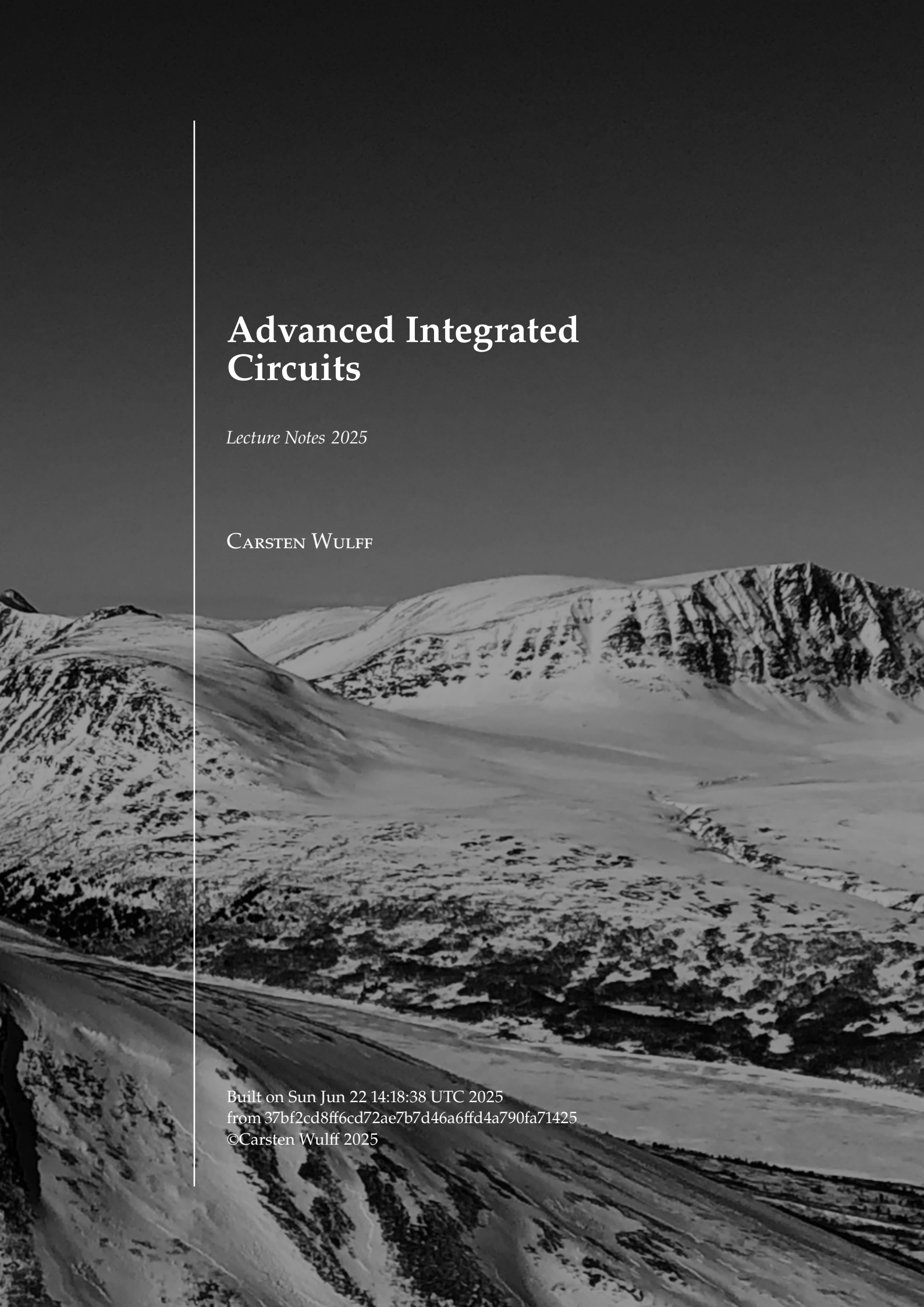
Advanced Integrated Circuits

Lecture Notes 2025

Carsten Wulff



Built on Sun Jun 22 14:18:38 UTC 2025
from 37bf2cd8ff6cd72ae7b7d46a6ffd4a790fa71425
©Carsten Wulff 2025

Contents

Background 1

In the spring of 2025 I lectured Advanced Integrated Circuits for the fourth 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 probably a few 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 %.

Date	Attendance
2024-02-02	19
2024-02-09	17
2024-02-16	16
2024-03-01	14
2024-03-07	14
2024-03-15	12
2024-03-22	13
2024-04-12	16
2024-04-19	10

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 %.

Wk	Attendance
2	21
3	21
4	23
5	20
6	22
7	24
9	20
9	24
11	20
12	17
14	16
15	14

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 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 2026 I should do the following changes:

- Wait until after M0 for group selection
- Talk about layout early. Force full M0 tutorial
- Make them do TR layout early
- Re-introduce milestone 3
- Write a detailed project description and milestone and expectation description
- Reduce time for milestone 1. Maybe make a ready schematic hierarchy to force names? ideal OTA?
- Find a good sigma delta intro circuit
- Add to analog systemverilog

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 aic2024, 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.

Introduction 2

Status: 1.0

2.1 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 make 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

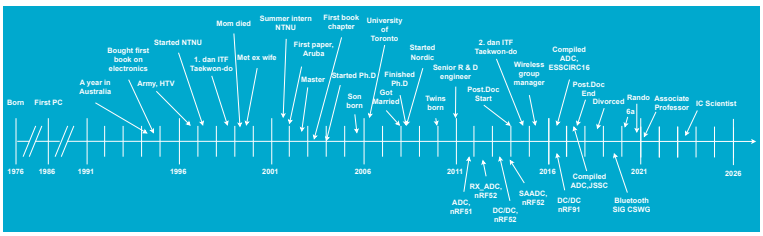


Figure 1: My life

2.2 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

Figure 2: Research Universe

2.3 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 were 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.

> Making integrated circuits (that work reliably) is not rocket science, it's much harder.

2.4 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 is 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 do 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 an insatiable desire to understand how the world *really* works at a fundamental level, then it's likely that you'll find analog flow interesting.

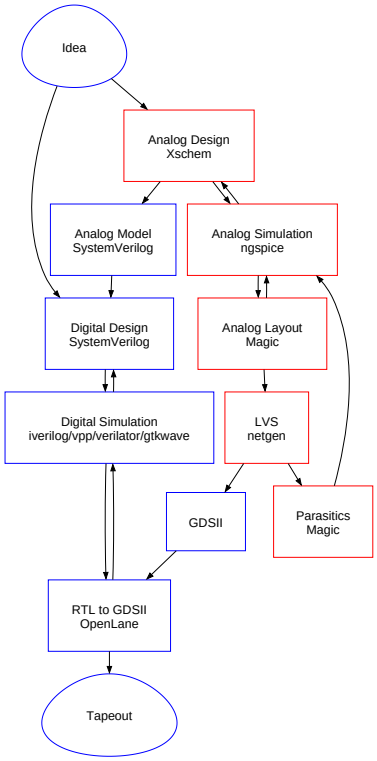


Figure 3: Analog and Digital design process

–>

2.5 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.

Together with Efabless there are cheap alternatives, like tinytapeout, which makes it possible for a private citizen to tape-out their own integrated circuit.

2.5.1 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 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**, bulid 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

2.5.2 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 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.

2.5.3 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
```

2.5.4 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. 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 your done.

On failure, go back as far as necessary

2.6 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.

I want to:

- 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.

2.7 Syllabus

The syllabus will be from Analog Integrated Circuit Design (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.

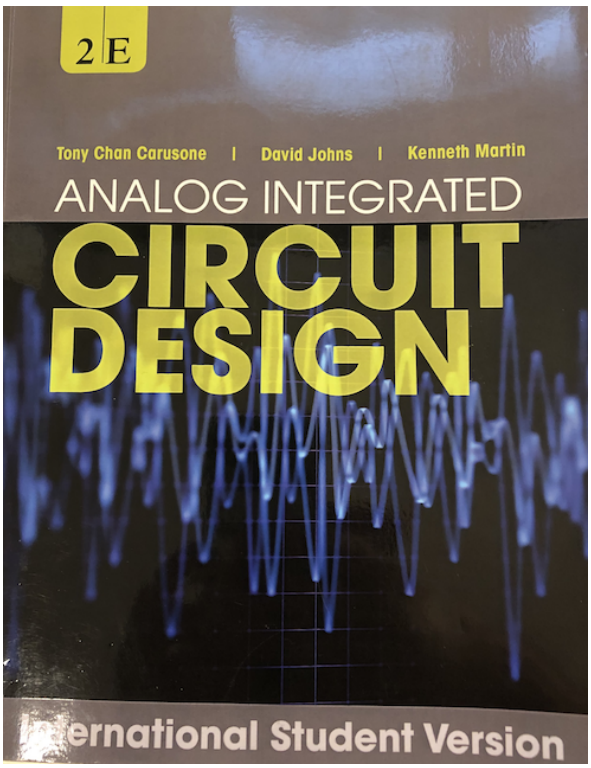


Figure 4: The book we'll use

2.8 Project JNW (2025)

"You can use logic to justify almost anything. That's its power. And its flaw." - Kathryn Janeway, Star Trek Voyager: Prime Factors

The project for 2025 is to

Design a integrated temperature sensor with digital read-out

An outline of the plan is shown below.

At the end of the project you will have a function that converts temperature to a digital value.

$$D = f_0(T)$$

I've broken down the challenge into three steps, first convert Temperature into a current

$$I = f_1(T)$$

Then convert current into a time

$$t = f_2(I)$$

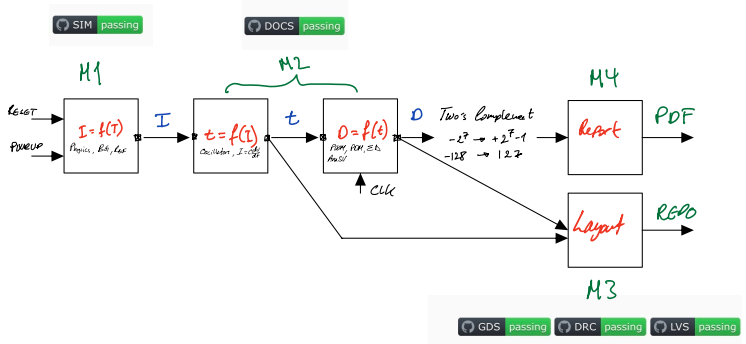
then time to digital

$$D = f_3(t) = f_3(f_2(f_1(T))) = f_0(T)$$

The third milestone is the layout, while the fourth milestone is the report.

You can find an example of last years designs at cnr_gr02_-sky130nm

You will be using a repository on github for all your design data. In that repository I've made it possible to run github actions, or github workflows. For each of the milestones there are associated workflows (SIM/DOCS/GDS/DRC/LVS).



Milestone 0: The zero milestone is not really part of the project, but it does introduce you too how you will work with the files in the project. It's important that you do this right away. To complete the milestone, upload a link to blackboard with your github repository for the tutorial Skywater 130 nm Tutorial

Milestone 1: The first milestone is to make a circuit that can convert from a temperature, to a current that is proportional to temperature. You will run a simulation on github that demonstrates that the circuit works. That is the SIM workflow.

Milestone 2: In the second milestone you will complete the schematic design of the circuit, and possibly also do some SystemVerilog to demonstrate that you get a digital value out that is proportional to temperature. Here, the simulations on github may be too long, so it's sufficient to describe the circuit, and how it works in detail in the documentation. This is the DOC workflow.

Milestone 3: The third milestone, making the layout, is optional, however, it will be impossible to get an A without getting some points from the layout milestone. Once the layout is complete, I expect that the design rule checks (DRC), Layout versus Schematic (LVS), and GDS (stream out to a GDSII file) is passing on github.

Milestone 4: 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.

2.8.1 Grading

Milestone	What does it mean	Condition for more than 0 points	Possible Points
M1 I=f(T)	Circuit that can convert a temperature into a current	SIM passing	10
M2 D=f(T)	Circuit that can convert from temperature into a digital value	DOC passing	20
M3 Lay-out	Layout of your circuit	DRC/LVS/GDS passing	20
M4 Re-port	Individual report	Uploaded to blackboard	48
Coolness	Extra points that I may choose to award		10
Total			108

2.8.2 Group dynamics

How you work together is important. No-one can do everything by them self. I know from experience it can be magical when bright brains come together. The collective brain can be smarter, better, faster, than anyone in the group.

That's why I think it's important not to just work in groups, but also focus on how we work in groups.

A group shall be maximum 4 members. There must be at least 3 that don't know each-other that well.

The group will meet once per week in the exercise hours.

2.8.2.1 Check-in

All group session must start with a Check-in (10 minutes)

Some example questions could be

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

Some examples answers could be:

- My dog died yesterday, so I'm not feeling great today.
- I woke up early, had an omelet, and went running, so I feel motivated and fantastic.
- I feel *blaaah* today, motivation is lacking.
- I went running yesterday and did not discover before I got home that I'd forgotten to put my pants on, even though it was -10 C.

The point of this exercise is to get to know each other a bit, and attempt to create psychological safety in the group.

2.9 Software

We'll use professional Open source software (xschem, ngspice, sky130A PDK, Magic VLSI, netgen)

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. I strongly recommend you start with that first.

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

A Refresher 3

Status: 0.8

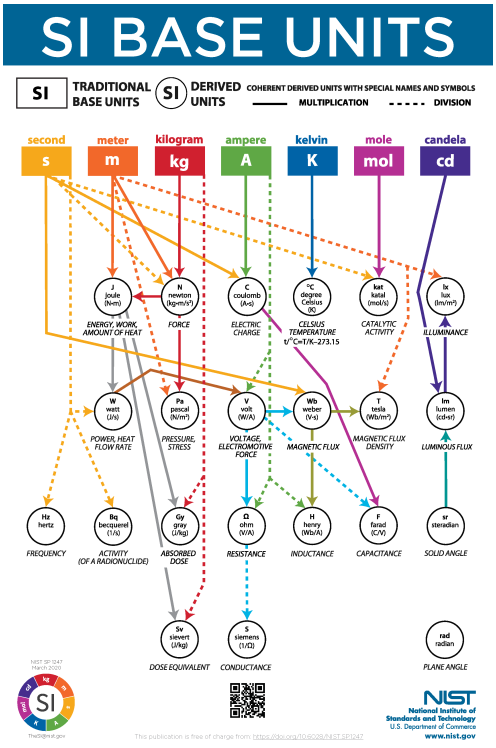
3.1 There are standard units of measurement

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
- 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.

Figure 1: Si base units, from https://www.nist.gov/pml/owm/metric-si/si-units

3.2 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

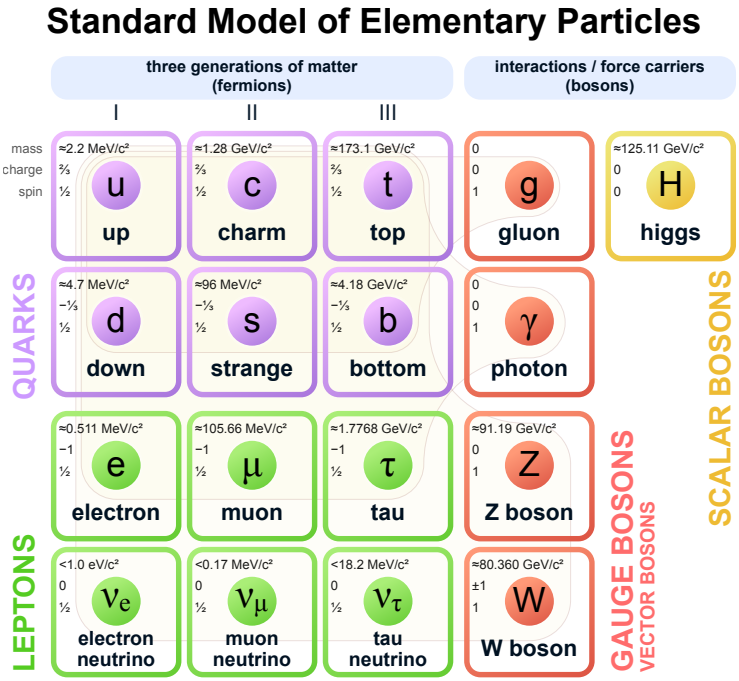


Figure 2: Standard model of particle physics, Wikipedia

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 that applies to all Fermions (particles with spin of 1/2)

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

3.3 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.

3.4 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

3.5 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.

3.6 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.

3.7 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.

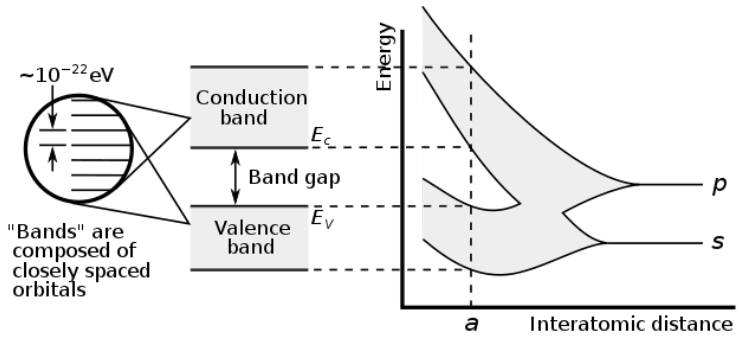


Figure 3: Electronic band structure, Wikipedia

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

3.8 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.

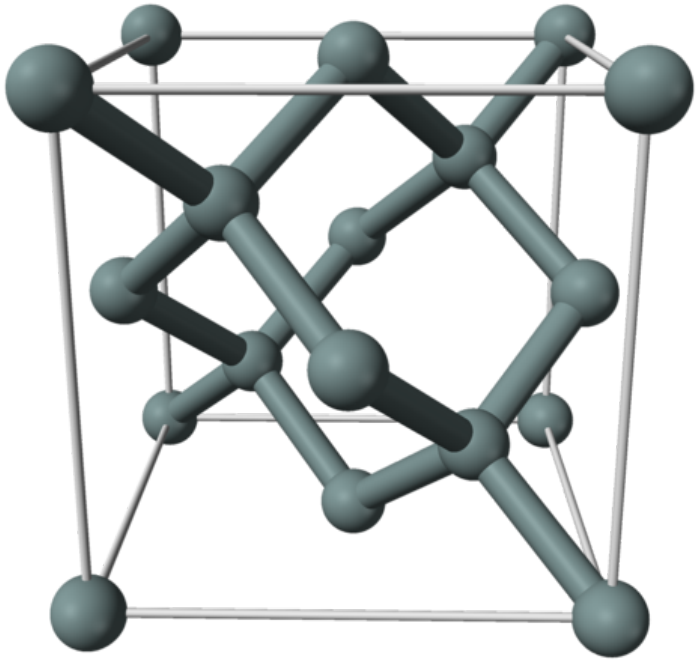


Figure 4: Silicon, Wikipedia

3.9 Band structure

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

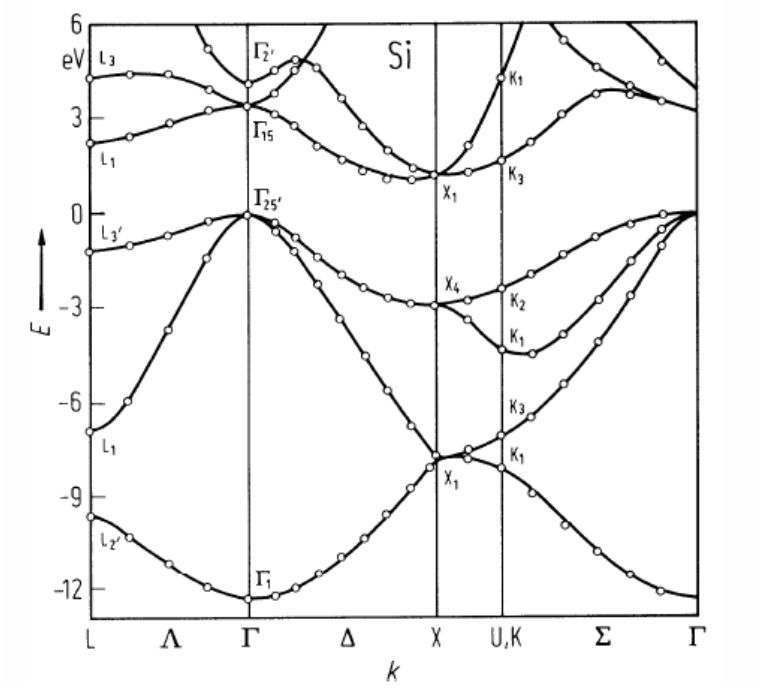


Figure 5: Silicon Band Structure

3.10 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.

$$E_G = E_C - E_V$$

3.11 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.

3.12 Metals

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

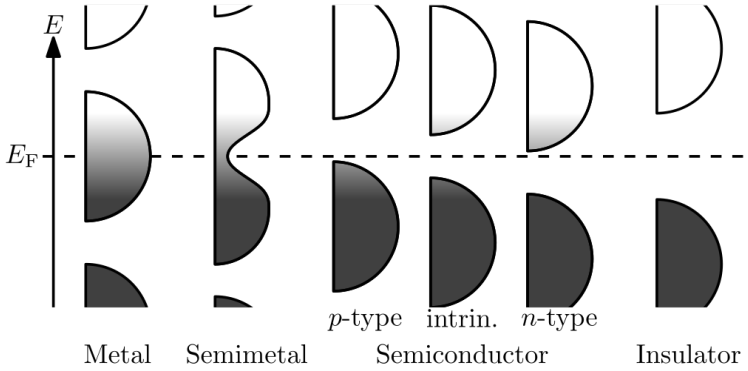


Figure 6: Band splitting in materials. Electronic Band Structure, Wikipedia

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.

3.13 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))
```

3.14 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.

3.15 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.

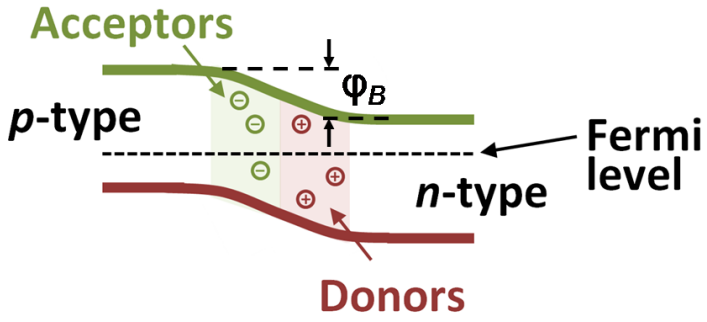


Figure 7: Band diagram of a PN junction, Wikipedia

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

3.16 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.

3.17 Fields

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

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

,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.

3.18 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.

3.19 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

3.20 Voltage

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

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

3.21 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.

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

3.22 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

3.23 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.

3.24 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.

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

3.25 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.

3.26 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.

3.27 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.

3.28 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.

Diodes 4

Status: 1.0

4.1 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 †

4.2 Silicon

Integrated circuits use single crystalline silicon. The silicon crystal is grown with the Czochralski method which forms a 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

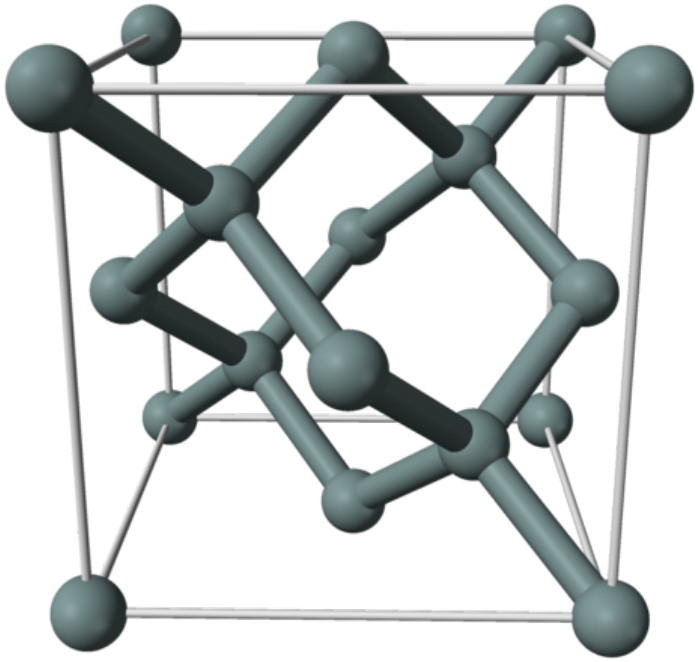


Figure 1: Silicon crystal unit cell

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" orbitial, 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 an 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.

4.3 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.

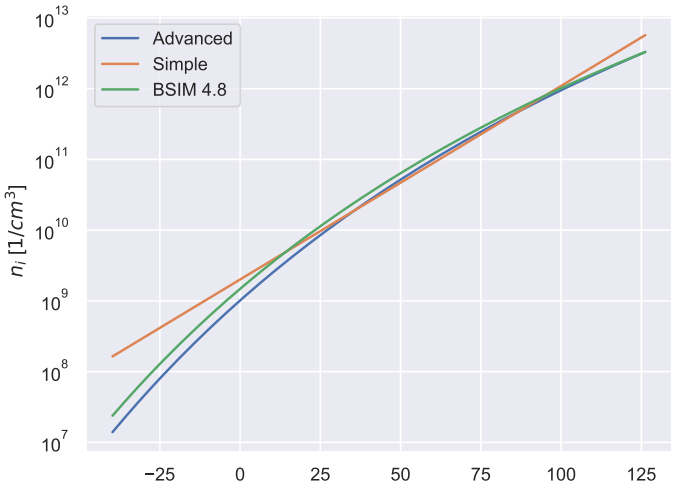


Figure 2: Intrinsic carrier concentration versus temperature

At room temperature the intrinsic carrier consentration 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 $n_i = \frac{1\times10^{16}}{(1\times10^6)^3}$ 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.

4.4 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

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

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

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

4.4.1 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 approxmiation 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

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

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 checmical 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 observiation 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 equlibrium, 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 $n_i^2 = n_0 p_0$

4.4.2 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 literaly 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.

4.5 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. 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 probabilty 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.

4.6 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.

4.6.1 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

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

4.6.2 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

Asssume 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 probabilty 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 start's 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

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

4.6.3 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 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.

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.

Although it's not trivial to see that the diode voltage has a negative temperature coefficient, if you do compute it as in vd.py, then you'll see it decreases.

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

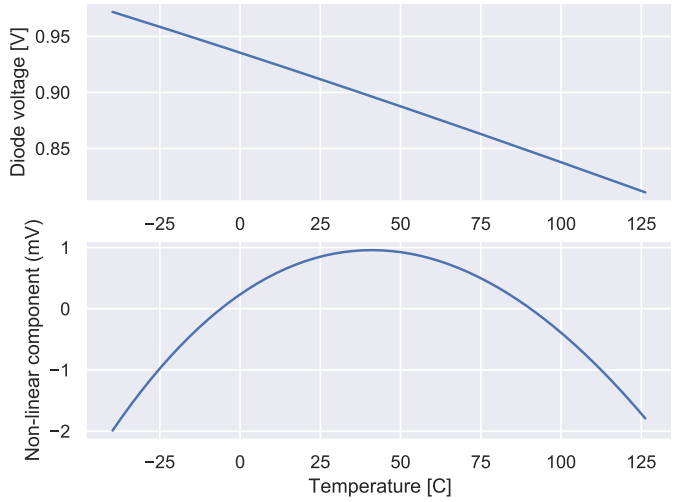


Figure 3: Diode forward voltage as a function of temperature

4.6.4 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.

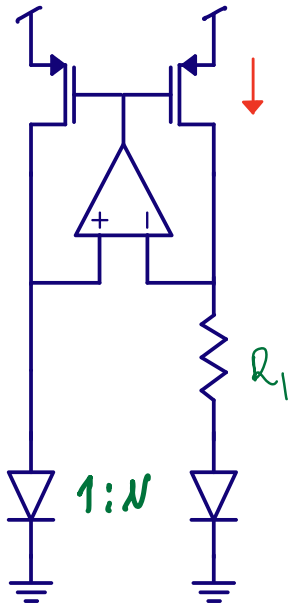


Figure 4: Circuit to generate a current proportional to kT

4.7 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.

References

[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/

Noise 5

Status: 0.5

5.1 Noise

Noise is a phenomena 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 bandwith of 96kHz. There exist an equation that relates number of bits to signal to noise ratio 1, the equation specifies that $SNR = 6.02 * Bits + 1.76 = 146.24dB$. As of 12.2005 the best digital to analog converter (DAC) that Analog Devices (a very big semiconductor company) has is 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!

5.2 Statistics

The mean of a signal x(t) is defined as

$$\overline{x(t)} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t)dt$$

The mean square of x(t) defined as

$$\overline{x^2(t)} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x^2(t)dt$$

The variance of x(t) defined as

$$\sigma^2 = \overline{x^2(t)} - \overline{x(t)}^2$$

For a signals with a mean of zero the variance is equal to the mean square. The auto-correlation of x(t) is defined as

$$\begin{aligned} R_x(\tau) &= \overline{x(t)x(t+\tau)} \\ &= \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t)x(t+\tau)dt \end{aligned}$$

5.3 Average Power

Average power is defined for a continuous system as ([eq:powcont]) and for discrete samples it can be defined as ([eq:powsamp]).

P_{av} usually has the unit A^2 or V^2, so we have to multiply/devide 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 $\sqrt{P_{av}}$.

$$P_{av} = \lim_{T \to \infty} \frac{1}{T} \int_{-T/2}^{+T/2} x^2(t) dt$$

$$P_{av} = \frac{1}{N} \sum_{i=0}^{N} x^2(i)$$

If x(t) has a mean of zero then, according to ([eq:var]), 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 Table [t:avgpow] and Table 2. 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 paper 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}
$\sqrt{\overline{V_n^2}}$	$\sqrt{\overline{I_n^2}}$
$\sqrt{\overline{v_n^2}}$	$\sqrt{\overline{i_n^2}}$

5.4 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 $\frac{I}{\sqrt{Hz}}$ for current.

The power spectral density is defined as two times the Fourier transform of the auto-correlation function 2

$$S_x(f) = 2\int_{-\infty}^{\infty} R_x(\tau)e^{-j2\pi f\tau}d\tau$$

This can also be written as

$$\begin{aligned}
S_x(f) = \quad & 2\left[\int_{-\infty}^{\infty} R_x(\tau)\cos(\omega\tau)d\tau - \int_{-\infty}^{\infty} R_x(\tau)j\sin(\omega\tau)d\tau\right] \\
= \quad & 2\left[\int_{-\infty}^{0} R_x(\tau)\cos(\omega\tau)d\tau + \int_{0}^{\infty} R_x(\tau)\cos(\omega\tau)d\tau\right] \\
- \quad & 2j\left[\int_{-\infty}^{0} R_x(\tau)\sin(\omega\tau)d\tau + \int_{0}^{\infty} R_x(\tau)\sin(\omega\tau)d\tau\right] \\
= \quad & 4\int_{0}^{\infty} R_x(\tau)\cos(\omega\tau)d\tau \\
- \quad & 2j\left[-\int_{0}^{\infty} R_x(\tau)\sin(\omega\tau)d\tau + \int_{0}^{\infty} R_x(\tau)\sin(\omega\tau)d\tau\right] \\
= \quad & 4\int_{0}^{\infty} R_x(\tau)\cos(\omega\tau)d\tau
\end{aligned}$$

, 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

$$\overline{x^2(t)} = \int_{0}^{\infty} S_x(f)df$$

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

$$S_y(f) = S_x(f)|H(f)|^2$$

, 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 ([eq:psd_hf]) into ([eq:ms_psd]), 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 equation ([eq:psd_hf]) 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.

5.5 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 white.* 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 movments with a large number of electrons will give us a Gaussian distribution, thus thermal noise is white.

* Gaussian distribution = normal distribution. Noise sources with Gaussian distribution are called white

5.6 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 equation ([eq:autocor]) we have that

$$\begin{aligned} R_x(\tau) = & \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x(t)x(t-\tau)dt \\ = & \left[\lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{+T/2} x^2(t)dt \right] \delta(\tau) \\ = & \overline{x^2(t)}\delta(\tau) \end{aligned}$$

The reason being that in a true random process $x(t)$ is uncorrelated with $x(t+\tau)$ where τ is an integer. If we use equation ([eq:psd]) 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.

5.7 Summing noise sources

Summing noise sources is usually trivial, but we need to know why and when it is not. We 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 ([eq:noisesum]) and just write

$$\overline{e_{tot}^2} = \overline{e_1^2} + \overline{e_2^2}$$

Most natural noise sources are uncorrelated.

5.8 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

$$\begin{aligned}
SNR = & \quad 10\log\left(\frac{Signal\ power}{Noise\ power}\right) \\
= & \quad 10\log\left(\frac{\overline{v_{sig}^2}}{\overline{e_n^2}}\right) \\
= & \quad 20\log\left(\frac{v_{rms}}{\sqrt{\overline{e_n^2}}}\right)
\end{aligned}$$

Another useful ratio is Signal to Noise and Distortion (SNDR), since most real systems exibit 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

$$SNDR = 10\log\left(\frac{Signal\ power}{Noise\ power}\right)$$

5.9 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. 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

$$F = F_1 + \sum_{i=1}^{N} \frac{F_{i+1} - 1}{\prod_{k=1}^{i-1} G_i}$$

Friiss 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!

5.10 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 3.

5.10.1 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 4, 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 ([eq:rxequal]) is true.

5.10.2 Sources of Confusion

The problem with spectral density arises when reading literature from different communities, for example 4 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 sporadicly) I eventually found the original source of the definition of spectral density 5. Having the original source helped, but I still don't know when the original definition split into ([eq:psd1]) and ([eq:psd2]). However, I'm pretty sure the it's just a matter of convenience. To see why ([eq:psd2]) is the most common among sources concerning noise we look at the inverse Fourier Transform. By the way, if you had not noticed yet, ([eq:psd1]) says

that *Spectral density is the Fourier Transform of the Auto-Correlation function*. The inverse Fourier Transform of ([eq:psd1]) 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 ([eq:psd2])

$$R_{x2}(\tau) = \frac{1}{2}\int_{-\infty}^{\infty} S_{x2}(f)e^{jw\tau}df$$

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 ([eq:psd1]) 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

$$\begin{aligned} S_{x1}(f) &= \int_{-\infty}^{\infty} R_{x1}(\tau)\cos(\omega\tau)d\tau \\ R_{x1}(\tau) &= \int_{-\infty}^{\infty} S_{x1}(f)\cos(\omega\tau)df \end{aligned}$$

and

$$\begin{aligned} S_{x2}(f) &= 2\int_{-\infty}^{\infty} R_{x2}(\tau)\cos(\omega\tau)d\tau \\ R_{x2}(\tau) &= \frac{1}{2}\int_{-\infty}^{\infty} S_{x2}(f)\cos(\omega\tau)df \end{aligned}$$

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 spectral density definition ([eq:psd2]) 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 ([eq:psd1]) 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 $S_{x1}(f) \neq S_{x2}(f)$.

Definition ([eq:psd1]) is called the two-sided spectral density and ([eq:psd2]) is called the one-sided spectral density.

5.10.3 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 ([eq:othermal]) is the spectral density when it is defined as in ([eq:psd2]). If we were to use ([eq:psd1]) 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 ([eq:psd1]) and ([eq:psd2]).[†]

5.10.4 Einstein: The source

In his 1914 paper 5 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

$$I(\theta) = \int_0^T \mathfrak{M}(\Delta)\cos(\pi\frac{\Delta}{\theta})d\Delta$$

[†] 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.

,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 ([eq:psd1]) is often referred to as the *Wiener-Khintchine* theorem.

Sky130nm tutorial 6

Status: 1.0

6.1 Tools

I would strongly recommend that you install all tools locally on your system.

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
- ngspice for simulation
- netgen for LVS
- xschem
- 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.

6.1.1 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
```

6.1.2 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

6.1.3 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"
```

6.1.4 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, chose 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
```

6.1.5 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
```

6.1.6 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
```

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
```

6.1.7 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 ../..
```

6.1.8 Install cicsim

cIcSim is used for simulation orchestration.

```
cd aicex/ip/cicsim
python3 -m pip install -e .
cd ../..
```

6.1.9 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
```

6.2 Check that magic and xschem works

To check that magic and xschem works

```
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 &
```

6.3 Design tutorial

6.3.1 Create the IP

I've made some scripts to automatically generate the IP.

To see what files are generated, see `tech_sky130A/cicconf/ip_-template.yaml`

```
cd aicex/ip
cicconf newip ex
```

6.3.2 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.

6.3.2.1 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 grading of the milestones is determined by passing github workflows.

We will also check that you have not cheated, and modified the workflows just to get them passing.

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
   sim.yaml  # Run a simulation
```

6.3.2.2 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
```

6.3.2.3 Design files

A "cell" in the open source EDA world should consists of the following files

- Schematic (.sch)
- Layout (.mag)
- Documenation (.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
  JNW_EX_SKY130A
  JNW_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 `JNW_EX`, then you would have

- `design/JNW_EX_SKY130A/JNW_EX.sch`: Schematic (xschem)
- `design/JNW_EX_SKY130A/JNW_EX.sym`: Schematic (xschem)
- `design/JNW_EX_SKY130A/JNW_EX.mag`: Layout (Magic)
- `design/JNW_EX_SKY130A/JNW_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)

6.3.2.4 Simulations

All simulations shall be stored in `sim`. Once you have a Schematic ready for simulation, then

```
cd sim
make cell CELL=JNW_EX
```

This will make a simulation folder for you. Repeat for all your cells.

```
sim
  Makefile
  cicsim.yaml -> ../tech/cicsim/cicsim.yaml
```

6.3.2.5 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
```

6.3.3 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 `jnw_ex_sky130a`.

```
cd jnw_ex_sky130a
git remote add origin \
 git@github.com:<your username>/jnw_ex_sky130a.git
```

6.3.4 Start working

6.3.4.1 Edit README.md

Open README.md in your favorite text editor and make necessary changes.

6.3.4.2 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.

6.3.5 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/JNW_EX_SKY130A/JNW_EX.sch &
```

6.3.5.1 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.

6.3.5.2 Add transistors

Use 'I' or 'Shift+i' (note the letter case) to open the library manager. Click the `jnw_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.

Press ESC to deselect everything

Select the input transistor, and change the name to ‘xi’

Select the output transistor, and change the name to ‘xo[3:0]’. Using bus notation on the name will create 4 transistors

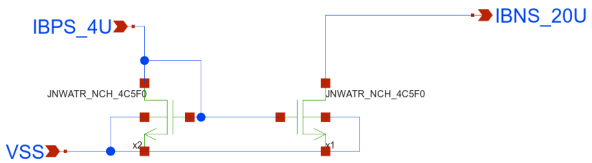
Select ports, and use ‘m’ to move the ports close to the transistors.

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



6.3.5.3 Netlist schematic

Check that the netlist looks OK

In work/

```
make xsch CELL=JNW_EX
cat xsch/JNW_EX.spice
```

6.3.6 Typical corner SPICE simulation

I’ve made cicsim that I use to run simulations (ngspice) and extract results

6.3.6.1 Setup simulation environment

Navigate to the `jnw_ex_sky130a/sim/` directory.

Make a new simulation folder

```
cicsim simcell  JNW_EX_SKY130A JNW_EX \
    ../tech/cicsim/cell_spice/template.yaml
```

I would recommend you have a look at simcell_template.yaml file to understand what happens.

6.3.6.2 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 JNW_EX
make typical
```

6.3.6.3 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)
```

6.3.6.4 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.

6.3.6.5 Modify result specification (tran.yaml)

Add the result specifications, for example

```
ibn:
  src:
    - ibns_20u
  name: Output current
  min: -20%
  typ: 20
  max: 20%
  scale: 1e6
  digits: 3
  unit: uA

vgs:
  src:
    - vgs_m1
  name: Gate-Source voltage
  typ: 0.6
```

```
  min: 0.3
  max: 0.7
  scale: 1
  digits: 3
  unit: V
```

Re-run the measurement and result generation

```
make typical OPT="--no-run"
```

Open `result/tran_Sch_typical.html`

6.3.6.6 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

6.3.7 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.

For the current mirror we don't need to vary voltage, since we don't have a VDD.

6.3.7.1 Remove Vh and Vl corners (Makefile)

Open Makefile in your favorite text editor.

Change all instances of "Vt,Vl,Vh" and "Vl,Vh" to Vt

6.3.7.2 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.

6.3.7.3 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 ;-)

6.3.7.4 Generate simulation summary

Run

```
make summary
```

Install pandoc if you don't have it

Run

```
pandoc -s  -t slidy README.md -o README.html
```

to generate a HTML slideshow that you can open in browser. Open the HTML file.

6.3.7.5 Viewing results without GUI browser

If your 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
```

6.3.7.6 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 +- 20 %. I'll leave it up to you to fix it.

6.3.8 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/JNW_EX_SKY130A/JNW_EX.mag
```

Now brace yourself, Magic VLSI was created in the 1980's. For it's time it was extremely modern, however, today it seems dated. However, it is free, so we use it.

6.3.8.1 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

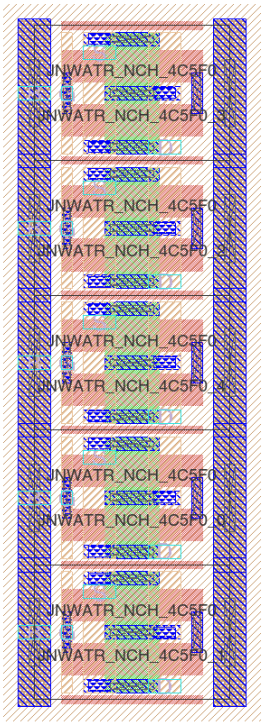
Hotkey	Function
v	View all
shift-z	zoom out
z	zoom in
x	look inside box (expand)
shift-x	don’t look inside box (unexpand)
u	undo
d	delete
s	select
Shift-Up	Move cell up
Shift-Down	Move cell down
Shift-Left	Move cell left
Shift-Right	Move cell right

6.3.8.2 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.



6.3.8.3 Add Ground

In the command window, type

```
see no *
see viali
see locali
see m1
see via1
see m2
```

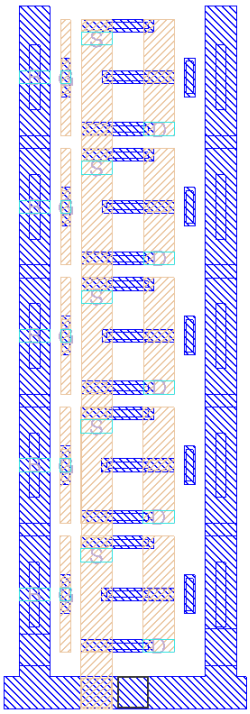
Change to the ‘wire tool’ with spacebar. Press the top transistor ‘S’ and draw all the way down to connect all of the transistors’ source terminals.

Change grid to 0.5 um.

Select a 0.5 um box below the transistors and paint the rectangle with locali (middle click on locali)

Connect guard rings to ground. Use the ‘wire tool’

Connect the sources to ground. Use the ‘wire tool’. Use ‘shift-right click’ to change layer down

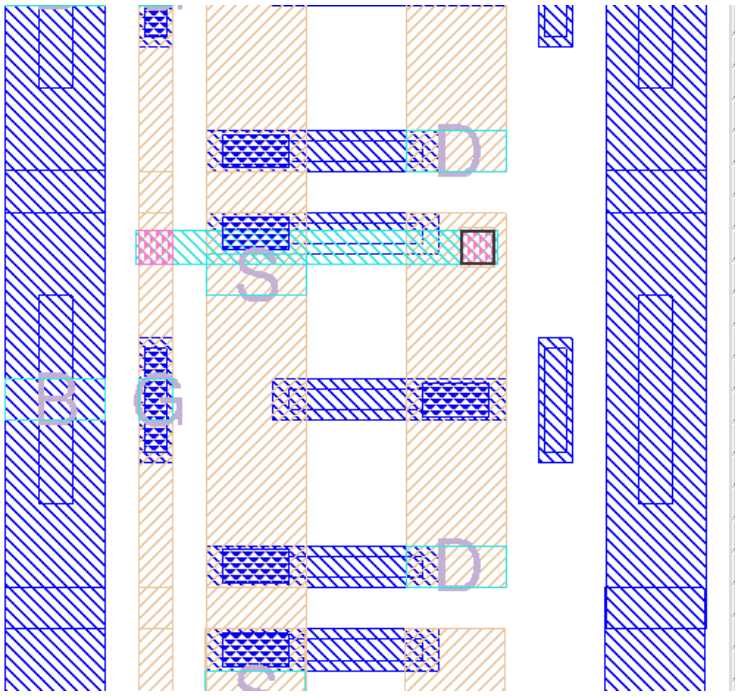


6.3.8.4 Route Gates

Press “space” to enter wire mode. Left click 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.

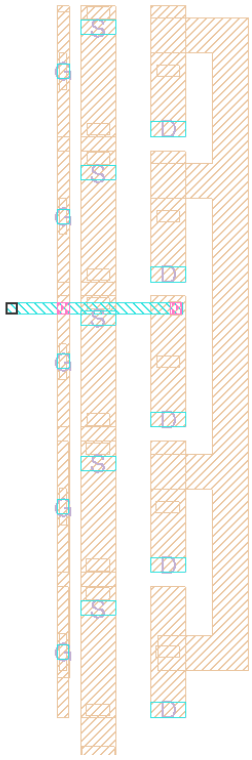
Start the route, press ‘shift-left click’ to go up one layer, route over to drain, and ‘shift-right click’ to go down.



6.3.8.5 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.



6.3.8.6 Add labels

Select a box on a metal, and use “Edit->Text” to add labels for the ports. Select the port button.

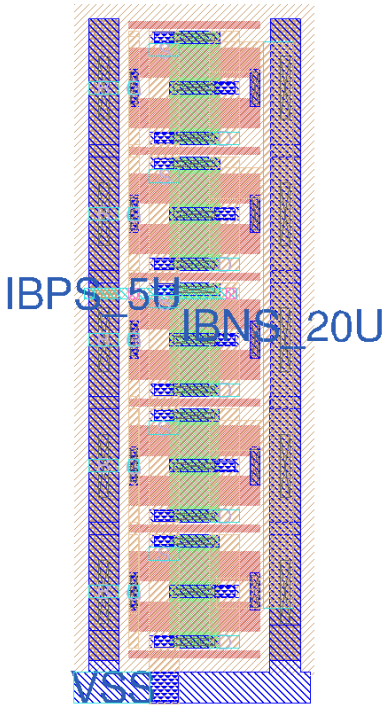
6.3.9 Layout verification

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
```

If you’ve routed correctly, then the LVS should be correct.



6.3.10 Extract layout parasitics

With the layout complete, we can extract parasitic capacitance.

```
make lpe
```

Check the generated netlist

```
cat lpe/JNW_EX_lpe.spi
```

6.3.11 Simulate with layout parasitics

Navigate to sim/JNW_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
```

6.3.11.1 Typical simuation

Run

```
make typical
```

6.3.11.2 Corners

Navigate to sim/JNW_EX. Run all corners again

```
make all
```

6.3.11.3 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  -t slidy 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.

6.3.12 Make documentation

Make a file (or it may exists) `design/JNW_EX_SKY130A/JNW_EX.md` and add some docs.

6.3.13 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: JNW_EX_SKY130A
cell: JNW_EX
author: Carsten Wulff
github: wulffern
tagline: The answer is 42
email: carsten@wulff.no
url: analogicus.github.io
doc:
  libraries:
    JNW_EX_SKY130A:
      - JNW_EX
sim:
  JNW_EX: make typical
```

6.3.14 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://analogicus.github.io/jnw_ex0_sky130a/.

6.3.15 Frequency 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 ../JNW_ATR_SKY130A
```

It's the last `../JNW_ATR_SKY130A` that sometimes is missing.

Analog Design 7

Status: 0.5

7.1 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.

7.1.1 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		

Item	Description	Yes	Action
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.		

7.1.2 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)		

7.1.3 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)		

7.2 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.

7.2.0.1 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?

7.2.0.2 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

7.2.0.3 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.

7.2.0.4 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”

7.2.0.5 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.

7.3 Layout rules

7.3.0.1 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 regulator. A bunch of different Ls and Ws is a pain when you do layout

7.3.0.2 Use pre-defined transistors for regular layout

Do: Use JNW_ATR_SKY130A and JNW_TR_SKY130A

7.3.0.3 Always use two fingers for analog transistors

That way, you don’t have to worry about current direction in the layout.

7.3.0.4 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.

7.3.0.5 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.

7.3.0.6 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.

IC and ESD 8

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

Status: 1.0

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.

8.1 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.

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.

8.1.0.1 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) provide the digital supply (1.8 V).

See more at Analog Specs

8.1.0.2 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.

8.1.0.3 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.

8.1.0.4 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.

8.1.0.5 Bias

The TinyTapeout does not provide bias currents (that I found), so that is something you will need to make.

8.1.0.6 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.

8.1.0.7 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 compared 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).

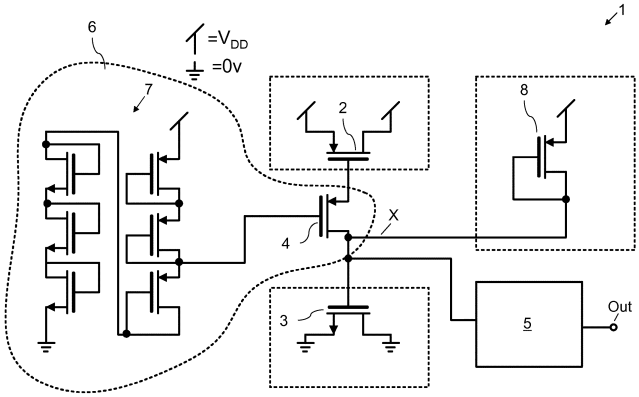


Figure 1

Figure 1: Power On Reset using gate tunneling

8.2 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, shown below, and rely on you understanding of PN-junctions.

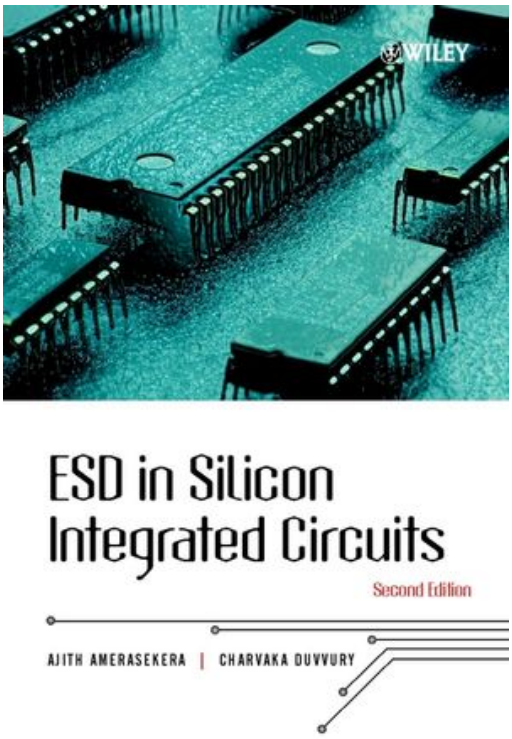


Figure 2: A book on ESD

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.

8.2.1 Human body model (HBM)

HBM is the "simple" version of ESD, a model can be seen in Figure 3. 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

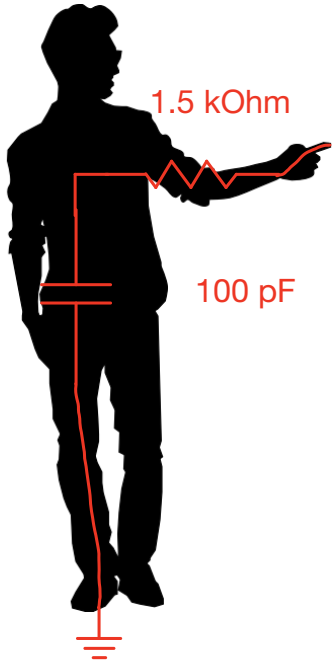


Figure 3: Human body model (HBM)

More on circuits that protect from HBM later.

8.2.2 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-Norheim tunneling to charge and discharge a capacitor that keeps it's 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 it's back, place it on an metal plate with an insulator in-between, and charge the metal plate to 1 kV, as shown in Figure 4.

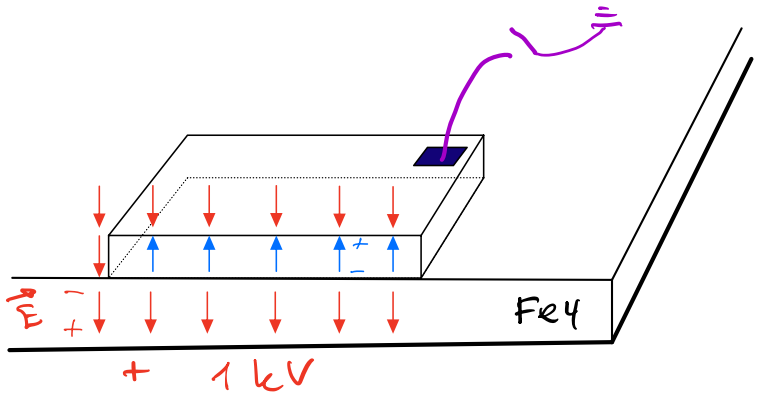


Figure 4: Charged Device Model (CDM) testing

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 lighting 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?

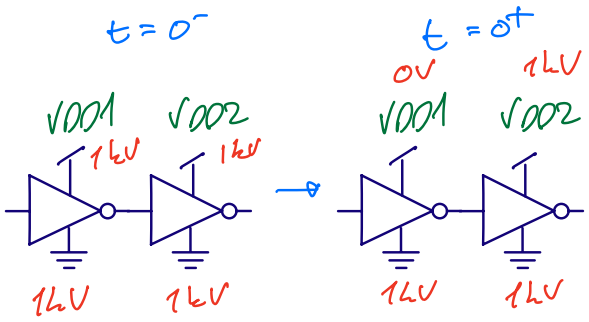


Figure 5: Cross domain voltage problem with CDM (or indeed HBM) events

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 will 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.

8.3 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.

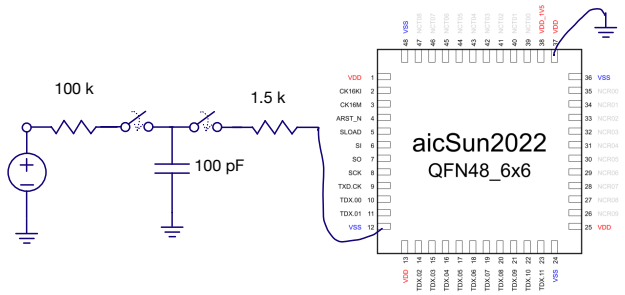


Figure 6: ESD HBM zap example

Let's simplify and think of the possible permutations, shown in Figure 7. 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.

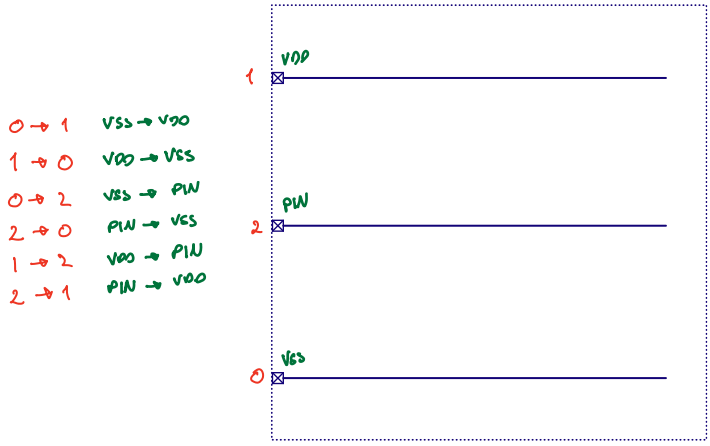


Figure 7: ESD zap permutations

When the current enters VSS and must leave via VDD, then it's simple, we can use a diode, as shown in Figure 8.

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.

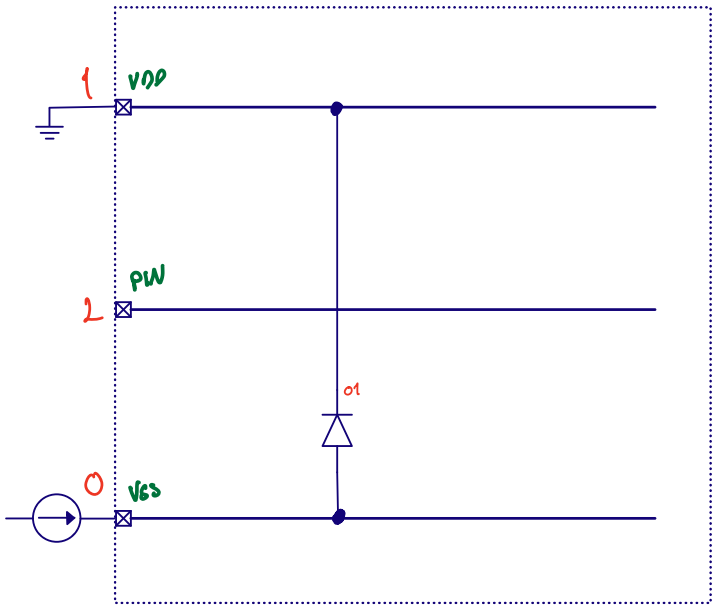


Figure 8: Protection for zap from ground to VDD

The same is true for current in on VSS and out on PIN. Here we can also use a diode, as shown in Figure 9.

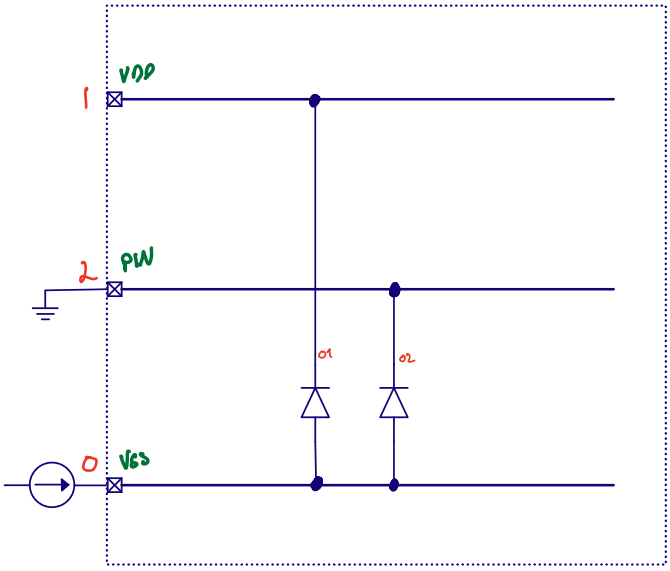


Figure 9: Protection for zap from ground to PIN

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 your 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 10.

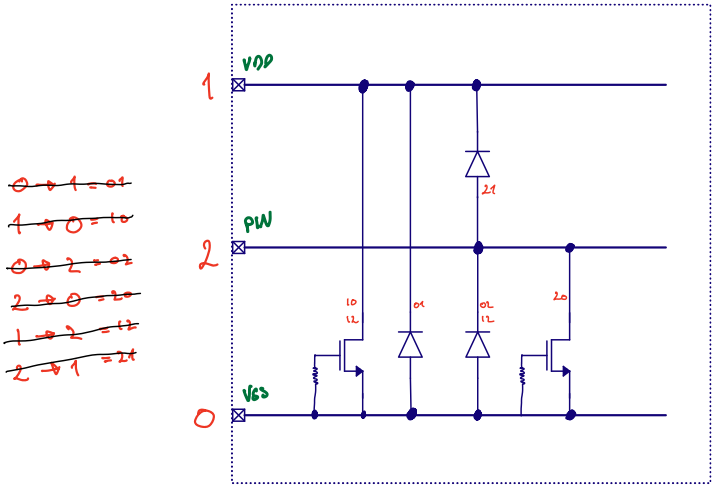


Figure 10: Full protection with diodes and grounded gate NMOS (GGNMOS)

8.4 The grounded gate NMOS

If you try the circuit in Figure 11 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.

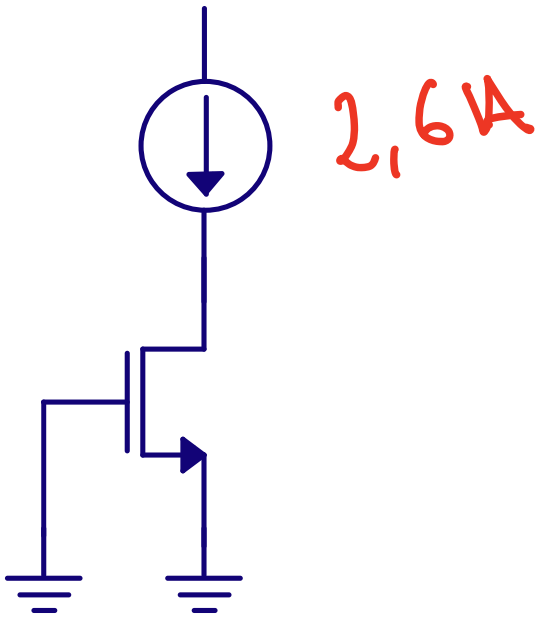


Figure 11: The grounded gate NMOS (GGNMOS)

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 12. The gate, source and bulk is connected to ground. The drain is connected to a high voltage.

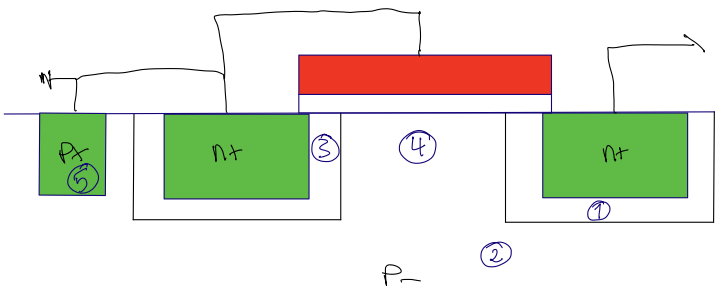


Figure 12: Cross section of the grounded gate NMOS

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 of 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 (unscilicided 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 for the pretty pictures you'll get when the drain/source breaks.

8.5 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 circutis. 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.

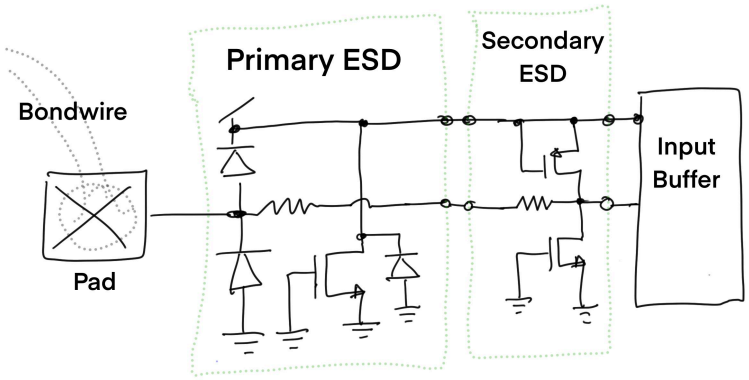


Figure 13: Full protection of an input including secondary protection

8.5.1 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. _

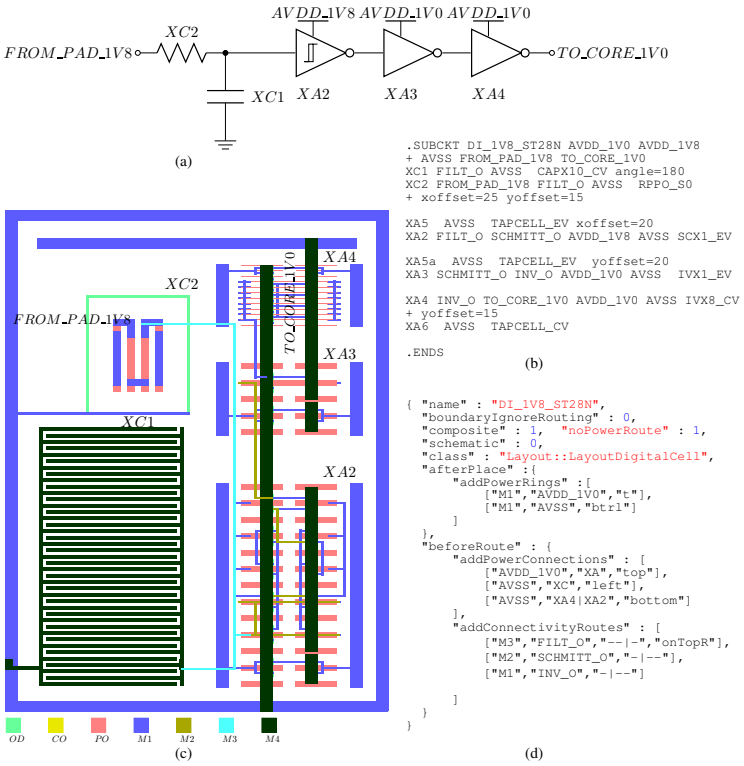


Figure 14: Full digital input including Schmitt trigger and level shifters

8.6 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 15. Under certain conditions we can get a short from VDD to ground.

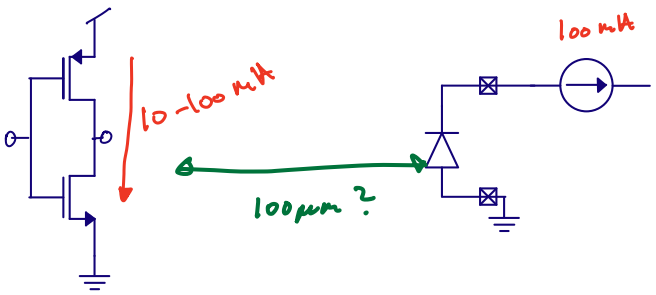


Figure 15: Inverter that suddenly shorts from VDD to ground located close to a PAD

Consider the cross section of the inverter in Figure 16. 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 .

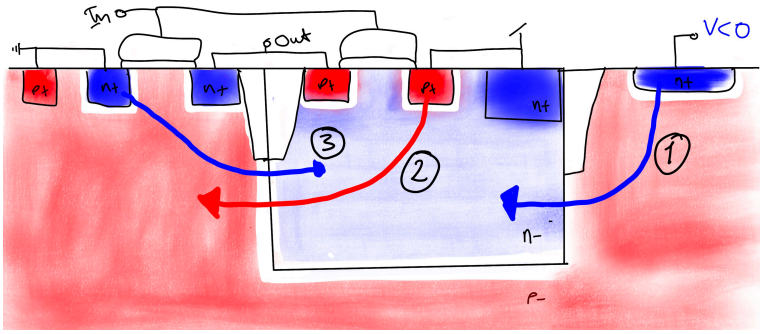


Figure 16: Cross section of an inverter

8.6.0.1 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.

8.6.0.2 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.

8.6.0.3 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).

8.6.0.4 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 phenomena 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 to 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 17. 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.

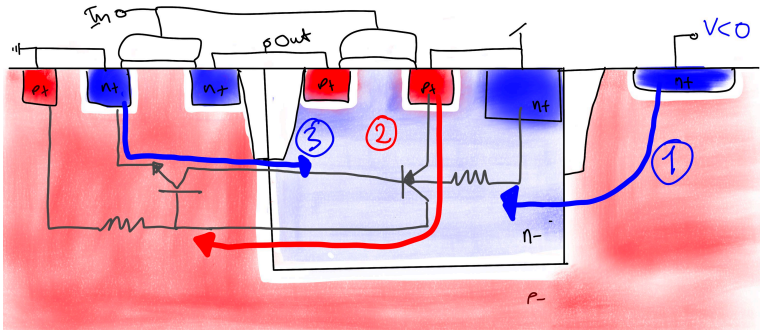


Figure 17: Cross section of an inverter including the parasitic bipolars

You must **always handle ESD** on an IC

- Do everything yourself
- Use libraries from foundry
- Get help www.sofics.com

8.7 Want to learn more?

ESD (Electrostatic Discharge) Protection Design for Nanoelectronics in CMOS Technology

Overview on Latch-Up Prevention in CMOS Integrated Circuits by Circuit Solutions

Overview on ESD Protection Designs of Low-Parasitic Capacitance for RF ICs in CMOS Technologies

9 References and bias

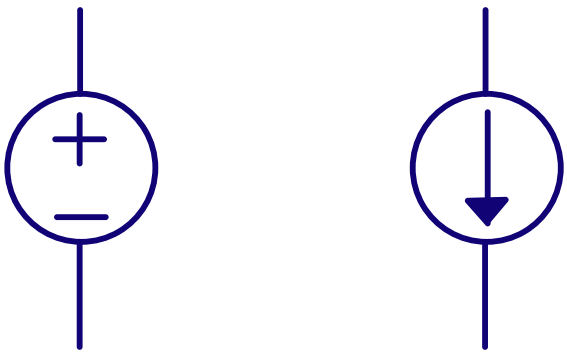
Keywords: VREF, IREF, VD, BGAP, LVBGAP, VI, GMCELL

Status: 0.5

In our testbenches, and trial schematics, it's common to include voltage sources and current sources. However, 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.

So how do we create something that is a *good enough* voltage and current source on an IC?



9.1 Routing

Before we take a 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.

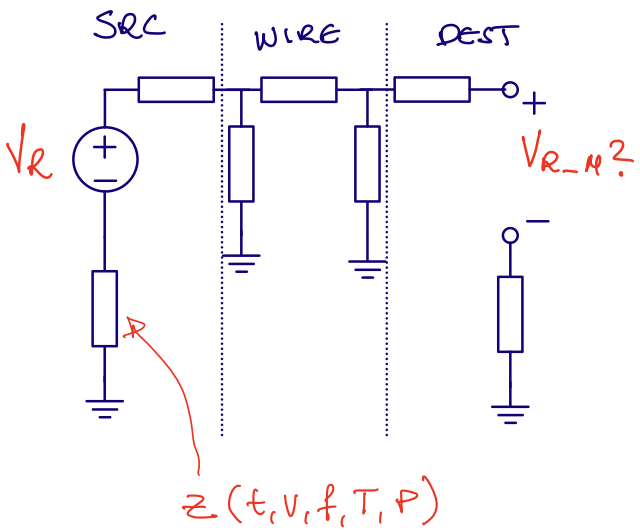
Assume we have a known voltage on our IC. How can we make sure we can share that voltage across an IC?

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*.

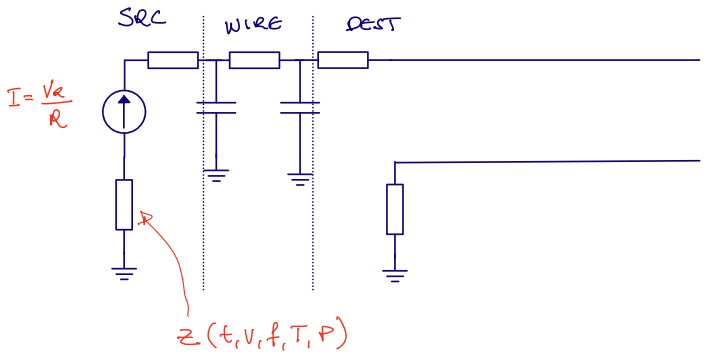
We've invented this magical place called *ground*, the final resting place of all electrons, and we have agreed that all voltages refer to that point.

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.



Most of the time, in order not to think about the ground impedance, we choose to route a known quantity 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.



Resistors have finite matching across die, let's say 2 % 3-sigma variation. As a result, if we need a accurate voltage reference, then we must distribute voltage.

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?

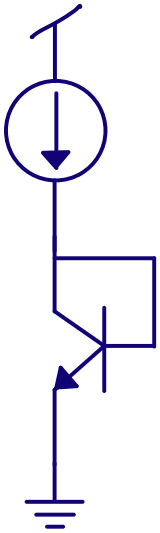
Imagine I have a 0.5 % 3-sigma accurate voltage reference at 1.22 V, that's a sigma of 2 mV. I need this reference voltage on a block on the other side of the IC, I don't want to distribute voltage, because I don't know that the ground is the same on the other side, at least not to a precision of 2 mV. 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 is two references?" And the answer is "Silicon area is expensive, to make things cheep, we must make things small", in other words, we should not duplicate features unless we absolutely have to.

9.2 Bandgap voltage reference

9.2.1 A voltage complementary to temperature (CTAT)



A diode connected bipolar transistor, or indeed a PN diode, assuming a fixed current, will have a voltage across that is temperature dependent

$$I_D = I_S \left(e^{\frac{V_{BE}}{V_T}} - 1 \right) + I_B \approx I_S e^{\frac{V_{BE}}{V_T}}$$

As I_S is much smaller that I_D we can ignore the -1, and we assume that the base current is much smaller than the drain 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 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 the bandgap voltage of silicon (approx 1.12 eV).

To see the temperature coefficient, I find it easier to re-arrange the equation above.

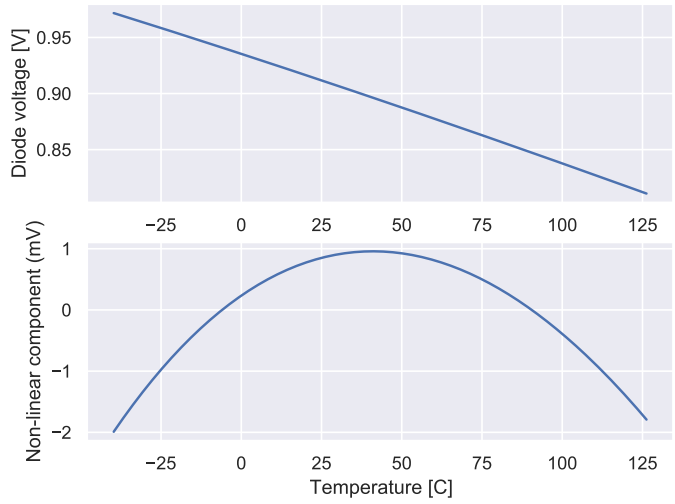
Some algebra (see Diodes)

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

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.



