

CMOS Logic

Carsten Wulff, carsten@wulff.no

I. CMOS LOGIC

Status: 0.3

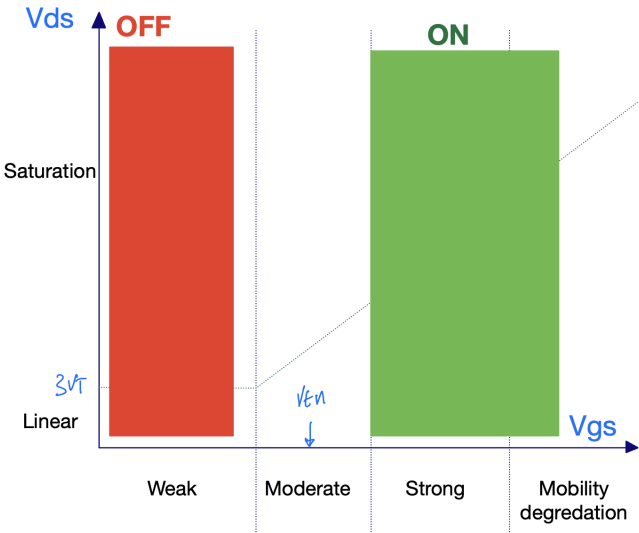
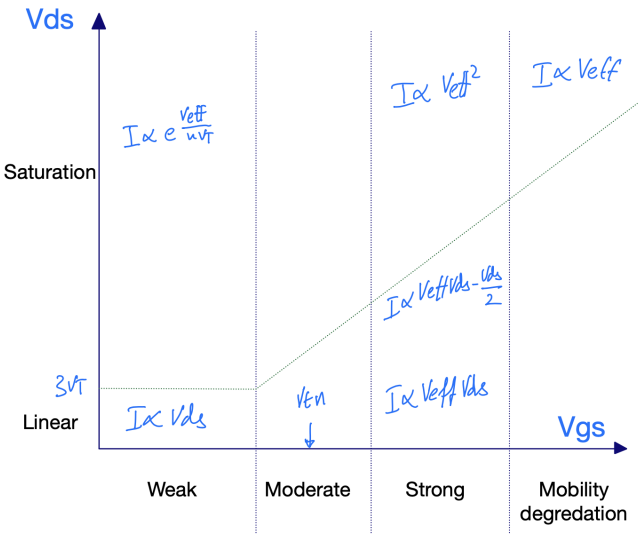
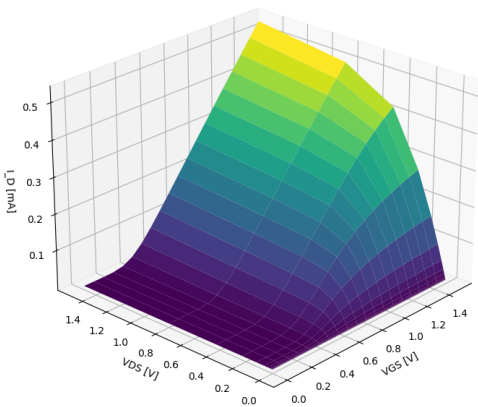
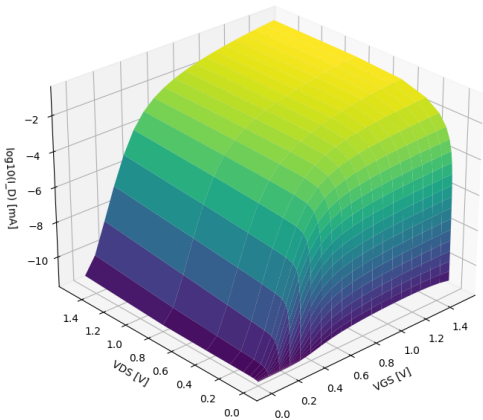
II. ANALOG TRANSISTOR TO DIGITAL TRANSISTOR
NMOS current ($W = 0.4\mu$ $L=0.15\mu$) as a function of

$$V_{GS}$$

and

$$V_{DS}$$

dicex/lectures/l13/mos.py



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

III. CMOS STATIC LOGIC ASSUMPTIONS

NMOS source is connected to low potential

$$V_{GS} > V_{TH}$$

when

IV. DON'T BREAK RULES UNLESS YOU KNOW EXACTLY WHY IT WILL BE OK

$$V_G = V_{DD}$$

V. LOGIC CELLS

PMOS source is connected to high potential

$$V_{GS} < V_{TH}$$

when

$$V_G = 0$$

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

$$\overline{AB} = \overline{A+B} \quad \leftarrow \text{DM}$$

$$\overline{A+B} = \overline{A} \overline{B} \quad \leftarrow \text{DM}$$

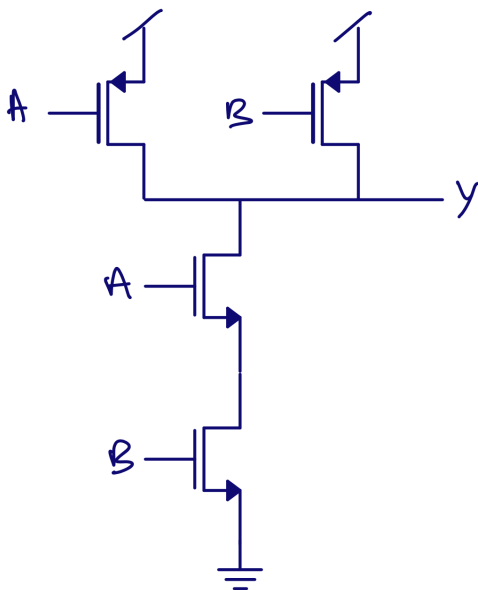
$$AB = \overline{\overline{A+B}}$$

$$A+B = \overline{\overline{A} \overline{B}}$$

$\overline{A}$	$\overline{B}$	$\overline{A+B}$	$\overline{AB}$	$\overline{\overline{A+B}}$	$\overline{\overline{AB}}$
1	1	1	1	0	0
1	0	1	0	0	1
0	1	1	0	0	1
0	0	0	0	1	1

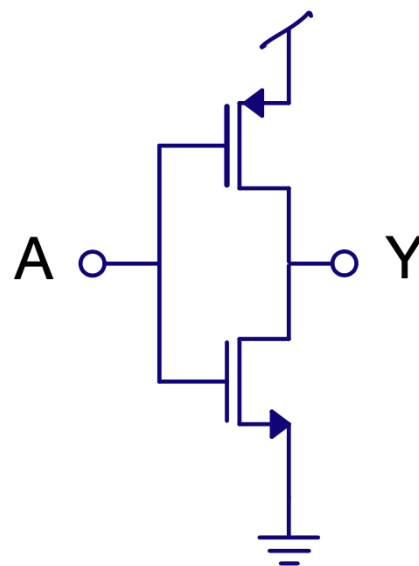
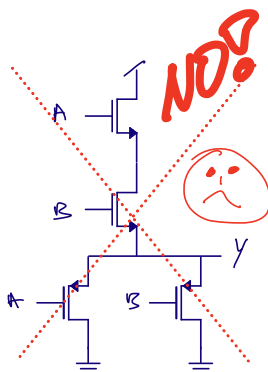
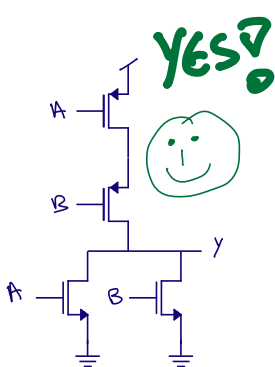
$$\overline{AB} \neq \overline{\overline{AB}}$$

