 527
27.1.2 Thermoelectric generators 528
27.2 Photovoltaic 529
27.3 Piezoelectric 532
27.4 Electromagnetic . . 533
27.4.1 "Near field" harvesting 533
27.4.2 Ambient RF Harvesting 534
27.5 Triboelectric generator 536
27.6 Comparison 537
27.7 Summary 539
27.8 Would you like to know more? 539

Integrated circuits are wasteful of energy. Digital circuits charge transistor gates to change states, and when discharged, the charges are dumped to ground. In analog circuits the transconductance requires a DC current, a continuous flow of charges from positive supply to ground.

Integrated circuits are incredibly useful though. Life without would be different.

A continuous effort from engineers like me have reduced the power consumption of both digital and analog circuits by order of magnitudes since the invention of the transistor 75 years ago.

One of the first commercial ADCs, the DATRAC on page 24, was a 11-bit 50 kSps that consumed 500 W. That's Walden figure of merit of 4 μJ/conv.step. Today's state-of-the-art ADCs in the same sampling range have a Walden figure of merit of 0.6 fJ/conv.step [1].

4 μ / 0.6 f = 6.7e9, a difference in power consumption of almost 10 billion times !!!

Improvements to power consumption have become harder and harder, but I believe there is still far to go before we cannot reduce power consumption any more.

Towards a Green and Self-Powered Internet of Things Using Piezoelectric Energy Harvesting [2] [1] has a nice overview of power consumption of technologies, seen in the next figures below.

In the context of energy harvesting, there is energy in electromagnetic fields, temperature, and mechanical stress, and there are ways to translate between them the energy forms.

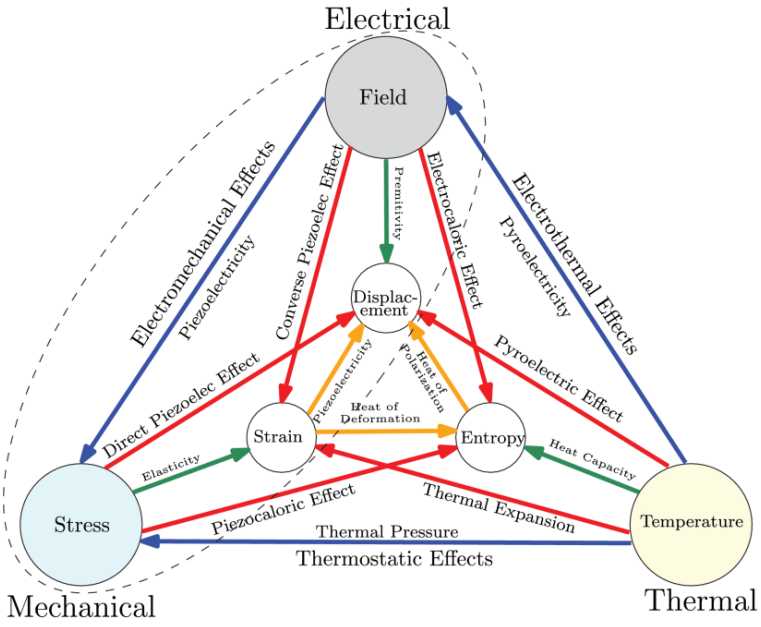


Figure 1: The couplings between the electrical, mechanical and thermal energy domains, including the piezoelectric, pyroelectric and thermoelectric effects. From Shirvanimoghaddam et al., IEEE Access *7 (2019) [2], licensed CC BY 4.0*

Below we can see a figure of the potential energy that can be harvested per volume, and the type power consumption of technologies [1].

As devices approach average power consumption of μW it becomes possible to harvest the energy from the environment, and do away with the battery.

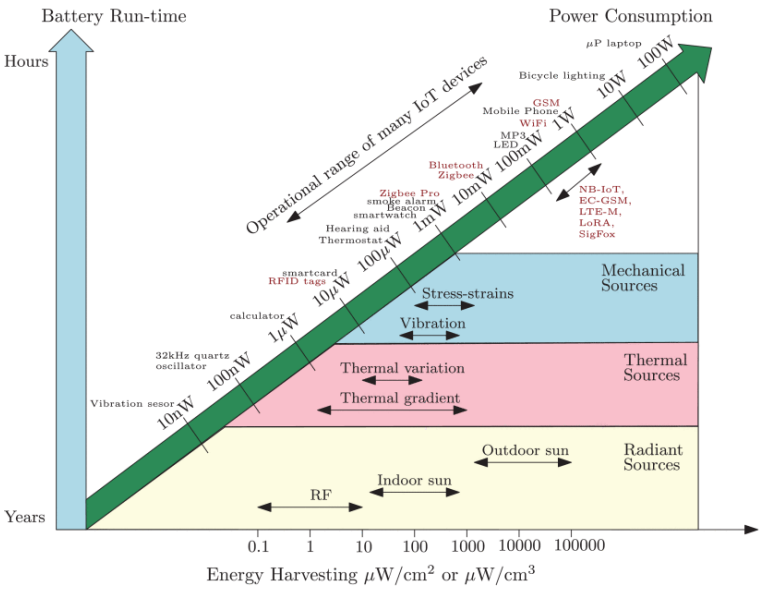


Figure 2: Battery run-time and power consumption of devices from a 32 kHz quartz oscillator to a laptop, plotted against the energy available from mechanical, thermal and radiant sources in uW/cm2. From

Shirvanimoghaddam et al., IEEE Access *7 (2019) [2], licensed CC BY 4.0*

For wireless standards, there are some that can be run on energy harvesting. Below is an overview from [1]. Many of us will have a NFC card in our pocket for payment, or entry to buildings. NFC card has a integrated circuit that is powered from the electromagnetic field from the NFC reader.

Other standards, like Bluetooth, WiFi, LTE are harder to run battery less, because the energy requirement above 1 mW.

Technologies like Bluetooth LE, however, can approach < 10 μW for some applications, although the burst power may still be 10 mW to 100 mW. As such, although the average power is low, the energy harvesting cannot support peak loads and a charge storage device is required (battery, super-capacitor, large capacitor).

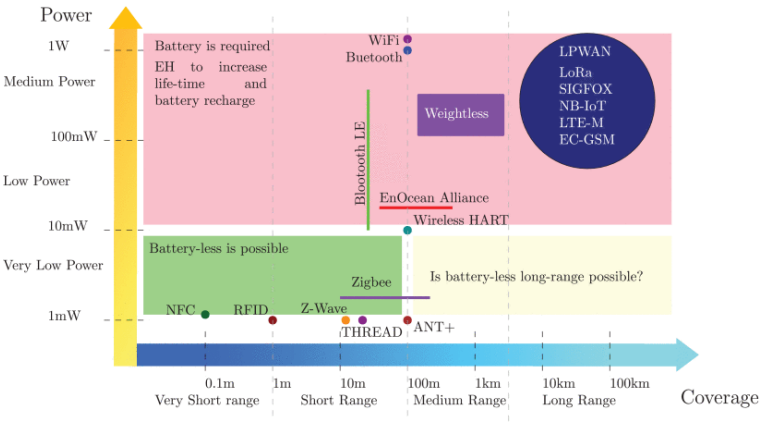


Figure 3: Power versus coverage for wireless standards, showing that NFC, RFID, Z-Wave and Zigbee can be battery-less while WiFi, Bluetooth and the cellular standards need a battery. From Shirvanimoghaddam et al., IEEE Access *7 (2019) [2], licensed CC BY 4.0*

I'd like to give you an introduction to the possible ways of harvesting energy. I know of five methods: - thermoelectric - photovoltaic - piezoelectric - electromagnetic - triboelectric

27.1 Thermoelectric

Apply heat to one end of a metal wire, what happens to the free electrons? As we heat the material we must increase the energy of the free electrons at the hot end of the wire. The atoms wiggle more, and when the free electrons scatter off the atomic structure there should be an exchange of energy. Think of the electrons at the hot side as high energy electrons, while on the cold side there are low energy electrons, I think.

There will be diffusion current of electrons in both directions in the material, however, if the mobility of electrons in the material is dependent on the energy, then we would get a difference in current of low energy electrons and high energy electrons. A difference in current would lead to a charge difference at the hot end and cold end, which would give a difference in voltage.

Take a copper wire, bend it in half, heat the end with the loop, and measure the voltage at the cold end. Would we measure a voltage difference?

NO, there would not be a voltage difference between the two ends of the wire. The voltage on the loop side would be different, but on the cold side, where we have the ends, there would be no voltage difference.

Gauss law tell us that inside a conductor there cannot be a static field without a current. As such, if there was a voltage difference between the cold ends, it would quickly be dissipated, and no DC current would flow.

The voltage difference in the material between the hot and cold end will create currents, but we can't use them if we only have one type of material.

Imagine we have Iron and copper wires, as shown below, and we heat one end. In that case, we can draw current between the cold ends.

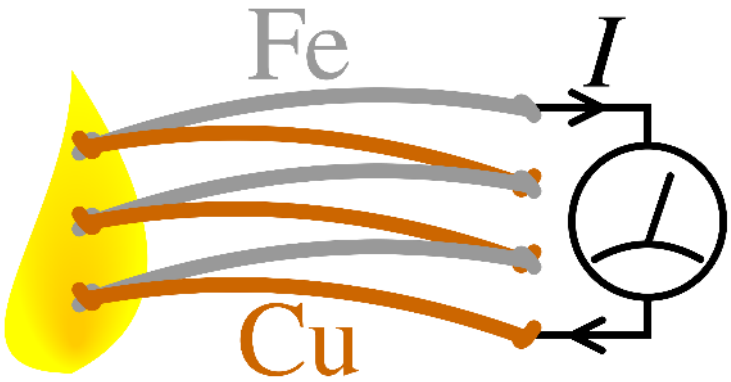


Figure 4: The thermoelectric effect - iron and copper wires joined at a heated end drive a current through a meter connected at the cold end. Image: Cmglee, CC BY-SA 4.0, via Wikimedia Commons

The voltage difference at the hot and cold end is described by the

Seebeck coefficient

Imagine two parallel wires with different Seebeck coefficients, one of copper (6.5 $\mu V/K$) and one of iron (19 $\mu V/K$). We connect them at the hot end. The voltage difference between hot and cold would

be higher in the iron, than in the copper. At the cold end, we would now measure a difference in voltage between the wires!

In silicon, the Seebeck coefficient can be modified through doping. A model of Seebeck coefficient is shown below. The value of the Seebeck coefficient depends on the location of the Fermi level in relation to the Conduction band or the valence band.

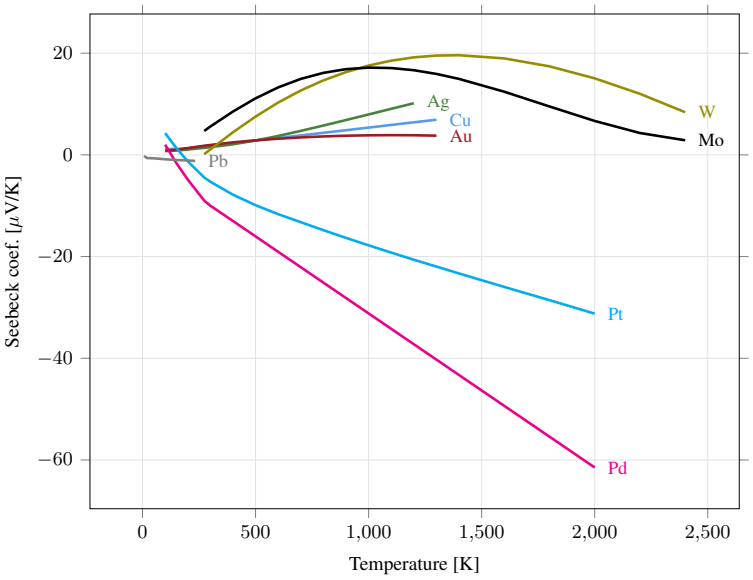


Figure 5: Absolute Seebeck coefficient versus temperature for a range of metals, spanning roughly +20 uV/K for tungsten to -60 uV/K for palladium. Data digitized from Nanite's figure, CC0, via Wikimedia Commons

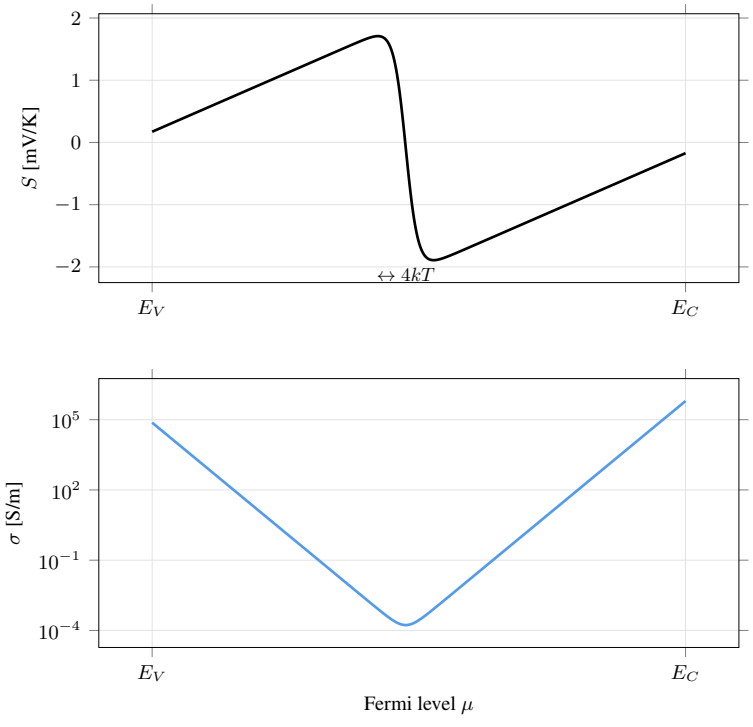


Figure 6: Seebeck coefficient and conductivity of silicon as a function of Fermi level, with the sign flipping over a window of about 4kT where the minority carriers take over, and the conductivity bottoming out at the same spot. Computed from a two-carrier Boltzmann model, after Nanite's figure, CC0, via Wikimedia Commons

In the picture below we have a silicon (the cyan and yellow colors).

Assume we dope with acceptors (yellow, p-type), that shifts the Fermi level closer to the Valence band (E_V), and the dominant current transport will be by holes, maybe we get 1 mV/K from the picture above.

For the material doped with donors (cyan, n-type) the Fermi level is shifted towards the Conduction band (E_C), and the dominant charge transport is by electrons, maybe we get -1 mV/K from the picture above.

Assume we have a temperature difference of 50 degrees, then maybe we could get a voltage difference at the cold end of 100 mV. That's a low voltage, but is possible to use.

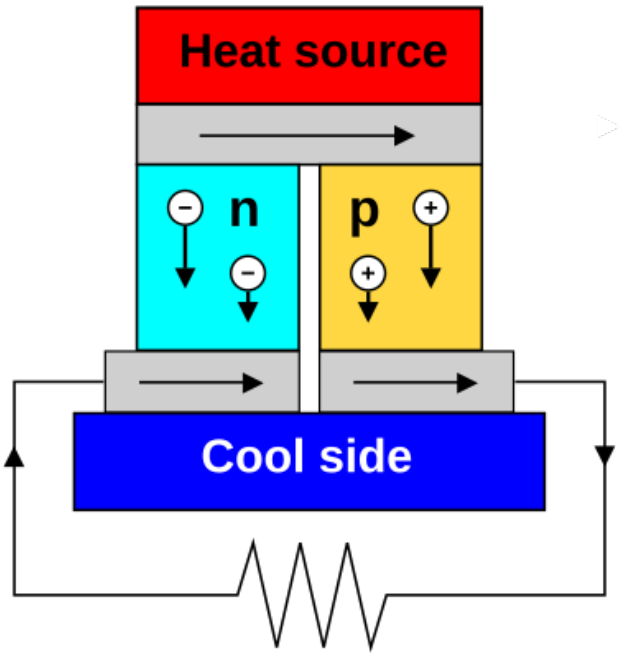


Figure 7: Thermoelectric generator, where n-type and p-type legs between a heat source and a cool side drive a current through the load resistor. Image: Ken Brazier, CC BY-SA 4.0, via Wikimedia Commons

The process can be run in reverse. In the picture below we force a current through the material, we heat one end, and cool the other. Maybe you've heard of Peltier elements.

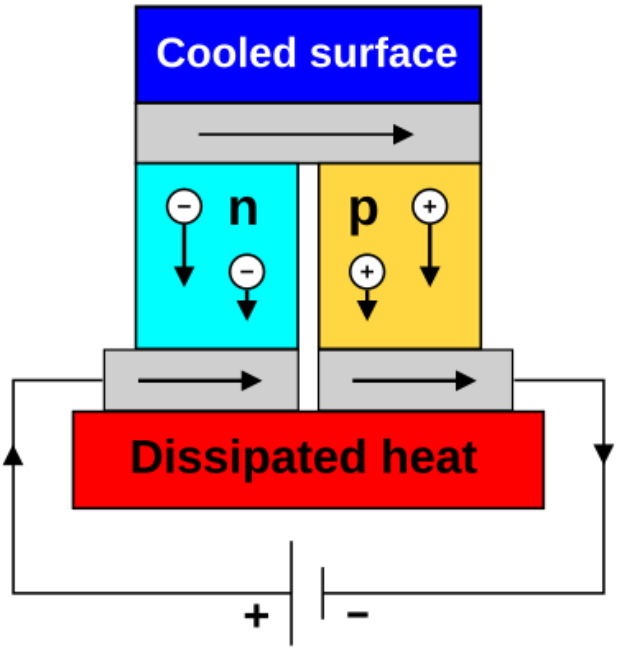


Figure 8: The same structure run in reverse as a Peltier cooler, where a forced current cools the top surface and dissipates heat at the bottom. Image: Ken Brazier, CC BY-SA 4.0, via Wikimedia Commons

27.1.1 Radioisotope Thermoelectric generator

Maybe you've heard of a nuclear battery. Sounds fancy, right? Must be complicated, right?

Not really, take some radioactive material, which generates heat, stick a thermoelectric generator to the hot side, make sure you can cool the cold side, and we have a nuclear battery.

Nuclear batteries are "simple", and ultra reliable. There's not really a chemical reaction. The nucleus of the radioactive material degrades, but not fast. In the thermoelectric generator, there are no moving parts.

In a normal battery there is a chemical reaction that happens when we pull a current. Atoms move around. Eventually the chemical battery will change and degrade.

Nuclear batteries were used in Voyager, and they still work to this day. The nuclear battery is the round thing underneath Voyager in the picture below. The radioisotopes provide the heat, space provides the cold, and voila, 470 W to run the electronics.

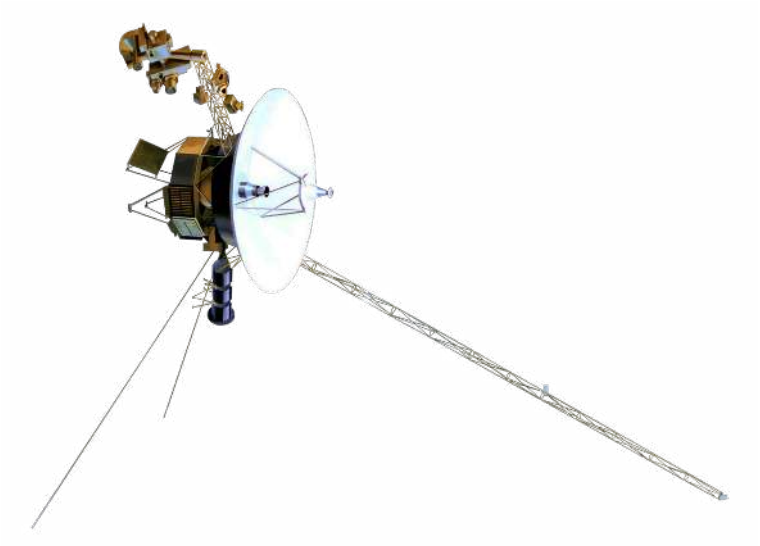


Figure 9: The Voyager spacecraft, whose radioisotope thermoelectric generator is the finned cylinder below the dish. Image: NASA, public domain, via Wikimedia Commons

27.1.2 Thermoelectric generators

Assume we wanted to drive a watch from a thermoelectric generator (TEG). The skin temperature is maybe 33 degrees Celsius, while the ambient temperature is maybe 23 degrees Celsius on average.

From the model of a thermoelectric generator below we'd get a voltage of 10 mV to 500 mV, too low for most integrated circuits.

In order to drive an integrated circuit we'd need to boost the voltage to maybe 1.8 V.

The main challenge with thermoelectric generators is to provide a cold-boot function where the energy harvester starts up at a low voltage.

In silicon, it is tricky to make anything work below some thermal voltages (kT/q). We at least need about 3 – 4 thermal voltages to make anything function.

The key enabler for an efficient, low temperature differential, energy harvester is an oscillator that works at low voltage (i.e 75 mV). If we have a clock, then we can boost with capacitors

In A 3.5-mV Input Single-Inductor Self-Starting Boost Converter With Loss-Aware MPPT for Efficient Autonomous Body-Heat Energy Harvesting [3] they use a combination of both switched capacitor and switched inductor boost.

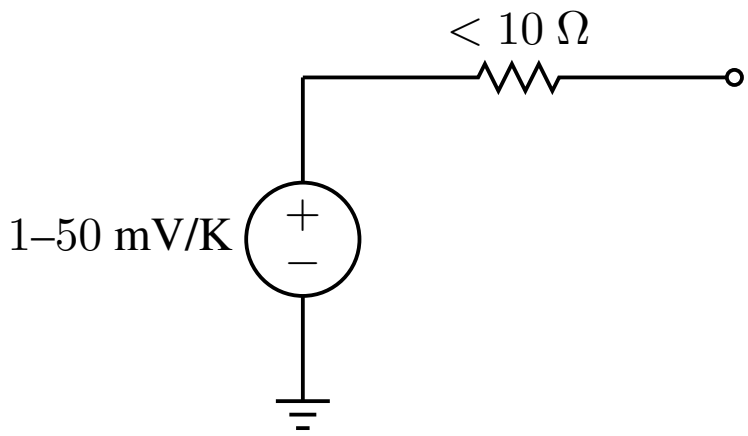


Figure 10: Model of a thermoelectric generator - a 1 to 50 mV/K source in series with a source resistance below 10 ohm - together with the self-starting boost converter architecture from Bose et al.

27.2 Photovoltaic

In silicon, photons can knock out electron/hole pairs. If we have a PN junction, then it's possible to separate the electron/holes before they recombine as shown in figure below.

An electron/hole pair knocked out in the depletion region (1) will separate due to the built-in field. The hole will go to P and the electron to N. This increases the voltage VD across the diode.

A similar effect will occur if the electron/hole pair is knocked out in the P region (2). Although the P region has an abundance of holes, the electron will not recombine immediately. If the electron diffuses close to the depletion region, then it will be swept across to the N side, and further increase VD.

On the N-side the same minority carrier effect would further increase the voltage (3).

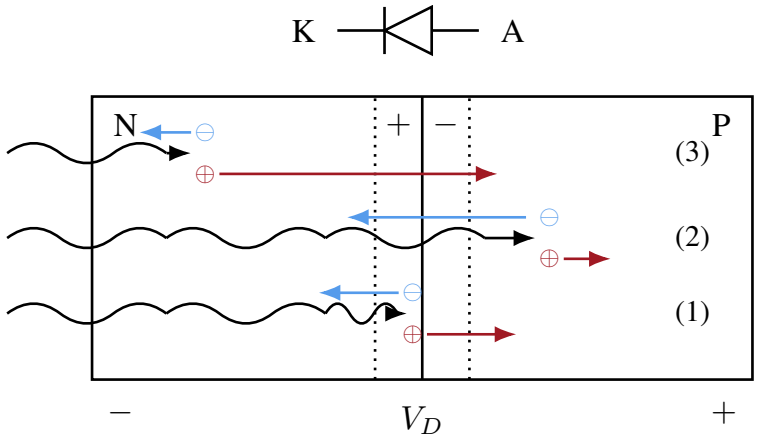


Figure 11: Photon absorption in a PN junction, where electron/hole pairs generated in the depletion region (1), the P region (2) and the N region (3) all increase the diode voltage VD

A circuit model of a Photodiode can be seen in figure below, where it is assumed that a single photodiode is used. It is possible to stack photodiodes to get a higher output voltage.

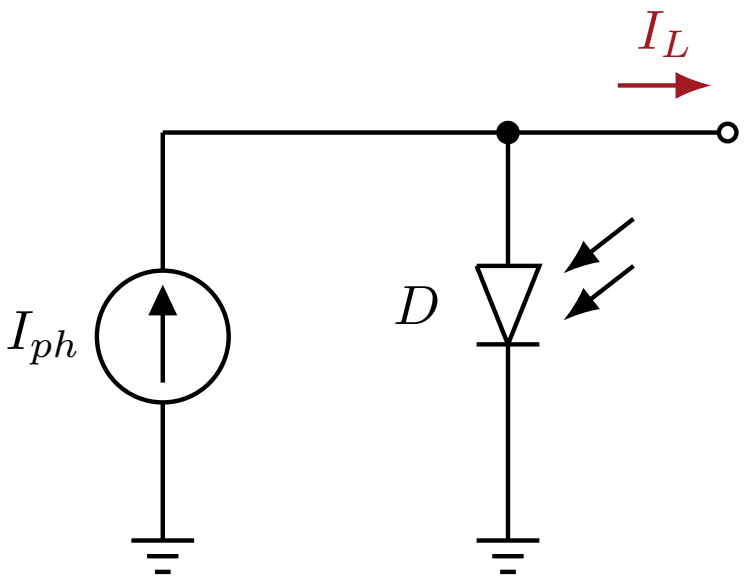


Figure 12: Circuit model of a photodiode - a photo current source in parallel with a diode, delivering a load current IL

As the load current is increased, the voltage VD will drop. As the photo current is increased, the voltage VD will increase. As such, there is an optimum current load where there is a balance between the photocurrent, the voltage VD and the load current.

$$I_D = I_S \left(e^{\frac{V_D}{V_T}} - 1 \right)$$

$$I_D = I_{Photo} - I_{Load}$$

$$V_D = V_T ln \left(\frac{I_{Photo} - I_{Load}}{I_S} + 1 \right)$$

$$P_{Load} = V_D I_{Load}$$

Below is a model of the power in the load as a function of diode voltage

```
#!/usr/bin/env python3
import numpy as np
import matplotlib.pyplot as plt

m = 1e-3
i_load = np.linspace(1e-5,1e-3,200)

i_s = 1e-12  # saturation current
i_ph = 1e-3  # Photocurrent

V_T = 1.38e-23*300/1.6e-19  #Thermal voltage

V_D = V_T*np.log((i_ph - i_load)/(i_s) + 1)

P_load = V_D*i_load

plt.subplot(2,1,1)
plt.plot(i_load/m,V_D)
plt.ylabel("Diode voltage [mA]")
plt.grid()
plt.subplot(2,1,2)
plt.plot(i_load/m,P_load/m)
plt.xlabel("Current load [mA]")
plt.ylabel("Power Load [mW]")
plt.grid()
plt.savefig("pv.pdf")
plt.show()
```

From the plot below we can see that to optimize the power we could extract from the photovoltaic cell we'd want to have a current of 0.9 mA in the model above.

The interactive version of this example plots the same equation both ways round, marks the maximum power point and reports the fill factor. Sweeping the photocurrent over six decades shows why harvesting in a dim room is a current problem rather than a voltage one.

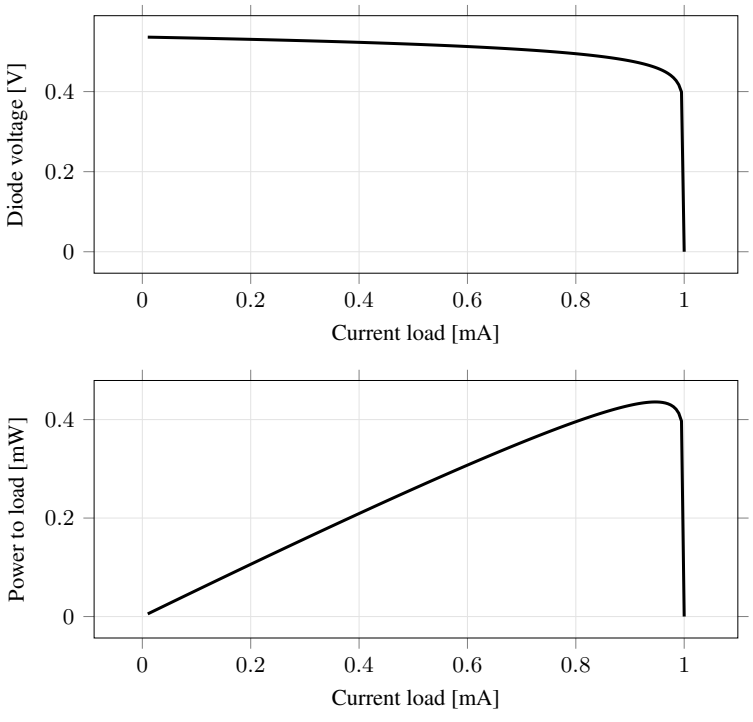


Figure 13: Diode voltage and delivered power versus load current for the photodiode model, with the maximum power point near 0.9 mA

Most photovoltaic energy harvesting circuits will include a maximum power point tracker as the optimum changes with light conditions.

In A Reconfigurable Capacitive Power Converter With Capacitance Redistribution for Indoor Light-Powered Batteryless Internet-of-Things Devices [4] they include a maximum power point tracker and a reconfigurable charge pump to optimize efficiency.

27.3 Piezoelectric

I'm not sure I understand the piezoelectric effect, but I think it goes something like this.

Consider a crystal made of a combination of elements, for example Gallium Nitride. In GaN it's possible to get a polarization of the unit cell, with a more negative charge on one side, and a positive charge on the other side. The polarization comes from an asymmetry in the electron and nucleus distribution within the material.

In a polycrystaline substance the polarization domains will usually be random, and no electric field will observable. The polarization domains can be aligned by heating the material and applying a electric field. Now all the small electric fields point in the same direction.

From Gausses law we know that the electric field through a surface is determined by the volume integral of the charges inside.

$$\oint_{\partial\Omega} \mathbf{E} \cdot d\mathbf{S} = \frac{1}{\epsilon_0} \iiint_V \rho \cdot dV$$

Although there is a net zero charge inside the material, there is an uneven distribution of charges, as such, some of the field lines will cross through the surface.

Assume we have a polycrystaline GaN material with polarized domains. If we measure the voltage across the material we will read 0 V. Even though the domains are polarized, and we should observe an external electric field, the free charges in the material will redistribute if there is a field inside, such that there is no current flowing, and thus no external field.

If we apply stress, however, all the domains inside the material will shift. Now the free charges do not exactly cancel the electric field in the material, the free charges are in the wrong place. If we

have a material with low conductivity, then it will take time for the free charges to redistribute. As such, for a while, we can measure an voltage across the material.

Assuming the above explanation is true, then there should not be piezoelectric materials with high conductivity, and indeed, most piezoelectric materials have resistance of 10^{12} to 10^{14} Ohm.

Vibrations on a piezoelectric material will result in a AC voltage across the surface, which we can harvest.

A model of a piezoelectric transducer can be seen below.

The voltage on the transducer can be on the order of a few volts, but the current is usually low (nA – µA). The key challenge is to rectify the AC signal into a DC signal. It is common to use tricks to reduce the energy waste due to the rectifier.

An example of piezoelectric energy harvester can be found in A Fully Integrated Split-Electrode SSHC Rectifier for Piezoelectric Energy Harvesting [5]

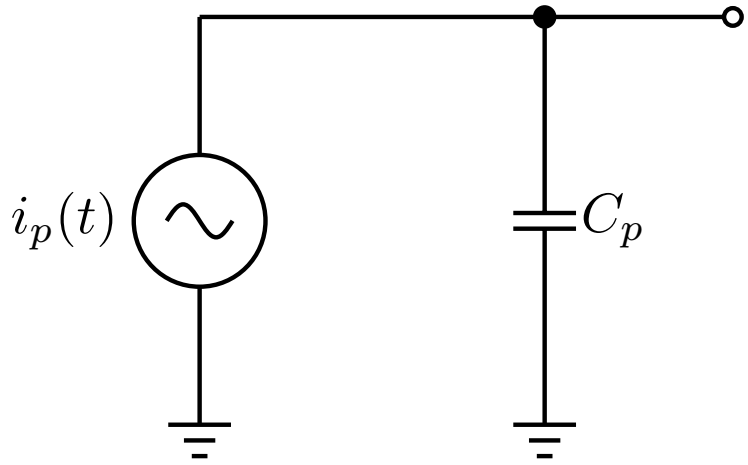


Figure 14: Model of a piezoelectric transducer - an AC current source in parallel with the transducer capacitance

27.4 Electromagnetic

27.4.1 "Near field" harvesting

Near Field Communication (NFC) operates at close physical distances

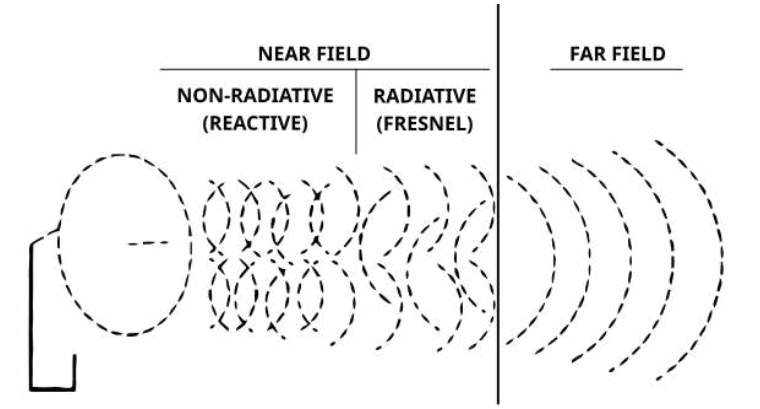


Figure 15: The reactive near field, radiative (Fresnel) near field and far field regions around an antenna. Image: Goran M Djuknic, public domain (US patent), via Wikimedia Commons

Reactive near field or inductive near field

$$\text{Inductive} < \frac{\lambda}{2\pi}$$

Within the inductive near field the antenna's can "feel" each other. The NFC reader inside the card reader can "feel" the antenna of the NFC tag. When the tag get's close it will load down the NFC reader by presenting a load impedance. As the circuit inside the tag is powered, it can change the impedance of it's antenna, which is sensed by the reader, and thus the reader can get data from the tag. The tag could lock in on the 13.56 MHz frequency and decode both amplitude and phase modulation from the reader.

Since the NFC or Qi system operates at close distances, then the coupling factor between antenna's, or really, inductors, can be decent, and it's possible to achieve efficiencies of maybe 70 %.

At Bluetooth frequencies, as can be seen below, it does not really make sense to couple inductors, as they need to be within 2 cm to be in the inductive near field. The inductive near field is a significant problem for the coupling between inductors on chip, but I don't think I would use it to transfer power.

Standard	Frequency [MHz]	Inductive [m]
AirFuel Resonant	6.78	7.03
NFC	13.56	3.52
Qi	0.205	232
Bluetooth	2400	0.02

27.4.2 Ambient RF Harvesting

Extremely inefficient idea, but may find special use-cases at short-distance.

Will get better with beam-forming and directive antennas

There are companies that think RF harvesting is a good idea.

AirFuel RF

I think that ambient RF harvesting should tingle your science spidy senses.

Let's consider the power transmitted in wireless standards. Your cellphone may transmit 30 dBm, your WiFi router maybe 20 dBm, and your Bluetooth LE device 10 dBm.

In case those numbers don't mean anything to you, below is a conversion to watts.

dBm	W
30	1
0	1 m
-30	1 u
-60	1 n
-90	1 p

Now ask your self the question "What's the power at a certain distance?". It's easier to flip the question, and use Friis to calculate the distance.

Assume

$$P_{TX}$$

= 1 W (30 dBm) and

$$P_{RX}$$

= 10 uW (-20 dBm)

then

$$D = 10^{\frac{P_{TX} - P_{RX} + 20log_{10}\left(\frac{c}{4\pi f}\right)}{20}}$$

In the table below we can see the distance is not that far!

Freq	$20log_{10}(c/4\pi f)$ [dB]	D [m]
915M	-31.7	8.2
2.45G	-40.2	3.1
5.80G	-47.7	1.3

I believe ambient RF is a stupid idea.

Assuming an antenna that transmits equally in all direction, then the loss on the first meter is 40 dB at 2.4 GHz. If I transmitted 1 W, there would only be 100 µW available at 1 meter. That's an efficiency of 0.01 %.

Just fundamentally stupid. **Stupid, I tell you!!!**

Stupidity in engineering really annoys me, especially when people don't understand how stupid ideas are.

27.5 Triboelectric generator

Although static electricity is an old phenomenon, it is only recently that triboelectric nanogenerators have been used to harvest energy.

An overview can be seen in Current progress on power management systems for triboelectric nanogenerators [6].

A model of a triboelectric generator can be seen in below. Although the current is low (nA) the voltage can be high, tens to hundreds of volts.

The key circuit challenge is the rectifier, and the high voltage output of the triboelectric generator.

Take a look in A Fully Energy-Autonomous Temperature-to-Time Converter Powered by a Triboelectric Energy Harvester for Biomedical Applications [7] for more details.

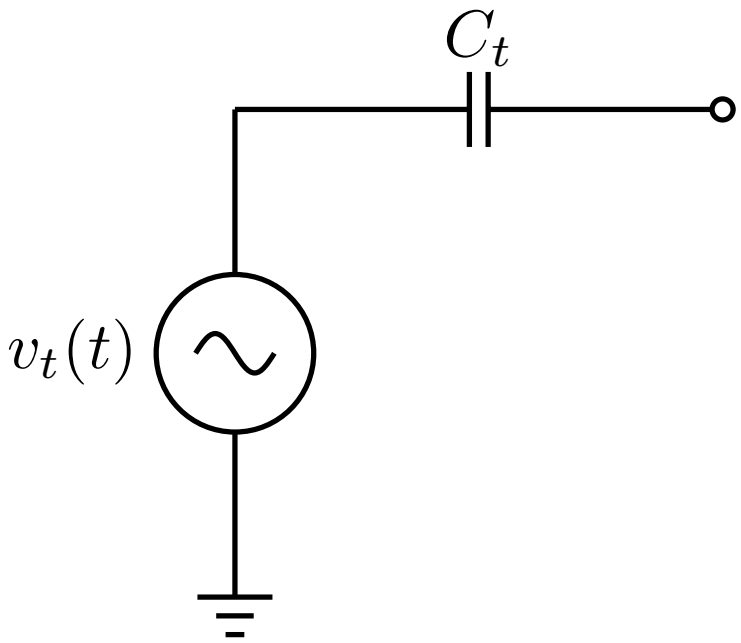


Figure 16: Model of a triboelectric generator - an AC source in series with the transducer capacitance

Tan et al. [7] built exactly this, and it is worth walking through because it is a complete system rather than a circuit.

Their transducer is a square centimetre of conductive nickel against nickel-PFA, and the energy source is human motion below 1 Hz. So the input is not a waveform in any useful sense: it is a few sparse pulses a second, each of them high voltage and almost no current, and between them nothing at all. The model in the figure above is the right picture for it, with one addition - a real transducer also has a shunt resistance across it, which is one more path for the little charge there is to leak away before it can be used.

That shapes the whole design. The rectifier cannot be a diode bridge, because two diode drops of the little charge available leaves nothing, and it cannot leak, because the next pulse may be a second away. Their answer is a low leakage rectifier feeding a power management unit whose static consumption is measured in nanowatts, storing charge on a capacitor until there is enough to do something with. On their measurements a conventional rectifier never reaches the 600 mV the circuit needs to start; theirs does.

