- Introduction
  - Transistor switch model
- Logic Circuits
  - Regeneration of the level
  - Capacitance
    - Effective fan-out
    - Signal in reality
  - Pass gate logic
    - Level restoration
- Circuit Timing / Dynamic Logic
  - Latch/Register Implementation
  - Sequencing, pipelining revisited
    - Clock skew
  - Dynamic logic
    - Charge coupling
    - Cascading dynamic logic
    - Clocked CMOS or C^2MOS
    - Conclusion

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 :

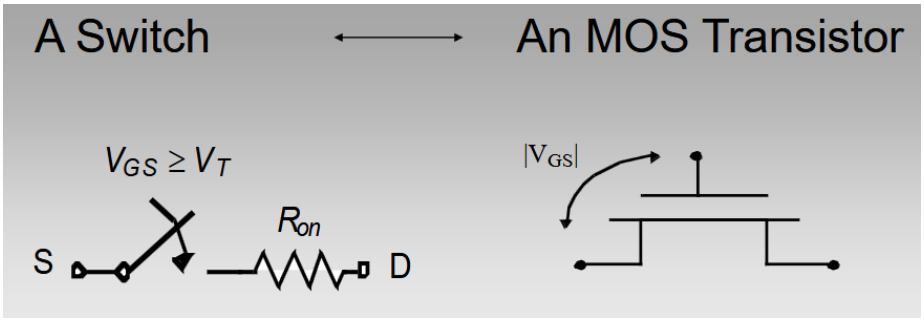


Figure 1: Switch

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.

- 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

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 :