$$\overline{A+B} \neq \overline{\overline{A+B}}$$



A. CMOS static logic is inverting

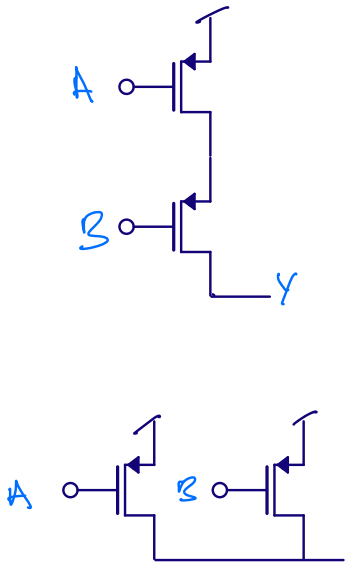
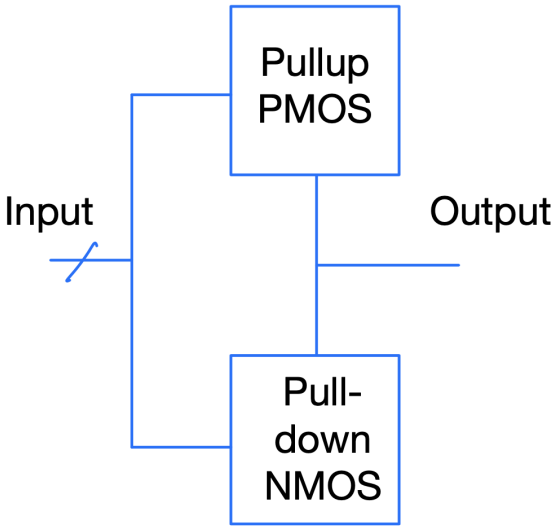
A	Y
1	0
0	1



PD \ PU	OFF	ON
	OFF	ON
OFF	Z	1
ON	0	X

PD = Pull-down PU = Pull-up

```
logic => [0,1,Z,X];
```



[.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

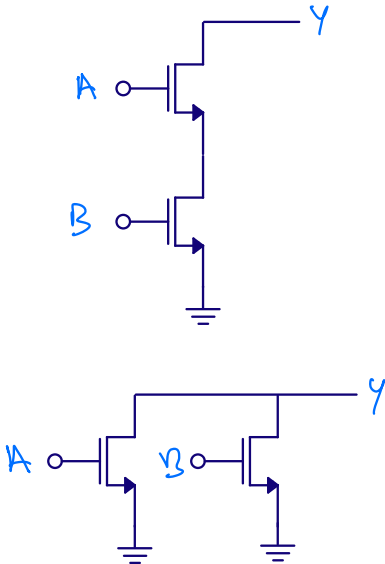
[.table-separator: #000000, stroke-width(1)] [.table: margin(8)]

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



B. Rules for inverting logic

Pull-up OR

=>

PMOS in series

=>

POS AND

⇒

PMOS in paralell

⇒

PAP

Pull-down OR

⇒

NMOS in paralell

⇒

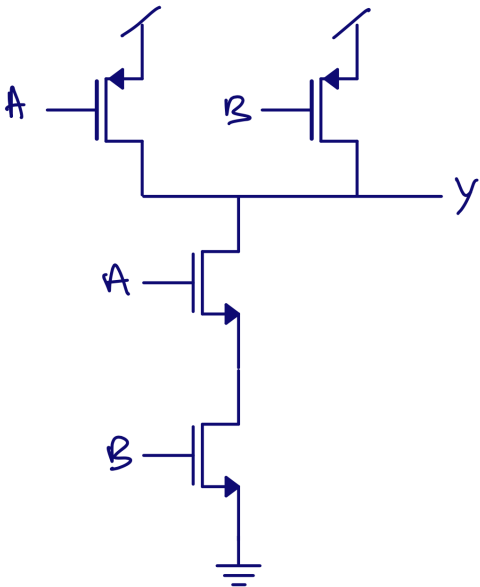
NOP AND

⇒

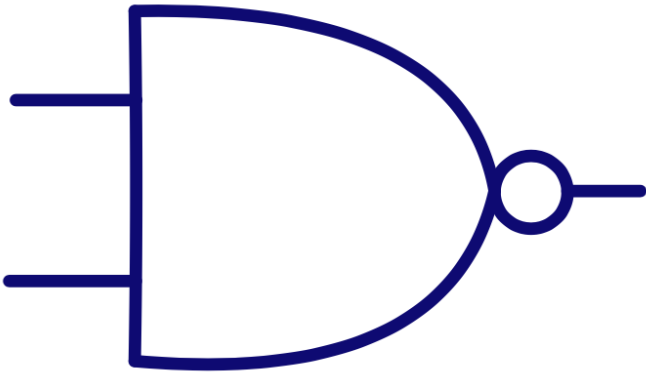
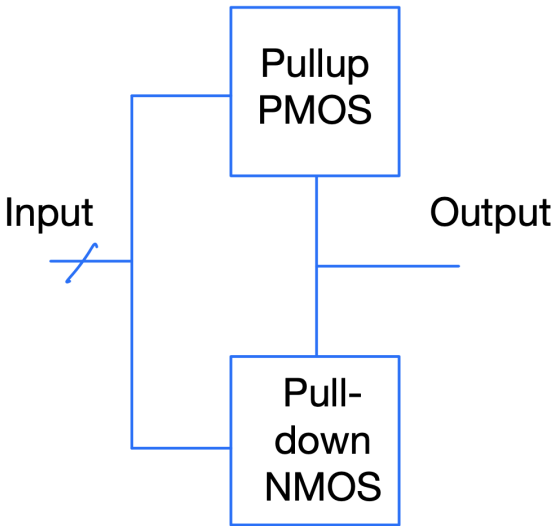
NMOS in series

⇒

NAS



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)]

[.table-separator: #000000, stroke-width(1)] [.table: margin(8)]

$Y = \overline{A + B} = \text{NOT} (A \text{ OR } B)$

$Y = \overline{AB} = \text{NOT} (A \text{ AND } B)$

OR PU

⇒

AND PU

⇒

PMOS in paralell PD

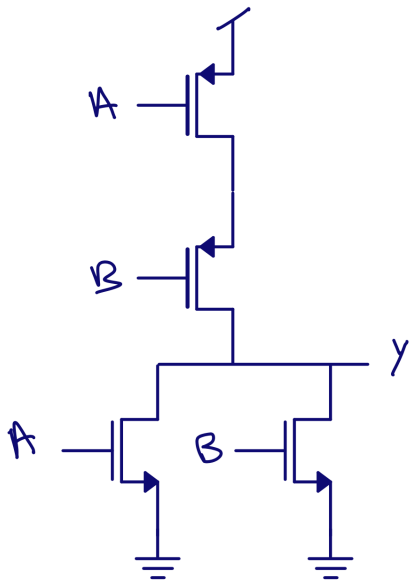
⇒

PMOS in series PD

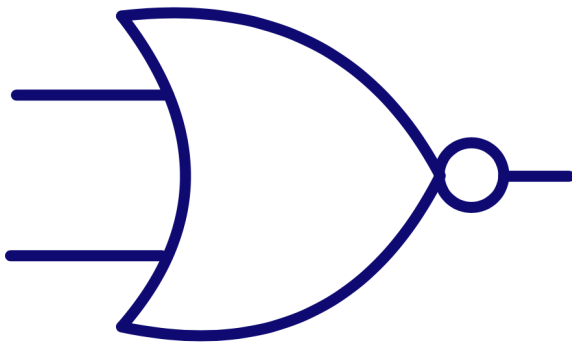
⇒

NMOS in series

NMOS in paralell



A	B	$\text{NOT}(A \text{ OR } B)$
0	0	1
0	1	0
1	0	0
1	1	0



VI. SR-LATCH

Use boolean expressions to figure out how gates work.