9.2.2 A current proportional to temperature (PTAT)

If we take two diodes, or bipolars, biased at different current densities, as shown in the figure below, 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

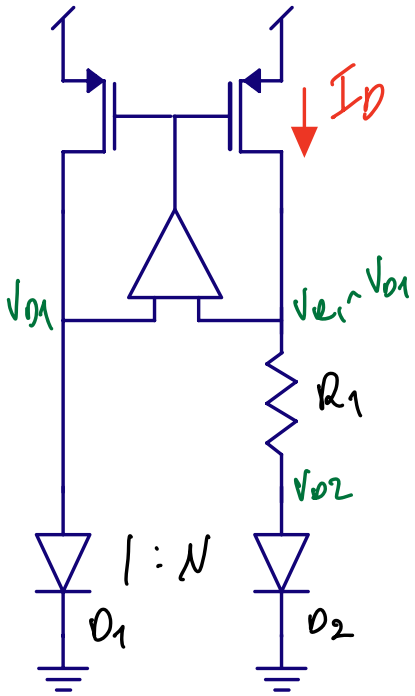
$$V_{D1} - V_{D2} = V_T \ln \frac{I_D}{I_{S1}} - V_T \ln \frac{I_D}{I_{S2}} = V_T \ln \frac{I_{S2}}{I_{S1}} = V_T \ln N$$

This is a remarkable result. The difference between two voltages is only defined by boltzmann's constant, temperature, charge, and a know size difference.

This differential voltage can be used to read out directly the temperature on an IC, provided we have a known voltage to compare with.

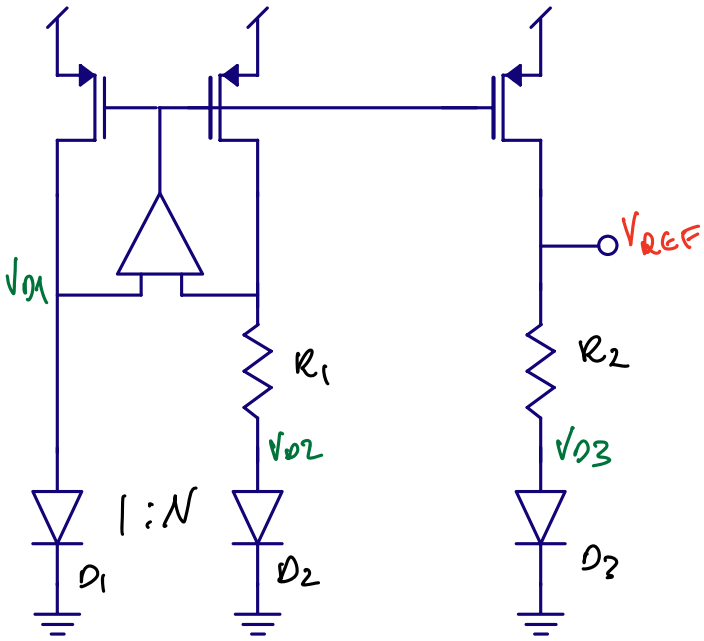
We often call this voltage ΔV_D or ΔV_{BE}, and we can clearly see it's proportional to absolute temperature.

We know that the V_D decreases linearly with temperature, so if we combined a multi-plum of the ΔV_{BE} with a V_D voltage, then we should get a constant voltage.

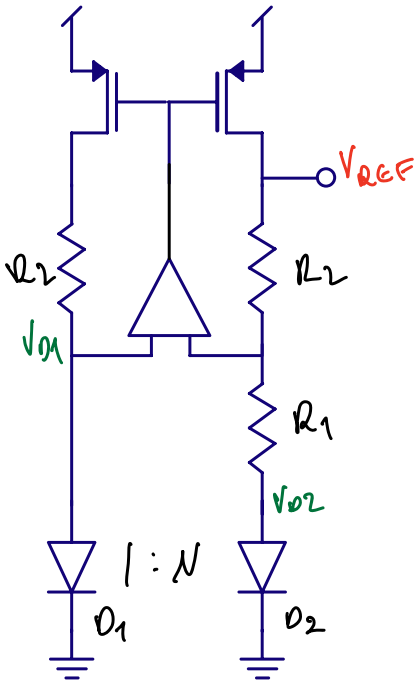


9.2.3 How to combine a CTAT with a PTAT ?

One method is the figure below. The voltage across resistor R_2 would compensate for the decrease in V_{D3}, as such, R_2 would be bigger than R_1.



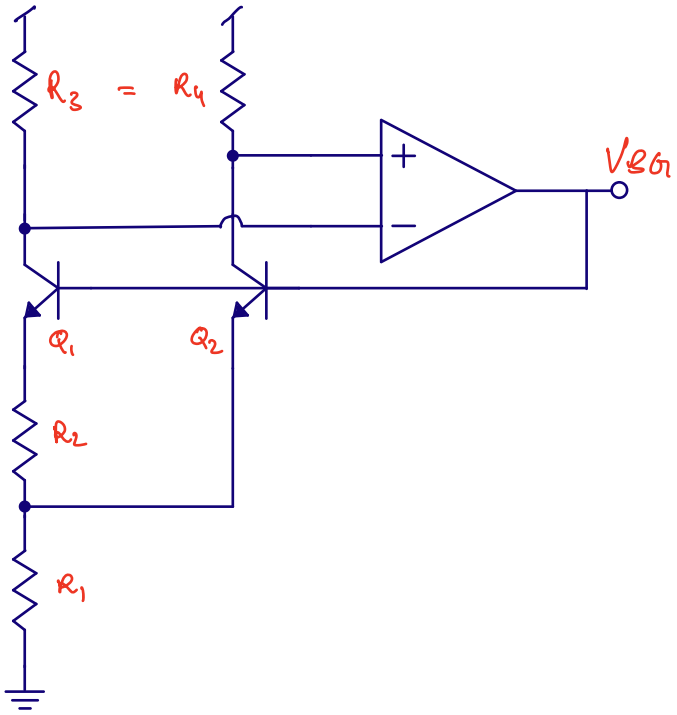
Another method would be to stack the R_2 on top of R_1 as shown below.



9.2.4 Brokaw reference

Paul Brokaw was a pioneer within reference circuits. Below is the Brokaw reference, which I think was first published in A simple

three-terminal IC bandgap 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.

$$V_{BG} = V_{G0} + (m-1)\frac{kT}{q}\ln\frac{T_0}{T} + T\left[\frac{k}{q}\ln\frac{J_2}{J_1}\frac{2R2}{R1} - \frac{V_{G0}-V_{be0}}{T_0}\right]$$

where V_{G0} is the bandgap, V_{be0} is the base emitter measured at a temperature T_0 and the J's are the current densities.

To get a constant output voltage, the relationship between the resistors should be approximately

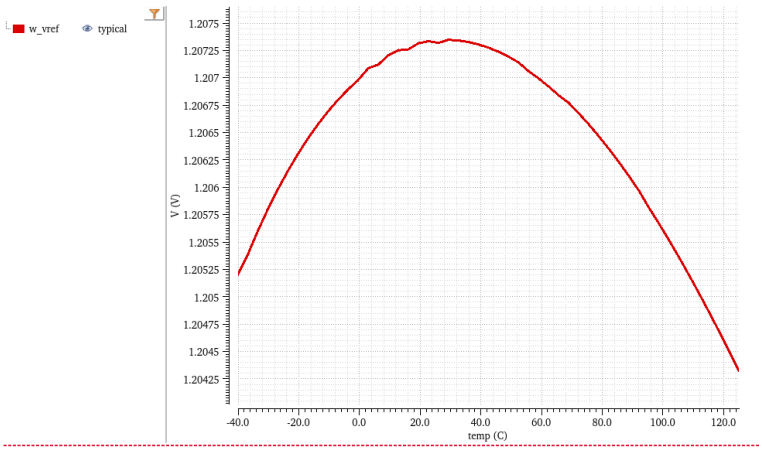
$$\frac{R2}{R1} = \frac{V_{G0}-V_{be0}}{2T_0\frac{k}{q}\ln(\frac{J_2}{J_1})}$$

In typical simulations, the variation can be low over the temperature range. The second order error is the remaining error from

$$V_{BG} = V_{G0} + (m-1)\frac{kT}{q}\ln\frac{T_0}{T} + T\left[\frac{k}{q}\ln\frac{J_2}{J_1}\frac{2R2}{R1} - \frac{V_{G0}-V_{be0}}{T_0}\right]$$

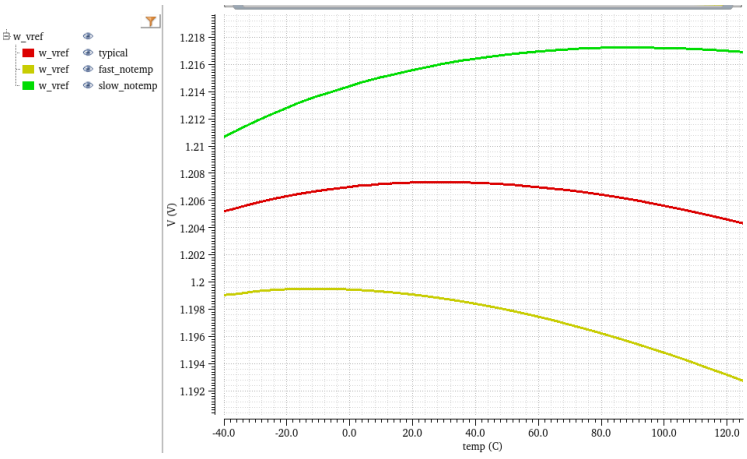
Where the last term is zero, so

$$V_{BG} = V_{G0} + (m-1)\frac{kT}{q}\ln\frac{T_0}{T}$$



Over corners, I do expect that there is variation. It may be that the V_D modeling is not perfect, which means the cancellation of the last term is incomplete.

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 too expensive for low-cost devices.



9.2.5 Low voltage bandgap

The Brokaw reference, and others, have a 1.2 V output voltage, which is hard 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

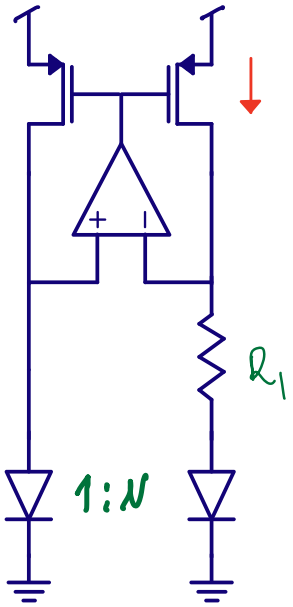
In real ICs though, you should ask yourself long and hard whether you really need these low-voltage references. Most ICs today still have a high voltage, either 1.8 V or 3.0 V.

If you do need them though, consider the circuit below. 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 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.



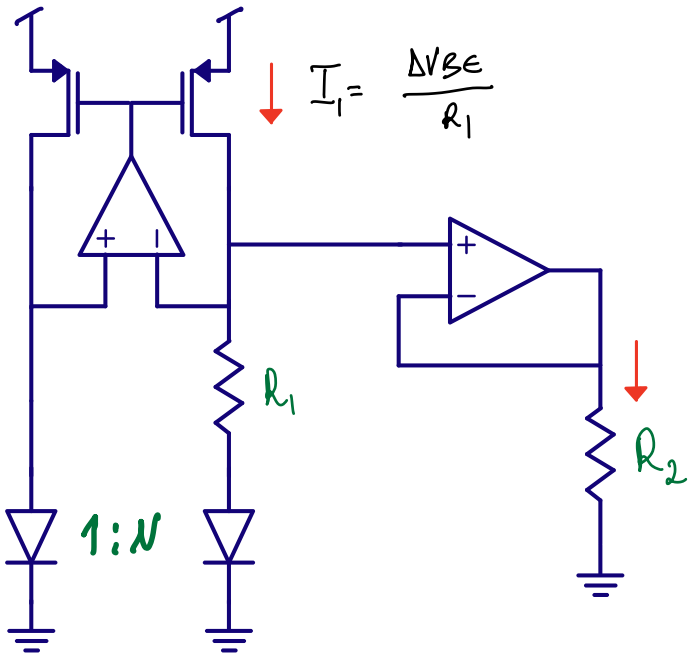
In the figure below I've used ΔV_{BE}, it's the same as ΔV_D, so ignore that error.

Assume we copy the V_D to another node, and place it across a second resistor R_2, as shown in the figure below. The current in this second resistor is then

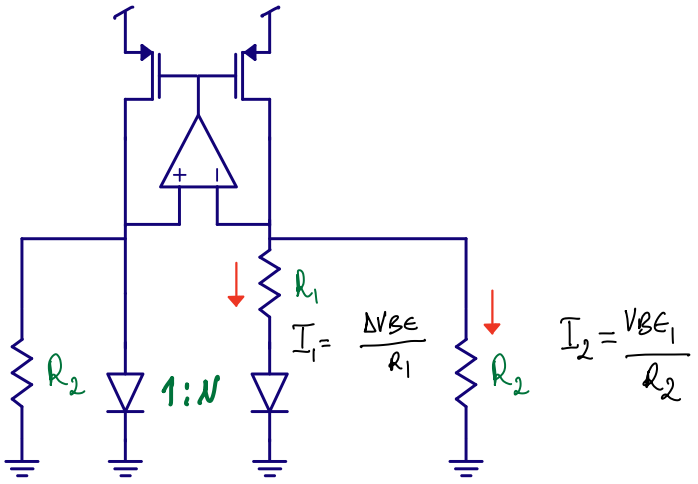
$$I_2 = \frac{V_D}{R_2}$$

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 with the correct proportions, then we can get a current that does not change with temperature.



Let's remove the OTA, and connect R_2 directly to V_D nodes, you should convince yourself of the fact that this does not change I_1 at all.



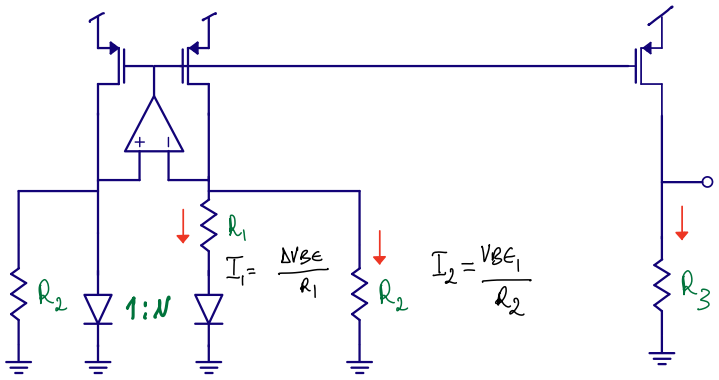
It does, however, change the current in the PMOS. Provided we scale R_2 correctly, then the PTAT I_1 can be compensated by the CTAT I_2, and we have a current that is independent of temperature.

$$I_{PMOS} = \frac{V_D}{R_2} + \frac{\Delta V_D}{R_1}$$

Assuming we copy the current into another resistor R_3, as shown below, we can get a voltage that is

$$V_{OUT} = R_3 \left[\frac{V_D}{R_2} + \frac{\Delta V_D}{R_1} \right]$$

Where the output voltage can be chosen freely, and indeed be lower than 1.2 V.



9.3 Bias

Sometimes we just need a current

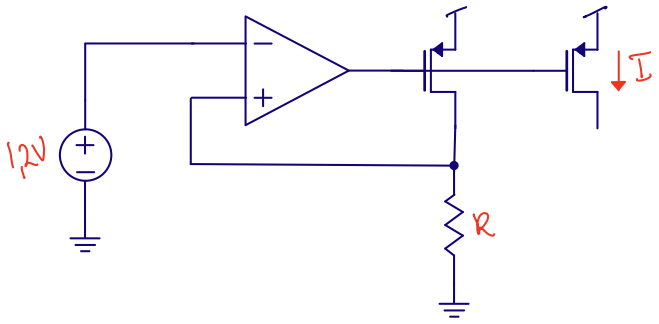
9.3.1 Voltage to current conversion

With a known voltage, we can convert to a known current with the circuit below.

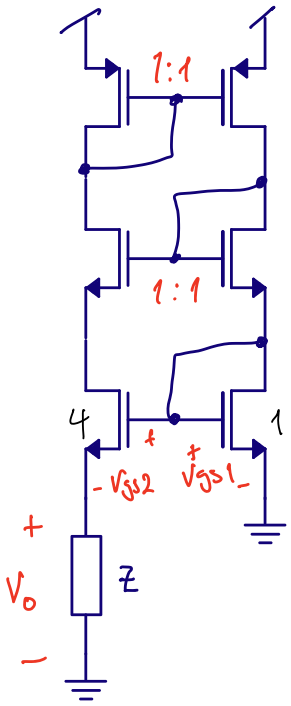
On-chip we don't have accurate resistors, but for bias currents, it's usually ok with + − 20 variation (the variation of R).

Across a IC, we can expect the resistors to match within a few percent, as such, we can recreate a voltage with a accuracy of a few percent 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 until the current is what we want.



9.3.2 GM Cell



Sometimes we don't need a full bandgap reference. In those cases, we can use a GM cell, where the impedance could be a resistor, in which case

$$V_o = V_{GS1} - V_{GS2} = V_{eff1} + V_{tn} - V_{eff2} - V_{tn} = V_{eff1} - V_{eff2}$$

Assuming 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 strong inversion, such that

$$g_m = \frac{2I_d}{V_{eff}}$$

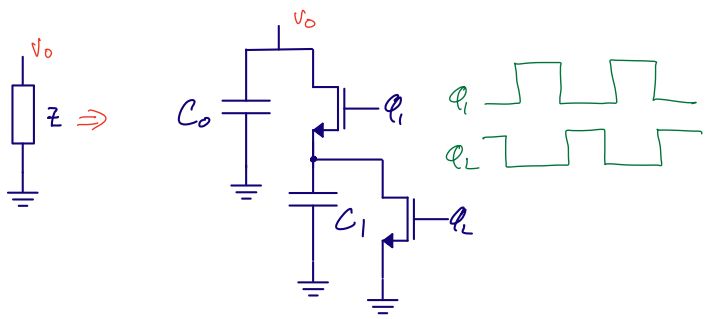
we find that

$$I = \frac{V_{eff1}}{2Z}$$

$$Z \Rightarrow \frac{1}{g_m}$$

If we use a resistor for Z, then we can get a transconductance that is proportional to a resistor, or a constant g_m bias.

We can use other things for Z, like a switched capacitor



9.4 Want to learn more?

A simple three-terminal IC bandgap reference

A CMOS bandgap reference circuit with sub-1-V operation

A sub-1-V 15-ppm//spl deg/C CMOS bandgap voltage reference without requiring low threshold voltage device

The Bandgap Reference

The Design of a Low-Voltage Bandgap Reference

Analog frontend and filters 10

Keywords: H(s**, BiQuad, Gm-C, Active-RC, OTA

Status: 0.5

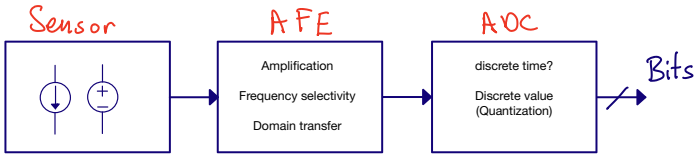
10.1 Introduction

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.



One example of an analog frontend is the recieve 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 ususally sufficient with low number of bits, assume 8-bits for the signal is more than enough.

If we assume the maximum of the ADC should be the blocker in the table below, and the resolution of the digital should be given by

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

$$\text{ADC resolution} \implies \ln \frac{7\text{ mV}}{28\text{ nV}} / \ln 2 \approx 18 \text{ bits}$$

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

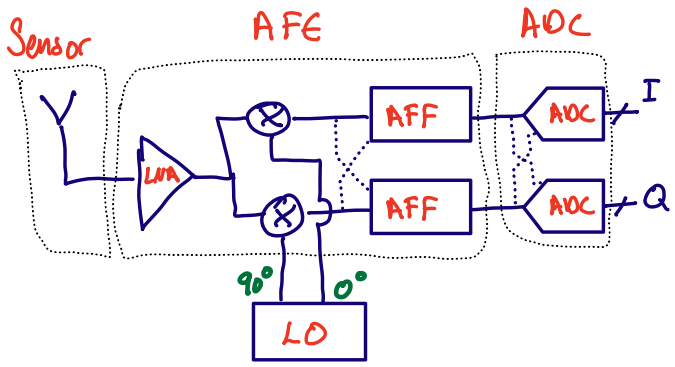
$$\text{Hours} = \frac{205 \text{ mAh}}{1.32 \text{ W}/3.8 \text{ V}} = 0.6 \text{ h}$$

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.



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.

10.2 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

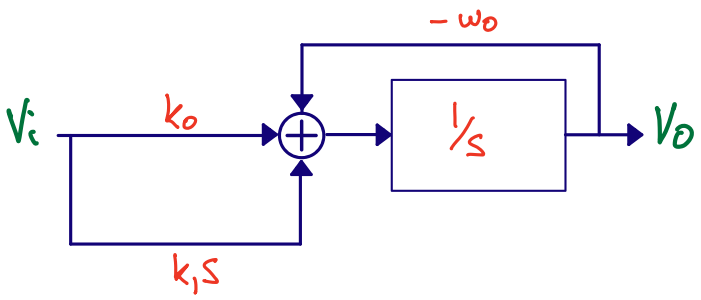
combination of 1'st and 2'nd order stages can synthesize any order filter.

Once we have the first and second order stages, we can start looking into circuits.

10.2.1 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.



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

$$u = (k_0 + k_1 s)V_i - \omega_0 V_o$$

$$V_o = u/s$$

$$u = V_o s = (k_0 + k_1 s)V_i - \omega_0 V_o$$

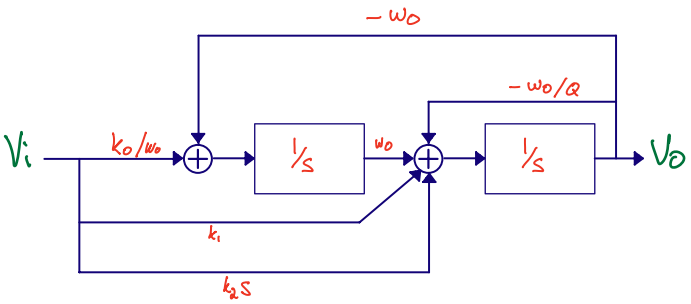
$$(s + \omega_0)V_o = (k_0 + k_1 s)V_i$$

$$\frac{V_o}{V_i} = \frac{k_1 s + k_0}{s + \omega_0}$$

10.2.2 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.



$$H(s) = \frac{k_2 s^2 + k_1 s + k_0}{s^2 + \frac{\omega_0}{Q} s + \omega_o^2}$$

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.

10.2.3 How do we implement the filter sections?

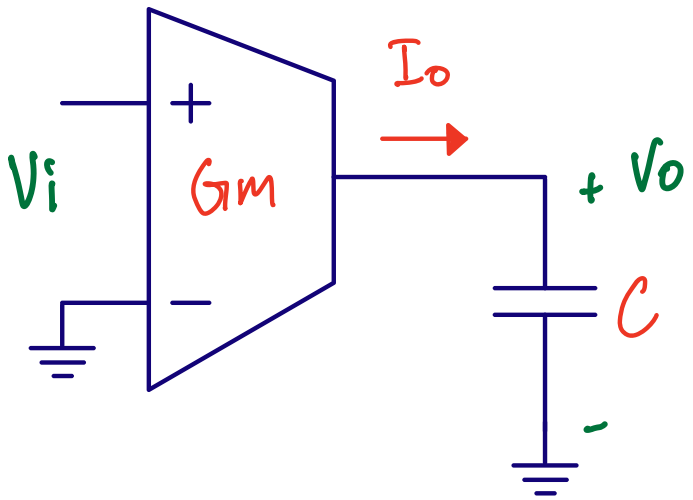
While I'm sure you can invent new types of filters, and there probably are advanced filters, I would say there is roughly three types. Passive filters, that do not have gain. Active-RC filters, with OTAs, and usually trivial to make linear. And Gm-C filters, where we use the transconductance of a transistor 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.

10.3 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.

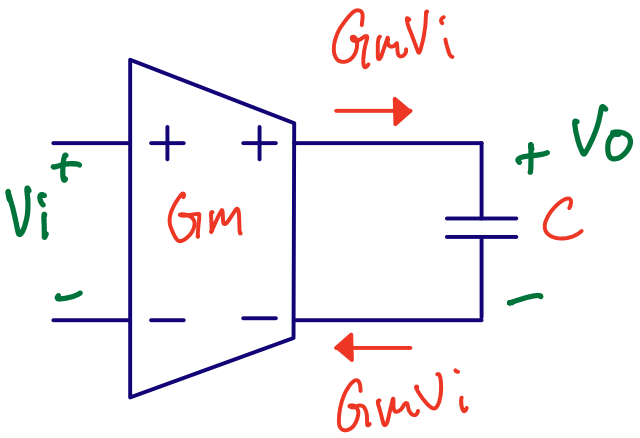


$$V_o = \frac{I_o}{sC} = \frac{\omega_{ti}}{s} V_i$$

$$\omega_{ti} = \frac{G_m}{C}$$

10.3.1 Differential Gm-C

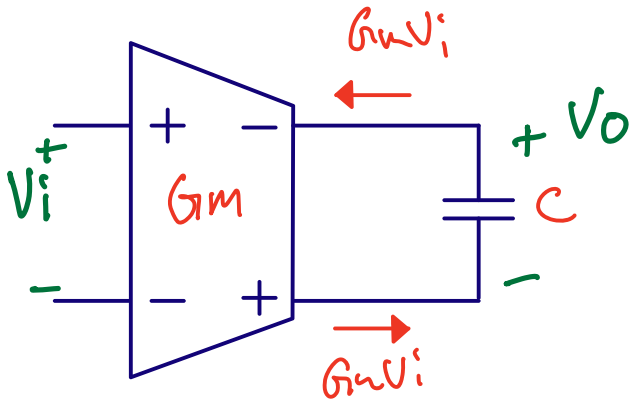
In a real IC we would almost always use differential circuit, as shown below. The transfer function is luckily the same.



$$sCV_o = G_m Vi$$

$$H(s) = \frac{V_o}{V_i} = \frac{G_m}{sC}$$

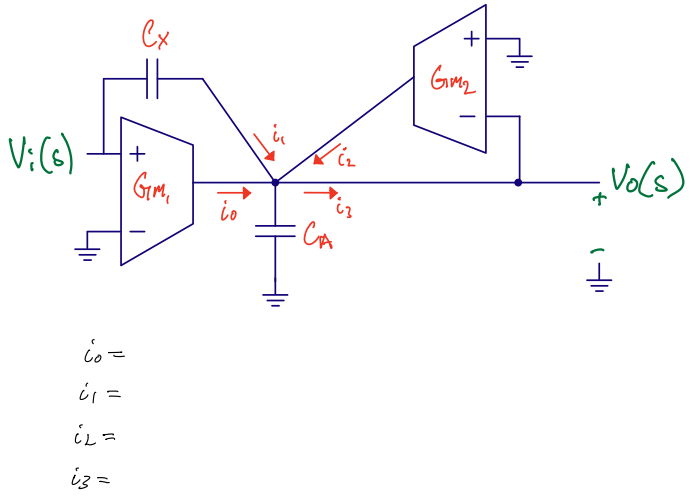
Differential circuits are fantastic for multiple reasons, power supply rejection ratio, noise immunity, symmetric non-linearity, but the qualities I like the most is that the outputs can be flipped to implement negative, or positive gain.



$$H(s) = \frac{V_o}{V_i} = -\frac{G_m}{sC}$$

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.

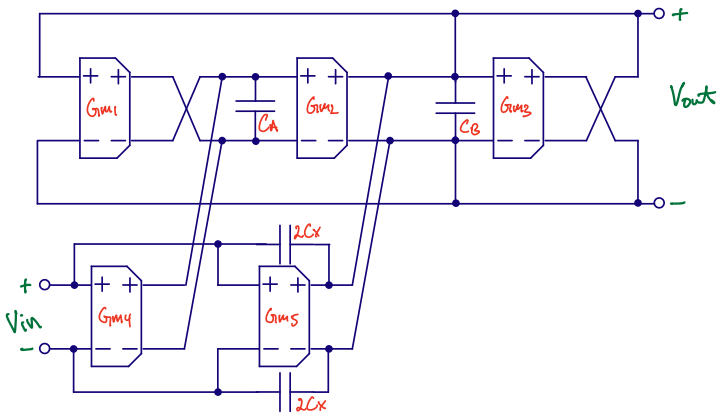


Given the transfer function from the signal flow graph, we see that we can select C_x, C_a and G_m to get the desired k's and ω_0

$$H(s) = \frac{k_1 s + k_0}{s + w_o}$$

$$H(s) = \frac{s\frac{C_x}{C_a+C_x} + \frac{G_{m1}}{C_a+C_x}}{s + \frac{G_{m2}}{C_a+C_x}}$$

Below is a general purpose Gm-C bi-quadratic system.



$$H(s) = \frac{k_2 s^2 + k_1 s + k_0}{s^2 + \frac{\omega_0}{Q} s + \omega_o^2}$$

$$H(s) = \frac{s^2 \frac{C_X}{C_X+C_B} + s\frac{G_{m5}}{C_X+C_B} + \frac{G_{m2}G_{m4}}{C_A(C_X+C_B)}}{s^2 + s\frac{G_{m2}}{C_X+C_B} + \frac{G_{m1}G_{m2}}{C_A(C_X+C_B)}}$$

10.3.2 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

And from Figure 10 a) we can see it's a similar Gm-C cell as chapter 12.5.4 in CJM.

One of my Ph.d's used the transonductor below on his master thesis Design Considerations for a Low-Power Control-Bounded A/D Converter.

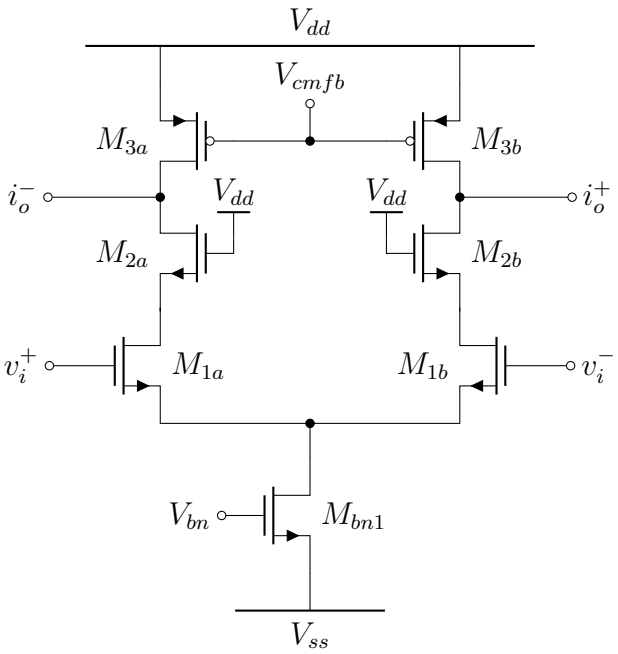


Figure 5.16: Transconductor schematic

10.4 Active-RC

The Active-RC filter should be well know at this point. However, what might be new is that the open loop gain A_0 and unity gain ω_{ta}

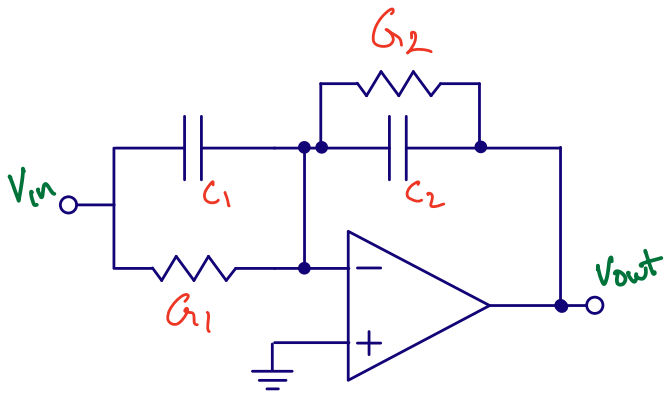
10.4.1 General purpose first order filter

Below is a general purpose first order filter and the transfer function. I've used the condutance $G = \frac{1}{R}$ 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.



$$H(s) = \frac{k_1 s + k_0}{s + w_o}$$

$$H(s) = \frac{-\frac{C_1}{C_2} s - \frac{G_1}{C_2}}{s + \frac{G_2}{C_2}}$$

Let's work through the calculation.

10.4.1.1 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

$G_{fb} = G_2 + sC_2$

10.4.1.2 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.

10.4.1.3 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 have not really checked that the OTA works porperly in the filter because "Someone else designed the OTA, I just used in my design".

But, for now, to make our lifes 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.

10.4.1.4 Step 4: Do the algebra

The current flowing from V_{out} to virtual ground is

$$I_{out} = G_{fb} V_{out}$$

The sum of currents into the virtual ground must be zero

$$I_{in} + I_{out} = 0$$

Insert, and do the algebra

$$G_{in} V_{in} + G_{out} V_{out} = 0$$

$$\Rightarrow -G_{in} V_{in} = G_{out} V_{out}$$

$$\frac{V_{out}}{V_{in}} = -\frac{G_{in}}{G_{out}}$$

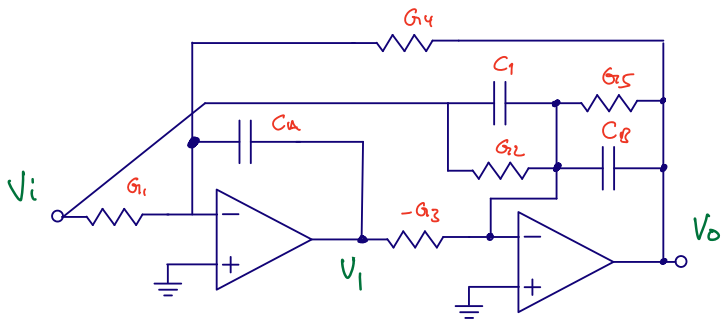
$$= -\frac{G_1 + sC_1}{G_2 + sC_2}$$

$$= \frac{-s\frac{C_1}{C_2} - \frac{G_1}{C_2}}{s + \frac{G_2}{C_2}}$$

10.4.2 General purpose biquad

A general bi-quadratic active-RC filter is shown below. These kind of general purpose filter sections are quite useful.

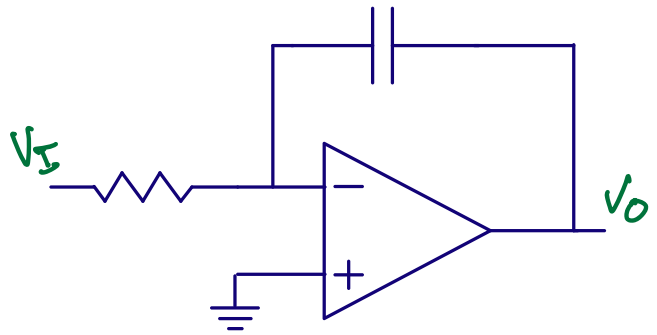
Imagine you wanted to make a filter, any filter. You'd decompose into first and second order sections, and then you'd try and match the transfer functions to the general equations.



$$H(s) = \frac{k_2 s^2 + k_1 s + k_0}{s^2 + \frac{\omega_0}{Q} s + \omega_o^2}$$

$$H(s) = \frac{\left[\frac{C_1}{C_B} s^2 + \frac{G_2}{C_B} s + (\frac{G_1 G_3}{C_A C_B})\right]}{\left[s^2 + \frac{G_5}{C_B} s + \frac{G_3 G_4}{C_A C_B}\right]}$$

10.5 The OTA is not ideal



$$H(s) \approx \frac{A_0}{(1 + sA_oRC)(1 + \frac{s}{w_{ta}})}$$

where

$$A_0$$

is the gain of the amplifier, and

$$\omega_{ta}$$

is the unity-gain frequency.

At frequencies above $\frac{1}{A_0RC}$ and below w_{ta} the circuit above is a good approximation of an ideal integrator.

See page 511 in CJM (chapter 5.8.1)

10.6 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 us 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 fantasitc 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.

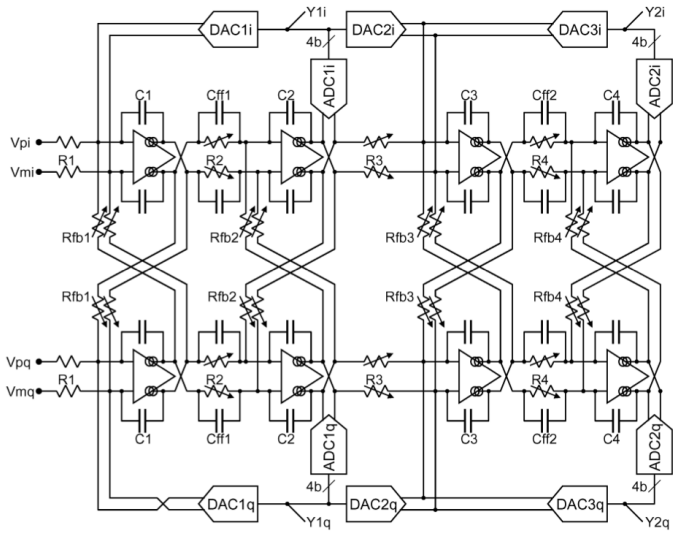
With a complex ADC like this, the first thing to understand is the rough structure.

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 sence.

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 $V(Vpi, Vpm)$ to ADC1i input is large, then $Y1i = V(Vpi, Vpm)$ at low frequencies.



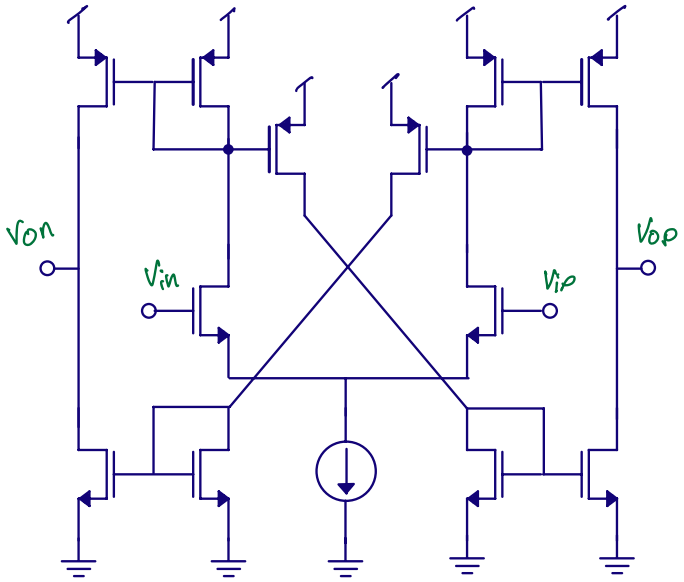
igma-delta modulator design.

10.7 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.

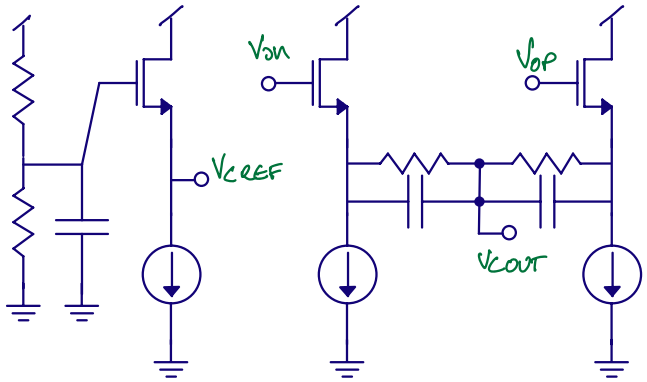
Below is the differential current mirror OTA.



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.

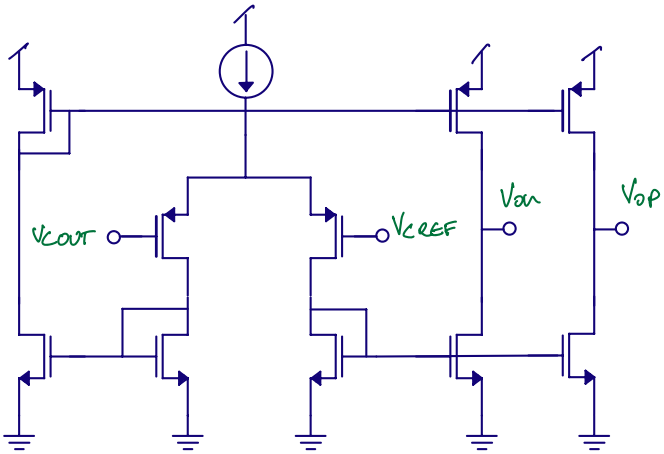
Below is a circuit I often use to sense the common mode. Ideally the source followers would be native transistors, but those are not always available.

The reference for the common mode can be from a bandgap, or in the case below, VDD/2.



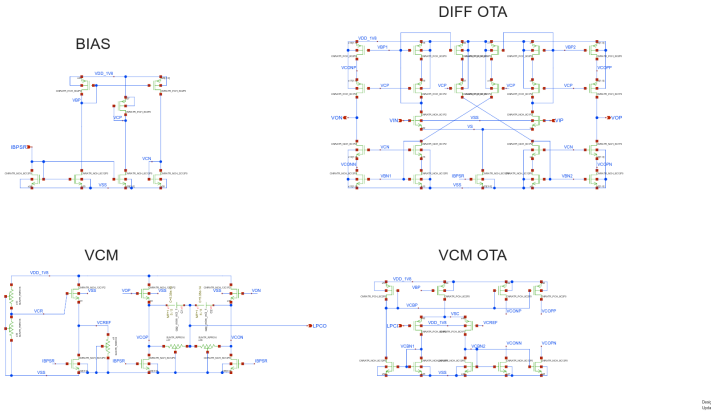
Once we have both the sensed common mode, and the common mode reference, we can use another OTA to control the common mode.

The nice thing about the circuit below is that the common mode feedback loop has the same dominant pole as the differential loop.



You can find the schematic for the OTA at

CNR_OTA_SKY130NM



10.8 Want to learn more?

A 77.3-dB SNDR 62.5-kHz Bandwidth Continuous-Time Noise-Shaping SAR ADC With Duty-Cycled Gm-C Integrator

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

Complex signal processing is not - complex

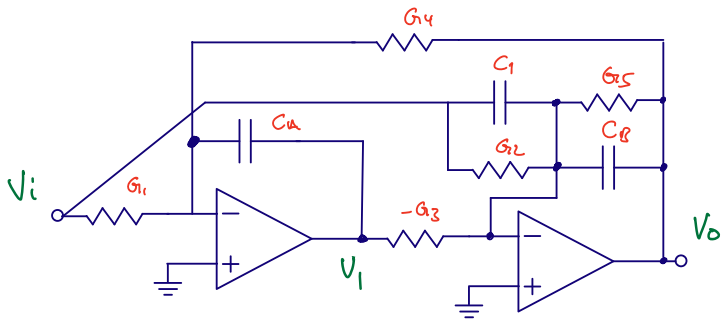
Switched-Capacitor Circuits 11

Keywords: SC DAC, SC FUND, DT, Alias, Subsample, Z Domain, FIR, IIR, SC MDAC, SC INT, Switch, Non-Overlap, VBE SC, Nyquist

Status: 0.5

11.1 Active-RC

A general purpose Active-RC bi-quadratic (two-quadratic equations) filter is shown below



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 is can have a 3-sigma value of

$$\sigma_{RC} = \sqrt{\sigma_R^2 + \sigma_C^2} = \sqrt{0.02^2 + 0.02^2} = 0.028 = 28\ \%$$

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 incredibility 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 kOhm 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 decent calibration algorithms, nor can we use binary search.

Analog designers need to deal with an almost infinite series of "But".

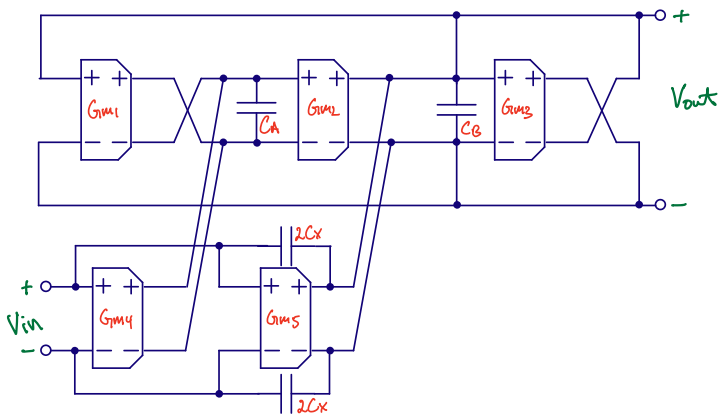
The experienced designer will know when to stop, when is the "But what if" not a problem anymore.

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.

11.2 Gm-C



$$H(s) = \frac{\left[s^2 \frac{C_X}{C_X+C_B} + s\frac{G_{m5}}{C_X+C_B} + \frac{G_{m2}G_{m4}}{C_A(C_X+C_B)}\right]}{\left[s^2 + s\frac{G_{m2}}{C_X+C_B} + \frac{G_{m1}G_{m2}}{C_A(C_X+C_B)}\right]}$$

The pole and zero frequency of a Gm-C filter is

$$\omega_{p|z} \propto \frac{G_m}{C}$$

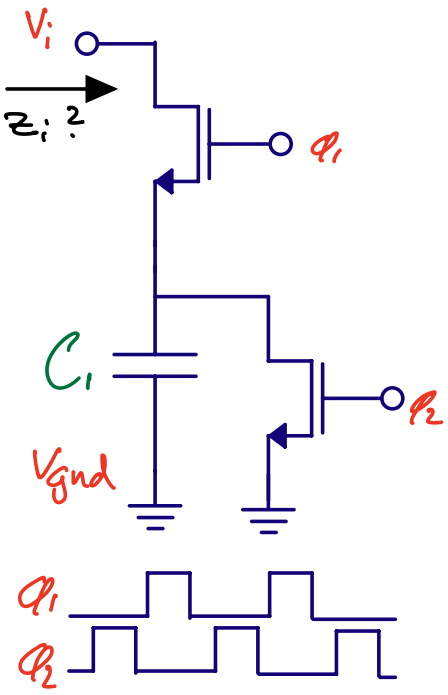
The transconductance accuracy depends on the circuit, and the bias circuit, so we can't give a general, applies for all circuits, sigma number. Capacitors do have 3-sigma 20 % variation, usually.

Same as Active-RC, Gm-C need calibration to get accurate pole or zero frequency.

11.3 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).



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

$$I_I = \frac{Q}{dt} = Q f_\phi$$

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.

$$Z_I = \frac{V_I - V_{GND}}{\left(Q_{\phi 1\$} - Q_{\phi 2\$}\right) f_\phi}$$

* I use the $ to mark the end of the period. It comes from Regular Expressions.

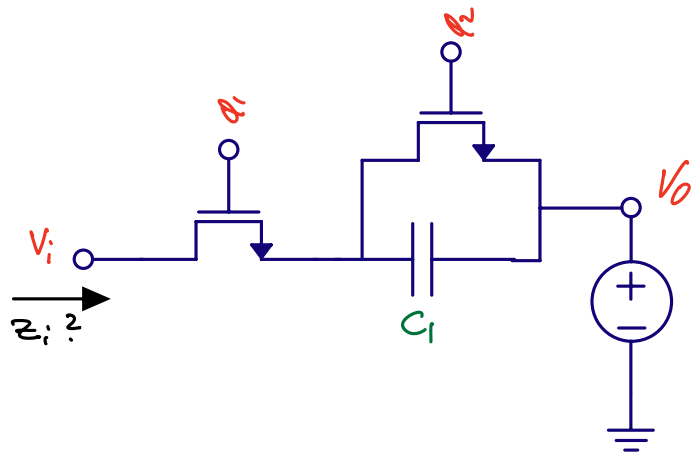
Inserting for the charges, we can see that the impedance is

$$Z_I = \frac{V_I}{(V_I C - 0)\, f_\phi} = \frac{1}{C_1 f_\phi}$$

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.



If we compute the impedance.

$$Z_I = \frac{V_I - V_O}{\left(Q_{\phi 1\$} - Q_{\phi 2\$}\right) f_\phi}$$

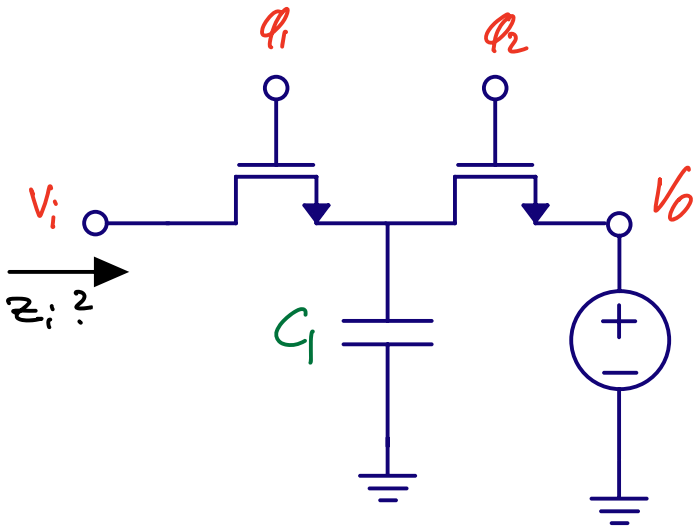
$$Q_{\phi 1\$} = C_1 (V_I - V_O)$$

$$Q_{\phi 2\$} = 0$$

$$Z_I = \frac{V_I - V_O}{(C_1 (V_I - V_O))\, f_\phi} = \frac{1}{C_1 f_\phi}$$

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.



$$Z_I = \frac{V_I - V_O}{\left(Q_{\phi 1\$} - Q_{\phi 2\$}\right) f_\phi}$$

$$Q_{\phi 1\$} = C_1 V_I)$$

$$Q_{\phi 2\$} = C_1 V_O$$

Inserted into the impedance we get the same result.

$$Z_I = \frac{V_I - V_O}{\left(C_1 V_I - C_1 V_O\right)) f_\phi} = \frac{1}{C_1 f_\phi}$$

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.

11.3.1 An example SC circuit

An example use of an SC circuit is

A pipelined 5-Msample/s 9-bit analog-to-digital converter

Shown in the figure below. You should think of the switched capacitor circuit as similar to a 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.

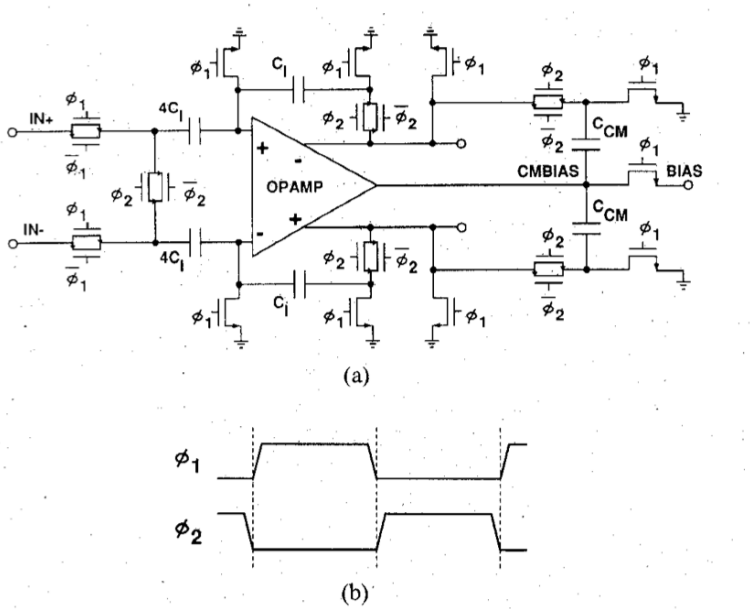


Fig. 6. (a) Schematic of S/H amplifier. (b) Timing diagram of a two-phase nonoverlapping clock.

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.

$$\omega_{p|z} \propto \frac{C_1}{C_2}$$

We can use these "switched capacitor resistors" to get pole or zero frequency or gain proportional to a 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.

11.4 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.

11.4.1 The mathematics

Define

$$x_c$$

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

$$\lim_{\tau \to 0} \rightarrow X_s(s) = \sum_{n=-\infty}^{\infty} x_c(nT)e^{-snT}$$

Or

$$X_s(j\omega) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X_c\left(j\omega - \frac{jk2\pi}{T}\right)$$

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

11.4.2 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" an "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 11001100, so our sample rate should be 1/2'th of the input sample rate. FFT's can be unwieldy beasts. I like to use coherent sampling, however, with multiple signals and samplerates I did not bother to figure out whether it was possible.

The alternative to coherent sampling is to apply a window function before the FFT, that's the reason for the Hanning window below.

dt.py

```
#- Create a time vector
N = 2**13
t = np.linspace(0,N,N)

#- Create the "continuous time" signal with multiple
#- "sinusoidal signals and some noise
f1 = 233/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)))
```

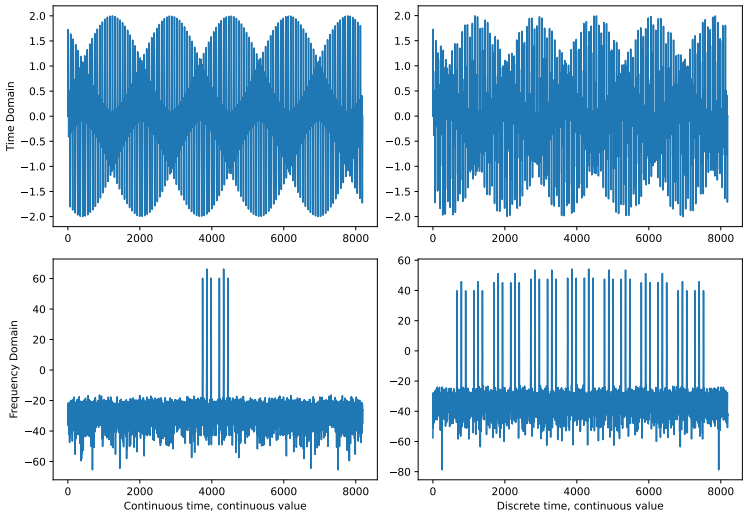
Try to play with the code, and see if you can understand what it does.

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 "0 Hz" would be at x-axis index 4096 ($2^{13}/2$).

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.

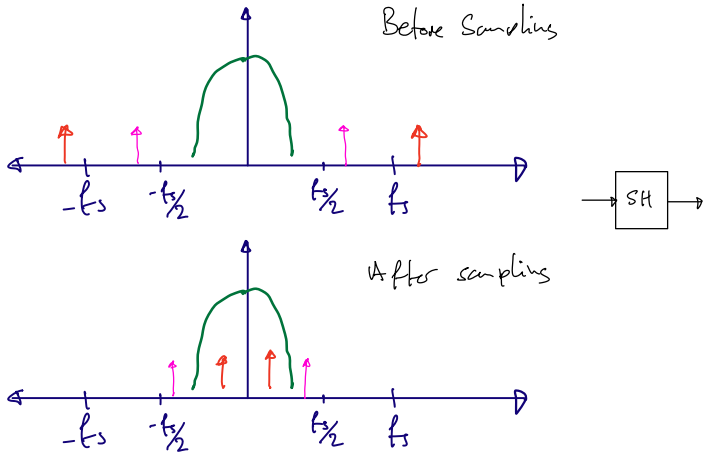


11.4.3 Aliasing, bandwidth and sample rate theory

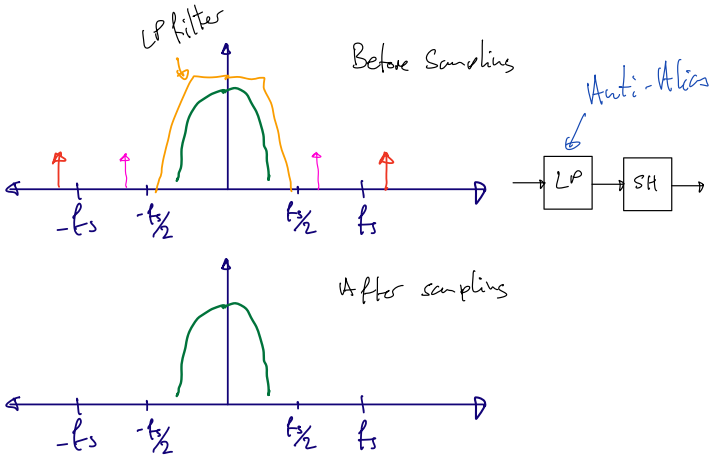
I want you to internalize that the spectral copies are real. They are not some "mathematical construct" that we don't have to deal

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 in may appear in-band. That's why it's important to band limit analog signals before they are sampled.



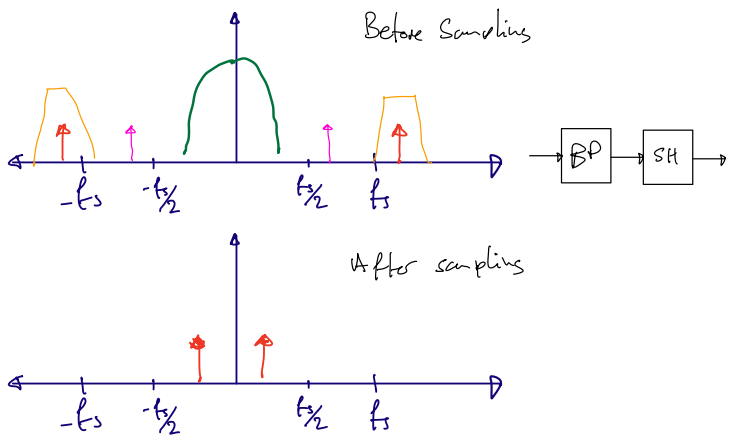
With an anti-alias filter (yellow) we ensure that the unwanted components are low enough before sampling. As a result, our wanted signal (green) is undisturbed.



Assume that we 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 is a good place to start to delve into the latest on signal reconstruction.



11.4.4 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

$$\frac{Y(z)}{X(z)} = \frac{1}{1 - z^{-1}}$$

11.4.5 Pole-Zero plots

If you're not comfortable with pole/zero plots, have a look at

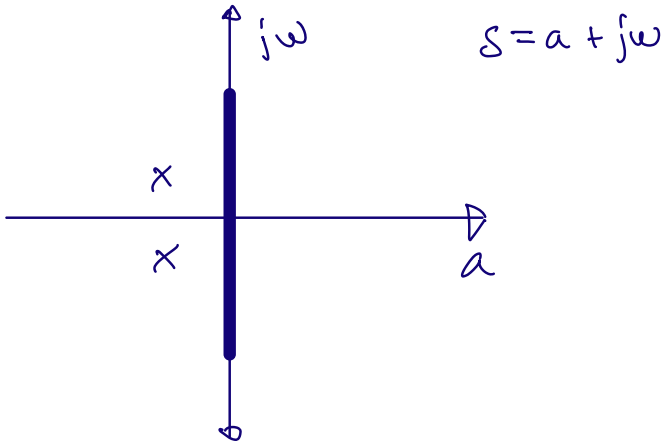
What does the Laplace Transform really tell us

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".



11.4.6 Z-domain

Spectra repeat every

$$2\pi$$

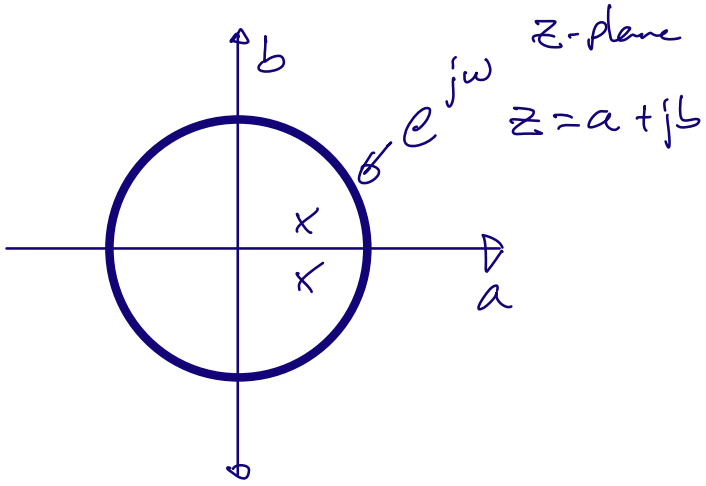
As such, it does not make sense to talk about a plane with a a and a $j\omega$. Rather we use the complex number $z = a + jb$.

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

$$s = \frac{z-1}{z+1}$$

Warning: First-order approximation https://en.wikipedia.org/wiki/Bilinear_transform



11.4.7 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

$$H(z) = \frac{b}{z-a}$$

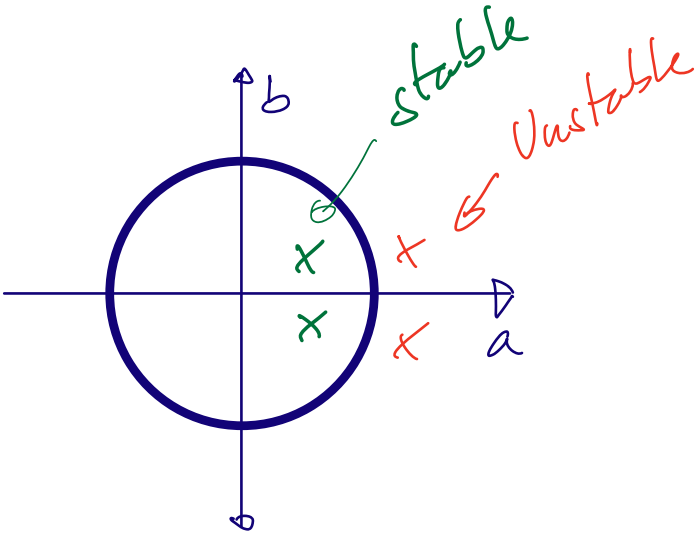
From the discrete time equation we can see that the impulse will never die out. We're adding the previous output to the current

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

From the impulse response it can be seen that if $a > 1$, then the filter is unstable. Same if $b > 1$. As long as $|a + jb| < 1$ the filter should be stable.

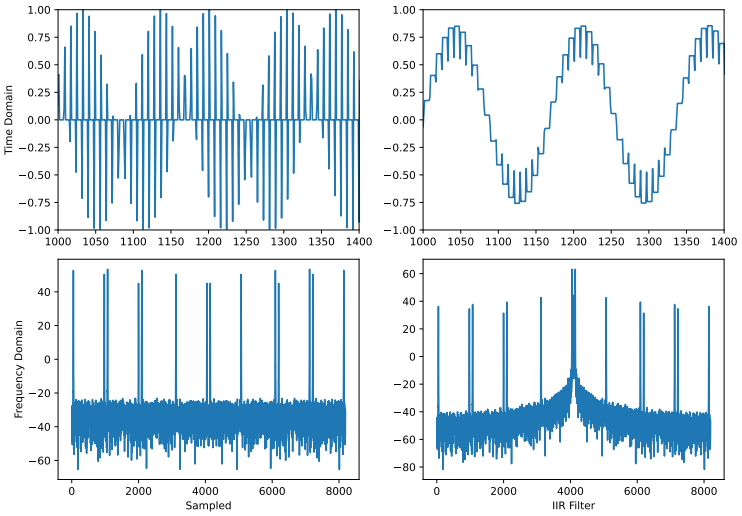


The first order filter can be implemented in python, and it's really not hard. See below. The $x_s n$ vector is from the previous python example.

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.

iir.py




```
#- IIR filter
b = 0.3
a = 0.25
z = a + 1j*b
z_abs = np.abs(z)
print("|z| = " + str(z_abs))
y = np.zeros(N)
y[0] = a
for i in range(1,N):
    y[i] = b*x_sn[i-1] + y[i-1]
```

The IIR filter we implemented above is a low-pass filter, and the filter partially rejects the copied spectra, as expected.

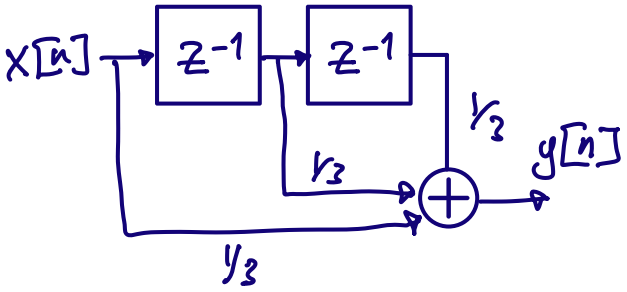
11.4.8 Finite-impulse response(FIR)

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, the 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.

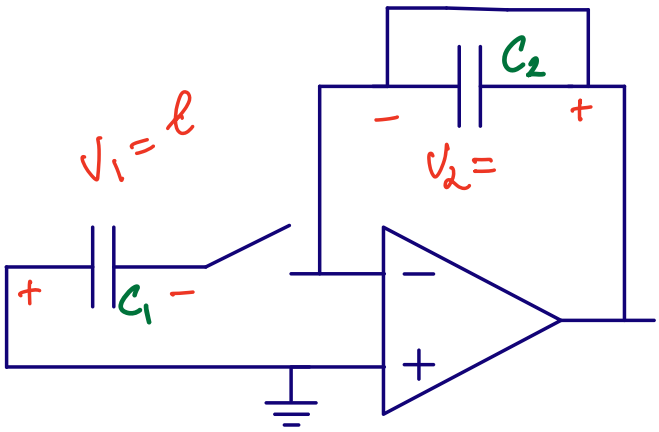
But be wary of rules like "IIR are always better than FIR" or visa versa. Especially if statements are written in books. Remember that the book was probably written a decade ago, and based on papers two decades old, which were based on three decades old state of the art. Our abilities to use computers for design has improved a bit the last three decades.

$$H(z) = \frac{1}{3}\sum_{i=0}^{2} z^{-1}$$



11.5 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.



This is the SC circuit during the sampling phase. Imagine that we somehow have stored a voltage $V_1 = \ell$ on capacitor C_1 (the switches for that sampling or storing are not shown). The charge on C_1 is

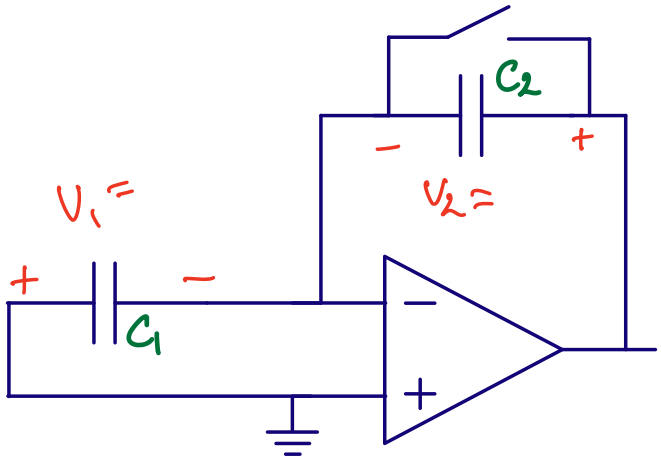
$$Q_{1\phi_1\$} = C_1 V_1$$

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.



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 it's 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 the voltages now have changed, there is zero volt across C_1, and thus there cannot be any charge across 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

$$Q_{2\phi_2\$} = Q_{1\phi_1\$} = C_1V_1 = C_2V_2$$

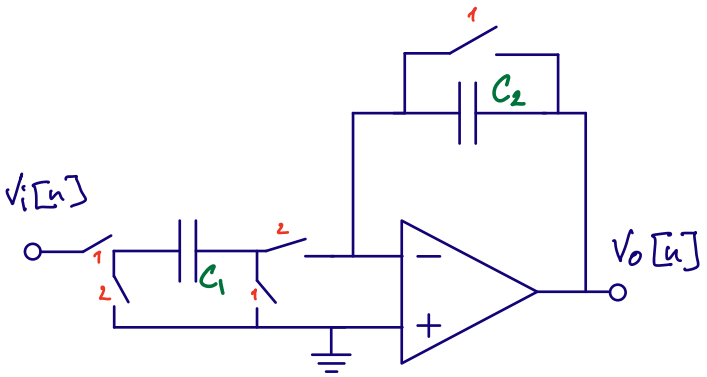
thus

$$\frac{V_2}{V_1} = \frac{C_1}{C_2}$$

11.5.1 Switched capacitor gain circuit

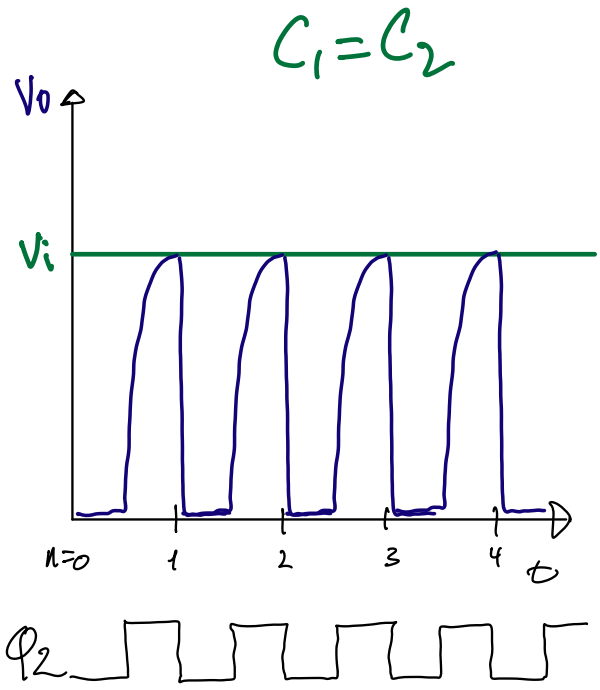
Redrawing the previous circuit, and adding a few more switches we can create a switched capacitor gain circuit.

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.



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.



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.

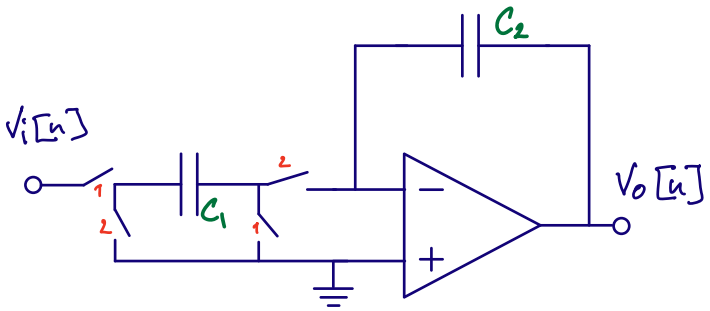
$$V_o[n+1] = \frac{C_1}{C_2} V_i[n]$$

$$V_o z = \frac{C_1}{C_2} V_i$$

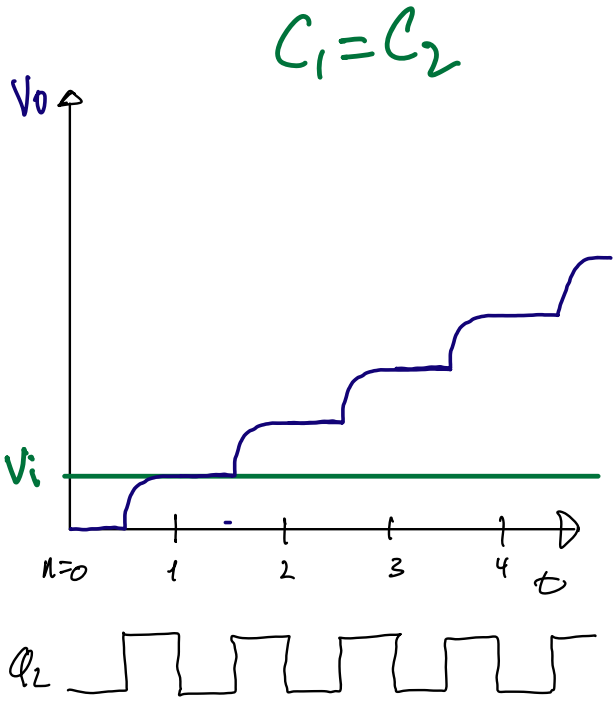
$$\frac{V_o}{V_i} = H(z) = \frac{C_1}{C_2} z^{-1}$$

11.5.2 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.



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.



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.

$$V_o[n] = V_o[n-1] + \frac{C_1}{C_2} V_i[n-1]$$

$$V_o - z^{-1}V_o = \frac{C_1}{C_2} z^{-1} V_i$$

Maybe one confusing thing is that multiple transfer functions can mean the same thing, as below.

$$H(z) = \frac{C_1}{C_2} \frac{z^{-1}}{1-z^{-1}} = \frac{C_1}{C_2} \frac{1}{z-1}$$

11.5.3 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

$$V_n^2 > \frac{2kT}{C}$$

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.