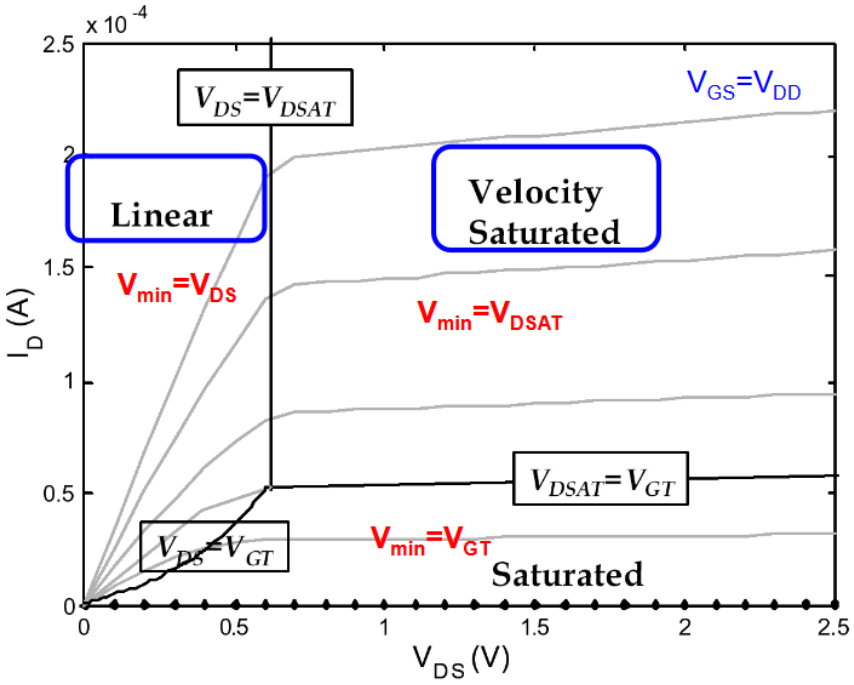


Figure 3: Regions

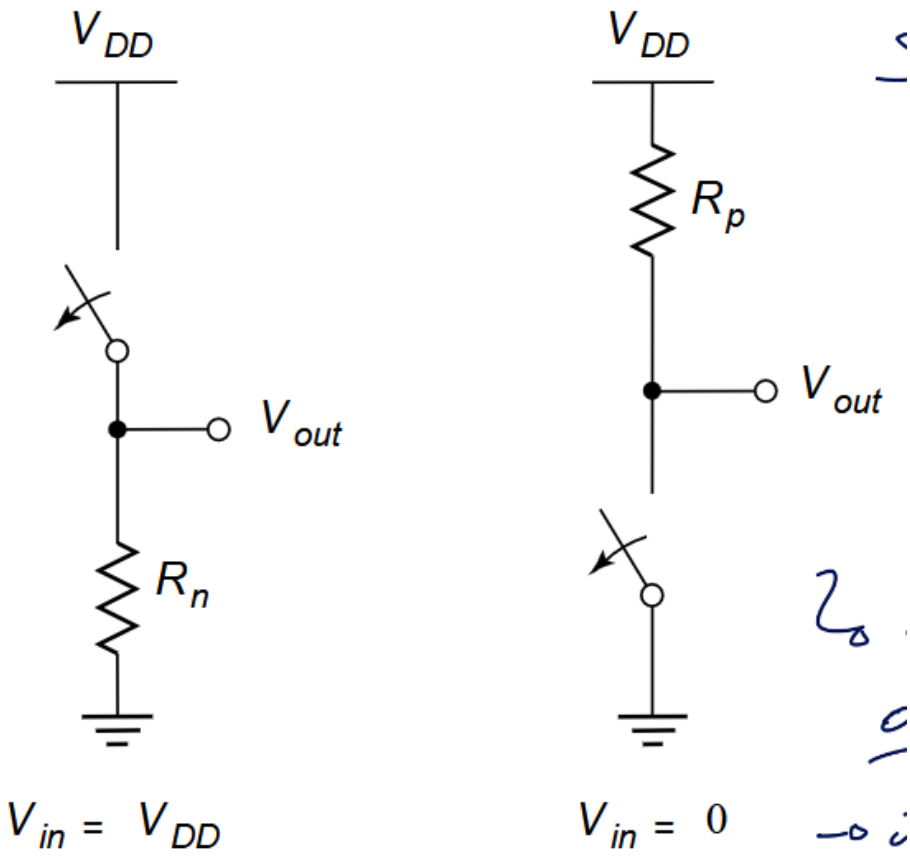


Figure 4: The static model

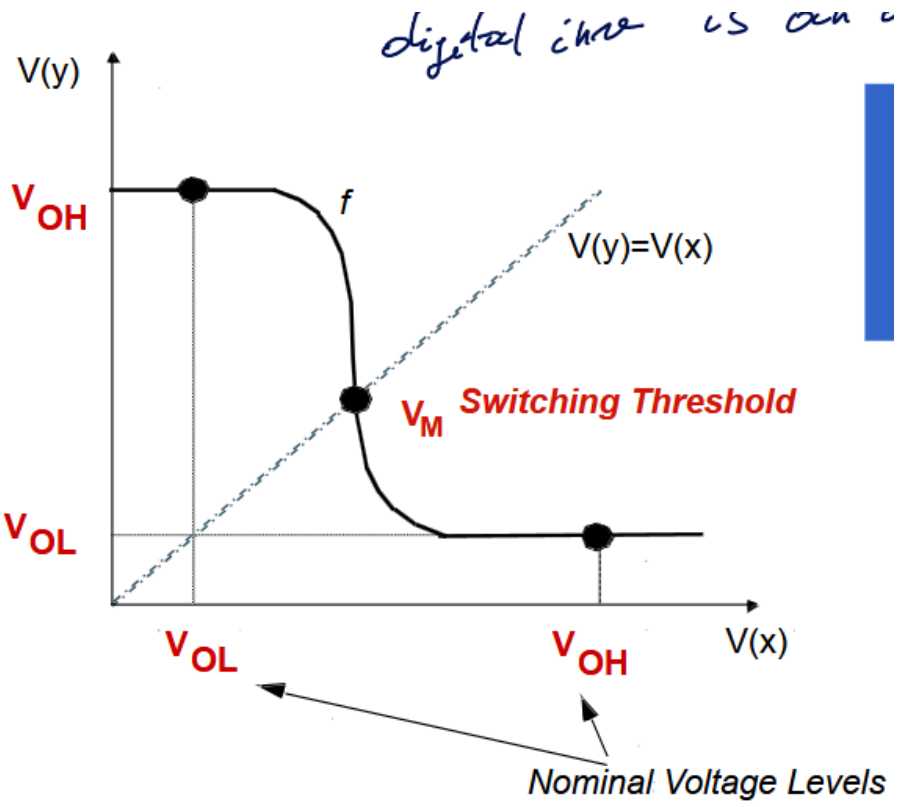


Figure 5: Switching threshold

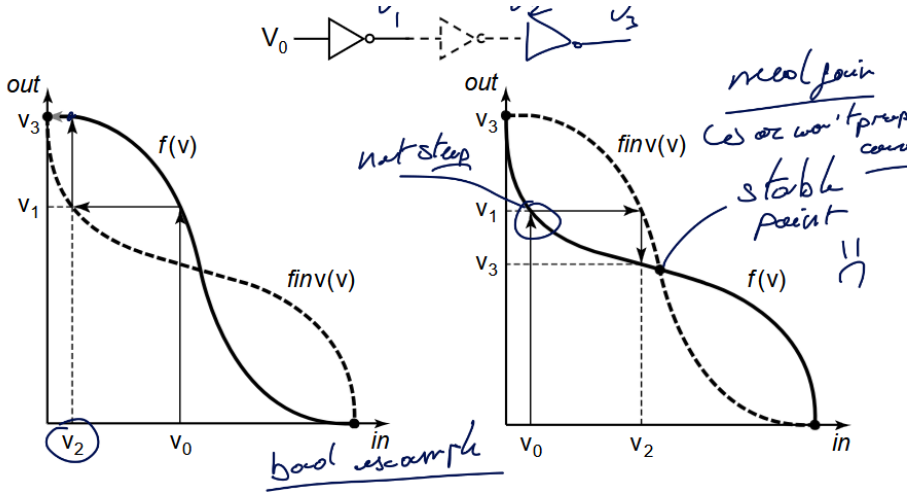


Figure 6: NAND gate regeneration

- 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.

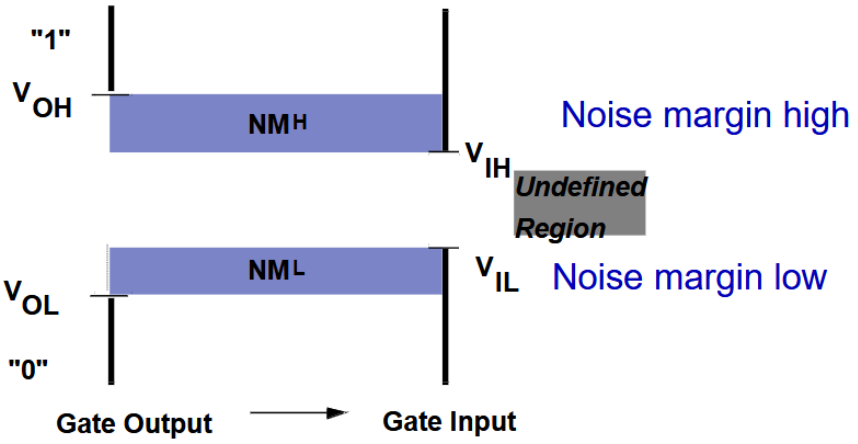


Figure 7: Noise margin

We have 3 different types of noise :

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

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

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

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

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

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

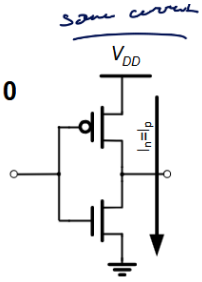


Figure 8: Switching threshold for INV

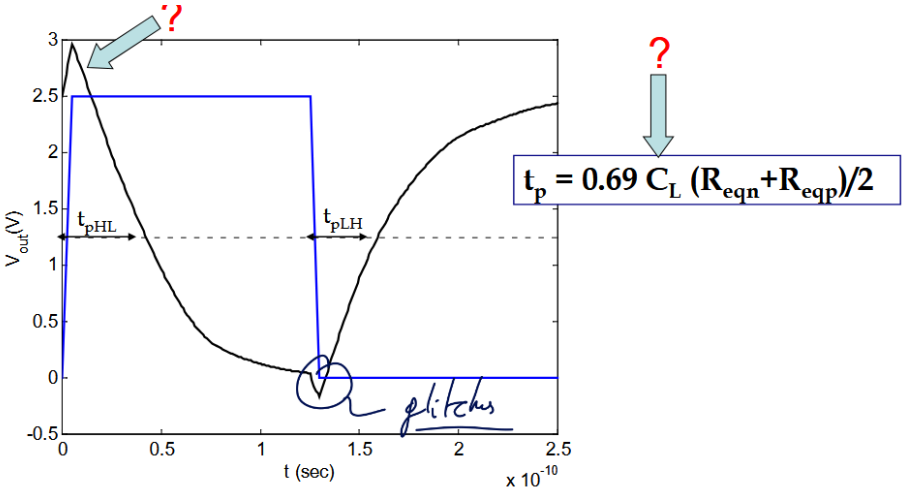


Figure 9: Glitches and Delay

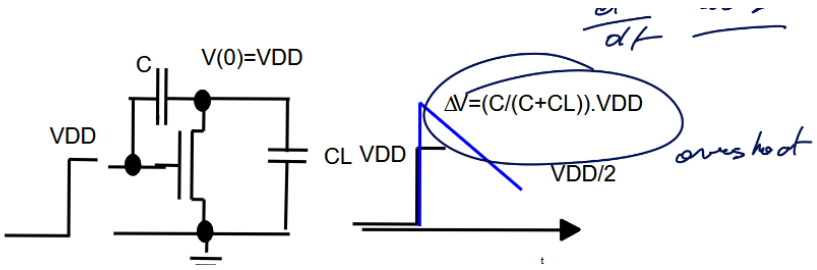


Figure 10: Explanation of the overshoot

reality and so the effect is less severe but still noticeable. We call it the *digital miller effect*:

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

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.