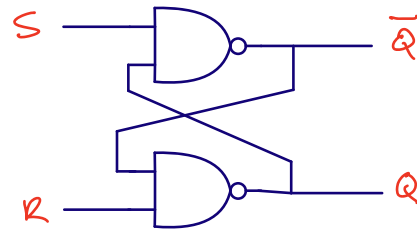
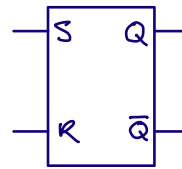
Remember De-Morgan

$$\overline{AB} = \overline{A} + \overline{B}$$

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$Q = \overline{R\overline{Q}} = \overline{R} + \overline{\overline{Q}} = \overline{R} + Q$$

$$\overline{Q} = \overline{S\overline{Q}} = \overline{S} + \overline{\overline{Q}} = \overline{S} + Q$$



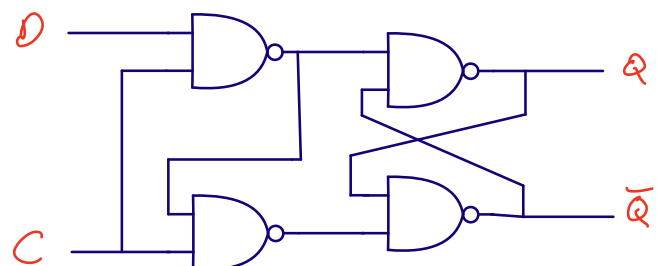
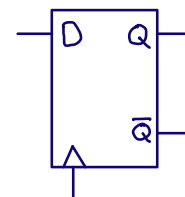
$$Q = \overline{R} + Q$$

$$\overline{Q} = \overline{S} + \overline{Q}$$

S	R	Q	$\sim Q$
0	0	X	X
0	1	0	1
1	0	1	0
1	1	Q	$\sim Q$

VII. D-LATCH (16 TRANSISTORS)

C	D	Q	$\sim Q$
0	X	Q	$\sim Q$
1	0	0	1
1	1	1	0



VIII. OTHER LOGIC CELLS

What about

$Y = AB$

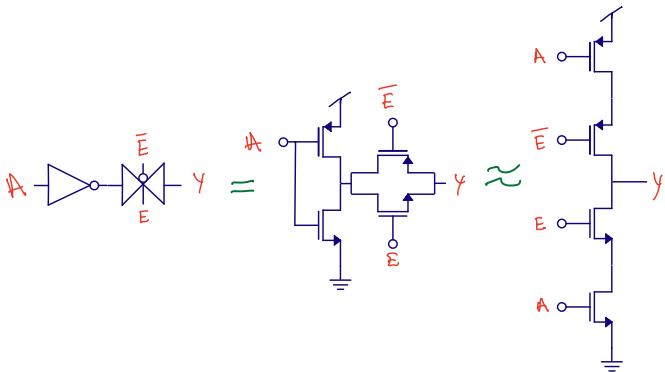
and

$Y = A + B$

?

$Y = AB = \overline{\overline{AB}}$

$Y = A \text{ AND } B = \text{NOT}(\text{NOT}(A \text{ AND } B))$



[.table-separator: #000000, stroke-width(1)] [.table: margin(8)]

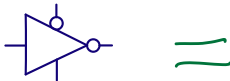
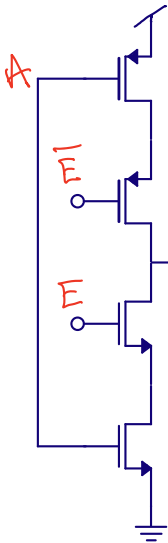
X. TRISTATE INVERTER

$Y = A+B = \overline{\overline{A+B}}$

$Y = A \text{ OR } B = \text{NOT}(\text{NOT}(A \text{ OR } B))$



E	A	Y
0	0	Z
0	1	Z
1	0	1
1	1	0

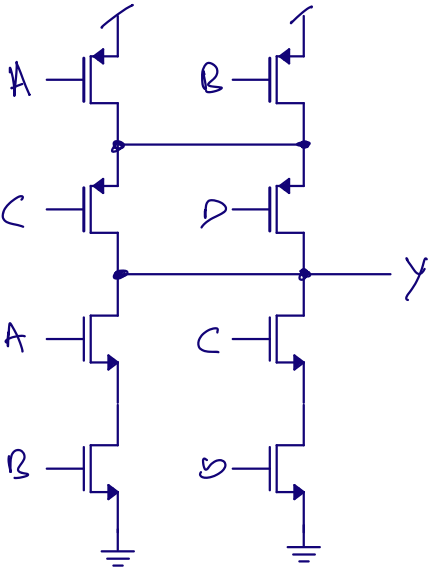


[.table-separator: #000000, stroke-width(1)] [.table: margin(8)]

IX. AOI22: AND OR INVERT

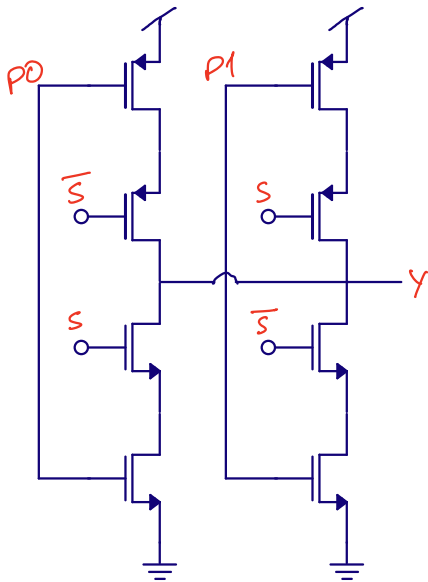
$Y = \text{NOT}(A \text{ AND } B \text{ OR } C \text{ AND } D)$

$Y = \overline{AB + CD}$

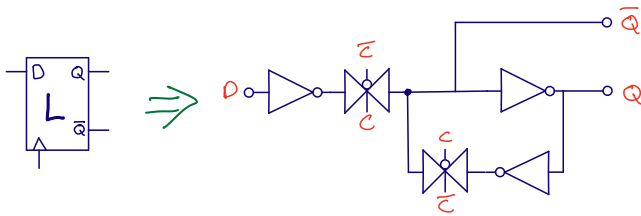


XI. MUX

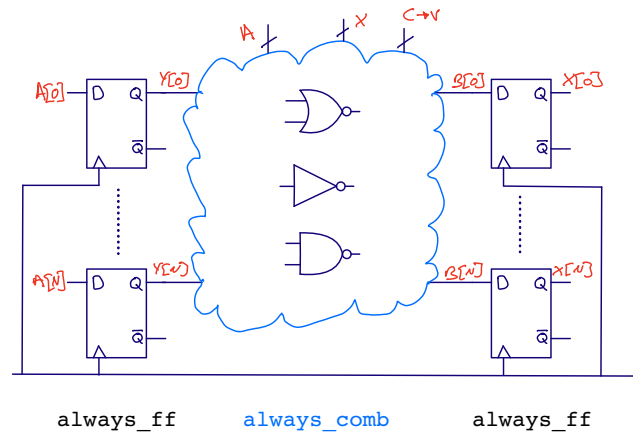
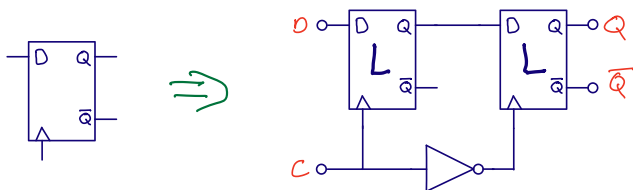
S	Y
0	NOT(P1)
0	NOT(P1)
1	NOT(P0)
1	NOT(P0)



D-Latch (12 transistors)