What it then does is neat. Rather than digitise a temperature, it starts a PTAT bandgap, charges a capacitor and waits for a comparator to trip, so the output is a pulse whose *width* is the temperature. No clock, no converter, no reference to speak of - the temperature falls out of a single one-shot discharge. If all you have is a millijoule now and then, answering in the time domain is a great deal cheaper than answering in the voltage domain.

One detail in their block diagram is worth more than the circuit though. There is a `VDD_ext` on it. The system is not fully harvested.

That is not a criticism, and it is a good illustration of the division of labour. They set out to show that a triboelectric harvester can run a temperature sensor, and they showed it; how the reading gets from the chip to wherever anyone would read it is left alone, and getting a radio to run off sub-1 Hz human motion is a different paper. It is academia's job to prove that something could be possible, and industry's job to make something that could be possible actually work. Read papers with that in mind and look for the block that is still plugged into the wall.

27.6 Comparison

Imagine you're a engineer in a company that makes integrated circuits. Your CEO comes to you and says "You need to make a power management IC that harvest energy and works with everything".

Hopefully, your response would now be "That's a stupid idea, any energy harvester circuit must be made specifically for the energy source".

Thermoelectric and photovoltaic provide low DC voltage. Piezoelectric and Triboelectric provide an AC voltage. Ambient RF is stupid.

For a "energy harvesting circuit" you must also know the application (wrist watch, or wall switch) to know what energy source is available.

Below is a table that show's a comparison in the power that can be extracted.

The power levels below are too low for the peak power consumption of integrated circuits, so most applications must include a charge storage device, either a battery, or a capacitor.

Energy source	Power density	Frequency	Characteristics
Solar / PV	10 uW/cm^2 indoor, 15 mW/cm^2 outdoor	DC	Requires exposure to light
RF	0.1 uW/cm^2 GSM, 0.01 uW/cm^2 WiFi	380 MHz–5 GHz	Poor indoors and out of line of sight
Thermal, body heat	40 uW/cm^2	DC	Requires a high temperature difference
Piezoelectric	4 uW/cm^2	> 30 Hz	Not limited to indoors or outdoors
Triboelectric (TENG)	1 uW/cm^2	1 Hz	Not limited to indoors or outdoors

Numbers from Tan et al. [7].

Read the first column against the last. Solar wins outdoors by three orders of magnitude and loses most of that the moment you go inside, RF is worse than either everywhere, and the two mechanical sources are the only ones that do not care where they are. That is the whole argument for bothering with piezoelectric and triboelectric harvesting despite their being at the bottom of the power column: a microwatt you can rely on beats a milliwatt that depends on someone opening the curtains.

The frequency column matters as much as the power column and is easier to overlook. DC sources want a boost converter; the mechanical ones deliver at a few hertz or less and want a rectifier and a reservoir, and at 1 Hz the circuit spends almost all of its life waiting. That is why the leakage of the storage path, rather than the efficiency of the conversion, is what decides whether a triboelectric system works.

27.7 Summary

The one-page version of this chapter:

- Harvesters deliver power, batteries deliver energy: a harvester design starts from the average load current, not the peak
- Thermoelectric generators give millivolts per kelvin of gradient - real designs start below 100 mV and need a boost converter that can cold-start from that little
- A photovoltaic cell is the photodiode from this chapter run in the fourth quadrant: half a volt a cell, current proportional to light
- Piezo and electromagnetic harvesters turn vibration into AC that must be rectified; triboelectric is the same story with contact charging
- Ambient RF sounds free until Friis has spoken: microwatts at best, and only near the transmitter
- Every source needs power management sized to its impedance - maximum power transfer is the whole game

27.8 Would you like to know more?

[1] Towards a Green and Self-Powered Internet of Things Using Piezoelectric Energy Harvesting [2]

A 3.5-mV Input Single-Inductor Self-Starting Boost Converter With Loss-Aware MPPT for Efficient Autonomous Body-Heat Energy Harvesting [3]

A Reconfigurable Capacitive Power Converter With Capacitance Redistribution for Indoor Light-Powered Batteryless Internet- of-Things Devices [4]

A Fully Integrated Split-Electrode SSHC Rectifier for Piezoelectric Energy Harvesting [5]

Current progress on power management systems for triboelectric nanogenerators [6]

A Fully Energy-Autonomous Temperature-to-Time Converter Powered by a Triboelectric Energy Harvester for Biomedical Applications [7]

[1] S.-E. Hsieh and C.-C. Hsieh, "A 0.4V 13b 270kS/s SAR-ISDM ADC with an opamp-less time-domain integrator," in *2018 IEEE international solid-state circuits conference - (ISSCC)*, 2018, pp. 240–242, doi: 10.1109/ISSCC.2018.8310273.

[2] M. Shirvanimoghaddam *et al.*, "Towards a green and self-powered internet of things using piezoelectric energy harvesting," *IEEE Access*, vol. 7, pp. 94533–94556, 2019, doi: 10.1109/ACCESS.2019.2928523.

[3] S. Bose, T. Anand, and M. L. Johnston, "A 3.5-mV input single-inductor self-starting boost converter with loss-aware MPPT for efficient autonomous body-heat energy harvesting," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 6, pp. 1837–1848, 2021, doi: 10.1109/JSSC.2020.3042962.

[4] H.-C. Cheng, P.-H. Chen, Y.-T. Su, and P.-H. Chen, "A reconfigurable capacitive power converter with capacitance redistribution for indoor light-powered batteryless internet-of-things devices," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 10, pp. 2934–2942, 2021, doi: 10.1109/JSSC.2021.3075217.

[5] S. Du, Y. Jia, C. Zhao, G. A. J. Amaratunga, and A. A. Seshia, "A fully integrated split-electrode SSHC rectifier for piezoelectric energy harvesting," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1733–1743, 2019, doi: 10.1109/JSSC.2019.2893525.

[6] T. Hu, H. Wang, W. Harmon, D. Bamgboje, and Z.-L. Wang, "Current progress on power management systems for triboelectric nanogenerators," *IEEE Transactions on Power Electronics*, vol. 37, no. 8, pp. 9850–9864, 2022, doi: 10.1109/T-PEL.2022.3156871.

[7] J. S. Y. Tan *et al.*, "A fully energy-autonomous temperature-to-time converter powered by a triboelectric energy harvester for biomedical applications," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 10, pp. 2913–2923, 2021, doi: 10.1109/JSSC.2021.3080383.

28 Analog SystemVerilog

28.1 Digital simulation . 542
28.1.1 Digital Simulators . 544
28.2 Transient analog simulation 545
28.3 Mixed signal simulation 547
28.4 Analog SystemVerilog Example 548
28.4.1 TinyTapeout TT06_-SAR 548
28.4.2 SAR operation . . . 549
28.5 Summary 552
28.6 Would you like to know more? 552

Design of integrated circuits is split in two, analog design, and digital design.

Digital design is highly automated. The digital functions are coded in SystemVerilog (yes, I know there are others, but don't use those), translated into a gate level netlist, and automatically generated layout. Not everything is push-button automation, but most is.

Analog design, however, is manual work. We draw schematic, simulation with a mathematical model of the real world, draw the analog layout needed for the foundries to make the circuit, verify that we drew the schematic and layout the same, extract parasitics, simulate again, and in the end get a GDSII file.

When we mix analog and digital designs, we have two choices, analog on top, or digital on top.

In analog on top we take the digital IP, and do the top level layout by hand in analog tools.

In digital on top we include the analog IPs in the SystemVerilog, and allow the digital tools to do the layout. The digital layout is still orchestrated by people.

Which strategy is chosen depends on the complexity of the integrated circuit. For medium to low level of complexity, analog on top is fine. For high complexity ICs, then digital on top is the way to go.

Below is a description of the open source digital-on-top flow. The analog is included into GDSII at the OpenRoad stage of the flow.

The GDSII is not sufficient to integrate the analog IP. The digital needs to know how the analog works, what capacitance is on every digital input, the propagation delay for digital input to digital outputs , the relation between digital outputs and clock inputs, and the possible load on digital outputs.

The details on timing and capacitance is covered in a Liberty file. The behavior, or function of the analog circuit must be described in a SystemVerilog file.

But how do we describe an analog function in SystemVerilog? SystemVerilog is simulated in an digital simulator.

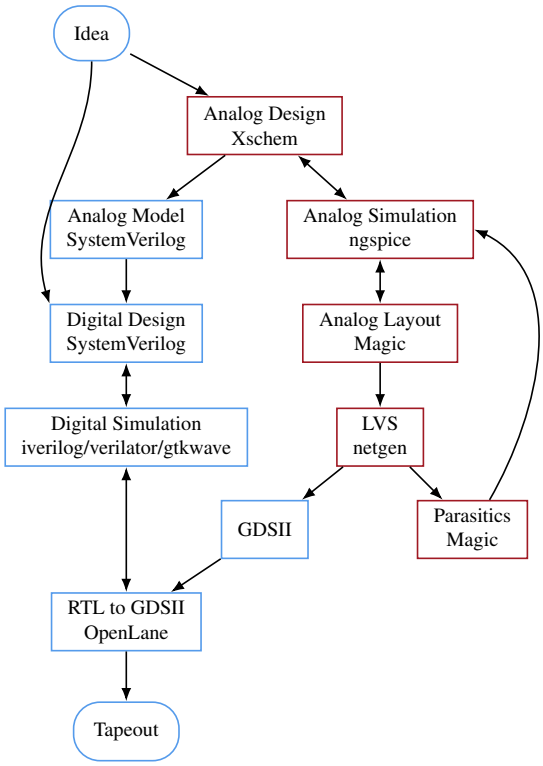


Figure 1: Mixed signal design flow, where the analog design is captured both as a schematic for ngspice and as a SystemVerilog model for the digital simulation

28.1 Digital simulation

Conceptually, the digital simulator is easy.

- The order of execution of events at the same time-step does not matter
- The system is causal. Changes in the future do not affect signals in the past or the now

In a digital simulator there will be an event queue, see below. From start, set the current time step equals to the next time step. Check if there are any events scheduled for the time step. Assume that execution of events will add new time steps. Check if there is another time step, and repeat.

Since the digital simulator only acts when something is supposed to be done, they are inherently fast, and can handle complex systems.

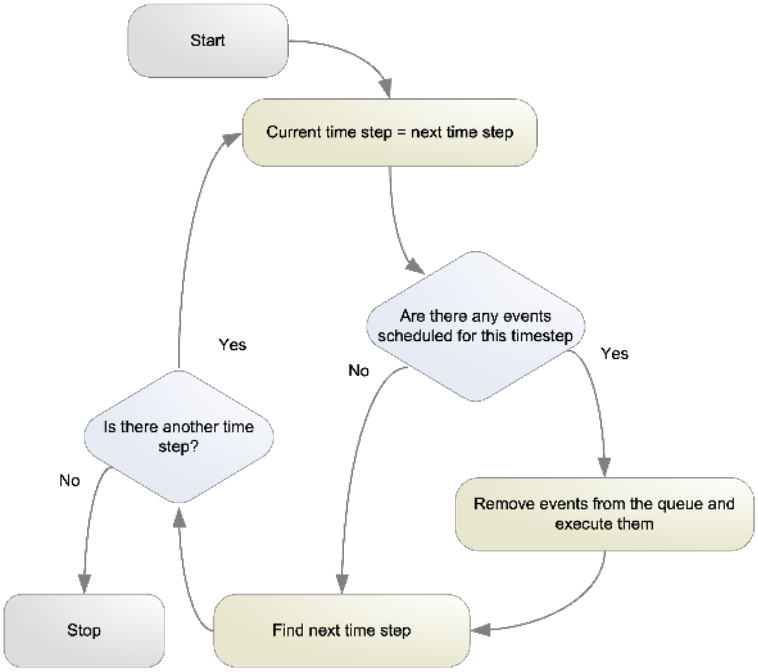


Figure 2: Flow chart of a digital simulator - advance to the next time step, execute the events scheduled there, and repeat until the queue is empty

It's a fun exercise to make a digital simulator. On my Ph.D I wanted to model ADCs, and first I had a look at SystemC, however, I disliked C++, so I made SystemDotNet

In SystemDotNet I implemented the event queue as a hash table, so it ran a bit faster. See below.

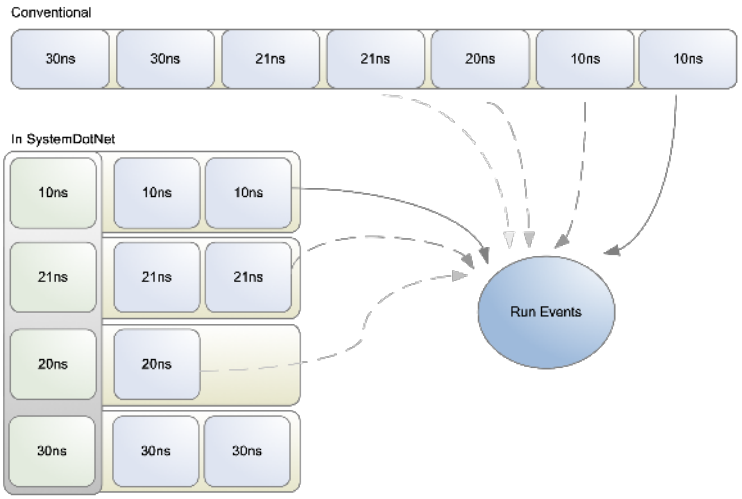


Figure 3: Event queue implemented as a hash table in SystemDotNet, where events at 10, 20, 21 and 30 ns are looked up by time step before being run. Source: the SystemDotNet project on SourceForge

28.1.1 Digital Simulators

There are both commercial and open source tools for digital simulation. If you've never used a digital simulator, then I'd recommend you start with iverilog. I've made some examples at dicex.

Commercial

- Cadence Excelium
- Siemens Questa
- Synopsys VCS

Open Source - iverilog/vpp - Verilator - SystemDotNet

28.1.1.1 Counter

Below is an example of a counter in SystemVerilog. The code can be found at counter_sv.

In the always_comb section we code what will become the combinatorial logic. In the always_ff section we code what will become our registers.

```
module counter(
               output logic [WIDTH-1:0] out,
               input logic              clk,
               input logic              reset
               );

   parameter WIDTH = 8;

   logic [WIDTH-1:0]                    count;
   always_comb begin
      count = out + 1;
   end

   always_ff @(posedge clk or posedge reset) begin
      if (reset)
        out <= 0;
      else
        out <= count;
   end

endmodule // counter
```

In the context of a digital simulator, we can think through how the event queue will look.

When the clk or reset changes from zero to 1, then schedule an event where if the reset is 1, then out will be zero in the next time step. If reset is 0, then out will be `count` in the next time step.

In a time-step where `out` changes, then schedule an event to set `count` to `out` plus one. As such, each positive edge of the clock at least 2 events must be scheduled in the register transfer level (RTL) simulation.

For example:

```
Assume `clk, reset, out = 0`

Assume event with `clk = 1`

0: Set `out = count` in next event (1)

1: Set `count = out + 1` using
   logic (may consume multiple events)

X: no further events
```

When we synthesis the code below into a netlist it's a bit harder to see how the events will be scheduled, but we can notice that clk and reset are still inputs, and for example the clock is connected to d-flip-flops. The image below is the synthesized netlist

It should feel intuitive that a gate-level netlist will take longer to simulate than an RTL, there are more events.

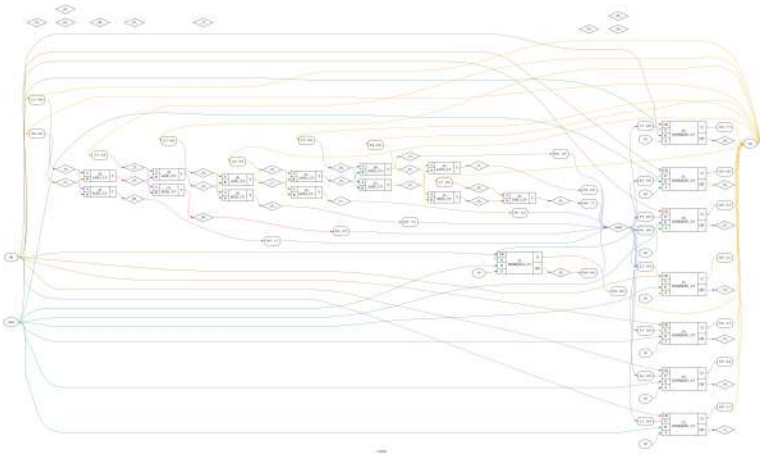


Figure 4: Gate level netlist of the counter after synthesis, with clk and reset driving eight D flip-flops and the adder logic

28.2 Transient analog simulation

Analog simulation is different. There is no quantized time step. How fast "things" happen in the circuit is entirely determined by the time constants, change in voltage, and change in current in the system.

It is possible to have a fixed time-step in analog simulation, for example, we say that nothing is faster than 1 fs, so we pick that as our time step. If we wanted to simulate 1 s, however, that's at least 1e15 events, and with 1 event per microsecond on a computer it's still a simulation time of 31 years. Not a viable solution for all analog circuits.

Analog circuits are also non-linear, properties of resistors, capacitors, inductors, diodes may depend on the voltage or current across, or in, the device. Solving for all the non-linear differential equations is tricky.

An analog simulation engine must parse spice netlist, and setup partial/ordinary differential equations for node matrix

The nodal matrix could look like the matrix below, i are the currents, v the voltages, and G the conductances between nodes.

$$\begin{pmatrix} G_{11} & G_{12} & \cdots & G_{1N} \\ G_{21} & G_{22} & \cdots & G_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ G_{N1} & G_{N2} & \cdots & G_{NN} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix} = \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{pmatrix}$$

The simulator, and devices model the non-linear current/voltage behavior between all nodes

as such, the G's may be non-linear functions, and include the v's and i's.

Transient analysis use numerical methods to compute time evolution

The time step is adjusted automatically, often by proprietary algorithms, to trade accuracy and simulation speed.

The numerical methods can be forward/backward Euler, or the others listed below.

- Euler
- Runge-Kutta
- Crank-Nicolson
- Gear [1]

If you wish to learn more, I would recommend starting with the original paper on analog transient analysis.

SPICE (Simulation Program with Integrated Circuit Emphasis) published in 1973 by Nagel and Pederson

The original paper has spawned a multitude of commercial, free and open source simulators, some are listed below.

If you have money, then buy Cadence Spectre. If you have no money, then start with ngspice.

Commercial - Cadence Spectre - Siemens Eldo - Synopsys HSPICE

Free - Aimspice - Analog Devices LTspice - xyce

Open Source - ngspice

28.3 Mixed signal simulation

It is possible to co-simulate both analog and digital functions. An illustration is shown below.

The system will have two simulators, one analog, with transient simulation and differential equation solver, and a digital, with event queue.

Between the two simulators there would be analog-to-digital, and digital-to-analog converters.

To orchestrate the time between simulators there must be a global event and time-step control. Most often, the digital simulator will end up waiting for the analog simulator.

The challenge with mixed-mode simulation is that if the digital circuit becomes to large, and the digital simulation must wait for analog solver, then the simulation would take too long.

Most of the time, it's stupid to try and simulate complex system-on-chip with mixed-signal , full detail, simulation.

For IPs, like an ADC, co-simulation works well, and is the best way to verify the digital and analog.

But if we can't run mixed simulation, how do we verify analog with digital?

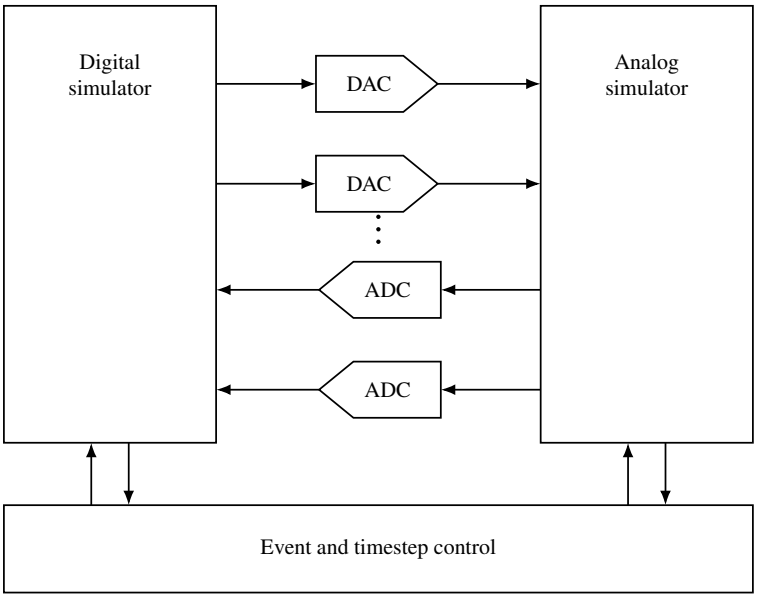


Figure 5: Mixed signal simulator, where a digital and an analog simulator exchange values through DAC and ADC connect modules under a shared event and timestep control

28.4 Analog SystemVerilog Example

28.4.1 TinyTapeout TT06_SAR

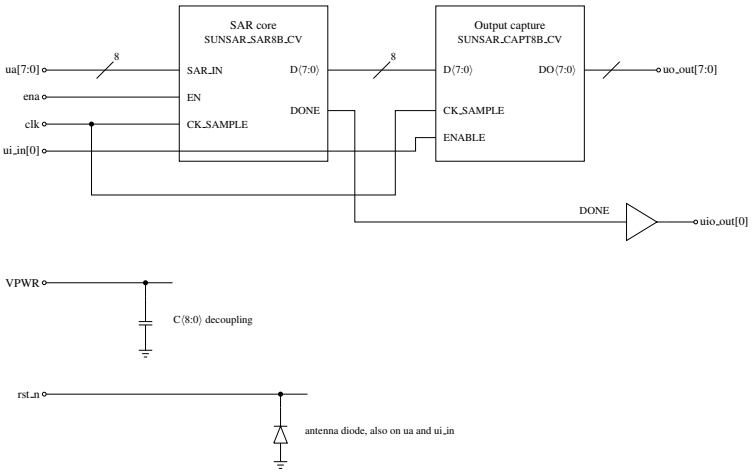


Figure 6: Top level of the TinyTapeout tt_um_TT06_SAR_wulffern 8-bit SAR ADC, redrawn from the xschem schematic: the SAR core, the output capture block, the buffered DONE on uio_out[0], the power decoupling and the antenna diodes on the inputs

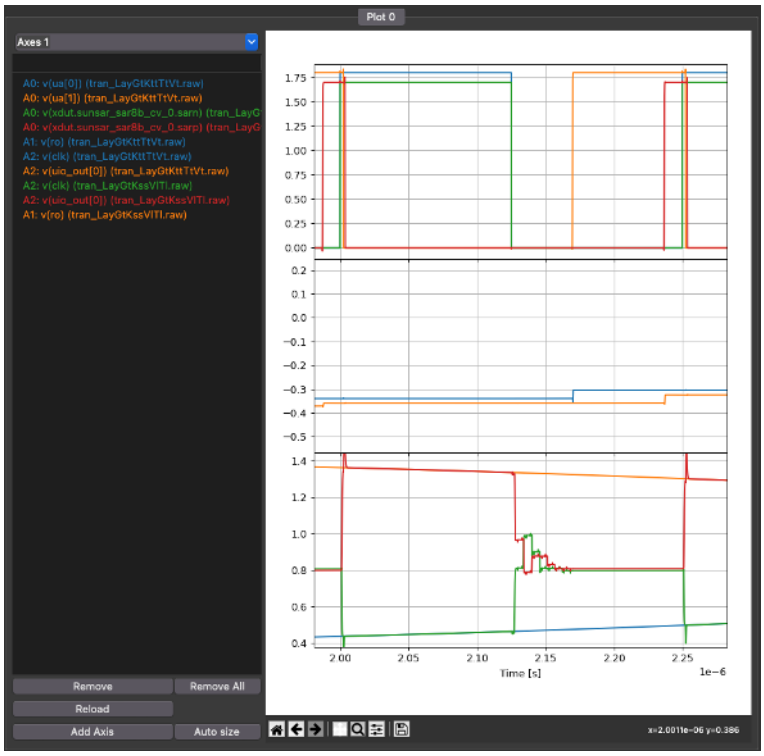


Figure 7: ngspice transient waveforms of the SAR, showing the sampled analog input, the clock and uio_out[0], and the sarp/sarn comparator inputs settling through the bit cycling

28.4.2 SAR operation

The key idea is to model the analog behavior to sufficient detail such that we can verify the digital code. I think it's best to have a look at a concrete example.

- Analog input is sampled when clock goes low (sarp/sarn)
- uio_out[0] goes high when bit-cycling is done
- Digital output (ro) changes when uio_out[0] goes high

```
//tt06-sar/src/project.v
module tt_um_TT06_SAR_wulffern (
                                input wire        VGND,
                                input wire        VPWR,
                                input wire [7:0]  ui_in,
                                output wire [7:0] uo_out,
                                input wire [7:0]  uio_in,
                                output wire [7:0] uio_out,
                                output wire [7:0] uio_oe,
`ifdef ANA_TYPE_REAL
                                input real        ua_0,
                                input real        ua_1,
`else
                                // analog pins
                                inout wire [7:0]  ua,
`endif
                                input wire        ena,
                                input wire        clk,
                                input wire        rst_n
                                );

//tt06-sar/src/tb_ana.v
`ifdef ANA_TYPE_REAL
   real        ua_0  = 0;
   real        ua_1 = 0;

`else
   tri [7:0]   ua;
   logic       uain = 0;
   assign ua = uain;
`endif

`ifdef ANA_TYPE_REAL
   always #100 begin
      ua_0 = $sin(2*3.14*1/7750*$time);
      ua_1 = -$sin(2*3.14*1/7750*$time);
   end
`endif


//tt06-sar/src/tb_ana.v
   tt_um_TT06_SAR_wulffern dut (
                                .VGND(VGND),
                                .VPWR(VPWR),
                                .ui_in(ui_in),
                                .uo_out(uo_out),
                                .uio_in(uio_in),
                                .uio_out(uio_out),
                                .uio_oe(uio_oe),
`ifdef ANA_TYPE_REAL
                                .ua_0(ua_0),
                                .ua_1(ua_1),
`else
                                .ua(ua),
```

```
`endif
                                .ena(ena),
                                .clk(clk),
                                .rst_n(rst_n)
                                );

#tt06-sar/src/Makefile
runa:
    iverilog  -g2012 -o my_design -c tb_ana.fl -DANA_TYPE_REAL
    vvp -n my_design

rund:
    iverilog  -g2012 -o my_design -c tb_ana.fl
    vvp -n my_design

   //tt06-sar/src/project.v
   //Main SAR loop
   always_ff @(posedge clk or negedge clk) begin
      if(~ui_in[0]) begin
         state <= OFF;
         tmp = 0;
         dout = 0;
      end
      else begin
         if(OFF) begin

         end
         else if(clk == 1) begin
            state = SAMPLE;
         end
         else if(clk == 0) begin
            state = CONVERT;
      `ifdef ANA_TYPE_REAL
            smpl = ua_0 - ua_1;
            tmp = smpl;

            for(int i=7;i>=0;i--) begin
               if(tmp >= 0) begin
                  tmp = tmp - lsb*2**(i-1);
                  if(i==7)
                    dout[i] <= 0;
                  else
                    dout[i] <= 1;
               end


               else begin
                  tmp = tmp + lsb*2**(i-1);
                  if(i==7)
                    dout[i] = 1;
                  else
                    dout[i] = 0;
               end
            end
`else
            if(tmp == 0) begin
               dout[7] <= 1;
               tmp <= 1;

            end
            else begin
               dout[7] <= 0;
               tmp  = 0;
            end
`endif

         end
```

```
      state = next_state;
   end // else: !if(~ui_in[0])
end // always_ff @ (posedge clk)
```

```
//tt06-sar/src/project.v
always @(posedge done) begin
   state = DONE;
   sampled_dout = dout;
end

always @(state) begin
   if(state == OFF)
     #2 done = 0;
   else if(state == SAMPLE)
     #1.6 done = 0;
   else if(state == CONVERT)
     #115 done = 1;
end
```

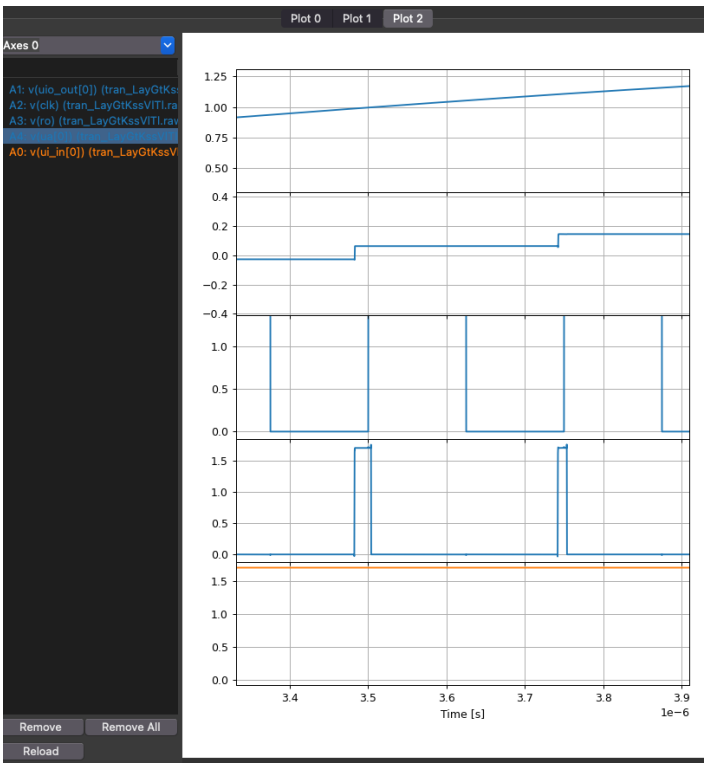


Figure 8: ngspice transient of the SAR at layout corner, with the ramping analog input, the digital output code, the clock and the uio_out[0] done pulses

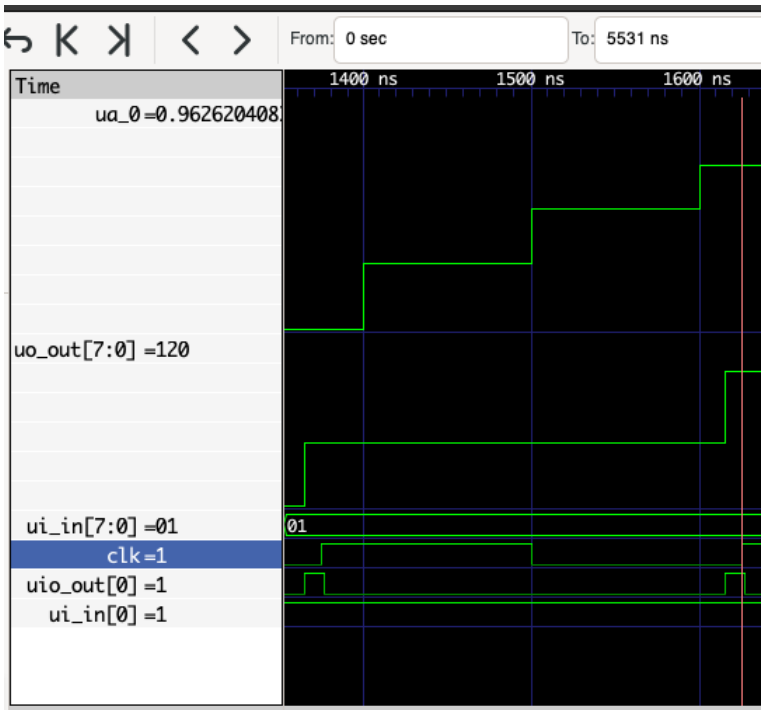


Figure 9: The same behaviour in gtkwave from the SystemVerilog real-number model, where uo_out[7:0] steps as ua_0 ramps

28.5 Summary

The one-page version of this chapter:

- Digital simulators are fast because they only compute at events; analog simulators are slow because they solve the whole matrix at every timestep
- Analog SystemVerilog models the analog blocks with real-valued signals inside the digital simulator - the mixed-signal chip simulates at digital speed
- The model is a contract: it must match the schematic's behavior at the pins, and drift between them is found in co-simulation
- A SAR ADC in TinyTapeout shows the whole flow: the same testbench drives the Verilog model and the transistor netlist
- Verification from zero supply, every time - models make that cheap enough to actually do

28.6 Would you like to know more?

For more information on real-number modeling I would recommend The Evolution of Real Number Modeling

[1] C. Gear, "Simultaneous numerical solution of differential-algebraic equations," *IEEE Transactions on Circuit Theory*, vol. 18, no. 1, pp. 89–95, 1971, doi: 10.1109/TCT.1971.1083221.

How to write a project report

29

Keywords: Writing, Paragraphs, Transitions, Structure, Introduction, Theory, Implementation, Results

29.1 Why 555
29.2 On writing English 555
29.2.1 Shorter is better . . 555
29.2.2 Be careful with adjectives 556
29.2.3 Use paragraphs . . . 556
29.2.4 Don't be afraid of I . 556
29.2.5 Transitions are important 556
29.2.6 However, is not a start of a sentence . 557
29.3 Report Structure . . 557
29.3.1 Introduction 557
29.3.2 Theory 558
29.3.3 Implementation . . 558
29.3.4 Result 558
29.3.5 Discussion 558
29.3.6 Future work 559
29.3.7 Conclusion 559
29.3.8 Appendix 559
29.4 Checklist 559
29.5 Summary 560
29.6 Would you like to know more? 560

29.1 Why

> Them who has a Why? in life can tolerate almost any How?

You're writing the report on the project for me to be able to see inside your head, and grade how much of the project you have understood.

- Have you learned what is to be expected?
- Do you understand what you're trying to explain?

You will work on the project in groups, however, on the report, you will write on your own.

That means, that there will be X projects reports that describe the same circuit. You shall not copy someone elses report text.

It's fine to share figures between reports, and also references.

I'm also forcing you to use a report format that matches well with what would be expected if you were to publish a paper.

29.2 On writing English

Writing well is important. I would recommend that you read On writing Well.

Most of you won't buy the book, as such, a few tips.

29.2.1 Shorter is better

I can write the section title idea in many words:

> A shorter text will more eloquently describe the intricacies of your thoughts than a long, distinguished, tirade of carefully, wonderfully, chosen words.

or

Shorter is better

Describe an idea with as few words as possible. The text will be better, and more readable.

29.2.2 Be careful with adjectives

Words like "very, extremely, easily, simply, . . . " don't belong in a readable text. They serve no purpose. Delete them.

29.2.3 Use paragraphs

You write a text to place ideas into anothers head. Ideas and thoughts are best communicated in chunks. I can write a dense set of text, or I can split a dense set of text into multiple paragraphs. The more I try to cram into a paragraph, for example, how magical the weather has been the last weeks, with lots of snow, and good skiing, the more difficult the paragraph is to read.

One paragraph, one thought. For example:

You write a text to place ideas into anothers head. Ideas and thoughts are best communicated in chunks.

I can write a dense set of text, or I can split a dense set of text into multiple paragraphs.

The more I try to cram into a paragraph, for example, how magical the weather has been the last weeks, with lots of snow, and good skiing, the more difficult the paragraph is to read.

29.2.4 Don't be afraid of I

If you did something, then say "I" in the text. If there were more people, then use "we".

29.2.5 Transitions are important

Sentences within a paragraph are sometimes linked. Use

- As a result,
- As such,
- Accordingly,
- Consequently,

And mix them up.

29.2.6 However, is not a start of a sentence

If you have to use "However" it should come in the middle of the sentence.

I want to go skiing, however, I cannot today due to work.

29.3 Report Structure

The sections below go through the expected structure of a report, and what the sections should contain.

29.3.1 Introduction

The purpose of the introduction is to put the reader into the right frame of mind. Introduce the problem statement, key references, the key contribution of your work, and an outline of the work presented. Think of the introduction as explaining the "Why" of the work.

Although everyone has the same assignment for the project, you have chosen to solve the problem in different ways. Explain what you consider the problem statement, and tailor the problem statement to what the reader will read.

Key references are introduced. Don't copy the paper text, write why they designed the circuit, how they chose to implement it, and what they achieved. The reason we reference other papers in the introduction is to show that we understand the current state-of-the-art. Provide a summary where state-of-the-art has moved since the original paper.

The outline should be included towards the end of the introduction. The purpose of the outline is to make this document easy to read. A reader should never be surprised by the text. All concepts should be eased into. We don't want the reader to feel like they've been thrown in at the end of a long story. As such, if you chosen to solve the problem statement in a way not previously solved in a key references, then you should explain that.

A checklist for all chapters can be seen in table below.

29.3.2 Theory

It is safe to assume that all readers have read the key references, if they have not, then expect them to do so.

The purpose of the theory section is not to demonstrate that you have read the references, but rather, highlight theory that the reader probably does not know.

The theory section should give sufficient explanation to bridge the gap between references, and what you apply in this text.

29.3.3 Implementation

The purpose of the implementation is to explain what you did. How have you chosen to architect the solution, how did you split it up in analog and digital parts? Use one subsection per circuit.

For the analog, explain the design decisions you made, how did you pick the transistor sizes, and the currents. Did you make other choices than in the references? How does the circuit work?

For the digital, how did you divide up the digital? What were the design choices you made? How did you implement readout of the data? Explain what you did, and how it works. Use state diagrams and block diagrams.

Use clear figures (i.e. circuitikz), don't use pictures from schematic editors.

29.3.4 Result

The purpose of the results is to convince the reader that what you made actually works. To do that, explain testbenches and simulation results. The key to good results is to be critical of your own work. Do not try to oversell the results. Your results should speak for themselves.

For analog circuits, show results from each block. Highlight key parameters, like current and delay of comparator. Demonstrate that the full analog system works.

Show simulations that demonstrate that the digital works.

29.3.5 Discussion

Explain what the circuit and results show. Be critical.

29.3.6 Future work

Give some insight into what is missing in the work. What should be the next steps?

29.3.7 Conclusion

Summarize why, how, what and what the results show.

29.3.8 Appendix

Include in appendix the necessary files to reproduce the work. One good way to do it is to make a github repository with the files, and give a link here.

29.4 Checklist

Item	Description	OK
Is the problem description clearly defined?	Describe which parts of the problem you chose to focus on. The problem description should match the results you've achieved.	
Is there a clear explanation why the problem is worth solving?	The reader might need help to understand why the problem is interesting	
Is status of state-of-the-art clearly explained?	You should make sure that you know what others have done for the same problem. Check IEEEXplore. Provide summary and references. Explain how your problem or solution is different	
Is the key contribution clearly explained?	Highlight what you've achieved. What was your contribution?	
Is there an outline of the report?	Give a short summary of what the reader is about to read	
Is it possible for a reader skilled in the art to understand the work?	Have you included references to relevant papers	
Is the theory section too long	The theory section should be less than 10 % of the work	
Are all circuits explained?	Have you explained how every single block works?	
Are figures clear?	Remember to explain all colors, and all symbols. Explain what the reader should understand from the figure. All figures must be referenced in the text.	
Is it clear how you verified the circuit?	It's a good idea to explain what type of testbenches you used. For example, did you use dc, ac or transient to verify your circuit?	
Are key parameters simulated?	You at least need current from VDD. Think through what you would need to simulate to prove that the circuit works.	

