Digital IC - Summary

Thomas Debelle

Introduction

One of the main thing about this class is finding how to optimize the ratio of $Op/s/W = Op/J$. We want to enhance this ratio. Less and less foundry have smaller and smaller technology. We want to reduce it both for *plugged* and

battery device. Either we don't have the requirements to pull out or the space to store the energy.

This class is all about Gate and transistor. Low level is king.

Transistor switch model

We can model a switch (MOSFET) with an ideal switch and a resistor :

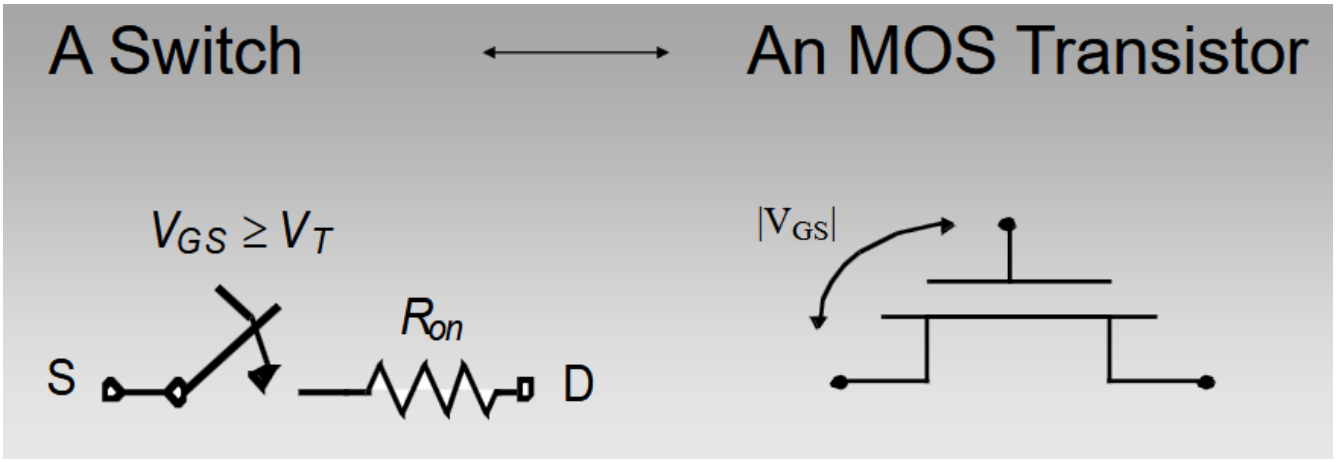


Figure 1: Switch

By the transistor scaling, short channel are behaving differently due to the **velocity saturation**. The relation becomes linear and not quadratic like it used to be.

Logic Circuits

The swing here is equal to V_{dd} so we have a high noise margin. It is not a **ratioed** logic so we can't use tricks to minimize mismatch by taking advantage of ratios. We only have 1 resistor on so low output impedance but the input is the gate of MOS so we have a high input impedance. There is no static power consumption since no direct path from Vdd to ground. That's nice :)

In the dynamic model we need to add an output cap C_L. The load cap is simply the sum of all capacitance at the output node. Transition time is determined by the charging of this cap by a resistor. The sizing impacts the dynamic behavior of the gate.

Ideally we want V_M to be at the middle of the other nominal voltage. We call the region in between the *undefined region*.

Regeneration of the level

With using this type of gate we have some regeneration level, it will amplify the signal and so we won't have undefined level and we will have the signal that will reach one defined state. If we have no regeneration, we will reach meta-stability. We have to meet some conditions :

- The transient or undefined region in the VTC should have a gain |dVout/dVin| larger than 1
- In the "legal" or defined regions the VTC gain should be smaller than 1 in absolute value
- The boundary between the defined and undefined regions are V_{IH} and V_{IL} where the gain = -1

We need gain or the signal will be lost.

We have 3 different types of noise :

1. Inductive coupling
2. Capacitive coupling
3. Power and ground noise

- case 1 : $V_{min} = V_{DS} \rightarrow$ Linear region

$$I_{DSAT} = \mu C_{ox} \frac{W}{L}\left((V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2}\right)$$

- case 2 : $V_{min} = V_{GS} - V_T \rightarrow$ (Channel) Saturation

$$I_{DSAT} = \frac{1}{2}\mu C_{ox} \frac{W}{L}\left((V_{GS} - V_T)^2\right)(1 + \lambda V_{DS})$$

- case 3 : $V_{min} = V_{DSAT} \rightarrow$ Velocity Saturation

$$I_{DSAT} = \mu C_{ox} \frac{W}{L}\left((V_{GS} - V_T)V_{DSAT} - \frac{V_{DSAT}^2}{2}\right)(1 + \lambda V_{DS})$$

Figure 2: Equations

The noise margin in CMOS is rather high which is a good thing seeing the low output impedance.

We see that the ratio of PMOS and NMOS determine the V_M voltages.

Capacitance

We know that the delay of a switch is $t_{phl} = f(R_{on}C_L) = ln(1/2)R_{on}C_L = ln(1/2)(R_{eqn} + R_{eqp})/2 \cdot C_L$. We are still observing some glitches when we switch on and off. This **isn't** due to the miller effect. This **overshoot** is due to the gate drain capacitor.

This is due to charges and sudden and "infinite" steep step at the input which will create an extra unwanted voltage. Thankfully the input isn't as steep in reality and so the effect is less severe but still noticeable. We call it the *digital miller effect*:

$$t_d = \frac{C_{total} \cdot (\Delta V + V_{DD}/2)}{I_{max}} = \frac{(C + C_L)}{I_{max}}\left(\frac{C}{C + C_L}V_{DD} + \frac{V_{DD}}{2}\right) = \frac{(3C + C_L) \cdot V_{DD}}{2I_{max}}$$

So it is like the cap becomes 3 times larger (similar to the miller effect) but in reality we are closer to 2 since we never have a perfect step at the input.

We can move those C_{gd} to the inside and see it as an impact on C_L. Again we can reuse the theory of DDP with the intrinsic and extrinsic load where $C_L = C_{int} + C_{ext}$:

$$t_p = 0.69R_{eq}(C_{int} + C_{ext}) = 0.69R_{eq}C_{int}\left(1 + \frac{C_{ext}}{C_{int}}\right) = t_{p0}\left(1 + \frac{C_{ext}}{C_{int}}\right)$$

So the sizing can help up to a certain point where we have an *irreducible* delay.