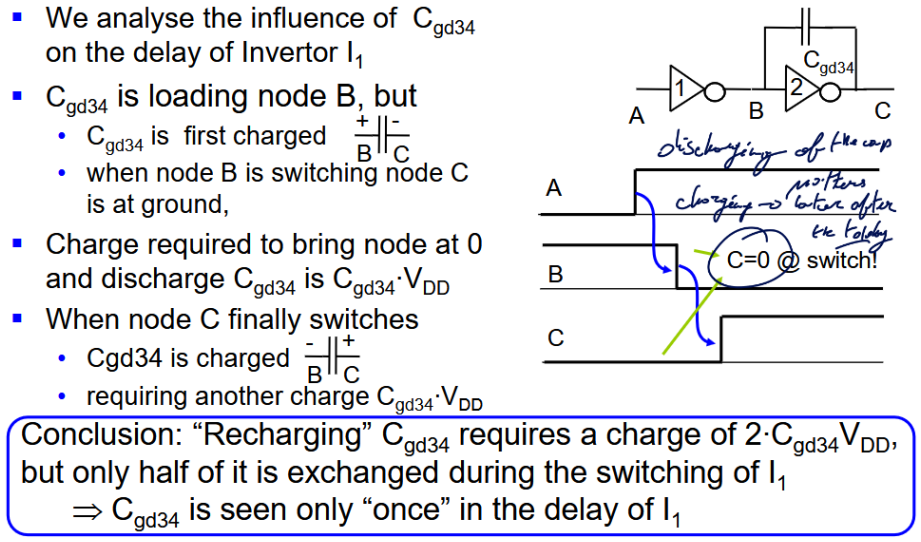


Figure 11: Loading issue

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 $C_L = C_{int} + C_{ext}$:

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

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

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 $C_{ext} = f C_g$. 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

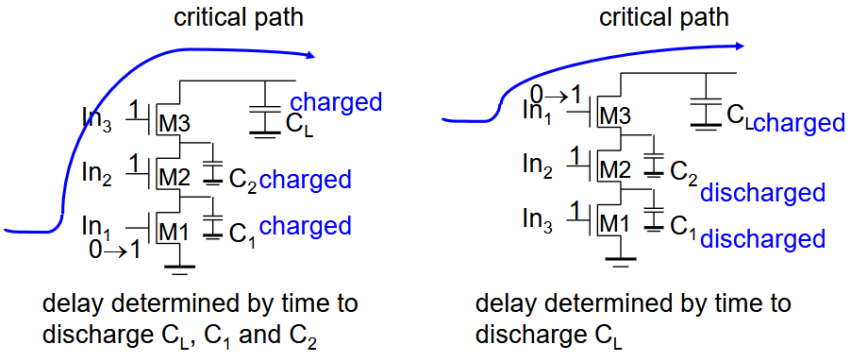


Figure 13: Critical path & Charging

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.

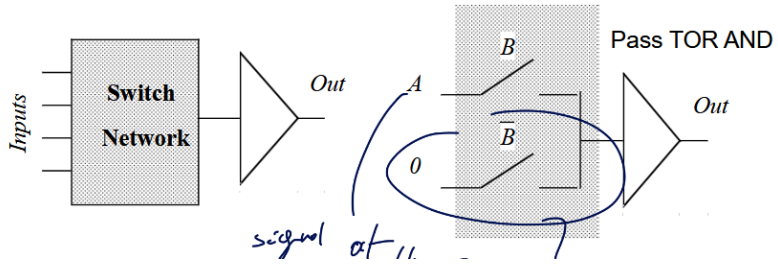


Figure 14: Pass gate logic

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.

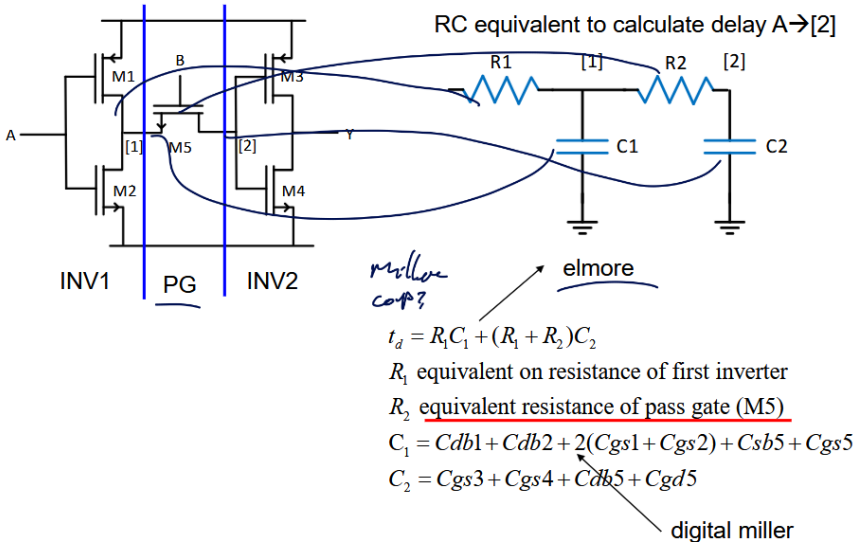


Figure 15: Static analysis

Level restoration

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

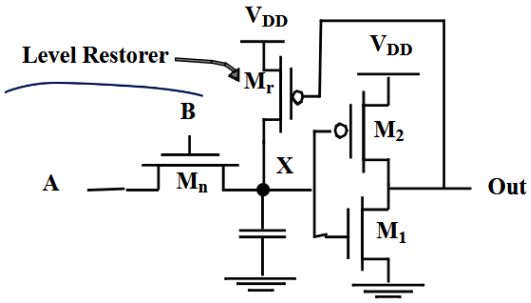


Figure 16: Level restorer

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 sizing.

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

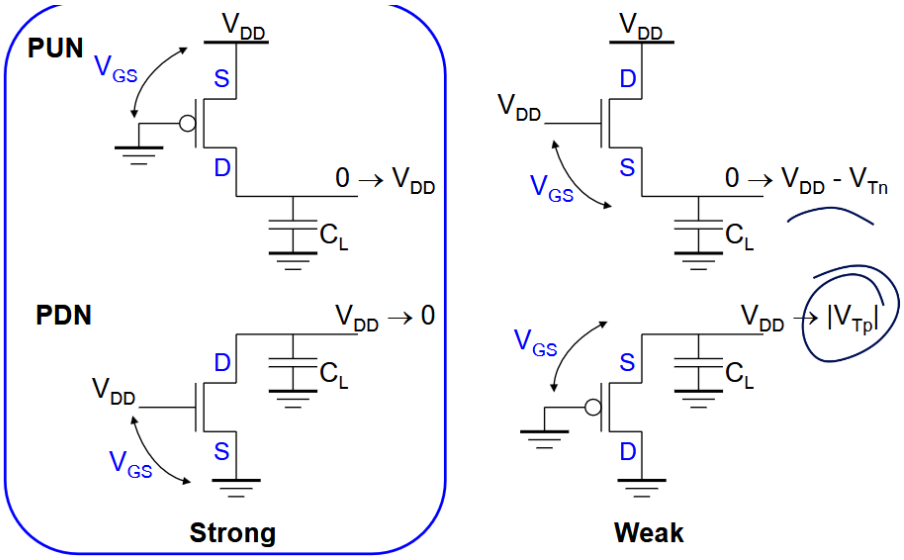


Figure 17: Threshold drops generalized

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.