Item	Description	OK
Have you tried to make the circuit fail?	Knowing how circuits fail will increase confidence that it will work under normal conditions.	
Have you been critical of your own results?	Try to look at the verification from different perspectives. Play devil's advocate, try to think through what could go wrong, then explain how your verification proves that the circuit does work.	
Have you explained the next steps?	Imagine that someone reads your work. Maybe they want to reproduce it, and take one step further. What should that step be?	
No new information in conclusion.	Never put new information into conclusion. It's a summary of what's been done	
Story	Does the work tell a story, is it readable? Don't surprise the reader by introducing new topics without background information.	
Chronology	Don't let the report follow the timeline of the work done. What I mean by that is don't write "first I did this, then I spent huge amount of time on this, then I did that". No one cares what the timeline was. The report does not need to follow the same timeline as the actual work.	
Too much time	How much time you spent on something should not be correlated to how much text there is in the report. No one cares how much time you spent on something. The report is about why, how, what and does it work.	
Length	A report should be concise. Only include what is necessary, but no more. Shorter is almost always better than longer.	
Template	Use IEEEtran.cls. Example can be seen from an old version of this document at https://github.com/wulffern/dic2021/tree/main/2021-10-19_project_report. Write in LaTeX. You will need LaTeX for your project and master thesis. Use http://overleaf.com if you're uncomfortable with local text editors and LaTeX.	
Spellcheck	Always use a spellchecker. Misspelled words are annoying, and may change content and context (peaked versus piqued)	

29.5 Summary

The one-page version of this chapter:

- The report is the product: a reader must be able to repeat what you did and believe what you claim
- Abstract states the claim, introduction the why, method the how, results the evidence, conclusion the so-what
- Figures carry the argument - every one referenced, every one earning its place
- Cite what you used, and let the results speak for themselves

29.6 Would you like to know more?

Strunk and White, *The Elements of Style* - still the shortest useful book on writing

Zinsser, *On Writing Well* - the friendliest one

A handful of JSSC papers, read for their structure rather than their content

30 Layout Generation

Keywords: Layout Generation, CICPY, Placement, Routing, Automation

30.1 Layout 563
30.2 Setup 563
30.3 CICPY 563
30.4 Placement 564
30.5 Summary 566
30.6 Would you like to know more? 566

30.1 Layout

The open source tools don't have any automatic analog layout. To my knowledge, there is no general purpose analog automagic layout anywhere in the world. It's an unsolved problem. Many have tried (including myself), but none have succeeded with a generic analog layout engine.

There are a few things, though, that could help you on the way.

30.2 Setup

I assume that you have the latest and greatest `aicex\ip` setup.

See SKY130NM Tutorial if aicex is unfamiliar.

Let's assume we use `jnw_gr05_sky130a` to test out our layout

```
cd aicex/ip/
cd jnw_gr05_sky130a
git checkout a1e3dfc324194729e042f5e653777b052759863b
cd work
```

30.3 CICPY

The first thing we need to do is to place all transistors. I do have a script to help. Install cicpy.

```
cd aicex/ip/cicpy
git checkout master
git pull
python3 -m pip install -e .
cd ..
cd cicspi
git checkout main
git pull
python3 -m pip install -e .
```

30.4 Placement

To generate an initial placement we can do the command below. If a layout exists it will be overridden

```
cd jnw_gr05_sky130a/work
cicpy sch2mag JNW_GR05_SKY130A OTA_Manuel
```

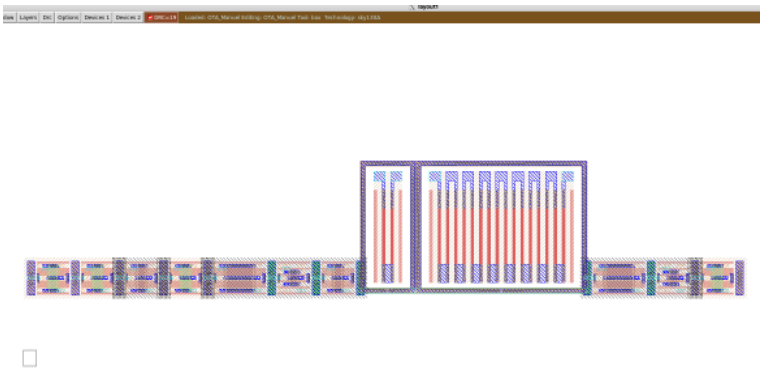


Figure 1: Initial cicpy sch2mag placement of OTA_Manuel in Magic - all transistors in one row, 19 DRC errors

The layout engine has no idea what components belong together, for example, the current mirror below should have been placed together

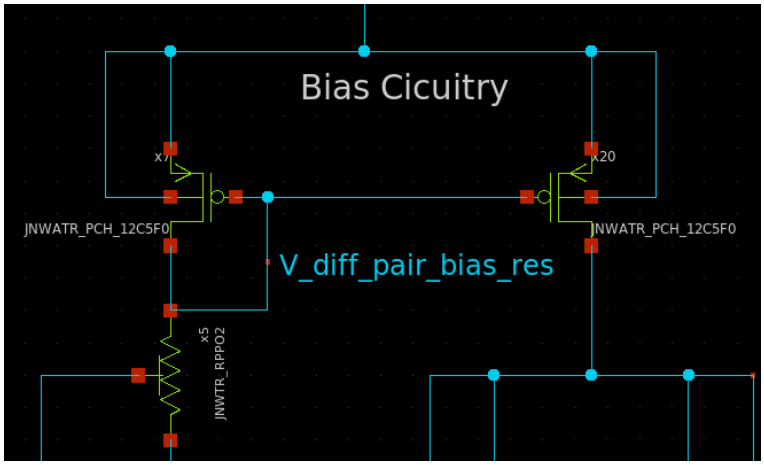


Figure 2: Xschem schematic of the OTA bias circuitry, where the current mirror pair xa07/xa20 should have been placed together

We can instruct the layout engine by adding a "group" name to the instance name. The instance name always starts with `x<something><number>` where the something can be nothing, or a group name (a,b, not a number).

The rules for placement are:

1. Sort all instances by groups
2. Sort all groups by instance name

3. Place the first instance.
4. For all instances: If the next instance has the same group, then add on top. Otherwise increment the x location.

As such, if I rename my instances, as shown below,

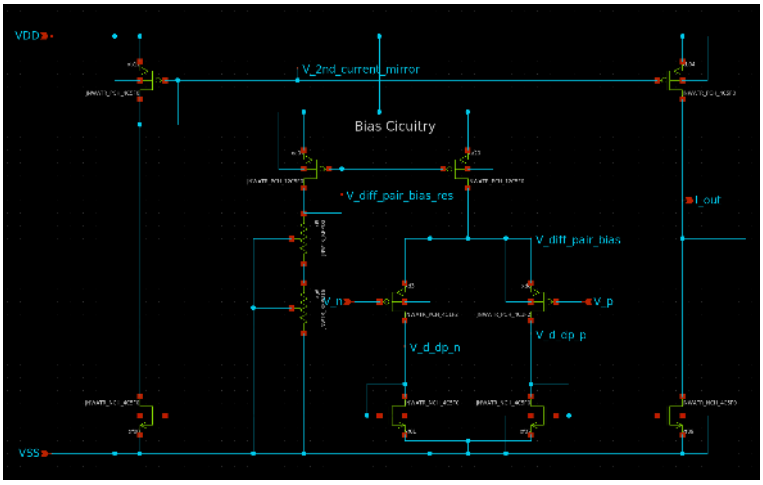


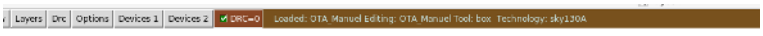
Figure 3: The same OTA schematic after renaming instances with group prefixes (xa, xb, xd, xf, xg) to guide the placer

Then the layout becomes a bit better

```
cicpy sch2mag JNW_GR05_SKY130A OTA_Manuel --gbreak 3 --xspace 34000 --yspace 30000
```

The gbreak command inserts a "group break" after the fourth group, such that a new Y coordinate is selected.

The X and Y space is for the distance between groups. The unit is "Ångstrøm", so 1 um is 10 000 Å.



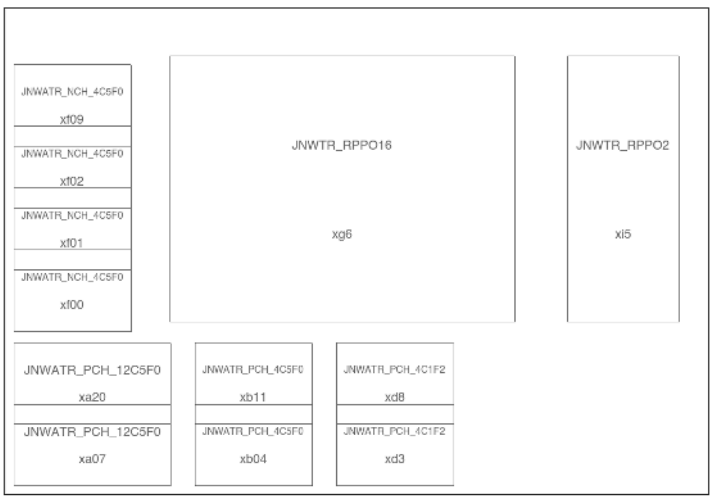
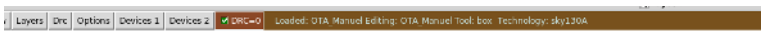


Figure 4: Placement after grouping, shown as instance boxes: devices of the same group stack vertically, and –gbreak 3 starts a new row



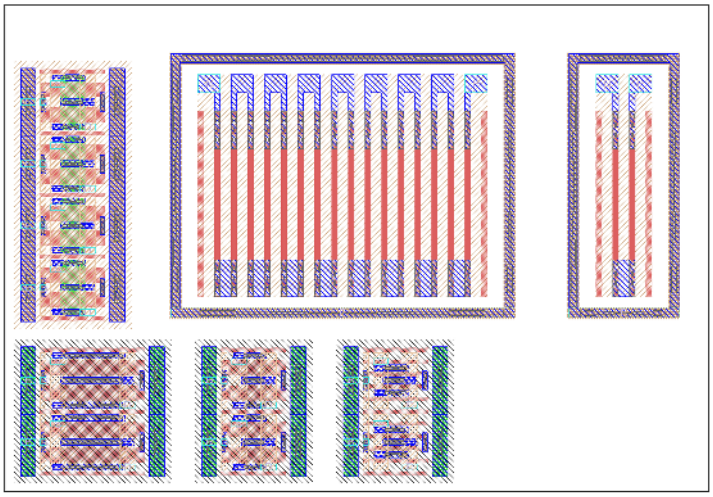


Figure 5: The same grouped placement with all layers drawn, DRC clean

30.5 Summary

The one-page version of this chapter:

- Layout is the final translation: from schematic to the polygons the fab will etch
- Matching is geometry: symmetry, proximity, dummies, and common centroid where it counts
- The parasitics are part of the circuit - extract and re-simulate before believing any layout
- DRC and LVS are not suggestions: clean both, every time

30.6 Would you like to know more?

Hastings, *The Art of Analog Layout* - the book on matching, and worth owning

The Magic and netgen documentation, which is the practical companion for the flow used here

Thoughts and Advice 31

Keywords: Advice, Skills, Zen of IC Design, Design Process, Career

31.1 Advice 567
31.1.1 What the team needs to know to design ICs 568
31.1.2 Zen of IC design (stolen from Zen of Python) 569
31.1.3 IC design mantra . . 569
31.1.4 Analog Design Process 569
31.2 Stuff to ponder . . . 570
31.2.1 AC open, DC closed switch 570
31.3 Summary 576
31.4 Would you like to know more? 576

31.1 Advice

This is some advice, use it, or ignore it, who cares.

Try to figure out what makes you happy, and do more of that

If you don't know how to say sorry when you do something stupid, learn.

When life sucks, run, or exercise, it's the only thing that works

Get a mac, time machine, and offsite backup. That ensures you'll never lose data.

Find a problem that you really want to solve, and learn a programming language to solve it. There is absolutely no point in saying "I want to learn programming", then sitting down with a book to read about programming, and expect that you will learn programming that way. It will not happen. The only way to learn programming is to program, a lot.

Learn to check your assumptions. You will make mistakes, and you need to get good at finding the mistakes you made.

Take your time to write a verification plan. And stick to it. Without sufficient simulation your circuit will not work.

The table below is my current view of the abstraction levels of analog design automation - what is solved, what is in progress, and what is still risky:

Status	Abstraction	Design	Layout	Why
:construction:	Chip	SystemVerilog	digital	Complex connections, few analog interfaces
:construction:	Module	SystemVerilog	digital	Large amount of digital signals, few analog signals
:warning:	Block	Schematic	programmatic	Large amount of critical analog interfaces, few digital
:white_check_mark:	Cell	Netlist/JSON	compiled	Few analog interfaces, few digital interfaces
:white_check_mark:	Device	JSON	compiled	Polygon pushing

Status	Abstraction	Design	Layout	Why
:white_-check_-mark:	Technology	JSON/Rules	compiled	Custom for each technology

31.1.1 What the team needs to know to design ICs

There are a multitude of tools and skills needed to design professional ICs. It's not likely that you'll find all the skills in one human, and even if you could, one human does not have sufficient bandwidth to design ICs with all it's aspects in a reasonable timeline

That is, unless we can find a way to make ICs easier.

The skills needed are

- *Project flow support*: **Confluence**, JIRA, risk management (DFMEA), failure analysis (8D)
- *Language*: **English**, **Writing English (Latex, Word, Email)**
- *Psychology*: Personalities, convincing people, presentations (Powerpoint, Deckset), **stress management (what makes your brain turn off?)**
- *DevOps*: **Linux**, build systems (CMake, make, ninja), continuous integration (bamboo, jenkins), **version control (git)**, containers (docker), container orchestration (swarm, kubernetes)
- *Programming*: Python, C, C++, Matlab *Since 1999 I've programmed in Python, Go, Visual BASIC, PHP, Ruby, Perl, C#, SKILL, Ocean, Verilog-A, C++, BASH, AWK, VHDL, SPICE, MATLAB, ASP, Java, C, SystemC, Verilog, Assembler, and probably a few I've forgotten.*
- *Firmware*: signal processing, algorithms, software architecture, security
- *Infrastructure*: **Power management**, **reset**, **bias**, **clocks**
- *Domains*: CPUs, peripherals, memories, bus systems
- *Sub-systems*: **Radio's**, **analog-to-digital converters**, **comparators**
- *Blocks*: **Analog Radio**, Digital radio baseband
- *Modules*: Transmitter, **receiver**, de-modulator, timing recovery, state machines
- *Designs*: **Opamps**, **amplifiers**, **current-mirrors**, adders, random access memory blocks, standard cells
- *Tools*: **schematic**, **layout**, **parasitic extraction**, synthesis, place-and-route, **simulation**, (System)Verilog, **netlist**
- *Physics*: transistor, pn junctions, quantum mechanics

31.1.2 Zen of IC design (stolen from Zen of Python)

When you learn something new, it's good to listen to someone that has done whatever it is before.

Here is some guiding principles that you'll likely forget.

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts (especially schematics).
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one **and preferably only one** obvious way to do it.
- Now is better than never.
- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.

31.1.3 IC design mantra

To copy an old mantra I have on learning programming

> Find a problem that you really want to solve, and learn programming to solve it. There is no point in saying "I want to learn programming", then sit down with a book to read about programming, and expect that you will learn programming that way. It will not happen. The only way to learn programming is to do it, a lot. – Carsten Wulff

And run the perl program

```
s/programming/analog design/ig
```

31.1.4 Analog Design Process

- Define the problem, what are you trying to solve?
- Find a circuit that can solve the problem (papers, books)
- Find right transistor sizes. What transistors should be weak inversion, strong inversion, or don't care?
- Write a verification plan. Plan to simulate everything that could go wrong.
- Check operating region of transistors (.op)

- Check key parameters (.dc, .ac, .tran)
- Check function. Exercise all inputs. Check all control signals
- Check key parameters in all corners. Check mismatch (Monte-Carlo simulation)
- Do layout, and check it's error free. Run design rule checks (DRC). Check layout versus schematic (LVS)
- Extract parasitics from layout. Resistance, capacitance, and inductance if necessary.
- On extracted parasitic netlist, check key parameters in all corners and mismatch (if possible).
- If everything works, then you're done.

On failure, go back as far as necessary

31.2 Stuff to ponder

Over a period of 10 months I was fortunate to spend some time at Electronics and Computer Engineering Department, University of Toronto. I was there as a grad student doing research on Comparator Based Switched Capacitor Circuits. Each Wednesday we had a meeting with the other Ph.D. and Master students, which was attended by Professor Ken Martin, Professor David Johns and Professor Trond Ytterdal. The quotes and tips here should not be taken as facts, but rather as "heads-up'' statements. Take these tips as something that should be checked and thought about. I do not remember which quotes/tips came from which professor, or indeed which student. So here goes

This is important: Do not worry about unknowns. Make a list of unknowns and find a test to check whether the unknown is a problem. Fixing things based on guesses will cause trouble.

31.2.1 AC open, DC closed switch

In SPICE there is usually a switch or capacitor/inductor that has the behavior of being open at AC and closed at DC or visa versa. Useful for setting common mode voltages in simulation of differential operational transconductance amplifiers.

31.2.1.1 Measure capacitance

To measure capacitance on a node in a circuit simulation.

Bias the block Put a small dc current into the node Measure the delta V over a short time period Calculate capacitance from i = C

dv/dt Always include a replica with a known capacitance value, i.e. a capacitor, to check your testbench.

31.2.1.2 On Analog Design

Normally source jitter will dominate

This comment was made in reference to a 10-bit 50MHz ADC. So if you're designing such a ADC you probably don't have to worry about jitter in the clock circuit. However, you should be careful about your clock input. I know some people do differential sinusoidal clocks and create a square wave clock on the inside of the chip. Using differential signaling will help with possible interference from nearby lines.

31.2.1.3 Distortion from ESD protection circuits

When you go to high resolutions (> 10 bit) and high speed (>50MHz) the non-linear capacitance of the ESD protection starts to matter. If you're doing an ADC above this area you should read Analysis and Measurement of Signal Distortion due to ESD Protection Circuits.

31.2.1.4 Add net names in layout

Always add net names to layout nets, this will help LVS to match nets. It will also save you when tracking down shorts.

31.2.1.5 Decouple to source node

In current mirrors, decouple to the source node. By decoupling between source and for example vss, any high frequency jumps on vss will also appear on the gate, thus the gate source voltage will stay constant and current will not change

31.2.1.6 Worry about current densities when routing > 20um

Metal wires on-chip have a maximum allowed dc current. This is due, among other things, to electromigration. At high current densities the aluminum atoms may migrate, and thus leave a void that might grow over time into a discontinuity. Why exactly >20um I don't know, but it was mentioned in a meeting as a rule of thumb. Current densities are usually around 1mA/square, but varies with technology

31.2.1.7 Always shield signal lines above 10 bit level

If you're doing an ADC, or indeed any circuit, that requires > 10 bit accuracy you should shield your signal lines. On chip you use metal below, above and sides. The same for PCBs. Sensitive signals can be routed in in-between layers.

31.2.1.8 Use a current source to feed inverter based oscillators

An inverter ring fed from a current source oscillates at a frequency set by the current rather than the supply, which buys supply rejection - see the current-starved ring in the oscillator chapter.

31.2.1.9 Check non-overlapping clocks in slow, high temp and low vdd

For non overlapping clocks you should check that the two clocks just meet in slow corner, high temperature and low vdd. By meet I mean one clock should start to rise when the other is almost at zero. Supposedly this PVT corner is the worst for non-overlap, but I have not checked.

31.2.1.10 Analog Sampling

If possible, you should sample analog just before digital IO switches. In other words, sample during quiet time.

31.2.1.11 Cascode devices should be minimum size

You get less capacitance this way.

31.2.1.12 The unit transistor W/L should be around 10-20

31.2.1.13 Place noisy digital blocks in deep N-well

By separating substrates you improve noise immunity

31.2.1.14 Shield analog blocks with deep N-well ring

Same thing as above

31.2.1.15 Always route differential signals differentially

Mismatch between parasitic capacitances/resistors in differential signal routing (differential means; two signals where one signal is phase shifted 180 degrees) can introduce errors. The error is reduced if the parasitics are matched, since the differential system cancels some of the errors.

31.2.1.16 On-chip decoupling

Remember to check whether you need on-chip decoupling of references and power. In most designs you do need decoupling, especially if you run at high speeds (> 10MHz).

31.2.1.17 Variable delayed clock

If jitter is not important, and you want a variable delayed clock, you can use a current starved inverter Place a current source inside or on the outside of your inverter, and use a current mirror to control the maximum current.

31.2.1.18 On Calibration & Test

Use serial shift registers for calibration bits

It is common to include some off-line startup calibration circuits in ICs. For example to tune transconductances, resistances, capacitances, offset voltages etc. Usually this leads to some form of DAC that needs a digital input. For these digital inputs a serial shift register should be used. Indeed for any digital input that does not have to be synchronous, use serial shifting. The best is to have a commercial bus like SPI or I2S, but this might be overkill.

Two registers should be used, one long shift register and one parallel load register. First you shift in all you calibration bits, then you load them into the parallel register. The calibration DACs are connected to the parallel register. This is to avoid any funny stuff happening when you load your bits. It's good to have control over the state of your circuit at all times. A long shift register is not a problem, it does not cost much to add some more bits. In a recent ADC I made, the shift register was 272 bits long.

The calibration register should have 4 inputs: data, data clock, reset, load. The load signal is used to do the parallel load after shifting in all the bits. You should also include a data output so you can check what was loaded in.

On all inputs you should use a Schmitt trigger to improve noise immunity. Especially since the input data and clock may be fed from a computer with slow rise and fall times.

31.2.1.19 Design for test

Make sure that all on-chip DC voltages (bias points, power, references) are available off-chip for measurement. Either through probe pads or analog test multiplexers.

Use analog test multiplexer

Use an analog test multiplexer with T-switches to give you access to internal nodes. A T-switch has two transmission gates and one NMOS. The first transmission gate is placed close to the analog node you want to test, the second close the test output. The NMOS is placed on the output side (not analog node side) of the first transmission gate. When the T-switch is ON the two transmission gates are closed and the NMOS is open. When the T-switch is OFF the NMOS grounds the long line between the transmission gates, thus preventing leakage between different test points of the analog multiplexer.

31.2.1.20 Calibration currents

If you have a current source with off-line calibration, the calibration current should be +- 50% versus nominal.

31.2.1.21 Calibration DACs

For calibration DACs use 3 or 4 thermometer encoded bits and the LSBs binary encoded.

31.2.1.22 Access to gain boosters

If you're using gain boosters the boost voltage should be accessible off-chip.

31.2.1.23 Default calibration state

The calibration DACs should start up in a default state close to the expected state. Use inverters between the calibration register and calibration DACs to set the default state.

31.2.1.24 Crystal Oscillators

Crystal oscillators are used to generate a clean, low phase noise (low jitter), clock signal.

31.2.1.25 Careful circuit board design

If you move into the +10 bit accuracy range the circuit board (PCB) becomes important. Especially if you're running at high frequencies. If you have designed and produced an ADC, you want to test the ADC performance, not the PCB. So you might want to exclude the circuit board as an error source. To do this you can include a parallel ADC of same or higher resolution to test the PCB performance.

I know of cases where a guy did a 12 bit ADC, made a circuit board, and tested. The test showed very poor performance < 10 bit, and it turned out to be the circuit board. Most of the extra noise was because he hadn't shielded his input signal and taken care of routing of input signal. He spent in excess of 3 months to track down the problem. And the solution was to redo the circuit board and take better care of the input signal.

Short circuit input on your ADC to measure noise floor

31.2.1.26 Single ended to differential converter

In the range 0Hz - 2MHz you can use ADC driver, AD8138 Analog Devices For single ended to differential conversion in the range > 2MHz a transformer should be used.

31.2.1.27 Non-harmonic spurs

When measuring your ADC, if you have spurs that are not harmonics in your FFT you should try to change the clock frequency (sampling frequency) to see if the spurs change frequency as well. If they do, they may be aliased interference from nearby RF transmitters.

31.2.1.28 Low capacitance probes are available, as low as 0.1pF - 2pF

31.2.1.29 Information sources

Handbook on filter synthesizing: Martin Snelgrove Phd thesis

State-Space Adaptive IIR Filters, David A. Johns

The Data Conversion Handbook, Walt Kester

31.3 Summary

The one-page version of this chapter:

- Automation climbs the abstraction ladder: devices and cells are solved, blocks are the frontier, chips are stitching
- Whatever is text - netlists, JSON, code - can be versioned, diffed, reviewed and compiled; keep everything as text
- The practical tips are paid for in silicon: feed ring oscillators from current sources, check non-overlap in the slow corner, leave a known capacitance to measure
- The goal is not to remove the designer; it is to stop the designer redrawing what a compiler can regenerate

31.4 Would you like to know more?

The ciccreator documentation, for the compiler this chapter argues for

OpenLane, the digital counterpart, for how far the same idea has gone on that side

SPICE 32

32.1 SPICE

Keywords: SPICE, Sources, Passives, Transistor Models, BSIM, Foundries, Unit Transistors, gm/ID

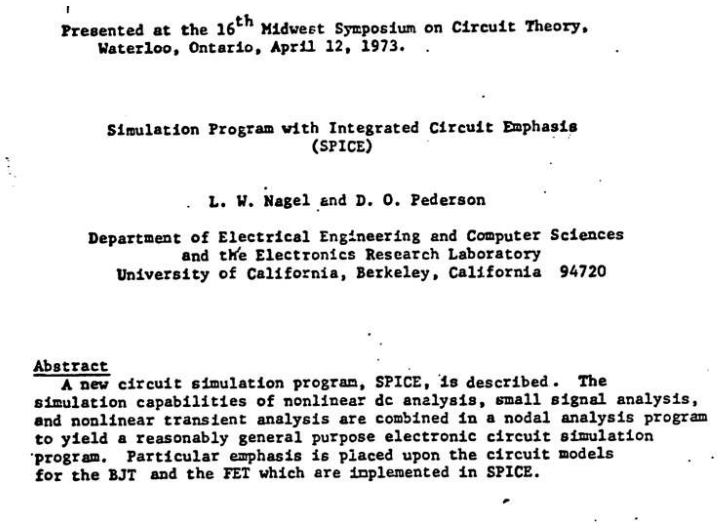
32.2 Simulation Program with Integrated Circuit Emphasis

To manufacture an integrated circuit we have to be able to predict how it's going to work. The only way to predict is to rely on our knowledge of physics, and build models of the real world in our computers.

One simulation strategy for a model of the real world, which absolutely every single integrated circuit in the world has used to come into existence, is SPICE.

Published in 1973 by Nagel and Pederson

SPICE (Simulation Program with Integrated Circuit Emphasis)



Presented at the 16th Midwest Symposium on Circuit Theory,
Waterloo, Ontario, April 12, 1973.

Simulation Program with Integrated Circuit Emphasis
(SPICE)

L. W. Nagel and D. O. Pederson

Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

Abstract

A new circuit simulation program, SPICE, is described. The simulation capabilities of nonlinear dc analysis, small signal analysis, and nonlinear transient analysis are combined in a nodal analysis program to yield a reasonably general purpose electronic circuit simulation program. Particular emphasis is placed upon the circuit models for the BJT and the FET which are implemented in SPICE.

Figure 1: Title page of Nagel and Pederson's 1973 SPICE paper from the 16th Midwest Symposium on Circuit Theory. Source: L. W. Nagel, SPICE2 memorandum ERL-M520, UC Berkeley, 1975

32.1 SPICE 577
32.2 Simulation Program with Integrated Circuit Emphasis . 577
32.2.1 Today 578
32.2.2 But 578
32.2.3 Sources 579
32.2.4 Passives 580
32.2.5 Transistor Models . 580
32.2.6 Transistors 581
32.2.7 Foundries 582
32.3 Find right transistor sizes 582
32.3.1 Use unit size transistors for analog design 583
32.3.2 What about gm/Id ? 583
32.3.3 Characterize the transistors 584
32.4 More information . 584
32.5 Analog Design . . . 584
32.6 Demo 584
32.7 Summary 585
32.8 Would you like to know more? 585

32.2.1 Today

There are multiple SPICE programs that has been written, but they all work in a similar fashion. There are expensive ones, closed source, and open source.

Some are better at dealing with complex circuits, some are faster, and some are more accurate. If you don't have money, then start with ngspice.

Commercial Cadence Spectre Siemens Eldo Synopsys HSPICE

Free Aimspice Analog Devices LTspice

Open Source ngspice

32.2.2 But

All SPICE simulators understand the same language (yes, even spectre can speak SPICE). We write our testbenches in a text file, and give it to the SPICE program. That's the same for all programs. Some may have built fancy GUI's to hide the fact that we're really writing text files, but text files is what is under the hood.

Pretty much the same usage model as 50-odd years ago

```
<spice program> testbench.cir
```

for example

```
ngspice testbench.cir
```

Or in the most expensive analog tool (Cadence Spectre)

```
spectre  input.scs  +escchars +log ../psf/spectre.out
-format psfxl -raw ../psf   +aps +lqtimeout 900 -maxw 5
-maxn 5 -env ade  -ahdllibdir
/tmp/wulff/virtuoso/TB_SUN_BIAS_GF130N/TB_SUN_BIAS/maestro/
results/maestro/Interactive.15/sharedData/CDS/ahdl/input.ahdlSimDB
+logstatus
```

The expensive tools have built graphical user interface around the SPICE simulator to make it easier to run multiple scenarios.

Corner	Typical	Fast	Slow	All
Mosfet	Mtt	Mff	Mss	Mff,Mfs,Msf,Mss
Resistor	Rt	Rl	Rh	Rl,Rh
Capacitors	Ct	Cl	Ch	Cl,Ch
Diode	Dt	Df	Ds	Df,Ds
Bipolar	Bt	Bf	Bs	Bf,Bs
Temperature	Tt	Th,Tl	Th,Tl	Th,Tl
Voltage	Vt	Vh,Vl	Vh,Vl	Vh,Vl

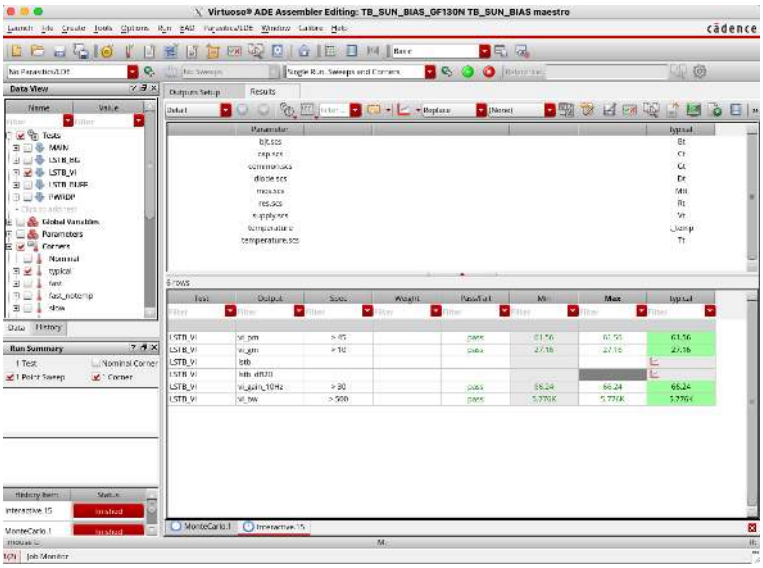


Figure 2: Cadence Virtuoso ADE Assembler, showing the corner definitions on the left and a pass/fail table of simulated specifications on the right. Screenshot of Cadence Virtuoso

I'm a fan of launching multiple simulations from the command line. I don't like GUI's. As such, I wrote cicsim, and that's what I use in the video and demo.

32.2.3 Sources

The SPICE language is a set of conventions for how to write the text files. In general, it's one line, one command (although, lines can be continued with a +).

I'm not going to go through an extensive tutorial in this document, and there are dialects with different SPICE programs. You'll find more info at ngspice

32.2.3.1 Independent current sources

Infinite output impedance, changing voltage does not change current

```
I<name> <from> <to> dc <number> ac <number>

I1 0 VDN dc In
I2 VDP 0 dc Ip
```

32.2.3.2 Independent voltage source

Zero output impedance, changing current does not change voltage

```
V<name> <+> <-> dc <number> ac <number>

V2 VSS 0 dc 0
V1 VDD 0 dc 1.5
```

32.2.4 Passives

Resistors

```
R<name> <node 1> <node 2> <value>

R1 N1 N2 10k
R2 N2 N3 1Meg
R3 N3 N4 1G
R4 N4 N5 1T
```

Capacitors

```
C<name> <node 1> <node 2> <value>

C1 N1 N2 1a
C2 N1 N2 1f
C4 N1 N2 1p
C3 N1 N2 1n
C5 N1 N2 1u
```

32.2.5 Transistor Models

Needs a model file describing the transistor model

BSIM (Berkeley Short-channel IGFET Model) http://bsim.berkeley.edu/models/bsim4/

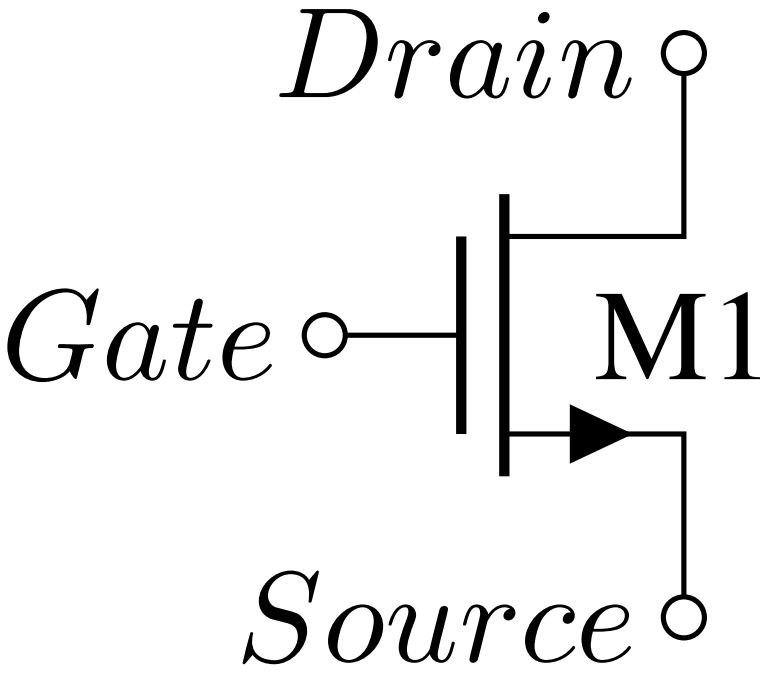


Figure 3: Circuit symbol for an NMOS transistor with its gate, drain and source terminals

284 parameters in BSIM 4.5

```
.MODEL N1 NMOS LEVEL=14 VERSION=4.5.0 BINUNIT=1
PARAMCHK=1 MOBMOD=0 CAPMOD=2 IGCMOD=1 IGBMOD=1
GEOMOD=1  DIOMOD=1 RDSMOD=0 RBODYMOD=0 RGATEMOD=3
PERMOD=1 ACNQSMOD=0 TRNQSMOD=0 TEMPMOD=0  TNOM=27
TOXE=1.8E-009 TOXP=10E-010 TOXM=1.8E-009  DTOX=8E-10
EPSROX=3.9 WINT=5E-009 LINT=1E-009 LL=0 WL=0 LLN=1
WLN=1  LW=0 WW=0 LWN=1 WWN=1  LWL=0 WWL=0 XPART=0
TOXREF=1.4E-009  SAREF=5E-6 SBREF=5E-6 WLOD=2E-6
KU0=-4E-6  KVSAT=0.2 KVTH0=-2E-8 TKU0=0.0 LLODKU0=1.1
WLODKU0=1.1 LLODVTH=1.0 WLODVTH=1.0 LKU0=1E-6
WKU0=1E-6 PKU0=0.0 LKVTH0=1.1E-6 WKVTH0=1.1E-6
PKVTH0=0.0 STK2=0.0 LODK2=1.0 STETA0=0.0  LODETA0=1.0
LAMBDA=4E-10  VSAT=1.1E 005 VTL=2.0E5 XN=6.0 LC=5E-9
RNOIA=0.577 RNOIB=0.37 LINTNOI=1E-009  WPEMOD=0
WEB=0.0 WEC=0.0 KVTH0WE=1.0  K2WE=1.0 KU0WE=1.0 SCREF=5.0E-6
TVOFF=0.0 TVFBSDOFF=0.0  VTH0=0.25  K1=0.35 K2=0.05
K3=0  K3B=0 W0=2.5E-006 DVT0=1.8 DVT1=0.52  DVT2=-0.032
DVT0W=0 DVT1W=0 DVT2W=0  DSUB=2 MINV=0.05 VOFFL=0
DVTP0=1E-007  DVTP1=0.05 LPE0=5.75E-008 LPEB=2.3E-010
XJ=2E-008  NGATE=5E 020 NDEP=2.8E 018 NSD=1E 020 PHIN=0
CDSC=0.0002 CDSCB=0 CDSCD=0 CIT=0  VOFF=-0.15 NFACTOR=1.2
ETA0=0.05 ETAB=0  UC=-3E-011  VFB=-0.55 U0=0.032
UA=5.0E-011 UB=3.5E-018  A0=2 AGS=1E-020 A1=0 A2=1
B0=-1E-020 B1=0  KETA=0.04 DWG=0 DWB=0 PCLM=0.08
PDIBLC1=0.028 PDIBLC2=0.022 PDIBLCB=-0.005 DROUT=0.45
PVAG=1E-020 DELTA=0.01 PSCBE1=8.14E 008 PSCBE2=5E-008
RSH=0 RDSW=0 RSW=0 RDW=0 FPROUT=0.2 PDITS=0.2 PDITSD=0.23
PDITSL=2.3E 006  RSH=0 RDSW=50 RSW=150
RDW=150  RDSWMIN=0 RDWMIN=0 RSWMIN=0 PRWG=0  PRWB=6.8E-011
WR=1 ALPHA0=0.074 ALPHA1=0.005  BETA0=30 AGIDL=0.0002
BGIDL=2.1E 009 CGIDL=0.0002 EGIDL=0.8  AIGBACC=0.012
BIGBACC=0.0028 CIGBACC=0.002  NIGBACC=1 AIGBINV=0.014
BIGBINV=0.004 CIGBINV=0.004  EIGBINV=1.1 NIGBINV=3 AIGC=0.012
BIGC=0.0028  CIGC=0.002 AIGSD=0.012 BIGSD=0.0028 CIGSD=0.002  NIGC=1
POXEDGE=1 PIGCD=1 NTOX=1  VFBSDOFF=0.0  XRCRG1=12 XRCRG2=5
CGSO=6.238E-010 CGDO=6.238E-010 CGBO=2.56E-011 CGDL=2.495E-10
CGSL=2.495E-10 CKAPPAS=0.03 CKAPPAD=0.03 ACDE=1  MOIN=15
NOFF=0.9 VOFFCV=0.02  KT1=-0.37 KT1L=0.0 KT2=-0.042 UTE=-1.5
UA1=1E-009 UB1=-3.5E-019 UC1=0 PRT=0 AT=53000  FNOIMOD=1
TNOIMOD=0  JSS=0.0001 JSWS=1E-011 JSWGS=1E-010 NJS=1
IJTHSFWD=0.01 IJTHSREV=0.001 BVS=10 XJBVS=1  JSD=0.0001
JSWD=1E-011 JSWGD=1E-010 NJD=1  IJTHDFWD=0.01 IJTHDREV=0.001
BVD=10 XJBVD=1  PBS=1 CJS=0.0005 MJS=0.5 PBSWS=1  CJSWS=5E-010
MJSWS=0.33 PBSWGS=1 CJSWGS=3E-010  MJSWGS=0.33 PBD=1 CJD=0.0005
MJD=0.5  PBSWD=1 CJSWD=5E-010 MJSWD=0.33 PBSWGD=1
CJSWGD=5E-010MJSWGD=0.33 TPB=0.005 TCJ=0.001 TPBSW=0.005
TCJSW=0.001 TPBSWG=0.005 TCJSWG=0.001  XTIS=3 XTID=3  DMCG=0E-006
DMCI=0E-006 DMDG=0E-006 DMCGT=0E-007  DWJ=0.0E-008 XGW=0E-007
XGL=0E-008  RSHG=0.4 GBMIN=1E-010 RBPB=5 RBPD=15  RBPS=15 RBDB=15
RBSB=15 NGCON=1 JTSS=1E-4 JTSD=1E-4 JTSSWS=1E-10 JTSSWD=1E-10
JTSSWGS=1E-7 JTSSWGD=1E-7  NJTS=20.0 NJTSSW=20 NJTSSWG=6
VTSS=10 VTSD=10 VTSSWS=10 VTSSWD=10  VTSSWGS=2 VTSSWGD=2
XTSS=0.02 XTSD=0.02 XTSSWS=0.02 XTSSWD=0.02 XTSSWGS=0.02
XTSSWGD=0.02
```

32.2.6 Transistors

```
M<name> <drain> <gate> <source> <bulk> <modelname> [parameters]
```

```
M1 VDN VDN VSS VSS nmos W=0.6u L=0.15u
```

```
M2 VDP VDP VDD VDD pmos W=0.6u L=0.15u
```

32.2.7 Foundries

Each foundry has their own SPICE models bacause the transistor parameters depend on the exact physics of the technology!

https://skywater-pdk.readthedocs.io/en/main/

32.3 Find right transistor sizes

Assume active (

$$V_{ds} > V_{eff}$$

in strong inversion, or

$$V_{ds} > 3V_T$$

in weak inversion). For diode connected transistors, that is always true.