D-Flip Flop (< 26 transistors)



always_ff

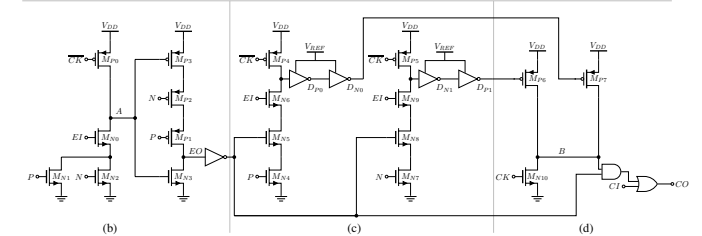
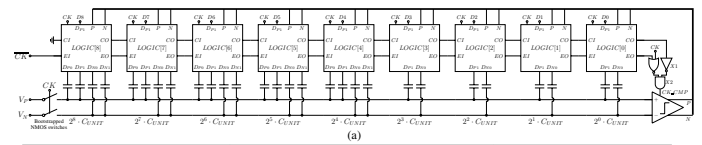
always_comb

always_ff

XII. THERE ARE OTHER TYPES OF LOGIC

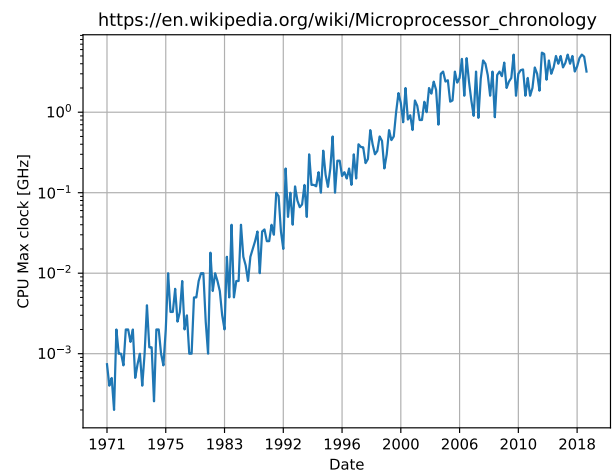
- True single phase clock (TSPC) logic
- Pass transistor logic
- Transmission gate logic
- Differential logic
- Dynamic logic

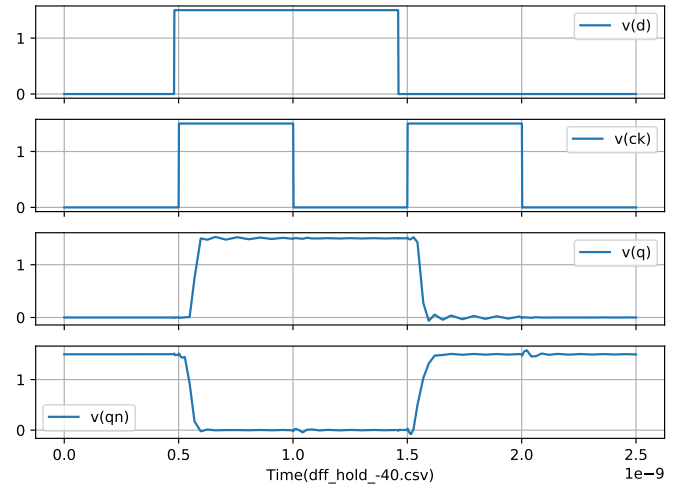
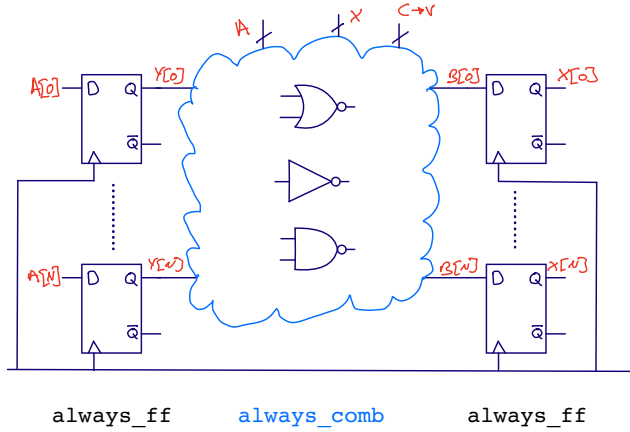
Consider other types of logic “rule breaking”, so you should know why you need it.



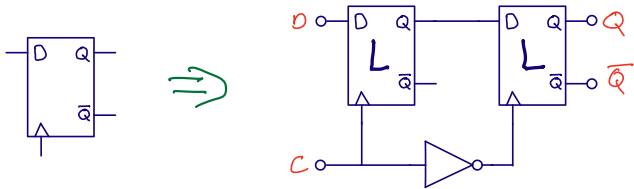
Dynamic logic => A Compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for Bluetooth Low Energy Receivers

XIII. SPEED





XIV. FLIP-FLOPS AND SPEED



XV. TIMING ANALYSIS

Analyze arrival times of all nodes in a combinatorial circuit

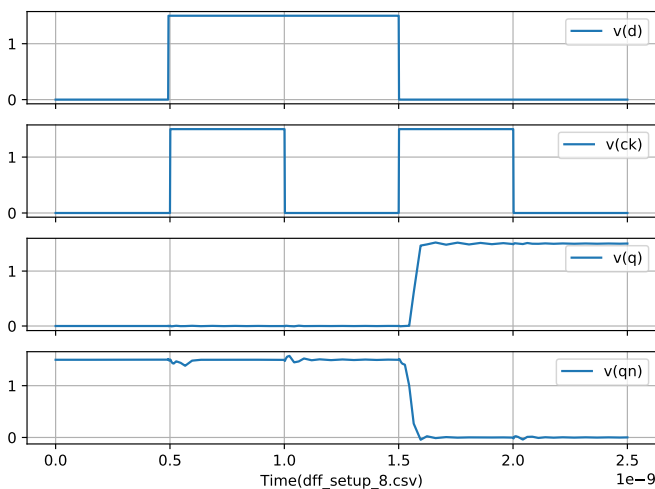
$$arrival_j = \max_{j \in fanin(i)} arrival_j + t_{pd_i} \Rightarrow a_j = \max_{j \in fanin(i)} a_j + t_{pd_i}$$

$$slack_i = required_i - arrival_j$$

dicex/lib/SUN_TR_GF130N.spi:

```
.SUBCKT DFRNQN1_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 NDIRIX1_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

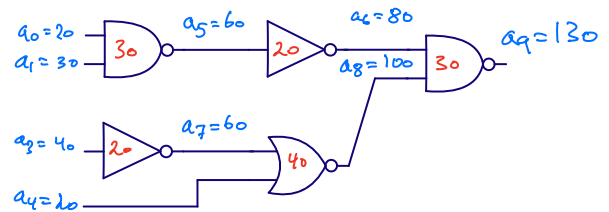
Positive slack (over PVT¹)

⇒

Timing is OK Negative slack (over PVT²)

⇒

Timing is not OK



XVI. TIMING ANALYSIS TOOLS

Commercial [Cadence Tempus](#)

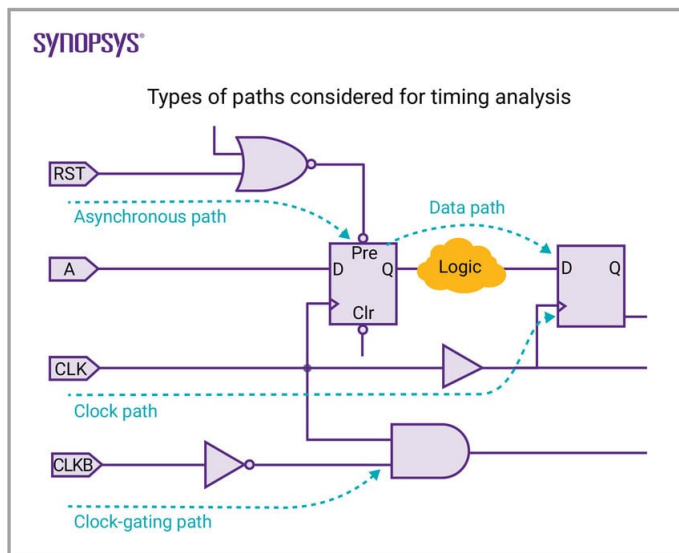
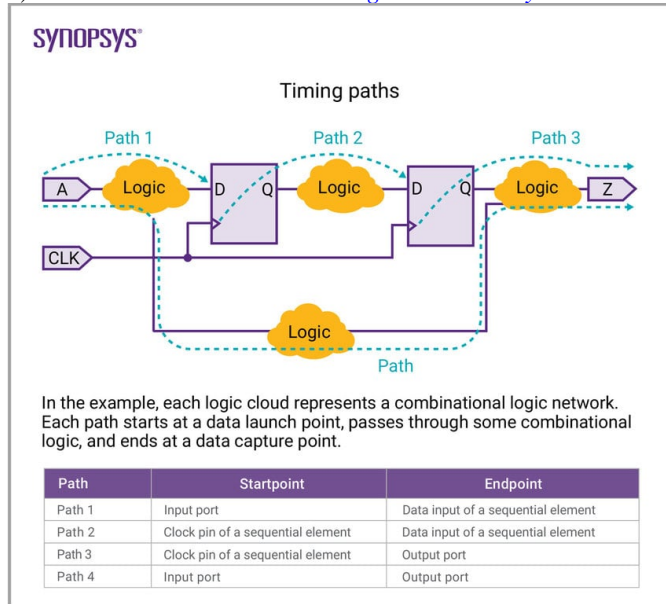
[Synopsys PrimeTime](#)

Free [OpenTimer](#)

¹Often called *clock gating*

²Often called *clock gating*

1) What is timing analysis:



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");
  }
}
}
```