11.5.4 Sub-circuits for SC-circuits

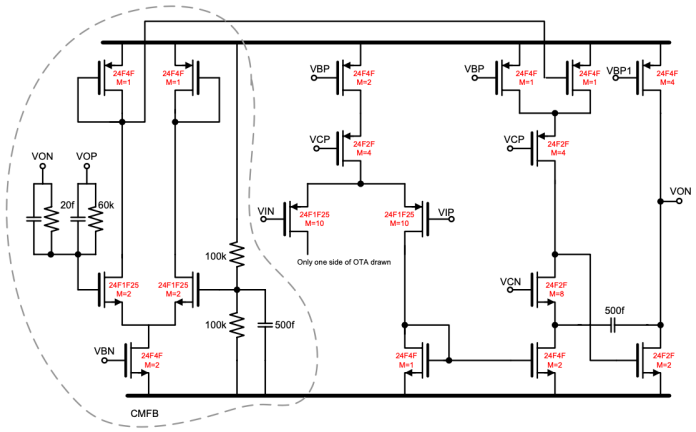
Switched-capacitor circuits are so common that it's good to delve a bit deeper, and understand the variants of the components that make up SC circuits.

11.5.4.1 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.

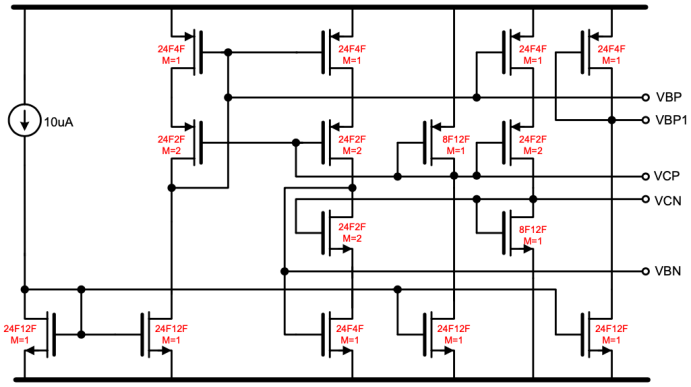
Not all SC circuits use OTAs, there are also comparator based SC circuits.

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.



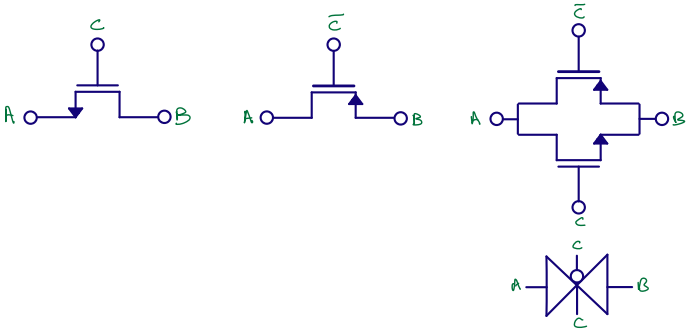
F = minimum transistor gate length. For example 24F4F => W = 24 x min gate, L= 4 x min gate

As bias circuit to make the voltages the below will work



11.5.4.2 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



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

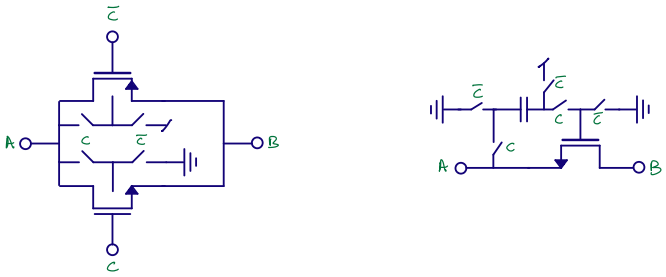
$$t > -\log(\text{error})\tau$$

For example, for a 10-bit ADC we need $t > -\log(1/1024)\tau = 6.9\tau$. 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

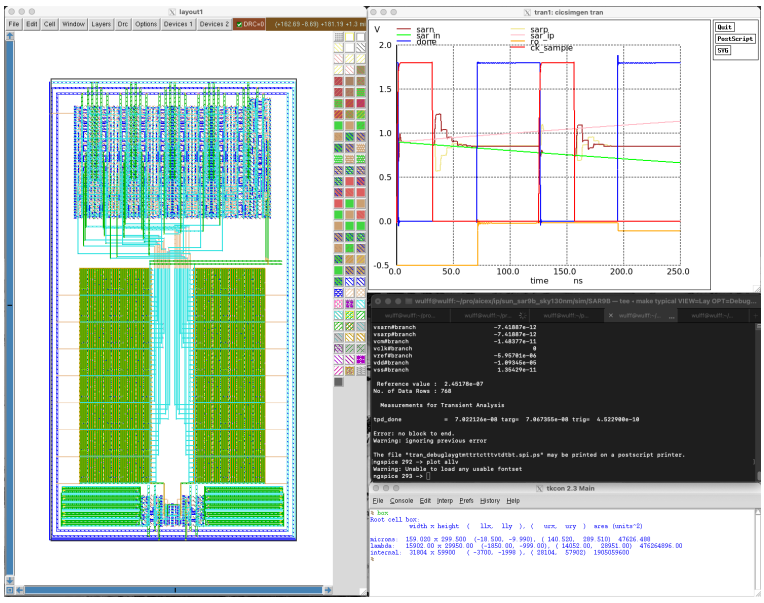
switches have smaller resistance, however, they also have more charge in the inversion layer, which leads to charge injection when the switches are turned of. Accordingly, larger switches are not always the solution.

Sometimes it may be sufficient to switch the bulks, as shown on the left below. But more often that one would like, we have to implement bootstrapped switches as shown on the right.

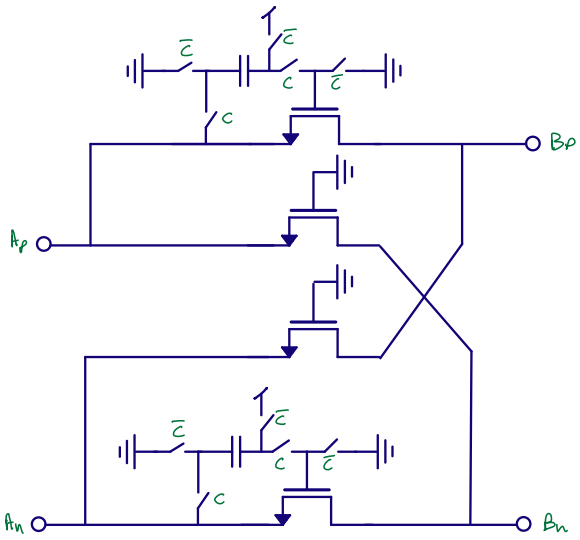


The switch I used in my JSSC SAR is a fully differential boostrapped switch with cross coupled dummy transistors. The JSSC SAR I've also ported to GF130NM, as shown below. The switch is at the bottom.

wulffern/sun_sar9b_sky130nm

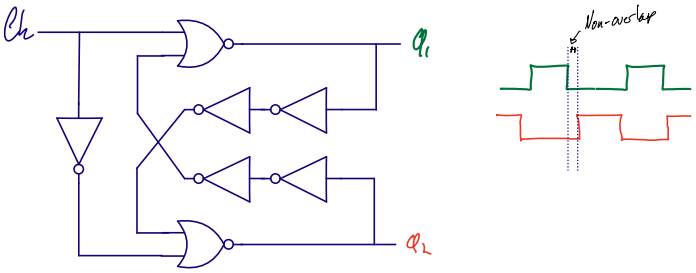


looks like the one below.



11.5.4.3 Non-overlapping clocks

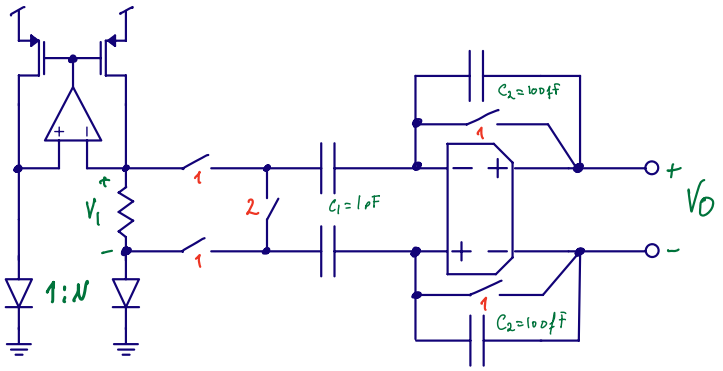
The non-overlap generator is standard. Use the one shown below. Make sure you simulate that the non-overlap is sufficient in all corners.



11.5.5 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

$$V_O(z) = 10\frac{kT}{q}\ln(N)z^{-1}$$



11.6 Want to learn more?

Blind Multiband Signal Reconstruction: Compressed Sensing for Analog Signal

Comparator-based switched-capacitor pipelined analog-to-digital converter with comparator preset, and comparator delay compensation

A Compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for Bluetooth Low Energy Receivers

A 10-bit 50-MS/s SAR ADC With a Monotonic Capacitor Switching Procedure

Low Voltage, Low Power, Inverter-Based Switched-Capacitor Delta-Sigma Modulator

Ring Amplifiers for Switched Capacitor Circuits

A Switched-Capacitor RF Power Amplifier

Design of Active N-Path Filters

Oversampling and Sigma-Delta ADCs

12

Keywords: Quantization, OSR, NEG FB, STF, NTF, SAR, First Order, SC SD, CT SD, INCR, FOM

Status: 0.5

12.1 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, one of the plot's is shown below.

1999, R. Walden: Analog-to-digital converter survey and analysis

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.

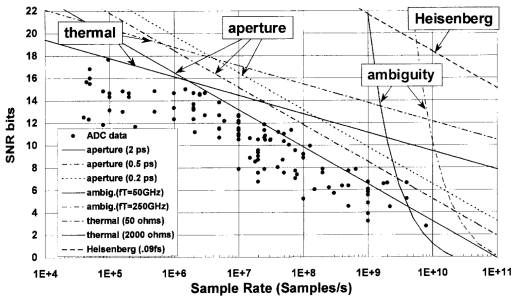


Fig. 7. Signal-to-noise ratio according to SNR-bits = (SNR(dB) − 1.76)/6.02. Three sets of curves show performance limiters due to thermal noise, aperture uncertainty, and comparator ambiguity. The Heisenberg limit is also displayed.

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

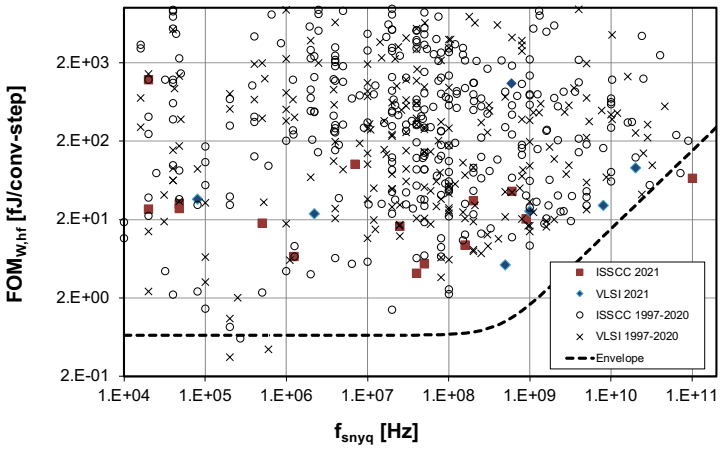
A common figure of merit for low-to-medium resolution ADCs is the Walden figure of merit, defined as

$$FOM_W = \frac{P}{2^B f_s}$$

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.



12.1.1 What makes a state-of-the-art ADC

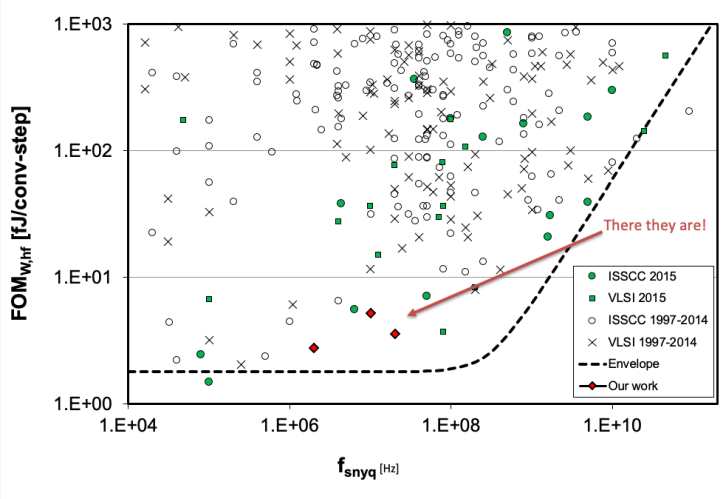
People from NTNU have made some of the worlds best ADCs

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 a 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

[2] A 68 dB SNDR Compiled Noise-Shaping SAR ADC With On-Chip CDAC Calibration



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.

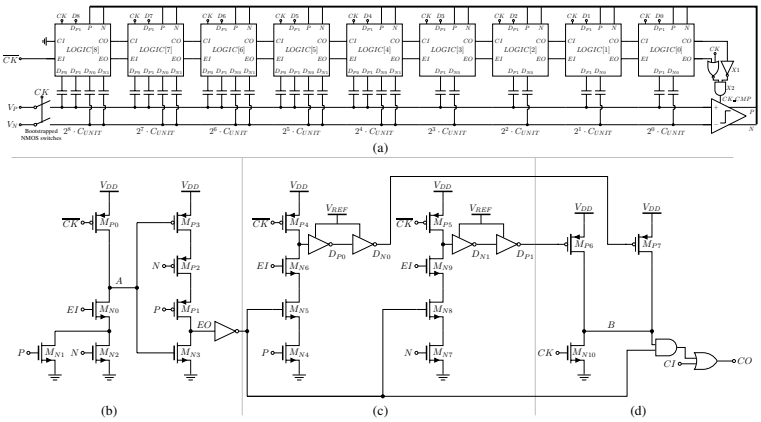
As soon as the comparator makes a decision, P or N goes high, A will be pulled low, if EI is enabled.

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.

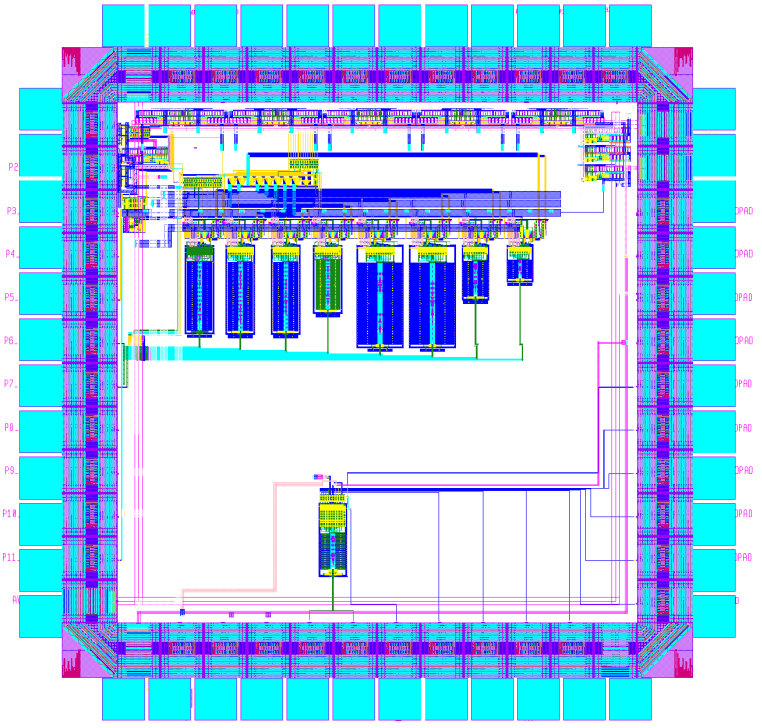
How fast the D_{XX} settle depend on the size of the capacitors, as such, the comparator clock will be slow for the MSB, and very fast for the LSB. This was my main circuit contribution in the paper. I think it's quite clever, because both the VDD and the capacitor corner will change the settling time. 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.

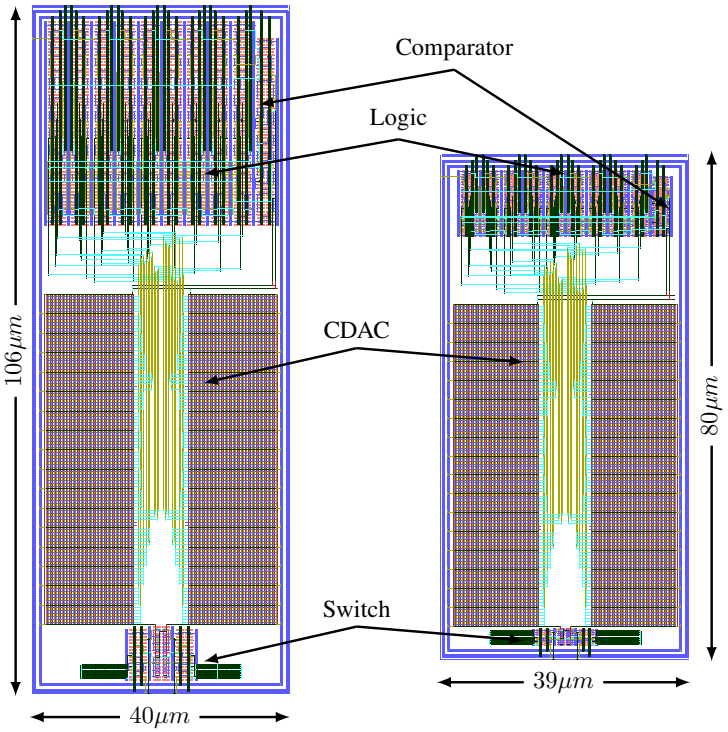


For state-of-the-art ADC papers it's not sufficient with the idea, and simulation. There must be proof that it actually works. No-one will really believe that the ADC works until there is measurements of an actual taped out IC.

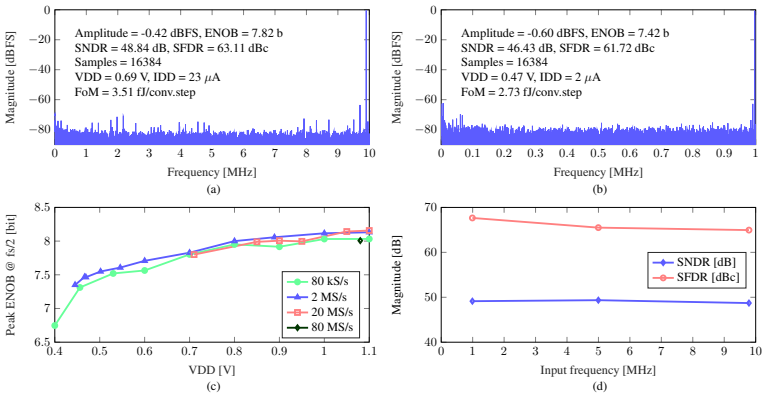
Below you can see the layout of the IC I made for the paper. Notice that there are 9 ADCs. I had many ideas that I wanted to try out, and I was not sure what would actually be state of the art. As a result, I taped out multiple ADCS.



The two ADCs that I ended up using in the paper is 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.



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.

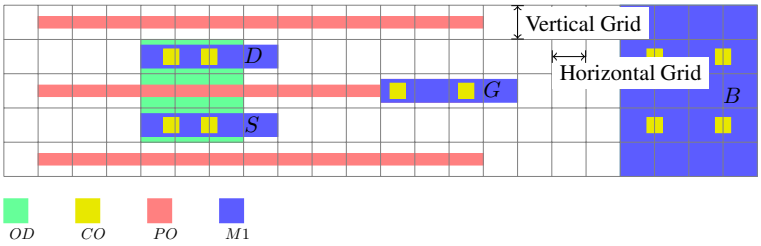


Comparing my ADCs to others, we can see that the FOM is similar to others. Based on the FOM it might not be clear why the paper was considered state-of-the-art.

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.

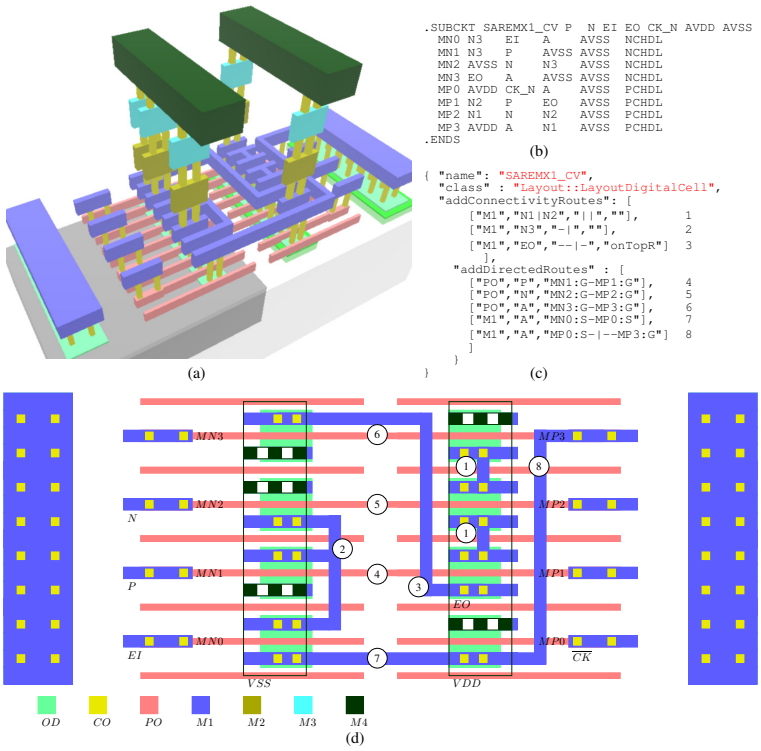
	Weaver [5]	Harpe [9]	Patil [10]	Liu [11]	This work	
Technology (nm)	90	90	28 FDSOI	28	28 FDSOI	
Fsample (MS/s)	21	2	No sampling	100	2	20
Core area (mm^2)	0.18	0.047	0.0032	0.0047	0.00312	
SNDR (dB)	34.61	57.79	40	64.43	46.43	48.84
SFDR (dBc)	40.81	72.33	30	75.42	61.72	63.11
ENOB (bits)	5.45	6.7 - 9.4	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	
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



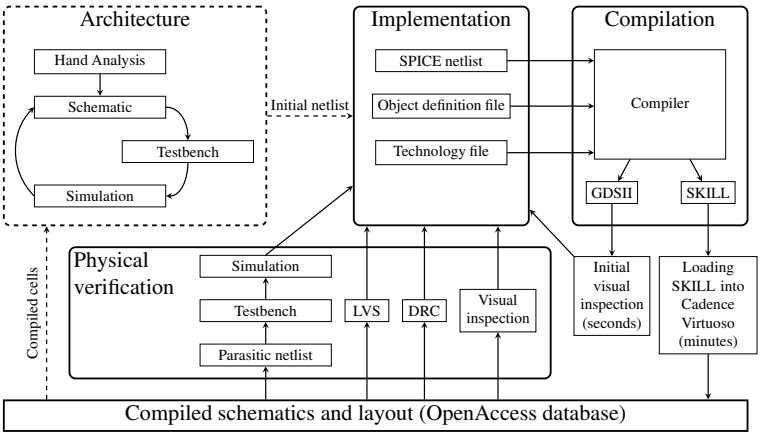
And then wrote a compiler (in Perl, later C++ ciccreator) to compile a object definition file, a SPICE netlist and a technology rule file into the full ADC layout.

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).



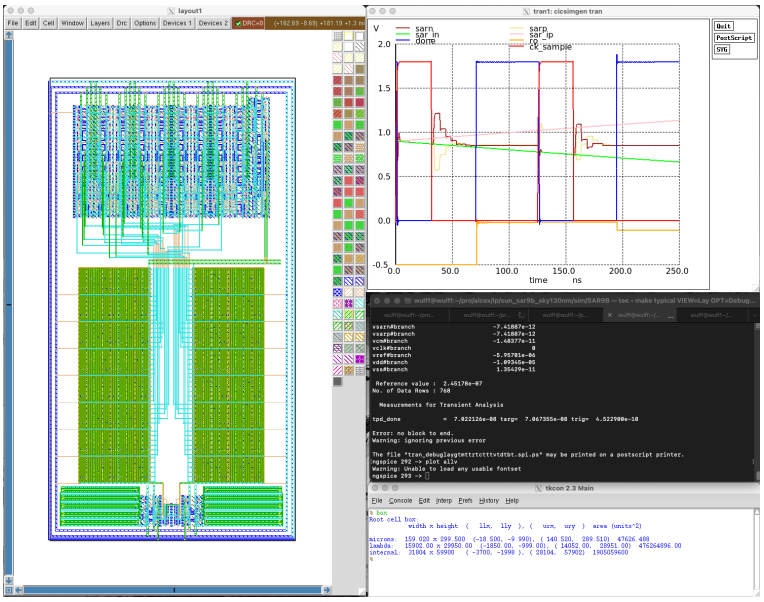
The implementation is the SPICE netlist, and the object definition file (JSON) and the rule file.

What I really like is the fact that the compilation could generate GDSII or SKILL, or these days, Xschem schematics and Magic layout.

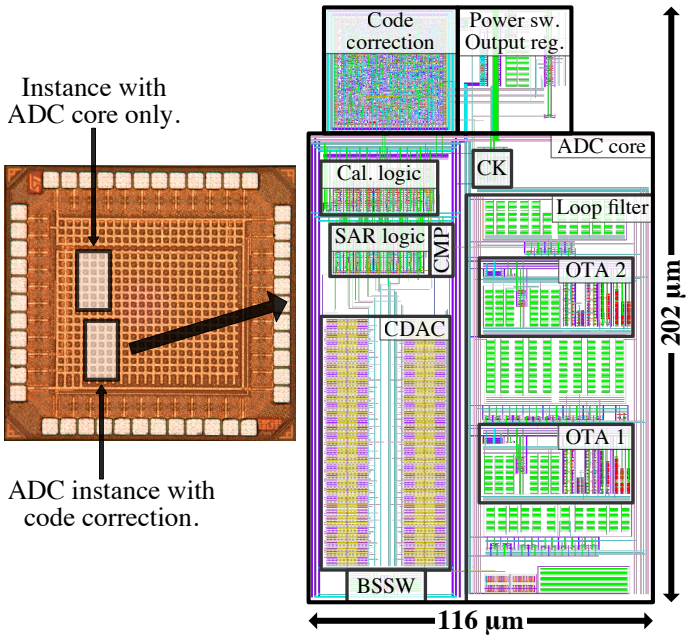


The cool thing with a compiled ADC is that it's easy to port between technologies. Since the original ADC, I've ported the ADC to multiple closed PDKs (22 nm FDSOI, 22 nm, 28 nm, 55 nm, 65 nm and 130nm). In the summer of 2022 I made an open source port to skywater 130nm.

SUN_SAR9B_SKY130NM



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.



12.1.2 High resolution FOM

For high-resolution ADCs, it's more common to use the Schreier figure of merit, which can also be found in

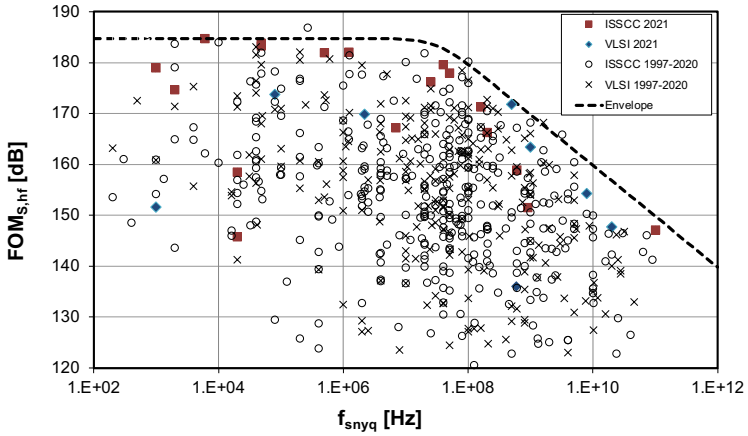
B. Murmann, ADC Performance Survey 1997-2022 (ISSCC & VLSI Symposium)

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.

$$FOM_S = SNDR + 10\log\left(\frac{f_s/2}{P}\right)$$

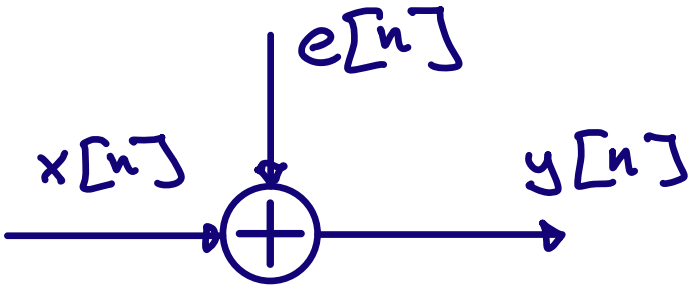
Above 180 dB is extreme



12.2 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 $x[n] = y[n] - e[n]$.



Maybe you've even heard the phrase "Quantization noise is white" or "Quantization noise is a random Gaussian process"?

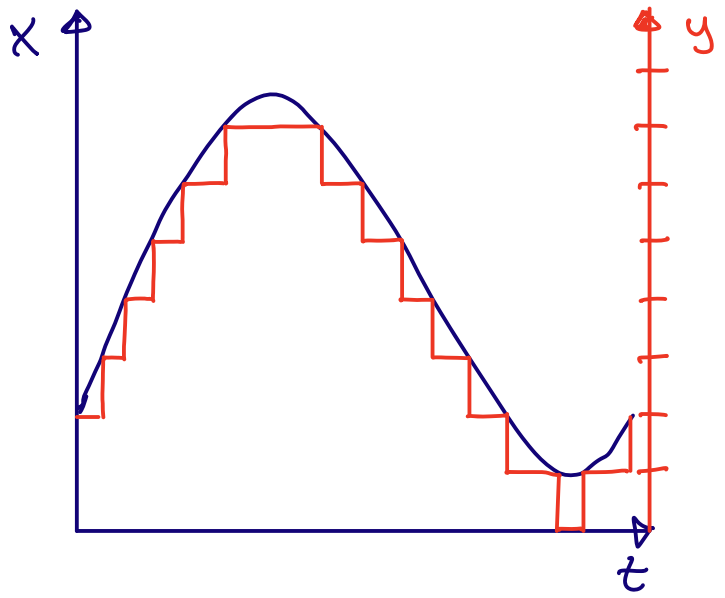
I'm here to tell you that you've been lied to. Quantization noise is not white, nor is it a Gaussian process. Those that have lied to you may say "yes, sure, but for high number of bits it can be considered white noise". I would say that's similar to saying "when you look at the earth from the moon, 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?

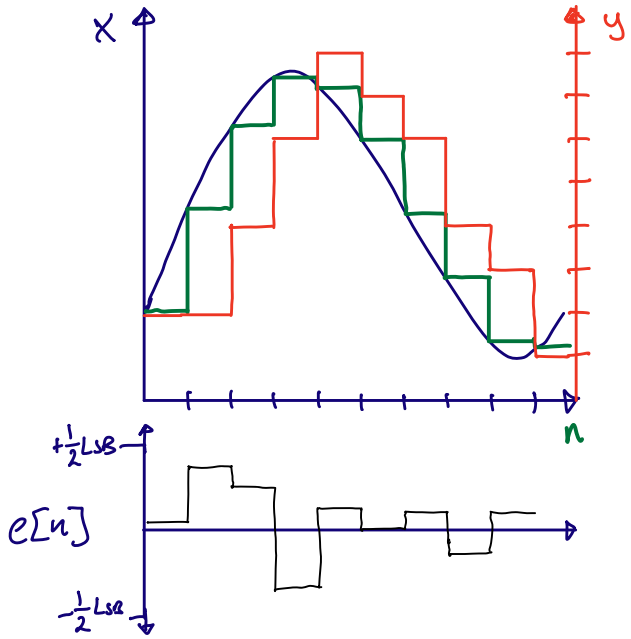
The figure below shows the input signal x and the quantized signal y.



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.



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 for details

In short, quantization noise is defined as

$$e_n(t) = \sum_{p=1}^{\infty} A_p \sin p\omega t$$

where p is the harmonic index, and

$$A_p = \begin{cases} \delta_{p1} A + \Sigma_{m=1}^{\infty} \frac{2}{m\pi} J_p(2m\pi A) & , p = \text{odd} \\ 0 & , p = \text{even} \end{cases}$$

$$\delta_{p1} \begin{cases} 1 & , p = 1 \\ 0 & , p \neq 1 \end{cases}$$

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

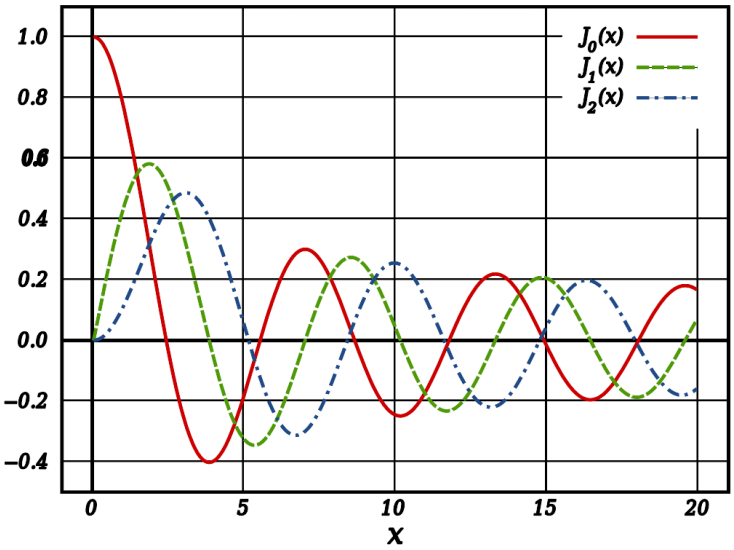
$$e_n(t) = \sum_{p=1}^{\infty} A_p \sin p\omega t$$

$$A_p = \delta_{p1} 2^{n-1} + \sum_{m=1}^{\infty} \frac{2}{m\pi} J_p(2m\pi 2^{n-1}), p = odd$$

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



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

$$\overline{e_n(t)} = 0$$

and variance (mean square, since mean is zero), or noise power, can be approximated as

$$\overline{e_n(t)^2} = \frac{\Delta^2}{12}$$

12.2.1 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

$$SQNR = 10\log\left(\frac{A^2/2}{\Delta^2/12}\right) = 10\log\left(\frac{6A^2}{\Delta^2}\right)$$

$$\Delta = \frac{2A}{2^B}$$

$$SQNR = 10\log\left(\frac{6A^2}{4A^2/2^B}\right) = 20B\log 2 + 10\log 6/4$$

$$SQNR \approx 6.02B + 1.76$$

You may have seen the last equation before, now you know where it comes from.

12.2.2 Understanding quantization

Below I've tried to visualize the quantization process q.py.

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 the FFT bin (not the frequency). Notice that there are two spikes, which should not be surprising, since a sinusoidal signal is a combination of two frequencies.

$$sin(x) = \frac{e^{ix} - e^{-ix}}{2i}$$

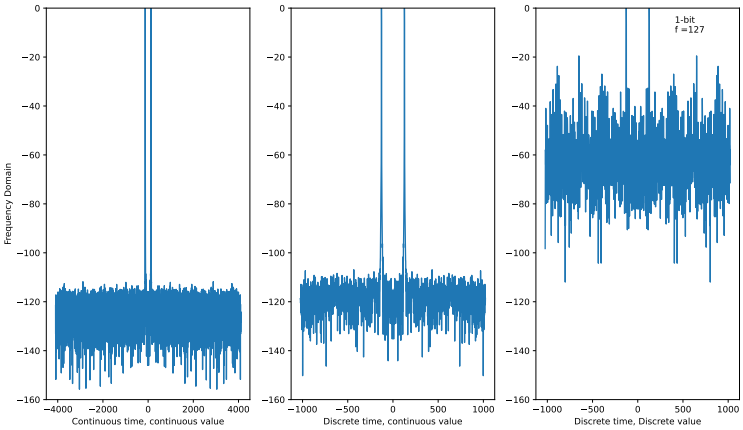
The second plot from the left is after sampling, notice that the noise level increases. The increase in the noise level should be due to noise folding, and reduced number of points in the FFT, but I have not confirmed (maybe you could confirm?).

The right plot is after quantization, where I've used the function below.

```
def adc(x,bits):
    levels = 2**bits
    y = np.round(x*levels)/levels
    return y
```

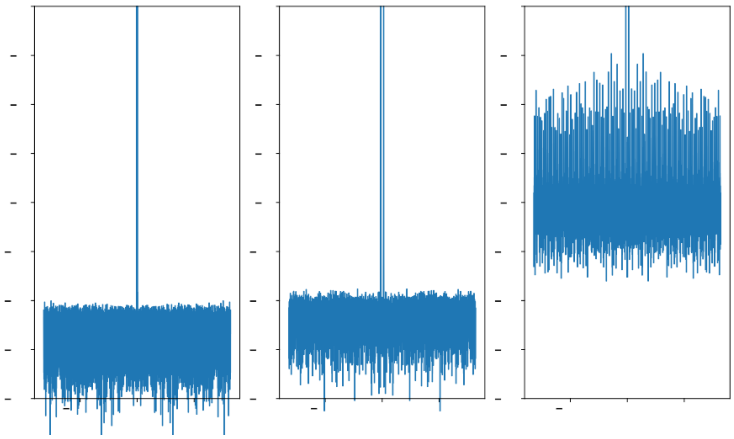
I really need you to internalize a few things from the right most plot. Really think through what I'm about to say.

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.



If you run the python script you can zoom in and check the highest spikes. The fundamental is at 127, so odd harmonics would be 381, 635, 889, and from the function of the quantization noise we would expect those to be the highest harmonics (at least when we look at the Bessel function), however, we can see that it's close, but that bin 396 is the highest. Is the math's wrong?

No, the math is correct. Never bet against mathematics. 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, you'll see it's the 11'th harmonic that is highest.



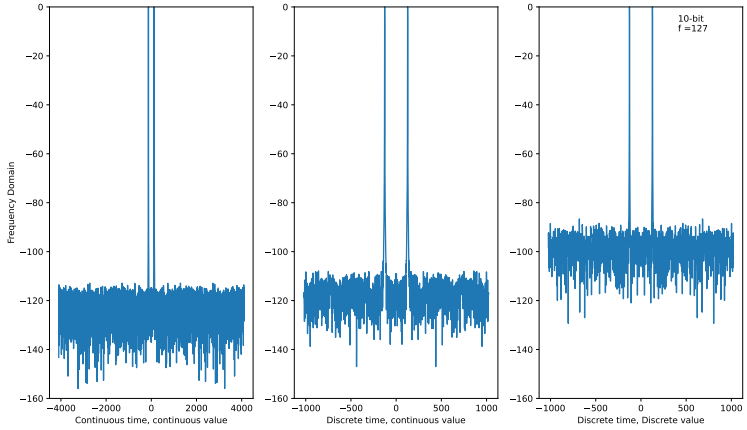
All the other spikes are the odd harmonics above the sample rate that fold. The infinite sum of harmonics will fold, some in-phase, some out of phase, depending on the sign of the Bessel function.

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.



12.2.3 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.

12.3 Oversampling

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?

12.3.1 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

$$\overline{(b_1 + b_2)^2} = 2\overline{b^2}$$

12.3.2 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 $(OSR \times A)^2/2$.

12.3.3 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**

12.3.4 Signal to Quantization Noise Ratio

The in-band quantization noise for a oversampling ratio (OSR)

$$\overline{e_n(t)^2} = \frac{\Delta^2}{12OSR}$$

And the improvement in SQNR can be calculated as

$$SQNR = 10\log\left(\frac{6A^2}{\Delta^2/OSR}\right) = 10\log\left(\frac{6A^2}{\Delta^2}\right) + 10\log(OSR)$$

$$SQNR \approx 6.02B + 1.76 + 10\log(OSR)$$

For an OSR of 2 and 4 the SQNR improves by

$$10\log(2) \approx 3dB$$

and for OSR=4

$$10\log(4) \approx 6dB$$

which is roughly equivalent to a 0.5-bit per doubling of OSR

12.3.5 Python oversample

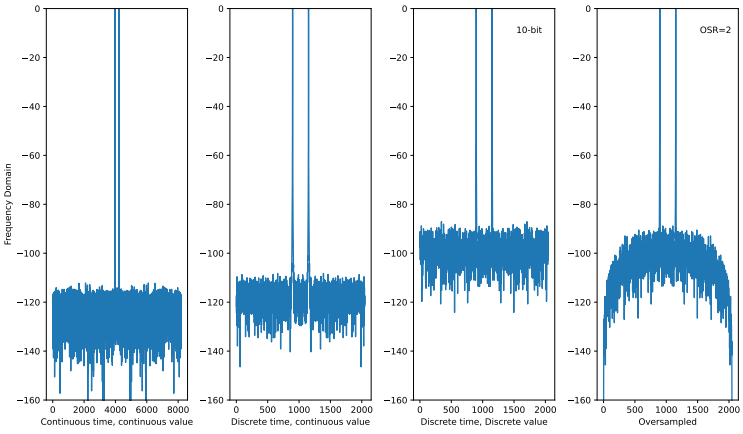
There are probably more elegant (and faster) ways of implementing oversampling in python, but I like to write the dumbest code I can, simply because dumb code is easy to understand.

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
```

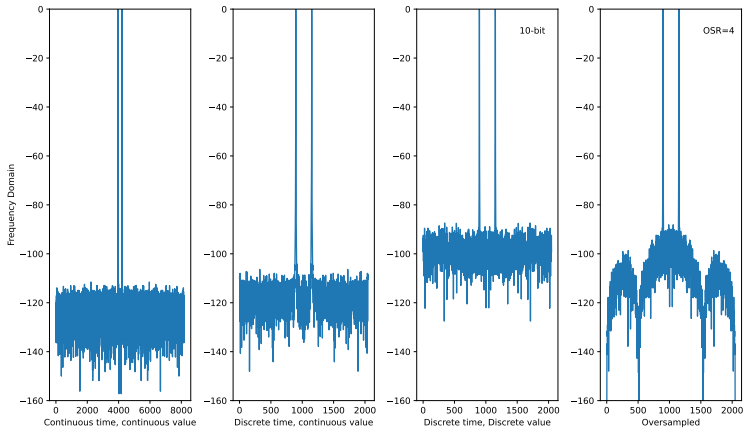
Below we can see the plot for OSR=2, the right most plot is the oversampled version.

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.



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 notice how the noise floor has 4 zero's.



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.

12.4 Noise Shaping

Look at the OSR=4 plot above. The OSR=4 does decrease the noise compared to the discrete time discrete value plot, however, the noise level of the discrete time continuous value is much lower.

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?

12.4.1 The magic of feedback

A generalized feedback system is shown below, it could be a regulator, a unity-gain buffer, or something else.

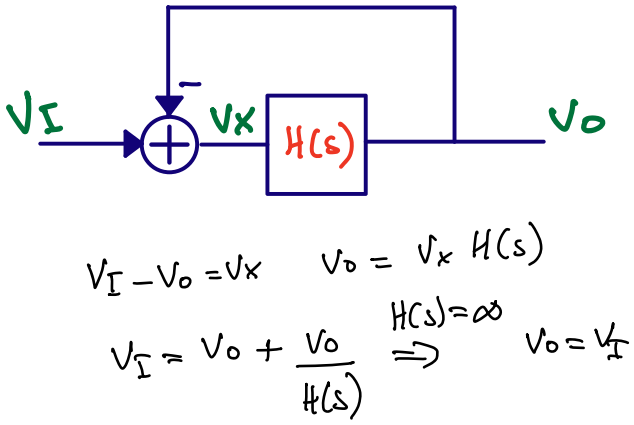
The output V_o is subtracted from the input V_i, and the error V_x is shaped by a filter $H(s)$.

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.

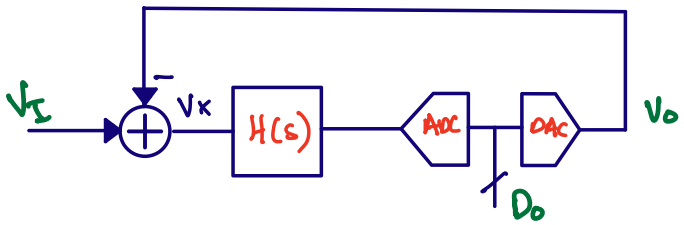


12.4.2 Sigma-delta principle

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

$$D_o = adc\left[H(s)\left(dac(D_o) - V_i\right)\right]$$

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.



12.4.2.1 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.

12.4.2.2 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.

12.4.2.3 The modified equation

With the assumptions we can change the equation into

$$D_o = adc\left[H(s)\left(V_i - dac(D_o)\right)\right] = H(s)\left(V_i - AD_o\right) + e$$

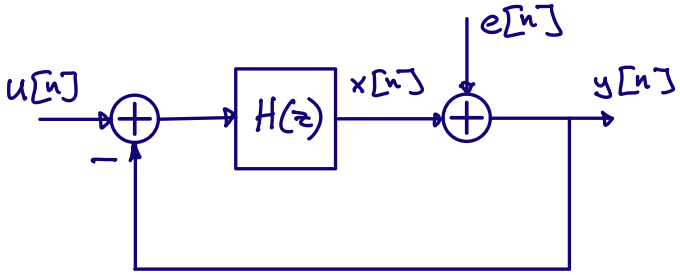
In noise-shaping texts it's common to write the above equation as

$$y = H(s)(u - y) + e$$

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.



in the Z-domain the equation would turn into

$$Y(z) = E(z) + H(z)\,[U(z) - Y(z)]$$

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.

12.4.3 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.

12.4.4 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.

12.4.5 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.

12.5 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

$$NFT = \frac{1}{1+1/(z-1)} = \frac{z-1}{z} = 1 - z^{-1}$$

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 with the continuous time variable

$$z = e^{sT} \overset{s=j\omega}{\rightarrow} e^{j\omega T} = e^{j2\pi f/f_s}$$

inserted into the NTF we get the function below.

$$NTF(f) = 1 - e^{-j2\pi f/f_s}$$

$$= \frac{e^{j\pi f/f_s} - e^{-j\pi f/f_s}}{2j} \times 2j \times e^{-j\pi f/f_s}$$

$$= \sin\frac{\pi f}{f_s} \times 2j \times e^{-j\pi f/f_s}$$

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)

$$|NFT(f)| = \left|2\sin\left(\frac{\pi f}{f_s}\right)\right|$$

The signal power for a sinusoid is

$$P_s = A^2/2$$

The in-band noise power for the shaped quantization noise is

$$P_n = \int_{-f_0}^{f_0} \frac{\Delta^2}{12}\frac{1}{f_s}\left[2\sin\left(\frac{\pi f}{f_s}\right)\right]^2 dt$$

and with a bunch of tedious maths, we can get to the conclusion

$$\vdots$$

$$SQNR = 6.02B + 1.76 - 5.17 + 30\log(OSR)$$

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.

12.5.1 SQNR and ENOB

Below is the signal-to-quantization noise ratio's for Nyquist up to second order sigma-delta.

$$SQNR_{nyquist} \approx 6.02B + 1.76$$

$$SQNR_{oversample} \approx 6.02B + 1.76 + 10\log(OSR)$$

$$SQNR_{\Sigma\Delta 1} \approx 6.02B + 1.76 - 5.17 + 30\log(OSR)$$

$$SQNR_{\Sigma\Delta 2} \approx 6.02B + 1.76 - 12.9 + 50\log(OSR)$$

We could compute an effective number of bits, as shown below.

$$ENOB = (SQNR - 1.76)/6.02$$

The table below shows the effective number of bits for oversampling, and sigma-delta modulators. For a 1-bit quantizer, pure oversampling does not make sense at all. For first-order and second-order sigma delta modulators, and a OSR of 1024 we can get high resolution ADCs.

Assume 1-bit quantizer, what would be the maximum ENOB?

OSR	Oversampling	First-Order	Second Order
4	2	3.1	3.9
64	4	9.1	13.9
1024	6	15.1	23.9

12.6 Examples

12.6.1 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

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).

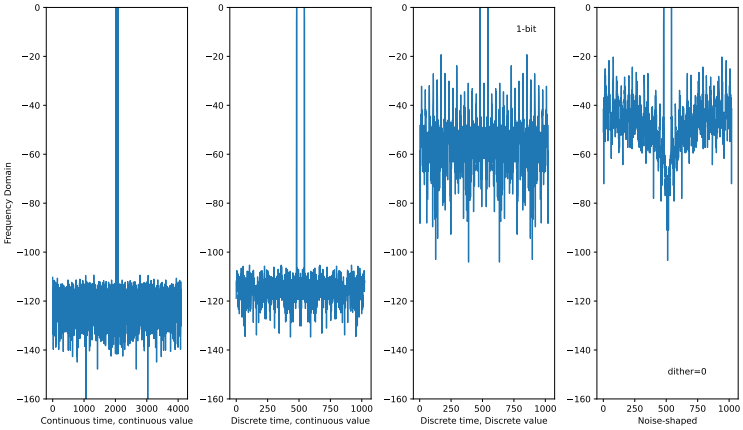
For each sample in the input vector u I compute the input to the quantizer x, which is the sum of the previous input to the quantizer and the difference between the current input and the previous output y_{sd}.

The quantizer generates the next y_{sd} and I have the option to add dither.

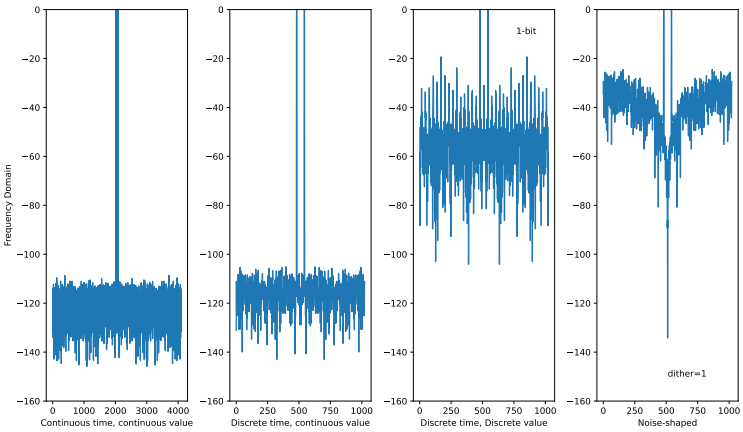
```
# u is discrete time, continuous value input
M = len(u)
y_sd = np.zeros(M)
x = np.zeros(M)
for n in range(1,M):
    x[n] = x[n-1] + (u[n]-y_sd[n-1])
    y_sd[n] = np.round(x[n]*2**bits
    + dither*np.random.randn()/4)/2**bits
```

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.

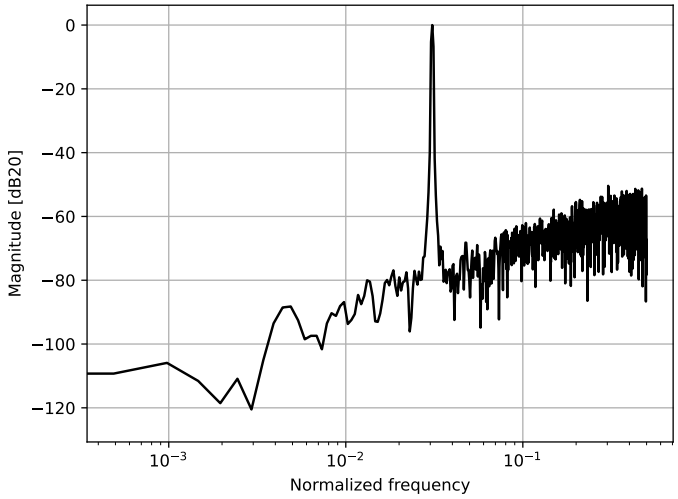
If we look at the noise we can also see the non-white quantization noise, which will degrade our performance. I hope by now, you've grown tired of me harping on the point that **quantization noise is not white**



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.



In papers it's common to use a logarithmic x-axis for the power spectral density, 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.



12.6.2 The wonderful world of SD modulators

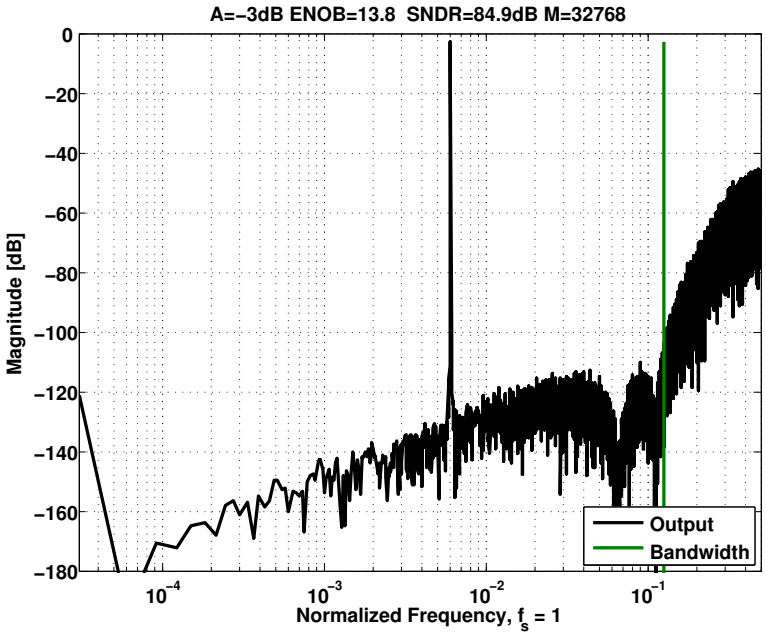
12.6.2.1 Open-Loop Sigma-Delta

On my Ph.D I did some work on

Resonators in Open-Loop Sigma-Delta Modulators

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 NFT where there are two complex conjugate zeros, and a zero at zero frequency. With a higher order filter one can use a lower OSR, and still achieve high ENOB.



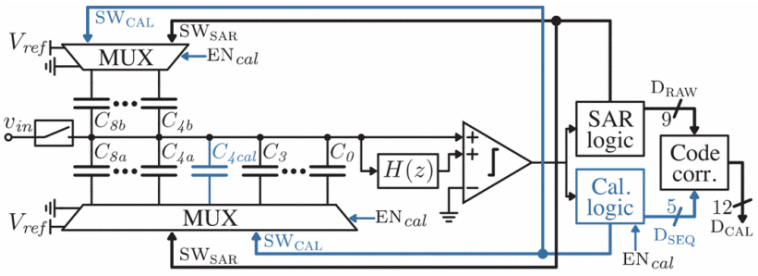
12.6.2.2 Noise Shaped SAR

One of my Ph.d students made a

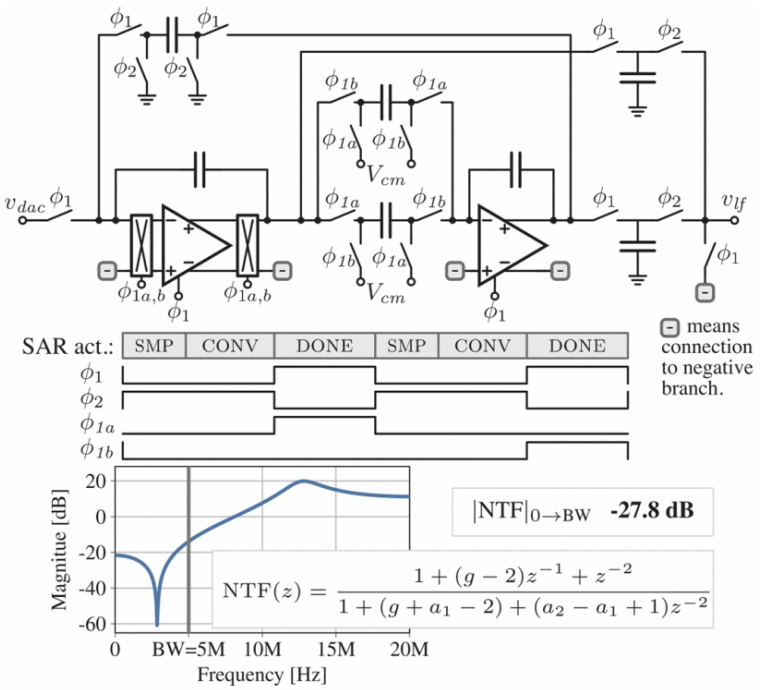
A 68 dB SNDR Compiled Noise-Shaping SAR ADC With On-Chip CDAC Calibration

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.



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.



12.6.2.3 Control-Bounded ADCs

One of my current Ph.D students is working 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.

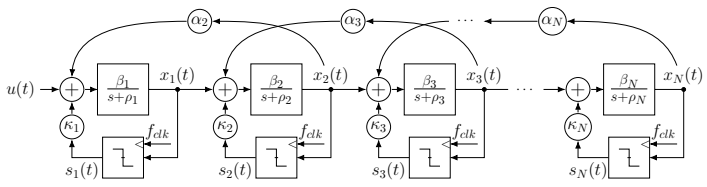
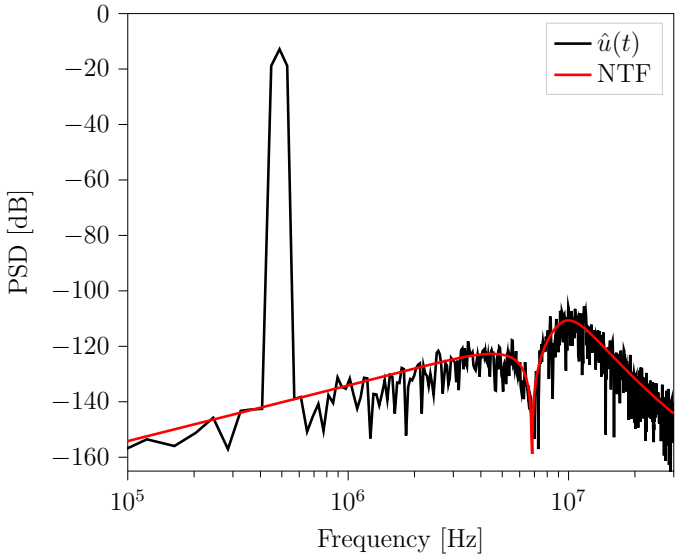


Figure 3.1: The general structure of the Leapfrog ADC

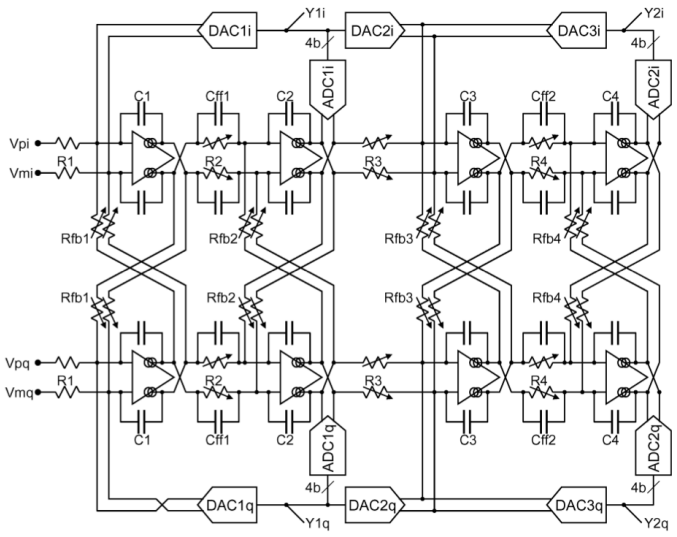
Below we can see a power spectral density plot of the ADC, and we can observe how the quantization noise is shaped. I think it's a third order NTF with a zero at zero frequency and a complex conjugate pole at 8 MHzish.



12.6.2.4 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 recievers

A 56 mW Continuous-Time Quadrature Cascaded Sigma-Delta Modulator With 77 dB DR in a Near Zero-IF 20 MHz Band



igma-delta modulator design.

12.6.2.5 My first Sigma-Delta

The first sigma-delta modulator I made in "real-life" was similar to the one shown below.

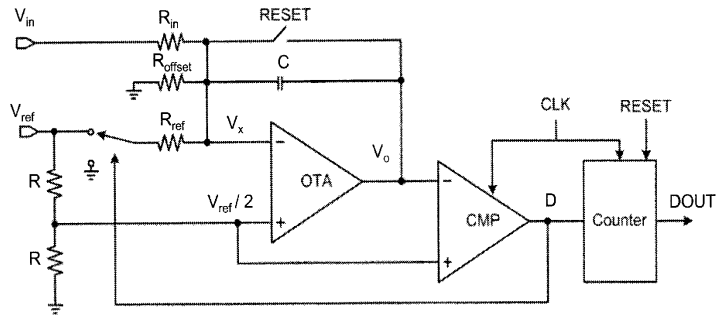
The input voltage is translated into a current, and the current is integrated on capacitor C. The R_{offset} is to change the mid-level voltage, while R_{ref} is the 1-bit feedback DAC. The comparator is the quantizer. When the clock strikes the comparator compares the V_o and $V_{ref}/2$ and outputs 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



12.7 Want to learn more?

The design of sigma-delta modulation analog-to-digital converters

Delta-sigma modulation in fractional-N frequency synthesis

A CMOS Temperature Sensor With a Voltage-Calibrated Inaccuracy of ± 0.15 C (3sigma) From -55 Cto 125 C

A 20-mW 640-MHz CMOS Continuous-Time Sigma-Delta ADC With 20-MHz Signal Bandwidth, 80-dB Dynamic Range and 12-bit ENOB

A Micro-Power Two-Step Incremental Analog-to-Digital Converter

Voltage regulation 13

Keywords: Battery, Vreg, LDOP, LDON, Flipped voltage follower, Buck, Boost, Load, Line, PSRR, MAX C, Quiescent, Settling, Efficiency, PWM, PFM

Status: 0.5

13.1 Voltage source

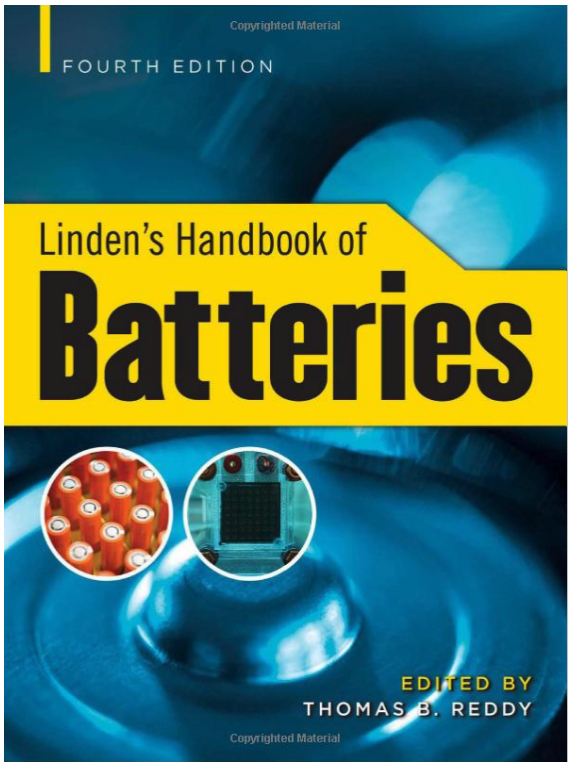
Most, if not all, integrated circuits need a supply and ground to work.

Assume a system is AC powered. Then there will be 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 Lanny explains how to design around some of the breakdown effects.

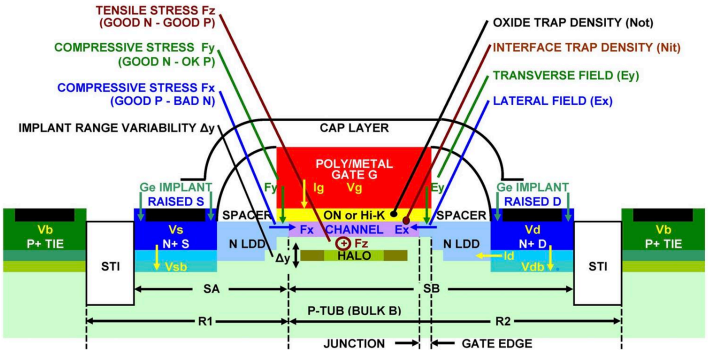


Fig. 2. *NMOS cross-section. In addition to stress from cap layers and Ge raised source-drain (S-D) implants, device dimensions such as distance from source-channel boundary to nearby STI (SA and SB), proximity and regularity of overlying metal patterns, and short distances to other device patterns within the local (< 2 μm) stress field induce transverse (F_y) and lateral (F_z and F_x) stress components, which affect threshold and mobility. Increasing the distance to P+ ties increases local tub (bulk) resistance components R1 and R2, which isolate the device MOS model substrate node from the device subcircuit symbol V_b node and degrade HF performance. Hot carrier reliability stress is dependent on the sum of transverse and lateral fields E_y and E_x. These fields are increased near the drain by increasing source to bulk (V_{sb}) and drain (V_d) to gate (V_g) or source (V_s) voltages in various combinations. As hot carrier stress increases, damage to channel from interface trap density (N_{it}) affects threshold and mobility, while gate oxynitirde (ON) or high-dielectric-constant (Hi-K) insulator trap density (N_{ot}) affects threshold and gate leakage.*

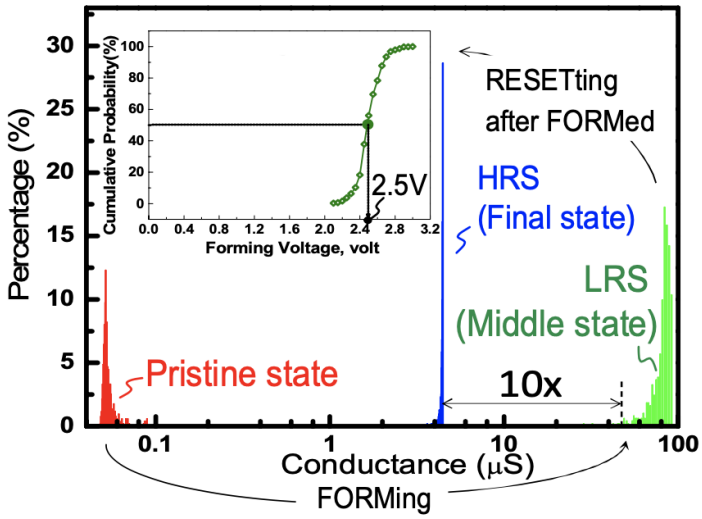
The transistors in a particular technology (from GlobalFoundries, TSMC, Samsung or others) have a maximum voltage that they can survive for a certain time. Exceed that time, or voltage, and the transistors die.

13.1.0.1 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 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.



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 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.

13.1.1 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

13.1.2 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.

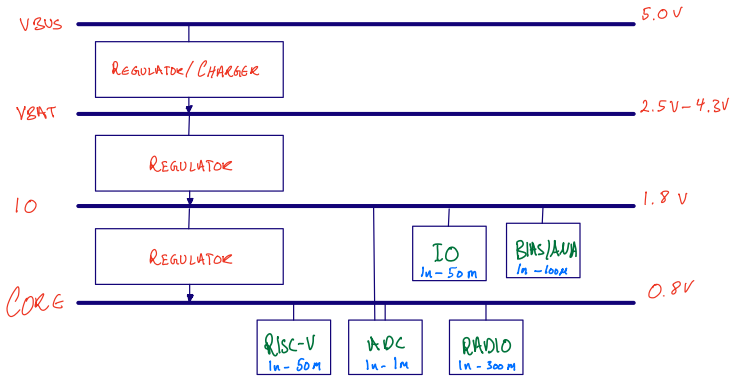
Voltage [V]
5.0
3.0
1.8
1.2

13.1.3 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.



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 picture below is from nRF5340 (page 23)

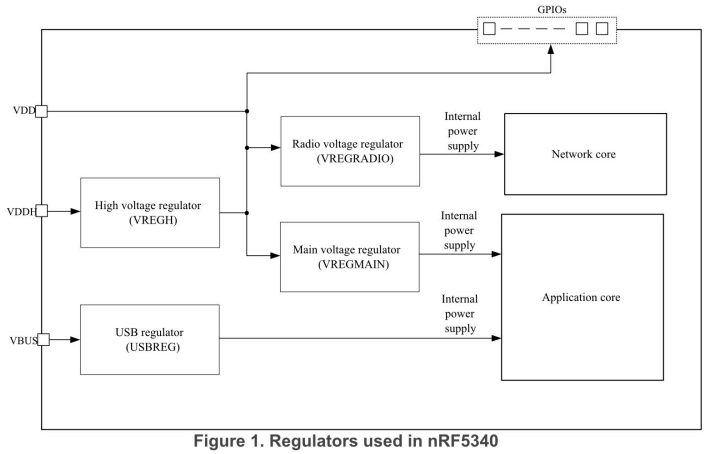
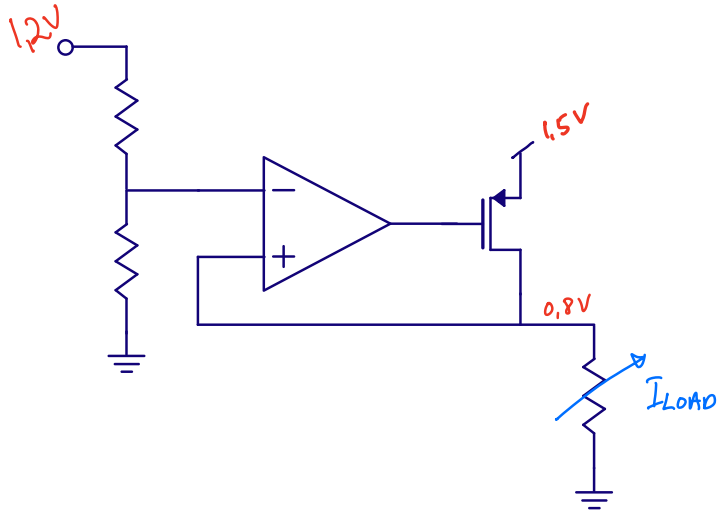


Figure 1. Regulators used in nRF5340

13.2 Linear Regulators

13.2.1 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.



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

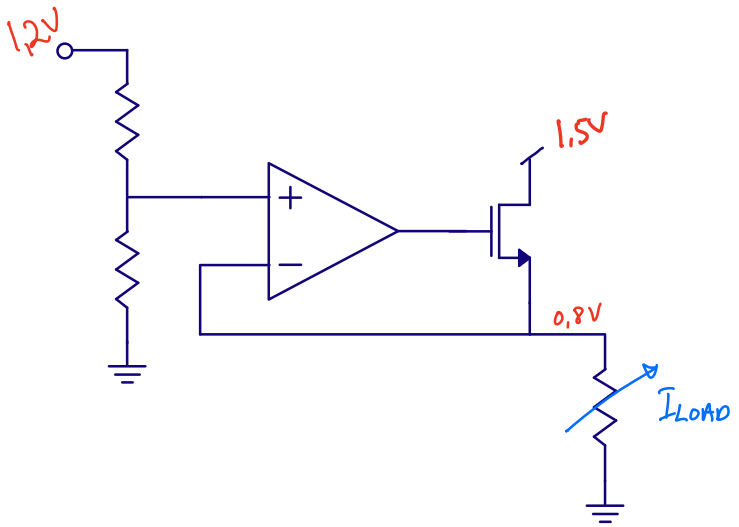
Parameter	Description	Unit
Load regulation	How much does the output voltage change with load current	V/A
Line regulation	How much does the output voltage change with input voltage	V/V
Power supply rejection ratio	What is the transfer function from input voltage to output voltage? The PSRR at DC is the 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.

13.2.2 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.



13.2.3 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

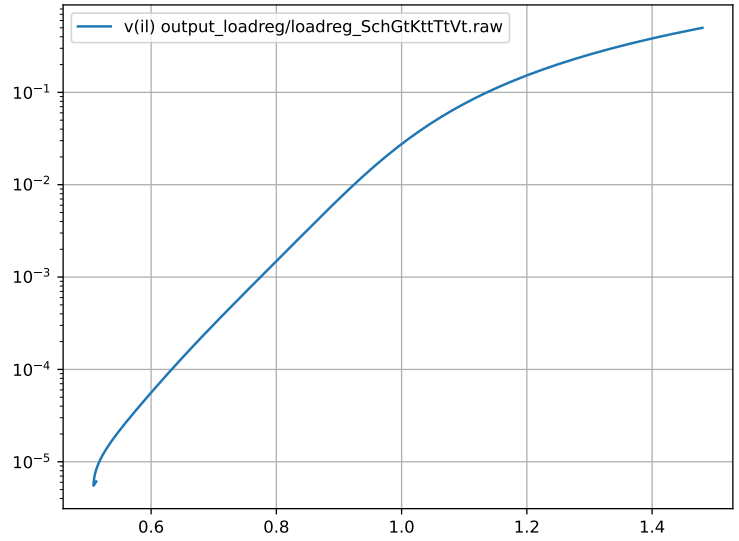
Below is an excerpt from the testbench. The pass-fet size has been determined by iteration.

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
```

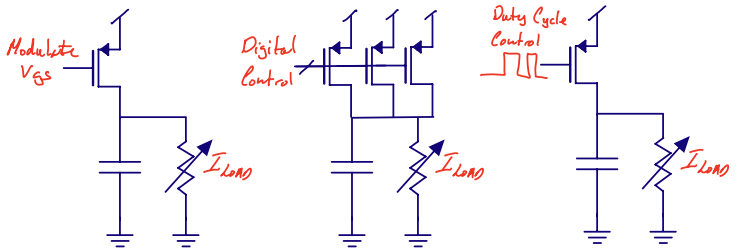
Below is a plot of the current on the y-axis as a function of the Vgs on the x-axis. Although it's possible to have almost 6 orders of magnitude change in current in the transistor it does become hard to make the loop stable over such a large range.