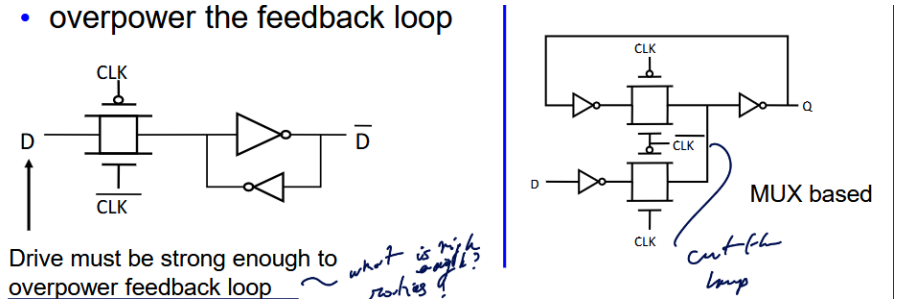


Figure 18: Latches

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.

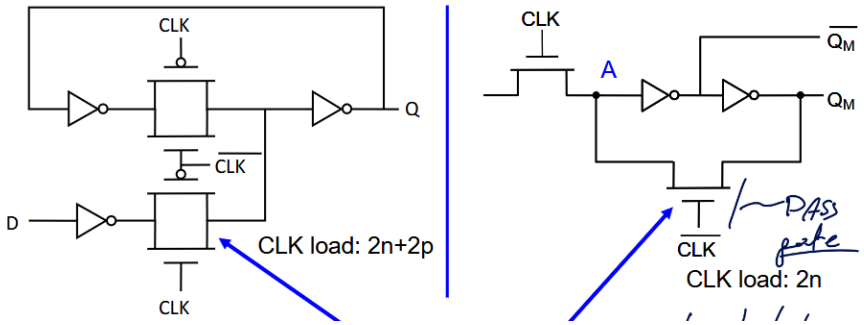


Figure 19: Mux-Based latch : Transmission Gate vs Switch

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}.

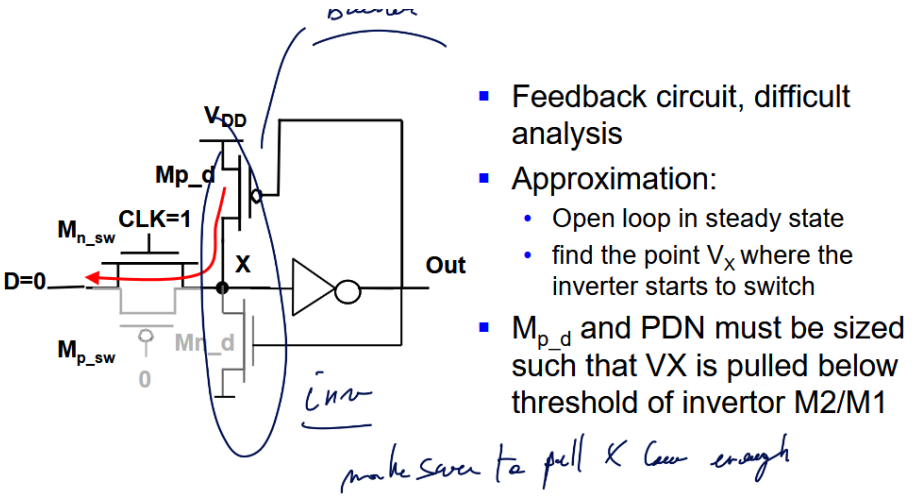


Figure 20: David Goliath

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.