Weak inversion:

$$I_D = I_{D0}\frac{W}{L}e^{V_eff/nV_T}$$

,

$$V_{eff} \propto \ln I_D$$

Strong inversion:

$$I_D = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}V_{eff}^2$$

,

$$V_{eff} \propto \sqrt{I_D}$$

Operating region for a diode connected transistor only depends on the current

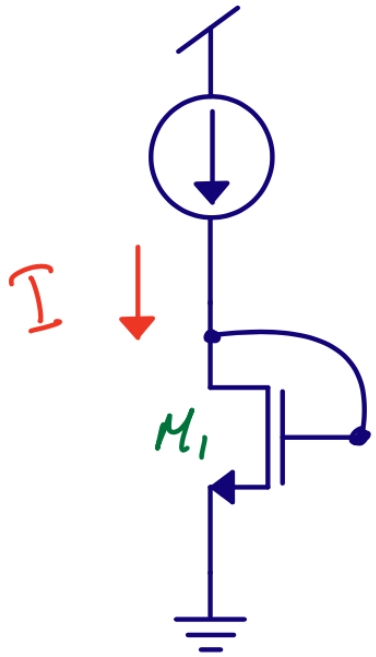


Figure 4: Diode connected NMOS biased by a current source, where the current alone sets the operating region

32.3.1 Use unit size transistors for analog design

$$W/L \approx \in [4, 6, 10]$$

, but should have space for two contacts

Use parallel transistors for larger W/L

Amplifiers

$$\Rightarrow L \approx 1.2 \times L_{min}$$

Current mirrors

$$\Rightarrow L \approx 4 \times L_{min}$$

Choose sizes that have been used by foundry for measurement to match SPICE model

32.3.2 What about gm/Id ?

Weak

$$\frac{g_m}{I_d} = \frac{1}{nV_T}$$

Strong

$$\frac{g_m}{I_d} = \frac{2}{V_{eff}}$$

32.3.3 Characterize the transistors

http://analogicus.com/cnr_atr_sky130nm/mos/CNRATR_NCH_2C1F2.html

32.4 More information

Ngspice Manual

Installing tools

32.5 Analog Design

1. **Define the problem, what are you trying to solve?**
2. **Find a circuit that can solve the problem (papers, books)**
3. **Find right transistor sizes. What transistors should be weak inversion, strong inversion, or don't care?**
4. **Check operating region of transistors (.op)**
5. **Check key parameters (.dc, .ac, .tran)**
6. **Check function. Exercise all inputs. Check all control signals**
7. Check key parameters in all corners. Check mismatch (Monte-Carlo simulation)
8. Do layout, and check it's error free. Run design rule checks (DRC). Check layout versus schematic (LVS)
9. Extract parasitics from layout. Resistance, capacitance, and inductance if necessary.
10. On extracted parasitic netlist, check key parameters in all corners and mismatch (if possible).
11. If everything works, then you're done.

On failure, go back

32.6 Demo

https://github.com/analogicus/jnw_spice_sky130A/tree/main

32.7 Summary

The one-page version of this chapter:

- SPICE has run the same way for fifty-odd years: a netlist in, operating points and waveforms out
- The quartet to master: op, dc, ac and tran - everything else is decoration on those four
- ngspice speaks the dialect this course uses, and the transistor models come from the PDK, not from the simulator

32.8 Would you like to know more?

Nagel's 1975 thesis, where SPICE comes from, and which still reads well

The ngspice manual, the reference for everything this chapter does

Mixed Signal Simulation in NGSPICE 33

33.1 Mixed Signal Simulation in ngspice . . 587
33.2 Digital simulation . 587
33.3 Transient analog simulation 588
33.4 Demo 590
33.5 The circuit 590
33.6 The digital code . . 590
33.7 Compile RTL 591
33.8 Import object into SPICE file 591
33.9 Import in testbench 592
33.10 Override default digital output voltage . 592
33.11 Running 592
33.12 Summary 592
33.13 Would you like to know more? 593

33.1 Mixed Signal Simulation in ngspice

Keywords: Mixed Signal, ngspice, Verilog, RTL, d_cosim, Testbench

33.2 Digital simulation

- The order of execution of events at the same time-step does not matter
- The system is causal. Changes in the future do not affect signals in the past or the now

There are both commercial and open source tools for digital simulation. If you've never used a digital simulator, then I'd recommend you start with iverilog. I've made some examples at dicex.

Commercial

- Cadence Excelium
- Siemens Questa
- Synopsys VCS

Open Source

- iverilog/vpp
- Verilator
- SystemDotNet

Below is an example of a counter in SystemVerilog. The code can be found at counter_sv.

In the always_ff sections we code what will become our registers; combinatorial logic would go in an always_comb section.

```
module dig(
           input wire         clk,
           input wire         reset,
           output logic [4:0] b
           );

   logic                      rst = 0;

   always_ff @(posedge clk) begin
      if(reset)
        rst <= 1;
      else
        rst <= 0;
```

```
    end

    always_ff @(posedge clk) begin
       if(rst)
         b <= 0;
       else
         b <= b + 1;
    end // dig

endmodule
```

33.3 Transient analog simulation

Analog simulation is different. There is no quantized time step. How fast "things" happen in the circuit is entirely determined by the time constants, change in voltage, and change in current in the system.

It is possible to have a fixed time-step in analog simulation, for example, we say that nothing is faster than 1 fs, so we pick that as our time step. If we wanted to simulate 1 s, however, that's at least 1e15 events, and with 1 event per microsecond on a computer it's still a simulation time of 31 years. Not a viable solution for all analog circuits.

Analog circuits are also non-linear, properties of resistors, capacitors, inductors, diodes may depend on the voltage or current across, or in, the device. Solving for all the non-linear differential equations is tricky.

An analog simulation engine must parse spice netlist, and setup partial/ordinary differential equations for node matrix

The nodal matrix could look like the matrix below, i are the currents, v the voltages, and G the conductances between nodes.

$$\begin{pmatrix} G_{11} & G_{12} & \cdots & G_{1N} \\ G_{21} & G_{22} & \cdots & G_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ G_{N1} & G_{N2} & \cdots & G_{NN} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_N \end{pmatrix} = \begin{pmatrix} i_1 \\ i_2 \\ \vdots \\ i_N \end{pmatrix}$$

The simulator, and devices model the non-linear current/voltage behavior between all nodes

as such, the G's may be non-linear functions, and include the v's and i's.

Transient analysis use numerical methods to compute time evolution

The time step is adjusted automatically, often by proprietary algorithms, to trade accuracy and simulation speed.

The numerical methods can be forward/backward Euler, or the others listed below.

- Euler
- Runge-Kutta
- Crank-Nicolson
- Gear [1]

If you wish to learn more, I would recommend starting with the original paper on analog transient analysis.

SPICE (Simulation Program with Integrated Circuit Emphasis) published in 1973 by Nagel and Pederson

The original paper has spawned a multitude of commercial, free and open source simulators, some are listed below.

If you have money, then buy Cadence Spectre. If you have no money, then start with ngspice.

Commercial - Cadence Spectre - Siemens Eldo - Synopsys HSPICE

Free - Aimspice - Analog Devices LTspice - xyce

Open Source - ngspice

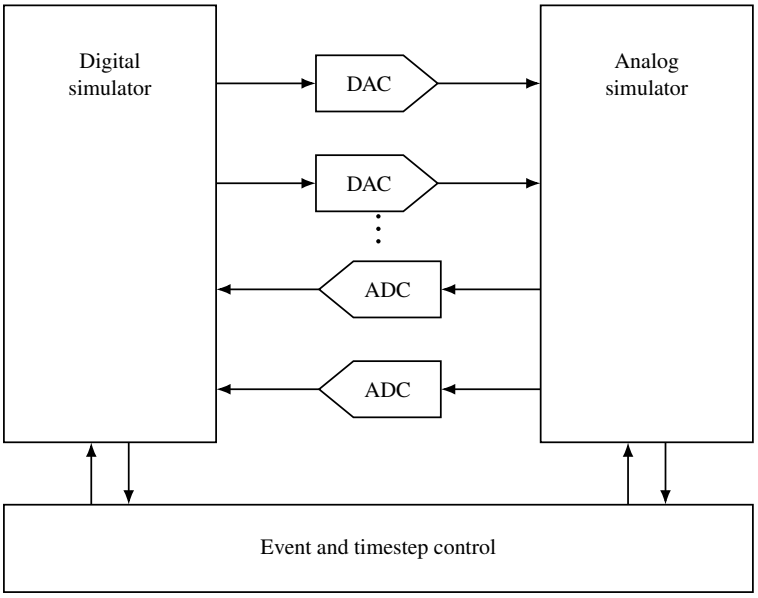


Figure 1: Mixed signal simulator, where a digital and an analog simulator exchange values through DAC and ADC connect modules under a shared event and timestep control

33.4 Demo

Tutorial at http://analogicus.com/jnw_sv_sky130a/

Repository at https://github.com/wulffern/jnw_sv_sky130a

Assumes knowledge of Tutorial

33.5 The circuit

In `design/JNW_SV_SKY130A/JNWSW_CM.sch` you'll find a current mirror, and a 5-bit current DAC.

What we want from the digital is to control the binary value of the current DAC.

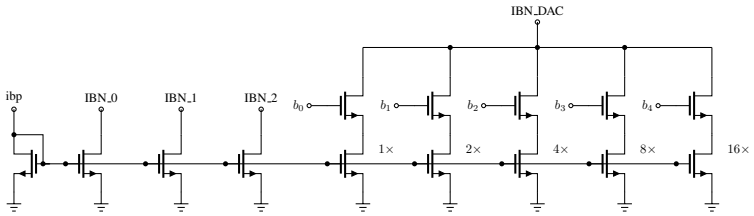


Figure 2: The JNWSW_CM schematic - a current mirror from ibp plus a 5-bit binary weighted current DAC controlled by b[4:0]

33.6 The digital code

The digital code is shown below. The `clk` controls the stepping, while the `reset` sets the output `b=0`. When `reset` is off, then the `b` increments.

```
module dig(
           input wire         clk,
           input wire         reset,
           output logic [4:0] b
           );

   logic                      rst = 0;

   always_ff @(posedge clk) begin
      if(reset)
        rst <= 1;
      else
        rst <= 0;
   end

   always_ff @(posedge clk) begin
      if(rst)
        b <= 0;
      else
        b <= b + 1;
   end // dig
endmodule
```

33.7 Compile RTL

The first thing we need to do is to translate the verilog into a compiled object that can be used in ngspice.

```
cd sim/JNWSW_CM
ngspice vlnggen ../../rtl/dig.v
```

33.8 Import object into SPICE file

I'm lazy. So I don't want to do the same thing multiple times. As such, I've written a small script to help me instanciate the verilog

```
perl ../../tech/script/gensvinst ../../rtl/dig.v dig
```

The script generates an `svinst.spi` file. The first section imports the digital compiled library

```
adut [clk
+ reset
+ ]
+ [b.4
+ b.3
+ b.2
+ b.1
+ b.0
+ ] null dut
.model dut d_cosim
+ simulation="../dig.so" delay=10p
```

Turns out that ngspice needs the digital inputs and outputs to be connected to something to calculate them (I think), so connect some resistors

```
* Inputs
Rsvi0 clk 0 1G
Rsvi1 reset 0 1G

* Outputs
Rsvi2 b.4 0 1G
Rsvi3 b.3 0 1G
Rsvi4 b.2 0 1G
Rsvi5 b.1 0 1G
Rsvi6 b.0 0 1G
```

For the busses I find it easier to read the value as a real, so translate the buses from digital `b[4:0]` to a real value `dec_b`

```
E_STATE_b dec_b 0 value={( 0
+ + 16*v(b.4)/AVDD
+ + 8*v(b.3)/AVDD
+ + 4*v(b.2)/AVDD
+ + 2*v(b.1)/AVDD
+ + 1*v(b.0)/AVDD
+)/1000}
.save v(dec_b)
```

33.9 Import in testbench

An example testbench can be seen below (`sim/JNWSW_CM/tran.spi`)

```
...

.include ../xdut.spi
.include ../svinst.spi

* Translate names
VB0 b.0 b<0> dc 0
VB1 b.1 b<1> dc 0
VB2 b.2 b<2> dc 0
VB3 b.3 b<3> dc 0
VB4 b.4 b<4> dc 0

...
```

33.10 Override default digital output voltage

We can override the output dac from digital to analog to ensure that the digital signals have the right levels

```
*- Override the default digital output bridge.
pre_set auto_bridge_d_out =
     + ( ".model auto_dac dac_bridge(out_low = 0.0 out_high = 1.8)"
     +   "auto_bridge%d [ %s ] [ %s ] auto_dac" )
```

33.11 Running

You can run the whole thing with

```
cd sim/JNWSW_CM/
make typical
```

33.12 Summary

The one-page version of this chapter:

- SystemVerilog describes hardware as events: always_ff becomes registers, combinational logic goes in always_comb
- Event-driven simulation only computes when something changes, which is why digital simulation is fast
- Mixed-signal co-simulation bridges the two worlds: the svinst flow connects SPICE and the digital testbench

33.13 Would you like to know more?

The IEEE 1800 standard is the authority; Sutherland's tutorials are the readable way in

The Verilator and iverilog manuals, for the two simulators used here

[1] C. Gear, "Simultaneous numerical solution of differential-algebraic equations," *IEEE Transactions on Circuit Theory*, vol. 18, no. 1, pp. 89–95, 1971, doi: 10.1109/TCT.1971.1083221.

Analog Design 34

Keywords: Checklist, Specification, Design, Tapeout, Schematic Rules, Layout Rules

34.1 Checklist 595
34.1.1 Specification 595
34.1.2 Design 596
34.1.3 Tapeout 596
34.2 Schematic rules . . 596
34.2.1 Only uppercase names allowed . . . 597
34.3 Layout rules 598
34.3.1 Limit the amount of transistor Width's and Length's that you use 598
34.4 Would you like to know more? 599

34.1 Checklist

There are roughly 3 phases of analog design.

- Specification
- Design
- Tapeout

The specification phase is where you think deeply through the design.

Can the design meet the key parameters you need? How will I verify the circuit? Do I know how to make the circuit?

These questions and more, are so common that most companies will have checklists that we use when we review the specification, design, and tapeout.

These checklists are closely guarded secrets, as the content contain significant amount of knowledge accumulated over numerous blunders, mistakes, failure to imagine, and physics teaching us a lesson.

The design phase is where we make the schematic, and simulate the schematic. We explore circuit architectures, fix problem corners, check our design over temperature, voltage, process corners (slow transistors, fast transistors, mismatch)

The tapeout phase is where we translate the schematic into layout, check the design rules (DRC), do layout versus schematic (LVS) and extract circuit parasitics to check layout parasitic effects (LPE). And, of course, simulate most things again.

I've made a checklist below for the most common questions that you need to ask yourself.

34.1.1 Specification

Item	Description	Yes	Action
Functional description	Have you described what the IP shall do?		
Key parameters	Have you updated your key parameters in the README		
Architecture	Have you described the circuit architecture? How should it work?		
Realism	Do you know how to do what you plan to do?		
Verification plan	Have you described exactly what you need to check? For example, stability of OTAs, current consumption, key parameters		
Specification	Have you added a specification for all parameters you intend to check. For example, phase margin should always be larger than 45 degrees.		

34.1.2 Design

Item	Description	Yes	Action
git	Have you committed the schematics to the repository?		
git push	Are the schematics pushed to github? Are you sure?		
git tag	Is the current version tagged		
Implementation	Are all schematics described with their own markdown file?		
Verification plan	Are all items on the verification plan completed? If not, have you described why it's no longer relevant?		
Electrical parameters	Are the electrical parameters updated with simulated results?		
Spec violations	Have you explained why the specification violations are not an issue?		
Simulation	Are the required corners run (typical, slow, fast, mc)		

34.1.3 Tapeout

Item	Description	Yes	Action
git	Are all schematics and layout committed? Are you sure?		
git push	Have you pushed to github?		
git tag	Is the latest version tagged?		
LVS	Is the LVS on github passing (green)?		
DRC	Is the DRC on github passing (green)?		
LPE	Is the LPE on github passing (green)?		
Simulation	Are the required corners re-run with layout parasitics (typical, slow, fast, mc)		

34.2 Schematic rules

Rules are nice. They reduce the cognitive load since a decision already has been made. You may disagree with the rule, but in analog design it's not that important what the "rule" is, but sometimes more important that the rule is followed.

Naming rules are one example. We can have a philosophical discussion till the end of time of whether it's best with uppercase

or lower case. CamelCase, however, is wrong in schematics and SPICE, since SPICE is case-insensitive, so "vref" is the same as "Vref".

Below are the rules I like. You may disagree, but if you're my student, then you don't have a choice. Follow them, or the grade may suffer.

34.2.1 Only uppercase names allowed

Do: AVDD, **Don't:** aVdd

Although editors can handle mix of upper case and lower case, SPICE, cannot. SPICE is case insensitive. That means AVDD == aVdd in SPICE, but AVDD != aVdd in editors. A such mixing case is a bad idea, so one must be picked, and uppercase the chosen one. Why? Why not?

34.2.1.1 Use same net throughout hierarchy

Do: VDD -> VDD -> VDD, **Don't:** VDD -> LOCALVDD -> CELLVDD

Debugging becomes a lot simpler if a net keeps it's name throughout the schematic hierarchy. Especially bad are cases where a net name is reused on multiple levels of hierarchy. Imagine the following scenario

```
     | (sub block)          |
     |                      |
VDD -|-LVDD---/ ---- VDD    |
     |                      |
```

Here the net name VDD is used on the top level, while LVDD is used in the sub-block. In the sub-block there is a power switch between LVDD and VDD. In this case VDD != VDD on top level, which can lead to long debugging times. There are, however, a few exceptions. For example, using VDD on standard cells (inverters, ANDs etc) is ok, even though the power supply is not called VDD

34.2.1.2 Spend time on making schematics pretty

It matters how schematics look. Think of it like this. In 10 years, you will be asked to port, re-simulate, fix a bug, on your design. If you have spent some time adding comments, making things look pretty, etc, then the job will be much, much easier. "A pretty schematic is a love letter to your future self".

PS: This applies to documentation, code, and everything you make.

34.2.1.3 All digital nets must be "active high" naming

Example: PWRUP_3V3, PWRUP_N_3V3

The PWRUP_3V3 should be understood as "When PWRUP_3V3 is high, then the block is powered up"

The PWRUP_N_3V3 should be understood as "When PWRUP_N_3V3 is high then the block is powered down"

34.2.1.4 Bias currents shall be named IB<device sending current (P|N)>

Do: IBP_1U, **Don't:** IBIAS_1U

The "P" post-fix tells us the current comes from a PMOS, so we can put it into a NMOS diode connected transistor. On the IBIAS we have no idea which way the current flows.

34.3 Layout rules

34.3.1 Limit the amount of transistor Width's and Length's that you use

Do: W = 1.0 um, L = 180 nm, Use multiplier for other sizes

Don't: W1 = 1 um, W2 = 1.1 um, W3 = 1.2 um, W4 = 2 um, W5 = 2.1 um

You want the layout to be relatively regular. A bunch of different Ls and Ws is a pain when you do layout

34.3.1.1 Use pre-defined transistors for regular layout

Do: Use JNW_ATR_SKY130A and JNW_TR_SKY130A

34.3.1.2 Always use two fingers for analog transistors

That way, you don't have to worry about current direction in the layout.

34.3.1.3 Always run gates in the same direction

Mobility of transistors (especially PMOS) is affected by strain, so if you rotate a transistor it will not have the same current, and change in current as a function of stress.

I've seen ICs have to be taped out again due to rotated transistors.

34.3.1.4 Always have dummy poly gates

For large lengths (> 500 nm) the lithography effects are not that severe, but the etching of the gate material will be asymmetric if there are no poly dummies. Make sure the poly dummy is exactly the same spacing on both sides.

For small lengths (< 200 nm) the lithography effects start to matter. The light used in most litho is 193 nm. 193 nm is used all the way down to about 7 nm.

Due to diffraction effects, it's common to have extremely regular poly spacing, an exact distance such that the interference from neighboring poly's align perfectly to the next poly.

34.3.1.5 Always place transistors away from well edge

Close to the N-well edge the donor consentration will be higher. Ion implantation is used for the wells, and the ions will scatter of the oxide wall, and increase the doping concentration close to the edge. As such, the transistor threshold voltage will increase close to a well edge.

Keep transistors about 3 um away from the N-well edge if threshold voltage is important.

34.4 Would you like to know more?

Johns and Martin, the textbook path through analog design [1]

Razavi, a second voice on the same material [2]

The JSSC and ISSCC archives, for how it is actually being done now

[1] D. Johns and K. Martin, *Analog integrated circuit design*. John Wiley & Sons, Inc., 1997.

[2] B. Razavi, *Design of analog CMOS integrated circuits*. McGraw-Hill, 2001.

35 The cic tools

This chapter was written by Claude, Anthropic's AI, from an outline and direction by Carsten Wulff, who reviewed and edited the result. The commit history of the book's repository records precisely who wrote what.

Analog design that cannot be reproduced from a shell command is not done. That is the thesis of this chapter, and the reason the tools in it exist. Every schematic in this course simulates from a script, every testbench sweeps its corners from a one-line command, and one of the ADCs in the converter chapter compiles its own layout. None of that needs a licence server.

The tools share a prefix - cic, for Custom IC Creator - and a philosophy: plain text in, plain text out, and git in between. Each one is small. Together they are a design flow.

35.1 cicsim - simulations you can rerun 601
35.2 ciccreator - the layout compiler 602
35.3 cicpy and cicconf - the glue 602
35.4 cicwave - looking at the waves 603
35.5 cictikz - the figures of this book 603
35.6 Summary 603
35.7 Would you like to know more? 603

35.1 cicsim - simulations you can rerun

`cicsim` is the workhorse: a script package that controls ngspice. You met it in the tutorial when `cicsim simcell` built the simulation directory for the current mirror. What it buys you over running ngspice by hand:

`cicsim run` takes a testbench and a corner specification - typical, slow, fast, high and low temperature, supply corners - and runs the cross product, in parallel if you ask (`--threads`), with a progress bar and a timeout to kill the simulation that wedged. Add `--count N` and each corner runs N times with mismatch - Monte Carlo from the same command line.

The measurements come back as a results table, one row per corner, which is the artifact that matters: the OTA chapter's advice to verify gain, noise, slew and start-up *over PVT* is one `cicsim run` per testbench, and the table is the evidence.

The yaml files that define a simulation live next to the schematic and go into git. Six months later, `cicsim run` again and you get the same table - or a diff that tells you exactly what the PDK update broke.

- `cicsim run` : corners, in parallel, with a results table
- `--count N` : Monte Carlo from the same command
- yaml + git : the simulation is reproducible, or it is not done

35.2 ciccreator - the layout compiler

`ciccreator` is the tool behind the compiled SAR ADC in the converter chapter. The idea dates to 2013: describe the circuit as a SPICE netlist, describe the layout intent as a JSON object definition, add a technology rule file, and let a compiler place the polygons. The prototype was 16 thousand lines of Perl; the current tool is a C++ rewrite of the same input language, fast enough to recompile an ADC while you watch.

The point is portability. The netlist and the object file do not mention a technology; the rule file does. Porting the ADC to a new process - and it has been ported to 22, 28, 55, 65, 130 and 180 nm - means writing a new rule file, not redrawing a layout. That is why the comparison table in the converter chapter has a "compiled" line with a competitive figure of merit, and why the same ADC exists as an open source SKY130 implementation you can read.

- SPICE netlist + JSON object definition + technology rules = layout
- Port to a new process: new rule file, not new polygons
- The compiled SAR ADC of the converter chapter is the proof

35.3 cicpy and cicconf - the glue

`cicpy` translates. It takes ciccreator output and transpiles it to whatever the rest of your flow speaks: SKILL for Cadence, SPICE, Verilog, Xschem schematics, Magic layout, or SVG when you just want to look at a cell. It also carries two small workhorses - `sch2mag` and `spi2mag` - that netlist a schematic and place-and-route it to Magic, which is how the standard-cell-like blocks in the aicex IPs appear.

`cicconf` manages the projects themselves. An aicex IP is a git repository, and a chip is many of them; `cicconf` keeps the dependency list in one `config.yaml` and clones, updates and templates the collection. When the tutorial told you to run `cicconf clone`, this is what stitched your project together.

- cicpy: transpile the compiled IC to SKILL, SPICE, Verilog, Xschem, Magic or SVG
- cicconf: one config.yaml for a project of many git repositories

35.4 cicwave - looking at the waves

`cicwave` is the waveform viewer: it reads ngspice raw files and draws them with a PyQtGraph/Qt6 backend fast enough for long transients. It is the third viewer of its line - the first was written during a summer internship in 2001 - and it exists for the same reason as the rest of the family: the open source flow deserves a viewer that starts instantly, works on every OS, and is scriptable.

35.5 cictikz - the figures of this book

The newest member drew the book you are reading. `cictikz` packages the TikZ symbol library behind every schematic figure in these chapters, gives an AI assistant a render-and-look loop so figures can be drawn and reviewed programmatically, and converts between the book's TikZ dialect and Xschem schematics - so a figure can start life as a real, simulated schematic and end as a book drawing, or the other way around.

- One symbol library for every schematic in the book
- TikZ to Xschem and back: figures that are also schematics

35.6 Summary

The one-page version of this chapter:

- If it does not rerun from a shell command, it is not done
- cicsim: corners and Monte Carlo as one command, results as a table, all of it in git
- ciccreator: netlist + object definition + rules compile to layout; porting is a rule file
- cicpy transpiles to the flow you have; cicconf holds multi-repo projects together
- cicwave views the waves; cictikz draws the book
- None of it needs a licence server

35.7 Would you like to know more?

Every tool lives on GitHub with its own documentation: cicsim, ciccreator (docs), cicpy, cicconf, cicwave (docs) and cictikz (docs). The IPs built with them are collected in aicex.

- github.com/wulffern/aicex - the IPs built with these tools

CMOS Logic 36

36.1 CMOS Logic

Keywords: CMOS Logic, Inverter, NAND, Static Logic, SR-Latch, D-Latch, Flip-Flop, Tristate, AOI

36.2 Analog transistor to digital transistor

NMOS current (W = 0.4u L=0.15u) as a function of

$$V_{GS}$$

and

$$V_{DS}$$

__dicex/lectures/l13/mos.py

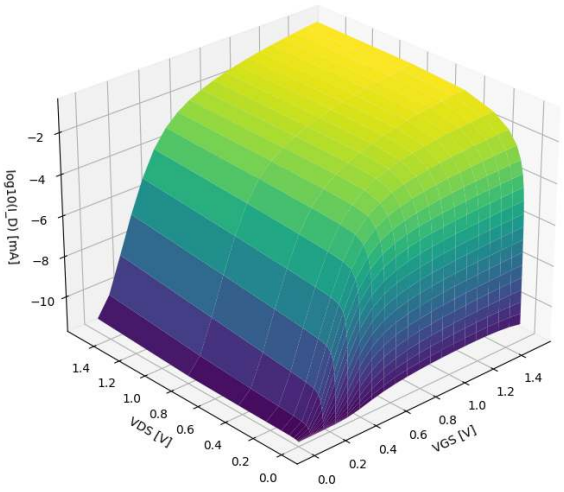


Figure 1: NMOS drain current on a log10 scale as a surface over gate-source and drain-source voltage, for W = 0.4u and L = 0.15u

36.1 CMOS Logic . . . 605
36.2 Analog transistor to digital transistor 605
36.3 CMOS static logic assumptions 607
36.4 Don't break rules unless you know exactly why it will be OK 609
36.5 Logic cells 609
36.5.1 CMOS static logic is inverting 609
36.5.2 Rules for inverting logic 612
36.6 SR-Latch 614
36.7 D-Latch (16 transistors) 615
36.8 Other logic cells . 616
36.9 AOI22: and or invert 617
36.10 Tristate inverter . 618
36.11 Mux 619
36.12 There are other types of logic . . . 620
36.13 Speed 621
36.14 Flip-flops and speed 622
36.15 Timing analysis . 625
36.16 Timing analysis tools 626
36.16.1 What is timing analysis 626
36.17 Every gate must be simulated to provide behavior over input transition and load capacitance 628
36.18 All analog blocks must have associated liberty file to describe behavior and timing paths *If you integrate analog into digital top flow* 628
36.19 Gate Delay 628
36.19.1 Delay definitions . 628
36.20 Delay estimation . 629
36.20.1 Inverter with in-

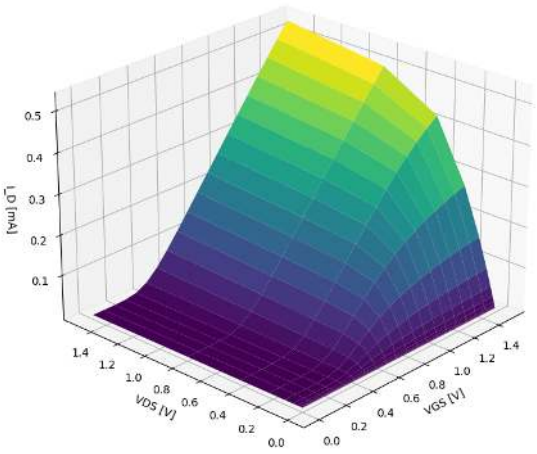


Figure 2: The same NMOS drain current on a linear scale in mA over gate-source and drain-source voltage

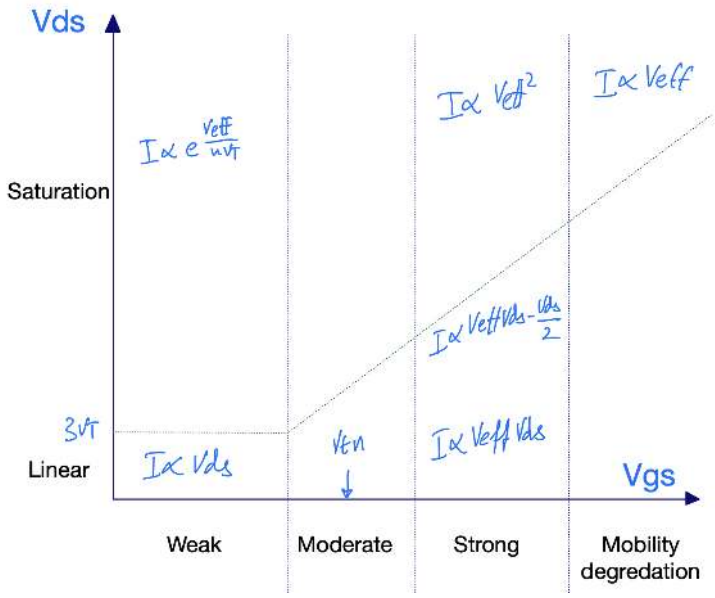


Figure 3: The analog view of the NMOS: current equations for the linear and saturation regions across weak, moderate and strong inversion and mobility degradation

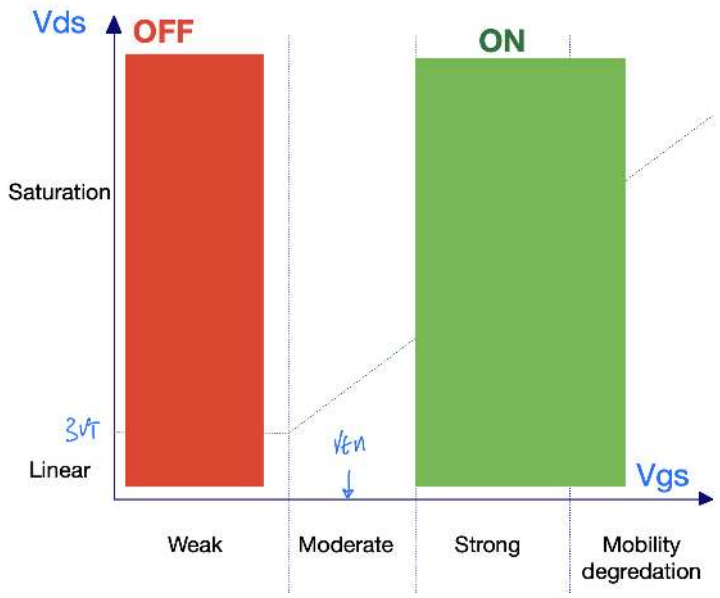


Figure 4: The digital view of the same plane, where weak inversion is treated as OFF and strong inversion as ON, and everything else is ignored

Gate	NMOS	PMOS
VDD	ON	OFF
VDD -> VSS	X	X
VSS -> VDD	X	X
VSS	OFF	ON

Gate	NMOS	PMOS
1	ON	OFF
1 -> 0	X	X
0 -> 1	X	X
0	OFF	ON

36.3 CMOS static logic assumptions

NMOS source is connected to low potential

$$V_{GS} > V_{TH}$$

when

$$V_G = V_{DD}$$

PMOS source is connected to high potential

$$V_{GS} < V_{TH}$$

when

$$V_G = 0$$

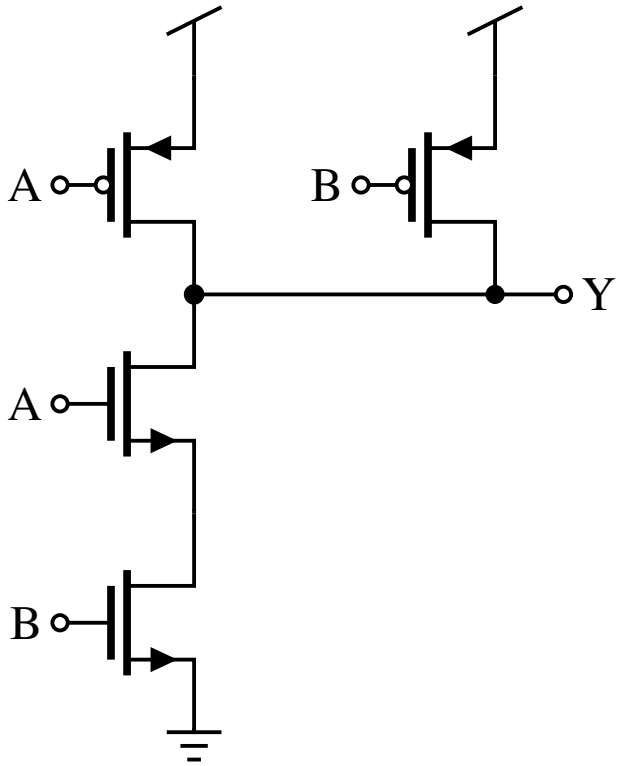


Figure 5: Transistor schematic of a two input NAND, with PMOS A and B in parallel to the supply and NMOS A and B in series to ground

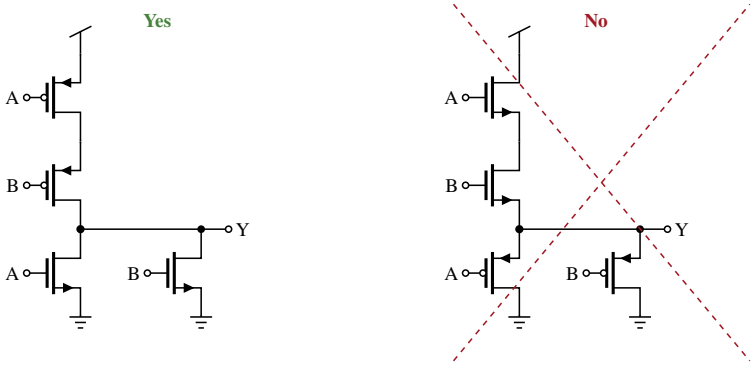


Figure 6: Two ways to wire a NOR - the accepted one with PMOS pulling up and NMOS pulling down, and the rejected one with the device types swapped so the sources sit at the wrong rail

36.4 Don't break rules unless you know exactly why it will be OK

36.5 Logic cells

A	B	$\overline{AB}$	$\overline{A+B}$	AB	$A+B$
0	0	1	1	0	0
0	1	1	0	0	1
1	0	1	0	0	1
1	1	0	0	1	1

$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$	$\overline{A}\,\overline{B}$	$\overline{\overline{A}+\overline{B}}$	$\overline{\overline{A}\,\overline{B}}$
1	1	1	1	0	0
1	0	1	0	0	1
0	1	1	0	0	1
0	0	0	0	1	1

$\overline{AB} = \overline{A} + \overline{B}$ ← DM

$\overline{A+B} = \overline{A}\,\overline{B}$ ← DM

$AB = \overline{\overline{A} + \overline{B}}$

$A + B = \overline{\overline{A}\,\overline{B}}$

$\overline{AB} \neq \overline{A}\,\overline{B}$

$\overline{A} + \overline{B} \neq \overline{A+B}$

Figure 7: Truth tables for NAND, NOR, AND and OR with inverted inputs, illustrating the two De Morgan identities

36.5.1 CMOS static logic is inverting

A	Y
1	0
0	1

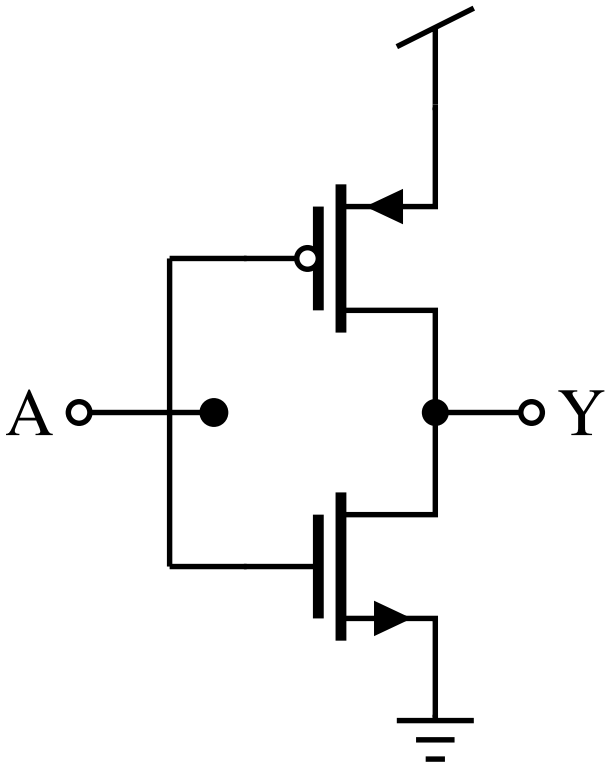


Figure 8: Transistor schematic of the CMOS inverter, one PMOS to the supply and one NMOS to ground sharing gate A and output Y

PD \ PU	OFF	ON
OFF	Z	1
ON	0	X

Figure 9: Output state as a function of the pull-up and pull-down networks - Z when both are off, 1 or 0 when one conducts, and X when both conduct

