

DIGITAL INTEGRATED CIRCUIT DESIGN

Luca Colombo | Politecnico di Milano | a. a. 2022/2023

DISCLAIMER

These notes cover the arguments of the course 'Digital Integrated Circuit Design' held by Professor A. Bonfanti at Politecnico di Milano during the academic year 2022-2023.

Since they have been authored by a student, errors and imprecisions can be present.

These notes don't aim at being a substitute for the lectures of Professor Bonfanti, but a simple useful tool for any student (life at PoliMi is already hard as it is, cooperating is nothing but the bare minimum).

Please remember that for a complete understanding of the subject there is no better way than directly attending the course (DIY), which is an approach that I personally suggest to anyone. Indeed, the course is really enjoyable and the professor very clear.

In any case, if you found these notes particularly helpful and want to buy me a coffee for the effort, you're more than welcome: <https://paypal.me/LucaColombox>

luccolombo29@gmail.com

DIGITAL INTEGRATED CIRCUIT DESIGN

- Digital refers to a way of coding a signal opposed to analog way. Which advantages?
 - ✓ Robustness to noise
 - ✓ Ease to store data
 - ✓ Ease to process data

- Integrated means that the circuits are implemented on a single chip, i.e., on the same silicon substrate, with advantages in terms of area, delay and power consumption with respect to a discrete component circuit.

Digital means binary, 0 and 1. In digital electronic noise has to be intended as a sort of interference, not actually noise. There is also a deterministic noise in binary digital electronics, that corrupts the signal, and it is the quantization noise, which can be reduced increasing the number of bits.

The other advantage is that in the digital domain it is very easy to store data, exploiting the concept of positive feedback or with a capacitor (capacitor charged to V_{dd} or discharged is called dynamic memory). The last advantage is that it is easy to process data with respect to the analog domain. Also filtering in the digital domain is very easy.

DIGITALIZATION OF INTEGRATED CIRCUIT

If we consider an integrated circuit that wants to read the signal from a sensor and to elaborate it, typically we have: the sensor, an amplifier with low noise to increase the dynamic range of the signal (voltage or current), a simple filter (1st or 2nd order) and then an ADC. After the ADC we have a digital circuit (DCP) to perform all the complex operations. The analog part is mandatory in the IC because the signal we acquire is for sure analog.

- There is a growing trend to move toward digital circuits due to the benefits of digital with respect to analog signal processing;
- Digital circuits (e.g., microprocessors, memories..) are becoming more and more complex, fast and power-hungry.

How these two trends were possible and are still ongoing?

- Process technology scaling
- Efficient design tools

Then, let's consider truly digital circuit, like the microprocessors. They are becoming very complex, fast and power hungry. This is possible thanks to two great advancements in the electronics in general, that are:

- Process technology scaling: reduction of the dimensions of the transistors.
- Efficient design tools: for instance the semi-custom approach based on the standard cell library.

CMOS TECHNOLOGY SCALING

It is the reduction of the transistor dimensions, typically the length, together with the oxide thickness, the power supply and the threshold voltage. This is the technology scaling. In fact, the reduction of the dimensions leads to a reduction of the area and hence the cost per transistor, and we can also implement more transistors in the same area. Moreover, we also increase the speed of the transistor (cut-off frequency) because the parasitic capacitances are smaller if we reduce the dimensions. If we reduce the length, the electron time to move from source to drain is smaller and so the transistor is faster.

□ What's scaling?

Reduction of the dimension of different parameters (minimum length, oxide thickness, power supply)

□ Why scaling?

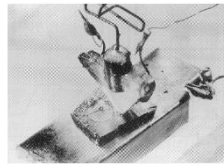
- A. Reduction of area per transistor/cost
- B. Increase of speed
- c. Reduction of power consumption (at a given frequency)

The **dynamic power consumption** is $P = C * V_{dd}^2 * f$.

C is the capacitance we have to charge and discharge, f is the frequency with which we do it.

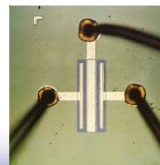
BREAKTHROUGHS

First BJT transistor at Bell Labs, 1948
(Bardeen, Brattain, Shockely)



First Integrated Circuit, 1958
Jack Kilby, Nobel Prize in 2000

First MOS transistor, 1959
Mohamed Atalla and Dawon Kahng
at Bell Labs



Before 1948, all circuits were implemented using thermionic valves, so vacuum tubes, which were not reliable and power hungry.

Kilby designed the first IC; he put more transistors on the same substrate of Germanium. Then, in 1959 Noyce (founder of Intel with Moore) designed an IC. The difference with Kilby is that it is less rudimental and more similar to modern IC. It considers 3 aspects:

- Transistors assembled on the same substrate.
- Isolation of transistors: done with reverse bias pn junction, otherwise transistors talk to each other.
- Interconnections on the IC (Kilby did it with flying wires outside the IC).

In 1959 the first MOS transistor was implemented. From 1968 on, MOSFET overwhelmed BJT, because we can implement very powerful switches, which are very important in digital electronics. BJTs are more of analog devices (e.g. to implement current generators).

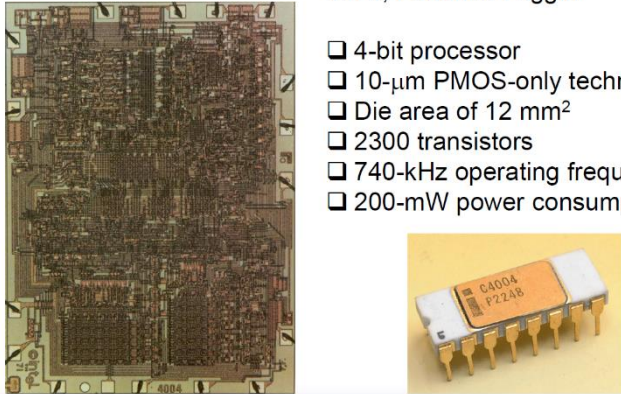
The first problem with the MOS transistor was that the gate was difficultly placed over the substrate, e.g. leaving spaces between the gate area and the n+ wells. Otherwise, the gate was overlapped on the n+ wells. In the first case the transistor is not working, while in the latter case we are creating a large parasitic capacitance, so the device is very slow.

Then Federico Faggin, in 1968, implemented the gate first and then the n+ well regions. We still might have a lateral spread under the gate area, but below the gate there are no ions because the polysilicon gate prevents the ion to reach the channel → **self-aligned gate technology**. Now the device is working and also very fast.


In 1971, Faggin also implemented the first microprocessor, the Intel 4004.

1971, Federico Faggin

- ❑ 4-bit processor
- ❑ 10- μm PMOS-only technology
- ❑ Die area of 12 mm²
- ❑ 2300 transistors
- ❑ 740-kHz operating frequency
- ❑ 200-mW power consumption



Nowadays, the technology is of 10 nm, and billions of transistors with an operating frequency of GHz. The power consumption is increased, but 250W is something we can deal with.



- ❑ FinFET 10-nm technology
- ❑ Size of 257 mm²
- ❑ Billions of transistors
- ❑ Frequency up to 6 GHz
- ❑ 250-W power consumption

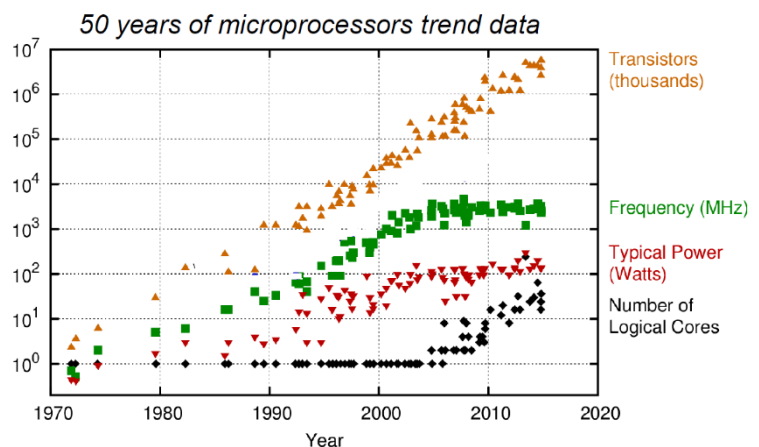


Watch out!
AMD is already producing processors (Ryzen 7000) in 5nm process, as well as Apple (M1).

GLOBAL TREND FOR MICROPROCESSORS

In the chart we describe the evolution of microprocessors in the last years. The number of transistor we find in an IC is increasing exponentially (logarithmic scale). Also the frequency is increasing, even if we are reaching a plateau.

Power consumption is almost flat, constant. This is possible because if we consider the previous formula for P, and C is the capacitance of a single transistor, the overall power consumption is the sum of all the single power consumptions.



If the number of transistor is exponentially increasing, V_{dd} is decreasing and f is increasing, but the C for the single transistor is decreasing \rightarrow we are in the end counterbalancing the effects.

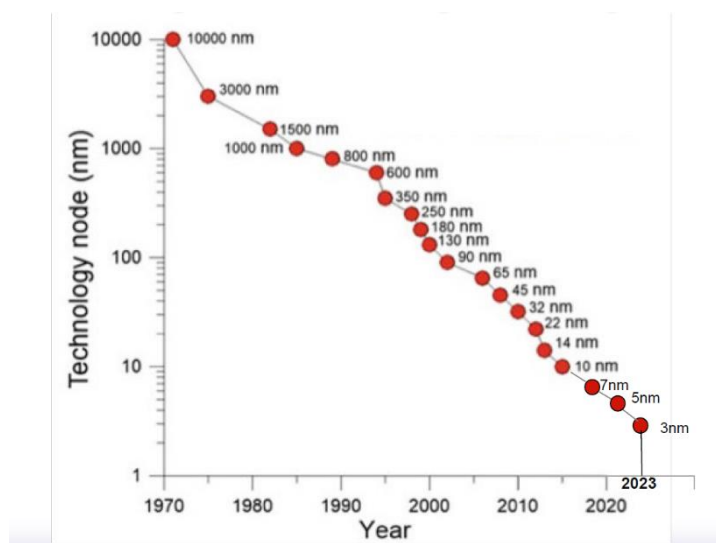
In the chart we can thus observe all the advantages of the technology scaling:

- Less cost per transistor
- Higher running frequency
- Small power consumption

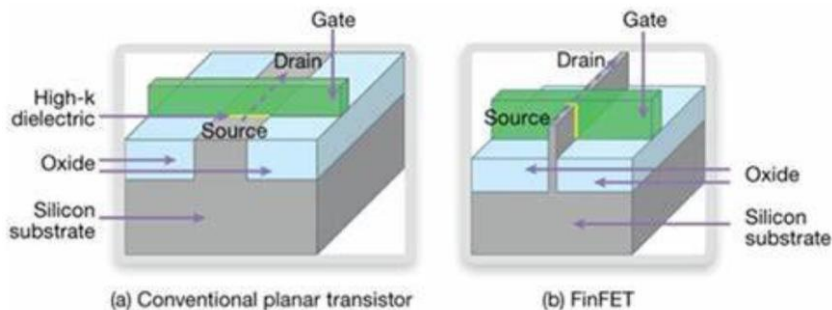
The operating frequency is rather constant after 2000 more or less at 6 GHz because the problem is the power density. If we increase the frequency, for a small area, we increase the power, and Silicon can withstand at max 100 W/cm^2 , so we eventually run the risk to burn Silicon. So what we do is, rather than increasing the frequency of one single core, we implement more cores in a microprocessor.

Scaling of CMOS gate length

Plot of the minimum length vs year of entrance in the market. It is an exponential decrease, and it is called Moore's law. Moore predicted, in 1965, that every two years we will have at disposal a new technology with a decrease in length of a factor $\sqrt{2}$.



Year 2011 is a fork. Before it we have planar devices (classical MOS transistors) and from 2011 on FINFET are used. It follows the same equations of a planar MOS transistor, but the gate surrounds by three sides the channel. In fact, the scaling of the planar MOS is problematic because of the increase of the subthreshold current and the DIBL effect arises (threshold voltage depends on the drain voltage, so we have no more a transistor).



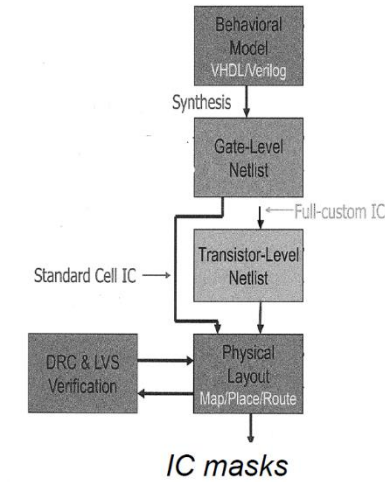
With the FINFET the gate gets back control of the channel.

SEMI-CUSTOM ASICs

It is not enough to have a powerful technology, we need to have efficient design tools to design digital ICs.

An ASIC is the application-specific integrated circuit. And ASIC for which the logic cells are pre designed is called semi-custom ASIC. Using predesigned cells from a library makes the design easier and faster. The most common approach is the standard-cell based approach.

An example is in the following image.



If we want to produce the mask of the IC to design it, we can do two things:

- We design the circuit at transistor level, and this is the **custom approach**. With this approach, which is possible if the circuit is not complex, we maximize the performances, because each transistor is designed. If we want to decrease the design time, we use another approach. **NB:** the used length is always the minimum, and the only parameter the designer has to consider is the width (increasing the length, we improve the offset, we reduce the flicker noise and increase the gain, but there are all analog parameters, so not important in the digital domain).
- We describe the circuit in a behavioral way (VHDL) and then there are tools to translate the behavioral code into a netlist of standard cells (NOT, AND, memories ecc.). Then the gates have also the layout (mask) available, so also the layout is performed automatically. Design time is reduced but maybe the circuit is not optimized.

DESIGN METRICS OF DIGITAL CIRCUITS

- Cost (area)
- Reliability
- Speed (dynamic performance τ_p , propagation delay)
- Power consumption

COST

In general, the cost of an IC is due to two factors:

1. Recurring engineering costs: variable costs
 - a. Silicon processing, packaging, test
 - b. Proportional to volume
 - c. Proportional to chip (die) area
2. Non recurring engineering costs: fixed costs, they are not proportional to production volume:
 - a. Design time and effort
 - b. Mask generation: most impacting contribution
 - c. Manufacturing machines and building

The general cost for an IC can be calculated as:

$$\text{IC cost} = \text{RE costs} + \frac{\text{NRE costs}}{\text{production volume}}$$

$$\text{RE costs} = \text{die cost} + \text{package cost} + \text{testing cost}$$

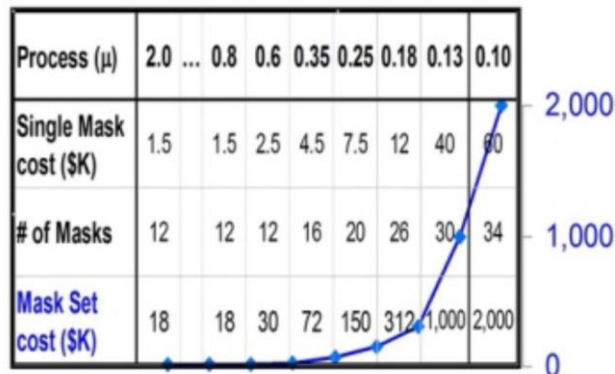
The first term is the RE costs per chip, the second one is divided by the number of chips we are producing. If we produce a lot of chips, the second term is negligible.

The die cost is the cost to process the silicon by means of photolithographic process. All the three terms in the RE costs are proportional to the area. Also the packaging, if we increase the die, increases. Hence if we increase the complexity of the circuit, also the testing cost increases.

Considering this expression, in the RE costs the die cost account for 80% of the overall RE costs.

Furthermore, the NRE costs term at the numerator is very large, but negligible because it is divided by the number of chips we are producing. The more impacting factor here is the mask cost.

The mask cost is exponentially increasing with the increase of scaling. In old processes, number of masks was smaller, while now we have a larger number but also the cost increases. However, this cost is not impacting because divided by the production volume.



How to create a chip

We want to find a formula that relates the area and the cost.

We start from an ingot, which is sliced in 1mm slices. The diameter of a wafer can be 20 or 30 cm. In the wafer, the chip is replicated as many times as possible with photolithographic process. Of course some area of the wafer is wasted.

Die cost

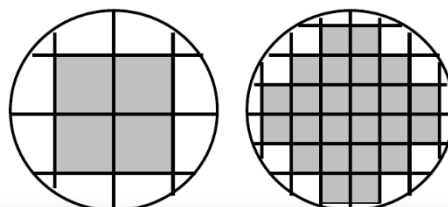
The wafer cost is around 1000\$, it is a fixed cost. Then, in order to assess the die cost, we need to divide by the number of dies per wafer. The dies per wafer must be multiplied by the yield (resa), because not all the dies we create are working. So at the denominator we have the number of working dies.

The number of dies we can implement in a wafer is computed considering the area of the wafer divided by the die area. To this term it is subtracted the fact that some area is wasted, which is the perimeter of the circle divided by the diagonal of the die.

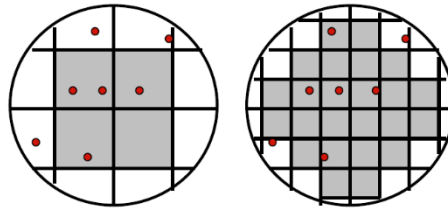
$$\text{die cost} = \frac{\text{wafer cost}}{\text{dies per wafer} \times \text{die yield}}$$

$$\text{die yield} = \frac{\text{No. of good chips per wafer}}{\text{Total number of dies per wafer}} \times 100\%$$

$$\text{dies per wafer} = \frac{\pi \times (\text{wafer diameter}/2)^2}{\text{die area}} - \frac{\pi \times \text{wafer diameter}}{\sqrt{2} \times \text{die area}}$$



Die yield



$$\text{die yield} = \left(1 + \frac{\text{defects per unit area} \times \text{die area}}{\alpha} \right)^{-\alpha}$$

α is approximately 3

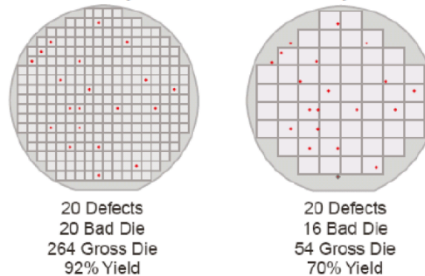
$$\text{die cost} = f(\text{die area})^4$$

Circuits with red spots are not working. In the second case we have a lot of working dies \rightarrow the larger the area of the die, the smaller the yield.

Defects per unit area is 1 or 0.1 defects per cm^2 . We also notice that if we double the area, the cost is 4 times larger according to the last formula.

Example of die yield

- wafer size of 12 inches, die size of 2.5 cm^2 , 1 defects/ cm^2 , $\alpha = 3$ (measure of manufacturing process complexity)
- 252 dies/wafer (remember, wafers round & dies square)
- die yield of **16%**
- $252 \times 16\% =$ only 40 dies/wafer die yield !



20 Defects
20 Bad Die
264 Gross Die
92% Yield

20 Defects
16 Bad Die
54 Gross Die
70% Yield

□ Die cost is strong function of die area

- proportional to the third or fourth power of the die area

Let us make an example considering a wafer of 30 cm diameter, a die area of 2.5 cm^2 and a defect density of 1 defect/ cm^2 . The number of chips per wafer is $282 - 42 = 240$, if we consider the above-mentioned formula. Among these 240 dies, we have to consider that not all the dies work correctly. Applying the empiric formula of the yield, only 16% of the dies work fine, but because we start from a very large area. So only 40 dies are working \rightarrow the smaller the die the better.

Thus, summarizing, we can observe that the die cost drastically depends on the die area. So, the area is a very important parameter at any level: at gate level, at module level and at system level. As far as the gate area is concerned, the number of transistors and their relative size (W and L) are the major parameters that set the area, but also the complexity of the gate can influence the area due to the routing of the interconnections.

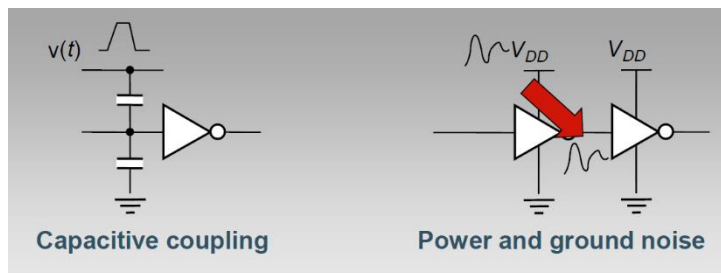
Chip	Metal layers	Line width	Wafer cost	Def./cm ²	Area mm ²	Dies/wafer	Yield	Die cost
386DX	2	0.90	\$900	1.0	43	360	71%	\$4
486 DX2	3	0.80	\$1200	1.0	81	181	54%	\$12
Power PC 601	4	0.80	\$1700	1.3	121	115	28%	\$53
HP PA 7100	3	0.80	\$1300	1.0	196	66	27%	\$73
DEC Alpha	3	0.70	\$1500	1.2	234	53	19%	\$149
Super Sparc	3	0.70	\$1700	1.6	256	48	13%	\$272
Pentium	3	0.80	\$1500	1.5	296	40	9%	\$417

This slide shows some examples of integrated circuits with their corresponding area and cost. The table refers to microprocessors. First of all, note the cost of the wafer: It costs approximately 1000 dollars. Notice the number of defects per cm²: approximately 1 defect per cm².

Finally, note that the die area increases from top to bottom and the corresponding cost increases more than linearly with the die area. Finally, it's worth noting that also the package cost is related to the die area: the larger the area, the larger the package (and maybe also with more pins due to a larger complexity of the circuit). We can notice that also the yield is decreasing, and the result is that the cost is increasing from top to bottom. The area is increased more or less of a factor of 5, the cost of 100.

RELIABILITY

Noise in digital integrated circuit



Reliability means the ability of a circuit to be immune to digital noise → robustness against digital noise.

Which are the sources of noise in a digital circuit? In an analog circuit, we deal with thermal, shot and flicker noise. This noise can alter the value of the voltage for example, which can assume all the values in the range between ground and power supply. In binary digital circuits, the signals can assume only two logic values, high or low, "1" or "0" typically corresponding to power supply voltage and ground. So **thermal, shot and flicker noise are not important** since they cannot alter the logic value of a voltage signal. In this slide you can see two examples of noise in digital circuits.

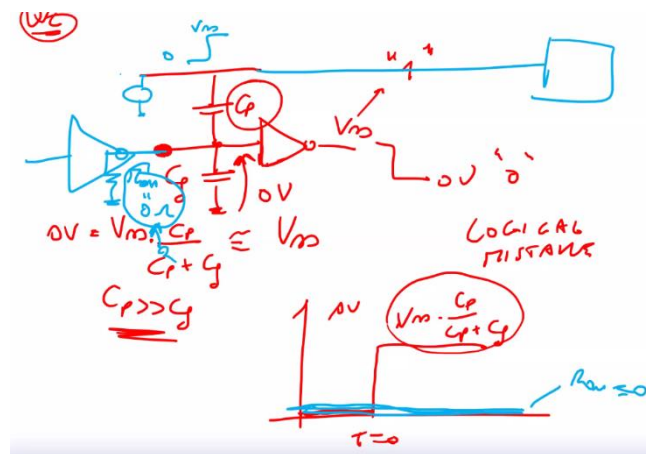
Examples of digital noises are the following.

Capacitive coupling

The first case is a **capacitive coupling**. If we consider a node whose voltage transitions from 0V to power supply value (step variation), this variation can couple to the input of a nearby inverter (supposed at 0V in this case), through a parasitic capacitance. In fact, the bottom capacitance is the gate capacitance C_g . We are implicitly assuming that the input of the inverter is a high impedance node (it can be modelled as a capacitance) and that it is driven by a high-impedance driver (which is a worst-case scenario).

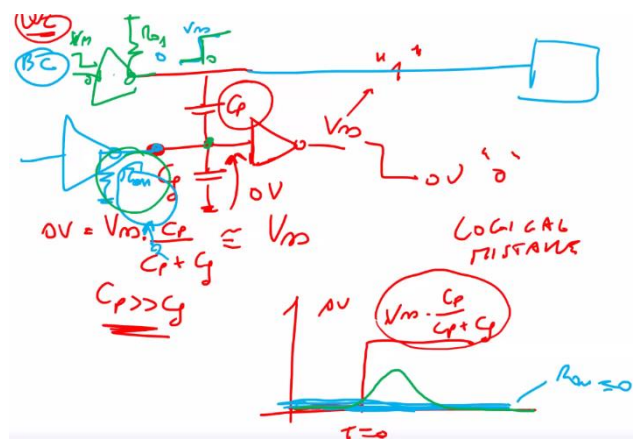
So we have the upper node that moves from 0 to V_{dd} , then the parasitic C_p and the C_g of the gate, which initially is at 0V, so the output of the inverter is V_{dd} . I'm considering the input of the inverter floating. The input of the inverter increases according to the capacitive voltage divider: $V_{dd} \cdot C_p / C_p + C_g$. If $C_p \gg C_g$, the ratio is almost V_{dd} , so we have in output of the inverter not V_{dd} but 0V, and this is an unwanted logical mistake.

In a realistic scenario, the approximation $C_p \gg C_g$ doesn't hold and the input of the inverter is not floating. Also the variation of voltage is not due to an ideal voltage generator, but due to another inverter and so on. If we suppose that instead of a realistic inverter for the supply of the capacitive divider we still have a voltage generator, if the r_{on} of the input inverter is 0 Ohm, we have a short and the generator cannot change the voltage in input of the inverter. This is a means with which we can reduce noise, using a small resistance. This can be done increasing the area of the transistor, so the width.



Let's create now a real situation, so $r_{on} \neq 0$ and the 'aggressor' not an ideal generator. The input of my inverter in the end goes to zero because the r_{on} is very small, but we have a peak in the middle. If the pulse flips the inverter we have a logical mistake. To reduce the probability of error we can:

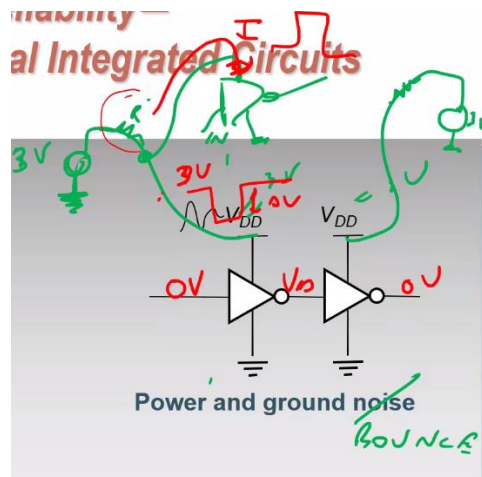
1. We improve the inverter reducing the output r_{on} of the input inverter. If r_{on} decreases, the pulse peak decreases.
2. We decrease the coupling capacitor C_p moving away the aggressor, so that C_p decreases because the dielectric increases.



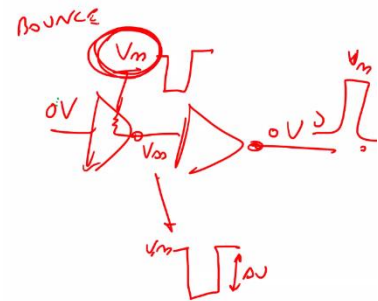
If the coupled variation is large enough, it can cause the inverter to change its output, from “1” to “0” in this case, determining a logic error. Obviously, the impedance of the driver is not infinite and the smaller it is the smaller the capacitive coupling is effective. The best case-scenario is an ideal driver, with a nil impedance at the output: in this case the capacitive coupling is not able to change the voltage at the input of the gate.

Power and ground bounce

We have two inverters in cascade but with two different PS. Let’s suppose that connected to the PS we have also other circuits that can switch on and off. When they turn on, the circuits sink current from the power line. Because of this current drawn, we have a bounce on the PS, so e.g. we go to 3V minus something.



At the input of the inverter we suppose to have 0V. In the inverter we have a pMOS and a nMOS transistor, and the pMOS is on, having 3V at the output (Vdd). The pMOS is hence working in the linear (ohmic) region. So it behaves as a resistance r_{on} , that is smaller the bigger the area. If I have a bounce of the PS voltage, the output node of the first inverter bounces the same, because we have a resistance. hence the second inverter receives in input a bounce. If the ΔV of the bounce is large enough, we can commute the output.



The same reasoning can then be repeated considering ground.

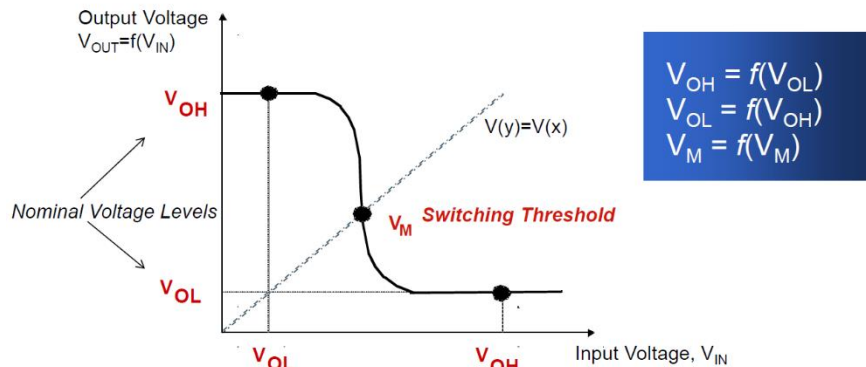
In the second case, we deal with bouncing of power supply or ground → **power and ground bounce**. In this case, there is an interference on the two rails or one of the two rails of the power supply. This can cause a bounce of the output voltage. You need to consider that the inverter output is connected through a low impedance path to voltage supply or ground depending on its state. This bounce is not a problem for the first gate but for the following gate, whose rails are supposed not to be affected by noise and disturbances. Thus, the output variation of the first inverter can reflect on the second inverter that can toggle.

Thus, the reliability is the capacity of a gate, or in general of a digital circuit, to be insensitive to digital noise. In particular, the concept of reliability is strictly related to the static performance of a gate, thus to its static characteristic.

Voltage transfer characteristic (VTC)

In reality, digital circuit are very immune to noise even with strong interferences. Its reliability and immunity to noise is quantified by the voltage transfer characteristic.

It quantifies the reliability of a circuit and it is the relationship between the input voltage and the output voltage. It is a steady state, DC characteristic.



The curve we get in the image is the one of an inverter, but we have also a transition region in the middle. V_{OH} is the **nominal output high voltage**. V_{OL} is the **nominal output low voltage**. In order to asses V_{OH} and V_{OL} we need to use a recursive formula by definition, that is the one in the blue box. In reality they will be V_{DD} and ground.

The difference between V_{OH} and V_{OL} is called **voltage swing**. The bigger the voltage swing, the better for the noise. So the voltage swing is somehow related to the reliability.

The other voltage is the threshold voltage, that it is on the bisector. On the left we have a logical 1 at the output and on the right a logical 0 at the output. V_M defines somehow if the output is 0 or 1.

Let us consider the simplest digital gate, the inverter. The inverter is a digital circuit that inverts the input logic signal. This means that if the input is high, the output is low, whereas if the input is low the output is high. We are talking about logic signals that can assume only two values, “0” and “1”. However, the inverter is a circuit which can receive at its input a voltage in the range between 0V and V_{DD} , that is, from ground to the circuit supply voltage. The voltage static characteristic, known as VTC (Voltage Transfer Characteristic), is the function that relates the output to the input voltage.

This slide shows the classical shape of the voltage transfer characteristic of an inverter. For a low input voltage, the output voltage is high and for a high input voltage the output voltage is low. In this characteristic, we can recognize some very important parameters: V_{OH} , V_{OL} , and V_M .

V_{OH} stands for “high output voltage”, and it is equal to the output voltage corresponding to the high level (i.e., a logical “1”), whereas V_{OL} stands for “low output voltage”, and it’s equal to the output voltage corresponding to the low level (i.e., a logical “0”). In other words, they are the nominal output voltage levels.

If we set the input of the inverter to V_{OL} , the output is V_{OH} and vice-versa. If we consider V_{OH} at the input of the inverter, the output voltage value is equal to V_{OL} . These voltage values can be evaluated in a recursive way being:

$$V_{OH} = f(V_{OL})$$

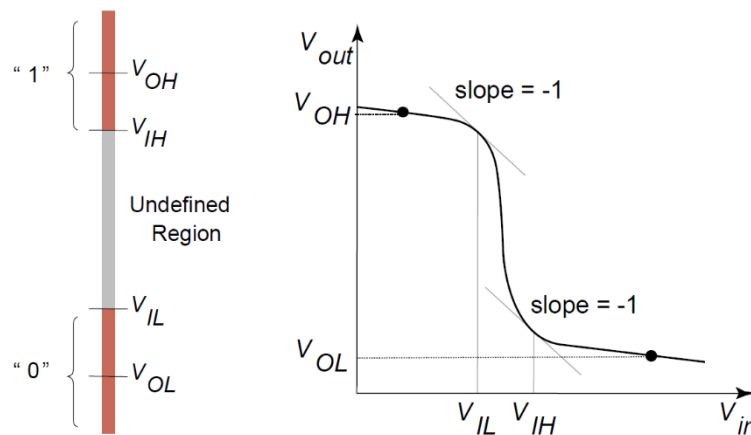
$$V_{OL} = f(V_{OH})$$

The difference between V_{OH} and V_{OL} is called signal swing.

The third important parameter of the VTC is the threshold voltage. This is defined as the input voltage that produces the same voltage at the output, i.e., $V_M=f(V_M)$. In other words, it's the abscissa and the ordinate of the intercept point between the VTC and the first quadrant bisector, the straight line that forms a 45° angle between the two axes. The threshold can be simply evaluated taking the circuit and connecting the input node to the output one.

Why this intermediate voltage is so important? Because it defines the trigger voltage of the inverter. If we consider a very steep characteristic, the threshold voltage marks two regions, one corresponding to a low input (and thus high output), and one corresponding to a high input (and thus low output). Thus, at a first order approximation, we can consider that for input voltages smaller than V_M the output is high and for voltages greater than V_M the output is low.

Mapping between analog and digital signal



We have 3 regions: flat, transition and flat. The flat regions correspond to logical 1 and logical 0. If the inverter is an analog circuit, the dV_{out}/dV_{in} is the small signal gain. In the flat region it is close to 0, in the transition region it is very large. Hence the flat region is defined up to the point where the gain is -1. These points define the V_{IL} (low input voltage, **the maximum input voltage recognized as logical 0**) and V_{IH} (high input voltage, **minimum input voltage recognized as logical 1**). The problem in defining the output is when we are in the transition region.

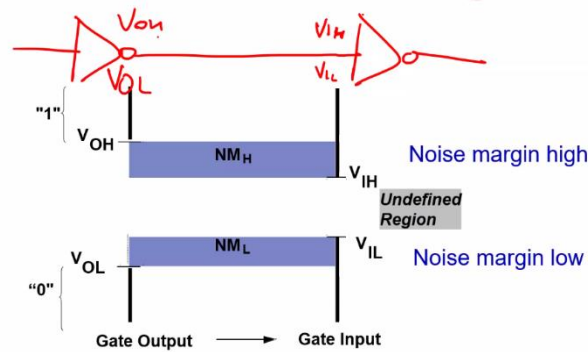
Let us consider the VTC of an inverter, which implies to consider the inverter as an analog circuit with an voltage input, V_{IN} , and a voltage output, V_{OUT} . We can define three regions: one region corresponding to the high logic level, one corresponding to the low logic level, and a transition or undefined region. The output voltage of the gate has to be in the two regions corresponding to "1" or "0" in order to correctly identify the logic values.

These two regions are defined by two voltages, V_{IL} and V_{IH} , which stand for "low input voltage" and "high input voltage", respectively. They correspond to the points on the characteristic having a slope of -1. In other words, they identify two flat regions in the voltage transfer characteristic, one corresponding to a low input (thus, high output) and the other one to a high input (thus, low output). We can consider V_{IL} as the maximum input voltage corresponding to a logic "0" at the input, and V_{IH} as the minimum input voltage corresponding to a logic "1" at the input.

DEFINITION OF NOISE MARGINS

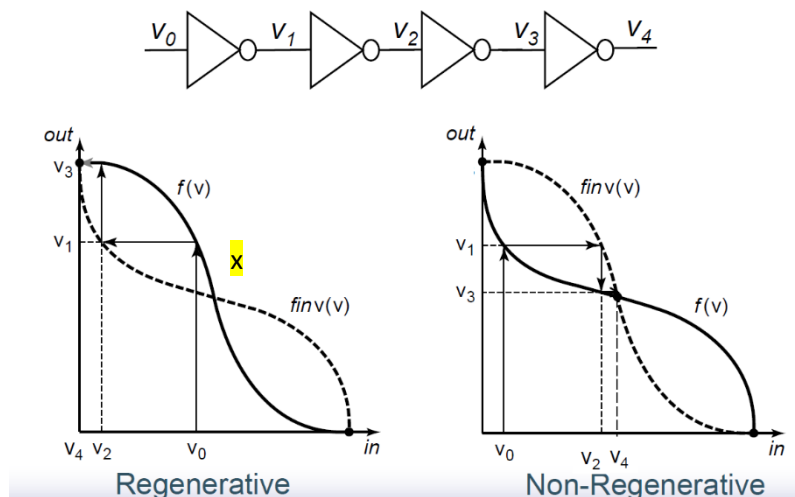
Let's consider two inverters. The output of the inverter at steady state can be either V_{OH} or V_{OL} . The next one is characterized by V_{IH} and V_{IL} .

The margin from V_{OH} to V_{IH} is the one that grants to have a 1 in the input of the second inverter and remain in the flat region. The $V_{OH} - V_{IH}$ is called **noise margin high (NMH)**. Same but opposite reasoning with V_{OL} and V_{IL} (**NML**). These noise margins quantify the reliability of the circuit, the larger the better.



We can define the so-called noise margins which assess the reliability of a gate. Let us consider that our reference inverter is fed by another inverter whose output can be "0" or "1". Thus, its output voltage, if no digital noise is present, can be V_{OH} ("1") or V_{OL} ("0"). The second inverter, our reference inverter, has a margin of $V_{OH} - V_{IH}$ to be sure that its input voltage is recognized as logic "1", and a margin of $V_{IL} - V_{OL}$ to be sure that its input voltage is recognized as "0". These two intervals are called noise margin high (NMH) and noise margin low (NML), respectively. They represent the maximum level of "noise" that a gate can tolerate at its input terminal without being affected by a logic error. Obviously, they have to be as large as possible.

REGENERATIVE PROPERTY



The inverter has the margins previously introduced to be immune to noise, but moreover the digital circuits have also the regenerative property. Even with a strong noise way beyond the noise margin we can solve the problem and have the output, they will never remain in the transition region with the output, at steady state the output will always be V_{OH} or V_{OL} , even if eventually wrong.

In this case I'm considering 4 inverters in cascade. $f(v)$ is the voltage transfer characteristic of all the inverters (bold line). The dashed line is the inverse transfer characteristic ($finv(v)$), I'm basically switching the axes, it is just a mathematical trick. So one point on the axis will be V_{DD} , the other $0V$.

Let's suppose that V_0 has to be nominally $0V$, but let's suppose that because of digital noise we have something larger than $0V$. We know the value of $V_m(x)$. The other assumption that I make is that $V_0 < V_m$. V_1 is the voltage on the direct characteristic that I have in input of the second inverter. Now I use the inverse characteristic, so I can read V_2 from it. Now V_2 is the input of the third inverter, so I use the direct characteristic. As last the inverse one form V_4 .

So V_0 is affected by noise that is larger over the noise margins but smaller than V_m , and what happens is that the output V_4 is 0 like if there was no noise at the input.

If V_0 is so big that overcomes the switching threshold, the convergence is towards V_{dd} , it is a mistake but it is still convergent. The bottom line is that the circuits are able to regenerate. If the noise is not overcoming the V_m , the output is good.

To say that a digital circuit is able to regenerate we need that the intermediate region has a gain (slope) in absolute value greater than 1. So every digital circuit must have two flat regions and a transition region with a slope greater than 1.

On the right we have the opposite situation, so we are not regenerative.

A large noise margin is desirable but there is another property that we desire for a digital gate. It's the regenerative property.

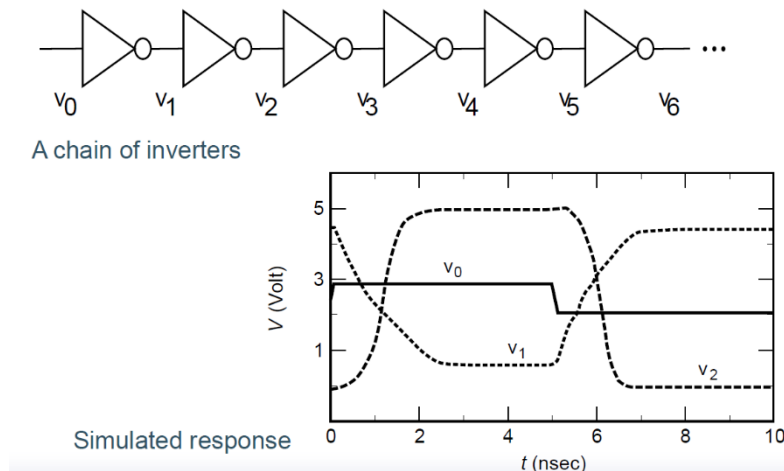
Let us suppose that the signal at the input of a simple gate, an inverter for example, is affected by digital noise. If the signal plus noise remains inside the margin there is no problem and the gate works fine even if the output voltage slightly differs from the nominal value, that is, V_{OH} or V_{OL} . On the contrary, if the noise is sufficiently high, it can bring the input signal outside the margin. However, even if the signal falls in the transition region, the output signal is regenerated if we consider some gates, inverters in our example, in cascade. This happens if the inverters show the regenerative property. Let's explain the concept referring to four inverters in cascade. Let's plot on the same graph both the direct function $V_{OUT}=f(V_{IN})$ and the inverse function $V_{IN}=f_{inv}(V_{OUT})$. This second function allows to evaluate the input voltage at the gate once its output voltage is known. So, for what concerns the inverse function, we read the output voltage on the horizontal axis and the corresponding input voltage on the y-axis.

Now, let us suppose that V_0 ideally has to be V_{OL} , ground in the case shown in this slide. Because of an interference, this voltage becomes larger and falls in the transition region (but below the switching threshold voltage, V_M). The voltage V_1 can be evaluated considering the direct function. Now, since V_1 is the input voltage of the second inverter, its output voltage V_2 can be inferred considering the inverse function. The voltage V_3 can be again evaluated considering the direct function. Now, since V_3 is close to V_{OH} , V_4 is close to V_{OL} . Thus, after an even number of inverters we have regenerated the initial value of V_0 , being now the voltage close to $0V$.

It's worth pointing out two remarks. First, the regenerative property consists of a transition region with a negative slope and an absolute value larger than 1. Second, in order to reestablish the correct value, the noise has not to bring the initial value over the threshold. The interference can bring the input signal inside the transition region but not over the threshold voltage, otherwise the regenerated signal is the opposite one, V_{OH} in this case.

Let us consider another example, plot in the right graph of this slide. Also, in this case we plot both the direct characteristic as solid line and the inverse function as dotted line. The inverter does not own the regenerative property since the transition region has a negative slope, but its absolute value is not larger than 1. It's almost flat.

In fact, let us consider again the four cascaded inverters. The input voltage of the first inverter affected by a disturbance is V_0 . In this case, it can be easily verified that the voltage V_4 is equal to the threshold, which is obviously in the middle of the transition region and thus it corresponds to neither “0” nor “1”.



If the inverters have the regenerative property, we notice that even if the input is not rail to rail (V_0), V_1 is already almost a full swing signal. After two inverters I have a full swing signal.

This slide shows the result of a simulation involving 5 cascaded inverters, all supplied with 5V and having a switching threshold of 2.5V. The first one is fed by a periodic signal with a reduced swing, which runs from 2.1V to 2.9V. Note that after two inverters the signal has been already regenerated.

KEY RELIABILITY PROPERTIES

- Absolute noise margin values are important since they measure the immunity of a circuit to interference and noise;
- “Noise” transfer functions and output impedance of the driver have to be minimized in order to increase the reliability of a circuit.

To maximize the immunity to noise we need to maximize the noise margins. Moreover, we can improve the reliability acting on the noise transfer function and on the output impedance of the driving inverter (r_{on}). So to reduce the effect of digital noise we can act on: noise margins, parasitic capacitances, driving circuits.

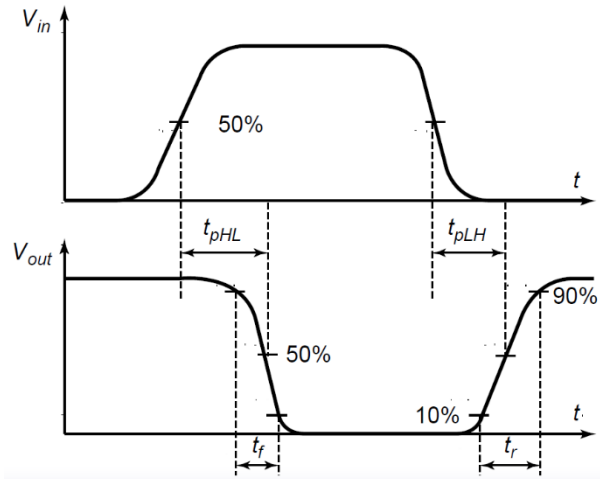
Thus, absolute noise margins are important since they measure the ability of a circuit to be immune to digital noise, i.e., interferences and supply bounces.

However, it’s always better to minimize, when possible, the “noise” transfer function and the output resistance of the driver in order to increase the reliability of a circuit.

As far as the driver of a circuit is concerned, it’s clear that a floating node is more prone to cause logic errors than a node forced by a low-resistance driver. This will be an issue in dynamic gates. Dynamic gates are based on the storage of a charge on a capacitor that then is left floating. Thus, they are very prone to interferences and noise coupling. Indeed, static gates are characterized by a low-impedance driving of the output voltage, which is always connected through a low impedance path to either VDD or ground. The lower the output resistance of the driver, the better. Obviously, this comes at the expense of a larger area (cost) and a larger power consumption (due to increased parasitic capacitances).

SPEED: PROPAGATION DELAY OR MAXIMUM OPERATING FREQUENCY

A digital circuit is typically characterized by a propagation delay between the input and the output. Let's consider an inverter, and input voltage and an output one. The input signal is a square wave. The propagation delay is the time that elapses from the crossing of half of the range at the input and half of the range at the output. In the image the t are in reality τ .



To simplify the analysis, we consider an ideal square at the input and a real one at the output. We use 50% as the value because typically it is where the switching threshold is placed. In fact, if V_m is not at half V_{dd} we compromise either the low or high noise margin.

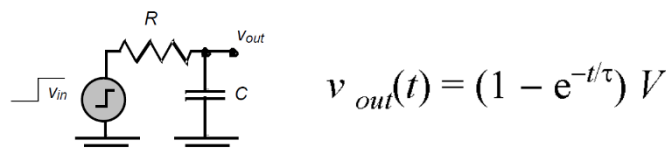
In general, since τ_{plh} and τ_{phl} are different, the propagation delay τ_p is the average of the two.

Let us consider the input signal of a gate, an inverter for example, and its corresponding output signal. To define the propagation delay we need to consider the point where the curve reaches the 50% of the swing both for the input and the output signal. The time interval between these two points is the propagation delay. This 50% of the swing is not chosen randomly. It has a meaning since the threshold is usually chosen to be in the middle of the supply range, or in other words at $V_{DD}/2$, in order to maximize the noise margins (thus, crossing 50% of the swing means crossing the threshold causing the inverter to toggle).

We will distinguish between a propagation delay from low to high (referring to the output transition), and from high to low. Thus, we have τ_{pLH} and τ_{pHL} , respectively. The arithmetic mean between these two values is the so called "propagation delay" of a gate.

Other two key times are the rise and the fall time, which are defined as the time interval between the 10% and the 90% of the swing.

A first order RC network



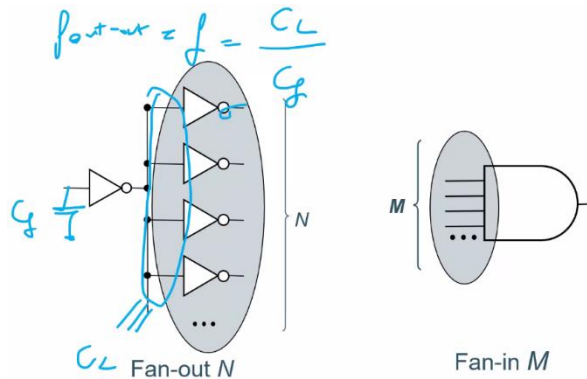
$$t_p = \ln(2) \tau = 0.69 RC$$

Important model – matches delay of inverter

In digital electronics a transistor is always modelled as the combination of a resistance and a capacitor. For steady behaviour we have a resistance, in the case of transient we model it as a capacitor. If we apply a step in input of an inverter, the output will have an exponential transition, and to assess the propagation delay we need to assess the cross of 50% of the V_{DD}, that is nothing else than 0.69*RC.

FAN-OUT AND FAN-IN

If we consider a circuit with a propagation delay of tau_p from input to output, the maximum operating frequency will be f_{max} = 1/tau_p. Knowing tau_p, the maximum bit rate we can feed to the digital circuit is 1/period of one bit, and the period of the bit is tau_p, so the bit rate is f_{max}.



Fan-out will be indicated as f and it is the load capacitance divided by the gate capacitance: $f = C_L / C_g$, so it is the ratio of capacitances. Fan-in is the number of inputs.

In order to define the ideal digital inverter, let's introduce the concepts of fan-out and fan-in. The fan-out is the number of ports connected to the output. Typically, it's defined as the ratio between the capacitance connected to the output node (due to the external load, the so-called extrinsic capacitance), and the input capacitance of the gate. We will come back to this definition in the next lessons.

The fan-in is the number of input terminals. The inverter has a fan-in of 1, whereas if we consider a 2-input NAND gate, its fan-in is 2. The gates with high fan-in tend to be more complex causing both static and dynamic performance to worsen.

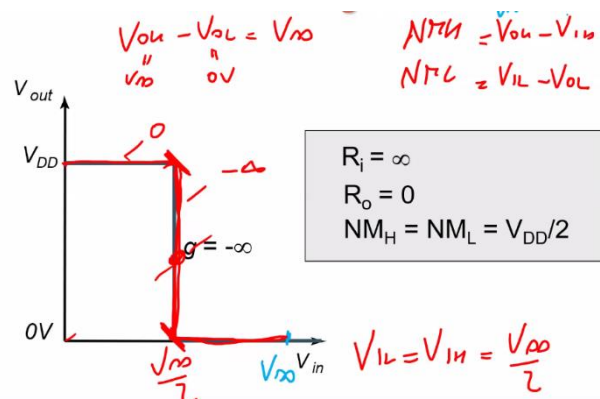
THE IDEAL INVERTER

The one in the image (red) is the ideal characteristic of an inverter. In fact, I have the largest possible swing (difference between V_{OH} and V_{OL}), that is V_{DD}. Moreover, the switching threshold is exactly half V_{DD}. It is important to have it in the middle because of noise margins.

$$NM_H = V_{OH} - V_{IH}$$

$$NM_L = V_{IL} - V_{OL}$$

In this case $V_{IL} = V_{IH} = V_{DD}/2$, so both the noise margins will be V_{DD}/2.

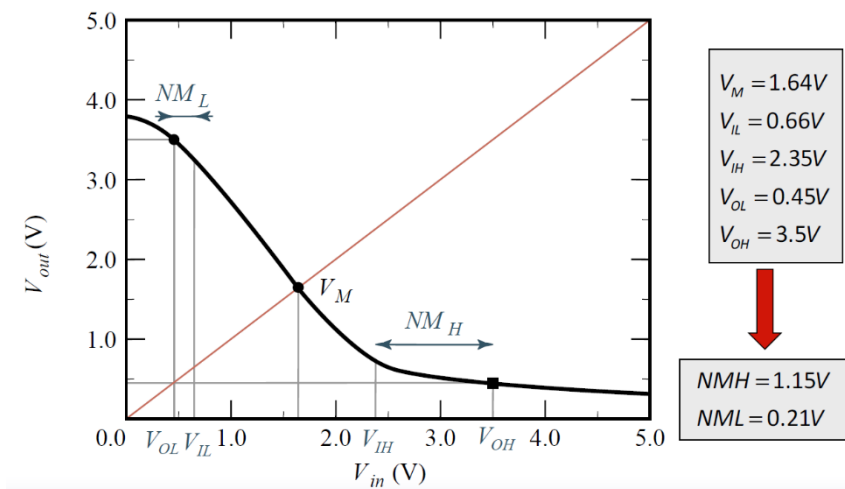


If there is noise on the ground, do we prefer to maximize NM_H or NM_L? NM_L, so maybe we can change the position of the switching threshold, greater than V_{DD}/2. So we need to custom design the inverter and not using libraries, which are putting the threshold in the middle.

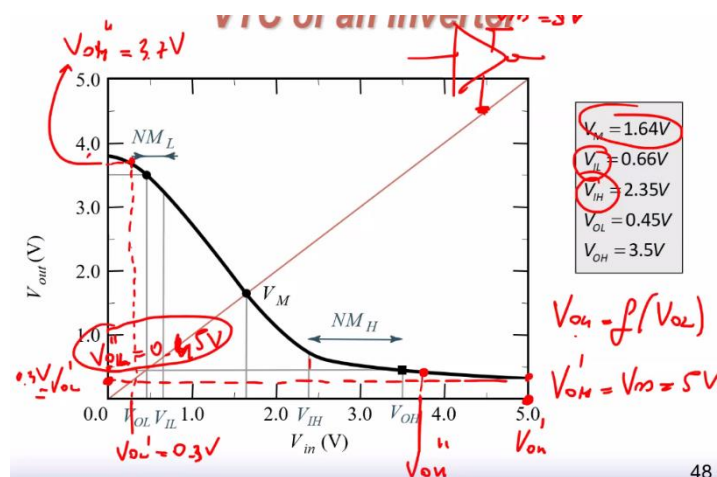
Moreover, the ideal inverter has an input infinite impedance and 0 output impedance (in reality 100 to 10k). In reality we have a capacitive impedance in the case of an inverter.

VTC of an inverter

In the image we have a very bad inverter and we need to assess all the static parameters. The switching threshold is easy to be found, we need the bisector $\rightarrow 1.64V$. PS is supposed to be between 5V and ground. The other parameters we can find are the V_{il} and V_{ih} , drawing the slope of -1. Moreover, the gain in the transition region is not so high, more or less -1.5.



To compute V_{oh} and V_{ol} we cannot use the PS value, nor the recursive formula. So we make a guess. I suppose that $V_{oh}' = V_{dd}$ and I measure a first estimate of V_{ol}' , $0.3V$. This estimate is put at the input of the inverter and assess $V_{oh}'' = 3.7V$. Now we use V_{oh}'' at the input and estimate V_{ol}'' . In the end, $V_{oh} = 3.5V$.



This inverter is very bad because the swing is not equal to the PS, the switching threshold is not in the middle but very low, and this results in two different noise margins, with the NML that is very small.

This slide shows the VTC characteristic of a real inverter. In truth, this is an old inverter, but it can be considered as an example.

On this characteristic we can evaluate all the parameters of interests: V_M , V_{OH} , V_{OL} , V_{IH} and V_{IL} . Starting from the threshold voltage, this parameter can be evaluated designing the bisector of the first quadrant. V_M is equal to $1.64V$. Now, we can sketch the two voltages V_{IL} and V_{IH} remembering that they correspond to the point of the VTC having slope equal to -1. Thus, $V_{IL} = 0.66V$ and $V_{IH} = 2.35V$.

Graphically we can derive also V_{OH} and V_{OL} , which are equal to 3.5V and 0.45V respectively. These two values can be computed by means of a recursive analysis, starting from an initial guess of $V_{OL}=0.3V$, for example. At this voltage it corresponds $V_{OH}=3.7V$. Then, we can use this voltage to improve the estimate of V_{OL} , obtaining $V_{OL}=0.45V$. Since the characteristic is rather flat in this region, this estimate is already correct. Finally, from $V_{OL}=0.45V$ we obtain $V_{OH}=3.5V$.

The two noise margins result: $NMH=V_{OH}-V_{IH}=1.15V$ and $NML=V_{IL}-V_{OL}=0.21V$.

Obviously, this is not a good inverter for what concerns the static characteristic. The threshold is not at $V_{DD}/2$ and the two noise margins are not symmetrical. The NMH is large, whereas the NML is very narrow.

POWER CONSUMPTION

The power consumption is always related to the Power supply generator, it is the power wasted by it. It is the power delivered to the circuit by the power supply generator.

Instantaneous power:

$$P(t) = v(t)i(t) = V_{supply}i(t)$$

Peak power:

$$P_{peak} = V_{supply}i_{peak}$$

Average power:

$$P_{ave} = \frac{1}{T} \int_t^{t+T} P(t) dt = \frac{V_{supply}}{T} \int_t^{t+T} i_{supply}(t) dt$$

Instantaneous power

Product of the V_{dd} value and the current.

Peak power

We consider the maximum current delivered by the power supply.

Average power

What matters is the average power, which establishes the heat to be removed-

It is what counts in digital electronics. It sets the duration of the battery. It is the average power wasted by the power supply generator.

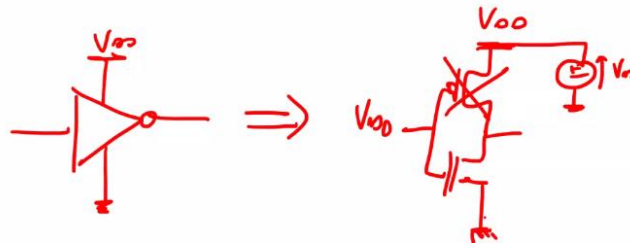
It is the integral over a period T and we need to evaluate the average current drawn by the power supply.

This average power is defined as the average product between power supply and current. Since the power supply is supposed to remain constant, the average power consumption can be estimated as the product between power supply voltage and the average current drawn by the supply, evaluated here as the integral over a period T (normalized to T) of the supply current. We can distinguish between static and dynamic power consumption.

The former is due to the current that flows from positive rail to negative rail (due to a leakage for example), the latter is the power associated to a commutation and it's the main contribution to the overall power being proportional to the operating frequency. This is mainly due to the current flowing from the supply to charge the output capacitance of the gates.

Let's suppose to deal with an inverter featuring a high input. Now, if the input is pulled down, the output is pulled up, thus the output capacitance of the inverter is charged to VDD. This causes a current to flow from power supply toward the capacitance, thus causing a power consumption. Conversely, if the input is driven high, the output capacitance is discharged to ground, but no current is drawn from the power supply, which does not waste power.

Let's consider an inverter and that there is no current drawn from the PS if the circuit is steady. If we consider 0V at the input, the nMOS is off, so there is no current that can be drawn, because the pMOS is in series with an open circuit. The same is true with a logic 1 at the input.

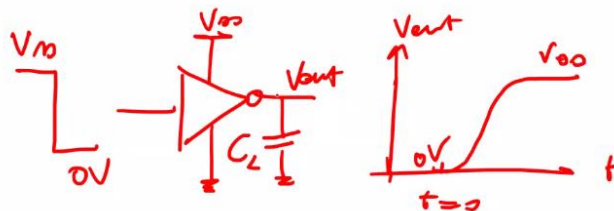


There is still however the leakage current, but it is negligible (pA). There is current only during a transition at the input, from 0 to Vdd (pull up) or from Vdd to 0 (pull down).

Input Pull-down

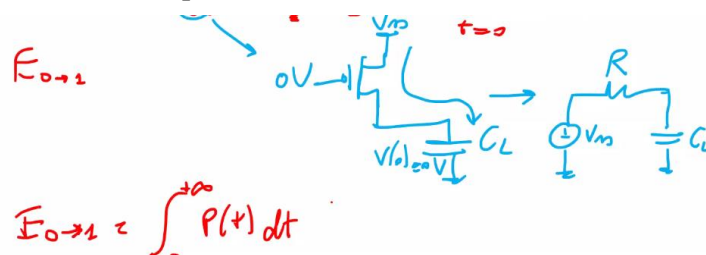
From logical 1 to logical 0. At the output I should have a transition in the opposite way, from 0V to Vdd. Let's try to assess the energy for the 0 → 1 transition. It is the energy spent by the power supply generator to perform the transition at the output, in fact we are dealing with the output.

In fact, at the output we have a capacitor C_L. The transition correspond to a current drawn from the PS.



During the pull up transition of the output it is the pMOS that comes into play. The pMOS is connected to a capacitor C_L across which at t = 0 we have 0V. The current across the pMOS charges the capacitor. I want to assess the energy spent by the PS generator for this charge event.

The transistor can be represented as a resistor R, and I don't care about its value, because I care about the capacitance. E_{0→1} is nothing more than the integral of the power; since I have just this event, I can consider the integral of the instantaneous power P(t) in dt.



The power is the one of the PS generator, that is Vdd, and the current is i(t) in the pMOS. Vdd can then be taken out of the integral and we have just the integral of the current. i(t) flows in the capacitor, and it is C_L * dVout/dt.

Now it can be simplified and we change the extremes of the integral because dV_{out} goes from 0 to V_{dd} .

$$E_{0 \rightarrow 1} = \int_0^{+\infty} P(t) dt = \int_0^{+\infty} V_{DD} \cdot i(t) dt = V_{DD} \cdot \int_0^{+\infty} C_L \frac{dV_{out}}{dt} dt = V_{DD} \int_0^{V_{DD}} C_L dV_{out} = C_L V_{DD}^2$$

The energy depends only on the PS voltage and the capacitor we are charging, not on R.

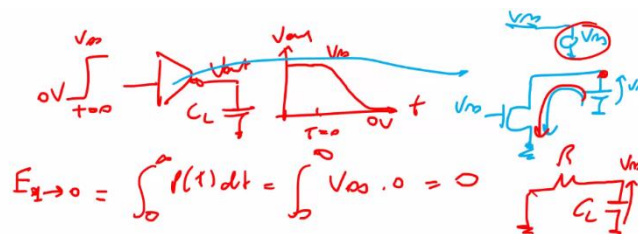
Moreover, the capacitor is charging, so there is energy in it. The energy in the capacitor is E_c and the power is P_c . I want to assess the energy absorbed and stored in the capacitor.

$$E_c = \int_0^{+\infty} P_c(t) dt = \int_0^{+\infty} C_L \frac{dV_{out}}{dt} \cdot V_{out} dt = \left[\frac{V_{out}^2}{2} \right]_0^{V_{DD}} \cdot C_L$$

The energy spent by the PS is in the $C_L \cdot V_{DD}^2$ term, and half is stored in the capacitor. The other half? It is dissipated in the resistance in heat, but it is independent on the resistance.