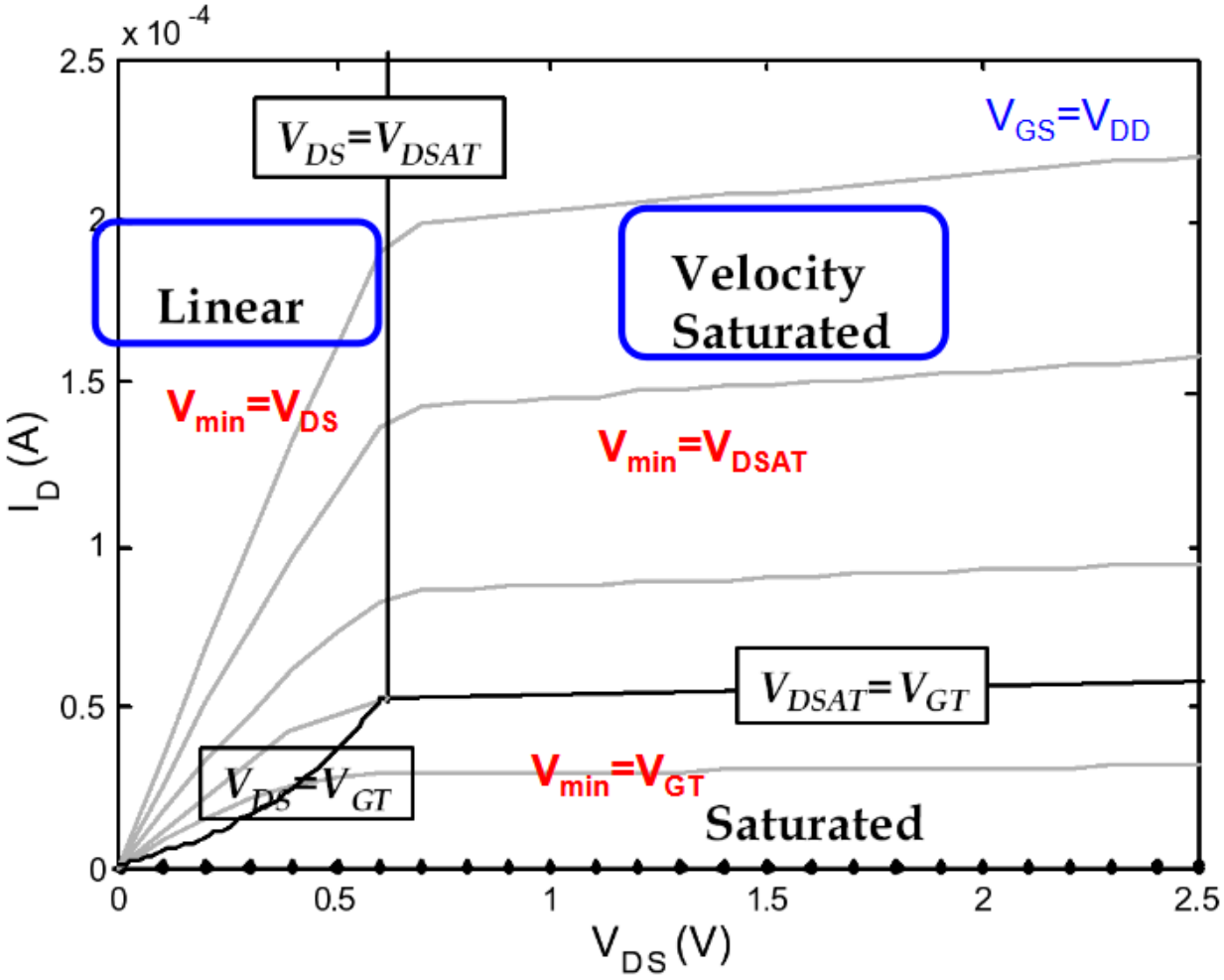


Figure 3: Regions

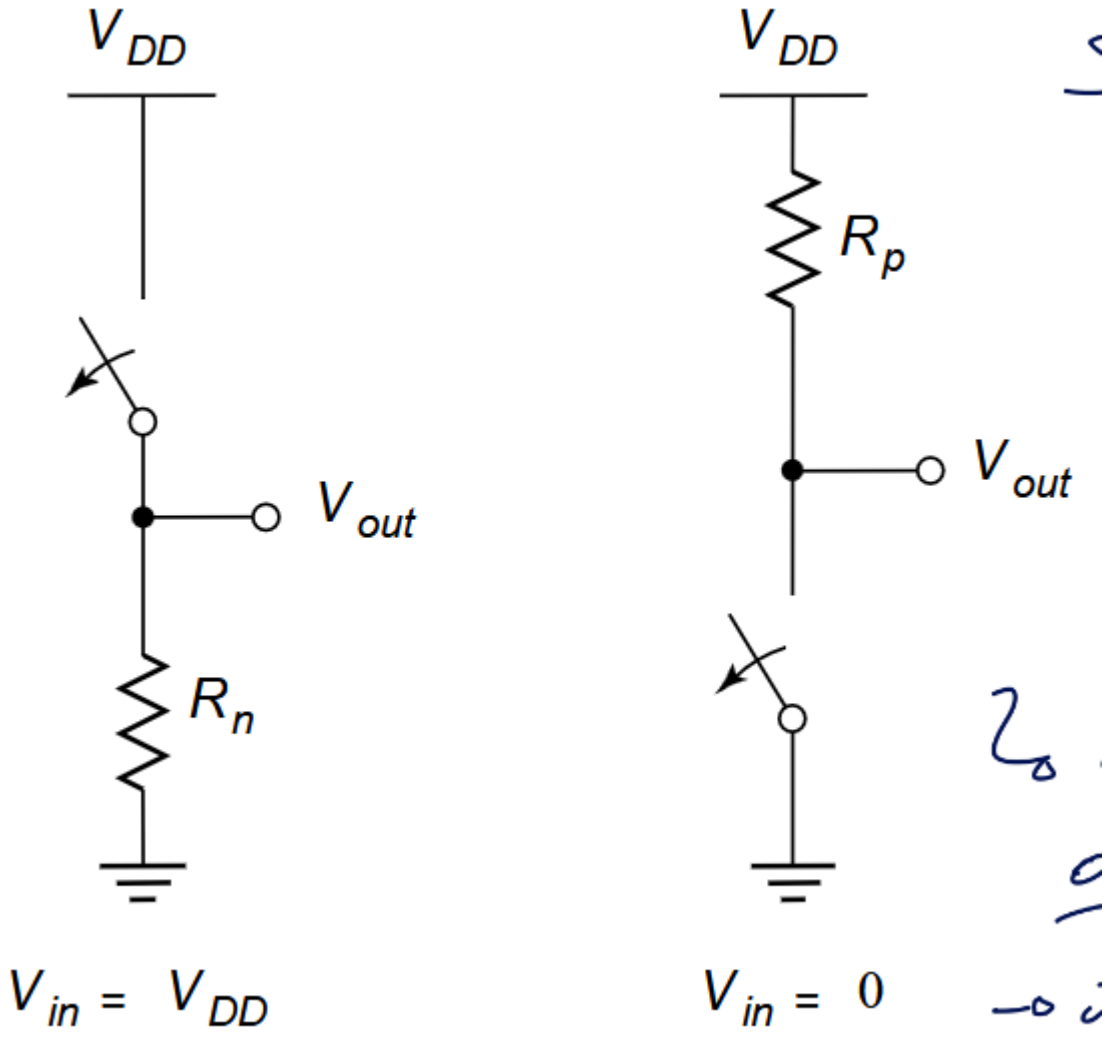


Figure 4: The static model

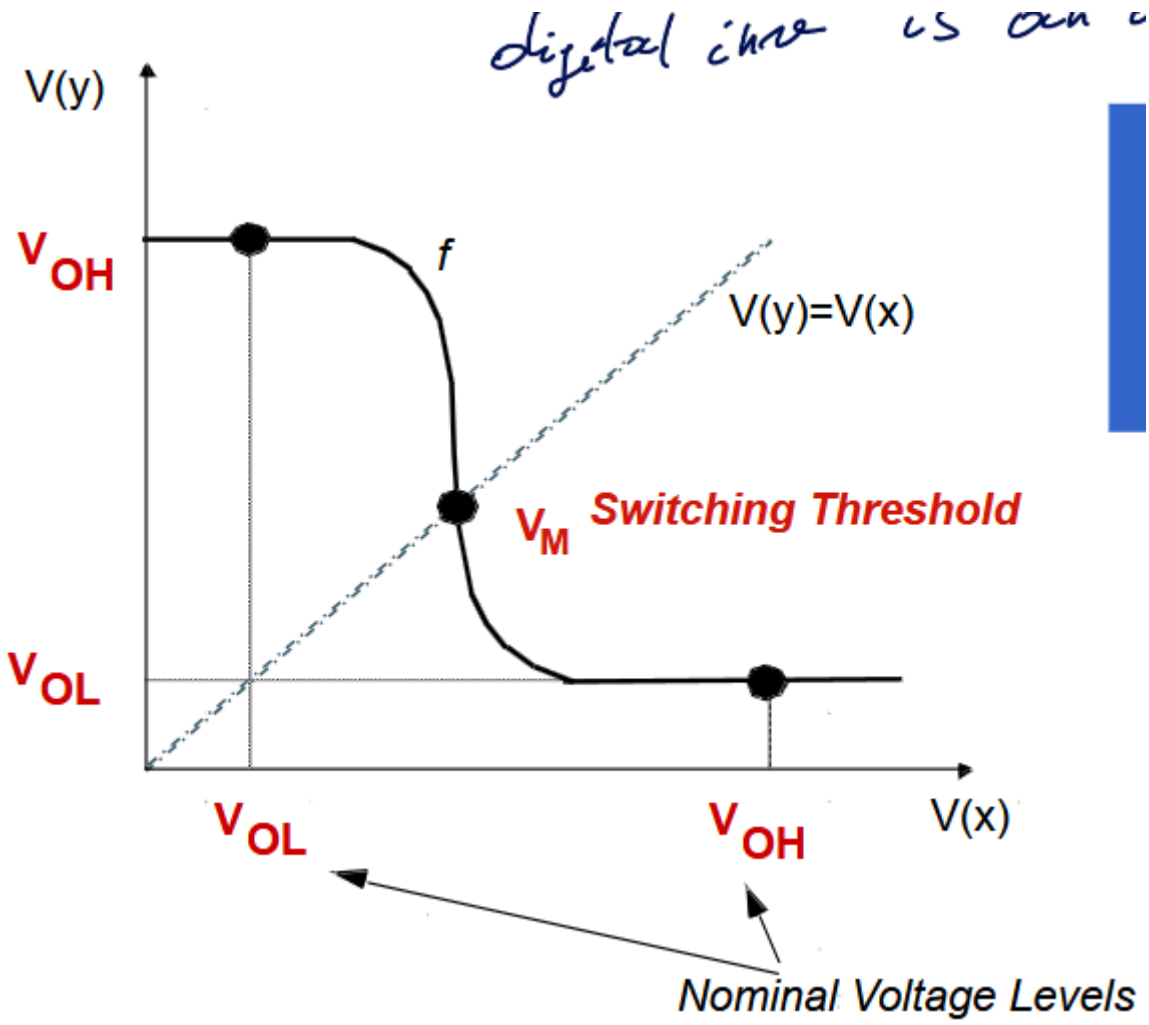


Figure 5: Switching threshold

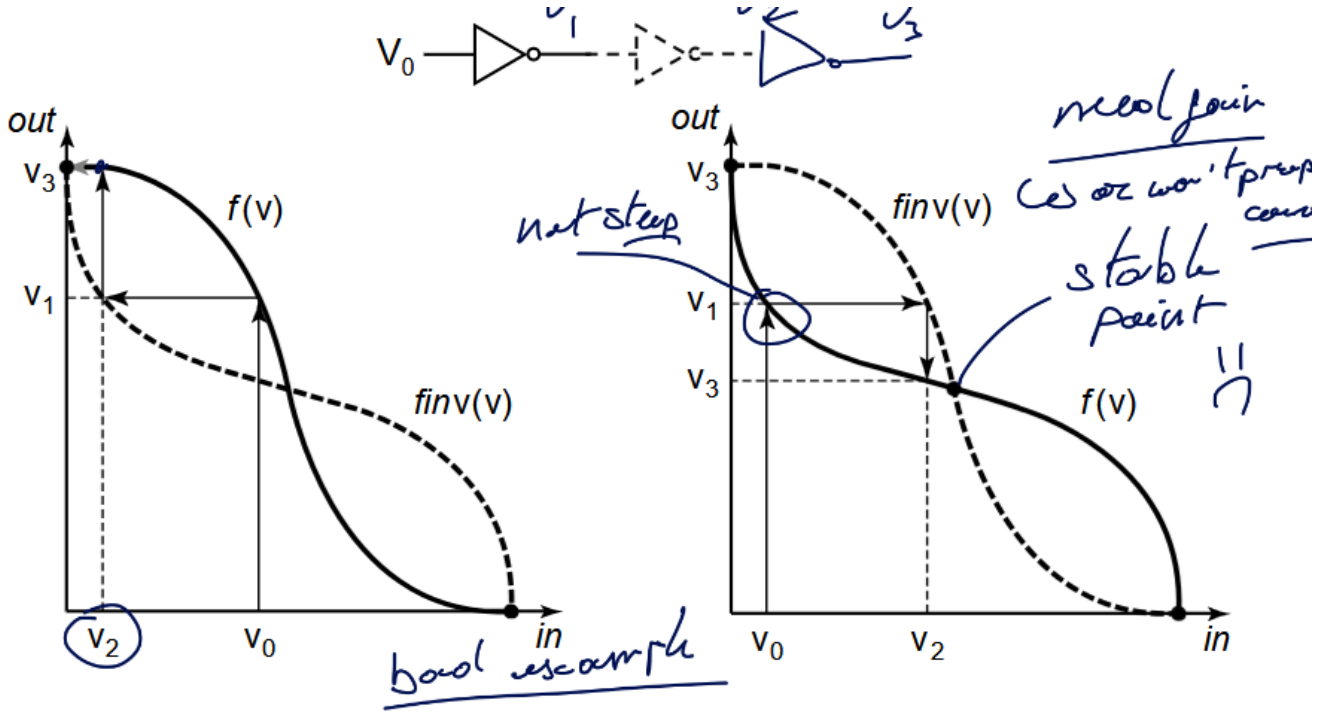


Figure 6: NAND gate regeneration

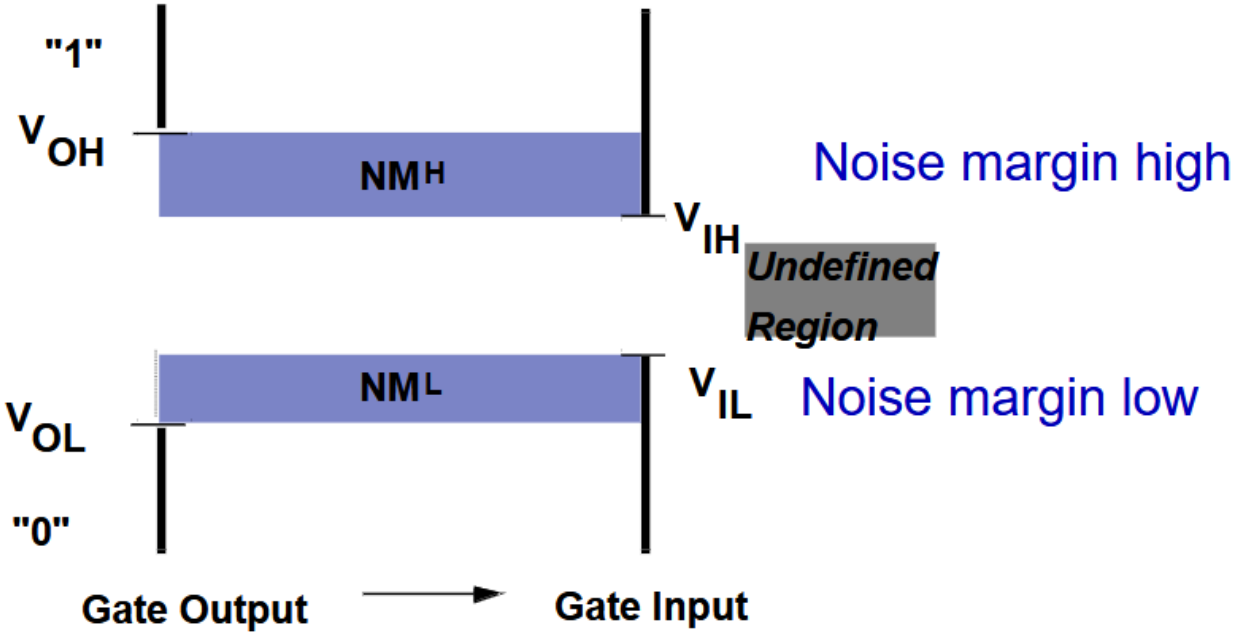


Figure 7: Noise margin

$$I_n(V_{GS} = V_M) + I_p(V_{GS} = V_M - V_{DD}) = 0$$

$$k_n V_{DSATn}(V_M - V_{Tn} - \frac{V_{DSATn}}{2}) + k_p V_{DSATp}(V_M - V_{DD} - V_{Tp} - \frac{V_{DSATp}}{2}) = 0$$

Solving for V_M yields

$$V_M = \frac{(V_{Tn} + \frac{V_{DSATn}}{2}) + r(V_{DD} + V_{Tp} + \frac{V_{DSATn}}{2})}{1+r} \text{ with } r = \frac{k_p V_{DSATp}}{k_n V_{DSATn}}$$

The ratio $(W/L)_p/(W/L)_n$ to set a certain V_M is given by

$$\frac{(W/L)_p}{(W/L)_n} = \frac{k_n' V_{DSATn}(V_M - V_{Tn} - V_{DSATn}/2)}{k_p' V_{DSATp}(V_{DD} - V_M + V_{Tp} - V_{DSATp}/2)} \text{ (both TOR in velocity saturation!)}$$

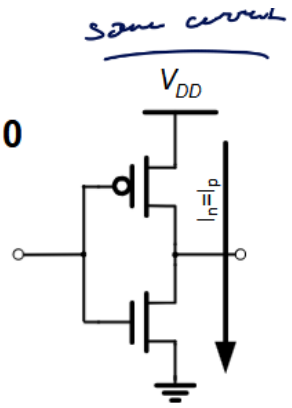


Figure 8: Switching threshold for INV

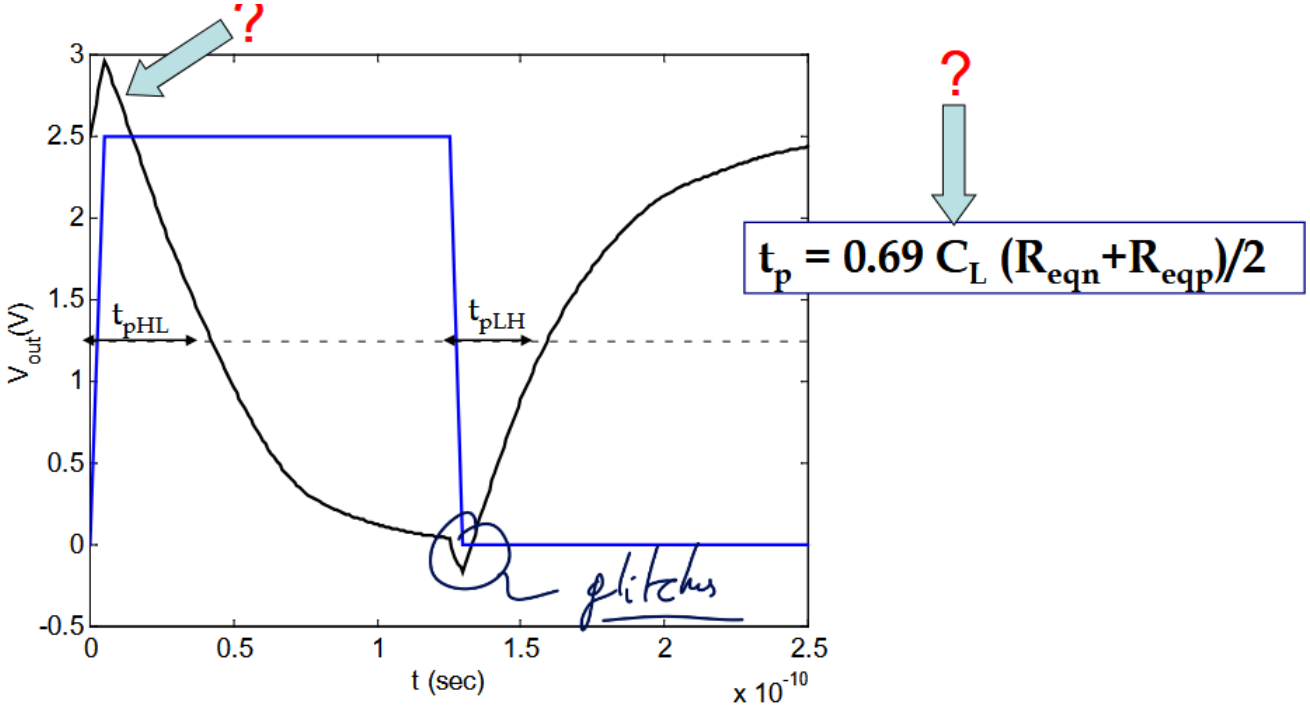


Figure 9: Glitches and Delay

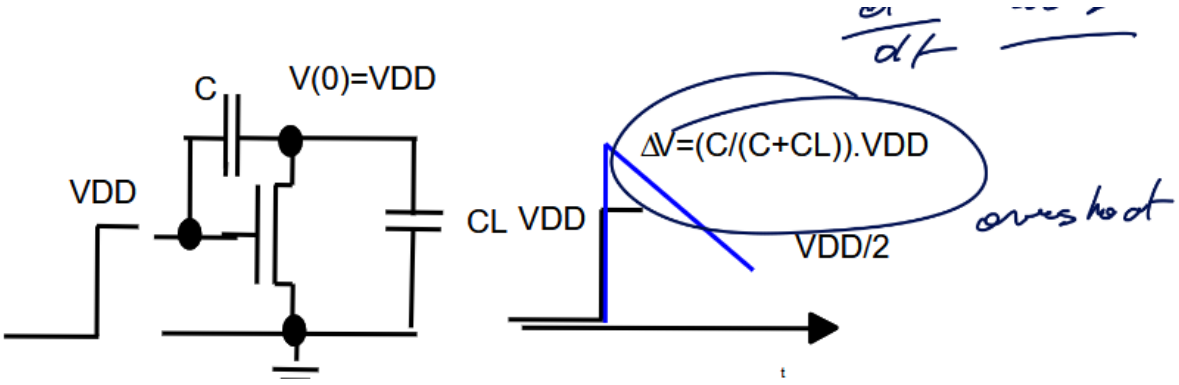
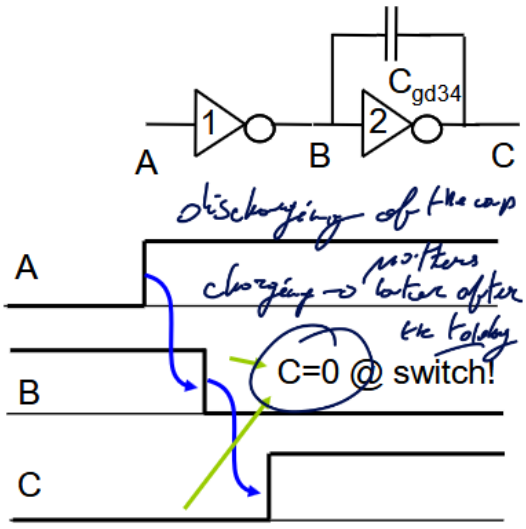


Figure 10: Explanation of the overshoot

- We analyse the influence of C_{gd34} on the delay of Invertor I_1
- C_{gd34} is loading node B, but
  - C_{gd34} is first charged + B - C
  - when node B is switching node C is at ground,
- Charge required to bring node at 0 and discharge C_{gd34} is $C_{gd34} \cdot V_{DD}$
- When node C finally switches
  - Cgd34 is charged - B + C
  - requiring another charge $C_{gd34} \cdot V_{DD}$



Conclusion: "Recharging" C_{gd34} requires a charge of $2 \cdot C_{gd34} V_{DD}$, but only half of it is exchanged during the switching of I_1

$\Rightarrow C_{gd34}$ is seen only "once" in the delay of I_1

Figure 11: Loading issue

Effective fan-out

The input C_g and intrinsic cap are always proportional to the sizing : $C_{int} = \gamma C_g$ where γ is a technological constant. Same goes for the extrinsic load C where it is the input C of the next invertor proportional to the sizing $C_{ext} = fC_g$. So we can summarize the delay t_p by :

$$t_p = t_{p0}\left(1 + \frac{f}{\gamma}\right)$$

The delay depends on the ratio between its external load capacitance and its input capacitance, te ratio is called the *effective fan-out*.

The newly introduced γ is not valid for dynamic logic or more exotic technologies. If this $\gamma = 1$ then it means that $C_{int} = C_{in}$. It is an acceptable approximation in standard CMOS logic.

$$\mathbf{t_p = kR_W C_{int}\left(1 + C_L / C_{int}\right) = t_{p0}\left(1 + f/\gamma\right)}$$

$C_{int} = \gamma C_{gin}$ with $\gamma \approx 1$

$f = C_L/C_{gin}$ - effective fanout

$R = R_{unit}/W$; $C_{int} = WC_{unit}$

$t_{p0} = 0.69 R_{unit} C_{unit}$

$R_{unit} \sim 1/V_{DD}$

Figure 12: The golden formula

For the ring oscillator where we assume equal size $f = 1$ is independent of the size. In real technology we see only a weak dependency of timing on sizing.

Signal in reality

We know that we can't have infinitely steep signal and even worse, a too slow rise and fall time could lead to metastability issues ! We could also have some actual short circuit for a brief amount of time leading to waste of energy. So in most of design software we will leave some headroom to avoid possible short-circuit and we will flag it with *max transition violation*.

Note on sizing When we see the a or b next to a transistor it is its *relative sizing* compared to a classic $\frac{W_{min}}{L_{min}}$. For complex gate and due to the various sizing we can have, we will transform a little bit our formula :

$$t_p = t_{p0}\left(p + \frac{gf}{\gamma}\right) = t_{p0} d$$

- f : electrical effort
- g : logical effort
  - due to the fact a logical gate is always slower than an invertor with equal current drive. Ratio of input cap to the cap of an inverter that delivers equal current.
- p : ratio of intrinsic delay of the gate to the intrinsic delay of an invertor

- d : gate delay
  - relative to the intrinsic delay of the reference invertor

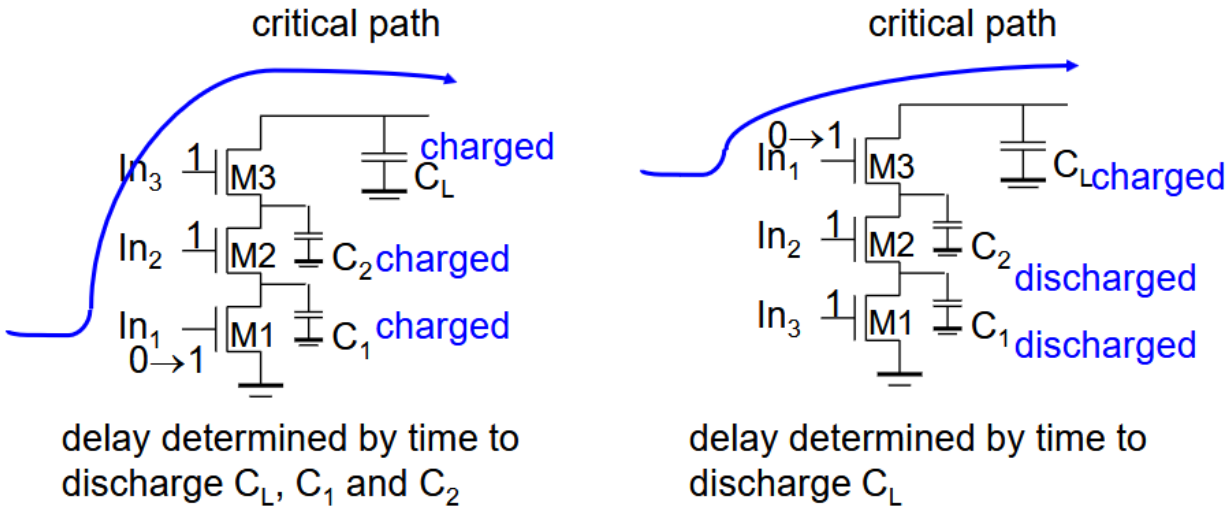


Figure 13: Critical path & Charging

Pass gate logic

Another approach than PUN and PDN as seen previously is the pass gate logic where we will not only use the gate but also the source of a transistor to create some gate.