PD = Pull-down PU = Pull-up

```
logic => [0,1,Z,X];
```

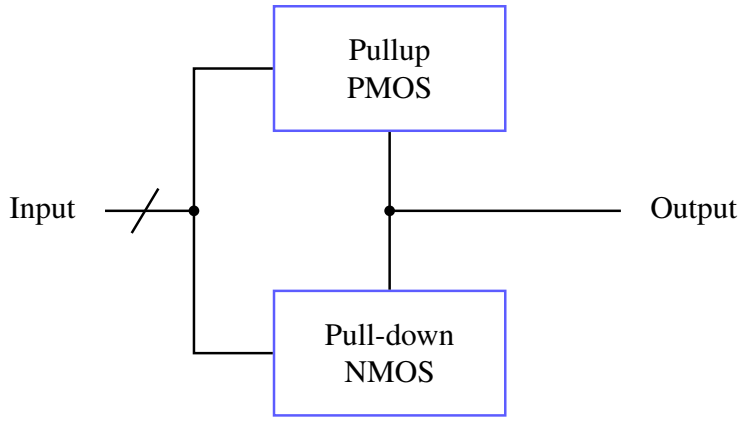


Figure 10: Block view of a static CMOS gate, where the inputs drive a PMOS pull-up network and an NMOS pull-down network that share the output node

Pull-up series

A	B	Y
0	0	1
0	1	Z
1	0	Z
1	1	Z

Pull-up parallel

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	Z

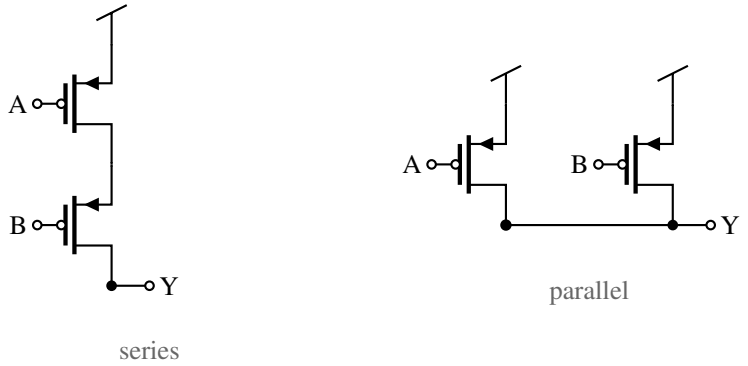


Figure 11: PMOS pull-up networks - two devices in series conduct only when both A and B are 0, two in parallel conduct unless both are 1

Pull-down series

A	B	Y
0	0	Z
0	1	Z
1	0	Z
1	1	0

Pull-down parallel

A	B	Y
0	0	Z
0	1	0
1	0	0
1	1	0

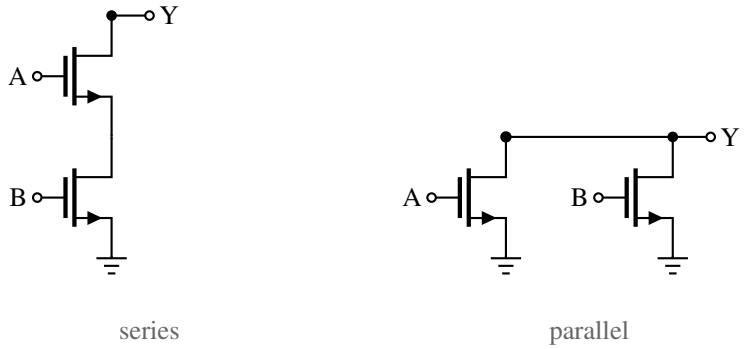


Figure 12: NMOS pull-down networks - two devices in series conduct only when both A and B are 1, two in parallel conduct when either is 1

36.5.2 Rules for inverting logic

Pull-up OR => PMOS in series => POS AND => PMOS in parallel => PAP

Pull-down OR => NMOS in parallel => NOP AND => NMOS in series => NAS

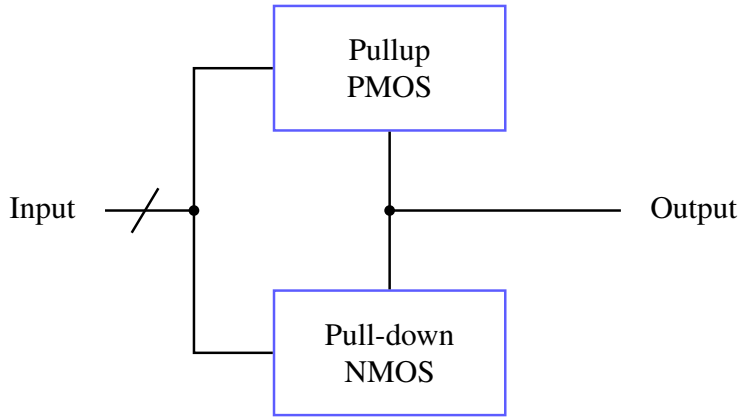


Figure 13: The same pull-up and pull-down blocks, read through the rules that OR maps to PMOS in series and NMOS in parallel, AND to PMOS in parallel and NMOS in series

$$Y = \overline{AB} = \text{NOT (A AND B)}$$

AND PU => PMOS in parallel PD => NMOS in series

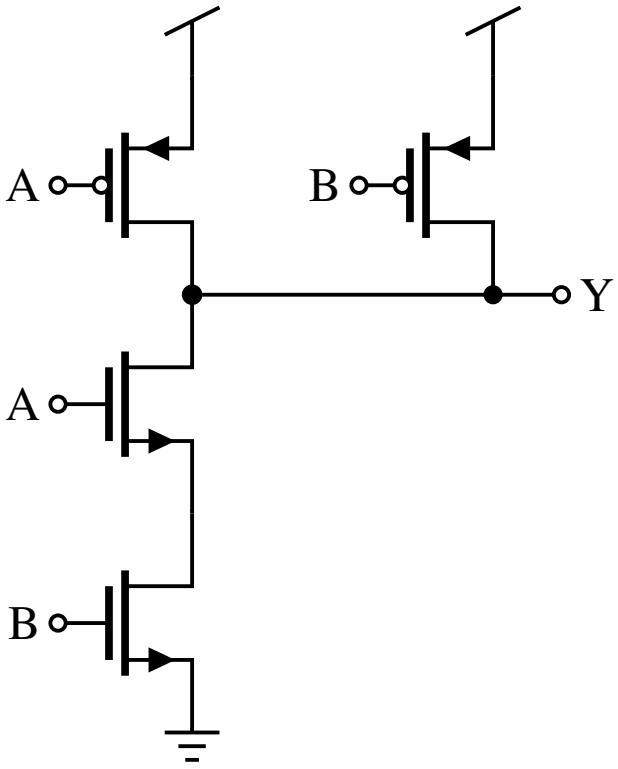


Figure 14: NAND built from the AND rule, with PMOS A and B in parallel for the pull-up and NMOS A and B in series for the pull-down

A	B	*NOT(A AND B)*
0	0	1
0	1	1
1	0	1
1	1	0

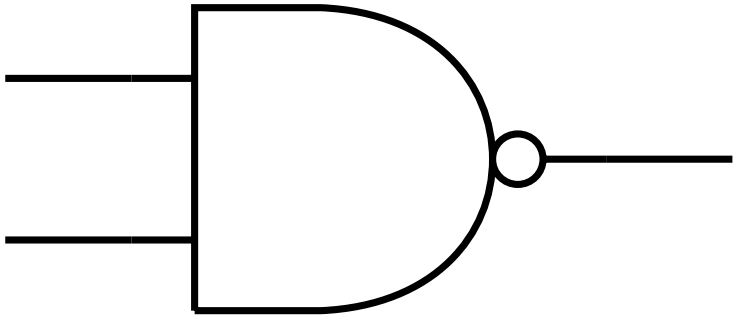


Figure 15: Schematic symbol for the two input NAND gate

$$Y = \overline{A + B} = \text{NOT (A OR B)}$$

OR PU => PMOS in series PD => NMOS in parallel

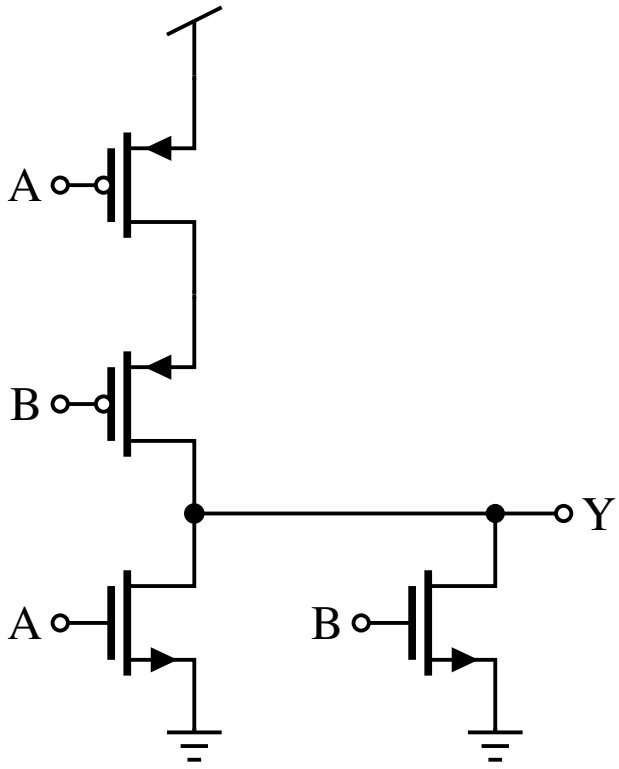


Figure 16: NOR built from the OR rule, with PMOS A and B in series for the pull-up and NMOS A and B in parallel for the pull-down

A	B	*NOT(A OR B)*
0	0	1
0	1	0
1	0	0
1	1	0

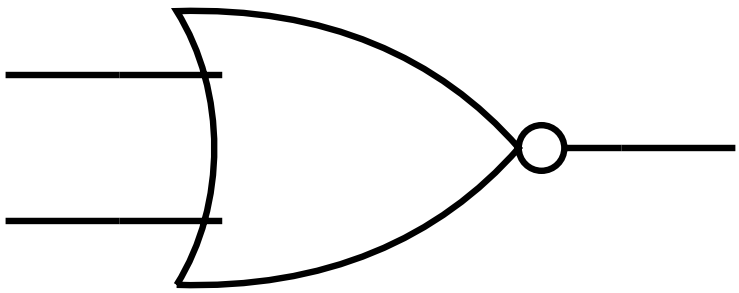


Figure 17: Schematic symbol for the two input NOR gate

36.6 SR-Latch

Use boolean expressions to figure out how gates work.

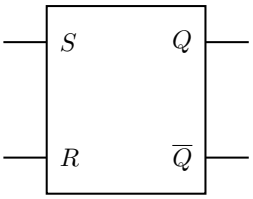
Remember De-Morgan

$$\overline{AB} = \overline{A} + \overline{B}$$
$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$Q = \overline{R\overline{Q}} = \overline{R} + \overline{\overline{Q}} = \overline{R} + Q$$

$$\overline{Q} = \overline{S\overline{Q}} = \overline{S} + \overline{Q} = \overline{S} + \overline{Q}$$

S	R	Q	$\overline{Q}$
0	0	X	
0	1	0	1
1	0	1	0
1	1	hold	



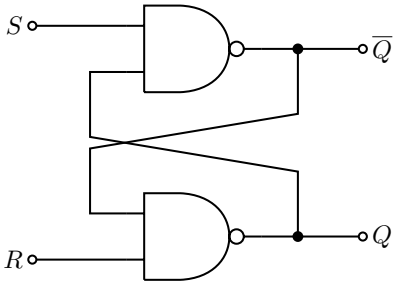


Figure 18: SR-latch truth table and symbol, and the cross-coupled NAND implementation

$$Q = \overline{R} + Q$$

,

$$\overline{Q} = \overline{S} + \overline{Q}$$

S	R	Q	~Q
0	0	X	X
0	1	0	1
1	0	1	0
1	1	Q	~Q

36.7 D-Latch (16 transistors)

C	D	Q	~Q
0	X	Q	~Q
1	0	0	1
1	1	1	0

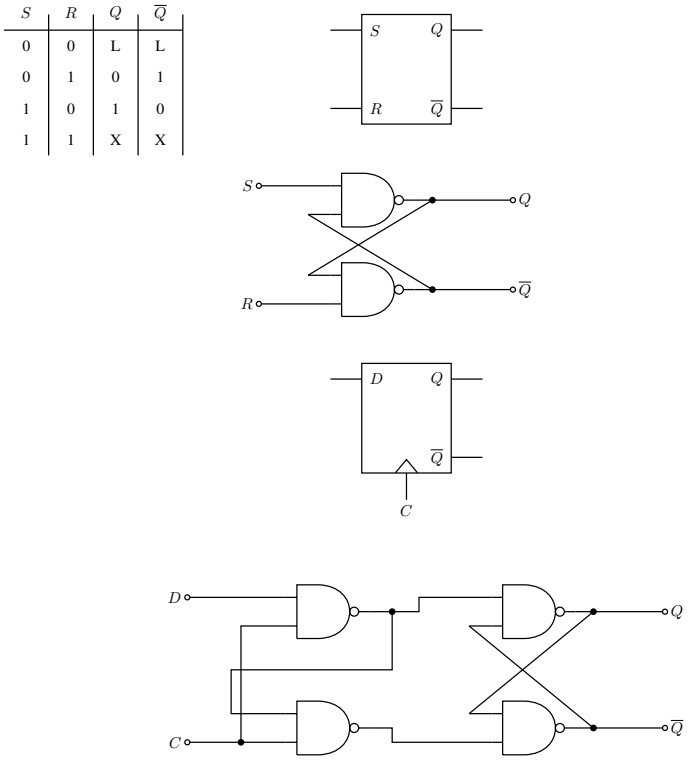


Figure 19: Gate level D latch built from NAND gates, with the cross-coupled NAND SR latch and its truth table above and the full latch driven by clock C below

36.8 Other logic cells

What about

$$Y = AB$$

and

$$Y = A + B$$

?

$$Y = AB = \overline{\overline{AB}}$$

Y = **A** AND **B** = NOT(NOT(**A** AND **B**))

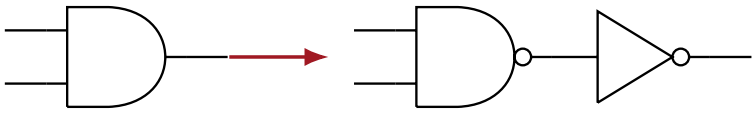


Figure 20: An AND gate built as a NAND followed by an inverter

$$Y = A+B = \overline{\overline{A+B}}$$

Y = **A** OR **B** = NOT(NOT(**A** OR **B**))

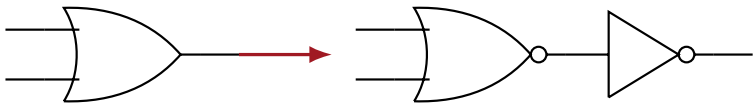
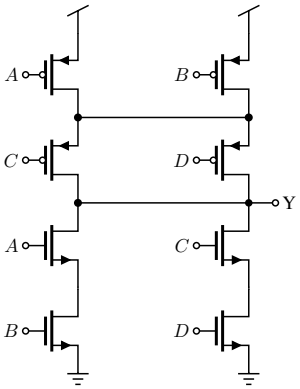


Figure 21: An OR gate built as a NOR followed by an inverter

36.9 AOI22: and or invert

Y = NOT(**A** AND **B** OR **C** AND **D**)

$$Y = \overline{AB + CD}$$



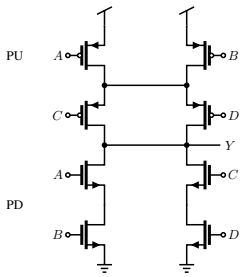


Figure 22: AOI22 gate at transistor level, with the series-parallel pull-up network above the output and the pull-down network below

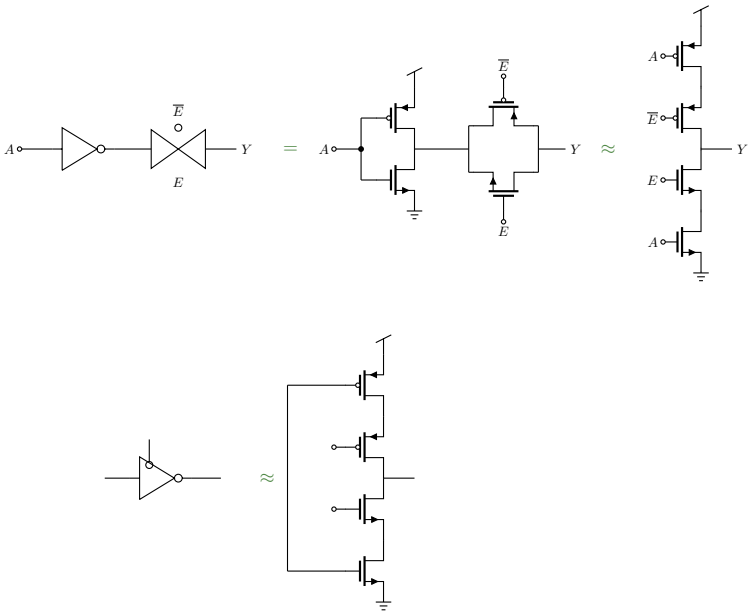


Figure 23: An inverter followed by a transmission gate is equivalent to a tristate inverter, redrawn as a single stack of four series transistors

36.10 Tristate inverter

E	A	Y
0	0	Z
0	1	Z
1	0	1
1	1	0

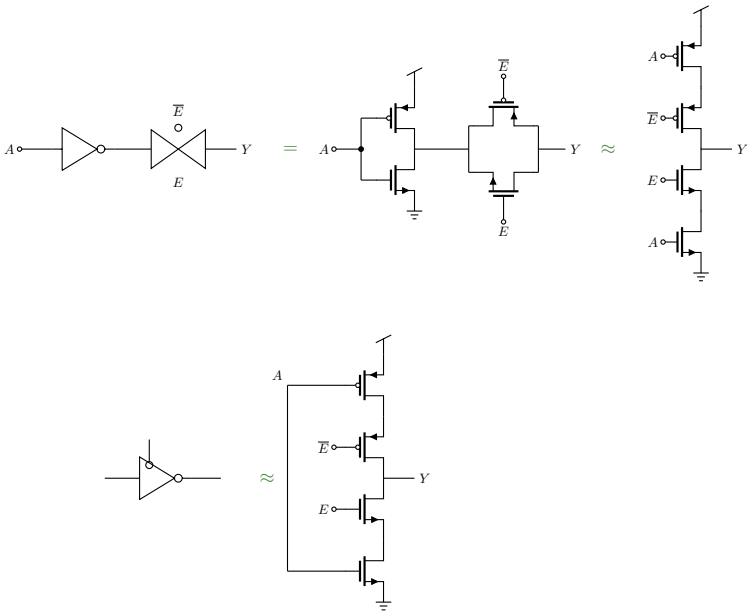


Figure 24: Tristate inverter - symbol and the four transistor stack where input A drives the outer devices and the enable E and its complement drive the inner devices

36.11 Mux

S	Y
0	NOT(P1)
1	NOT(P0)

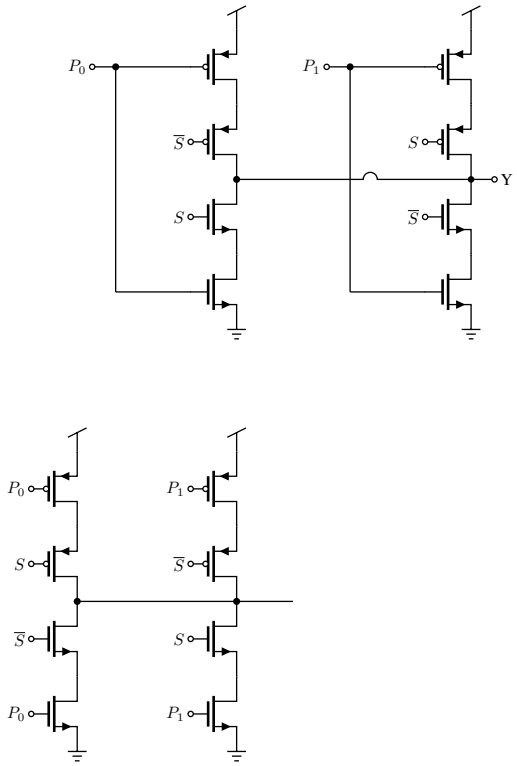


Figure 25: Two input multiplexer built from two tristate inverters, with select S and its complement steering either P0 or P1 to output Y

D-Latch (12 transistors)

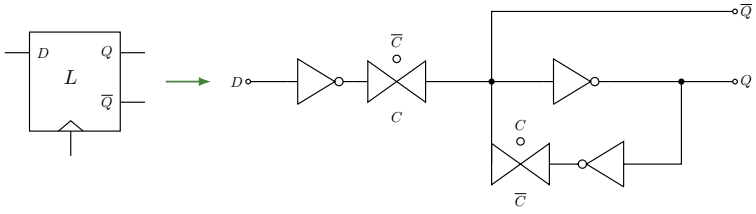


Figure 26: Twelve transistor D latch - input inverter, clocked transmission gate and a feedback inverter pair holding the state, shown as symbol, gate level and transistor level

D-Flip Flop (< 26 transistors)

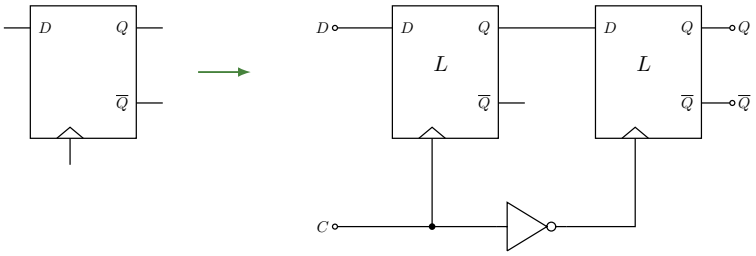


Figure 27: D flip-flop drawn as two D latches in series clocked on opposite phases of C

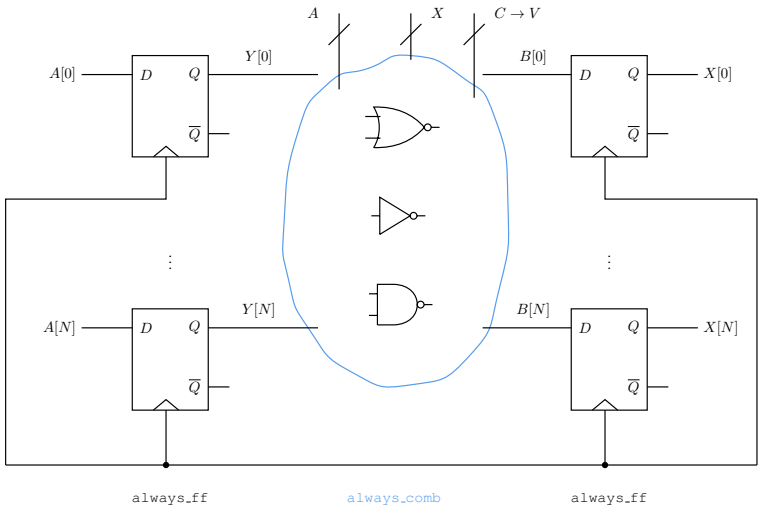


Figure 28: Synchronous digital design, with clocked flip-flop banks at the input and output and a cloud of combinational logic between them

36.12 There are other types of logic

- True single phase clock (TSPC) logic
- Pass transistor logic
- Transmission gate logic
- Differential logic
- Dynamic logic

Consider other types of logic "rule breaking", so you should know why you need it.

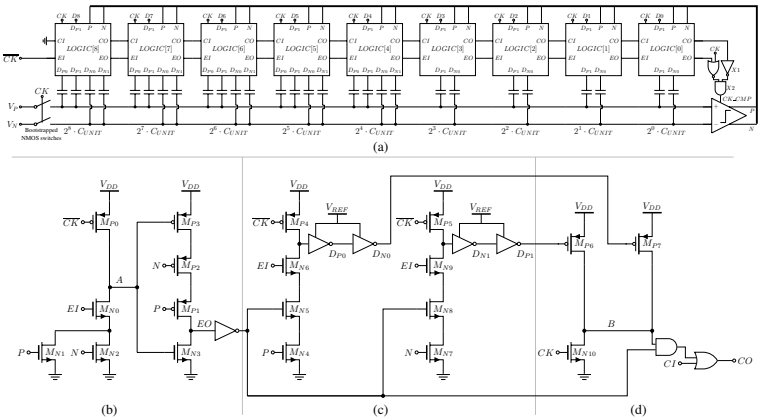


Figure 29: Dynamic logic in a 9-bit SAR ADC - the binary weighted capacitor array with per-bit logic slices above, and the transistor level dynamic cells below. © 2017 IEEE. Reprinted, with permission, from [1]

Dynamic logic => A Compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for Bluetooth Low Energy Receivers [1]

36.13 Speed

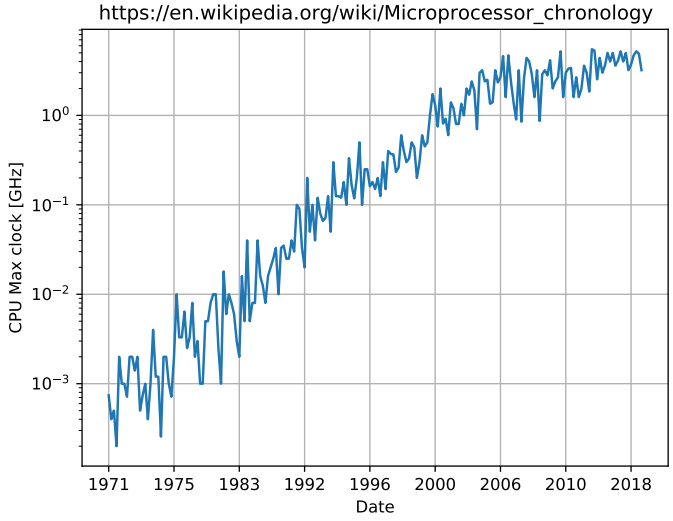


Figure 30: Maximum microprocessor clock frequency in GHz plotted against year from 1971 to 2018, on a logarithmic axis

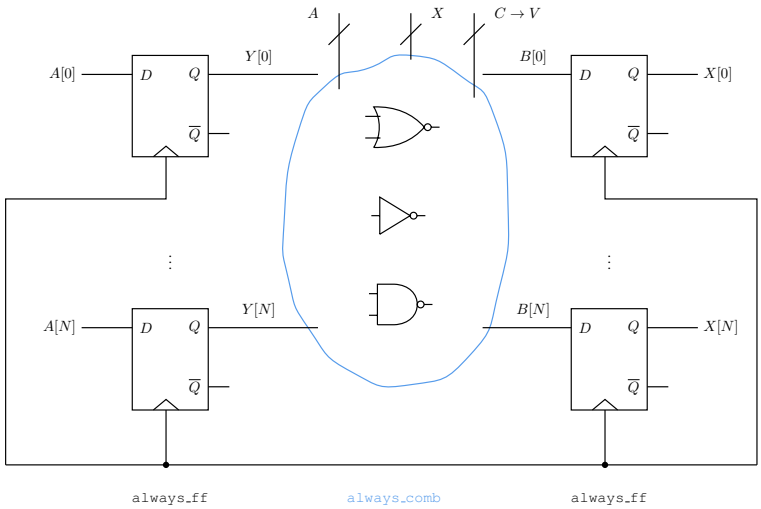


Figure 31: The same flip-flop and combinational logic structure, whose maximum clock rate is set by the delay through the logic between two flip-flops

36.14 Flip-flops and speed

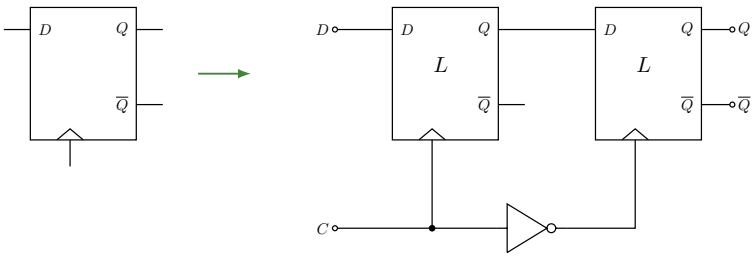


Figure 32: D flip-flop as two cascaded latches, alongside the SPICE subcircuit of the DFRNQNX1 standard cell that implements it

dicex/lib/SUN_TR_GF130N.spi:

```
.SUBCKT DFRNQNX1_CV D CK RN Q QN AVDD AVSS
XA0 AVDD AVSS TAPCELLB_CV
XA1 CK RN CKN AVDD AVSS NDX1_CV
XA2 CKN CKB AVDD AVSS IVX1_CV
XA3 D CKN CKB A0 AVDD AVSS IVTRIX1_CV
XA4 A1 CKB CKN A0 AVDD AVSS IVTRIX1_CV
XA5 A0 A1 AVDD AVSS IVX1_CV
XA6 A1 CKB CKN QN AVDD AVSS IVTRIX1_CV
XA7 Q CKN CKB RN QN AVDD AVSS NDTRIX1_CV
XA8 QN Q AVDD AVSS IVX1_CV
.ENDS
```

Setup time: How long before clk does the data need to change

The setup time is not a number the flip-flop advertises, it is a number you measure. Sweep the moment the data changes relative to the rising clock edge, simulate, and look at where the output stops following the input. The two plots below are two points either side of that boundary, 8 ps and 10 ps.

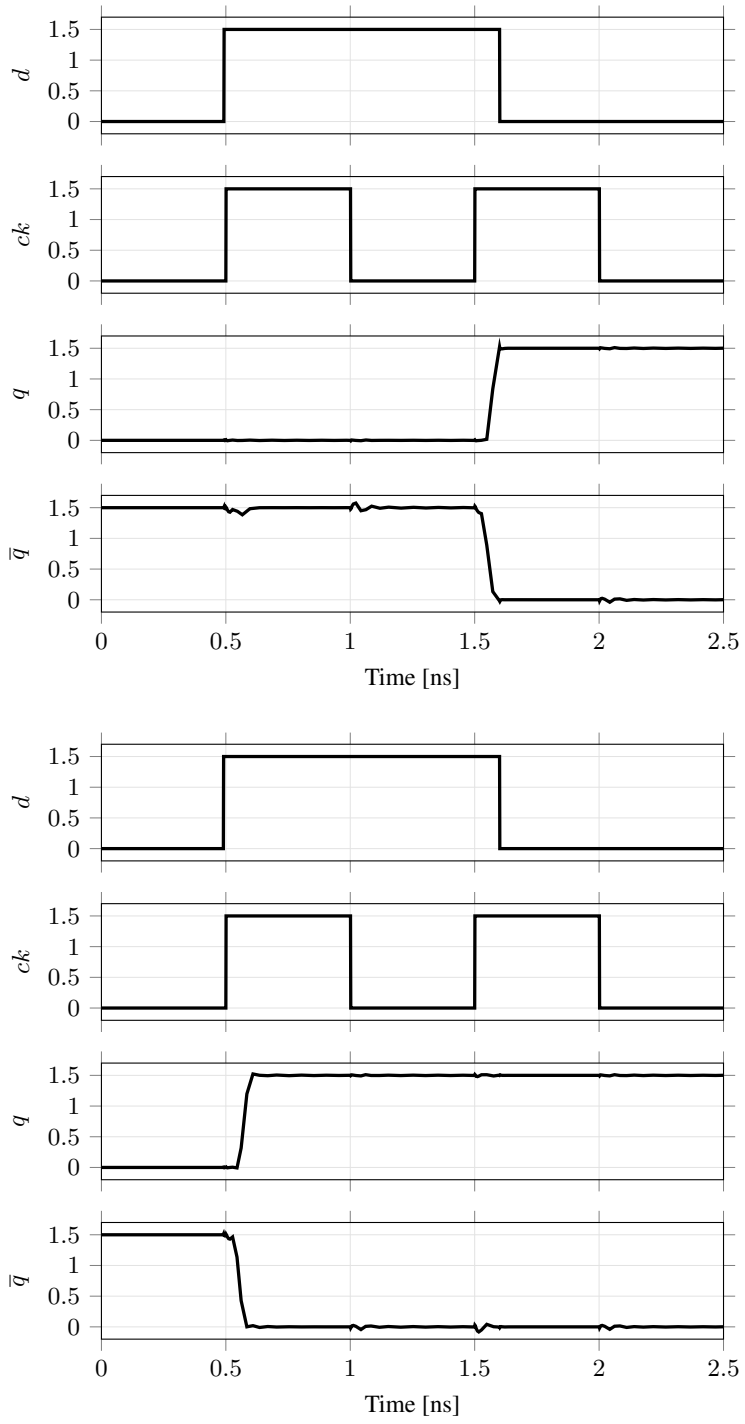


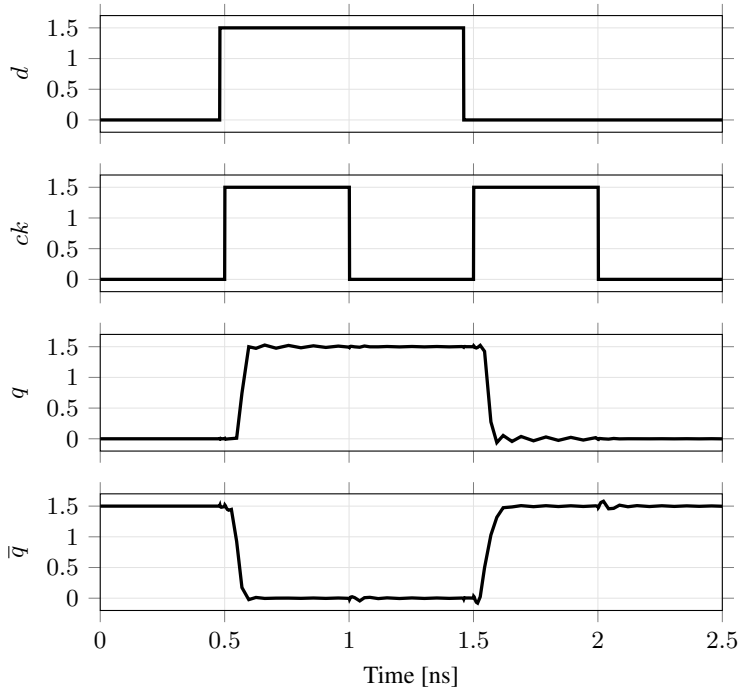
Figure 33: Simulated d, ck, q and qn of the D flip-flop for two setup times, 8 ps and 10 ps. With 8 ps the data changes too close to the rising clock edge at 0.5 ns, the flip-flop does not capture it, and q only goes high at the second edge at 1.5 ns. With 10 ps the data has settled early enough and q rises with the first edge. Two picoseconds separate the two

Left of that boundary the flip-flop still switches, but late: notice how q in the 8 ps plot rises a full clock period after it should. That is the failure mode setup violations produce in a real chip. Nothing is stuck, nothing looks broken on a scope, the data simply

arrives one cycle behind, and it only happens on the corners and the temperatures where the launching path is slowest.

Hold time: How long after clk can the data change

Hold time is the same experiment run from the other side. Now the question is not whether the data arrived early enough to be captured, but whether it stayed put long enough afterwards for the capture to finish. Move the data edge towards the clock edge and the flip-flop eventually samples the *new* value instead of the old one.



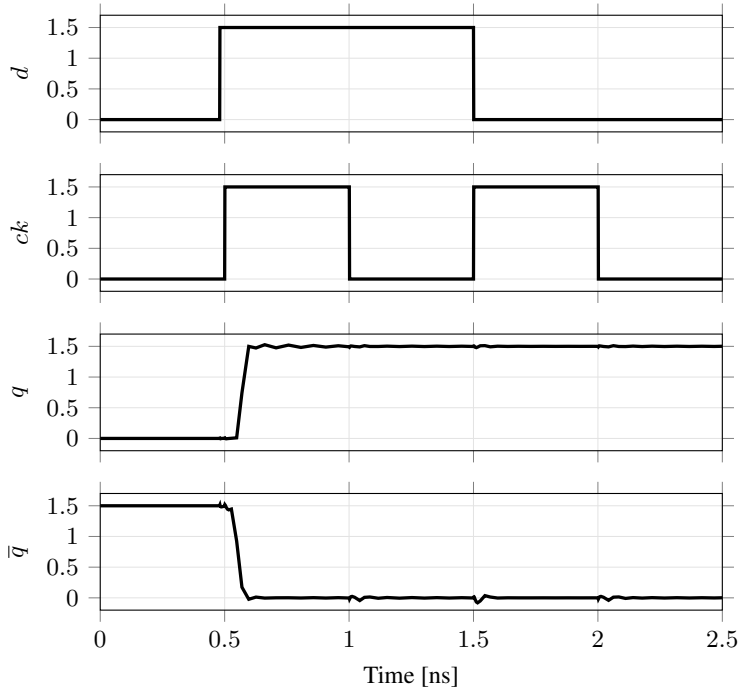


Figure 34: The same signals for two hold times, -40 ps and -30 ps, that is, the data changing 40 ps and 30 ps before the second rising clock edge at 1.5 ns. At -40 ps the flip-flop takes the new low value and q falls. At -30 ps the change is not taken and q stays high for another period

Hold violations are worse than setup violations, and it is worth being clear about why. A setup violation you can fix after the fact by slowing the clock down; the path simply needs more time, and the same silicon works at a lower frequency. A hold violation does not care what the clock frequency is. The data races the clock over a distance that has nothing to do with the period, so a chip that fails hold fails at every frequency, including DC, and the only fix is more silicon: buffers inserted in the fast path, which means another place and route pass.