Input Pull-up

Now the nMOS transistor is on. Now the current of the PS generator is 0 because there is a pull down at the output, so there is no current provided from the PS.



At the beginning, the C_L is charged to V_{DD} and it is discharged. The energy in the capacitor is dissipated in the resistance R of the model circuit through heat. E_c has the same value as the previous case.

If we suppose to have a signal that is a series of square waves (alternation between 0 and 1), which is the average power wasted by the inverter? The PS generator wastes power only if it has to charge the capacitor, not if it has to discharge it. So only if we have a transition $E_{0 \rightarrow 1}$.

The power is the energy multiplied by the frequency with which the transition occurs, e.g. $f_{0 \rightarrow 1}$, which is called **pull up frequency**. It is not the frequency of the square wave, but the average frequency with which we charge the capacitor. And the pull up frequency is somehow proportional to the clock frequency.

$$P = E_{0 \rightarrow 1} \cdot f_{0 \rightarrow 1} + \cancel{E_{1 \rightarrow 0} \cdot f_{1 \rightarrow 0}}$$

The PS spends energy only for the pull up, because in the pull down the charge goes directly into ground

This is the formula for the **dynamic power consumption P_d** . The heat dissipated during the transition is independent on the resistance value.

THE MOSFET TRANSISTOR

With MOSFET we can implement very good switches. In an analog circuit I would use a BJT, which is a better current generator (larger gain, faster and bigger output resistance).

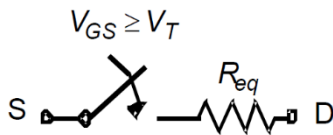
In digital electronics we care about switches, not current generators, but we need switches to connect a load to V_{DD} or to ground. MOSFET can be turned on and off very easily, so they are good. Moreover, in CMOS, pMOS and nMOS are good as well, even if typically $u_n C'_{ox} < u_n C'_{ox}$.

Moreover, we can have no static power consumption with MOSFET wrt BJT. They are also smaller and their production process is simpler, so we need fewer masks.

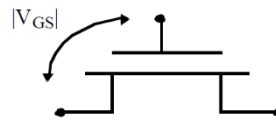
CMOS technology	bipolar technology
1) Better switches	1) Current drive capability
2) Good complementary devices (PMOS and NMOS)	2) Faster devices
3) No static power consumption	3) Large g_m and output resistance (analog features)
4) Larger integration density	
5) Simpler process (less masks, thus cheaper)	

PURSUING A GOOD SWITCH

Switch



An NMOS Transistor



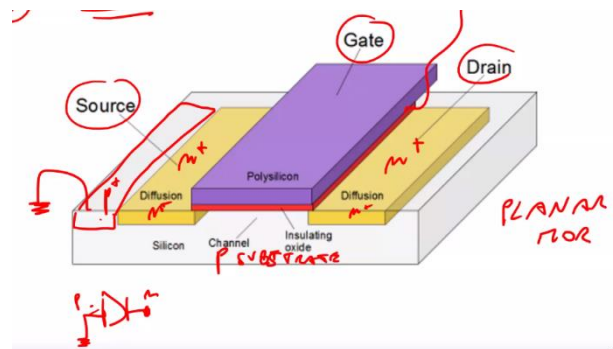
In digital electronics, a transistor operates as:

- open-circuit (switch off) with $V_{GS}=0$
- short-circuit (switch on) with $V_{GS}=V_{DD}$

We want a device that performs as a switch in series with a resistance. in the case of a MOS, if $V_{gs} > V_t$ the device closes. The short circuit is in reality a very small resistance. we want switches that, once closed, connect the V_{out} either to V_{DD} or ground.

THE MOS TRANSISTOR

The one in the image is a nMOS transistor. We have the two n⁺ regions diffused with the self-aligned process, the red one is SiO₂ and then we have above the Polysilicon, which acts as the gate. Everything is implemented on a p substrate, which is slightly doped.

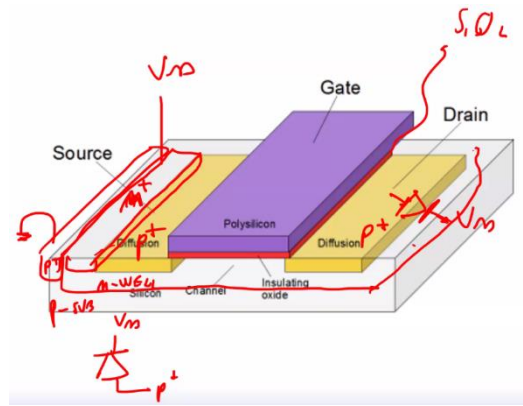


There is something missing, because we need to implement a p⁺ region to bias also the bulk. It is biased to the lowest voltage that we have available, that in our case is ground. The substrate is in common to all the devices and we want the pn junctions reverse biased, so the only way is to have the bulk to ground. So the MOSFET is a 4 terminal device.

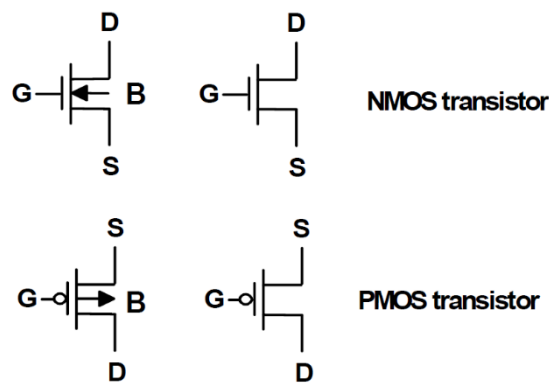
As for the pMOS transistor, it is the same but the main difference is that the substrate is still p doped, but in the p substrate we implement a n well region, lightly doped with phosphorus. In the n well we implement two p+ wells for source and drain. Then the rest is the same but changing the doping.

In the n well we also implement a n+ diffused region to bias the n well at the most positive power supply V_{dd} to have a reverse bias junction.

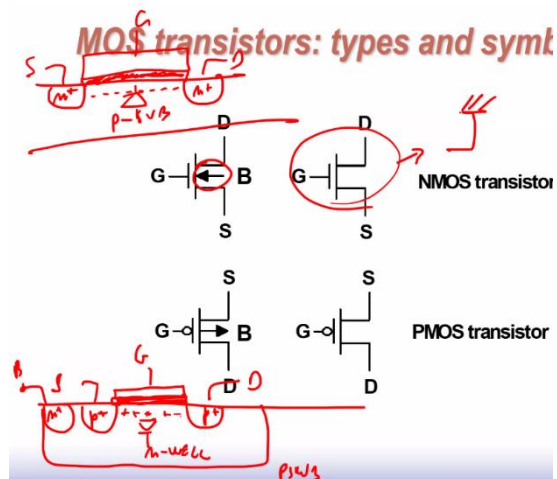
Furthermore, we need to add also a p+ region to ground for the substrate.



Symbols



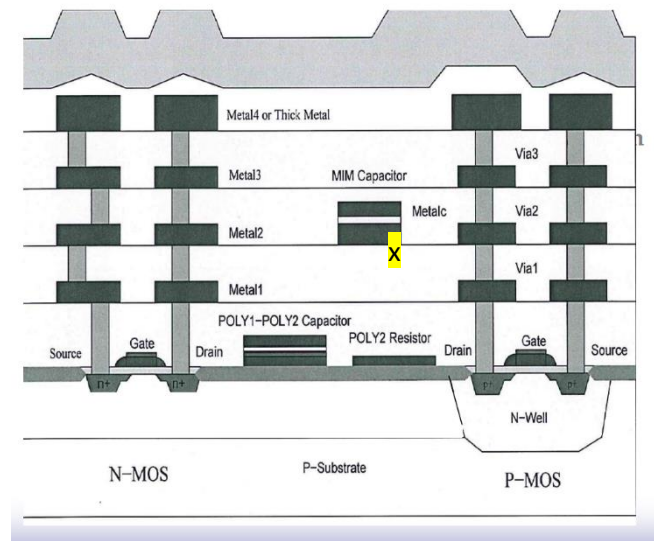
The body of the transistor is represented as a diode because if we cut and we make a section of the transistor, we have something like a diode (because of charge distribution). If the bulk is not represented it is supposed as connected to ground (for the nMOS, V_{dd} for the pMOS).



In digital electronic we never indicate where the source is, unless we want to investigate the transients. In all the other cases we have the symbols as in the image, without the source indicated. This because in digital electronic, if the signal is steady, there is no current, while in analog electronic we have the bias current and so we need to indicate the source. Now we care about switches.

The dot on the gate of the pMOS means 'active low', that is the fact that the mosfet is active with low voltages on the gate.

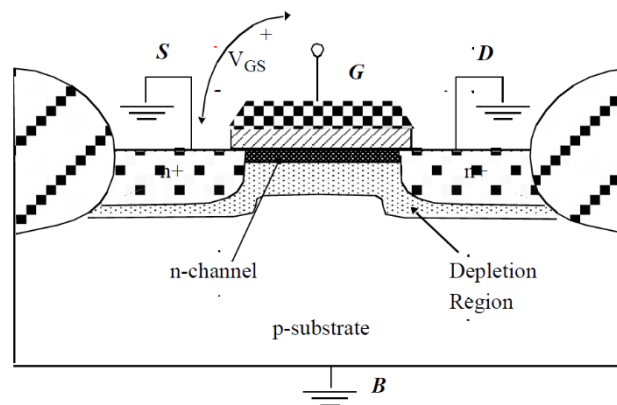
CROSS SECTION OF AN IC



We have the Silicon layer where we implement the n^+ and p^+ wells. The body terminal is not represented, so we should add a p^+ region connected to ground and a n^+ region connected to V_{dd} . The metal1, metal2, metal3 etc. are interconnections (we are in section). The white regions are SiO_2 . The light grey portions are vias, with which we interconnect contacts at different level of metalization. The first via in contact with the silicon (n^+ or p^+ region) is called **contact**, not via.

Furthermore, also analog passive devices are represented (x). It is a capacitor. The other one is POLY1-POLY2 capacitor.

THRESHOLD VOLTAGE



If $V_g = 0$ we create a depletion layer below the gate, positive charges are removed because holes are attracted by the grounded bulk and the depletion region has a negative fixed charge. It is a fixed charge. If I apply a voltage at the drain greater than 0, nothing happens because there is no free charge below the SiO_2 .

Now $V_g > 0$. At a certain point I create free electrons below the gate that come from the source and the drain. These are free charges, so if $V_{ds} > 0$ the electrons can move from the source to the drain and we have a current. The threshold is the voltage to apply to the gate to have a density of e^- under the gate equal to the original concentration of ions, so that we invert the channel. So we create an inversion layer equal to the original concentration of ions of acceptors in the substrate.

$$V_T = \Phi_{GC} - 2\Phi_F - \frac{Q_B}{C_{OX}} - \frac{Q_{OX}}{C_{OX}} \quad \text{x}$$

Φ_{GC} = work function difference between gate and channel

Φ_F = Fermi voltage (-0.3V)

Q_B = depletion layer charge

Q_{OX} = eventual charge trapped in the oxide

If a source-to-body voltage, V_{SB} , is applied:

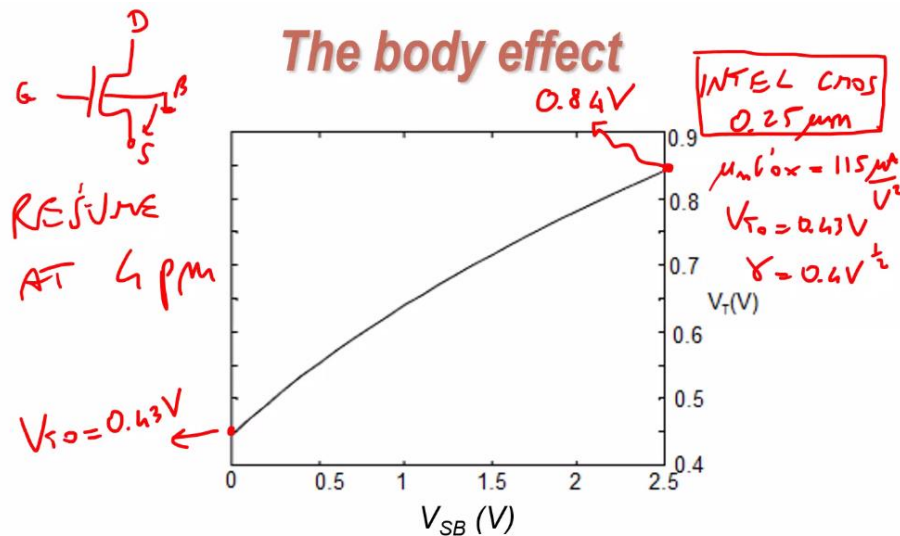
$$V_T = V_{T0} + \gamma \left(\sqrt{|-2\Phi_F + V_{SB}|} - \sqrt{|-2\Phi_F|} \right) \quad \text{y}$$

γ = body effect coefficient

The nominal threshold V_{T0} is x. The nMOS transistor is a 4-terminal device and if the source is not at ground and we have a body to source voltage, we have the body effect and V_T increases. V_T increases as much as we increase the V_{SB} , according to y. γ is the body effect coefficient that in our case is $0.4V^{1/2}$. $-2\Phi_F = 0.6V$.

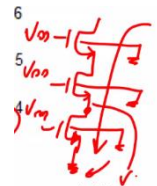
Since V_{SB} , V_T increases always wrt V_{T0} . In digital electronics we want a small threshold to have a good short circuit, but if we have a small threshold it is difficult to turn off fastly the device (the smaller the threshold voltage, the larger the leakage current).

THE BODY EFFECT

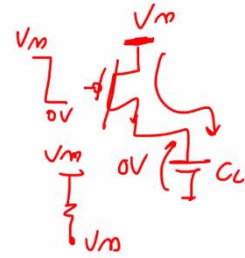
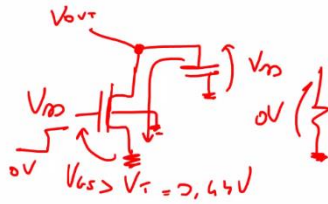


On the right it is indicated the reference technology, with nominal $V_{DD} = 2.5V$. When V_{SB} is equal to 0, $V_T = V_{T0}$. Increasing V_{SB} , the V_T increases.

Why do I have to have the source at a voltage larger than ground? In an AND gate we have nMOS transistor in series. The bottom one has the source to ground, but the others above not, and so I have the body effect. During the transient there is a current in the transistor, and we have the body effect, which reduces the driving capability of the transistor.

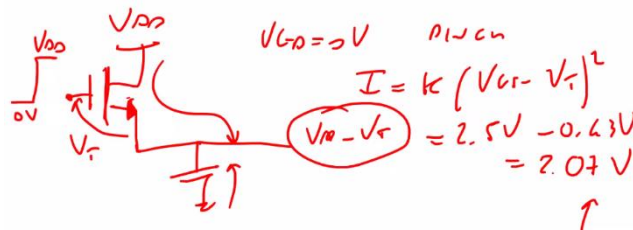


Let's consider a nMOS transistor connected to a capacitor which is to V_{DD} . If the voltage on the gate of the nMOS reaches V_{DD} , $V_{GS} > V_T$, the transistor is on and we have a current that discharges the capacitor. At the end of the transient, $V_{out} = 0$. This because in DC at steady state the capacitor is open, so no



current flows, so the transistor is ohmic and 0V across drain and source. Hence the transistor can pull the node to ground. In the opposite situation we consider a capacitor that is charged up to Vdd and a pMOS.

Let's see what happens if we use a nMOS to perform a pull up and a pMOS for the pull down. Let's start from the former. What happens is that once I have Vdd on the gate we have a transient, so we need to indicate which is the terminal that behaves as a source, and it is the one connected to the capacitor. Current starts to flow from Vdd to ground (always). The voltage on the capacitor increases. When we reach $V_{dd} - V_t$ the current is 0. This is why the nMOS transistor is not used for pull-up, because we want the largest possible swing to have the largest noise margin.



But the voltage won't be 2.07V because V_t is not 0.43V, which is the nominal threshold voltage, but I have to consider the body effect, so $V_t \neq V_{t0} = 0.43V$, because the source voltage changes.

$$V_{DD} - V_{t0} - \gamma \left[\sqrt{0.6 + \frac{V_{DD} - V_t}{V_{SB}}} - \sqrt{0.6} \right]$$

Since the V_{sb} is $V_{dd} - V_t$, I'm stuck and I need an iterative procedure. I start with $V_t = 0.43V$ and $V_{sb} = 2.07V$ and get a second estimate of V_t .

$$V_t' = V_{DD} - V_{t0} - \gamma \left[\sqrt{0.6 + \frac{2.07V}{V_{SB}}} - \sqrt{0.6} \right] = 0.774V$$

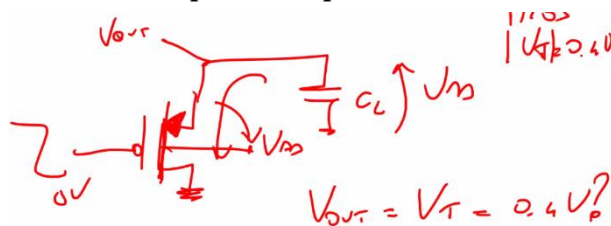
$$V_{m} - V_t' = 2.5V - 0.774V \approx 1.73V$$

Then I go on.

$$V_t'' = V_{DD} - V_{t0} - \gamma \left[\sqrt{0.6 + \frac{1.73V}{V_{SB}}} - \sqrt{0.6} \right] = 0.73V$$

I continue with the iterations and the final point that the output voltage reaches, after a reasonably time, 1.764V, which is different from the 2.5V that I can use if I would have used a pMOS for the pull-up.

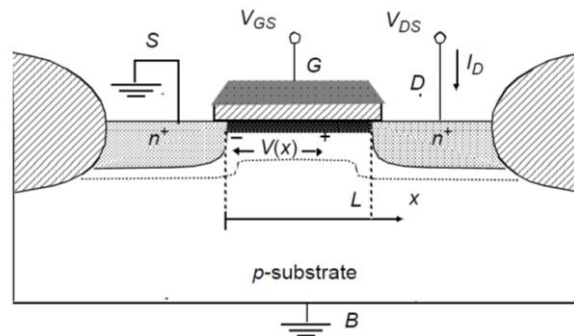
The same reasoning can be done with the pMOS for pull-down.



Performing the calculations, we get $V_{out} = 0.7V$ more or less.

TRANSISTOR IN OHMIC (LINEAR) REGION

We have free electrons both at the source and drain side. This means that the gate to drain voltage is greater than V_t .

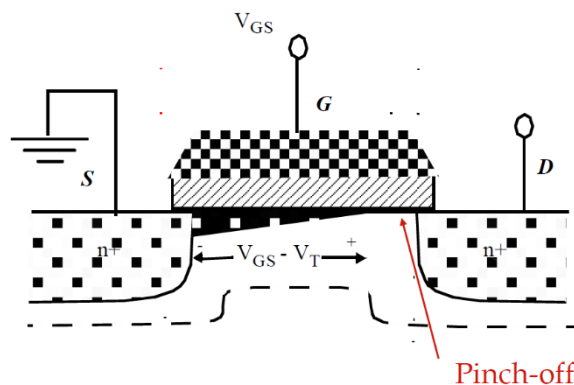


$$V_{GS} > V_T \quad (\text{transistor is on})$$

$$V_{GD} > V_T \Rightarrow V_{DS} < V_{GS} - V_T = V_{OV}$$

PINCH-OFF SATURATION

The pinch off region of the transistor is met when there are no more free electrons at the drain side. This doesn't mean that the current is 0.



$$V_{GS} > V_T$$

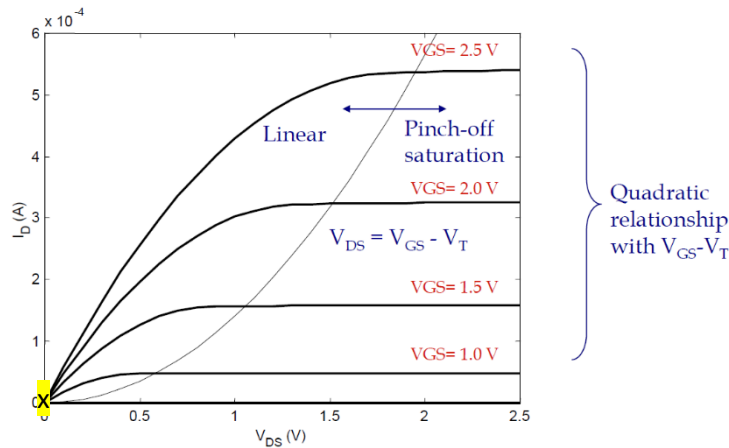
$$V_{GD} < V_T \Rightarrow V_{DS} > V_{GS} - V_T = V_{OV}$$

The transistor has to be on to be in pinch off and then $V_{gd} < V_t$. The voltage where the channel is pinched off, across the channel, is V_{ov} . In the pinch off region, the current depends quadratically on the overdrive voltage.

The current depends quadratically on the V_{ov} because it is proportional to the charge across the channel Q and to the velocity of the carriers along the channel. The charge is a density of charge Q' , which is proportional to $C'_{ox}(V_{gs} - V_t)$; the electrons move from source to drain and the velocity is proportional to the mobility and the electric field across the channel. The electric field across the channel is the voltage drop across the channel divided by the length of the channel. But the voltage difference across the channel is V_{ov} . If I put them together I have in both a dependence on V_{ov} , so finally the dependence is quadratic.

The final characteristic of a transistor is the following.

The parabola defines the ohmic region and the pinch off region. The parabola corresponds to the locus of point for which we enter the pinch off region, which is $V_{gs} - V_t$. It is the minimum V_{ds} for which we have the pinch off region.



In the linear region we have a strong dependence on V_{ds} , while in the saturation region we have a small dependence on V_{ds} because of the Early effect (channel modulation effect, due to the fact that the pinch off point moves), so the current slightly increases.

Equations

$$V_{DS} > V_{GS} - V_T = V_{OV} \quad \text{saturation region (pinch-off)}$$

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

$$V_{DS} < V_{GS} - V_T = V_{OV} \quad \text{ohmic region}$$

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

A region that is missing is the cut-off region, where $V_{gs} < V_t$ and the transistor is off, $I_{ds} = 0$. For the saturation formula we use λ that is $1/V_a$, where V_a is the Early voltage.

We can notice that in linear region, if we neglect the V_{ds}^2 term, we have the equation of a resistance, and we can get the on resistance: $r_{on} = 1/(\mu C'_{ox}(W/L) \cdot V_{ov})$.

$\mu C'_{ox}$ is called **process transconductance**.

Let's neglect the quadratic term in linear region. Even if $V_{gs} > V_t$, the current can still be equal to zero even if the transistor is on. This happens if $V_{ds} = 0$, so we are working in point x of the previous image. This is the working point of transistor in digital circuits. So this condition occurs either if the transistor is off or if it is on but $V_{ds} = 0$.

Carrier velocity saturation

So far we have considered the following equation.

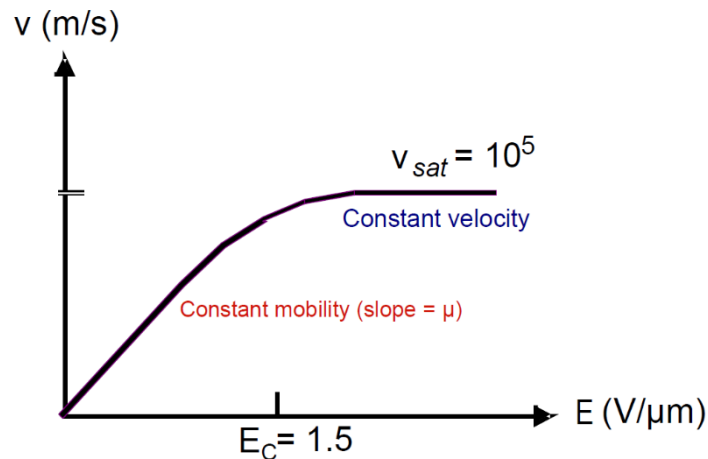
The channel can be ohmic, pinched off or cut off. Moreover, the fact that the velocity is proportional to the electric field is valid for any technology but now might have carrier velocity saturation. In fact, if we increase the E across a

$$v = \mu E = \mu \frac{\Delta V}{L}$$

ΔV IS THE VOLTAGE ACROSS THE CHANNEL
 $\mu = 10^5$
 velocity

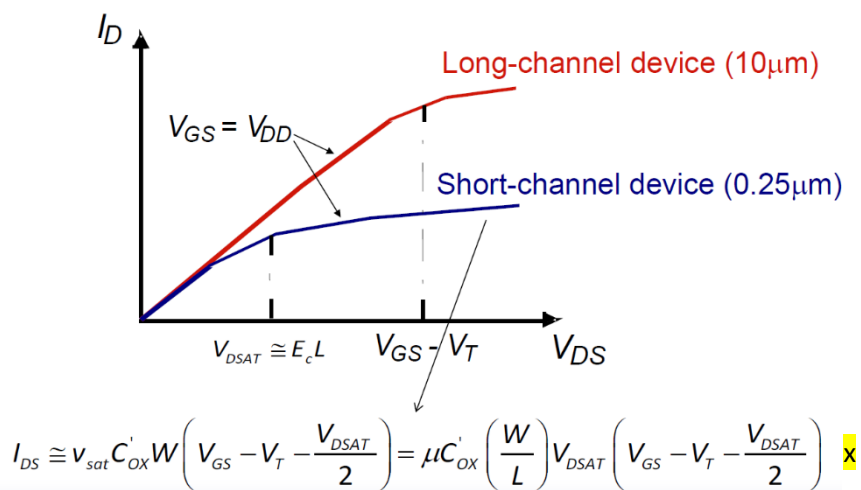
we

semiconductor, when we reach a critical electric field the velocity saturates, we reach a plateau and it saturates to the value in the image.



This is the reason for which current becomes linear with the overdrive voltage (no more squared in saturation) and this is what happens in modern technology

Drain current in velocity saturation region



I consider INTEL 0.25 um CMOS technology with $W/L = 10\mu/10\mu$ and a device with the same aspect ratio, but $0.2\mu/0.2\mu$.

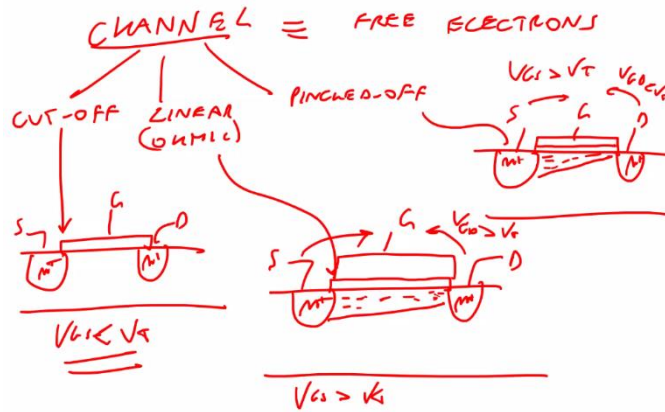
We expect to have the same behaviour in terms of characteristic, because the current depends on the aspect ratio, but we have the short channel effects in the second case, so we have velocity saturation (blue curve).

If this effect comes into play, the current has the formula in the right side of the equation x. V_{dsat} is the voltage to apply across the channel to have the short channel effect. It is a number for each technology, for the 0.25um it is 0.63V. If we apply this voltage across the channel the velocity saturates.

Let's consider the channel (channel = having free electrons that can conduct a current) of a mosfet. We have three possibilities: cut-off, linear and pinched off.

The velocity saturation occurs at the current level, not at the channel level.

What about the voltage across the channel ΔV ?



In the linear region it is established by V_{ds} . In the pinched off condition it is $V_{ov} = V_{gs} - V_t$. So what counts to have saturation of the velocity of the carriers is that the corresponding ΔV divided by the length is equal to the critical electric field.

$$\frac{\Delta V}{L} = E_c \left(1.5 \frac{V}{\mu m} \right)$$

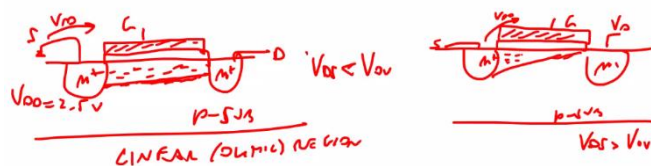
Hence if $\Delta V > E_c * L$, we have velocity saturation, and $E_c * L$ is a number. This also means that ΔV is proportional to the length. The higher the length, the lower the probability the velocity saturation occurs for high voltages.

The ΔV that makes the velocity saturate is $V_{dsat} = 0.63V$.

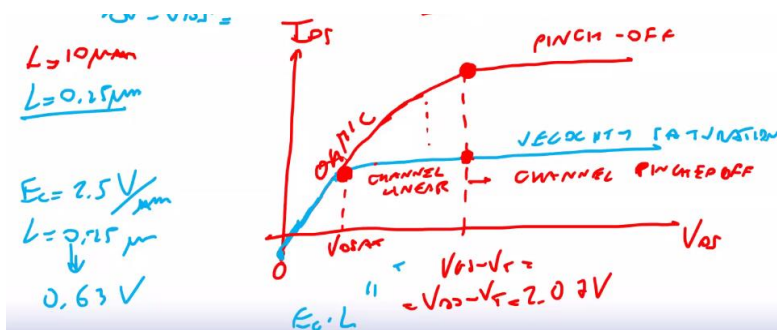
The V_{dsat} is the voltage that, applied to the drain, creates an electrical field so large, equal to the critical electrical field E_c (typically $2V/\mu m$) to saturate the velocity of the carriers. This explains the behaviour of the current that doesn't depend on the V_{ds} voltage anymore. More or less, the electrical field in the channel is V_{ds}/L .

Difference between long channel device and short channel device

In the linear or ohmic region I have a channel both on the source and drain (top left image). In the saturation region the channel is present only at the source side, I have the pinch-off at the drain. In this latter case, $V_{ds} > V_{ov}$.



Now let's plot I_{ds} vs V_{ds} for a long channel device ($L = 10\mu m$) and for the short one ($L = 0.25\mu m$). In the long channel device I have the square ohm law.



For the short channel, the E_c is more or less $2.5\text{V}/\mu\text{m}$, and if $L = 0.25\mu\text{m}$, $V = 0.63\text{V}$. As soon as we apply this voltage across the channel (at the drain), the velocity of the carrier saturates and there is no more dependence of V_{ds} .

If we consider the blue curve, which is the behaviour of the channel? At the beginning it's linear, we are in the ohmic region, and the current is the one of an ohmic transistor. Then, immediately after V_{dsat} , the channel is again linear. The channel becomes pinched off at the same pinch off point of the long channel transistor.

In the case of the $10\mu\text{m}$ there is no velocity saturation because the E_c is the same, but the L is larger, so the voltage at which we reach V_{dsat} is 25V , which is not available, we cannot create it.

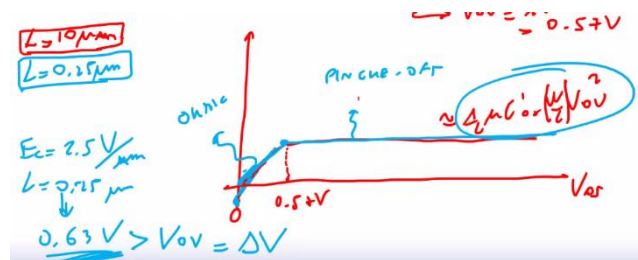
Let's now change the parameter V_{gs} . Everything else is kept the same, and $V_{gs} = 1\text{V}$ (small V_{gs}). The overdrive voltage in this case, being the threshold set to $V_t = 0.43\text{V}$, it is $1\text{V} - 0.43\text{V} = 0.57\text{V}$.

In this case there is not a difference between the $10\mu\text{m}$ case and the $0.25\mu\text{m}$ case.

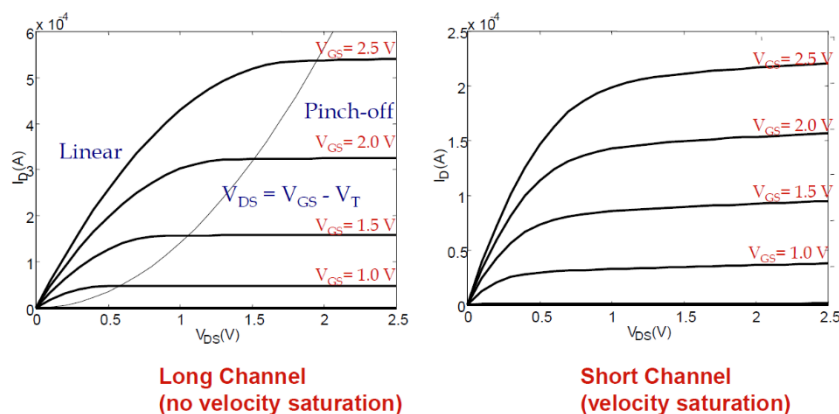
At 0.57V we enter the pinch off region for the $10\mu\text{m}$.

From this point on, the voltage across the channel, considering a very rough approximation, doesn't change. But 0.57V is smaller than 0.63V , so it cannot create a saturation of the carriers, so I'm not entering the velocity saturation, because $V_{dsat} > V_{ov}$.

So for small V_{ov} voltages, the velocity saturation doesn't occur. It happens on small length and large voltages V_{ds} .



Characteristic curves



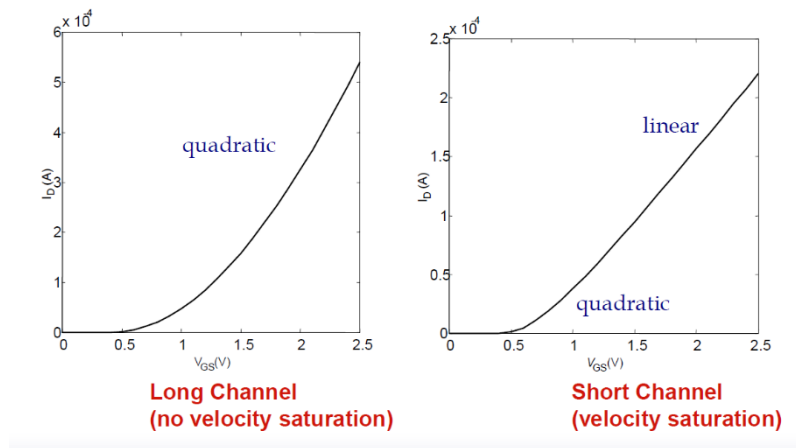
The W/L is the same for the two transistors, and these are the results of the simulations. In the short channel case, the characteristics are different because the scales are different. For $V_{gs} = 1\text{V}$ the curve is the same for the long channel, but then it changes for higher values of V_{gs} .

Three remarks:

1. Once we consider the velocity saturation, the available current is much smaller. So once they behave as current generators, transistors affected by velocity saturation can draw smaller current

- less available current. With smaller current the propagation delay of the inverter increases because it takes more time to charge the parasitic capacitance
- 2. In the velocity saturation case it seems that the transistor behaves like a current generator for a larger region, the ohmic region for the current (not channel) is much smaller than in the case of the long channel → the velocity saturation transistor works for most of the V_{ds} voltages in the current generator state, so current flat with respect to V_{ds} .
- 3. The Early voltage is proportional to the length; in our model is the lambda parameter that is more or less 0 in the long channel. In the short channel the slope is higher, so the channel length modulation effect takes place even if we are saturating the velocity of the carriers.

Trans-characteristic I_{ds} vs V_{gs}



It is current vs V_{gs} for $V_{ds} = V_{dd} = 2.5$ V. On the left we have the classical square law model, where the current is nihil up to the point $V_{gs} = V_t = 0.43$ V. Then the current is the pinch off current; if on, the channel is always pinched off because $V_{ds} < V_{dsat}$.

For the short channel I have still no current below threshold, and then two regions. One is the velocity saturation, but for small V_{gs} we have the previously described condition, so we cannot create an electrical field in the channel to have velocity saturation, so we have the square law. In both regions, however, the channel is always pinched off. More or less for $V_{gs} = 1.06$ V we have the boundary between velocity saturation and not.

SHORT CHANNEL MOSFET OPERATIVE REGIONS

1. The I_{ds} current is approximately 0, there is still a leakage current however. In the other three regions the device is on, because $V_{gs} > V_t$.
2. $V_{ds} < V_{gs} - V_t$. This means that the channel is ohmic (or linear). If we suppose that the $V_{ds} < V_{dsat} = 0.63$ V, we are not saturating the carriers, so the current is ohmic.
3. $V_{gs} - V_t < V_{ds}$ means that the channel is pinched off. Moreover, if $V_{gs} - V_t = V_{ov} < V_{dsat}$, so there is still no saturation of the carriers, so the current in this case is pinched off.
4. The channel can be ohmic or pinched off, I don't care. What counts for the current is that V_{dsat} is smaller than both V_{ov} and V_{ds} , so we have velocity saturation.

1) CUT OFF

$$V_{GS} - V_T \leq 0$$

2) LINEAR or TRIODE

$$V_{GS} - V_T > 0$$

$$\left. \begin{array}{l} V_{DS} < V_{GS} - V_T \\ V_{DS} < V_{DSAT} \end{array} \right\} V_{DS} \text{ is the lowest}$$

3) PINCH-OFF SATURATION

$$V_{GS} - V_T > 0$$

$$\left. \begin{array}{l} V_{GS} - V_T < V_{DS} \\ V_{GS} - V_T < V_{DSAT} \end{array} \right\} V_{OV} \text{ is the lowest}$$

4) VELOCITY SATURATION

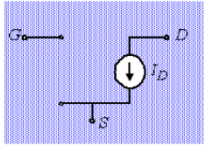
$$V_{GS} - V_T > 0$$

$$\left. \begin{array}{l} V_{DSAT} < V_{DS} \\ V_{DSAT} < V_{GS} - V_T \end{array} \right\} V_{DSAT} \text{ is the lowest}$$

UNIFIED MODEL FOR MANUAL ANALYSIS

$$V_{GS} < V_T \Rightarrow I_{DS} \cong 0$$

otherwise:



$$I_{DS} = \mu C'_{OX} \left(\frac{W}{L} \right) V_{min} \left(V_{GS} - V_T - \frac{V_{min}}{2} \right) (1 + \lambda V_{DS})$$

with:

$$V_{min} = \min(V_{GS} - V_T, V_{DS}, V_{DSAT})$$

If $V_{GS} - V_T > 0$ the transistor is on.

To explain the concept of V_{min} , we have to resort to the previous image. The operating region of the transistor in terms of current depends on which voltage is the smallest among V_{ds} and V_{dsat} . The smallest establishes the operating region in terms of current.

Let's suppose, for instance, that $V_{GS} = 1V$, so that $V_{ov} = 0.57V$. $V_{ds} = 2.5V$ and V_{dsat} is a number, $0.63V$. The current will be the one of the pinched off transistor with $V_{min} = V_{ov}$ (if we neglect the channel modulation effect in the last paranthesis).

$$I_{DS} = \mu C'_{OX} \left(\frac{W}{L} \right) V_{ov} \left(V_{ov} - \frac{V_{ov}}{2} \right) = \frac{1}{2} \mu C'_{OX} \left(\frac{W}{L} \right) V_{ov}^2$$

If instead $V_{ds} = 0.1V$, the current has the expression in the ohmic region.

$$I_{DS} = \mu C'_{OX} \left(\frac{W}{L} \right) \left[(V_{GS} - V_T) V_{ds} - \frac{V_{ds}^2}{2} \right]$$

Now V_{dsat} is the smallest. We get the following.

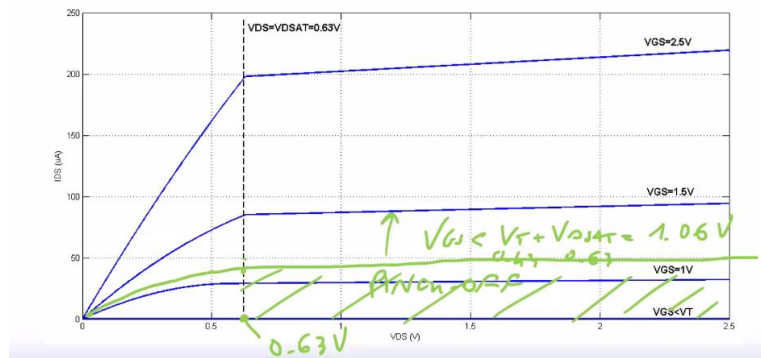
$$I_{DS} = \mu C'_{OX} \left(\frac{W}{L} \right) V_{DSAT} \left(V_{GS} - V_T - \frac{V_{DSAT}}{2} \right)$$

Simulation of short channel MOSFET

For small V_{ov} , that is $V_{GS} < V_T + V_{dsat} = 1.06V$, we have the classical square law model and for the current we are in pinch off. Above it we have the velocity saturation.

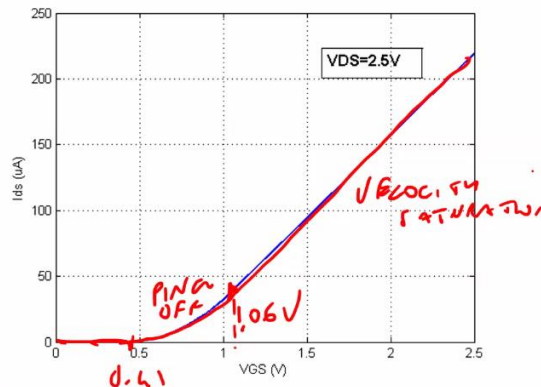
	V_{T0} (V)	γ ($V^{-0.5}$)	V_{DSAT} (V)	K' (A/V^2)	λ (V^{-1})
NMOS	0.43	0.4	0.63	115×10^{-6}	0.06

$$(W/L) = 0.375 / 0.25 \quad V_{DD} = 2.5V$$



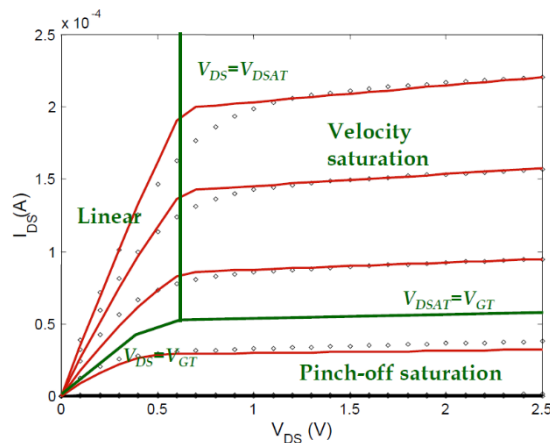
So when V_{dsat} is the smallest voltage among the three, we are in velocity saturation region, and it is a very large area.

	V_{TH} (V)	γ ($V^{0.5}$)	V_{DSAT} (V)	k' (A/V^2)	λ (V^{-1})
NMOS	0.43	0.4	0.63	115×10^{-6}	0.06



In the upper table we have the parameters that will be used during lecture.

Model – simulation comparison



Red curves are the simulations, which are very similar to the measurements. The dots are the results of the model. The model is very accurate in pinch off saturation and in velocity saturation region. The discrepancy is at the boundary between linear and velocity saturation for large V_{GS} .

I don't care about this discrepancy because in static conditions (DC) the transistor is either off or ohmic with current equal to 0. The problem is during the transient. If we consider the pull down transient, performed by the nMOS transistor, from the propagation delay stand point, the range I have to consider is not from V_{DD} to ground, but it is smaller, from V_{DD} to $V_{DD}/2$ in terms of V_{DS} (definition of the propagation delay, half the supply range). In this range there is no discrepancy, the dots are overlapped with the red characteristics and I'm ok.

For the pull-up, also this is the region of interest (V_{DD} to $V_{DD}/2$ in terms of absolute value of V_{DS}), so we are good.

Reference 0.25 um CMOS technology

Table 3.2 Parameters for manual model of generic 0.25 μm CMOS process (minimum length device).

	V_{T0} (V)	γ (V ^{0.5})	V_{DSAT} (V)	k' (A/V ²)	λ (V ⁻¹)
NMOS	0.43	0.4	0.63	115×10^{-6}	0.06
PMOS	-0.4	-0.4	-1	-30×10^{-6}	-0.1

For the PMOS transistors, your textbook consider all the voltages and the current negative.

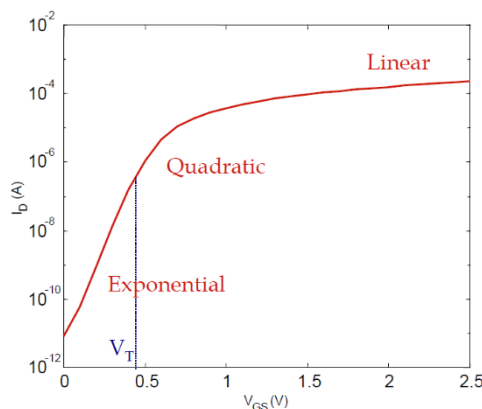
We work with positive quantities considering the absolute value.

Remember that a PMOS is on if V_G is smaller than V_S by $|V_T|$, i.e., $|V_{GS}| - |V_T| > 0$

We should consider the absolute value in the tables and keep the same functional form.

SUBTHRESHOLD CONDUCTION

We are in log scale and under threshold. For $V_{GS} < V_T$, the current is much smaller than in the linear region, pA. We know that for $V_{GS} < V_T$ we have an exponential dependency of the current according to x.



$$I_D = I_0 e^{\frac{qV_{GS}}{nkT}} \left(1 - e^{-\frac{qV_{DS}}{kT}} \right) \quad \times$$

n = slope factor

$$n = 1 + \frac{C_D}{C_{ox}}$$

S is ΔV_{GS} for $I_{D2}/I_{D1} = 10$

$$S = n \left(\frac{kT}{q} \right) \ln(10)$$

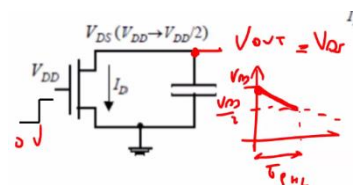
Typical values for S:
60 .. 100 mV/decade

$n = 1.5$ in general. The only thing that counts is that, once we switch off a transistor, the current is not properly nihil, it is in the order of pA. With billion of inverters, the overall current cannot be considered negligible. **The current is exactly 0 only if $V_{ds} = 0$.**

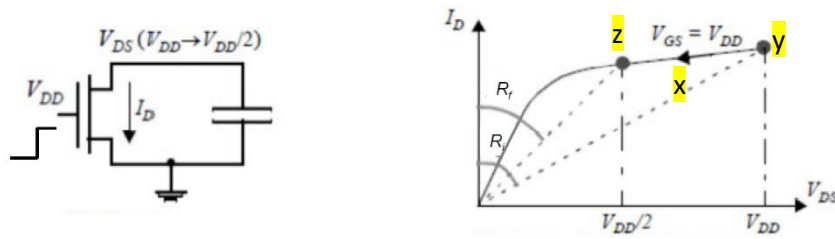
EQUIVALENT RESISTANCE

I want to assess the propagation delay as $\tau_p = \ln(2) \cdot R_{eq} \cdot C$. So the transistor must be modelled with resistances.

In the image I apply a low to high transition to an inverter, so the pMOS is off (not represented in the image) and the nMOS is on. The voltage was V_{DD} at the beginning, and now I want the high to low propagation delay. V_{out} goes from V_{DD} to 0 at the end of the day, but we care, in digital electronic, from V_{DD} to $V_{DD}/2$.



So the V_{GS} of the transistor is fixed to V_{DD} , and for large V_{GS} we have the curve on the right of the next image.



$$R_{EQ} = \frac{R_i + R_f}{2} = \frac{1}{2} \left(\frac{V_{DD}}{I_{DSi}} + \frac{V_{DD}/2}{I_{DSf}} \right) = \frac{1}{2} \left[\frac{V_{DD}}{I_{DSAT} (1 + \lambda V_{DD})} + \frac{V_{DD}/2}{I_{DSAT} (1 + \lambda V_{DD}/2)} \right] = \frac{3}{4} \frac{V_{DD}}{I_{DSAT}} \left(1 - \frac{5}{6} \lambda V_{DD} \right)$$

$$I_{DSAT} = \mu C_{ox} \left(\frac{W}{L} \right) V_{DSAT} \left[V_{DD} - V_T - \frac{V_{DSAT}}{2} \right]$$

I'm interested in the region x from $V_{DD}/2$ to V_{DD} . This is the current drawn by the transistor once I discharge the capacitance.

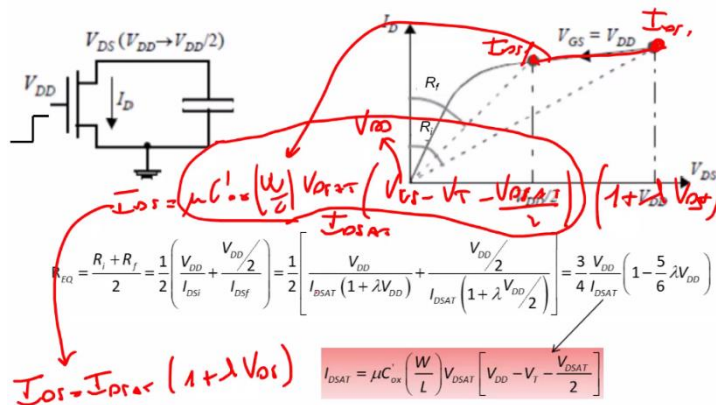
I want to express the behaviour of the transistor along the transient considering a resistor. Current y is the initial current of the transient, I_{DSi} . The voltage in y is V_{DD} , so the ratio is R_i , the initial resistance. Then I move in the final point z; I will have a final resistance R_f .

Then I compute the equivalent resistance as the arithmetic average between R_i and R_f : **$R_{EQ} = (R_f + R_i)/2$** .

In reality, I should have computed the equivalent resistance with an integral.

$$R_{eq} = \frac{1}{\Delta V} \int_{V_m}^{V_m} \frac{V_{DS}}{I_{DS}} dV$$

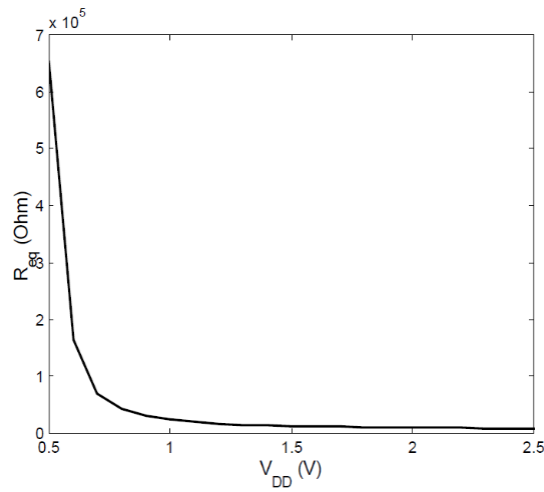
The expression of the current I have to consider is the one with the modulation voltage. The first part is constant, I_{DSAT} , the thing that changes is the part with the channel modulation effect.



The formula must not be known, the important concept is the averaging of the resistance. Moreover, the equivalent resistance depends on the PS voltage and I_{DSAT} . And I_{DSAT} depends on the aspect ratio, so **to decrease the equivalent resistance of a transistor we can change the aspect ratio increasing W** .

To be honest, R_{eq} depends at the denominator to $V_{DD} - V_T - V_{DSAT}/2$ (in the I_{DSAT} dependance). The term $V_T + V_{DSAT}/2$ is called **effective threshold V_{TE}** . In our technology, $V_{TE} = 0.43 + 0.63/2 = 0.7V$. So why not changing V_{DD} ? If I decrease it, at a certain point I have a steep increase of R_{eq} as in the next image. This happens because the denominator goes to 0.

Similarly, applying a large Vdd is not useful, even if the Req decreases, because we create breakdown in the transistor and we destroy the transistor. In fact, every technology has a maximum Vdd above which we cannot go. If we decrease L of the transistor, the Vdd allowed decreases.



Hence to decrease the resistance we can act only on one parameter, the W.

Results

Table 3.3 Equivalent resistance R_{eq} ($W/L = 1$) of NMOS and PMOS transistors in 0.25 μm CMOS process (with $L = L_{min}$). For larger devices, divide R_{eq} by W/L .

V_{DD} (V)	1	1.5	2	2.5
NMOS (k Ω)	35	19	15	13
PMOS (k Ω)	115	55	38	31



nominal supply voltage

nMOS and pMOS transistors have an equivalent resistance of 13k and 31k respectively, for $W/L = 1$. The difference arises because we have the term $un^*C'_{ox}$, the process transconductance. In fact, the one of the nMOS is more or less three times larger than the one of the pMOS.

ASSESSING THE PROPAGATION DELAY

The transistor in the image is not feasible, because the aspect ratio is one, it can be done only on paper. Typically, the minimum width we can design is $W_{min} = 1.5 \cdot L_{min}$ at least.

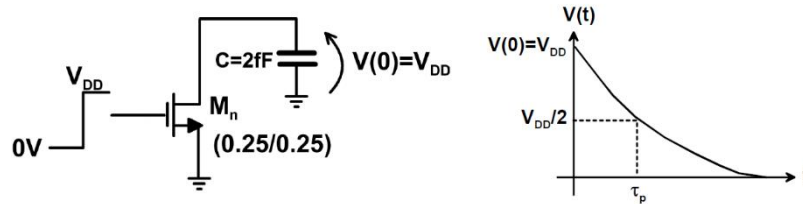
For this transistor we have a resistance of 13k and I want to assess the propagation delay, assuming it is the pull down transistor of an inverter. The $C = 2\text{fF}$ (capacitance associated to the minimum size inverter).

The voltage at the gate experiences a low to high transition, the transistor turns on and I have a discharge current. The propagation delay of this circuit is computed on the first part of the transient, from V_{dd} to $V_{dd}/2$, and it is an RC transient. The τ_p is the one in the image.

Is this the only way to assess the delay?

No, we can do this because it is a very simple circuit and we know that the transistor is working in velocity saturation from V_{dd} to $V_{dd}/2$. Thus $I_{ds} = I_{dsat}(1 + \lambda V_{ds})$.

Then I have the capacitor, whose relationship is well known (image). Then once I have this expression I can assess the propagation delay performing the integral. With this exact analysis we get a similar result with respect to the model.



$$\tau_p = \ln(2) R_{eq} C_L = 0.69 \cdot 13k\Omega \cdot 2fF = 18ps$$

Carrying out the exact analysis, i.e., $I_{DS} = -C \frac{dV_{DS}(t)}{dt}$

$$\tau_p = -C \int_{2.5V}^{1.25V} \frac{1}{I_{DSAT} (1 + \lambda V_{DS})} dV_{DS} = \frac{C}{\lambda I_{DSAT}} \ln \left(\frac{1 + \lambda V_{DD}}{1 + 0.5 \lambda V_{DD}} \right) = 17.7ps$$

MOS CAPACITANCES DYNAMIC BEHAVIOUR

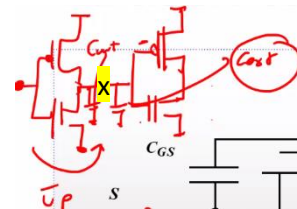
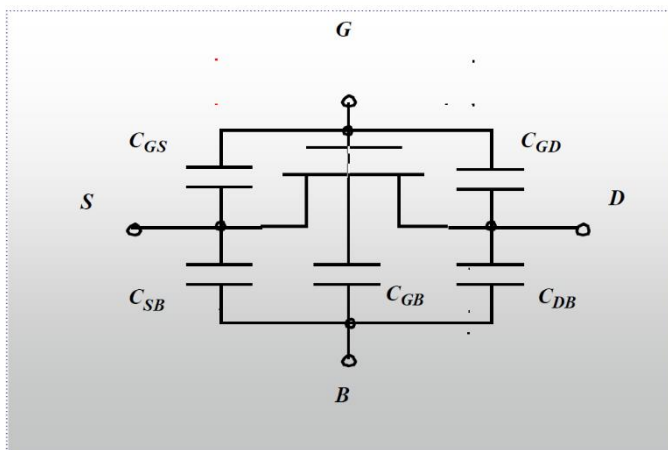
The capacitances that come into play in a transistors are:

1. Geometrical capacitances, a.k.a. overlap capacitances
2. Channel capacitances, a.k.a. intrinsic capacitances
3. Junction capacitances

For the 2, the intrinsic capacitances is not a correct term, it refers to another thing, because the intrinsic capacitance is the parasitic capacitance of a digital gate.

3 occurs because to isolate the transistor we have to reverse bias the diodes. Since we have diodes to reverse bias, a diode in a reverse bias region means a depletion capacitance.

TRANSISTOR MODEL WITH PARASITICS

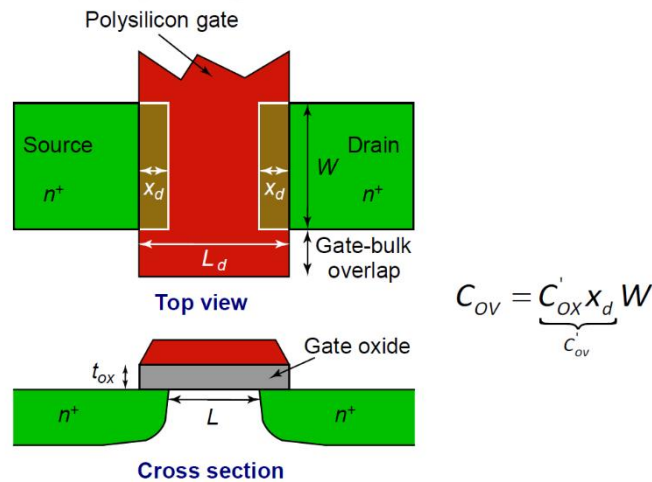


Bulk is ground for nMOS and Vdd for pMOS. We want to estimate the capacitance at the input and at the output of the inverter. In fact, the inverter may drive another gate, e.g. another inverter. If I'm interested in assessing the propagation delay of the inverter, I know that I can model the transistor as a resistor, but I need to assess also the parasitics contributions, Cint and Cext.

During the transient, node x corresponds to the drain of the transistor (either for the pull up or the pull down), so at the output we have the drain of the transistors during transient (during transient the transistors are current generators, so one terminal behaves as a source and the other as a drain).

Cdb and Csb are the parasitic capacitances related to the reverse bias diodes.

Overlap capacitances (always fixed, due to production process)



I'm considering an nMOS transistor. The transistor is implemented with the self-gate aligned technology (Federico Faggin). Firstly we implement the gate and then the source and drain regions.

We have overlap capacitances with the dielectric SiO2 in the middle. It is estimated as the specific capacitance per unit area ($C'_{ox} = \epsilon_{ox}/t_{ox} = \epsilon_r \cdot \epsilon_0/t_{ox}$), a well-known number depending on the technology. $\epsilon_{ox} = 1/3 \cdot \epsilon_{Si}$ with $\epsilon_{Si} = 1 \text{ pF/cm}$. In the end, $\epsilon_{ox} = 1/3 \text{ pF/cm} = 33 \text{ aF/um}$, leading to a $C'_{ox} = 6 \text{ fF/um}^2$.

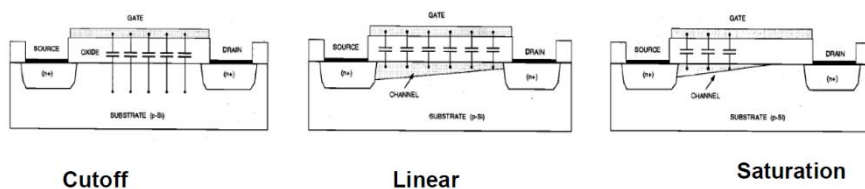
Then this C'_{ox} is multiplied by the overlap region, which depends on the technology.

Typically, $x_d = 1/5 \cdot L_{min} = 0.05 \text{ um}$ in our technology.

The other dimension to determine the overlap area is the width of the transistor, the value the designer can change → if we increase W to decrease the resistance, we are increasing the parasitic capacitance.

NB: in pinch off region there is no channel contribution and the only contribution that survives is the overlap contribution.

Channel capacitances



Parameter	Cutoff	Linear	Saturation
C_{gb}	C_{ox}/C_{dep}	0	0
C_{gs}	0	$0.5C_{ox}$	$0.67C_{ox}$
C_{gd}	0	$0.5C_{ox}$	0
$C_g = C_{gs} + C_{gd} + C_{gb}$	C_{ox}/C_{dep}	C_{ox}	$0.67C_{ox}$

We have a gate and the channel below it that can be: cut-off (no channel), linear, pinch off saturation.

Cut off region (// means in series)

In the cut off region, since there is no channel it doesn't mean we have an open circuit, because in the region under the gate I have a depletion layer, so fixed negative charge, because the holes are attracted by the p+ substrate contact, so negative fixed ions remain (with fixed charges I cannot have a current). Hence I have the series of two capacitances, the oxide capacitance and the depletion layer capacitance, which is between the gate and bulk.

The other contributions C_{gs} and C_{gd} are zero in the cut off region. In reality we still have the overlap contribution that generates a capacitance, but we are speaking about channel contributions, so no contribution from C_{gs} and C_{gd} .

Linear region

Gate is well above threshold but $V_{ds} = 0$, for instance.

In this case the channel is inverted, I have an inversion layer of free electrons. The overall oxide capacitance can be split in two contributions, one toward the source and the other toward the drain. $C_{ox} = C'_{ox} * W * L$ (L is the effective length, excluding the overlap regions).

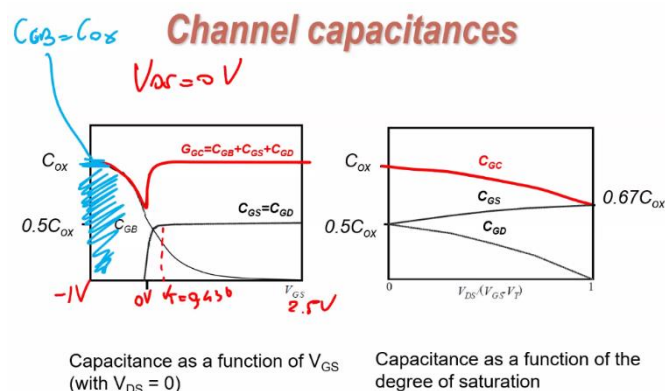
The two contributions are equal because the channel is formed in mostly equal measure on both sides.

Pinchoff saturation

The overall capacitance is toward the source, but between gate and source there is only a fraction of the oxide capacitance, 2/3 of it. No contribution to the drain and to the bulk.

The overall capacitances can be visualized in the following image.

On the left we have all the contributions in the case $V_{ds} = 0V$, spanning the value of V_{gs} .



We have a contribution from gate to bulk only in the left part; in this part I'm applying a negative voltage to the gate, so I'm attracting free holes and an accumulation region.