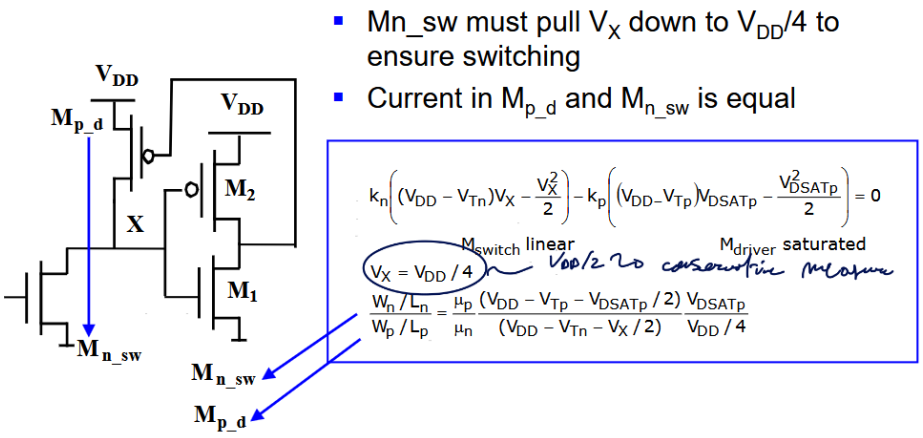


Figure 21: Sizing of the latch

NEED TO READ THIS PART CAREFULLY

PAGE 15

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

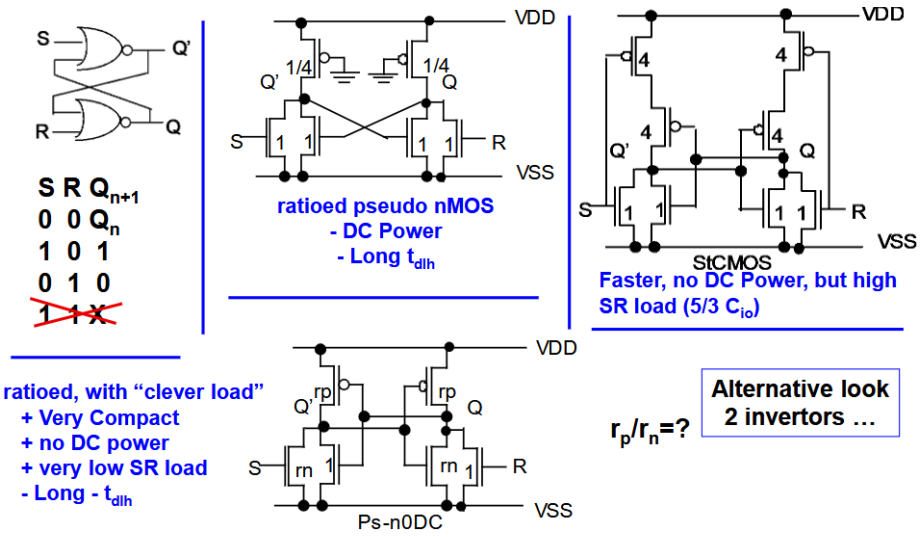


Figure 22: NOR-NOR latch

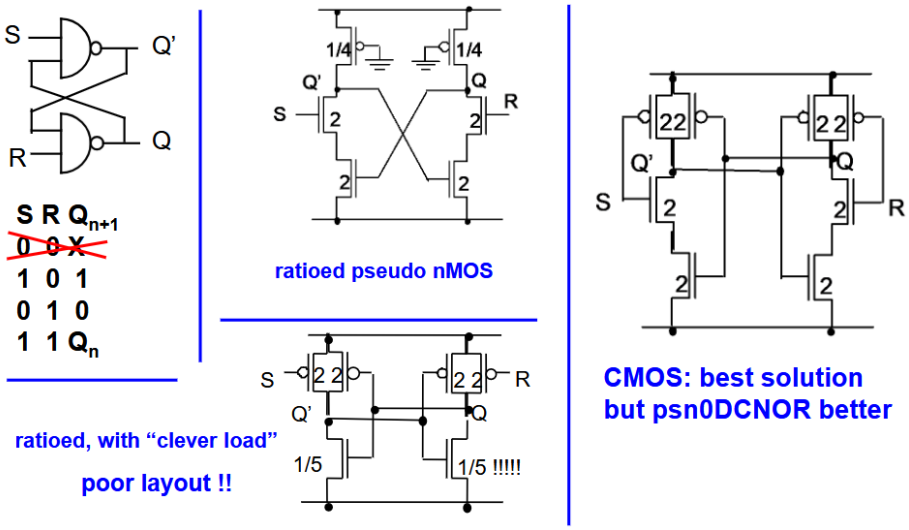


Figure 23: NAND-NAND latch

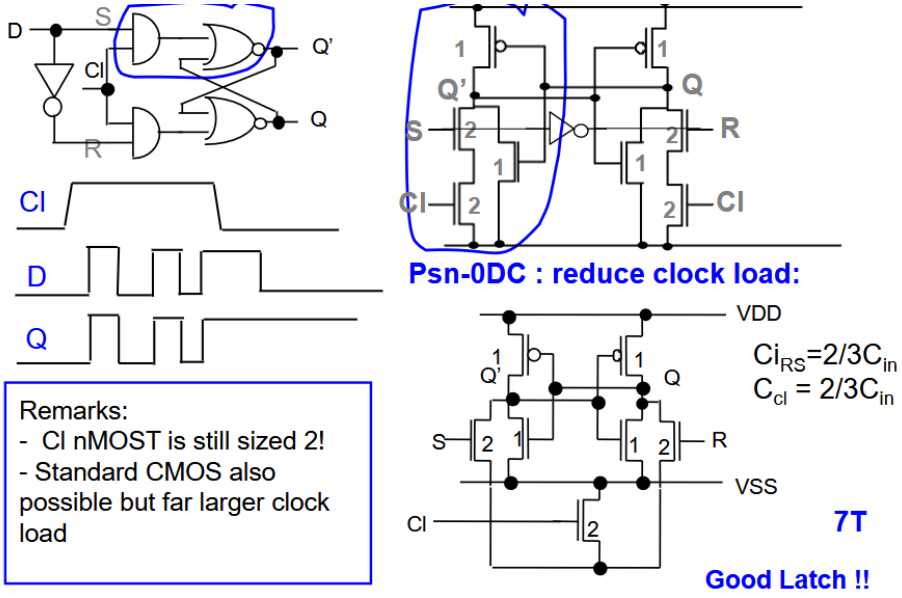
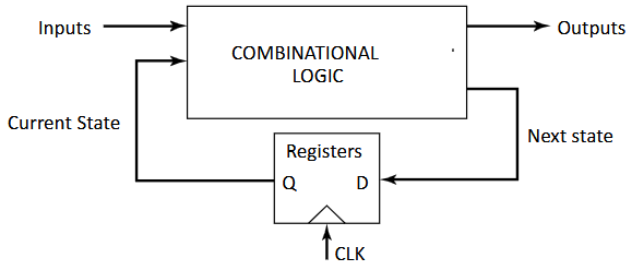