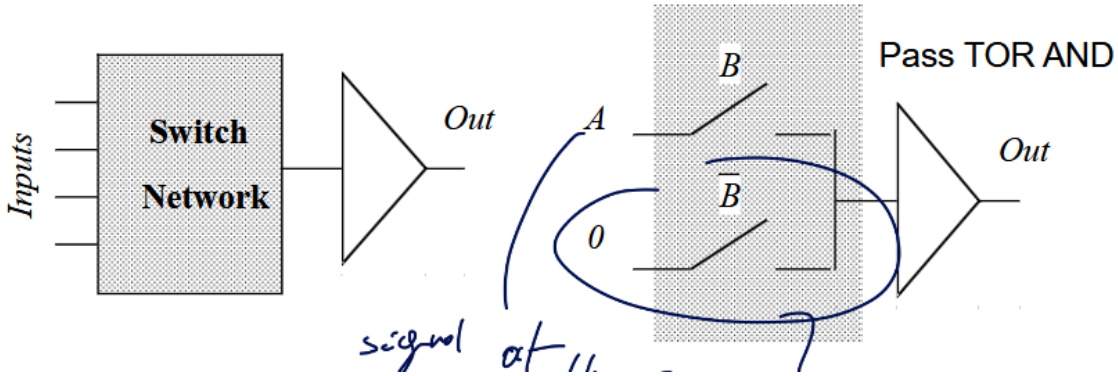


Figure 14: Pass gate logic

This technique uses less transistors. But the NMOS isn't a really good pull up transistor. It won't give a nice and crisp high voltage but rather a voltage with a $\Delta V = V_{Tloss}$. So to keep this signal crisp and nice we are obligated to add an INV to output a valid signal. This goes in the same idea as the *regeneration of the signal* logic.

So cascading of passes tor logic isn't a smart choice since the signal will rapidly deprecate.

Level restoration

So we need to do what we call *level restoration*.

It is compatible with the full swing and the static power consumption is gone. But we need a bleeder which will increases capacitance at node X and takes away the pull down current. Leads to speed degradation and needs proper

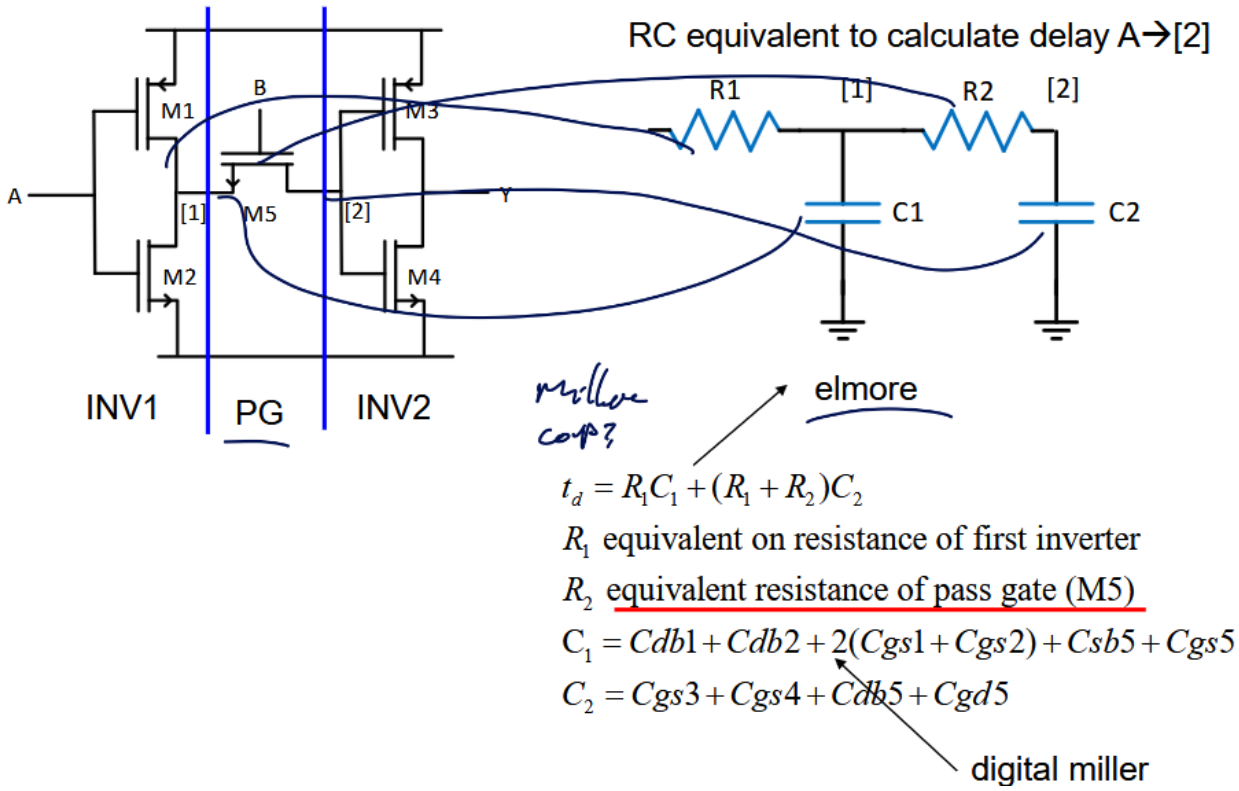


Figure 15: Static analysis

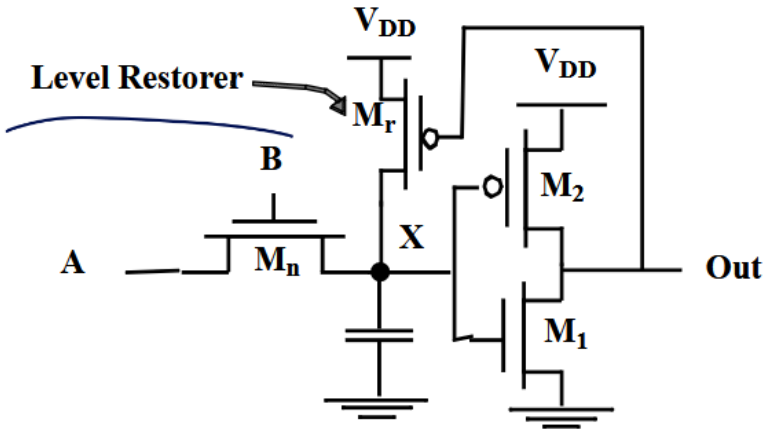


Figure 16: Level restorer

sizing.

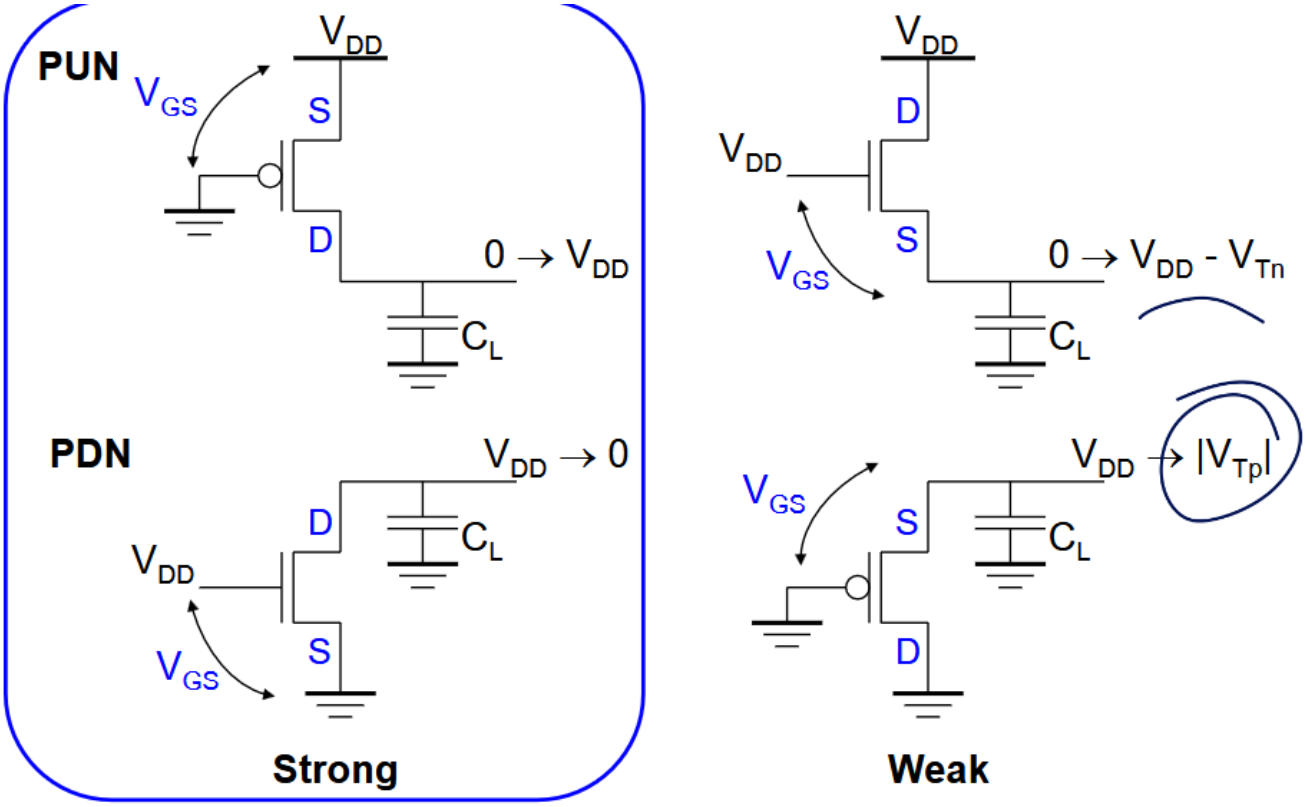


Figure 17: Threshold drops generalized

Transmission gate We can add a PMOS to the switch to create a **transmission gate**. This requires another transistor but also a complementary signal is needed so this will result in extra cost.

Circuit Timing / Dynamic Logic

Latch/Register Implementation

We know that having 2 INV back to back could lead to meta-stability so how can we reliably store data ? We want to remember the data *no triggering* but also sample and so stop looping the data *triggering*. In the second approach we simply *overpower the feedback loop* due to the asymmetry of the INV and so the input D will be more important than the output of the small INV (**David-Goliath latch**). Or we can cut the loop like in the second example.

Specifications	Positive Latch	Register
Storage ?	store when clock is low	Stores on rising edge
Transparent ?	Yes when clock is high	No

For latches we can either go for positive or negative latches and a register is simply two latch back to back. We can do latches using MUX for example.

The second option is more preferable because it will have less load on the CLK which reduce the energy waste. But we need to remember we have to *regenerate* the signal since we will have a V_{Tloss}.

The latch is pretty similar to the idea we have seen with pass gate. Here we can see again the bleeder on top and NMOS on the bottom which forms a basic INV.

NEED TO READ THIS PART CAREFULLY

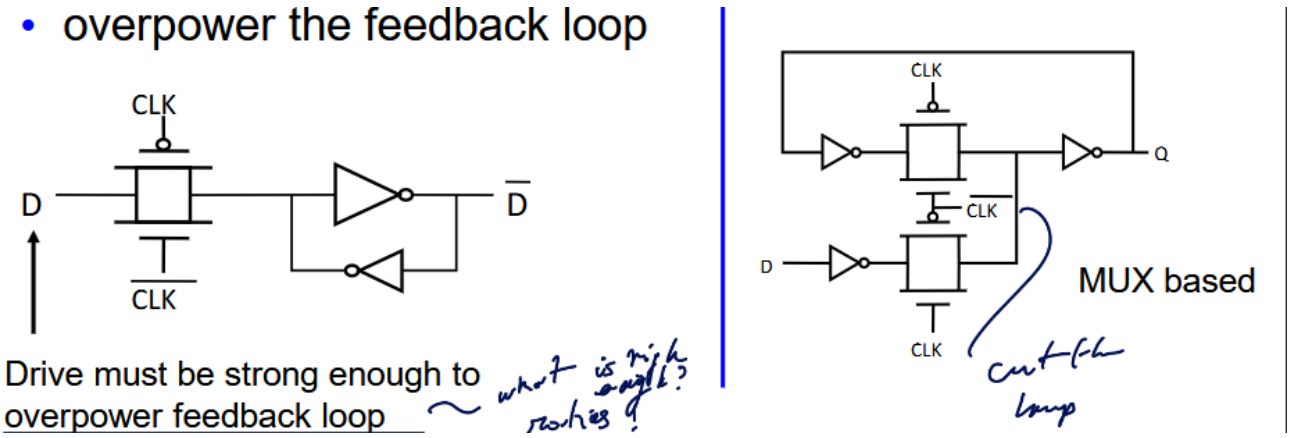


Figure 18: Latches

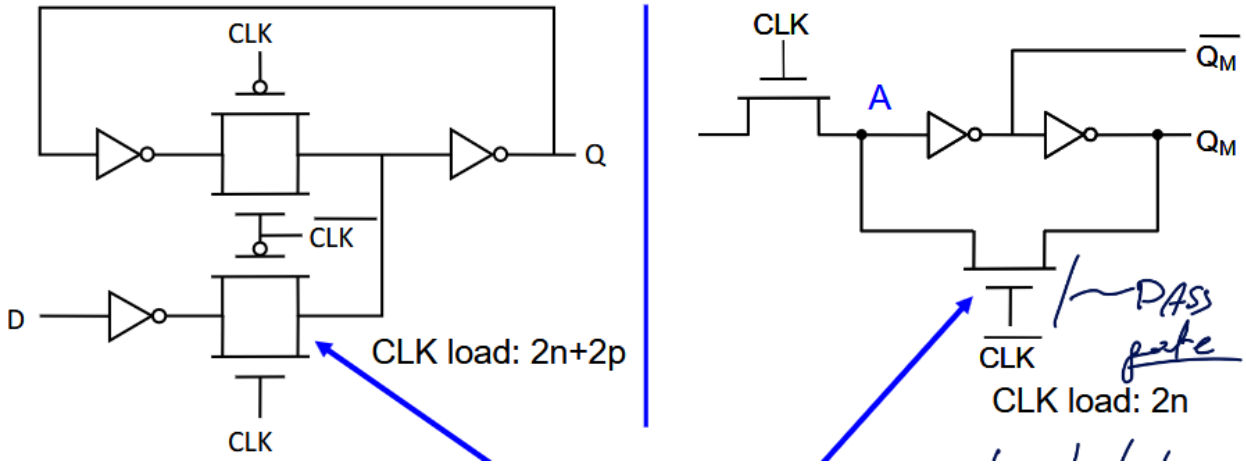
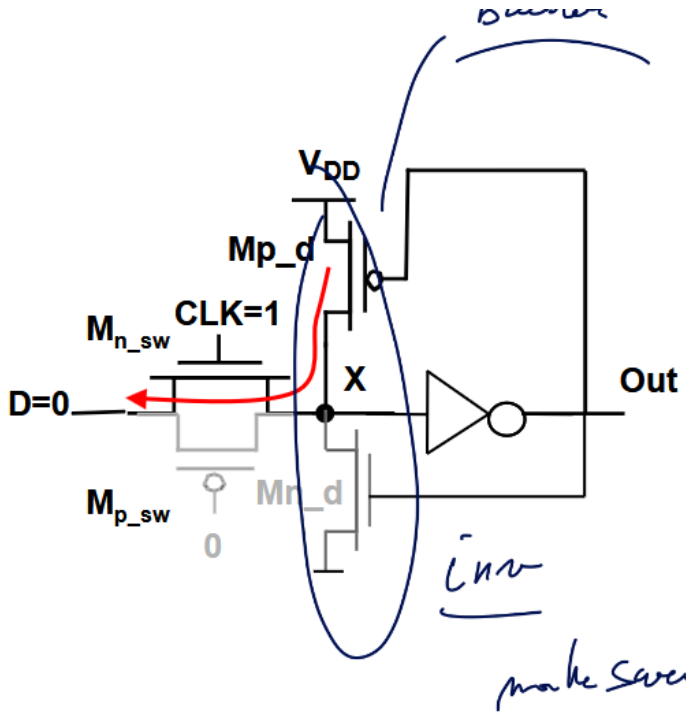
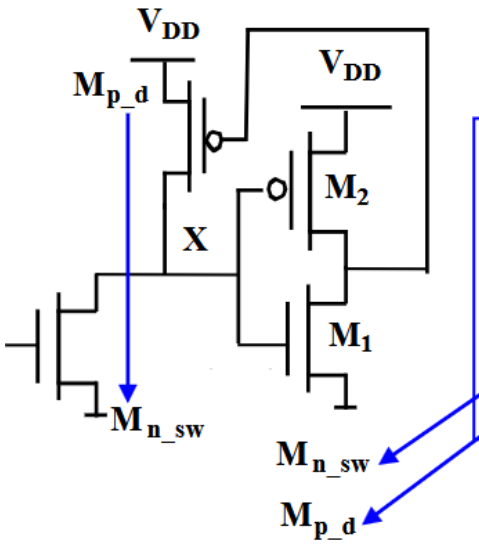


Figure 19: Mux-Based latch : Transmission Gate vs Switch



- Feedback circuit, difficult analysis
- Approximation:
  - Open loop in steady state
  - find the point V_X where the inverter starts to switch
- M_{p_d} and PDN must be sized such that VX is pulled below threshold of invertor M2/M1

Figure 20: David Goliath



- Mn_sw must pull V_X down to $V_{DD}/4$ to ensure switching
- Current in M_{p_d} and M_{n_sw} is equal

$$k_n\left((V_{DD}-V_{Tn})V_X-\frac{V_X^2}{2}\right)-k_p\left((V_{DD}-V_{Tp})V_{DSATp}-\frac{V_{DSATp}^2}{2}\right)=0$$

M_{switch} linear M_{driver} saturated

$V_X = V_{DD}/4$ $V_{DD}/2$ to conservative measure

$$\frac{W_n/L_n}{W_p/L_p}=\frac{\mu_p}{\mu_n}\frac{(V_{DD}-V_{Tp}-V_{DSATp}/2)}{(V_{DD}-V_{Tn}-V_X/2)}\frac{V_{DSATp}}{V_{DD}/4}$$

This is a worst case approach. In practice the bleeder will already be "weakened" because at V_X=V_{DD}/4, the invertor has already begun to switch off.