Where we have the valley we are in the cut off region. After this region there is no more gate to bulk capacitance contribution.

The red line is the overall capacitance I see from the gate. Apart from the pinch off region, I notice that C_g is almost $C_{ox} = C'_{ox} * W * L$ ($C'_{ox} = 6 \text{ fF}/\mu\text{m}^2$).

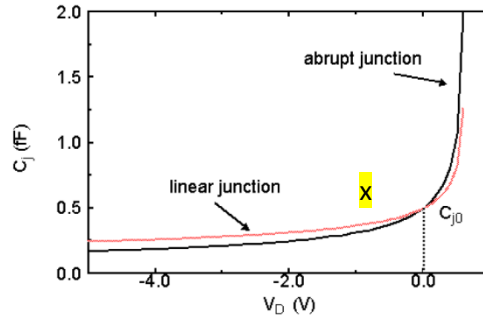
In the right plot, instead, I'm considering $V_{gs} - V_t = V_{dd}$ and I'm spanning V_{ds} . **The ratio between V_{ds} and $V_{gs} - V_t$ is also called ratio of saturation.** If equal to 0 we are in ohmic, if 1 we are in pinchoff saturation, so from left to right we move from ohmic to pinchoff saturation.

In ohmic condition the overall oxide capacitance is split in two equal contributions. Then one of the two increases, and the other one goes to 0. The overall capacitance we see from the gate is the red one. It is not properly constant but almost $2/3 * C_{ox}$.

So whatever the operating region, from the gate we see almost the gate capacitance C_{ox} .

Recap – Behaviour of a diode

A pn junction has a capacitance that depends on the applied voltage according to the formula. Firstly, it is proportional to the area of contact of the junction. Then on the specific parameter per unit area. The built in potential ϕ_0 is 0.9V, and $m = 0.5$.



$$C_j = \frac{A \cdot C'_{j,0}}{\left(1 - \frac{V_D}{\phi_0}\right)^m}$$

A = diode area
 $C'_{j,0}$ = specific capacitance per unit area ($V=0$)
 V_D = direct voltage across the diode
 ϕ_0 = built-in potential

26

This capacitance depends mostly on the depletion region we create at the boundary and inversely related to the direct voltage we apply.

X is the region of reverse bias, where it is biased the diode of the transistor.

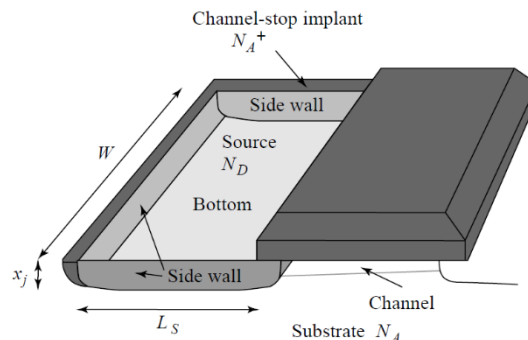
Diffusion capacitance

Let's consider the diffusion region corresponding either source or the drain, it is the same, because the device is symmetric.

The depth x_j is 0.15 μ m, quite similar to the length of the channel, non negligible.

The width is the most important parameter we can change. The substrate is p doped, then we have a n+ region and the diode is a 3d diode. So for sure we have a bottom area contribution, but then we have also the sidewalls contribution for each of the 3 sides, but not on the one on the channel side because we have a 'shortcircuit' because the channel exists.

Two contributions: bottom-area and side-wall capacitances



$$C_{diff} = C'_j L_s W + \underbrace{C'_j x_j}_{C'_{j,sw}} (2L_s + W)$$

The first term is the bottom area contribution. C'_j is the specific capacitance per unit area, and the area of the bottom side is $W \cdot L_s$, with L_s that is fixed by technology and it is 0.625 μ m.

Then we have the sidewall contributions; $x_j = 0.12 \text{ } \mu\text{m}$.

EQUIVALENT CAPACITANCE ALONG A TRANSISTOR

The capacitances seen are voltage dependent, so if we experience a transient, the capacitances vary along the transient. Hence we have to consider that the capacitance is not constant.

To address the variation of the capacitance we could use the first approach of the image. A capacitance is a variation of charge wrt a variation of voltage. The variation of charge, divided by the voltage variation, means that I have to perform an integral. But this is a very complex approach, because we have to solve an integral.

- Replace non-linear capacitance by large-signal equivalent linear capacitance which displaces equal charge over voltage swing of interest

$$C_{EQ} = \frac{\Delta Q}{\Delta V} = \frac{Q(V_2) - Q(V_1)}{V_2 - V_1} = \frac{1}{V_2 - V_1} \int_{V_1}^{V_2} C(V) dV$$

↓ simplified approach

$$C_{EQ} \cong \frac{C_2 + C_1}{2}$$

- To further simplify the analysis, let's consider constant capacitances, i.e., not dependent on voltage, and equal to their maximum value

The other approach is a simplified one. The capacitance varies along the transient from a value C1 to a value C2, and I don't know how it varies. So I do the average. It works if the change is not so abrupt.

The last possibility is to consider the maximum value of the capacitance if we have it varying during the transient. In fact, the worst case approximation is always ok. This simplifies a lot the analysis.

Capacitive device model

Let's consider the maximum values. Gate-channel means that, from the gate, in terms of channel capacitance I see three regions, source, bulk and drain. I can say that the maximum capacitance is the oxide capacitance.

Then the gate overlap capacitance is a fixed geometric capacitance. $C_{ov} = C'_{ox} \cdot x_d = 0.3 \text{ fF}/\mu\text{m}$.

Then we have the junction capacitance, whose maximum value is for an applied voltage equal to 0V (if I look at the C_j formula of the diode).

- Gate-channel capacitance

- $C_{GC} \cong C'_{ox} WL$

- Gate-overlap capacitance

- $C_{GSov} = C_{GDov} = C'_{ov} W$

- Junction capacitance

- $C_{diff} = C'_J L_s W + C'_{J,SW} (2L_s + W)$

At a first order approximation, all the capacitances are proportional to W

The only parameter we can change is W.

Gate and drain capacitances

- Gate and drain capacitance of a transistor are proportional on the width of the device, at a first order approximation

$$C_g \cong C_{ox} + 2C_{ov} = \frac{\epsilon_{ox}}{t_{ox}}(L + 2x_d)W$$

$$C_d \cong C_{ov} + C'_j L_s W + C'_{j,sw} (2L_s + W)$$

- The specific capacitance per unit width are:

$$C'_g \cong \frac{\epsilon_{ox}}{t_{ox}}(L + 2x_d)$$

$$C'_d \cong \frac{\epsilon_{ox}}{t_{ox}}x_d + C'_j L_s + C'_{j,sw}$$

Let's try to assess more quantitatively the gate and drain capacitances. In a nMOS transistor, from the gate we see the overall oxide capacitance, which is the channel capacitance and then the two overlap contributions, one toward the drain and the other one toward the source.

From the drain we see mostly the overlap contribution between gate and drain and, during a transient, e.g. from Vdd to Vdd/2, we see the overlap contribution because the channel is pinchoff. So either the transistor is off or in pinchoff, from the drain I see the overlap contribution. Then I have also the diode contribution, split in bottom area and sidewall contributions.

Let's try to divide the gate and drain contributions by the width W, so assessing them per unit width.

If we change the technology, these expressions were evaluated for 0.25um technology. Still staying in the planar technology, let's look at C'g and C'd.

If I move from a 0.25um to 0.18um, also all the other dimensions must decrease, so in the end C'g remains constant, because numerator and denominator are scaled equally.

As for C'd, the first term remains the same; the second one has C'j increasing and Ls decreasing, and also C'j,sw remains constant → also C'd remains constant.

Parameter

The first column is C'ox, the second one C'o. In the end, C'g = C'd = 2 fF/um and they are almost constant for most of the technology → **C'g and C'd technology independent at first approximation**. So we see the same **specific** capacitance at the gate and drain (not the same capacitance).

Capacitances in 0.25-μm CMOS process

	C_{ox} (fF/μm ²)	C_o (fF/μm)	C_j (fF/μm ²)	m_j	ϕ_b (V)	$C_{j,sw}$ (fF/μm)	$m_{j,sw}$	$\phi_{b,sw}$ (V)
NMOS	6	0.31	2	0.5	0.9	0.28	0.44	0.9
PMOS	6	0.27	1.9	0.48	0.9	0.22	0.32	0.9

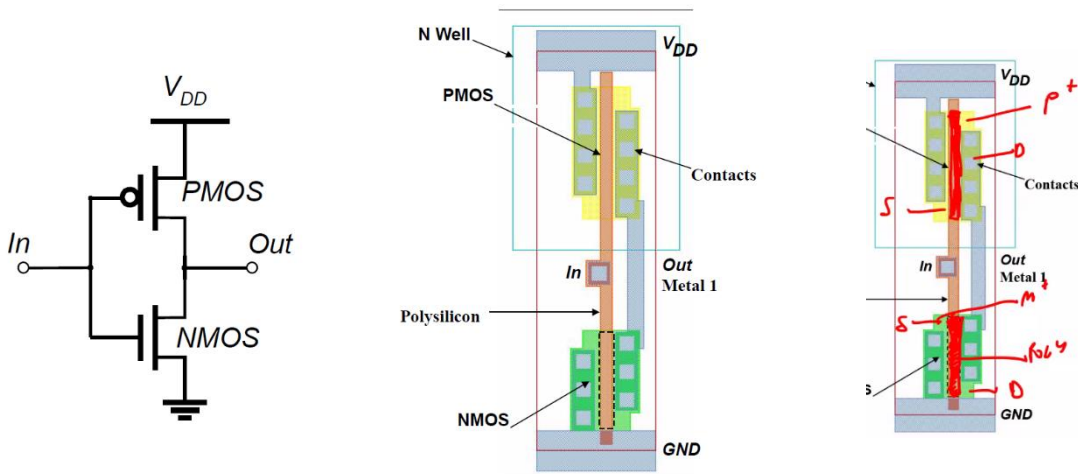
- In the 0.25-μm process: $C'_g \cong C'_d \cong 2 \text{ fF}/\mu\text{m}$ (of width, W)
- C'_g and C'_d are almost technology independent (for example, L, x_d, t_{ox} scale with the technology)
- However, smaller dimensions allowed by scaled technology lead to smaller capacitances!

So what's the benefit of the scaling?

It is that we can reduce the width and the length and the capacitance decreases (not the specific capacitances). In terms of resistance, it is the same (depends on the aspect ratio, and the minimum aspect ratio is 1.5). So **the only parameter that changes with technology scaling is the capacitance** → smaller capacitance means faster transistors.

FC – CMOS INVERTER

Fully complementary CMOS inverter. I want to implement a circuit that is able to invert my logical signal at transistor level.



The simplest thing to do is to implement two complementary switches that work with two different voltages.

On the right we have the layout of the FC-CMOS inverter. The light green area is n+, the red is polysilicon gate of the nMOS transistor. Connections between drains is done with metal, while the grey squares are contacts.

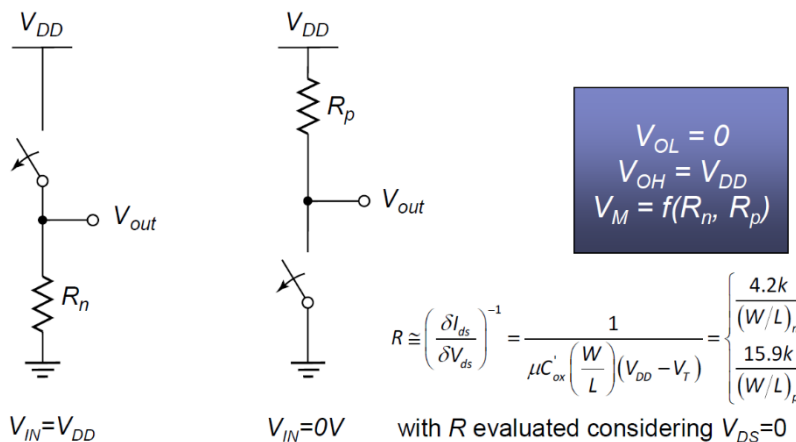
In this top view there is something missing, that are the substrate contacts and, since the pMOS is implemented in a n well region (light blue), I need a n+ contact to bias the n well region to Vdd.

FIRST ORDER DC ANALYSIS

Let's apply a static voltage at the input (steady state) and we measure the output after the transient is over.

In this case, if we apply a large enough voltage at the input, the nMOS is on and the pMOS is off. Applying Vdd at the input of the inverter, the output is 0V. The pMOS is in fact off, and the nMOS is in ohmic with Vds = 0 and Ids more or less 0A, because the pMOS is off.

Rn is a real resistance, not an ideal one.



Let's consider the opposite voltage at the input, so 0V. Same reasoning, Ids = 0A in the pMOS because the nMOS is off. Vds = 0V and so Vout = Vdd.

The resistances R_n and R_p have the value in the formula, computed in the ohmic region.

The image shows handwritten equations in red ink on a blue background. The first equation is $I_{DS} = \mu C'_{ox} \left(\frac{W}{L}\right) V_{DS} (V_{GS} - V_t - \frac{V_{DS}}{2})$. The second equation is $\frac{dI_{DS}}{dV_{GS}} = \mu C'_{ox} \left(\frac{W}{L}\right) [V_{GS} - V_t - \frac{V_{DS}}{2}]$. The text 'rcuit Design' is visible on the left and 'Invert' on the right.

The derivative must be assessed for $V_{DS} = 0$, and the derivative is nothing else than equal to $1/R_{on}$.

Is there a reason to reduce R_{on} ?

If we increase W , R_{on} decreases and we improve the reliability because the output of one inverter is the input of another one, so the better the connection to ground or to V_{DD} , the more improved the digital noise, so there is no digital noise able to change the level of the second inverter. So **by reducing R_{on} we reduce the transfer function of the digital noise.**

In term of reliabilities the things that matter are: noise margins, R_{on} , reduced wire parasitic coupling.

STATIC PROPERTIES

- FC-CMOS logic is ratioless: $V_{OH} = V_{DD}$, $V_{OL} = 0V$ whatever the sizing of the gate
- Switching threshold depends on the relative sizing of PMOS and NMOS transistor
- No static power consumption (assuming negligible leakage)
- Capacitive input (infinite input resistance)
- Static gate: low impedance path from output node to either VDD or GND
- Output impedance depending on the transistor aspect ratio

Ratioless means that we can sizing the pMOS and the nMOS as we want. What depends on the relative sizing is the switching threshold: $V_m = f(W_n, W_p)$, even if in reality it is a function of the aspect ratios.

Moreover, there is no static power consumption because in DC one of the switches is open and there is no current drawn from the power supply generator at first approximation.

The FC device belongs to the FC logic family, together e.g. with the pass transistor logic. And all these logic belong to the static gate logics; this means that, whatever the state, there is always a low impedance (resistance) path to ground or V_{DD} at **steady state**. This doesn't happen in dynamic families, because there may be a state where the output is floating.

This means that the static gate is a very reliable technology.

VOLTAGE TRANSFER CHARACTERISTIC (VTC)

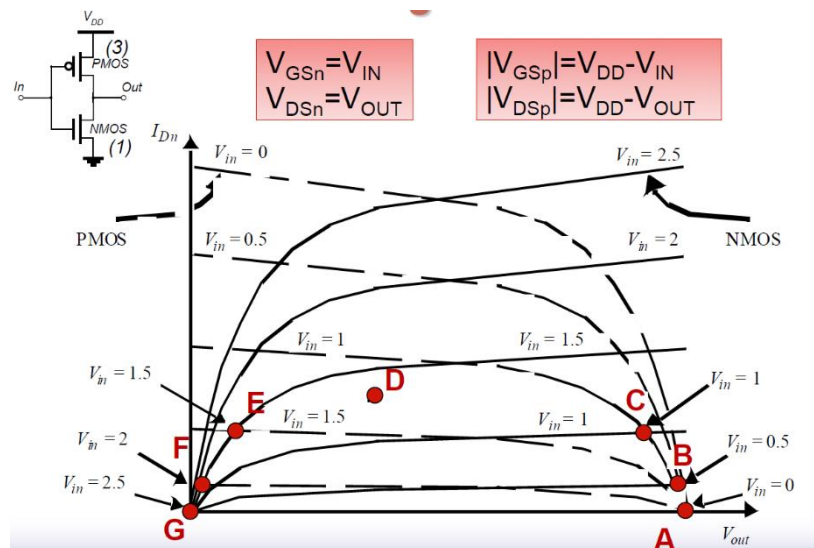
I want to derive the static VTC of the gate, meaning that I'm plotting the relationship between V_{out} and V_{in} at steady state, without any transient.

Surprisingly, the pMOS in the image has an aspect ratio of 3, and the nMOS of 1. It is expected because of reliability purposes, to have a switching threshold in the middle.

Symmetric devices means electrically equivalent, so that they have the same current for the same applied voltage. The switching threshold is defined so that if I apply $V_m = V_{dd}/2$ at the input I get $V_m = V_{dd}/2$ at the output.

To build the VTC I have to plot on the same graph the current of the two devices (I_{dn} and I_{dp}) as function of V_{out} for different values of V_{in} . The curves of the nMOS are the classical V_{ds} vs I_{ds} curves. For the pMOS, $V_{out} = V_{dd} - |V_{ds}|$. The current in the pMOS is 0 if $V_{out} = V_{dd}$.

The curve of the nMOS for $V_{in} = 2V$ and of the pMOS for $V_{in} = 0.5V$ are very similar because the V_{gs} is approximately the same.

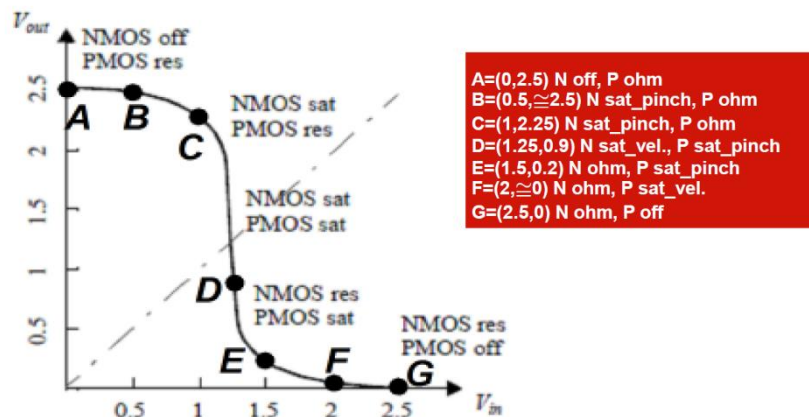


Since we want to evaluate the i/o characteristic, we know that the two transistor share the same current, because they are in series, in DC. So I take e.g. $V_{in} = 0V$ and I take the point where the two current characteristics are the same for the two transistors: point A, this is the working point.

We can repeat this procedure for other input voltages.

Point D corresponds to a characteristic that is not depicted in the image, and it is related to $V_{dd}/2 = 1.25V$. It is shifted leftwards, meaning that the threshold is not in the middle, it is lower. The point D corresponds to a $V_{out} = 0.9V$.

Now let's plot the points A, B, C ... on the characteristic. For a given V_{in} , we plot the point of the corresponding V_{out} .



The one in the image is the VTC at steady state. We notice a transition region in the middle and the two almost flat region at the boundaries → characteristic of an inverter.

The transition in the middle has a very big small-signal gain (very steep). In point A, B and C the input signal is small, the pMOS is well turned on because we have a large V_{gs} and the pMOS works in ohmic region, we have a shortcircuit to V_{dd} . In this region the pMOS is in ohmic region and the nMOS works in saturation region as a current generator, and since the resistance of the pMOS is small, the output is connected like to V_{dd} .

On the other side the opposite happens, so the nMOS is in ohmic (it can be represented with a real small resistance) and the pMOS is a current generator. The output is not properly ground, because some current flows in the nMOS, but since the resistance is small, the output is approximately 0V.

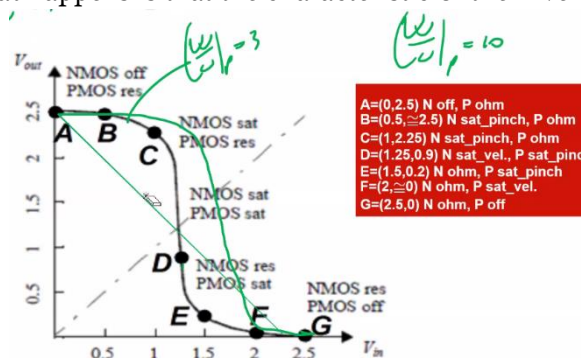
The particular region is the middle one where we have a steep transition, the output voltage drops to 0 for a small interval of the input voltage. This means that the small signal gain is very high; if we bias the inverter with $V_t = 1.22V$ (threshold voltage from the plot where the bisector intercept the inverter characteristic), the output should be 1.22V. If I then apply a small signal, I have an amplification, and the gain is the derivative, which is negative, so the output will be the input one amplified.

In order to have a large gain, the two transistor in the transition region must work as current generators and so in saturation region, not in ohmic. The current in the pMOS and nMOS transistor is the same.

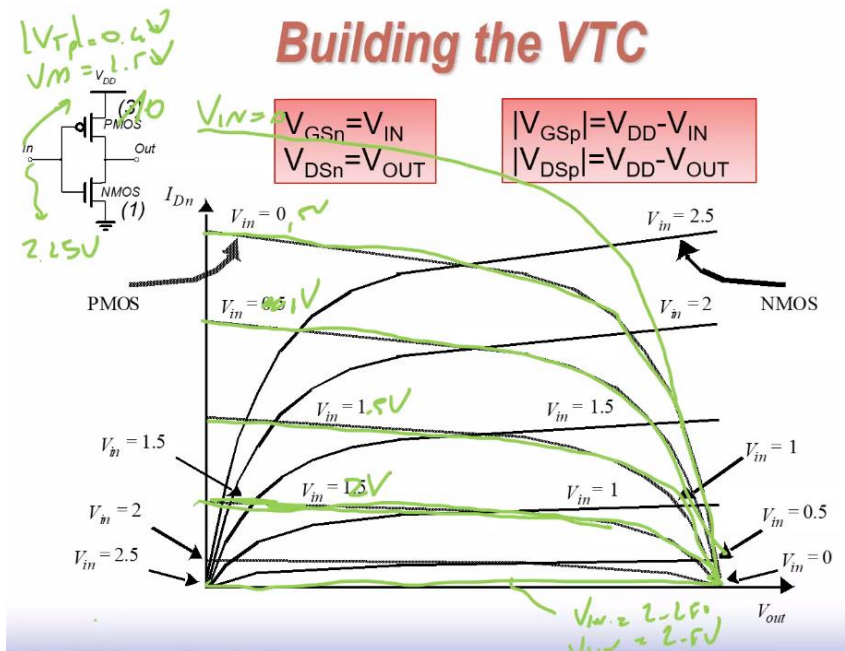
As a second remark, the sizing of the pMOS with an aspect ratio 3 times bigger than the nMOS has a rationale that is that the two devices must be symmetric, i.e. electrically equivalent, they must draw the same current for the same V_{ds} and V_{gs} , so that the two characteristics intersect exactly in the middle for a voltage in input equal to the threshold voltage. **In order to have the V_t in the middle, the two devices, working as current generators (I don't care if in pinchoff or velocity saturation), must have the same V_{gs} and V_{ds} , so they must be electrically equivalent.**

Since in our case we have that the threshold is not in the middle at 1.25V, so what we can do is to increase the aspect ratio of the pMOS from 3 to 3.5 so that the characteristic of the pMOS is shifted above and the intercept between the two characteristics is in the middle of the PS.

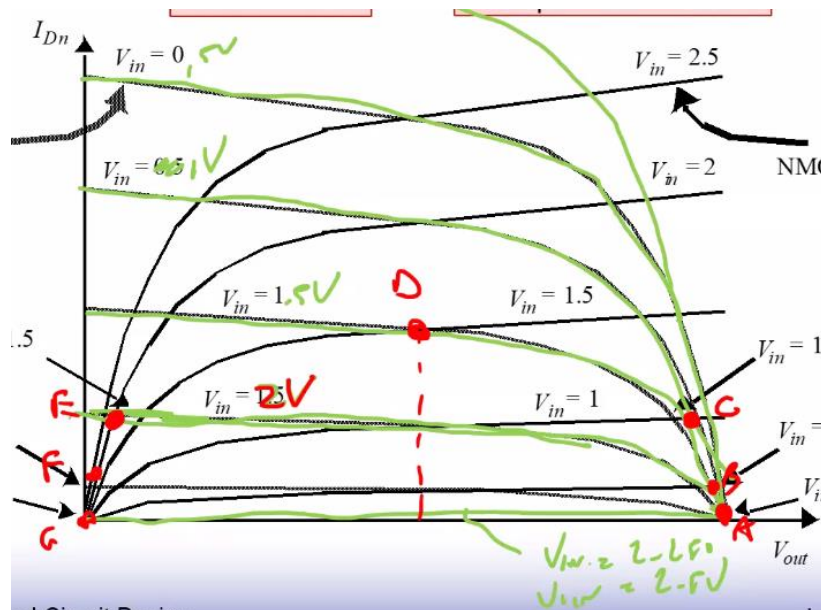
If we size $W/L|_p = 10$, what happens is that the characteristic of the inverter moves rightwards.



The nMOS transistor curves are not changing because the aspect ratio is the same; however, the pMOS is larger, so the current it carries for the same V_{gs} is bigger.



Then I have to assess the working points, seeing the intersection points for the same input voltage. In this case the threshold voltage is approximately 1.5V.



The threshold voltage is larger because I've increased the driving capabilities of the pMOS, and so the pMOS and nMOS behave as current generator for higher input voltages.

If e.g. the pMOS was sized with an aspect ratio of 1, the VTC is shifted towards the left, so the two transistor are electrically equivalent for a smaller input voltage. To have the same current, we will need $V_{gs,n} < V_{gs,p}$ to have them behaving as current generators.

Once we have two current generators, I_p must be equal to I_n for sure because they are in series. If we stack two transistors with two different currents one over the other, to have the condition $I_p = I_n$ satisfied, if e.g. $I_p > I_n$, the output voltage increases so that the pMOS is pushed in ohmic region and the balance stands. So I_p decreases and becomes equal to I_n (the weakest always wins).

This is why we need to size the two transistors to be electrically equivalent to have the threshold in the middle.

SWITCHING THRESHOLD

So the two transistors are working as current generators at the switching threshold. Let's assume that:

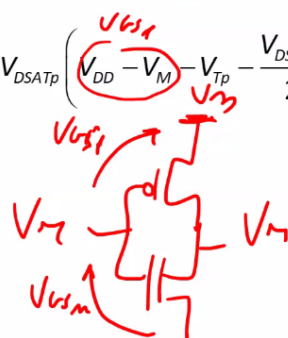
- They work in velocity saturation region.
- Neglect the channel modulation effect.

Assuming both transistors working in velocity saturation region:

$$x \quad k_n' \left(\frac{W}{L} \right)_n V_{DSATn} \left(V_M - V_{Tn} - \frac{V_{DSATn}}{2} \right) = k_p' \left(\frac{W}{L} \right)_p V_{DSATp} \left(V_{DD} - V_M - V_{Tp} - \frac{V_{DSATp}}{2} \right)$$

defining
$$r = \frac{k_p' \left(\frac{W}{L} \right)_p V_{DSATp}}{k_n' \left(\frac{W}{L} \right)_n V_{DSATn}},$$

it turns out:
$$V_M = \frac{\left(V_{Tn} + \frac{V_{DSATn}}{2} \right) + r \left(V_{DD} - V_{Tp} - \frac{V_{DSATp}}{2} \right)}{1 + r}$$



I'm considering the input voltage equal to Vm (threshold voltage) and so also the output will be. k'p is u_p*C'ox. We can also call beta the ratio between the aspect ratios.

Once r is defined, we have a linear equation that can be solved in the unknown Vm. This is derived directly from the fact that the two transistors are working as current generators, so I'm equating the two currents assuming they work in velocity saturation region (x).

To have Vm = 1.25V, r has to be a value that is r = 1.443.

But what counts is that the beta factor beta = (W/L)_p / (W/L)_n = 3.5.

A couple of remarks. The expression is used to assess the beta factor that assesses the Vm in the middle, but the two transistors are actually working in velocity saturation region?

We need to refine the analysis.

Let's take 3.5 as factor between the aspect ratios. The Vgs,n = 1.25V at threshold, so Vov = 0.82V (Vt,most = 0.43V). If so, the transistor is pinched off in the channel and the transistor is in velocity saturation because 0.82V > 0.63V which is the saturation voltage.

For the pMOS, Vov = 0.85V, but Vdsat,p = 1V. The channel is still pinched off but 0.85V is not enough to work in velocity saturation, because smaller than 1V.

So I should have written the saturation pinchoff equation for the pMOS.

$$\frac{1}{2} \mu_p C_{ox} \left(\frac{W}{L} \right)_p \left(V_{DD} - V_n - V_{Tp} \right)^2$$

However, the result is very similar than in the case of velocity saturation, so the approximation is ok.

Another remark. Let's suppose that Vdsat,p = Vdsat,n and Vtp = Vtn, but the process transconductance is different (k'). To have the switching threshold in the middle, r = 1, which means that (W/L)_p:

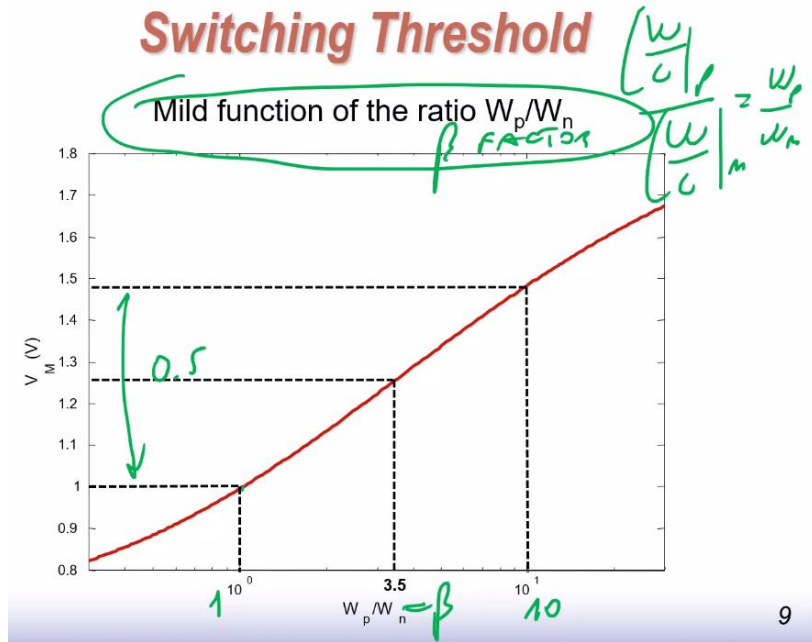
$$\left(\frac{W}{L} \right)_p = \frac{\mu_n}{\mu_p} \left(\frac{W}{L} \right)_n$$

Unfortunately, in our technology it is not true that $V_{dsat,p} = V_{dsat,n}$ and $V_{tp} = V_{tn}$, but the result we can get is the same, since the ratio between the process transconductances is something like 3.8, very much similar to the found 3.5.

Moreover, **the threshold voltage is a function of the beta factor, not an absolute function of the aspect ratios.**

Simulations result

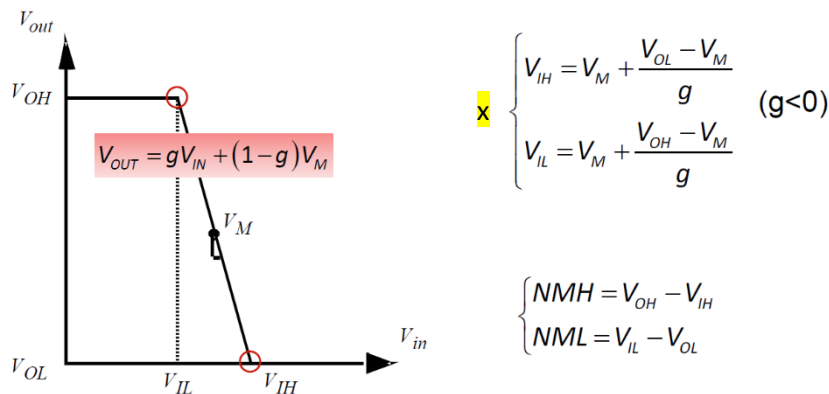
X axis is in log scale, y axis in linear scale. The result is that even if I vary the beta, the threshold voltage is not changing much → threshold voltage is a mild function of beta.



DETERMINING THE NOISE MARGINS – NMH and NML

The VTC is not linear. Let's approximate it with a piecewise linear approximation with 3 pieces of straight lines.

A simple approach: piecewise linear approximation



$V_{ol} = 0V$ and $V_{oh} = V_{dd}$.

The transition region has the same gain at threshold as the original characteristic.

The equation of the straight line is the one in the red box. How can we assess V_{il} and V_{ih} ? We substitute in the line formula V_{ol} (for V_{ih}) and V_{oh} (for V_{il}).

V_m has to be in the middle to maximize the noise margins because if we develop the expressions for NMH and NML, and $V_{oh} = V_{dd}$ and $V_{ol} = 0V$ in a FC-CMOS inverter, if the gain g is large, in formulas only the V_m term survives.

$$NMH = V_{dd} - V_m$$

$$NML = V_m$$

Hence $V_m = V_{dd}/2$ we maximize the noise margins.

Gain determination

Let's consider $(W/L)_p = 3.5$ and for the nMOS = 1, so that the beta factor is 3.5.

We need to compute the real gain of the inverter at the switching threshold, the analog small signal gain.

Determining the gain

The transistors at the switching threshold work as current generator:

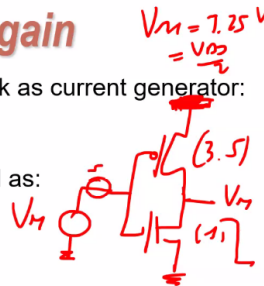
$$g = -(g_{mn} + g_{mp})(r_{op} || r_{on})$$

For both transistor g_m and r_0 can be assessed as:

$$g_m = \frac{\partial I}{\partial V_{GS}} \cong k' \left(\frac{W}{L} \right) V_{DSAT} \left(1 + \lambda \frac{V_{DD}}{2} \right) \cong k' \left(\frac{W}{L} \right) V_{DSAT}$$

$$r_0 = \left(\frac{\partial I}{\partial V_{DS}} \right)^{-1} = \frac{1}{\lambda \mu C_{ox} \left(\frac{W}{L} \right) V_{DSAT} \left(V_M - V_T - \frac{V_{DSAT}}{2} \right)} \cong \frac{1}{\lambda I_{DSAT}(V_M)}$$

$$\text{Thus, it turns out: } g \cong - \frac{k'_n \left(\frac{W}{L} \right)_n V_{DSATn} + k'_p \left(\frac{W}{L} \right)_p V_{DSATp}}{\lambda_n I_{DSATn}(V_M) + \lambda_p I_{DSATp}(V_M)} \cong -30$$



An inverter can be seen as two transistors in common source configuration. Firstly I compute the shortcircuit current: $icc = (g_{m,n} + g_{m,p}) * v_{in}$.

The R_{out} is then the parallel of the two output resistances: $R_{out} = r_{op} || r_{on}$.

Then the gain is $R_{out} * icc$ and it is equal to -30.

Then to compute the g_m we use the equations for the transistors assuming they are working in velocity saturation.

In the equations for g_m and r_0 $V_{gs} = V_m$ because there we are evaluating the parameters. λ is the only parameter that changes with the length, because it is equal to $1/V_a$.

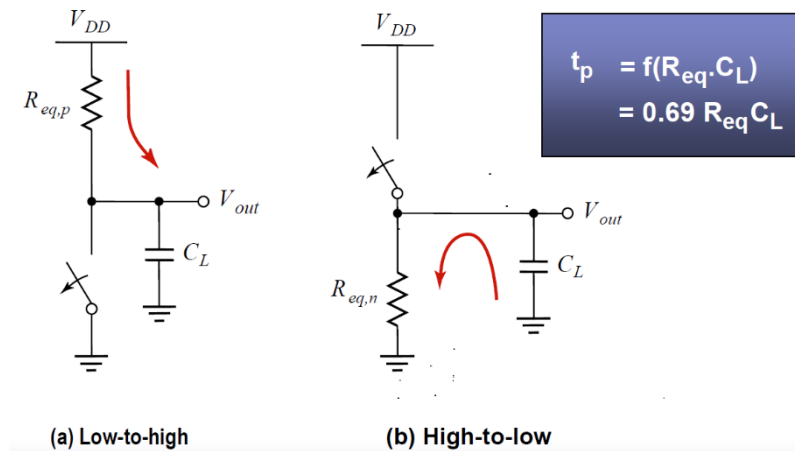
If I perform the derivative of the voltage transfer characteristic (the real one), we notice that the gain is not reaching the value of -30, but it is more or less -17.

So if we assess the real noise margins are: $V_{il} = 1.03V$, $V_{ih} = 1.45V$ and $NMH = 1.05V$, $NML = 1.03V$ (with a value of -30, $NML = NMH = 1.21V$).

DYNAMIC PERFORMANCES

We want to assess the propagation delay to get to a formula $\tau_p = \ln(2) \cdot R_{eq} \cdot C_L$.

RC MODEL TO EVALUATE THE DELAY



We want to model the transistor as a single resistance. Let's consider the pull up transition, it is performed by the pMOS, which can be modelled by an equivalent resistance, it is not that the transistor works as a resistance, it is not a static resistance R_{on} , but it is an equivalent resistance, the transistor is not in ohmic region.

For the pull down transistor it is the nMOS that instead discharges the output capacitance. The equivalent resistance can be easily computed, the problem is to assess the capacitance value.

For the pMOS let's assume an aspect ratio three times the aspect ratio of the nMOS, so $\beta = 3$. Moreover, the minimum aspect ratio for a transistor, in any technology, is 1.5. **The size of the inverter corresponds to the aspect ratio of the nMOS transistor.**

Inverter size

$$\left(\frac{W}{L}\right)_p / \left(\frac{W}{L}\right)_n \approx 3 \quad \text{assures } V_M \cong V_{DD}/2$$

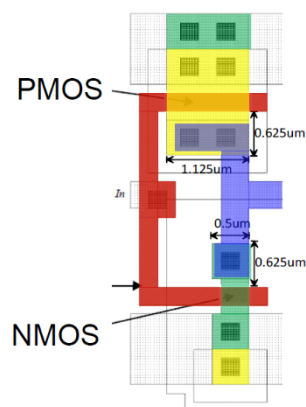
$$\left(\frac{W}{L}\right)_n = \frac{0.375 \mu\text{m}}{0.25 \mu\text{m}} = 1.5$$

$$\left(\frac{W}{L}\right)_p = \frac{1.125 \mu\text{m}}{0.25 \mu\text{m}} = 4.5$$



$$s = 1.5$$

(inverter size)



In terms of parasitic capacitances, the protagonist is the pMOS because it has larger dimensions, it has a larger W .

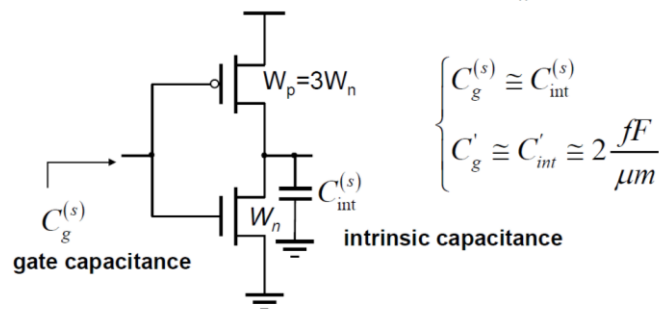
GATE AND INTRINSIC CAPACITANCES

The purpose is to assess the gate capacitance of the inverter and the capacitance at the output. I know the specific capacitance seen at the drain and at the gate of a transistor, $C'_g = C'_d = 2 \text{ fF}/\mu\text{m}$.

This said, the length of the transistor is the minimum, 0.25um (always the minimum in digital electronics).

The beta factor is chosen equal to 3, so that the Vm is approximately in the middle of the characteristic. This ensures also a good equivalence between the equivalent resistances of the transistors, so an equal propagation delay for the two transitions.

- Minimum length for both transistors
- $(W_p/W_n)=3$ to assure threshold in the middle and good equivalence between equivalent resistances, i.e., propagation delays
- The size s of in inverter corresponds to $(W/L)_n$



For $s=1.5$, $W_n=0.375\mu m$, $C_g^{(1.5)} \cong C_{int}^{(1.5)} = 2 \frac{fF}{\mu m} (W_n + W_p) = 3 fF$ 16

For a pMOS transistor with $W/L = 1$, the $Req = 31k$, while for the nMOS $14k$. So with $\beta = 3$, $V_m = V_{dd}/2$ and $Req,n = Req,p$, more or less.

The upper script s indicates the generic size s , where $s = (W/L)_n$ as previously said. We multiply the specific capacitance for the width of the transistor and we get C_g and C_{int} . This is the worst case analysis and it works fine.

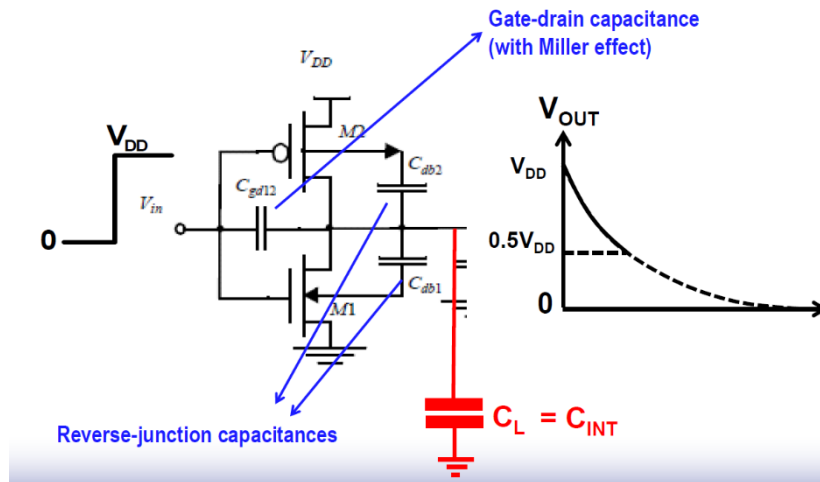
Detailed analysis

- The analysis carried out so far is approximated but works fine
- A more detailed analysis requires to assess all the contributions taking into account:
 - 1) Voltage dependence of capacitances on the voltage
 - 2) Miller effect when both nodes across the capacitance transition
- Aim of this analysis is just to verify how the approximation is good enough to compute an equivalent capacitance (at both gate and drain terminal)

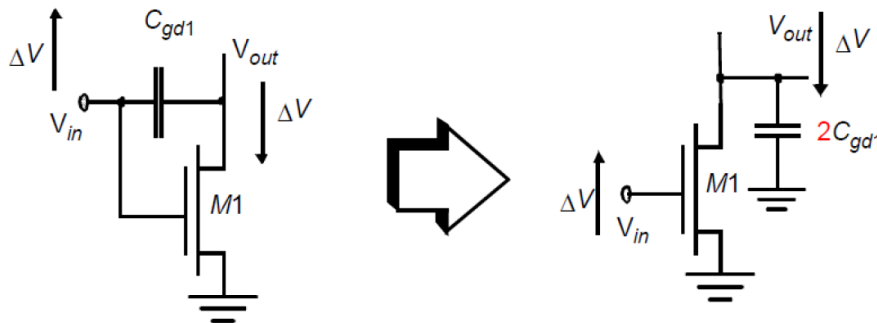
The capacitance that experiences the Miller effect is the overlap capacitance, that is the only contribution that matters in the gate-drain capacitance.

Intrinsic capacitance

Let's consider just the low-to-high transition of the input.



Miller effect



"A capacitor experiencing identical but opposite voltage swings at both its terminals can be replaced by a capacitor to ground, whose value is two times the original value."

Contributions to the intrinsic capacitance

For the low to high transition, the C_{int} is 3.07fF. With our analysis we assessed it equal to 3fF. The dominant contributions are the drain bulk capacitances, the diode capacitances, and the most impacting one is the one on the p side.

CAPACITOR	EXPRESSION
C_{GDn}	$2C'_{ov,n}W_n = 0.232 \text{ fF}$
C_{GDp}	$2C'_{ov,p}W_p = 0.607 \text{ fF}$
C_{DBn}	$0.58C'_{j,n}AD_n + 0.62C'_{j,SWn}PD_n = 0.348 \text{ fF} + 0.325 \text{ fF}$
C_{DBp}	$0.83C'_{j,p}AD_p + 0.88C'_{j,SWp}PD_p = 1.1 \text{ fF} + 0.46 \text{ fF}$

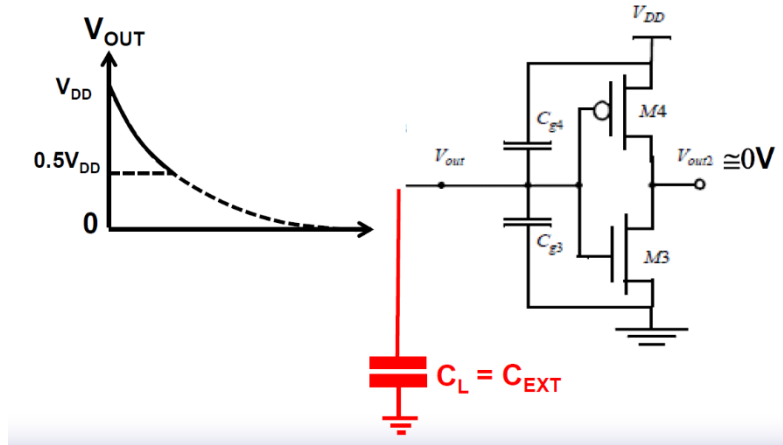
$$C_{int}^{(1.5)} \cong 3.07 \text{ fF}$$

A similar capacitance can be found for the high-to-low transition

A similar result can be obtained with the high to low transition. So the two transitions have similar equivalent capacitances equal to 3 fF.

Contributions to the Cext

Also for the gate capacitance Cext (extrinsic capacitance), the result is of 3.09 fF.



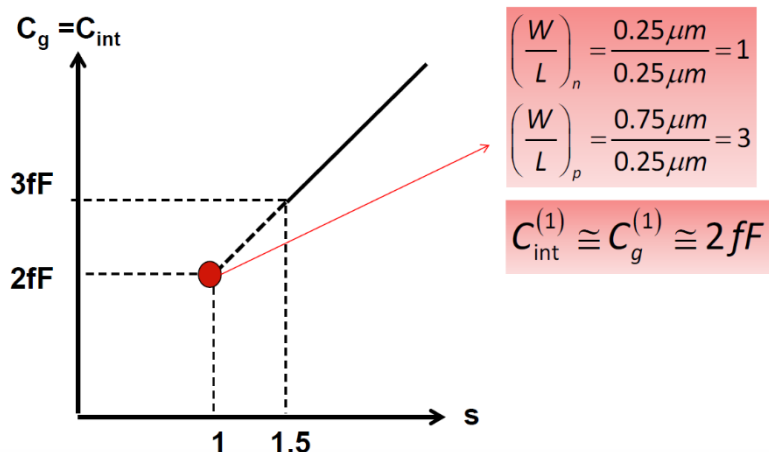
CAPACITOR	EXPRESSION
$C_{gn} = C_{GCn} + C_{ov,sn} + C_{ov,d3}$	$C'_{ox} W_n L_n + 2C'_{ov,n} W_n = 0.562 \text{ fF} + 0.232 \text{ fF} = 0.794 \text{ fF}$
$C_{gp} = C_{GCp} + C_{ov,sp} + C_{ov,d}$	$C'_{ox} W_p L_p + 2C'_{ov,p} W_p = 1.69 \text{ fF} + 0.608 \text{ fF} = 2.298 \text{ fF}$

$$C_g^{(1.5)} \cong 3.09 \text{ fF}$$

A similar capacitance can be found for the high-to-low transition

IDEAL MINIMUM SIZE INVERTER

Let's consider an inverter with size s and beta = 3 (threshold in the middle and approximately equal propagation delays). If I increase the size of the inverter, the capacitances increase because the aspect ratio of the nMOS is increasing, as well as the one of the pMOS.



Cext = Cint as seen before, and we have a linear relationship, so if I increase s from 1.5 to 15, the Cext = Cint = 30 fF, a factor 10 bigger.

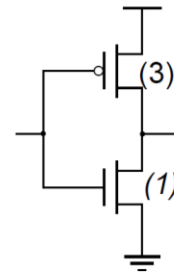
Moreover, an inverter with $s = 1$ is not physically implementable, but if I could implement it, its $C_g = C_{int} = 2 \text{ fF}$.

So there is a linear relationship between the size of the inverter and the gate and intrinsic capacitances, and the capacitance doesn't depend on the transition high to low or low to high.

An inverter with $s = 1$ and $\beta = 3$ will be considered the minimum size inverter.

INTRINSIC PROPAGATION DELAY

- Minimum length devices, $L=0.25\mu\text{m}$
- $W_p = 3W_n = 0.725\mu\text{m}$
 - switching threshold in the middle of supply range
 - approx. same pull-up and pull-down currents
 - approx. equal resistances, i.e., $R_N \cong R_P$
 - approx. equal rise t_{pLH} and fall t_{pHL} delays
- Analyze as an RC network:



$$\begin{aligned} \tau_{pHL} &= \ln(2) R_n^{(1)} C_{int}^{(1)} \cong 0.69 \cdot 13\text{k}\Omega \cdot 2\text{fF} \cong 18\text{ps} \\ \tau_{pLH} &= \ln(2) R_p^{(1)} C_{int}^{(1)} \cong 0.69 \cdot \frac{31\text{k}\Omega}{3} \cdot 2\text{fF} \cong 14\text{ps} \\ \tau_{p0} &= \frac{\tau_{pHL} + \tau_{pLH}}{2} = 16\text{ps} \end{aligned}$$

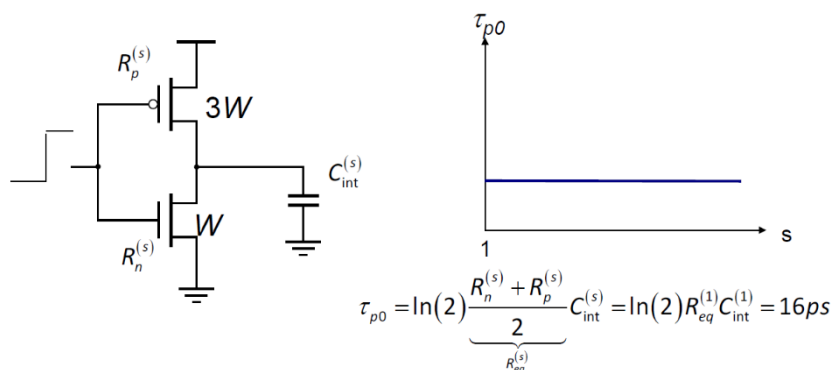
For the transition from H to L the nMOS plays, so I need its equivalent resistance (x). Furthermore, I'm also considering a minimum size inverter.

Of course we are computing these values for the 0.25um technology.

The one in the red box is the intrinsic propagation delays, without having the inverter loaded.

16 ps will be the intrinsic propagation delay of our inverter.

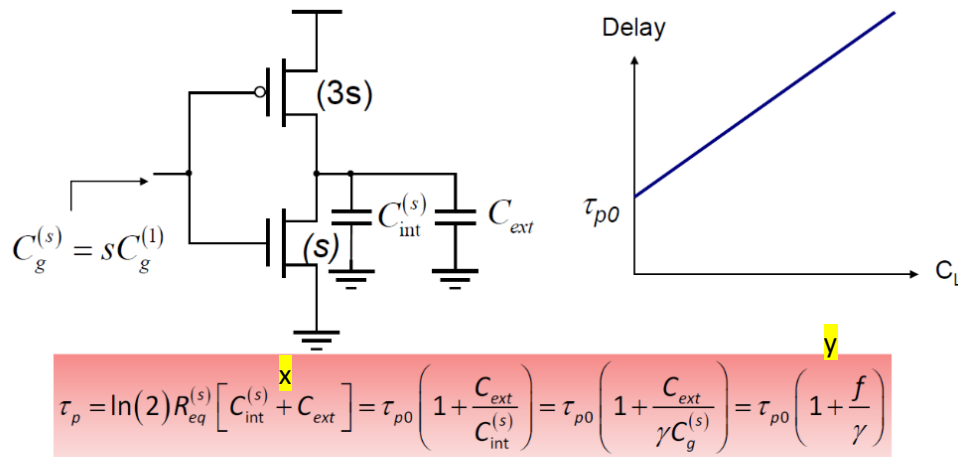
If I increase the size e.g. to 10, the $C_g = C_{int} = 20 \text{ fF}$, but also the equivalent resistance changes, they decrease by the scaling factor of the size (10 in this case). So overall the intrinsic propagation delay stays the same.



Intrinsic delay does not change with the size "s":
 R_{eq} decreases with s, whereas C_{int} increases with s

In reality, the R_{eq} is proportional to $1/s$, and the C to s , so they cancel out in the intrinsic propagation delay.

LOADED PROPAGATION DELAY



f = fan-out (or electrical effort)

γ = self-loading factor (=1 in almost all the technologies)

We are adding an extrinsic contribution. Now the capacitance in the formula of the delay is the sum of two contributions, C_{int} and C_{ext} .

I can factor out C_{int} from the propagation delay formula x.

Let's then define gamma the ratio C_{int}/C_g , and in most of the technologies gamma = 1. Gamma is called **self-loading factor**.

With this trick we arrive at the final formula y. In fact, C_{ext}/C_g is the so-called **fan-out**. It doesn't consider the intrinsic contribution, but just the extrinsic capacitance and the gate capacitance.

Tau_p0 depends on the technology and on the beta factor, but since everything is fixed, it is a constant number.

The only parameter with which we can change the delay of a gate is to change the fan-out. If I increase C_{ext} (C_L), the fanout increases linearly and also the propagation delay increases linearly.

How can I reduce the delay of an inverter?

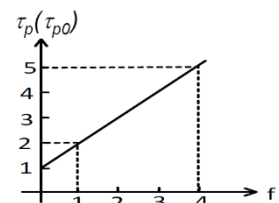
The only trick we can adopt is increasing the sizing, so the width of the transistor so that C_g (and C_{int}) is increased and also R_{eq} is decreased, and the fanout is decreased.

The best delay we can get is tau_p0, the intrinsic delay.

Inverter sizing

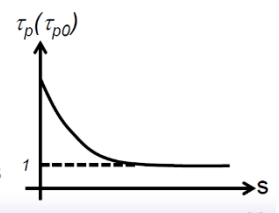
If we increase C_{ext} , the τ_p increases linearly. If instead we consider a fixed C_{ext} , if we increase the size τ_p decreases up to the minimum value τ_{p0} . The size that corresponds to 16 ns is the maximal one, no reason to increase it further.

- Once fixed the size of the inverter, increasing the load brings to a linear increase of the delay

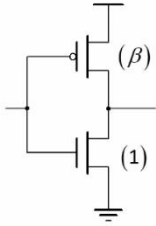


- Increasing the size for a given load makes the delay subside

increasing the size until the delay does not improve anymore (self. loading)



INVERTER SIZING



✓ Switching threshold at $V_{DD}/2$ to maximize both noise margins

↓

$\beta = 3.5$

✓ But let's take a look at τ_{pHL} and τ_{pLH}

$$\tau_{pHL} = \ln(2)(13k\Omega) \underbrace{c'(W_n + W_p)}_{C_{int} = 4.5c'l_{min}} = 20.3ps$$

$$\tau_{pLH} = \ln(2) \left(\frac{31k\Omega}{3.5} \right) \underbrace{c'(W_n + W_p)}_{C_{int} = 4.5c'l_{min}} = 13.8ps$$

To get τ_{pHL} closer to τ_{pLH} without compromising noise margins:

$\beta = 3$

$$\left. \begin{array}{l} \tau_{pHL} = 18ps \\ \tau_{pLH} = 14.3ps \end{array} \right\} \tau_{p0} \cong 16ps \quad V_M = 1.22V$$

Typically beta is the ratio of the aspect ratios, pMOS over nMOS. To have the V_m (switching threshold of the inverter) at half V_{dd} to maximize both noise margins, $\beta = 3.5$, so it is an inverter with the best reliability (highest noise margins).

But if we look at the intrinsic propagation delay for the two transitions. R_{eq} in our technology, $0.25\mu m$, is $13k\Omega$ for the nMOS. Then $C_{int} = C'$ (specific capacitance per unit width) multiplied by $W_n + W_p$, where W_i indicates the width $\rightarrow C'(W/L_p + W/L_n) * L_{min} = C_{int}$.

With a $\beta = 3.5$ we get $\tau_{HL} = 20.3ps$.

In the second case from low to high, the equivalent resistance scales down with the aspect ratio, so we need to divide by the beta. $\tau_{LH} = 13.8ps$ (at first approximation the C_{int} is the same).

Then to assess the propagation delay I take the average, but in this case the two propagations delays are very different.

Typically, in digital electronics the worst-case scenario is what counts the most, but we still take care only of the average propagation delay whatever the transition because somehow we can have both the transition in some nodes, so the average is more important, because it is not that in any node we have the worst transition \rightarrow average = 17ps.

Let's try to compromise the reliability to improve the delay time. So we choose $\beta = 3$. In this case the threshold is not in the middle exactly, but not very far from it, very close to it. With $\beta = 3$, pMOS size changes and the $\tau_{HL} = 18ps$ (faster), but $\tau_{LH} = 14.3ps$ (slower). However, overall we have an average of 16ps, better.

So we don't have the best reliability but the best propagation delay, also because in this case the equivalent resistances of the pMOS and nMOS are more similar \rightarrow **the idea is to better balance the transistors in terms of R_{eq} .**

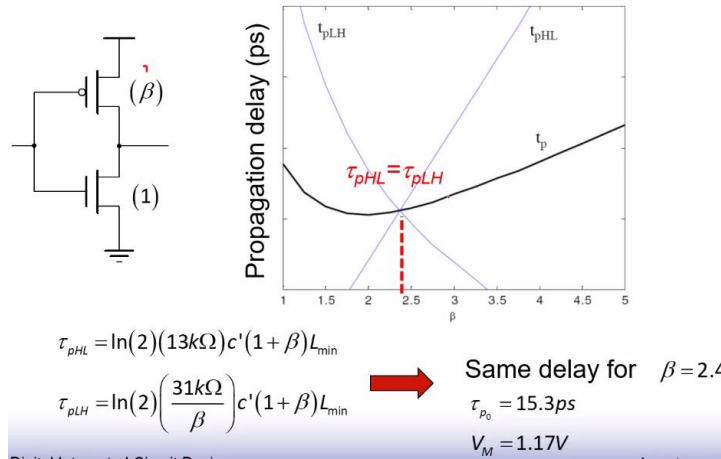
Equal delay

$$R_{eq,p} = R_{eq,p0}/\beta$$

The beta factor acts on the R_{eq} but also on C_{int} , where I have the term $W_n + W_p$, and W/L_p is beta.

The high to low transition increases proportionally to beta, while the low to high dependance on beta is more complex because we have beta both at numerator and denominator, so we have like a parabola. When beta tends to inf, we reach a plateau of more or less 10ps for the low to high transition, and the dependance on beta is not present because it is like if the nMOS transistor is negligible.

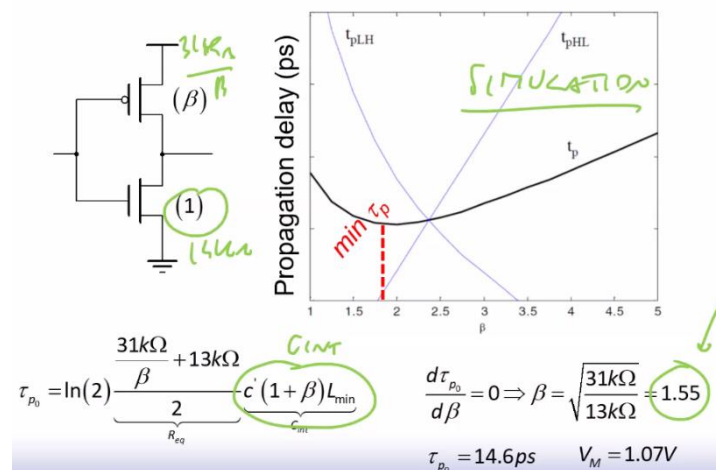
For $\beta = 2.4$ we have the same propagation delay. It can be assessed graphically or numerically, it's the same.



Now the delay is minimum, 15.3ps, but I have worse noise margins, compromising the reliability to have better delays.

In the point where the delay is the same, also the Req of the two transistors are the same.

In the graph, it seems that there is a minimum point in the average propagation delay curve t_p . Mathematically speaking, to get the minimum we need to take the derivative of the average of the delays with respect to beta and put it to 0.



The minimum delay is 14.6ps, but the $V_m = 1.07V$.

A value of $\beta = 2.4$ has the advantage, since the aspect ratio of the pMOS is reduced, of reducing the area of the pMOS, and also of reducing the power consumption of the pMOS, because the capacitance is smaller. Hence the **best beta factor is 2 or 2.4** (still however compromising the reliability).

In our case, we will however use $\beta = 3$.

ENERGY VS DELAY TRADE-OFF

For sure there is a balance between power consumption and propagation delay. For a given technology, we consider the inverter and for the inverter we assess the energy delay product, which has to be minimized.

Since energy and delay are two features traded-off, this product is a figure of merit for a given technology. Let's consider the energy for a switching event (low to high or high to low). To be honest, only for the low to high transition energy is spent (pull-up) of the voltage across a capacitor, and this energy is $C(V_{dd})^2$. For the pull down no energy is consumed. C and V_{dd} depend on the technology.

$$E_{0 \rightarrow 1} = C(V_{DD})^2$$

$$E_{1 \rightarrow 0} = 0$$

$$\tau_p \cong \ln(2)C \left[\frac{3}{4} \frac{V_{DD}}{KV_{DSAT}(V_{DD} - V_T - 0.5V_{DSAT})} \right]$$

**Energy-delay product:
average energy per transition x delay**

$$EDP = E_{ave} \tau_p \propto \frac{(V_{DD})^3}{V_{DD} - V_T - 0.5V_{DSAT}}$$

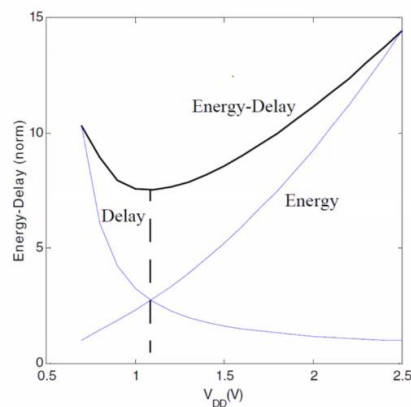
EDP is minimized for a power-supply voltage equal to:

$$V_{DD} = \frac{3}{2}(V_T + 0.5V_{DSAT}) \cong 1.1V$$

Let's now assess the intrinsic propagation delay of the inverter. Again, the reference technology is 0.25um from Intel. C_{int} is not a function of V_{dd}, while R_{eq} depends on the technology because depends on V_{dd} (formula seen previously). Of course, if we change the technology the value changes.