Figure 24: Transparent (n-)latch

- Finite State Machines - Controllers



- Datapath

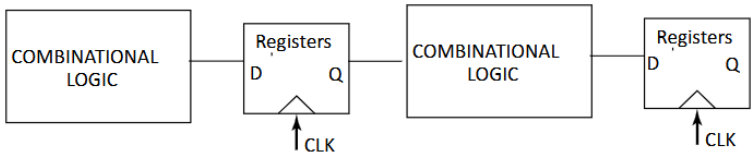


Figure 25: Registers in the system

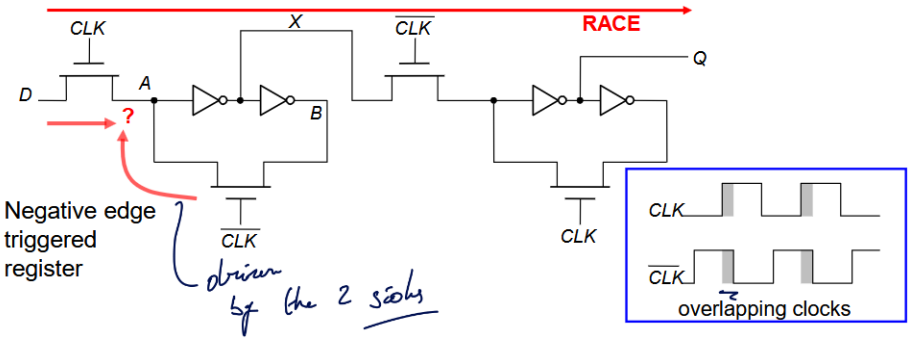


Figure 26: CLK must not overlap

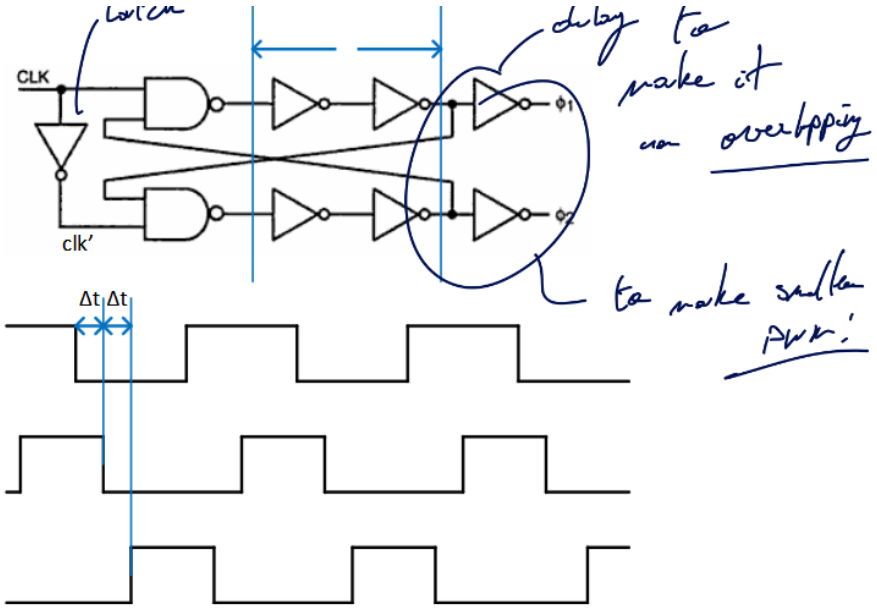


Figure 27: Creating non-overlapping clock

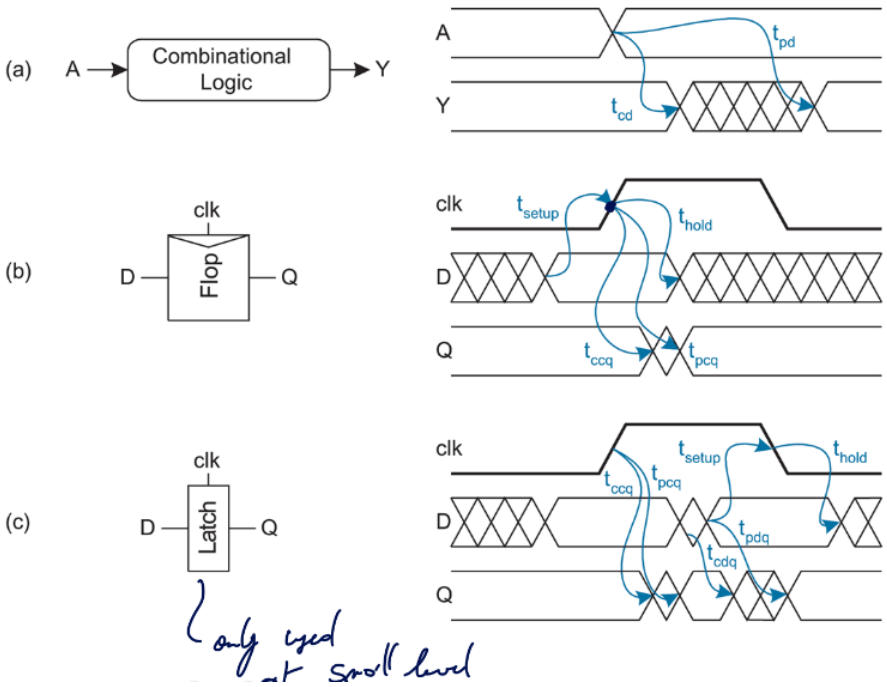


Figure 28: Timing

For registers

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

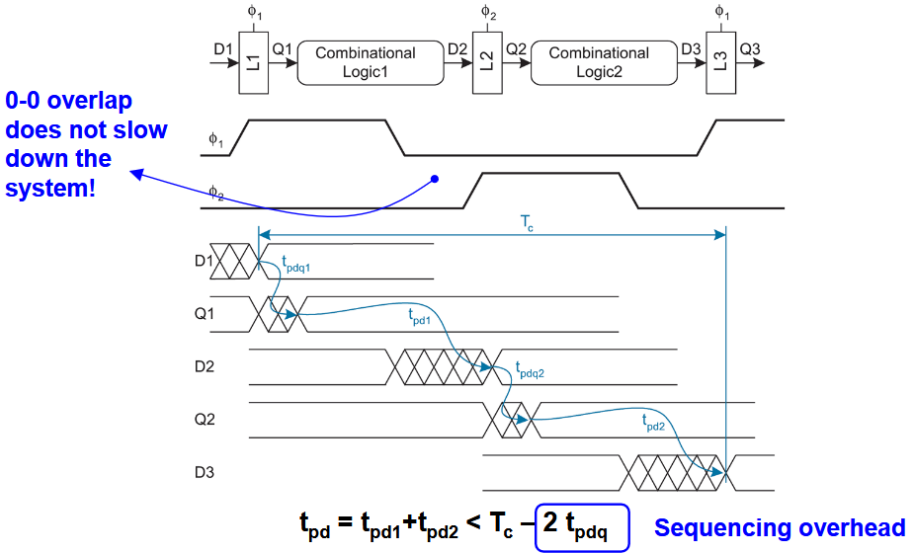


Figure 29: latches, max. delay

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

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.

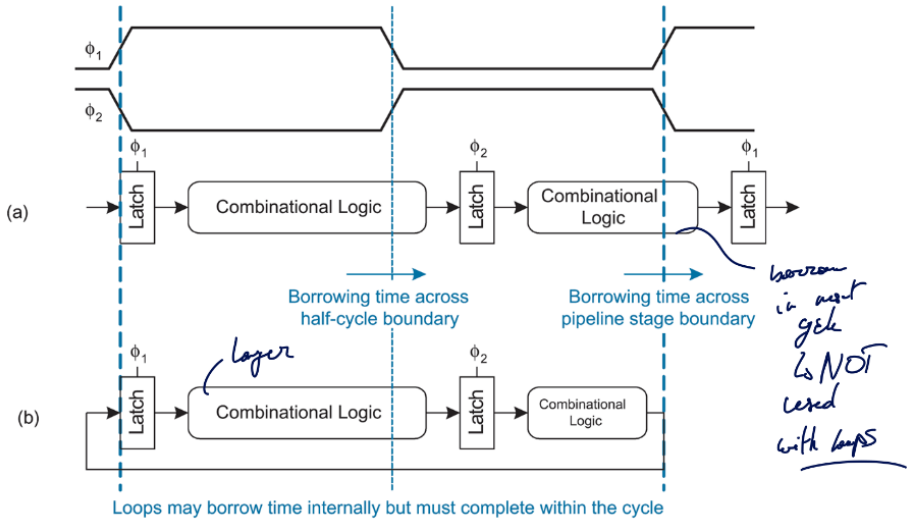


Figure 30: Time borrowing technique

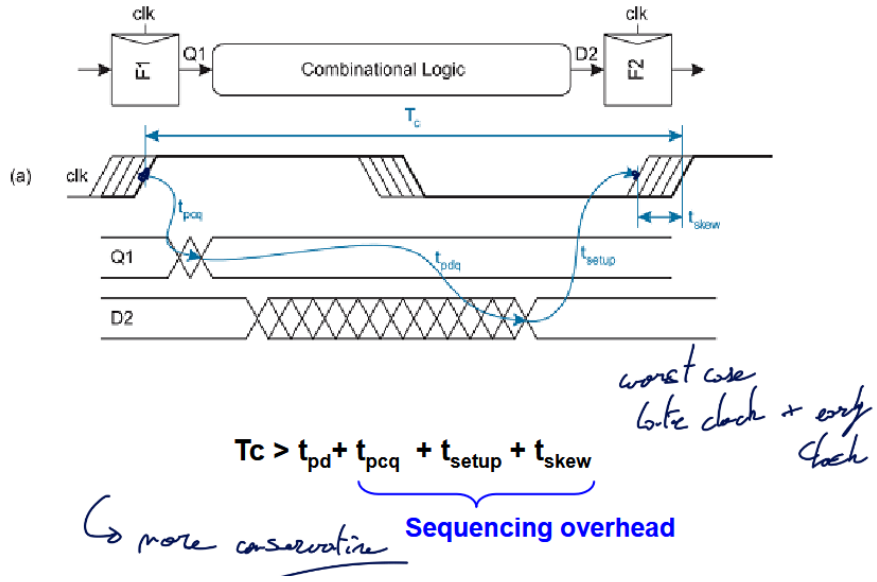


Figure 31: Clock skew effect

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 :

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

	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.

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

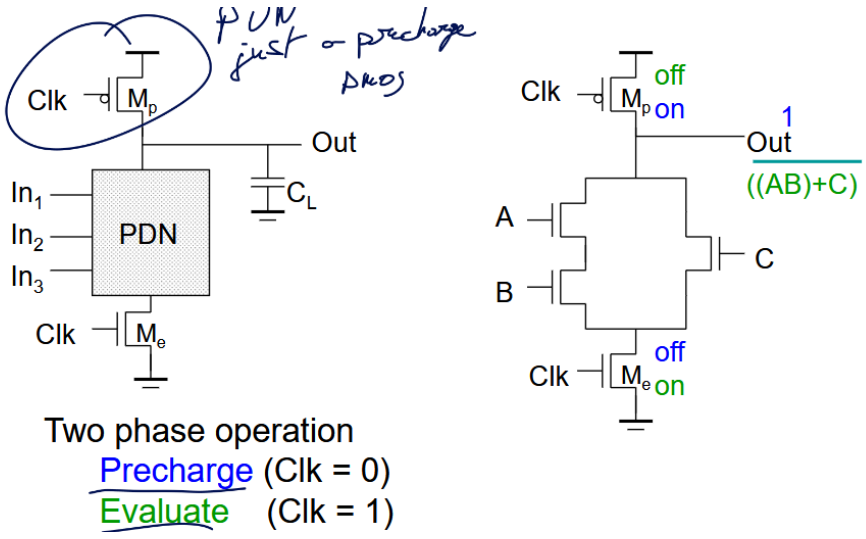


Figure 32: Example of a dynamic logic circuit

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.