Figure 21: Sizing of the latch

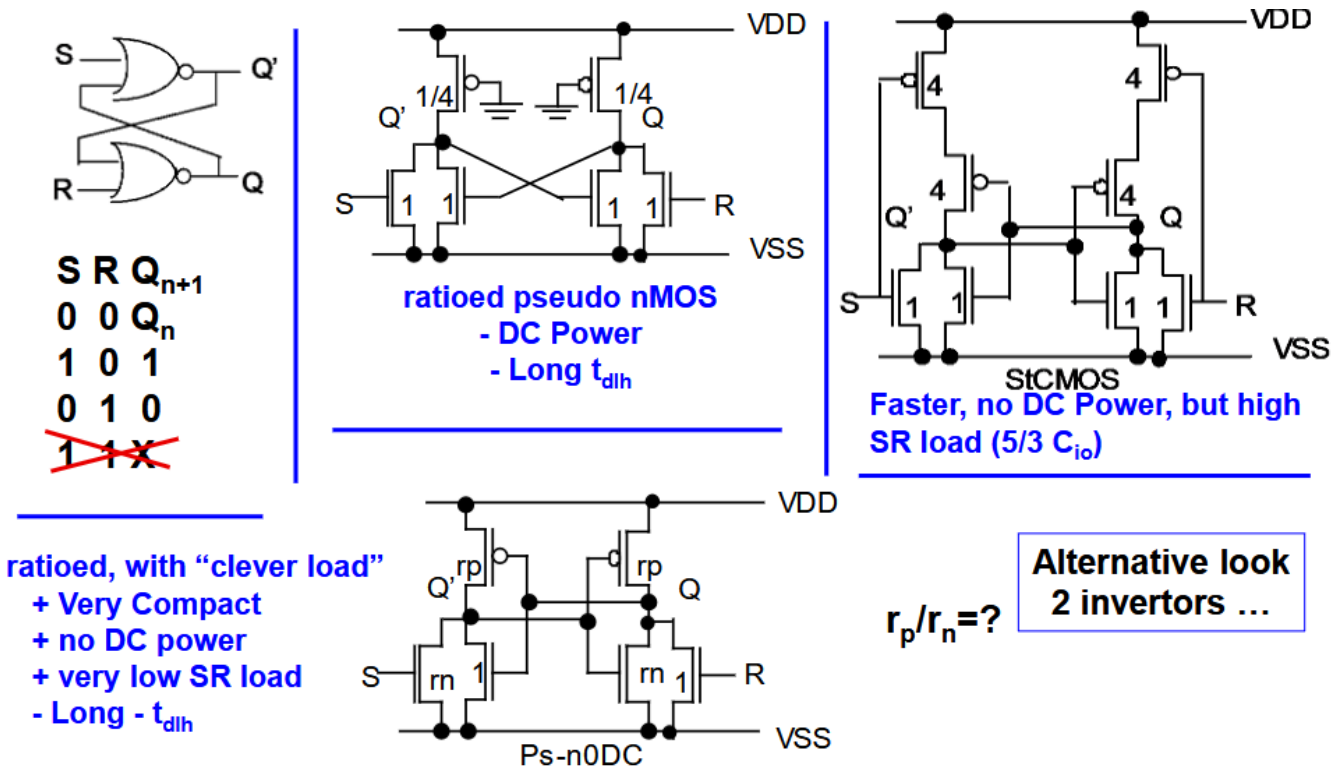


Figure 22: NOR-NOR latch

Sequencing, pipelining revisited

We can also have a *latch based pipelines*, the sequencing is much more soft but we could have race condition if the clocks are overlapping.

Here we can have a 1-1 overlap on both latches are transparent. We can also have some undefined states where a node is driven by multiple nets.

When we have a 0-0 overlap it is like a pseudo static latch since there is no change.

Term	Name
t_{pd}	Logic Propagation Delay
t_{ad}	Logic Contamination Delay
t_{pcq}	Latch/Flop Clock-to-Q Propagation Delay
t_{ccq}	Latch/Flop Clock-to-Q Contamination Delay
t_{pdq}	Latch D-to-Q Propagation Delay
t_{cdq}	Latch D-to-Q Contamination Delay
t_{setup}	Latch/Flop Setup Time
t_{hold}	Latch/Flop Hold Time

For registers

$$T_c > t_{pd} + \underbrace{t_{pcq} + t_{setup}}_{\text{sequencing overhead}} \qquad t_{cd} + t_{ccq} > t_{hold}$$

$t_{cd} + t_{ccq} + t_{nonoverlap} > t_{hold}$

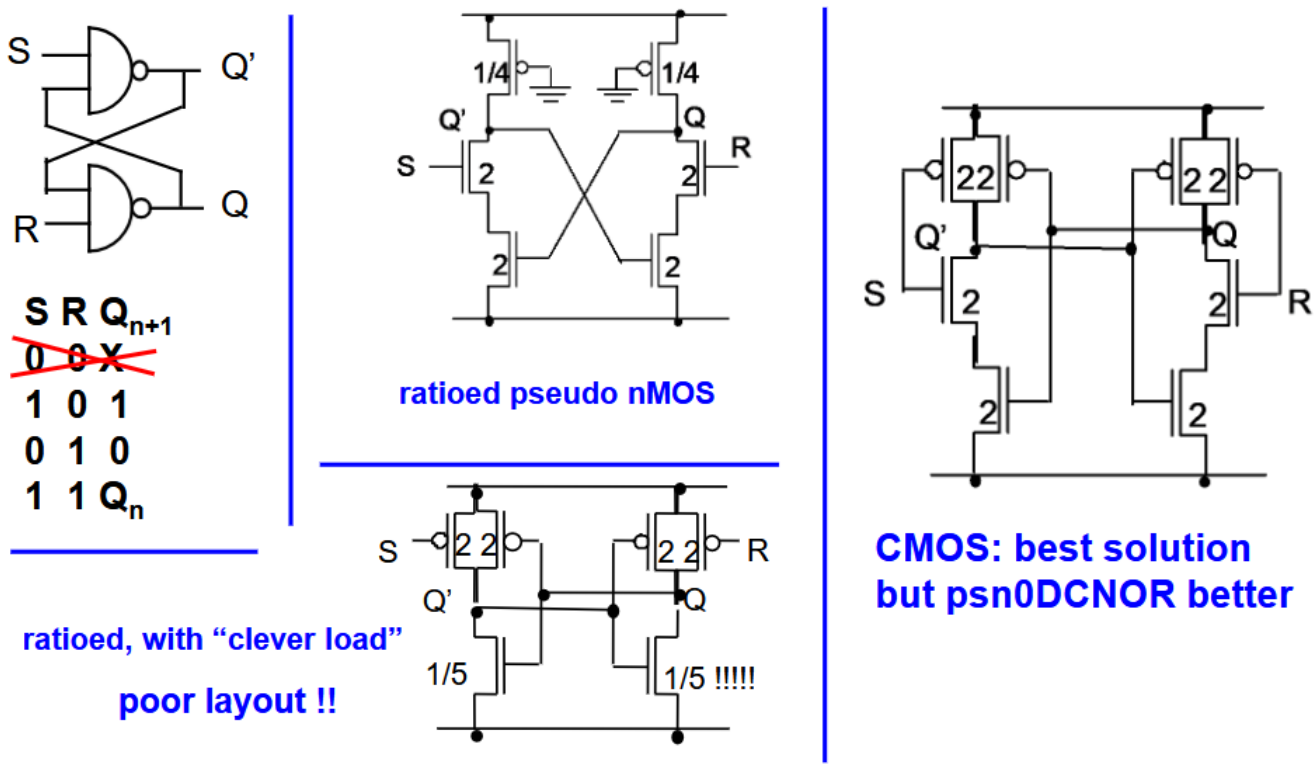


Figure 23: NAND-NAND latch

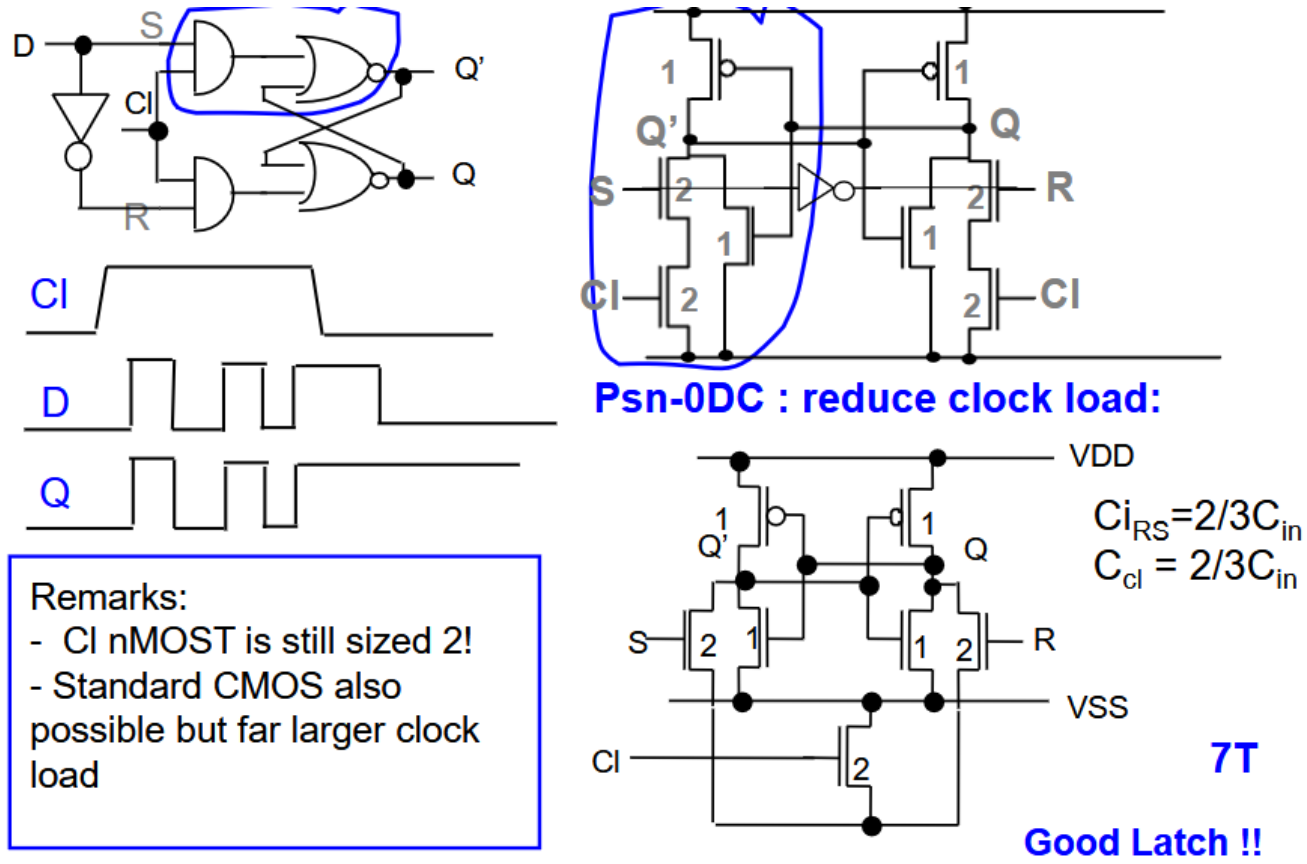
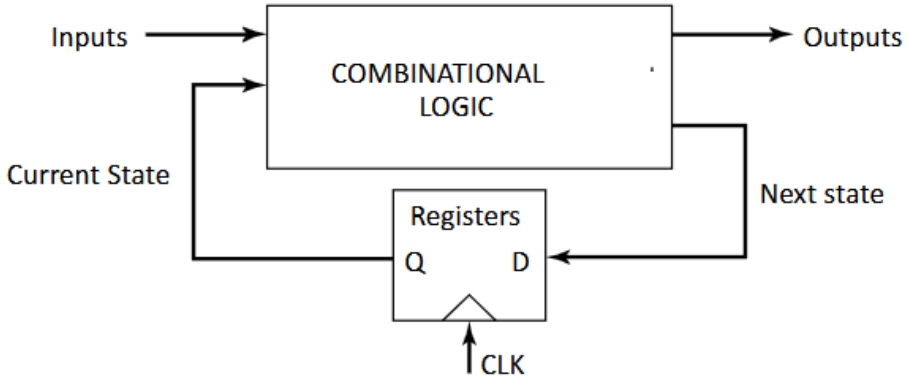