If we multiply the average energy (1/2 * C * V_{dd}²) per event spent by the power supply generator by the intrinsic delay, we have a V_{dd}³ at the numerator, and V_{dd} - V_{te} (V_{te} = effective threshold = V_t + 0.5*V_{dsat}) at the denominator.

As we can see in the plot below, the delay decreases with V_{dd}, and then it reaches a plateau for V_{dd} that tends to inf. Energy instead increases quadratically.



In our circuit, we can change V_{dd} up to a maximum value of 2.5V in 0.25um. We could overpower the circuit, it works for a while but then it collapses.

Now, if I have a circuit that doesn't have to work as fast as it can, I don't want to optimize the dynamic performances. To decrease the power consumption, which is the V_{dd} to choose? Of course not the maximal one, the minimum value permitted is the value that grants for the circuit to work more or less at 1kHz, e.g. 0.7V. Thus we minimize the power consumption.

Vice versa, if I don't care about the power consumption but I want a much larger frequency, I can use 2.5V, eventually also overpowering the circuit, but not for a long time.

If we decrease V_{dd} the energy consumed decreases but the dynamic performances (the delay) is worsened.

If we decrease Vdd, the characteristic of the inverter becomes flatter, so also the small signal gain in the transition region decreases. There is a minimum value corresponding to $3 \cdot V_{thermal} = 75mV$ to have the inverter working and a reliable characteristic so that the small signal gain is bigger than 1. In fact, for an inverter to work fine the slope must be at least 1. Of course, with this value the transistor is very slow, because it works in the subthreshold regime.

Design for performance

Cext is the extrinsic contribution of another circuit. Cg(s) means that the gate capacitance of the inverter is $C_g(s) = s \cdot C_g(1)$, so linear proportionality. Cg(1) is the gate capacitance of the minimum size inverter. Same reasoning for Cint(s)

$\beta = 5$ $\left(\frac{W}{L}\right)_p = 3 \left(\frac{W}{L}\right)_n$

\square Keep capacitances small

\square Increase transistor sizes

- watch out for self-loading!

\square Increase V_{DD} (????)

$C_g^{(n)} = 5 C_g^{(1)}$

\downarrow

$5 = 1$

Instead, Req scales down with the size (1/s).

$$t_p = \frac{C_L}{I} \cdot R_{eq}^{(n)} \cdot (C_{int}^{(n)} + C_{ext}) = \frac{C_L}{I} \cdot R_{eq}^{(n)} \cdot C_{ext} + \frac{C_L}{I} \cdot R_{eq}^{(n)} \cdot C_{int}^{(n)}$$

x

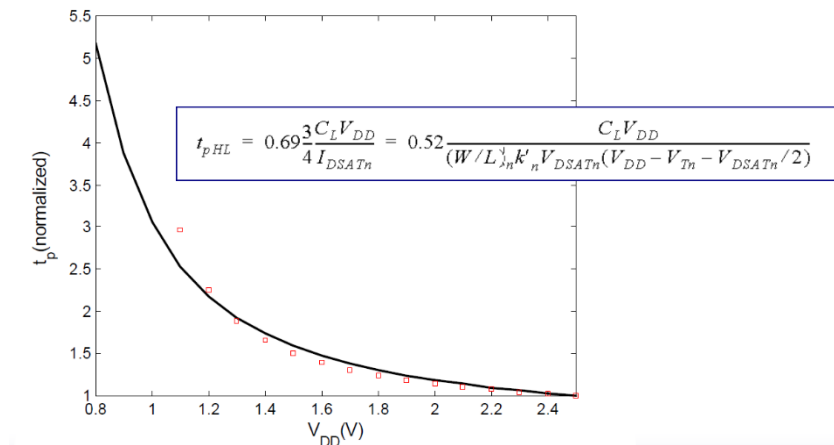
Term x is the intrinsic propagation delay, it is a number independent on the size, because capacitance and resistance scale with opposite trends.

In the other term we have a 1/s dependence in Req(s).

To improve performance, to decrease the propagation delay, I can act on different terms:

- Keep the capacitances as small as possible, even if sometimes it cannot be done, e.g. if we have a load \rightarrow use minimum area transistors and circuits, because this means reducing the capacitances, so power consumptions and delays..
- For what concerns the minimum size inverter, we can increase the size but at a certain point we reach a plateau τ_{p0} , over which there is no more benefit.
- Increasing Vdd decreases the intrinsic propagation delay, but there is a limit to Vdd. In any case, better not to use a smaller Vdd to optimize the intrinsic propagation delay \rightarrow use the maximum Vdd we can apply to the technology.

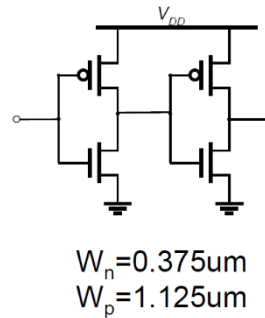
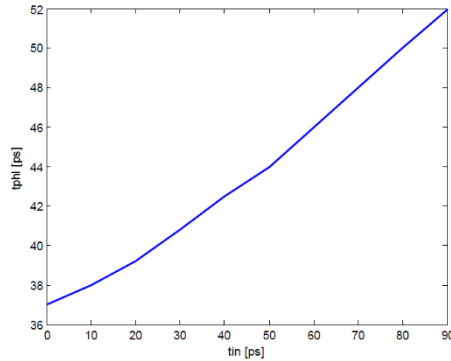
Vdd has kept to the maximum value as possible because the capacitance is a mild function of Vdd, only the junction contribution to it depends on Vdd. What changes with Vdd is Req, which improves increasing Vdd.



In the image we have two inverters in cascade. Up to now we have considered an abrupt transition at the input, a step function, which is not present in real circuits.

If we don't have an ideal signal at the input, let's assess τ_p as a function of $\tau_{p,in}$. In the plot, τ_p vs $\tau_{p,in}$ different values.

τ_p has a quite linear dependence with respect to $\tau_{p,in}$.



$$x \quad t_p^{(i)} = t_{step}^{(i)} + \eta t_{step}^{(i-1)}$$

$$\eta \cong 0.25$$

x is the real expression. τ_{step} is the classical expression considering an abrupt transition at the input, and ideal one. Factor η depends on the technology. However, since $\eta = 0.25$, the dependence on $\tau_{p,in}$ is mild.

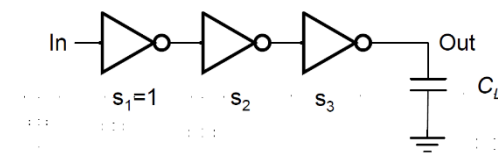
If we have similar propagation delays, considering just the ideal delay time leads to an error of 25% more or less, so even if the circuit is real, let's consider the intrinsic propagation delay as if the input signal is ideal, so the ideal delay.

INVERTER CHAIN

Let's suppose to have a capacitive load to be driven outside the IC package with the minimum propagation delay as possible.

Given C_L , let's connect directly one inverter to the capacitance. The size s_1 that minimizes the propagation delay is the smallest possible one, but we have to drive the inverter with a uC, and the uC may struggle to drive a very big inverter. So at the output of the digital circuit uC we put a minimum size inverter so that the processor doesn't struggle in driving it. But in this way the propagation delay increases, because the inverter is smaller (fanout is very large).

To reduce the propagation delay from the output of the processor to the C_L we add inverters in the middle, even if it seems counter-intuitive.



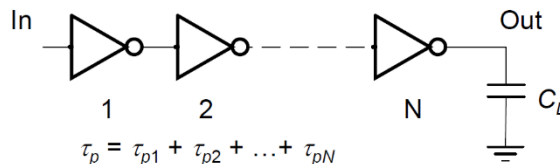
If C_L is given:

- How many stages are needed to minimize the delay?
- How sizing the inverters?

Once we have circuits in cascade, **the overall propagation delay is the sum of the propagation delays assessed in ideal conditions, so the ideal propagation delays** with an abrupt transition at the input.

INVERTER CHAIN DELAY

C_L and s_1 (minimum size) are fixed.



$$\tau_p = \tau_{p1} + \tau_{p2} + \dots + \tau_{pN}$$

$$\tau_{p,j} = 0.69R_{eq}^{(1)}C_{int}^{(1)} \left(1 + \frac{C_{g,j+1}}{\gamma C_{g,j}} \right) = \tau_{p0} \left(1 + \frac{C_{g,j+1}}{\gamma C_{g,j}} \right)$$

$$\tau_p = \sum_{j=1}^N \tau_{p,j} = \tau_{p0} \sum_{i=1}^N \left(1 + \frac{C_{g,i+1}}{\gamma C_{g,i}} \right), \quad C_{g,N+1} = C_L$$

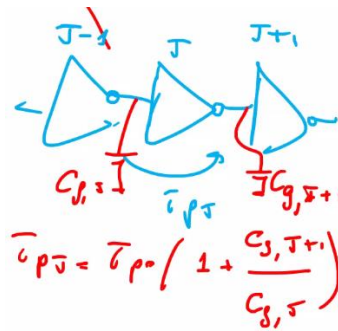
Let's express the propagation delay as the expression of physical terms.

$$\tau_p = \tau_{p1} + \tau_{p2} + \dots + \tau_{pN}$$

$$= \sum_{j=1}^N \tau_{p,j} = \sum_{i=1}^N \tau_{p0} \left(1 + \frac{f_i}{\gamma} \right)$$

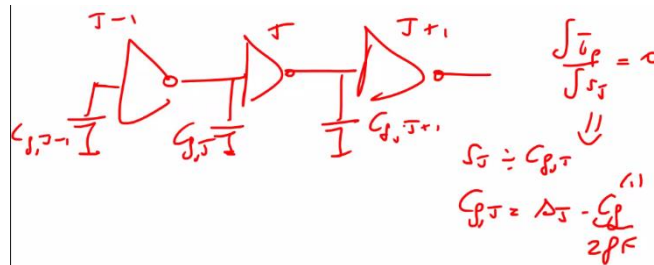
Now we explicit f_i , the fanout of the stage i . Let's design the i staged, preceded by the $i-1$ staged and followed by the $i+1$ stage.

I'm considering the delay of the stage i.

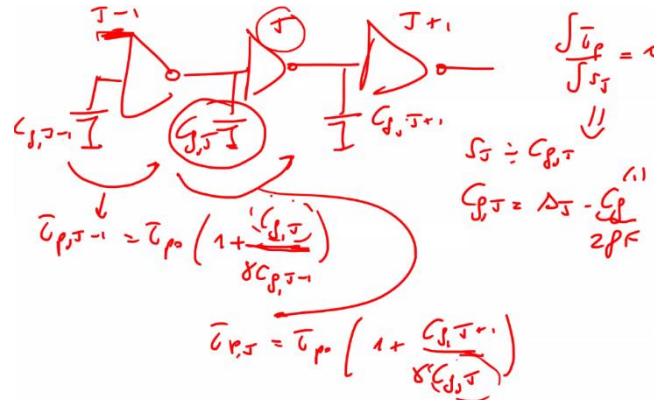


The unknowns of this design, if the number N of inverters is fixed, are the sizes of the inverters up to s_N , except for s_1 . The propagation delay is $\tau_p = f(s_1, s_2, \dots, s_n)$. We should take the derivative of τ_p with respect to any variable s_1, s_2 and so on and equate it to 0.

Let's consider a generic J stage. I have to put $d\tau_p/ds_j = 0$. But s_j , size of the generic J stage, is proportional to $C_{g,j}$, so we can assess the derivative in terms of $C_{g,j}$.



$C_{g,j}$ appears in both the propagation delays of inverter J-1 and inverter J.



So in assessing the derivative $d\tau_p/dC_{g,j}$, I have to consider the two terms.

$$\frac{d\tau_p}{dC_{g,j}} = 0 \rightarrow \cancel{\tau_{po}} \frac{1}{C_{g,j-1}} - \cancel{\tau_{po}} \frac{C_{g,j+1}}{\delta(C_{g,j})} = 0$$

The result is the following.

$$\left\{ \begin{array}{l} C_{g,j} = \sqrt{C_{g,j-1} \cdot C_{g,j+1}} \\ \frac{C_{g,j}}{C_{g,j-1}} = \frac{C_{g,j+1}}{C_{g,j}} \end{array} \right.$$

The first term of the second equation is the fanout of the J-1 stage, and the same for the inverter J: $f_{j-1} = f_j$. So the delay is minimized if the generic inverter J has the same fanout of the preceding one.

This means that **all the inverters must have the same fanout**, which also means that all the inverters have the same delay. So if we put in cascade inverters, we minimize the delay if all the inverters have the same fanout, i.e. same electrical effort.

Instead, if we look at the first formula, it means that **if we take a capacitance, this capacitance is the geometrical mean between the following and previous capacitance**.

Since the last stage has to drive the load, the last capacitance is indicated as C_L .

1. Delay equation has $N-1$ unknowns, i.e., $C_{g,2}, C_{g,3}, \dots, C_{g,N}$
2. Minimize the delay, find $N - 1$ partial derivatives
3. Result: $C_{g,j+1}/C_{g,j} = C_{g,j}/C_{g,j-1}$
4. Size of each stage is the geometric mean of two neighbors

$$C_{g,j} = \sqrt{C_{g,j-1} C_{g,j+1}}$$

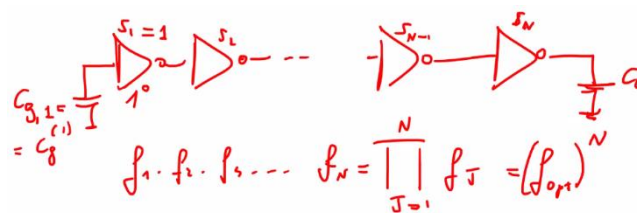
- each stage has the same effective fan-out, $f_j = f_{j+1}$
- each stage has the same delay, $\tau_{p,j} = \tau_{p,j+1}$

Since the capacitance at the gate and output node are proportional to the size, we can assess the generic capacitance $C_{g,j}$ as the gate capacitance at size s_1 times the size.

The results we can get are the one in the image above, already described.

Let's now consider the chain of inverters. The first capacitance is C_{g1} , but since $s_1 = 1$, $C_{g,1} = C_g^{(1)}$, but only in the first inverter, if $s_1 = 1.5$, $C_{g,1} = C_g^{(1.5)}$.

This said, let's write the product of all the fanouts all the way down to f_n . All the stages have the same fanout, that we can call f_{opt} , the one we want to assess.



Let's compute the product expressing the fanouts.

$$f_1 \cdot f_2 \cdot f_3 \cdot \dots \cdot f_N = \prod_{j=1}^N f_j = (f_{opt})^N$$

$$\frac{C_{g,2}}{C_{s,4}} \cdot \frac{C_{s,3}}{C_{g,2}} \cdot \frac{C_{g,4}}{C_{s,3}} \cdot \dots \cdot \frac{C_{g,N}}{C_{s,N-1}} \cdot \frac{C_L}{C_{s,N}} = (f_{opt})^N$$

The product cancels out, and in the end the only two terms remaining are $C_L/C_{g,1}$. This ratio is called **F**, that is the **Path Fanout** (or path electrical effort).

Then, $F = (f_{opt})^N$.

Since N is fixed, we can assess the optimal fanout.

$$F = \frac{C_L}{C_{g,1}} = (f_{opt})^N \rightarrow f_{opt} = \sqrt[N]{\frac{C_L}{C_{g,1}}}$$

This is the optimal fanout for all the inverters of the chain of inverters to minimize the delay.

Optimal fanout

When each stage is sized by f and has same fan-out f :

$$f^N = F = C_L / C_{g,1} \quad F = \text{path fan-out}$$

Thus, the fan-out for each stage is:

$$f_{opt} = \sqrt[N]{F}$$

Minimum path delay:

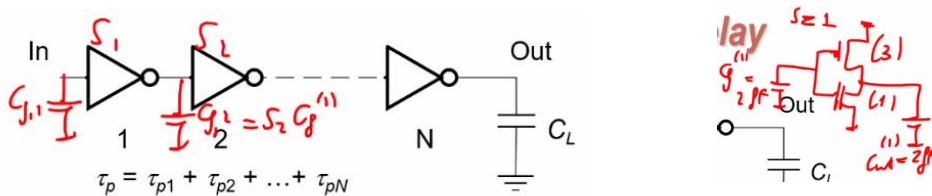
$$\tau_p = N\tau_{p0} \left(1 + \frac{\sqrt[N]{F}}{\gamma} \right)$$

Stage sizing:

$$S_j = S_1 f_{opt}^{j-1} \quad \mathbf{x}$$

The propagation delay is the sum of all the propagation delays, but since the fanout is the same, the propagation delay is the same for any stage. So the total delay is the N sum of the delay of a single stage inverter. This is the minimum delay we can get.

As for the stages sizing, the condition that we have to grant is to have a fanout = f_{opt} . Let's write the capacitances of the chain. The reference inverter is the one on the right.



Let's write the fanout of the first stage. We find that is the ratio between the size of the following inverter and the size of the inverter itself, and it must be equal to f_{opt} .

$$\frac{C_{g,2}}{C_{g,1}} = \frac{S_2 C_{g,1}}{S_1 C_{g,1}} = \frac{S_2}{S_1} = f_{opt} \quad S_2 = S_1 f_{opt}$$

We can do the same thing for the second inverter, getting s_3/s_2 , but s_2 is already determined.

$$\frac{C_{g,3}}{C_{g,2}} = \frac{S_3}{S_2} \rightarrow S_3 = S_2 f_{opt} = S_1 (f_{opt})^2$$

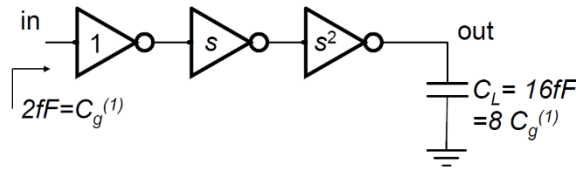
Generalizing, we get the expression x.

Example

$N = 3$, $C_L = 16$ fF and $s_1 = 1$ as data.

Since $s_1 = 1$, $C_{g,1} = 2$ fF.

In the end, $F = C_L / C_{g,1} = 8$



$C_L/C_g^{(1)}$ has to be equally distributed across $N = 3$ stages:

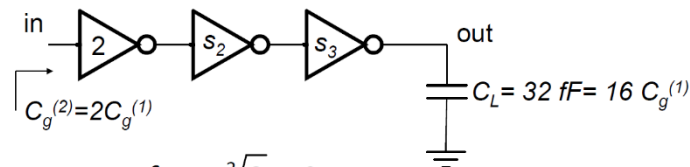
$$f_{opt} = \sqrt[3]{8} = 2$$

Since the first inverter is minimum, being $s_1=1$:

$$s = f_{opt} = 2$$

As for the delay, it is N times, so 3 times the single delay, that is 9 times τ_{p0} . We can notice that if the minimum size inverter is directly connected to the load, the delay would be the same.

Example of non-minimum input stage



$$f_{opt} = \sqrt[3]{8} = 2$$

but since the first inverter is not minimum:

$$s_2 = 2(f_{opt})^1 = 4$$

$$s_3 = 2(f_{opt})^2 = 8$$

Same procedure of the previous example. Now the F is the same as before because the input stage is not minimum size, but twice the minimum size, and also the load is (32 fF).

OPTIMUM NUMBER OF STAGES

Now s_1 is fixed, C_L is fixed but N is a variable. The formula for f_{opt} is valid whether N is a parameter or not.

For a given load, C_L and given input capacitance C_g
find the optimal fan-out f_{opt} :

$$\tau_p = N\tau_{p0} \left(1 + \frac{\sqrt[N]{F}}{\gamma} \right)$$

Searching for the best number of stages:

$$x \quad \frac{\partial \tau_p}{\partial N} = \gamma + \sqrt[N]{F} - \frac{\sqrt[N]{F} \ln(F)}{N} = 0 \quad \longrightarrow \quad f_{opt} = e^{\left(1 + \frac{\gamma}{f_{opt}} \right)}$$

A closed-form solution only if $\gamma = 0$: $f_{opt} = e$

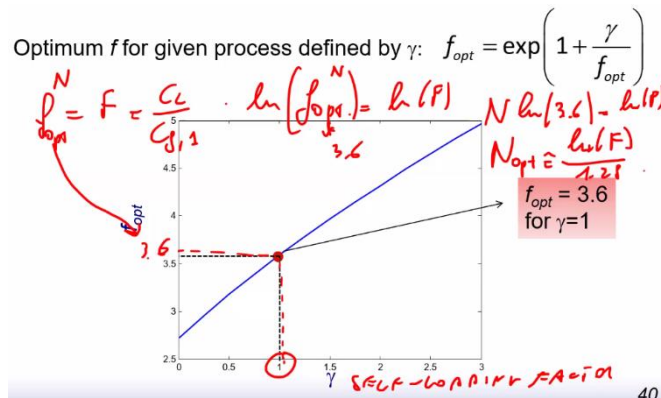
$$N = \ln(F)$$

Let's now reason on the propagation delay of the chain of inverters. To obtain the best N value, we take the expression of the delay and we take the derivative (not the partial one because N is the only unknown) and put it to 0.

The Nth-root of F is f_{opt} . Our variable of interest in x is f_{opt} , but we get an implicit result, because f_{opt} is in the both sides of the equation.

But if $\gamma = 0$, so the $C_{int} = 0$ (the inverter has an output node without parasitics), $f_{opt} = e = 2.72$.

We can plot f_{opt} as a function of γ .

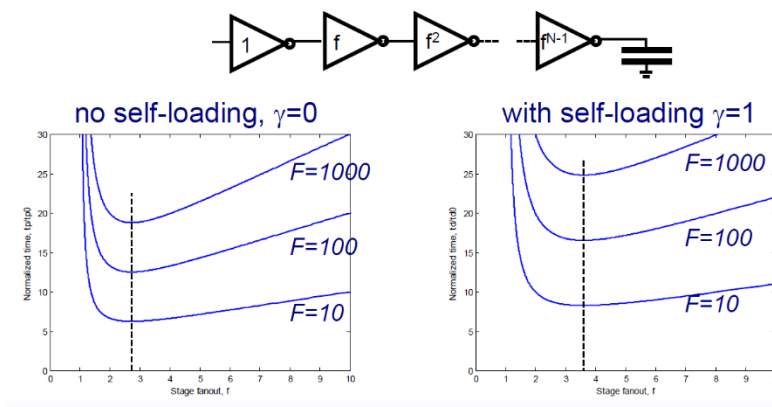


If I can decide also the number of stages, the best is to select the number of stages, from a mathematical perspective (not electronic one), to get the minimum delay, that corresponds to a $f_{opt} = 3.6$. We have to select N such that $f_{opt} = 3.6$, i.e. $f_{opt}^N = F$ (if $\gamma = 1$).

The problem is that $N = \ln(F)/\ln(3.6)$ is not an integer number typically. So what should we do? The answer will be given later on.

Impact of self-loading on the optimum delay τ_{p0}

Chain of inverters with the same fan-out (f) and $N = \ln(F)/\ln(f)$



I'm plotting the delay, normalized by τ_{p0} , as a function of the fanout of the stage. On the x axis I'm changing f and accordingly the number of stages N . On the left, if $\gamma = 0$, the minimum is always reached at 2.7, that is e . on the right, we have the minimum always at 3.6. So whatever the load, the minimum of the delay is reached for the sizing corresponding to a fanout = 3.6. This might correspond to a N value that is not an integer, because it is a pure mathematical analysis.

An important thing. The minimum of the three curves on the right is rather flat if we move from left to right on the curves, so 3.6 is not used, typically something bigger is used, like 4 or 5. The advantage of

selecting a higher f allows to reduce the overall area of the circuit, which is a benefit in terms of cost and power consumption.

BUFFER DESIGN

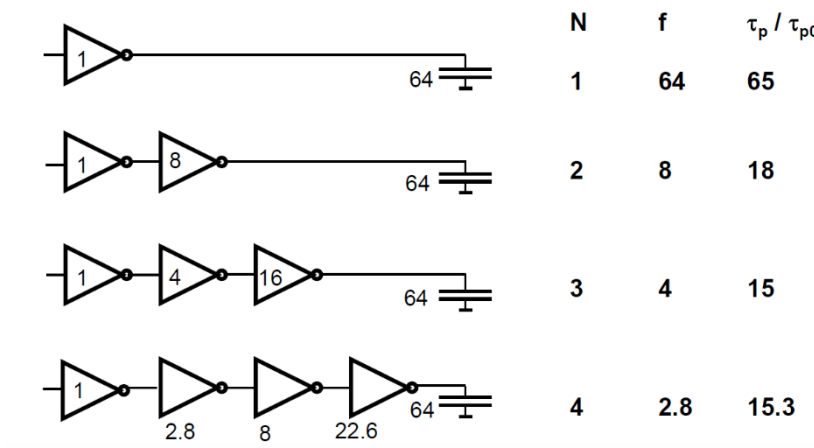
S_1 is fixed, then C_L is fixed ($64 \cdot C_g^{(1)}$). Hence $F = 64$. The goal is to minimize τ_p , from the input to the load.

Initially, $N = 1$, so the delay expressed in terms of τ_{p0} is 65.

Let's add an inverter in the chain. Once N is fixed, the optimum fanout is not 3.6, because 3.6 is when N is a free parameter. So $f_{opt} = \sqrt{F} = \sqrt{64} = 8$.

Now let's add a third inverter. Of course, I need to reoptimize all the stages. $f_{opt} = \sqrt[3]{F} = \sqrt[3]{64}$, which is 4, so we are close to 3.6.

The 4 inverters solution is good in terms of inverters, but the problem is that, despite the delay that is slightly bigger, we have an area of $2.8 + 8 + 22.4$, and it is a very huge size, which means huge area. So the circuit with 3 inverters is much smaller than the one with 4 inverters.



Remark

If I had just s_1 and C_L , the first thing to do is to compute N_{opt} .

$$N_{opt} = \frac{\ln(F)}{\ln(f_{opt})} = \frac{\ln(64)}{\ln(3.6)} \approx 3.25$$

$N_{opt} = 3.25$ stages, and for each stage the fanout is 3.6, whatever F , because we have $f_{opt} = 3.6$ in concordance with N_{opt} . But unfortunately I cannot design 3.25 stages, but do I use 3 or 4 stages? The closer, 3.

But with $N = 3$, we need to assess the $f_{opt} = \sqrt[3]{F} = 4$.

In general, the value of f_{opt} that I want to get is very close to 3.6, so if it is very far from 3.6 it is maybe an error.

NORMALIZED DELAY FUNCTION OF F

Unbuffered is with the minimum size inverter directly connected to the load capacitor. The values are the ones of τ_{p0} . The delay can be assessed as $\tau_p = \tau_{p0} \cdot (1 + F/\gamma)$.

The other possibility is to use two stages. N is fixed to 2, so $f_{opt} = \sqrt{F}$. The delay is $\tau_p = 2 \cdot \tau_{p0} \cdot (1 + f_{opt}/\gamma)$.

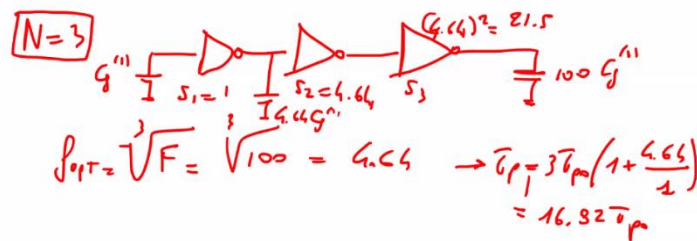
The last case is the inverter chain, so I can choose N because it is treated as a variable. It increases area and power, but in terms of delay it is the best solution.

Let's assess N_{opt} for the inverter chain. $\tau_p = N_{opt} \tau_{p0} (1 + 3.6)$. We have 3.6 because it is the optimum fanout.

$$\tau_p = N \tau_{p0} \left(1 + \frac{\sqrt[N]{F}}{\gamma} \right)$$

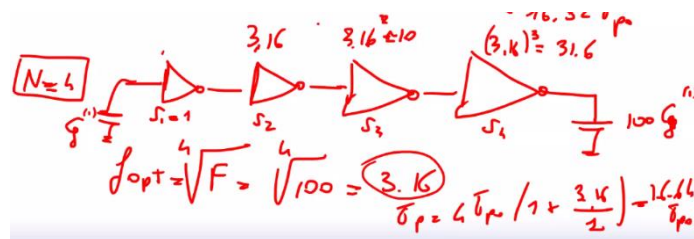
F	Unbuffered	Two Stage	Inverter Chain	N_{opt}
10	11	8.3	8.3	1.8
100	101	22	16.5	3.6
1000	1001	65	24.8	5.4
10,000	10,001	202	33.1	7.2

If $F = 100$, do I select $N = 3$ or 4?



If $N = 3$, the fanout is 4.64 for all the inverters for sure, so we have an equation more to verify that everything is fine. Moreover, I know that the fanout = $C_L / s_3 * C_g^{(1)}$, so the last inverter can be used to double check the sizing.

If $N = 4$:



In terms of delay, the best solution is to have $N = 4$, and in fact 3.6 is closer to 4 than to 3. However, if we consider area and power consumption, the $N = 3$ solution is better.

POWER DISSIPATION

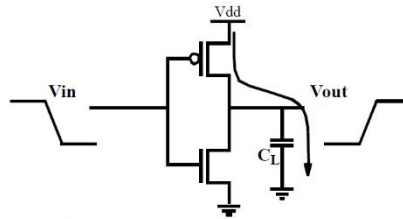
- **Dynamic power consumption**
Charging and discharging output capacitance
- **Cross-conduction current**
Short-circuit path between supply rails during switching
- **Leakage**
Leaking diodes and transistors

In digital circuits, there are 3 contributions to the power consumption. In good designs, the first one is the dominant one and the other two are negligible. The second occurs if, for whatever reason, both the pull up and pull down network are on at the same time, so it is like having a shortcircuit. This might occur during transients, of course not at steady state. The current that passes is called cross-conduction current.

Then the last contribution is the static DC contribution.

DYNAMIC POWER CONSUMPTION

It is the power that the PS generator delivers to charge the output capacitance. This corresponds to an energy $C_L V_{DD}^2 = V_{DD} Q = V_{DD} (V_{DD} C)$.



$$\text{Energy} = C_L V_{DD}^2$$

$$\text{Power} = \text{Energy per transition} = C_L V_{DD}^2 f_{0 \rightarrow 1}$$

- $f_{0 \rightarrow 1}$ is the frequency with which the output is pulled up
- Need to reduce C_L , V_{DD} and operating frequency to reduce power

If the pull up occurs with a frequency called **pull up frequency $f_{0 \rightarrow 1}$** , the power is the energy multiplied by $f_{0 \rightarrow 1}$.

If we can, reducing V_{DD} would be good because it contributes with the square power to the power consumption.

The problem is how to assess the pull up frequency.

Switching activity

An inverter is fed with a digital signal, and the duration of the bit is T_{bit} . I can define the clock frequency f_{clock} as $1/T_{bit}$, so the bit per second (**bitrate**).

Let's consider a stint of time in which we have n transitions from 0 to 1, $n_{0 \rightarrow 1}$. So for n times we charge the capacitance to V_{DD} . The overall energy spent to charge the capacitance is $C_L V_{DD}^2 n_{0 \rightarrow 1}$.

Now, let's define the switching activity as in the image. N is the number of bits in series. The duration of the bitstream is $N T_{bit}$. The probability with which we observe a pull up is the ratio in the image x .

- Consider to switch an inverter for N clock cycles:

$$E = C_L V_{DD}^2 n_{0 \rightarrow 1}$$

- E is the energy drawn by the supply for N clock cycles
- $n_{0 \rightarrow 1}$ is the number of 0->1 transitions at the output

- If the data at the input changes with a rate f_{ck} :

$$\alpha_{SW} = \lim_{N \rightarrow \infty} \left(\frac{n_{0 \rightarrow 1}}{N} \right) = P(0)P(1) \quad \text{switching activity}$$

$$P = C_L V_{DD}^2 \alpha_{SW} f_{ck}$$

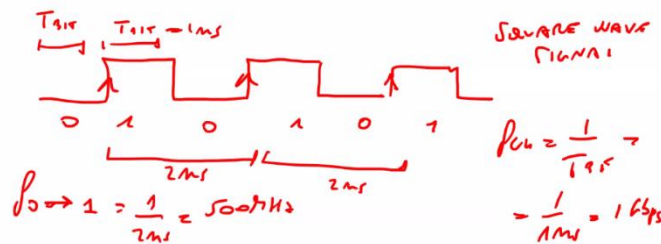
- In the case of equiprobable signal: $\alpha_{SW} = 0.5 * 0.5 = 0.25$

The limit of ratio x is the switching activity, which is just the probability of having a 0 in one bit period times the probability of having a 1 in the following bit period if in the previous one we had a 0 (conditional probability).

If we have a switching activity, we can assess the power consumption multiplying the energy per pull up event per the bitrate and the switching activity, whose product is $f_{0 \rightarrow 1}$. Of course, the switching activity is always smaller than 1.

Let's consider two different cases, and in the first one we have a deterministic signal in which every Tbit we change from 0 to 1 and viceversa. How much is the switching activity?

If Tbit = 1ns, every 2ns we have a pull up, so $f_{0 \rightarrow 1}$ is $1/2ns = 500MHz$. The switching activity is 0.5.



We could arrive to the same result considering the probability. In fact, the probability $P(0)$ to have a zero in a bit period is 0.5, because 50% of the times we have a 0 or a 1. Instead, $P(1)$ is 1 because we are sure that after a zero, the probability to have a 1 is 1. So the switching activity is $0.5 * 1 = 0.5$.

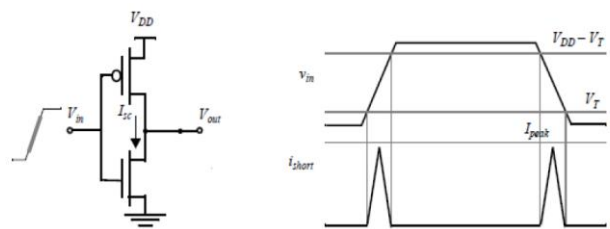
Let's consider now a random signal that can be 0 and 1 with the same probability. So in one Tbit I can have 50% probability of 1 and 50% of 0, and the same in the following Tbit.

The switching activity is now $0.5 * 0.5 = 0.25$, because the periods are unrelated. Hence the pull up frequency, if we consider the same signal, is 250MHz.

The case in which we deliver power from the PS generator is in the transition from 0 to 1. If the bit sequence is $1 \rightarrow 1$, $0 \rightarrow 0$ or $1 \rightarrow 0$, no power is dissipated.

CROSS-CONDUCTION POWER CONSUMPTION

It is due to the current from V_{DD} to GND during a transition event. Let's consider an ideal inverter with no input and output capacitances. If $C = 0$ at the output, the BW is infinite and the propagation delay is 0.



- Short-circuit current between the rails during the input transitions;
- Due to a finite slope of the input signal that causes NMOS and PMOS to be on simultaneously;
- It lessens increasing the output capacitance.

Let's consider now not an ideal input signal, with abrupt changes, but a real one with finite slopes for the transitions in a time t_{in} to go from 0V to Vdd.

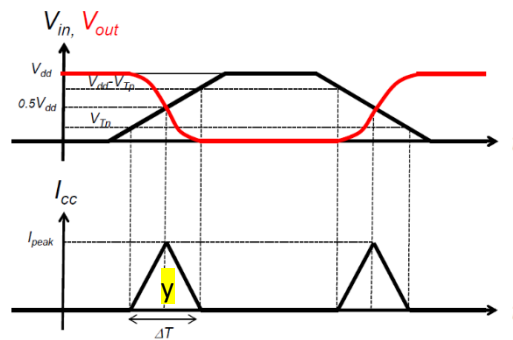
During this t_{in} , once we crossed $V_{dd}/2$, the output is $V_{dd}/2$ because we are in transition region and the two transistors work as current generator, and a current from Vdd to ground comes from Vdd, so we have power consumption.

The current is delivered both during the transition up and transition down. This form of power consumption is also called **short circuit current**, because we have a pull up and pull-down transistors on at the same time.

This current that flows occurs only because we have real signals with finite slope at the input. In the case of ideal signals, the shortcircuit current is 0. In fact, during the rising edge we move immediately from 0V (nMOS on and pMON off) to Vdd, and so immediately we switch off the nMOS and turn on the pMOS. But this transition is immediate, so there is no current because the two transistors are never on at the same time.

In the realistic case, i.e. with a capacitance at the output, this power dissipation contribution is negligible if the slope at the input, so the propagation delay of the input signal, is more or less equal to the propagation delay of the output signal. This happens in the superbuffer, so if we size the superbuffer in order to minimize the propagation delay, this power dissipation contribution is negligible with respect to the dynamic one due to the charge and discharge of the output capacitance.

Cross-conduction current



$$\Delta T = T_{IN} \frac{V_{DD} - V_{tn} - V_{tp}}{V_{DD}} \cong \frac{V_{DD} - 2V_T}{V_{DD}} \cdot T_{IN}$$

Tin is missing

$$I_{peak} = k_n' \left(\frac{W}{L}\right)_n V_{DSATn} \left(\frac{V_{DD}}{2} - V_T - \frac{V_{DSATn}}{2}\right) \left(1 + \lambda \frac{V_{DD}}{2}\right) \cong 39 \mu A \cdot \left(\frac{W}{L}\right)_n$$

The input signal is the black one in the upper plot, whose signal transitions are linear. No output capacitance is present (ideal inverter). The switching threshold is at $V_{dd}/2$.

We want to assess the triangular shape current. As long as the input voltage is 0V, the output voltage is Vdd and the current is 0A. We start to observe a current from Vdd to ground from the V_{in} value such that the $V_{t,n}$, threshold voltage of the nMOS is crossed. The nMOS is in pinchoff, because the V_{gs} is so small that the voltage across the channel is close to 0, V_{dsat} is not overcome. Hence the current depends on the square of the voltage.

Then, as soon as $V_{ov} > V_{dsat}$, the nMOS enter the velocity saturation region. **At the threshold** (of the inverter) **we have the maximum amount of current** (both transistors behave as current generators in velocity saturation region). After this point the pMOS works as a current generator (with decreasing current because we are reducing the V_{gs} of the pMOS) and the nMOS as a Req.

The peak current can be computed with formula x. Everything in this formula is fixed except for the aspect ratio. The base of the triangle representing the peak current is a fraction of T_{in} , because current is

not nihil in the range between the V_t of the transistors. So the peak current depends on the aspect ratio (i.e. on the size of the inverter), this is what counts.

To assess power consumption we need to resort to the energy concept, energy spent by the PS generator. The energy is wasted in correspondence of each transition, low to high or high to low. It doesn't depend on the sign of the transition. We can consider just one transition, because the two are equal.

The energy is the area of the triangle subtended by the I_{peak} current y multiplied by the PS voltage. Technically, $E = V_{DD} \cdot Q$, and Q is the area of the triangle y .

Hence $E = \frac{1}{2} \cdot \Delta T \cdot I_{peak} \cdot V_{DD}$.

Whatever the input transition, if the transition has a finite slope, we have the following (**E_{cc} = cross conduction energy**).

□ In the case of an input transition:

$$E_{cc} = V_{DD} \left(\frac{I_{peak} \Delta T}{2} \right) \cong 33 \mu W \cdot \left(\frac{W}{L} \right)_n \cdot T_{IN}$$

□ Considering an input data rate of f_0 and an equiprobable signal:

Worst case scenario

$$P_{cc} = V_{DD} \left(\frac{I_{peak} \Delta T}{2} \right) \frac{f_0}{2} = \frac{(V_{DD} I_{peak} \Delta T) f_0}{4} \cong 16.5 \mu W \cdot \left(\frac{W}{L} \right)_n \cdot T_{IN} \cdot f_0$$

being $f_0/2$ the average transition rate.

Let's consider an input signal with a bitrate of f_0 (or f_{ck}) and that is equiprobable, so the same probability of having either a 0 or a 1 (random signal). If f_0 is the bitrate, we have a transition rate of $\frac{1}{2}$ of f_0 , because only in the cases $0 \rightarrow 1$ and $1 \rightarrow 0$ out of 4 ($0 \rightarrow 0$ and $1 \rightarrow 1$) I have a transition.

So to assess power consumption we need to multiply the energy and the transition rate. On average, we have a transition $2 \cdot T_{bit}$, with $T_{bit} = 1/f_{ck}$. E.g. if $f_{ck} = 1$ Gbps, $T_{bit} = 1$ ns and we have a transition every 2 ns.

Again, also the **power consumption due to the cross conduction current is proportional to the duration of the input signal, the bitrate and the aspect ratio of transistors.**

It depends on the bitrate because it depends on how many transitions we have in a unit time.

This cross conduction current is proportional to the bitrate as the dynamic power consumption was. Moreover, P_{cc} doesn't depend on the output capacitance at first approximation, with respect to the dynamic one.

As an important remark, this P_{cc} expression was derived without considering the output capacitance, considering an ideal inverter (no delay at all). The next step is to add the capacitance.

Addition of a capacitance at the output

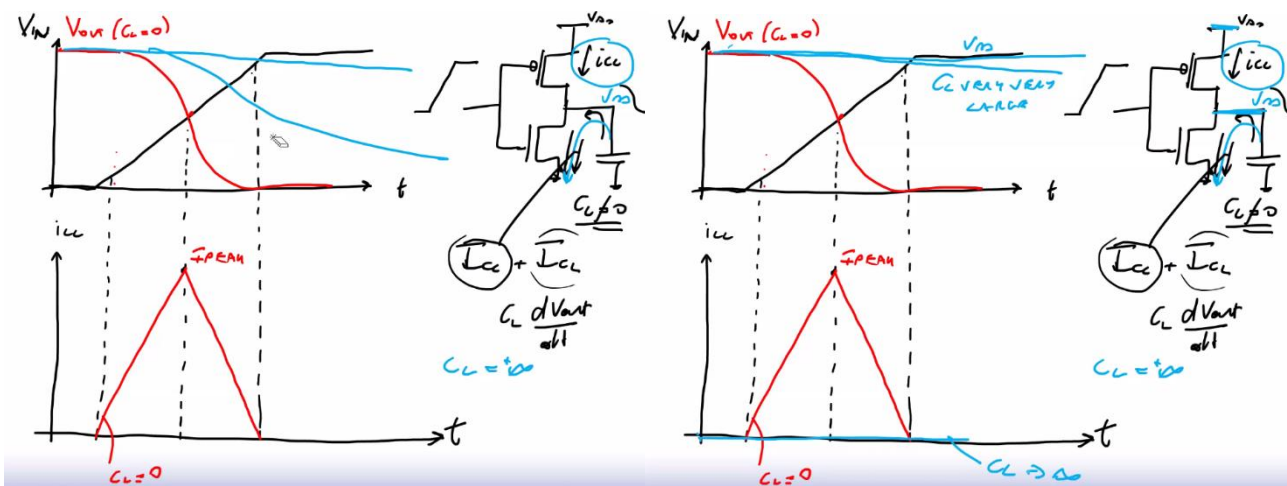
Let's consider just one transition.

The cross conduction current is the one flowing in the pMOS transistor. In fact, let's consider an ideal step in input and a capacitance at the output $\neq 0$. In this case the pMOS is switched off immediately, and the current from V_{DD} to ground 0 if the pMOS is switched off immediately. But we have a current through the nMOS to ground, discharging the output capacitor. **With an abrupt transition there is no cross conduction current.**

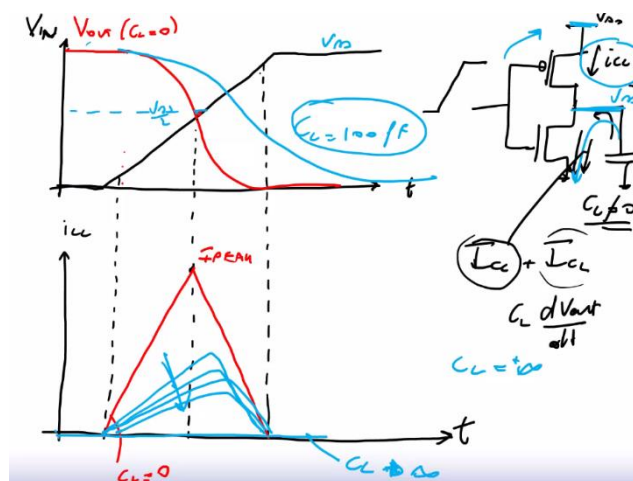
Instead of an abrupt one, let's consider an input with a finite slope. During the transition both transistors are on and we have a current from Vdd that is the cross conduction current. In particular, in the nMOS we have two currents: the cross conduction and the one coming from the capacitance discharge. Instead, in the pMOS flows uniquely the cross conduction current i_{cc} , so to assess i_{cc} we have to consider only the pMOS.

If we add a capacitor, the red triangular shape is modified. Let's consider the case $C_L = \infty$, the characteristic of the inverter becomes flat because we are not able to discharge C_L in a short amount of time. In fact, a $C_L = \infty$ can be assumed as a voltage generator at first approximation. In an intermediate case, knowing that if $C_L = 0$ we have the fastest possible transition, and increasing the capacitance we move 'upwards'.

Let's try to depict the current if $C_L = \infty$. The $V_{ds,p}$ is 0V, so the current in the pMOS is 0 whatever the V_{gs} , so there is no current i_{cc} .



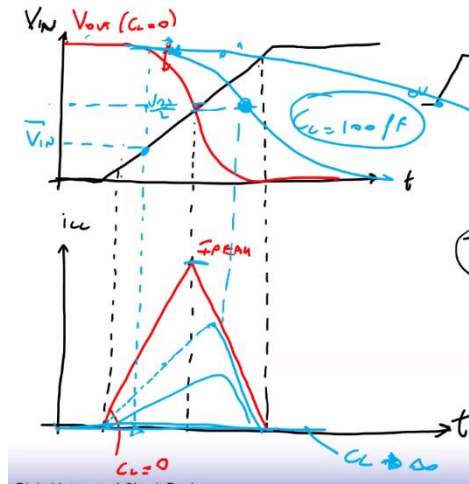
Let's consider now the intermediate case. The i_{cc} has a lower peak value I_{peak} , and the triangle is skewed, the peak doesn't correspond anymore to the crossing of the inverter's threshold of the input signal. The area underneath the curve decreases and the P_{cc} decreases.



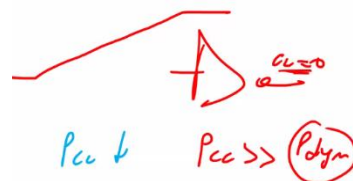
Hence in terms of P_{cc} , considering a situation with no capacitor at the output is the worst case.

When the input ramp starts to increase the operating region of the pMOS transistor is the ohmic one. then the output voltage decreases (discharging of C). Given that V_{ds} is the difference between V_{dd} and

the VTC, in the red case with no output capacitor the V_{ds} is bigger given a certain input voltage with respect to the blue case. Hence the current in the blue case is smaller than in the red one. Then if we go on along the input slope, approximately at $V_{DD}/2$ we have both transistors acting as current generators. In the blue case the current is smaller because the pMOS is operating with a V_{gs} (distance between the input black curve and V_{DD}) smaller than in the red case around $V_{DD}/2$.



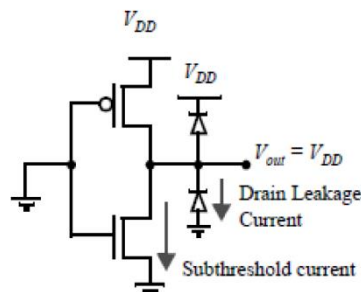
NB: we can have a i_{cc} only in the region where we have a finite slope, not when $V_{in} = 0V$ or $V_{in} = V_{DD}$. The contribution of P_{cc} comes into play when the slope of the input signal is very small, so the area of the i_{cc} is not negligible with respect to the dynamic power consumption, the cross conduction power is overwhelming the dynamic one.



LEAKAGE CURRENT

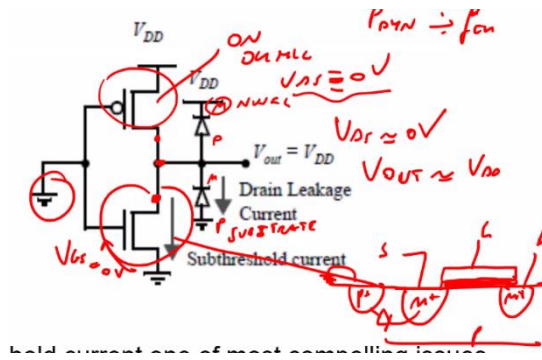
It is a static contribution that is typically negligible. It becomes dominant only if $f_{ck} = 0$, where the P_{cc} and P_{dyn} are 0 because both directly proportional to f_{ck} .

Let's consider an inverter with a steady voltage at the input, e.g. $0V$. nMOS is off and pMOS is ohmic with V_{ds} more or less $0V$, so V_{out} more or less V_{DD} .



Sub-threshold current one of most compelling issues of low-energy circuit design in scaled technologies!

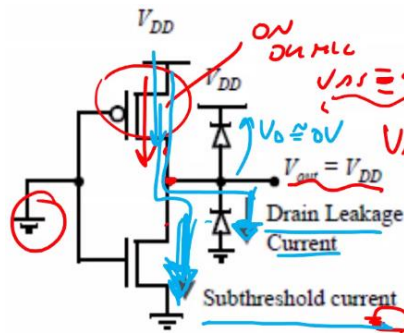
We have to consider that even if $V_{gs,n} = 0V$ there is a leakage current. Moreover, the drain and the bulk contacts form a diode (ground to output node, that is the drain).



In the nMOS we have two currents flowing. The subthreshold current and the leakage current of the reverse biased diode (the one at the bottom). These two currents come from the pMOS, which is in ohmic region.

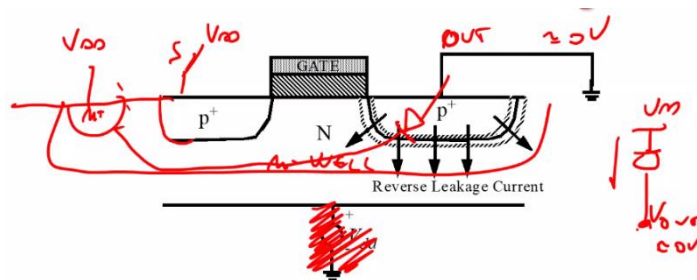
The current in the upper diode is negligible because the voltage across its pn junction is approximately 0V.

We can identify a static contribution: $P_{dc} = V_{dd} \cdot I = V_{dd} \cdot (I_{sub} + I_{leak})$



Let's consider the opposite case with static Vdd in input. The output will be close to 0V, and the leakage current is in the pMOS transistor. Moreover the upper diode has now Vdd of reverse bias, so we have a leakage current. The I_{leak} and I_{sub} sum and flow in the nMOS, which is now ohmic. Because of this, the output voltage is not exactly 0V, but approximately 0V.

REVERSE-BIASED DIODE LEAKAGE



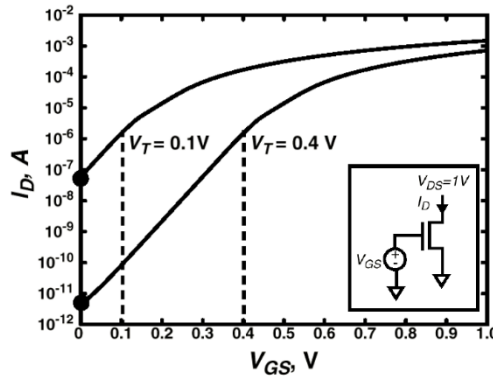
$$I_{DL} = J_S \times A$$

$J_S \cong 100 \text{ pA}/\mu\text{m}^2$ at 25 °C in 0.25- μm CMOS
 J_S doubles for every 9 °C!

In this case we have a current from Vdd to the output node, which is approximately 0V. This leakage current strongly depends on the temperature of the circuit.

SUBTHRESHOLD CURRENT

In the image it is represented the I_D vs V_{GS} characteristic (transfer characteristic of the transistor). It is in linear-log scale to highlight the subthreshold current. Below $V_T = 0.4V$ we have a non-negligible subthreshold current, even if at $V_{GS} = 0V$. For $V_{GS} = 0V$ we have more or less $I_{sub} = 10$ pA.



$$I_{leak}|_{V_{GS}=0} = \mu \left(\frac{W}{L} \right) e^{\frac{-V_T}{nU_{TH}}}$$

$I_{leak} \cong 10$ pA for a NMOS
with $V_T=0.5V$ and $(W/L)=1$

The current of the transistor in subthreshold region depends exponentially on V_T (check Compagnoni's notes), so if V_T is increased, I_{sub} increases. In the scaling down process, V_T decreases, so I_{sub} increases. E.g. if $V_T = 0.1V$, with the same aspect ratio, the current increases up to 100nA. This is due to the exponential dependence.

Also in subthreshold, even for $V_{ds} = 0V$ the current I_{sub} is 0.

PRINCIPLES FOR POWER REDUCTION

- **Reduce gate size**, i.e. physical capacitance. In fact, during operations, only $P_{dyn} = C \cdot V_{dd}^2 \cdot f_0 \rightarrow 1$ counts. To reduce it, it is better to use circuits with minimum area, keeping the size small.
- **Reduce voltage**, at the cost of reduced time performance, i.e. operating frequency.
- **Reduce switching activity**. Once we have a f_{sw} , this frequency depends on the switching activity. We can theoretically reduce it to reduce the power consumption.

IMPACT OF TECHNOLOGY SCALING

It corresponds to the improvement of the CMOS technology, L is reduced together with the oxide thickness, overlap length and so on. on the other side, also the Vdd and Vt are reduced.

Rationales for technology scaling

Make things cheaper:

- Want to sell more functions (transistors) per chip for the same money.
- Build same products cheaper, sell the same part for less money, because we use less silicon.
- Price of a transistor has to be reduced.

But also want to be faster, smaller, and to lower power consumption.

TECHNOLOGY SCALING MODELS

- **Full scaling** (also called constant electrical field): dimensions and voltage scale together by the same factor S. However, if we continue to reduce Vt, leakage current becomes a problem because we are no more able to switch off the transistor. If we scale down the oxide thickness in the channel and we keep constant the voltage between gate and bulk, the electrical field across the oxide increases, with drastic effects on the reliability of the circuit.
- **Fixed voltage scaling**: most commonly used model so far. Only the dimensions scale, voltages remain constant.
- **General scaling**: today's approach. Voltages and dimensions scale with different factors. To keep the maximum overdrive voltage constant, better not to use the same scaling.

Scaling relationship in full scaling

Parameter	Relationship	Full Scaling
W, L, t_{ox}		1/S
V_{DD} , V_T		1/S
N_{sub}		S
area	WL	1/S ²
C_g	$\epsilon_{ox}WL/t_{ox}$	1/S
I_{DSAT}	$C'_{ox}(W/L)V_{DSAT}V_{DD}$	1/S
R_{eq}	V_{DD}/I_{DSAT}	1
delay (τ)	$R_{eq}C_g$	1/S
frequency	1/ τ	S
energy	$C(V_{DD})^2$	1/S ³
power	$C(V_{DD})^2f$	1/S ²

Not only C_g , but also the C_{int} capacitance is reduced by a factor 1/S.

As for I_{dsat} (that is the maximum current), it depends on C'_{ox} , which is ϵ_{ox}/t_{ox} and V_{dsat} , which is decreasing with the technology scaling. Also V_{dd} is scaled down. In the end I_{dsat} decreases, which is not so good because it slows down the speed of the circuit. However, in digital electronics it's R_{eq} that matters; in fact, I_{dsat} reduces, but also the voltage does, and since R_{eq} is voltage over current **→ R_{eq} is almost the same regardless the technology we are using.**

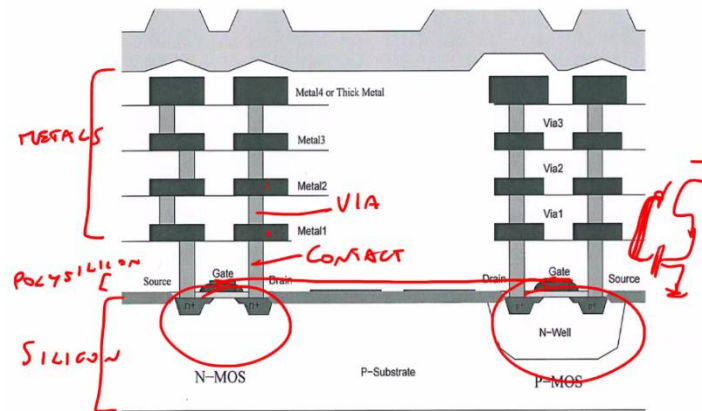
As for the delay, it is $R_{eq} * C_g$, so overall it decreases by a factor 1/S due to the decrease in C_g . **The circuit is faster just because we are using a smaller capacitance**, the benefit is in terms of capacitance.

As for the energy per event, C and Vdd are both reduced, so a scaling with 1/S³. As for the power, we have a dependance on the pull-up frequency, which can improve with a factor S. Hence power

consumption is reduced by a factor $1/S^2$. If we compare instead two circuits with the same frequency f , the improvement in technology scaling is $1/S^3$ because f is the same in both.

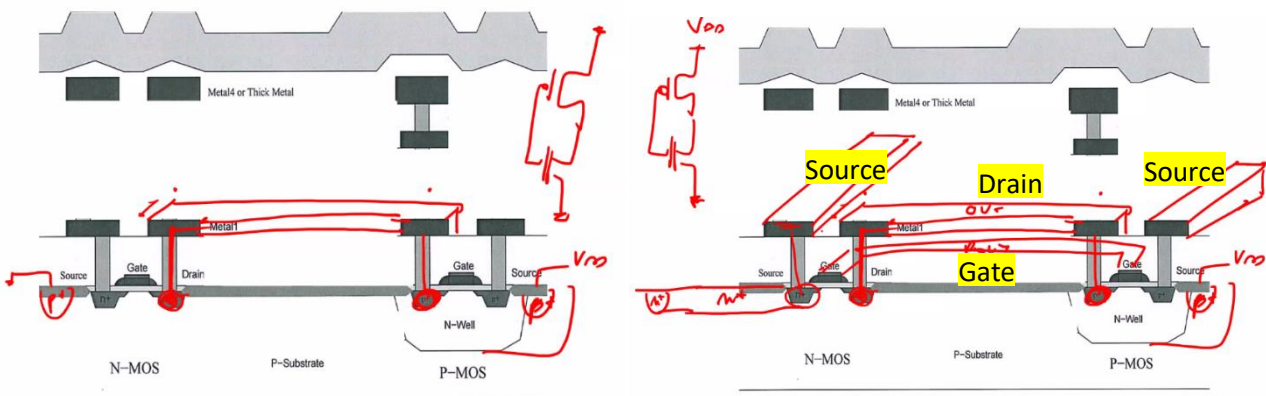
WIRES IN ICs

INTERCONNECTS IN CHIPS



The white region is SiO₂, so different layers are built inside the silicon dioxide. Plugs that allow to connect two different layers of metals are called via, while from the polysilicon surface to first layer of metal is called contact.

The cross-section can be different, but still the drains of the inverters are to be connected. The same has to be done with the gates, and then the two source contacts separated. To be honest, also the n+ diffusion region can be used to perform a connection at silicon level.

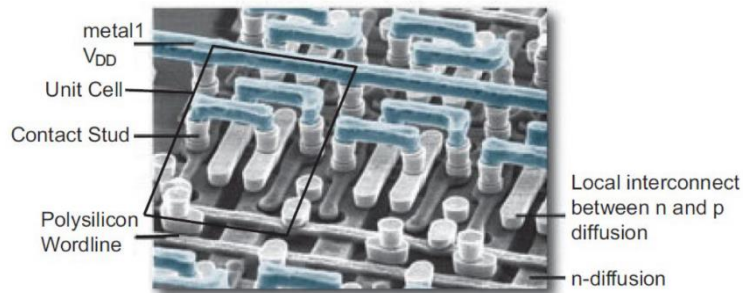


Hence an interconnection can be done with n+ diffusion region, polysilicon or metal.

A designer can choose, typically, the type of interconnection, that usually is made out of metal (copper or aluminum) because it has a much better conductivity with respect to the other two. This is a procedure at layout level, not at a design level.

As for the wire connection, the vertical dimensions are fixed by the technology (thickness of the wire and of the dielectric), and we can change the width and the length (planar dimension) plus the separation between two nearby wires. To be honest, also the level of metal with which the connection is implemented can be changed (metal1, metal2, metal3, ecc.). This last choice has an effect in terms of capacitance.

PHYSICAL VIEW



Partially completed 6-transistor SRAM array using local interconnect

In the image it is represented an SRAM. Blue parts are metal wires that perform interconnections. The Wordline is an activation signal to activate the cell, and it is an interconnection implemented with polysilicon.

Furthermore, the ground in the SRAM is brought to all the cells with an n diffusion interconnection line. This is not the best interconnection, because it is a semiconductor and its resistance is quite high with respect to metal. Moreover, an interconnection done with metal at higher level in the chip has a smaller capacitance than one closer to the substrate, because the distance is smaller, and the thickness of the dielectric is smaller → bigger parasitic capacitance.

IMPACT OF INTERCONNECT PARASITICS

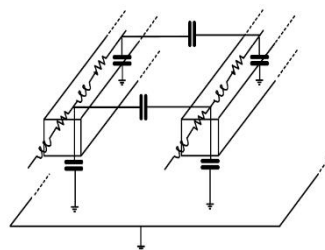
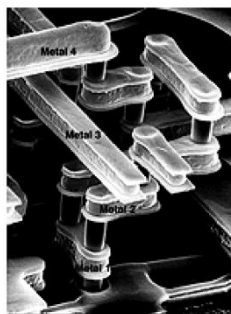
Classes of parasitics:

- Resistive
- Capacitive
- Inductive

Interconnect parasitics:

- Affect propagation delay.
- Reduce reliability, because if we consider two wires that connect two circuits, between them we have dielectric and so a parasitic capacitive coupling.
- Increase power consumption.

In the image layers above the silicon are represented. The matrix in the middle is SiO₂, so we have parasitic capacitances and resistances and inductances of the wires. Capacitances are between the wires and all the wires to the substrate ground.



Simplifications

- Inductance can be neglected if the wire resistance is large or if the rise/fall time of the input signal is large.
- When the wire is short and the equivalent resistance of the driver is large, the wire resistance can be neglected.
- When the separation between nearby wires is large or when the wires run in parallel for a short distance, the inter-wires capacitance can be neglected.