XVII. EVERY GATE MUST BE SIMULATED TO PROVIDE BEHAVIOR OVER INPUT TRANSITION AND LOAD CAPACITANCE

XVIII. ALL ANALOG BLOCKS MUST HAVE ASSOCIATED LIBERTY FILE TO DESCRIBE BEHAVIOR AND TIMING PATHS *If you integrate analog into digital top flow*

XIX. GATE DELAY

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

XX. DELAY ESTIMATION

How can we get a reasonably accurate hand calculation model of delay?

$$C \approx 1 \text{ fF}/\mu\text{m}$$

$$R \approx 1 \text{ k}\Omega/\mu\text{m}$$

1) Inverter with inverter load:

$$C \approx 1 \text{ fF}/\mu\text{m}$$

,

$$R \approx 1 \text{ k}\Omega/\mu\text{m}$$

$$t_{pd} = R \times 6C = 6RC$$

$$t_{pd} = 6 \times 1 \times 10^3 \times 1 \times 10^{-15} \text{ s}$$

$$t_{pd} = 6 \times 10^{-12} = 6 \text{ ps}$$

XXI. ELMORE DELAY

$$t_{pd} \approx \sum_{\text{nodes}} R_{\text{nodes-to-source}} C_i$$

$$= R_1 C_1 + (R_1 + R_2) C_2 + \dots + (R_1 + R_2 + \dots + R_N) C_N$$

Good enough for hand calculation

XXII. 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

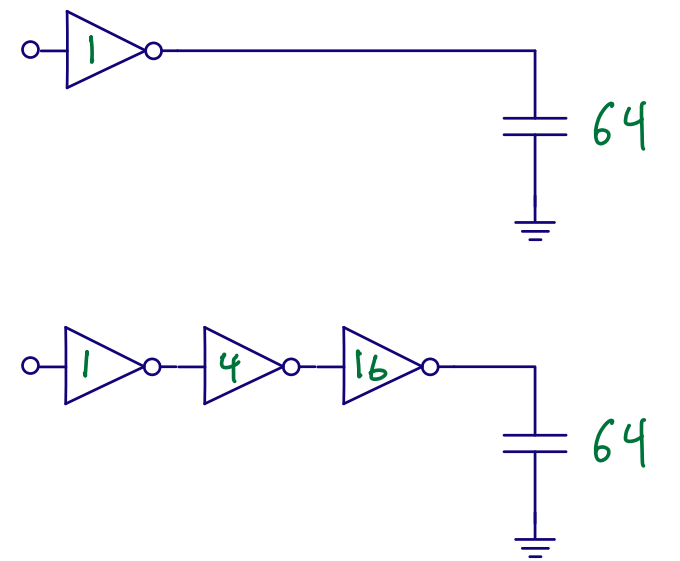
$$\Rightarrow d = f + p = gh + p = 5\frac{2}{3}$$

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	$D_F = \sum f_i$
parasitic delay	p	$P = \sum p_i$
delay	$d = f + p$	$D = \sum d_i = D_F + P$

- XXIII. MODERN IC TIMING ANALYSIS REQUIRES
COMPUTERS WITH ADVANCED PROGRAMS³
- XXIV. BEST NUMBER OF STAGES
- XXV. 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	$d = f + p$	$D = \sum d_i = D_F + P$

$$H = C_{cout}/C_{in} = 64$$
$$G = \prod g_i = \prod 1 = 1$$
$$B = 1$$
$$F = GBH = 64$$

One stage

$$f = 64 \Rightarrow D = 64 + 1 = 65$$

Three stage with

$$f = 4$$

$$D_F = 12, p = 3 \Rightarrow D = 12 + 3 = 15$$

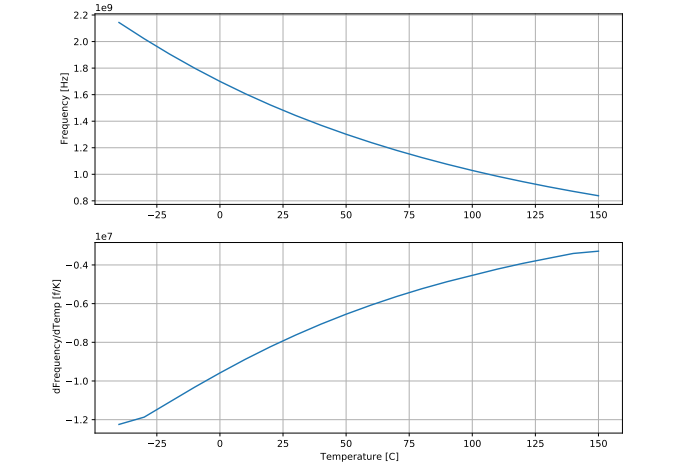
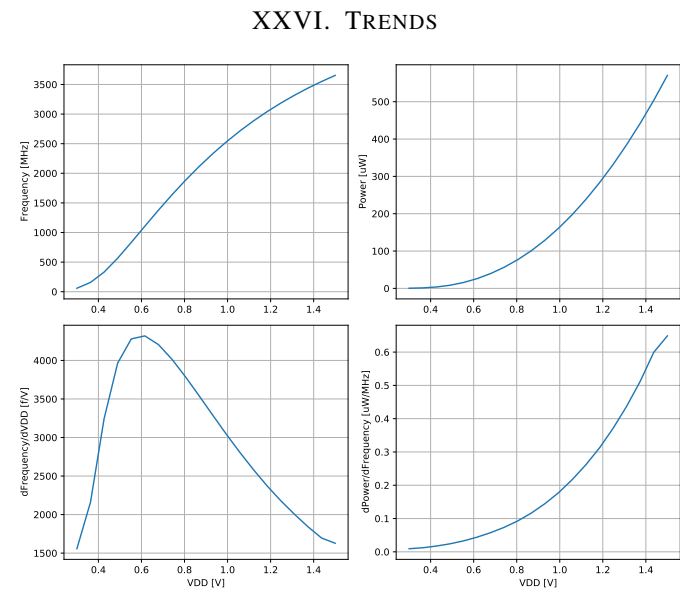
For close to optimal delay, use