Figure 24: Transparent (n-)latch

- Finite State Machines - Controllers



- Datapath

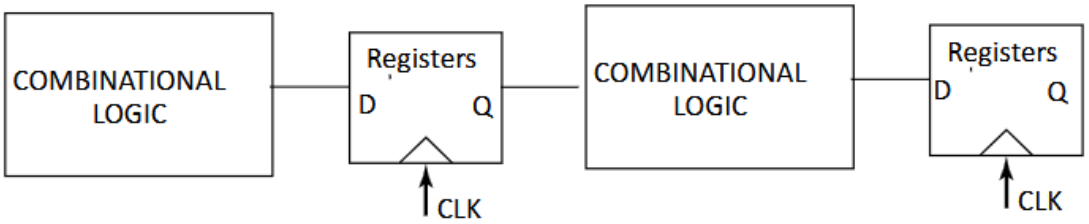


Figure 25: Registers in the system

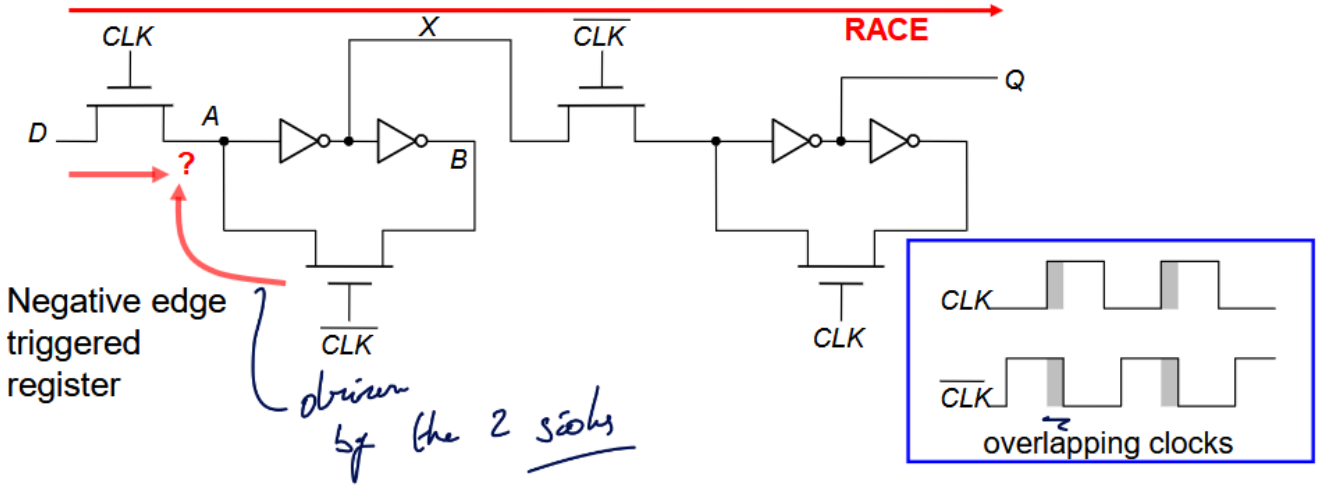


Figure 26: CLK must not overlap

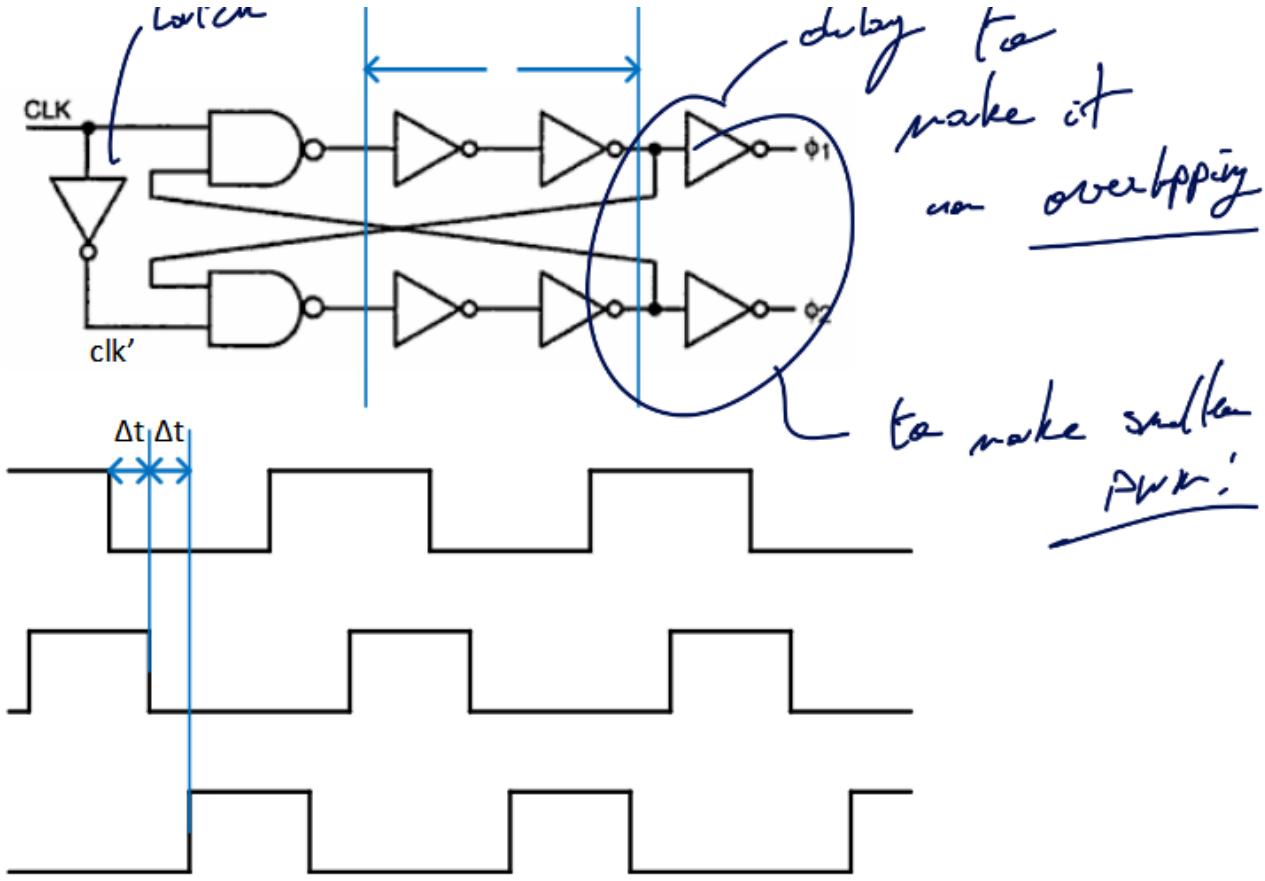


Figure 27: Creating non-overlapping clock

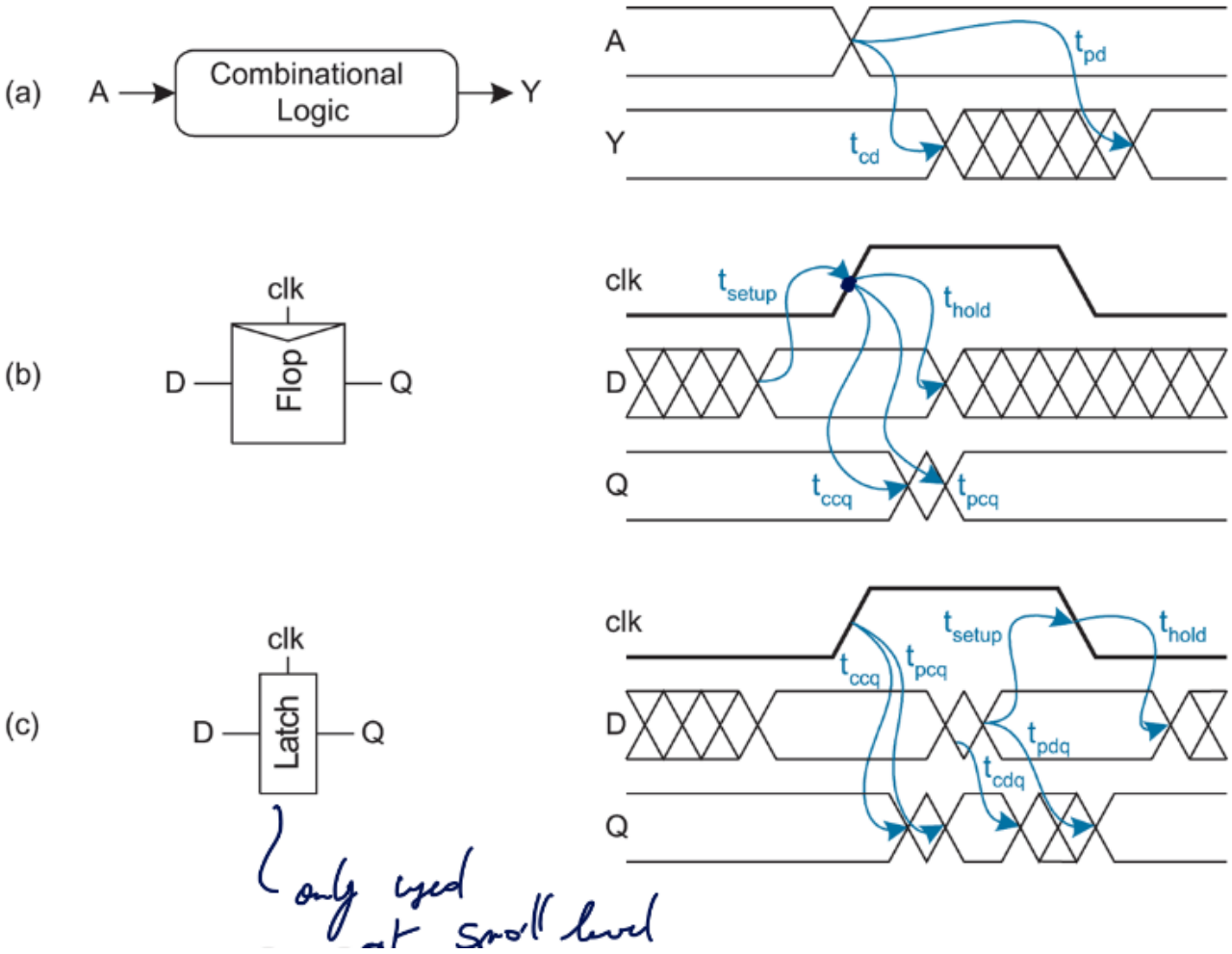


Figure 28: Timing

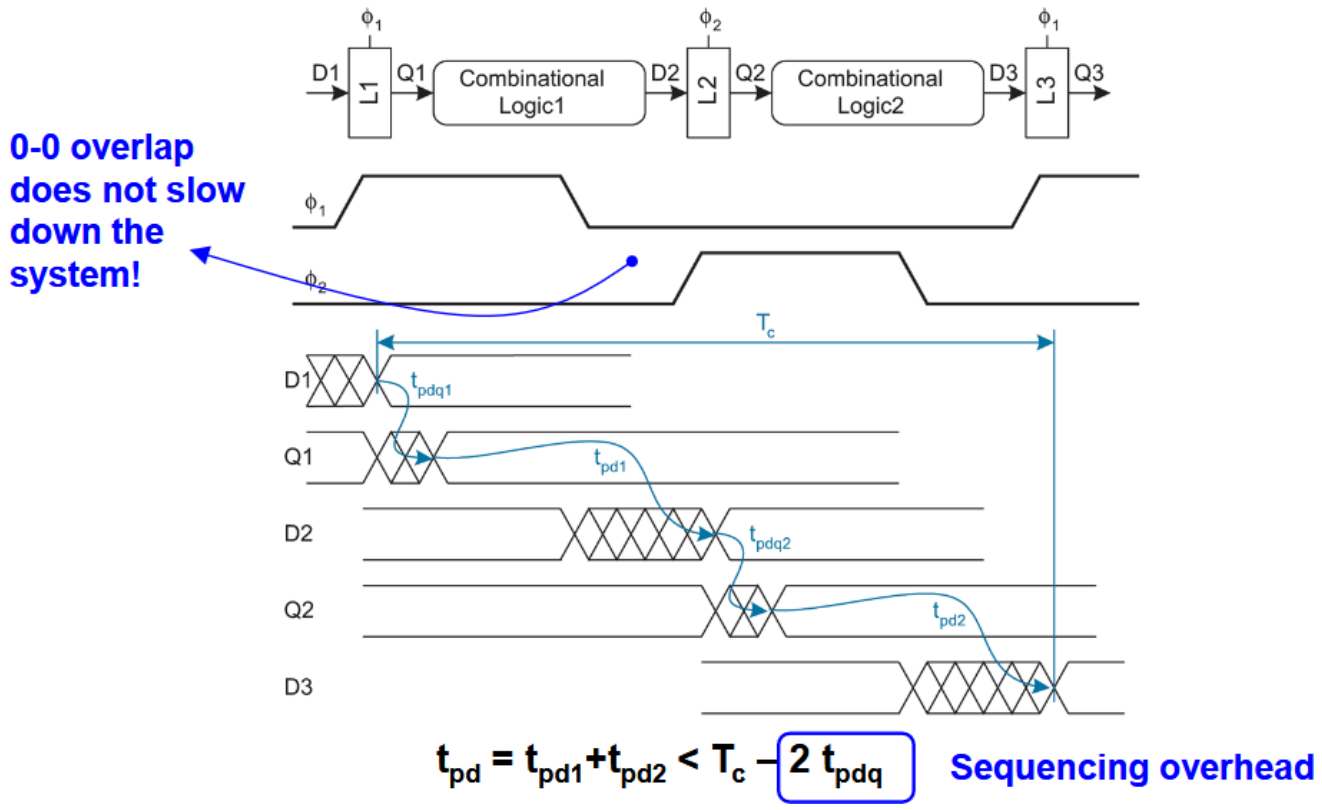


Figure 29: latches, max. delay

In the registers logic, we set the clock speed based on the slowest logic section. But in latch logic, we can do some **time borrowing** technique to take some time from the next cycle. We can't do this if we loop the data over in which case we will have some overlap of processing data.

To compute the maximum allowed borrow time is based also on the setup time :

$$t_{borrow} < T_c/2 - (t_{setup} + t_{non\_overlap})$$

Clock skew

It is the fact the clock will have to propagate to gates and it can take less or more time depending on the travel distance. It is a deterministic phenomena. The statistical one is jitter which is not taken into account in this course.

We can use a **clock tree** structure to balance the clock and to make sure that the path taken for all the components is of the same length. It is not always feasible. It is also influenced by interconnect properties.

If we do latch logic the timing is not impacted by t_{skew} since the *signal has the whole transparent phase to arrive.*

$$t_{pd} = t_{pd1} + t_{pd2} < T_c - 2 \cdot t_{pdq}$$

And for time borrowing we get :

$$t_{borrow} < T_c/2 - (t_{setup} + t_{non\_overloap} + t_{skew})$$

For the validity we have :

$$\text{FF: } t_{cd} + t_{ccq} - t_{skew} > t_{hold} \qquad \text{Latch: } t_{cd} + t_{ccq} + t_{non\_overlap} - t_{skew} > t_{hold}$$

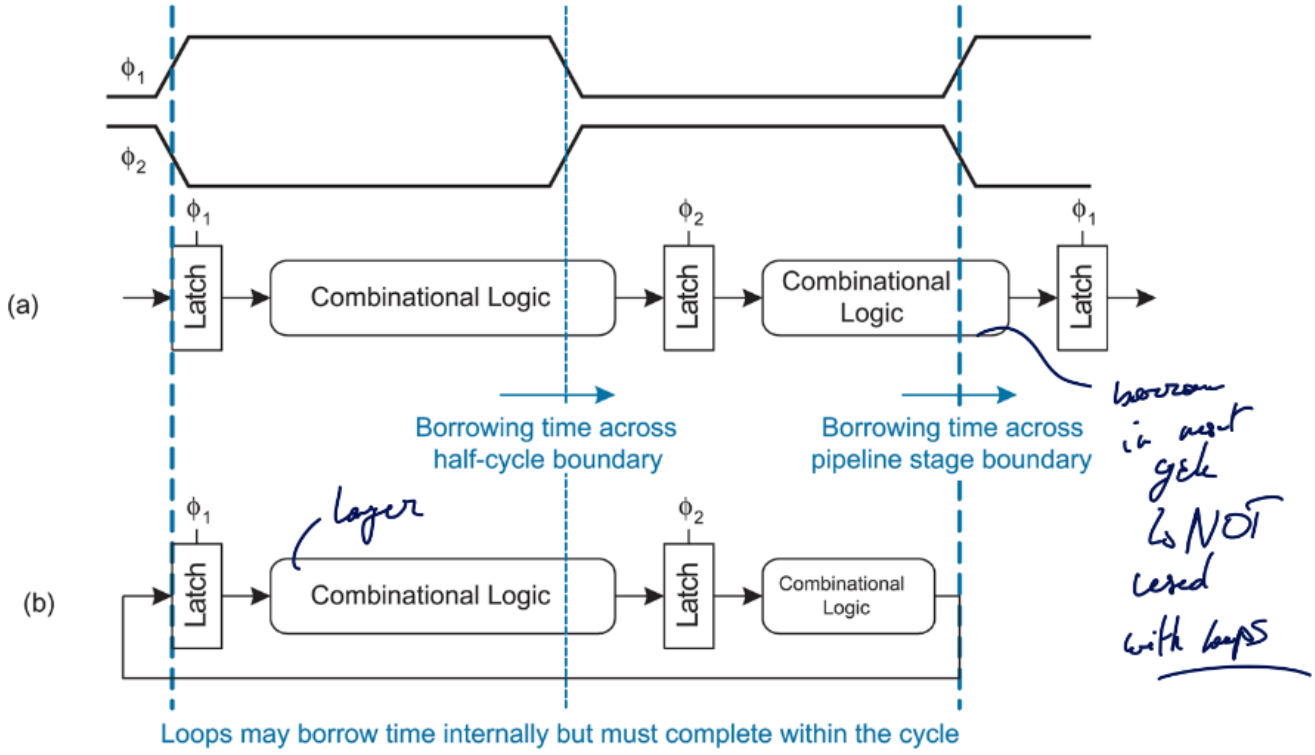


Figure 30: Time borrowing technique

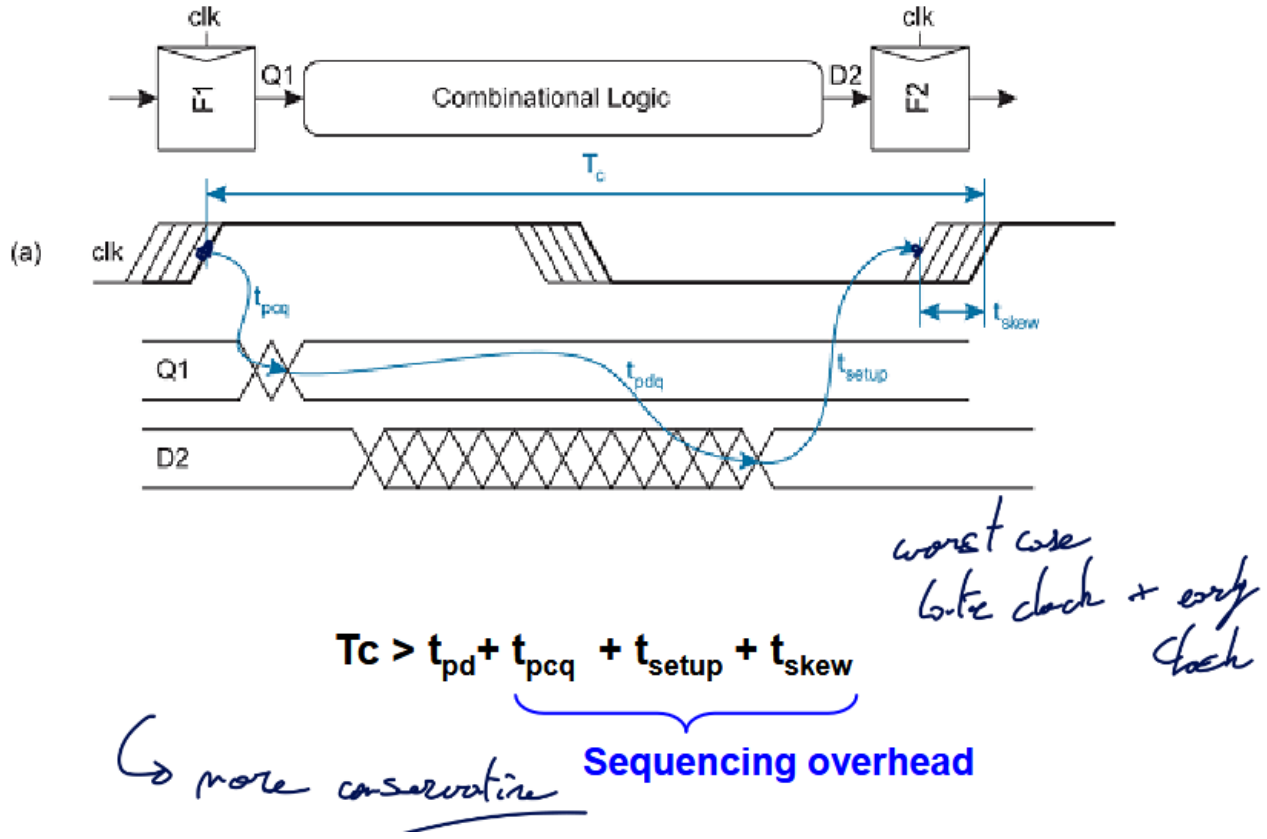


Figure 31: Clock skew effect

	Sequencing overhead ($T_c - t_{pd}$)	**Minimum logic delay** (t_{cd})	**Time borrowing** (t_{borrow})
Flip-Flops	$t_{pcq} + t_{setup} + t_{skew}$	$t_{hold} - t_{ccq} + t_{skew}$	0 (*doesn't exist*)
Two-Phase Transparent Latches	$2 \cdot t_{pdq}$	$t_{hold} - t_{ccq} - t_{non\_overlap} + t_{skew}$ in each half-cycle	$\frac{T_c}{2} - (t_{setup} + t_{non\_overlap} + t_{skew})$

Dynamic logic

The idea behind *dynamic logic* is to charge and discharge a node that has a high impedance. While in static logic the output is either connected to VDD or GND via a low resistive path.