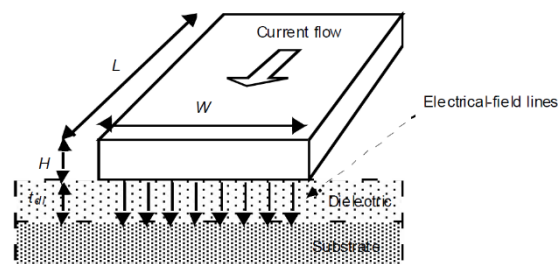
The inductance of the wire can be neglected because the impedance due to the inductance is negligible with respect to the resistance.

As a second approximation, when the wire is short or the equivalent resistance of the driver is large with respect to the one of the wire, the resistive effect of the wire can be neglected. Exceptions are if the wire is very long, e.g. in the case of the clock line.

Even if we neglect R and L, we still have a lot of parasitic capacitances. If the separation d between two wires is large, we can neglect the interwire capacitance. The same if two wires are close but run in parallel for a small distance.

CAPACITANCE

Let's consider a wire like in the image. The designer can change the width and the length.



$$C_{pp} = \frac{\epsilon_{di}}{t_{di}} WL$$

The substrate acts as a ground plane and let H be the height of the wire, which is not negligible. The capacitance from the wire to ground is called **parallel plate contribution**, and it is proportional to the dielectric permittivity and directly proportional to the overlap area. Given the technology, H is a fixed parameter. Also t_{di} is a fixed parameter.

Moreover, the side wall of the wire is not negligible and it may happen that we have a very small width but a big height, so the side wall has a larger area with respect to the bottom area, so we cannot use the parallel plate capacitor model, we need to add the contribution of the side wall. The side walls give a contribution called **fringing contribution**.

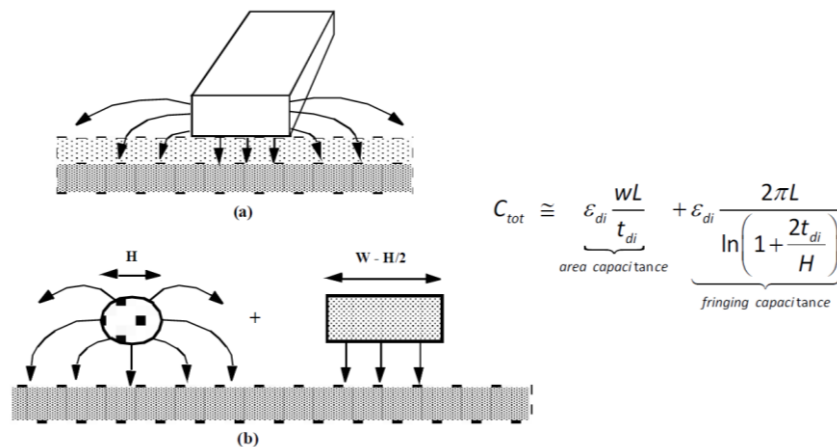
Permittivity

Why do we use SiO₂ instead for instance of silicon nitride? Because the parasitic capacitance would be higher due to the dielectric permittivity.

Material	ϵ_r
Free space	1
Aerogels	~1.5
Polyimides (organic)	3-4
Silicon dioxide	3.9
Glass-epoxy (PC board)	5
Silicon Nitride (Si_3N_4)	7.5
Alumina (package)	9.5
Silicon	11.7

FRINGING CAPACITANCE

The parallelepiped can be split in a cylinder and another parallelepiped where just the bottom plate is considered. So we have a parallel plate contribution and a second contribution of the fringing capacitance. This contribution is a strong function of L and mildly dependently on H (diameter) and t_{di} . So it reduces with a mild function of t_{di} .

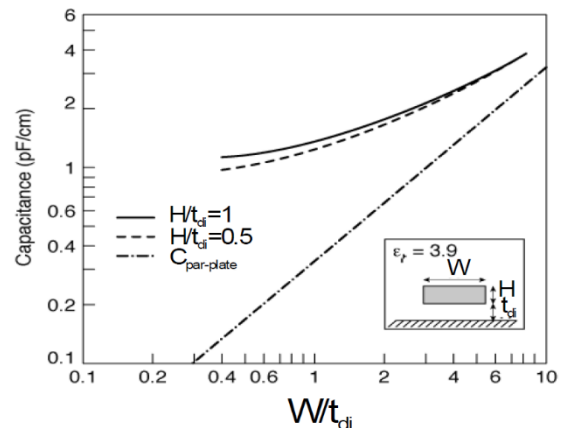


If we consider the specific capacitance per unit length C' , it is C_{tot}/L .

$$C' = \frac{C_{tot}}{L} = \epsilon_{di} \left(\frac{W}{t_{di}} \right) + \frac{2\pi \epsilon_{di}}{\ln\left(1 + \frac{2t_{di}}{H}\right)}$$

Aside from the length, the designer can change W and t_{di} (moving to an upper metal layer). The real dependance is on the first term, because we have W/t_{di} . So to reduce C' we can use the minimum W possible allowed by the technology, and increase the oxide thickness, implementing the wire not in the bottom layer, but in a higher one.

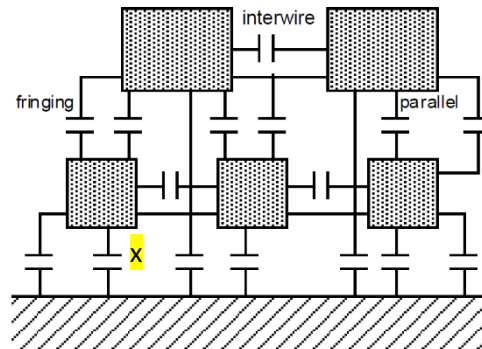
In the next image C' is represented as a function of W/t_{di} . The parallel plate contribution is the straight line. For $W/t_{di} = 1$, the parallel plate contribution is ϵ_{di} , which is $1/3 \text{ pF/cm}$. Then the overall capacitance is represented by the other two curves, including the fringing contribution. If we increase H and t_{di} , the curve shifts down, but the dependance is mild for large W/t_{di} , for smaller value of this ratio, C' is dominated by the fringing contribution.



So if we implement a wire we want not to work with a large W/t_{di} ratio, but where this ratio is small. This means that we have to decrease W and increase t_{di} using a higher level of meta (metal3, metal4 and so on).

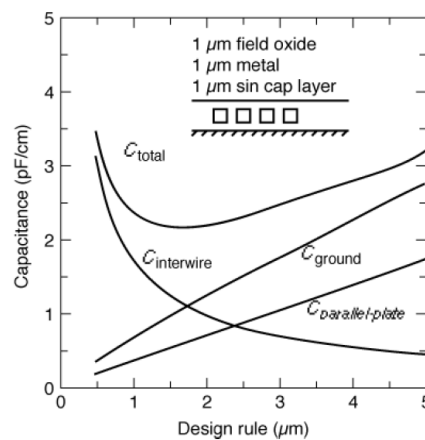
INTERWIRE CAPACITANCE

If I consider two wires that are running in parallel on the same layer, we have an interwire capacitance. In the image below we have the cross-section. Gray parts are the wires. X is bottom plate contribution.



If we have a small overlap between the wires I have also a parallel plate contribution. I want to have a C_{wire} contribution summing all the contributions. The thing we can do to solve the problem is to decouple all the elements, and consider C_{wire} as the sum of independent contributions. If we do so, we are performing an overestimation.

Impact of interwire capacitance



Typically the dominant contribution is the capacitance of wires running in parallel on the same layer, that is the interwire capacitance. Let's consider 4 wires at the same height with respect to the silicon substrate. Above them we have a large metal that acts as a ground plane; the distance between the wires and the top and bottom layers is 1μm.

In the graph we are representing C' vs the CMOS technology node (we are scaling L , W and d). I'm plotting the total C' considering a wire, e.g. the one in the middle.

The parallel plate contribution decreases as the technology decreases, because W is decreasing and so the bottom area is decreasing for the same length.

Instead, the interwire contribution increases as an hyperbola because I'm decreasing d , and wires are closer and closer. This suggests that if we have a critical wire and we want to minimize its C_{wire} , and the length cannot be changed, we can reduce W or push the wire away from the other wires (**isolating the critical wire is the most important thing to do**), implementing it in the top layer of metal.

Once we have a technology, we can assess the fringing and parallel plate contributions between two wires according to the following table.

	Field	Active	Poly	Al1	Al2	Al3	Al4
Poly	88						
	54						
Al1	30	41	57				
	40	47	54				
Al2	13	15	17	36			
	25	27	29	45			
Al3	8.9	9.4	10	15	41		
	18	19	20	27	49		
Al4	6.5	6.8	7	8.9	15	35	
	14	15	15	18	27	45	
Al5	5.2	5.4	5.4	6.6	9.1	14	38
	12	12	12	14	19	27	52

Al in the column indicates the level of metal in which we are implementing the wire, while the row indicates the level we want to assess the capacitance with. In the white boxes we have the parallel plate capacitance per unit area (aF/um²), while in the grey ones we have the fringing contribution per unit length (aF/um). Field is the substrate, active is the n+ or p+ region.

For instance, if we consider a metal2, it is row Al2. Why polysilicon cannot be build above the n+ and p+ regions? If we implement polysilicon, below it we cannot implement n+ and p+ regions because it acts as a barrier, and the polysilicon is always implemented before the diffusion region, according to the self-aligned gate process.

We should avoid the connection with polysilicon for sure because the resistivity is very large, but also because capacitances related to the polysilicon are very high compared to the metal, if we look at the table → better to use metal to convey a critical signa.

Interwire capacitance assessment

Let's consider two wires on the same layer placed at the minimum distance allowed by the technology we are using.

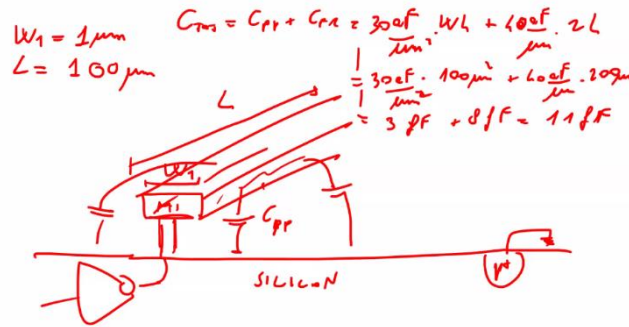
Table 4.3 Interwire capacitance per unit wire length for different interconnect layers of typical 0.25µm CMOS process. The capacitances are expressed in aF/µm, and are for minimally-spaced wires.

Layer	Poly	Al1	Al2	Al3	Al4	Al5
Capacitance	40	95	85	85	85	115

The difference in terms of numbers is due to the fact that polysilicon has a smaller H, so two contiguous wires have small sidewalls and so less capacitance. The upper level of metal has a larger contribution because typically the upper level has the largest height, almost a factor 2 with respect to the other layers.

Let's suppose that we want to bring the signal in output of the inverter far away in another portion of the chip. Let's assess the capacitance of the wire with respect to the silicon layer. Since we are in metal1 and we are implementing over the substrate, we have to consider the field column and the metal1 row in the table.

I have the Cpp contribution and two fringing contributions.



So we have 11fF to be added at the output of the inverter.

Let's now add a wire in metal2 with a width of 0.5um that runs above the metal1 wire for a length of 50um. Let's also add another wire in metal1 at the minimum distance from the original wire that faces the initial wire for the whole length.

We can assess the interwire contributions of the metal1 wires.

$$C_{\text{int}} = \frac{95\text{af}}{\mu\text{m}} \cdot 100\mu\text{m} = 9.5\text{fF}$$

As for the one between metal1 and metal2, considering the overlapping area (100um is 2*50um):

$$C_{12} = \frac{36\text{af}}{\mu\text{m}^2} \cdot (0.5 \times 50)\mu\text{m}^2$$

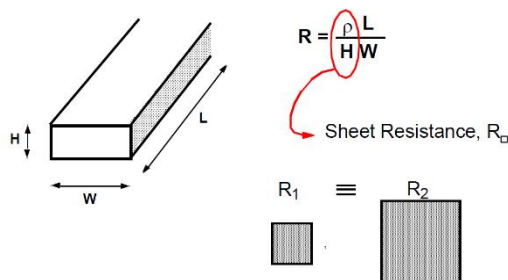
$$+ \frac{15\text{af}}{\mu\text{m}} \cdot 100\mu\text{m} =$$

$$= 0.9\text{fF} + 1.5\text{fF} =$$

In the end C_{tot} is 25.4fF, summing the three contributions.

RESISTANCE

WIRE RESISTANCE



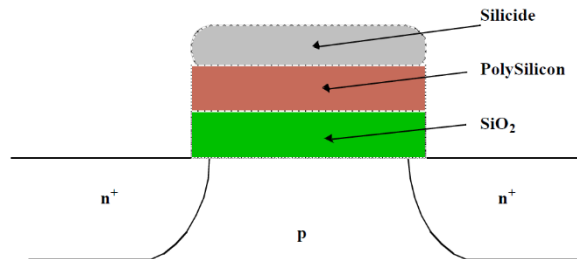
In the R expression the designer can change L and W. The sheet resistance is the resistance of a wire with a length equal to the width, and once we have the technology, the sheet resistance ρ_0/H is fixed.

There are different materials with which the interconnection can be made. Typically aluminum is used because it's very compatible with silicon dioxide, or copper in modern technology or for upper levels of metal.

Material	ρ ($\Omega\text{-m}$)
Silver (Ag)	1.6×10^{-8}
Copper (Cu)	1.7×10^{-8}
Gold (Au)	2.2×10^{-8}
Aluminum (Al)	2.7×10^{-8}
Tungsten (W)	5.5×10^{-8}

Polycide gate mosfet

Polysilicon alone is never used, also the gate of a transistor is never just implemented with PolyS, because it is very resistive. To increase its conductivity, typically it is covered with another material called silicides. Then the circuit is heated, and this material enters in the PolyS creating a compound called **polycide**. We can improve the conductivity by a factor of 10 with respect to just polysilicon alone.



Silicides: WSi_2 , $TiSi_2$, $PtSi_2$ and $TaSi$
 Conductivity 8-10 times better than poly

Sheet resistance

In the table we have the materials we can use to implement an interconnection and their sheet resistance.

Material	Sheet Resistance (Ω/\square)
n- or p-well diffusion	1000 – 1500
n^+ , p^+ diffusion	50 – 150
n^+ , p^+ diffusion with silicide	3 – 5
n^+ , p^+ polysilicon	150 – 200
n^+ , p^+ polysilicon with silicide	4 – 5
Aluminum	0.05 – 0.1

INDUCTANCE

At first order approximation it's negligible. Let's consider a wire at t_{di} from the ground plane, thickness H and width W, with length L. Let's assess its inductance, with W and H negligible with respect to t_{di} , so that it is like having a line.

The inductance per unit length is a mild function of the physical parameters of the wire, so it is more or less a constant. At first order approximation, a wire, whatever the material, has an inductance per length of 0.4 pH/ μm .

- Assuming $W < t_{di}$ and a negligible thickness H of the wire:

$$\frac{\text{inductance}}{L} = l = \frac{\mu_0}{2\pi} \ln \left(\frac{8t_{di}}{W} + \frac{W}{4t_{di}} \right)$$

- It turns out that the inductance is a mild function of the geometry:

$$l \cong 0.4 \frac{pH}{\mu m}$$

- Considering $r = 0.075 \Omega/\mu m$, it's negligible for frequency smaller than:

$$2\pi fl = r \Rightarrow f = \frac{r}{2\pi l} \cong 30GHz$$

If for the wire we consider a sheet resistance of 0.075 Ohm/square and a $W = 1\mu m$.

$$R = R_{sheet} * L/W.$$

The resistance per unit length will be $r = 0.075 \text{ Ohm}/\mu m$.

Now I want to compare the two values of l and r . The absolute value of the inductance impedance is: $|Z_l| = 2\pi f l L$, where L is the length. The resistance is $R_w = r L$.

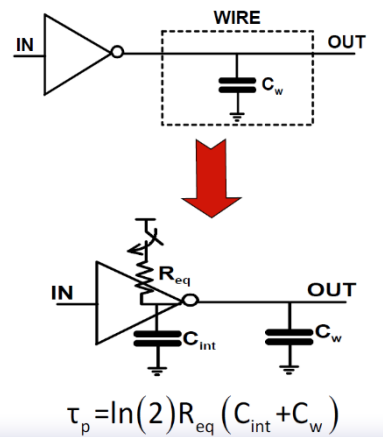
The frequency at which the impedance of the inductance is the same of the resistance is 30GHz. For smaller frequencies the resistance dominates, for higher ones the inductance dominates. Since in general the frequency of operation of circuits is smaller than 30GHz, the resistance dominates at the frequency of interest.

INTERCONNECT MODELING

There are some models we can use for the wire. The goal is to consider the wire as a circuit made of passive elements.

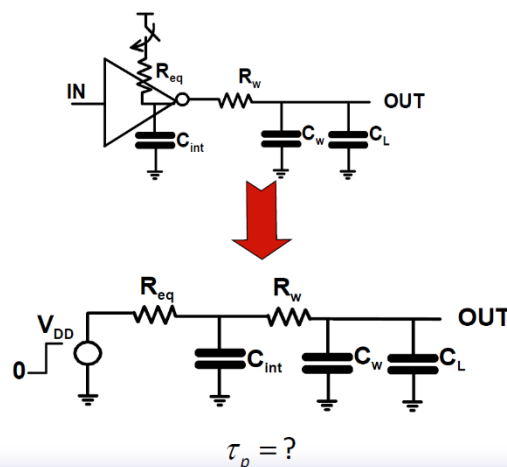
Lumped C model

Once the equivalent resistance (R_{eq}) of the system that drives the wire is much larger than the resistance of the wire, R_w resistance of the wire can be neglected, and the wire is just a capacitor C_w towards ground (of course given by the sum of all the previously seen contributions, C_{pp} , C_{fr} and a contribution that is not toward ground, but with a resistance to ground, that is the interwire contribution. The resistance that puts the interwire capacitance to ground is the static r_{on} resistance of the nMOS transistor in ohmic region, it is not R_{eq}) \rightarrow lumped C model.



The delay can be written with the classical formula.

Lumped RC model



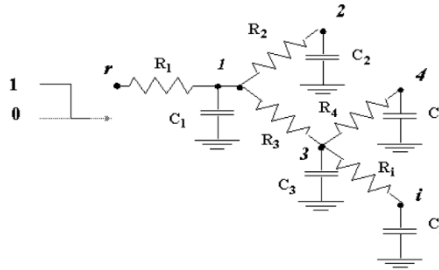
Now R_w is comparable with R_{eq} .

I have to model the wire with a RC network. R_{eq} and C_{int} represent the inverter, R_w and C_w the wire and C_L eventually the C_g of another inverter. The network has 2 poles because the C_L is in parallel with C_w (only two independent capacitors).

To assess the delay we can use the Elmore theorem.

Elmore theorem

It allows to assess the delay of a network with more than one pole. Let's suppose to have a circuit like in the following image and we want to assess the delay in a certain point of the circuit, e.g. from point r to point i.



$$R_{ik} = \sum R_j \Rightarrow (R_j \in [\text{path}(s \rightarrow i) \cap \text{path}(s \rightarrow k)])$$

$$\tau_{Di} = \sum_{k=1}^N C_k R_{ik}$$

To apply this theorem the circuit has to fulfill three conditions:

1. One single input
2. No feedback
3. All capacitors are grounded. The capacitor must be all towards a fixed point.

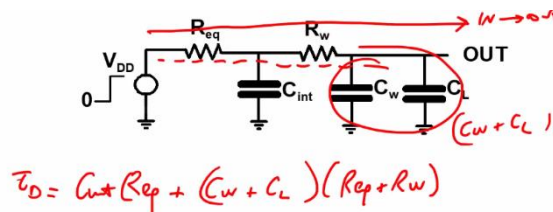
The Elmore theorem states that the delay can be assessed considering the shared path resistance for every capacitor of the circuit. The delay of a circuit involves the assessment of a time constant for each capacitor, and for each capacitor I have to consider its shared path resistance. Then all the time constants are summed.

The shared path resistance is the resistance in common between the path from the input to the output and the path from the input to the considered capacitor.

E.g., for C1 it's R1 because the first one is R1 + R3 + Ri, and the second is R1, and so the one in common is R1. For C2 it's again R1. For C3 it's R1 + R3. For C4 is R1 + R3. Ci is R1 + R3 + Ri.

Then I multiply the respective capacitance for the shared path resistance and I sum the time constants.

Now we want to apply this theorem to the lumped RC model.



We have the capacitance Cint and the sum of Cw and Cl, so in the Elmore computations we have just two contributions.

Let's rearrange highlighting the Req resistance. In this way it's like if the current to charge Cint, Cw and Cl passes from Req, while Rw is responsible of the current contribution to Cw and Cl.

$$\begin{aligned} \tau_D &= C_w * (R_{eq} + (C_w + C_L)) * (R_{eq} + R_w) \\ &= R_{eq} (C_w + C_w + C_L) + R_w (C_w + C_L) \end{aligned}$$

Let's express it in a third way. Let's consider the load capacitance equal to the gate capacitance, that is equal to C_{int} . The output is to the input of the load inverter, that is the inverter after the wire.

$$\tau_D = C_{int}(R_{eq} + (R_w + C_L)(R_{eq} + R_w)) \quad \tau_p = \ln(2) \tau_D$$

$$= R_{eq}(C_{int} + C_w + C_g) + R_w(C_w + C_g) \quad C_g = C_{int}$$

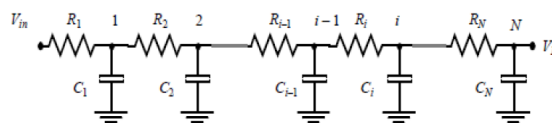
$$\rightarrow 2R_{eq}C_{int} + R_{eq}C_w + R_wC_w + R_wC_{int}$$

$\ln(2) \cdot 2 \cdot R_{eq} \cdot C_{int}$ is $2 \cdot \tau_{p0}$, the delay of the inverter. $R_w \cdot C_w$, apart from $\ln(2)$, it is the delay of the wire T_w . $R_{eq} \cdot C_w$ is the delay of a resistance that has to charge the capacitance of the wire, apart from the term $\ln(2)$, so we can call this term T_{iw} . Then the last term is $R_w \cdot C_{int}$ is T_{wi} .

I want to improve the delay of the wire T_w . Let's suppose to have a wire driven by an inverter and to receive the signal at the far end of the wire. We need to compute the delay of the wire. We want to improve the T_w expression.

The Elmore delay applied to a wire

Let's split the resistance of the wire R_w in N equal pieces $R = R_w/N$. The same for the capacitance of the wire C_w , divided in $C = C_w/N$ capacitances.



□ Assuming equal all the resistances and all the capacitances:

$$\tau_p = \ln(2) [CR + 2CR + 3CR + \dots + NCR] = \ln(2) \frac{N(N+1)}{2} CR$$

□ If $R_w = NR = rL$ and $C_w = NC = cL$ (r and c specific cap. and res., respectively):

$$\tau_p = \ln(2) \frac{r c L^2}{2} = \ln(2) \frac{R_w C_w}{2}$$

I want to assess the delay from V_{in} to V_n . For the first contribution the shared path resistance is R , for the second is $2 \cdot R$, and so on. The overall resistance path is $N \cdot R$. Let's now factor out C and R at x .

If N is much larger, $N(N+1)$ is more or less N^2 .

Then I replug R_w and C_w in the expression $\rightarrow \ln(2) \cdot R_w \cdot C_w / 2$.

It seems that the delay is the same estimated with the lumped model for the wire ($\ln(2) \cdot R_w \cdot C_w$), but with a factor $/2$ in addition. So the lumped model performs like an overestimation, assuming this one just found is the correct one.

If we express R_w as r , specific resistance per unit length, times the length of the wire, and the same for C_w with c . We get the expression $y \rightarrow$ the propagation delay depends quadratically on the length of the wire.

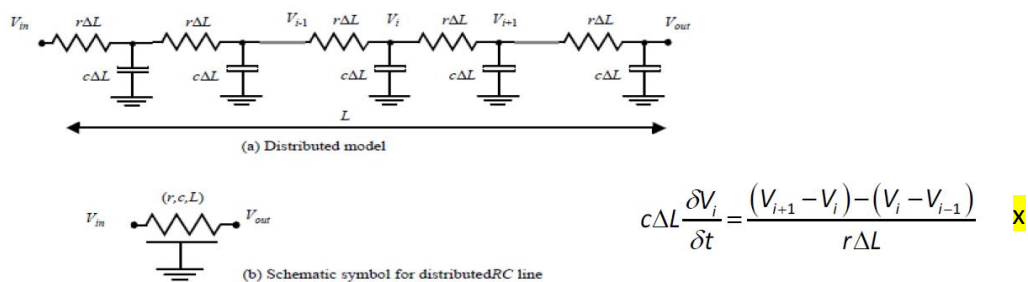
If ideally we would neglect the resistance, using the model in theory the delay is 0. But what comes into play is the speed of light, because the signal travels with the speed of light. If the length is increased, we are doubling the propagation delay linearly. If we remove the resistance, the inductive effect comes into play, e.g. at PCB level where the wire resistance is negligible.

Few considerations

- The delay of a line increases with the square of the length, because both R_w and C_w are proportional to the length L ;
- the lumped RC model overestimates the delay since we get a time constant $R_w C_w$, whereas with this more accurate model that resembles a distributed line, we get $R_w C_w / 2$.

THE DISTRIBUTED MODEL

We want to assess the delay from the input to the output. we divide the resistance in infinite pieces and each piece has a length of ΔL , with ΔL that tends to 0. x is the diffusion equation that is a differential equation with which we can assess the delay and write the expression of the voltage in every point of the wire as a function of time.



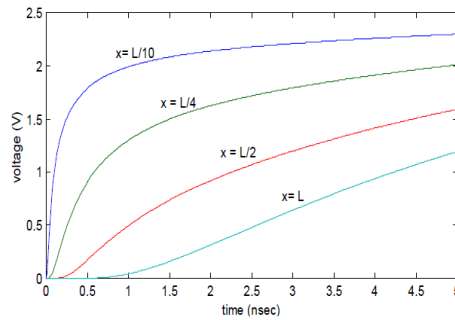
$$\begin{cases}
 V_{out}(t) = 2 \operatorname{erf} \left(\sqrt{\frac{R_w C_w}{4t}} \right) & \operatorname{erf}(x) := \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt & t \ll R_w C_w \\
 V_{out}(t) = 1 - 1.366 e^{-\frac{2.5359}{R_w C_w} t} + 0.366 e^{-\frac{9.4641}{R_w C_w} t} & & t \gg R_w C_w
 \end{cases}$$

The one in the red box is the solution of the expression.

STEP RESPONSE

L is the length of the wire and on the y axis we have the output voltage. The input step is from 0 to 2.5V. I'm interested in the point where I cut at half V_{dd} and the delay is $0.38 * R_w * C_w$ according to the table. This is the value of time after which we reach $V_{dd}/2$, and the distributed model result is obtained solving the equation x of the previous image. With the rough distributed model $\ln(2) * R_w * C_w / 2$ I would get 0.345, so it is a good approximation.

The lumped RC model result gives a higher value, an overestimation.



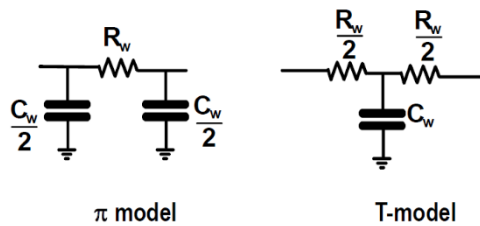
Voltage Range	Lumped RC	Distributed
0 → 50%	0.69 $R_W C_W$	0.38 $R_W C_W$
0 → 63%	$R_W C_W$	0.5 $R_W C_W$
10 → 90%	2.2 $R_W C_W$	0.9 $R_W C_W$
0 → 90%	2.3 $R_W C_W$	$R_W C_W$

How can we cope with the overestimation of the lumped model?

DISTRIBUTED-LIKE MODELS

We can use two distributed-like models that give the same result.

In the pi-model the capacitance is split in two elements, in the T-model it is the resistance that is split in two terms.



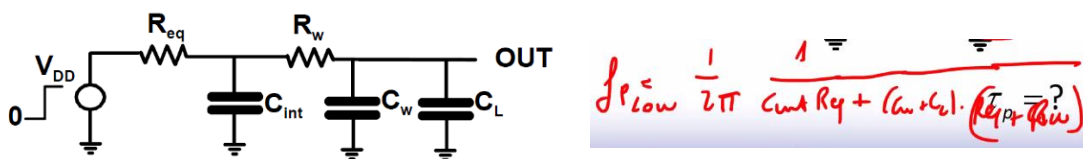
$$\tau_p = \ln(2) \frac{R_w C_w}{2}$$

The delay of the pi-model is $\ln(2)$ times the Elmore time constant, so since I have two capacitances I have to consider two terms. $\ln(2) * [C_w/2 * 0 + C_w/2 * R_w] = \ln(2) * R_w * C_w/2$.

The same reasoning can be done for the T-model, where the shared path resistance is $R_w/2$.

Which model to use then? If the resistance is negligible we don't care about the type of model, the wire is just a capacitance, we can neglect the resistance. If the resistance is not negligible, if the delay of the wire is not negligible with respect to the remaining term, better to use the T-model or pi-model, if instead it is negligible we can use any model because all the other terms in the delay formula are the same.

Moreover, if we have a RC network, poles can be computed with the Middlebrook theorem.

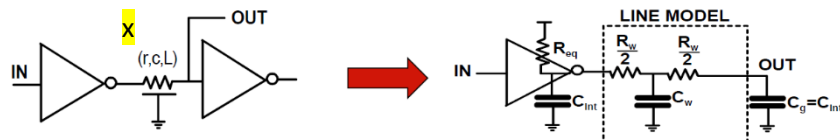


The response to the step function will be an exponential increase (if the two poles are very split apart) and the $V_{dd}/2$ is crossed after a τ that is the same computed with the Elmore delay \rightarrow the Elmore delay is a way to express the dominant time constant of a circuit.

Applying the pi or T model

X is the symbol to illustrate the effect of the distributed resistance and capacitance. Typically, $r = 0.1$ Ohm/ μm and $c = 100$ aF/ μm for a Metal 1 in 0.25 μm Intel CMOS process.

The wire has been substituted with the T-model, and the symbol x identifies the distributed effect of the components.



$$\tau_p = \ln(2) \left[C_{int} (R_{eq}) + C_w \left(R_{eq} + \frac{R_w}{2} \right) + C_{int} (R_{eq} + R_w) \right] =$$

$$= \ln(2) \left[C_{int} (2R_{eq} + rL) + R_{eq} cL + \frac{crL^2}{2} \right]$$

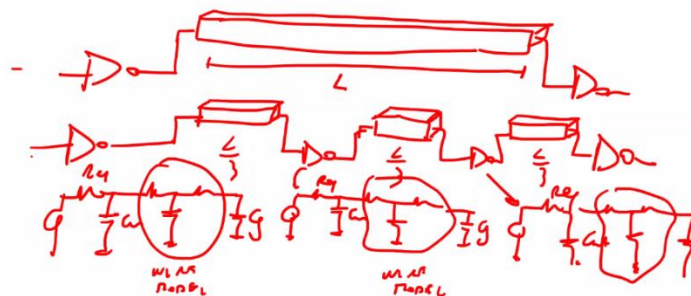
In case of long line the quadratic dependence on L dominates!

Since I have 3 capacitances, I have 3 contributions in the Elmore formula. $C_g = C_{int}$. Again, $R_w = r \cdot L$ and $C_w = c \cdot L$.

For the final expression we notice that we have constant terms in the delay τ_p plus a term that depends on the length ($C_{int} \cdot r \cdot L$), and a term proportional to the L^2 . The term that depends on L^2 is the delay of the wire. This term is typically negligible, but if we consider long wires (mm or cm), it is no more negligible and we have to consider it.

Which is the length for which the L^2 term is non negligible? We have to compare term y with $2 \cdot \tau_{p0}$, and the result is 2.5 mm in our technology. At this length, $T_{inv} = T_w$.

Let's suppose to have a very long wire. How can we reduce the dependence on L^2 ? If we are able to split the wire into pieces and decouple them, we would get something like below. Between each piece I put an inverter, and then I can replace the wires and inverters with their models, e.g. the wire with its T-model. Of course I need to split with C_g and voltage generator to model the inverters.



I want to minimize the delay up to the input of the last inverter input. I have 3 pieces of wires.

If we are splitting the wire in pieces, we are like cutting the square dependence, the sum of the three wires delays is smaller than the original delay because it is like:

$$10^2 = 100$$

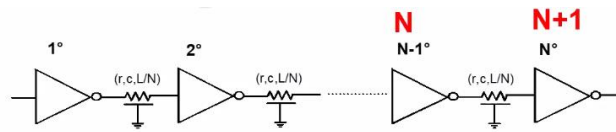
$$(5)^2 + (5)^2 = 25 + 25 = 50$$

$$(2.5)^2 + (2.5)^2 + (2.5)^2 = 25$$

For instance the delay of the wires now is: $\ln(2)/2 * C_w/3 * R_w/3$. If I sum the three delays, i.e. I multiply by 3, I have a factor of 3 of improvement in the delay, just for the wire. Of course, the drawback is that we are adding the inverter delay with respect to the original problem.

Maybe there is an optimal number of pieces in which we can split the wire to optimize the delay.

INSERTING BUFFERS IN A WIRE



$$\tau_p = \ln(2)N \left[C_{int} (R_{eq}) + \frac{cL}{N} \left(R_{eq} + \frac{rL}{2N} \right) + C_{int} \left(R_{eq} + \frac{rL}{N} \right) \right] \quad \text{X}$$



$$\frac{d\tau_p}{dN} = 2C_{int} R_{eq} - \frac{crL^2}{2N^2} = 0 \Rightarrow N = L \sqrt{\frac{cr}{4C_{int} R_{eq}}} \quad \text{Y}$$

It's worth inserting a buffer if: $L \geq \sqrt{\frac{16C_{int} R_{eq}}{cr}} = \sqrt{\frac{23\tau_{p0}}{cr}}$

Let's suppose that all the inverters have the same size S. The different parts are inverter-wire-gate cap of inverter, then inverter-wire-gate cap of the inverter and so on. Hence I have N equal parts.

The difference is that in the original problem the wire length was L, now it is L/N. Aside from this, each piece is equal to the original problem.

I want to minimize the delay from the input point to the input of the load final inverter. The propagation delay is the single delay of each piece multiplied by N. The Elmore delay constant is the same for each piece, that is the same of the whole original system with length L, but with the difference of L/N.

We can rearrange expression x.

$$= \ln(2) N \left[T_{inv} + T_{iw} + T_{wg} + T_w \right]$$

$$= \ln(2) N \left[\underbrace{2C_{int} R_{eq}}_{T_{inv}} + \underbrace{\frac{C_w R_{eq}}{N}}_{T_{iw}} + \underbrace{\frac{R_w C_{wg}}{N}}_{T_{wg}} + \underbrace{\frac{R_w C_w}{2N^2}}_{T_w} \right]$$

Let's now multiply by N the terms

$$= \ln(2) \left[2N R_{eq} C_{int} + C_w R_{eq} + R_w C_{wg} + \frac{R_w C_w}{2N} \right]$$

We have two fixed terms independent on N, then there is a term directly proportional to N and one other inversely proportional to N.

The terms that are independent on N are the mixed terms. It seems that the resistance of the inverter has to drive the overall capacitance of the wire in the term $C_w \cdot R_{eq}$, and that the overall resistance of the wire has to drive the capacitance of an inverter. So it's like the inverter has to drive all the N capacitances, hence the overall capacitance of the wire. Also for the other mixed term, we can split the wire in N pieces, in any case each piece has a small resistance but each piece of wire has to drive the inverter and its gate capacitance. However, these two terms depend on the size of the inverter even if they are independent on the number of pieces. What depends on the number of pieces is the delay of the single inverter and the delay of the wire.

Equation y has in the sqrt the physical parameters of the technology we are using. Since $C_{int} \cdot R_{eq}$ is constant whatever the size, there is no reason to highlight the size. It is a number once chosen the technology.

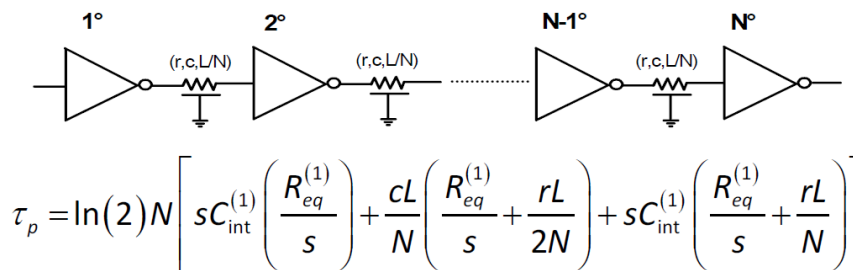
$$\frac{d\tau_p}{dN} = 2C_{int} R_{eq} - \frac{crL^2}{2N^2} = 0 \Rightarrow N = L \sqrt{\frac{cr}{4C_{int} R_{eq}}} = \sqrt{\frac{C_w R_w}{4C_{int} R_{eq}}}$$

If I look at this expression, I notice we have R_w and C_w and the property of the inverter. This is the ratio between the wire delay and the propagation delay of the inverter, aside from factors. This ratio suggests the number of pieces we have to break the wire in. If the delay of the wire is larger than the intrinsic propagation of the inverter, cutting the wire is reasonable.

For $N = 2$ we have the **critical length** of the wire ($N =$ number of pieces), and we assess the minimum length for which it is worth to cut the wire and insert a buffer. The result is $L = 7\text{mm}$.

OPTIMIZING THE SIZE OF THE BUFFERS

Now we do the same analysis without highlighting N but the size S. $C_{int} = s \cdot C_{int}^{(1)}$ and $R_{eq} = R_{eq}^{(1)}/s$. The expression is the same as before.



$$\frac{d\tau_p}{ds} = -\frac{cL}{N} \left(\frac{R_{eq}^{(1)}}{s^2} \right) + C_{int}^{(1)} \left(\frac{rL}{N} \right) = 0 \Rightarrow s = \sqrt{\frac{R_{eq}^{(1)} c}{r C_{int}^{(1)}}} = \sqrt{\frac{R_{eq}^{(1)} C_w}{R_w C_{int}^{(1)}}}$$

Now I want to optimize the propagation delay derivating the τ_p with respect to s. Before taking the derivative, better to perform the s multiplications and grouping the terms as before.

$$\left[\underbrace{2C_{int}^{(1)} R_{eq}^{(1)}}_{T_{INV}} + \underbrace{\frac{C_w R_w}{2N^2}}_{T_W} + \underbrace{\frac{C_w R_w}{N}}_{T_{IW}} + \underbrace{\frac{R_w}{N} \cdot C_{int}^{(1)}}_{T_{WE}} \right]$$

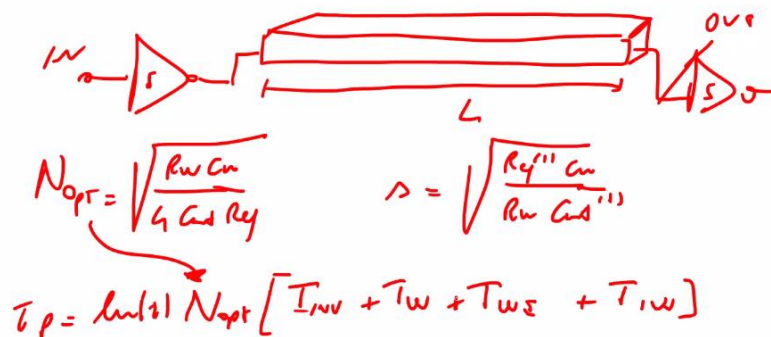
Again, we have two constant terms, the inverter delay and the wire delay, but the mixed terms are size dependent, one increases, the other decreases, because the capacitance of the inverter scales with the size, and so also the resistance of the inverter. The optimum value of s that minimizes the delay is x of the previous image.

The size of the inverter has the Req of the minimum size inverter at the numerator because the size that we have to choose for these inverters is related inversely to the Req of the minimum size inverter. The larger Req⁽¹⁾ the larger s we have to select to compensate for it, and the opposite is value for Cint⁽¹⁾. The same is true for Cw and Rw. The larger Cw, the larger the size we have to select.

If we consider our reference technology, Req⁽¹⁾ is 11.6 kOhm, Cint⁽¹⁾ = 2 fF, then if c = 100 aF/um and r = 0.1 Ohm/um, s = 92, so a large inverter.

If we have to drive inverters with size 92 we have to drive a large capacitance, so the way to proceed in input is to put a superbuffer to drive the first 92*Cint⁽¹⁾ capacitance.

Practical meaning



The size comes into play in T_{iw} and T_{wi}.

$$\begin{aligned} \tau_p &= \ln(2) N_{opt} [T_{inv} + T_w + T_{w_s} + T_{iw}] \\ &= \ln(2) N_{opt} [2\tau_p + 2\tau_p + 2\tau_p + 2\tau_p] \\ &= N_{opt} (8\tau_p) \approx 2.7 \sqrt{R_w C_w C_{nd}^{11} R_{eq}^{11}} \quad x \end{aligned}$$

I get the same delay for all the 4 contributions. So the optimized delay for this problem is Nopt(8*tau_p0). Let's plug Nopt. **Once we have optimized the problem cutting the wire in N equal pieces, each optimized piece has a delay equal to 4 identical terms of 2*tau_p0**, whatever the technology we choose, so there is a benefit in improving the technology because tau_p0 decreases. We cannot do better than this. So the delay is technology dependent (tau_p0), but the way in which it's computed is the same.

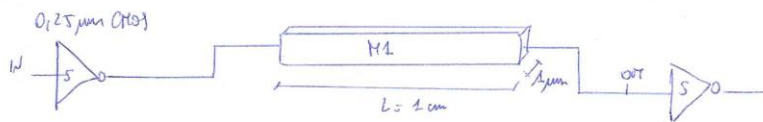
If we consider Nopt, then we have the expression x, that is proportional to the geometrical mean between the wire delay Tw and tau_p0, so the overall delay is technology dependent.

So we start from a large delay of the wire and tau_p0 is small. We can improve the situation basically averaging the two delays of the wire and of the inverter if we cut the wire.

Numerical example

0.25 um CMOS with wire of L = 1 cm and W of the wire W = 1 um. It is implemented in metal 1.

Let's assess Cw and Rw. Cw is made of the parallel plate contribution times the bottom area of the wire plus the fringing contribution times the perimeter.



$$C_w = C'_{pp}(WL) + C'_{p0}(2L) =$$

$$= 30 \frac{\text{aF}}{\mu\text{m}^2} \cdot (1 \mu\text{m} \cdot 10^4 \mu\text{m}) + 40 \frac{\text{aF}}{\mu\text{m}} \cdot (2 \cdot 10^4 \mu\text{m}) = 1,4 \text{ pF}$$

$$R_w = R_{\square} \left(\frac{L}{W} \right) = 0,35 \frac{\Omega}{\square} \cdot 10^4 = 750 \Omega$$

$$\text{Let's normalize by the length: } c' = \frac{C_w}{L} = 140 \frac{\text{aF}}{\mu\text{m}} \quad ; \quad r' = \frac{R_w}{L} = 0,075 \frac{\Omega}{\mu\text{m}}$$

R_w is the square resistance times the number of squares L/W .

Then we normalize everything per the length.

Now we want to minimize the delay and demonstrate that it is worth cutting the wire. The wire delay and inverter delays are:

$$T_w = \frac{R_w C_w}{L} = 286 \text{ ps}$$

$$T_{inv} = 2 \tau_{p0} = 32 \text{ ps}$$

It is worth cutting the wire because the $T_w \gg T_{inv}$.

In fact, if we compute N_{opt} :

$$N_{opt} = \sqrt{\frac{R_w C_w}{4 R_{eq} C_{in}}} = 3$$

\downarrow
11.67 kΩ

The original problem delay with $s = 1$ and $N = 1$ has a dominant contribution that is not $T_w = 286 \text{ ps}$ or $T_{inv} = 32 \text{ ps}$, but it is the $T_{iw} = R_{eq}^{(1)} \ln(2) \cdot C_w = 8.6 \text{ ns}$ (like if the inverter has to drive the overall capacitance).

The model to use in this case is the lumped C model for the wire because the wire acts as a capacitance driven by the large resistance of the inverter. The resistance of the wire is completely negligible because it is more than a factor 10 smaller.

Now we can act on the size s and N . Let's start by acting on the size.

IMPACT OF CMOS PROCESS SCALING ON WIRE PERFORMANCE

In the first column we have the parameters of the wire. Last column to be neglected. Oxide thickness are all reduces, so it is like if for the same metal layer the distance from the oxide is reduced.

As for the length we can distinguish between local wires, which are wires for which the length is reduced, and constant length wires for which the length remains the same.

As for the capacitance C of the wire, we consider just the parallel plate contribution. It is proportional to the bottom area $L*W$ divided by the oxide thickness t . Since all these dimensions scale down with S , in the end the C is reduced with S in local wires.

The problem is however the resistance, which is inversely proportional to the cross-section. The resistance increases by a factor S in the local wires, but S^2 in the constant length wires. Since what counts is the product between C and R , in the constant length wires the delay increases quadratically.

Parameter	Relation	Local Wire	Constant Length	Global Wire
W, H, t		$1/S$	$1/S$	$1/S$
L		$1/S$	1	$1/S_C$
C	LW/t	$1/S$	1	$1/S_C$
R	L/WH	S	S^2	S^2/S_C
CR	L^2/Ht	1	S^2	S^2/S_C^2

The delay remains constant for local wires while increasing for global interconnections!

So with technology scaling we can implement much better transistors, but the wires are worsened.

To address this problem, materials have been changed, **switching to copper from aluminum**, because copper has a smaller resistivity. The other step is to **substitute the silicon dioxide with aerogels**, a material that has an electrical permittivity similar to air.

Constant resistance scaling.

Parameter	Relation	Local Wire	Constant Length	Global Wire
W, t		$1/S$	$1/S$	$1/S$
H		1	1	1
L		$1/S$	1	$1/S_C$
C	$\epsilon_c LW/t$	ϵ_c/S	ϵ_c	ϵ_c/S_C
R	L/WH	1	S	S/S_C
CR	L^2/Ht	ϵ_c/S	$\epsilon_c S$	$\epsilon_c S/S_C^2$

Unchanging the height of the wires only gives a small advantage since it brings into foreground the fringing and inter-wire capacitance problems

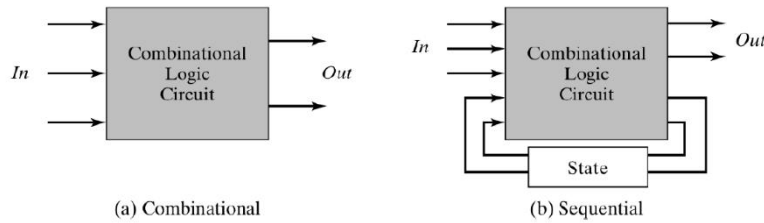
Let's consider $\epsilon_c = 1$ through the table. In order to cope with the previous problem, the height of the wire is kept constant. The aim of technology scaling was to keep the electric field constant, and this is the reason why all the dimensions were reduced. So let's try to keep H constant. Implementing a thin wire very high and so very thin is a real problem with respect to a wire that is 'fatter'. W and t are both reduced by a factor of S .

The capacitance C is scaled with a factor $1/S$ in case of local wires and it remains constant in constant length wires, and so nothing has changed with respect to the previous case, because the C_{pp} is not influenced by the height of the wire. What changes is the resistance. Thus delay decreases for local wires (improvement, but the delay of the wire is not a matter because the wires are short and they act just like capacitor) and scales increasing linearly for constant length wires (wrt S^2 previously).

If with technology scaling the wire is becoming thinner (higher), the interwire capacitance increases, so better not to put close two wires. Also the fringing contribution increases, because the side wall is bigger, but it is a smaller worsening than the interwire capacitance.

FC-CMOS LOGIC GATES

COMBINATIONAL VS SEQUENTIAL



Output = $f(In)$

Output = $f(In, Previous In)$

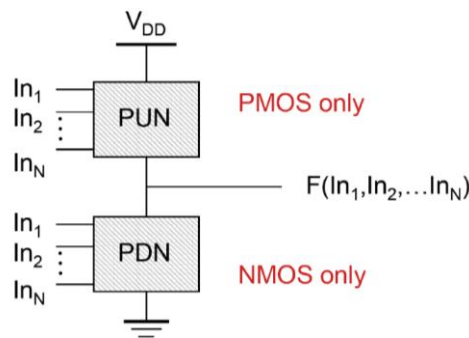
Combinational are digital circuit where the output is a function of the input at a certain time. In case of sequential we have also a memory (logic gates + flipflops), so that the output is a function of the input at the previous time instants. Combinational will be divided in static and dynamic families. In static we have FC-CMOS, ratioed logic (divided in pseudo-NMOS logic and DCVSL) and pass-transistor.

STATIC LOGIC GATES

- **At every point in time** (except during the switching transients) **each gate output is connected to either VDD or GND through a low impedance** (resistance) **path**. At steady state there is always a pMOS from the output to Vdd or a nMOS from the output to ground. During transients it is possible to have both transistors on.
- **The output of the gates always assumes the value of the Boolean function implemented by the circuit** (ignoring the transient effects during the switching periods).
- Static gates are in contrast with dynamic ones, where the output can be a high impedance for one of the two logic values.

IMPLEMENTING A FC-CMOS GATE

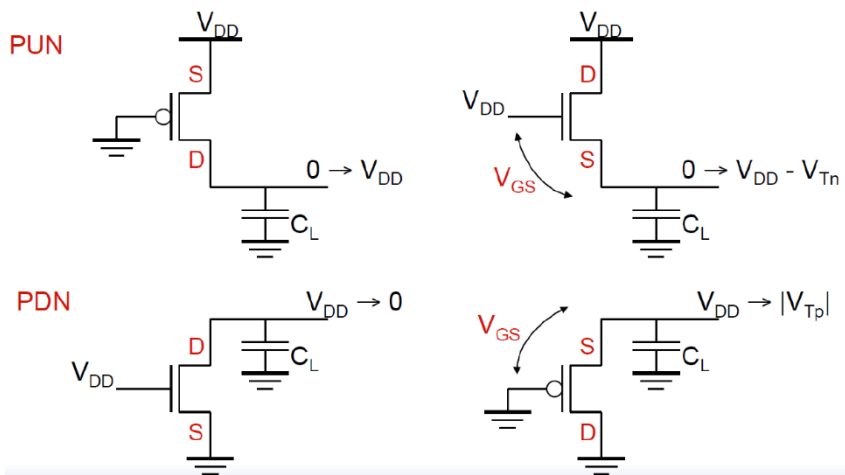
We always have a pull-up network made of pMOS and a pull-down network made of nMOS. If N is the fan-in, we have N pMOS and nMOS in the pull up and pull down networks.



- PUN is implemented with N PMOS transistors, PDN with N NMOS ones
- PUN and PDN are complementary, i.e., if PDN is on, PUN is off, and viceversa

The pull up and pull down are complementary and cannot be on at the same time, otherwise a short circuit between Vdd and ground.

Recap



Let's consider a pMOS (active low device) whose source is to Vdd and the drain to output. $V_{gs} = V_{dd}$, so the pMOS can pull up the capacitor up to Vdd, so to create a direct path between the output and Vdd. During transient the transistor is always on since $V_{gs} = V_{dd}$ and in the end it operates in ohmic with $V_{ds} = 0$.

Let's use a nMOS to pull up a node. Its gate voltage is Vdd, and the output capacitor initially is discharged. At the beginning of the transient $V_{gs} = V_{dd}$, but during the transient it shrinks and the nMOS becomes a much worse current generator. As soon the threshold is passed, at $V_{dd} - V_{tn}$ (where the threshold is affected by the body effect, so higher) we stop the transient, so we are not able to reach Vdd. This is the reason why not to use nMOS in PUN (pull up networks).

The same reasoning can be done for PDN (pull down networks).

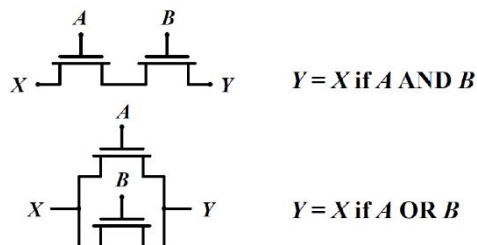
The reasons for nMOS in PDN and pMOS in PUN are:

1. High large swing, and we want the swing large for noise margins, so for reliability issues.
2. Once the transient is over and we look in the drain, it is theoretically infinite, which is good \rightarrow better output resistance.
3. Body effect

PARALLEL AND SERIES OF NMOS TRANSISTORS

□ NMOS transistor is a switch controlled by the gate voltage: It turns on for a high input

□ NMOS passes a strong "0" but a weak "1"



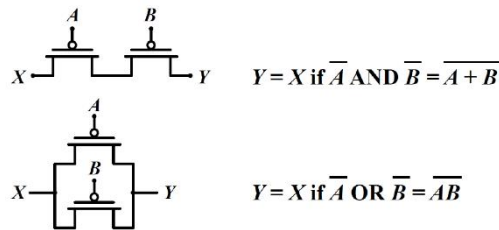
The nMOS is a very good element if we want to transfer a zero (0), but not a 1. Let's put two nMOS in series and on the left side we have a signal X. A and B are logic 1, so the nMOS are on. So the node Y becomes equal to X. Since $Y = X$ if $A \cdot B = 1$ and $Y = 0$ if $A \cdot B = 0$, it seems that $Y = \overline{A \cdot B}$.

Let's put them in parallel. $Y = X$ if $A + B = 1$.

$X = 0 \rightarrow Y = 0$ if $A + B = 1$.

PARALLEL AND SERIES OF PMOS TRANSISTORS

- PMOS transistor is a switch controlled by the gate voltage: It turns on for a low input
- PMOS passes a strong "1" but a weak "0"



pMOS are very effective in passing a logic 1.

NB: $\overline{A \cdot B} = \overline{A} + \overline{B}$ and viceversa (De Morgan's law)

The series of pMOS is the PUN of an NOR gate. The parallel is the PUN of a NAND gate.

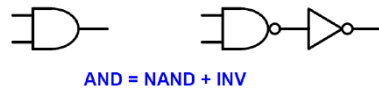
COMPLEMENTARY CMOS LOGIC STYLE

- PUN is the dual to PDN (it can be shown using the DeMorgan's Theorems): NMOS in series correspond to PMOS in parallel and vice versa

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

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

- FC-CMOS gates are always inverting



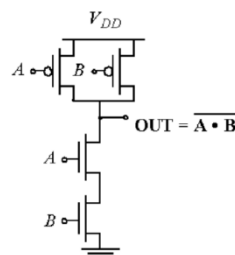
PUN and PDN are dual, i.e. nMOS transistor in series correspond to pMOS transistor in parallel, and viceversa. So if we have a PUN with pMOS in parallel, PDN is of nMOS in series.

Moreover, **FC-CMOS gates are always inverting**. In fact, a nMOS is high for a high input signal, but it implements a PDN, so we have an inversion. The opposite for the pMOS. So to implement an AND gate we need to implement a NAND + Inverter.

Example: NAND and NOR gates

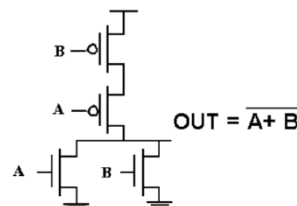
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table of a 2 input NAND gate



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table of a 2 input NOR gate



NAND: $y = \overline{A \cdot B}$. NOR: $y = \overline{A + B}$.

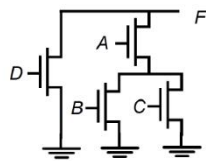
Let's focus on the PDN responsible for the 0. This happens if $A = B = 1$, so I put two nMOS transistors in series for the NAND that close at the same time. Then the PUN is automatically define as the dual of the PDN.

Also for the NOR we can focus on the 0 and I implement a parallel connection in the PDN, then for the PUN again dual property.

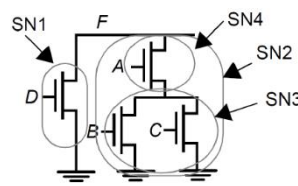
GENERIC FC-CMOS GATE

Let's design the FC-CMOS gate that synthesizes the function:

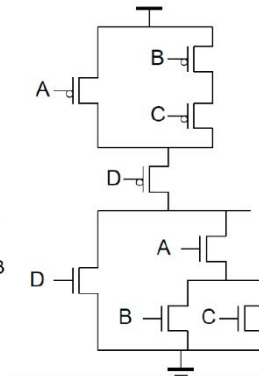
$$Y = D + A \cdot (B + C)$$



1) Design PDN



2) Identify sub-nets



3) Design PUN

$Y = 0$ if $D = 1$ or $A = 1$ and B or $C = 1$. This can be also seen as $Y = 0$ if $D = 1$ or $AB = 1$ or $AC = 1$. $AB = 1$ corresponds to $A = B = 1$, $AC = 1$ to $A = C = 1$.

Once the PDN is identified, I have to identify the subnets SN. The dual property is valid also for bundle of transistors, e.g. SN1 and SN2 are in parallel, SN4 and SN3 are in series. Hence to design the PUN we have to switch the SN in parallel to in series and vice versa.

For instance, let's suppose to have $D = 1$. The output is 0, and the D in the middle of the PUN is off. As a second example to verify that the two networks are complementary, let's suppose $A = 0$. The output is 0 and the PDN is deactivated.

Properties of FC-CMOS logic

FC-CMOS logic is very reliable, and $V_{ol} = 0V$ and $V_{oh} = V_{dd}$ whatever the sizing (**ratioless logic**). Depending on the relative sizing of PUN and PDN, the threshold voltage V_m can be put in the middle.

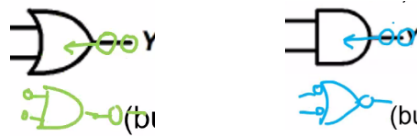
- High reliability
 - $V_{OL} = 0V$
 - $V_{OH} = V_{DD}$
 - V_M can be set approximately in the middle
 - Low-output impedance
- Ratioless logic
- Comparable rise and fall propagation delay
- Negligible static power consumption (no direct path between V_{DD} and GND)

If we can put V_m in the middle, we can theoretically equalize PD and PU times (τ_{plh} and τ_{phl}) as it was for the inverter.

Moreover, the negligible static power consumption is a direct consequence of the fact that PDN and PUN are complementary, never active at the same time.

Bubble pushing theorem

The negation pushed into the NOR gives a NAND with input negated and viceversa.

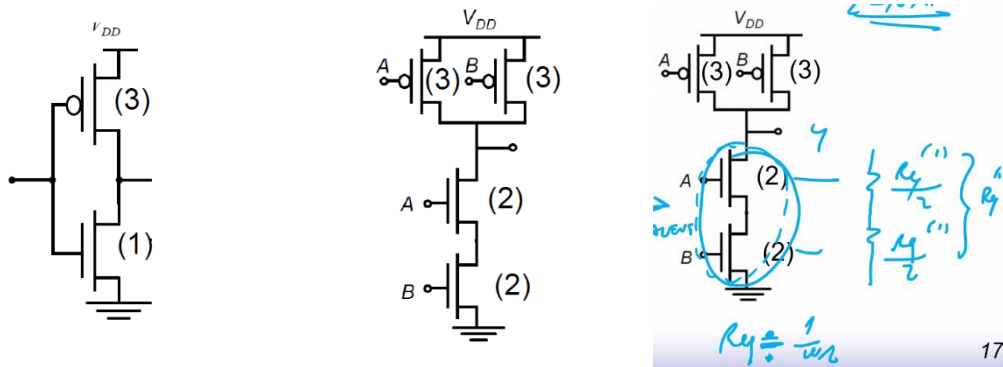


SIZING OF AN FC-CMOS GATE

In the inverter we had $\beta = 3$ to have $V_m = V_{dd}/2$ and τ_{phl} more or less equal to τ_{plh} , so that the pMOS and nMOS networks are electrically equivalent. In the image, $S = 1$ because the nMOS transistor aspect ratio is the size.

A similar reasoning can be applied to the 2 port NAND. The PDN has two transistors in series and I want it equal to the nMOS of the inverter in terms of current and equivalent resistance (electrically equivalent). Since I have two transistors in series, I have two resistances in series, and since the resistance is inversely proportional to the aspect ratio, if I increase the aspect ratio, R_{eq} decreases. So I increase the aspect ratios of the transistors by a factor 2 (two transistors in the series).

- Sizing like the one adopted for the inverter:
 - Threshold approximately in the middle
 - Comparable pull-up and pull-down time



The two transistors in series share one n+ well, so it is like if the electrons have to travel across a length that is $2L$, but the width is the same \rightarrow I need to increase the width and so the aspect ratio to compensate the fact that they are in series.



Instead, for the pMOS parallel, **in the worst case the pull up is performed just by one single transistor.** Then the best case scenario is with both on at the same time. So let's size the pMOS with the same aspect ratio as in the inverter. So there is a difference only when we have a series of transistors.

Now we can say that the classical inverter and the circuit on the right are equivalent in terms of current capabilities. So also the NAND gate has a size of 1 (size = current capability) as the inverter.

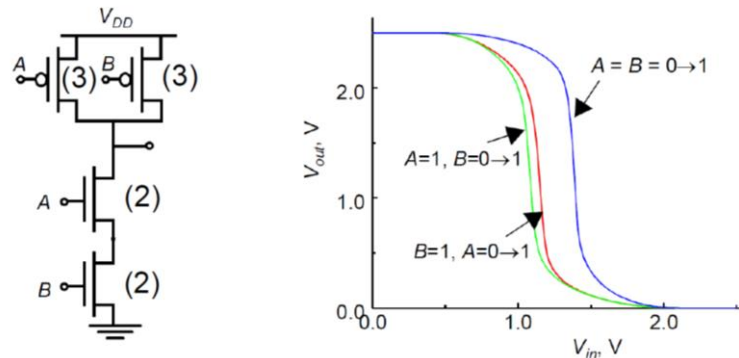
If we had a size 2, in the inverter the nMOS aspect ratio is 2, the pMOS is 6. As for the NAND, 6, 6, 4, 4. The equivalent size of two transistors n series is half the aspect ratio of the series. In parallel is twice the aspect ratio of one transistor.

The NAND gate on the right has a size of 1, because the equivalent size of the pull down path is 1.

STATIC CHARACTERISTIC

The idea is to have a similar property for the NAND gate than for the fully complementary CMOS inverter.

The static characteristic is V_{out} vs V_{in} , but in the NAND gate we have 2 inputs.