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.

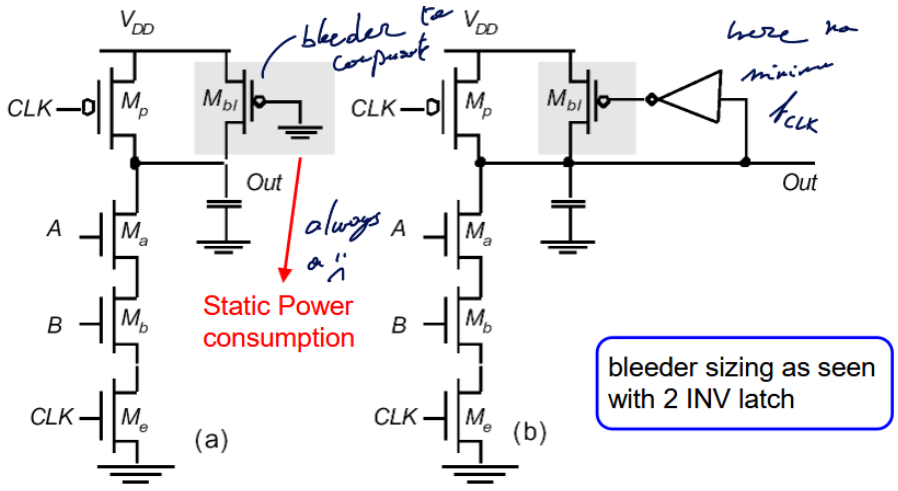


Figure 33: Bleeder and Level restorer

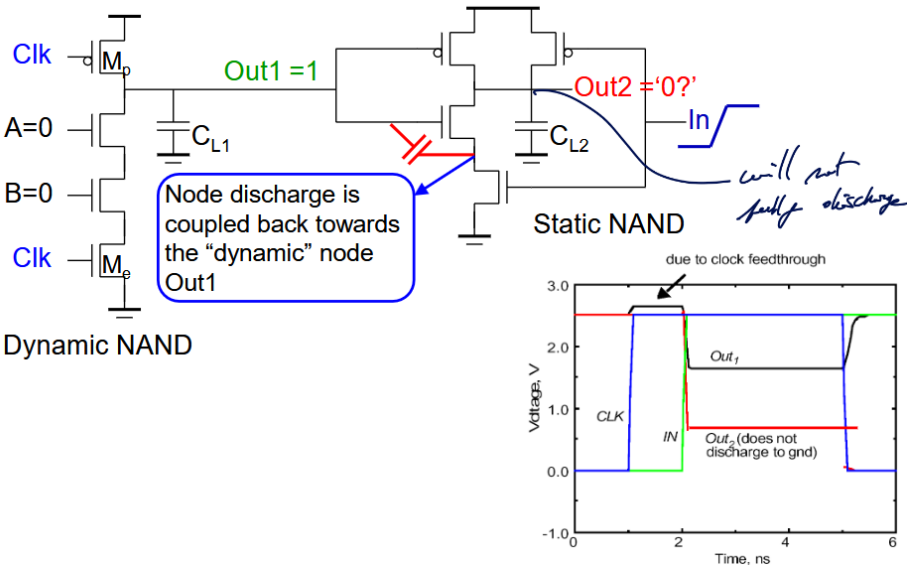


Figure 34: Output to Input coupling

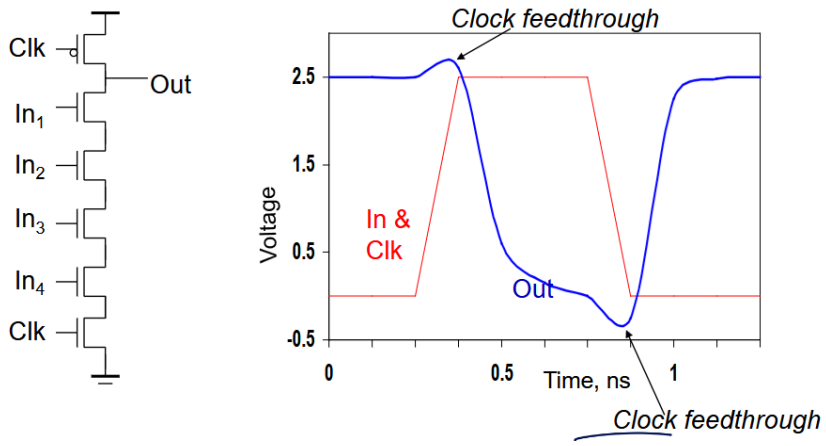


Figure 35: Clock feedthrough

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*)

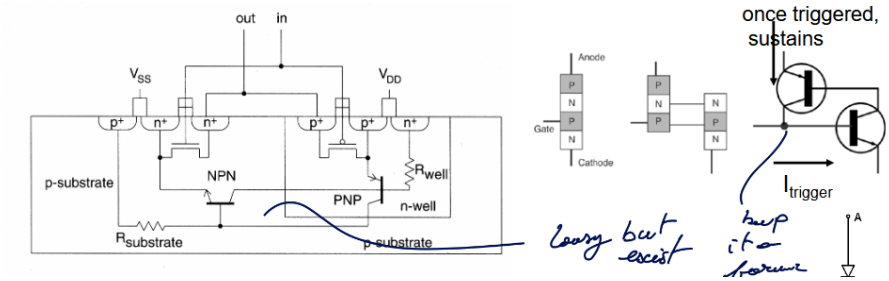


Figure 36: Latch-up

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

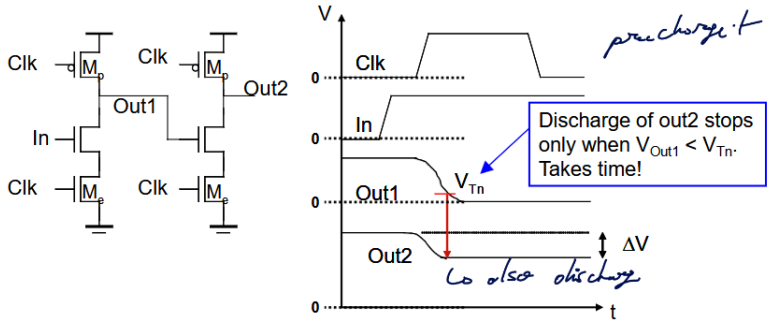


Figure 37: Cascading dynamic logic

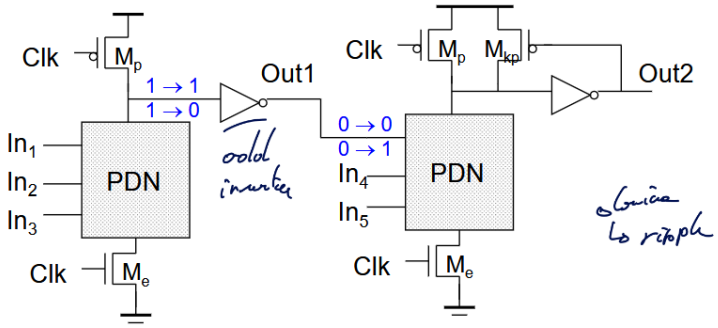


Figure 38: Domino logic

$1 \rightarrow 0$ 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**.