Sometimes it's easier to split the range into multiple ranges.



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.



13.3 Switched Regulators

Linear regulator 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 W/0.93 = 86 W, as such, only 6 W is wasted as heat. A temperature increase of 6 W × 20 °C/W = 120°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).

13.3.1 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

So before we go into details, let's have a look at the principles.

13.3.1.1 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

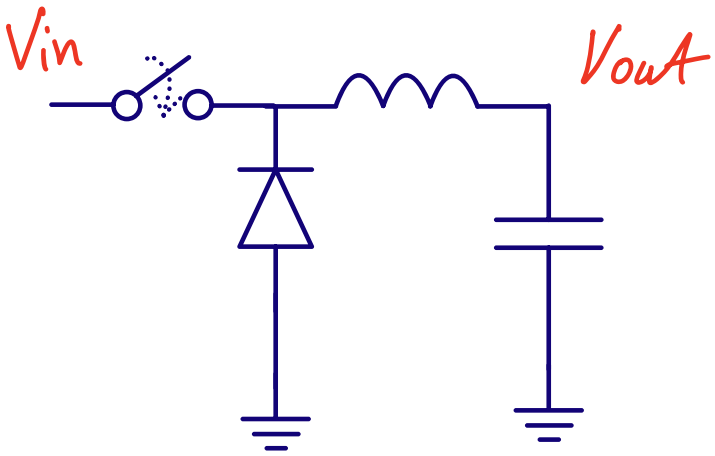
$$V = L\frac{dI}{dt}$$

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.



13.3.1.2 Capacitive BUCK DC/DC

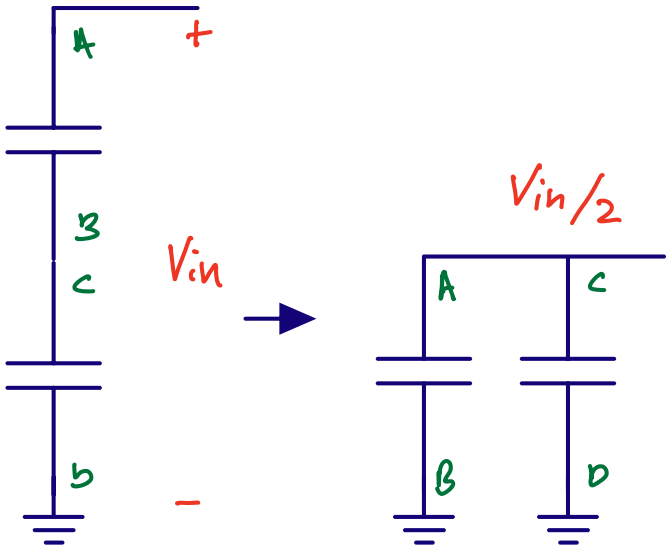
In a capacitive buck below what we're doing is charging two capacitors in series to a high voltage, Vin, and then re-configuring the capacitors to be in parallel.

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.

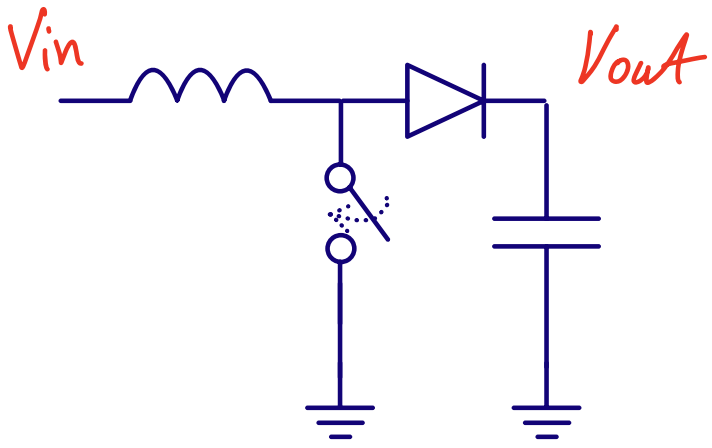
Often we have to insert an LDO after a capacitive buck to make the output voltage independent of input voltage.



13.3.1.3 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.



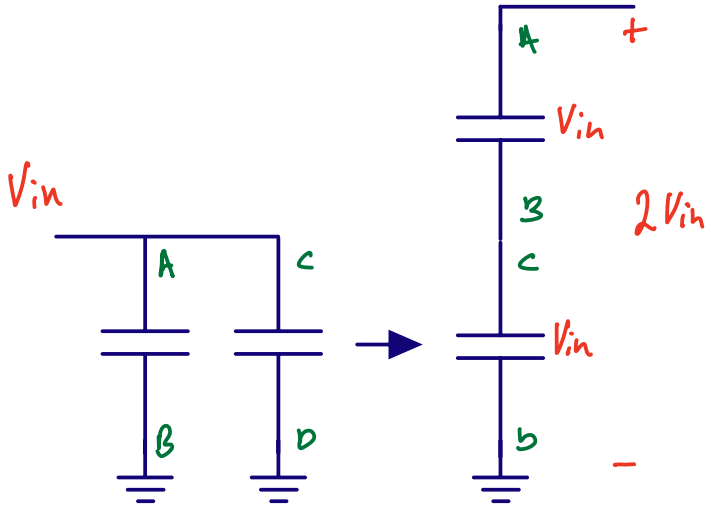
13.3.1.4 Capacitive BOOST DC/DC

In a capacitive boost we start with a parallel connection, charge the capacitors to Vin, then reconfigure the circuit to a series combina-

tion.

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.



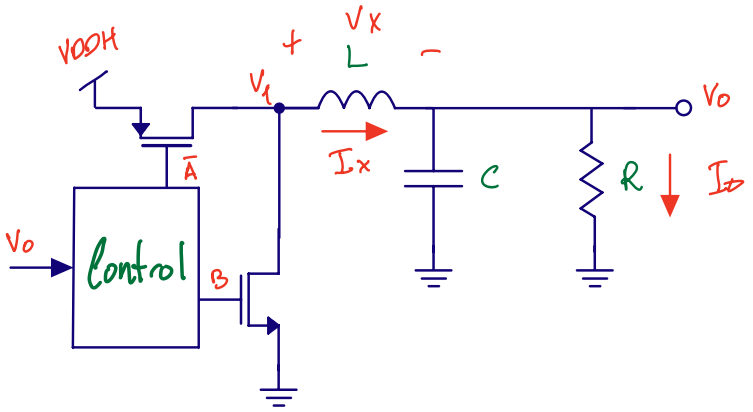
13.3.2 Inductive DC/DC converter details

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.



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.

13.3.3 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 $V_x = V_{VDDH} - V_O$. 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 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 $B = 1$.

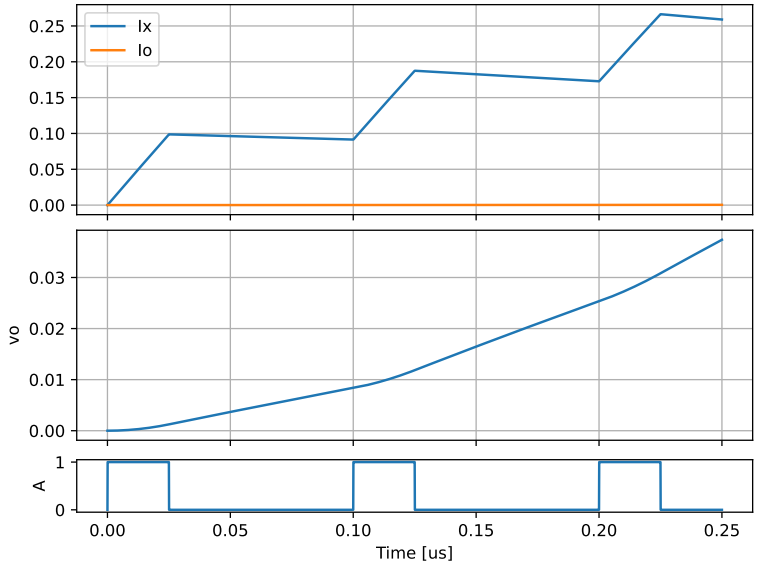
The V_x is now negative, so the current will decrease, however, since V_x is small, it does not decrease much.

I've made a

Jupyter PWM BUCK model

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.

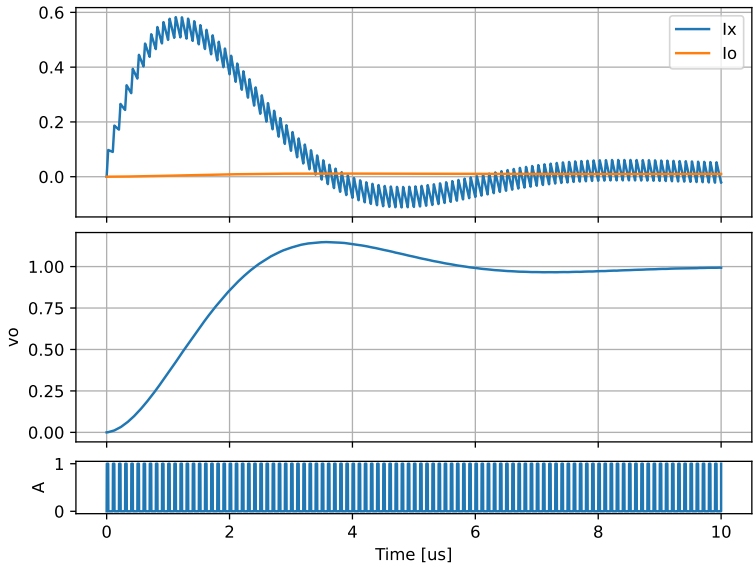


If we run the simulation longer, see plot below, the DC/DC will start to settle into a steady state condition.

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

$$V_o = V_{in} \times \text{Duty-Cycle}$$

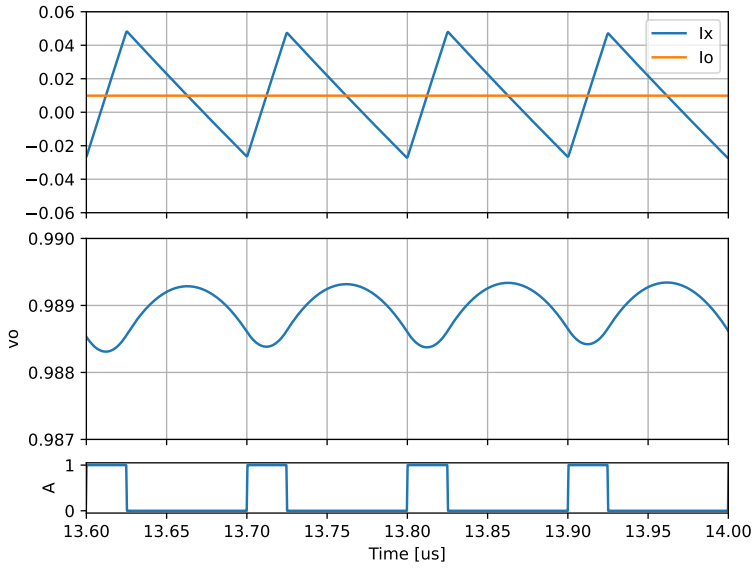
, where the duty-cycle is the ratio between the duration of $A = 1$ and $B = 1$.



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.

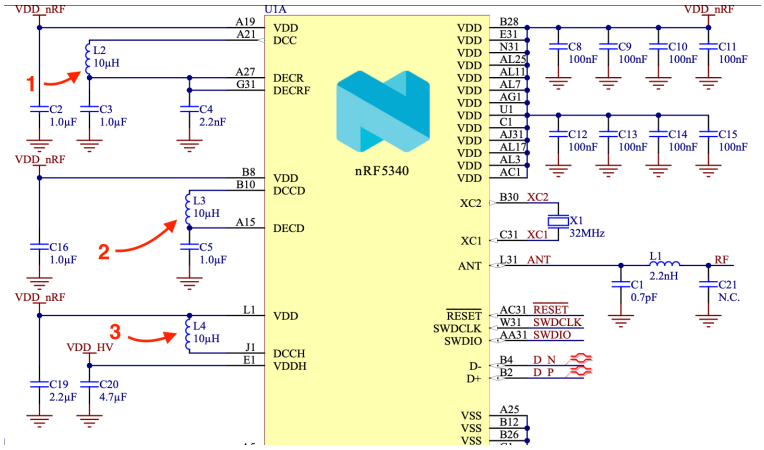
If the DC/DC was 100% efficient, then the current from the 4 V input supply would be 1/4'th of the 1 V output supply. 100% efficient DC/DC converters violate the laws of nature, as such, we can expect to get up to 9X% under optimal conditions.



13.3.4 Real world use

DC/DC converters are used when power efficiency is important. Below is a screenshot of the hardware description in the nRF5340 Product Specification.

We can see 3 inductor/capacitor pairs. One for the "VDDH", and two for "DECRF" and "DECD", as such, we can make a good guess there are three DC/DC converters inside the nRF5340.

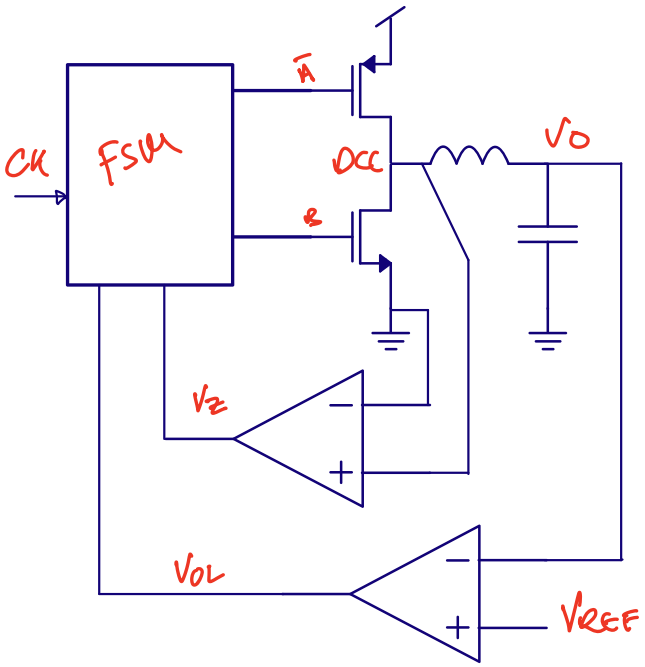


13.3.5 Pulsed Frequency Mode (PFM)

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 load's 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 impedant.



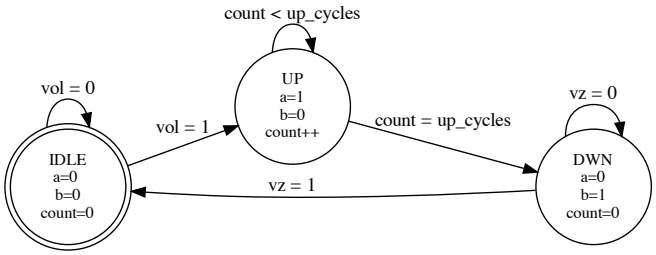
Imagine V_o is at 1 V, and we apply a constant output load. According to the integral equations the V_o would decrease linearly.

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 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 "DOWN" 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 $V_o < V_{REF}$.

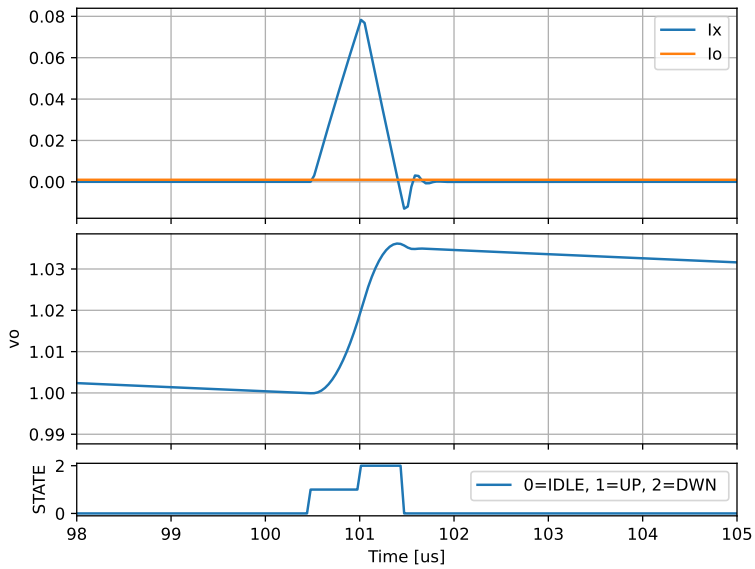


I think the name "pulsed-frequency mode" refers to the fact that the frequency changes according to load current, however, I'm not sure of the origin of the name. The name is not important. What's important is that you understand that mode 1 (PWM) and mode 2 (PFM) are two different "operation modes" 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



13.4 Want to learn more?

Search terms: regulator, buck converter, dc/dc converter, boost converter

13.4.1 Linear regulators

A Scalable High-Current High-Accuracy Dual-Loop Four-Phase Switching LDO for Microprocessors Overview of fancy LDO schemes, digital as well as analog

Development of Single-Transistor-Control LDO Based on Flipped Voltage Follower for SoC 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 Some insights into large power systems.

13.4.2 DC-DC converters

Design Techniques for Fully Integrated Switched-Capacitor DC-DC Converters 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 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 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 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 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.

Clocks and PLLs 14

Keywords: Systems, Feedback, PLL, Integer Divider, SD, SD PLL, Modulation, linear phase model

Status: 0.5

14.1 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.

14.1.1 A customer story

Take an example.

14.1.1.1 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 a 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 want's 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 an 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.

14.1.1.2 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.

14.1.2 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-harmnoics, where the fundamental is given by the period of the train.

14.1.3 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.

14.1.4 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.

14.1.5 Conclusion

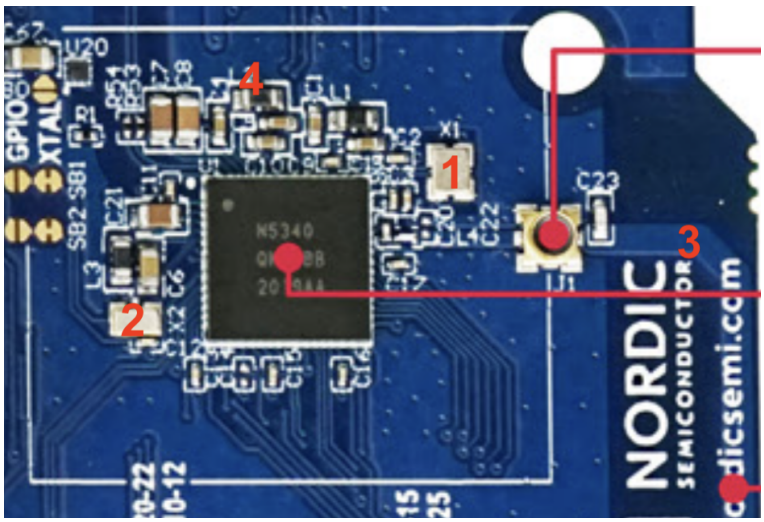
Each "clock problem" will have different frequency, noise and stability requirements. You must know the order of magnitude of those before you can design a clock source. There is no "one-solution fits all" clock generation IP.

14.2 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.

In the figure below you can see the following items.

1. 32 MHz crystal
2. 32 KiHz crystal
3. PCB antenna
4. DC/DC inductor



14.2.1 32 MHz crystal

Any Bluetooth radio will need a frequency reference. We need to generate an accurate 2.402 MHz - 2.480 MHz 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-accustic-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.

14.2.2 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.

14.2.3 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.

14.2.4 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.

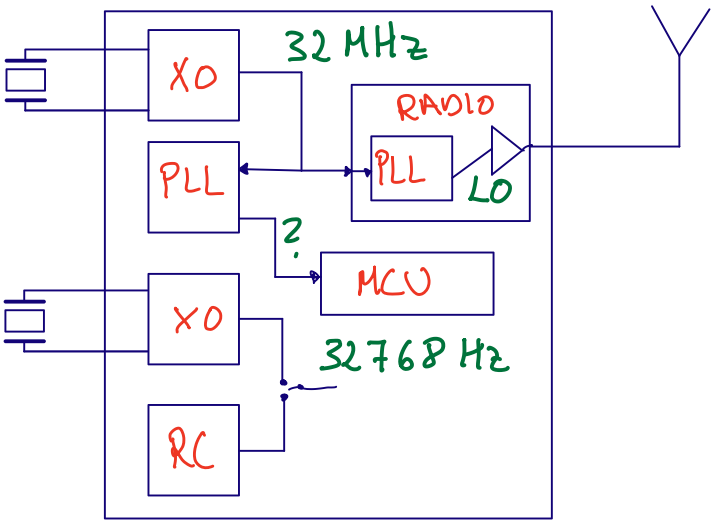
From our assumptions we could make a guess what must be inside the IC, something like the picture below.

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.



Most of the digital blocks on an IC will be synchronous logic, see figure below. A fundamental principle of sychnronous 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.

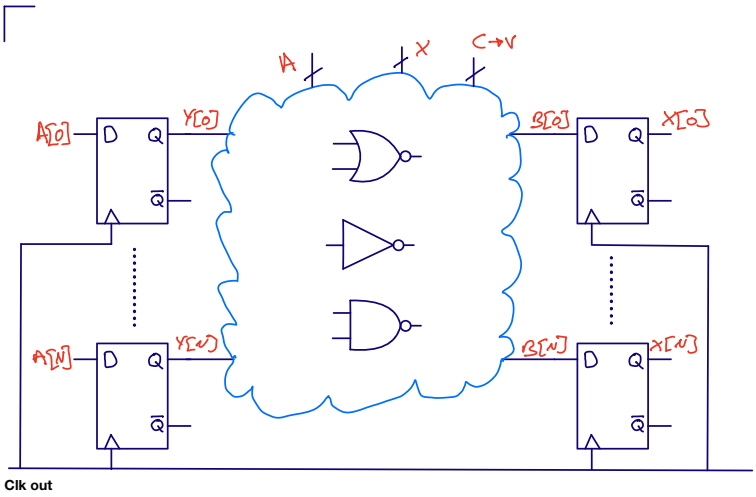
We also must check for all paths that the B[M] are held for long enough after the clock strikes such that our flip-flop does not change state. The hold time is the distance from the clock edge to where the data is allowed to change. Negative hold times are common in DFFs, so the data can start to change 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,

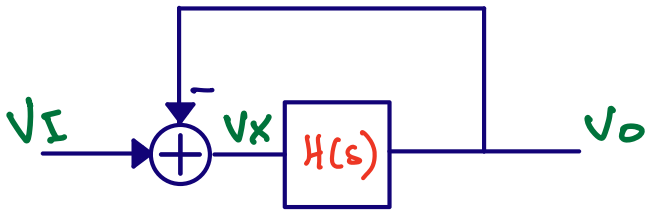
wire propagation delays, clock frequency (or period), and the variation in the clock frequency. The process, voltage, temperature variation must also be checked for all components, so the number of liberty files can quickly grow large.

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.



14.3 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.



$$V_I - V_o = V_x \qquad V_o = V_x H(s)$$

$$V_I = V_o + \frac{V_o}{H(s)} \quad \overset{H(s)=\infty}{\Rightarrow} \quad V_o = V_I$$

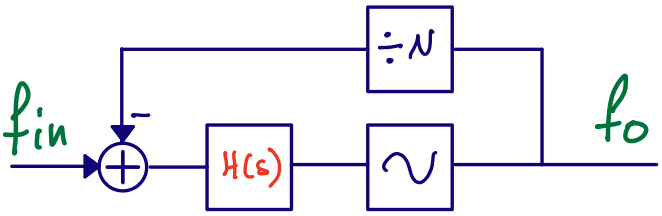
14.3.1 Integer PLL

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).

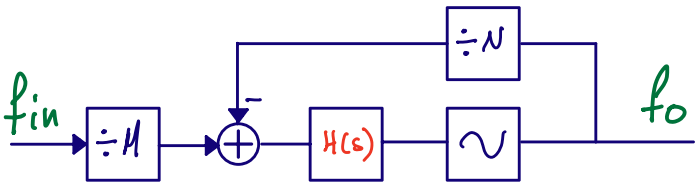
We then take the error, apply a transfer function $H(s)$ with high gain, and control our oscillator frequency.

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 $f_o = N \times f_{in}$



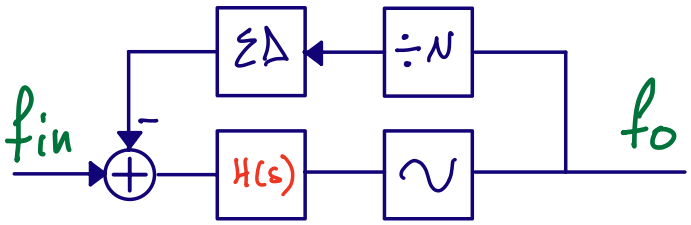
Sometimes you want a finer frequency resolution, in that case you'd add a divider on the reference and get $f_o = N \times \frac{f_{in}}{M}$..



14.3.2 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, integeres in a sigma-delta fashion in the divider.



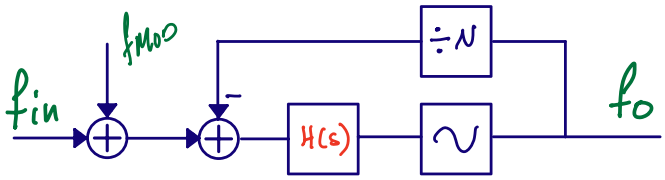
14.3.3 Modulation in PLLs

From your signal processing, or communication courses, you may recognize the equation below.

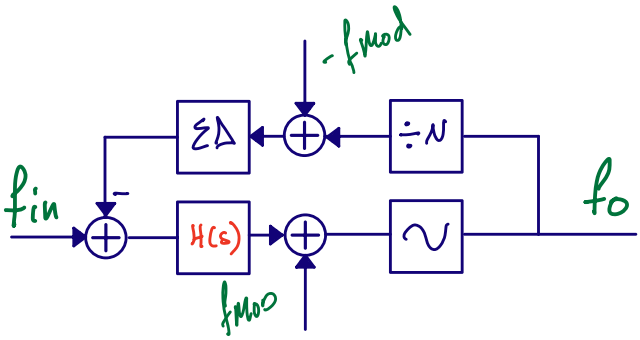
$$A_m(t) \times cos\left(2\pi f_{carrier} t + \phi_m(t)\right)$$

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.



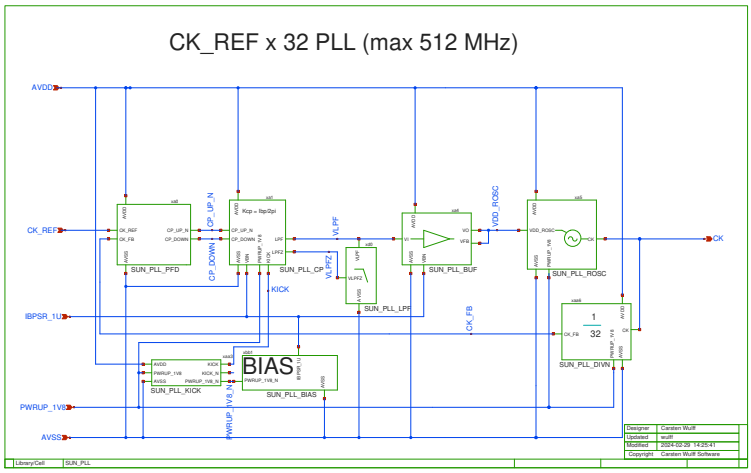
Most modern radios, however, will have a two-point modulation. The modulation signal is applied to the VCO (or DCO), and the opposite signal is applied to the feedback divider. As such, the modulation is not seen by the loop.



14.4 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 to 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 $H(s)$.



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!

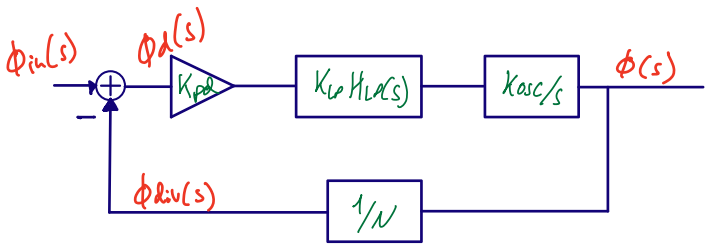
I have no idea who first thought of the idea, but it turns out, that one can model a PLL as a linear system if one consider the phase of the voltages inside the PLL, especially when the PLL is locked (phase of the output and reference is mostly aligned). Where the phase is defined as

$$\phi(t) = 2\pi \int_0^t f(t)dt$$

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.



14.4.1 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 approaces 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 translate can translate 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_{vco}, which must be determined by simulation.

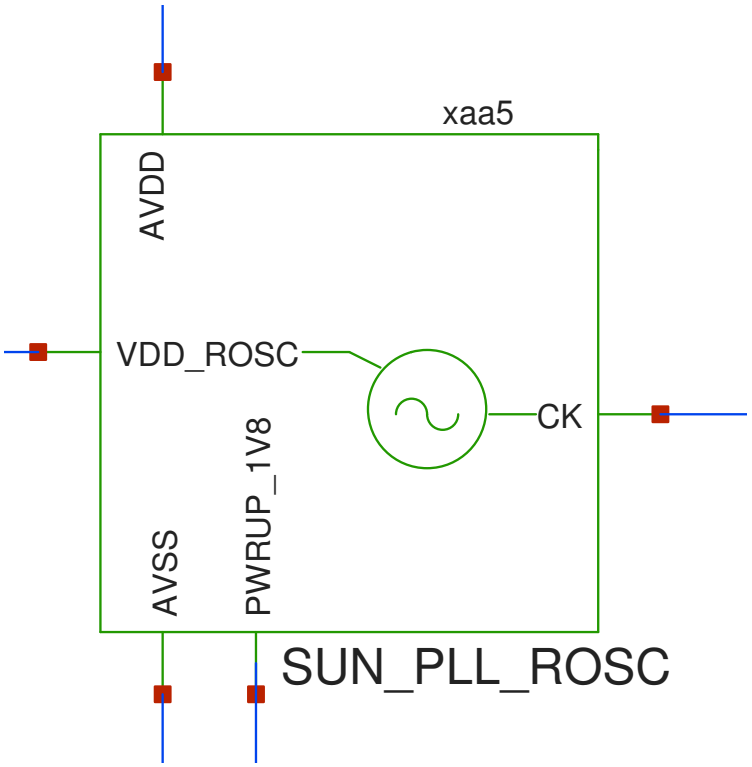
14.4.2 Controlled oscillator

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.

$$K_{osc} = 2\pi \frac{df}{dV_{cntl}}$$



14.4.2.1 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
```

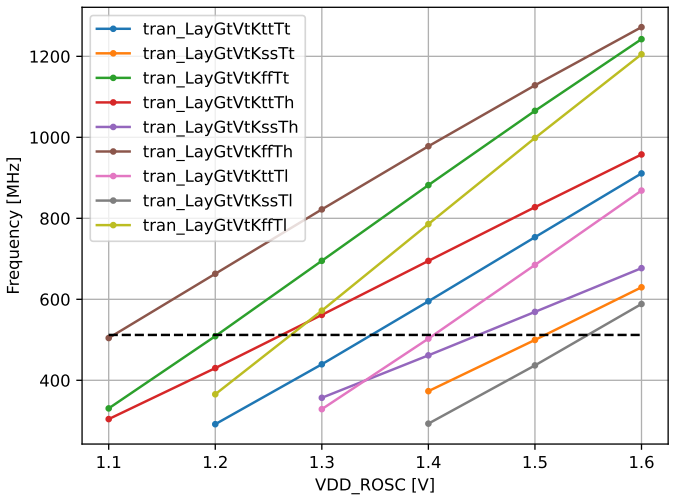
I use `tran.py` to extract the time-domain signal from ngspice into a CSV file.

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.

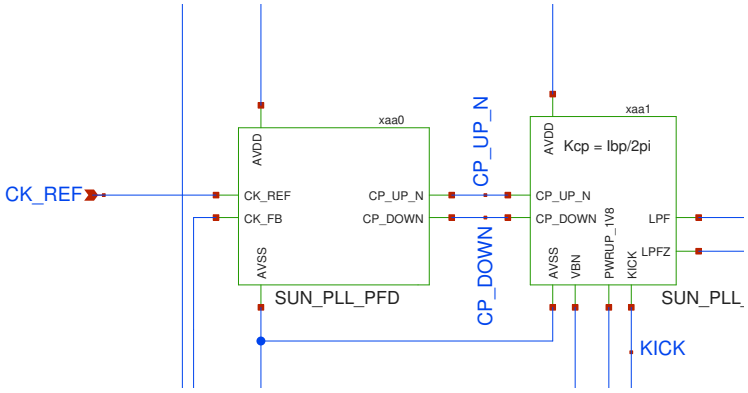


14.4.3 Phase detector and charge pump

The gain of the phase-detector and charge pump is the current we feed into the loop filter over a period. I don't remember why, check in the book for a detailed description.

The two blocks compare our reference clock to our feedback clock, and produce an error signal.

$$K_{pd} = \frac{I_{cp}}{2\pi}$$



14.4.4 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.

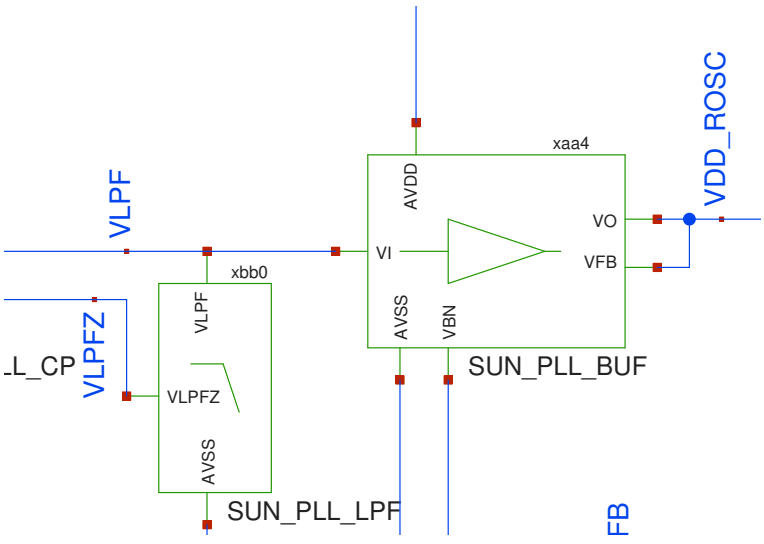
I would start with the "known to work" loop filters before you explore on your own.

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 VPLF node is high impedant, so I can't draw current from the loop filter without changing the filter transfer function.

$$K_{lp}H_{lp}(s) = K_{lp}\left(\frac{1}{s} + \frac{1}{\omega_z}\right)$$

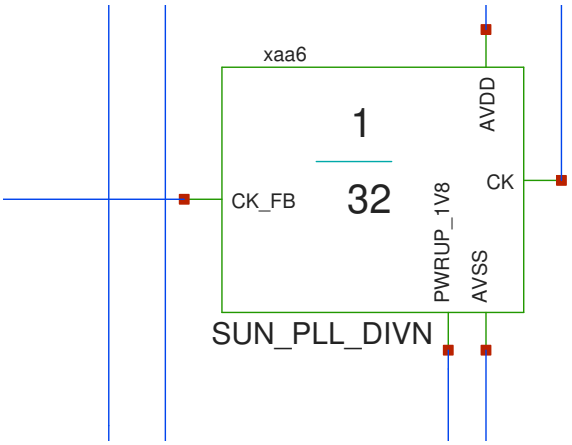
$$K_{lp}H_{lp}(s) = \frac{1}{s(C_1 + C_2)}\frac{1 + sRC_1}{1 + sR\frac{C_1C_2}{C_1+C_2}}$$



14.4.5 Divider

The divider is modelled as

$$K_{div} = \frac{1}{N}$$



14.4.6 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.

$$L(s) = \frac{K_{osc} K_{pd} K_{lp} H_{lp}(s)}{Ns}$$

14.4.6.1 Python model

I've made a python model of the loop, you can find it at sun_pll_-sky130nm/jupyter/pll

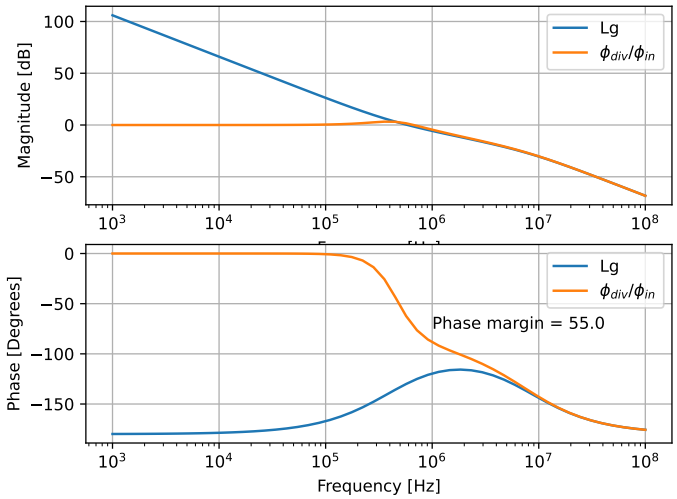
In the jupyter notbook below you can find some more information on the phase/frequency detector, and charge pump.

sun_pll_sky130nm/jupyter/pfd

Below is a plot of the loop gain, and the transfer function from input phase to divider phase.

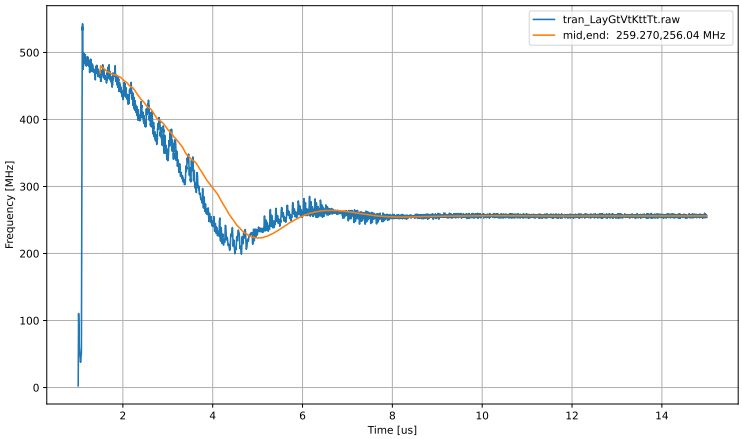
We can see that the loop gain at low frequency is large, and proportional to $1/s$. As such, the phase of the divided down feedback clock is the same as our reference.

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.



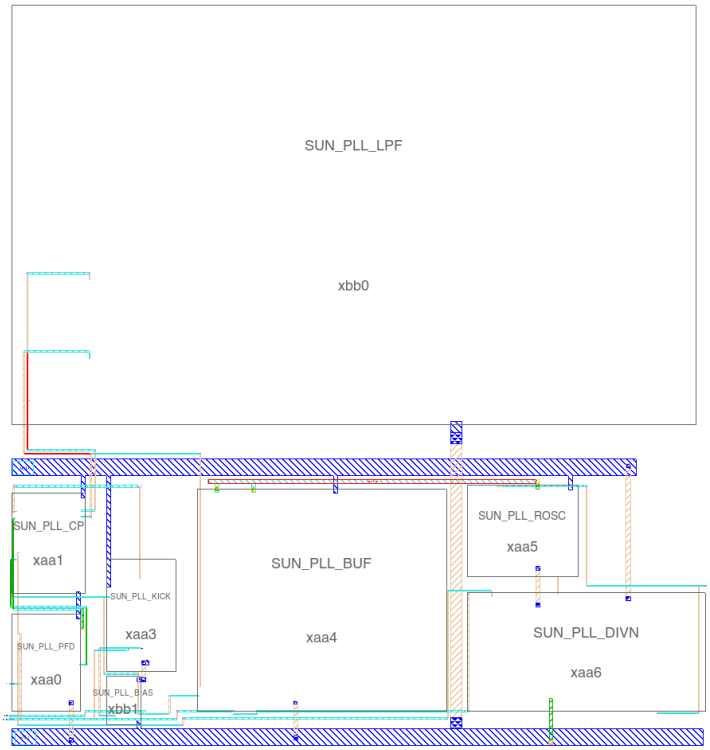
The top testbench for the PLL is tran.spi.

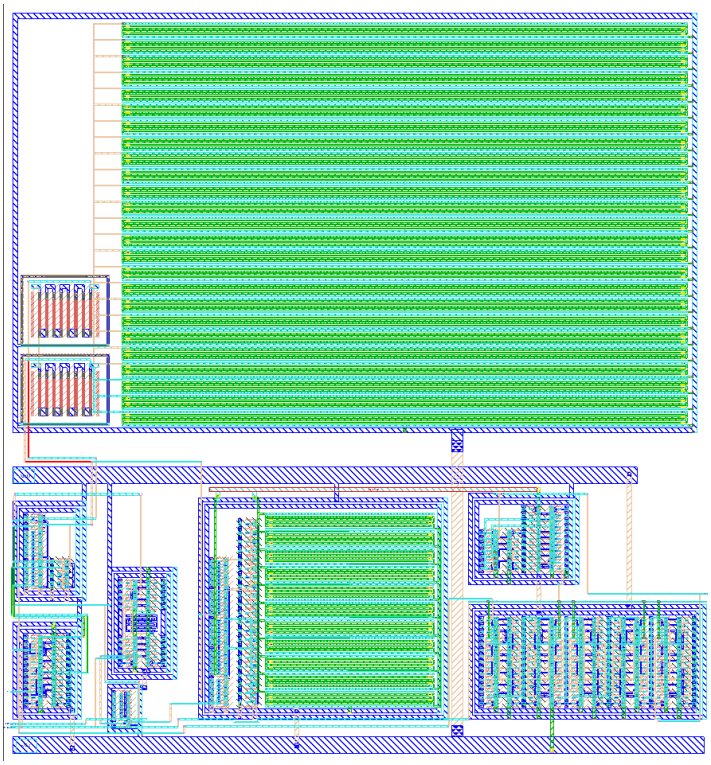
I power up the PLL and wait for the output clock to settle. I use freq.py to plot the frequency as a function of time. The orange curve is the average frequency. We can see that the output frequency settles to 256 MHz.



You can find the schematics, layout, testbenches, python script etc at SUN_PLL_SKY130NM

Below are a couple layout images of the finished PLL





14.5 Want to learn 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

All-digital PLL and transmitter for mobile phones

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

15 Oscillators

Keywords: Crystal model, Pierce, Temperature, Controlled oscillator, Ring osc, Ictrl Rosc, DCO Ring, LCOSC, RCOSC

Status: 0.5

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.

15.1 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.

15.1.1 Microchip 5071B Cesium Primary Time and Frequency Standard

One example of a ultra precise time piece is shown below. 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

15.1.2 Rubidium standard

Rubidium standard, use the rubidium hyper-fine transition of 6.8 GHz (6834682610.904 Hz)

and can actually be made quite small. Below is a picture of a tiny atomic clock. According to the marketing blurb:

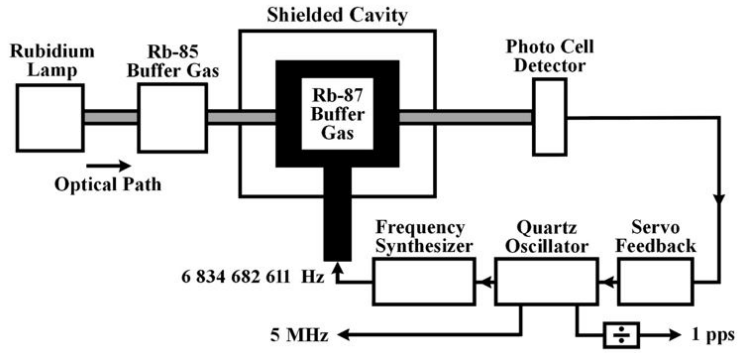
The MAC is a passive atomic clock, incorporating the interrogation technique of Coherent Population Trapping (CPT) and operating upon the D1 optical resonance of atomic Rubidium Isotope 87.

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 (freqency synthesizer) to generate the exact frequency needed.

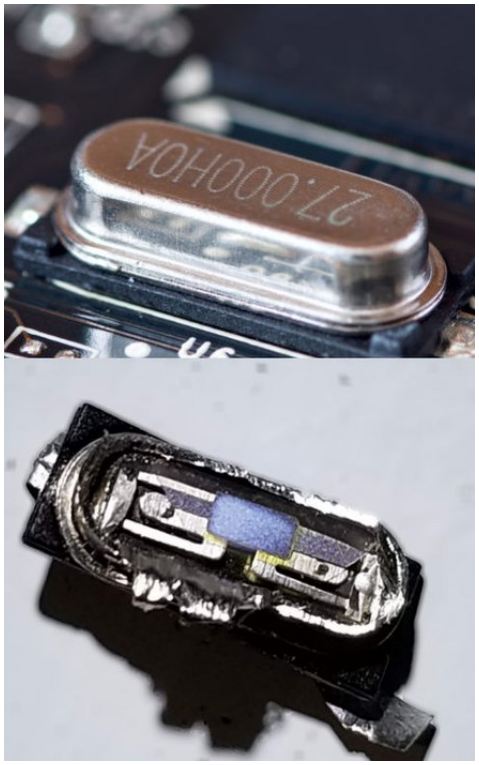
The negative feedback loop ensures that the 5 MHz clock coming out is proportional to the hyper-fine energy levels in the Rubidium atoms. Negative feedback is cool! Especially when we have a pole at DC and infinite gain.



15.2 Crystal oscillators

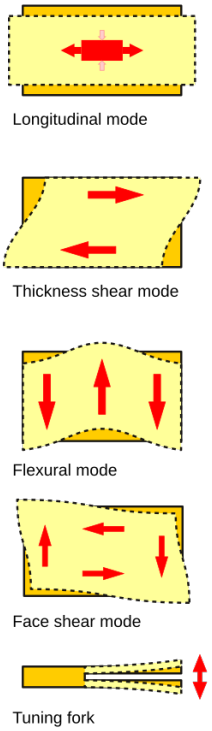
For accuracy's of parts per million, which is sufficient for your wrist watch, or most communication, it's possible to use crystals.

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



The vibrations in the crystal lattice can have many modes, as illustrated by figure below.

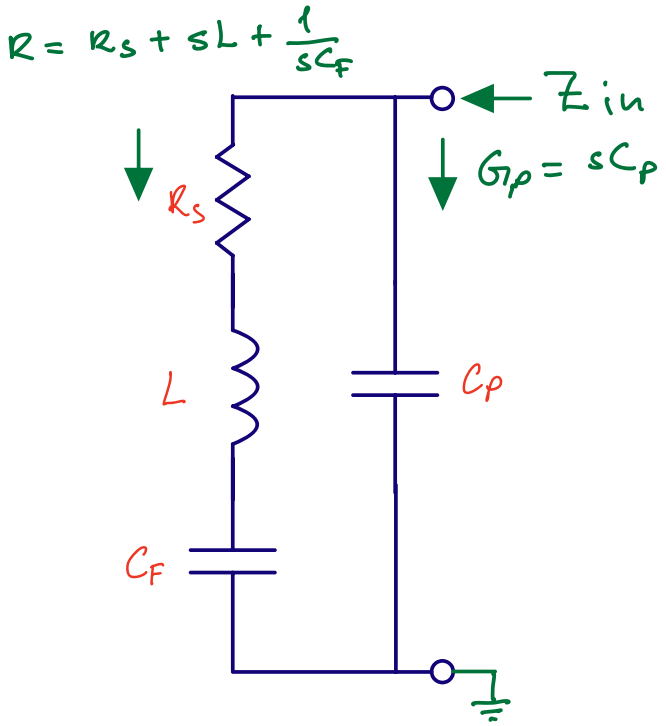
All we need to do with a crystal is to inject sufficient energy to sustain the oscillation, and the resonance of the crystal will ensure we have a correct enough frequency.



15.2.1 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.



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 impedant at DC.

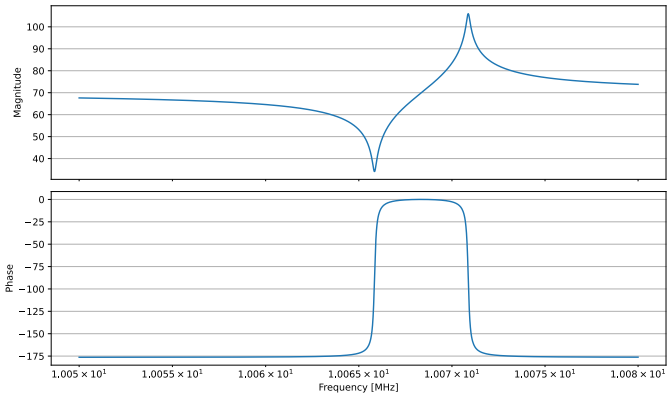
Since the 1/(sCp) does not change much at resonance, then

$$Z_{in} \approx \frac{L C_F s^2 + 1}{L C_F C_p s^2 + C_F + C_P}$$

See Crystal oscillator impedance for a detailed explanation.

In the impedance plot below we can clearly see that there are two "resonance" points. Usually noted by series and parallel resonance.

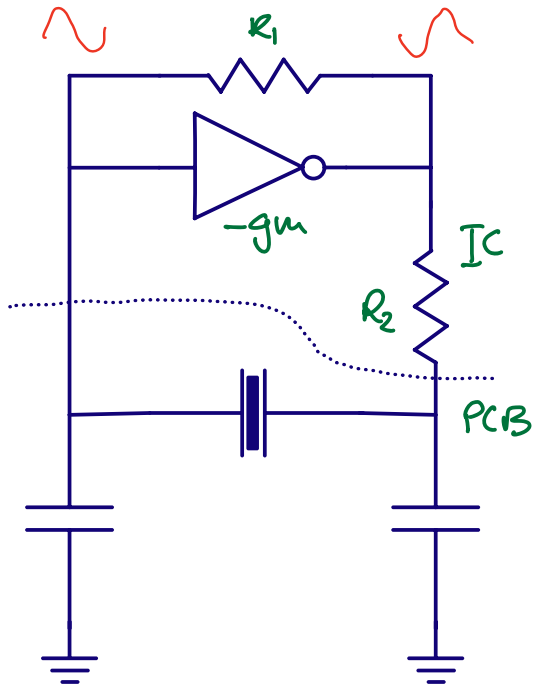
I would encourage you to read The Crystal Oscillator for more details.



15.2.2 Circuit

Below is a common oscillator circuit, a Pierce Oscillator. The crystal is the below the dotted line, and the two capacitance's are the on-PCB capacitance's.

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).



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 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.

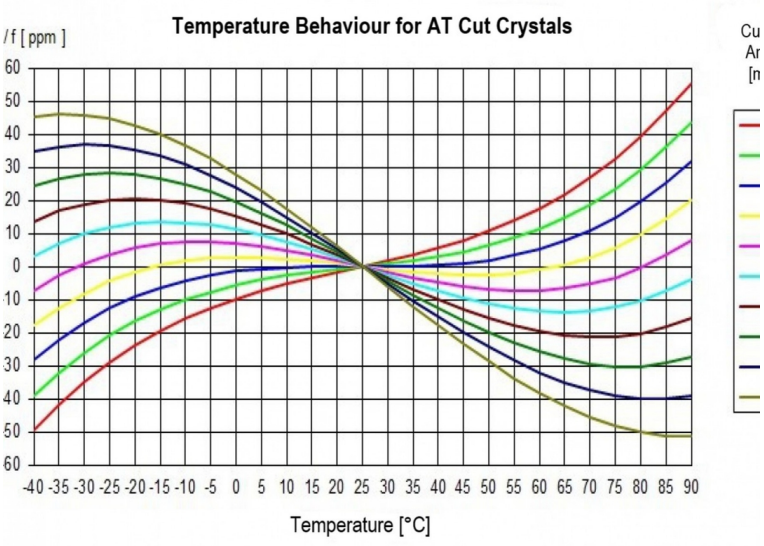
Can fine tune frequency with parasitic capacitance

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 capacitance's to support multiple crystal vendors without changing the PCB.

15.2.3 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.



15.3 Controlled Oscillators

On an integrated circuit way 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.

15.3.1 Ring oscillator

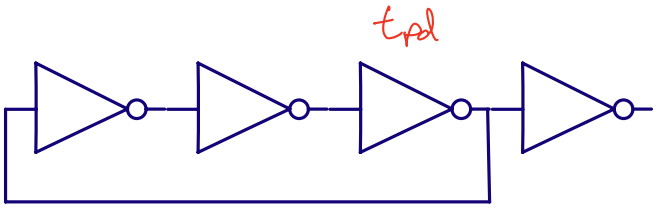
The simplest oscillator is a series of inverters biting their own tail, a ring oscillator.

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.

$$t_{pd} \approx RC$$

$$R \approx \frac{1}{gm} \approx \frac{1}{\mu_n C_{ox} \frac{W}{L}(VDD - V_{th})}$$

$$C \approx \frac{2}{3} C_{ox} WL$$



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" of 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, then number of inverter stages should be $\in [3, 5, 7, 11, 13, 17, 19, 23, 29, 31]$

15.3.2 Capacitive load

The oscillation frequency of the ring oscillator can also be changed by adding capacitance.

$$f = \frac{\mu_n C_{ox} \frac{W}{L}(VDD - V_{th})}{2N\left(\frac{2}{3}C_{ox}WL + C\right)}$$

$$K_{vco} = \frac{2\pi\mu_n C_{ox} \frac{W}{L}}{2N\left(\frac{2}{3}C_{ox}WL + C\right)}$$

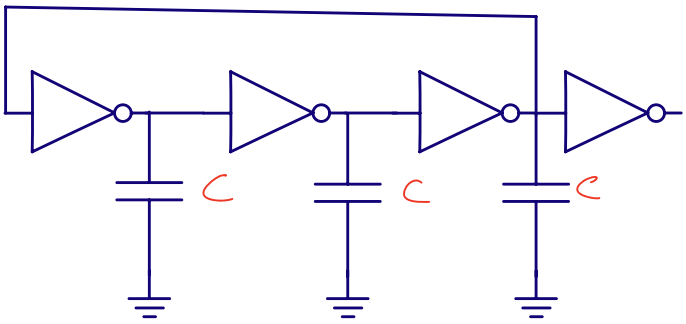
Assume that the extra capacitance is much larger than the gate capacitance, then

$$f = \frac{\mu_n C_{ox} \frac{W}{L}(VDD - V_{th})}{2NC}$$

$$K_{vco} = \frac{2\pi\mu_n C_{ox} \frac{W}{L}}{2NC}$$

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.



15.3.3 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.

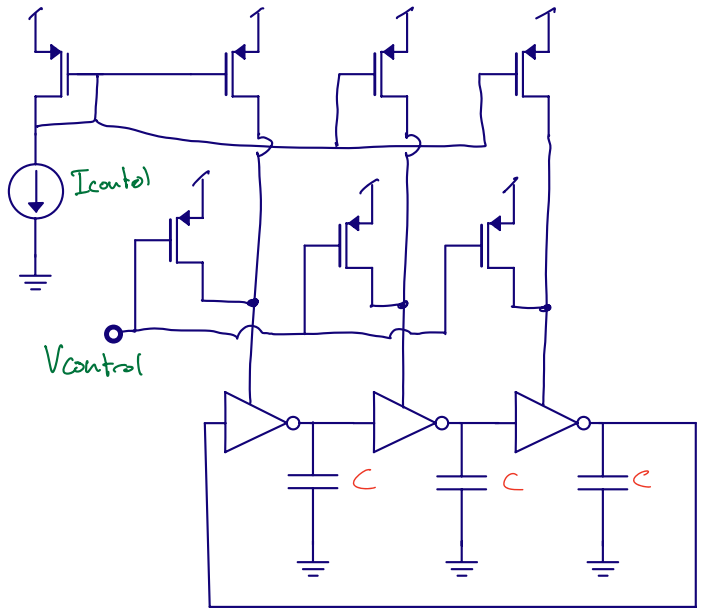
It's also possible to run a counter on the output frequency of the VCO, and count the edges between two reference clocks. That way we can get a precise estimate of the oscillation frequency.

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 recievers that use a PLLs to directly demodulate the incoming frequency shift keyed modulated carrier.



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.

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

$$f \approx \frac{I_{control} + \frac{1}{2}\mu_p C_{ox}\frac{W}{L}(VDD - V_{control} - V_{th})^2}{C\frac{VDD}{2}N}$$

$$K_{vco} = 2\pi\frac{\partial f}{\partial V_{control}}$$

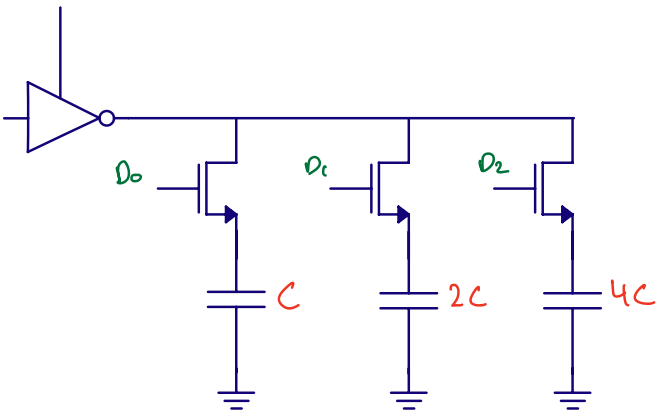
$$K_{vco} = 2\pi \frac{\mu_p C_{ox} W/L}{C\frac{VDD}{2}N}$$

15.3.4 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.



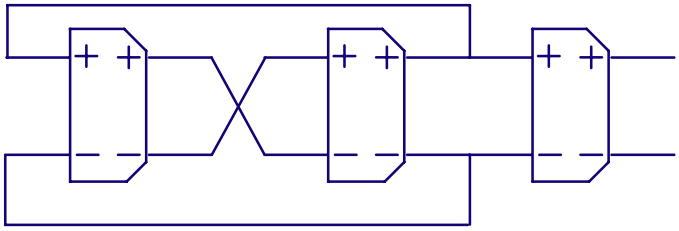
15.3.5 Differential

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.

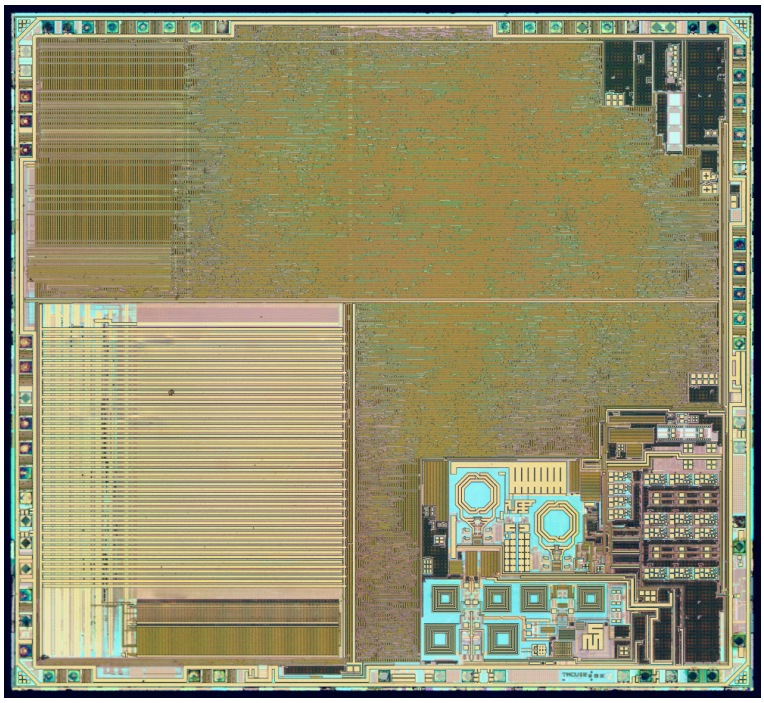


15.3.6 LC oscillator

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.

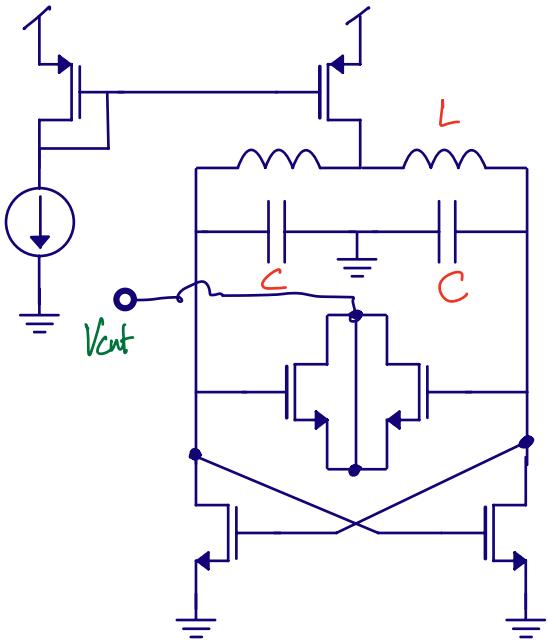
Most ring oscillators are too high noise for radio's, we must use a inductor and capacitor to create the resonator.

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 reciever



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"

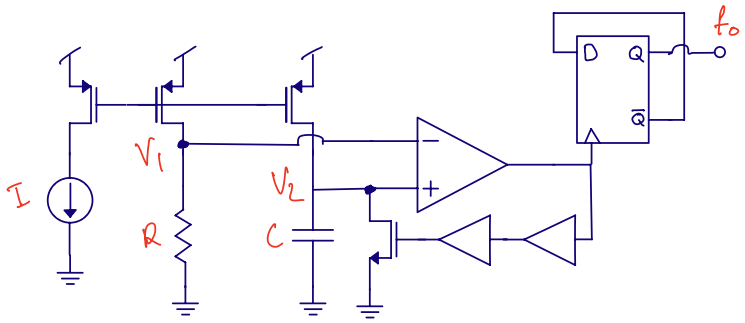
The NMOS at the bottom provide the "negative transconductance" to compensate for the loss in the LC tank.



$$f \propto \frac{1}{\sqrt{LC}}$$

15.4 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.



$$V_1 = IR$$

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

$$dt = \frac{CV_2}{I} = \frac{CIR}{I}$$

$$f = \frac{1}{dt} = \frac{1}{RC}$$

$$f_o = \frac{1}{2}f = \frac{1}{2RC}$$

15.5 Want to learn more?

15.5.1 Crystal oscillators

The Crystal Oscillator - A Circuit for All Seasons

High-performance crystal oscillator circuits: theory and application

Ultra-low Power 32kHz Crystal Oscillators: Fundamentals and Design Techniques

A Sub-nW Single-Supply 32-kHz Sub-Harmonic Pulse Injection Crystal Oscillator

15.5.2 CMOS oscillators

The Ring Oscillator - A Circuit for All Seasons

A Study of Phase Noise in CMOS Oscillators

An Ultra-Low-Noise Swing-Boosted Differential Relaxation Oscillator in 0.18-um CMOS

Ultra Low Power Frequency Synthesizer

Low Power Radio 16

Keywords: Range, Antenna Size, Modulation, OFDM, GFSK, pi/4-qpsk, 8-psk, 16 QAM, Bluetooth LE, LP RX, LNA, MIxer, AAF, ADC, BB

Status: 0.5

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.

16.1 Data Rate

16.1.1 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 reciever 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	
CRC	4	Bit errors

What	Bits	Why
Buttons	16	One-hot coding. Most mice have buttons
Preamble	8	Synchronization
Address	32	Unique identifier
Total	76	

16.1.2 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

16.1.3 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
```

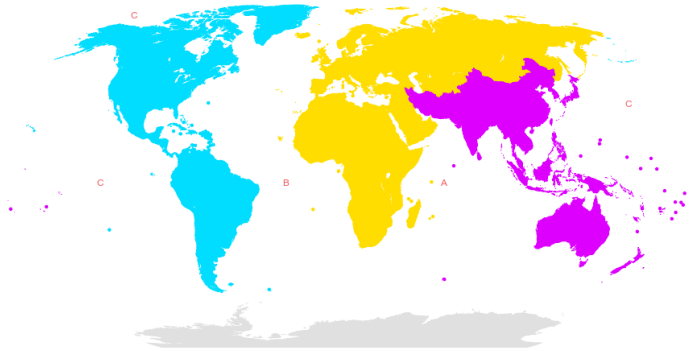
The above statements are a exact copy of what happens in industry when we start design of something. We make an educated guess and multiply by a number. More optimistic people would multiply with e.

16.2 Carrier Frequency & Range

16.2.1 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.



- 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

16.2.2 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 = light-speed/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:

16.2.3 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 reciever 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

$$D = 10^{\frac{P_{TX} - P_{RX} + 20log_{10}\left(\frac{c}{4\pi f}\right)}{20}}$$

16.2.4 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	$20log_{10}\left(c/4\pi f\right)$ [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:

16.2.5 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.

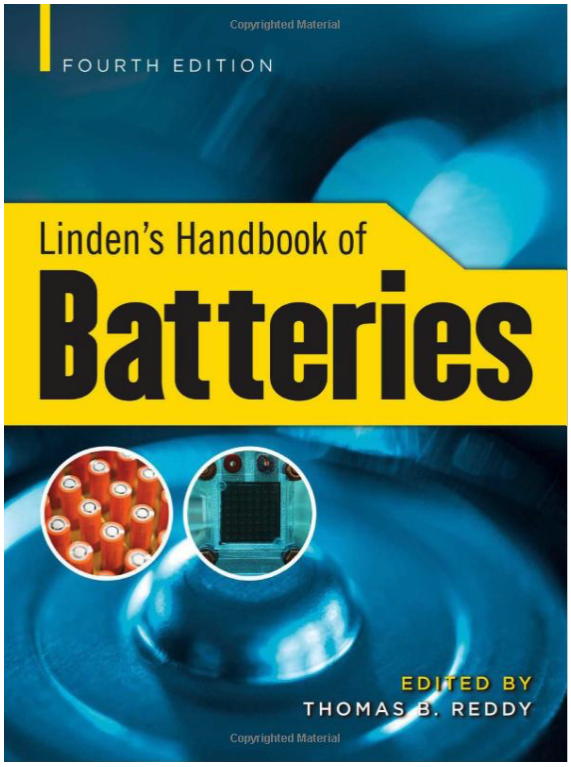
	$20log_{10}\left(c/4\pi f\right)$			
Freq	[dB]	D@n=2 [m]	D@n=6 [m]	OK/NOK
2.45 GHz	**-40.2**	**97.4**	**4.6**	:white_-check_mark:
5.80 GHz	-47.7	41.2	3.45	:white_-check_mark:
24.12 GHz	-60.1	9.9	2.1	:x:

16.3 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, see picture below. It's worth a read if you want to dive deeper into chemistry and properties of primary (non-chargeable) and secondary (chargeable) cells.

16.3.1 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

16.4 Decisions

Now we know that we need a 1 Mbps radio at 2.4 GHz that runs of 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?

16.4.1 Modulation

Any modulation can be described by the function below.

$$A_m(t) \times cos\left(2\pi f_{carrier}(t)t + \phi_m(t)\right)$$

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	> 1024 bits/symbol	Really non-constant envelope

16.4.2 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 light we send the phase and amplitude is usually real.

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.

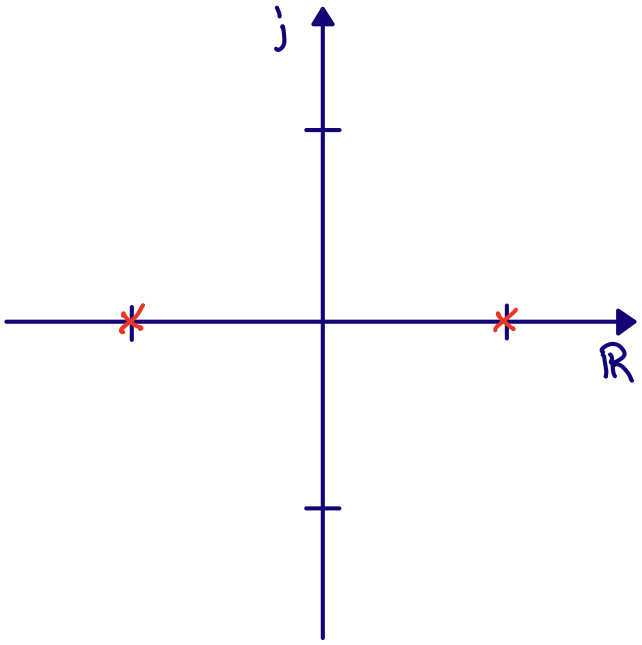
The equation for the unit circle is $y = e^{i(\omega t+\phi)}$ where ϕ is the phase, and ω is the angular frequency.

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

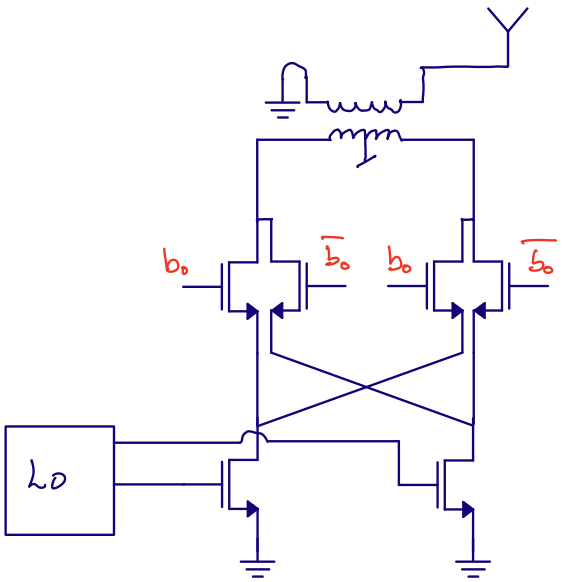
rotation was slightly slower, the red dot would move the other way around the circle.

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.

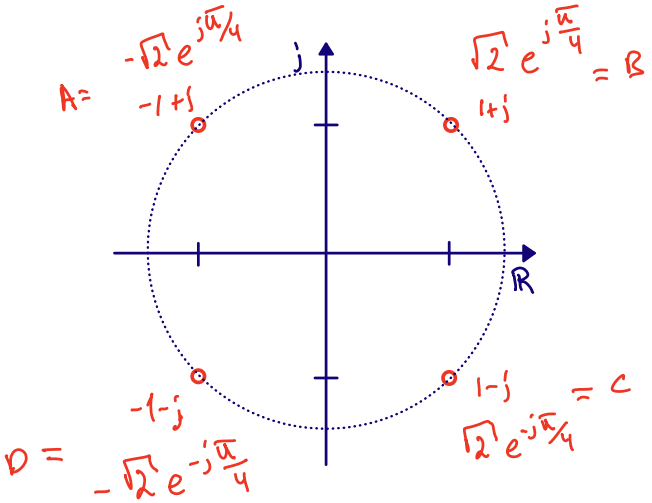


There is another way to change phase 180 degrees, and that's simply to swap the phase in the transmitter circuit. Imagine as below we have a local oscillator driving pseudo differential common source stages with switches on top. If we flip the switches we can change the phase 180 degrees pretty fast.

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.

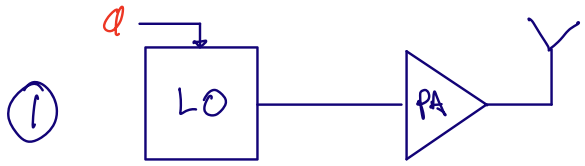


Standards like Zigbee used offset quadrature phase shift keying, with a constellation as shown below. With 4 points we can send 2 bits per symbol.

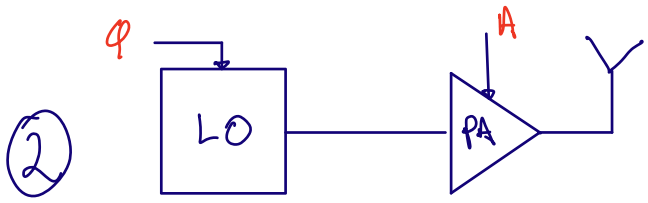


In ZigBee, or 802.15.4 as the standard is called, the phase changes is actually done with a constant envelope.

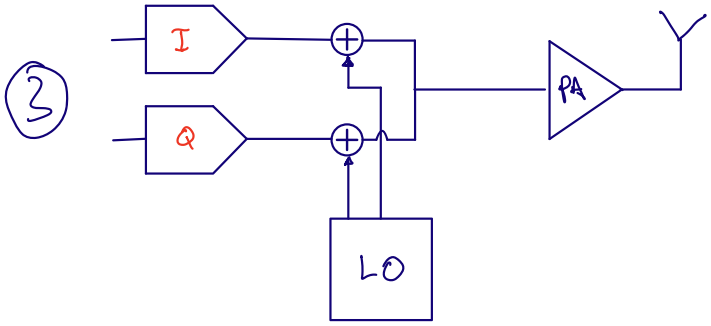
The nice thing about constant envelope is that the radio transmitter can be simple. We don't need to change the amplitude. If we have a PLL as a local oscillator, where we can change the phase (or frequency), then we only need a power amplifier before the antenna.



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.