- Since N-devices appear in series, their width has been doubled
- The three transitions have threshold quite in the middle
- In the transitions involving two P-devices in parallel V_M is larger, like in an inverter with $(W/L)_p=6$ and $(W/L)_n=1$

The static characteristic is something related to the noise margins, to reliability, to the fact that we have a transition in input and a transition in output. In this case we have 3 cases; for instance $A = 1$ and B that goes from 0 to 1. Then $B = 1$ and A that goes from 0 to 1. Then the case in which both A and B go to from 0 to 1 at the same time, it's like inputs are connected together, so it is like having an inverter in this last case.

If we look at the three characteristics, at first approximation the three are different, but all similar to the characteristic of an inverter. Since the characteristics are very close to 1.25 (V_m), it is a good sizing.

When A and B are both transitioning we have a larger threshold. Let's try to understand why the blue case is above the middle.

If we consider the green and red curves, the circuit has $A = 1$ and B that goes from 0 to V_{dd} . $A = 1$ means that the pMOS is off and the two nMOS on, and the two nMOS can be considered as a single transistor turning on with size 1 (they are in series). The PUN instead is made of a transistor of size (3), so it is an inverter (1) – (3) and the inverter has beta 3.

Instead, for the blue case the pMOS are both on before transitioning, so we have two transistors of size 3 in parallel, hence it is like a unique transistor of size 6 (for the same length, two transistor in parallel is like having a larger transistor in terms of width) → inverter with beta factor 6 ((1) and (6)).

Better than this we cannot do, we cannot have 3 characteristics overlapping.

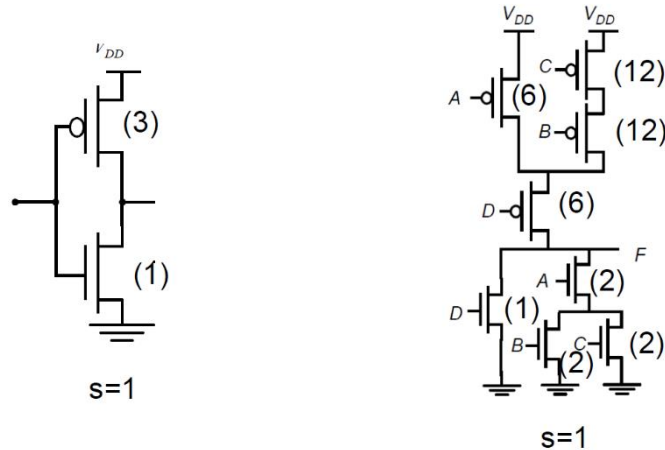
GENERIC FC-CMOS GATE SIZING

$$C = P + \frac{A \cdot (B + C)}{V_{nn}}$$

I want to implement a gate with size 1, so the PUN has to have 3 in the worst case as strength and the PDN has to have 1 in the worst case, as in the image on the left. The function to implement is the one above.

To make sure that in the worst case the size is 1, since A and B are in series, I have to size them with aspect ratio of 2, so that the equivalent aspect ratio is 1. Thus in the worst case, considering all the

possible paths, all the, have (1), so the same equivalent resistance of the fully complementary CMOS inverter.



Same equivalent resistance in the worst-case paths

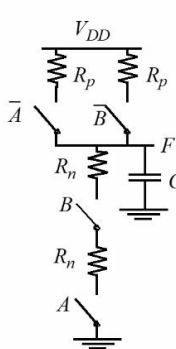
For the PUN, since we want an equivalent resistance for the path $R_{eq}^{(3)}$, since D is sized with (6), C and B have both (12), so that they correspond to an equivalent transistor with (6). Thus in the end the equivalent aspect ration is (3) for every path.

This is not the only possible sizing. If we start from the right PUN, we have C, B and D, we could have used the three transistors in series with (9). Hence the A must be sized with (4.5).

$$\begin{aligned}
 & R_{eq}^{(3)} = R_{eq}^{(x)} + R_{eq}^{(s)} \quad A \text{---} \\
 & \frac{R_{eq}^{(1)}}{3} = \frac{R_{eq}^{(1)}}{8} + \frac{R_{eq}^{(1)}}{9} \quad D \text{---} \\
 & R_{eq}^{(1)} \cdot \frac{1}{3} = \frac{1}{8} + \frac{1}{9} \quad D \text{---} \\
 & \frac{1}{x} = \frac{1}{8} + \frac{1}{9} = \dots \\
 & \frac{1}{x} = \frac{2}{9} \Rightarrow \boxed{x = 4.5}
 \end{aligned}$$

The best solution is the one that minimizes the output capacitance C_{int} . The previous one is the best because the pMOS is smaller, and C_{int} is proportional to the width.

PROPAGATION DELAY IN THE NAND GATE



2-input NAND

- Each transistor modeled as its equivalent resistance $R_p = R_p^{(3)}, R_n = R_n^{(1)}/2$
- Determine the worst-case transitions:
 - As for the pull-up when 1 PMOS charges C
 $\tau_{pLH} = 0.69R_p^{(3)}C$
 - As for the pull-down when 2 NMOS discharges C
 $\tau_{pHL} = 0.69(2R_n)C = 0.69R_n^{(1)}C$
- C is twice the one of a minimum-size inverter
 $C = \frac{C_{int}^{(1)}}{(1+3)}(2+3+3) = 4fF$

19

Now between parenthesis we have the aspect ratio of the transistor, not the size of the inverter, because we are considering the pMOS transistor. So if I consider transistors, between parenthesis we have the aspect ratio, if I consider the inverter I put the size between parenthesis. $R_p^{(3)}$ is the equivalent resistance of the pMOS, which is $13 \text{ k}\Omega/2$, where $13 \text{ k}\Omega$ is the R_{eq} of the nMOS.

Let's assess the worst case PU. It involves just one pMOS transistor. C is C_{int} , the intrinsic capacitance of the gate. τ_{pLH} is the pull up time.

As for the PD path, it involves two nMOS in series whose sum is $2 \cdot R_n$, that is $R_n^{(1)}$. The worst case is when both are on.

The problem is assessing the value of C_{int} .

Cint assessment

For a minimum size inverter of $s = 1$, $C_{int}^{(1)} = 2 \text{ fF}$. If we consider our gate, $(3) || (3) + (2) + (2)$, since the length is the same for all the transistors, we can say that C_{int} is proportional to the aspect ratio, so for $s = 2$, $C_{int} = 4 \text{ fF}$.

On the output node we have two pMOS in parallel with twice the aspect ratio. So we take $C_{int}^{(1)}$ and we divide it by the sum of the aspect ratio of the minimum size inverter, then multiply by the transistors' aspect ratios connected to the output. It is a mathematical proportion.

Let's assess C_{int} in the classical way. At the drain node, the specific capacitance is $2 \text{ fF}/\mu\text{m}$. We have 3 drains connected to the output in our case.

$$C_{int} = C' (W_p + W_p + W_n)$$

However, we have the aspect ratios, not the widths. So I can rewrite it as the sum of the aspect ratios times the length, that is the same for all the transistors (it is not $C_{int}^{(1)}$).

$$C_{int} = C' \cdot \sum \left(\frac{W}{L} \right) \cdot L$$

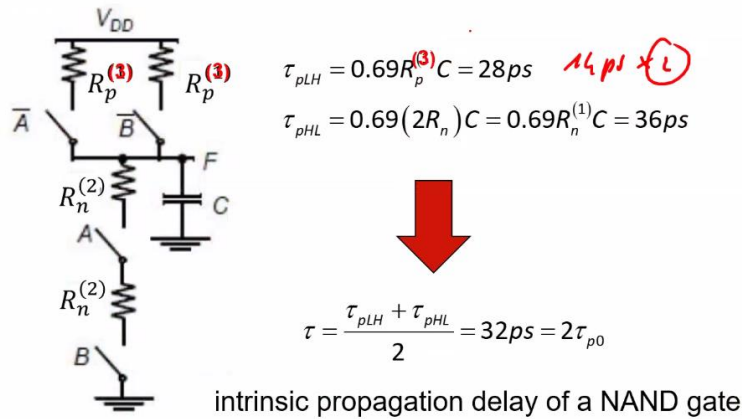
In the end, for a NAND with size 1 $C_{int} = 4 \text{ fF}$. So the resistance is the same, but the capacitance is twice. So the intrinsic propagation delay of the NAND is twice the one of the inverter.

Does the *intrinsic* propagation delay depend on the size of the NAND gate?

The resistance is smaller, the capacitance is increased if we increase the size, so overall the intrinsic delay stays the same.

Propagation delay

The factor of 2 comes from the capacitance.



So the intrinsic propagation delay (should be tau_p in the image) is two times larger due to the capacitance. What changes is the Cint, not the Req.

The one below are the results of the simulations.

$\tau_{pLH} (A = B = 1 \rightarrow 0) \cong 17ps$ $\tau_{pLH} (A = 1, B = 1 \rightarrow 0) \cong 38ps$ $\tau_{pLH} (B = 1, A = 1 \rightarrow 0) \cong 27ps$	}	pull-up delays
$\tau_{pHL} (A = 1, B = 0 \rightarrow 1) \cong 40ps$ $\tau_{pHL} (B = 1, A = 0 \rightarrow 1) \cong 32ps$ $\tau_{pHL} (B = A = 0 \rightarrow 1) \cong ?$	}	pull-down delays

What's happening? The parasitic capacitance between the 2 NMOS transistors is coming into play

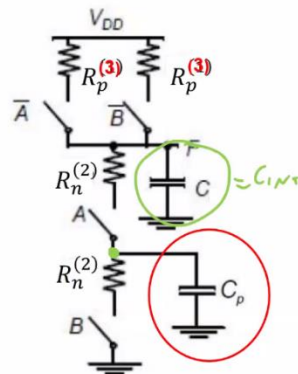
For a NAND gate we can have different possibilities; in the first three cases we have the PU. The first case is with both inputs shorted to 1. The smaller PU time is smaller and reasonable because we have two pMOS together pulling up the output node. This is the best case. The third is the worst case, pulling up the output capacitor with just one transistor. 27 ps is similar to the rough analysis previously done (28ps). However, when I swap the inputs, and B is turned on (second case), the pull up takes a longer time, 38 ps.

For the PD, if both A and B transition from 0 to 1, the delay can be either 40 ps or 32 ps.

There is something we are neglecting to consider to understand why the simulations results are different from the pen and paper analysis. The problem is the parasitic capacitance Cp in the middle of the two nMOS transistors.

ASSESSING THE PARASITIC CAPACITANCE

In a generic transistor, for the gate and drain we have $C' = 2\text{fF}/\mu\text{m}$. At the source, it is not so far from this value, so let's also consider $2\text{fF}/\mu\text{m}$ at the source. We want to estimate with this approximation the C_p .

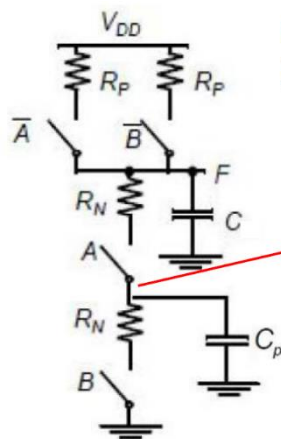


C_p is connected to 2 sources of $W_n=0.5\mu\text{m}$ NMOS transistors:

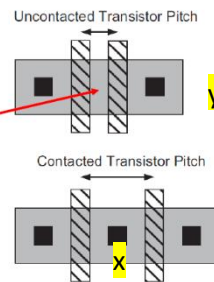
$$C_p = \frac{2\text{fF}}{\mu\text{m}}(2W_n) = 2\text{fF}$$

C_p is proportional to the sum of the widths of the transistors connected to the node, so it is $C' \cdot [W_n + W_n]$. Again I can express the width as sum of the aspect ratios multiplied by the same length. In the end we get $2\text{fF} = C_p$. Since $C_{int} = 4\text{fF}$, we cannot neglect C_p .

Better estimate for C_p



C_p is reduced by the fact the junction is shared and narrowed



$$C_p = \alpha \frac{2\text{fF}}{\mu\text{m}}(2W_n) < 1\text{fF}$$

The two nMOS transistors share the source-drain regions in the intermediate node. Furthermore, contact x means that we need a large area for the source and the drain, but it is not a node of interest, so I can remove the contact and if we look at the view of two nMOS in series, we can have a small junction removing the node (y), so that C_p can be reduced by a factor alpha (more or less alpha is 1/3).

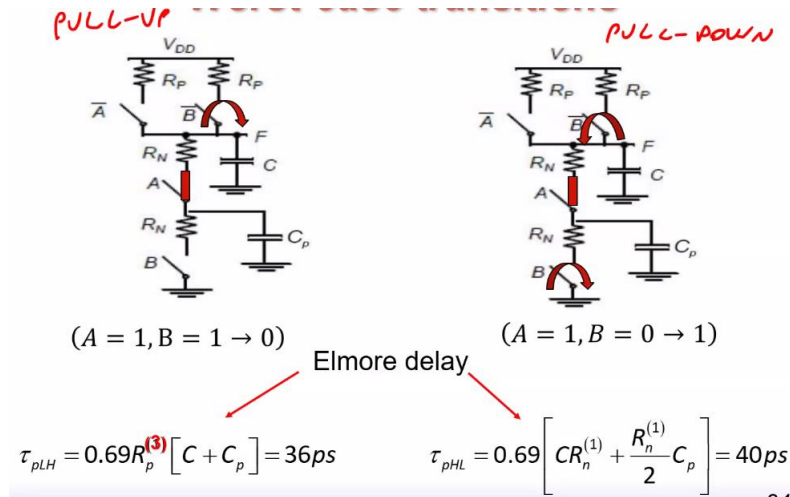
So C_p in the end is a bit smaller than 1 fF. Let's re-assess the two worst case transitions.

Worst-case transitions

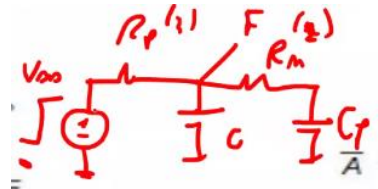
\bar{A} and \bar{B} indicate an active low signal.

At the beginning B is 0, but the first nMOS A is on, so there is a connection with C_p . At $t = 0$, C_p is 0V because B nMOS was on, so C_p was discharged to ground. Once the pMOS B is on, we charge C_{int} and also C_p . C_p is charged up to $V_{dd} - V_t$, because then the nMOS A is off.

We can draw an equivalent electrical model.



Equivalent electrical model for PUN



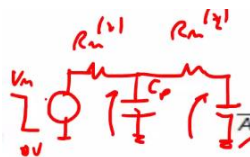
Let's apply the Elmore delay theorem for the delay.

$$\tau_{pLH} = 0.69 \left[CR_p^{(1)} + C_p R_p^{(2)} \right]$$

So the charging of C_p was the factor missing previously.

We can now shift to the pull-down. It is the same as before. When $A = 1$ and B is turned on, we have the pull down, so C_{int} is discharged, but also C_p is discharged. Before the transition, C_p was charged to $V_{DD} - V_t$.

Equivalent electrical model for PDN



In the end I get 40 ps.

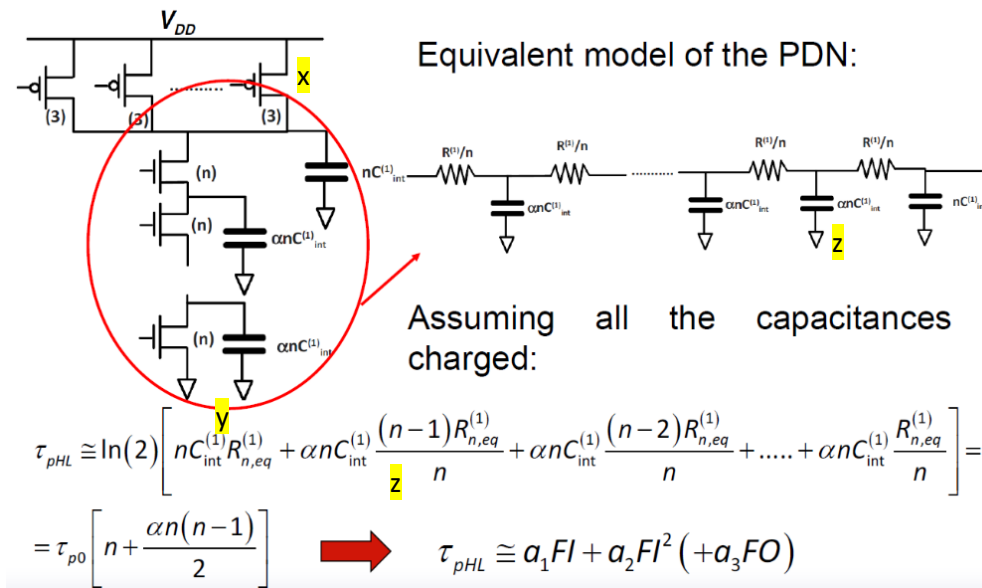
Instead, the 32 ps in the simulations for the PD cannot be explained by the model, it comes from the fact that the series of two transistors is not actually equal to one single transistor with half the aspect ratio. The 40 ps delay is instead justified as above.

Other transitions

In the other transitions C_p is not coming into play because it is already discharged/charged. Actually, even if it is discharged, the intermediate node to which C_p is attached increases its voltage and then decreases it to accommodate the current to discharge C_{int} in the pull down transition.

GENERAL CASE

In the image we have a n inputs NAND (fan-in = n). We want to compute the worst-case pull down. This happens when we turn on the transistor at the bottom of the PDN, and the other ones are already on, while the pMOS are already off except for the last one on the right, which is turned off (x).



Let's assess all the capacitances involved. The NAND has a size of 1.

I want to size the NAND gate with n inputs with S = 1. This is made by a lot of pMOS in parallel, and n nMOS transistors in series. For the worst case PDN we want to have the equivalent sizes of an inverter (3) – (1). So all the pMOS will have an aspect ratio of (3), and the nMOS of (n), so that their equivalent resistance is Req⁽¹⁾. As for Cint, it is n times Cint⁽¹⁾, so n*2 fF.

Then I have the intermediate nodes, whose capacitances can come into play in the transitions. They are proportional to alpha, n and Cint⁽¹⁾, with alpha much smaller than 1.

As for the first Cp of the pile, we can compute it as:

$$C_p = c' \cdot (W_p + W_n)$$

$$= c' \cdot L \left[\left(\frac{W}{L} \right)_p + \left(\frac{W}{L} \right)_n \right]$$

$$= c' \cdot L \cdot (Z_m)$$

This capacitance can be reduced because we can share the junctions, so Cp is smaller than this computed value (k << 1).

$$C_p \approx k c' L (Z_m) = (2) \cdot n \cdot \frac{c' L}{c' L} \cdot c' L$$

Worst case PD path

The network is sketched in the previous slide. The resistances are all equal and equal to Rn⁽¹⁾/n, 13 kOhm/n.

The model is complex because it involves n capacitances. Y is the classical contribution of the current that passes through all the resistances to discharge Cint at the output.

Then we have the contribution of the capacitance z.

I can rewrite it grouping ln(2)*Rn⁽¹⁾*Cint⁽¹⁾, which is actually tau_p0 (16ps).

$$= \tau_{p0} [n + \alpha(n-1) + \alpha(n-2) + \dots + 1] =$$

$$= \tau_{p0} \left[n + \alpha \frac{n(n-1)}{2} \right] \left(+ \ln(n) \text{ times } C_c \right)$$

↑
sum of n-1 integers (x)

INTRINSIC PROPAGATION DELAY

Aside from C_{int} , if I would have also an external contribution C_c , I should add the given term proportional to C_c .

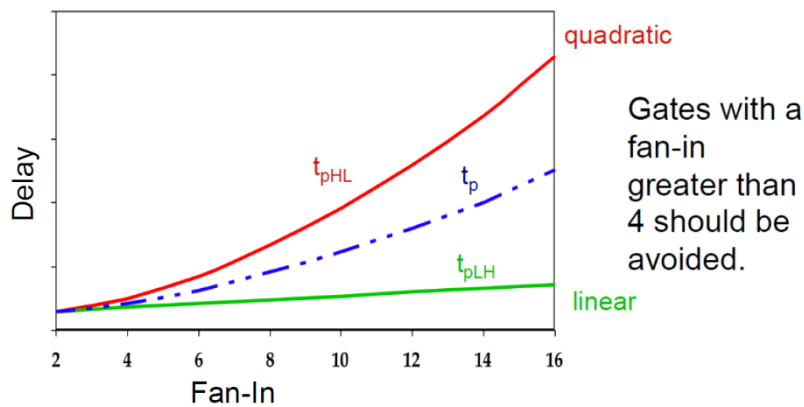
$\tau_{p0} * n$ is a term that increases linearly with n and it is expected because what changes between an inverter and a NAND for the same size is the output capacitance. In the NAND, the C_o is n times the $C_{int}^{(1)}$ of the minimum size inverter. So we expect a larger intrinsic propagation delay because the output capacitance is greater.

Then we have also another term, x , which cannot be neglected because it is proportional to n^2 . It is hence a quadratic dependence in the worst case. For a small number of inputs this term can be neglected because α is much smaller than 1, but once we implement a NAND with a lot of inputs it cannot be neglected. This is why we will never see NAND gates or NOR gates with a number of inputs greater than 4.

In the worst case, the propagation delay is linear with the fan in FI ($FI = n$), then quadratically depends on n and if we have a load capacitor we also have a third term which depends on the fanout.

$$\tau_{pHL} \cong a_1 FI + a_2 FI^2 (+ a_3 FO)$$

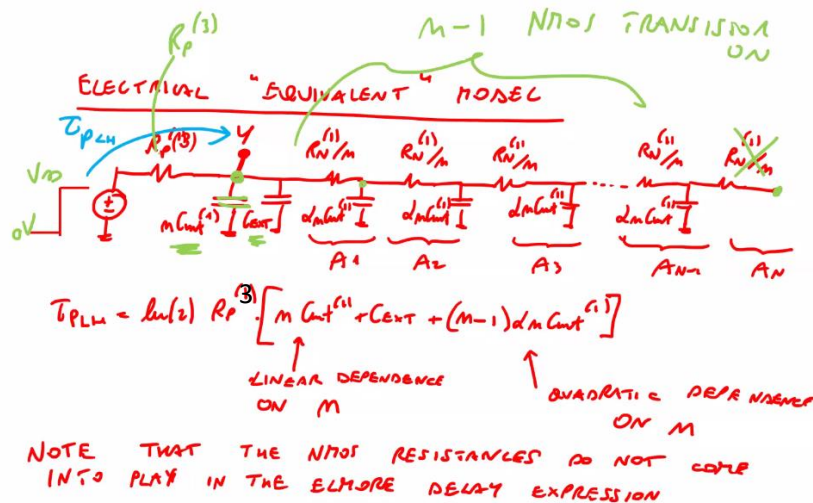
DELAY AS A FUNCTION OF THE FAN-IN



In the plot, we are considering the intrinsic delay in the worst case to show its dependence on the fan-in. For a small number of inputs we have a straight line, which then becomes a parabola, with a quadratic dependence on the fan-in. So for a small number of inputs the delay dependence is linear with n .

The green curve is wrong. In the worst case we don't have a linear relationship for the PU transition, also it has a quadratic dependence as for τ_{pHL} .

Worst case PU path

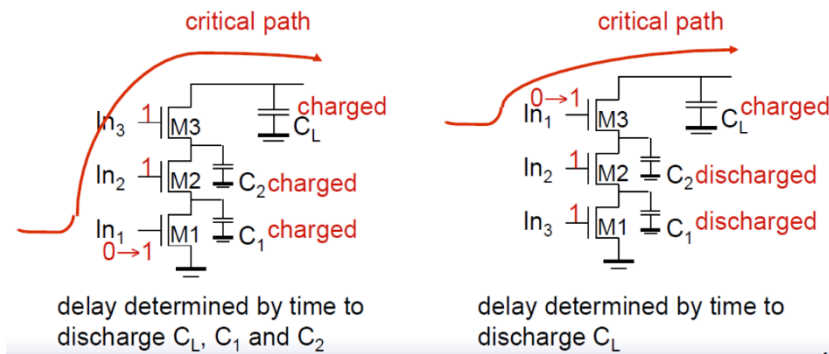


So the tau_pLH trend is quadratic with n^2.

CLEVER DESIGN – TRANSISTOR ORDERING

Transistor ordering

The last signal to transition close to the output node



Let's consider a 3 inputs NAND gate, and the 3 inputs are not equivalent, e.g. the path corresponding to C has some delay before transitioning, it is the last input to reach the NAND. Where is the best place to put the input C?

We can consider two cases. In terms of nMOS transistors, in one case C is at the bottom, in the other case connected to the top transistor.

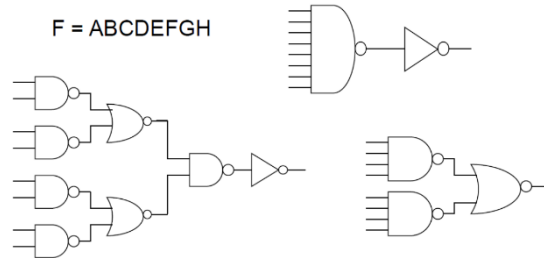
In the left case, before the transition of C occurs all the parasitic capacitances are charged, because the other nMOS are already on. Then I have the transition and all the capacitances must be discharged.

In the right case, the parasitic capacitances are already discharged and once the transition of C occurs, only C1 has to be discharged, so this case is the better. So it is better to connect the slowest signal to the transistor closer to the output node.

Large fan-in gates

I want to implement the AND function F . We can use a NAND gate with 8 inputs followed by an inverter or use simpler gates with less inputs.

Avoid large fan-in gates



Which is the best circuit to implement an AND function?

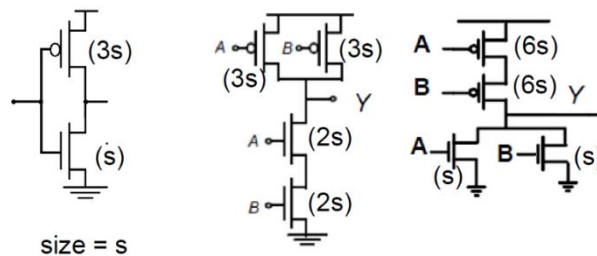
The one with a very large fan-in is not the best one because it has 8 inputs and so it involves a delay that is not linear with the number of inputs.

In the left case we have a lot of gates in cascade, while on the right more inputs but less in series, so they seem almost equivalent. The difference is when driving a load, having more stages in cascade it's easier to drive a load. So for small C_l it is better the solution on the right, for high C_l the solution on the left.

Once the load is given, how can we size the gates in the path?

Sizing of a gate

- **Size of a gate:** a generic gate has a **size s** if it shows the same equivalent resistance (worst case) of an inverter with size s



- Let's **neglect internal parasitic capacitances** (analysis valid only for small fan-in)

The first is the NOT, then the NAND and NOR.

Two gates with the same size have the same equivalent resistance for the worst case path. The inverter (NOT) has a size s . The one in the image is the sizing to have an equivalent resistance. For instance, the NOT has the PDN with size s because in the worst case only one transistor nMOS is on, so it has to have the same size of the inverter.

What changes between the three implementations is the overall output and input capacitance. Once the length is fixed, the capacitance is proportional to the width (in reality to the aspect ratio), and so the input capacitance of the NAND ($5s$) is larger than the inverter. It is even larger for the NOR.

As for the output capacitance, e.g. in the case of the NOR, it is $8s$ ($6s + s + s$), while in the inverter it is $4s$, so a factor 2 greater.

From now on we will neglect the internal parasitic capacitances, so the analysis is valid for small fan-in gates.

LOGICAL EFFORT AND ELECTRICAL EFFORT

We want to compare a generic FC-CMOS gate with the reference inverter.

Aim: compare the generic gate with the reference inverter

- **Intrinsic delay factor (p):** how the intrinsic capacitance is worsened by the complexity of the gate with respect to the inverter

$$p = \frac{C_{int}(s=1)}{C_{int}^{(1)}}$$

- **Logical effort (g):** how the gate capacitance is worsened by the complexity of the gate with respect to the inverter case

$$g = \frac{C_g(s=1)}{C_g^{(1)}}$$

- **Electrical effort or fan-out (f):** ratio between external and gate capacitance

$$f = \frac{C_{ext}}{C_g}$$

3:

Intrinsic delay factor

p is the ratio between the capacitance at the output of a generic gate and the capacitance at the output of the inverter, for the same size. For a NAND and NOR gate, p corresponds to the fan-in, and the C_{int} to put in the formula is not the C_{int} of the inverter, obviously, but of the gate itself. Always, in a gate, p > 1. It quantifies the complexity of the gate at the output node, compared to the inverter.

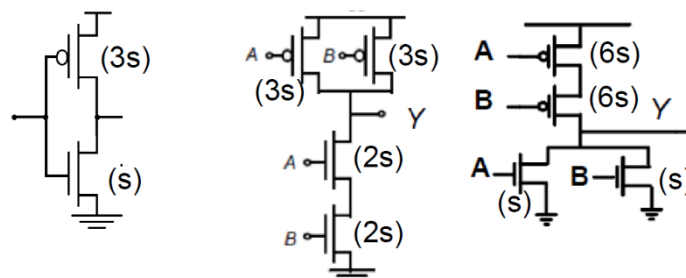
Logical effort

Ratio between the gate capacitance of the FC-CMOS gate and the one of the inverter for the same size. For an inverter of s = 1, C_g(1) = C_{int}(1) = 2 fF. For a NAND of s = 1, the ratio is 5/4 (5s of the nand and 4s for the inverter). It quantifies the complexity of the gate at the input, compared to the inverter.

Electrical effort

It is the fanout, ratio between the external capacitance and the gate capacitance. If I increase the size, the fanout decreases linearly with s, because C_g is proportional to the size also in generic gates.

Example



Comparing the gate with the same size, e.g. s=1:

$$p = 1$$

$$g = 1$$

$$p = 2$$

$$g = 5/4$$

$$p = 2$$

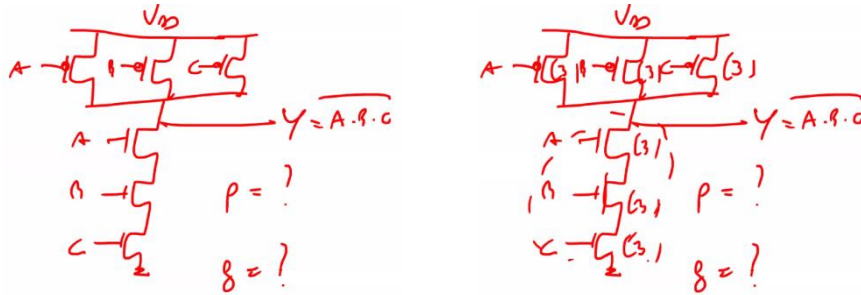
$$g = 7/4$$

Watch out: p and g depend on the topology, not on the size!

For the NOR, g = 7/4 because C_g = C'*[W_p + W_n] = C'*L_{min}*(6+1) = 7C'*L_{min}, and in the inverter C_g(1) = L(3+1)C'.

PROPAGATION DELAY OF A GENERIC GATE

Let's assess the delay of a NAND gate of 3 inputs and let's assess p and g to compare it with a reference inverter ((3), (1)). To have the same size, all the transistors of the NAND have an aspect ratio of 3.



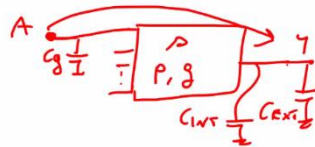
Connected to the output node we have 4 transistors with aspect ratio of 3, while in the reference inverter a transistor of aspect ratio 3 (pMOS) and one 1 (nMOS). The C_{int} in output of the gate is the $C_{int}(1)$ (output of the inverter) times the sum of the aspect ratio of the gate divided by the sum of the aspect ratio of the inverter.

$$C_{int} = C_{int}^{(1)} \cdot \frac{\sum \frac{W}{L}}{1+3}$$

$$= C_{int}^{(1)} \cdot \frac{4 \cdot 3}{4} = 3 \cdot C_{int}^{(1)}$$

So $p = 3 = \text{fan-in}$, and $g = 6/4 = 1.5$ (we have to consider one input for the calculations, e.g. A. For NAND and NOR the g is the same regardless the input, for other gates we might have different logical efforts depending on the input).

Let's generalize the result, for a generic gate of size s for which I want to assess the delay τ_p from A to y.



Considering a generic gate with size s and an external load C_{ext} :

$$\begin{aligned} \tau_p &= \ln(2) R_{EQ} (C_{int} + C_{ext}) \\ &= \ln(2) \frac{R_{EQ}^{(1)}}{s} (psC_{int}^{(1)} + C_{ext}) \quad \text{X} \\ &= \ln(2) R_{EQ}^{(1)} C_{int}^{(1)} \left(p + \frac{C_{ext}}{sC_{int}^{(1)}} \right) \quad \text{Y} \\ &= \tau_{p0} \left[p + \frac{C_{ext}}{\gamma s C_g^{(1)}} \right] = \tau_{p0} \left[p + \frac{C_{ext} C_g}{\gamma s C_g^{(1)} C_g} \right] \\ &= \tau_{p0} \left[p + \frac{1}{\gamma} \cdot \underbrace{\frac{C_{ext}}{C_g}}_f \cdot \underbrace{\frac{C_g}{s C_g^{(1)}}}_g \right] = \tau_{p0} \left[p + \frac{fg}{\gamma} \right] \end{aligned}$$

34

The resistance scales down with the size with respect to size (1), but also the C_{int} increases with s. According to x, if we have an external load and we want to decrease the propagation delay, we can act on the size.

We can look at y and, if we increase s, the delay decreases up to the point where the delay is $\tau_{p0} \cdot p$, so there is a dependance of the delay on the size, increasing the size we decrease the delay.

Then we change $sC_{int}(1) = \gamma \cdot s \cdot C_g(1)$. Then I multiply numerator and denominator by C_g so that we can highlight the fanout = C_{ext}/C_g . Then also $C_g/sC_g(1)$ is the logical effort.

$$= \tau_{p0} \left[\rho + \frac{1}{\gamma} \cdot \frac{C_{ext}}{C_g} \cdot \frac{C_g}{sC_g(1)} \right] = \tau_{p0} \left[\rho + \frac{fg}{\gamma} \right]$$

$\tau_{p0} \cdot \rho$ INTRINSIC DELAY
 $\tau_{p0} \cdot \frac{fg}{\gamma}$ EFFORT DELAY

The delay doesn't depend on the logical effort of the gate g , so the dependance in the formula is fake, in fact if we express f , the g cancels out.

However, the expression above is useful when we want to minimize the delay of a chain of gates.

INTRINSIC DELAY AND EFFORT DELAY

The following expression loses the physical meaning of the delay, which is $\ln(2) \cdot R_{eq} \cdot (C_{int} + C_{ext})$. In the end, increasing the size we can decrease the propagation delay, the size is the only parameter on which we can act.

$$\tau_p = \tau_{p0} \left[\rho + \frac{fg}{\gamma} \right] \leftarrow \tau_{p0} \left[\rho + \frac{C_{ext}}{sC_{int}(1)} \right]$$

$$\tau_p = \tau_{p0} \left[\rho + \frac{fg}{\gamma} \right]$$

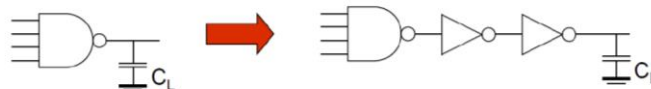
- The delay depends:
 - on the complexity of the gate through ρ and g
 - on the capacitive load through f
 - on the adopted technology through τ_{p0} (and γ)
- The delay is made by two terms:
 - parasitic (or intrinsic) delay $\tau_{p0}\rho$
 - effort (or extrinsic) delay $\tau_{p0}(fg/\gamma)$
- Only the fan-out, f , depends on the size being the ratio between C_{ext} and C_g , the latter depending on the size

35

OPTIMIZING THE GATE SIZE

Question #2

Is it better to drive a large capacitive load directly or after some buffering?



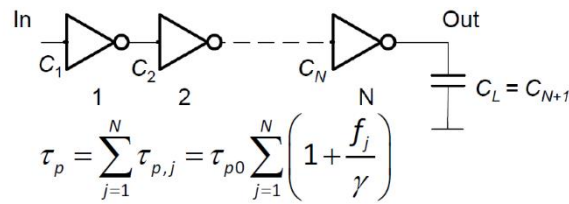
METHOD

- ✓ Logic effort method (suggested by Ivan Sutherland)
- ✓ Original paper published in 1991 (uploaded in the beep portal)
- ✓ Extension of inverter chain theory

37

Inverter chain sizing

The delay is minimized if all the added inverters have the same fanout if N is fixed. If N can be changed, we have to select N so that $f_{opt} = 3.6$ for all the inverters.



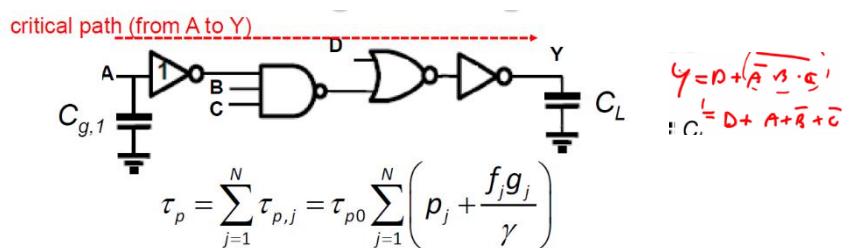
□ Minimum delay for equal fan-out of all the inverters, f_{opt}

□ Path fan-out: $F = \prod_{j=1}^N f_j = \frac{C_L}{C_{g,1}}$

□ If N is fixed: $f_{opt} = \sqrt[N]{F}$

□ If N can be set: $N_{opt} = \ln(F) / \ln(f_{opt})$ $f_{opt} \cong 3.6$

Chain of generic gates



$$\tau_p = \sum_{j=1}^N \tau_{p,j} = \tau_{p0} \sum_{j=1}^N \left(p_j + \frac{f_j g_j}{\gamma} \right)$$

□ Path electrical effort (or path fan-out): $F = \prod_{j=1}^N f_j = \frac{C_L}{C_{g,1}}$

□ Path logical effort: $G = \prod_{j=1}^N g_j$

□ Path effort: $H = F \cdot G$

□ Minimum delay for equal stage effort $h_{opt} = f \cdot g = \sqrt[N]{H} = \sqrt[N]{FG}$

We can define a **critical path**, which is the path that corresponds to the largest delay, that in our case is from A to Y. The other delays are smaller. Once we have a path, we define the delay as the sum of the ideal delays.

For each gate I have to assess p and g. Of course, for the inverters $p = g = 1$. As done for the inverter chain, we can define a path fanout F. Now, since we deal with more complex gates, we can also consider the path logical effort, product of the logical efforts of all the gates in the chain. Multiplying F and G we get the **path effort H**.

Once we have H, we can easily verify that the delay of **the critical path delay is minimized if we size the gates so that each gate has the same stage effort $f \cdot g$** .

It is a result similar to the inverter chain theory, the only thing that changes is the addition of the path logical effort.

Design flow

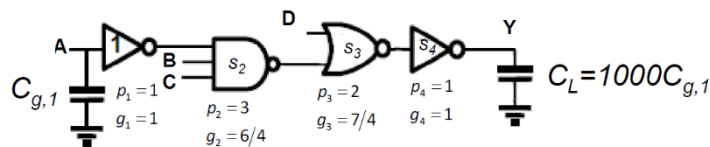
1. Assess the path electrical effort (path fan-out), F
2. Assess the path logical effort, G
3. Assess the path effective effort, $H=FG$
4. Assess the optimum stage effort, h_{opt}
5. Assess the size of the gates in order to have equal effort:
6. Assess the minimum delay, $\tau_{p,opt}$:

$$\tau_{p,opt} = \tau_{p0} \sum_{j=1}^N \left(p_j + \frac{h_{opt}}{\gamma} \right) = \tau_{p0} \left[\left(\sum_{j=1}^N p_j \right) + N \frac{h_{opt}}{\gamma} \right]$$

1. $F = Cl/Cg,1$
2. $G = \text{prod}_j(g_j)$
3. $H = FG$
4. $h_{opt} = \text{sqrt}_N(H)$

So each stage has the same $h_{opt} = f^*g$. This is not equal to say that each stage has the same delay, what it's in common between the stages is the effort delay, extrinsic delay. The intrinsic delay depends on the gate we are considering.

Example of optimization

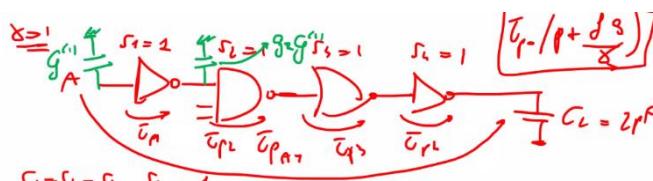


Considering all stages with size $s=1$:

$$\begin{aligned} \tau_p &= \tau_{p0} \sum_j \left(p_j + \frac{f_j g_j}{\gamma} \right) = \tau_{p0} (1 + g_2) + \tau_{p0} \left(3 + \frac{g_3}{g_2} g_2 \right) + \tau_{p0} \left(2 + \frac{g_4}{g_3} g_3 \right) + \tau_{p0} \left(1 + \frac{C_L}{g_4 C_g^{(1)}} \right) = \\ &= \tau_{p0} \left(1 + \frac{6}{4} \right) + \tau_{p0} \left(3 + \frac{7}{4} \right) + \tau_{p0} (2 + 1) + \tau_{p0} (1 + 1000) = \\ &= 1011.2 \tau_{p0} \end{aligned}$$

Watch out: the input capacitance of a gate is $sgC_g^{(1)}$!

$$F = Cl/Cg,1 = 1000.$$



Let's assess the delay of the circuit before the optimization, with s_1, s_2, s_3 and $s_4 = 1$.

$$\tau_{p,1} = \tau_{p1} + \tau_{p2} + \tau_{p3} + \tau_{p4}$$

Let's use the formula that involves the fanout (f_i = fanout of the i -th stage).
 For the 3-input NAND $p = 3$, for the 2 input NOR it is 2.

$$= \tau_{p0}(1+f_1) + \tau_{p0}(3+f_2g_1) + \tau_{p0}(2+f_3g_2) + \tau_{p0}(1+f_4)$$

Now we have to consider the gate capacitance of the NAND, that is the load capacitance for the inverter,

$$= \tau_{p0}(1+g_2) + \tau_{p0}(3+\frac{g_3}{g_2} \cdot g_2) + \tau_{p0}(2+\frac{g_3}{g_2}) + \tau_{p0}(1+\frac{C_L}{C_{g1}})$$

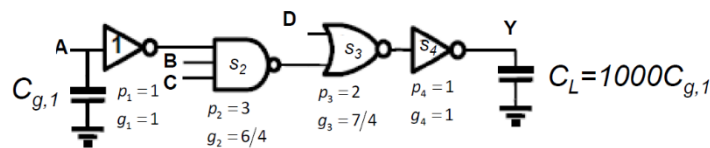
and the C_g of the inverter. The gate capacitance of the ANNAD is $g_2 \cdot C_g(1)$. As for the delay of the NAND, the gate capacitance of the NOR is $g_3 \cdot C_g(1)$, and it is the load capacitance of the NAND. In this case g_2 cancels out.

The largest delay in this chain is the last one, overwhelming the other contributions.

Let's try to minimize this delay.

The steps to be performed are in the next image.

$$\tau_{p0}(1+\frac{C_L}{C_{g1}}) + \tau_{p0}(3+\frac{C_L}{C_{g1}}) + \tau_{p0}(2+1) + \tau_{p0}(1+1000)$$



- | | |
|----------------------------|--|
| 1. Path electrical effort: | $F=1000$ |
| 2. Path logical effort | $G=(1) \cdot (1.5) \cdot (1.75) \cdot (1)=2.625$ |
| 3. Path effort: | $H=FG=2625$ |
| 4. Optimum stage effort: | $h_{opt} = \sqrt[4]{H} \cong 7.16$ |
| 5. Sizes: | $s_1=1, s_2=4.8, s_3=19.5, s_4=140$ |
| 6. Minimum delay: | |

$$\tau_{p,opt} = \tau_{p0} \left[\left(\sum_{j=1}^N p_j \right) + N \frac{h_{opt}}{\gamma} \right] = 35.64 \tau_{p0}$$

We can rewrite the situation. The unknowns of the problem are s_2, s_3 and s_4 . In order to minimize the delay from A to Y each stage must have an effort $h = f \cdot g = 7.16$. The size is inside the term f .

For the first inverter $s_1, h = f_1 = 7.16$, which is the ratio of two capacitances. In fact at the input of the NAND the C is $s_2 \cdot g_2 \cdot C_{g1}$.

$$h = f_1 = 7.16 \quad [f_1=1]$$

$$= \frac{s_2 g_2 C_{g1}}{C_{g1}} = s_2 g_2 = 7.16 \rightarrow s_2 = \frac{7.16}{g_2}$$

Let's move to h_2, g_2 is $6/4$. Capacitance in input of the NOR is $g_3 \cdot s_3 \cdot C_{int}(1)$.

$$h_2 = f_2 \cdot g_2 = 7.16 = \frac{s_3 g_3 C_{int}}{s_2 g_2 C_{g1}} \cdot g_2 = \frac{s_3}{s_2} g_3 = 7.16$$

We get $s_3 = 19.4$. Then we have the final inverter.

Finally, $s_4 = 140$.

Now all the gates are sized, and I can use the last gate to assess if the calculations are correct.

$$h_4 = f_4 = \frac{C_L}{s_4 C_{in}} = \frac{2pF}{140 \cdot 2pF} = \frac{1000}{140} = 7.14$$

s > 3 > 2

$$\tau_p = \tau_{p0}(1 + h_{opt}) + \tau_{p0}/3 + h_{opt} + \tau_{p0}(2 + h_{opt}) + \tau_{p0}(1 + h_{opt})$$

The overall propagation delay is:

All the delay contributions have in common the effort delay.

What can we do to increase the number of stages without changing the function Y?

The only element with logical effort 1 is the inverter. If we add a couple of inverters in series we are not changing the function, nor F nor G, so H stays the same. If we look at formula x, $h_{opt} = \sqrt[N]{H}$, so maybe if we increase N and H is the same, we get a benefit in terms of delay.

Optimum number of stages

- For a given load and a given input capacitance of the first stage, the delay of the optimum sized chain is:

$$\tau_{p,opt} = \tau_{p0} \left[\left(\sum_{j=1}^N p_j \right) + N \frac{\sqrt[N]{H}}{\gamma} \right]$$

- The delay is minimum if the number of stage can be chosen:

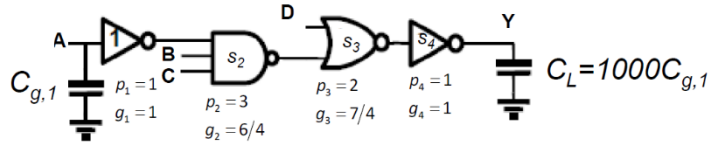
$$\begin{cases} h_{opt} = 3.6 \\ (h_{opt})^N = H \Rightarrow N_{opt} = \frac{\ln(H)}{\ln(3.6)} \quad \mathbf{x} \end{cases}$$

- In order to reach the optimum number N_{opt} it's possible to insert inverters (i.e. buffer). Remember: inverters feature $g_{inv} = 1$!

If we add inverters in the circuit, in even number, we are not changing function nor path effort, because an inverter has $g = 1$, we are acting on N.

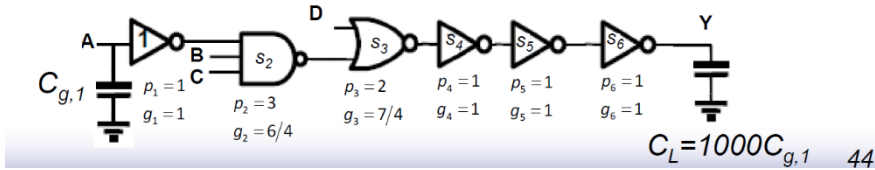
So we minimize the delay if we can change the number of stages so that we get a stage effort of 3.6, i.e. the product of the fanout and logical effort. The formula is x.

Buffering a chain adding inverters



$$N_{opt} = \frac{\ln(H)}{\ln(3.6)} \cong 6.1$$

In order to get N=6, we can add two inverters at the end of the chain and then optimize the sizes of all the gates



Since $N_{opt} = 6.1$, we need to add a couple of inverters, which must be properly sized so that each stage has the effort h_{opt} . Can we do better than this with this heavy load?

7.16 is much larger than 3.6, which is the best stage effort for $\gamma = 1$. So increasing the number of stages we can improve h_{opt} and further minimize the propagation delay.

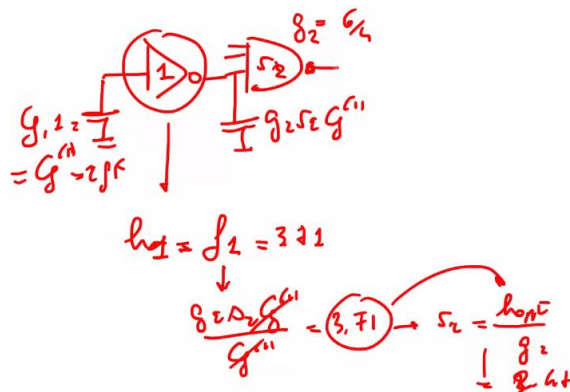
In any case, the product of all the stage efforts has to be H, that is in the h_{opt} case, $(h_{opt})^N = H$. From this we can get the optimum number of stages N_{opt} .

Which is the stage effort that minimizes the propagation delay? More or less 3.6 with 6.1 stages, but we have to choose an integer number of stages. So $N_{opt} = 6$, and $h = 3.71$.

The question is where to put the couple of inverters. We can put them wherever we want, but the best solution is to put them at the end of the chain.

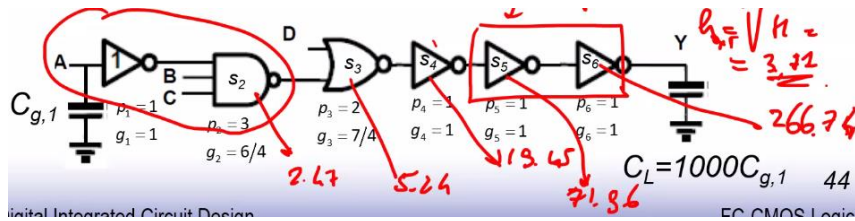
Indeed, when we size the circuit the size increases progressively from the input to the output of the chain, so **better to have simpler gates in the final positions and more complex gates at the beginning**.

Now we are left with the sizing of the circuit. To do so, we have to consider that each stage must have the stage effort of h_{opt} . Let's start with the first gate.



As for $s_2, h_2 = f_2 * g_2 = h_{opt} = 3.71$.

$f_2 = (s_3 * g_3 * C_{g(1)}) / (s_2 * g_2 * C_{g(1)})$ and the only unknown is $s_3 \rightarrow s_3 = 5.24$.



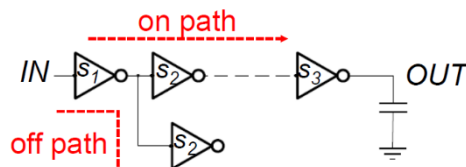
The ratio between two contiguous gates' sizes is 3.71.

So far we considered a chain of gates, but we might add branches

ADDING BRANCHING EFFORT

- ❑ It can be addressed considering a branching effort, **b**
- ❑ A stage that drives also parallel paths has a branching effort:

$$b = \frac{C_{on_path} + C_{off_path}}{C_{on_path}}$$



$$b_1 = \frac{s_2 C_g^{(1)} + s_2 C_g^{(1)}}{s_2 C_g^{(1)}} = 2$$

Let's consider a simple branching where the branch has the same gate than in the main path and with the same size. This branching can be treated defining for the driving gate a branching factor. The driving gate is the inverter s1 in their case, which has to drive two equal paths.

The **branching effort** is **b**, that in the case of s1 is b1. Basically it indicates how many capacitances we are adding with the branching with respect to the case without branching.

If we have a third path in parallel, still with size s2, the branching effort is 3, because we are driving 3 equal capacitances. The on path is the critical path from the input to the output where we want to minimize the delay. In this case the branching effort corresponds to the number of gates we are driving in parallel, because they have the same size.

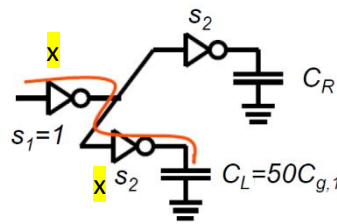
In general, considering the branching effort, the propagation delay can be written as:

$$\tau_{p,j} = \tau_{p,j} \left(f + \frac{f b}{g} \right)$$

That, in case of branching, adds the term **b** at the numerator $\rightarrow f \cdot g \cdot b$, where **f** is the fanout along the path and **b** is the branching effort.

$f \cdot b$ is exactly, for a gate, the overall external capacitance divided by the C_g of the gate we are considering, so it is the real fanout, that has been decomposed in fanout along the path **f** and branching effort **b**.

Example – 1



Path branching effort:

$$B = \prod_{j=1}^N b_j$$

1. Path electrical effort: $F=50$
2. Path logical effort: $G=1$
3. Path branching effort: $B=2$
4. Path effort: $H=FG B=100$
5. Optimum stage effort: $h_{opt} = \sqrt[3]{H} \cong 10$
6. Sizes: $s_1=1, s_2=5$
7. Minimum delay:

$$\tau_{p,opt} = \tau_{p0} \left[\left(\sum_{j=1}^N p_j \right) + N \frac{h_{opt}}{\gamma} \right] = 22\tau_{p0}$$

We have a minimum size inverter that drives, in the reference technology of 0.25um from INTEL, two paths with two inverters of size s2. The orange one is the path I want to minimize.

In case of multiple nodes along the branch, we can define a **path branching effort**, which is the product of all the branching efforts of all the gates involved in the branching. In our case we have just $b_1 = 2$.

$F = C_L / C_{g,1} = 50$ (ratio of capacitances in the critical path)

$G = g_1 * g_2 = 1$

$B = b_1 = 2$ (in this case only of the first inverter that drives two equal paths)

$h_{opt} = 10$ (optimum stage effort)

Inverters x both belong to the critical path, so they must have $h_1 = h_2 = 10$. The first inverter has a stage effort $h_1 = f_1 * b_1 = 2 * s_2 * C_{g,1} / C_{g,1} = 2 * s_2$, because the C_g of the second inverter is $s_2 * C_{g,1}$. So $s_2 = 5$.

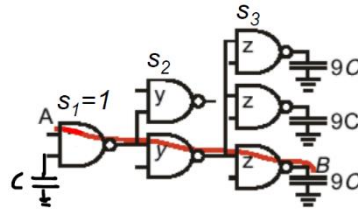
Hence the two inverters in the fork have the same size of 5. Let's now assess the propagation delay of the optimized path. The propagation delay of the first inverter is the one of an inverter that drives two inverters of size 5, so $\tau_{p1} = 11 * \tau_{p0}$.

The propagation delay of the second stage is $11 * \tau_{p0}$, so in this case it happens that they have the same delay, but it is only because we are using inverters.

Let's remove the upper branch. With respect to $22 * \tau_{p0}$ the delay decreases because we are like removing a capacitance, $\tau_{p1} = 6 * \tau_{p0}$, and the overall delay is $17 * \tau_{p0}$.

If we could change s2, still having 2 inverters, 5 is not the best size, but $7 = \sqrt{50}$. Now $\tau_{p1} = 8 * \tau_{p0}$ and also the second inverter, so overall $\tau_{p} = 16 * \tau_{p0}$.

Example – 2



1. Path electrical effort: $F=9$
2. Path logical effort: $G=(5/4)^3$
3. Path branching effort: $B=2 \cdot 3=6$
4. Path effort: $H=FGB=105.5$
5. Optimum stage effort: $h_{opt} = \sqrt[3]{H} \cong 4.7$
6. Sizes: $s_1=1, s_2=1.9, s_3=2.4$
7. Minimum delay:

$$\tau_{p,opt} = \tau_{p0} \left[\left(\sum_{j=1}^N p_j \right) + N \frac{h_{opt}}{\gamma} \right] = 20.1 \tau_{p0}$$

Longest path is from A to B, that is the critical one. First branch has $b_1 = 2$, and the second one $b_2 = 3$. For the NAND, $p = 2$ and $g = 5/4$ if the size is (3);(2). Hence $G = (5/4)^3$, because we have three NANDs in series. Every stage in the critical path must have a stage effort of h_{opt} to minimize the propagation delay of the critical path.

$$h_1 = 4.7 = f_1 b_1 g_1 \sqrt[3]{\frac{F}{b_1}}$$

$f_1 = s_2/s_1$, and $s_2 = (s_1 \cdot h_{opt}) / (b_1 \cdot g_1)$, and we get $s_2 = 1.9$. The same procedure has to be repeated for the second gate.

$$\begin{aligned} \dots & \\ h_2 &= f_2 b_2 g_2 \sqrt[3]{\frac{F}{b_2}} \\ &= \frac{5}{5} b_2 g_1 \\ s_3 &= \frac{h_{opt} \cdot 5}{b_2 g_2} \\ &= 2.4 \end{aligned}$$

The gates in the critical paths have in common the effort delay, and moreover they are equal, so the three gates have the same delay. The delay of each gate is $\tau_{p0} \cdot (2 + h_{opt}) = 6.7 \cdot \tau_{p0}$.

The final delay of $20.1 \cdot \tau_{p0}$ of the critical path is not the optimal one, because h_{opt} is 4.7 and not 3.6, so we can add buffers in the chain to minimize the delay. Since 4.7 is close to 3.6, let's try adding two inverters.

With 5 gates, $h_{opt} = \sqrt[5]{H} = 2.5$.

If I size the gates in the path now, I get a delay that is:

$$\tau_p \cong \tau_{p0} [2 + 2 + 1 + 2 + 1 + 5 \cdot h_{opt}]$$

$$\cong \tau_{p0} [8 + 12.5]$$

In the end I get $20.5 \cdot \tau_{p0}$, so I don't optimize the delay. This because 4.7 was already quite close to 3.6.

Let's suppose we can add just one inverter, and not a couple. $h_{opt} = \sqrt[4]{H} = 3.2$. Now we have a benefit.

$$\tau_p \approx \tau_{p0} \left[2 + 2 + 2 + 2 + 4 \frac{\log 2}{s} \right]$$

With a final result of $19.8 \cdot \tau_{p0}$, which is quite close to $20.1 \cdot \tau_{p0}$. So better not to use it to save area and power consumption.

POWER CONSUMPTION IN A GENERIC GATE

- Consider to switch a gate for N clock cycles:

$$E = C_L V_{DD}^2 n_{0 \rightarrow 1}$$

- E is the energy drawn by the supply for N clock cycles
- $n_{0 \rightarrow 1}$ is the number of 0→1 transitions at the output

- If the data at the input changes with a rate f_{ck} :

$$\alpha_{sw} = \lim_{N \rightarrow \infty} \left(\frac{n_{0 \rightarrow 1}}{N} \right) = P(0)P(1) \quad \text{switching activity}$$

$$P = C_L V_{DD}^2 \alpha_{sw} f_{ck}$$

- **The switching activity depends on the logic function!**

The product of the switching activity and bitrate f_{ck} is the pullup frequency. We want to extend this theory to a FC-CMOS gate.

NOR GATE

A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

Assume **signal probabilities**

$$p_{A=1} = 1/2$$

$$p_{B=1} = 1/2$$

Then **transition probability**

$$p_{0 \rightarrow 1} = p_{Out=0} \times p_{Out=1}$$

$$= 3/4 \times 1/4 = 3/16$$

If inputs switch every cycle

$$\alpha_{0 \rightarrow 1} = 3/16$$

I want to assess the power consumption spent by the PS generator. The output is typically 0 (see truth table) and we can try to assess the power consumption assuming equiprobable input signals and an output load C.

$$\text{Alpha}_{sw} = P(0) \cdot P(1) = 3/4 \cdot 1/4 = 3/16.$$

For the same capacitance, it seems that the inverter has a larger power consumption than the NOR, because the switching activity is bigger.

NAND GATE

The output is more likely 1 than 0. Again, if we consider an equiprobable signals in output to A and B, the 4 cases are equiprobable at the output. The switching activity is still 3/16, so the power consumption is smaller than in the inverter case for the same load capacitance.

Typically, the more complex the function the smaller the switching activity.

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Assume signal probabilities

$$p_{A=1} = 1/2$$

$$p_{B=1} = 1/2$$

Then transition probability

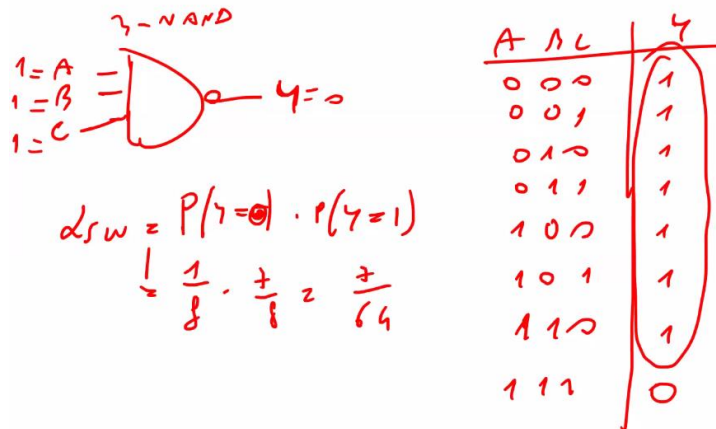
$$p_{0 \rightarrow 1} = p_{Out=0} \times p_{Out=1}$$

$$= 3/4 \times 1/4 = 3/16$$

If inputs switch every cycle

$$\alpha_{0 \rightarrow 1} = 3/16$$

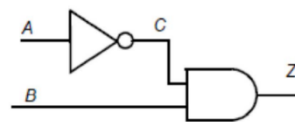
Let's consider the case of a 3 inputs NAND gate.



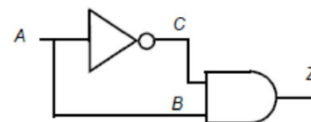
This switching activity has been computed assuming equiprobable signals, so P(0) and P(1) are uncorrelated.

RECONVERGENT FAN-OUT

We have two cases with inputs connected in different manners.