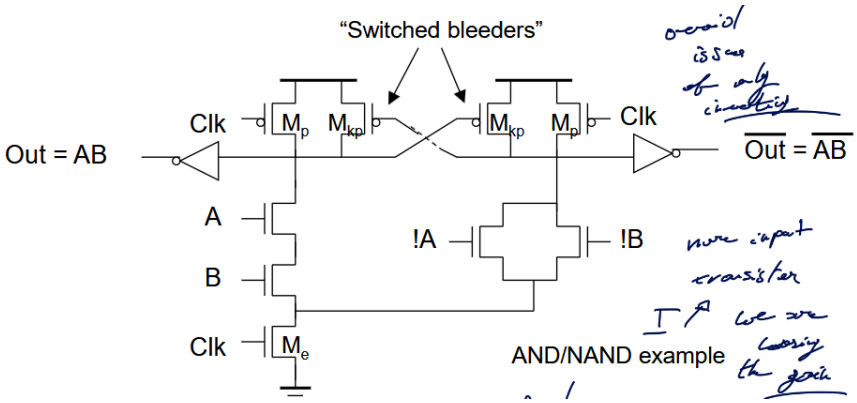


Figure 39: Dual rail domino logic - using switched bleeders

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.

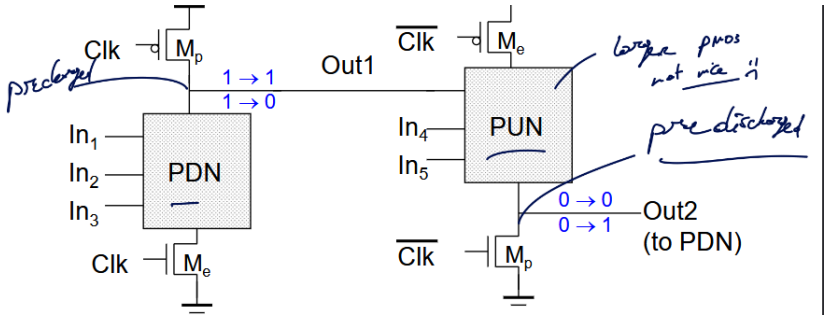


Figure 40: Alternative

Alternative to cascade of dynamic logic A good alternative is to switch between PDN and PUN networks so the $0 \rightarrow 1$ transition are allowed at inputs of PDN and $1 \rightarrow 0$ 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 $0 \rightarrow 0$ and $1 \rightarrow 1$ transition and to clock overlap. But we have good performance

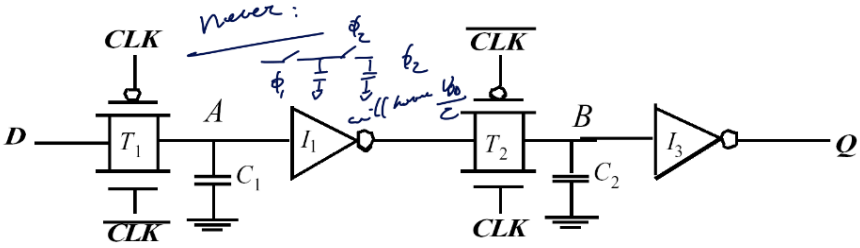


Figure 41: Dynamic edge triggered register

Clocked CMOS or C^2MOS

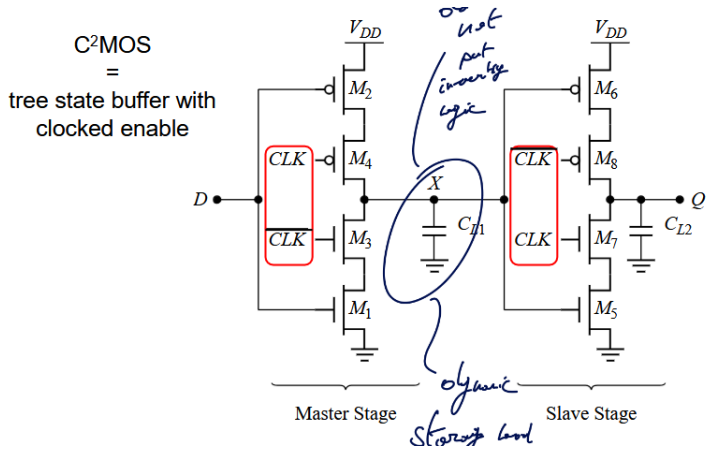


Figure 42: C^2MOS

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 $CLK = 0$, 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.

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.

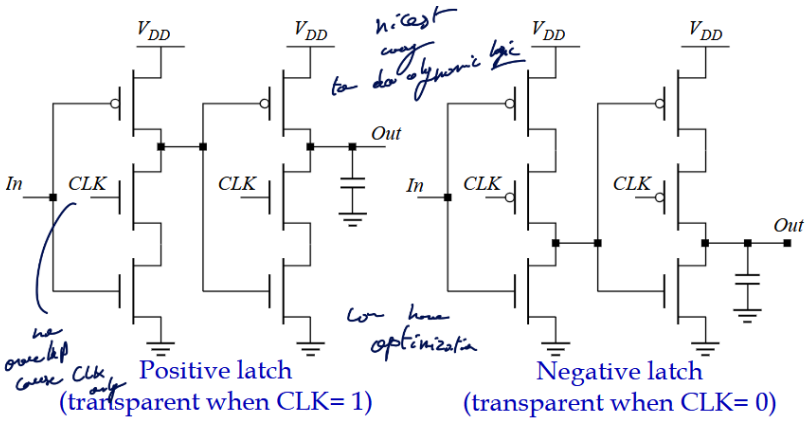


Figure 43: TSPC logice

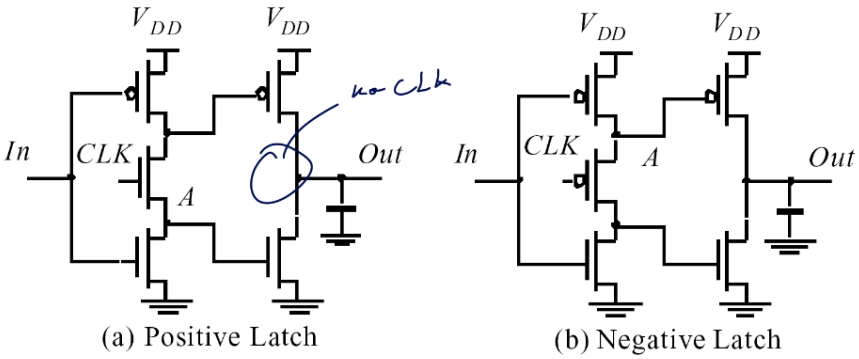


Figure 44: TSPC simplification - split output

- 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...