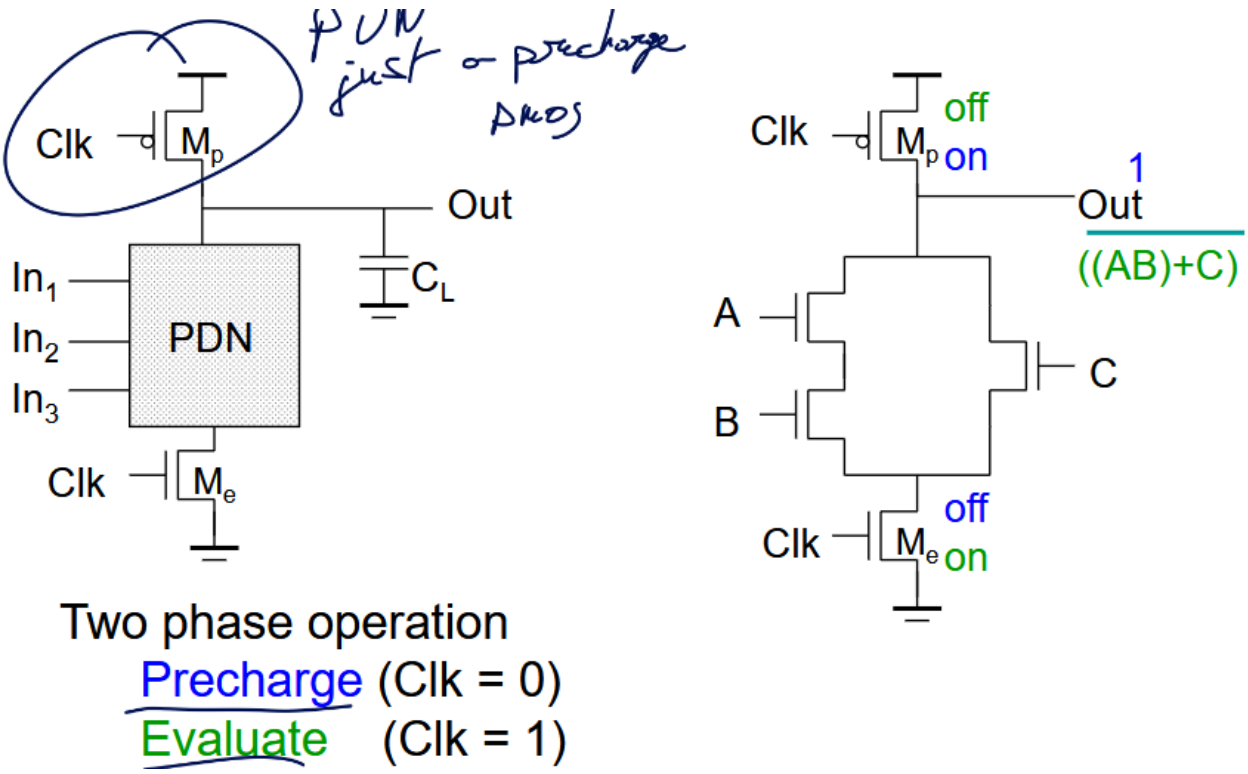


Figure 32: Example of a dynamic logic circuit

We use the low CLK to charge and then evaluate when it is high. So if we discharge by mistake, we will have to wait the next clock cycle. There is 0 or 1 transition during evaluation. We can keep the node high before or after evaluation.

We only need to create a PDN which represent the function we are trying to produce only $N + 2$ (so less transistor compared to static logic $2 \cdot N$). We have full swing outputs. No need to ratio size and we have a faster switching speed since the output and input line has less capacitance. Less logical effort.

But we have a higher power dissipation than static logic. The line is always active and for a continuous set of 0 we need to load and discharge all the time the cap. We have a low noise margin since the PDN will start working as soon as the input signal exceed V_{thn} and so $V_M = V_{IH} = V_{IL} = V_{thn}$.

More over, there is some diode on the substrate that causes leakage (*junction leakage of diode in reverse bias*). We also have some *subthreshold leakage* which is dominant.

I we don't compensate for the leakage, we force a higher clock frequency to avoid falling in the gray zone. We however compensates for this with using a bleeder or level restorer. This will induce some extra static power consumption.

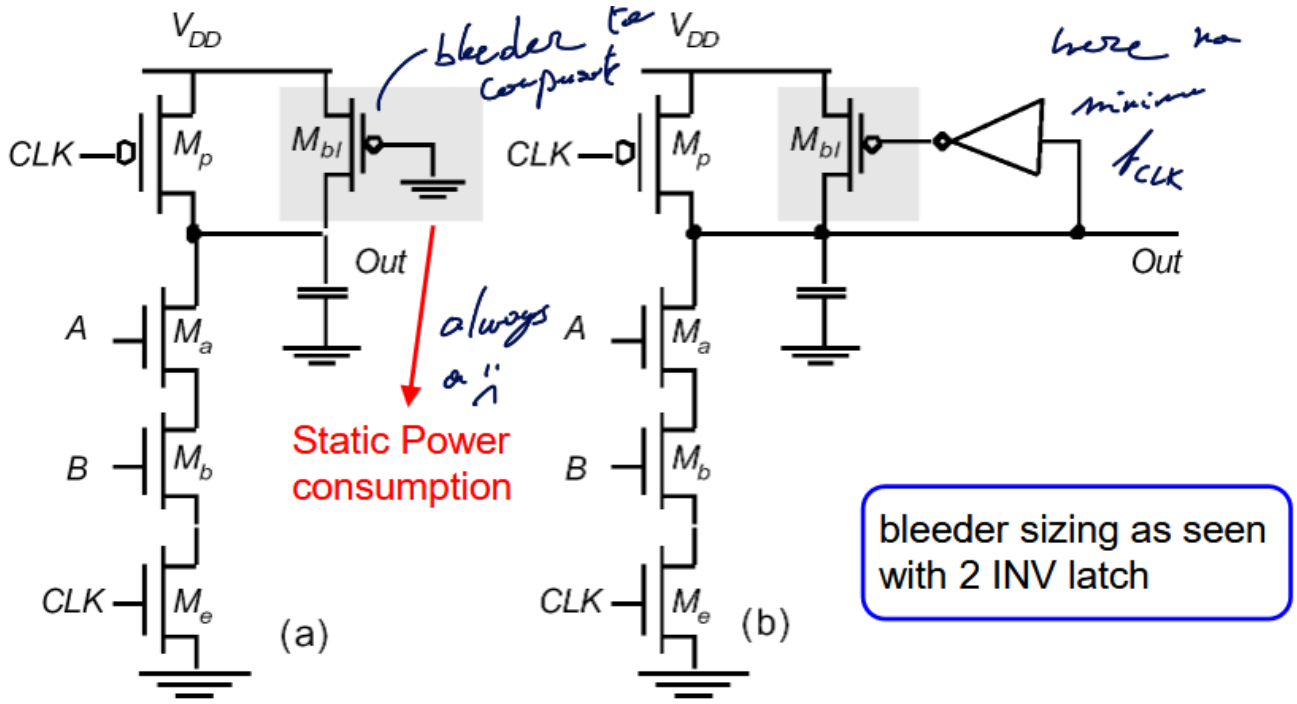


Figure 33: Bleeder and Level restorer

Charge sharing Something that can happen due to noise, is that the transistor A will discharge by mistake C_{out} and the charge will go to the node between A and B. The charges will split between those two cap reducing the output voltage and have a higher noise sensitivity.

One way to avoid this is to precharge the internal nodes. But we need more transistor, more energy and it will increase the C making the circuit slower.

Charge coupling

It is also pretty sensitive to charge coupling. So any wire-to-wire crosstalk can be disastrous. There is also the backgate coupling or output to input coupling that can be problematic. We should also avoid clock feedthrough and overshoot.

Danger: charge injection in the substrate. Charge can be collected by HiZ node or lead to latch-up.

Latch-up It is an annoying phenomena where the substrate of MOS transistors has a parasitic thyristor junction we can cause latch-up and put the transistor in an unwanted state. To reset, we need to cut-off the power of the circuit and start again. (*More info: Weste N., e.a. "Principles of CMOS VLSI design – a systems perspective. Second Edition", Addison Wesley, 1993*)

Cascading dynamic logic

The finite propagation delay from in to out1 will cause a partial discharge at out2 which can again makes the output 2 invalid resulting in unwanted behavior.

Domino logic To avoid this we can add an inverted between the two stages. So any transition from charged to discharge or keep will result into a valid state. The static invertor is like a buffer and we need some extra logic to restore the correct output.

We can add some skew to the inverter since the only critical transition is the $1 \rightarrow 0$ one. We reduce the impact impedance. Only non-inverting logic can be implemented ! So we either need to do some logic transformation (not always possible) or use **dual rail domino**.