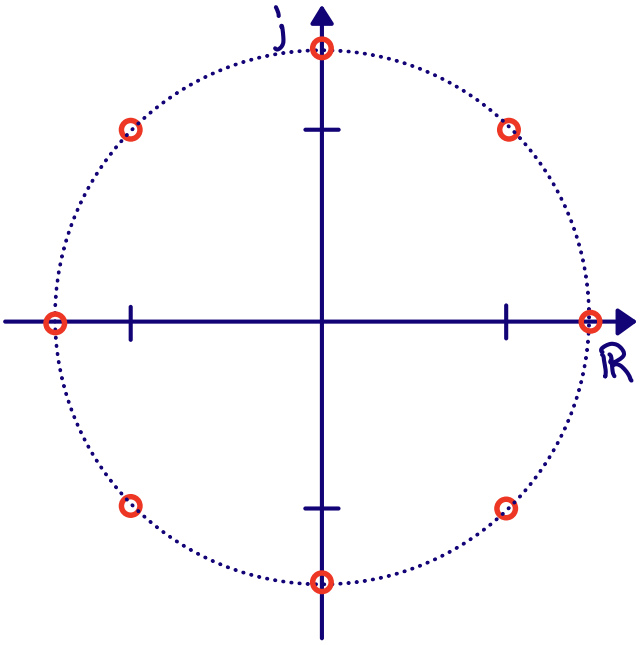


Or a Cartesian architecture where we make the in-phase component, and quadrature-phase components in digital, then use two digital to analog converters, and a set of complex mixers to encode onto the carrier. The power amplifier would not need to change the amplitude, but it does need to be linear.



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.



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 reciever 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 reciever 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.

BLUETOOTH CORE SPECIFICATION Version 5.3 | Vol 2, Part A page 394

Radio Specification

GFSK modulation mode, whereas the subsequent synchronization sequence, payload, and trailer sequence are transmitted using the Enhanced Data Rate PSK modulation mode.

3.2.1 Modulation characteristics

During access code and packet header transmission the Basic Rate GFSK modulation mode shall be used. During the transmission of the synchronization sequence, payload, and trailer sequence a PSK type of modulation with a data rate of 2 Mb/s or optionally 3 Mb/s shall be used. The following subsections specify the PSK modulation for this transmission.

3.2.1.1 Modulation method overview

The PSK modulation format defined for the 2 Mb/s transmission shall be π/4 rotated differential encoded quaternary phase shift keying (π/4-DQPSK).

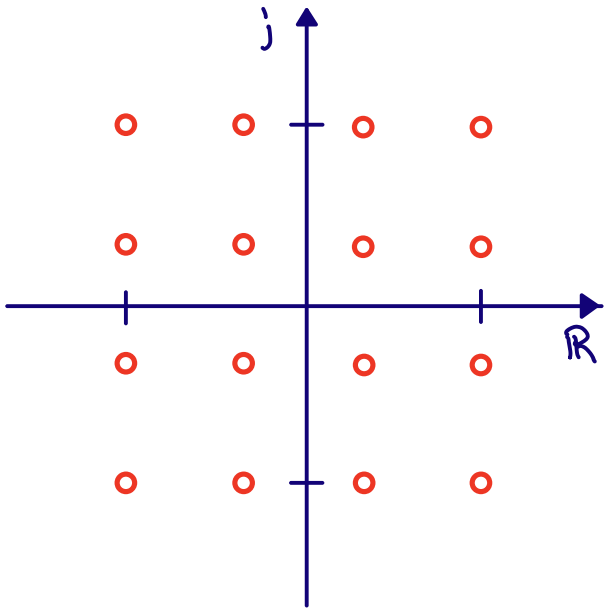
The PSK modulation format defined for the 3 Mb/s transmission shall be differential encoded 8-ary phase shift keying (8DPSK).

The modulation shall employ square-root raised cosine pulse shaping to generate the equivalent lowpass information-bearing signal v(t). The output of the transmitter shall be a bandpass signal that can be represented as

$$S(t) = Re\left[v(t)e^{j2\pi F_c t}\right] \quad \text{(EQ 1)}$$

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.

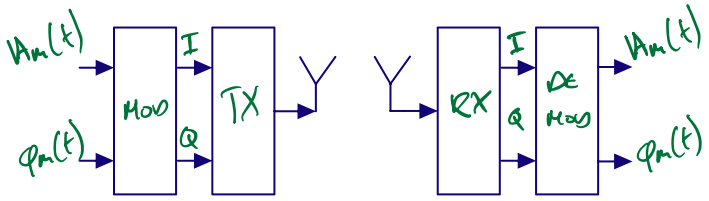
If you wanted to research "new fancy modulation schemes" I'd think about Sphere packing.



16.4.3 Single carrier, or multi carrier?

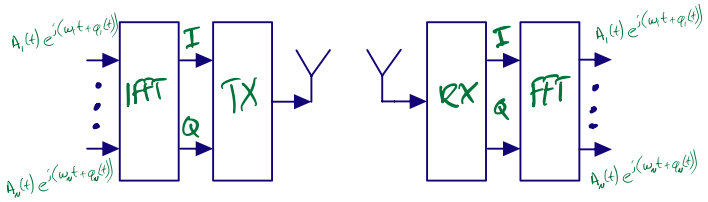
Assume we wanted to send 1024 Mbps over the air. We could choose a bandwidth of a about 1 GHz with 1-bit per symbol, or have a bandwidth of 1 MHz if we sent 1024 QAM at 1MS/s. Both cases would look like the figure below.

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.



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 reciever 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.



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.

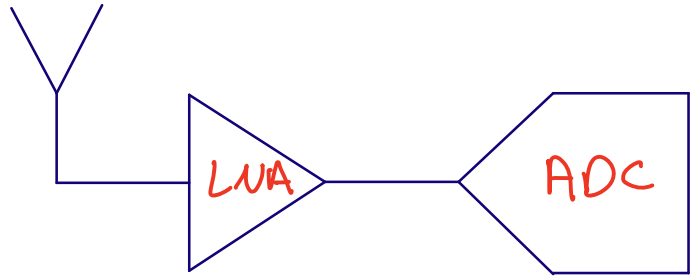
It's possible to do the same with single carrier modulation also. Imagine we made a 128-QAM modulation on a single carrier. As long as we constructed the time domain signal correctly (cyclic prefix to make the FFT work nicely, some preamble to measure the communication channel, then we could take an FFT at the reciever, 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.

16.4.4 Use a Software Defined Radio

For our mouse, what radio scheme should we choose? One common instances of "how to make a choice" in industry is "Delay the choice as long as possible so your sure the choice is right".

Maybe the best would be to use a software defined radio reciever? 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?



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

$$\approx 5 \text{ fJ/step}$$

$$BW = 2.5 \text{ GHz}$$

$$DR = 100 \text{ dB} = (96 - 1.76)/6.02 \approx 16 \text{ bit}$$

$$P = 5 \text{ fF} \times 5 \text{ GHz} \times 2^{16} = 1.6 \text{ W}$$

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.

16.5 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.

16.5.1 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)

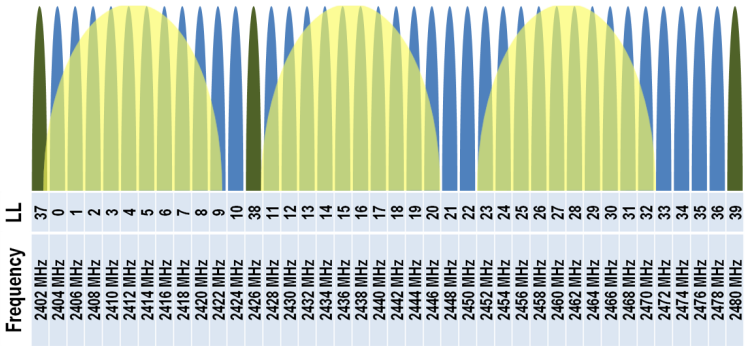
You'll find BR/EDR in most audio equipment, cars and legacy devices. For new devices (also audio), there is now a transition to Bluetooth Low Energy.

16.5.2 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 yellow is the WiFi channels.

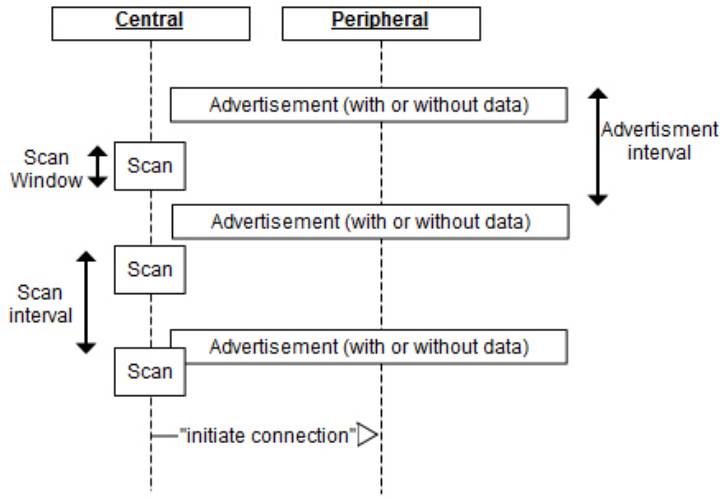
The advertiser channels have been intentionally placed where there is space between the WiFi channels to decrease the probability of collisions.



Any Bluetooth LE peripheral will advertise it's 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.

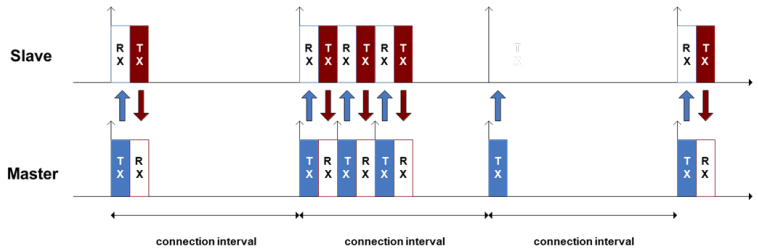
A Bluetooth LE central will camp in receive on a advertiser channel and look for these short messages from peripherals. If one is observed, the Central may choose to respond.

Take any spectrum analyzer anywhere, and you'll see traffic on 2402, 2426, and 2480 MHz.



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.



png further information Building a Bluetooth application on nRF Connect SDK

Bluetooth Specifications in Development

16.6 Algorithm to design state-of-the-art LE radio

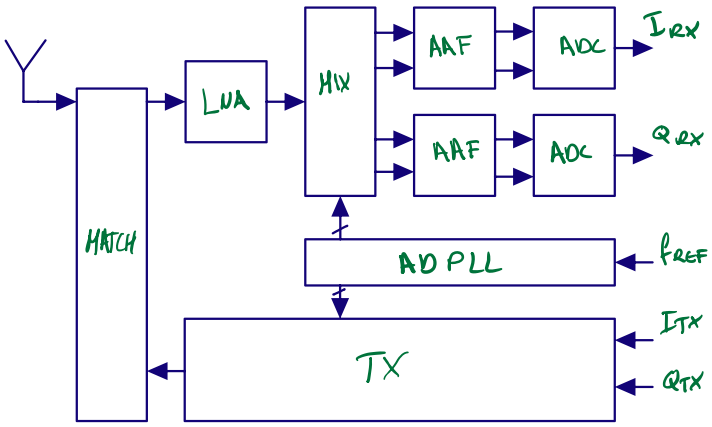
- Find most recent digest from International Solid State Circuit Conference (ISSCC)
- Find Bluetooth low energy papers
- Pick the best blocks from each paper

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.



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

16.6.1 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.

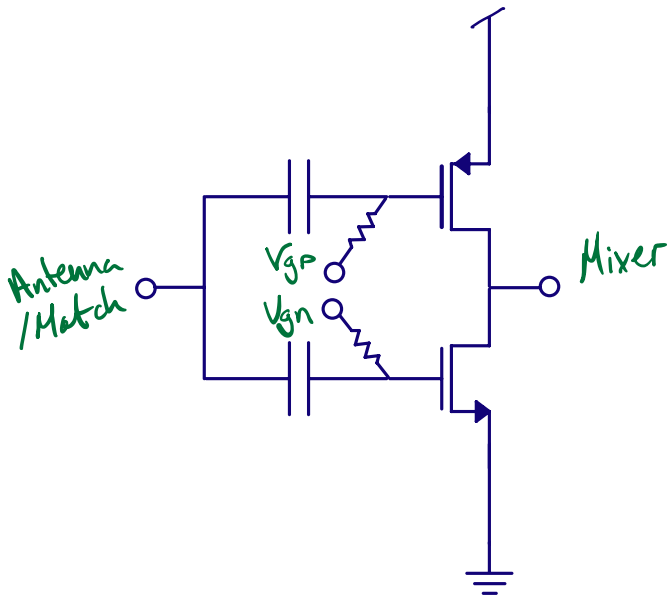
The challenges in the low noise amplifier is to provide the right gain. If there is a strong input signal, then reduce the gain. If there is a low input signal, then increase the gain.

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

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.



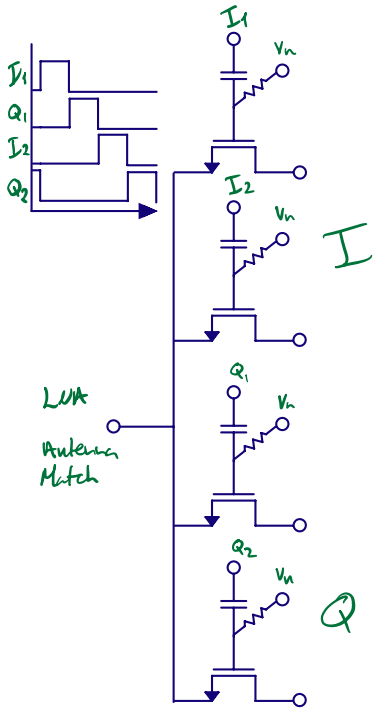
16.6.2 MIXER

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

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



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.

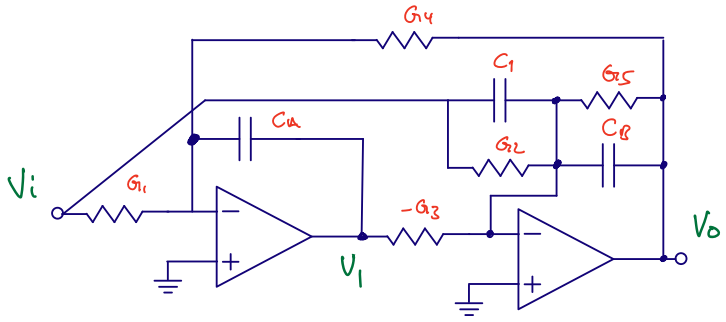
There is no "one correct choice", there are trade-offs that both ways. KISS (Keep It Simple Stupid) is one of my guiding principles when working on radio architecture.

These days most de-modulation happens in digital, and we need to convert the analog signal to digital, but first AAF.

16.6.3 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 6.2 nV RMS, while 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 $20\log(0.1/6.2e-9) = 76$dB in the reciever.



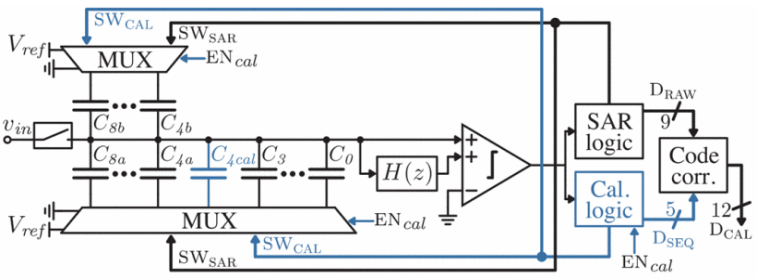
16.6.4 ADC

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.

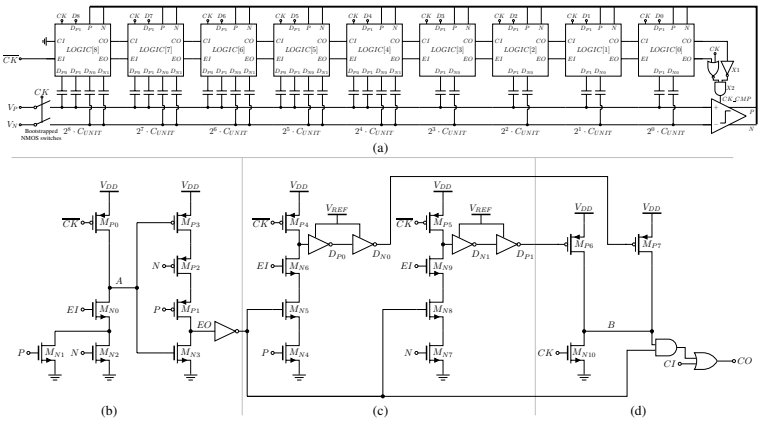
If I were to pick, I'd make something like A 68 dB SNDR Compiled Noise-Shaping SAR ADC With On-Chip CDAC Calibration as shown in the figure below.



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.

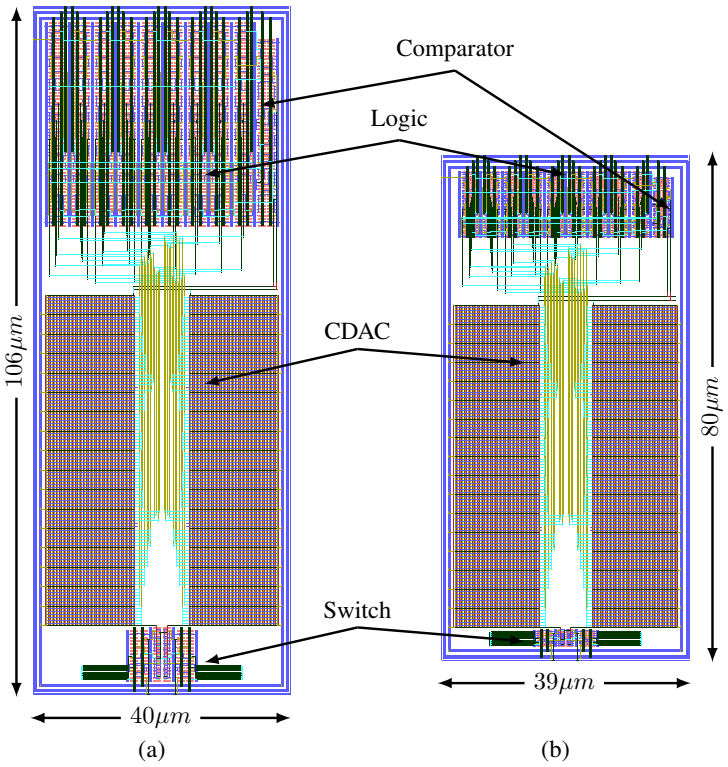
The main selling point of that ADC was that it's compiled from a JSON file, a SPICE file and a technology file into a DRC/LVS clean layout.

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



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.

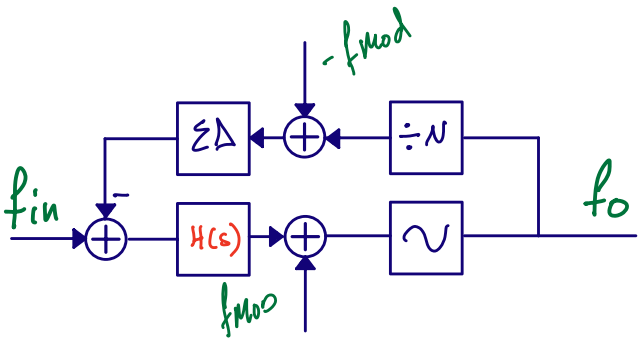
I detest doing anything twice, so I love the fact that I never have to re-draw that ADC again. I just fix the technology file (and maybe some tweaks to the other files), and I have a completed ADC.



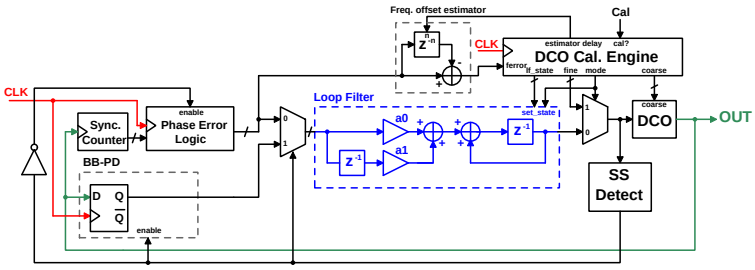
16.6.5 AD-PLL

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.



AD-PLL with Bang-Bang phase detector for steady-state



16.6.6 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

16.7 What do we really want, in the end?

The reciever part can be summed up in one equation for the sensitivity. The noise in a certain bandwidth. The Noise Figure of the analog reciever. The Energy per bit over Noise of the demodulator.

$$P_{RX_{sens}} = -174dBm + 10 \times log10(DR) + NF + Eb/N0$$

for example, for nRF5340

$$P_{RX_{sens}} + 174 - 60 = NF + Eb/N0 = 17dB$$

- *Bluetooth*® 5.1, IEEE 802.15.4-2006, 2.4 GHz transceiver
  - -97.5 dBm sensitivity in 1 Mbps Bluetooth low energy mode
  - -20 to +3 dBm configurable TX power
  - On-air compatible with nRF52, nRF51, nRF24L, and nRF24AP Series
  - Supported data rates:
    - *Bluetooth* 5.1: 2 Mbps, 1 Mbps, 500 kbps, and 125 kbps
    - IEEE 802.15.4-2006: 250 kbps
    - Proprietary 2.4 GHz: 2 Mbps, 1 Mbps
  - Single-ended antenna output (on-chip balun)
  - 128-bit AES/ECB/CCM/AAR co-processor (on-the-fly packet encryption)
  - 3.2 mA run current in TX (0 dBm)
  - 2.6 mA run current in RX
  - RSSI (1 dB resolution)

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.

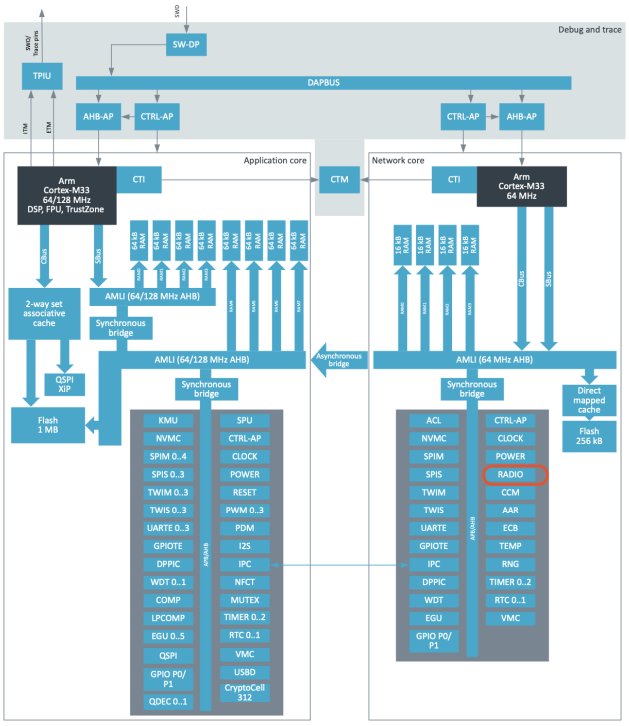


Figure 1: Simplified block diagram

16.8 Want to learn 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, 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, B. J. Thijssen, University of Twente, Enschede, The Netherlands

A 68 dB SNDR Compiled Noise-Shaping SAR ADC With On-Chip CDAC Calibration, 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 Recievers, C. Wulff, T. Ytterdal

Cole Nielsen, https://github.com/nielscol/thesis_presentations

"Python Framework for Design and Simulation of Integer-N AD-PLLs", Cole Nielsen, https://github.com/nielscol/tfe4580-report/blob/master/report.pdf

Design of CMOS Phase-Locked Loops, Behzad Razavi, University of California, Los Angeles

17 Energy Sources

Status: 0.3

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.

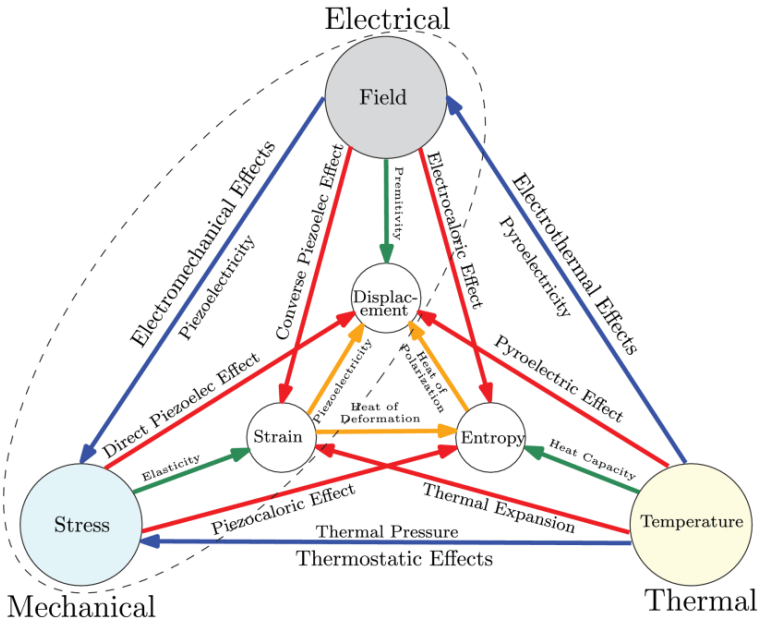
One of the first commercial ADCs, the DATRAC on page 24, was a 11-bit 50 kSps that consumed 500 W. That's Walden figure of merit of 4 μJ/conv.step. Today's state-of-the-art ADCs in the same sampling range have a Walden figure of merit of 0.6 fJ/conv.step.

4 μ / 0.6 f = 8.1e9, 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.

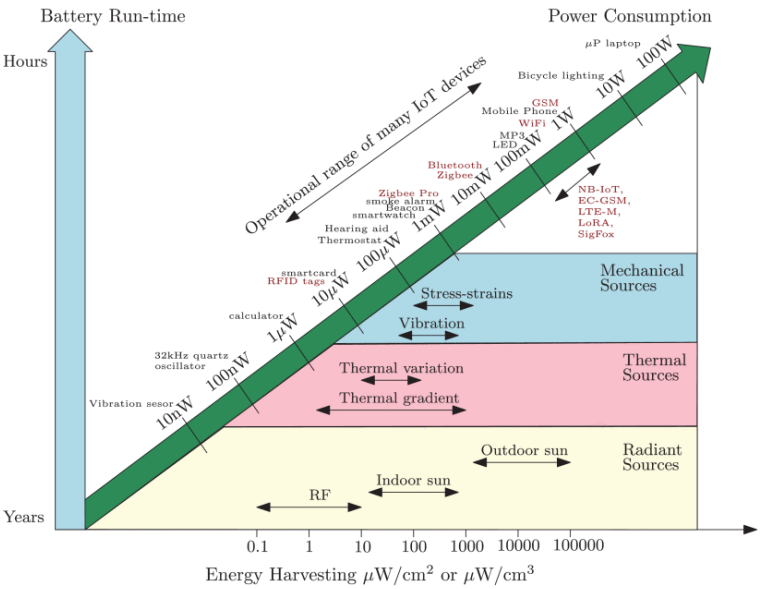
Towards a Green and Self-Powered Internet of Things Using Piezoelectric Energy Harvesting [1] has a nice overview of power consumption of technologies, seen in the next figures below.

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.



Below we can see a figure of the potential energy that can be harvested per volume, and the type power consumption of technologies [1].

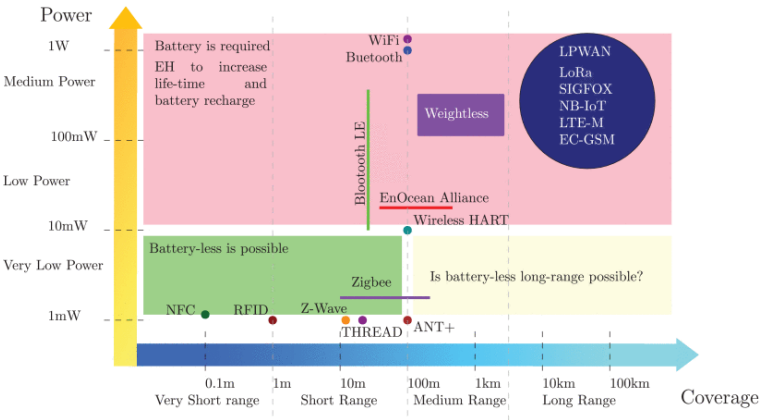
As devices approach average power consumption of μW it becomes possible to harvest the energy from the environment, and do away with the battery.



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).



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

17.1 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?

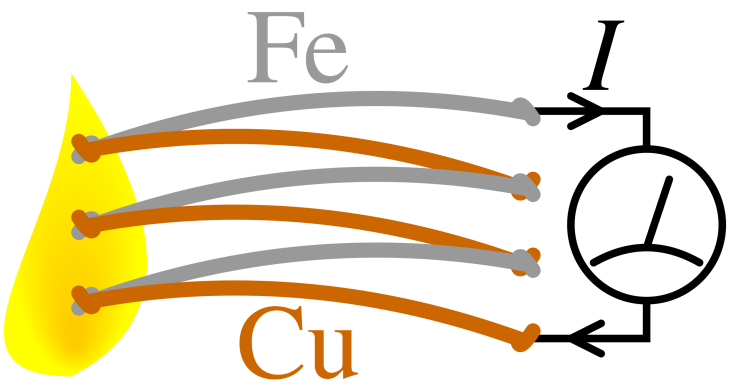
NO, there would not be a voltage difference between the two ends of the wire. The voltage on the loop side would be different, but on the cold side, where we have the ends, there would be no voltage difference.

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 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.

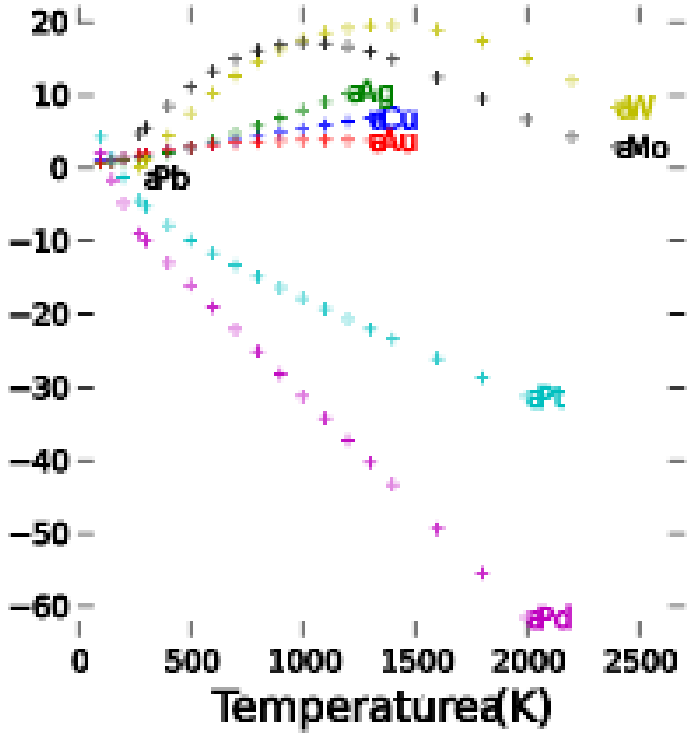


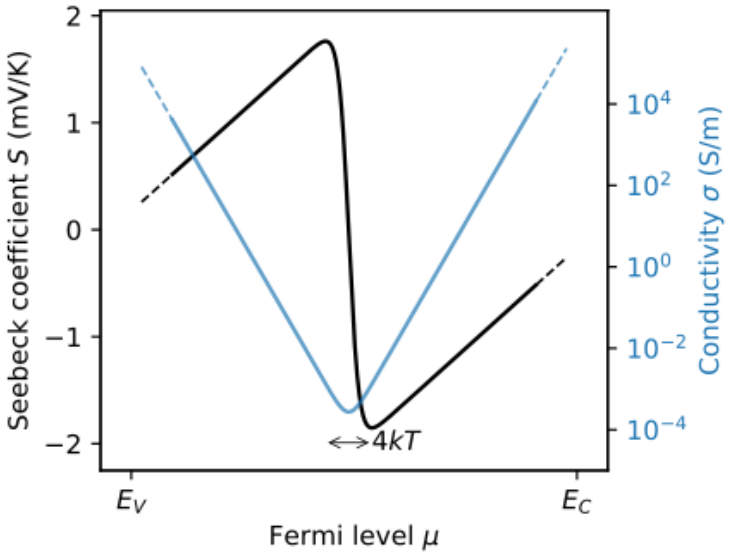
The voltage difference at the hot and cold end is described by the

Seebeck coefficient

Imagine two parallel wires with different Seebeck coefficients, one of copper (6.5 $\mu V/K$) and one of iron (19 μ/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 V valence band.



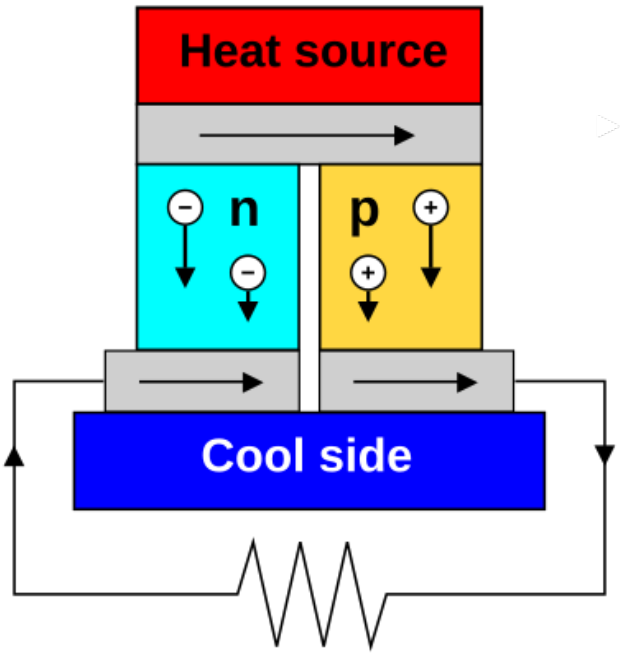


In the picture below we have a silicon (the cyan and yellow colors).

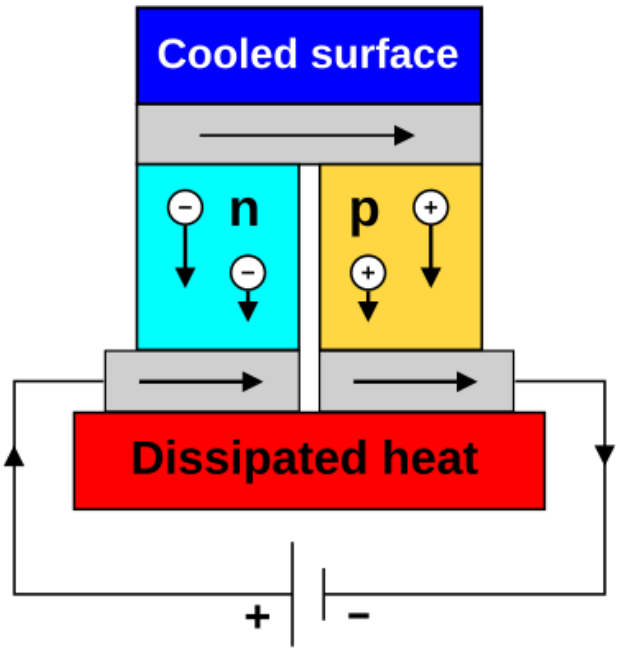
Assume we dope with acceptors (yellow, p-type), that shifts the Fermi level closer to the Valence band (E_V), and the dominant current transport will be by holes, maybe we get 1 mV/K from the picture above.

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.



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.



17.1.1 Radioisotope Thermoelectric generator

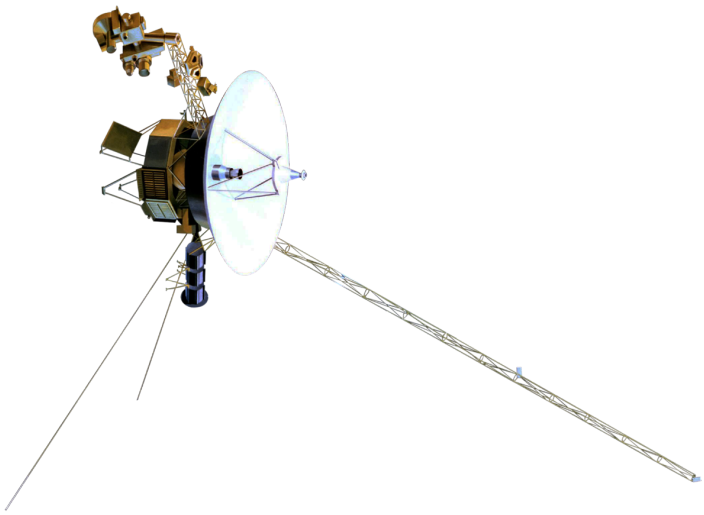
Maybe you've heard of a nuclear battery. Sounds fancy, right? Must be complicated, right?

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.



17.1.2 Thermoelectric generators

Assume a 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.

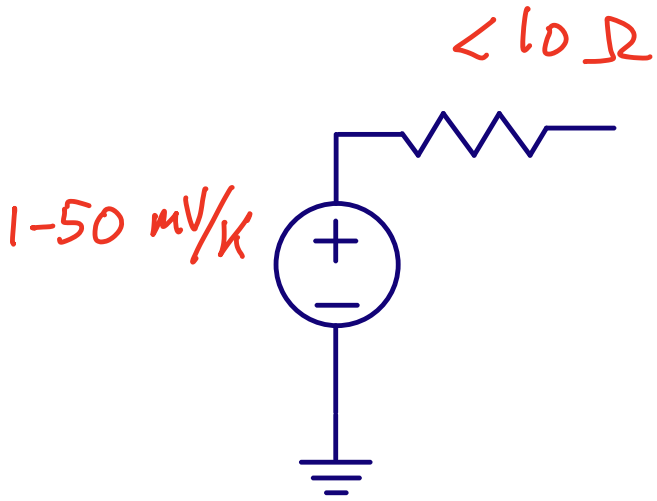
In order to drive an integrated circuit we'd need to boost the voltage to maybe 1.8 V.

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 they use a combination of both switched capacitor and switched inductor boost.



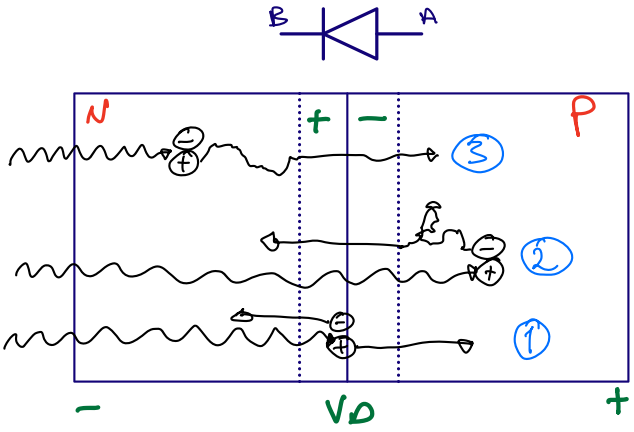
17.2 Photovoltaic

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.

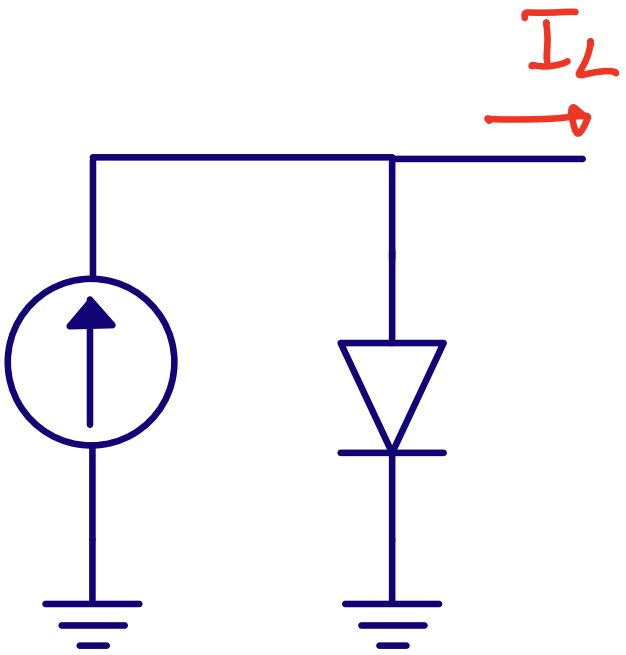
An electron/hole pair knocked out in the depletion region (1) will separate due to the built-in field. The hole will go to P and the electron to N. This increases the voltage VD across the diode.

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).



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.



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.

$$I_D = I_S \left(e^{\frac{V_D}{V_T}} - 1 \right)$$

$$I_D = I_{Photo} - I_{Load}$$

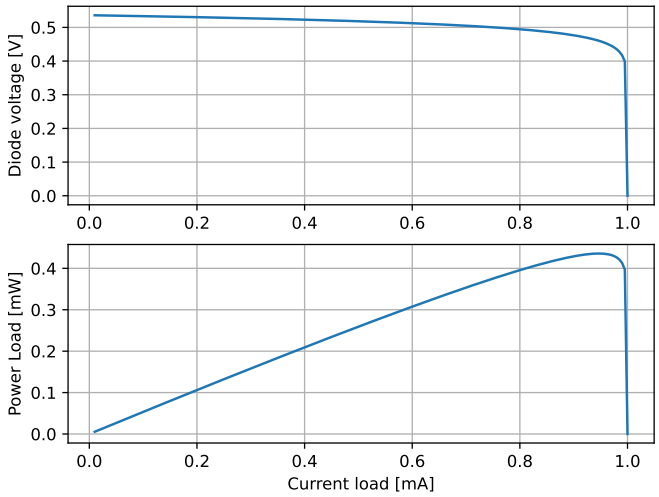
$$V_D = V_T ln\left(\frac{I_{Photo} - I_{Load}}{I_S} + 1\right)$$

$$P_{Load} = V_D I_{Load}$$

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.



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 they include a maximum power point tracker and a reconfigurable charge pump to optimize efficiency.

17.3 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.

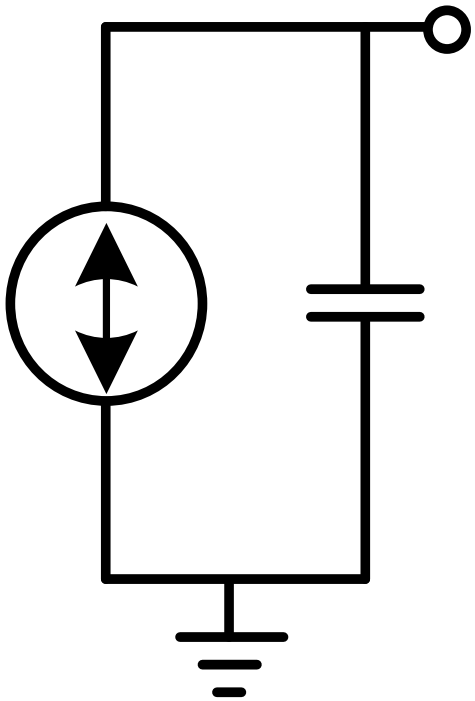
Assuming the above explanation is true, then there should not be piezoelectric materials with high conductivity, and indeed, most piezoelectric materials have resistance of 10^{12} to 10^{14} Ohm.

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.

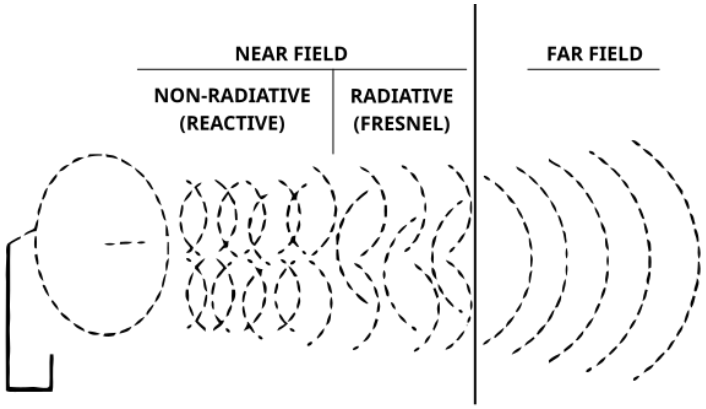
An example of piezoelectric energy harvester can be found in A Fully Integrated Split-Electrode SSHC Rectifier for Piezoelectric Energy Harvesting



17.4 Electromagnetic

17.4.1 "Near field" harvesting

Near Field Communication (NFC) operates at close physical distances



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

17.4.2 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	$20log_{10}\left(c/4\pi f\right)$ [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.

17.5 Triboelectric generator

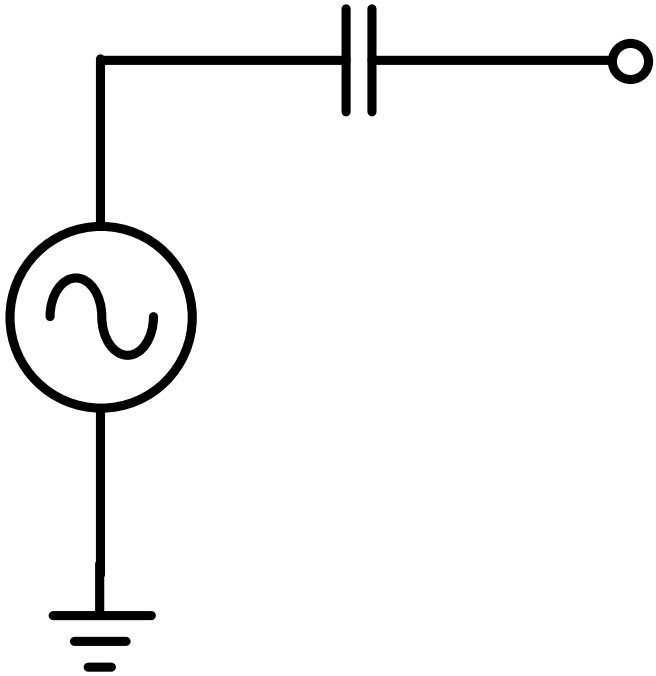
Although static electricity is an old phenomenon, it is only recently that triboelectric nanogenerators have been used to harvest energy.

An overview can be seen in Current progress on power management systems for triboelectric nanogenerators.

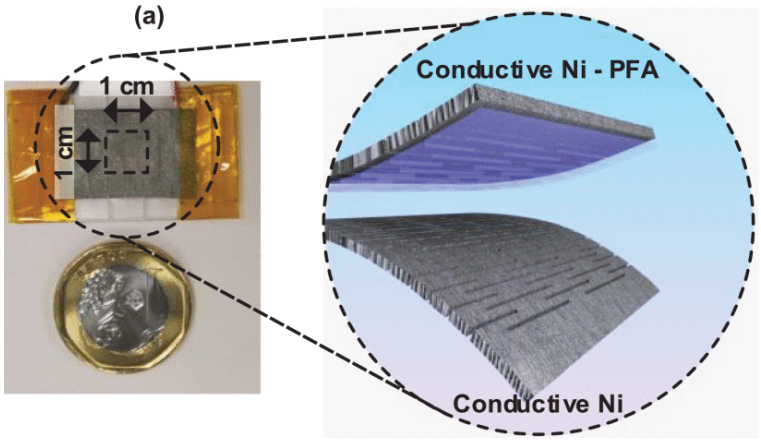
A model of a triboelectric generator can be seen in below. Although the current is low (nA) the voltage can be high, tens to hundreds of volts.

The key circuit challenge is the rectifier, and the high voltage output of the triboelectric generator.

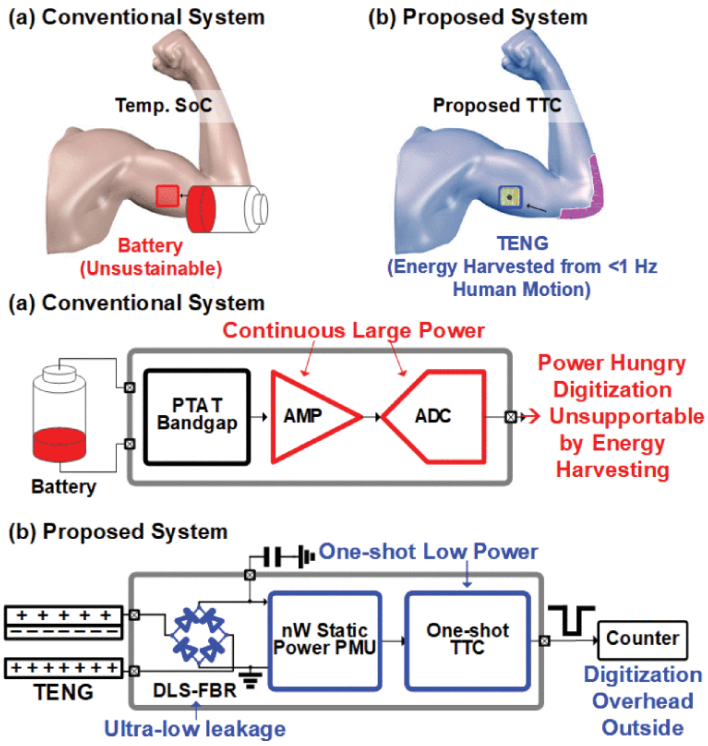
Take a look in A Fully Energy-Autonomous Temperature-to-Time Converter Powered by a Triboelectric Energy Harvester for Biomedical Applications for more details.



Below is a custom triboelectric material that converts friction into a sparse electric field.



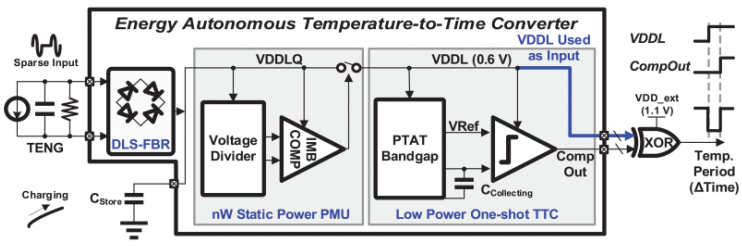
The key idea of the triboelectric circuit below is to rectify the sparse voltage pulses and store the charge on a capacitor. Once the voltage is high enough, then a temperature sensor is started.

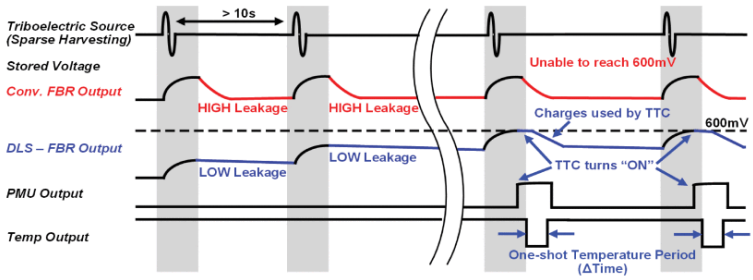


Below is some more details on the operation of the harvesting circuit, and the temperature sensor. Notice how the temperature sensor part of the circuit (PTAT bandgap, capacitor and comparator) produce a pulse width modulated signal that depends on temperature.

Also notice the "VDD_ext" in the figure. That means the system is not fully harvested. The paper is a prime example on how we in academia can ignore key portions of a system. They've focused on the harvesting part, and making the temperature dependent pulse width modulated signal. Maybe they've completely ignored how the data is transmitted from the system to where it would be used, and that's OK.

It's academia's job to prove that something could be possible. It's industry's job to make some that could be possible actually work.





17.6 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	10μW/cm²(indoor) 15mW/cm²(outdoor)	DC	Requires exposure to light
RF Energy	0.1μW/cm²(GSM) 0.01μW/cm²(WiFi)	380M ~ 5 Hz	Low efficiency for indoor and out of line-of-sight
Thermal – body heat	40μW/cm²	DC	Requires high temperature differences
Piezoelectric	4μW/cm²	> 30 Hz	Not limited by indoors or outdoors
Triboelectric (TENG)	1μW/cm²	1 Hz	Not limited by indoors or outdoors

17.7 Want to learn more?

[1] Towards a Green and Self-Powered Internet of Things Using Piezoelectric Energy Harvesting

A 3.5-mV Input Single-Inductor Self-Starting Boost Converter With Loss-Aware MPPT for Efficient Autonomous Body-Heat Energy Harvesting

A Reconfigurable Capacitive Power Converter With Capacitance Redistribution for Indoor Light-Powered Batteryless Internet- of-Things Devices

A Fully Integrated Split-Electrode SSHC Rectifier for Piezoelectric Energy Harvesting

Current progress on power management systems for triboelectric nanogenerators

A Fully Energy-Autonomous Temperature-to-Time Converter Powered by a Triboelectric Energy Harvester for Biomedical Applications

Analog SystemVerilog 18

Status: 0.3

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.

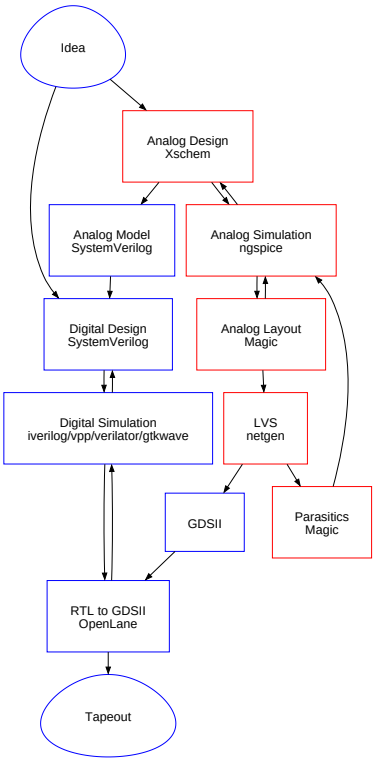
Which strategy is chosen depends on the complexity of the integrated circuit. For medium to low level of complexity, analog on top is fine. For high complexity ICs, then digital on top is the way to go.

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.



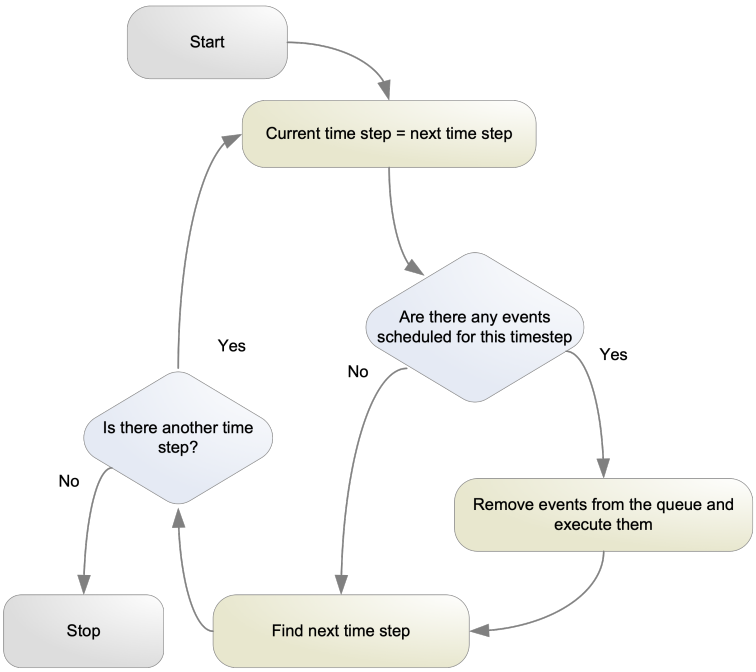
18.1 Digital simulation

Conceptually, the digital simulator is easy.

- The order of execution of events at the same time-step do not matter
- The system is causal. Changes in the future do not affect signals in the past or the now

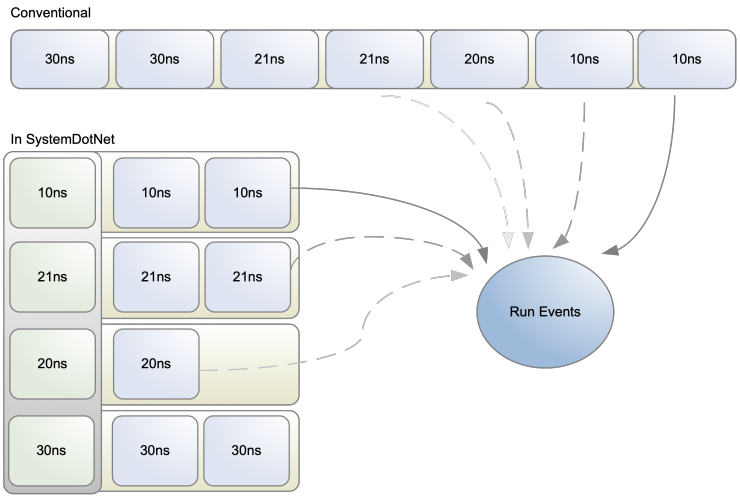
In a digital simulator there will be an event queue, see below. From start, set the current time step equals to the next time step. Check if there are any events scheduled for the time step. Assume that execution of events will add new time steps. Check if there is another time step, and repeat.

Since the digital simulator only acts when something is supposed to be done, they are inherently fast, and can handle complex systems.



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.



18.1.0.1 Digital Simulators

There are both commercial an 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

18.1.0.2 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.

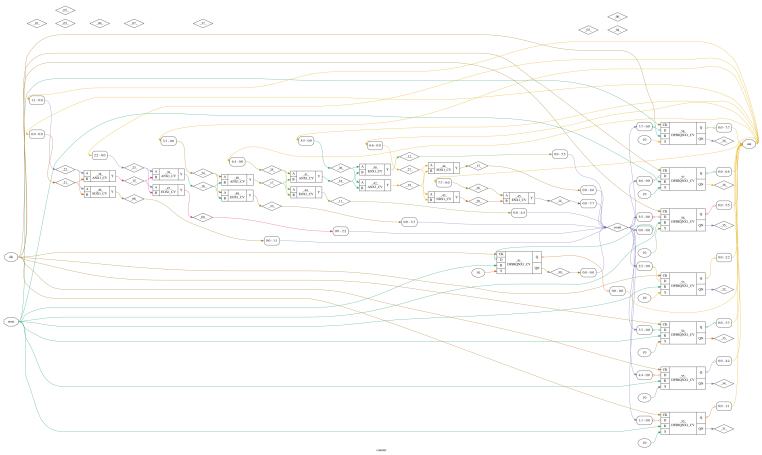
In a time-step where `out changes, then schedule an event to setcounttoout‘` plus one. As such, each positive edge of the clock at least 2 events must be scheduled in the register transfer level (RTL) simulation.

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.



18.2 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

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

18.3 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.

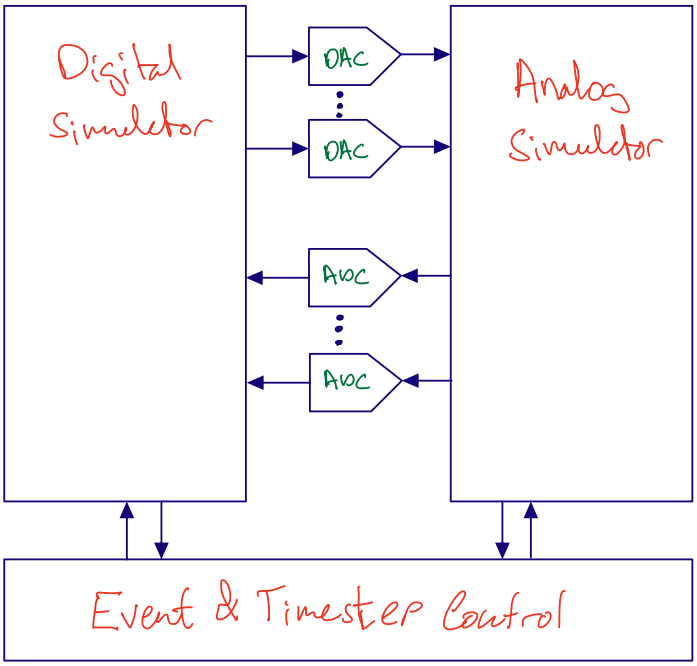
To orchestrate the time between simulators there must be a global event and time-step control. Most often, the digital simulator will end up waiting for the analog simulator.

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.

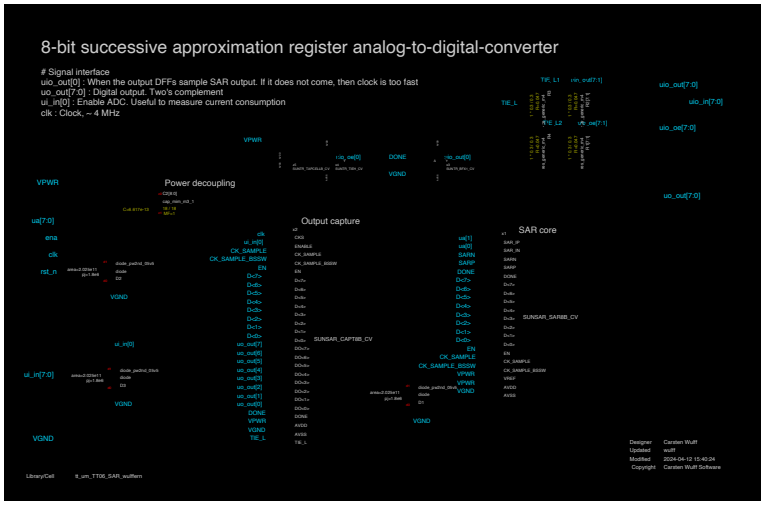
For IPs, like an ADC, co-simulation works well, and is the best way to verify the digital and analog.

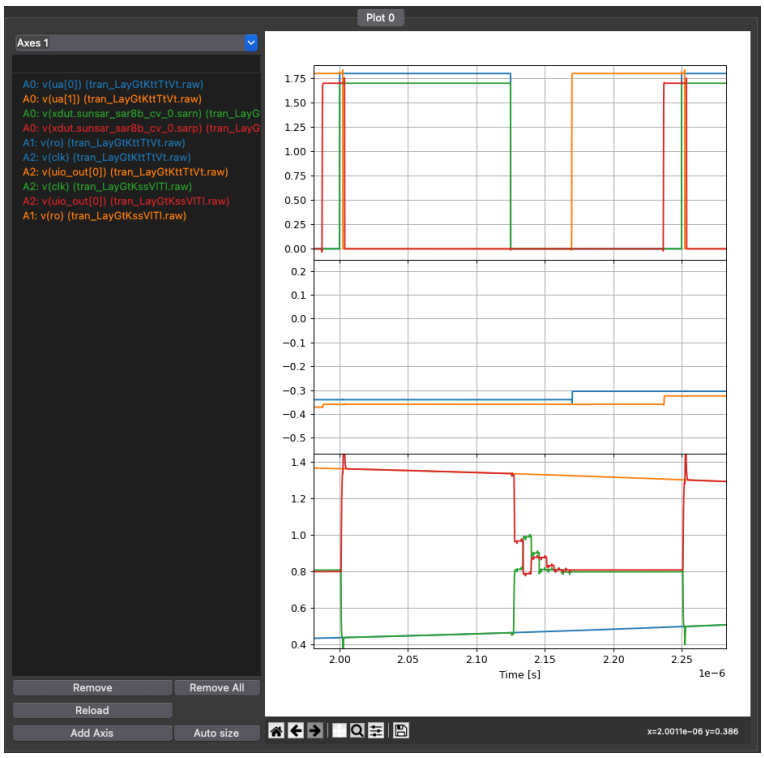
But if we can't run mixed simulation, how do we verify analog with digital?



18.4 Analog SystemVerilog Example

18.4.1 TinyTapeout TT06_SAR





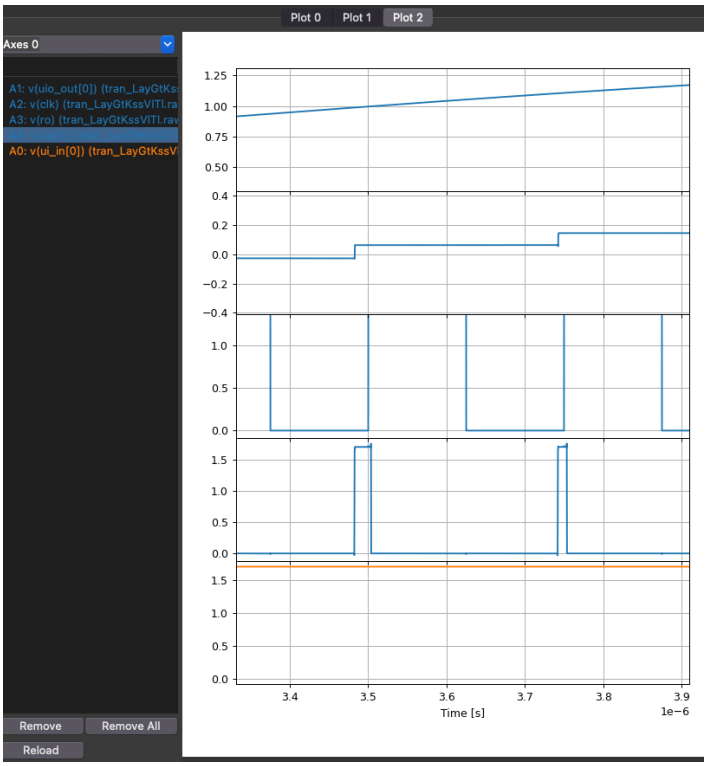
18.4.2 SAR operation

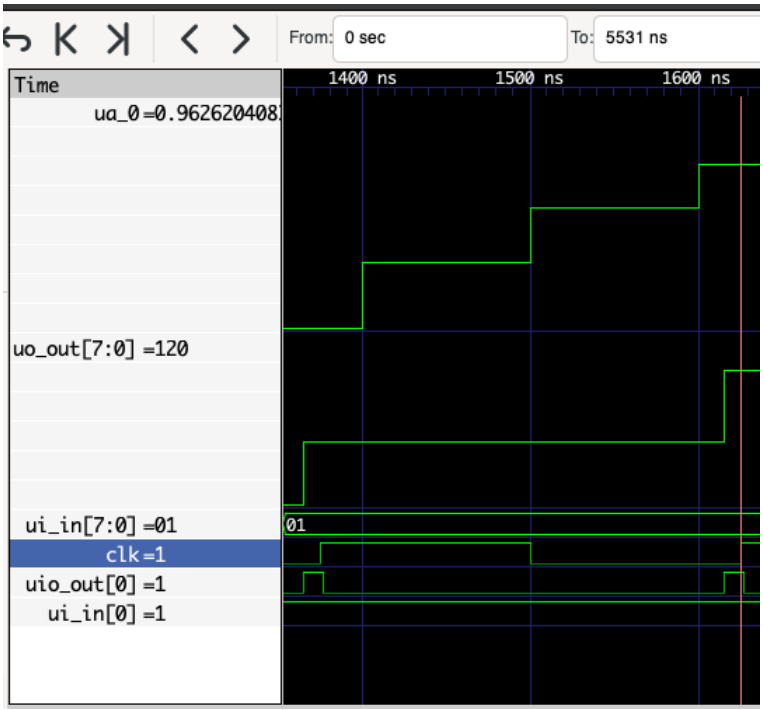
The key idea is to model the analog behavior to sufficient detail such that we can verify the digital code. I think it's best to have a look at a concrete example.

- 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
```





18.5 Want to learn more?

For more information on real-number modeling I would recommend The Evolution of Real Number Modeling

How to write a project report

19

Status: 0.8

19.1 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.

Should you make a fantastic temperature sensor, and maybe even reach close to a tapeout I would strongly suggest you submit a paper to NorCas. The deadline is August 15 2024.

19.2 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.

19.2.1 Shorter is better

I can write the section title idea in many words:

> A shorter text will more elequently describe the intricacies of your thoughts than a long, distinguished, tirade of carefully, wonderfully, choosen words.

or

> Shorter is better

Describe an idea with as few words as possible. The text will be better, and more readable.

19.2.2 Be careful with adjectives

Words like "very, extremely, easily, simply, . . . " don't belong in a readable text. They serve no purpose. Delete them.

19.2.3 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.

19.2.4 Don't be afraid of I

If you did something, then say "I" in the text. If there were more people, then use "we".

19.2.5 Transitions are important

Sentences within a paragraph are sometimes linked. Use

- As a result,
- As such,
- Accordingly,
- Consequently,

And mix them up.

19.2.6 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.

19.3 Report Structure

The sections below go through the expected structure of a report, and what the sections should contain.

19.3.1 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 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.

19.3.2 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.

19.3.3 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.

19.3.4 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 result should speak for themself.

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.

19.3.5 Discussion

Explain what the circuit and results show. Be critical.

19.3.6 Future work

Give some insight into what is missing in the work. What should be the next steps?

19.3.7 Conclusion

Summarize why, how, what and what the results show.

19.3.8 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.

19.4 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.	
Have you tried to make the circuit fail?	Knowing how circuits fail will increase confidence that it will work under normal conditions.	

Item	Description	OK
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)	

Layout Generation 20

Status: 0.5

20.1 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 unsovled 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.

20.2 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
```

20.3 CICPY

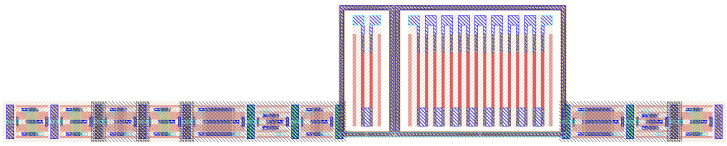
The first thing we need to do is to place all transistors. I do have a script to help. Install cicpy.

```
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 .
```

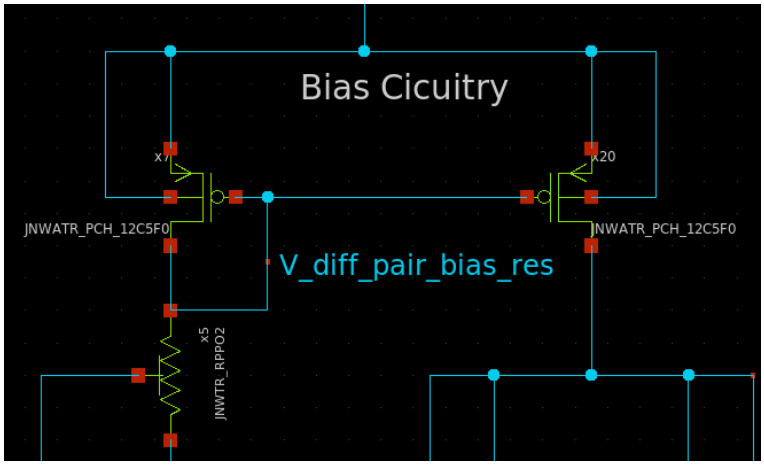
20.4 Placement

To generate an initial placement we can do the command below. If a layout exists it will be overriden

```
cd jnw_gr05_sky130a/work
cicpy sch2mag JNW_GR05_SKY130A OTA_Manuel
```



The layout engine has no idea what components belong together, for example, the current mirror below should have been place together

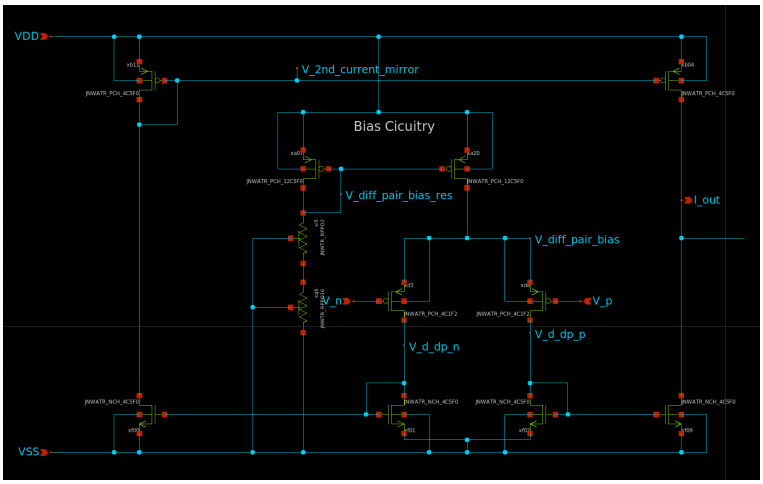


We can instruct the layout engine by adding a "group" name to the instance name. The instance name always starts with `x<something><number>` where the something can be nothing, or a group name (a,b, not a number).