36.15 Timing analysis

Analyze arrival times of all nodes in a combinatorial circuit

$$arrival_i = max_{j \in fanin(i)} arrival_j + t_{pd_i} \Rightarrow a_i = max_{j \in fanin(i)} a_j + t_{pd_i}$$

$$slack_i = required_i - arrival_i$$

Positive slack (over PVT*) => Timing is OK Negative slack (over PVT†) => Timing is not OK

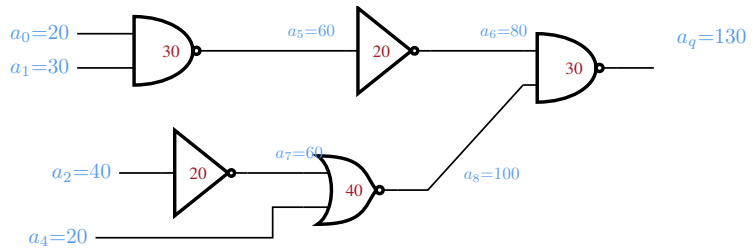


Figure 35: Arrival time propagation through a small combinational circuit, where each gate adds its delay to the largest arrival time at its inputs to give 130 at the output

* PVT => Process, Voltage, Temperature
† PVT => Process, Voltage, Temperature

36.16 Timing analysis tools

Commercial Cadence Tempus

Synopsys PrimeTime

Free OpenTimer

36.16.1 What is timing analysis

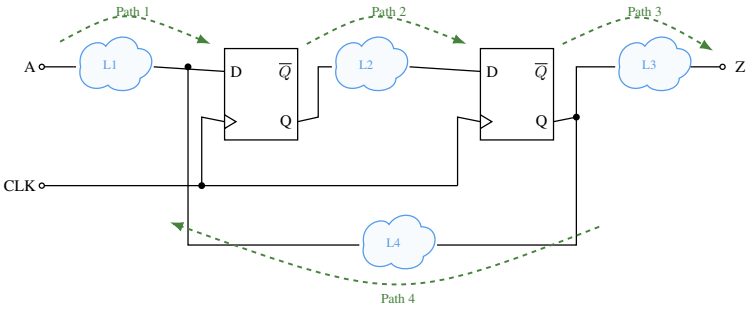


Figure 36: Timing paths through a circuit, each running from a launch point through combinational logic to a capture point, with a table of the four startpoint and endpoint combinations

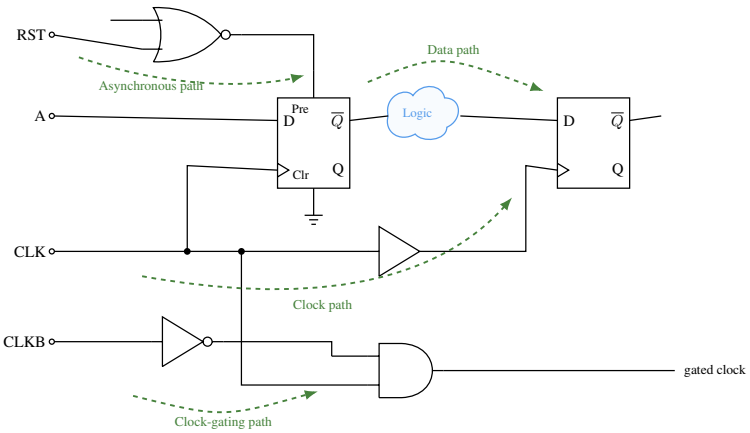


Figure 37: The path types considered in timing analysis - data path, clock path, asynchronous reset path and clock-gating path

36.16.1.1 What do the tools need?

Input and output delay paths as a function of input transition time and capacitive load, setup and hold time.

osu018_stdcells.lib

```
cell (INVX1) {
  cell_footprint : inv;
area : 16;
  cell_leakage_power : 0.0221741;
  pin(A)  {
    direction : input;
    capacitance : 0.00932456;
    rise_capacitance : 0.00932196;
    fall_capacitance : 0.00932456;
  }
  pin(Y)  {
    direction : output;
    capacitance : 0;
    rise_capacitance : 0;
    fall_capacitance : 0;
    max_capacitance : 0.503808;
    function : "(!A)";
    timing() {
      related_pin : "A";
      timing_sense : negative_unate;
      cell_fall(delay_template_5x5) {
        index_1 ("0.005, 0.0125, 0.025, 0.075, 0.15");
        index_2 ("0.06, 0.18, 0.42, 0.6, 1.2");
        values ( \
          "0.030906, 0.037434, 0.038584, 0.039088, 0.030318", \
          "0.04464, 0.057551, 0.073142, 0.077841, 0.081003", \
          "0.064368, 0.091076, 0.11557, 0.126352, 0.144944", \
          "0.139135, 0.174422, 0.232659, 0.261317, 0.321043", \
          "0.249412, 0.28434, 0.357694, 0.406534, 0.51187");
      }


      fall_transition(delay_template_5x5) {
        index_1 ("0.005, 0.0125, 0.025, 0.075, 0.15");
        index_2 ("0.06, 0.18, 0.42, 0.6, 1.2");
        values ( \
          "0.032269, 0.0648, 0.087, 0.1032, 0.1476", \
          "0.036025, 0.0726, 0.1044, 0.1236, 0.183", \
          "0.06, 0.0882, 0.1314, 0.1554, 0.2286", \
          "0.1494, 0.1578, 0.2124, 0.2508, 0.3528", \
          "0.288, 0.2892, 0.3192, 0.3576, 0.492");
      }
      cell_rise(delay_template_5x5) {
        index_1 ("0.005, 0.0125, 0.025, 0.075, 0.15");
        index_2 ("0.06, 0.18, 0.42, 0.6, 1.2");
        values ( \
          "0.037639, 0.056898, 0.083401, 0.104927, 0.156652", \
          "0.05258, 0.083003, 0.119028, 0.141927, 0.207952", \
          "0.07402, 0.112622, 0.162437, 0.191122, 0.271755", \
          "0.15767, 0.201007, 0.284096, 0.331746, 0.452958", \
          "0.285016, 0.326868, 0.415086, 0.481337, 0.653064");
      }
      rise_transition(delay_template_5x5) {
        index_1 ("0.005, 0.0125, 0.025, 0.075, 0.15");
        index_2 ("0.06, 0.18, 0.42, 0.6, 1.2");
        values ( \
          "0.031447, 0.059488, 0.0846, 0.0918, 0.138", \
          "0.047167, 0.0786, 0.1044, 0.1224, 0.1734", \
          "0.072, 0.096, 0.1398, 0.1578, 0.222", \
          "0.1866, 0.1914, 0.2358, 0.2748, 0.3696", \
          "0.3648, 0.3648, 0.384, 0.4146, 0.5388");
      }
    }
    internal_power() {
      related_pin : "A";
      fall_power(energy_template_5x5) {
```

```
        index_1 ("0.005, 0.0125, 0.025, 0.075, 0.15");
        index_2 ("0.06, 0.18, 0.42, 0.6, 1.2");
        values ( \
          "0.009213, 0.004772, 0.00823, 0.018532, 0.054083", \
          "0.009047, 0.005677, 0.005713, 0.015244, 0.049453", \
          "0.008669, 0.006332, 0.002998, 0.01159, 0.04368", \
          "0.007879, 0.007243, 0.001451, 0.004701, 0.030385", \
          "0.007605, 0.007297, 0.003652, 0.000737, 0.020842");
      }
      rise_power(energy_template_5x5) {
        index_1 ("0.005, 0.0125, 0.025, 0.075, 0.15");
        index_2 ("0.06, 0.18, 0.42, 0.6, 1.2");
        values ( \
          "0.023555, 0.029044, 0.041387, 0.051684, 0.087278", \
          "0.023165, 0.028621, 0.039211, 0.048916, 0.083039", \
          "0.023574, 0.02752, 0.036904, 0.045723, 0.077971", \
          "0.024479, 0.025247, 0.032268, 0.039242, 0.066587", \
          "0.024942, 0.025187, 0.029612, 0.034835, 0.057524");
      }
    }
  }
}
```

36.17 Every gate must be simulated to provide behavior over input transition and load capacitance

36.18 All analog blocks must have associated liberty file to describe behavior and timing paths *If you integrate analog into digital top flow*

36.19 Gate Delay

36.19.1 Delay definitions

Parameter	Name	Description
t_pdr	max rising propagation delay	input to rising output cross 50 %
t_pdf	max falling propagation delay	input to falling output cross 50 %
t_pd	propagation delay	t_pd = (t_pdr + t_pdf)/2
t_r	rise time	20 % to 80 %

Parameter	Name	Description
t_f	fall time	80 % to 20 %
t_cdr	min rising contamination delay	input to rising output cross 50 %
t_cdf	min falling contamination delay	input to falling output cross 50 %
t_cd	contamination delay	t_cd = (t_cdr + t_cdf)/2

36.20 Delay estimation

How can we get a reasonably accurate hand calculation model of delay?

$$C \approx 1 \text{ fF}/\mu\text{m}$$

$$R \approx 1 \text{ k}\Omega\mu\text{m}$$

36.20.1 Inverter with inverter load

$$C \approx 1 \text{ fF}/\mu\text{m}$$

,

$$R \approx 1 \text{ k}\Omega\mu\text{m}$$

$$t_{pd} = R \times 6C = 6RC$$

$$t_{pd} = 6 \times 1 \times 10^3 \times 1 \times 10^{-15} \text{ s}$$

$$t_{pd} = 6 \times 10^{-12} = 6 \text{ ps}$$

36.21 Elmore Delay

$$t_{pd} \approx \sum_{\text{nodes}} R_{\text{nodes}-to-source} C_i$$

$$= R_1C_1 + (R_1 + R_2)C_2 + ... + (R_1 + R_2 + ... + R_N)C_N$$

Good enough for hand calculation

36.22 Delay components

Parasitic delay (p)

p = 9 or 12 RC

Independent of load capacitance

Effort delay (f)

f = 5h RC

Proportional to load capacitance

Let's use process independent unit

$$d = \frac{d_{real}}{\tau}$$

,

$$\tau = 3RC$$

Parasitic delay

$$\Rightarrow p = 12RC/3RC = 4$$

Effort delay

$$\Rightarrow f = 5hRC/3RC = \frac{5}{3}h$$

Delay

$$\Rightarrow d = f + p = \frac{5}{3}h + 4$$

Logical effort (g) is the ratio of the input capacitance of a gate to the input capacitance of an inverter delivering the same output current

Parasitic delay

$$\Rightarrow p = 4$$

Logic effort

$$\Rightarrow g = \frac{5}{3}$$

Electrical effort

$$\Rightarrow h = 1$$

Effort

$$\Rightarrow f = gh$$

Delay

$$\Rightarrow d = f + p = gh + p = 5\frac{2}{3}$$

Real delay

$$\Rightarrow d = 5\frac{2}{3} \times 3 \text{ ps} = 17 \text{ ps}$$

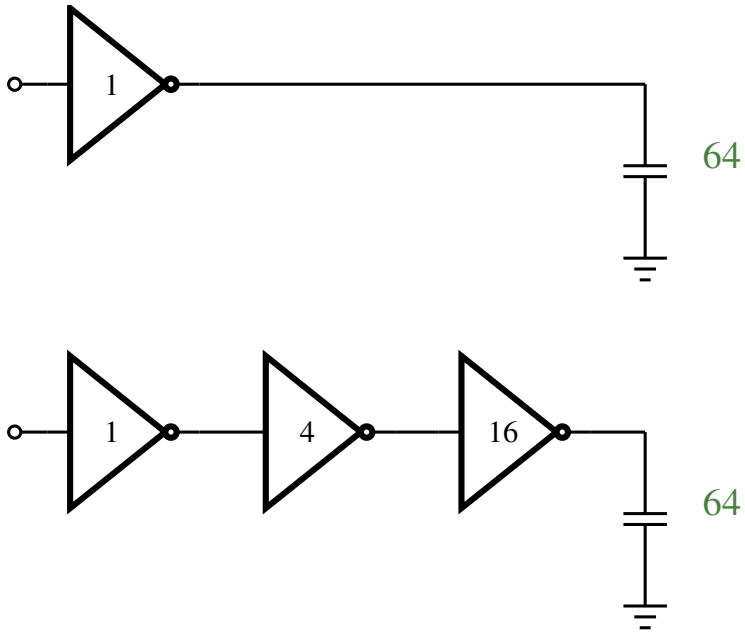
Term	Stage Expression	Path Expression
number of stages	1	N
logical effort	g	$G = \prod(g_i)$
electrical effort	$h = \frac{C_{in}}{C_{out}}$	$H = \frac{C_{out(path)}}{C_{in(path)}}$
branching effort	$b = \frac{C_{onpath}+C_{offpath}}{C_{onpath}}$	$B = \prod b_i$
effort	$f = gh$	$F = GBH$
effort delay	f	$D_F = \sum f_i$
parsitic delay	p	$P = \sum p_i$
delay	$d = f + p$	$D = \sum d_i = D_F + P$

Figure 38: Logical effort summary table giving the stage and path expressions for number of stages, logical effort, electrical effort, branching effort, effort, effort delay, parasitic delay and total delay

36.23 Modern IC timing analysis requires computers with advanced programs‡

36.24 Best number of stages

36.25 Which has shortest delay?



‡ Opportunity for good programmers

Figure 39: Two ways to drive a load of 64 - a single inverter, and a chain of three inverters sized 1, 4 and 16, with the path effort delay worked out for each

Term	Stage Expression	Path Expression
number of stages	1	N
logical effort	g	$G = \prod(g_i)$
electrical effort	$h = \frac{C_{in}}{C_{out}}$	$H = \frac{C_{out(path)}}{C_{in(path)}}$
branching effort	$b = \frac{C_{onpath}+C_{offpath}}{C_{onpath}}$	$B = \prod b_i$
effort	$f = gh$	$F = GBH$
effort delay	f	$D_F = \sum f_i$
parsitic delay	p	$P = \sum p_i$
delay	$d = f + p$	$D = \sum d_i = D_F + P$

Figure 40: The logical effort table used to evaluate the two candidate paths, giving H equal to 64, G equal to 1, B equal to 1 and path effort F equal to 64

$$H = C_{cout}/C_{in} = 64$$

$$G = \prod g_i = \prod 1 = 1$$

$$B = 1$$

$$F = GBH = 64$$

One stage, with Sutherland's classic

$$p \approx 1$$

per inverter

$$f = 64 \Rightarrow D = 64 + 1 = 65$$

Three stage with

$$f = 4$$

$$D_F = 12, p = 3 \Rightarrow D = 12 + 3 = 15$$

The classic textbook numbers above use Sutherland's parasitic delay of about 1 per inverter; our diffusion-heavy estimate earlier gave $p = 4$. Redo the sums with $p = 4$ and you get 68 against 24 -

the absolute numbers move, the conclusion does not: split the path into stages of effort around four.

For close to optimal delay, use

$$f = 4$$

(Used to be

$$f = e$$

)

36.26 Trends

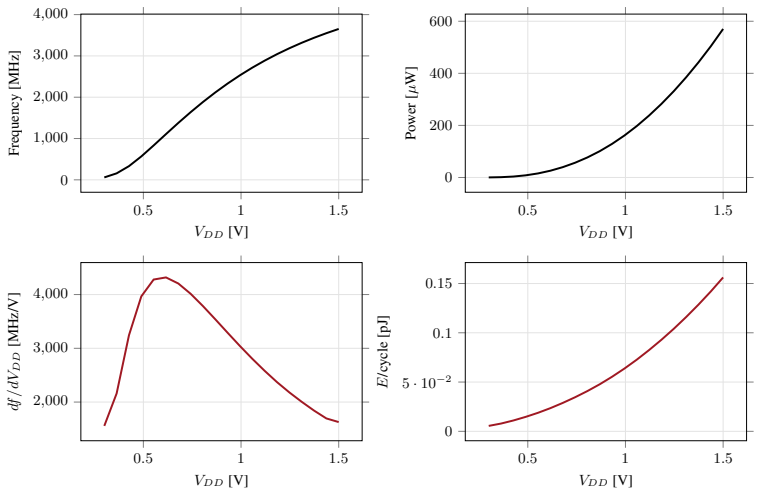


Figure 41: Ring oscillator frequency and power against supply voltage, with the supply sensitivity and the energy per cycle below. The sensitivity peaks near 0.6 V at more than 4 GHz per volt, and the energy per cycle is worst where the ring is fastest

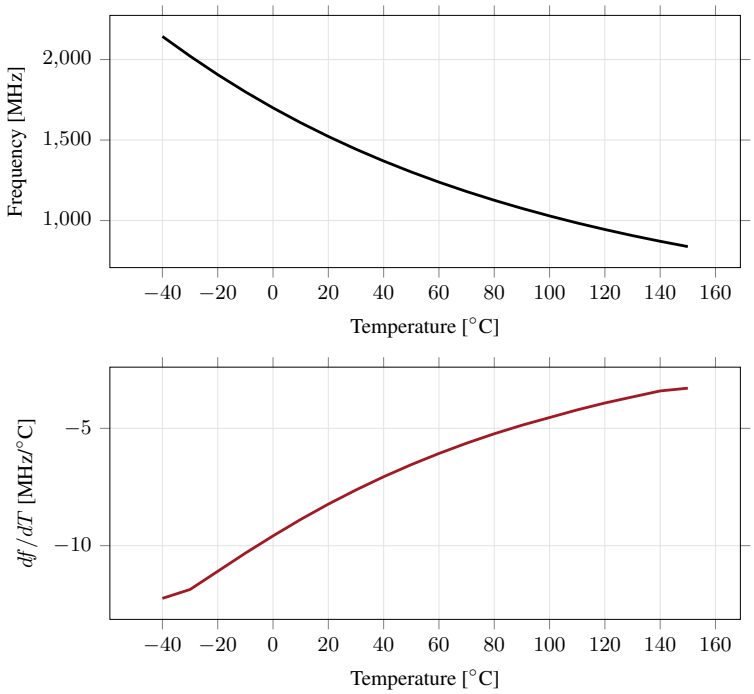


Figure 42: Ring oscillator frequency falling from 2.14 GHz to 0.84 GHz, a factor of 2.6, as temperature rises from minus 40 to 150 degrees Celsius. The slope below runs from about minus 12 MHz per degree in the cold to minus 3 when hot, so the sensitivity is itself temperature dependent

36.27 Attack vector

```
module counter(
               output logic [WIDTH-1:0] out,
               input logic              clk,
               input logic              reset
               );

   parameter WIDTH = 8;

   logic [WIDTH-1:0]                    count;
   always_comb begin
      count = out + 1;
   end

   always_ff @(posedge clk or posedge reset) begin
      if (reset)
        out <= 0;
      else
        out <= count;
   end

endmodule // counter
```

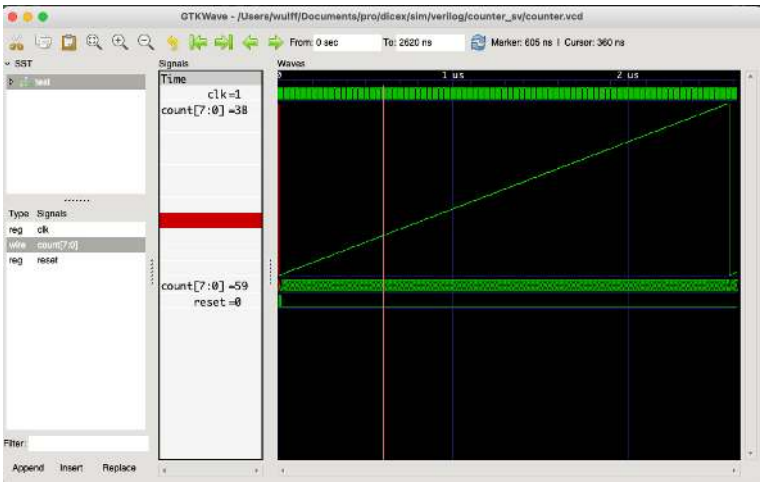


Figure 43: GTKWave view of the 8-bit counter testbench, where the count bus ramps over 2.6 us while clk toggles and reset stays low

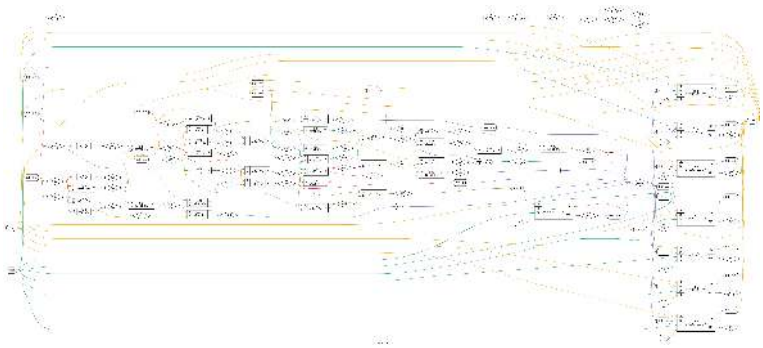


Figure 44: The same 8-bit counter synthesized to the SUN TR GF130N standard cell library, with eight D flip-flops on the right and the increment logic to the left

```
.SUBCKT counter out_7 out_6 out_5 out_4 out_3 out_2 out_1 out_0 clk reset AVDD AVSS
* SPICE netlist generated by Yosys 0_9 (git sha1 1979e0b1, gcc 10_3_0-1ubuntu1~20_10 -fPIC -Os
X0 out_2 1 AVDD AVSS IVX1_CV
X1 out_3 2 AVDD AVSS IVX1_CV
X2 out_4 3 AVDD AVSS IVX1_CV
X3 out_5 4 AVDD AVSS IVX1_CV
X4 out_6 5 AVDD AVSS IVX1_CV
X5 out_0 6 AVDD AVSS IVX1_CV
X6 out_1 7 AVDD AVSS IVX1_CV
X7 6 7 8 AVDD AVSS NRX1_CV
X8 out_0 out_1 9 AVDD AVSS NDX1_CV
X9 1 9 10 AVDD AVSS NRX1_CV
X10 10 11 AVDD AVSS IVX1_CV
X11 2 11 12 AVDD AVSS NRX1_CV
X12 out_3 10 13 AVDD AVSS NDX1_CV
X13 out_3 10 14 AVDD AVSS NRX1_CV
X14 12 14 15 AVDD AVSS NRX1_CV
X15 3 13 16 AVDD AVSS NRX1_CV
```

```
X16 16 17 AVDD AVSS IVX1_CV
X17 out_4 12 18 AVDD AVSS NRX1_CV
X18 16 18 19 AVDD AVSS NRX1_CV
X19 4 17 20 AVDD AVSS NRX1_CV
X20 out_5 16 21 AVDD AVSS NDX1_CV
X21 out_5 16 22 AVDD AVSS NRX1_CV
X22 20 22 23 AVDD AVSS NRX1_CV
X23 5 21 24 AVDD AVSS NRX1_CV
X24 out_6 20 25 AVDD AVSS NRX1_CV
X25 24 25 26 AVDD AVSS NRX1_CV
X26 out_7 24 27 AVDD AVSS NRX1_CV
X27 out_7 24 28 AVDD AVSS NDX1_CV
X28 28 29 AVDD AVSS IVX1_CV
X29 27 29 30 AVDD AVSS NRX1_CV
X30 out_0 out_1 31 AVDD AVSS NRX1_CV
X31 8 31 32 AVDD AVSS NRX1_CV
X32 out_2 8 33 AVDD AVSS NRX1_CV
X33 10 33 34 AVDD AVSS NRX1_CV
X34 35 clk AVSS reset out_0 35 AVDD AVSS DFSRQNX1_CV
X35 32 clk AVSS reset out_1 36 AVDD AVSS DFSRQNX1_CV
X36 34 clk AVSS reset out_2 37 AVDD AVSS DFSRQNX1_CV
X37 15 clk AVSS reset out_3 38 AVDD AVSS DFSRQNX1_CV
X38 19 clk AVSS reset out_4 39 AVDD AVSS DFSRQNX1_CV
X39 23 clk AVSS reset out_5 40 AVDD AVSS DFSRQNX1_CV
X40 26 clk AVSS reset out_6 41 AVDD AVSS DFSRQNX1_CV
X41 30 clk AVSS reset out_7 42 AVDD AVSS DFSRQNX1_CV
V0 count_0 35 DC 0
V1 43 out_2 DC 0
V2 44 out_3 DC 0
V3 count_3 15 DC 0
V4 45 out_4 DC 0
V5 count_4 19 DC 0
V6 46 out_5 DC 0
V7 count_5 23 DC 0
V8 47 out_6 DC 0
V9 count_6 26 DC 0
V10 48 out_7 DC 0
V11 count_7 30 DC 0
V12 49 out_0 DC 0
V13 50 out_1 DC 0
V14 count_1 32 DC 0
V15 count_2 34 DC 0
.ENDS
```

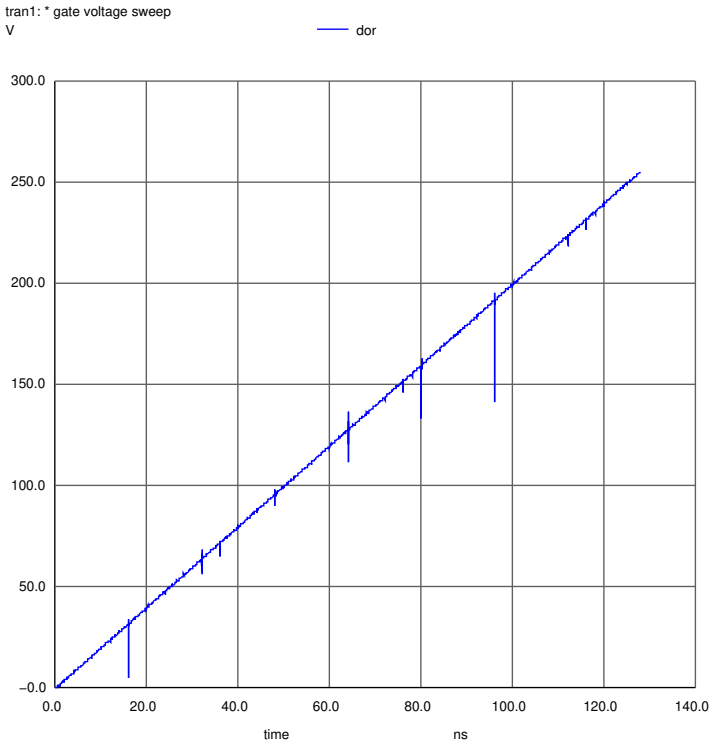


Figure 45: ngspice transient of the reference counter output dor, counting linearly from 0 to 255 over 128 ns

dicex/sim/verilog/counter_sv/counter_attack_tb.cir

```
VDDA AVDD_ATTACK 0 dc 0.5 pulse(1.5 0.6 tcd trf trf tapw taper)
```

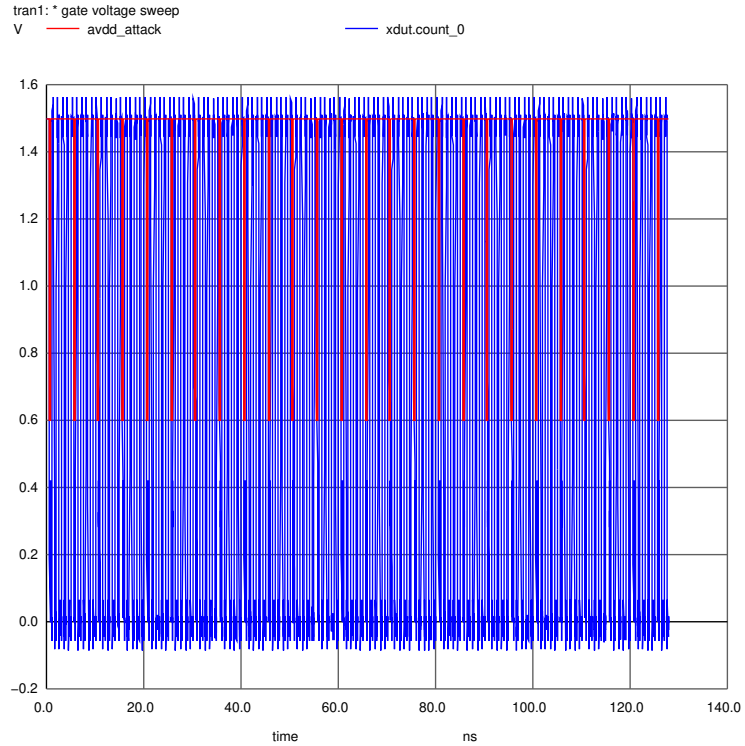


Figure 46: ngspice transient of the attacked supply, pulsed between 1.5 V and 0.6 V, together with the least significant counter bit

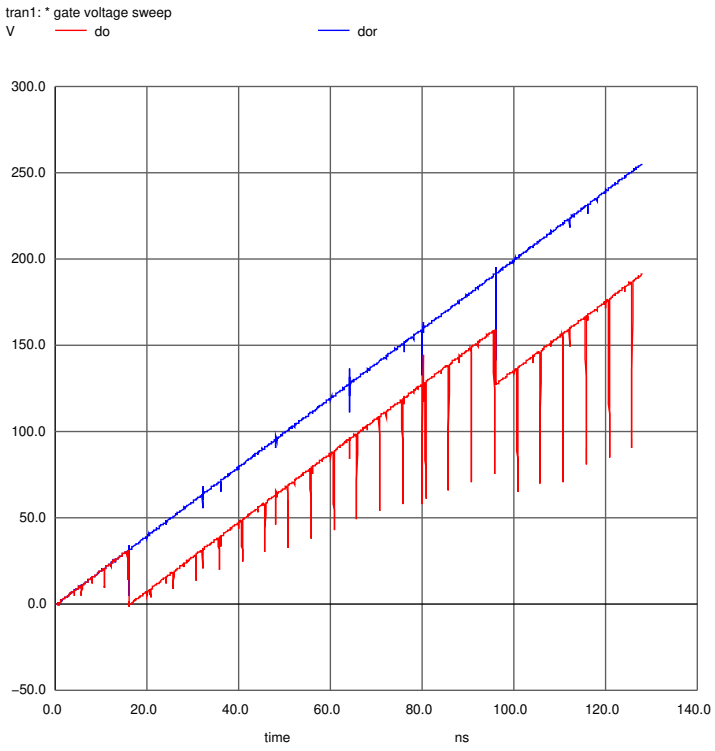


Figure 47: The attacked counter output do in red against the reference output dor in blue, showing the counts lost each time the supply is glitched



Figure 48: The ChipWhisperer web page, an open source platform for side channel power analysis and fault injection attacks on embedded systems. Screenshot of chipwhisperer.io

36.28 Pick two

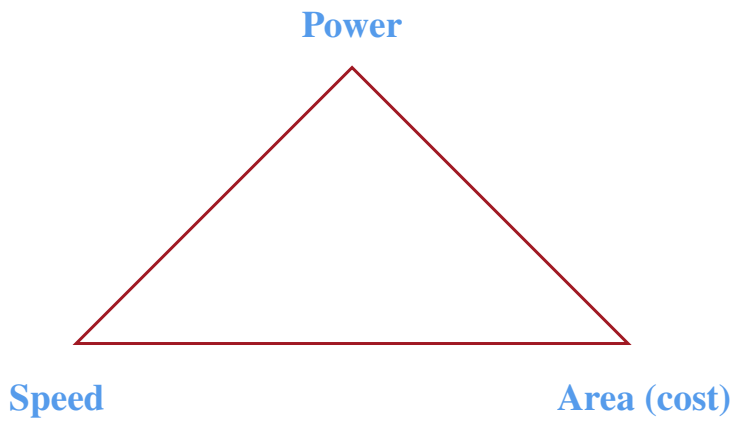


Figure 49: The design trade-off triangle between power, speed and area or cost

36.29 Power

36.30 What is power?

Instantanious power:

$$P(t) = I(t)V(t)$$

Energy :

$$\int_0^T P(t)dt$$

[J]

Average power:

$$\frac{1}{T}\int_0^T P(t)dt$$

[W or J/s]

36.31 Power dissipated in a resistor

Ohm's Law

$$V_R = I_R R$$

$$P_R = V_R I_R = I_R^2 R = \frac{V_R^2}{R}$$

36.32 Charging a capacitor to VDD

Capacitor differential equation

$$I_C = C\frac{dV}{dt}$$

$$E_C = \int_0^\infty I_C V_C dt = \int_0^\infty C\frac{dV}{dt}V_C dt = \int_0^{V_C} CV dV = C\left[\frac{V^2}{2}\right]_0^{V_{DD}}$$

$$E_C = \frac{1}{2}CV_{DD}^2$$

36.33 Energy to charge a capacitor to a voltage VDD

$$E_C = \frac{1}{2}CV_{DD}^2$$

$$I_{VDD} = I_C = C\frac{dV}{dt}$$

$$E_{VDD} = \int_0^\infty I_{VDD}V_{DD}dt = \int_0^\infty C\frac{dV}{dt}V_{DD}dt = CV_{DD}\int_0^{V_{DD}} dV = CV_{DD}^2$$

Only half the energy is stored on the capacitor, the rest is dissipated in the PMOS

36.34 Discharging a capacitor to 0

$$E_C = \frac{1}{2}CV_{DD}^2$$

Voltage is pulled to ground, and the power is dissipated in the NMOS

36.35 Power consumption of digital circuits

$$E_{VDD} = CV_{DD}^2$$

In a clock distribution network (chain of inverters), every output is charged once per clock cycle

$$P_{VDD} = CV_{DD}^2 f$$

36.36 Sources of power dissipation in CMOS logic

$$P_{total} = P_{dynamic} + P_{static}$$

Dynamic power dissipation

Charging and discharging load capacitances

short-circut current, when PMOS and NMOS conduct at the same time

$$P_{dynamic} = P_{switching} + P_{shortcircuit}$$

Static power dissipation

Subthreshold leakage in OFF transistors

Gate leakage (tunneling current) through gate dielectric

Source/drain reverse bias PN junction leakage

$$P_{static} = \left(I_{sub} + I_{gate} + I_{pn}\right) V_{DD}$$

36.37 Switching Power in logic gates

Only output node transitions from low to high consume power from

$$V_{DD}$$

Define

$$P_i$$

to be the probability that a node is 1

Define

$$\overline{P_i} = 1 - P_i$$

to be the probability that a node is 0

Define **activity factor (**

$$\alpha_i$$

) as the **probability of switching a node from 0 to 1**

If the probabilty is uncorrelated from cycle to cycle

$$\alpha_i = \overline{P_i} P_i$$

36.38 Switching probability

Random data

$$P = 0.5$$

,

$$\alpha = 0.25$$

Clocks

$$\alpha = 1$$

Gate	P_Y
AND2	$P_A P_B$
OR2	$1 - \bar{P_A}\bar{P_B}$
NAND2	$1 - P_A P_B$
NOR2	$\bar{P_A}\bar{P_B}$
XOR2	$P_A \bar{P_B} + \bar{P_A} P_B$

Figure 50: Probability that the output of an AND2, OR2, NAND2, NOR2 or XOR2 gate is 1, expressed in the input probabilities PA and PB

Gate	P_Y
AND2	$P_A P_B$
OR2	$1 - \bar{P}_A \bar{P}_B$
NAND2	$1 - P_A P_B$
NOR2	$\bar{P}_A \bar{P}_B$
XOR2	$P_A \bar{P}_B + \bar{P}_A P_B$

Figure 51: The gate output probability table applied to the network below, where each NAND2 output has probability 3 over 4 of being 1

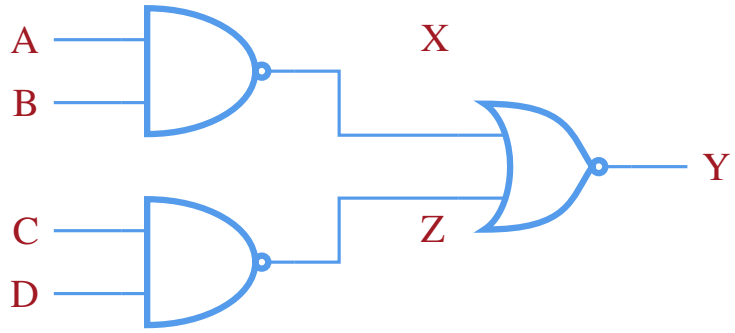


Figure 52: Two NAND2 gates driving a NOR2 gate, giving output probability PY of 1 over 16 and activity factor 15 over 256 when all inputs have probability one half

Assume

$$P = P_A = P_B = P_C = P_D = \frac{1}{2}$$

$$P_X = P_Z = 1 - PP = 1 - \frac{1}{4} = \frac{3}{4}$$

$$\overline{P_X} = \overline{P_Y} = \frac{1}{4}$$

$$P_Y = \frac{1}{4} \times \frac{1}{4} = \frac{1}{16}$$

$$\alpha = \frac{1}{16}\left(1 - \frac{1}{16}\right) = \frac{15}{16}\frac{1}{16} = \frac{15}{256}$$

Gate	P_Y
AND2	$P_A P_B$
OR2	$1 - \bar{P_A}\bar{P_B}$
NAND2	$1 - P_A P_B$
NOR2	$\bar{P_A}\bar{P_B}$
XOR2	$P_A\bar{P_B} + \bar{P_A}P_B$

Figure 53: The same gate output probability table, used here alongside the De Morgan simplification of the network

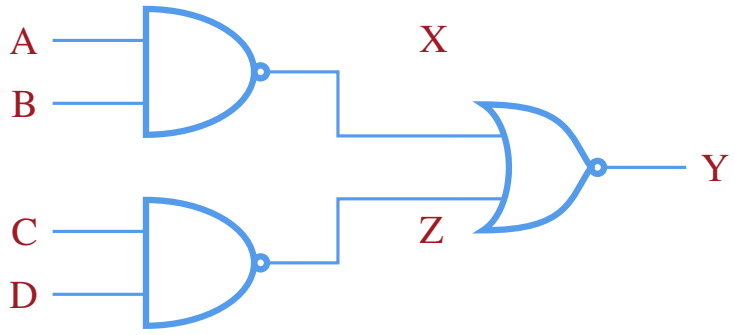


Figure 54: The same NAND-NAND-NOR network, where De Morgan reduces the function to ABCD so that PY is the product of the four input probabilities, 1 over 16

$$\overline{\overline{AB} + \overline{CD}}$$

Use *De Morgan* first

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$\overline{\overline{AB} + \overline{CD}} = \overline{\overline{ABCD}} = ABCD$$

$$\Rightarrow P_Y = P_A P_B P_C P_D = \left(\frac{1}{2}\right)^4 = \frac{1}{16}$$

$$P_{tot} = \alpha C V_{DD}^2 f$$

36.39 Strategies to reduce dynamic power

1. Stop clock
2. Stop activity
3. Reduce clock frequency
4. Turn off VDD
5. Reduce VDD

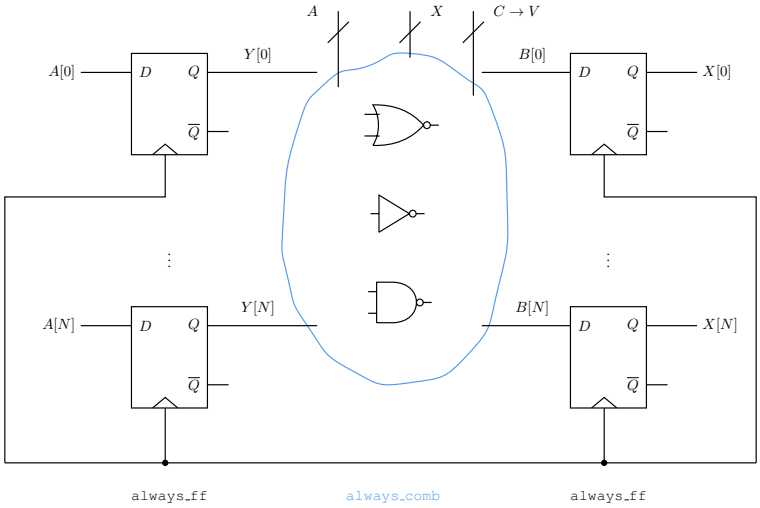


Figure 55: Synchronous logic drawn as banks of D flip-flops separated by a combinational cloud, the switching activity that costs the dynamic power

36.39.1 Stop clock §

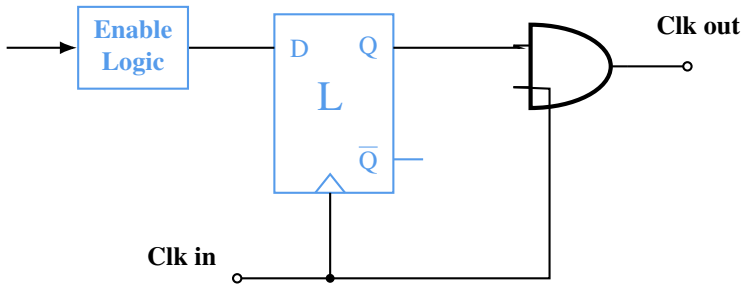


Figure 56: Clock gating, where enable logic feeds a latch and an AND gate so that the output clock only toggles the flip-flops and combinational cloud when the block is enabled

36.39.2 Stop activity

Stopping the clock stops the flip-flops, but it does not necessarily stop the combinational cloud. If the inputs to the cloud keep moving, every gate inside it keeps charging and discharging its load,

§ Often called *clock gating*

and the $\alpha C V_{DD}^2 f$ bill arrives whether or not anything downstream cares about the answer. Stopping the activity means breaking the data path into the cloud, so its inputs are held constant and the logic inside simply stops toggling.