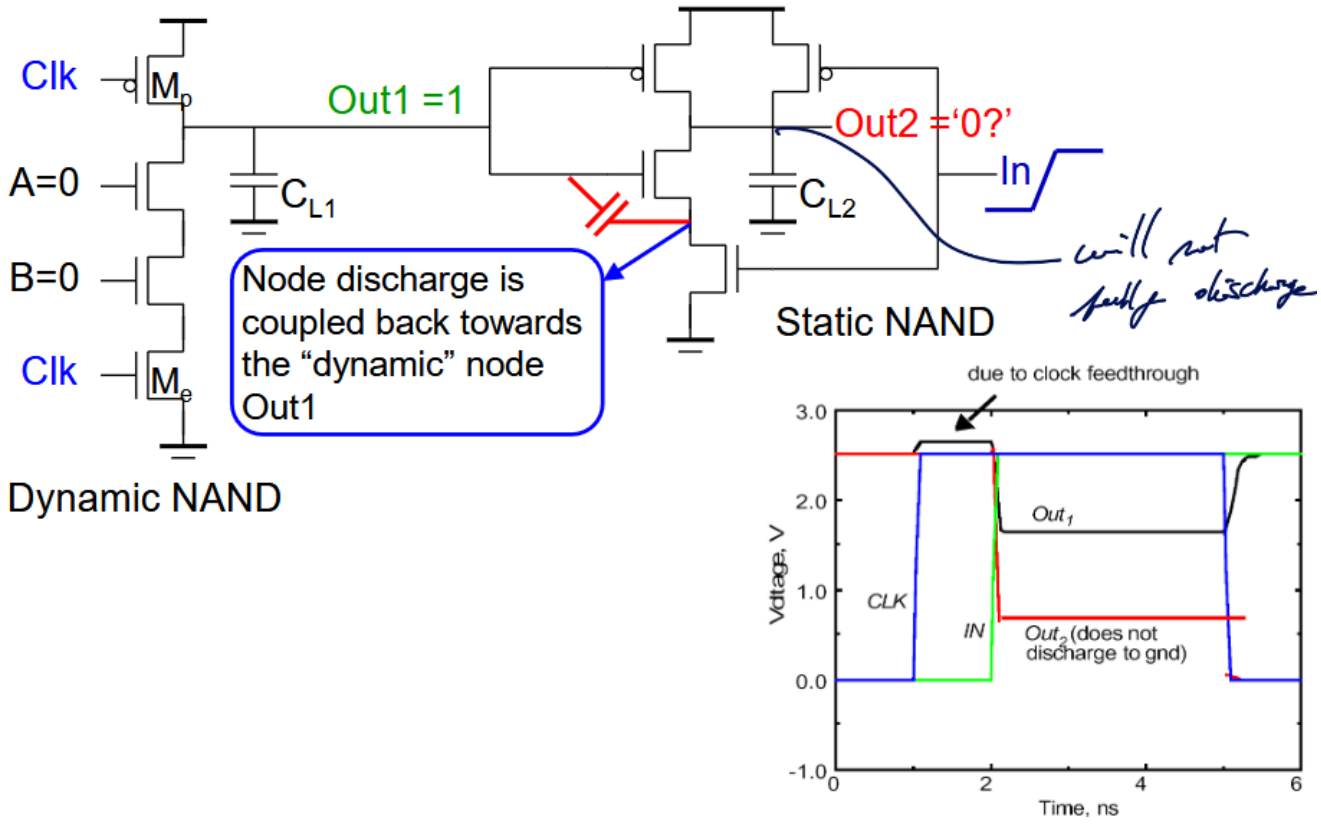


Figure 34: Output to Input coupling

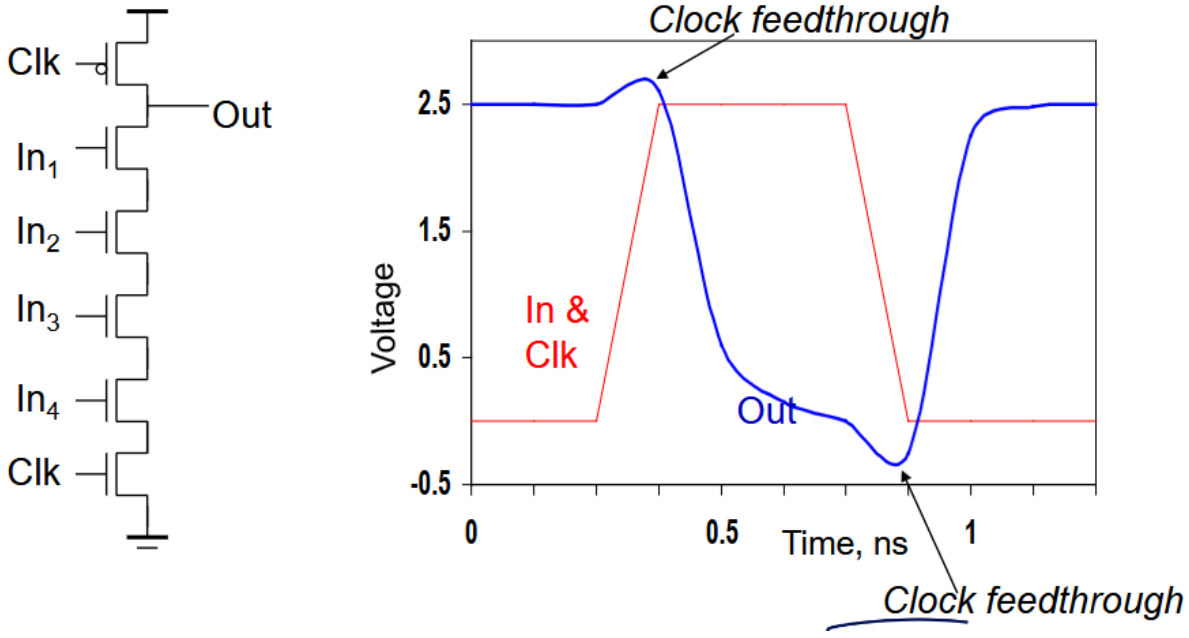


Figure 35: Clock feedthrough

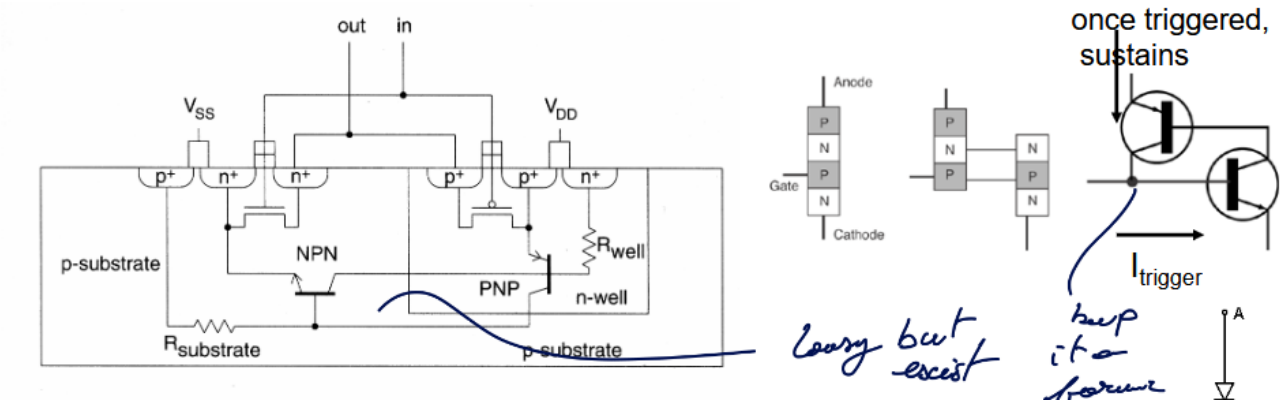


Figure 36: Latch-up

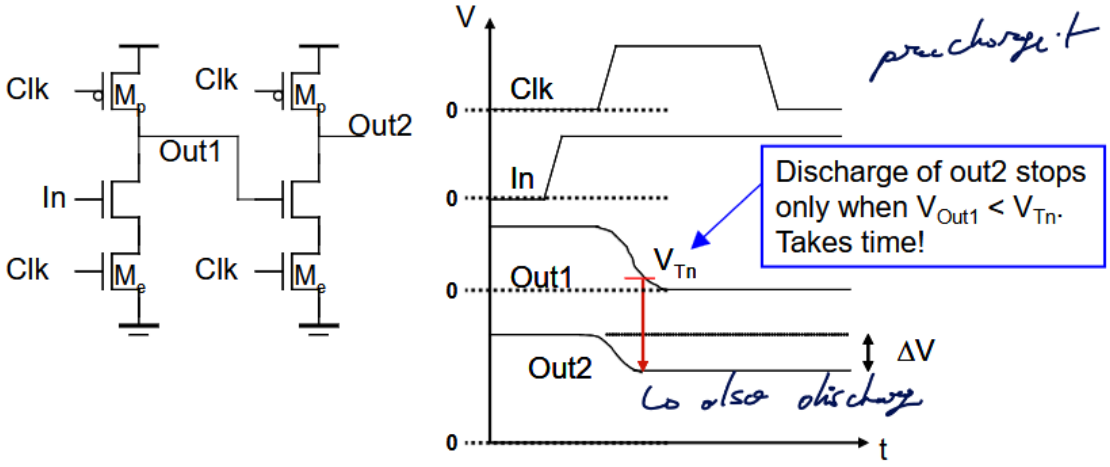


Figure 37: Cascading dynamic logic

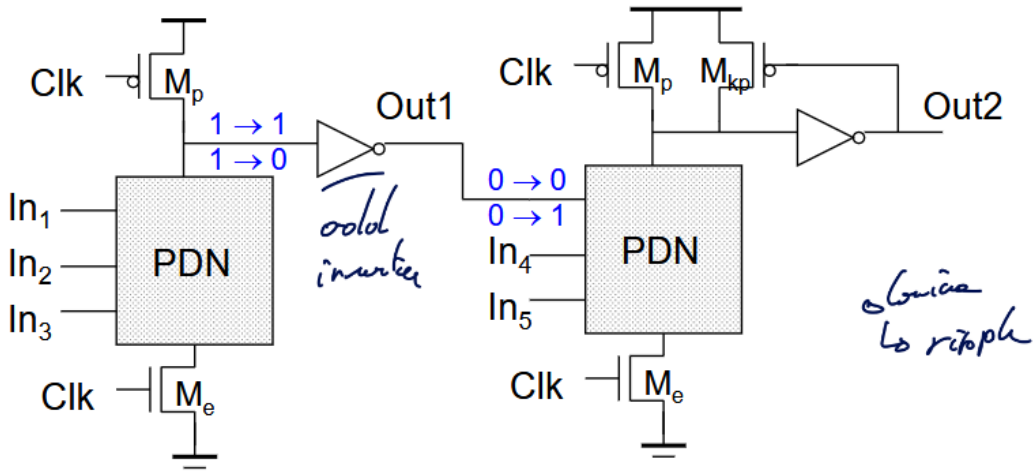


Figure 38: Domino logic

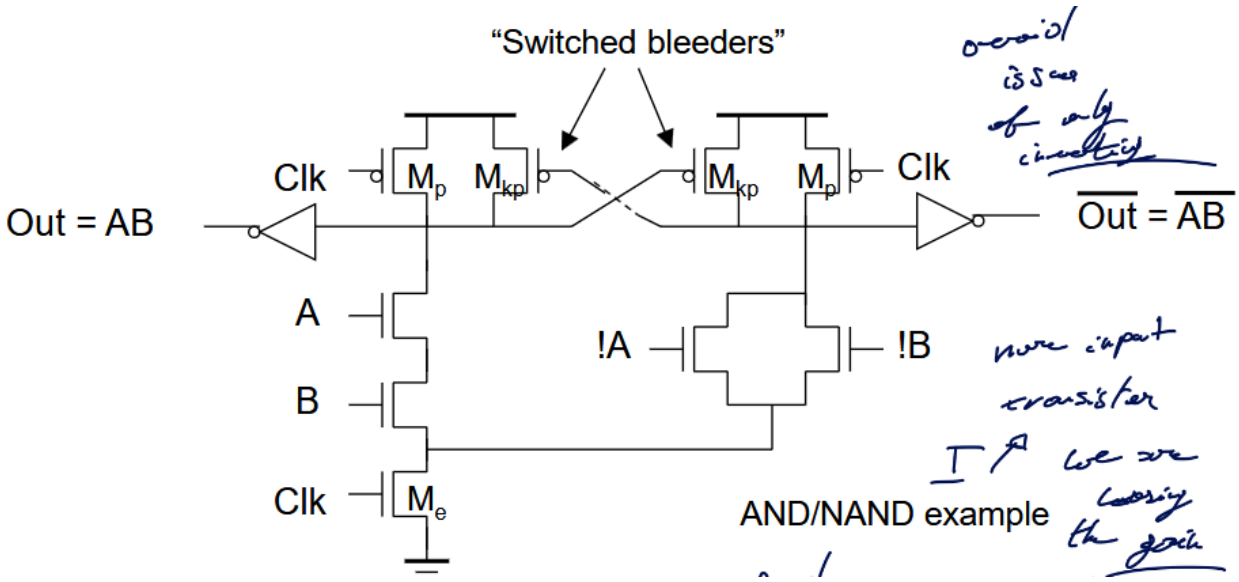


Figure 39: Dual rail domino logic - using switched bleeders

Dual rail domino logic We need to construct 2 PDN one where we do $F(A, B, C) = (A\&B)|C$ or any function and then its NOT version $\overline{F(A, B, C)}$ using *De Morgan*'s theorem. So only one branch will switch on and only one side will make a transition. But it comes at the expense of area and continuous switching no matter the result.

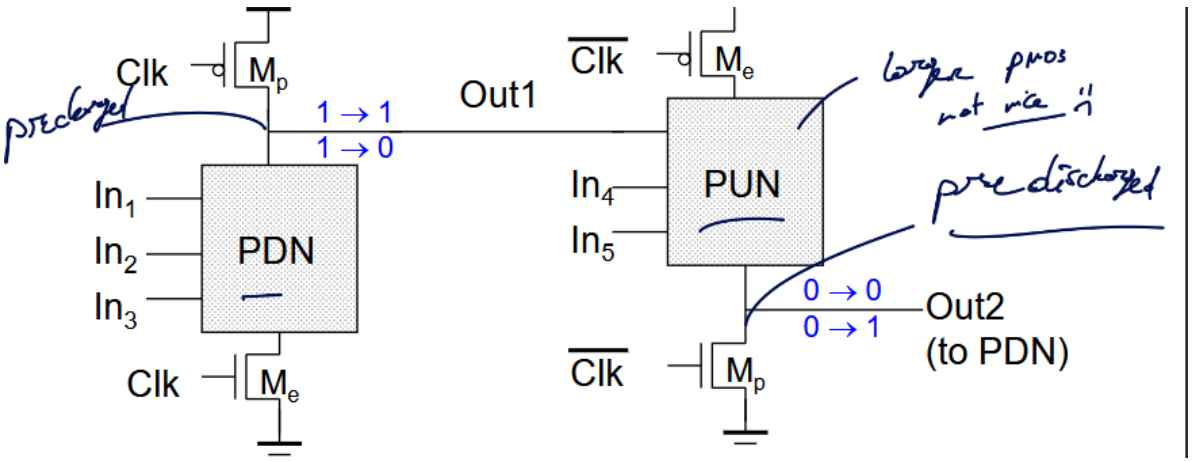


Figure 40: Alternative

Alternative to cascade of dynamic logic A good alternative is to switch between PDN and PUN networks so the $0 \to 1$ transition are allowed at inputs of PDN and $1 \to 0$ transitions are allowed at inputs of PUN.

Dynamic edge triggered register We use a capacitor as a storage source that we need to refresh. We can't simply halt the clock. It is sensitive to $0 \to 0$ and $1 \to 1$ transition and to clock overlap. But we have good performance

Clocked CMOS or C^2MOS

Now it is no longer sensitive to clock overlap just need a fast enough rise and fall times.

True Single Phase Clock (TSPC) logic Here we only have 1 clock phase so easy to distribute the clock and no overlap by construction. At first when $CLK = 1$ we have like 2 invertors and so the latch is transparent. But when

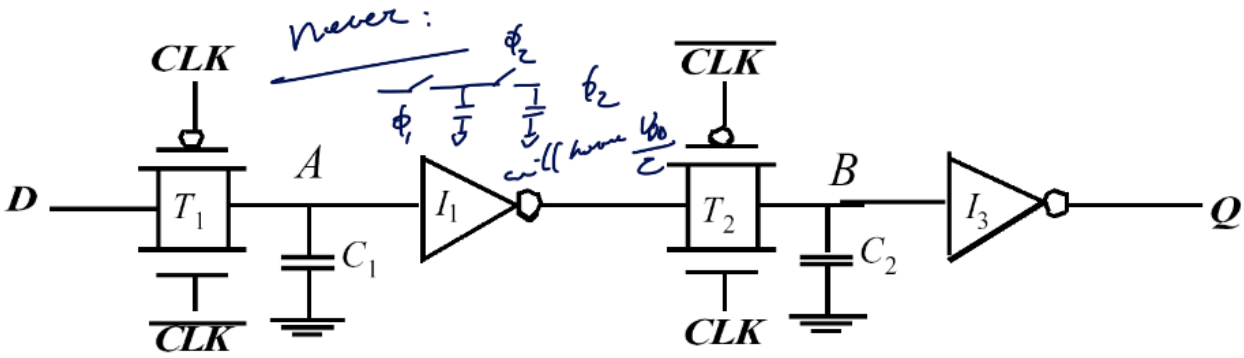


Figure 41: Dynamic edge triggered register

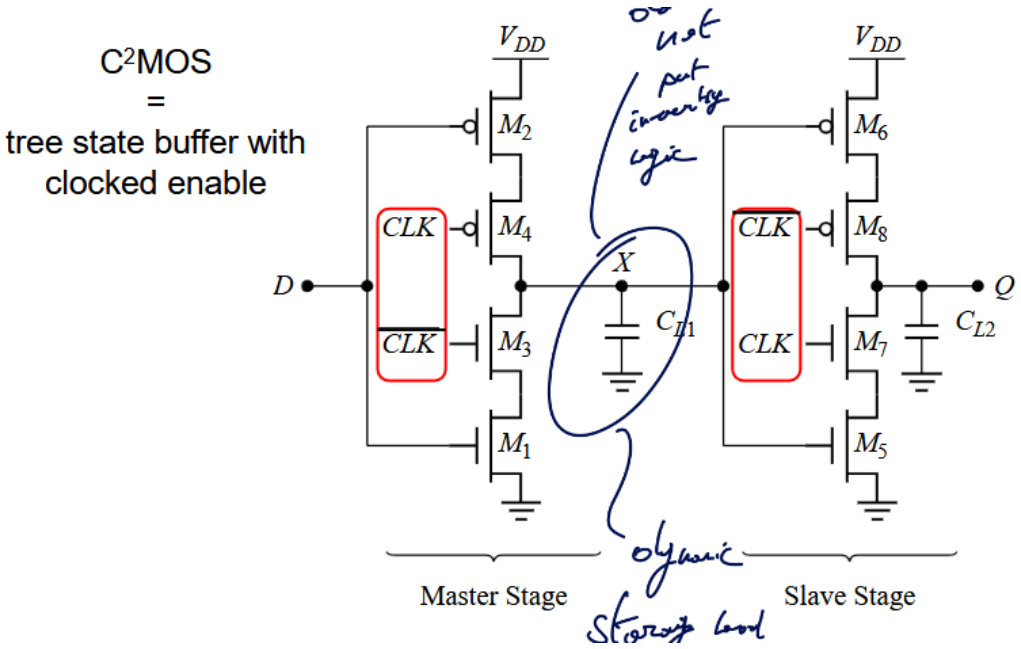


Figure 42: C^2MOS

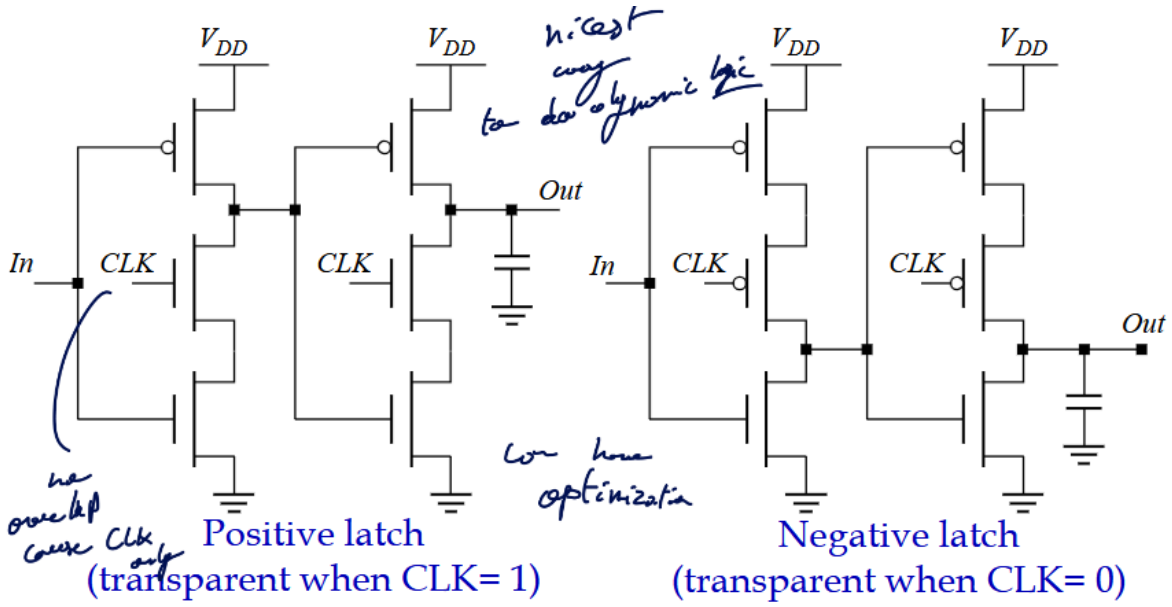


Figure 43: TSPC logice

$CLK = 0$, the charge can't be pulled down and if we have $In = 0$, the output won't change since the inverter will stop the propagation of that value.

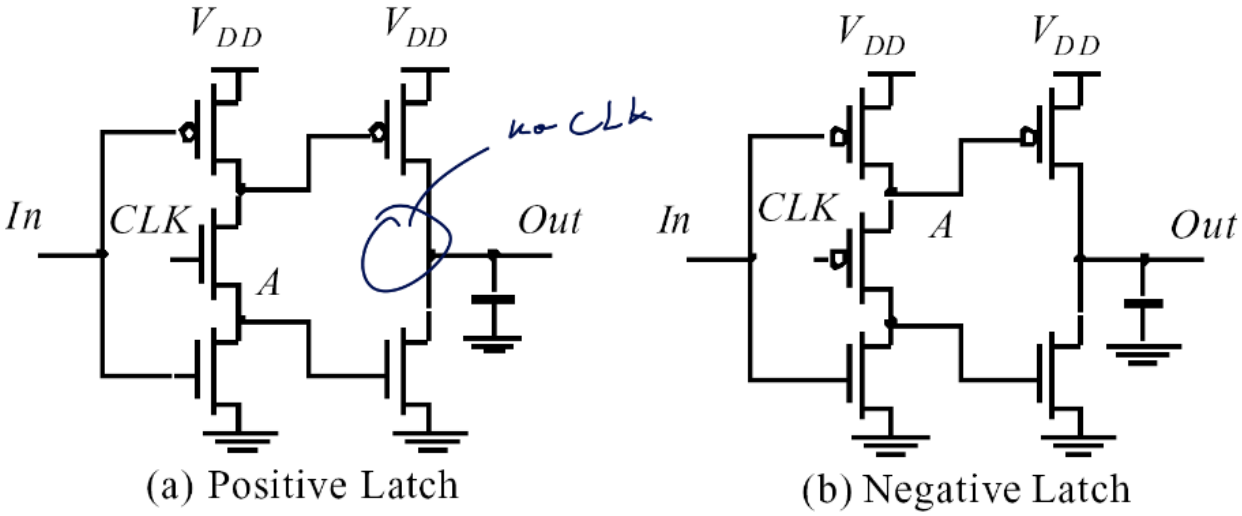


Figure 44: TSPC simplification - split output

It has less TOR and clock load but the split outputs don't have full swing. We have less drive, less VDD scaling possibility.

Conclusion

- Due to its high impedance nature, design of dynamic circuits is tricky and requires extreme care at the circuit level.
- Hard to automate in a synthesis – P&R flow based on static CMOS standard cells
- Power consumption of dynamic logic is usually higher.
- Nothing is as easy as standard CMOS...

5 - Production Test

We cannot build a working chip that is not testable, no one would buy our chip if we can't demonstrate that a specific die is working. This is why we need production test, to demonstrate that each of our sample is working as expected.

Introduction: what's the problem

Let's dive in.

Ad hoc versus structured test

We have two type of cost:

- Non recurring cost: development cost, engineer salary
- Recurring cost: testing the chip, the time it takes us, the reduced throughput

We need to increase the *observability* and *controlability* but adding more pins is expensive for IC, so we need to use a **scanning technique** where we load in a buffer we can observe all the data. We can also add more controlability by using muxes that are driven by a test mode signal. This a **ad hoc method** of testing.

But we don't always have this available. We don't always know how everything works.

Structured test

We could just enumerate all possible state, But with n inputs and m states, we have 2^{m+n} tests, which quickly explode and we can't just test all possible combinations.

Structural testing

It is based on the fault models and knowledge of the circuit structure. We can predict or check for specific defect and use the **Automatic Test Pattern Generation (ATPG)**

Design for testability

TOOD

Summary

TODO

6 - Low Energy Design

We have some fundamental equations related to power and energy:

$$E_{dyn} = QV_{dd} = CV_{dd}^2 \qquad P_{stat} = V_{dd}I_{leak} \qquad E_{stat} = V_{dd}I_{leak}t_d$$

Fight the battle at all levels

$$P = \alpha CV_{dd}^2 f_{clk} + I_{subth}V_{dd}$$

The most simple idea is to:

- Kill activity α
- Reduce cap C
- Reduce V_{dd}
- Clock slower f_{clk}
- Less leakage I_{subth}

Activity factor is a software dependent value and analysis tools can use statistic to estimate the switching probability but it isn't perfect.

In this course we focus on circuit design and architecture. So we can play with V_{DD}, use more or less parallelism, do some signal and clock gating, ...

All this level of abstraction can sometimes obfuscate the fact that two circuits can have different power performance. It can be hard to locate and to monitor it with the complexity of modern tools.

(micro)Architectural choices

Due to combination logic and the fact that we may go through multiple level of gates, we can have some glitches: so an unwanted, unnecessary transition that is synonym of wasted energy.

Typically, this glitch issue is present in Ripple Carry Adder (RC Adder). To avoid it, we can adopt a **tree structure** to balance the delay path and so to avoid unwanted transition. We can think of the *Log adder*.

The **logic depth** is the source of this glitch and so pipelining will help it.

Signal Gating

To avoid such glitches or overall unwanted transition, we can do signal gating and avoid signal propagation. But we need more circuitry and power, so it is a trade-off and optimization problem.

Signal gating is only interesant if we toggle the gate less often than the signal that we block or let pass. Even better is sharing a control signal between multiple part of the circuit. We usually uses clock, data buses, ...

Guarded evaluation One implementation of this idea is **guarded evaluation** where we will trigger the evaluation only if we know we want its result next clock cycle.

Precomputation Here we check the if condition earlier and avoid propagating useless inputs.