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,

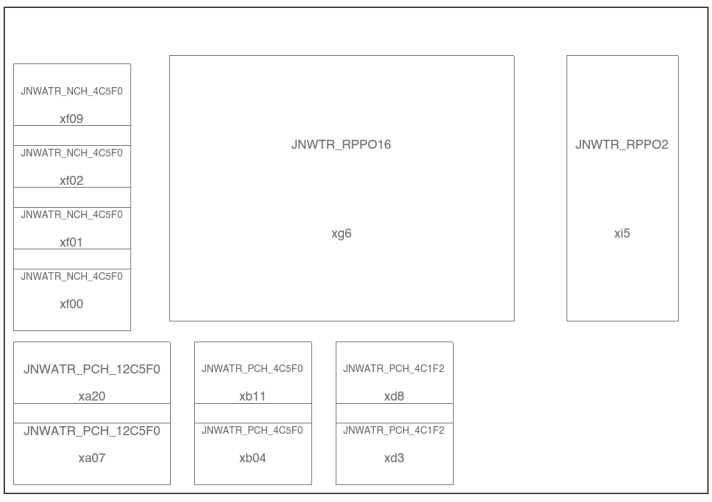


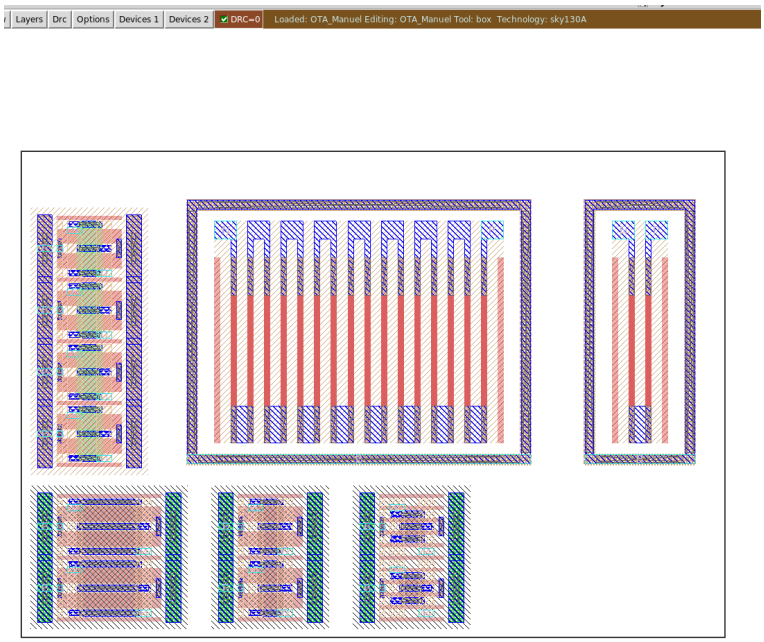
Then the layout becomes a bit better

```
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.




Layers
Drc
Options
Devices 1
Devices 2
DRC=0
Loaded: OTA_Manuel Editing: OTA_Manuel Tool: box Technology: sky130A

MOSFETs 21

Status: 0.3

I'm stunned if you've never heared the word "transistor". I think most people have heard the word. What I find funny is that almost nobody understand 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.

21.1 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.

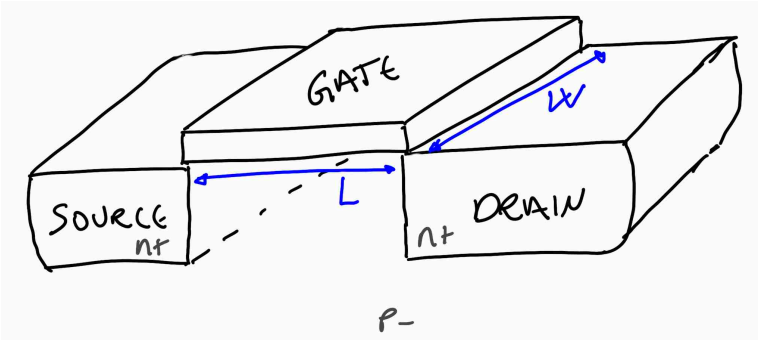


Figure 1: 3D crossection of a transistor

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.

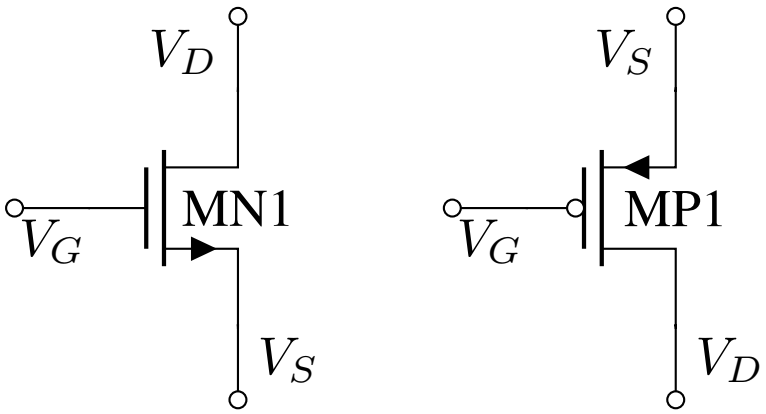


Figure 2: Transistor symbols

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.

21.2 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 co-valent 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.

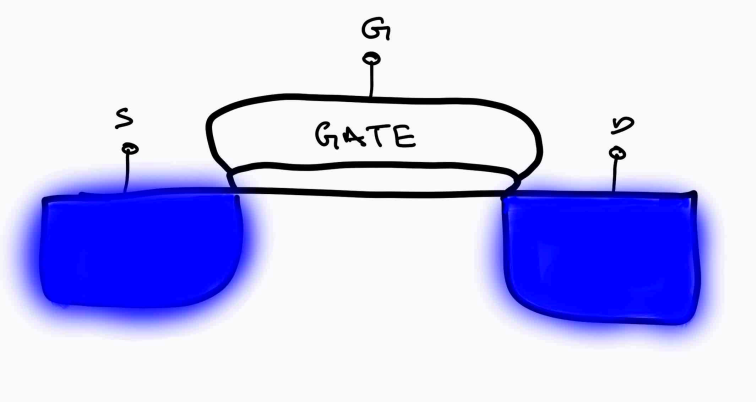


Figure 3: MOSFET in “off” state

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.

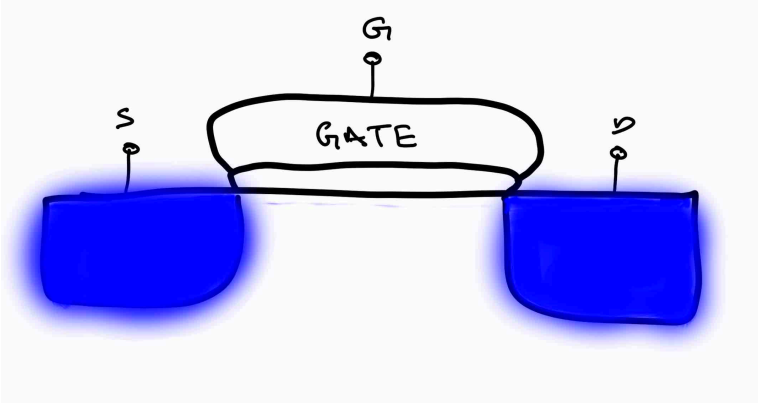


Figure 4: MOSFET in subthreshold

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 a 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.

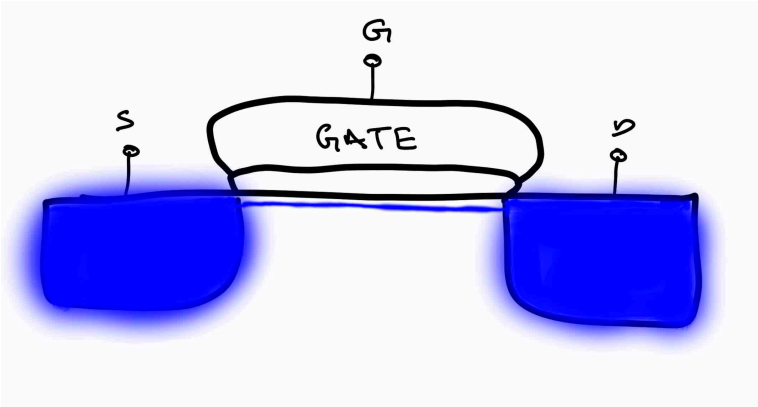


Figure 5: MOSFET in strong inversion

This thin blue sheet extend 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.

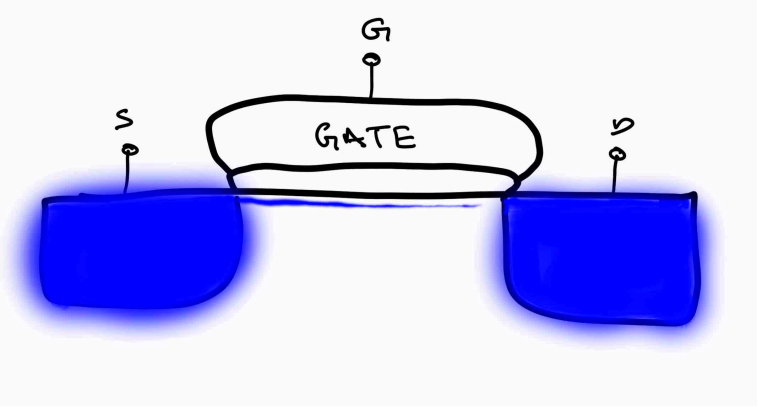


Figure 6: MOSFET in strong inversion and saturation

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 make transistor 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.

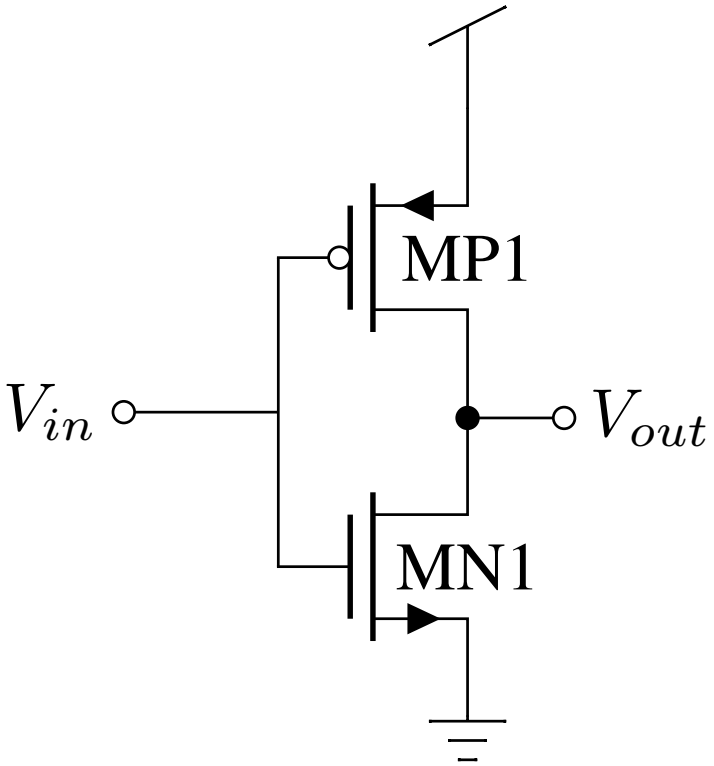


Figure 7: Inverter

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.

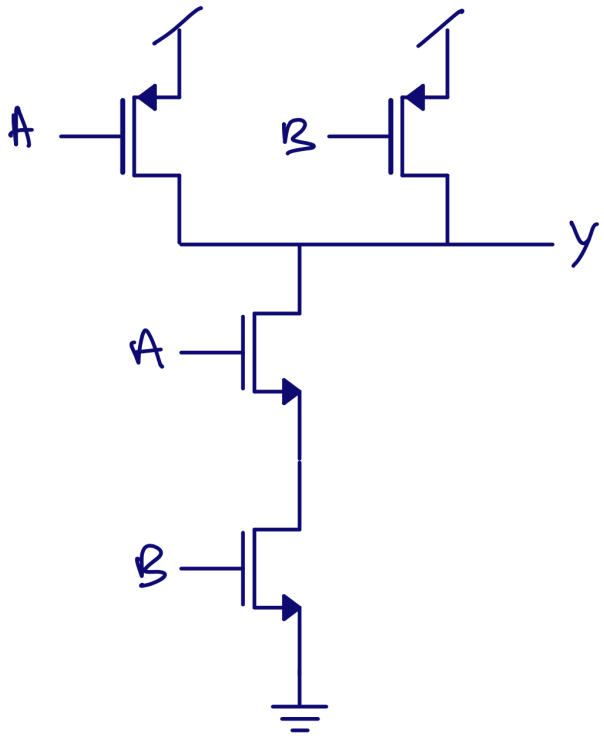


Figure 8: NAND

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.

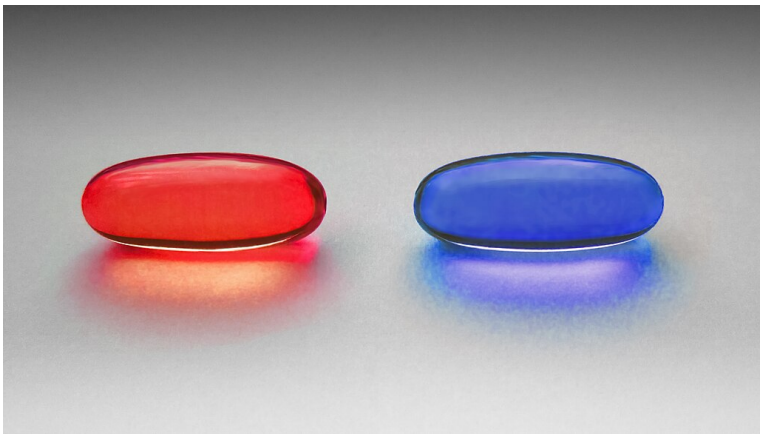


Figure 9: The choice

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 does not 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.

21.3 Analog transistors in the books

In the books we learn the equations for weak inversion

$$I_D \propto (e^{(V_{gs}-V_{th})/U_T} - 1)$$

, where I_D is the drain current, V_{gs} is the gate source voltage, V_{th} is the threshold voltage and $U_T = kT/q$, 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 a 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

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

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 relativistc version

of Schrodinger (which includes magnetic fields, and if you did not know, magnetic fields is just a relativistic effect of the electric field) and the wave function to compute how an 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 excelent books and resources. I would recommend Mark Lundstrom for the best in detail description of MOSFETs.

21.4 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.

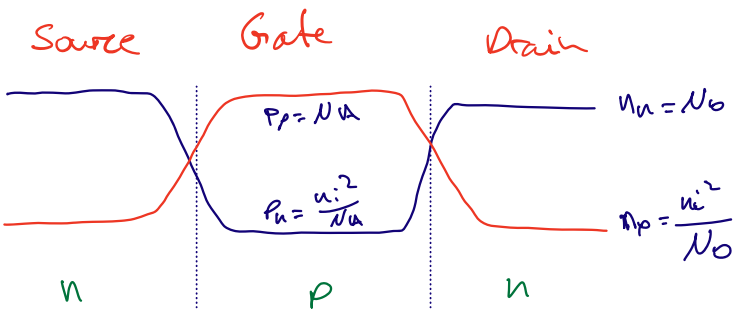


Figure 10: Charge carrier density in a MOSFET

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

An electron in the source would see a 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.

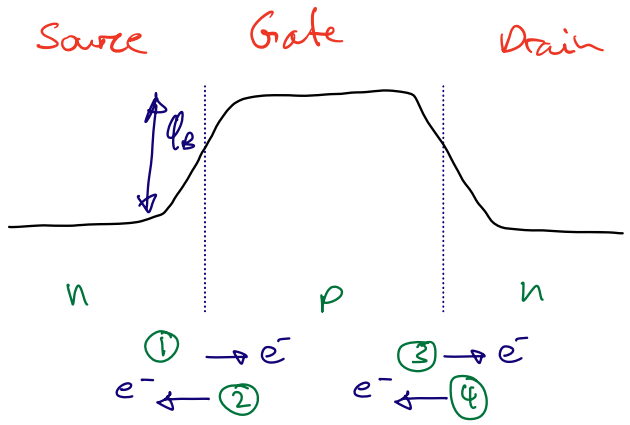


Figure 11: MOSFET subthreshold , $V_{DS} = 0$

There would also be hole currents flowing between source/bulk/drain and visa 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.

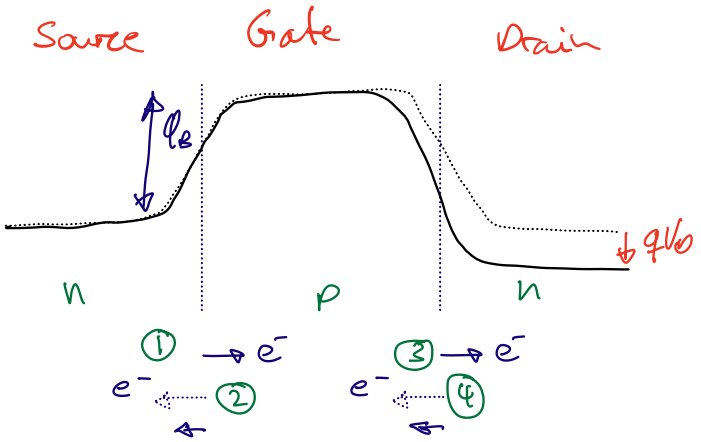


Figure 12: MOSFET subthreshold, $V_S = 0$ V, $V_D > 0$ V

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

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

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_{TH}

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.

21.5 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.

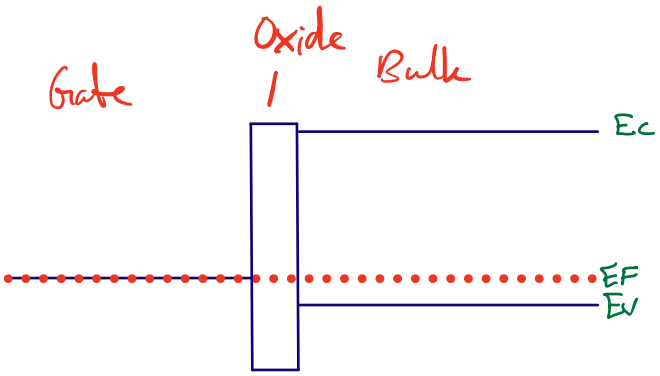


Figure 13: Band diagram of a fictive MOSFET.

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

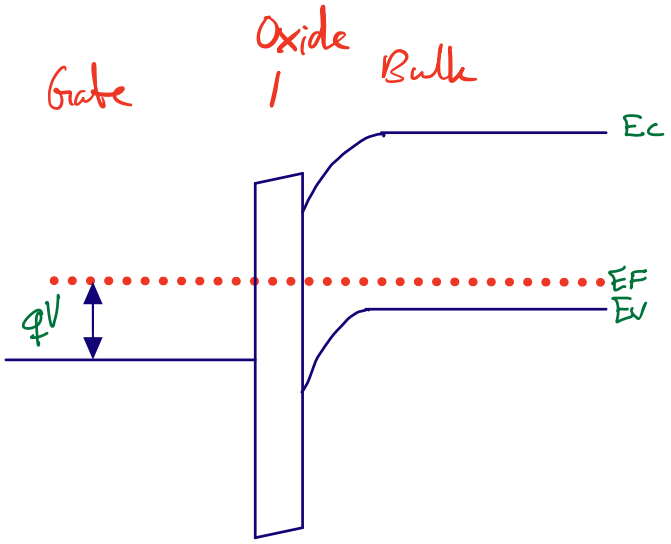


Figure 14: Band diagram with gate-source voltage applied

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 signficantly. 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.

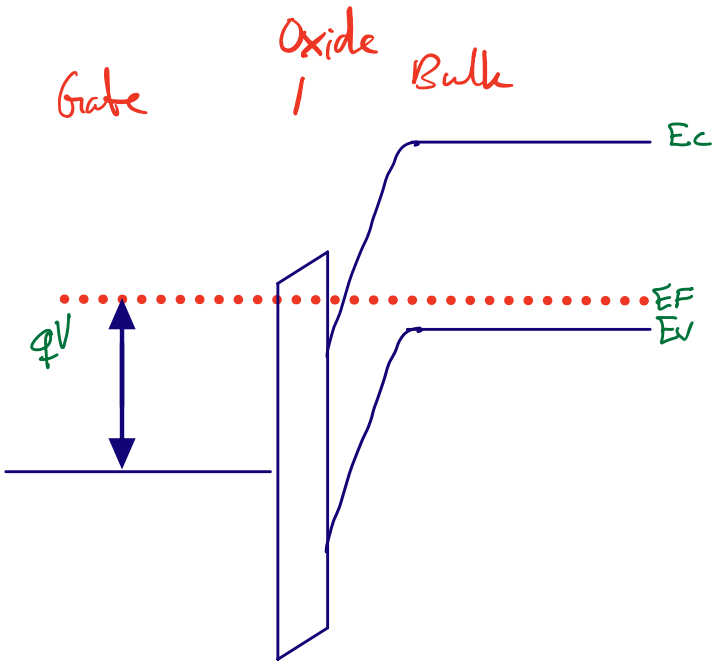


Figure 13: Band diagram with high gate-source voltage applied

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 it's 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 Columb 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 ampltiudes 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 a electric field to silicon can free co-valent 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 in

Carrier transport near the Si/SiO2 interface of a MOSFET

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

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

21.6 How should I size my transistor?

The method that makes most sense to me, is to use the inversion-coefficient method, described in Nanoscale MOSFET Modeling: Part 1 and Nanoscale MOSFET Modeling: Part 2.

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.

I should caveat my proposal for method. For the past 7 years I've not had the luxury to do full time, hardcore, analog design. As my career progressed, most of my time is now spent telling others what I think is a good idea to do, and not doing hardcore analog design myself. I think, however, I have a pretty decent understanding of analog circuits, and how to design them, so I think I'm correct in the proposal. If I were to start hardcore analog design now, I would go all in on inversion-coefficient based transistor size selection.

21.7 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.

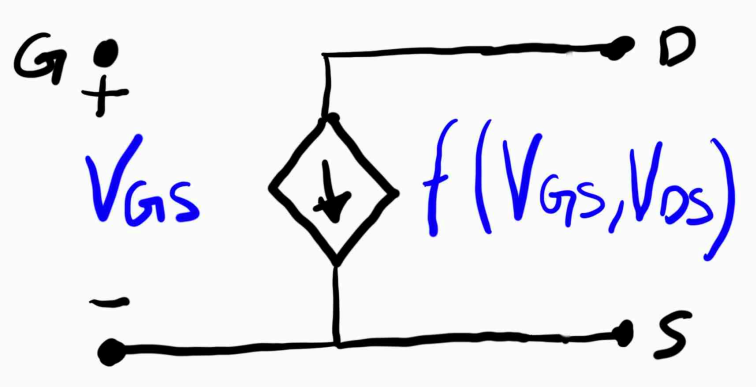
You can find the testbenches at Testbenches at dicex/sim/spice/N-CHIO

21.7.1 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



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?

21.7.2 Gate-source voltage

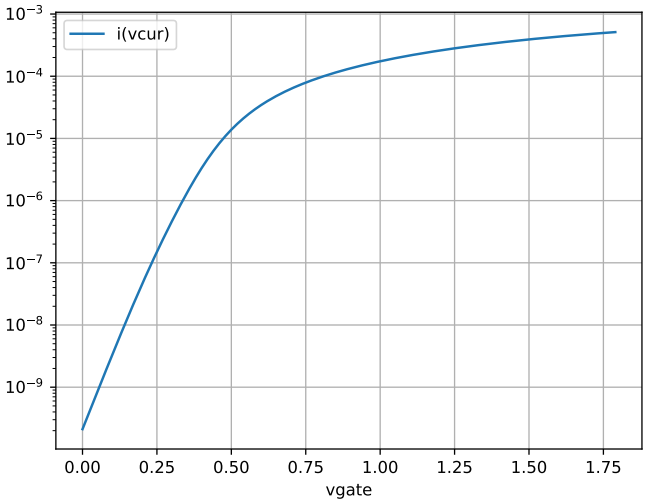
Below are the conditions I've used in the testbench. Notice there is a V_B that is the $p-$ substrate, or bulk, of the transistor. When we draw symbols of a transistor we don't always include the bulk node, because that's most of the time connected to ground for NMOS.

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.

$$i(vcur) = I_{DS}$$



21.7.3 Inversion level

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

Weak inversion

The drain current is low, but not zero, when

$$V_{eff} << 0$$

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

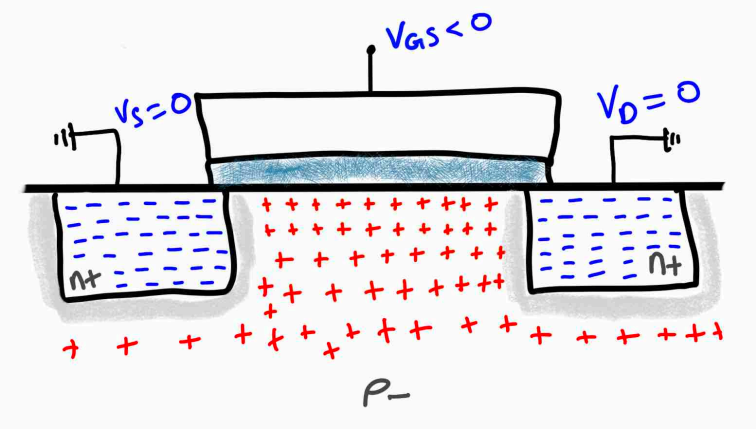
$$n \approx 1.5$$

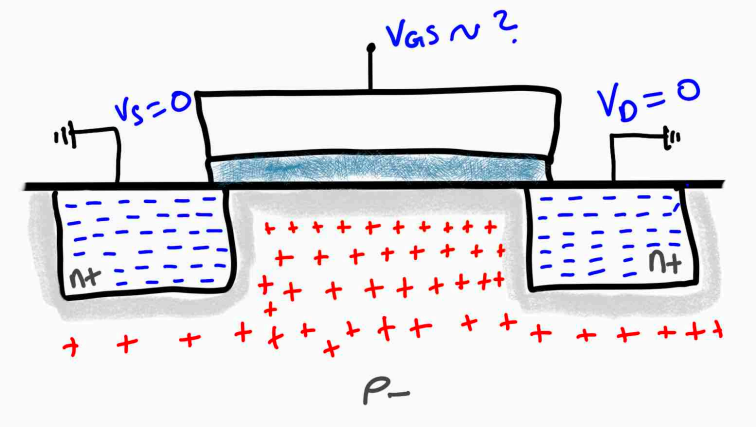
Moderate inversion

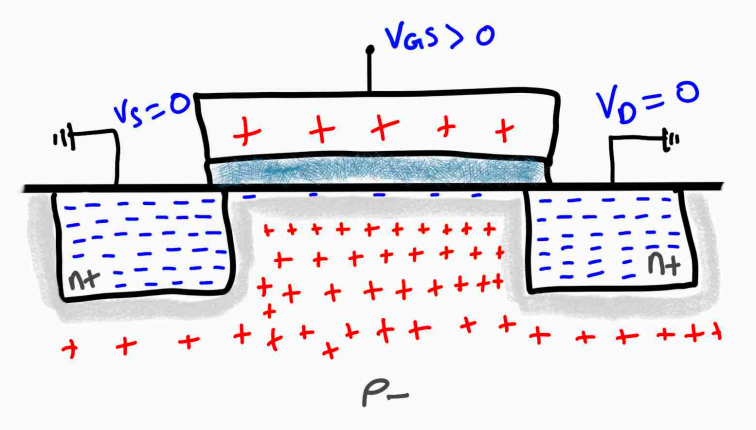
Very useful region in real designs. Hard for hand-calculation. Trust the model.

Strong inversion

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



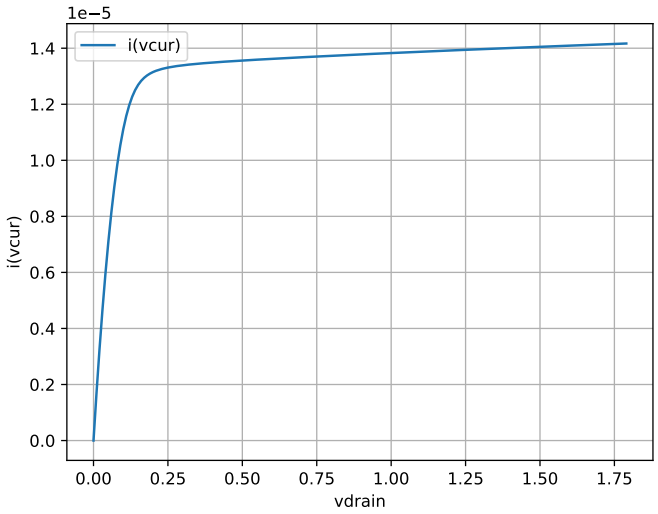




21.7.4 Drain source voltage

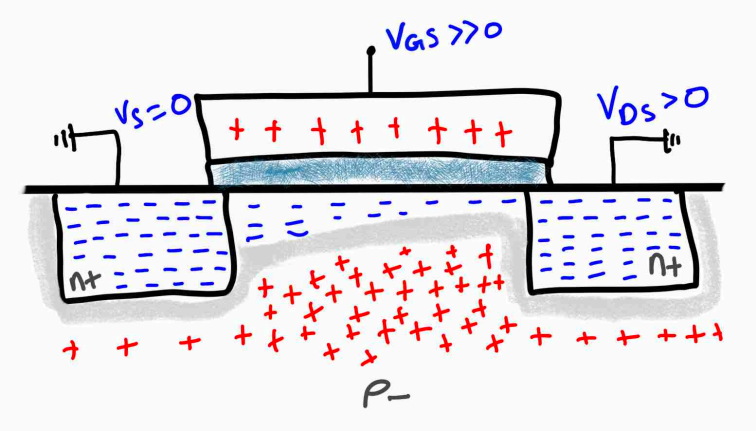
Param	Voltage [V]
VGS	0.5
VDS	0 to 1.8
VS	0
VB	0

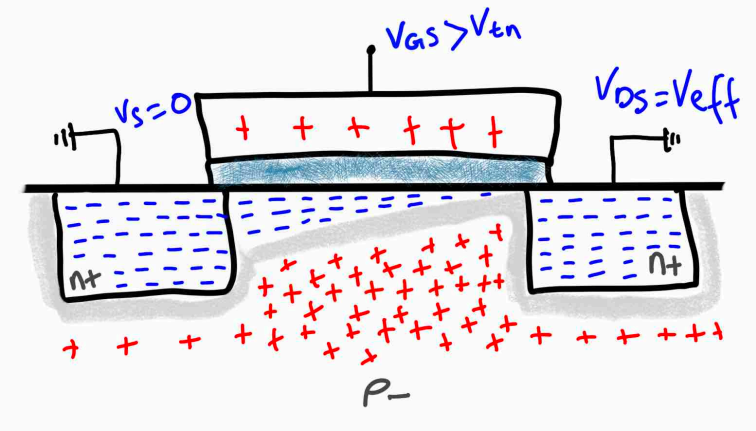
$$i(vcur) = I_{DS}$$

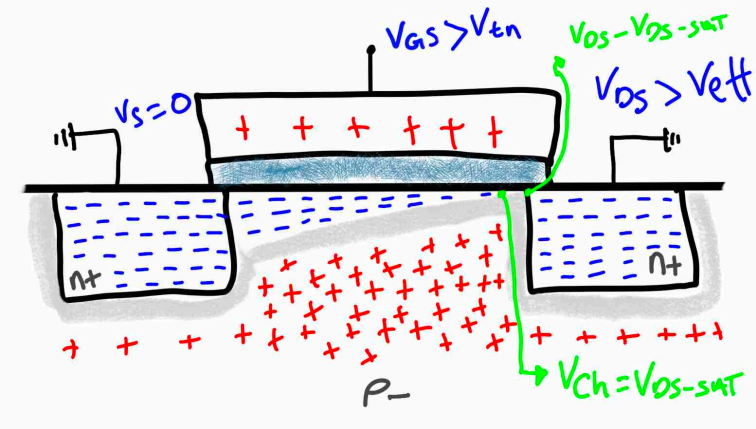


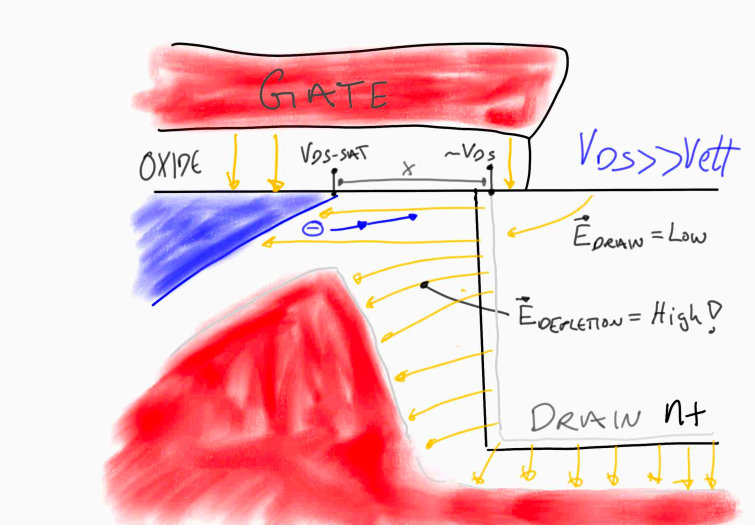
21.7.5 Strong inversion

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





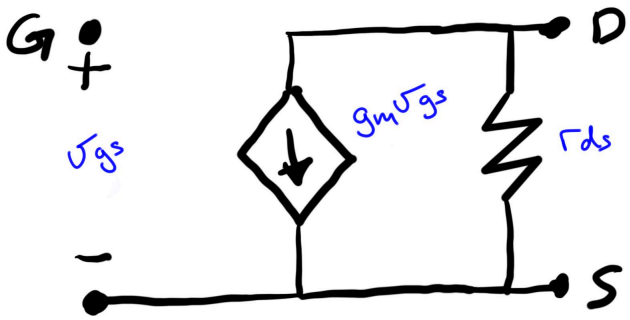




21.7.6 Low frequency model

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

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



21.7.7 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

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

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

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

Define

$$\ell = \mu_n C_{ox}\frac{W}{L}$$

and

$$V_{eff} = V_{GS} - V_{tn}$$

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

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

Assume channel length modulation is not there, then

$$I_D = \frac{1}{2}\ell V_{eff}^2$$

which means

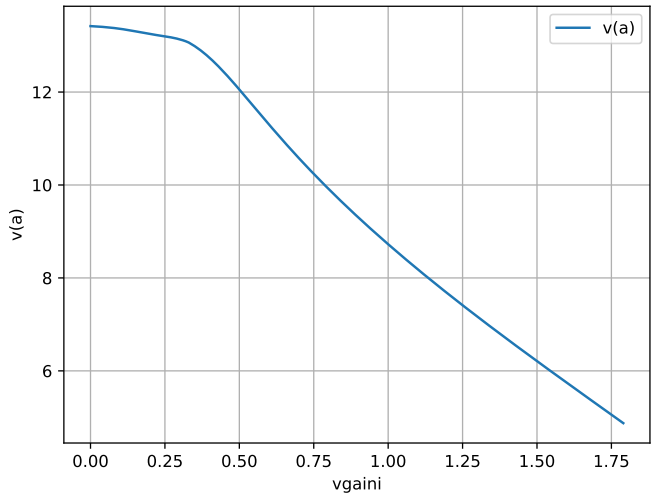
$$\frac{1}{r_{ds}} = g_{ds} \approx \lambda I_D$$

21.7.8 Intrinsic gain

Define intrinsic gain as

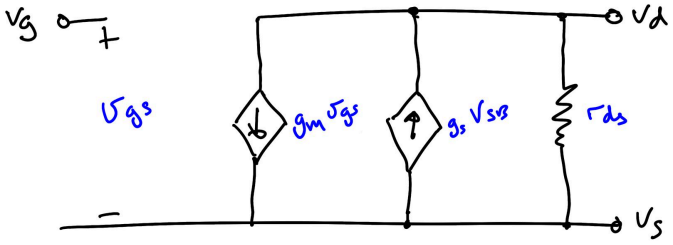
$$A = \left|\frac{v_{out}}{v_{in}}\right| = g_m r_{ds} = \frac{g_m}{g_{ds}}$$

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

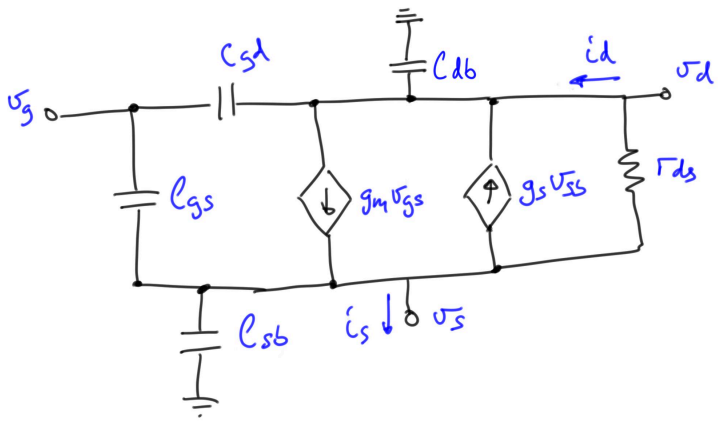


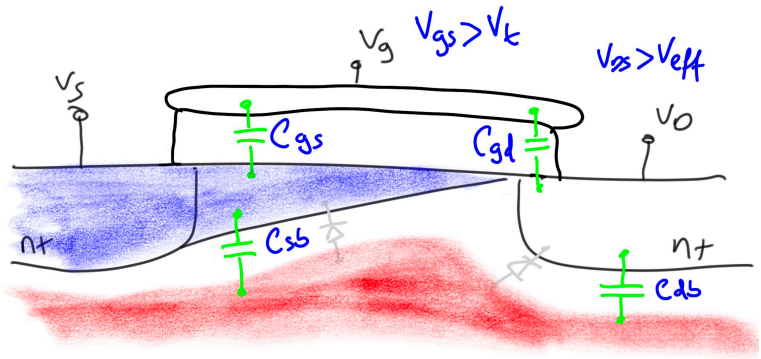
vgaini = Gate Source Voltage =

$$V_{eff} + V_{tn}$$



21.7.9 High frequency model





$$C_{gs}$$

and

$$C_{gd}$$

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

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

$$C_{sb}$$

and

$$C_{db}$$

Both are depletion capacitances

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

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

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

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

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

21.7.10 Be careful with Cgd (blame Miller)

If

$$Y(s) = 1/sC$$

then

$$Y_1(s) = 1/sC_{in}$$

and

$$Y_2(s) = 1/sC_{out}$$

where

$$C_{in} = (1 + A)C$$

,

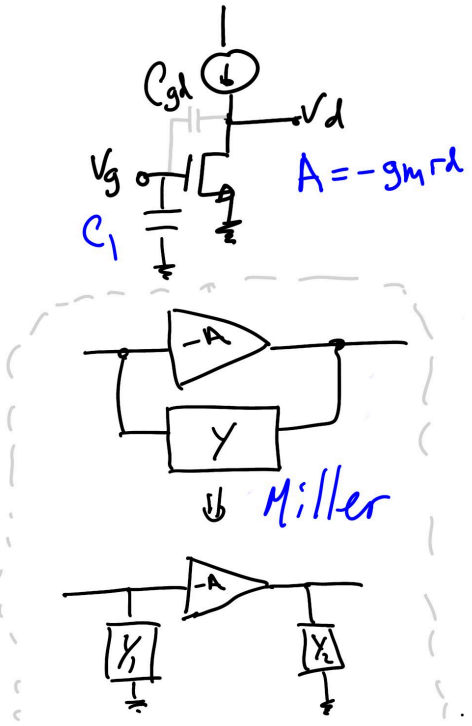
$$C_{out} = (1 + \frac{1}{2})C$$

$$C_1 = C_{gd} g_m r_{ds}$$

C_{gd}

can appear to be 10 to 100 times larger!

if gain from input to output is large



21.8 Weak inversion

If

$$V_{eff} < 0$$

diffusion currents dominate.

$$I_D = I_{D0} \frac{W}{L} e^{V_eff/nV_T}$$

, where

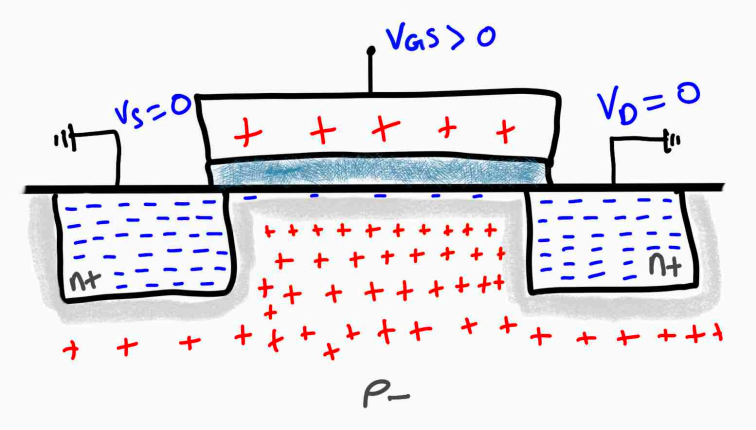
$$V_T = kT/q$$

,

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

$$I_{D0} = (n-1)\mu_n C_{ox} V_T^2$$

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



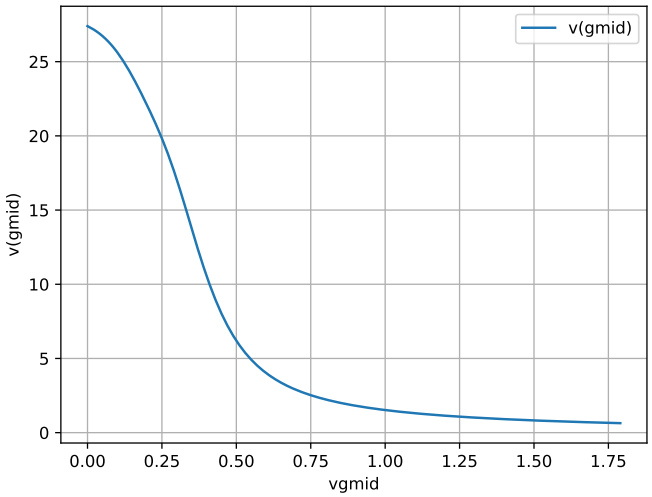
Bang for the buck

Subthreshold:

$$\frac{g_m}{I_D} = \frac{1}{nV_T} \approx 25.6 \text{ [S/A] @ 300 K}$$

Strong inversion:

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



21.9 Velocity saturation

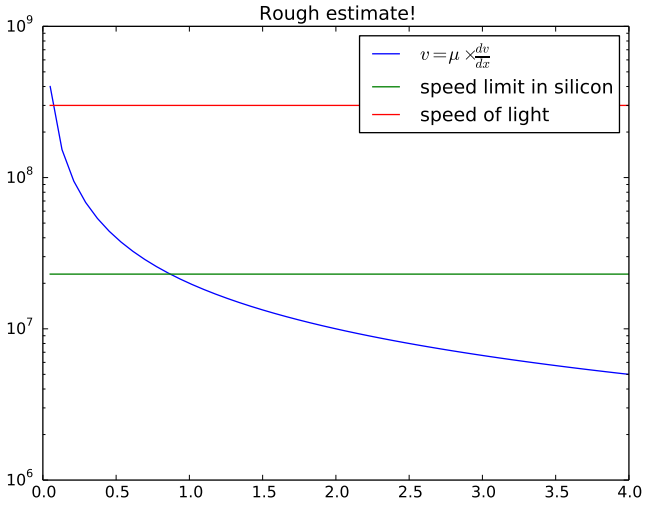
Electron speed limit in silicon

$$v \approx 10^7 cm/s$$

$$v = \mu_n E = \mu_n \frac{dV}{dx}$$

$$\mu_n \approx 100 \text{ to } 600 \; cm^2/Vs$$

in nanoscale CMOS



21.9.1 Square law model

$$Q(x) = C_{ox}\left[V_{eff} - V(x)\right]$$

$$v = \mu_n E = \mu_n \frac{dV}{dx}$$

$$\ell = \mu_n C_{ox} \frac{W}{L}$$

$$I_D = WQ(x)v = \ell L\left[V_{eff} - V(x)\right]\frac{dV}{dx}$$

$$I_D dx = \ell L\left[V_{eff} - V(x)\right]dV$$

$$I_D \int_0^L dx = \ell L \int_0^{V_{DS}} \left[V_{eff} - V(x)\right]dV$$

$$I_D\left[x\right]_0^L = \ell L\left[V_{eff}V - \frac{1}{2}V^2\right]_0^{V_{DS}}$$

$$I_D L = \ell L\left[V_{eff}V_{DS} - \frac{1}{2}V_{DS}^2\right]$$

$$@V_{DS} = V_{eff} \Rightarrow I_D = \frac{1}{2}\ell V_{eff}^2$$

21.9.2 Mobility Degradation

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

$$g_{m(mob-deg)} = \frac{\ell}{2\theta}$$

21.9.3 What about holes (PMOS)

In PMOS holes are the charge-carrier (electron movement in valence band)

$$\mu_p < \mu_n$$

In intrinsic silicon:

$$\mu_n \leq 1400[cm^2/Vs] = 0.14[m^2/Vs]$$
$$\mu_p \leq 450[cm^2/Vs] = 0.045[m^2/Vs]$$

$$\mu_n \approx 3\mu_p$$

$$v_{n\_max} \approx 2.3 \times 10^5[m/s]$$
$$v_{p\_max} \approx 1.6 \times 10^5[m/s]$$

Doping (

$$N_A \textbf{or} N_D$$

) reduces

$$\mu$$

21.10 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

CONTRIBUTED PAPER

Analog Circuit Design in Nanoscale CMOS Technologies

Classic analog designs are being replaced by digital methods, using nanoscale digital devices, for calibrating circuits, overcoming device mismatches, and reducing bias and temperature dependence.

By Lanny L. Lewyn, *Life Senior Member IEEE*, Trond Ytterdal, *Senior Member IEEE*, Carsten Wulff, *Member IEEE*, and Kenneth Martin, *Fellow IEEE*

https://ieeexplore.ieee.org/document/5247174

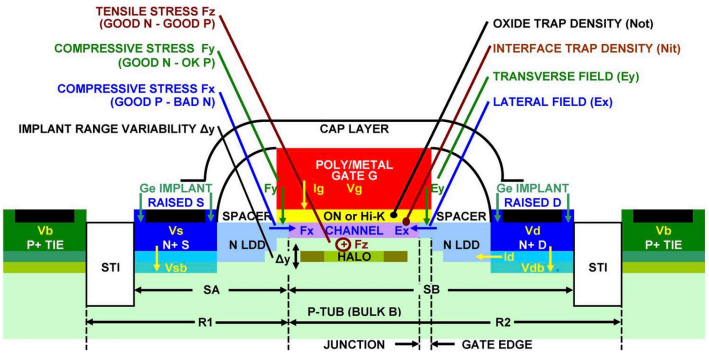
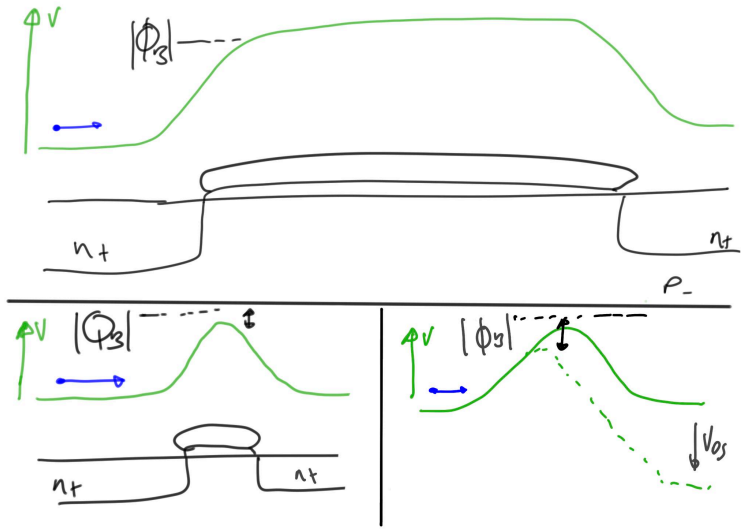
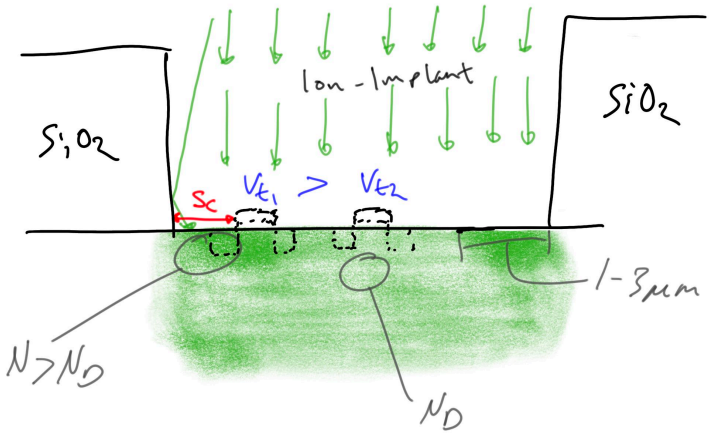


Fig. 2. *NMOS cross-section. In addition to stress from cap layers and Ge raised source-drain (S-D) implants, device dimensions such as distance from source-channel boundary to nearby STI (SA and SB), proximity and regularity of overlying metal patterns, and short distances to other device patterns within the local (< 2 μm) stress field induce transverse (F_y) and lateral (F_x and F_z) stress components, which affect threshold and mobility. Increasing the distance to P+ ties increases local tub (bulk) resistance components R1 and R2, which isolate the device MOS model substrate node from the device subcircuit symbol V_b node and degrade HF performance. Hot carrier reliability stress is dependent on the sum of transverse and lateral fields E_y and E_x. These fields are increased near the drain by increasing source to bulk (V_{sb}) and drain (V_d) to gate (V_g) or source (V_s) voltages in various combinations. As hot carrier stress increases, damage to channel from interface trap density (N_{it}) affects threshold and mobility, while gate oxynitirde (ON) or high-dielectric-constant (Hi-K) insulator trap density (N_{ot}) affects threshold and gate leakage.*

21.10.1 Drain induced barrier lowering (DIBL)



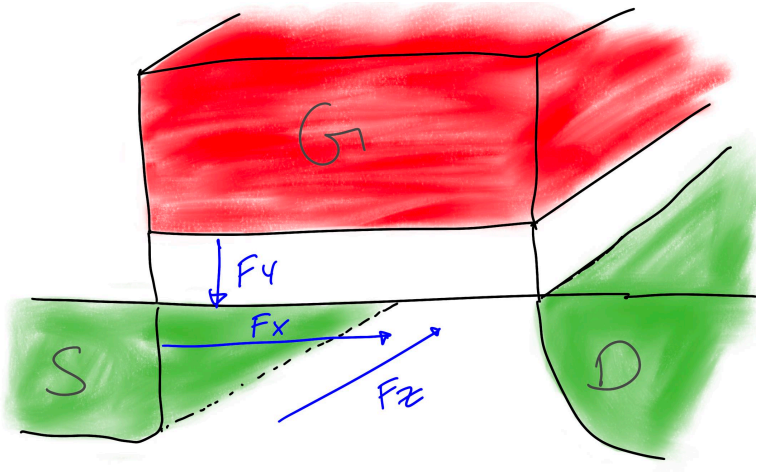
21.10.2 Well Proximity Effect (WPE)



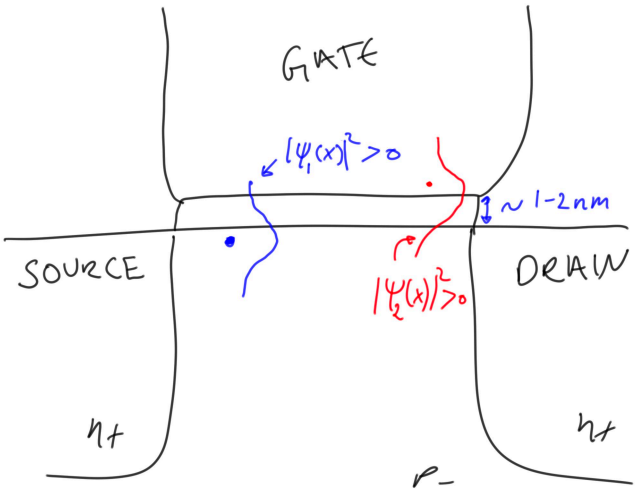
21.10.3 Stress effects

Stress	PMOS	NMOS
Stretch Fz	Good	Good
Compress Fy	OK	Good
Compress Fx	Good	Bad

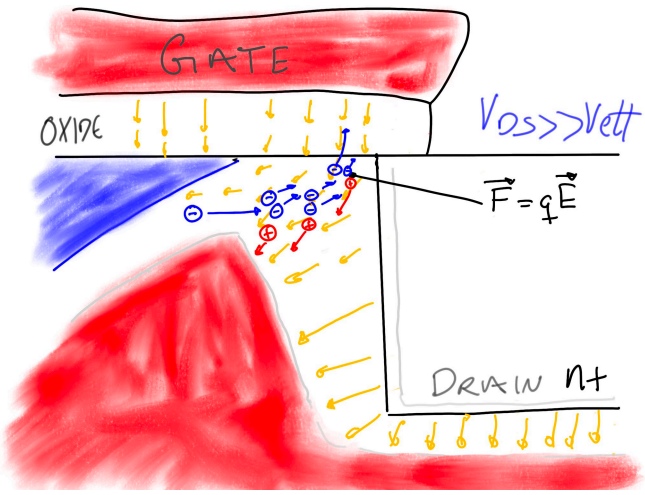
What can change stress?



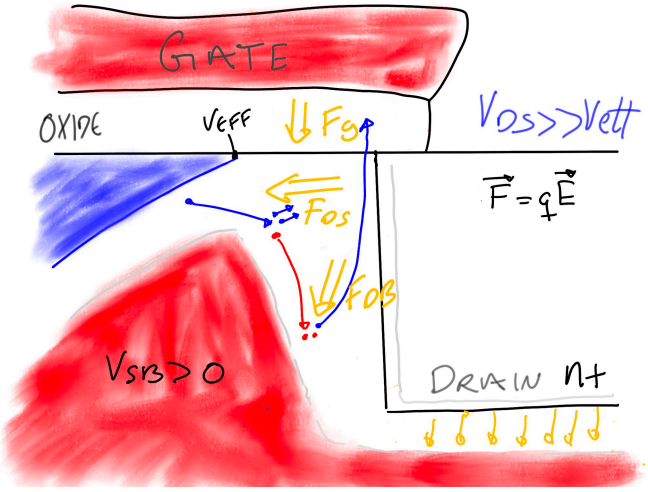
21.10.4 Gate current



21.10.5 Hot carrier injection



21.10.6 Channel initiated secondary-electron (CHISEL)



21.11 Variability

Provide

$$I_2 = 1\mu A$$

Let's use off-chip resistor

$$R$$

, and pick

$$R$$

such that

$$I_1 = 1\mu A$$

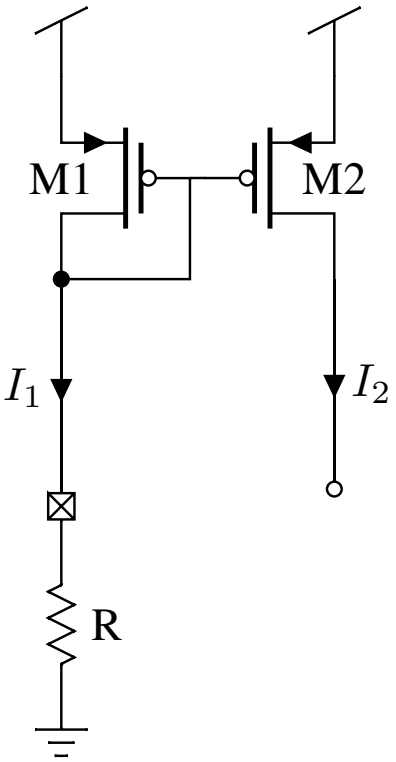
Use

$$\frac{W_1}{L_1} = \frac{W_2}{L_2}$$

What makes

$$I_2 \neq 1\mu A$$

?



- Voltage variation
- Systematic variations
- Process variations
- Temperature variation
- Random variations
- Noise

21.11.1 Voltage variation

$$I_1 = \frac{V_{DD} - V_{GS1}}{R}$$

If

$$V_{DD}$$

changes, then current changes.

Fix: Keep

$$V_{DD}$$

constant

21.11.2 Systematic variations

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

$$Stress_1 \neq Stress_2 \rightarrow I_1 \neq I_2$$

...

21.11.3 Process variations

Assume strong inversion and active

$$V_{eff} = \sqrt{\frac{2}{\mu_p C_{ox} \frac{W}{L}} I_1}$$

,

$$V_{GS} = V_{eff} + V_{tp}$$

$$I_1 = \frac{V_{DD} - V_{GS}}{R} = \frac{V_{DD} - \sqrt{\frac{2}{\mu_p C_{ox} \frac{W}{L}} I_1} - V_{tp}}{R}$$

$$\mu_p$$

,

$$C_{ox}$$

,

$$V_{tp}$$

will all vary from die to die, and wafer lot to wafer lot.

21.11.4 Process corners

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

21.11.5 Fix process variation

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 (approximately 1 M

$$\Omega$$

)

21.11.6 Temperature variation

Mobility decreases with temperature

Threshold voltage decreases with temperature.

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

High

$$I_D =$$

fast digital circuits

Low

$$I_D =$$

slow digital circuits

What is fast? High temperature or low temperature?

21.11.7 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

21.11.8 How do we fix temperature variation?

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)

21.11.9 Random Variation

$$\ell = \mu_p C_{ox} \frac{W}{L}$$

$$I_D = \frac{1}{2}\ell(V_{GS} - V_{tp})^2$$

Due to doping , length, width,

$$C_{ox}$$

,

$$V_{tp}$$

, . . . random varation

$$\ell_1 \neq \ell_2$$

$$V_{tp1} \neq V_{tp2}$$

As a result

$$I_1 \neq I_2$$

, but we can make them close.

21.11.10 Pelgrom's* law

Given a random gaussian process parameter

$$\Delta P$$

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 (

$$D \approx 0$$

)

$$\Rightarrow \sigma^2(\Delta P) = \frac{A_P^2}{WL}$$

21.11.11 Transistors with same

$$V_{GS}$$

†

$$\frac{\sigma_{I_D}^2}{I_D^2} = \frac{1}{WL}\left[\left(\frac{gm}{I_D}\right)^2 \sigma_{vt}^2 + \frac{\sigma_\ell^2}{\ell}\right]$$

Valid in weak, moderate and strong inversion

$$\frac{\sigma_{I_D}^2}{I_D^2} = \frac{1}{WL}\left[\left(\frac{gm}{I_D}\right)^2 \sigma_{vt}^2 + \frac{\sigma_\ell^2}{\ell}\right]$$

$$\frac{\sigma_{I_D}}{I_D} \propto \frac{1}{\sqrt{WL}}$$

---

* M. J. M. Pelgrom, C. J. Duinmaijer, and A. P. G. Welbers, "Matching properties of MOS transistors," IEEE J. Solid-State Cir- cuits, vol. 24, no. 5, pp. 1433–1440, Oct. 1989.

† Peter Kinget, see CJM

Assume

$$\frac{\sigma_{I_D}}{I_D} = 10\%$$

, We want

$$5\%$$

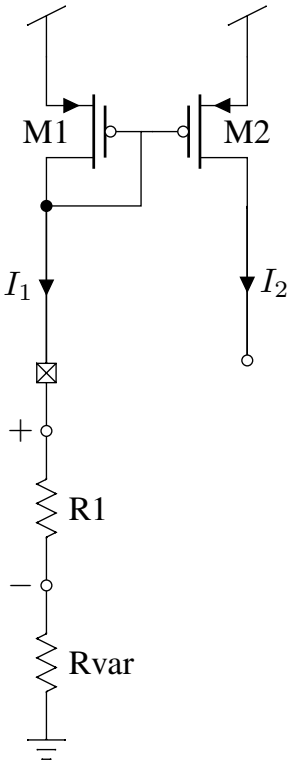
, how much do we need to change WL?

$$\frac{\frac{\sigma_{I_D}}{I_D}}{2} \propto \frac{1}{2\sqrt{WL}} = \frac{1}{\sqrt{4WL}}$$

We must quadruple the area to half the standard deviation

$$1\%$$

would require **100** times the area



21.11.12 What else can we do?

$$\frac{\sigma_{I_D}^2}{I_D^2} = \frac{1}{WL}\left[\left(\frac{gm}{I_D}\right)^2 \sigma_{vt}^2 + \frac{\sigma_\ell^2}{\ell}\right]$$

Strong inversion

$$\Rightarrow \frac{gm}{I_D} = \frac{1}{2V_{eff}} = low$$

Weak inversion

$$\Rightarrow \frac{gm}{I_D} = \frac{q}{nkT} \approx 25$$

Current mirrors achieve best matching in strong inversion

$$\frac{\sigma_{I_D}^2}{I_D^2} = \frac{1}{WL}\left[\left(\frac{gm}{I_D}\right)^2 \sigma_{vt}^2 + \frac{\sigma_\ell^2}{\ell}\right]$$

$$\sigma_{I_D}^2 = \frac{1}{WL}\left[gm^2\sigma_{vt}^2 + I_D^2\frac{\sigma_\ell^2}{\ell}\right]$$

Offset voltage for a differential pair

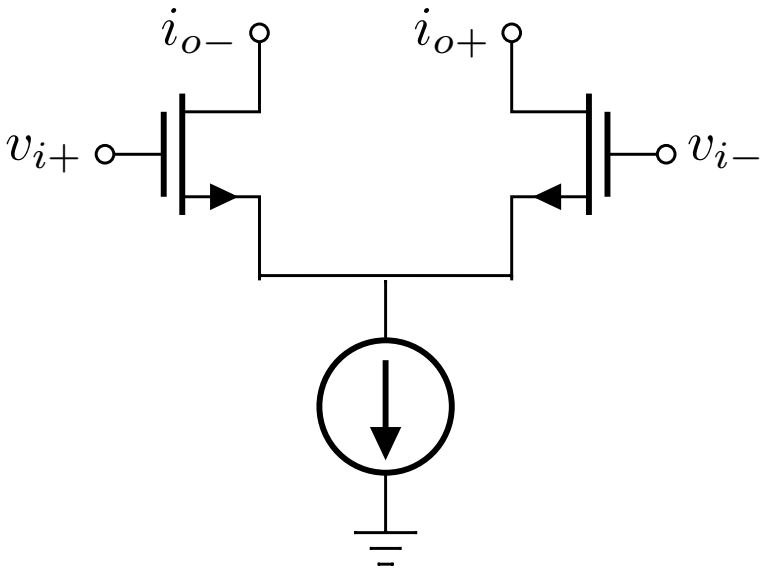
$$i_o = i_{o+} - i_{o-} = g_m v_i = g_m(v_{i+} - v_{i-})$$

$$\sigma_{v_i}^2 = \frac{\sigma_{I_D}^2}{gm^2} = \frac{1}{WL}\left[\sigma_{vt}^2 + \frac{I_D^2}{gm^2}\frac{\sigma_\ell^2}{\ell}\right]$$

High

$$\frac{gm}{I_D}$$

is better (best in weak inversion)



21.11.13 Transistor Noise

Thermal noise Random scattering of carriers, generation-recombination in channel?

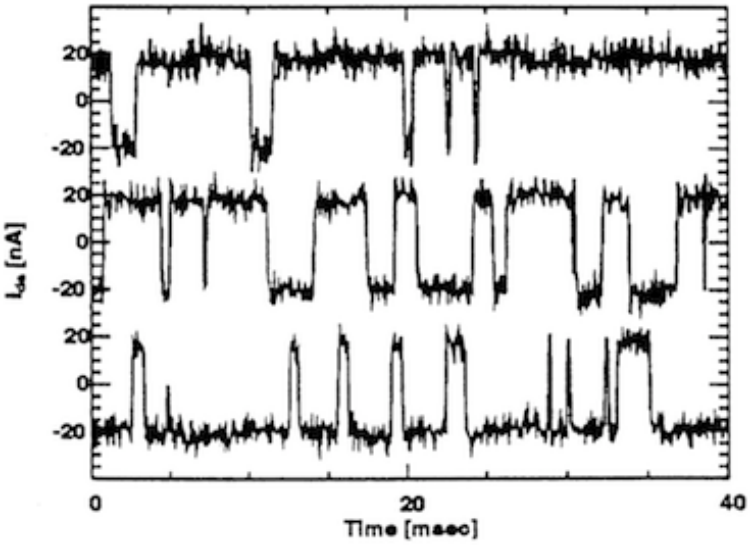
$$PSD_{TH}(f) = \text{Constant}$$

Popcorn noise Carriers get "stuck" in oxide traps (dangling bonds) for a while. Can cause a short-lived (seconds to minutes) shift in threshold voltage

$$PSD_{GR}(f) \propto \text{Lorentzian shape} \approx \frac{A}{1 + \frac{f^2}{f_0}}$$

Flicker noise Assume there are many sources of popcorn noise at different energy levels and time constants, then the sum of the spectral densities approaches flicker noise.

$$PSD_{flicker}(f) \propto \frac{1}{f}$$



Circuits 22

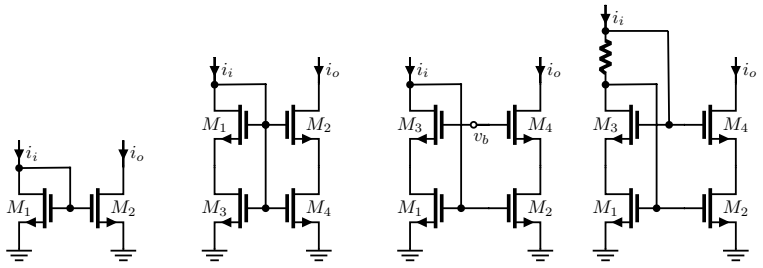
22.1 Current Mirrors

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 mirros. 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

22.1.1 Normal current mirror

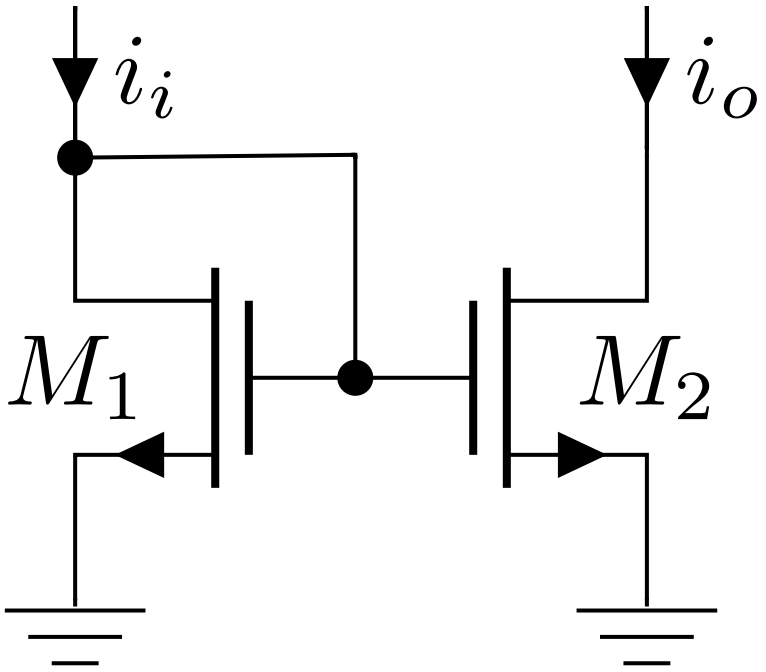
The normal current mirror consists of a diode connected transistor (M_1) and a common source transistor M_2.

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.

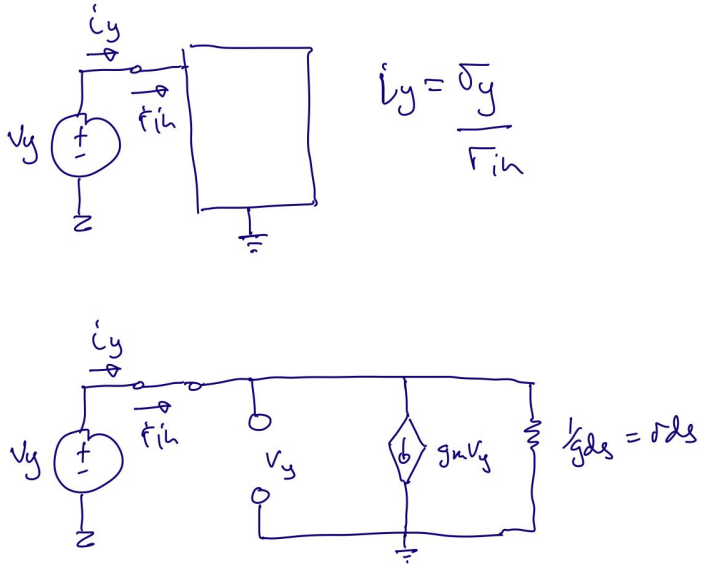
A current pushed into M_1 will cause the V_{GS1} to rise, and at some point, find a stable point where the current pushed in is equal to the current in M_1

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 transitor is in weak-inversion.



22.1.1.1 Input resistance

To see the small signal input resistance we can apply a test voltage to the diode connected resistor, as shown in the figure below.



Observe the current

$$i_y = g_{ds} v_y + g_m v_y$$

While the input resistance

$$r_{in} = \frac{v_y}{i_y} = \frac{1}{g_m + g_{ds}}$$

which, assuming $g_{ds} >> g_m$, reduces to

$$r_{in} \approx \frac{1}{g_m}$$

.

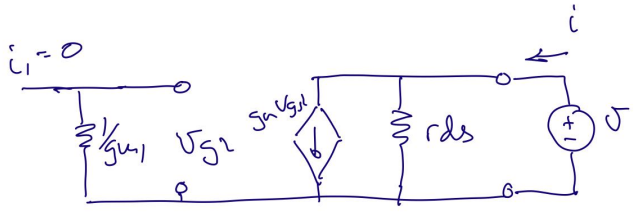
Assume now I apply 1 μA current to the diode connected transistor, and the $g_m = 1\ \mu S$.

Would the voltage be $v_y = r_{in} i_y = \frac{1\ \mu A}{1\ \mu S} = 1\ V$? NO! It's important to understand the difference between the small signal input resistance, and the large signal impedance.

The large signal impedance is a highly non-linear function (we've seen before that the current in a MOSFET has both an exponential, and a square-law, and sometimes a linear with voltage), as such, there is no single function describing what the gate-source voltage will be.

To see the DC voltage, apply a current in SPICE, and use a simulator to find the voltage.

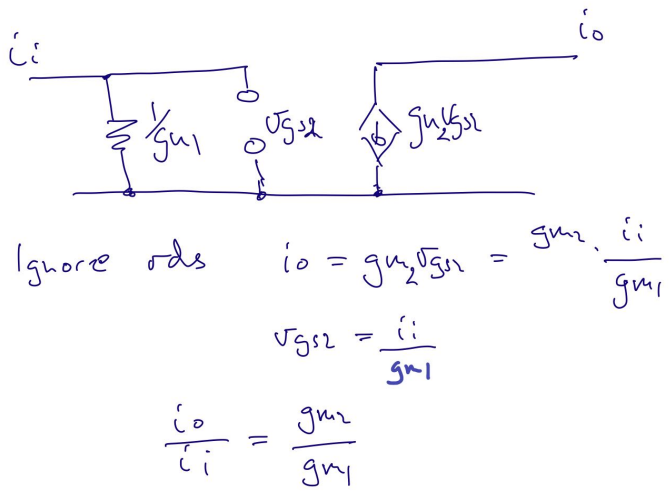
22.1.1.2 Output resistance



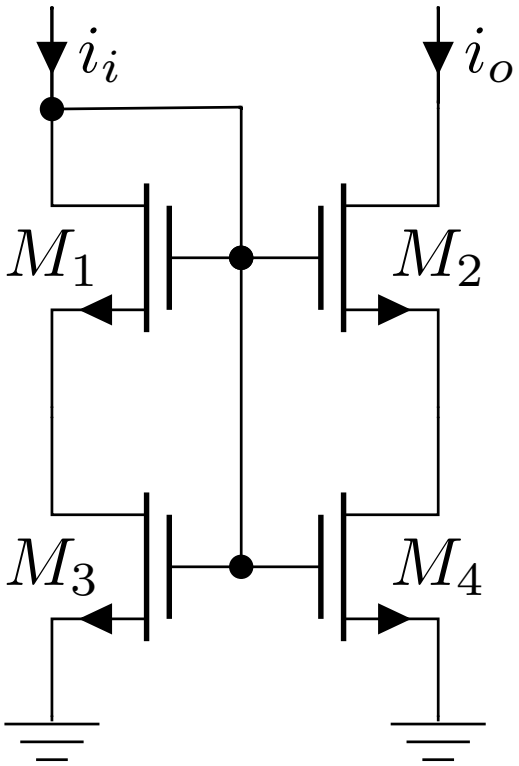
V_y does not affect V_{gs2}

$r_{out} = r_{ds}$

22.1.1.3 Current gain



22.1.2 Source degeneration



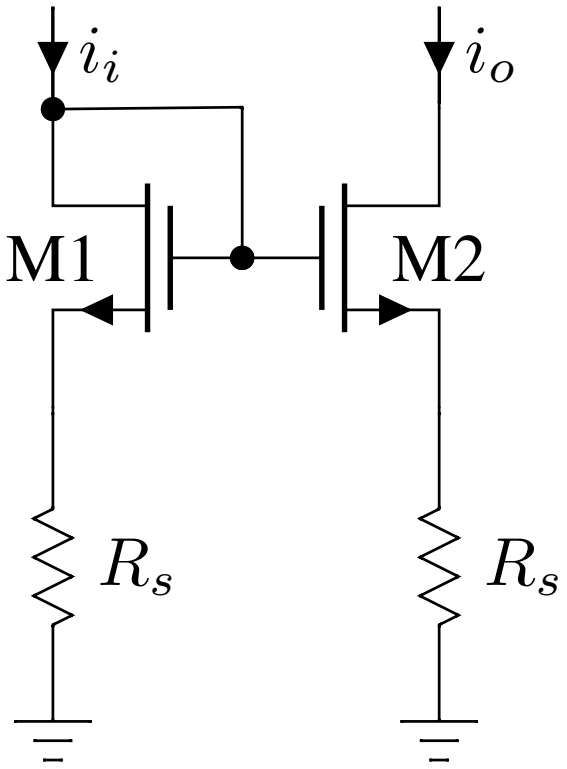
What is the operating region of M3 and M4?

What is the operating region of M1 and M2?

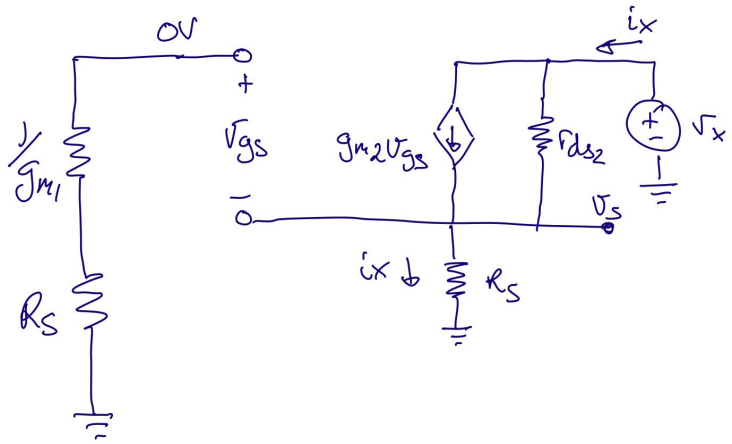
22.1.2.1 Input resistance

M1 and M2 are in linear region, can be simplified to resistors

$$r_{in} = \frac{1}{g_{m1}} + R_s$$



22.1.3 Output resistance



$$v_{gs} = -v_s$$

,

$$v_s = i_x R_s$$

,

$$r_{out} = \frac{v_x}{i_x}$$

$$i_x = g_{m2} v_{gs} + \frac{v_x - v_s}{r_{ds2}}$$

$$i_x = -i_x g_{m2} R_s + \frac{v_x - i_x R_s}{r_{ds2}}$$

$$v_x = i_x \left[r_{ds2} + R_s (g_{m2} r_{ds2} + 1) \right]$$

Rearranging

$$r_{out} = r_{ds2}[1 + R_s(g_{m1} + g_{ds2})] \approx r_{ds2}[1 + g_{m1} R_s]$$

22.1.3.1 Cascode output resistance

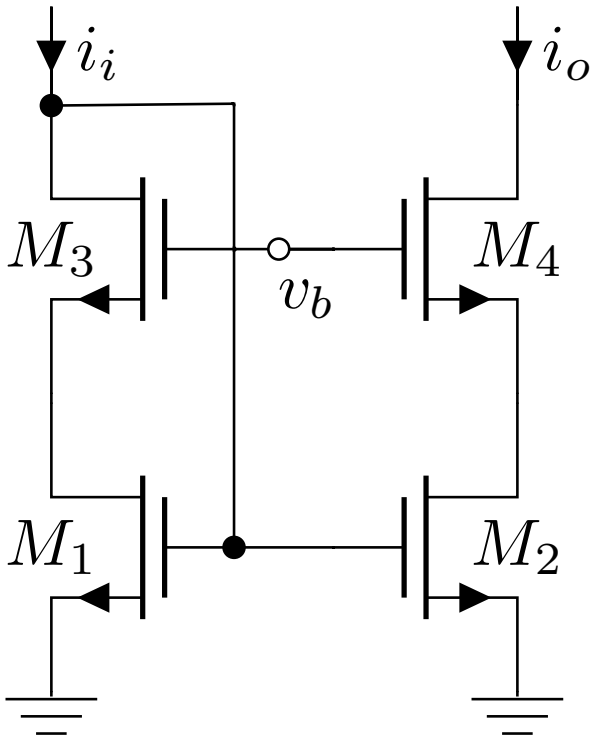
From source degeneration (ignoring bulk effect)

$$r_{out} = r_{ds4}[1 + R_s(g_{m4} + g_{ds4})]$$

$$R_S = r_{ds2}$$

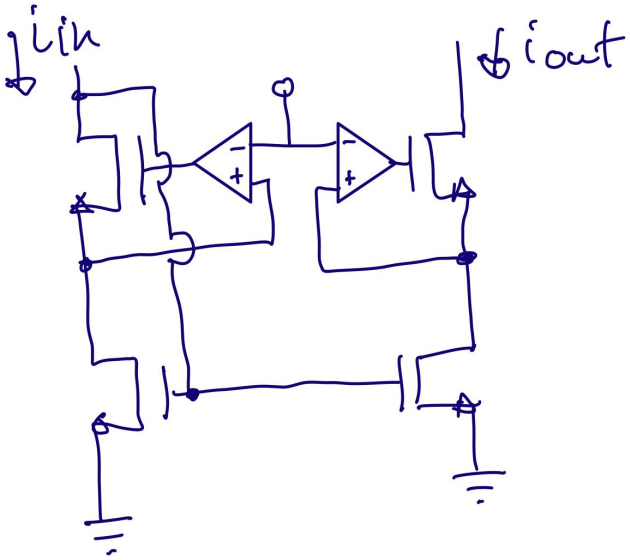
$$r_{out} = r_{ds4}[1 + r_{ds2}(g_{m4} + g_{ds4})]$$

$$r_{out} \approx r_{ds2}(r_{ds4} g_{m4})$$



22.1.3.2 Active cascodes

$$r_{out} \approx r_{ds2}(Ar_{ds4}g_{m4})$$



22.2 Amplifiers

22.3 Source follower

Input resistance

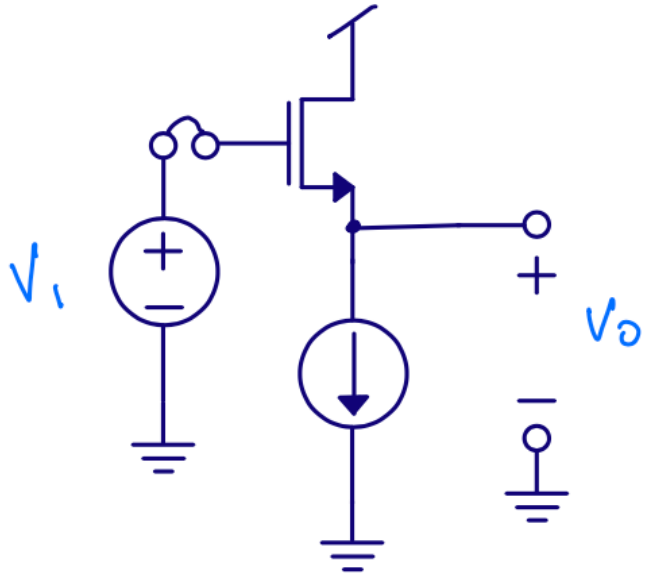
$$\approx \infty$$

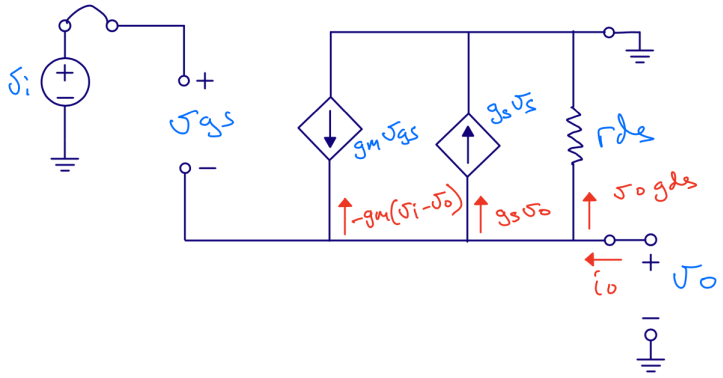
Gain

$$A = \frac{v_o}{v_i}$$

Output resistance

$$r_{out}$$





22.3.1 Output resistance

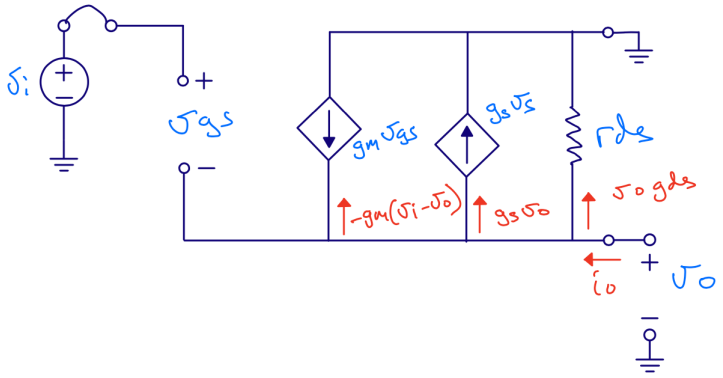
$$i_o = v_o(g_{ds} + g_s) - g_m v_i + v_o g_m$$

$$v_i = 0$$

$$i_o = v_o(g_{ds} + g_s + g_m)$$

$$r_{out} = \frac{v_o}{i_o} = \frac{1}{g_m + g_{ds} + g_s}$$

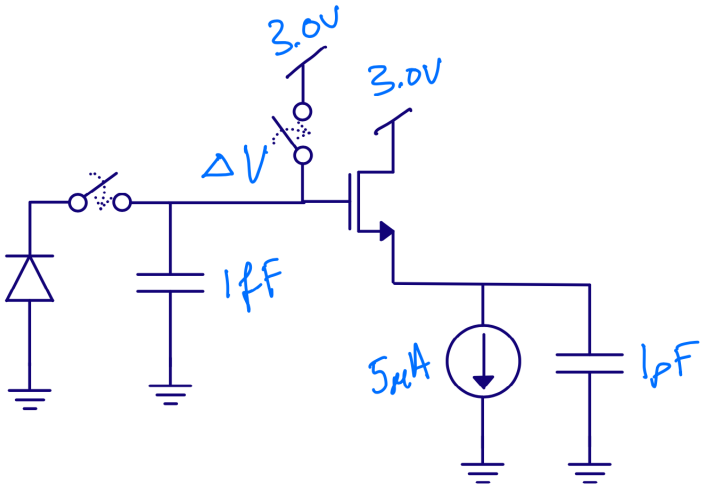
$$r_{out} \approx \frac{1}{g_m}$$



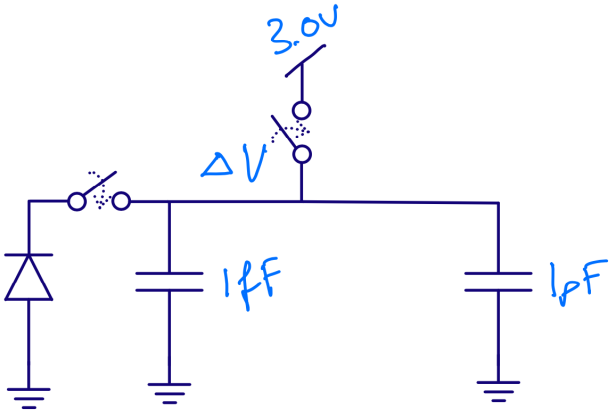
22.3.2 Why use a source follower?

Assume 100 electrons

$$\Delta V = Q/C = -1.6 \times 10^{-19} \times 100/(1 \times 10^{-15}) = -16 \text{ mV}$$



$$\Delta V = Q/C = -1.6 \times 10^{-19} \times 100/(1 \times 10^{-12}) = -16 \text{ uV}$$

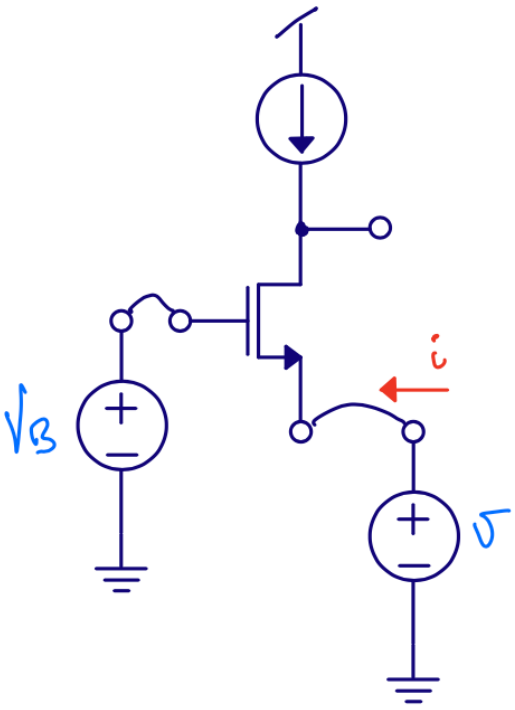


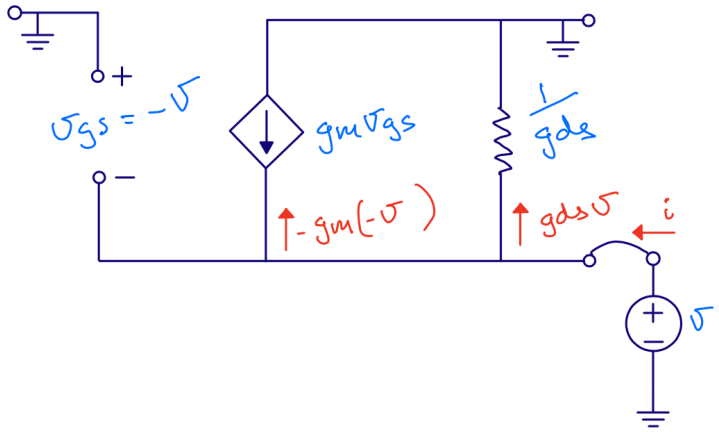
22.4 Common gate

Input resistance

Gain

Output resistance





22.4.1 Input resistance

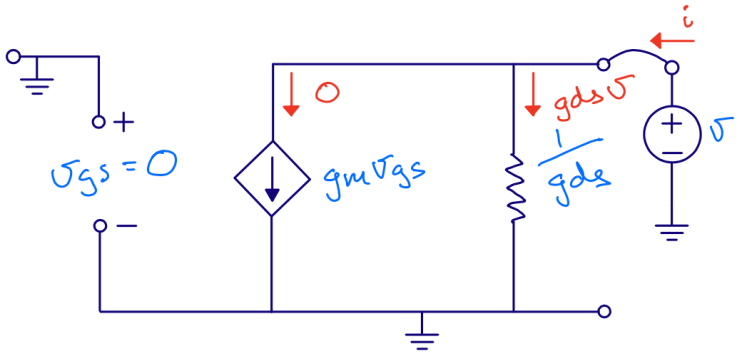
$$i = g_m v + g_{ds} v$$

$$r_{in} = \frac{1}{g_m + g_{ds}} \approx \frac{1}{g_m}$$

However, we've ignored load resistance.

$$r_{in} \approx \frac{1}{g_m}\left(1 + \frac{R_L}{r_{ds}}\right)$$

22.4.2 Output resistance



22.4.3 Gain

$$i_o = -g_m v_i + \frac{v_o - v_i}{r_{ds}}$$

$$i_o = 0$$

$$0 = -g_m v_i r_{ds} + v_o - v_i$$

$$v_i(1 + g_m r_{ds}) = v_o$$

$$\frac{v_o}{v_i} = 1 + g_m r_{ds}$$

We've ignored bulk effect (g_s), source resistance (R_S) and load resistance (R_L)

$$A = \frac{(g_m + g_s + g_{ds})(R_L || r_{ds})}{1 + R_S \left(\frac{g_m + g_s + g_{ds}}{1 + R_L / r_{ds}} \right)}$$

If $R_L >> r_{ds}$, $R_S = 0$ and $g_s = 0$

$$A = \frac{(g_m + g_{ds}) r_{ds}}{1} = 1 + g_m r_{ds}$$

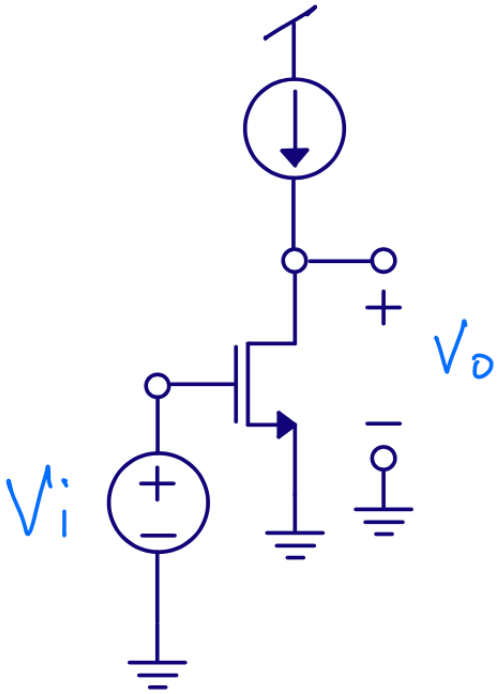
22.5 Common source

$$r_{in} \approx \infty$$

$$r_{out} = r_{ds}$$

, it's same circuit as the output of a current mirror

Gain



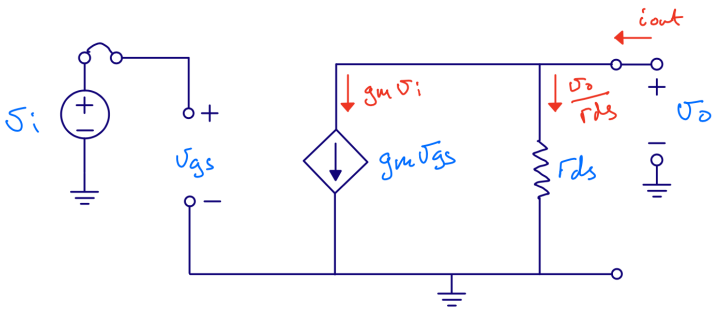
22.5.1 Gain

$$i_o = g_m v_i + \frac{v_o}{r_{ds}}$$

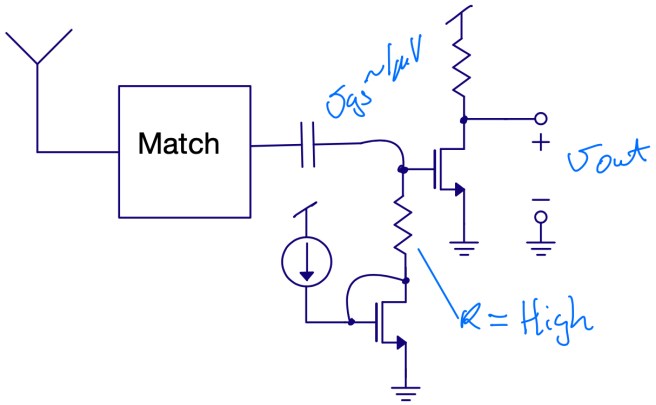
$$i_o = 0$$

$$-g_m v_i = \frac{v_o}{r_{ds}}$$

$$\frac{v_o}{v_i} = -g_m r_{ds}$$



22.5.2 Why common source?



22.6 Differential pair

Input resistance

$$r_{in} \approx \infty$$

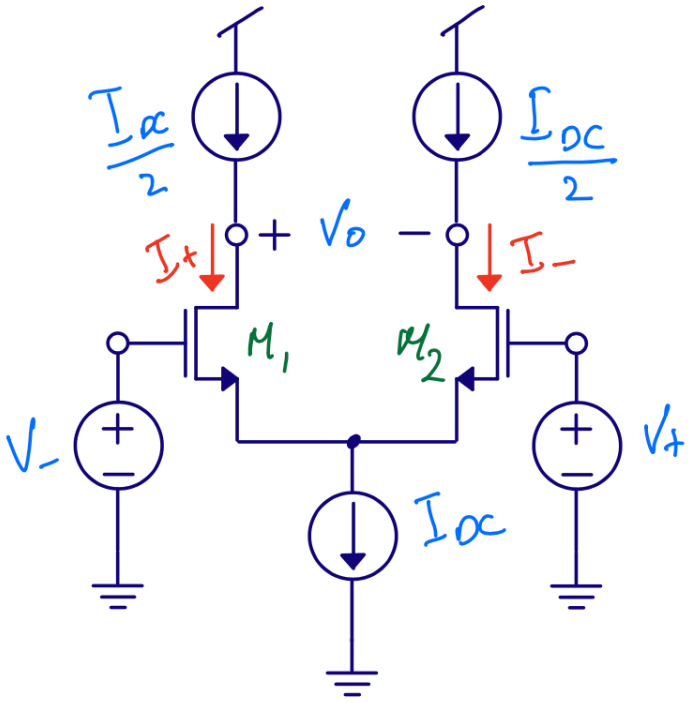
Gain

$$A = g_m r_{ds}$$

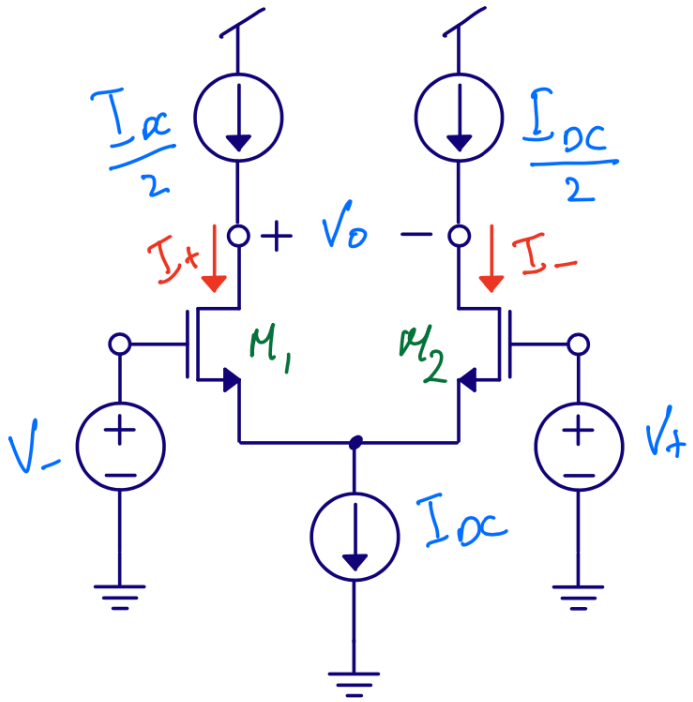
Output resistance

$$r_{out} = r_{ds}$$

Best analyzed with T model of transistor (see CJM page 31)



22.6.1 Diff pairs are cool



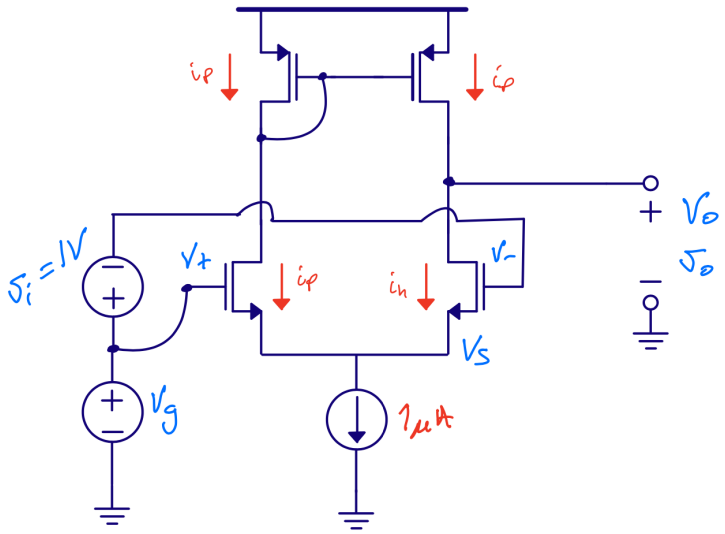
Can choose between

$$v_o = g_m r_{ds} v_i$$

and

$$v_o = -g_m r_{ds} v_i$$

by flipping input (or output) connections



23 Integrated Passives

Status: 0.3

23.1 Metal in ICs is not wire in schematic

Metal wires in an integrated circuit comes 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 determine what we must simulate. Everything needs to be simulated with parasitc capacitance and max current. Only RF, however, usually needs to be simulated with resistance, capacitance, inductance and maximum current.

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

23.2 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.

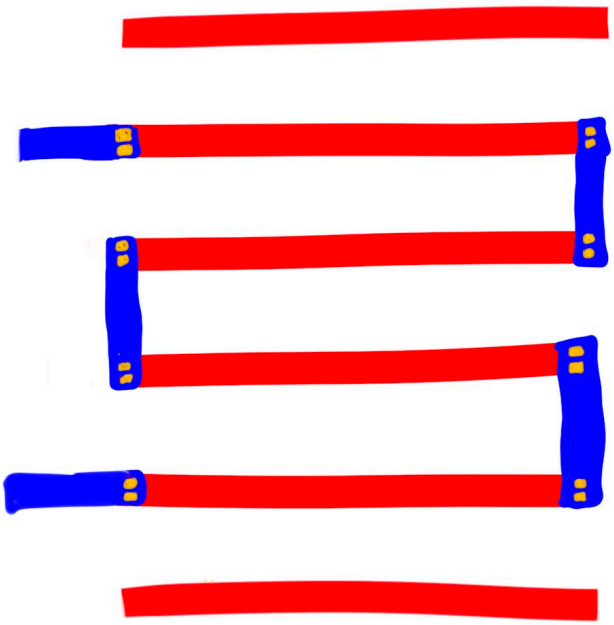
That's why almost all analog circuits rely on the relative sizes of passives, not the absolute value. If a circuit does rely on absolute values, then it usually needs to be trimmed in production.

23.2.1 Polysilicon

Can be both N-doped, and P-doped

Often with two flavors, with, and without silicide

Silicide reduces resistance of polysilicon



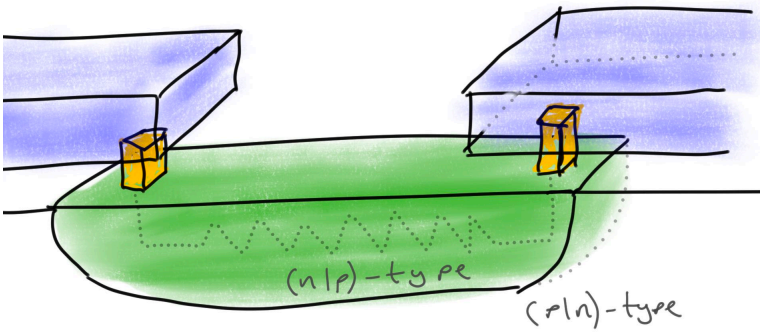
23.2.2 Diffusion

Use doped region as resistor

Usually without silicide

Non-linear capacitance

Tricky temperature dependence

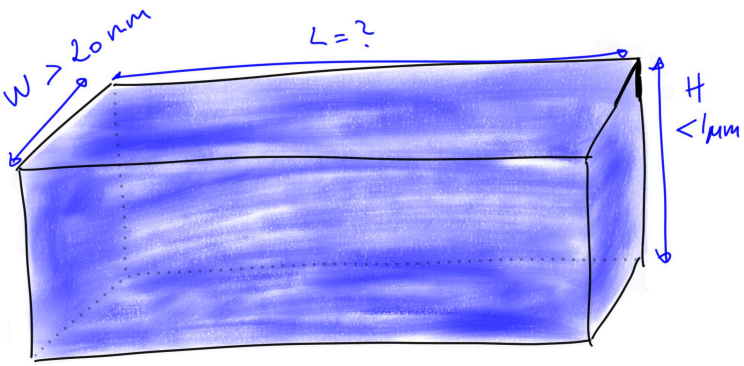


23.2.3 Metal

Usually too low omhic to be a useful resistor

Useful for “separating nets” in schematic and layout

Must be considered for power supply and ground routing (high currents)



23.3 Capacitors

23.3.1 What is S, M, L, XL on a chip?

nRF52832

$$3200\mu m \times 3000\mu m = 9600k\mu m^2$$

S

$$< 5\ k\mu m^2$$

M

$$< 50\ k\mu m^2$$

L

$$< 200\ k\mu m^2$$

XL

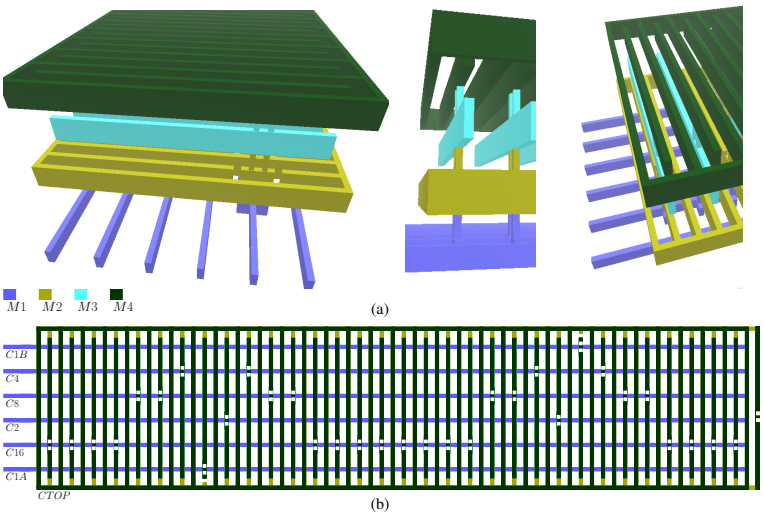
$$> 200\ k\mu m^2$$

23.3.2 Metal-Oxide-Metal finger capacitors

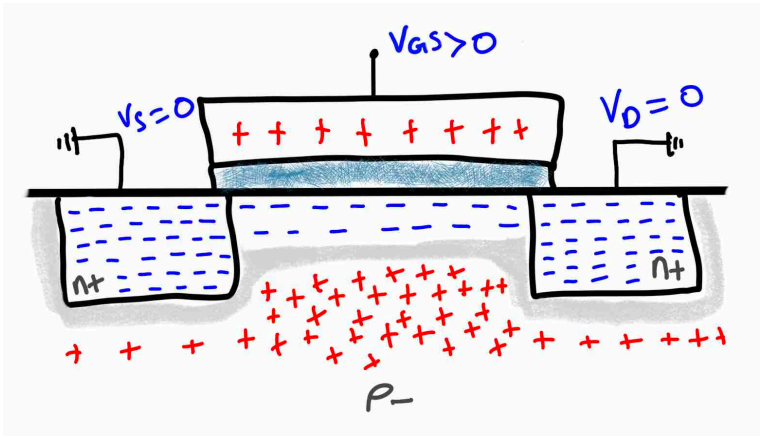
Unit capacitance

$$\approx 1fF/\mu m^2/layer$$

$$10pF = 100\mu m \times 100\mu m = 10k\mu m^2$$



23.3.3 MOS capacitors