$$f = 4$$

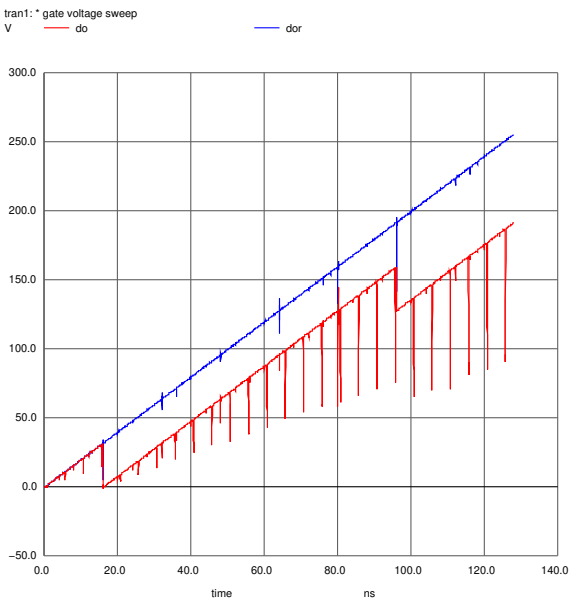
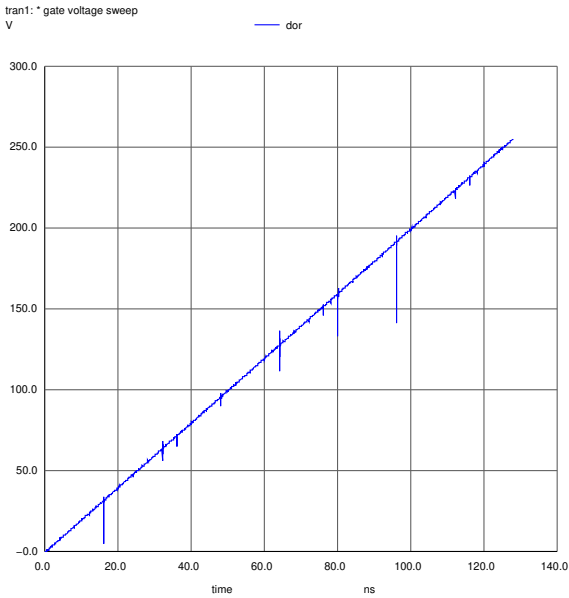
(Used to be

$$f = e$$

)



³Often called power gating



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)

ChipWhisperer

The ChipWhisperer® ecosystem presents the first open-source, low-cost solution to expose weaknesses that exist in embedded systems all around us.

Software Documentation Hardware Documentation

Single Board Solutions

Starter Kits

Target Boards

CW308 UFO Targets



conduct side-channel
power analysis and fault
injection attacks

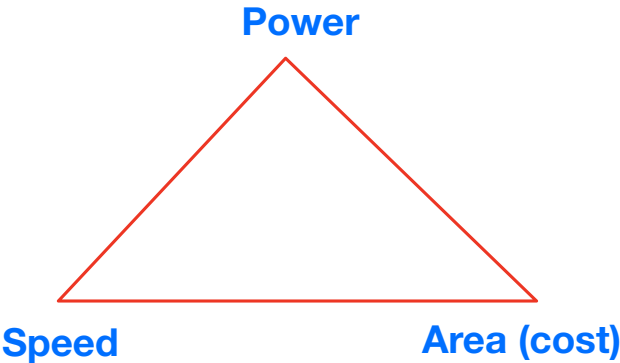
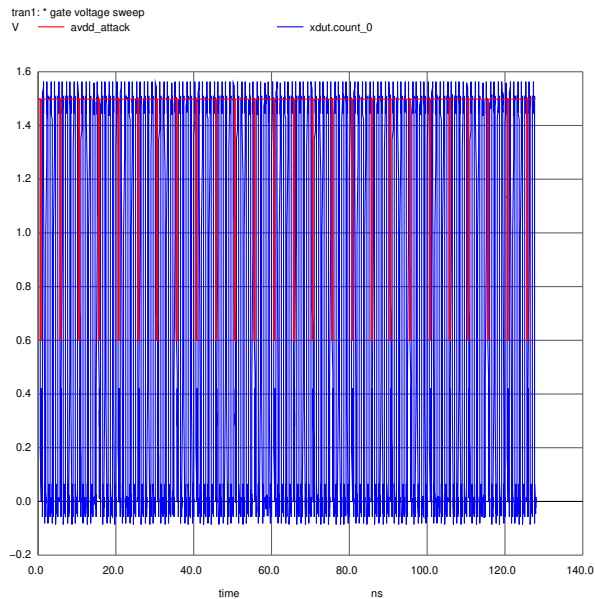


uncover vulnerabilities
in your own
embedded systems



implement
robust security
countermeasure

XXVIII. PICK TWO



XXIX. POWER

XXX. WHAT IS POWER?

Instantaneous 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]

XXXI. POWER DISSIPATED IN A RESISTOR

Ohm's Law

$$V_R = I_R R$$

$$P_R = V_R I_R = I_R^2 R = \frac{V_R^2}{R}$$

XXXII. CHARGING A CAPACITOR TO VDD

Capacitor differential equation

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

$$E_C = \int_0^\infty I_C V_C dt = \int_0^\infty C \frac{dV}{dt} V_C dt = \int_0^{V_C} C V dV = C \left[\frac{V^2}{2} \right]_0^{V_{DD}}$$

$$E_C = \frac{1}{2} C V_{DD}^2$$

XXXIII. ENERGY TO CHARGE A CAPACITOR TO A VOLTAGE VDD

$$E_C = \frac{1}{2} C V_{DD}^2$$

$$I_{VDD} = I_C = C \frac{dV}{dt}$$

$$E_{VDD} = \int_0^\infty I_{VDD} V_{DD} dt = \int_0^\infty C \frac{dV}{dt} V_{DD} dt = C V_{DD} \int_0^{V_{DD}} dV = C V_{DD}^2$$

Only half the energy is stored on the capacitor, the rest is dissipated in the PMOS

XXXIV. 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

XXXV. POWER CONSUMPTION OF DIGITAL CIRCUITS

$$E_{VDD} = C V_{DD}^2 f$$

In a clock distribution network (chain of inverters), every output is charged once per clock cycle

$$P_{VDD} = C V_{DD}^2 f$$

XXXVI. SOURCES OF POWER DISSIPATION IN CMOS LOGIC

$$P_{total} = P_{dynamic} + P_{static}$$

Dynamic power dissipation

Charging and discharging load capacitances

short-circuit 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

$$P_{static} = (I_{sub} + I_{gate} + I_{pn}) V_{DD}$$

XXXVII. 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 $\overline{P}_i = 1 - P_i$

$$\overline{P}_i = 1 - P_i$$

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 probability is uncorrelated from cycle to cycle

$$\alpha_i = \overline{P}_i P_i$$

XXXVIII. SWITCHING PROBABILITY

Random data

$P = 0.5$

,

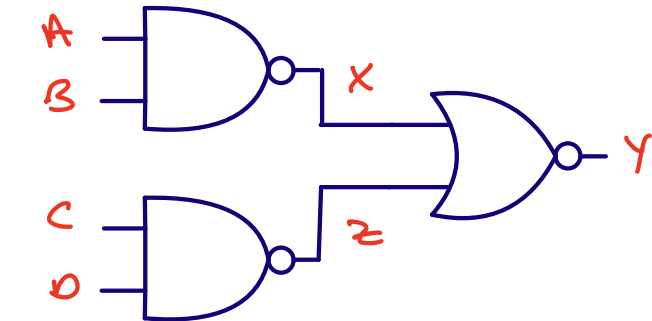
$\alpha = 0.25$

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

$P = P_A = P_B = P_C = P_D = \frac{1}{2}$

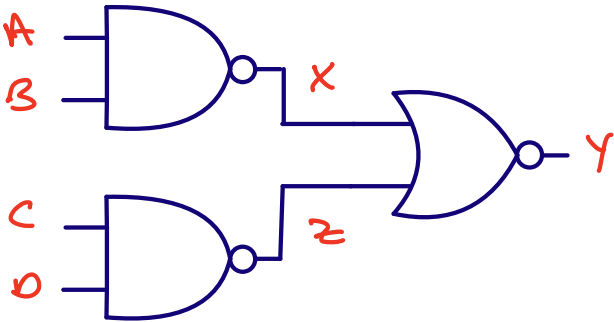
$P_X = P_Z = 1 - PP = 1 - \frac{1}{4} = \frac{3}{4}$

$\overline{P_X} = \overline{P_Y} = \frac{1}{4}$

$P_Y = \frac{1}{4} \times \frac{1}{4} = \frac{1}{16}$

$\alpha = \frac{1}{16} \left(1 - \frac{1}{16} \right) = \frac{15}{16} \frac{1}{16} = \frac{15}{256}$