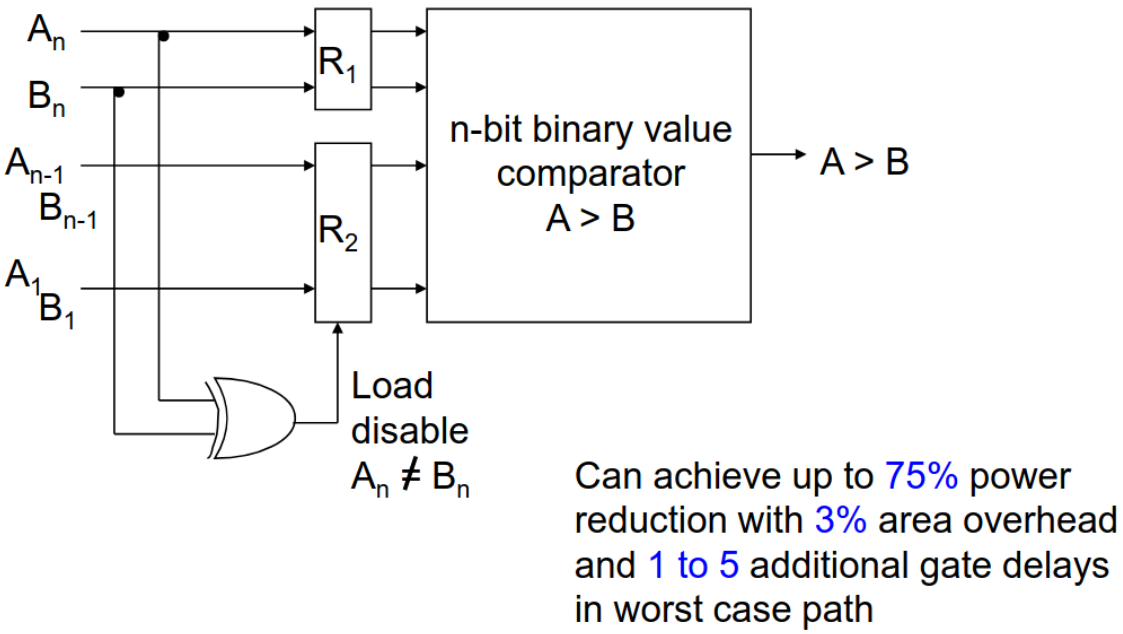


Figure 45: Precomputation

Clock gating Can be done automatically and efficiently but introduce *skew* and *testability problems*.

Dilemma

The biggest dilemma here is the trade-off between more hardware and slower speed which is better for the *dynamic POV*. But more hardware means more leakage which is bad for *static leakage*.

If we lower the supply near V_T we know that:

$$I_D = I_0 e^{\frac{V_{dd}-V_T}{nkT/q}}$$

So we will still have some current ... We can reduce the supply but we will have lower speed, less difference between I_{on} and I_{off} and more sensitive to variation. The speed isn't an issue for many application in reality and luckily enough, products that run at those frequency usually want low energy usage. Win-win situation for once.

But let's not get overwhelmed and keep in mind the actual FoM is E/op not who got the smallest supply. In fact, going to low will at some point increase the current consumption at some point.

Hard to quantify the *sign off* and it changes from one netlist to the other quite drastically.

The plumber's manual

Power gating

Usually, we want to cut-off the logic that is no longer in use. We will implement the logic in **Low VT** and the switch network in **High VT** because:

- **Low VT**: high leakage but good performance
- **High VT**: low leakage

But when turning off the logic, we still want to keep the results of our previous calculation. For this, we need a **keeper** that needs low leakage.

Also, switching the network on and off is not a good thing and we will have a penalty due to the charging and discharging of the cap C_p and C_n.

Variable Threshold CMOS

V_T isn't fixed as seen in DIAC, we can model it with:

$$V_T = V_{T0} \pm \gamma \left(\sqrt{2|\varphi_F - v_{BS}|} - \sqrt{2|\varphi_F|} \right)$$

So changing v_{BS} will bias the V_T. We call this Reverse Body Bias.

Issues

- γ becomes smaller with scaling
  - Need more negative PSS which is costly and not always reliable
- Substrate is a large cap, so pretty slow and costs energy
- V_T has an impact on performance

FDSOI is one solution that is talked about in the next chapter.

Forward body bias Used here to increaser performance but we can go up to a certain point or we won't reverse the junction properly. It is quite complicated to control and should only be used where performance is absolutely critical.

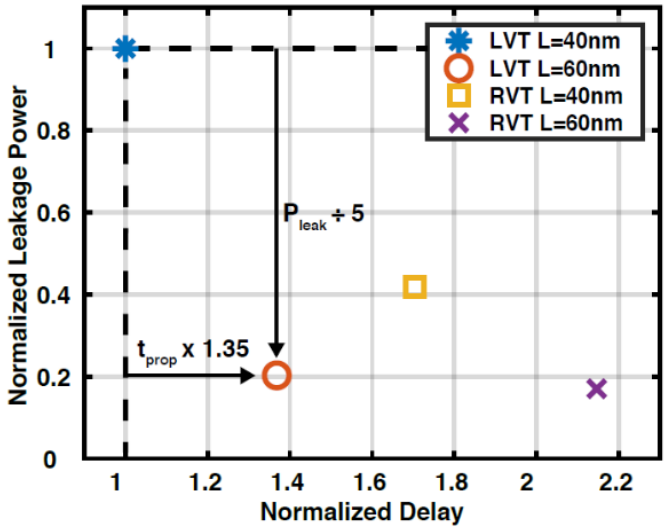
Multiple Threshold CMOS The best thing to do is to mix technologies of CMOS and use sometimes low or high V_T transistor.

Long-Le transistors

Another technique is to have longer length of transistor by about 10% compared to the nominal length. They are 10% slower but have **3 times lower leakage**. So whenever we have a path that have some slacks we will replace the transistor by this type of transistor.

One common practice, is to start designing with long Le transistor and then when we want to reach timing, we will reduce its length for the critical path.

Such Le transistors are often more appreciated as they don't require multiple power supply and is more effective.



[7]

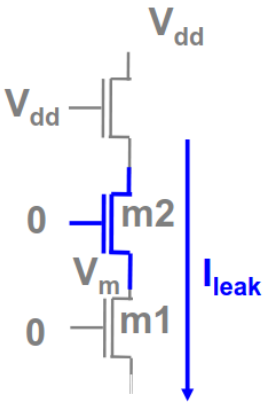
RVT is not that interesting after all

Figure 46: Techniques and their effects

Stacking effect

If one drain lets water out why not building another one a lil further down the river ?

That's what they thought when introducing this technique or something similar. It can drastically reduce leakage but will gives us less headroom for our design:



$V_m(I_{leak}) \sim 100mV$

I_L determined by transistor m2

$V_{GSm2} = -V_m$

$$I_{leak} = I_o \exp\left(\frac{-V_m - V_T}{nkT/q}\right) = I_{leak0} \exp\left(-\frac{V_m}{nkT/q}\right)$$

30..40 mV

Figure 47: Stacking effect

We can have some natural stacking (because of the nature of the logic) or *forced stacking* where we will split one NMOS into two parts to reduce leakage.

Standby vector

One last technique is to use *standby vector* that will force a 1 or 0 in standby mode.

Sizing and multiple supply voltages

We saw in platforms how minimizing delay and energy is an optimization problem where we need to find the best tradeoff. On top of that, we need to stay between the rail, have a large enough transistor and respect V_T.

Energy Delay Sensitivity

To compare various technology or implementation we can take a look at the sensitivity where:

$$S_A = \left.\frac{\partial E/\partial A}{\partial D/\partial A}\right|_{A=A_0}$$

It is a powerful tool as it indicates the *effectiveness of changes* in design. With this, comes a lot of theory about optimality but one thing that is important here is the **pareto-optimality**.

Pareto-optimal

> the best that can be achieved without disadvantaging at least one metric

So basically, we can reduce one metric without raising another one. The two curves have various sensitivity.

$$\Delta E = S_A \cdot (-\Delta D) + S_B \cdot \Delta D$$

Circuit analysis of Energy and Delay

When analyzing i and $i+1$ logic gates we can find the delay to be (like in DDP):

$$t_d = \frac{K_d V_{DD}}{(V_{DD} - V_T)^\alpha}\left(\frac{\gamma C_i + C_W + C_{i+1}}{\gamma C_i}\right) = \tau_{nom} : \left(1 + \frac{C'_{i+1}}{\gamma C_i}\right)$$

$$E_{dyn} = (\gamma C_i + C_W + C_{i+1}) \cdot V_{DD,i}^2 = C_i(\gamma + f'_i) \cdot V_{DD,i}^2$$

Where $C_i = K_e S_i$, $f'_i = (C_W + C_{i+1})/C_i = S'_{i+1}/S_i$, giving us:

$$E_i = K_e S_i (V_{DD,i-1}^2 + \gamma V_{DD,i}^2)$$

We can then derivate the sensitivity to sizing:

So the more a gate consumes, the more sensitives it is to sizing.

We can do the same thing for the sensitivity to the power supply:

Most sensitive to energy and fast gates. Maximal for $V_{DD,max}$

Example Chain of inverter

Ultra low voltage design examples

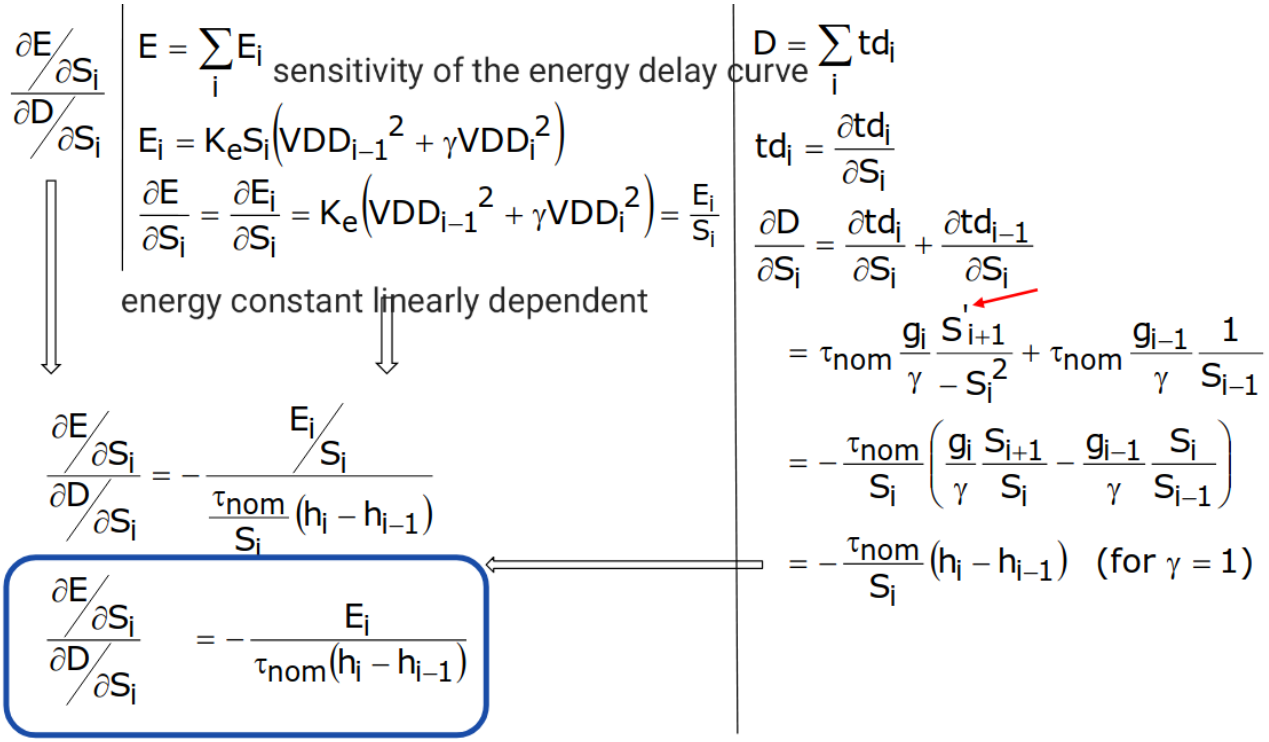


Figure 48: Sizing sensitivity

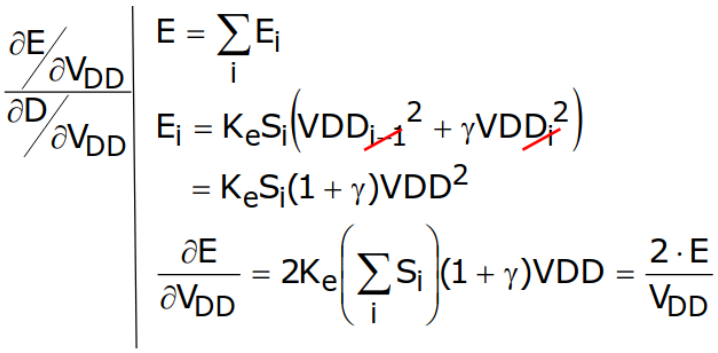


Figure 49: Sensitivity PSS 1

$$\frac{\partial E/\partial V_{DD}}{\partial D/\partial V_{DD}}$$

$$E = \sum_i E_i$$

$$\frac{\partial E}{\partial V_{DD}} = \frac{2 \cdot E}{V_{DD}}$$

$$D = \sum_i {}' td_i$$

$$= \sum_i \left(\frac{K_d V_{DD}}{(V_{DD} - V_T)^\alpha} \left(1 + \frac{1}{\gamma} \frac{S'_{i+1}}{s_i} \right) \right)$$

$$= K_d \frac{V_{DD}}{(V_{DD} - V_T)^\alpha} \sum_i \theta_i$$

$$\frac{\partial D}{\partial V_{DD}} = K_d \sum_i \theta_i \left(\frac{1}{(V_{DD} - V_T)^\alpha} - \frac{\alpha}{(V_{DD} - V_T)^{\alpha+1}} \right)$$

$$= K_d \sum_i \theta_i \frac{V_{DD}}{(V_{DD} - V_T)^\alpha} \left(\frac{1}{V_{DD}} - \frac{\alpha}{(V_{DD} - V_T)} \right)$$

$$= - D \left(\frac{- V_{DD} + V_T + \alpha V_{DD}}{V_{DD}(V_{DD} - VT)} \right)$$

Figure 50: Sensitivity PSS 2

$$\frac{\partial E/\partial V_{DD}}{\partial D/\partial V_{DD}}$$

$$E = \sum_i E_i$$

$$\ldots$$

$$\frac{\partial E}{\partial V_{DD}} = \frac{2 \cdot E}{V_{DD}}$$

$$D = \sum_i td_i$$

$$\ldots$$

$$\frac{\partial D}{\partial V_{DD}} = - D \left(\frac{- V_{DD} + V_T + \alpha V_{DD}}{V_{DD}(V_{DD} - VT)} \right)$$

$$\frac{\partial E/\partial V_{DD}}{\partial D/\partial V_{DD}} = - \frac{E}{D} \frac{2 V_{DD}(V_{DD} - V_T)}{V_{DD} + V_T + \alpha V_{DD}}$$

$$= - \frac{E}{D} \frac{2 V_{DD} \left(1 - \frac{V_T}{V_{DD}} \right)}{V_{DD} \left(- 1 + \frac{V_T}{V_{DD}} + \alpha \right)}$$

$$\frac{\partial E/\partial V_{DD}}{\partial D/\partial V_{DD}} = - \frac{E}{D} \frac{2 \left(1 - \frac{V_T}{V_{DD}} \right)}{V_{DD} \left(\alpha - 1 + \frac{V_T}{V_{DD}} \right)}$$

Figure 51: Sensitivity PSS 3

Minimize delay for given C_L and C_{in}=1?

$$t = t_{p0}.(1 + \frac{C_{in2}}{C_{in1}} + 1 + \frac{C_{in3}}{C_{in2}} + ... + 1 + \frac{C_L}{C_{inN}})$$

$$\Rightarrow \min\left(\frac{S_2}{S_1} + \frac{S_3}{S_2} + ... + \frac{F}{S_N}\right)$$

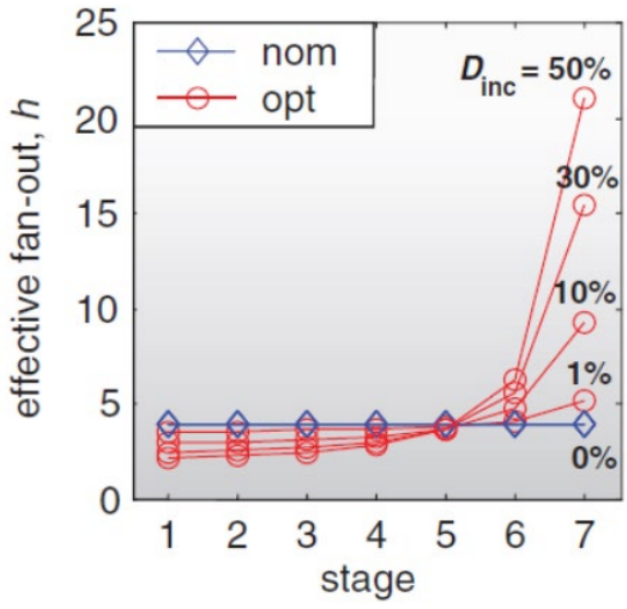
$$\Rightarrow S_j = \sqrt{S_{j-1} S_{j+1}}$$

simple case without logical effort

Each stage has the same effort f=Cext/Cin, also called 'fixed taper'

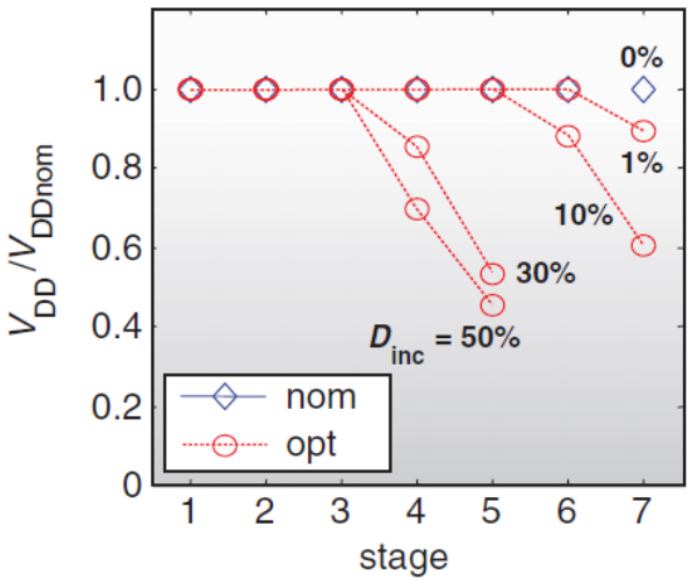
Each stage has the same delay

Figure 52: Chain of inverter example



This is a variable taper instead of the consant taper for minimum delay

Figure 53: Effective fan-out



- V_{DD} reduces energy of the final load first

Figure 54: Change of PSS