```
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
```

Moscap is

$$\approx 10fF/\mu m^2$$

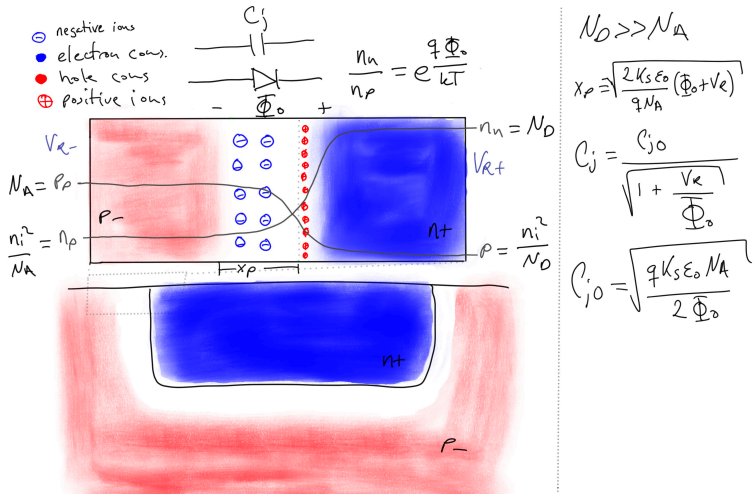
$$10pF = 31\mu m \times 31\mu m \approx 1k\mu m^2$$

```
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
```

23.3.4 Varactors

A varactor is a "variable capacitor", usually it's a device that varies the capacitance with the voltage across the device.

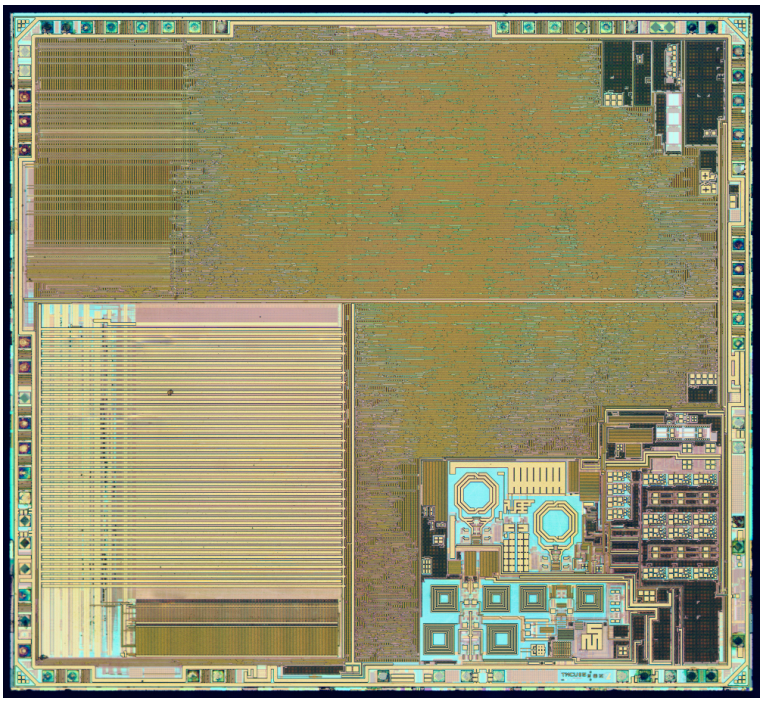


23.4 Inductors

Usually two top metals, because they are thick (low ohmic)

Use foundry model

3D electro magnetic simulation often needed



23.5 Variation in passives

Absolute value for resistors and capacitors

$$\approx \pm 10$$

% to

$$\pm 20$$

%

Relative precision for closely spaced devices

$$\approx$$

0.1 % to 1 %

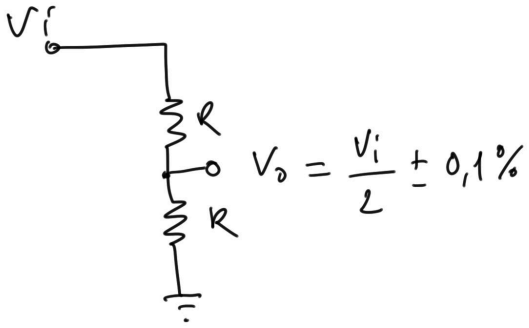
Relative precision for devices on same die

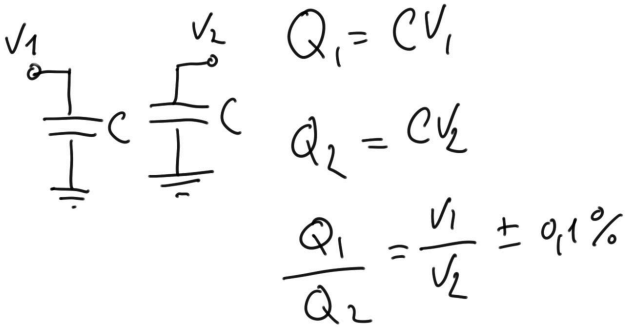
$$> 2$$

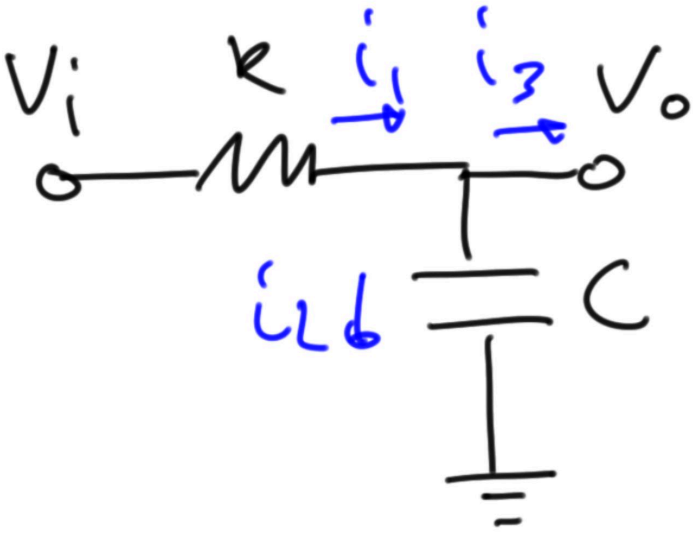
% or more

23.6 Relative precision

Resistors and Capacitors can be matched extremely well







$$i_3 = 0 = i_1 - i_2$$

$$0 = \frac{V_i - V_o}{R} - \frac{V_o}{1/sC}$$

$$0 = V_i - V_o - V_o sRC$$

$$V_o(1 + sRC) = V_i$$

$$\frac{V_o}{V_i} = \frac{1}{1 + sRC}$$

Assume standard deviation (

$$\sigma$$

)* of

$$\sigma_R = 20$$

%,

$$\sigma_C = 20$$

%

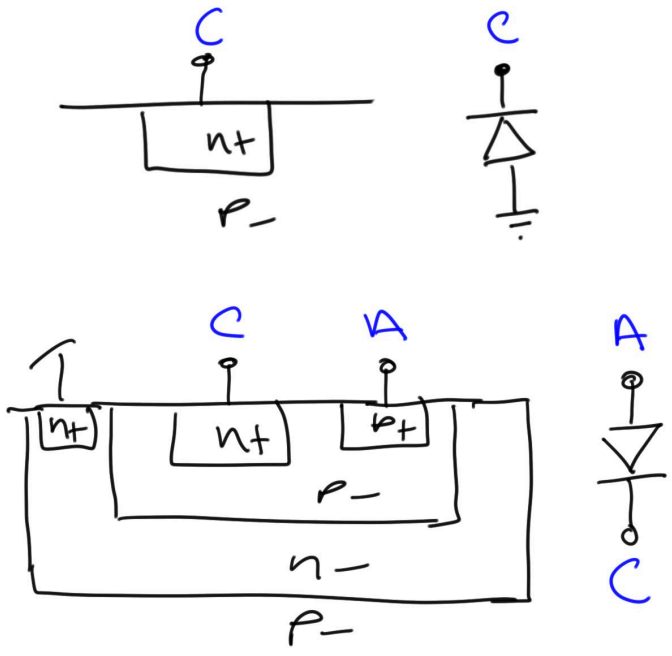
$$\sigma_{RC} = \sqrt{0.2^2 + 0.2^2} = 28$$

%

23.7 Diodes

Many, many ways

Reverse bias diodes to ground are useful for signals with long routing to transistor gate. Protects gate from breakdown during chemical mechanical polish.



* If you don't remember how standard deviation works, read Introduction to mathematics of noise sources

SPICE 24

24.1 SPICE

Status: 0.3

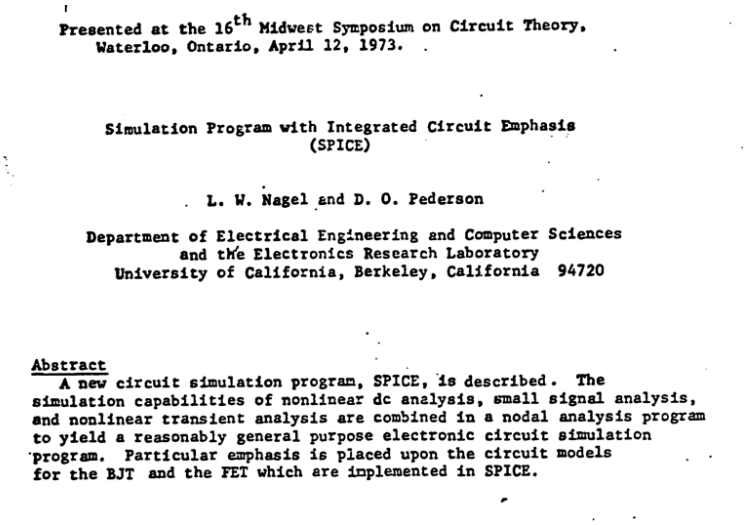
24.2 Simulation Program with Integrated Circuit Emphasis

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.

24.2.1 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

24.2.2 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 48 years ago

```
<spice program> testbench.cir
```

for example

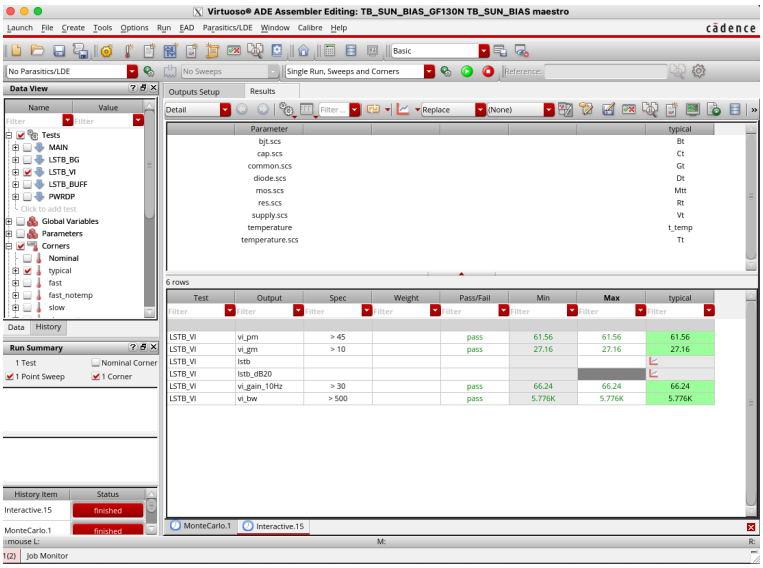
```
ngspice testbench.cir
```

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



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.

24.2.3 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

24.2.3.1 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
```

24.2.3.2 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
```

24.2.4 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
```

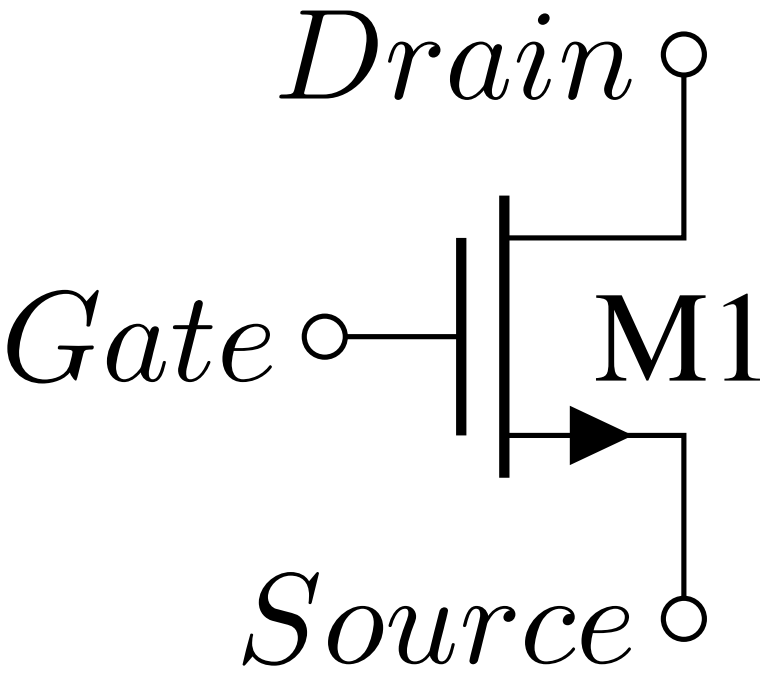
Capacitors

```
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
```

24.2.5 Transistor Models

Needs a model file the transistor model

BSIM (Berkeley Short-channel IGFET Model) http://bsim.berkeley.edu/models/bsim4/



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
```

24.2.6 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
```

24.2.7 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/

24.3 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:

$$I_D = I_{D0}\frac{W}{L}e^{V_eff/nV_T}$$

,

$$V_{eff} \propto \ln I_D$$

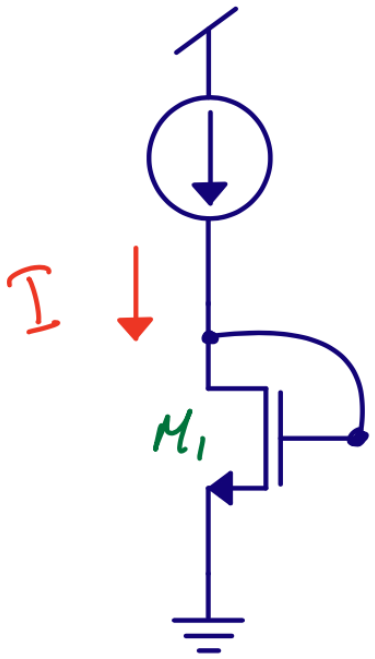
Strong inversion:

$$I_D = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}V_{eff}^2$$

,

$$V_{eff} \propto \sqrt{I_D}$$

Operating region for a diode connected transistor only depends on the current



24.3.1 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

24.3.2 What about gm/Id ?

Weak

$$\frac{g_m}{I_d} = \frac{1}{nV_T}$$

Strong

$$\frac{g_m}{I_d} = \frac{2}{V_{eff}}$$

24.3.3 Characterize the transistors

http://analogicus.com/cnr_atr_sky130nm/mos/CNRATR_NCH_2C1F2.html

24.4 More information

Ngspice Manual

Installing tools

24.5 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 your done.

On failure, go back

24.6 Demo

https://github.com/analogicus/jnw_spice_sky130A/tree/main

CMOS Logic 25

25.1 CMOS Logic

Status: 0.3

25.2 Analog transistor to digital transistor

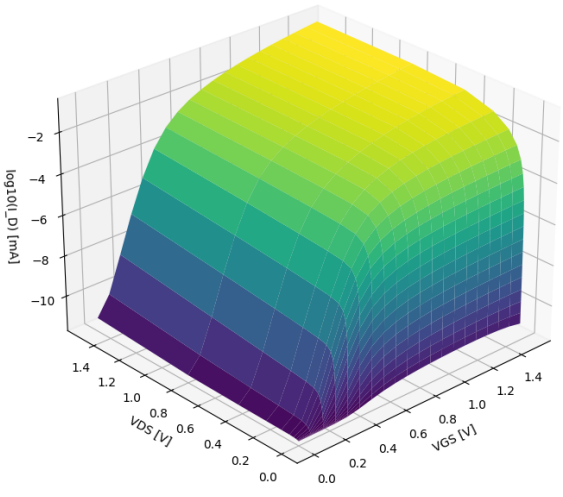
NMOS current (W = 0.4u L=0.15u) as a function of

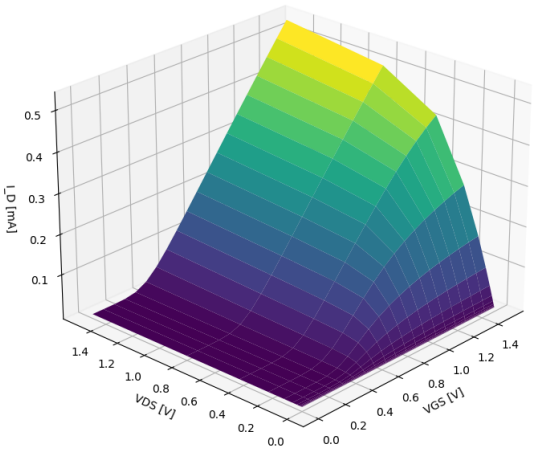
$$V_{GS}$$

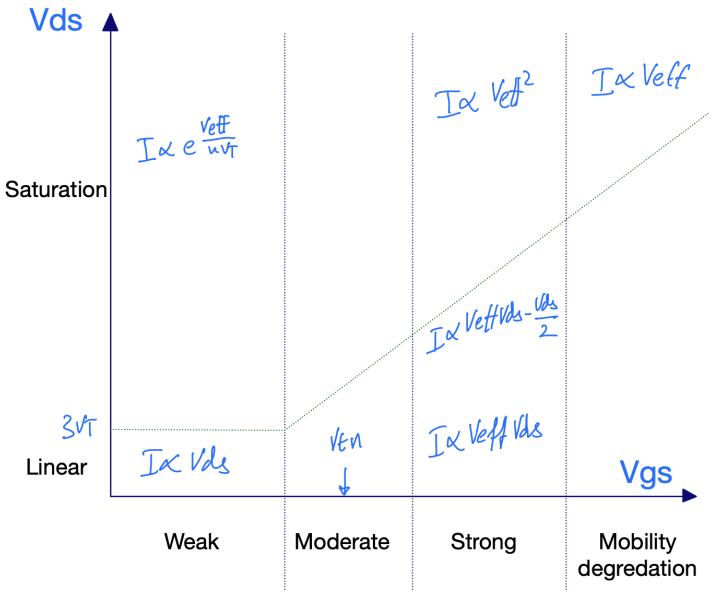
and

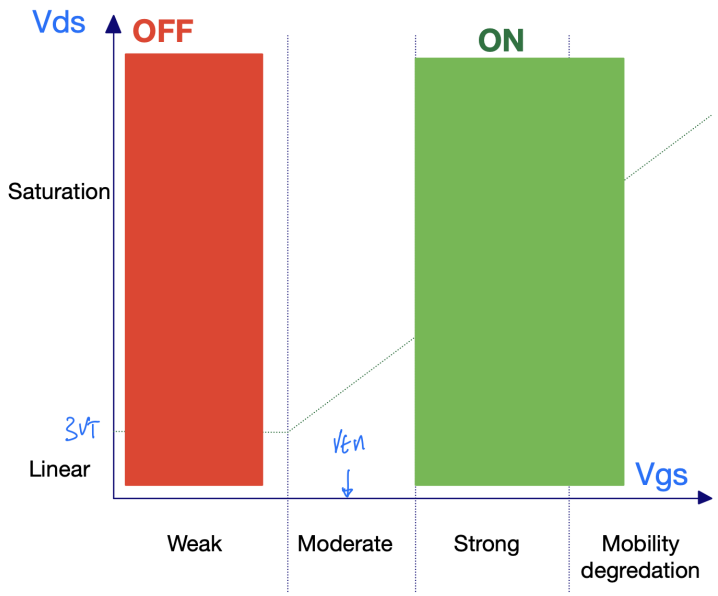
$$V_{DS}$$

dicex/lectures/l13/mos.py




0.5
0.4
0.3
0.2
0.1
i_D [mA]
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
VDS [V]
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
VGS [V]


Vds
Saturation
Linear
Vgs
Weak
Moderate
Strong
Mobility degredation
I∝ e^(Veff/nVT)
I∝ Veff²
I∝ Veff
I∝ VeffVds - Vds/2
3VT
I∝ Vds
VGN
I∝ Veff Vds



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

25.3 CMOS static logic assumptions

NMOS source is connected to low potential

$$V_{GS} > V_{TH}$$

when

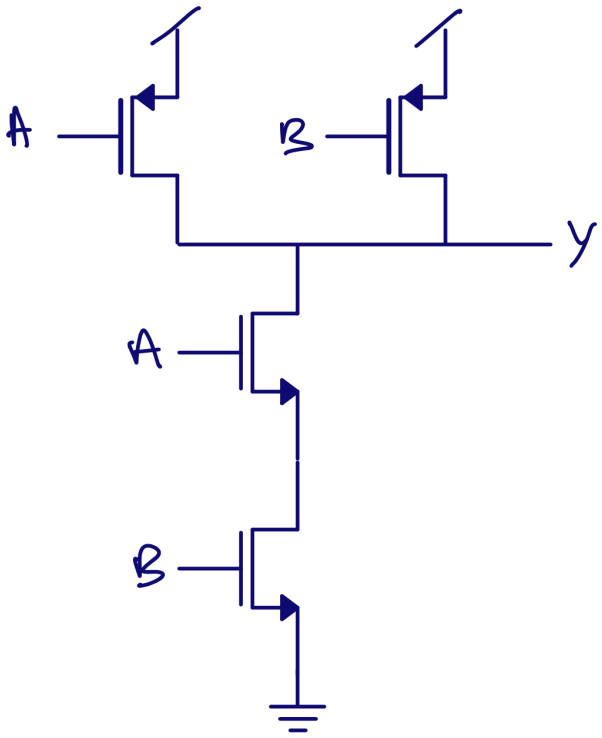
$$V_G = V_{DD}$$

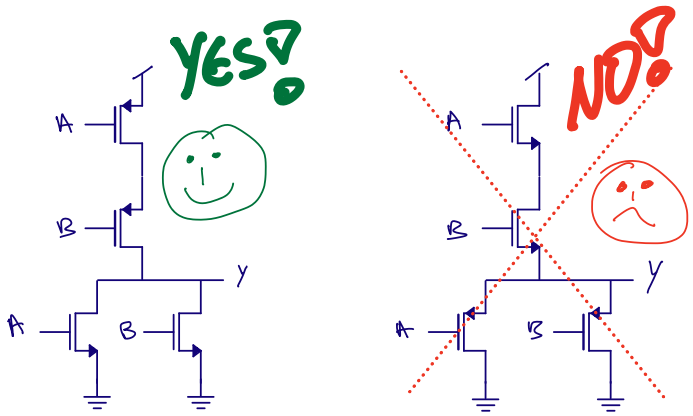
PMOS source is connected to high potential

$$V_{GS} < V_{TH}$$

when

$$V_G = 0$$


A
B
Y
A
B


YES!
A
B
Y
A
B
NO!
A
B
Y
A
B

25.4 Don't break rules unless you know exactly why it will be OK

25.5 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

$AB = \overline{\overline{A}+\overline{B}}$

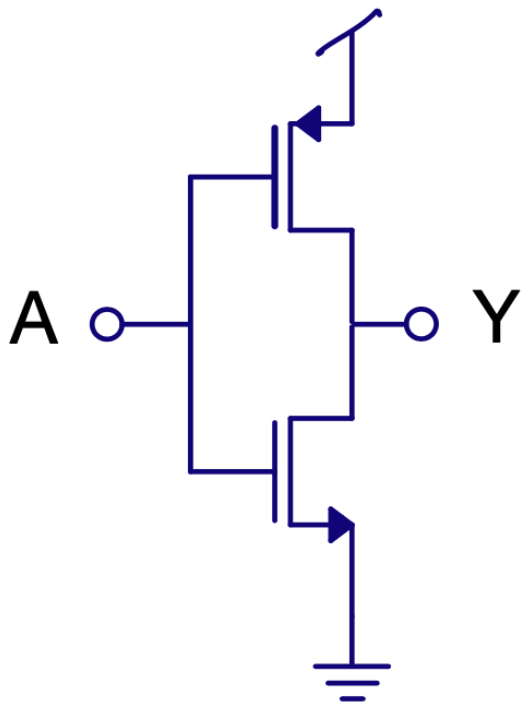
$A+B = \overline{\overline{A}\,\overline{B}}$

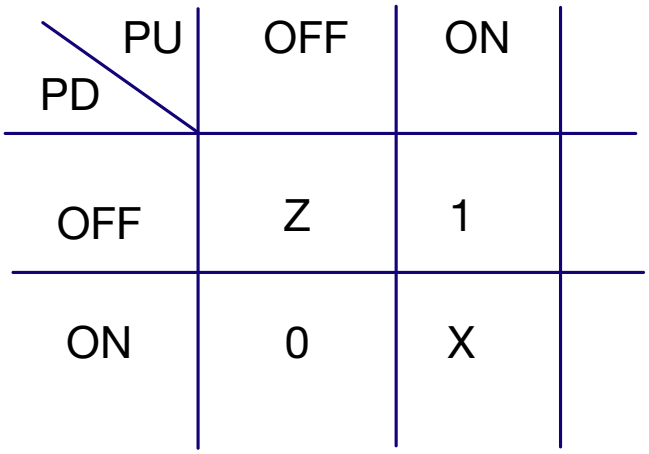
$\overline{AB} \neq \overline{A}\,\overline{B}$

$\overline{A}+\overline{B} \neq \overline{A+B}$

25.5.1 CMOS static logic is inverting

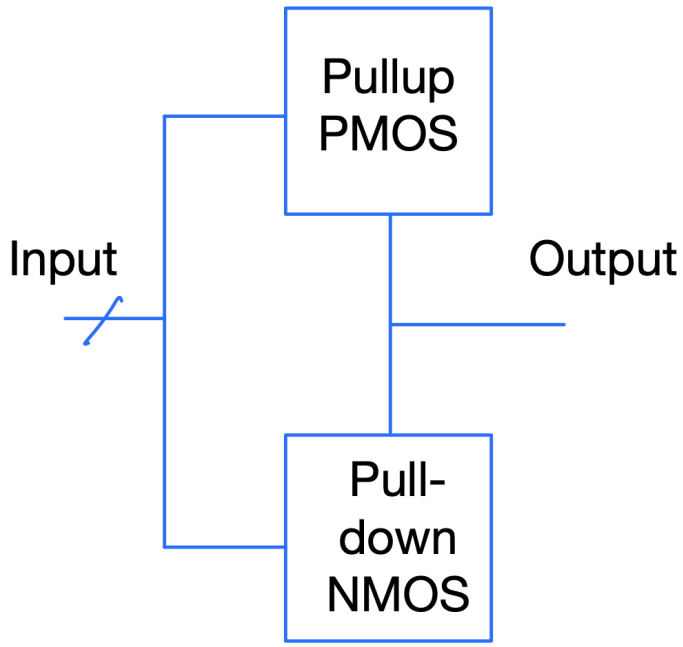
A	Y
1	0
0	1





PD = Pull-down PU = Pull-up

```
logic => [0,1,Z,X];
```



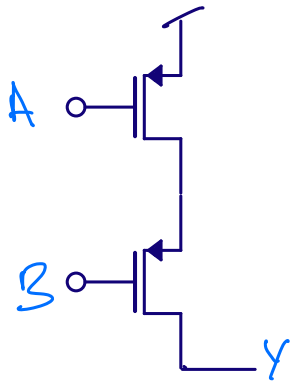
[.table-separator: #000000, stroke-width(1)] [.table: margin(8)]

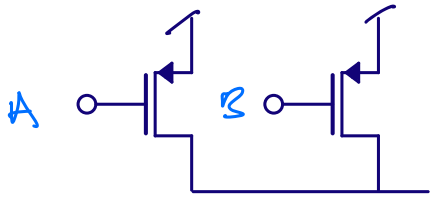
Pull-up series

A	B	Y
0	0	1
0	1	Z
1	0	Z
1	1	Z

Pull-up paralell

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	Z





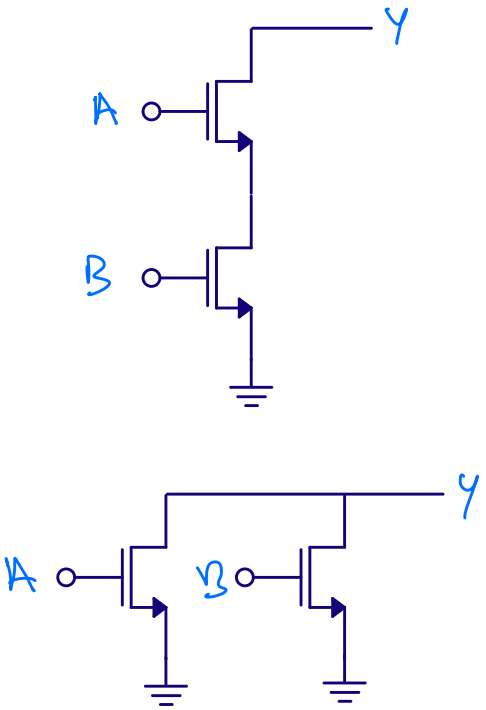
[.table-separator: #000000, stroke-width(1)] [.table: margin(8)]

Pull-down series

A	B	Y
0	0	Z
0	1	Z
1	0	Z
1	1	0

Pull-down paralell

A	B	Y
0	0	Z
0	1	0
1	0	0
1	1	0



25.5.2 Rules for inverting logic

Pull-up OR

⇒

PMOS in series

⇒

POS AND

⇒

PMOS in paralell

⇒

PAP

Pull-down OR

⇒

NMOS in paralell

⇒

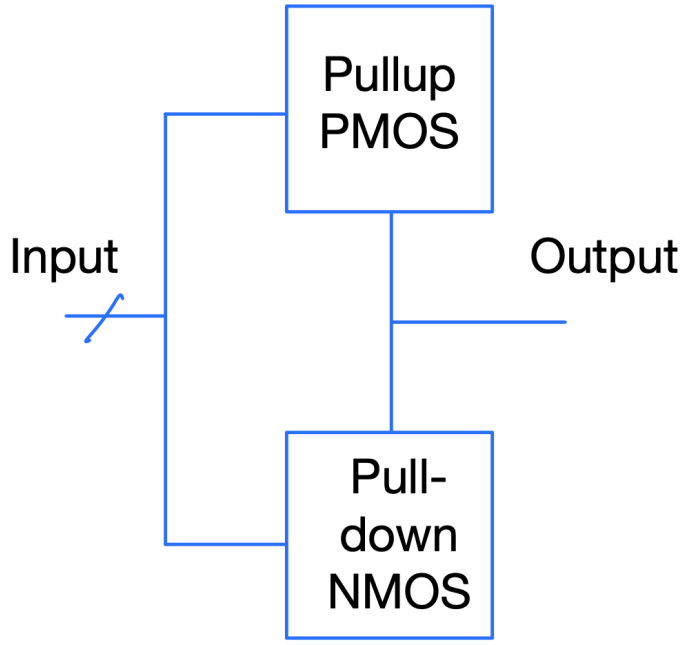
NOP AND

⇒

NMOS in series

⇒

NAS



[.table-separator: #000000, stroke-width(1)] [.table: margin(8)]

$$Y = \overline{AB} = \text{NOT (A AND B)}$$

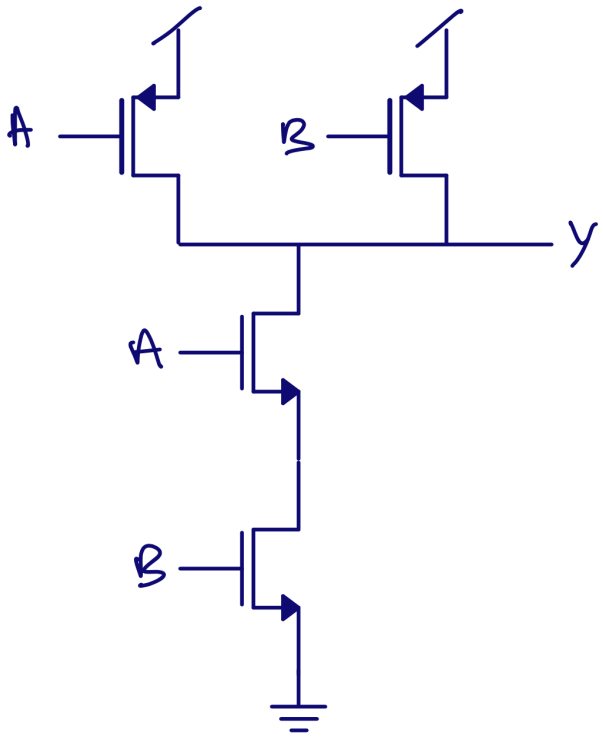
AND PU

$\Rightarrow$

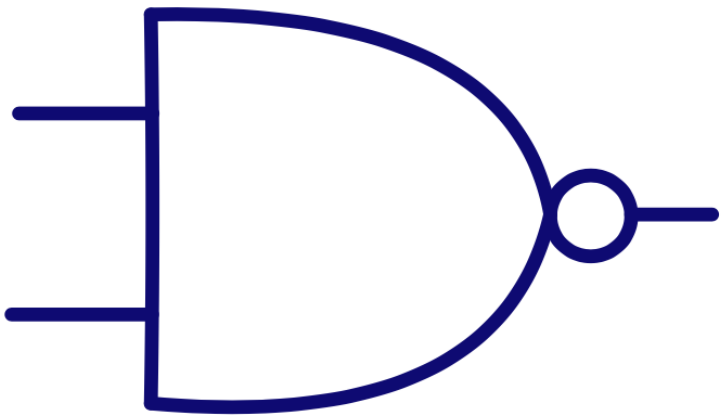
PMOS in paralell PD

$\Rightarrow$

NMOS in series



A	B	NOT(A AND B)
0	0	1
0	1	1
1	0	1
1	1	0



[.table-separator: #000000, stroke-width(1)] [.table: margin(8)]

$$Y = \overline{A + B} = \text{NOT (A OR B)}$$

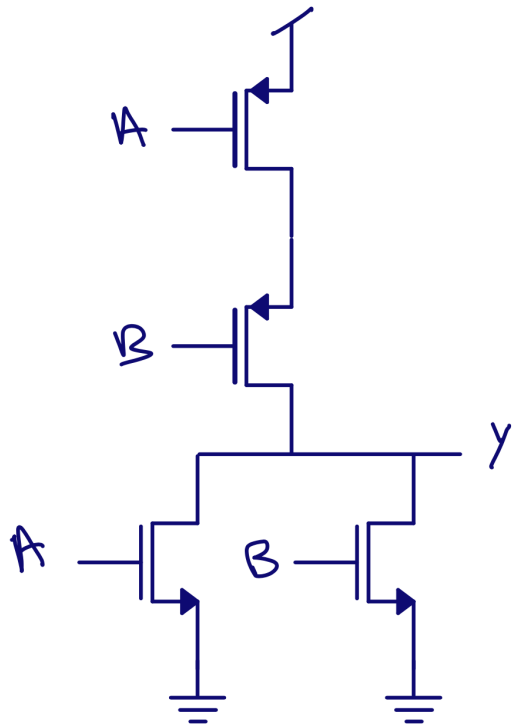
OR PU

$\Rightarrow$

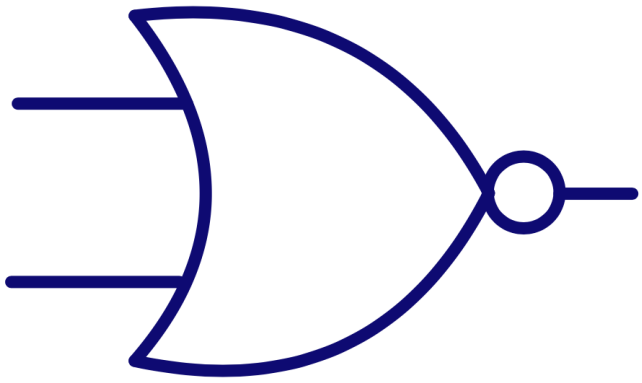
PMOS in series PD

$\Rightarrow$

NMOS in paralell



A	B	NOT(A OR B)
0	0	1
0	1	0
1	0	0
1	1	0



25.6 SR-Latch

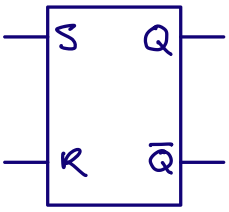
Use boolean expressions to figure out how gates work.

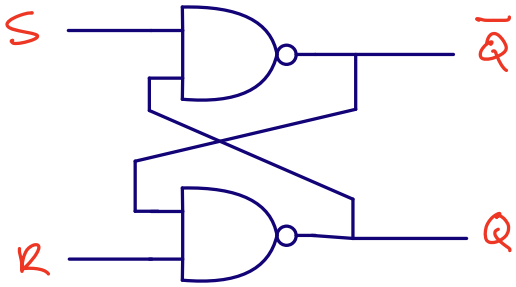
Remember De-Morgan

$$\overline{AB} = \overline{A} + \overline{B}$$
$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$Q = \overline{R\overline{Q}} = \overline{R} + \overline{\overline{Q}} = \overline{R} + Q$$

$$\overline{Q} = \overline{SQ} = \overline{S} + \overline{Q} = \overline{S} + \overline{Q}$$





$$Q = \overline{R} + Q$$

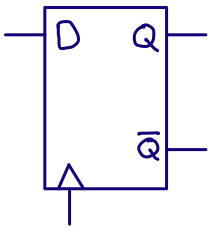
,

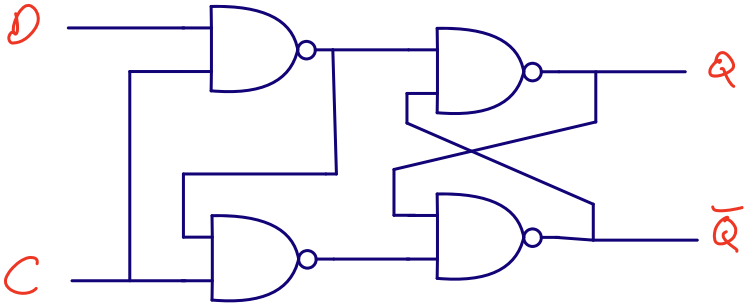
$$\overline{Q} = \overline{S} + \overline{Q}$$

S	R	Q	~Q
0	0	X	X
0	1	0	1
1	0	1	0
1	1	Q	~Q

25.7 D-Latch (16 transistors)

C	D	Q	~Q
0	X	Q	~Q
1	0	0	1
1	1	1	0





25.8 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**))

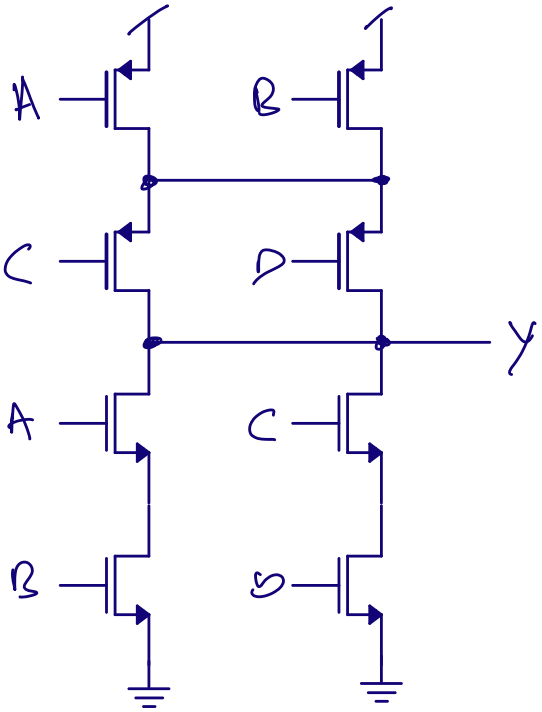
$$Y = A+B = \overline{\overline{A+B}}$$

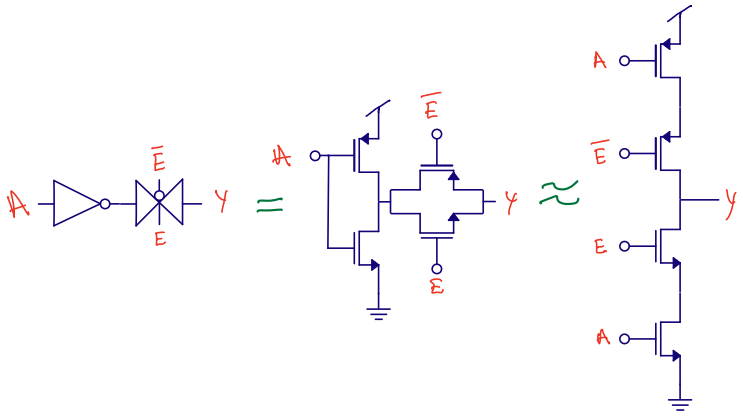
Y = **A** OR **B** = NOT(NOT(**A** OR **B**))

25.9 AOI22: and or invert

Y = NOT(**A** AND **B** OR **C** AND **D**)

$$Y = \overline{AB + CD}$$

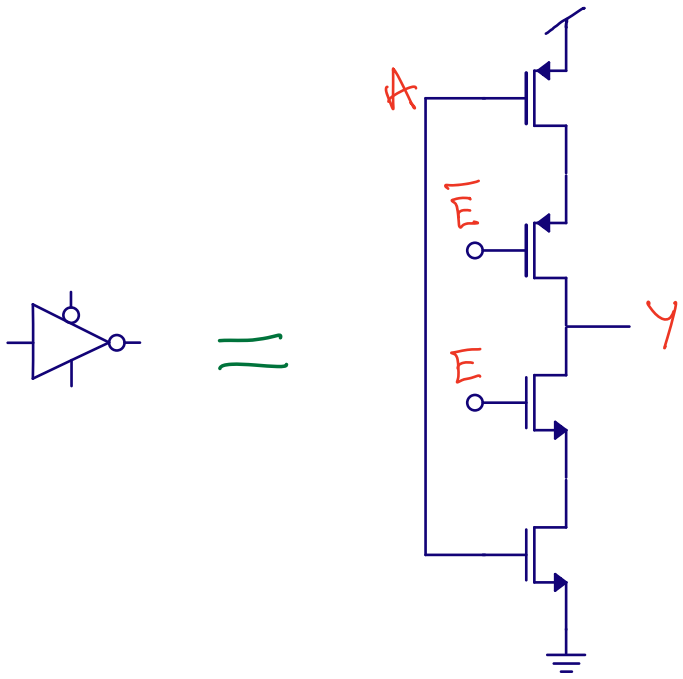




[.table-separator: #000000, stroke-width(1)] [.table: margin(8)]

25.10 Tristate inverter

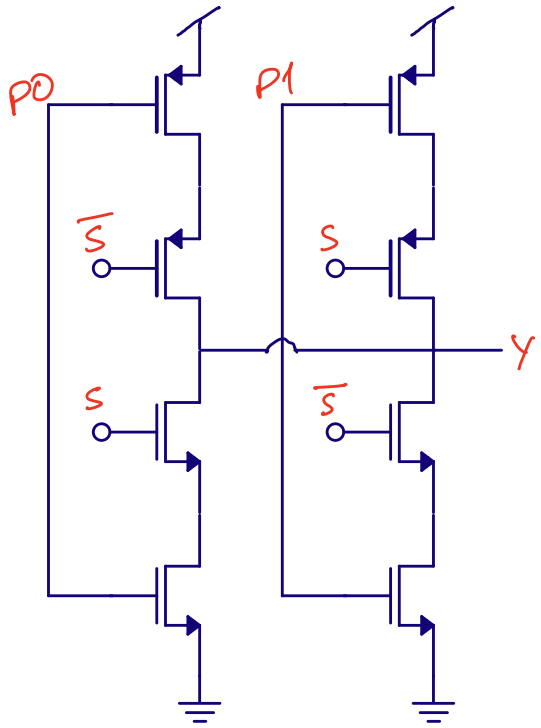
E	A	Y
0	0	Z
0	1	Z
1	0	1
1	1	0



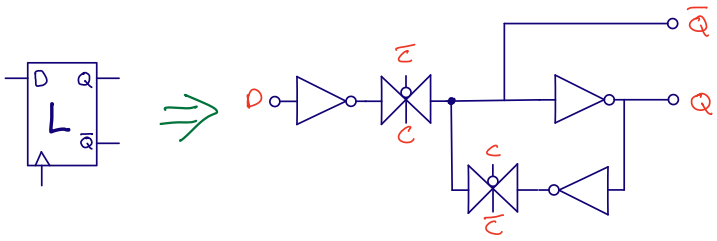
[.table-separator: #000000, stroke-width(1)] [.table: margin(8)]

25.11 Mux

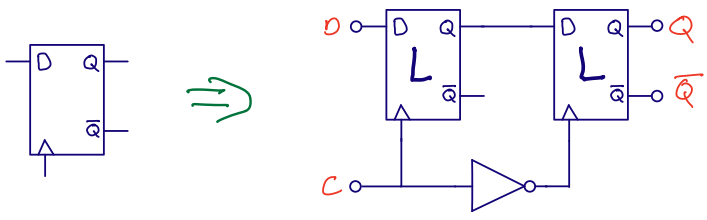
S	Y
0	NOT(P1)
0	NOT(P1)
1	NOT(P0)
1	NOT(P0)

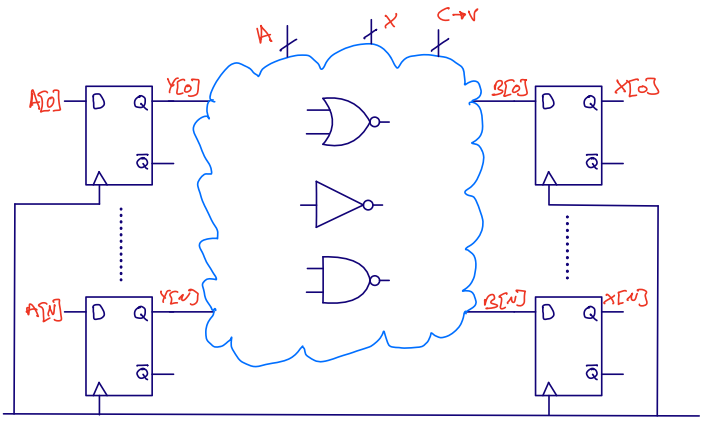


D-Latch (12 transistors)



D-Flip Flop (< 26 transistors)

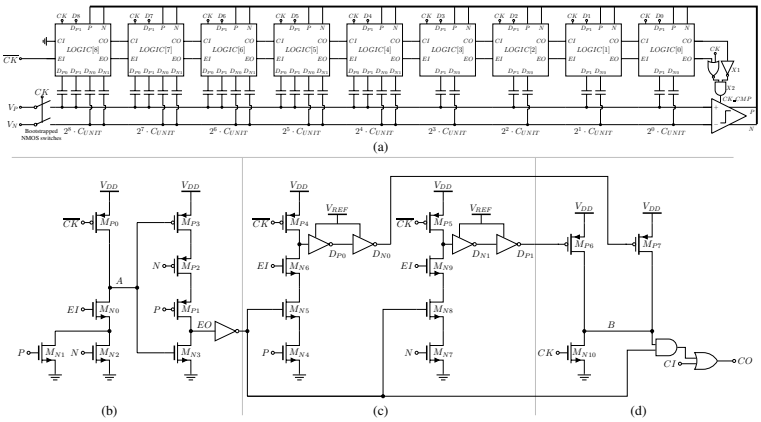




25.12 There are other types of logic

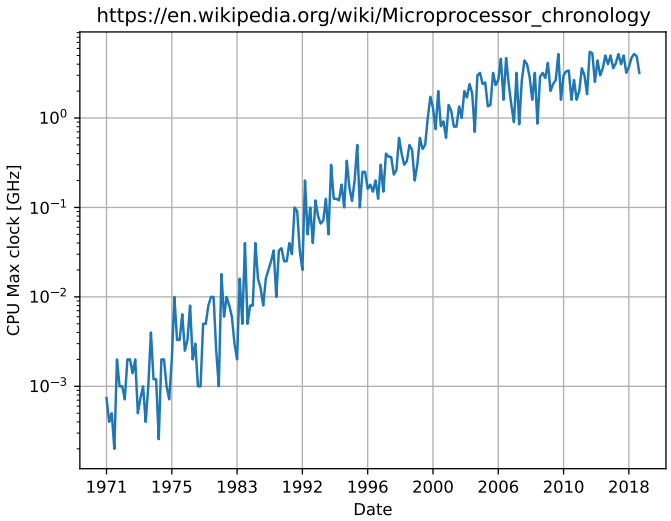
- True single phase clock (TSPC) logic
- Pass transistor logic
- Transmission gate logic
- Differential logic
- Dynamic logic

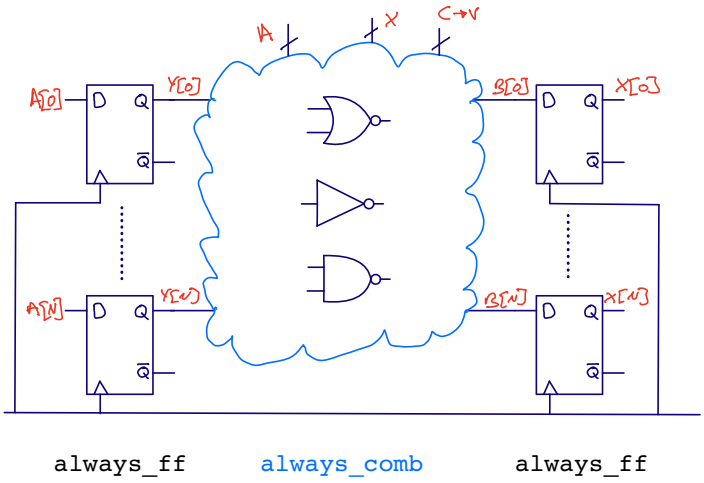
Consider other types of logic "rule breaking", so you should know why you need it.