(a) Logic circuit without reconvergent fanout



(b) Logic circuit with reconvergent fanout

❑ Case (a): $P(Z=1) = P(B=1) \cdot P(C=1) = 0.25$

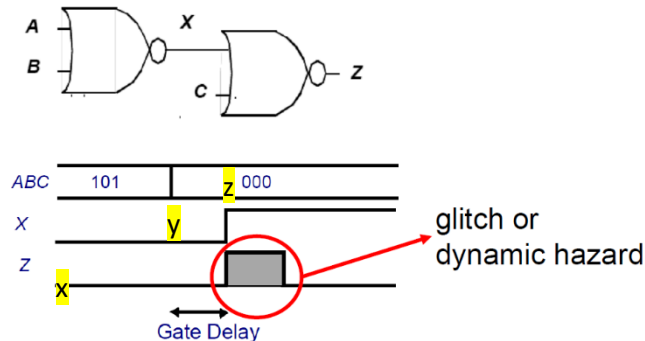
❑ Case (b): $P(Z=1) = P(B=1) \cdot P(C=1|B=1) = 0$

- Need to use conditional probability to account for inter-signal correlations
- CAD tools best to perform such complex analysis

If we follow the analysis carried on so far, considering equiprobable signals (random signals), we can focus on the circuit on the left. In output of the AND, $P(0) = 3/4$ and $P(1) = 1/4$, so switching activity is $3/16$ as before.

If we consider the circuit on the left, A is equiprobable, but B is the complement of C, so the output of the AND is always 0, so the switching activity is 0 because $P(1) = 0$. This is a condition of **reconvergent fanout**.

GLITCH IN STATIC CMOS LOGIC



Due the different delay of the paths (C and X), the result is correct but there is an additional contribution to the power consumption 53

It is a form of power consumption that can occur in complex circuit that is not taken into account in the classical P computation based on the switching activity. Inputs A and B are far from the output with respect to the input C. We are in presence of a **chain**.

Let's take $A = 1, B = 0$, so $X = 0$, and $C = 1 \rightarrow Z = 0$. Now A transition to 0 and also C, but they are in two different points of the circuit, so they have different delays. Let's suppose that $\tau_{p1} = \tau_{p2}$ for the two gates. We can say that after τ_{p1} we have a transition in output of the first gate, so X has transitioned from low to high. In the mean time, in reality at $t = 0$, also C has transitioned, and after a delay τ_{p2} the Z transitions to low.

The logic function implemented by the network is:

$$\begin{aligned} \overline{A+B+C} &= \overline{\overline{\overline{A+B+C}}} \\ &= \overline{(\overline{A+B}) \cdot \overline{C}} \\ &= \overline{(\overline{A+B})} \cdot \overline{\overline{C}} \end{aligned}$$

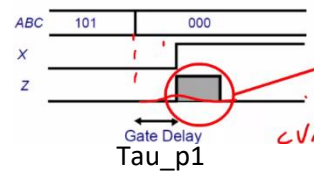
Let's consider the transition 101 to 000. At the beginning, x, I have $Z = 0$ in output, as well as at y, that has $A = B = C = 0$. At z, X goes to 1, C is already 0.

Due to the different propagation delay of the two different paths, even if after the glitch the output is $Z = 0$ as expected, I have a **glitch**, that is not predicted by the logical function. In fact, from the logical standpoint the output has never to transition, but it has indeed a glitch, which, if we consider the final capacitance, charges the capacitance, with an charge spent that is $C \cdot V_{dd}$. Then it is eventually discharged.

This glitch power consumption occurs typically when we have chains of gates with inputs that can have different propagation delays.

To equalize the delay, we can think of adding gates, e.g. a couple of inverters, in the path of the C input. In this case the two inputs of the final NOR gate transition simultaneously, so the output will always be 0. However, this solution has a higher area and power consumption.

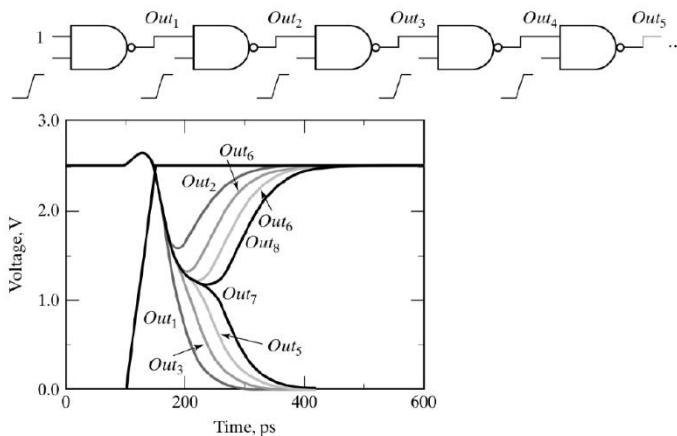
Another possible solution is to slow down the second NOR, having hence $\tau_{p2} > \tau_{p1}$. So the τ_{p1} is the same, and the second NOR takes a lot of time to transition, so X has the time to change and the glitch is almost null.



If we suppose that the peak voltage reached by the glitch is ΔV , the charge spent by the power supply generator is no more $C \cdot V_{dd}$ but $C \cdot \Delta V$, so the energy is $C \cdot \Delta V \cdot V_{dd}$. If this is repeated with a frequency f and ΔV is small, this consumption is decreased.

Furthermore, **the glitch is a logical mistake**, because from a logical standpoint the signal Z should remain 0 all the time, while here it goes to almost 1 for a while.

Example of additional power consumption

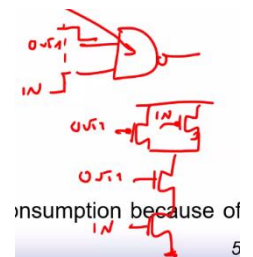


Even outputs contribute to the power consumption because of paths with different delay

54

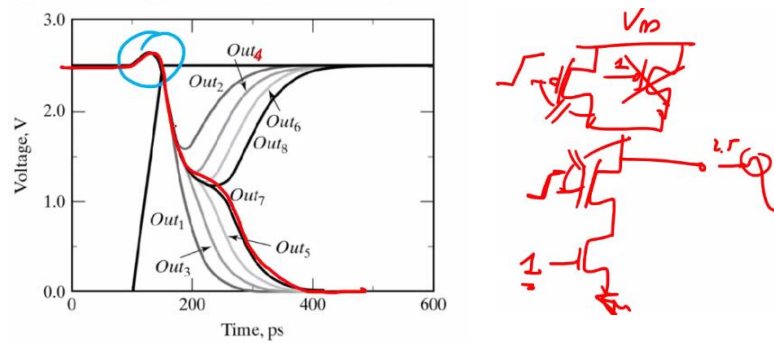
Again, we have a slow path and a fast one. At $t = 0$ the signal on the fast path changes from 0 to 1. The outputs in all the odd positions transition to 0, the ones in even positions remain 1. Node out1 and out3 and so on take some time to transition with respect to the fast node. Out1 transitions with a delay that is $\tau_{p,p}$ across the first gate.

As for out2, from a logical standpoint it has to remain 1. If we depict the situation for the second NAND, one input transitions immediately, the other one takes some time. For a brief amount of time the signals are both 1 in input of the NAND due to the different delays, so the PDN is activated, and the output is discharged. This is something unwanted. However then the output goes back to 1.



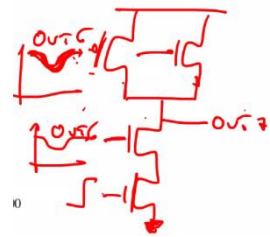
The thing is, as soon as we move in the gate, the glitch is increasing due to the larger delay we are accumulating. Since we have a pull down for a short time and then a pull up, this is related to power consumption.

Output 7 vs output 1



We start from 2.5V, at a certain point the output reaches 2.7V. This is due to **charge injection**, since we have a forward path from the input to the output with the Cgd capacitances. On a fast transition, the capacitance behaves as a short, so the signal passes to the output due to the short of Cgd, but then as soon as we have the plateau of the transition it is the mosfet that acts. This is due to the fact that we are using real switches with parasitic capacitances.

As a last remark, the out7 has also a sort of bouncing in the middle, a problem that worsens from out1 to out7. This because out6 is not always 1, but it has a glitch. So the pMOS in the PUN turns partially on for a small period of time, and this is a conflict between the pull down and pull up network. So we have transconduction current. So the bump is in part due to the fact that we are turning on the pMOS transistor in the PUN, and in part due to the fact that if we decrease the gate voltage of a transistor the Req of the transistor increases (it draws less current), so the PDN is becoming more resistive, and so we are slowing down the pull down. But then the out6 voltage goes up again and the situation is restored, the PDN is fast again.

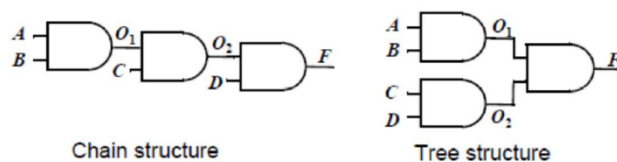


PRINCIPLES FOR POWER REDUCTION

- Reduce gate size, i.e., physical capacitance
- Reduce voltage (at the cost of reduced time performance)
- Reduce glitch (equalizing delay)
- Reduce switching activity
 - Logic reconstruction
 - Input reordering
 - Exploiting parallelism

Glitches reduction

The best way to reduce glitches is to implement a **tree structure**. In this case the propagation delay for all the paths is the same, so we can reduce the possibility of glitches because of the same propagation delays.



	O_1	O_2	F
p_1 (chain)	1/4	1/8	1/16
$p_0 = 1 - p_1$ (chain)	3/4	7/8	15/16
$p_{0 \rightarrow 1}$ (chain)	3/16	7/64	15/256
p_1 (tree)	1/4	1/4	1/16
$p_0 = 1 - p_1$ (tree)	3/4	3/4	15/16
$p_{0 \rightarrow 1}$ (tree)	3/16	3/16	15/256

The output F has the same switching activity in both cases but not the internal nodes (chain better than tree)

RATIOED LOGIC

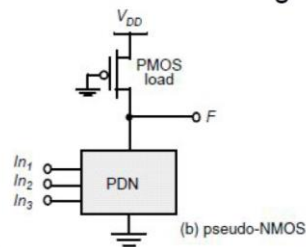
The FC-CMOS logic is ratioless, meaning that the static properties of the logic don't depend on the sizing of the pMOS and nMOS, $V_{oh} = V_{DD}$ and $V_{ol} = 0V$ whatever the sizing, only V_m is a function of the beta factor.

- ❑ Ratioed logics are such that their static properties depend on the relative size (i.e., aspect ratio) of the pull-up and pull-down network
- ❑ We shall analyze two ratioed logics:
 1. Pseudo-NMOS logic
 2. Differential Cascode Voltage Switch Logic (DCVSL)
- ❑ The goal is to build gates faster and smaller than FC-CMOS gates

The big issue in FC-CMOS logic is the usage of the pMOS transistor, which are big to balance the different process transconductance with respect to the nMOS. If they are bigger, their capacitance is larger and so the propagation delay is bigger. The objective of ratioed logics is to implement smaller and faster gates to avoid bulky transistors.

PSEUDO NMOS LOGIC

- ❑ FC-CMOS logic drawbacks:
 1. 2N transistors
 2. Large input capacitance, thus large logical effort
 3. Large intrinsic capacitance
- ❑ Let's substitute the cumbersome pull-up network of a FC-CMOS gate with a single PMOS transistor with the gate tied to ground (i.e., always on)
- ❑ Smaller area, smaller input and output capacitance

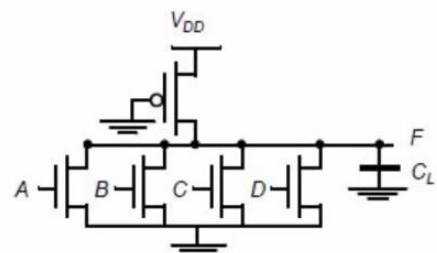


In a pseudo nMOS logic the PUN changes, with just one single pMOS transistors, whose gate is to ground, so it is always on.

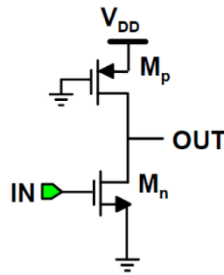
This has the benefit that the input is connected just to nMOS transistors, so **the capacitance is reduced both at the input and the output**. This leads to a circuit with smaller area also.

Let's suppose we want to implement a two inputs NAND gate with a pseudo nMOS approach. The PDN is the same of the FC-CMOS NAND, and the PUN is with a single pMOS that can be minimum size (1).

This is very good in terms of parasitic capacitances. The benefit is even greater in the case of a NOR gate. The one in the image is the 4 input NOR gate. In fact, we are avoiding the large size (12) pMOS of the FC-CMOS 4 input NOR.



In order to analyze the properties of the logic, let's consider the pseudo nMOS inverter as the reference to analyze the properties of this logics.



Main drawbacks:

1. $V_{OL} > 0V$ (but $V_{OH} = V_{DD}$)
2. Static power consumption for a $V_{IN} = V_{DD}$
3. Asymmetric propagation delays
4. Trade-off between static and dynamic performances

$V_{OL} > 0$ in pseudo nMOS, while in FC-CMOS it is exactly 0. Instead, $V_{OH} = V_{DD}$. We know that $V_{OL} = f(V_{OH})$ and $V_{OH} = f(V_{OL})$.

Let's suppose that $V_{in} = V_{OL} < V_{tn}$, threshold voltage of the nMOS. If so, the nMOS is off, and also the current I_p in the pMOS is 0 (static current at steady state). So to have this, $V_{ds,p} = 0V$. So $V_{OH} = V_{DD}$.

Let's consider now $V_{OH} = V_{in} = V_{DD}$. So we have a current from V_{DD} to ground. This means that the nominal output voltage corresponding to a logic 0 is not 0V, otherwise there would be no current in the nMOS, $I_n = 0$. So the nominal output voltage is not 0V.

The only way to keep the output voltage close to ground, if we suppose to have a large pMOS and a smaller nMOS, the pMOS is larger and wants to draw a large amount of current, while the nMOS is smaller and wants to draw a smaller amount of current, so the only possibility is that the output voltage increases pushing in ohmic region the pMOS transistor. So to have the output voltage close to ground, the nMOS must be large and the pMOS off, so that the nMOS is pushed in ohmic.

So the nMOS must be sized larger (i.e. more conductive) than the pMOS transistor. Moreover, when the input voltage is high, the output voltage is not zero (worsening of the noise margins), so we have a static power consumption, a static current. It is not a subthreshold current.

Another drawback is that, to balance drawbacks 1 and 2, the nMOS must be sized bigger than the pMOS, so eventually the pMOS is minimum size and the nMOS is bigger \rightarrow asymmetric propagation delays, $\tau_{pLH} \gg \tau_{pHL}$.

Another drawback from this analysis is that there is a trade off between static and dynamic performances. If we want to decrease the pull up time we need to increase the pMOS size, but at the same time the power consumption is increased and also V_{OL} , so the reliability is worsened.

VTC of pseudo nMOS inverter

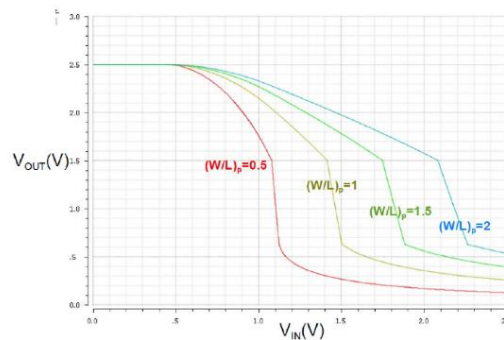
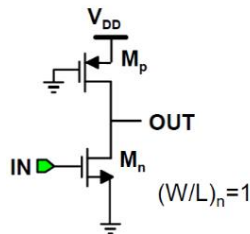
Let's suppose we can change the pMOS aspect ratio, and the nMOS is fixed with a size of 1. $(W/L)_p = 0.5$ can be obtained with an increased length, not using the minimum one ($L = 0.5 \mu m$ and $W = 0.25 \mu m$). Moreover, the table stops at $(W/L)_p = 2$ because if we further increase the aspect ratio, V_{OL} increases further and further up to the point where it is bigger than $V_{DD}/2$, and we don't have an inverter anymore.

Increasing the pMOS aspect ratio the nominal output voltage V_{OL} increases. $P_{low} = V_{DD} \cdot I_{DS}$ when $V_{out} = V_{OL}$, static power consumption. So increasing the dimension of the pMOS leads to an increase in the static power consumption, while in a FC-CMOS gate it is 0W.

As for the propagation delays, the propagation delay from low to high becomes smaller because the pMOS size is increased.

For the high to low transition we have the nMOS transistor involved, but its size is fixed. Why is the propagation delay increasing? This is because if we increase the pMOS size we are increasing the output capacitance, but there is also another reason. The pMOS transistor is on, so the current that discharges the capacitor through the nMOS is the current of the nMOS minus the one of the pMOS, which is on, according to the KCL. So the current that discharges the output capacitor is not the one of the nMOS, but **we are reducing the effective current discharging the output capacitor → hampering effect**.

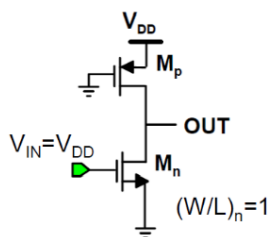
This effect can be seen as an increase of the pull-down resistance with respect to the equivalent resistance of the pMOS.



$(W/L)_p$	V_{OL} (V)	V_M (V)	P_{low} (μ W)	τ_{pLH}	τ_{pHL}
0.5	0.128	1.1	74	39.6ps	8.5ps
1	0.26	1.41	147	17.6ps	11.9ps
1.5	0.395	1.63	218	12.75ps	18.7ps
2	0.537	1.77	287	9.6ps	32.7ps

Low output voltage and static power

Let's consider $V_{OH} = V_{DD}$ at the input of the nMOS; the output goes to logical "0". Both transistors are on, and $I_p = I_n$ at steady state. If V_{OL} is close to ground (not properly zero), the nMOS is ohmic, and the pMOS is in velocity saturation (V_{ds} almost V_{DD}). In the analysis it is neglected the channel modulation effect (the term with $1 + \lambda \cdot V_{ds}$).



Let's assume M_n in ohmic region and the M_p in velocity saturation region:

$$k'_n \left(\frac{W}{L} \right)_n \left[(V_{DD} - V_{Tn}) V_{OL} - \frac{V_{OL}^2}{2} \right] = k'_p \left(\frac{W}{L} \right)_p \left[(V_{DD} - V_{Tp}) V_{DSATp} - \frac{V_{DSATp}^2}{2} \right]$$

$$V_{OL} \cong \frac{k'_p \left(\frac{W}{L} \right)_p \left[(V_{DD} - V_{Tp}) V_{DSATp} - \frac{V_{DSATp}^2}{2} \right]}{k'_n \left(\frac{W}{L} \right)_n [(V_{DD} - V_{Tn})]} \cong \frac{k'_p \left(\frac{W}{L} \right)_p V_{DSATp}}{k'_n \left(\frac{W}{L} \right)_n}$$

$$P_{low} = V_{DD} I_{LOW} = V_{DD} k'_p \left(\frac{W}{L} \right)_p \left[(V_{DD} - V_{Tp}) V_{DSATp} - \frac{V_{DSATp}^2}{2} \right] [1 + \lambda_p (V_{DD} - V_{OL})]$$

The unknown is V_{ol} , which is close to ground. Because of this, $V_{ol}^2/2$ can be neglected and we get a linear equation with respect to V_{ol} . Since also $V_{dsat}^2/2$ is small, we can neglect it, and furthermore V_{tp} almost V_{tn} .

In the end V_{ol} is a function of the beta factor, so to have a small output voltage the pMOS must be smaller than the nMOS, we want beta small to have V_{ol} close to ground. For $\beta = 1$, V_{dsat} is 1V, and $k_p/k_n = 1/3.5$, so $V_{ol} = 290mV$.

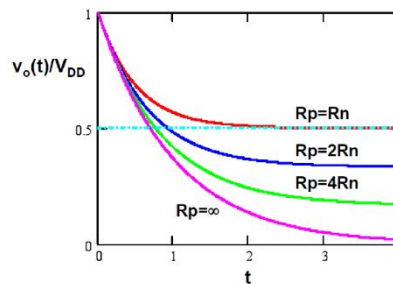
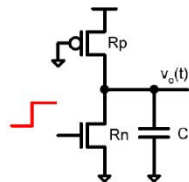
So V_{ol} is a function of the beta factor, so of the relative size of the pMOS and nMOS transistors → ratioed logic. Moreover in the low output state we have a power consumption, and the current is fixed and depends on the aspect ratio of the pMOS if we neglect the channel modulation effect. **Increasing the pMOS aspect ratio leads to an increase in the current and in the static power consumption P_{low} .** This is another reason to keep $(W/L)_p$ small.

Propagation delay

It is asymmetric if the pMOS is smaller than the nMOS one, so its driving capabilities are smaller. We cannot counterbalance the smaller process transconductance increasing the size, and this leads to a bigger $R_{eq,p} > R_{eq,n}$ and the PU time is bigger than the PD time, so larger τ_{pLH} .

In this circuits the PD is difficult to be assessed because the nMOS is hampered by the pMOS transistor with its current.

- ❑ Asymmetric propagation delays
- ❑ Pull-up time due to just the (weak) PMOS transistor
- ❑ Pull-down time due to the (strong) NMOS transistor hampered by the PMOS one



In the plot the pMOS aspect ratio is increasing from top chart to upper chart, because R_p decreases. $R_{eq,p}(1) = 31 \text{ k}\Omega$, and the generic $R_{eq,p} = 31\text{k}\Omega/(W/L)_p$. Instead, $R_{eq,n} = 13 \text{ k}\Omega$.

Increasing the pMOS aspect ratio decreases the pMOS equivalent resistance and so V_{ol} increases. So V_{ol} is inversely proportional to the $R_{eq,p}$.

The propagation delay instead increases (time needed to cross $V_{dd}/2$) as we increase the aspect ratio. If $R_p = R_n$, the pull down propagation delay is infinite, because physically the capacitor is not discharged, there is no current available for the capacitor to discharge. So we cannot size the pMOS to have the same R_{eq} of the nMOS. Keeping the pMOS small has instead a benefit in terms of propagation delay from high to low at the output, reducing the hampering effect.

The $R_{eq,p} = 31 \text{ k}\Omega$ was assessed with the pMOS in velocity saturation. But in this case the operating region of the transistor, at the beginning is, with $V_{out} = V_{dd}$, ohmic. So $31 \text{ k}\Omega$ is an approximation, a very rudimental estimate of the equivalent resistance of the pMOS.

With this rudimental analysis let's try to quantify the pull-down delay, τ_{pHL} .

Pull-down delay

We want to take into account the pMOS hampering effect. Currents I_p and I_n are related to the conductance. It is like the equivalent conductance of the pull down is the conductance of the nMOS minus the one of the pMOS. Then the conductance is $1/R_{eq}$.



Stated this, we can say that the equivalent resistance of the pull down is the difference between the inverse of the equivalent resistances. So the effective pull down resistance is the equivalent resistance of the nMOS divided by a term $1 - \text{something}$. The 'something', if $(W/L)_p < (W/L)_n$, is smaller than 1. So we are like magnifying the $R_{eq,n}$ (the denominator is < 1), that is the same of saying having less current for the pull down.

- ❑ Think in terms of current that discharges the output capacitance
- ❑ The discharge current is the difference between the NMOS current and the PMOS transistor one
- ❑ Current means conductance, thus the equivalent pull-down conductance is reduced as:

$$G_{pd} = G_{eq,n} - G_{eq,p} \quad \longrightarrow \quad R_{pd} = \left(\frac{1}{R_{eq,n}} - \frac{1}{R_{eq,p}} \right)^{-1} = \frac{R_{eq,n}}{1 - \frac{R_{eq,n}}{R_{eq,p}}}$$

So numerically we get as below.

Assume the inverter implemented in the INTEL CMOS 0.25- μm CMOS process

$R_{eq,p} = 31\text{k}\Omega$
 $R_{eq,n} = 13\text{k}\Omega$
 $C_{int} = 1\text{fF}$

$R_{pd} = \frac{R_{eq,n}}{1 - \frac{R_{eq,n}}{R_{eq,p}}} = 22.4\text{k}\Omega$

$\tau_{pLH} = \ln(2)R_{eq,p}C_{int} = 21.5\text{ps}$

$\frac{5\text{ k}\Omega}{(17.6\text{ ps})}$

$\tau_{pHL} = \ln(2)R_{pd}C_{int} = 15.5\text{ps}$

(11.9 ps)

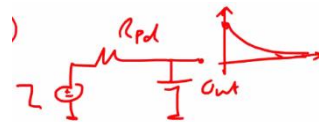
What about the capacitance? In the reference FC-CMOS inverter we had $C_{int} = 2\text{ fF}$, now $C_{int} = 1\text{ fF}$, because the sum of aspect ratio is 2 (in the reference inverter is 4). At the input A instead we have 0.5 fF in this case. So **smaller intrinsic and gate capacitances**.

Let's assess PU and PD times, as in the image above, with $C_{int} = 1\text{ fF}$.

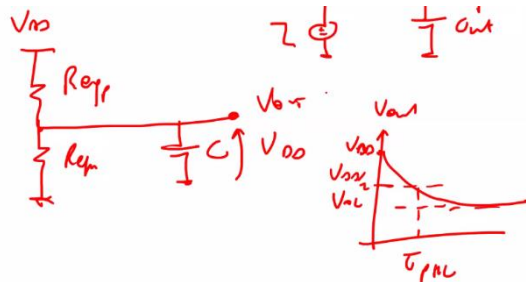
In the simulation we have a smaller propagation delay because the output voltage in the LH transition is not starting from 0V , but from a higher voltage (0.26V).

As for the pull down, the model is the same but we use the effective pull down resistance. Again, the simulation is better but because the approximations are very rudimental.

Estimating the HL time with the formula $\tau = RC$ is like using the following circuit to estimate the delay.



However, this model is not accurate for our circuit, we need to revert to another circuit. The plateau value is not 0, but V_{OL} .



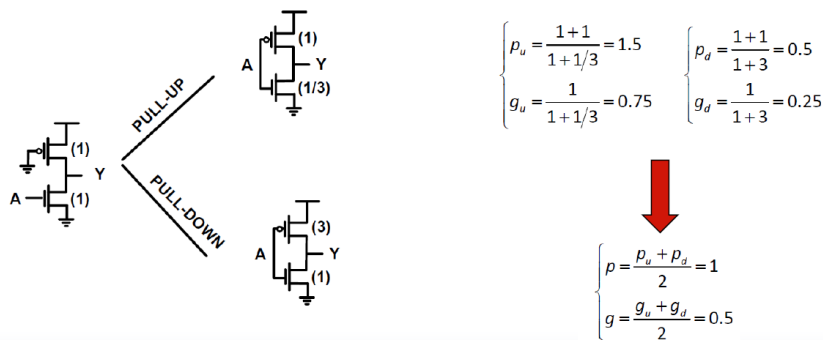
The other approximation that leads to a non-negligible error is that the pole of the circuit above is $\tau = C \cdot (Req,p \parallel Req,n)$, while we are not assessing R_{pd} as a parallel resistance.

Logical effort (g) and intrinsic propagation delay factor (p)

Let's assess them for the pseudo nMOS logic. They are not just one value, but they are different for the PU and PD transitions, because the transitions are asymmetric. The pseudo nMOS gate must be compared with an equivalent FC-CMOS inverter featuring the same equivalent resistance for the considered transition. $Req,p(1) = 31 \text{ k}\Omega$, so for the pull up the pMOS is (1), and the nMOS is (1/3) in the FC-CMOS inverter. Let's neglect also the hampering effect, considering $R_{pd} = Req,n$.

For the pull down, the $Req,n(1) = 13 \text{ k}\Omega$, so the inverter must have the nMOS with size (1) and the pMOS (3).

- ✓ Logical effort and intrinsic propagation delay (i.e., intrinsic delay factor) are different for pull-up and pull-down transitions
- ✓ Compare the gate with an FC-CMOS inverter featuring the same equivalent resistance for the considered transition



Pull-up

We have to consider the pseudo nmos logic and the FC-CMOS inverter on the top. The p is the ratio between the sum of the aspect ratio in the pseudo nmos gate and in the FC-CMOS inverter, so 1.5. For the logical effort g we look at the input and we do the ratio as for p. In the end $g = 0.75$.

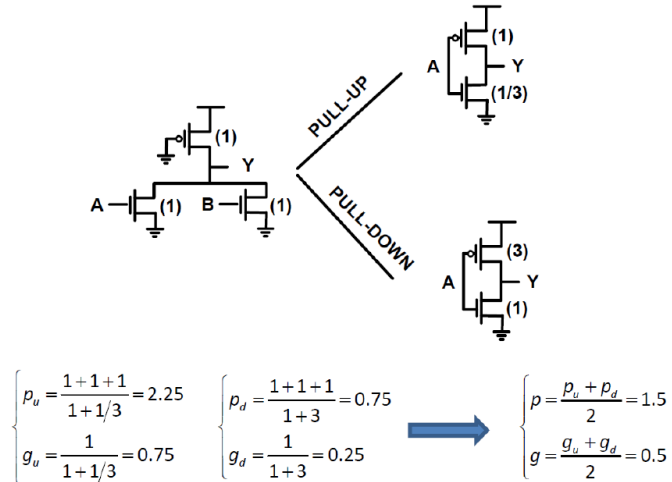
Pull-down

Same analysis as the pull-up, but with the reference inverter at the bottom.

Then the final g and p are obtained averaging them. The p is not so improved, the great benefit is in g at the input, with respect to the FC-CMOS case.

If $p = 1$, the pseudo inverter has an intrinsic propagation delay of $p \cdot \tau_{p0} = \tau_{p0}$, that is the same of a classical FC-CMOS inverter. The great benefit is in the input capacitance since g is much smaller.

Let's consider a more complex gate, e.g. a two input NAND.



For a FC-CMOS 2-NOR gate is: $p=2$ and $g=7/4$

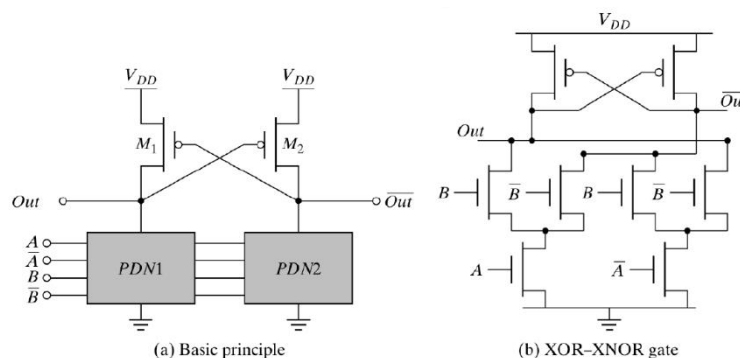
The pMOS is always considered with the minimum aspect ratio. So in the end there is a great benefit in terms of g , so of input capacitance.

The two drawbacks $V_{ol} > 0V$ and the $P_{low} > 0W$, which means static power consumption, are very bad, in particular the latter. FC-CMOS are used to implement pMOS and nMOS electrically equivalent so that there is never a connection between V_{dd} and ground at steady state, so static power consumption. So we can improve the circuit to cope with these two problems.

In analog electronics to solve this kind of problem concerning stability and bandwidth are solved with negative feedback. In digital electronics negative feedback is substituted by positive feedback, and we also need to resort to a **differential configuration**. The combination of positive feedback and differential configuration allow to solve problems.

DIFFERENTIAL CASCODE VOLTAGE SWITCH LOGIC (DCVSL)

It is the result of the combination of positive feedback and differential configuration. It is a **differential pseudo nMOS logic**. The gate is the one on the left, and it allows to have both the true function and the

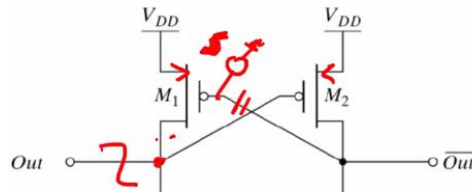


- ✓ DCVSL exploits differential inputs and a positive feedback load
- ✓ The aim is to have $V_{OL} = 0V$ and to avoid static power consumption

complement of the function in output. So we have two different PDN, which are **dual** and **complementary**, so if on the right PDN we have the NAND, on the left PDN we have the AND \rightarrow never simultaneously on. We will skip the gate on the right.

The positive feedback is in the **cross-coupled pMOS device**, also known as semi-latch.

We want to verify that it is a positive feedback. So we cut the loop and check that the signal that returns is positive. If it was an analog circuit, the pMOS would be in common source configuration.

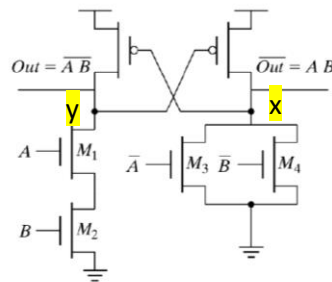


So we have two inversions in cascade and we get a positive feedback.

DCVSL sizing

The PDN1 (left) is the PDN of a NAND, while on the right we need to have the complementary PDN, so it is the PDN of an AND. To implement the AND we use the De Morgan theorem (it is the AND PDN for the complementary input signals).

Let's consider e.g. $A = 0$ and $B = 1$. The PDN1 is off, while on the PDN2 we are on and $out_bar = 0$.



- ✓ DCVSL is a ratioed logic
- ✓ The pull-down network has to be strong enough to pull-down the corresponding output node and turn the feedback on
- ✓ To turn the feedback on it's mandatory that the output voltage drops below $V_{DD} - V_{TP}$

If $B = 1$ and $A = 0$, M_1 and M_4 are off. Let's start from the condition that $out = V_{DD}$ and $out_bar = 0V$. Now A transitions from low to high, so M_3 is disabled (PDN2 is off) but M_1 is turned on. node x remains floating, at $0V$, where the capacitance was charged.

If the PDN > PUN, node y is discharged. But this means that the pMOS transistor on the right side is turned on and charged node x capacitance to V_{DD} , but if so, the pMOS on the left tends to be disabled, so node y can go to $0V$. So it is a very powerful circuit because the pMOS transistors are on when the node has to be 1, but then they are able to switch off when the node has to go to zero, thanks to the positive loop. It is not like in the pseudo-nMOS logic where the pMOS is always on.

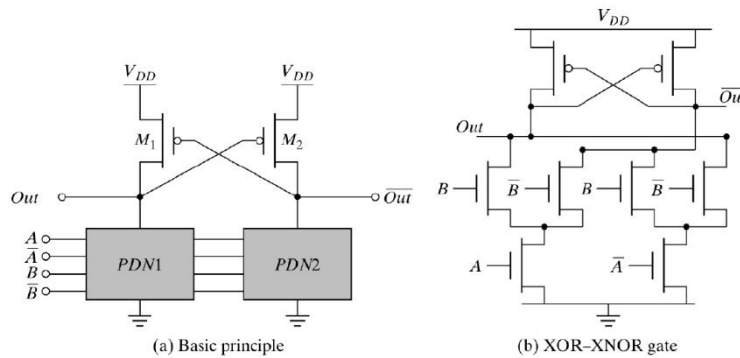
DCVSL – recap

The pseudo nMOS logic has some drawbacks:

- $V_{ol} > 0V$ and V_{ol} depends on the relative size of the pMOS and PDN \rightarrow reduced noise margins and reliability.
- Static power consumption, we have a DC static current when the output is 0.

To avoid these issues, we can resort to DCVSL. It is like resorting to a differential implementation and a positive feedback loop.

A generic gate implemented in DCVSL includes two complementary PDNs. These two are fed by the same input signals. Above we have the semi-latch (cross coupled device).



- ✓ DCVSL exploits differential inputs and a positive feedback load
- ✓ The aim is to have $V_{OL}=0V$ and to avoid static power consumption

Let's consider the gate on the right, which implements the XOR and XNOR functions. XOR goes to one if the two inputs are different $\rightarrow Y = A*B_bar + A_bar*B$.

	0	1
0	0	1
1	1	0

To implement a gate that synthesizes the XOR, I need to express it in an inverted form. So I apply a couple of bars and the De Morgan theorem or I can study the k-map.

Now I can consider the complement of this k-map, which is the XNOR, which is XOR_bar. So $Y_bar = (AB + A_bar*B_bar)$.

So now I can write Y in a complement fashion.

$$\bar{Y} = AB + \bar{A}\bar{B} \rightarrow Y = \overline{AB + \bar{A}\bar{B}}$$

This will result in the following PDN for the XOR:



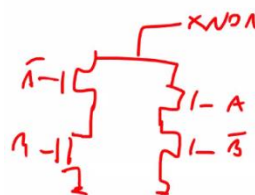
We can derive the complementary PDN. The complement of the XNOR function is the XOR function (k-map above).

$$\overline{XNOR(z)} \rightarrow z = \overline{AB + \bar{A}\bar{B}}$$

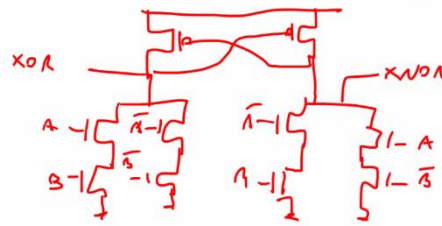
Summarizing:

$$\begin{aligned}
 XOR(z) &\rightarrow z = \overline{AB + \bar{A}\bar{B}} && z=1 \text{ if } A \neq B \\
 XNOR(z) &\rightarrow z = \overline{\overline{AB + \bar{A}\bar{B}}} && z=1 \text{ if } A=B
 \end{aligned}$$

For what concerns the PDN of the XNOR:



Then in implement the latch and I have the DCVSL.

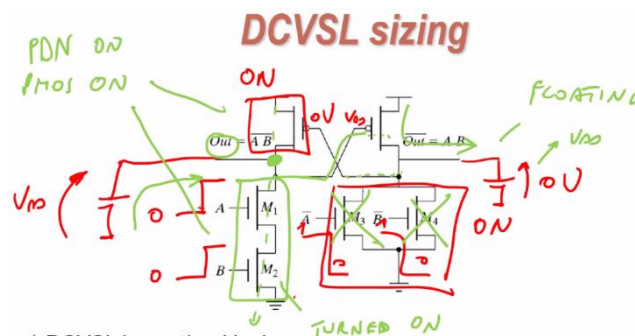


The circuit in the slide before is exactly the circuit here above, but some transistors are lumped together to spare area and power consumption.

This said, let's consider the following gate, and $A = B = 0$ at the beginning, with them then transitioning to 1. Before the transition of the inputs, PDN on the left is off, while the one on the right is on. This means that the output_bar is pulled down (left pMOS on), while Out is at Vdd.

When the input transitions, the left PDN is turned on and the PDN on the right is turned off. Now out_bar is floating (it's a capacitance charged to 0V, and the right pMOS is off). As for Out, we have a pMOS still on, but the left PDN is also on, so we have a cross conduction current since we have a direct path from Vdd to ground. This is a side effect of this logic during the transient.

As for the right side, if the left PDN is stronger than the left pMOS; the Out is pulled down, so we have some current discharging the capacitance at Out. If so, the pMOS on the right, which was previously floating, is switched on. So the capacitance on Out_bar is charged. As a consequence, the transistor on the left (pMOS) gets off, and so the PDN on the left is no more hampered by the pMOS and the Out can go to 0V, while on the right Out_bar we have Vdd.



In this case there is no more $V_{ol} > 0$ as in pseudo nMOS, thanks to the positive feedback loop.

During this analysis we assumed that the left PDN is stronger than the left pMOS. Let's try to redo this analysis removing this hypothesis and assume that the pMOS is stronger than the left PDN. Out capacitance is charged to Vdd, Out_bar to 0V. Initial situation is the same as before, with M1 and M2 and right pMOS off. then the transition occurs and M1 and M2 are on, while M3 and M4, previously on, are off. So again I have a crossconduction current on the left. If the pMOS is stronger, node Out remains stuck at almost Vdd (not Vdd because this implies a current equal to 0 in the pMOS, but the PDN is on and drawing current, so it cannot be Vdd), a bit smaller, so the pMOS on the right remains off. Hence also the capacitance on Out_bar remains floating to 0V.

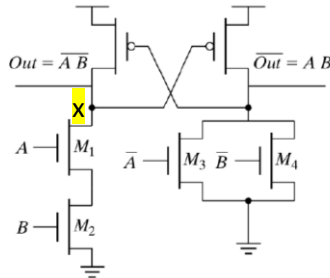
This circuit is not working because nodes cannot move from this situation.

We can assess the maximum aspect ratio of the pMOS transistor with respect to the PDN to have this circuit working.

Limit condition

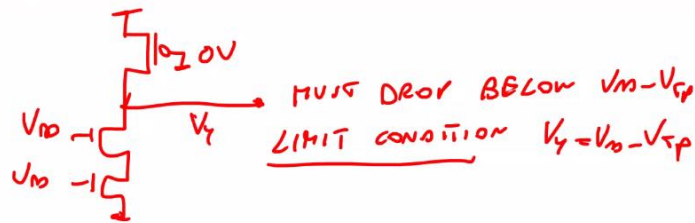
Which is the limit condition that makes the network working?

It depends on the 'turning on' of the right pMOS, we need to turn it on. The pMOS on the left is on because its gate is at 0V, while on the right node we have a floating pMOS. So voltage x must go below $V_{dd} - V_{tp}$. This means that the V_{gs} of the pMOS on the right is greater than V_{tp} and it is hence turned on.



The objective of the analysis is to assess the aspect ratio of the pMOS transistor with respect to the one of the nMOS (maximum beta factor) that allows the gate to work.

The situation on the left is the following.



We consider the PDN with a $(W/L)_{n,eq} = \frac{1}{2} (W/L)_n$, since they are in series, and I can substitute the PDN with a single nMOS of $(W/L)_{n,eq}$. In our reference technology, $V_{dsat,n} = 0.63V$, $V_{tn} = 0.43V$ and $V_{tp} = 0.4V$. This results in $V_y = 2.1V$

The pMOS is working in ohmic, with $V_{ds} = V_{tp}$. The nMOS is working in velocity saturation.

$$\mu_p \left(\frac{W}{L}\right)_p \frac{V_{ds,p}}{V_{tp}} (V_m - V_{tp} - \frac{V_{ds,p}}{2}) = \mu_n \left(\frac{W}{L}\right)_{n,eq} \frac{V_{ds,n}}{V_{dsat,n}} (V_m - V_{tn} - \frac{V_{ds,n}}{2})$$

We can now assess the maximum beta, which corresponds to this limiting condition.

$$\frac{\left(\frac{W}{L}\right)_p}{\left(\frac{W}{L}\right)_{n,eq}} = \beta_{max} = \frac{\mu_n \left(\frac{W}{L}\right)_{n,eq} V_{dsat,n} (V_m - V_{tn} - \frac{V_{dsat,n}}{2})}{\mu_p \left(\frac{W}{L}\right)_p V_{tp} (V_{dd} - V_{tp} - \frac{V_{tp}}{2})} = 5.6$$

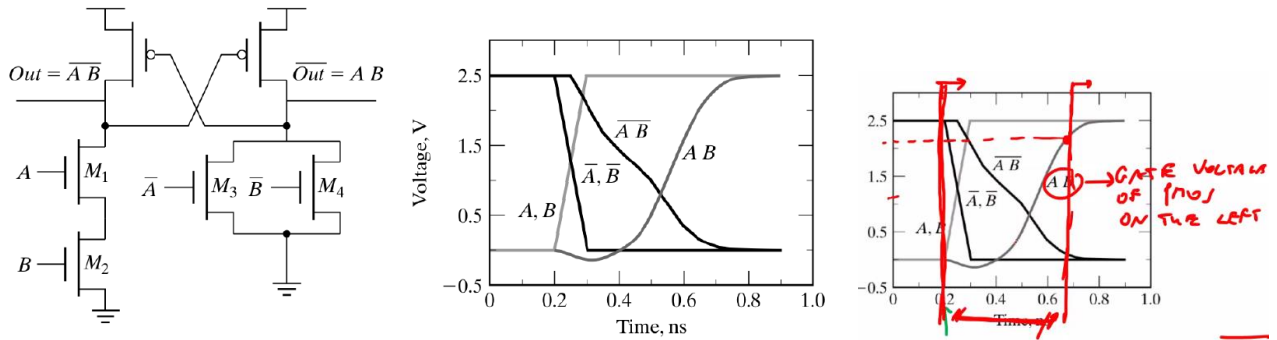
Of course, in the gate we need to grant a beta factor smaller than this one.

Transient response

A and B transition from 0 to 1, vice versa A_bar and B_bar. If we look at the transient response, we can notice that:

1. Pull down is faster than pull up. It is reasonable because it copes with the activation of the loop. In fact at the beginning we pull down Out, then the pMOS on the right is turned on and this corresponds to the pull up of Out_bar, and then we switch of the left pMOS. This also means that there is **no more hampering effect**, because we switch off the pMOS on the side we need to implement a logical '0'.
2. Cross conduction current. From 0.2ms on, the PDN is on; $A*B$ is the gate voltage of the pMOS on the left and when it reaches 2.2V, the pMOS on the left is off. So we have a cross conduction

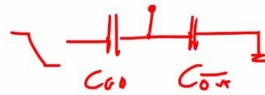
current in the left path in this period of time. Only after the second red bar the hampering effect is no more present and the pMOS is really off.



We notice that the $Out = (A*B)_{bar}$ has a change of concavity, like if it has a faster drop. In the first part of the transient, the PDN is hampered by the pMOS transistor; if the pMOS would have stayed on, the Out could have not reached 0V, it remains to a voltage greater than 0V (if the loop was cut). Fortunately, the voltage drops to zero because the loop switches off the pMOS. The change of concavity is because we are speeding up the pull down.

Let's now focus on Out_{bar} . We notice that it takes some time to start to rise because we are increasing the V_{gs} of the pMOS, and it requires some time. Once it is slightly on, the Out_{bar} capacitance starts to increase its voltage.

Furthermore, Out_{bar} has an underbounce, it is smaller than 0, negative, e.g. -150mV. We have an overlap capacitance between gate and drain of M_3 and M_4 , so something like a voltage divider made of capacitances, so we have a capacitive coupling to Out_{bar} . Also because in this small time the pMOS is off.



We don't have the same behaviour in the Out transition because the left pMOS is on at the beginning of the transient, so the node is fixed to Vdd. The network would be the following, with a RC network that keeps the voltage to Vdd.



PASS-TRANSISTOR LOGIC

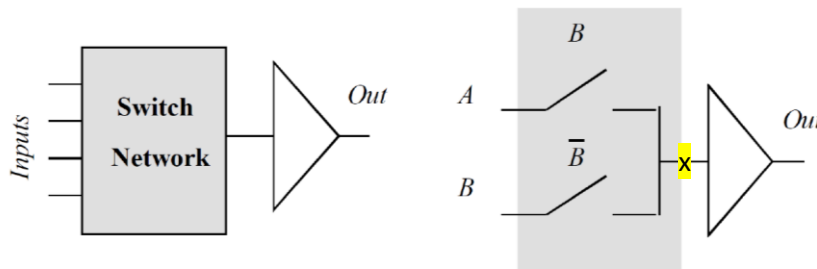
A static logic is a logic where the output is always connected to Vdd or GND with a low impedance path. Pass transistor logic is a static logic.

The idea is to improve the FC-CMOS logic, which has the drawback of pMOS transistor usage, which are not so conductive and hence must be increased in size (bulky).

In pass transistor the idea is to use nMOS transistors to convey a signal that can be Vdd or ground.

- ❑ The aim of pass-transistor logics is to build gates faster and smaller than FC-CMOS ones.
- ❑ FC-CMOS gates are reliable, offer great robustness and symmetrical delays but are cumbersome and slow
- ❑ Pass transistor logics use transistors as real switch, not just as PDN or PUN
- ❑ We will analyze the following topologies:
 1. Lean Integration with Pass-transistor (LEAP)
 2. Swing Restored Pass-transistor Logic (SRPL)
 3. Complementary Pass-transistor Logic (CPL)
 4. Transmission gate logic

Transistors as switches



- No static power consumption
- No PMOS transistors (large and capacitively heavy)
- Only NMOS transistors

We can implement an AND gate with just two transistors, as in the right grey box. This is the approach of multiplexing. The select signal of the multiplexer is B. When $B = 0$, the output is 0, so it's B, and when $B = 1$, the output is A. The expression is: $Y = A*B + B*\bar{B}$, but $B*\bar{B} = 0$, so we get $Y = A*B$.

The idea is to implement something like a multiplexer. The two paths for B and B_bar of course are never on at the same time.

Advantages

No static power consumption because we will never have a path between Vdd and ground since they will never be on simultaneously. So if e.g. $A = Vdd$ and $B = GND$ and the two are on at the same time, we would have a connection between Vdd and GND, but in this case it will never happen.

Furthermore, we are not using pMOS transistors, it is an nMOS only network.

Disadvantages

Let's remove the buffer and consider the output at node x. We put $B = 1$ and $A = V_{dd}$; the voltage at node x is $V_{dd} - V_t$, that in our reference technology is $2.5 - 0.43 = 2.1V$. However, we are neglecting the body effect in doing so, so the voltage is not 2.1 but 1.7V (real $V_t = 0.8V$), so losing 0.8V with respect to V_{dd} .

Once we reach 1.7V, the top transistor is turned off and not able to bring the voltage at node x at $V_{dd} - V_t = 1.7V$, so it is in cutoff, hence x is floating (also because the bottom transistor is off) and thus we are not implementing a static gate (we don't have a path to V_{dd} or GND), because the output node is floating.

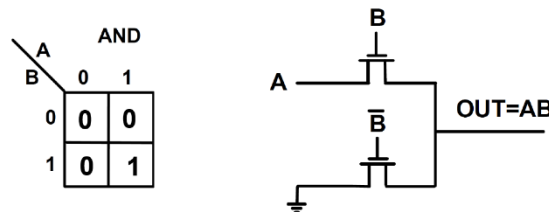
So as a first disadvantage, $V_{oh} = V_{dd} - V_{t^*}$, where V_{t^*} is the nominal threshold voltage increased by the body effect.

There is another drawback. Again, let's consider $B = V_{dd}$ and input A transitions from 0V to V_{dd} . Output x is 0V, because also A is at 0 at the beginning and there is no current. Then we apply the transition, current flows from left to right and so the source is at x. The $V_{gs} = V_{dd} - V_x$, and V_x is increasing, so the transistor is reducing its driving capability because we are shrinking the V_{gs} voltage. This is also a reason why nMOS cannot be used for pull-up (very slow pull-up), aside from the fact that we cannot reach V_{dd} .

If we now assume that the buffer is an inverter, a classical FC-CMOS inverter in reference technology with threshold in the middle, we have that $V_{ol} = 0V$ and $V_{oh} = V_{dd}$. If $A = 0$, out will be V_{dd} , but if $A = V_{dd}$, since x is not 1 but 1.7V, the output of the inverter is not exactly 0, but something slightly higher. If we apply 1.7V in input to the inverter, both pMOS and nMOS are on, so we have a static power consumption from V_{dd} to ground. This is a huge issue. Moreover, if the input is 1.7V, the output cannot be exactly 0 because both transistors are on.

Thankfully, the inverters have the regenerative property and we can regenerate the output at the output, we regenerate something close to 0V (even if not exactly 0V).

Example – AND

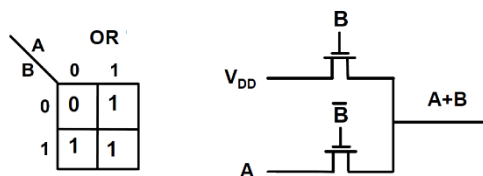


The gate is a multiplexer with a select bit (B):

- > $B=1 \rightarrow OUT=A$
- > $B=0 \rightarrow OUT=0$

From the k-map I select B as the select signal of the MUX. When $B = 0$, output is 0, and when $B = 1$ output is A. So one input is grounded and the other is to A, and B and B_{bar} command the switches.

Example – OR



The gate is a multiplexer with a select bit (B):

- > $B=1 \rightarrow OUT=1$
- > $B=0 \rightarrow OUT=A$

ISSUES

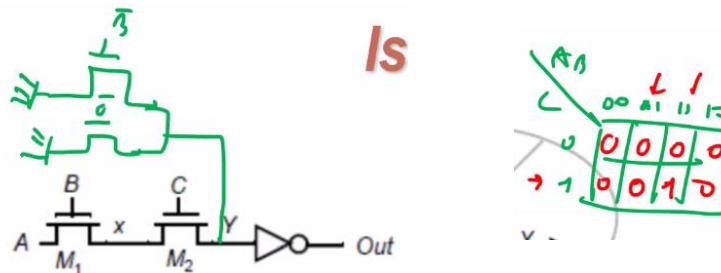


- V_y does not pull up to V_{DD} , but to $V_{DD} - V_{Tn}$
- V_T is increased by body effect ($V_{Tn} = 0.78V$)
- Threshold voltage loss causes static power consumption
- Pull-up is slow because V_{GS} is reduced along the transient
- Cascade is problematic

These are the other issues that are present. One is the problematic cascade.

Problematic cascade

If we want to cascade gates, we have a problem. Let's consider the circuit on the left (even if it is not the complete gate, some branches are missing). The complete circuit would be the following.



When B or C = 0, Y = 0, and also when B = C = 1, Y = A. So only when B and C are 1 the output is A. So we have a 3 input AND gate ($Y = A * B * C$).

Let's now consider A = Vdd = B = C. The current in the series is 0 because we are in an infinite impedance path (input of the inverter is at high impedance). Hence node x is at $V_{dd} - V_{tn}^*$ (considering body effect) = 1.7V. At node $V_y = V_{dd} - V_{tn}^* = 1.7V$. The first transistor is off, the other is in ohmic with $V_{ds} = 0V$. So V_y is not able to reach Vdd.

Let's focus on the circuit on the right of two images ago. We are trying to connect the output of a pass transistor network directly to the gate of another pass transistor. So they are not in series.

Let's suppose again A = B = C = Vdd. $V_x = V_{dd} - V_{tn1}^* = 1.7V$, considering body effect. When we reach 1.7V the transistor is turned off. Instead, $V_y = V_{dd} - V_{tn1}^* - V_{tn2}^*$. The two thresholds are not equal, the largest one is V_{tn2} because the body effect is smaller since the bulk source voltage is smaller, so it is something like 0.6V.

$$V_T = V_{Tn} + \gamma \left[\sqrt{0.6V + V_{as}} - \sqrt{0.6} \right]$$

This is a big issue because in input A we have a logical 1, so the output should be 0V. But since V_y is almost 1V, the output is Vdd, so it is a logical mistake. **I cannot cascade gates with nMOS transistors**, because we lose thresholds.

LEAP LOGIC

It solves all the issues explained so far. It involves:

- Pass transistor network

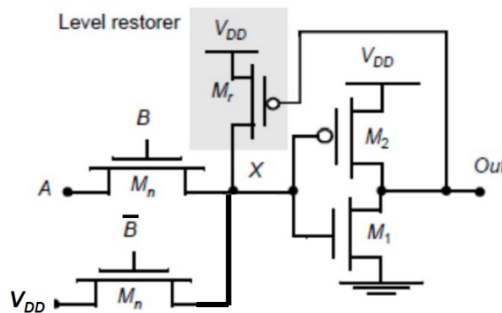
- Level restorer (weak pMOS transistor)
- Static inverter that drives the restorer

With this combination I can overcome all the issues seen so far.

The pass transistor network is done with Mn transistors. Then we have the FC-CMOS inverter with M1 and M2 and the Mr is the pMOS level restorer.

We have a positive loop at node x with the inverter and the pMOS in common source configuration.

LEAP: LEAn integration with Pass-transistor



- Level restorer exploits positive feedback to get $V_{OH} = V_{DD}$
- Add some parasitic capacitance
- Ratioed solution

Let's consider $B = V_{DD}$ always, and A transitioning from 0 to 1. When $A = 0$, $x = 0V$, out is V_{DD} and the pMOS is off.

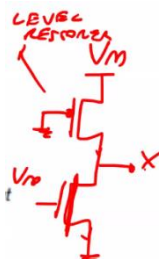
Let's consider now the transition at the input of A. Transistor is like aside. Node x can reach $V_{DD} - V_{tn}$. As soon as the node x passes the switching threshold increasing, the output of the inverter is toggled to 0V, so the pMOS turns on. If so, V_x can reach V_{DD} , because now pMOS and nMOS transistor are both pulling up the node, since the pMOS is performing in the last part of the transient. So $V_x = V_{DD}$ and $Out = 0V$.



Thanks to the positive feedback, **the static power consumption in the inverter has been solved, it is negligible the transient current. Moreover, we have also a full swing.**

Drawback

It is a ratioed solution. The problem, in fact, is for the transition from V_{DD} to GND of node A. At the beginning, $V_x = V_{DD}$, out = 0V and pMOS is on. When we have the transition, we can represent the circuit as the following.



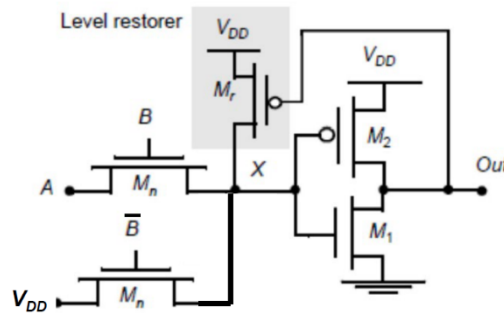
V_x value depends on the relative strength of the pMOS and nMOS, so on the beta factor between the two. If the pMOS is strong, $V_x = V_{DD}$ and it remains stuck to approximately V_{DD} even if we are trying to pull down it. Hence the circuit is not working, it's too strong. So **the pMOS has to be weaker than the nMOS.**

When x drops, we have the inverter and once the switching threshold of the inverter is passed, the inverter toggles to V_{DD} in output and the pMOS is switched off → V_x can reach exactly 0V, it is no more hampered by the pMOS.

Starting from the leap logic, let's consider B = 1, and A transitioning from 0 to 1. When A = 0, x = 0, the output of the inverter is V_{DD} and the pMOS is off.

Then we consider the transition, that is A = V_{DD}. Node x is the source of the transistor, the pMOS is off and so the current flows in the C_g of the inverter and x rises.

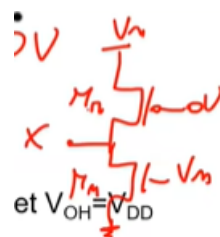
LEAP: LEAn integration with **P**ass-transistor



- Level restorer exploits positive feedback to get $V_{OH} = V_{DD}$
- Add some parasitic capacitance
- Ratioed solution

If the pMOS would be always off, x can reach $V_{DD} - V_{tn}^* = 1.7V$, affected by the body effect. Let's then take the inverter as a minimum size one (1) with $\beta = 3$ and $V_m = V_{DD}/2$. As soon x crosses $V_{DD}/2$, the inverter toggles and the output of the inverter goes to 0V. The pMOS is turned on and x node is charged both by the current of the pMOS and the above nMOS, allowing x to reach V_{DD} → pMOS is a **level restorer**, which performs the transition from 1.7V to 2.5V. So there is a positive loop acting.

Let's consider the opposite transition, A transitioning from 1 to 0. Node x is charged to V_{DD} at the beginning. The pMOS and the nMOS are both on.

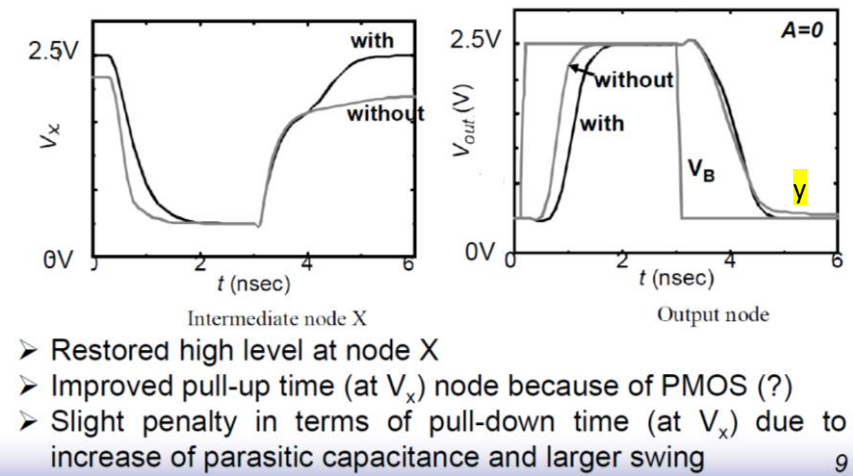


X starts from V_{DD} and at the end of the transient its final voltage depends on the aspect ratio of the two transistors. If the pMOS has a larger aspect ratio, x is close to V_{DD} because the pMOS is very conductive (small on resistance). If so, x stays at V_{DD}, so the output of the inverter stays at 0V and nothing happens, x is not pulled down even if A = 0 → logical mistake.

If instead the pMOS size is the same of the nMOS, so it is less strong due to the different process transconductance, x cannot reach 0 still. However, as soon as we cross $V_{DD}/2$, the inverter toggles, the pMOS is switched off and the nMOS is no more hampered by the pMOS and x can reach 0V. This is the correct way to size the gate → **pMOS must be weaker than the nMOS otherwise the pull down cannot be performed.**

Simulations result

Comparison with and without PMOS restorer (both with inverter)



The previous circuit is taken with $A = 0$ and B is transitioning (light gray). On the left I'm plotting node x behaviour, while on the right the output node.

From node x we expect the signal high at the beginning, then it goes down and up again when B returns 0. We are considering the cases with and without the level restorer.

Without the level restorer we are not able to reach V_{dd} . Of course the voltage increases slowly along time, and if we wait enough it reaches V_{dd} thanks to the subthreshold current.

If we consider instead the presence of the pMOS, we notice a change of concavity in the x , that is due to the fact that the inverter toggles and switches on the pMOS.

As for the pull-down transition of x , it seems that without the pMOS we are faster, and it is. First of all, we start from a smaller voltage because without the pMOS we are not able to reach V_{dd} . Then there is no hampering effect because there is no pMOS that 'fights' against the transistor to keep the node at V_{dd} .

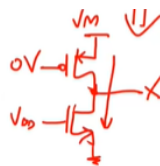
So the pull down is faster without the level restorer, but we still need it because if we cascade the leap logic with the FC-CMOS inverter, at 1.7V (final voltage reached by x when we don't have the pMOS) we have cross conduction current, because pMOS and nMOS are both on.

Then the overall output is a full swing because the inverter has the regenerative property.

The absence of the level restorer can also be seen in position y , where we have that the output is not exactly 0. This because the input voltage of the inverter is 1.8V and the voltage in output to the inverter cannot be exactly 0 because we have a cross conduction current.

Beta factor assessment

Let's call $(W/L)_p$ and $(W/L)_n$ the two aspect ratios. The objective is to assess the maximum beta factor so that the leap logic works fine in the pull down of x . x has to drop below the switching threshold of the inverter ($V_{dd}/2$) to turn on the positive loop and switch off the pMOS. The circuit to consider is the following, with pMOS on at the beginning.