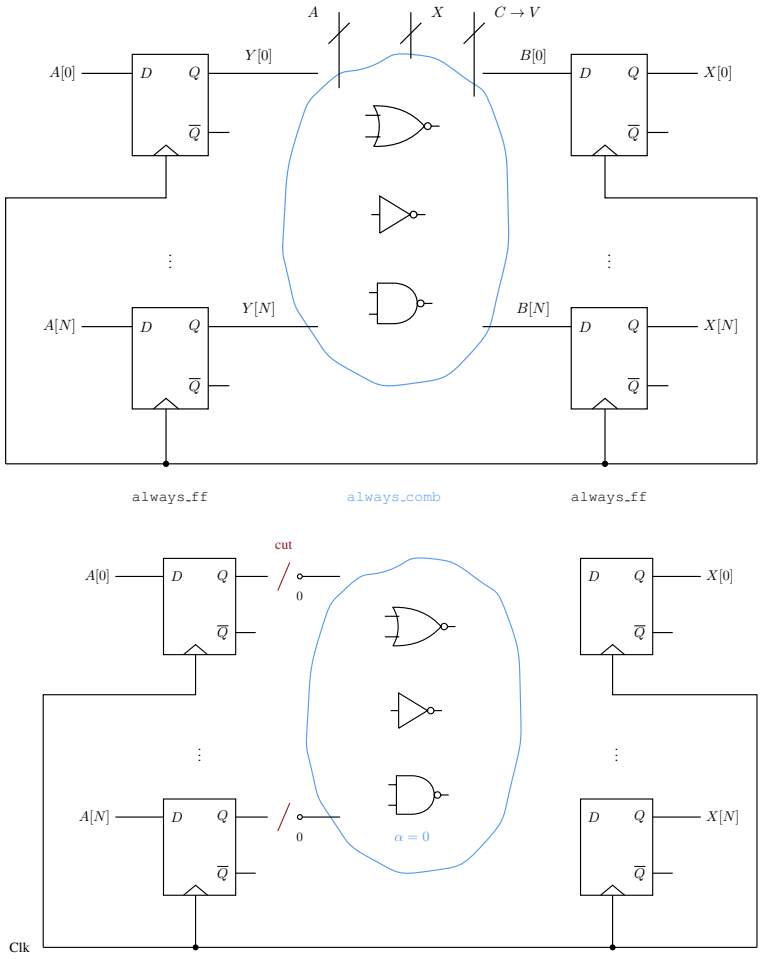


Figure 57: The same flip-flop banks and combinational cloud drawn twice. In the first drawing the cloud is marked at the points where the dynamic power equation can be attacked. In the second the data path from the first flip-flop bank into the cloud is broken, so the cloud sees a constant input and stops switching, while the flip-flops still receive their clock

36.39.3 Reduce frequency

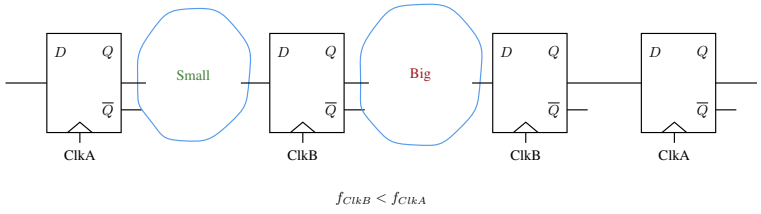


Figure 58: Reducing clock frequency by clocking the big combinational cloud with ClkB and the small cloud with the faster ClkA

36.39.4 Turn off power supply ¶

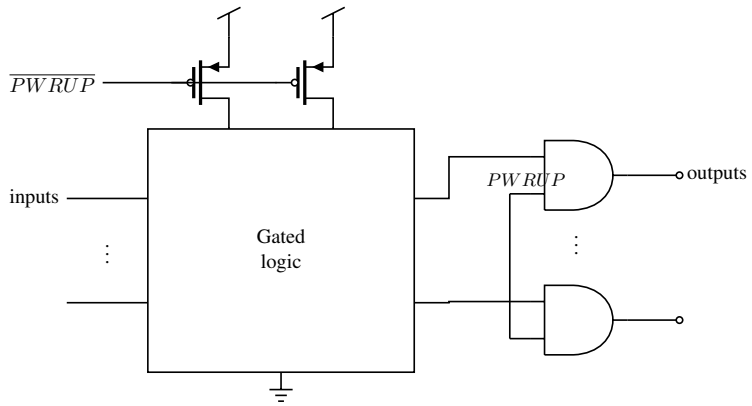


Figure 59: Power gating with PWRUP header switches above the gated logic block and AND gates holding the outputs

36.39.4.1 Reduce power supply

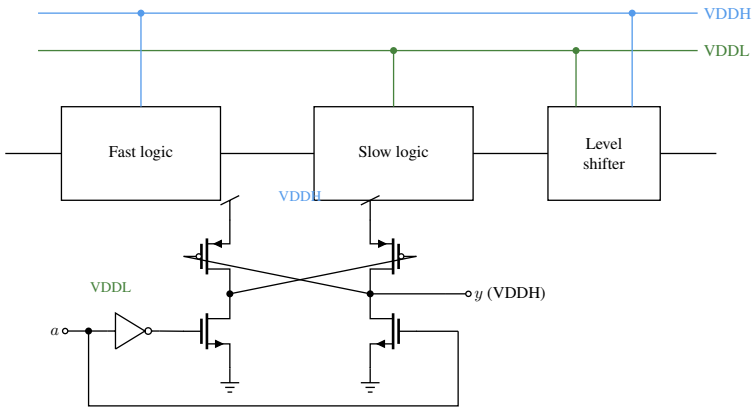


Figure 60: Two supply domains, fast logic on VDDH and slow logic on VDDL, joined by a cross-coupled level shifter

36.39.4.2 Energy-Delay Product

$$EDP = k\frac{C^2V_{DD}^3}{(V_{DD} - V_t)^{1 \text{ to } 2}}$$

Differentiating with respect to

$$V_{DD}$$

and setting the result to

$$0$$

it's possible to work out that

¶ Often called power gating

$$V_{DD-opt} = \frac{3}{3 - 1 \text{ to } 2} V_t \in [1.5, 3] V_t$$

36.40 Wires

36.41 Wire geometry

Pitch = w + s

Aspect ratio (AR) = t/w

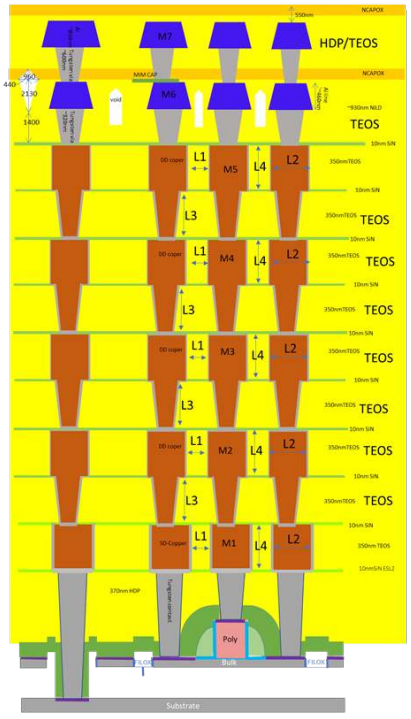
These days

$$AR \approx 2$$

36.42 Metal stack

Often 5 - 10 layers of metal

Metal	Material	Thickness	Purpose
Metal 1	Copper	Thin	in gate routing
Metal 3 - 5	Copper	Thicker	Between gates routing
RDL	Aluminium	Ultra thick	Can tolerate high forces during wire bonding.



2 layers of Aluminum Interconnect

5 layers of Copper Interconnect

Figure 61: Cross section of the SkyWater SKY90-FD metal stack, five copper layers M1 to M5 below two aluminium layers M6 and M7. Source: Google Open Source Blog, 2022

36.43 Metal routing rules on IC

Odd numbers metals => Horizontal routing (as far as possible)

Even numbers metals => Vertical routing (as far as possible)

36.44 Modeling Interconnect

Resistance narrow size impedes flow

Capacitance through under the leaky pipes

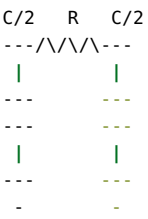
Inductance paddle wheel intertia opposes changes in flow rate

36.45 Lumped model

Use 1-segment

$$\pi$$

-model for Elmore delay



36.46 Wire resistance

$$\text{resistivity} \Rightarrow \rho \ [\Omega\text{m}]$$

$$R = \frac{\rho}{t}\frac{l}{w} = R_{\square}\frac{l}{w}$$

$$R_{\square} = \text{sheet resistance } [\Omega/\square]$$

To find resistance, count the number of squares

$$R = R_{\square} \times \#\text{ of squares}$$

36.47 Most wires: Copper

$$R_{sheet-m1} \approx \frac{1.7\mu\Omega cm}{200nm} \approx 0.1\Omega/\square$$

$$R_{sheet-m9} \approx \frac{1.7\mu\Omega cm}{3\mu m} \approx 0.006\Omega/\square$$

Pitfalls

Cu atoms diffuse into silicon and can cause damage

Must be surrounded by a diffusion barrier

Difficult high current densities (mA/

μ

m) and high temperature (125 C)

Metal	Bulk resistivity ($\mu\Omega$·cm)
Silver (Ag)	1.6
Copper (Cu)	1.7
Gold (Au)	2.2
Aluminium (Al)	2.8
Tungsten (W)	5.3
Titanium (Ti)	43.0

36.48 Contacts

Contacts and vias can have 2-20

Ω

Must use many contacts/vias for high current wires

36.49 Wire Capacitance

Dense wires has about

0.2 fF/μm

36.50 FSM

36.51 Mealy machine

An FSM where outputs depend on current state and inputs

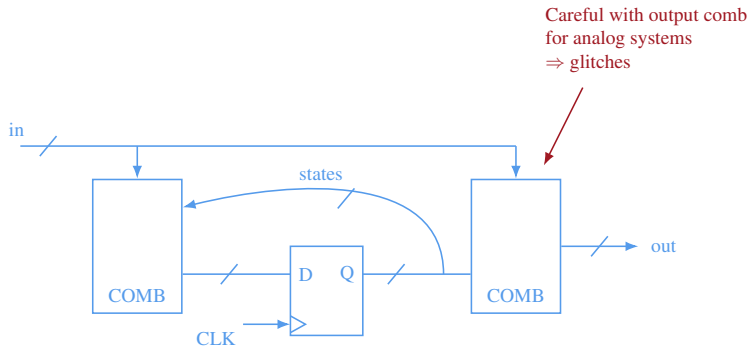


Figure 62: Mealy machine, where the input drives both the next-state logic and the output combinational block, so the output depends on input and current state

36.52 Moore machine

An FSM where outputs depend on current state

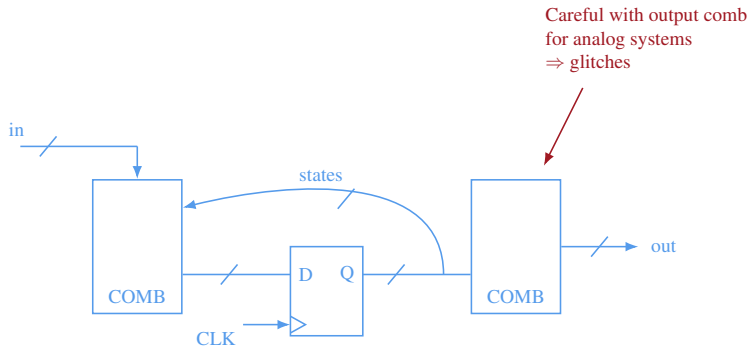


Figure 63: Moore machine, where the output combinational block is driven only by the state register

36.53 Mealy versus Moore

Parameter	Mealy	Moore
Outputs	depend on input and current state	output depend on current state
States	Same, or fewer states than Moore	
Inputs	React faster to inputs	Next clock cycle
Outputs	Can be asynchronous	Synchronous

Parameter	Mealy	Moore
States	Generally requires fewer states for synthesis	More states than Mealy
Counter	A counter is not a mealy machine	A counter is a Moore machine
Design	Can be tricky to design	Easy

36.53.1 dicex/sim/counter_sv/counter.v

```
module counter(
               output logic [WIDTH-1:0] out,
               input logic              clk,
               input logic              reset
               );
   parameter WIDTH                      = 8;
   logic [WIDTH-1:0]                    count;

   always_comb begin
      count = out + 1;
   end

   always_ff @(posedge clk or posedge reset) begin
      if (reset)
        out <= 0;
      else
        out <= count;
   end

endmodule // counter
```

36.54 Battery charger FSM

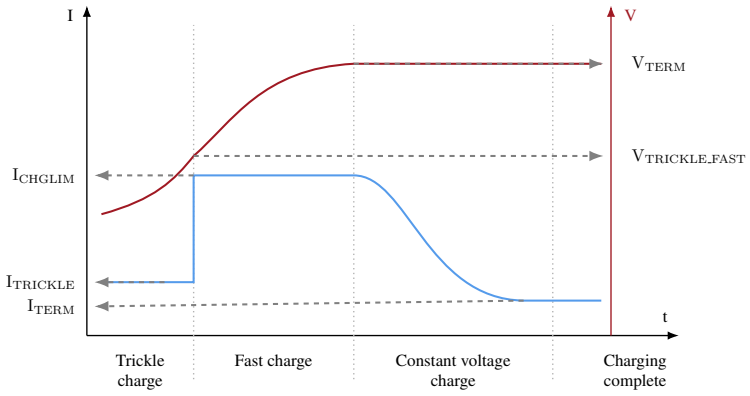


Figure 64: Li-Ion charging profile, charge current and battery voltage against time through trickle charge, fast charge, constant voltage charge and charging complete

36.54.1 Li-Ion batteries

Most Li-Ion batteries can tolerate 1 C during fast charge

For Biltema 18650 cells:

$$1\ C = 2950\ mA$$

$$0.1\ C = 295\ mA$$

Most Li-Ion need to be charged to a termination voltage of 4.2 V



Figure 65: A Biltema ICR18650 rechargeable Li-ion cell rated 2950 mAh at 3.7 V

Too high termination voltage, or too high charging current can cause growth of lithium dendrites, that short + and -. Will end in flames. Always check manufacturer datasheet for charging curves and voltages

36.54.2 Battery charger - Inputs

Voltage above

$$V_{TRICKLE}$$

Voltage close to

$$V_{TERM}$$

If voltage close to

$$V_{TERM}$$

and current is close to

$$I_{TERM}$$

, then charging complete

If charging complete, and voltage has dropped (

$$V_{RECHARGE}$$

), then start again

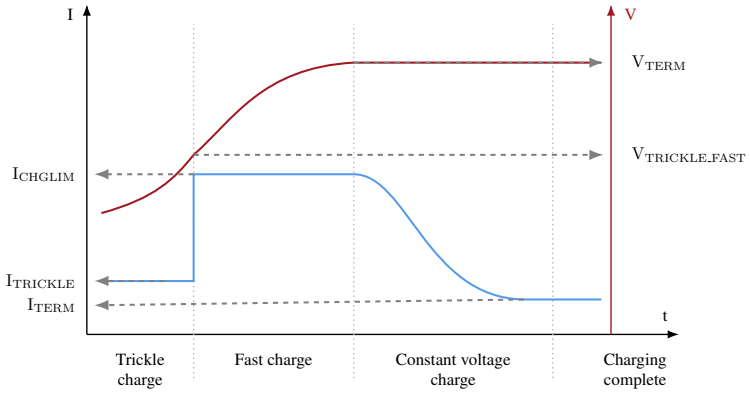


Figure 66: The charging profile marked with the thresholds the charger senses - the trickle to fast voltage, the termination voltage VTERM and the termination current ITERM

36.54.3 Battery charger - States

Trickle charge (0.1 C)

Fast charge (1 C)

Constant voltage

Charging complete

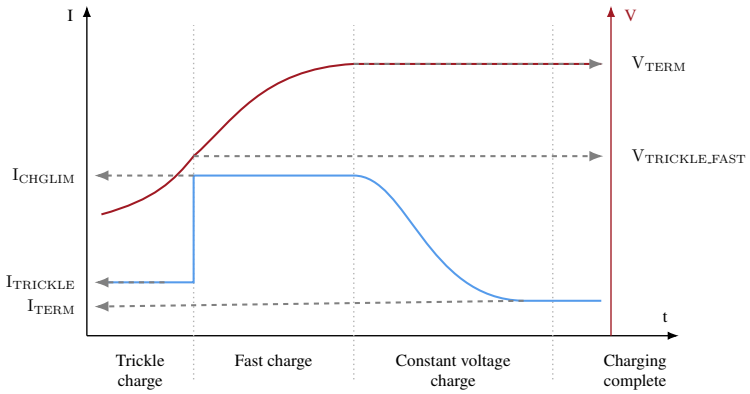


Figure 67: The charging profile divided into the four charger states - trickle charge at 0.1 C, fast charge at 1 C, constant voltage and charging complete

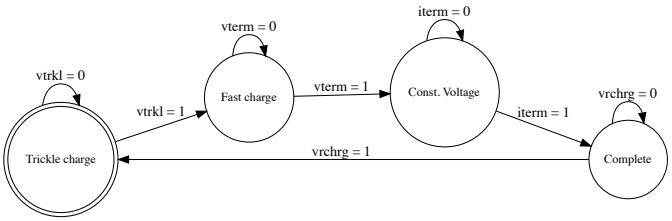


Figure 68: Battery charger state machine, cycling trickle charge, fast charge, constant voltage and complete on the vtrkl, vterm, iterm and vrchrg flags

36.54.3.1 One way to draw FSMs - Graphviz

```
digraph finite_state_machine {
    rankdir=LR;
    size="8,5"

    node [shape = doublecircle, label="Trickle charger", fontsize=12] trkl;
    node [shape = circle, label="Fast charge", fontsize=12] fast;
    node [shape = circle, label="Const. Voltage", fontsize=12] vconst;
    node [shape = circle, label="Done", fontsize=12] done;

    trkl -> trkl [label="vtrkl = 0"];
    trkl -> fast [label="vtrkl = 1"];
    fast -> fast [label="vterm = 0"];
    fast -> vconst [label="vterm = 1"];
    vconst-> vconst [label="iterm = 0"];
    vconst-> done [label="iterm = 1"];
    done-> done [label="vrchrg = 0"];
    done-> trkl [label="vrchrg = 1"];

}
```

```
dot -Tpdf bcharger.dot -o bcharger.pdf
```

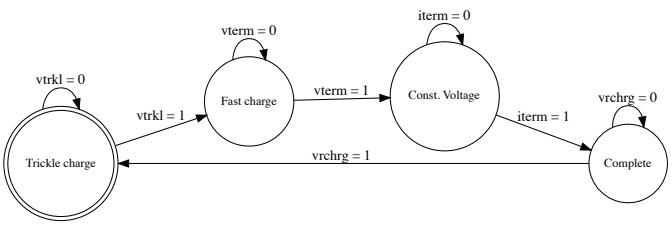


Figure 69: The battery charger state diagram beside the SystemVerilog case statement that implements the next-state logic

```
module bcharger( output logic trkl,
        output logic fast,
        output logic vconst,
        output logic done,
        input logic  vtrkl,
        input logic  vterm,
        input logic  iterm,
        input logic  vrchrg,
        input logic  clk,
        input logic  reset
                   );

   parameter TRLK = 0, FAST = 1, VCONST = 2, DONE=3;
   logic [1:0]                   state;
   logic [1:0]                   next_state;

   //- Figure out the next state
   always_comb begin
      case (state)
        TRLK: next_state = vtrkl ? FAST : TRLK;
        FAST: next_state = vterm ? VCONST : FAST;
        VCONST: next_state = iterm ? DONE : VCONST;
        DONE: next_state = vrchrg ? TRLK :DONE;
        default: next_state = TRLK;
      endcase // case (state)
    end

   //- Control output signals
   always_ff @(posedge clk or posedge reset) begin
      if(reset) begin
         state <= TRLK;
         trkl <= 1;
         fast <= 0;
         vconst <= 0;
         done <= 0;
      end
      else begin
         state <= next_state;
         case (state)
           TRLK: begin
              trkl <= 1;
              fast <= 0;
              vconst <= 0;
              done <= 0;
           end
           FAST: begin
              trkl <= 0;
              fast <= 1;
              vconst <= 0;
              done <= 0;

           end
           VCONST: begin
              trkl <= 0;
              fast <= 0;
              vconst <= 1;
              done <= 0;

           end
           DONE: begin
              trkl <= 0;
              fast <= 0;
              vconst <= 0;
              done <= 1;
```

```
          end
        endcase // case (state)
      end // else: !if(reset)
   end
endmodule
```

36.54.3.2 Synthesize FSM with yosys

dicex/sim/verilog/bcharger_sv/bcharger.ys

```
# read design
read_verilog -sv bcharger.sv;
hierarchy -top bcharger;

# the high-level stuff
fsm; opt; memory; opt;

# mapping to internal cell library
techmap; opt;
synth;
opt_clean;

# mapping flip-flops
dfflibmap  -liberty ../../../lib/SUN_TR_GF130N.lib

# mapping logic
abc -liberty ../../../lib/SUN_TR_GF130N.lib

# write synth netlist
write_verilog bcharger_netlist.v
read_verilog  ../../../lib/SUN_TR_GF130N_empty.v
write_spice -big_endian -neg AVSS -pos AVDD -top bcharger bcharger_netlist.sp

# write dot so we can make image
show -format dot -prefix bcharger_synth -colors 1 -width -stretch
clean
```

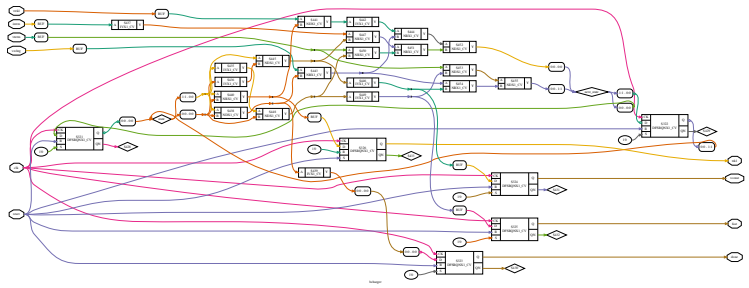


Figure 70: Gate level netlist of the battery charger after yosys synthesis to the SUN TR GF130N cell library, with six flip-flops and the next-state logic between them

36.55 Summary

The one-page version of this chapter:

- CMOS logic is a PMOS pull-up network against an NMOS pull-down: the inverter is the atom
- Delay is RC: Elmore for a hand estimate, logical effort for sizing - stage long paths at an effort around four
- Registers, combinational clouds and a clock make synchronous design; setup and hold are checked at every capture
- Static timing analysis walks every path over PVT: positive slack, or it does not ship
- Dynamic power is CV^2 f; leakage is what you pay even when nothing switches

36.56 Would you like to know more?

Weste and Harris, *CMOS VLSI Design* - the standard treatment of logic, timing and power

Sutherland, Sproull and Harris, *Logical Effort* - the short book that made stage sizing a method rather than a habit

[1] C. Wulff and T. Ytterdal, "A compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for bluetooth low energy receivers," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 7, pp. 1915–1926, 2017, doi: 10.1109/JSSC.2017.2685463.

FAQ 37

Keywords: FAQ, Git, Tools, Setup

37.1 **Frequently asked questions** **659**
37.1.1 I get asked for password when I access github 659

37.1 Frequently asked questions

37.1.1 I get asked for password when I access github

It's likely because the remote is set to https and not SSH

```
$git remote -v
origin https://github.com/analogicus/lelo_gr01_sky130a.git
```

then do

```
git remote set-url origin git@github.com:analogicus/lelo_gr01_sky130a.git
```

Or it could be because you have not setup public/private key access to github.

38 Equations

Keywords: Reference, Physics, Semiconductors, Diode, MOSFET, Noise, Circuits, References, Filters, Switched Capacitor, Data Converters, Regulators, PLL, Oscillators, Radio

Every equation of the lecture series in one place, ordered from fundamental physics to the highest abstraction. No derivations and almost no words — those live in the lectures — just the results, as a reference.

38.1 Fundamental physics 661
38.2 Semiconductors . . . 662
38.3 Diode 663
38.4 MOSFET 663
38.5 Noise 665
38.6 Circuits 666
38.7 References and bias 666
38.8 Filters 667
38.9 Switched capacitor . 667
38.10 Data converters . . . 667
38.11 Voltage regulation . 668
38.12 PLL 668
38.13 Oscillators 669
38.14 Radio 669
38.15 Would you like to know more? 669

38.1 Fundamental physics

The QED Lagrangian — everything in electronics follows from this

$$\mathcal{L} = \bar{\psi}[i\hbar c\gamma^\mu\partial_\mu - mc^2]\psi - q[\bar{\psi}\gamma^\mu\psi]A_\mu - \frac{1}{16\pi}F_{\mu\nu}F^{\mu\nu}$$

Schrödinger equation

$$i\hbar\frac{d}{dt}\psi(r,t) = \widehat{H}\psi(r,t)$$

Probability density of a particle

$$P = |\psi(r,t)|^2\,,\,\psi(r,t) = Ae^{i(kr-\omega t)}$$

Heisenberg uncertainty

$$\sigma_x\sigma_p \geq \frac{\hbar}{2}\,,\,\Delta E\Delta t > \frac{h}{2\pi}$$

Fermi-Dirac distribution, and its Boltzmann tail

$$f(E) = \frac{1}{e^{(E-E_F)/kT}+1} \approx e^{(E_F-E)/kT}$$

Maxwell's equations

$$\oint_{\partial\Omega}\mathbf{E}\cdot d\mathbf{S} = \frac{1}{\epsilon_0}\iiint_V \rho\cdot dV\,,\,\oint_{\partial\Omega}\mathbf{B}\cdot d\mathbf{S} = 0$$

$$\oint_{\partial\Sigma} \mathbf{E} \cdot d\ell = -\frac{d}{dt} \iint_{\Sigma} \mathbf{B} \cdot d\mathbf{S}$$

$$\oint_{\partial\Sigma} \mathbf{B} \cdot d\ell = \mu_0 \left(\iint_{\Sigma} \mathbf{J} \cdot d\mathbf{S} + \epsilon_0 \frac{d}{dt} \iint_{\Sigma} \mathbf{E} \cdot d\mathbf{S} \right)$$

Force on a charge

$$\vec{F} = q\vec{E}$$

38.2 Semiconductors

Density of electrons in the conduction band

$$n = \int_{E_C}^{\infty} N(E) f(E) dE$$

Effective density of states

$$N_c = 2 \left[\frac{2\pi k T m_n^*}{h^2} \right]^{3/2} , N_v = 2 \left[\frac{2\pi k T m_p^*}{h^2} \right]^{3/2}$$

Intrinsic carrier concentration

$$n_i = \sqrt{N_c N_v}\, e^{-E_g/(2kT)}$$

Doped silicon (mass action)

$$n_n = N_D \, , \, p_n = \frac{n_i^2}{N_D} \, ; \, p_p = N_A \, , \, n_p = \frac{n_i^2}{N_A}$$

Drift current

$$\vec{J} = qn\mu\vec{E}$$

Diffusion current

$$J = -qD_n \frac{d\rho}{dx}$$

Thermal voltage

$$V_T = \frac{kT}{q} \approx 25.85 \text{ mV @ } 300 \text{ K}$$

38.3 Diode

Built-in voltage of a pn junction

$$\Phi_0 = V_T \ln\left(\frac{N_A N_D}{n_i^2}\right)$$

Depletion width

$$W = \sqrt{\frac{2\varepsilon_{si}(\Phi_0 + V_R)}{q} \cdot \frac{N_A + N_D}{N_A N_D}}$$

Diode equation

$$I_D = I_S\left(e^{V_D/V_T} - 1\right),\ I_S = qAn_i^2\left(\frac{D_n}{L_n N_A} + \frac{D_p}{L_p N_D}\right)$$

Forward voltage temperature dependence

$$V_D = \frac{kT}{q}(\ell - 3\ln T) + V_G \Rightarrow \frac{dV_D}{dT} = \frac{k}{q}(\ell - 3\ln T - 3)$$

Difference of two diode voltages (PTAT)

$$V_{D1} - V_{D2} = V_T \ln N$$

Generation (leakage) current of a reverse biased junction

$$I_{gen} = \frac{qAn_i W}{\tau_g}$$

38.4 MOSFET

Weak inversion

$$I_D = I_{D0}\frac{W}{L}e^{V_{eff}/nV_T} \text{ if } V_{DS} > 3V_T$$

$$n = \frac{C_{ox} + C_{j0}}{C_{ox}},\ I_{D0} = (n-1)\mu_n C_{ox} V_T^2$$

Strong inversion, define

$$V_{eff} = V_{GS} - V_{tn} \, , \ell = \mu_n C_{ox} \frac{W}{L}$$

$$I_{DS} = \ell \begin{cases} V_{eff} V_{DS} & \text{if } V_{DS} << V_{eff} \\ V_{eff} V_{DS} - V_{DS}^2/2 & \text{if } V_{DS} < V_{eff} \\ \frac{1}{2} V_{eff}^2 \left[1 + \lambda (V_{DS} - V_{eff})\right] & \text{if } V_{DS} > V_{eff} \end{cases}$$

Transconductance

$$g_m = \frac{\partial I_{DS}}{\partial V_{GS}} = \ell V_{eff} = \sqrt{2\ell I_D} = \frac{2I_D}{V_{eff}} \text{ (strong)} \, , g_m = \frac{I_D}{nV_T} \text{ (weak)}$$

Transconductance efficiency

$$\frac{g_m}{I_D} = \frac{1}{nV_T} \text{ (weak)} \, , \frac{g_m}{I_D} = \frac{2}{V_{eff}} \text{ (strong)}$$

Output conductance and intrinsic gain

$$g_{ds} = \frac{1}{r_{ds}} \approx \lambda I_D \, , A = g_m r_{ds} = \frac{2}{\lambda V_{eff}}$$

Capacitances

$$C_{gs} = \frac{2}{3} WLC_{ox} \text{ (saturation)} \, , C_{gd} = C_{ox} W L_{ov}$$

$$C_{sb} = (A_s + A_{ch}) C_{js} \, , C_{js} = \frac{C_{j0}}{\sqrt{1 + \frac{V_{SB}}{\Phi_0}}}$$

Miller's theorem

$$C_{in} = (1 + A)C \, , C_{out} = \left(1 + \frac{1}{A}\right) C \Rightarrow C_{in} \approx C_{gd} \, g_m r_{ds}$$

Matching (Pelgrom)

$$\sigma^2(\Delta P) = \frac{A_P^2}{WL} + S_P^2 D^2$$

Current mismatch (Kinget)

$$\frac{\sigma_{I_D}^2}{I_D^2} = \frac{1}{WL}\left[\left(\frac{g_m}{I_D}\right)^2 \sigma_{vt}^2 + \frac{\sigma_\ell^2}{\ell^2}\right], \sigma_{v_i}^2 = \frac{\sigma_{I_D}^2}{g_m^2}$$

38.5 Noise

Mean square and power spectral density

$$\overline{x^2(t)} = \int_0^\infty S_x(f)df \, , S_y(f) = S_x(f)|H(f)|^2$$

Thermal noise of a resistor

$$S_{th}(f) = 4kTR$$

Sampled (kT/C) noise bandwidth of an RC

$$f_x = \frac{\pi f_0}{2} = \frac{1}{4RC}$$

Uncorrelated sources add in power

$$\overline{e_{tot}^2} = \overline{e_1^2} + \overline{e_2^2}$$

Signal-to-noise ratio

$$SNR = 10\log\left(\frac{\overline{v_{sig}^2}}{\overline{e_n^2}}\right)$$

Noise factor and Friis' formula

$$F = \frac{SNR_{input}}{SNR_{output}} \, , NF = 10\log(F)$$

$$F = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \cdots$$

38.6 Circuits

Common source

$$A = -g_m r_{ds} \text{ , } r_{out} = r_{ds}$$

Common drain (source follower)

$$A = \frac{g_m}{g_m + g_{ds} + g_s} \text{ , } r_{out} \approx \frac{1}{g_m}$$

Common gate

$$A = 1 + g_m r_{ds} \text{ , } r_{in} \approx \frac{1}{g_m}\left(1 + \frac{R_L}{r_{ds}}\right)$$

Source degeneration and cascode output resistance

$$r_{out} = r_{ds2}\left[1 + R_s(g_{m2} + g_{ds2})\right] \approx r_{ds2}\, g_{m2} R_s$$

38.7 References and bias

Bipolar / diode core

$$V_{BE} = V_T \ln \frac{I_C}{I_S} \text{ (CTAT) , } \Delta V_{BE} = V_T \ln N \text{ (PTAT)}$$

Brokaw bandgap output

$$V_{REF} = V_{BE3} + \frac{R_2}{R_3} V_T \ln \frac{R_2}{R_1}$$

Bandgap voltage with curvature

$$V_{BG} = V_{G0} + (m-1)\frac{kT}{q} \ln \frac{T_0}{T} + T\left[\frac{k}{q} \ln \frac{J_2}{J_1} \frac{2R_2}{R_1} - \frac{V_{G0} - V_{be0}}{T_0}\right]$$

38.8 Filters

Pole/zero frequency

$$\omega_{p|z} \propto \frac{1}{RC} \text{ (Active-RC)} , \omega_{p|z} \propto \frac{G_m}{C} \text{ (Gm-C)}$$

General biquad

$$H(s) = \frac{\frac{C_1}{C_B}s^2 + \frac{G_2}{C_B}s + \frac{G_1 G_3}{C_A C_B}}{s^2 + \frac{G_5}{C_B}s + \frac{G_3 G_4}{C_A C_B}}$$

38.9 Switched capacitor

SC resistance

$$Z_I = \frac{1}{C_1 f_\phi}$$

SC gain stage and integrator

$$H(z) = \frac{C_1}{C_2}z^{-1} , H(z) = \frac{C_1}{C_2}\frac{z^{-1}}{1 - z^{-1}}$$

First and second order IIR

$$H(z) = \frac{b}{z - a} , H(z) = \frac{bz}{z^2 - 2az + (a^2 + b^2)}$$

38.10 Data converters

Quantization noise

$$\overline{e_n^2} = \frac{\Delta^2}{12} \Rightarrow SQNR \approx 6.02B + 1.76 \text{ dB}$$

Oversampling

$$SQNR \approx 6.02B + 1.76 + 10\log(OSR)$$

First order noise shaping

$$Y(z) = STF(z)U(z) + NTF(z)E(z) , STF = z^{-1} , NTF = 1 - z^{-1}$$

$$SQNR = 6.02B + 1.76 - 5.17 + 30\log(OSR)$$

Figures of merit

$$FOM_W = \frac{P}{2^B f_s}\ , FOM_S = SNDR + 10\log\left(\frac{f_s/2}{P}\right)$$

DAC: a digital number scales a reference

$$V_{out} = D_{in} \times V_{ref}$$

38.11 Voltage regulation

The inductor and capacitor of a switcher

$$I_x(t) = \frac{1}{L}\int V_x(t)dt\ , V_o(t) = \frac{1}{C}\int (I_x(t) - I_o(t))dt$$

Ideal buck output

$$V_o = V_{in} \times \text{Duty-Cycle}$$

38.12 PLL

A modulated carrier, and phase versus frequency

$$A_m(t)\cos\left(2\pi f_c t + \phi_m(t)\right)\ , \phi(t) = 2\pi\int_0^t f(t)dt$$

Loop gain of a charge-pump PLL

$$L(s) = \frac{K_{osc}K_{pd}K_{lp}H_{lp}(s)}{Ns}$$

$$K_{osc} = 2\pi\frac{df}{dV_{cntl}}\ , K_{pd} = \frac{I_{cp}}{2\pi}\ , K_{lp}H_{lp}(s) = \frac{1}{s(C_1+C_2)}\frac{1+sRC_1}{1+sR\frac{C_1C_2}{C_1+C_2}}$$

38.13 Oscillators

Crystal input impedance

$$Z_{in} \approx \frac{LC_F s^2 + 1}{LC_F C_P s^2 + C_F + C_P}$$

Ring oscillator frequency

$$f = \frac{1}{2Nt_{pd}}\,,\, t_{pd} \approx RC \Rightarrow f = \frac{\mu_n(VDD - V_{th})}{\frac{4}{3}NL^2}$$

Current starved ring

$$f \approx \frac{I_{control}}{C\frac{VDD}{2}N}$$

38.14 Radio

Friis transmission (free space)

$$P_{RX} = \frac{P_{TX}}{D^2}\left[\frac{\lambda}{4\pi}\right]^2$$

Receiver sensitivity

$$P_{RX_{sens}} = -174\ \text{dBm} + 10\log_{10}(DR) + NF + E_b/N_0$$

38.15 Would you like to know more?

Every equation here is derived in its own chapter of this book; the section titles above point at them

Johns and Martin carry the same set with more algebra [1]

[1] D. Johns and K. Martin, *Analog integrated circuit design*. John Wiley & Sons, Inc., 1997.

References 39

[1] S. H. Lewis and P. R. Gray, "A pipelined 5-Msample/s 9-bit analog-to-digital converter," *ieee_j_jssc*, vol. 22, no. 6, 1987.

[2] B. Nauta, "A CMOS transconductance-c filter technique for very high frequencies," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 2, pp. 142–153, 1992, doi: 10.1109/4.127337.

[3] T. C. Carusone, D. Johns, and K. Martin, *Analog integrated circuit design*. Wiley, 2011 [Online]. Available: https://books.google.no/books?id=1OIJZzLvVhcC

[4] Berkeley, "Berkeley short-channel IGFET model." [Online]. Available: http://bsim.berkeley.edu/models/bsim4/

[5] C. C. Liu, "A 0.35mW 12b 100MS/s SAR-assisted digital slope ADC in 28nm CMOS," in *IEEE ISSCC dig. Tech. papers*, 2016, pp. 462–463, doi: 10.1109/ISSCC.2016.7418107.

[6] S. Patil, A. Ratiu, D. Morche, and Y. Tsividis, "A 3-10 fJ/conv-step Error-Shaping Alias-Free Continuous-Time ADC," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 908–918, 2016, doi: 10.1109/JSSC.2016.2519396.

[7] C. C. Liu, S. J. Chang, G. Y. Huang, and Y. Z. Lin, "A 10-bit 50-MS/s SAR ADC With a Monotonic Capacitor Switching Procedure," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 731–740, 2010, doi: 10.1109/JSSC.2010.2042254.

[8] L. L. Lewyn, T. Ytterdal, C. Wulff, and K. Martin, "Analog Circuit Design in Nanoscale CMOS Technologies," *Proc. IEEE*, vol. 97, no. 10, pp. 1687–1714, Oct. 2009, doi: 10.1109/JPROC.2009.2024663.

[9] S. Weaver, B. Hershberg, and U. K. Moon, "Digitally Synthesized Stochastic Flash ADC Using Only Standard Digital Cells," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 1, pp. 84–91, Jan. 2014, doi: 10.1109/TCSI.2013.2268571.

[10] P. Harpe, G. Dolmans, K. Philips, and H. de Groot, "A 0.7V 7-to-10bit 0-to-2MS/s flexible SAR ADC for ultra low-power wireless sensor nodes," in *Proc. 38th Eur. Solid-State Circuits Conf. (ESSCIRC)*, 2012, pp. 373–376, doi: 10.1109/ESSCIRC.2012.6341363.

[11] B. Murmann, "ADC Performance Survey 1997-2016," 2016. [Online]. Available: http://web.stanford.edu/~murmann/adcsurvey.html. [Accessed: Jul. 12, 2016]

[12] C. Wulff and T. Ytterdal, "A compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for bluetooth low energy receivers," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 7, pp. 1915–1926, 2017, doi: 10.1109/JSSC.2017.2685463.

[13] H. Garvik, C. Wulff, and T. Ytterdal, "A 68 dB SNDR compiled noise-shaping SAR ADC with on-chip CDAC calibration," in *2019 IEEE asian solid-state circuits conference (a-SSCC)*, 2019, pp. 193–194, doi: 10.1109/A-SSCC47793.2019.9056925.

[14] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching properties of MOS transistors," *ieee_j_jssc*, vol. 23, no. 5, pp. 1433–1439, 1989.