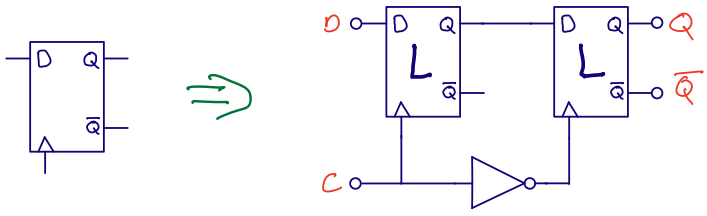
Dynamic logic => A Compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for Bluetooth Low Energy Receivers

25.13 Speed





25.14 Flip-flops and speed

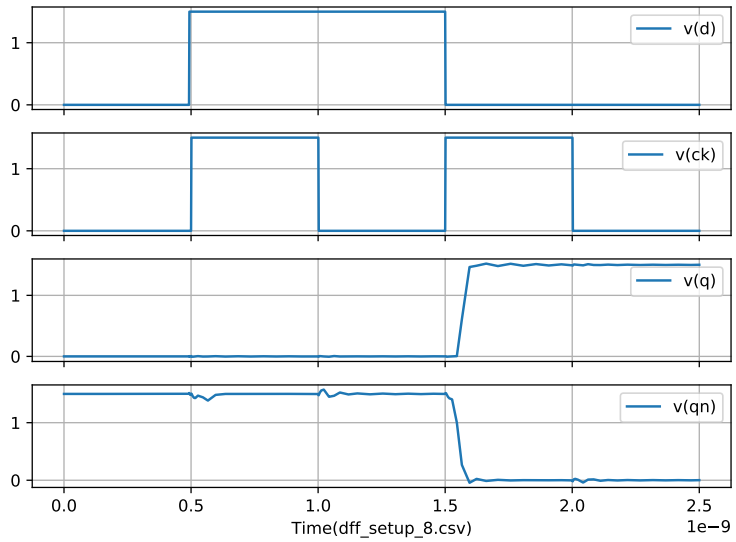


dicex/lib/SUN_TR_GF130N.spi:

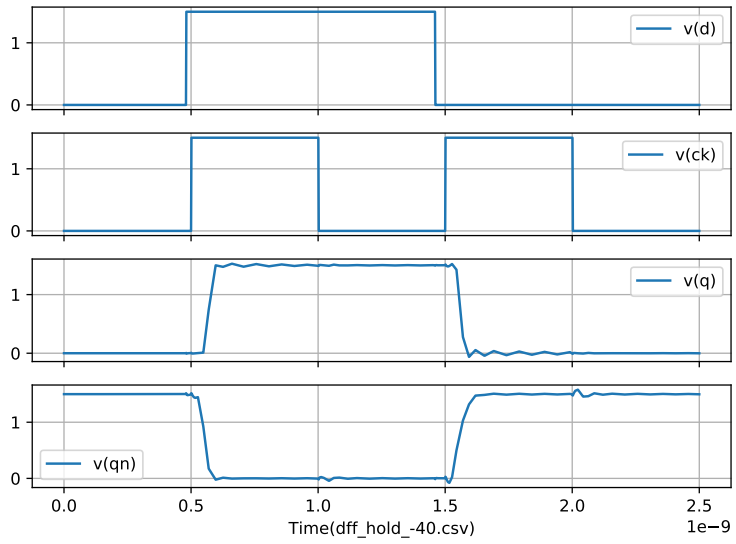
```
.SUBCKT DFRNQNX1_CV D CK RN Q QN AVDD AVSS
XA0 AVDD AVSS TAPCELLB_CV
XA1 CK RN CKN AVDD AVSS NDX1_CV
```

```
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



Hold time: How long after clk can the data change



25.15 Timing analysis

Analyze arrival times of all nodes in a combinatorial circuit

$$arrival_j = max_{j \in fanin(i)} arrival_j + t_{pd_i} \Rightarrow a_j = max_{j \in fanin(i)} a_j + t_{pd_i}$$

$$slack_i = required_i - arrival_j$$

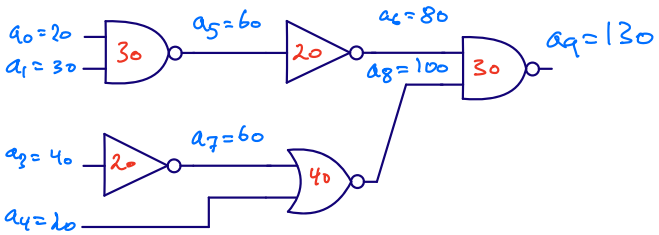
Positive slack (over PVT*)

$\Rightarrow$

Timing is OK Negative slack (over PVT†)

$\Rightarrow$

Timing is not OK



25.16 Timing analysis tools

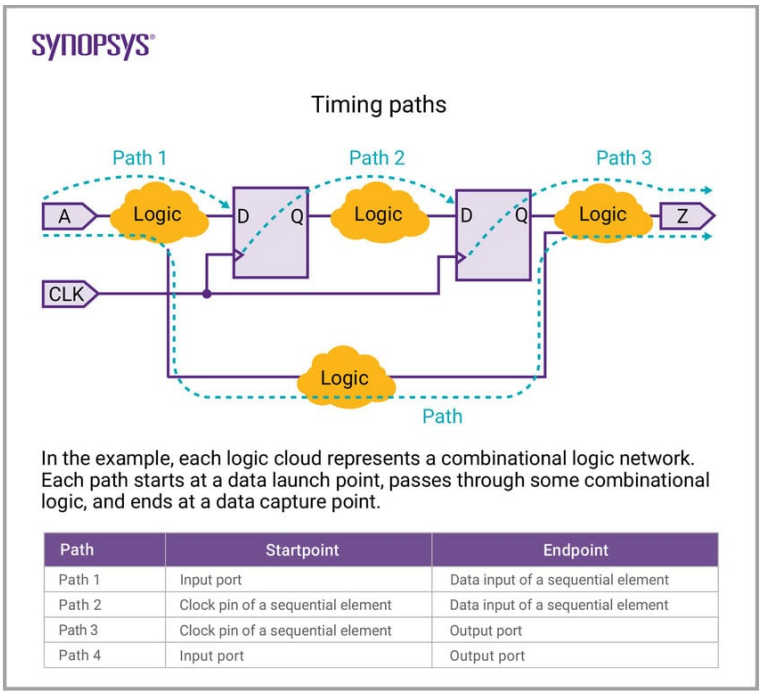
Commercial Cadence Tempus

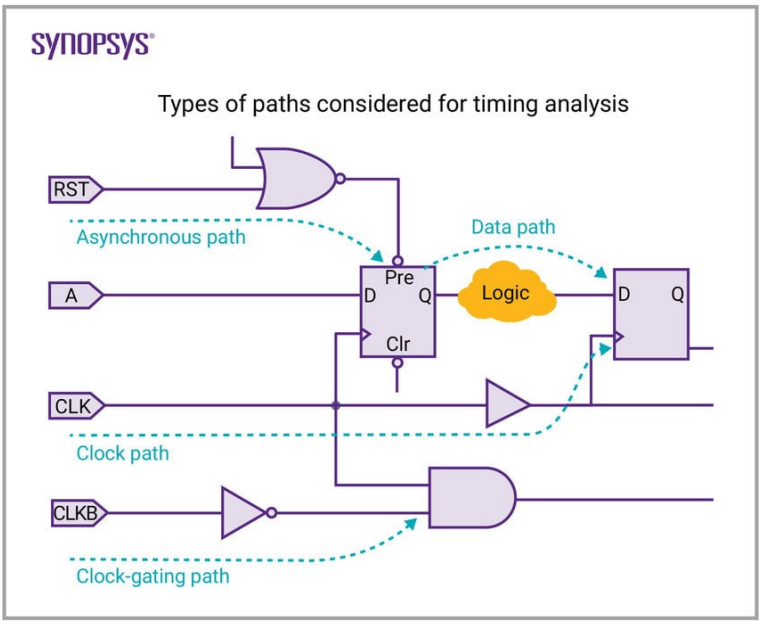
Synopsys PrimeTime

Free OpenTimer

* Often called *clock gating*
† Often called *clock gating*

25.16.0.1 What is timing analysis





25.16.0.2 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");
      }
    }
  }
}
```

25.17 Every gate must be simulated to provide behavior over input transition and load capacitance

25.18 All analog blocks must have associated liberty file to describe behavior and timing paths If you integrate analog into digital top flow

25.19 Gate Delay

25.19.0.1 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_pdf = (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

25.20 Delay estimation

How can we get a resonably accurate hand calculation model of delay?

$$C \approx 1 \text{ fF}/\mu\text{m}$$

$$R \approx 1\ \text{k}\Omega\mu\text{m}$$

25.20.0.1 Inverter with inverter load

$$C \approx 1\ \text{fF}/\mu\text{m}$$

,

$$R \approx 1\ \text{k}\Omega\mu\text{m}$$

$$t_{pd} = R \times 6C = 6RC$$

$$t_{pd} = 6 \times 1 \times 10^{3} \times 1 \times 10^{-15}\ \text{s}$$

$$t_{pd} = 6 \times 10^{-12} = 6\ \text{ps}$$

25.21 Elmore Delay

$$t_{pd} \approx \sum_{\text{nodes}} R_{\text{nodes}-to-source} C_i$$

$$= R_1C_1 + (R_1 + R_2)C_2 + \ldots + (R_1 + R_2 + \ldots + R_N)C_N$$

Good enough for hand calculation

25.22 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

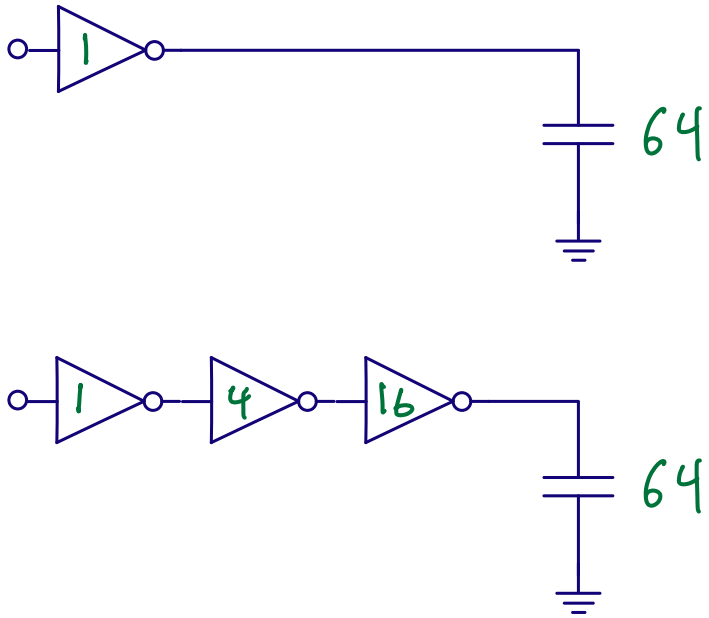
$$\Rightarrow d = 5\frac{2}{3} \times 3 \text{ ps} = 17 \text{ ps}$$

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	$d = f + p$	$D = \sum d_i = D_F + P$

25.23 Modern IC timing analysis requires computers with advanced programs‡

25.24 Best number of stages

25.25 Which has shortest delay?



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	$d = f + p$	$D = \sum d_i = D_F + P$

$$H = C_{cout}/C_{in} = 64$$

$$G = \prod g_i = \prod 1 = 1$$

‡ Often called power gating

$$B = 1$$

$$F = GBH = 64$$

One stage

$$f = 64 \Rightarrow D = 64 + 1 = 65$$

Three stage with

$$f = 4$$

$$D_F = 12, p = 3 \Rightarrow D = 12 + 3 = 15$$

For close to optimal delay, use

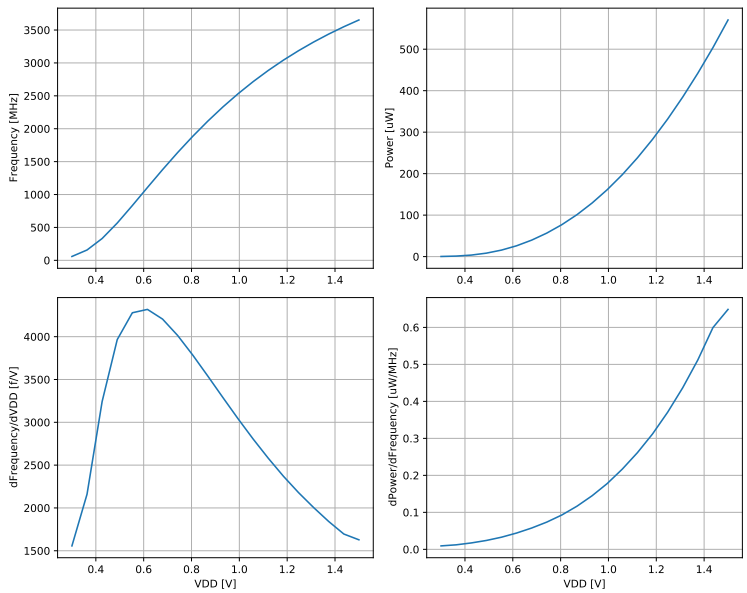
$$f = 4$$

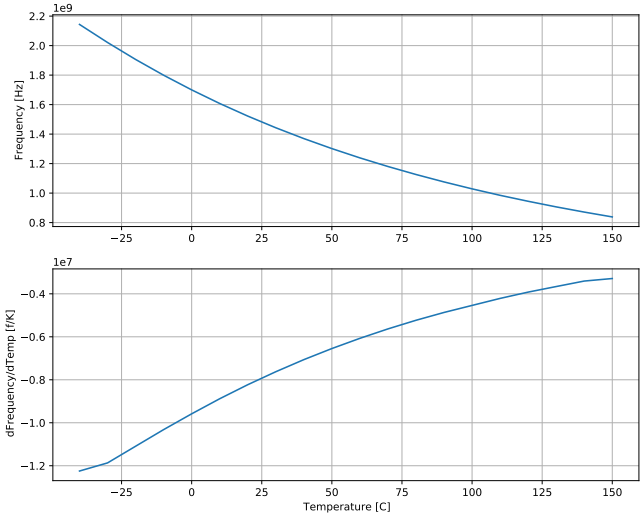
(Used to be

$$f = e$$

)

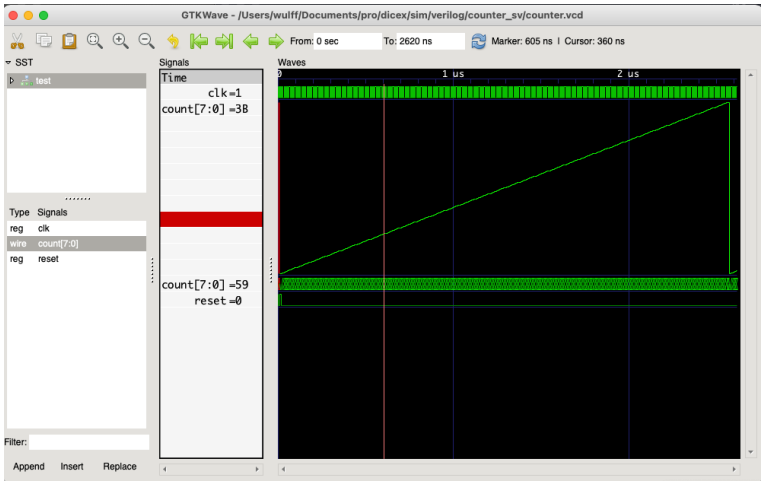
25.26 Trends

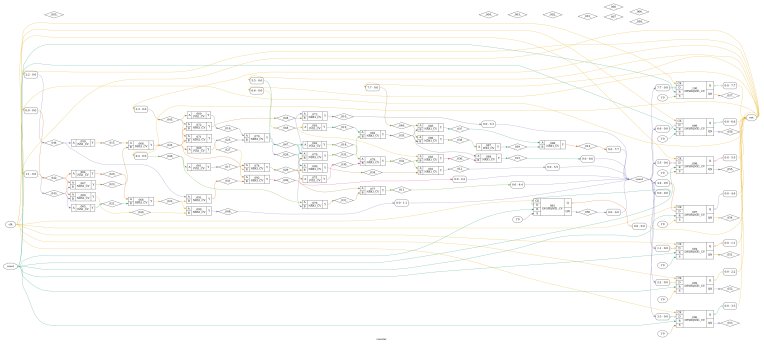




25.27 Attack vector

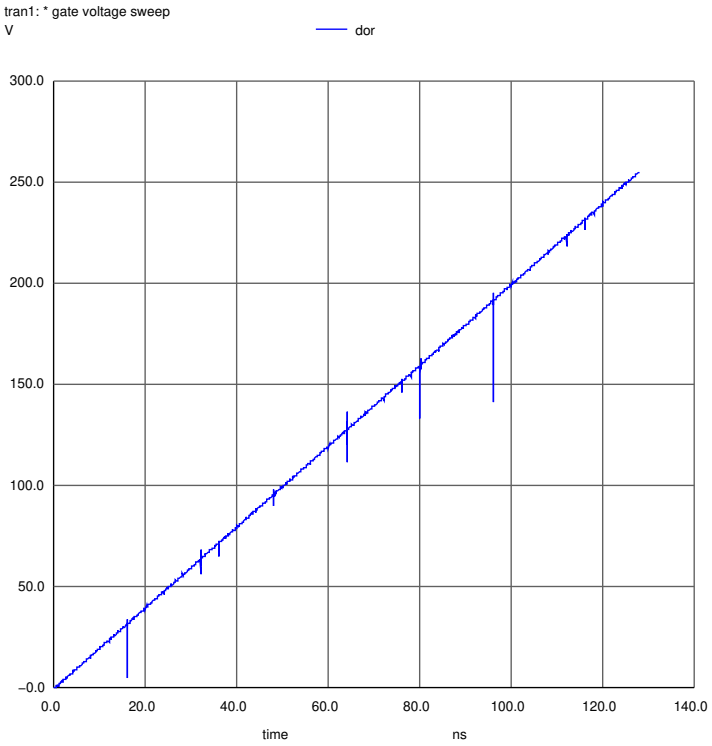
```
module counter(
               output logic [WIDTH-1:0] out,
               input logic              clk,
               input logic              reset
               );

   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
```





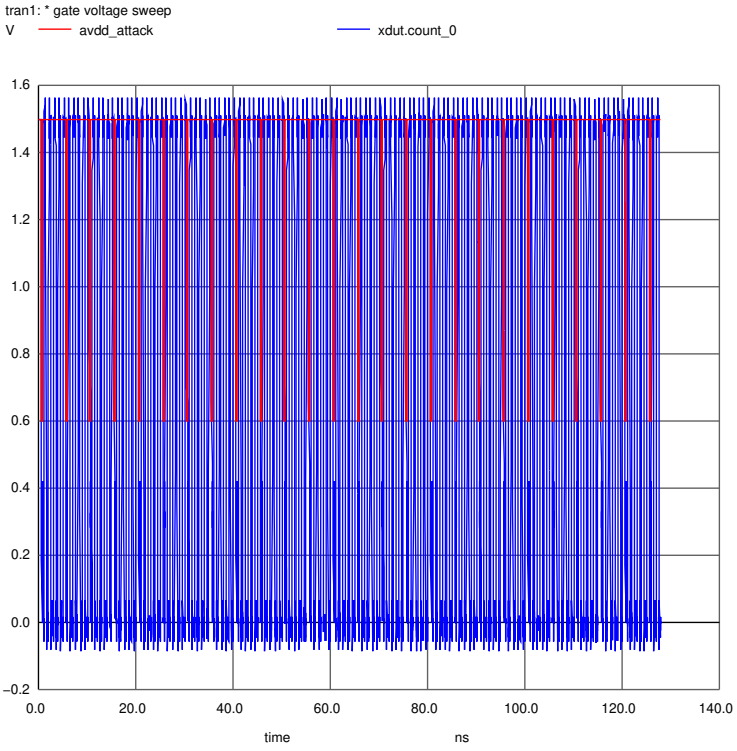
```
.SUBCKT counter out_7 out_6 out_5 out_4 out_3 out_2 out_1 out_0 clk reset AVDD AVSS
* SPICE netlist generated by Yosys 0_9 (git sha1 1979e0b1, gcc 10_3_0-1ubuntu1~20_10 -fPIC -Os
X0 out_2 1 AVDD AVSS IVX1_CV
X1 out_3 2 AVDD AVSS IVX1_CV
X2 out_4 3 AVDD AVSS IVX1_CV
X3 out_5 4 AVDD AVSS IVX1_CV
X4 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
```

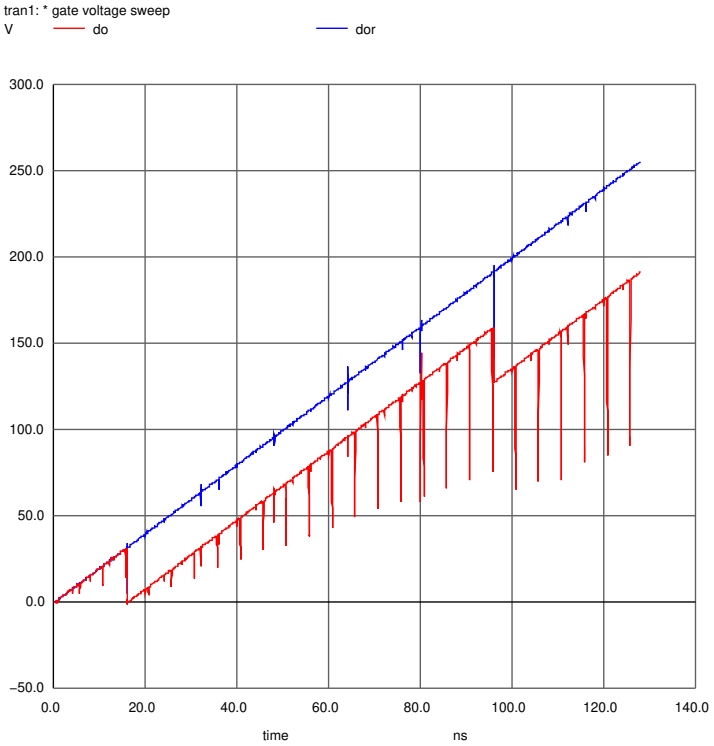
```
X38 19 clk AVSS reset out_4 39 AVDD AVSS DFSRQNX1_CV
X39 23 clk AVSS reset out_5 40 AVDD AVSS DFSRQNX1_CV
X40 26 clk AVSS reset out_6 41 AVDD AVSS DFSRQNX1_CV
X41 30 clk AVSS reset out_7 42 AVDD AVSS DFSRQNX1_CV
V0 count_0 35 DC 0
V1 43 out_2 DC 0
V2 44 out_3 DC 0
V3 count_3 15 DC 0
V4 45 out_4 DC 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
```



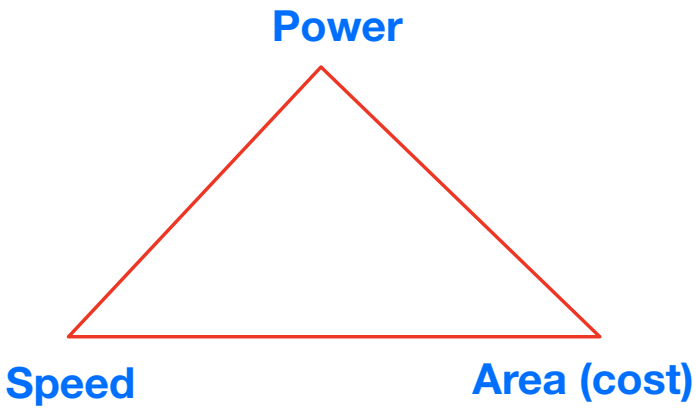
dicex/sim/verilog/counter_sv/counter_attack_tb.cir

```
VDDA AVDD_ATTACK 0 dc 0.5 pulse(1.5 0.6 tcd trf trf tapw taper)
```


tran1: * gate voltage sweep
V
avdd_attack
xdut.count_0
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
20.0
40.0
60.0
80.0
100.0
120.0
140.0
time
ns


tran1: * gate voltage sweep
V
do
dor
300.0
250.0
200.0
150.0
100.0
50.0
0.0
−50.0
0.0
20.0
40.0
60.0
80.0
100.0
120.0
140.0
time
ns

25.28 Pick two



25.29 Power

25.30 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]

25.31 Power dissipated in a resistor

Ohm's Law

$$V_R = I_R R$$

$$P_R = V_R I_R = I_R^2 R = \frac{V_R^2}{R}$$

25.32 Charging a capacitor to VDD

Capacitor differential equation

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

$$E_C = \int_0^\infty I_C V_C dt = \int_0^\infty C\frac{dV}{dt}V_C dt = \int_0^{V_C} CV dV = C\left[\frac{V^2}{2}\right]_0^{V_{DD}}$$

$$E_C = \frac{1}{2}CV_{DD}^2$$

25.33 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

25.34 Discharging a capacitor to 0

$$E_C = \frac{1}{2} C V_{DD}^2$$

Voltage is pulled to ground, and the power is dissipated in the NMOS

25.35 Power consumption of digital circuits

$$E_{VDD} = C V_{DD}^2$$

In a clock distribution network (chain of inverters), every output is charged once per clock cycle

$$P_{VDD} = C V_{DD}^2 f$$

25.36 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

$$P_{static} = \left(I_{sub} + I_{gate} + I_{pn}\right) V_{DD}$$

25.37 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

$$\alpha_i = \overline{P_i} P_i$$

25.38 Switching probability

Random data

$$P = 0.5$$

,

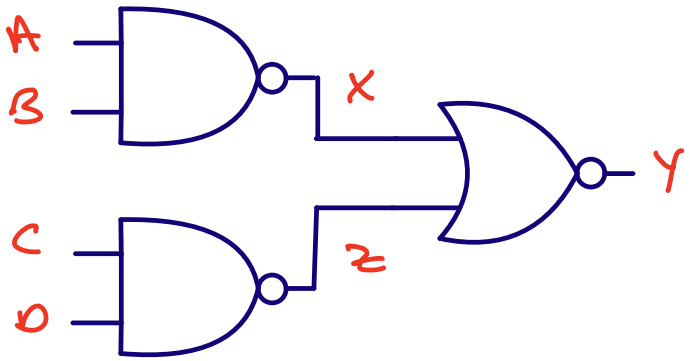
$$\alpha = 0.25$$

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	$P_A \bar{P_B} + \bar{P_A} P_B$

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	$P_A \bar{P}_B + \bar{P}_A P_B$



Assume

$$P = P_A = P_B = P_C = P_D = \frac{1}{2}$$

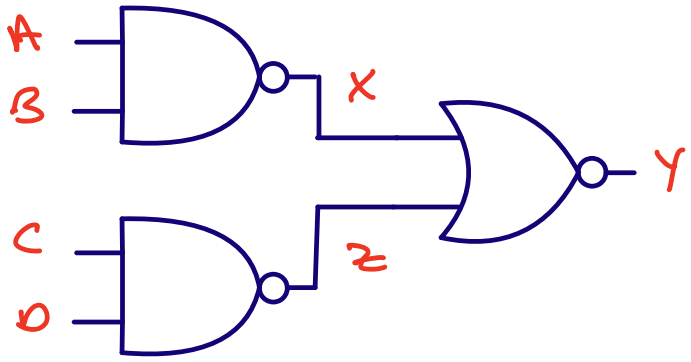
$$P_X = P_Z = 1 - PP = 1 - \frac{1}{4} = \frac{3}{4}$$

$$\overline{P_X} = \overline{P_Y} = \frac{1}{4}$$

$$P_Y = \frac{1}{4} \times \frac{1}{4} = \frac{1}{16}$$

$$\alpha = \frac{1}{16}\left(1 - \frac{1}{16}\right) = \frac{15}{16}\frac{1}{16} = \frac{15}{256}$$

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	$P_A \bar{P}_B + \bar{P}_A P_B$



$$\overline{\overline{AB} + \overline{CD}}$$

Use *De Morgan* first

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$\overline{\overline{AB} + \overline{CD}} = \overline{\overline{ABCD}} = ABCD$$

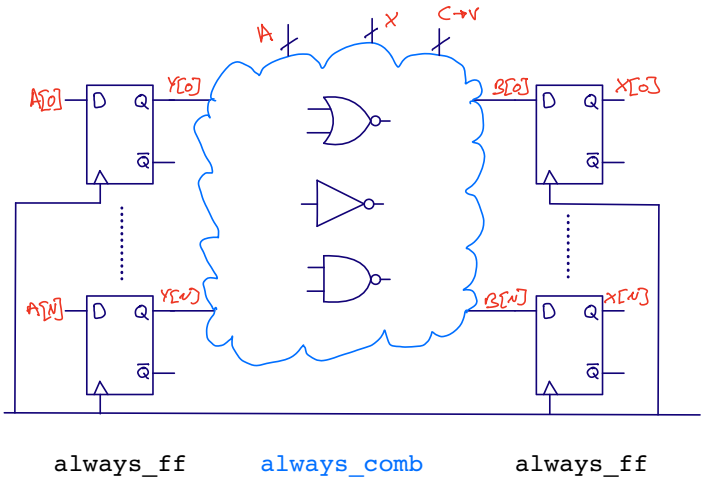
$$\Rightarrow P_Y = P_A P_B P_C P_D = \left(\frac{1}{2}\right)^4 = \frac{1}{16}$$

$$P_{tot} = \alpha C V_{DD}^2 f$$

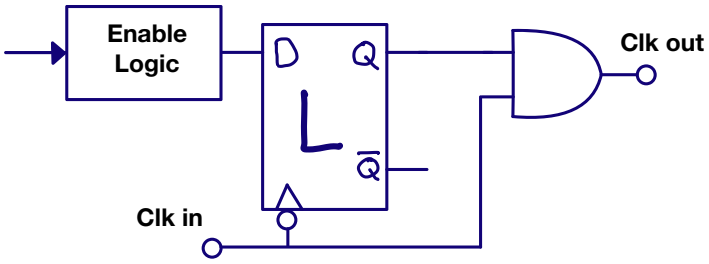
25.39 Strategies to reduce dynamic power

1. Stop clock
2. Stop activity
3. Reduce clock frequency

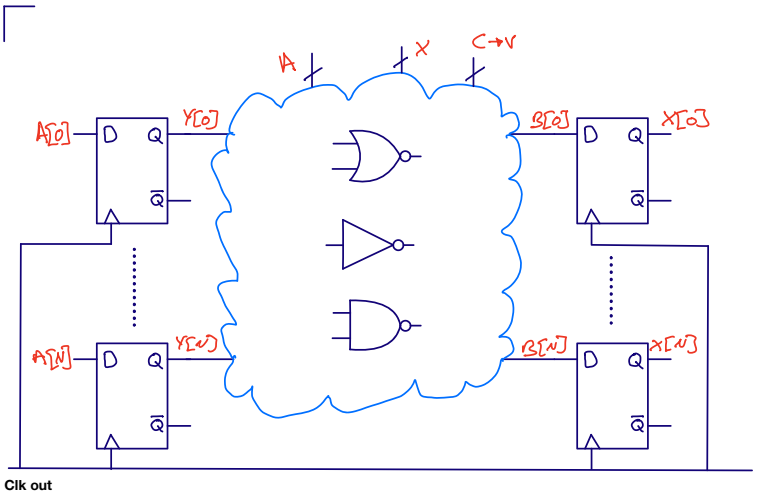
4. Turn off VDD
5. Reduce VDD



25.39.1 Stop clock §

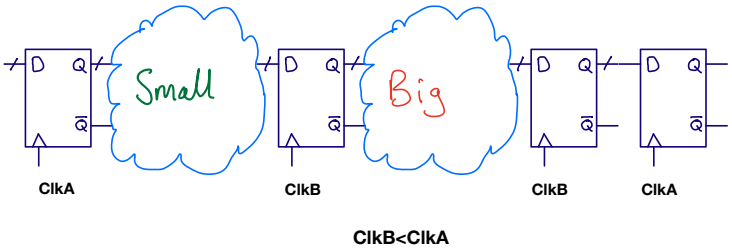


25.39.2 Stop activity

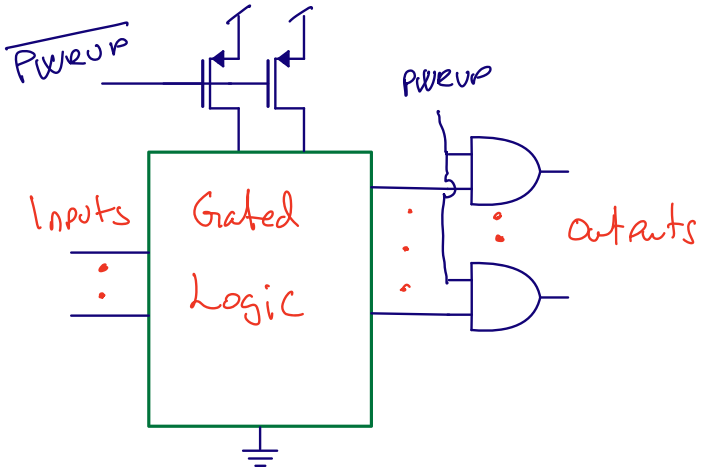


§ Often called *clock gating*

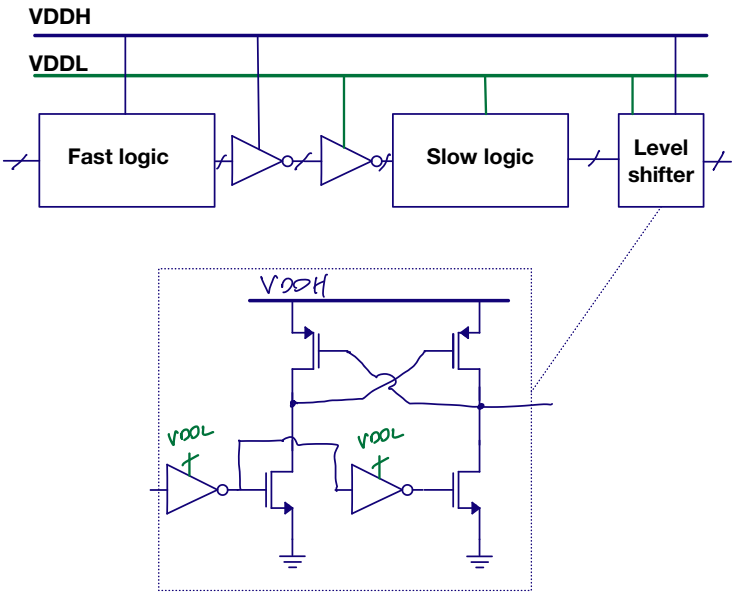
25.39.3 Reduce frequency



25.39.4 Turn off power supply ¶



25.39.4.1 Reduce power supply



¶ Often called power gating

25.39.4.2 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

$$V_{DD-opt} = \frac{3}{3 - 1\text{ to }2}V_t \in [1.5, 3]V_t$$

25.40 Wires

25.41 Wire geometry

Pitch = w + s

Aspect ratio (AR) = t/w

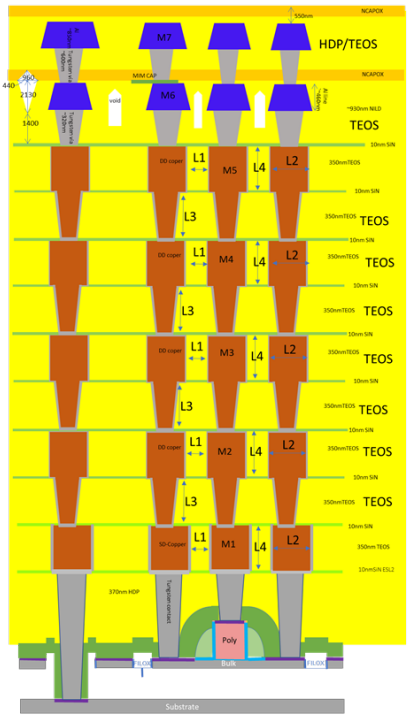
These days

$$AR \approx 2$$

25.42 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 tick	Can tolerate high forces during wire bonding.



2 layers of Aluminum Interconnect

5 layers of Copper Interconnect

25.43 Metal routing rules on IC

Odd numbers metals

$$\Rightarrow$$

Horizontal routing (as far as possible)

Even numbers metals

$$\Rightarrow$$

Vertical routing (as far as possible)

25.44 Modeling Interconnect

Resistance narrow size impedes flow

Capacitance through under the leaky pipes

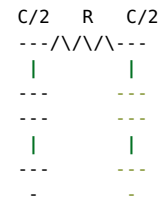
Inductance paddle wheel intertia opposes changes in flow rate

25.45 Lumped model

Use 1-segment

$$\pi$$

-model for Elmore delay



25.46 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

$$R = R_{\square} \times \#\text{ of squares}$$

25.47 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 (μΩ • cm)
Silver (Ag)	1.6
Copper (Cu)	1.7
Gold (Au)	2.2
Aluminum (Al)	2.8
Tungsten (W)	5.3
Titanium (Ti)	43.0

25.48 Contacts

Contacts and vias can have 2-20

$$\Omega$$

Must use many contacts/vias for high current wires

25.49 Wire Capacitance

Dense wires has about

$$0.2 \text{ fF}/\mu\text{m}$$

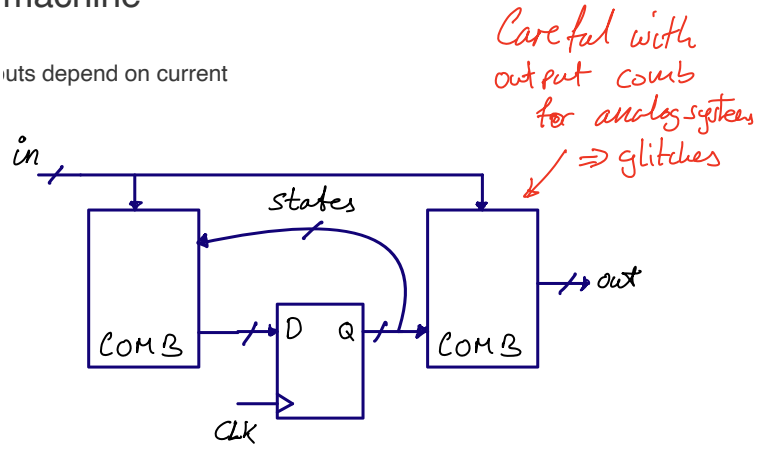
25.50 FSM

25.51 Mealy machine

An FSM where outputs depend on current state and inputs

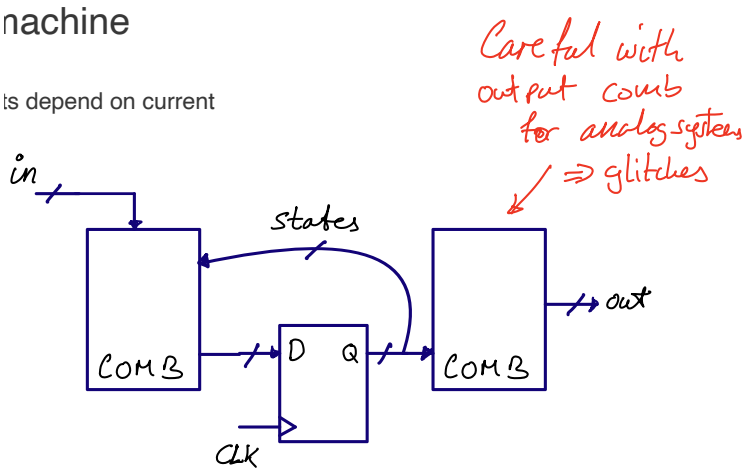
machine

uts depend on current



25.52 Moore machine

An FSM where outputs depend on current state



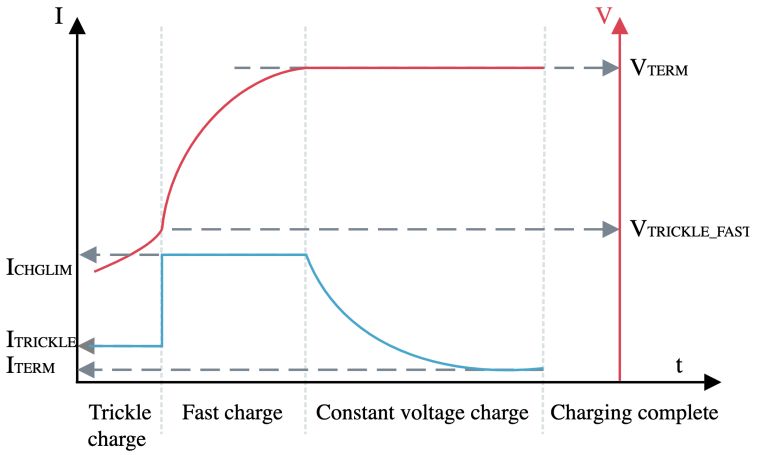
25.53 Mealy versus Moore

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
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

25.53.1 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
```

25.54 Battery charger FSM



25.54.1 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

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

25.54.2 Battery charger - Inputs

Voltage above

$$V_{TRICKLE}$$

Voltage close to

$$V_{TERM}$$

If voltage close to

$$V_{TERM}$$

and current is close to

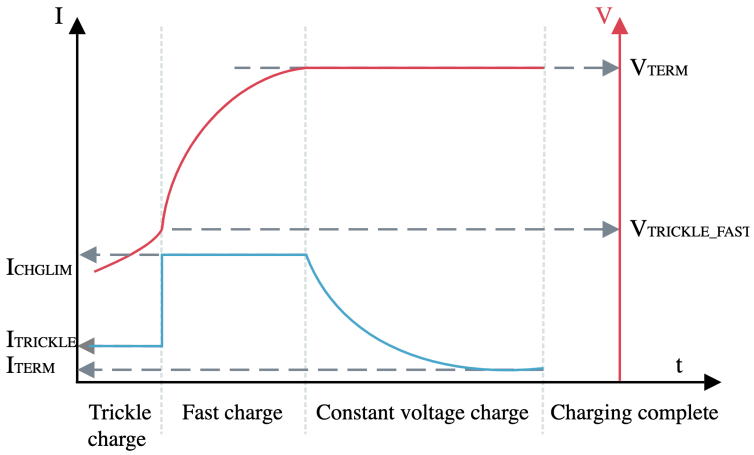
$$I_{TERM}$$

, then charging complete

If charging complete, and voltage has dropped (

$$V_{RECHARGE}$$

), then start again



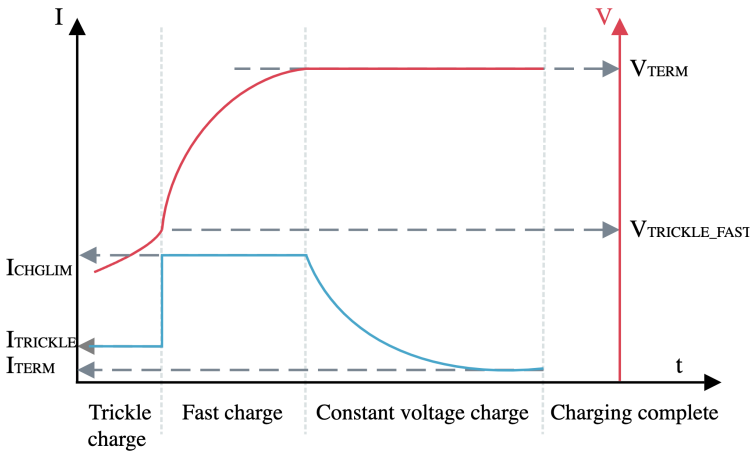
25.54.3 Battery charger - States

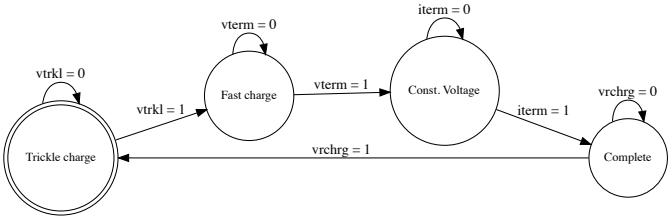
Trickle charge (0.1 C)

Fast charge (1 C)

Constant voltage

Charging complete

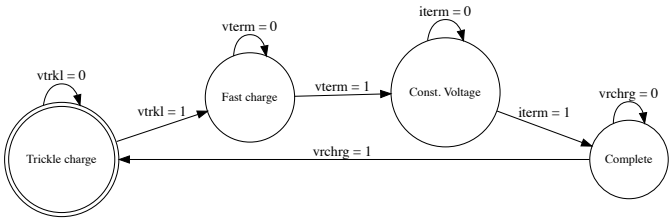




25.54.3.1 One way to draw FSMs - Graphviz

```
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
```



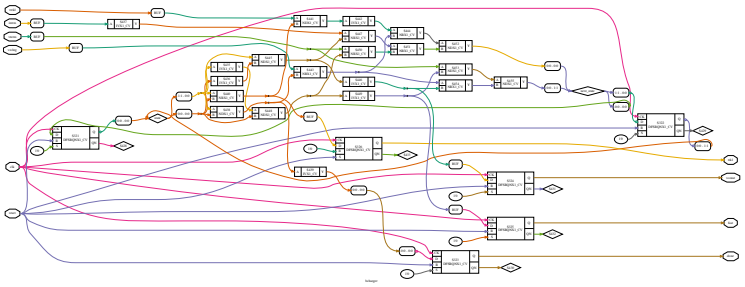

```
module bcharger( output logic trkl,
       output logic fast,
       output logic vconst,
       output logic done,
       input logic  vtrkl,
       input logic  vterm,
       input logic  iterm,
       input logic  vrchrg,
       input logic  clk,
       input logic  reset
                  );

   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
```

25.54.3.2 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
```



Mixed Signal Simulation in NGSPICE 26

26.1 Mixed Signal Simulation in ngspice

Status: 0.3

26.2 Digital simulation

- The order of execution of events at the same time-step do not matter
- The system is causal. Changes in the future do not affect signals in the past or the now

There are both commercial an 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_comb section we code what will become the combinatorial logic. In the always_ff section we code what will become our registers.

```
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
```

26.3 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

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

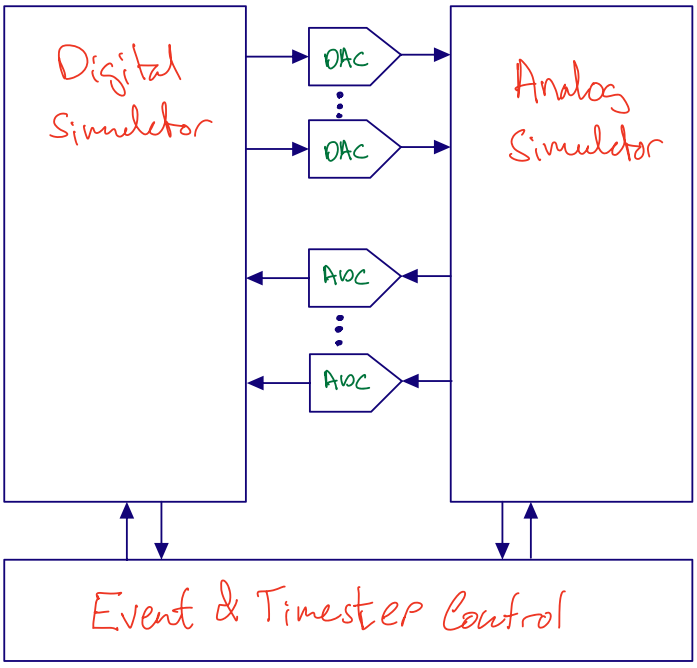
The original paper has spawned a multitude of commercial, free and open source simulators, some are listed below.

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



26.4 Demo

Tutorial at http://analogicus.com/jnw_sv_sky130a/

Repository at https://github.com/wulffern/jnw_sv_sky130a

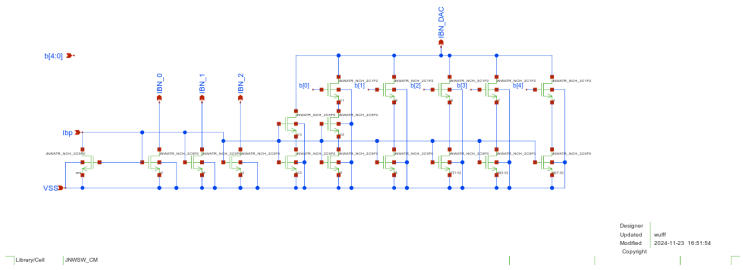
Assumes knowledge of Tutorial

26.5 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.

Current Mirror and Current DAC



26.6 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
```

26.7 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
```

26.8 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 `svninst.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)
```

26.9 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

...
```

26.10 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 =te 0.0 out_high = 1.8)"
     +   "auto_bridge%d [ %s ] [ %s ] auto_dac" )
```

26.11 Running

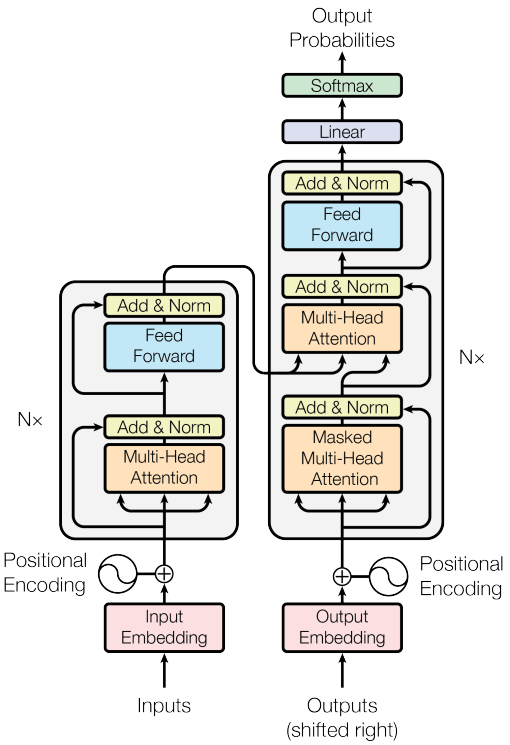
You can run the whole thing with

```
cd sim/JNWSW_CM/
make typical
```

Analog Neural Networks and Translinear Circuits

27

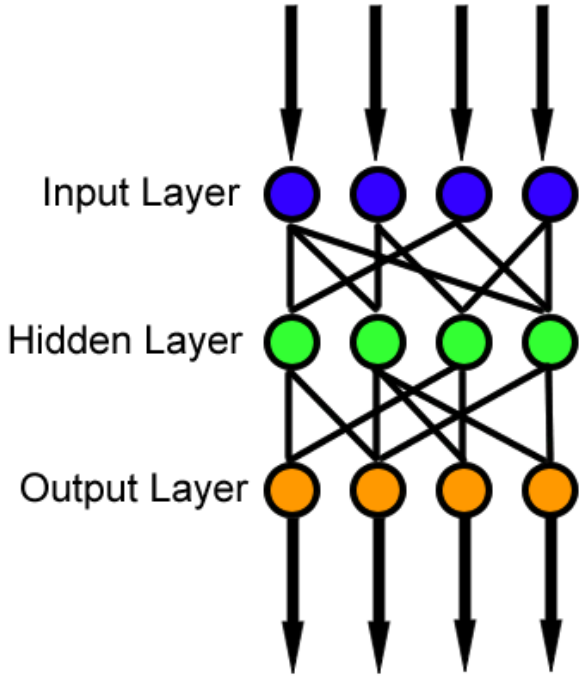
Attention Is All You Need



Neural Nets 3blue1brown

$$a_{l+1} = \sigma(W_l a_l + b_l)$$

A NN consists of addition, multiplication, and a non-linear function



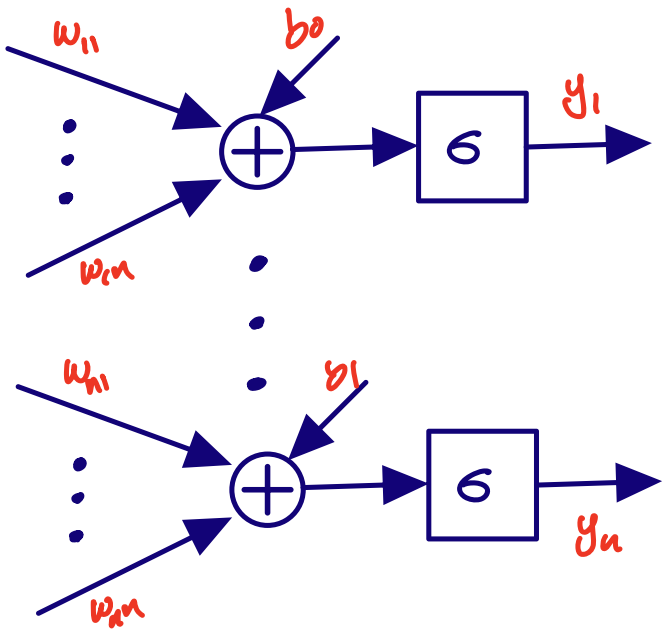
$$\mathbf{y} = \sigma\left(\begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}\right)$$

$$\mathrm{OA}_{(x,y,k)} = f\left(\sum_{i=0}^{R-1}\sum_{j=0}^{S-1}\sum_{c=0}^{C-1} \mathrm{IA}_{(x+i,y+j,c)} \times W_{(i,j,c,k)}\right)$$

Assume N neurons

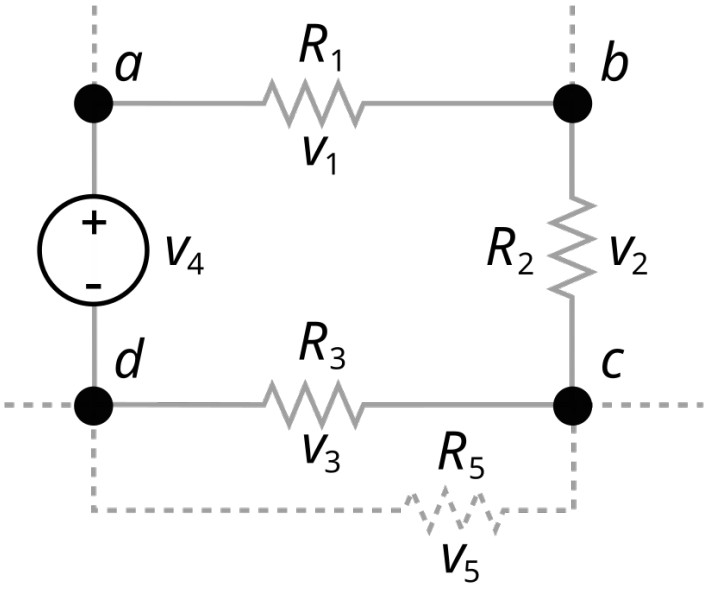
- N multiplications per neuron
- N + 1 additions per neuron
- 1 sigmoid per neuron

For efficient inference, additions and multiplications should be low power!

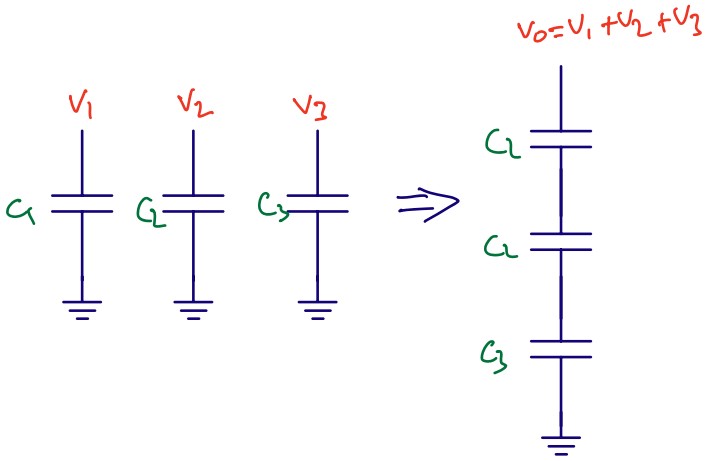


27.0.1 Kirchoff's voltage law

The directed sum of the potential differences around any closed loop is zero

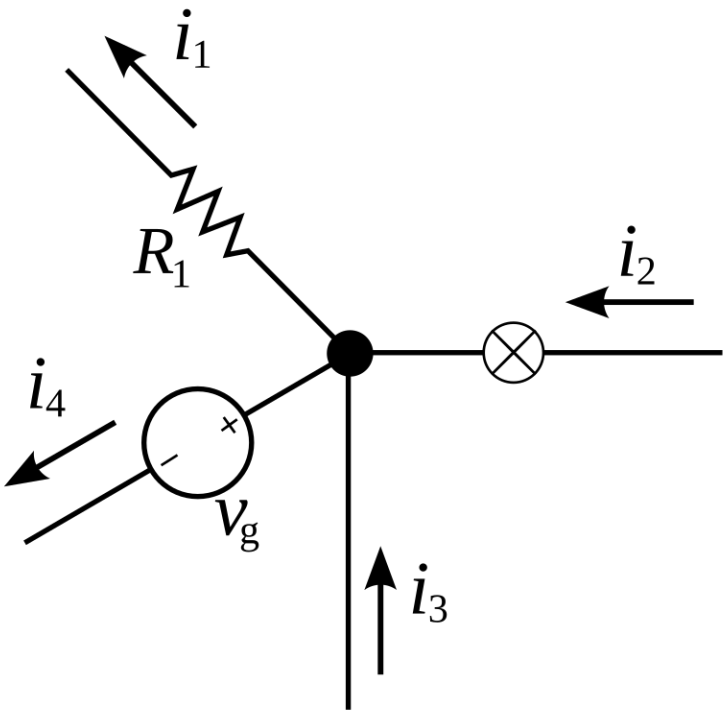


$$V_1 + V_2 + V_3 + V_4 = 0$$

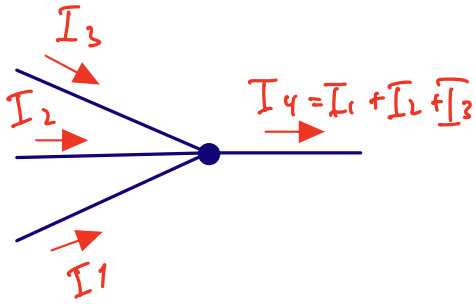


27.0.2 Kirchoff's current law

The algebraic sum of currents in a network of conductors meeting at a point is zero

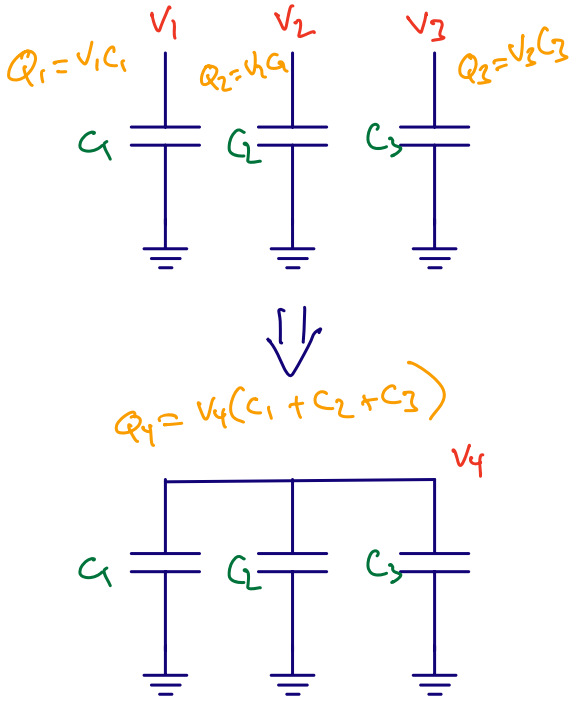


$$i_1 + i_2 + i_3 + i_4 = 0$$



27.0.3 Charge concervation

See Charge concervation on Wikipedia



$$Q_4 = Q_1 + Q_2 + Q_3$$

$$V_4 = \frac{C_1V_1 + C_2V_2 + C_3V_3}{C_1 + C_2 + C_3}$$

27.1 Multiplication

27.1.1 Digital capacitance

$$V_4 = \frac{C_1V_1 + C_2V_2 + C_3V_3}{C_1 + C_2 + C_3}$$

$$V_O = \frac{C_1}{C_{TOT}}V_1 + \cdots + \frac{C_N}{C_{TOT}}V_N$$

Make capacitors digitally controlled, then

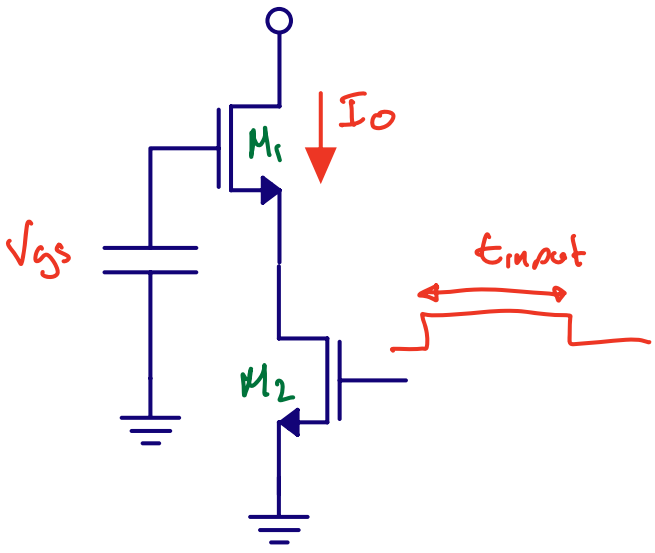
$$w_1 = \frac{C_1}{C_{TOT}}$$

Might have a slight problem with variable gain as a function of total capacitance

27.1.2 Mixing

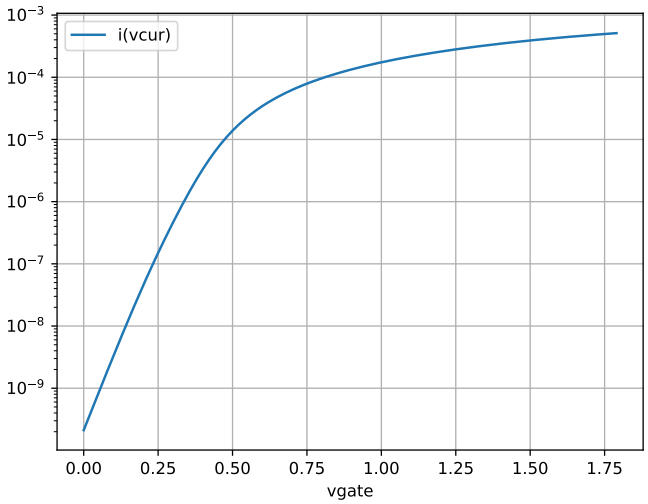
$$I_{M1} = G_m V_{GS}$$

$$I_o = I_{M1} t_{input}$$



27.1.3 Translinear principle

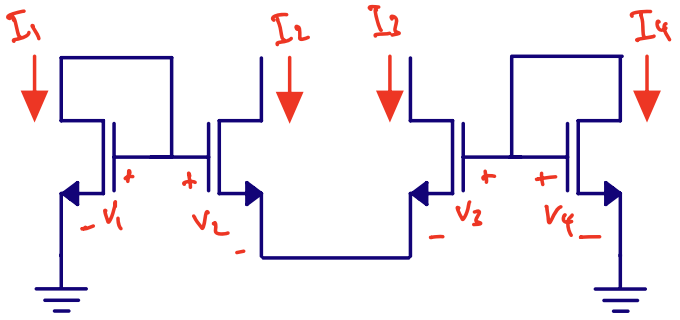
27.1.3.1 MOSFET in sub-threshold



$$I = I_{D0}\frac{W}{L}e^{(V_{GS}-V_{th})/nU_T} \text{ ,} U_T = \frac{kT}{q}$$

$$I = \ell e^{V_{GS}/nU_T} \text{ , } \ell = I_{D0}\frac{W}{L}e^{-V_{th}/nU_T}$$

$$V_{GS} = nU_T \ln\left(\frac{I}{\ell}\right)$$



$$V_1 + V_2 = V_3 + V_4$$

$$nU_T\left[\ln\left(\frac{I_1}{\ell_1}\right) + \ln\left(\frac{I_2}{\ell_2}\right)\right] = nU_T\left[\ln\left(\frac{I_3}{\ell_3}\right) + \ln\left(\frac{I_4}{\ell_4}\right)\right]$$

$$\ln\left(\frac{I_1 I_2}{\ell_1 \ell_2}\right) = \ln\left(\frac{I_3 I_4}{\ell_3 \ell_4}\right)$$

$$\frac{I_1 I_2}{\ell_1 \ell_2} = \frac{I_3 I_4}{\ell_3 \ell_4}$$

$$I_1 I_2 = I_3 I_4 \text{ , if } \ell_1 \ell_2 = \ell_3 \ell_4$$

$$I_1 I_2 = I_3 I_4$$

$$I_1 = I_a \text{ ,} I_2 = I_b + i_b \text{ ,} I_3 = I_b \text{ ,} I_4 = I_a + i_a$$

$$I_a(I_b + i_b) = I_b(I_a + i_a)$$

$$I_a I_b + I_a i_b = I_b I_a + I_b i_a$$

$$i_b = \frac{I_b}{I_a} i_a$$

$$\ell_1 \ell_2 = \ell_3 \ell_4$$

$$\ell_1 = I_{D0} \frac{W}{L} e^{-V_{th}/nU_T}$$

$$\ell_2 = I_{D0} \frac{W}{L} e^{-(V_{th} \pm \sigma_{th})/nU_T} = \ell_1 e^{\pm \sigma_{th}/nU_T}$$

$$\sigma_{th} = \frac{a_{vt}}{\sqrt{WL}}$$

$$\frac{\ell_2}{\ell_1} = e^{\pm \frac{a_{vt}}{\sqrt{WL}}/nU_T}$$

27.1.3.2 Demo

JNW_SV_SKY130A

27.2 Want to learn more?

An Always-On 3.8 u J/86 % CIFAR-10 Mixed-Signal Binary CNN Processor With All Memory on Chip in 28-nm CMOS

CAP-RAM: A Charge-Domain In-Memory Computing 6T-SRAM for Accurate and Precision-Programmable CNN Inference

ARCHON: A 332.7TOPS/W 5b Variation-Tolerant Analog CNN Processor Featuring Analog Neuronal Computation Unit and Analog Memory

IMPACT: A 1-to-4b 813-TOPS/W 22-nm FD-SOI Compute-in-Memory CNN Accelerator Featuring a 4.2-POPS/W 146-TOPS/mm2 CIM-SRAM With Multi-Bit Analog Batch-Normalization

Bibliography

D. Johns and K. Martin, *Analog Integrated Circuit Design*. John Wiley & Sons, Inc., 1997.

A. V. D. Ziel, *Noise in Solid State Devices and Circuits*. John Wiley & Sons Inc, 1986.

B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw-Hill, 2001.

R. M. Gray and L. D. Davisson, *An Introduction to Statistical Signal Processing*. Cambridge University Press, 2004, no. ISBN-0521838606.

A. Einstein, "Method for the determinination of the statistical values of observations concerning quantities subject to irregular fluctuations," vol. October, 1987.