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

$\overline{A + B} = \bar{A} \cdot \bar{B}$

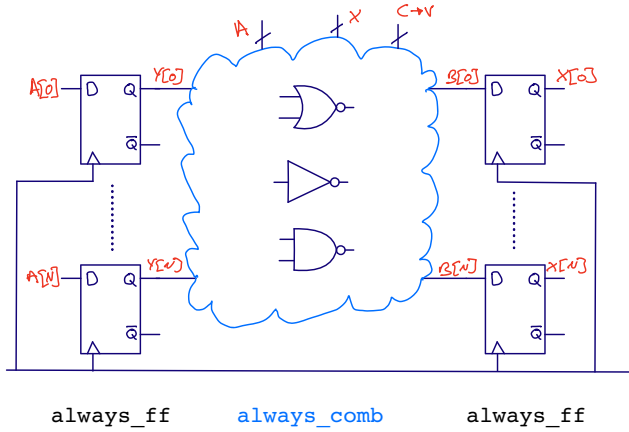
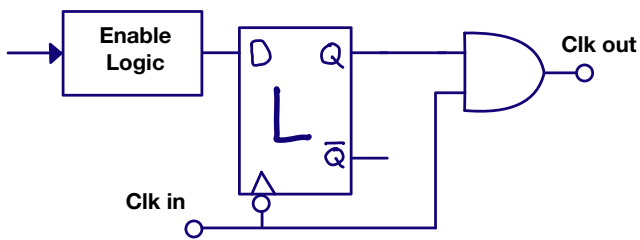
$\overline{\overline{AB} + \overline{CD}} = \overline{\overline{ABCD}} = ABCD$

$\Rightarrow P_Y = P_A P_B P_C P_D = \left(\frac{1}{2} \right)^4 = \frac{1}{16}$

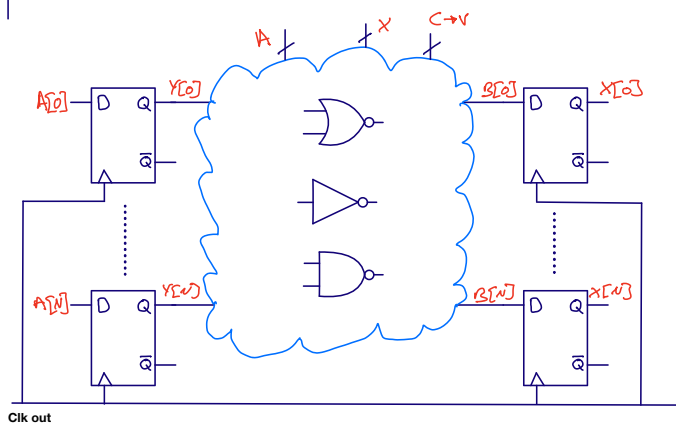
$P_{tot} = \alpha C V_{DD}^2 f$

XXXIX. STRATEGIES TO REDUCE DYNAMIC POWER

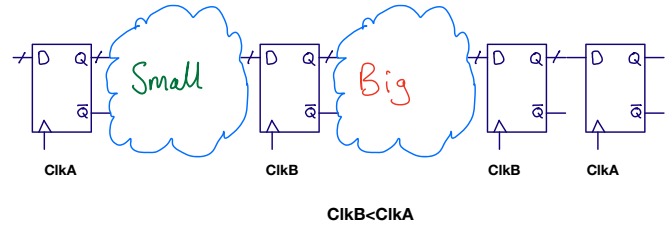
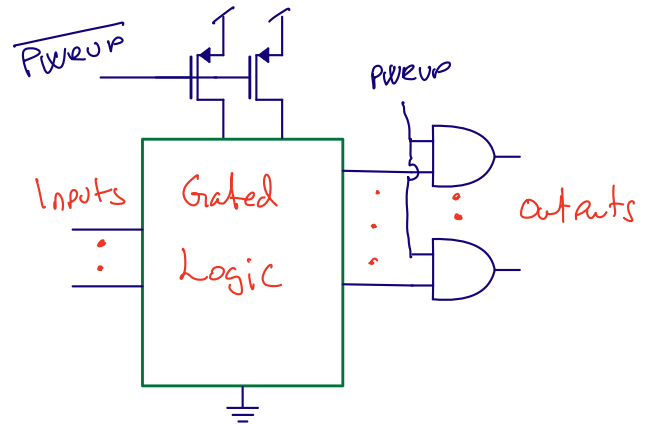
1. Stop clock
2. Stop activity
3. Reduce clock frequency
4. Turn off VDD
5. Reduce VDD

A. Stop clock⁴

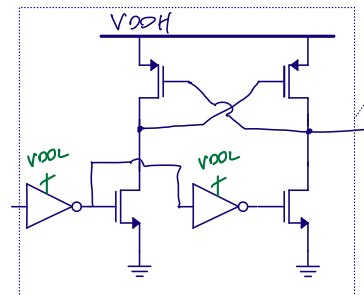
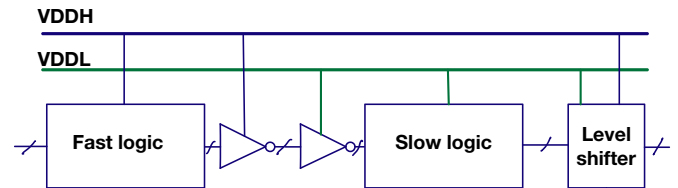
B. Stop activity

⁴Often called clock gating

C. Reduce frequency

D. Turn off power supply⁵

1) Reduce power supply:



2) Energy-Delay Product:

$$EDP = k \frac{C^2 V_{DD}^3}{(V_{DD} - V_t)^{1 \text{ to } 2}}$$

Differentiating with respect to

$$V_{DD}$$

⁵Often called power gating

and setting the result to

$$0$$

it's possible to work out that

$$V_{DD-opt} = \frac{3}{3 - 1 \text{ to } 2} V_t \in [1.5, 3] V_t$$

XL. WIRES

XLI. WIRE GEOMETRY

Pitch = w + s

Aspect ratio (AR) = t/w

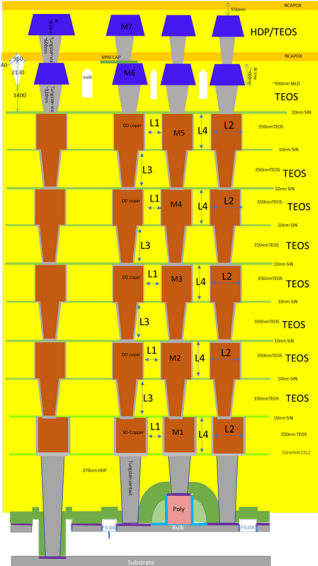
These days

$$AR \approx 2$$

XLII. 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.



XLVIII. CONTACTS

Contacts and vias can have 2-20

Ω

Must use many contacts/vias for high current wires

XLIX. WIRE CAPACITANCE

Dense wires has about

$0.2\text{ fF}/\mu\text{m}$

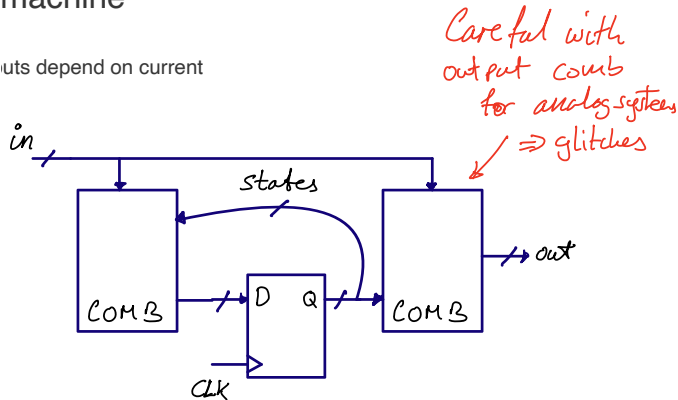
L. FSM

LI. MEALY MACHINE

An FSM where outputs depend on current state and inputs

machine

uts depend on current

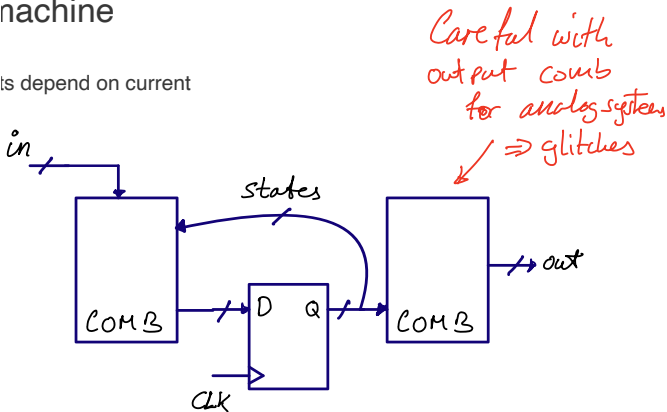


LII. MOORE MACHINE

An FSM where outputs depend on current state

nachine

ts depend on current



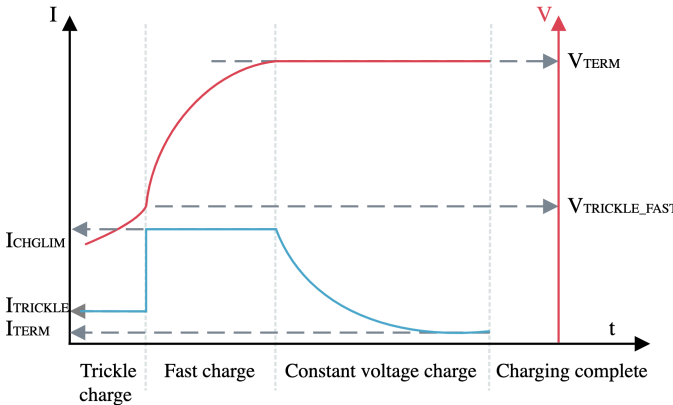
LIII. MEALY VERSUS MOORE

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

A. 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  
end  
endmodule // counter
```

LIV. BATTERY CHARGER FSM



A. Li-Ion batteries

Most Li-Ion batteries can tolerate 1 C during fast charge

For Biltema 18650 cells:

$1\text{ C} = 2950\text{ mA}$

$0.1\text{ C} = 295\text{ mA}$

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

B. Battery charger - Inputs

Voltage above

$$V_{TRICKLE}$$

Voltage close to

$$V_{TERM}$$

If voltage close to

$$V_{TERM}$$

and current is close to

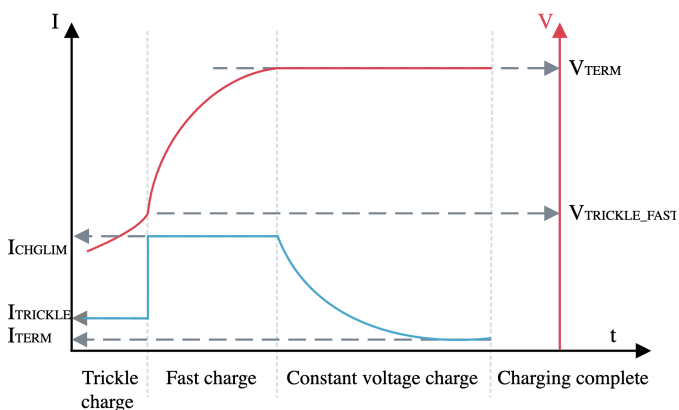
$$I_{TERM}$$

, then charging complete

If charging complete, and voltage has dropped (

$$V_{RECHARGE}$$

), then start again



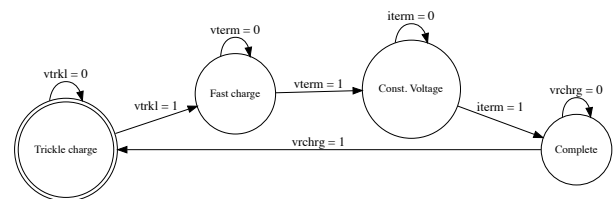
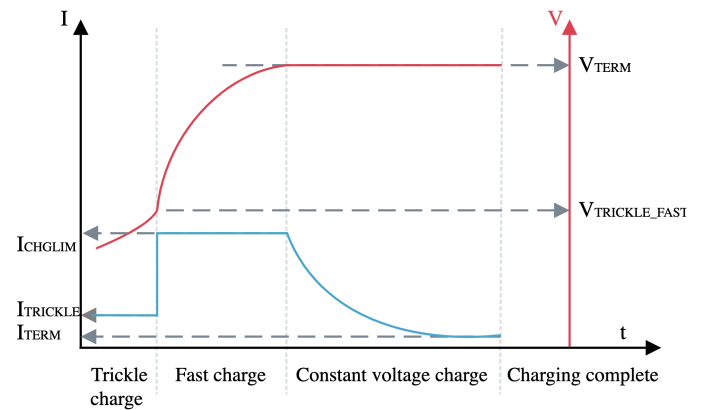
C. Battery charger - States

Trickle charge (0.1 C)

Fast charge (1 C)

Constant voltage

Charging complete



1) One way to draw FSMs - Graphviz:

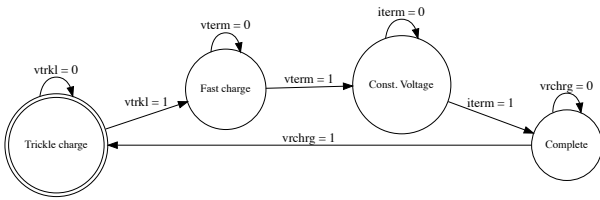
```
digraph finite_state_machine {
    rankdir=LR;
    size="8, 5"
```

```
node [shape = doublecircle, label="Trickle ch
node [shape = circle, label="Fast charge", fo
node [shape = circle, label="Const. Voltage",
node [shape = circle, label="Done", fontsize=
```

```
trkl -> trkl [label="vtrkl = 0"];
trkl -> fast [label="vtrkl = 1"];
fast -> fast [label="vterm = 0"];
fast -> vconst [label="vterm = 1"];
vconst-> vconst [label="item = 0"];
vconst-> done [label="item = 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

```

2) Synthesize *FSM* with *yosys*:
dicex/sim/verilog/bcharger_sv/bcharger.yosys

```

# 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;

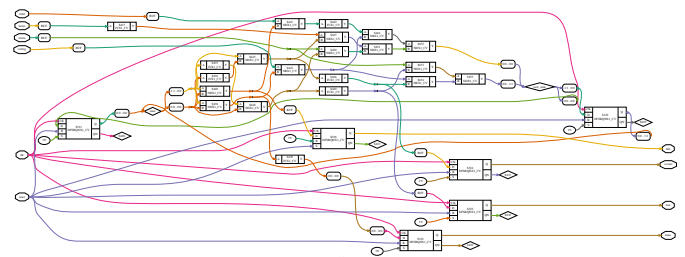
# mapping flip-flops
dfflibmap -liberty ../../lib/SUN_TR_GF130N.lib

# 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.spice

# write dot so we can make image
show -format dot -prefix bcharger_synth -colors 1 -width -stretch
clean

```



Carsten Wulff received the M.Sc. and Ph.D. degrees in electrical engineering from the Department of Electronics and Telecommunication, Norwegian University of Science and Technology (NTNU), in 2002 and 2008, respectively. During his Ph.D. work at NTNU, he worked on open-loop sigma-delta modulators and analog-to-digital converters in nanoscale CMOS technologies. In 2006-2007, he was a Visiting Researcher with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada. Since 2008 he's been with Nordic Semiconductor in various roles, from analog designer, to Wireless Group Manager, to currently Principle IC Scientist. From 2014-2017 he did a part time Post.Doc focusing on compiled, ultra low power, SAR ADCs in nanoscale technologies. He's also an Adjunct Associate Professor at NTNU. His present research interests includes analog and mixed-signal CMOS design, design of high-efficiency analog-to-digital converters and low-power wireless transceivers. He is the developer of Custom IC Compiler, a general purpose integrated circuit compiler, and makes the occasional video on analog integrated circuits at <https://www.youtube.com/@analogicus>. For full CV see <https://analogicus.com/markdown-cv/>.