[15] A. Einstein, "Method for the determinination of the statistical values of observations concerning quantities subject to irregular fluctuations," *ieee_m_assp*, vol. October, 1987.

[16] R. M. Gray and L. D. Davisson, *An introduction to statistical signal processing*. Cambridge University Press, 2004.

[17] D. Johns and K. Martin, *Analog integrated circuit design*. John Wiley & Sons, Inc., 1997.

[18] B. Razavi, *Design of analog CMOS integrated circuits*. McGraw-Hill, 2001.

[19] R. Schreier and G. C. Temes, *Understanding delta-sigma data converters*. John Wiley & Sons, 2005.

[20] R. H. Walden, "Analog to digital converter survey and analysis," *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 4, pp. 539–550, 1999, doi: 10.1109/49.761034.

[21] A. V. D. Ziel, *Noise in solid state devices and circuits*. John Wiley & Sons Inc, 1986.

[22] Z. Tang, Y. Fang, Z. Shi, X.-P. Yu, N. N. Tan, and W. Pan, "A 1770- μ m2 leakage-based digital temperature sensor with supply sensitivity suppression in 55-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 3, pp. 781–793, 2020, doi: 10.1109/JSSC.2019.2952855.

[23] S. Jeong, Z. Foo, Y. Lee, J.-Y. Sim, D. Blaauw, and D. Sylvester, "A fully-integrated 71 nW CMOS temperature sensor for low power wireless sensor nodes," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 8, pp. 1682–1693, 2014, doi: 10.1109/JSSC.2014.2325574.

[24] M. A. P. Pertijs, A. Niederkorn, X. Ma, B. McKillop, A. Bakker, and J. H. Huijsing, "A CMOS smart temperature sensor with a 3/spl sigma/ inaccuracy of /spl plusmn/0.5/spl deg/c from -50/spl deg/c to 120/spl deg/c," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 2, pp. 454–461, 2005, doi: 10.1109/JSSC.2004.841013.

[25] J.-H. Park, J.-H. Hwang, C. Shin, and S.-J. Kim, "A BJT-based temperature-to-frequency converter with +- 1°c (3σ) inaccuracy from - 40 c to 140 c for on-chip thermal monitoring," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 10, pp. 2909–2918, 2022, doi: 10.1109/JSSC.2022.3182708.

[26] H. T. Friis, "Noise figures of radio receivers," *Proceedings of the IRE*, vol. 32, no. 7, pp. 419–422, 1944, doi: 10.1109/JRPROC.1944.232049.

[27] K. P. Cheung, *Plasma charging damage*. Springer-Verlag London, 2001 [Online]. Available: https://link.springer.com/book/10.1007/978-1-4471-0247-2

[28] K. Hashimoto, "Charge damage caused by electron shading effect," *Japanese Journal of Applied Physics*, vol. 33, pp. 6013–6018, 1994, doi: 10.1143/JJAP.33.6013. [Online]. Available: https://iopscience.iop.org/article/10.1143/JJAP.33.6013

[29] S. Krishnan, A. Amerasekera, S. Rangan, and S. Aur, "Antenna device reliability for ULSI processing," in *International electron devices meeting (IEDM) technical digest*, 1998, pp. 247–250, doi: 10.1109/IEDM.1998.746430.

[30] Z. Huang, Z. Tang, X.-P. Yu, Z. Shi, L. Lin, and N. N. Tan, "A BJT-based CMOS temperature sensor with duty-cycle-modulated output and ±0.5°c (3σ) inaccuracy from −40 °c to 125 °c," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 8, pp. 2780–2784, 2021, doi: 10.1109/TCSII.2021.3068283.

[31] M.-D. Ker, W.-Y. Chen, W.-T. Shieh, and I.-J. Wei, "New ballasting layout schemes to improve ESD robustness of i/o buffers in fully silicided CMOS process," *IEEE Transactions on Electron Devices*, vol. 56, no. 12, pp. 3149–3159, 2009, doi: 10.1109/TED.2009.2031003.

[32] M.-D. Ker, "ESD (electrostatic discharge) protection design for nanoelectronics in CMOS technology," in *2006 advanced signal processing, circuit and system design techniques for communications*, 2006, pp. 217–279, doi: 10.1109/ASPCAS.2006.251127.

[33] M.-D. Ker and Z.-H. Jiang, "Overview on latch-up prevention in CMOS integrated circuits by circuit solutions," *IEEE Journal of the Electron Devices Society*, vol. 11, pp. 141–152, 2023, doi: 10.1109/JEDS.2022.3231822.

[34] M.-D. Ker, C.-Y. Lin, and Y.-W. Hsiao, "Overview on ESD protection designs of low-parasitic capacitance for RF ICs in CMOS technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 11, no. 2, pp. 207–218, 2011, doi: 10.1109/TDMR.2011.2106129.

[35] R. Widlar, "New developments in IC voltage regulators," *IEEE Journal of Solid-State Circuits*, vol. 6, no. 1, pp. 2–7, 1971, doi: 10.1109/JSSC.1971.1050151.

[36] A. Brokaw, "A simple three-terminal IC bandgap reference," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 6, pp. 388–393, 1974, doi: 10.1109/JSSC.1974.1050532.

[37] H. Banba *et al.*, "A CMOS bandgap reference circuit with sub-1-v operation," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 5, pp. 670–674, 1999, doi: 10.1109/4.760378.

[38] K. N. Leung and P. Mok, "A sub-1-v 15-ppm//spl deg/c CMOS bandgap voltage reference without requiring low threshold voltage device," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 4, pp. 526–530, 2002, doi: 10.1109/4.991391.

[39] B. Razavi, "The design of a low-voltage bandgap reference [the analog mind]," *IEEE Solid-State Circuits Magazine*, vol. 13, no. 3, pp. 6–16, 2021, doi: 10.1109/MSSC.2021.3088963.

[40] H. Li, Y. Shen, E. Cantatore, and P. Harpe, "A 77.3-dB SNDR 62.5-kHz bandwidth continuous-time noise-shaping SAR ADC with duty-cycled gm-c integrator," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 4, pp. 939–948, 2023, doi: 10.1109/JSSC.2022.3227678.

[41] L. J. Breems, R. Rutten, R. H. M. van Veldhoven, and G. van der Weide, "A 56 mW continuous-time quadrature cascaded ΣΔ modulator with 77 dB DR in a near zero-IF 20 MHz band," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 12, pp. 2696–2705, 2007, doi: 10.1109/JSSC.2007.908765.

[42] K. Martin, "Complex signal processing is not - complex," in *ESSCIRC 2004 - 29th european solid-state circuits conference (IEEE cat. no.03EX705)*, 2003, pp. 3–14, doi: 10.1109/ESSCIRC.2003.1257061.

[43] W. Wu *et al.*, "A 28-nm 75-fsrms analog fractional- N sampling PLL with a highly linear DTC incorporating background DTC gain calibration and reference clock duty cycle correction," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 5, pp. 1254–1265, 2019, doi: 10.1109/JSSC.2019.2899726.

[44] M. van Elzakker, E. van Tuijl, P. Geraedts, D. Schinkel, E. A. M. Klumperink, and B. Nauta, "A 10-bit charge-redistribution ADC consuming 1.9 μw at 1 MS/s," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 5, pp. 1007–1015, 2010, doi: 10.1109/JSSC.2010.2043893.

[45] Y. Chae, K. Souri, and K. A. A. Makinwa, "A 6.3 μw 20 bit incremental zoom-ADC with 6 ppm INL and 1 μv offset," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 12, pp. 3019–3027, 2013, doi: 10.1109/JSSC.2013.2278737.

[46] W.-H. Tseng, C.-W. Fan, and J.-T. Wu, "A 12-bit 1.25-GS/s DAC in 90 nm CMOS with >70 dB SFDR up to 500 MHz," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 12, pp. 2845–2856, 2011, doi: 10.1109/JSSC.2011.2164302.

[47] M. Mishali and Y. C. Eldar, "Blind multiband signal reconstruction: Compressed sensing for analog signals," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 993–1009, 2009, doi: 10.1109/TSP.2009.2012791.

[48] C. Wulff and T. Ytterdal, "Comparator-based switched-capacitor pipelined analog-to-digital converter with comparator preset, and comparator delay compensation," *Analog Integrated Circuits and Signal Processing*, vol. 67, no. 1, pp. 31–40, 2010, doi: 10.1007/s10470-010-9576-3.

[49] Y. Chae and G. Han, "Low voltage, low power, inverter-based switched-capacitor delta-sigma modulator," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 2, pp. 458–472, 2009, doi: 10.1109/JSSC.2008.2010973.

[50] B. Hershberg, S. Weaver, K. Sobue, S. Takeuchi, K. Hamashita, and U.-K. Moon, "Ring amplifiers for switched capacitor circuits," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 12, pp. 2928–2942, 2012, doi: 10.1109/JSSC.2012.2217865.

[51] S.-M. Yoo, J. S. Walling, E. C. Woo, B. Jann, and D. J. Allstot, "A switched-capacitor RF power amplifier," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 12, pp. 2977–2987, 2011, doi: 10.1109/JSSC.2011.2163469.

[52] M. Darvishi, R. van der Zee, and B. Nauta, "Design of active n-path filters," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 12, pp. 2962–2976, 2013, doi: 10.1109/JSSC.2013.2285852.

[53] N. Blachman, "The intermodulation and distortion due to quantization of sinusoids," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 6, pp. 1417–1426, 1985, doi: 10.1109/TASSP.1985.1164729.

[54] C. Wulff and T. Ytterdal, "Resonators in open-loop sigma–delta modulators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 10, pp. 2159–2172, 2009, doi: 10.1109/TCSI.2009.2015211.

[55] B. Boser and B. Wooley, "The design of sigma-delta modulation analog-to-digital converters," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 6, pp. 1298–1308, 1988, doi: 10.1109/4.90025.

[56] T. Riley, M. Copeland, and T. Kwasniewski, "Delta-sigma modulation in fractional-n frequency synthesis," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 5, pp. 553–559, 1993, doi: 10.1109/4.229400.

[57] K. Souri, Y. Chae, and K. A. A. Makinwa, "A CMOS temperature sensor with a voltage-calibrated inaccuracy of ± 0.15° c (3σ) from − 55° c to 125° c," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 292–301, 2013, doi: 10.1109/JSSC.2012.2214831.

[58] G. Mitteregger, C. Ebner, S. Mechnig, T. Blon, C. Holuigue, and E. Romani, "A 20-mW 640-MHz CMOS continuous-time ΣΔ ADC with 20-MHz signal bandwidth, 80-dB dynamic range and 12-bit ENOB," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 12, pp. 2641–2649, 2006, doi: 10.1109/JSSC.2006.884332.

[59] C.-H. Chen, Y. Zhang, T. He, P. Y. Chiang, and G. C. Temes, "A micro-power two-step incremental analog-to-digital converter," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 8, pp. 1796–1808, 2015, doi: 10.1109/JSSC.2015.2413842.

[60] T. Grasser *et al.*, "Simultaneous extraction of recoverable and permanent components contributing to bias-temperature instability," in *2007 IEEE international electron devices meeting*, 2007, pp. 801–804, doi: 10.1109/IEDM.2007.4419069.

[61] S. J. Kim, W.-S. Choi, R. Pilawa-Podgurski, and P. K. Hanumolu, "A 10-MHz 2–800-mA 0.5–1.5-v 90," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 3, pp. 814–824, 2018, doi: 10.1109/JSSC.2017.2776298.

[62] X. Mao, Y. Lu, and R. P. Martins, "A scalable high-current high-accuracy dual-loop four-phase switching LDO for microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 6, pp. 1841–1853, 2022, doi: 10.1109/JSSC.2021.3129620.

[63] T. Y. Man, K. N. Leung, C. Y. Leung, P. K. T. Mok, and M. Chan, "Development of single-transistor-control LDO based on flipped voltage follower for SoC," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 55, no. 5, pp. 1392–1401, 2008, doi: 10.1109/TCSI.2008.916568.

[64] Y.-J. Lee *et al.*, "A 200-mA digital low drop-out regulator with coarse-fine dual loop in mobile application processor," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 64–76, 2017, doi: 10.1109/JSSC.2016.2614308.

[65] H.-P. Le, S. R. Sanders, and E. Alon, "Design techniques for fully integrated switched-capacitor DC-DC converters," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 9, pp. 2120–2131, 2011, doi: 10.1109/JSSC.2011.2159054.

[66] S. J. Kim *et al.*, "High frequency buck converter design using time-based control techniques," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 4, pp. 990–1001, 2015, doi: 10.1109/JSSC.2014.2378216.

[67] M.-H. Huang and K.-H. Chen, "Single-inductor multi-output (SIMO) DC-DC converters with high light-load efficiency and minimized cross-regulation for portable devices," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 4, pp. 1099–1111, 2009, doi: 10.1109/JSSC.2009.2014726.

[68] C. F. Lee and P. Mok, "A monolithic current-mode CMOS DC-DC converter with on-chip current-sensing technique," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 1, pp. 3–14, 2004, doi: 10.1109/JSSC.2003.820870.

[69] X. Gao, E. A. M. Klumperink, M. Bohsali, and B. Nauta, "A low noise sub-sampling PLL in which divider noise is eliminated and PD/CP noise is not multiplied by N^2," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 12, pp. 3253–3263, 2009, doi: 10.1109/JSSC.2009.2032723.

[70] R. Staszewski *et al.*, "All-digital PLL and transmitter for mobile phones," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 12, pp. 2469–2482, 2005, doi: 10.1109/JSSC.2005.857417.

[71] D. Tasca, M. Zanuso, G. Marzin, S. Levantino, C. Samori, and A. L. Lacaita, "A 2.9–4.0-GHz fractional-n digital PLL with bang-bang phase detector and 560-fs_{rms} integrated jitter at 4.5-mW power," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 12, pp. 2745–2758, 2011, doi: 10.1109/JSSC.2011.2162917.

[72] B. Razavi, "The crystal oscillator [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 9, no. 2, pp. 7–9, 2017, doi: 10.1109/MSSC.2017.2688679.

[73] E. Vittoz, M. Degrauwe, and S. Bitz, "High-performance crystal oscillator circuits: Theory and application," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 3, pp. 774–783, 1988, doi: 10.1109/4.318.

[74] L. Xu, D. Blaauw, and D. Sylvester, "Ultra-low power 32kHz crystal oscillators: Fundamentals and design techniques," *IEEE Open Journal of the Solid-State Circuits Society*, vol. 1, pp. 79–93, 2021, doi: 10.1109/OJSSCS.2021.3113889.

[75] K.-M. Kim, S. Kim, K.-S. Choi, H. Jung, J. Ko, and S.-G. Lee, "A sub-nW single-supply 32-kHz sub-harmonic pulse injection crystal oscillator," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 6, pp. 1849–1858, 2021, doi: 10.1109/JSSC.2020.3016021.

[76] B. Razavi, "The ring oscillator [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 4, pp. 10–81, 2019, doi: 10.1109/MSSC.2019.2939771.

[77] B. Razavi, "A study of phase noise in CMOS oscillators," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 3, pp. 331–343, 1996, doi: 10.1109/4.494195.

[78] J. Lee, A. K. George, and M. Je, "An ultra-low-noise swing-boosted differential relaxation oscillator in 0.18-μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 9, pp. 2489–2497, 2020, doi: 10.1109/JSSC.2020.2987681.

[79] M. Tamura *et al.*, "30.5 a 0.5V BLE transceiver with a 1.9mW RX achieving −96.4dBm sensitivity and 4.1dB adjacent channel rejection at 1MHz offset in 22nm FDSOI," in *2020 IEEE international solid-state circuits conference - (ISSCC)*, 2020, pp. 468–470, doi: 10.1109/ISSCC19947.2020.9063021.

[80] K. Martin, "Complex signal processing is not complex," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 9, pp. 1823–1836, 2004, doi: 10.1109/TCSI.2004.834522.

[81] B. J. Thijssen, E. A. M. Klumperink, P. Quinlan, and B. Nauta, "30.4 a 370μw 5.5dB-NF BLE/BT5.0/IEEE 802.15.4-compliant receiver with >63dB adjacent channel rejection at >2 channels offset in 22nm FDSOI," in *2020 IEEE international solid-state circuits conference - (ISSCC)*, 2020, pp. 466–468, doi: 10.1109/ISSCC19947.2020.9062973.

[82] B. Razavi, *Design of CMOS phase-locked loops*. Cambridge University Press, 2020.

[83] C. Enz, F. Chicco, and A. Pezzotta, "Nanoscale MOSFET modeling: Part 1: The simplified EKV model for the design of low-power analog circuits," *IEEE Solid-State Circuits Magazine*, vol. 9, no. 3, pp. 26–35, 2017, doi: 10.1109/MSSC.2017.2712318.

[84] C. Enz, F. Chicco, and A. Pezzotta, "Nanoscale MOSFET modeling: Part 2: Using the inversion coefficient as the primary design parameter," *IEEE Solid-State Circuits Magazine*, vol. 9, no. 4, pp. 73–81, 2017, doi: 10.1109/MSSC.2017.2745838.

[85] H. Pretl and M. Eberlein, "Fifty nifty variations of two-transistor circuits: A tribute to the versatility of MOSFETs," *IEEE Solid-State Circuits Magazine*, vol. 13, no. 3, pp. 38–46, 2021, doi: 10.1109/MSSC.2021.3088968.

[86] B. Hernes *et al.*, "A 92.5mW 205MS/s 10b pipeline IF ADC implemented in 1.2V/3.3V 0.13μm CMOS," in *2007 IEEE international solid-state circuits conference. Digest of technical papers*, 2007, pp. 462–615, doi: 10.1109/ISSCC.2007.373494.

[87] M. Shirvanimoghaddam *et al.*, "Towards a green and self-powered internet of things using piezoelectric energy harvesting," *IEEE Access*, vol. 7, pp. 94533–94556, 2019, doi: 10.1109/ACCESS.2019.2928523.

[88] S. Bose, T. Anand, and M. L. Johnston, "A 3.5-mV input single-inductor self-starting boost converter with loss-aware MPPT for efficient autonomous body-heat energy harvesting," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 6, pp. 1837–1848, 2021, doi: 10.1109/JSSC.2020.3042962.

[89] S. Du, Y. Jia, C. Zhao, G. A. J. Amaratunga, and A. A. Seshia, "A fully integrated split-electrode SSHC rectifier for piezoelectric energy harvesting," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 6, pp. 1733–1743, 2019, doi: 10.1109/JSSC.2019.2893525.

[90] T. Hu, H. Wang, W. Harmon, D. Bamgboje, and Z.-L. Wang, "Current progress on power management systems for triboelectric nanogenerators," *IEEE Transactions on Power Electronics*, vol. 37, no. 8, pp. 9850–9864, 2022, doi: 10.1109/T-PEL.2022.3156871.

[91] J. S. Y. Tan *et al.*, "A fully energy-autonomous temperature-to-time converter powered by a triboelectric energy harvester for biomedical applications," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 10, pp. 2913–2923, 2021, doi: 10.1109/JSSC.2021.3080383.

[92] H.-C. Cheng, P.-H. Chen, Y.-T. Su, and P.-H. Chen, "A reconfigurable capacitive power converter with capacitance redistribution for indoor light-powered batteryless internet-of-things devices," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 10, pp. 2934–2942, 2021, doi: 10.1109/JSSC.2021.3075217.

[93] T. Masuhara and R. Muller, "Complementary DMOS process for LSI," *IEEE Journal of Solid-State Circuits*, vol. 11, no. 4, pp. 453–458, 1976, doi: 10.1109/JSSC.1976.1050758.

[94] C. Gear, "Simultaneous numerical solution of differential-algebraic equations," *IEEE Transactions on Circuit Theory*, vol. 18, no. 1, pp. 89–95, 1971, doi: 10.1109/TCT.1971.1083221.

[95] B. Razavi, "The bandgap reference [a circuit for all seasons]," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 3, pp. 9–12, 2016, doi: 10.1109/MSSC.2016.2577978.

[96] H.-S.-P. Wong *et al.*, "Metal–oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012, doi: 10.1109/JPROC.2012.2190369.

[97] S.-E. Hsieh and C.-C. Hsieh, "A 0.4V 13b 270kS/s SAR-ISDM ADC with an opamp-less time-domain integrator," in *2018 IEEE international solid-state circuits conference - (ISSCC)*, 2018, pp. 240–242, doi: 10.1109/ISSCC.2018.8310273.

[98] C. F. Wheatley and H. A. Wittlinger, "OTA obsoletes OP. AMP." in *Proc. National electronics conference*, 1969, pp. 152–157 [Online]. Available: https://class.ece.iastate.edu/ee435/miscHandouts/OTA%20Wheatley%20and%20Wittlinger%20Dec%2069.pdf

[99] B. J. Hosticka, "Dynamic CMOS amplifiers," *IEEE Journal of Solid-State Circuits*, vol. 15, no. 5, pp. 881–886, 1980, doi: 10.1109/JSSC.1980.1051488.

[100] R. L. Geiger and E. Sánchez-Sinencio, "Active filter design using operational transconductance amplifiers: A tutorial," *IEEE Circuits and Devices Magazine*, vol. 1, no. 2, pp. 20–32, 1985, doi: 10.1109/MCD.1985.6311946.

[101] P. R. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *ieee_j_jssc*, vol. 40, no. 6, pp. 1212–1224, 2005.

[102] C. Wulff, "JNW-TEMP: Two temperature sensors, measured." 2026 [Online]. Available: https://analogicus.com/jnw-tt-2025/presentation.html

[103] M. Venn and the Tiny Tapeout contributors, "Tiny tapeout." [Online]. Available: https://tinytapeout.com

[104] M. J. Kirton and M. J. Uren, "Noise in solid-state microstructures: A new perspective on individual defects, interface states and low-frequency (1/f) noise," *Advances in Physics*, vol. 38, no. 4, pp. 367–468, 1989, doi: 10.1080/00018738900101122.

[105] P. A. M. Dirac, "The quantum theory of the emission and absorption of radiation," *Proceedings of the Royal Society of London A*, vol. 114, no. 767, pp. 243–265, 1927, doi: 10.1098/rspa.1927.0039.

[106] H. Nyquist, "Thermal agitation of electric charge in conductors," *Physical Review*, vol. 32, no. 1, pp. 110–113, 1928, doi: 10.1103/PhysRev.32.110.

[107] S. Machlup, "Noise in semiconductors: Spectrum of a two-parameter random signal," *Journal of Applied Physics*, vol. 25, no. 3, pp. 341–343, 1954, doi: 10.1063/1.1721637.

[108] A. L. McWhorter, "1/f noise and germanium surface properties," in *Semiconductor surface physics*, R. H. Kingston, Ed. University of Pennsylvania Press, 1957, pp. 207–228.

[109] C. H. Henry and D. V. Lang, "Nonradiative capture and recombination by multiphonon emission in GaAs and GaP," *Physical Review B*, vol. 15, no. 2, pp. 989–1016, 1977, doi: 10.1103/PhysRevB.15.989.

[110] K. S. Ralls *et al.*, "Discrete resistance switching in submicrometer silicon inversion layers: Individual interface traps and low-frequency (1/f?) noise," *Physical Review Letters*, vol. 52, no. 3, pp. 228–231, 1984, doi: 10.1103/PhysRevLett.52.228.

[111] K. K. Hung, P. K. Ko, C. Hu, and Y. C. Cheng, "A unified model for the flicker noise in metal-oxide-semiconductor field-effect transistors," *IEEE Transactions on Electron Devices*, vol. 37, no. 3, pp. 654–665, 1990, doi: 10.1109/16.47770.

[112] M. Xiao, I. Martin, E. Yablonovitch, and H. W. Jiang, "Electrical detection of the spin resonance of a single electron in a silicon field-effect transistor," *Nature*, vol. 430, pp. 435–439, 2004, doi: 10.1038/nature02727.

[113] F. E. Feyling, "Design Considerations for a Low-Power Control-Bounded A/D Converter," Master's thesis, Norwegian University of Science; Technology (NTNU), 2021 [Online]. Available: https://hdl.handle.net/11250/2824253

Index

A complete OTA, sized, 194
A customer story, 453
AAF, 493, 513
AC open, DC closed switch, 570
Active cascodes, 168
Active-RC, 311, 320, 349
AD-PLL, 516
ADC, 493, 514
ADC state-of-the-art, 385
Add ground, 227
Add labels, 229
Add ports, 218
Add transistors, 218, 226
Advice, 567
Against a real reference, 249
AICEX, 209
Alias, 349
Allan deviation, 247
Ambient RF Harvesting, 534
Amplifiers, 169
An example SC circuit, 355
An HBM ESD zap example, 268
Analog design, 584
Analog design process, 16, 569
Antenna, 41, 495
Antenna size, 493
AOI, 605
AOI22: and or invert, 617
Arrhenius, 125
Ask questions relentlessly, 23
Atomic clocks, 473
Attack vector, 634
Automation, 563
Average power, 125, 126

Band diagrams, 25, 34
Band structure, 25, 31
Bandgap voltage reference, 284
Baseband, 517
Battery, 429, 497
Battery charger - Inputs, 653
Battery charger - States, 654
Battery charger FSM, 652
BB, 493
Bernoulli statistics, 247
Best number of stages, 631
BGAP, 281
Bias, 183, 261, 303
Bias circuits, 192
Bias point, 157
Big bang, 7
Binary coding, 342
Binary scaled DACs, 340
BiQuad, 311
Bluetooth, 507
Bluetooth LE, 493
Bluetooth low energy, 508
Body effect, 100
Boost, 429
BPSK, 499
Brokaw reference, 290
BSIM, 577
Buck, 429
Built-in voltage, 49, 59
Bulk driven input, 189

Calibration, 247
Capacitance, 41
Capacitances, 75
Capacitive load, 483
Capacitors, 40, 199, 202
Capture and emission, 142
Career, 567
Cascode, 157
Cascoded current mirror, 167
CD, 157
CG, 157
Charged device model (CDM), 266
Charging a capacitor to VDD, 640
Chase down what's off, 21
Check waveforms, 222
Checklist, 595
cicconf, 209, 601
ciccreator, 601
CICPY, 563
cicpy, 209, 601

cicsim, 209, 601
cictikz, 601
cicwave, 601
Circuit, 478
Circuits, 157, 661, 666
Clocks, 261
CMFB, 183
CMOS Logic, 605
CMOS oscillators, 491
Common gate, 173
Common mode feedback, 190
Common source, 177
Compile RTL, 591
Configuration files, 216
Constraint Is a Feature, 22
Control of pass-fet, 436
Controlled oscillator, 463, 473
Controlled oscillators, 481
Core voltage, 432
Corners, 215, 231, 601
Course goals, 11
Create the IP, 215
Crystal model, 473
Crystal oscillators, 475, 490
CS, 157
CT SD, 385
Current, 37, 60
Current loops, 41
Current Mirror OTA, 183
Current mirror OTA, 185
Current mirrors, 157, 163
Current mode DACs, 345
Currents in a semiconductor, 39
Curvature correction, 297

D-Latch, 605
D-latch (16 transistors), 615
d_cosim, 587
DAC complexity, 337
DAC errors, 335
Data, 493
Data converters, 661, 667
Data rate, 493, 494
DC-DC converters, 449
DC/DC inductor, 457
DCO Ring, 473
Delay components, 629
Delay definitions, 628
Delay estimation, 629
Density of electrons/holes, 34
Density of States, 49
Density of states, 54
Design, 595, 596
Design files, 216
Design process, 11, 567
DIBL, 75
Diff pairs are cool, 180
Differential, 486
Differential gm-c, 317
Differential pair, 157, 179
Diffusion, 199, 201
Diffusion current, 38
Diffusion leakage, 65
Digital, 261
Digital simulation, 542, 587
Digital simulators, 544
Diode, 661, 663
Diode equation, 49
Dirac's golden rule, 140
Discrete-time signals, 357
Displacement current, 41
Dither, 247
Divider, 468
Documentation, 215
Don't be afraid of I, 556
Doping, 49, 57
Drain of M2, 228
Drain source current, 89
Drain source voltage, 95
DRC, 215
Drift current, 37
DT, 349
Dynamic amplifiers, 183, 195

Edit info.yaml, 232
Edit readme.md, 217
Efficiency, 429
Einstein: The source, 136
Electromagnetic, 521, 533
Electron, 7
Electron injection, 277
Electrons, 26
Electrostatic discharge, 264
Elmore delay, 629
Equations aren't real, 71
Every wire is both, 44

Example, 381
Example circuit, 325
Example: thermal noise, 135
Examples, 416

FAQ, 659
Far field, 41
Faraday, 41
Feedback, 453
Fermi level, 25, 32
Fermi-Dirac, 49, 125, 138
Field effect, 75, 77
Fields, 25, 35
Figure of Merit, 240
Filters, 314, 661, 667
Find right transistor sizes, 582
Finding a transconductor, 319
FIR, 349
First order, 385
First order filter, 314, 365
First-order noise-shaping, 413
Five transistor, 183
Five transistor OTA, 184
Fix process variation, 117
Flicker noise, 125
Flip-Flop, 605
Flip-flops and speed, 622
Flipped voltage follower, 429
Focus, 21
Folded cascode, 183, 187
FOM, 235, 385
Foundries, 577, 582
Fractional PLL, 460
Frequency, 455
Frequently asked questions, 232, 659
Friis, 125
From OTA to op amp, 193
FSM, 651
Fully differential, 190
Fundamental physics, 661
Fusion, 7
Future work, 559

Gate current, 112
Gate delay, 628
Gate-source voltage, 90
gds, 75
General purpose biquad, 323
Get creative with python, 223
GFSK, 493
Git, 209, 659
Github setup, 217
Github workflows, 215
gm, 75
GM Cell, 304
Gm-C, 311, 316, 351
gm/ID, 75, 157, 577
GMCELL, 281
Golden rule, 125
Grade, 240
Ground, 261

H(s), 311
HCI, 75
Headroom, 183
High frequency model, 101
High resolution FOM, 394
Honesty, 21
Hot carrier injection, 113
How big is one step?, 145
How GR06 works, 147
How I see our roles, 12
How many traps?, 146
How to tell, 150
Human body model (HBM), 265

IC, 7
IC design mantra, 16, 569
Icarus, 541
Ictrl rosc, 473
IIR, 349
Impedance, 476
Implementation, 555, 558
Import in testbench, 592
INCR, 385
Inductance, 41
Inductors, 40, 199, 205
INL, 247
Input buffer, 275
Install cicconf, 212
Install cicpy, 212
Install cicsim, 212
Install tools, 211
Insulators, 25, 33
Integer divider, 453
Integer PLL, 459

Intrinsic carriers, 49
Intrinsic gain, 75, 99
Introduction, 555
Introduction to behavior, 89
Inversion level, 91
Inverter, 605
Inverter based OTAs, 188
Inverter-Based, 183
IO voltage, 433
IP, 215
IREF, 281
Is it the supply?, 148
It's all quantum, 52

Latch-up, 276
Layout, 215, 563
Layout compilation, 601
Layout generation, 563
Layout rules, 595, 598
LC oscillator, 487
LCOSC, 473
LDON, 429
LDOP, 429
Leakage, 49, 235
Li-Ion batteries, 652
Line, 429
linear phase model, 453
Linear regulators, 434, 449
Lithium battery, 521
LNA, 493
LNTA, 511
Load, 429
Logic cells, 609
Loop filter, 467
Loop gain, 463
Loop transfer function, 468
Lorentzian, 125
Low frequency model, 97
Low voltage bandgap, 294
LP RX, 493
LPE, 215
Lumped model, 649
LVBGAP, 281
LVS, 215

Magic, 209
Magic VLSI, 225
Make documentation, 231
Many traps, 151
MAX C, 429
Maxwell, 41
McWhorter, 125
Mealy machine, 651
Mealy versus Moore, 651
Measured silicon, 247
Memoryless, 149
Metal, 202
Metal oxide semiconductor, 75
Metal routing rules on IC, 649
Metal stack, 648
Metals, 25, 33
Milestone 0: The tutorial, 241
Milestone 1: The bandgap, 241
Milestone 5: the report, 244
Milestones, 235
Miller, 75
Miller compensation, 183
Mixed signal, 541, 587
Mixed signal simulation, 547
MIXER, 512
Mixer, 493
Mobility degradation, 108
Modeling interconnect, 649
Modulation, 453, 493, 498
Modulation in PLLs, 460
MOM, 199
Monte carlo, 601
Moore machine, 651
More information, 584
MOS Cap, 199
MOS capacitors, 203
MOS references, 299
MOSFET, 661, 663
Most wires: copper, 650
Multiphonon capture, 125
Mux, 619
My favorite OTA, 327

NAND, 605
Nauta, 183
Near field, 41
NEG FB, 385
Netlist schematic, 219
ngspice, 209, 215, 541, 587, 601
NMOS pass-fet, 435
Node voltage, 261

Noise, 75, 374, 455, 661, 665
Noise equations, 121
Noise figure, 125
Noise power, 404
Noise shaping, 408
Noise spectrum, 127
Noise transfer function, 412
Non-Overlap, 349
Normal current mirror, 164
Not one Gaussian, 144
NTF, 385
Nyquist, 349

OFDM, 493
On writing English, 555
One trap, 138, 257
Op amp, 183
Option 1: Full freedom, 161
Option 2: constrained, 161
Oscillators, 661, 669
OSR, 385
OTA, 157, 183, 311
OTAs, 183
OTHER, 109
Other logic cells, 616
Oversampling, 404
Ownership, 21

Package, 261
Paragraphs, 555
Passives, 577, 580
PCB antenna, 456
Pelgrom, 75
PFM, 429
Photovoltaic, 521
Physics, 661
pi/4-qpsk, 493
Pick two, 639
Pierce, 473
Piezoelectric, 521
Place devices, 226
Placement, 563, 564
Plasma charging currents, 67
PLL, 453, 459, 661, 668
PLL Example, 461
PMOS pass-fet, 434
PN Junction, 49
PN junctions, 58
Pole-Zero plots, 363
Polysilicon, 199, 200
Power, 639
Power supply, 497
Probability, 27
Probability distribution, 129
Process corners, 116
Process variations, 116
PSD, 125
PSD of a white noise source, 129
PSRR, 429
PTAT, 247
PWM, 429
Python noise-shaping, 416
Python oversample, 407

Quantisation, 247
Quantization, 385, 395
Quantum electrodynamics, 36
Quiescent, 429

Radiation, 41
Radio, 661, 669
Random telegraph noise, 125, 247
Random variation, 118
Range, 493
Range (free space), 496
Range (friis), 496
Range (real world), 497
Rate, 494
RCOSC, 473
Real number modeling, 541
Real world use, 446
Realistic, 484
Reduce frequency, 646
Reference, 661
References, 661
References and bias, 666
Regulator, 235
Regulators, 661
Relative precision, 207
Relaxation oscillators, 489
Report structure, 557
Reproducibility, 601
RESET (POR), 261
Resistor based DACs, 333
Resistors, 39, 199, 200
Result, 558

Results, 555
Return path, 41
Reverse leakage, 64
RF Harvesting, 521
Ring osc, 473
Ring oscillator, 481
Roles, 11
Root cause, 21
Root Causes or Nothing, 22
Route gates, 228
Routing, 282, 563
RTL, 587
Rubidium standard, 474
Rules for inverting logic, 612
Run all corners, 223
Running, 592

Sand, 7
SAR, 385, 541
SAR operation, 549
SC DAC, 349
SC FUND, 349
SC INT, 349
SC MDAC, 349
SC SD, 385
Schematic rules, 595, 596
Schrödinger, 125, 137
SD, 453
SD PLL, 453
Second order filter, 315, 366
Semiconductors, 25, 34, 661, 662
Sensing the common mode, 191
Separate Identity from Job, 23
Settling, 429
Setup, 563, 659
Setup github pages, 232
Shorter is better, 555
Shot noise, 125
SI Units, 25
Sigma-delta principle, 410
Signal power, 405
Signal to Noise Ratio, 405
Signal to Noise Ratios, 131
Signal transfer function, 412
Silicon, 7, 25, 49
Silicon unit cell, 30
Simulation summary, 231
Simulations, 217
Sizing, 157
Skills, 11, 567
SNR, 125
Source follower, 169
Sources, 577, 579
Sources of Confusion, 134
Specification, 235, 239, 595
Spectral density, 133
Speed, 621
SPICE, 577
SQNR and ENOB, 415
Square law model, 107
SR-Latch, 605, 614
Stability, 455
Standards, 21
Start working, 217
Static logic, 605
Statistics, 125, 126
STF, 385
Stop activity, 645
Stress effects, 111
Strong inversion, 75, 95
Structure, 555
Stuff to ponder, 570
Subsample, 349
Summing noise sources, 130
Supernova, 7
Supply, 262
Supply planning, 433
Switch, 349
Switched capacitor, 351, 661, 667
Switched capacitor CMFB, 192
Switched regulators, 438
Switched-Capacitor, 369
Switching probability, 642
Systematic variations, 115
Systems, 453
SystemVerilog, 541

Tapeout, 11, 235, 247, 595, 596
Temperature, 49, 473
Temperature behavior, 481
Temperature chamber, 247
Temperature sensor, 235
Temperature variation, 117
TempSense, 261
Testbench, 587
The capacitor falls out, 43

The challenge, 235
The circuit, 590
The clock in the loop, 252
The digital code, 590
The file structure, 215
The grounded gate NMOS, 271
The headroom budget, 183
The inductor falls out, 43
The lattice must move, 141
The Lorentzian, measured, 145
The magic of feedback, 409
The mathematics, 358
The model card knows, 152
The noise floor, 125
The OTA is not ideal, 324
The project, 238
The suspects, 149
The work, 217
Theory, 555, 558
Thermal noise, 125
Thermal noise is Planck, 136
Thermoelectric, 521
Thermometer encoding, 344
Think about the results, 224
Threshold, 75
Time-domain readout, 247
Timing analysis, 625
Timing analysis tools, 626
Tiny tapeout, 247
TinyTapeout, 541
Today, 578
Tools, 209, 659
Transconductance, 98
Transfer curve, 247
Transient analog simulation, 545, 588
Transistor models, 577, 580
Transistor sizing strategy, 161
Transistors, 581
Transit frequency, 103
Transitions, 555
Transitions are important, 556
Trends, 633
Triboelectric, 521
Triboelectric generator, 536
Tristate, 605
Tristate inverter, 618
Tunnelling, 125
Two sensors, one event, 255
Two stage (Miller) OTA, 186
Two states, one Lorentzian, 144
Two-Stage, 183
Typical simuation, 231

Uncertainty principle, 28
Unit transistors, 577
Use paragraphs, 556

Value, 21
Varactors, 199, 205
Variability, 75, 114
Variation in passives, 206
VBE SC, 349
VD, 281
VDD, 261
Velocity saturation, 75, 106
Verification, 183
Verifying the OTA, 196
Verilog, 587
VI, 281
Voltage, 36
Voltage regulation, 668
Voltage source, 429
Voltage variation, 115
VREF, 281
Vreg, 429

Wafer, 7
Weak inversion, 75, 104
What a trap is, 138
What about gm/Id ?, 583
What about holes (PMOS), 108
What calibration buys, 251
What else can we do?, 120
What is power?, 639
What one trap looks like, 143
What the silicon does, 249
What this buys you on-chip, 47
Which has shortest delay?, 631
Which transistor in GR06?, 147
White noise, 125
Why are there two currents?, 38
Why clocks?, 453
Why common source?, 178
Why ESD, 261
Why transistors die, 430

Why use a source follower?, 172
Widlar reference, 289
Will you tape-out an IC?, 14
Wire capacitance, 650
Wire geometry, 648
Wire resistance, 649
Wires, 648
Wires below 1 mm, 70
WPE, 75
Writing, 555
WSL, 209

xschem, 209, 215

Z Domain, 349
Z-domain, 364
Z-transform, 362
Zen of IC Design, 11, 567