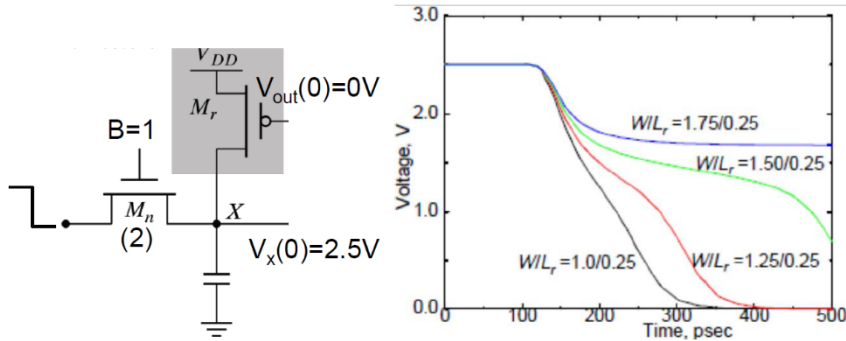


The limit condition for x , and also final point, is $V_m = V_{dd}/2$. In this final condition, the two transistors are both in velocity saturation region, so I equate the two currents in velocity saturation region of the two transistors.

$$\beta = \frac{(W/L)_p}{(W/L)_n} = \frac{k'_m V_{ovn} \tau_m (V_m - V_{tp} - \frac{V_{ovn}}{2})}{k'_p V_{ovp} \tau_p (V_m - V_{tp} - \frac{V_{ovp}}{2})}$$

$K'n = 115 \mu A/V^2$, $K'p = 30 \mu A/V^2$ in 0.25 μm CMOS from intel. In the end we get $\beta = 2.6$. This is the maximum aspect ratio of the pMOS to have the circuit working correctly.

Ratioed solution: PMOS must be weaker than NMOS



- PMOS is on at the beginning of the transient
- X must drop below the inverter threshold to switch the PMOS off
- If need be, the inverter can be low-skewed ($V_M < V_{DD}/2$) to fasten the turn-on of the PMOS and speed up the L-H transition of X 10

I cannot consider transistor with aspect ratio of 1 because it cannot be implemented. In the plot we have the voltage at x for different values of the $(W/L)_p$. Also in the pull down we have a change of concavity due to the elimination of the hampering effect of the pMOS. Before the concavity we have it, after we don't.

Up to now, the inverter has been consider a classical inverter with switching threshold in the middle. Typically, the inverter is sized so that the switching threshold is well below the middle of the supply range.

If we consider a pMOS weaker than the nMOS, the transition that 'struggles' for node x is the pull up, which is somehow not so good because we are using the nMOS for the pull up and the pMOS that acts only at the end of the transient, because before it's off.

Let's suppose to have a $V_m = 0.7V < V_{DD}/2$. If so, the inverter toggles earlier and the pMOS comes into play earlier in the pull up.

How can we get a V_m smaller than $V_{DD}/2$?

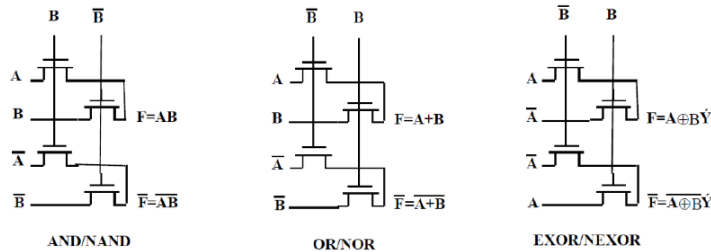
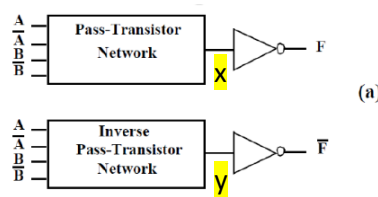
We reduce the pMOS size (of the inverter). This means that the transistor is **low-skewed**. This is very useful to speed up the pull up of the node x. So the advantage is not in the classical propagation delay, but in the time in which the transient is over.

LEAP logic is very powerful because we can implement complex gates with just few and small transistors, not in the FC-CMOS gates.

The problem is that it is a ratioed solution, the pMOS has a maximum aspect ratio over which the gate is not working correctly. In fact, sometimes we want to increase the pMOS ratio, to increase the pull up (pMOS here is the hampering one).

How can we solve this problem? We use a differential configuration and a feedback loop.

DIFFERENTIAL IMPLEMENTATIONS



Exploit the two complementary outputs to have full-swing

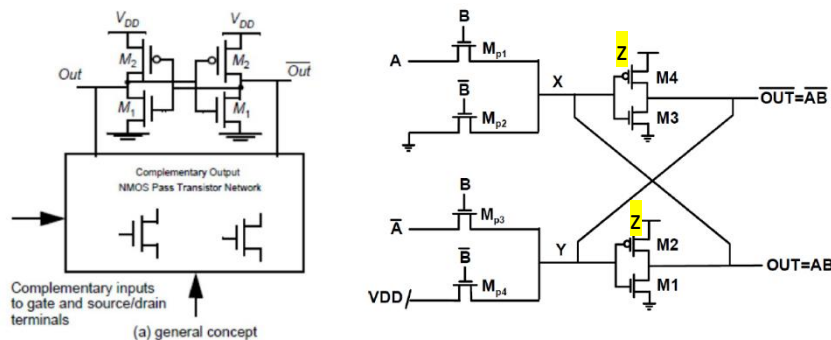
LEAP is single ended, and I want a differential gate. Instead of implementing one multiplex, we implement two of them; the two multiplexes have the same select signals, what we change are the inputs, complement inputs from one mux to the other.

We take $B = 0$ and consider the first gate $\rightarrow Y = \bar{A} * B + B * \bar{1}$.
 If we write the k-map, we see that it is indeed the NAND.

Let's consider the XOR. When $B = 0$, the top most transistor is on and $Y = \bar{B} * A$. Then in parallel we have another path that is on when $B = 1 \rightarrow Y = \bar{B} * A + B * \bar{A}$.
 Then we have to replicate the network and invert the inputs, and we get the XNOR.

The problem of the output node Y is that it cannot reach V_{DD} , it is $V_{DD} - V_{tn}^*$, so the two following inverters are subjected to cross-conduction current. Furthermore, from a logical perspective, this Y node is equal to F , and x to \bar{F} . However, F and \bar{F} are almost full swing thanks to the regenerative property of the inverter.

SWING-RESTORED PASS TRANSISTOR LOGIC (SRPT)



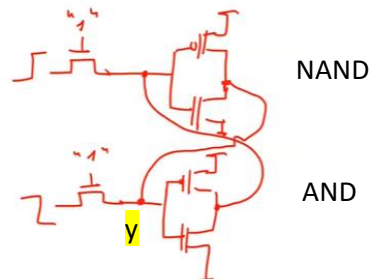
- Adopt two inverters in a positive feedback loop
- Still a ratioed solution
- PMOS transistors must be weaker than pass-transistors
- Inverters are typically LO-skewed ($V_M < V_{DD}/2$)

We have bridging connections since the connected outputs are the same from a logical perspective, but the output of the inverter has the advantage to be full swing. So from a logical perspective we have a level restorer after x and y , made of two inverters in cascade. It is a latch.

It is used to implement flip-flops and memories, particularly SRAM.

The latch implements a positive feedback loop. Unfortunately, it is a ratioed solution; in particular, the p MOS cannot be sized too large, because they **cannot be stronger than the nMOS transistors used in the pass transistor network**.

Let's consider the following example. $B = 1$ and A transitioning from 0 to 1.

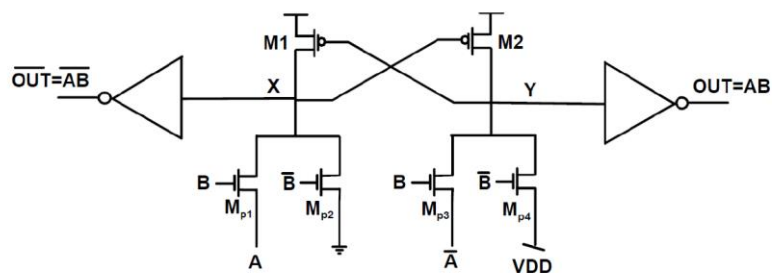


At the beginning of the transition we have $A = 0$, so in the upper inverter the p MOS is on and the n MOS is off, so the output is V_{dd} . In the lower part the p MOS is off and n MOS on. Then we have A transitioning. The upper part has the pass n MOS that is trying to pull up a node which is connected to the n MOS transistor of the bottom inverter, which is on, so the two transistors are fighting each other. If the two have the same size, the one of the inverter wins because the n MOS is stronger in performing the pull down.

But the bottom pass n MOS has to pull down a node connected to a p MOS connected to V_{dd} . If the p MOS is weaker, the n MOS wins, so we are able to pull down the node at position y . So in the bottom inverter the n MOS switches off and the p MOS switches on. Consequently, the bottom p MOS of the inverter helps to pull up the node in the upper path, so also the p MOS in the top inverter is switched off and the n MOS turned on. In the end this is a positive loop.

So the pass n MOS transistors have to be stronger than the p MOS transistor in the inverter, otherwise we cannot activate the positive loop.

COMPLEMENTARY PASS TRANSISTOR LOGIC (CPL)



- Adopt two PMOS in a positive feedback loop (like in DCVSL)
- But not a ratioed solution (pass-transistor can pull-up)
- Inverters used as buffer to drive other gates avoiding long chain of transistors (think about RC networks)

It is **ratioless**. In order to improve the LEAP function, we can exploit some expedients:

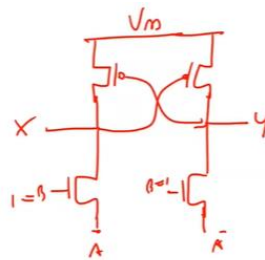
- Differential architecture, so that we can implement SRPL, which is however ratioed.
- Positive feedback.

These two expedients together lead to CPL.

Let's remove the two inverters for now, they are useless for the correct working for the gate, but used for another reason that will be explained later. The CPL is made of two pass transistor networks at the bottom, the true network and the inverse network, which are two equal multiplexers and the real inputs are complementary. Node X implements the function $X = A * B$, while Y is the complementary function $Y = B * \bar{A} + \bar{B} * A = (A * B)_{\bar{}}$. The restorer is made of a semilatch.

The CPL resembles a DCVSL in terms of restorer, but it is completely different in terms of bottom networks, which are not PDN, but pass transistor networks. So **nMOS are used as pass transistors, not pull down transistors**. So DCVSL is ratioed, while CPL is ratioless, because of the use of pass transistors.

We can demonstrate that whatever the size of the pMOS with respect to the pass transistors, the gate works fine, and X and Y are full swing signals. We start by designing the gate, considering just the left branch on ($B = 1$).



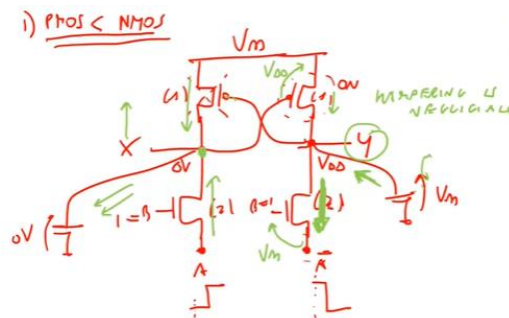
Let's consider A transitioning from low to high. Before the transition occurs, the pMOS on the right is on and keeps Y to Vdd. On the left, the pMOS is off.

pMOS < nMOS

Let's apply the transition and considering a sizing with the pMOS < (weaker) than the nMOS. According to the transition, node X has to be pulled up, Y down.

The first node to transition is Y because nMOS are more prone to perform the PD than the PU, so node Y is discharged with a discharge current drawn from the capacitance attached to Y. This current is somehow hampered by the right pMOS that is on, even though the nMOS draws a higher current because it is stronger than the pMOS. So the capacitor at the output Y is discharged, and the hampering effect is negligible.

If node Y is pulled down, the pMOS on the left is quickly turned on and pulls up node X together with the nMOS on the left, so also node X is pulled up (at the beginning only the nMOS pulls up, then the pMOS on the left cooperates).



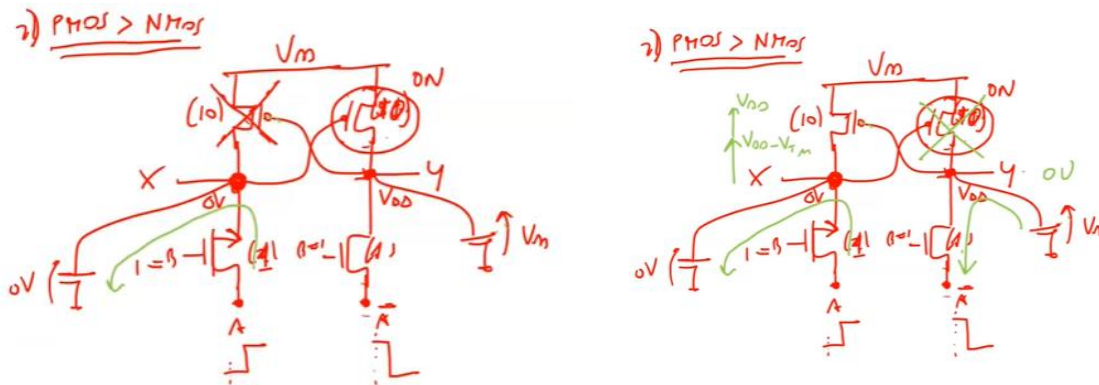
Hence chronologically we have:

1. Pull down with negligible hampering on node Y.
2. Pull up, firstly nMOS and then also pMOS for X.

Let's consider now a sizing with pMOS (10) > nMOS (1).

pMOS > nMOS

The first transition occurs at node X. In fact, at the beginning Y is hampered close to V_{dd} by the pMOS, which is stronger than the nMOS. On the left side, the pMOS is off, so the nMOS on the left can charge the capacitance at node X, with a very small current.



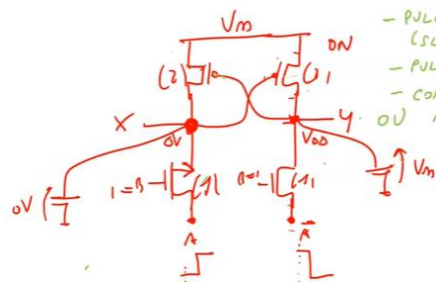
Node X increases, so the pMOS on the right is switching off, and at a certain point the right nMOS is no more hampered and can discharge node Y. As soon as this occurs, the left pMOS is turned on and it performs the last part of the pulling up of X. Chronologically:

1. Weak (slow) pull up of node X performed by a weak nMOS.
2. Pull down of Y.
3. Completion of pull up of X.

Of course this is not the right sizing, because the first transition to occur is a slow pull up made by an nMOS. So the gate works but it is very slow, the correct sizing was the previous one.

However, **since the pMOS transistor has not a maximum aspect ratio, we can increase the pMOS size to balance the two transitions, pull up and pull down. With this kind of gate we can have two more or less equal propagation delays.**

A possible sizing is the following, to have equal propagation delays.

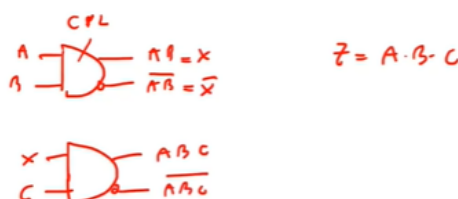


Role of the inverters

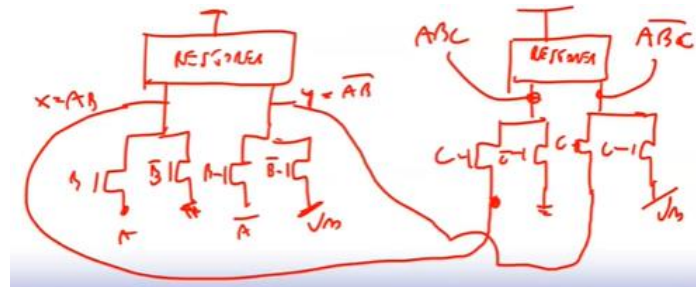
They are not mandatory for the correct working of the gate. **Inverters are used as buffer to drive other gates avoiding long chain of transistors.**

Let's consider a CPL with AND and NAND and let's suppose we want to implement the function $Z = A \cdot B \cdot C$.

The second gate can be fed with the signals X and C, and we get Z.



Without the inverters, we have the following networks.



From input A we have a certain path to ABC, and we are cascading two nMOS transistors, and if we cascade with other gates we might have a lot of nMOS transistors in cascade. The problem with cascading nMOS transistors is the capacitance in the middle nodes, because we have to discharge a lot of capacitances (or charge), increasing the propagation delay quadratically.

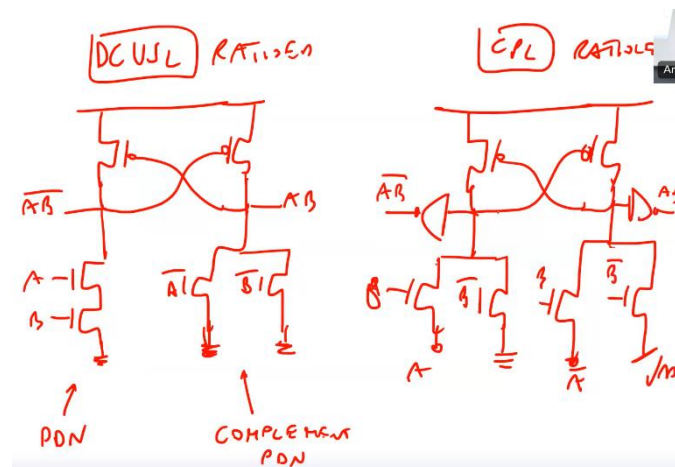
So the use of the inverter is suggested because it can lead to benefits in terms of propagation delay. Indeed, they act as buffers (same rationale as the wire cutting), avoiding a propagation delay that increases quadratically. If the increase in the delay is not a concern, we can actually remove the inverters.

Nodes X and Y are full swing regardless the presence of the inverter and whatever the sizing of the pMOS transistors, because we are implementing pass transistor network and not PDN.

Another advantage is that we have **two complementary outputs at the same time**.

DCVSL VS CPL

Let's design the same function, AND/NAND. For the DCVSL we have two complementary PDNs.

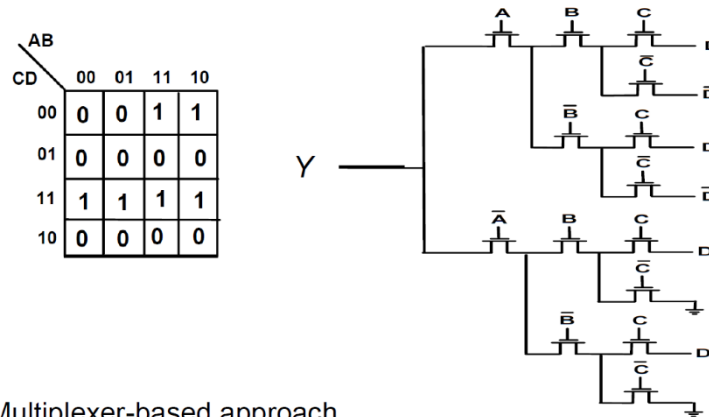


Aside from the inverters, the two networks are very similar, also with the same restorer (semilatch). The main difference is in the nMOS network; for the DCVSL the nMOS can only perform pull down, because connected to ground. For CPL we have pass transistor network which can perform both pull up and pull down.

As a second remark, in DCVSL we don't need buffers because we will never have a cascade of nMOS.

DESIGNING CPL GATES

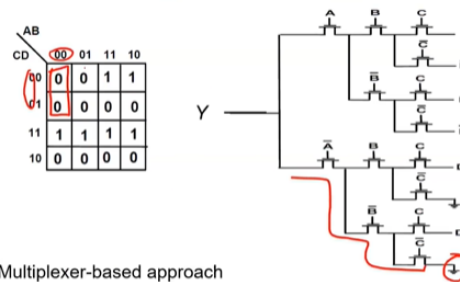
In the example we have a 3 multiplexer, where A, B and C are the select signals of the multiplexer and the inputs are D and D_bar.



- Multiplexer-based approach
- Allow to simply design whatever function
- Non-optimum solution

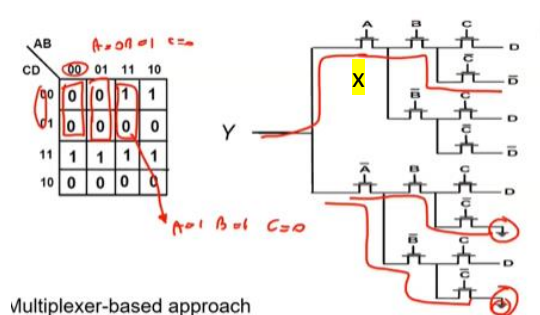
How to implement the pass transistor network

We start from A, B, C = 0, and Y = 0. This corresponds to the following path.



- Multiplexer-based approach
- Allow to simply design whatever function
- Non-optimum solution

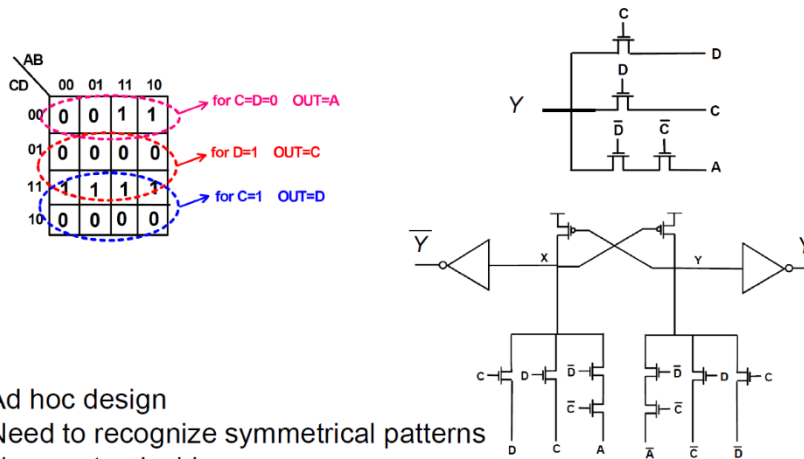
Let's take A, B = 1 and C = 0. At transistors level corresponds to the path x. In this case Y = D_bar.



In all the other cases in the last two rows, Y = D.

To optimize the gate, we need to look at the k-map and search for patterns of the output values. It is evident that in the last two rows we have 1 - 0 and the output is D whatever A and B value, so if C = 1, the output is D whatever the value of A and B.

Then in the middle I can recognize the combination of 0 - 1, and I don't care about the overlap, the important thing is to create the largest possible pattern.



- Ad hoc design
- Need to recognize symmetrical patterns
- Non-customizable

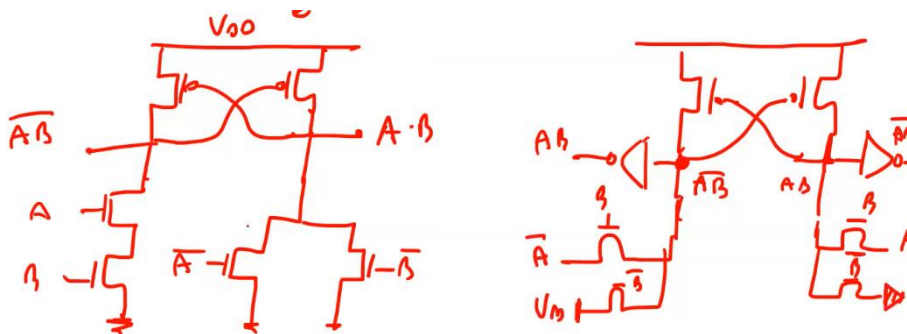
In the end I can get an ad-hoc solution for this k-map.

Let's consider the case $C = D = 1$ (row with all 1). We can notice an important property. In fact, for this configuration of the inputs, two paths are on, while in the general implementation of the CPL gate it never happens that two paths are on at the same time.

Recap – Difference between DCVSL and Pass Transistor Logic

Let's implement the function $Y = A \cdot B$. The two logics share the same level restorer, a couple of pMOS transistors connected in positive feedback. For DCVSL we have PDN, one complementary and dual of the other. In the DCVSL (left) we are implementing two complementary PDN, while in PTL we have pass transistor networks, not PDN, so they are able to perform both a PU and a PD.

Furthermore, DCVSL is ratioed, PTL is ratioless, so we can size the pMOS transistor in the way we want, without considering the strength of the level restorer. So Y and Y bar in the PTL are full swing



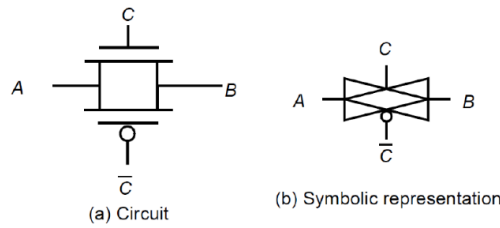
signals. Instead, for DCVSL the PDN must be strong enough to turn on the pMOS transistor, otherwise the feedback is not enabled, and the gate doesn't work correctly.

In the DCVSL gate we don't need inverters because the input is always connected to a gate of the next gate.

TRANSMISSION GATE

Sometimes it can be powerful to implement real switches that are able to convey both signals, 0 and 1. However, if we use circuits made only of nMOS as in the CPL gates, the nMOS is asked to perform both the PU and PD, but it performs well the PD. So the idea is to implement a switch with the pMOS and nMOS in parallel, activated by complementary signals. The nMOS is on for $C = 1$, the pMOS is on by $C = 1$, since inverting. So the two transistors are on for the same signal, not for different signals.

The advantage is that pMOS is good at PU, nMOS at PD, so this switch is good at whatever transition we are considering.

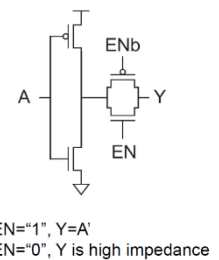


- Use of the PMOS to enhance the pull-up
- Both are used as real switch, not as just PU and PD devices
- Little penalty due the larger P-channel device

Example – Tristate Inverter

It's an inverter whose output can be a logical 0, 1, or high impedance. If we consider two DSP circuits connected to the same output, having an idle output is needed if one of the two has to communicate on the line.

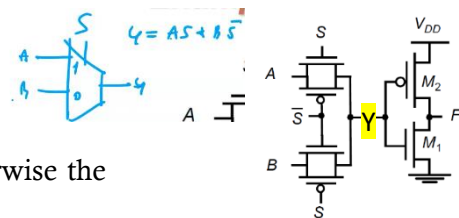
We need a transmission gate because if e.g. only an nMOS transistor was used, we have a problem for $A = 0$, because the output of the inverter reaches V_{DD} , but the nMOS can pull up only to $V_{DD} - V_t$.



Example - Multiplexer

This circuit works correctly, node Y is a full swing signal, since we are using transmission gate, so the inverter is not mandatory for the correct working of the gate.

It is used as a buffer; we need it to avoid a long series of transmission gates if we connect one after the other, otherwise the delay increases quadratically with the number of elements.

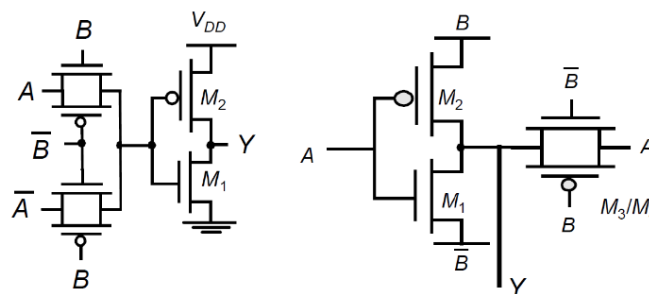


- Multiplexer (inverting): $Y = SA' + S'B'$
 - > $S=1$, $Y=A'$
 - > $S=0$, $Y=B'$
- Inverter (non mandatory) to buffer the output

Moreover, it is a very powerful circuit since we are implementing a powerful multiplexer with only 8 transistors. This circuit has good properties, in fact it is full swing and the output is buffered, with limited number of transistors.

In terms of area, maybe the LEAP gate is better than the transmission gate (less transistors), but in transmission gates we are not ratioed and in the case of LEAP (truthfully, in all pass transistor logics) we have to consider cross conduction current since it is ratioed, while for transmission gate it is not a problem because we don't have any level restorer.

Example – 2-inputs NOR



Classical implementation (mux like design)

Alternative implementation

Note that the FC-CMOS implementation requires 12 transistors

On the left we have the classical implementation with a multiplexer-based approach. When $B = 1$ the upper path is on and $Y = \bar{A}$, while when $B = 0$ the bottom part is on and $Y = A$.

This XOR requires 6 transistors + 2 inverters to implement \bar{A} and \bar{B} , so overall it is a 10 transistors implementation. I can implement the same function using a CPL.

The function is $Y = A \cdot \bar{B} + \bar{A} \cdot B$. We can transform this in an inverting function, which results in $Y = \overline{(A \cdot B + \bar{A} \cdot \bar{B})}$. We can implement this with a FC-CMOS gate, resulting in two pMOS transistors in series in the PDN and two in the PUN, so we have to counterbalance this effect increasing the size.

Moreover, in the mux-like design (left) we have two outputs, Y and \bar{Y} , but in order to cascade gates the output must be taken after the inverter. As a second remark, we have also to consider the two inverters to generate \bar{A} and \bar{B} because TGL is a **single ended logic**, the output is just Y . If the logic is single ended, if we need to cascade gate we need to complement \bar{Y} . Instead, CPL is differential, so we have at the same time Y and \bar{Y} .

The alternative implementation (on the right) is still a XOR gate, but very naïve. It is made of two parts, a strange circuit on the left (which is not an inverter) and a transmission gate. Let's neglect the final inverter in the classical mux-like design to compare the alternative implementation with it.

In the alternative implementation we have to consider the cases $B = 0$ or 1 .

B = 1

We notice that the transmission gate is off, and Y is \bar{A} (classical inverter), which is the same behaviour of the classical implementation.

B = 0

$Y = A$ according to the transmission gate. The part of the circuit that previously was working as an inverter is now different, we have a combination of two followers, two common drains, so from a logical standpoint we have A in output of it.

If the transmission gate is on, we cannot have a conflict, so it is expected to have A in output of the right part, which works as a buffer. To implement this naïve XOR gate we need only $4 + 2$ transistors, used to implement \bar{B} .

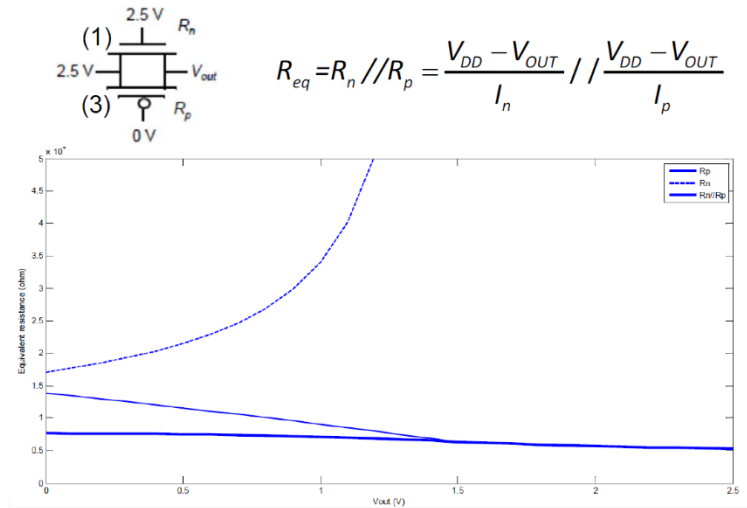
So this alternative implementation is very powerful because implementing the XOR gate with only 6 transistors, we cannot do better than this. So it is a powerful approach if we have to reduce the area. Moreover, since we don't have a level restorer, we don't have the cross-conduction current issue.

Remark

If we consider the TG as a switch, we are combining the properties of pMOS and nMOS transistors, and we can verify that the equivalent resistance, the large signal one defined as $R_{eq} = V_{ds}/I_{ds}$, is rather constant throughout all the transient from 0 to V_{dd} , and more or less equal to 6 kOhm.

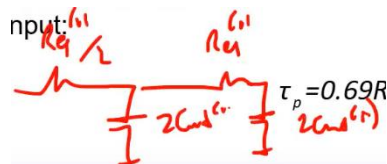
It is 6 kOhm since in our reference technology, during the PD, the nMOS has an equivalent resistance of $R_{eq(1),n} = 14$ kOhm. For the PU, for a pMOS of size 3, $R_{eq(3),p}$ is 31 kOhm/3 = 10 kOhm.

From the graph, it seems that the nMOS resistance R_n increases up to infinity. This because during the pullup, at a certain point, the nMOS switches off.

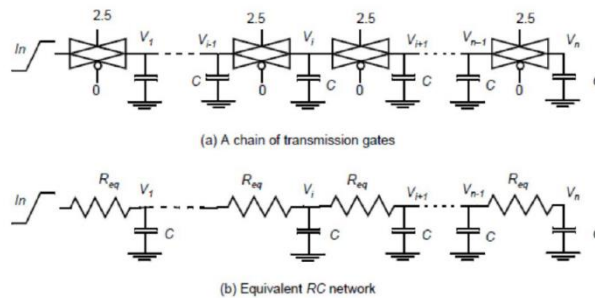


CASCADE OF TG

Let's consider transmission gates in series, so we have a cascade of switches. If we close these switches, the delay between input and output depends on the number of switches in series. Firstly, a TG, since it is made of transistors, can be seen as a resistance times a capacitor. The resistance of the switch, with sizing (1)n and (3)p, is more or less the $R_{eq}^{(1)}/2$, but as for the capacitance it is $C_{int}^{(1)} = 2 \text{ fF}$. However, we have to consider that we have two switches in series, so the situation is:



If we put these in cascade, the delay increases quadratically, since we have an RC network. It is the same reasoning done with the wire and the Elmore theory. This is why we cannot cascade TG without using inverters.



Assume all the internal nodes discharged and a step voltage is applied at the input:

$$\tau_p = 0.69 R_{eq} C \frac{N(N+1)}{2}$$

How many TG can we afford to cascade without a buffer?

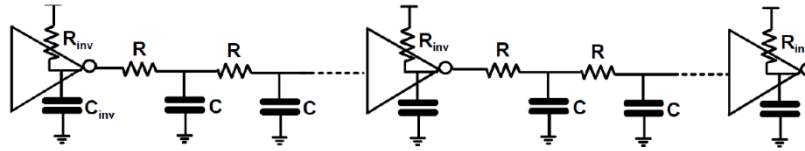
It is the same theory done for the wire in how many splits we have to cut it. Let's consider N TGs, and m is the number TGs per stage. So we have N/m stages.

To simplify the analysis, let's suppose to use minimum size inverters ((3) – (1)) and the TG is the combination of transistors with aspect ratio (3) and (1).

R_{inv} of the inverter is $R_{eq}^{(1)}$ and $C_{int} = C_{int}^{(1)}$. Since we are facing two TGs, $C = 2 * C_{int}^{(1)}$, while $R = R_{eq}^{(1)}/2$.

If we have N/m pieces, we can sum the delay of each piece and inside of each stage we can apply the Elmore delay formula. We have $m \cdot C$ and then we have the delay of the TGs, which increases quadratically with m , and then we have the combination of all the resistances of the TG that drive the final inverter.

So we get a delay that is a function of m . which is the optimum value for m ? We need to derive the delay with respect to m .



N transmission gates, m TGs per stage, N/m stages:

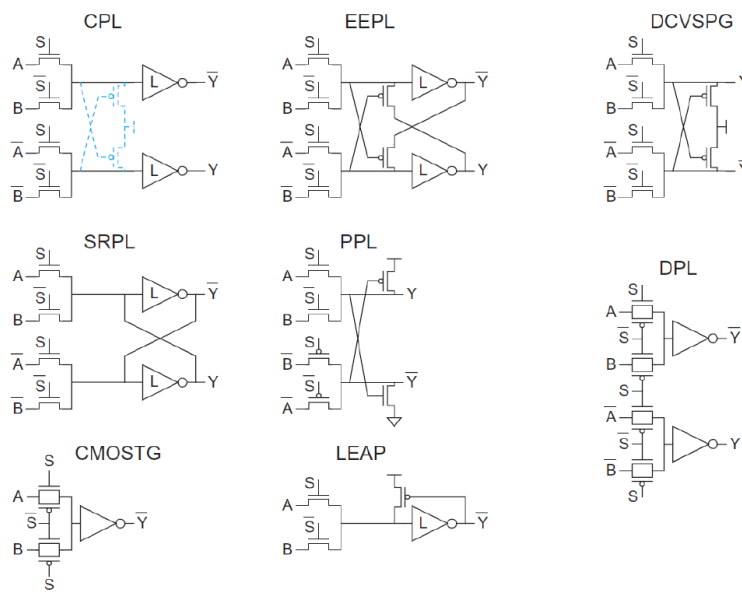
$$\tau_p = 0.69 \frac{N}{m} \left[R_{inv} (2C_{inv} + mC) + RC \frac{m(m+1)}{2} + mRC_{inv} \right]$$

Optimum number of TGs per stage:

$$m_{opt} = 2 \sqrt{\frac{R_{inv} C_{inv}}{RC}} \cong 2$$

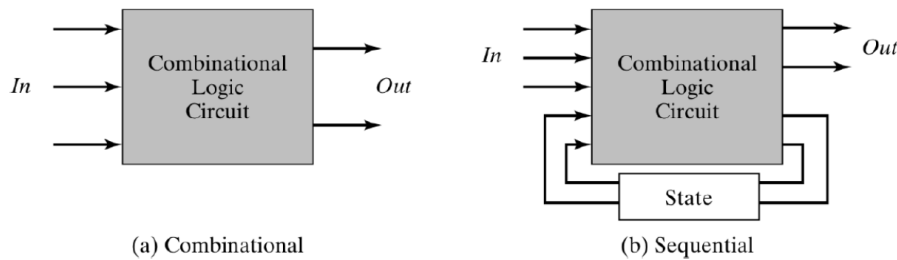
We get a final $m_{opt} = 2$. So in a single stage we can put two TGs and this leads to the best propagation delay with this sizing and parameters. Then we need to buffer, otherwise the delay increases.

RECAP



In the inverters sometime we see L, which means 'low skewed'. These inverters, in order to improve the performance of the gate, have a switching threshold that is not in the middle but smaller than $V_{dd}/2$ to enhance the pull up, because at the beginning the PU is performed by the pass transistor network, then the restorer comes into play and to speed up the turning on of the pMOS, we can size the inverter with a switching threshold below $V_{dd}/2$.

SEQUENTIAL CIRCUITS



$$\text{Output} = f(\text{In})$$

$$\text{Output} = f(\text{In}, \text{Previous In})$$

A combinational circuit is a circuit for which the output is a function of the input at that time stamp. Neglecting transients, assuming that the combinational circuit has a propagation delay of 0 ($\tau_p = 0$), so there is no memory, in the sequential circuit we have memory implemented using flip-flops rather than latches.

It is an example of finite state machine, it is a mixture of a combinational part and a memory in feedback. In reality, inside the combinational logic circuit of the sequential logic we have two circuits; one is the logic for the output, the other is the state logic. These two are made of gates and fed by the real inputs and the output of the memory element.

The memory element serves the role of creating a state for the machine. The bits that enter the state are called **next state**, while the output of the state is the **current state**.

If we consider that a sequential circuit needs a synchronization signal, i.e. a clock, let's suppose that the register 'state' samples the signal on the rising edge of the clock, storing in this instant the value.

Let's consider the behaviour of the overall circuit. The two blocks in the combinational logic are fed by the inputs and the current state. As for the output, we have two outputs; the real output of the circuit but also the state of the circuit. So at every rising edge of the clock we assess the output combining the inputs and the current state generating an output and a next state. The next rising edge of the clock the next state becomes the current state and the current state is evaluated together with the inputs and so on. So the 'state' path behaves as a delay element. This kind of memory is also called **foreground memory**.

LATCH AND FLIP FLOP

- ❑ Two basic memory devices able to store a bit:
 1. Latch
 2. Flip-flop (also called register)

- ❑ Latches and flip-flops are used:
 1. Foreground memory, e.g., as in Finite State Machines (FSMs)
 2. Frequency dividers and counters
 3. "Pipelining"

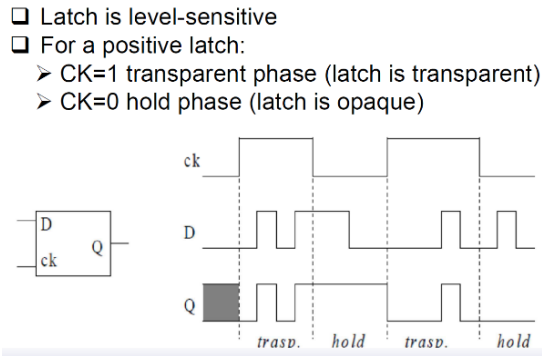
A foreground memory is a memory embedded in some combinational logic to implement a finite state machine. This is a way to differentiate it from the background memory, like SRAM, DRAM, FLASH. The difference is in the quantity of bits we can store. Foreground is in the order of kbits, background Mbit or Gbit. Moreover, the delay of foreground is in the order of ps, of background in the order of ns or even us.

Moreover, flip-flops are very cumbersome and large, while SRAM and DRAM are very small, of 6 or even 1 transistor. The most important application of flip-flops is in pipelined circuits.

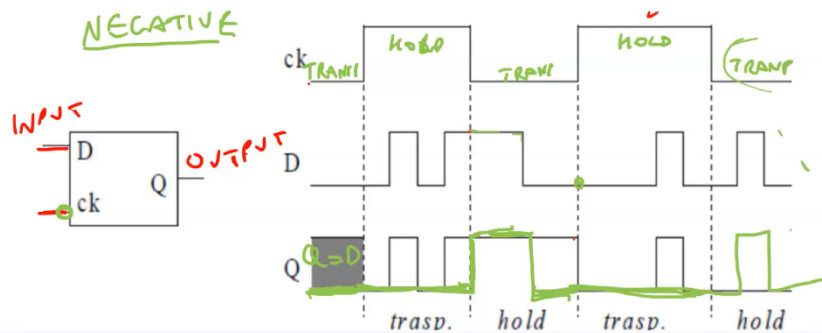
LATCH

The main difference between latch and flip-flop is that the latch is a **level-sensitive device**, able to store a bit when the clock is one or zero, so we distinguish **positive** or **negative latch**. The positive latch stores a bit in the whole period where the clock signal is high; the negative latch is a device that operates in a complementary way, able to store a bit for the whole period of time where the clock is 0. So for a positive latch we can distinguish:

- **Transparent phase:** clock = 1, so the output is equal to the input.
- **Hold phase:** clock = 0.



If we have a negative latch, it is indicated with a circle at the clock input. Now the latch is transparent when clock = 0. As in the positive latch, the last value stored remains in the hold phase, then when clock = 0 we are back in the transparent phase, and we follow the input.



So it is clearly a **level-sensitive device**.

FLIP-FLOP

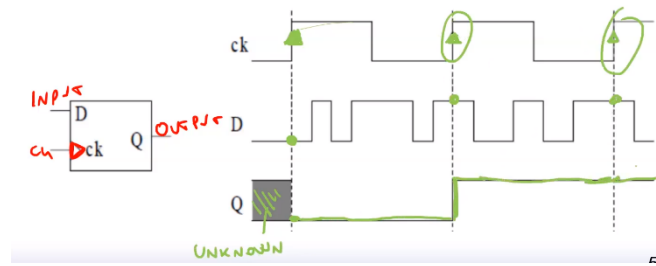
It is an **edge-triggered device**, and we can have positive or negative edge-triggered flip-flops. Positive means that the device samples the input signal on the rising edge of the clock, viceversa on the falling edge for negative edge-triggered flip-flops.

Its symbols resembles a latch but it has a triangle on the ck line.

Positive edge-triggered flip-flop

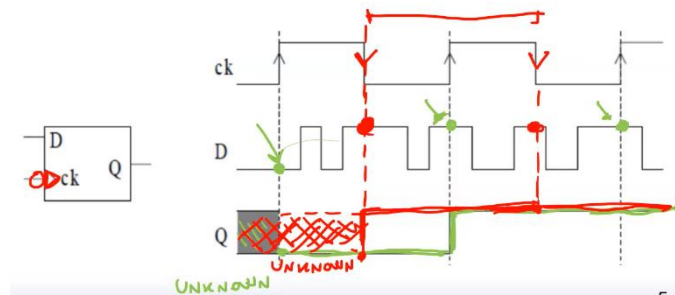
At the first edge, since it is the first one, we don't know the value of the output before the rising edge, but I'm sure that after it we sample the next value, which is a 0, and it remains 0 for all the clock period. In correspondence of the second rise edge I sample the second value. Then I have a third rising edge, and the sampled value remains 1. So we sampled 3 values.

- ❑ Flip-flop is edge-triggered
- ❑ A positive (negative) edge-triggered FF samples the input data on the rising (falling) edge



Negative edge-triggered flip-flop

The input is unknown up to the first falling edge, and at that point the output is 1 and it remains like that up to the next falling edge, where I sample another 1. So the behaviour is completely different depending on the sensing edge.

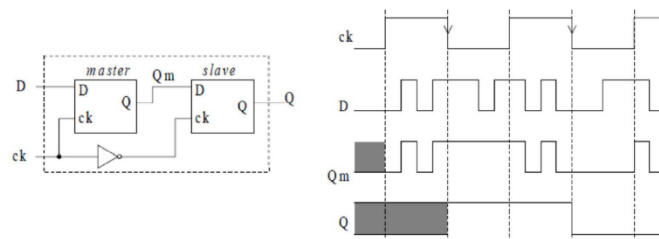


These are ideal behaviours.

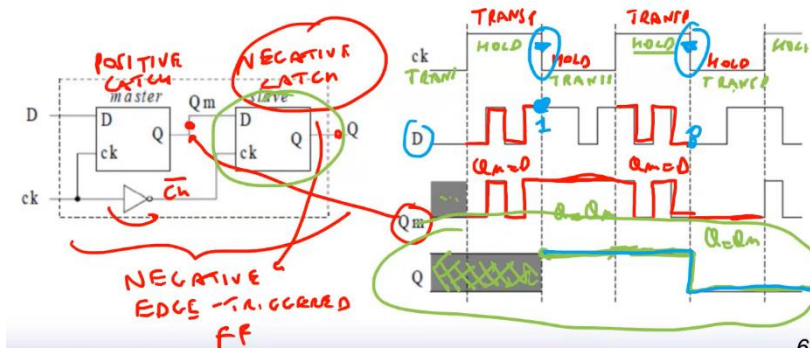
MASTER-SLAVE FLIP-FLOP CONFIGURATION

The flip flop is implemented combining two latches in cascade in the so-called master-slave configuration. We put in cascade two latches with opposite polarity (positive latch and negative one in the image), leading to a negative edge-triggered flip-flop. So the second latch defines the kind of flipflop. The second latch is fed by clock_bar, so for clock = 0, the second latch samples the input.

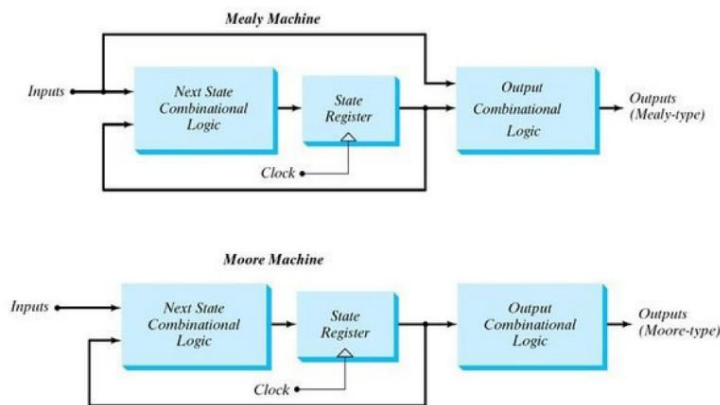
- ❑ Latch is typically used to implement a flip-flop in the so-called master-slave configuration
- ❑ A positive edge-triggered FF is the cascade of a negative and positive latch (and viceversa)



The intermediate node Q_m is the output of the first latch, that is a positive latch. So in the hold phase the output remains steady for the master latch. Now it is very simple to assess Q since Q_m is the input of a negative latch, but for the whole period gray in the last plot the Q_m is unknown, and $Q = Q_m$. So the signal at the bottom is like having sampled the signal D at the falling edges of the ck signal.



Example of finite state machine (FSM)



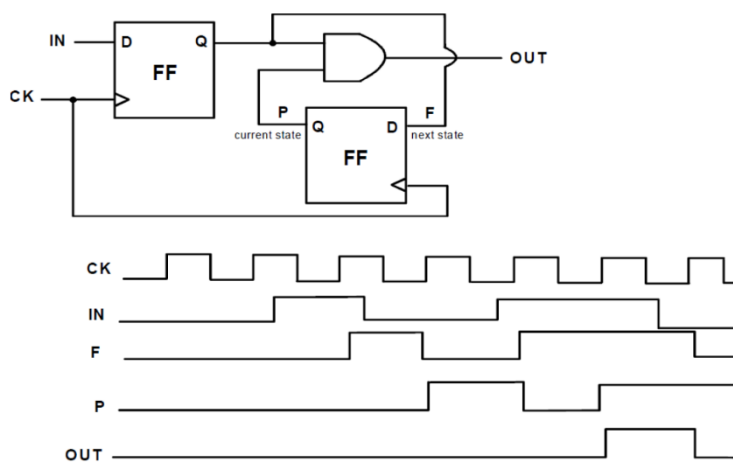
There are two kinds of FSM: Mealy and Moore. If the output is a function of only the current state we are dealing with Moore FSM, while if it is the function of both the inputs and the current state we have a Mealy machine.

State register is a flip flop that stores the input, not just a single bit but a bus of n bits.

The difference between the Mealy machine and the Moore one is that the output in the case of Moore is function only of the current state.

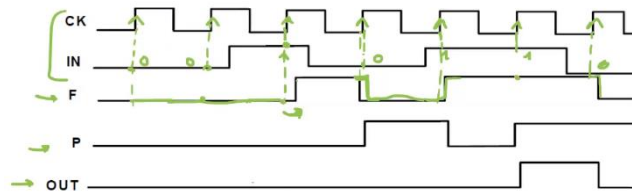
We will use Moore machines to implement counters and frequency dividers.

Example of sequential circuit (FSM) – 2



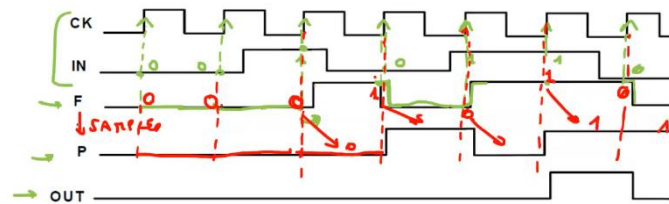
In this case we have a single bit input also sampled by a flip flop; the single AND gate is the output logic and the state register is made only of a single flip flop. The output is function of the sampled input and of the current state. This is a Mealy FSM.

Let's see how the circuit works. Ck is the synchronization signal, and the flip flops are positive sensing, so sensitive to a high transition. P is the current state, F is the sampled input (corresponding to the next state).



When I sample the first one, the operation of sampling takes some time, so we have a delay with which F goes to 1. This delay is not from the input to the output, but from the rising edge to the output.

As for P, P is the sampled version of F.



If we compare F and P, they are shifted in time by a clock period. This is the effect of the flip flop presence, which is a **delay element**.

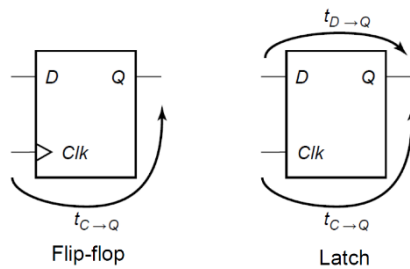
As for the output, it is the output of an AND gate, so only when P and F are 1. Hence this machine is able to recognize when there are two ones in cascade in the input signal.

With a simple combinational circuit we cannot recognize two 1 in series, we need for sure a memory element.

To recognize two zeros in cascade we need instead a NOR port.

TIMINGS

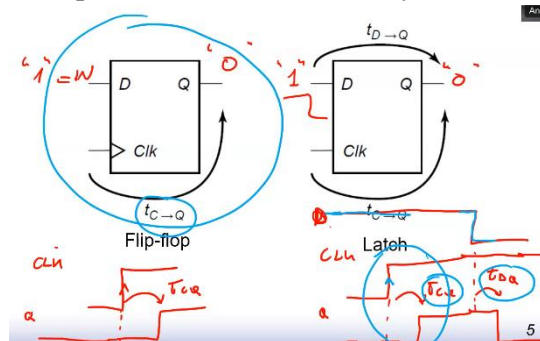
The presence or the absence of the circle in the clock line identifies the type of flip-flop (positive or negative). With the circle we have a negative edge triggered flip flop, and the same for the latch.



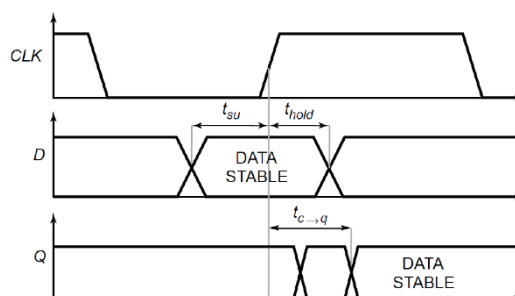
Since the behaviours of the flip flop and latch are completely different, they are characterized by different propagation delays. In the case of FF (flip-flop) we have just one propagation delay from input C to output Q. So the input signal on D is transferred at the output after a delay $t_{C \rightarrow Q}$. Time is evaluated starting from the edge of the clock.

Let's suppose that the input is 1, and the output is 0. Once the clock transitions from low to high, the output Q goes to 1 with a delay $t_{C \rightarrow Q}$. Since the FF is an edge triggered device, the delay is assessed starting from the edge of the clock.

Instead, the latch is able to copying the input signal for all the time when the clock is 1 (if positive latch). But it can happen another thing. Let's suppose that D transitions during the period when $ck = 1$, and it goes e.g. from 1 to 0. The input is updated but not with a delay $t_{C \rightarrow Q}$, but $t_{D \rightarrow Q}$.



FLIP-FLOP CHARACTERISTIC TIMES



Let's consider a positive edge triggered device. This device samples the input signal in correspondence of the rising edge of the clock. To work correctly, the FF has to satisfy some conditions; since it is edge triggered, it is able to sample the input signal in correspondence of the raising edge, but t_{su} has to be satisfied, so the input cannot change too close to the rising edge of the clock, so the input data D must be stable in the period of time before the rising edge \rightarrow must be steady to correctly acquire the signal. This condition is called **setup condition**, and t_{su} is the **setup time**. If data changes, two things can happen. Either the data is not sampled correctly, so we are not able to update the output, or the propagation delay becomes too large. So t_{su} cannot be nihil and must be > 0 .

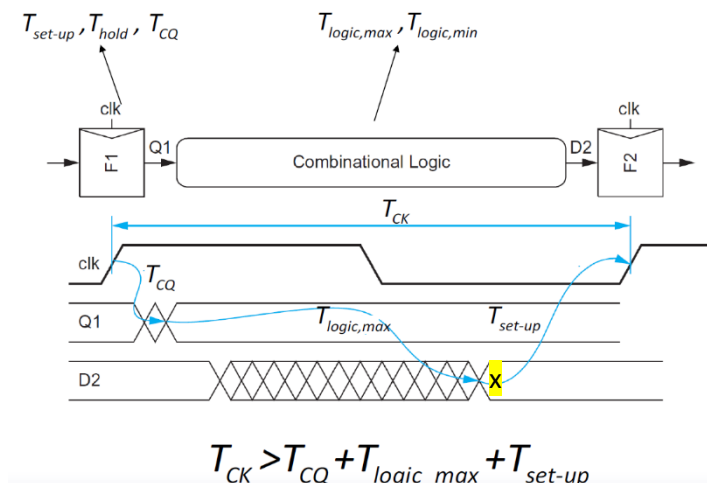
Another condition has to be fulfilled; the data must be stable also after the rising edge for a time t_{hold} . This means that **around the sensitive edge of the clock the data must remain stable**, otherwise the FF isn't working correctly.

If this condition is verified, after a propagation delay $t_{C \rightarrow Q}$ the data is updated at the output, from the sensitive rising edge.

- ❑ T_{set-up} is the time window before the sensitive edge of the FF in which the input signal D must remain stable to be correctly sampled
- ❑ T_{hold} is the time window after the sensitive edge of the FF in which the input signal D must remain stable to be correctly sampled
- ❑ T_{CQ} is the time after the sensitive edge of the FF when the signal becomes stable

T_{hold} can be positive or a 'negative time' or 0.

MAX DELAY CONSTRAINT

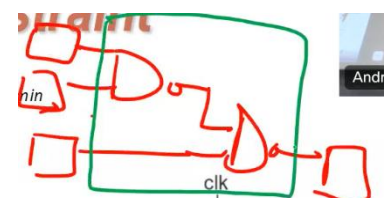


The depicted FF indicates an array of parallel FFs with ck signal in common. On the first rising edge of the clock, the first register samples the input signal, but in correspondence of the same rising edge of the clock, the receiving FF samples the signal on D2. After a delay $t_{C \rightarrow Q}$ the output Q1 is transitioning (the cross in the plot indicates the bit transitioning). Then we have the combinational logic in input that process Q1. After a propagation delay τ_p of the logic we have the signal available at D2. D2 is a single bit and it transitions at x.

The receiving FF has to sample D2. On the second rising edge, the FF at the end of the chain has to sample the signal. But in order to correctly acquire the signal, D2 must transition a setup time before the ck rising edge.

So in a clock period we have to: sample the input signal, process the input data through the combinational logic and be in advance of a setup time before the second rising edge of the clock.

Let's suppose to have 3 FFs in input, and the following combinational logic. In this circuit we have 2 delays, not just one, because it depends on the signal path we are considering. **$T_{logic,max}$ is the worst case delay of the combinational logic.**



Moreover, the inequation of the previous slide sometimes is rewritten re-arranging the terms.

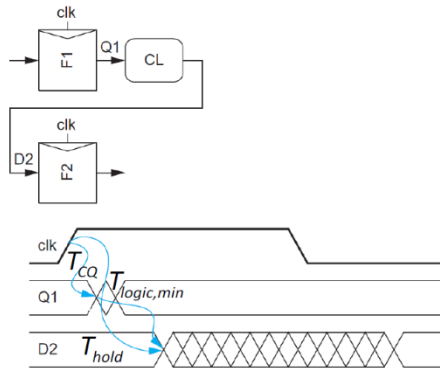
$$T_{logic_min} \leq T_{clk} - (T_{cq} + T_{su})$$

$T_{cq} + T_{su}$ is called **sequential overhead**. So it there exist a minimum clock period.

If we establish a ck period and we designed the FF with T_{cq} and T_{su} , the logic block must have a defined delay according to the formula above. It is a different way to see the same thing.

In the end, the sum of T_{cq} and T_{su} reduces the bitrate.

MIN DELAY CONSTRAINT



$$T_{HOLD} < T_{CQ} + T_{logic_min}$$

It refers to the hold time of the FF. We can consider the circuit of before, just with a different layout.

Let's consider the same rising edge of the clock and what the two FFs are doing in correspondence of this rising edge. The first FF, F1, is sampling the signal in input, and F2 is sampling D2. After a while, D2 changes. Since a FF has a hold time constraint, D2 cannot change too fast, otherwise F2 is not acquiring correctly the signal at its input. If F2 has $T_{hold} > 0$, the condition in the slide must be verified.

Conversely, if $T_{hold} \leq 0$, there is no constraint.

What happens if we don't respect the min-delay constraint?

Let's suppose $T_{cq} = 20\text{ps}$, $T_{hold} = 40\text{ps}$ and $T_{logic} = 0$ (so there is no logic). In correspondence of the rising edge, the first FF samples the input signal, and F2 samples D2.

Due to the selected times, the D2 is updated from 0 to 1 after 20 ps and F2 is not writing a 0 at the output but a 1, so we are committing a mistake.

So it is like the input is passing directly to the output, but this is due to the fact we are violating the minimum delay constraint.

We have two possible solutions:

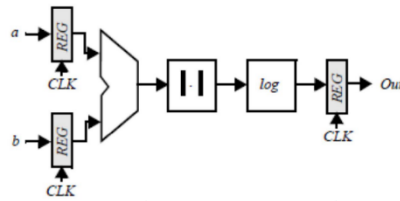
1. Design a FF with $T_{hold} < 20\text{ps}$.
2. We add a delay, e.g. two inverters in cascade.

Example – pipelining a circuit

We have two inputs, A and B that are digital words (i.e. not a single bit). The two digital words are summed and they pass through a block that assess the absolute value, and then another block that performs the logarithm. These three blocks are the combinational logic.

So we have T_{cq} , T_{add} , T_{mod} , T_{log} and T_{setup} . How much is the maximum operating frequency of the circuit? In a clock period we have to do 3 operations: sampling the input signal, processing it and being at the output T_{setup} time before the sensitive edge of the clock.

For the sake of simplicity, let's suppose that the 3 times of the logic block are equal and equal to T. Let's suppose also that Tcq is much smaller than T, as well as for Tsetup. This means that the sequential overhead is negligible with respect to the propagation delay of the logic. The maximum operating frequency of the circuit is 1/3T.

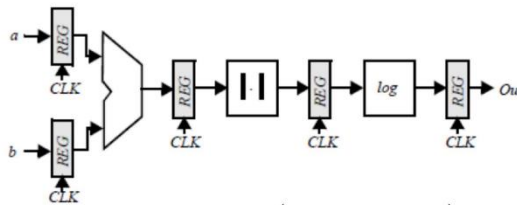


$$T_{CLK} > T_{CQ} + (T_{add} + T_{mod} + T_{log}) + T_{set-up}$$

Assuming equal delay ($T_{add}=T_{mod}=T_{log}=T$) and a negligible overhead:

$$T_{CLK} > 3T \Rightarrow f_{max} \cong \frac{1}{3T}$$

Let's add registers in the circuit at the output of each combinational circuit. For each part we have to satisfy the min and max delay constraints.



$$T_{CLK} > T_{CQ} + \max(T_{add}, T_{mod}, T_{log}) + T_{set-up}$$

Assuming equal delay ($T_{add}=T_{mod}=T_{log}=T$) and a negligible overhead:

$$T_{CLK} > T \Rightarrow f_{max} \cong \frac{1}{T}$$

Since constraints have to be fulfilled in each part, the worst critical circuit is the one related to the worst delay, and since the circuits are the same it is the same and unique.

$$T_{clk} \geq T_{cq} + T + T_{setup}$$

The maximum operating frequency if the sequential overhead is negligible, so Tcq << T and Tsetup << T is 1/T, much greater than before, we gained a factor of 3 in terms of propagation frequency.

The disadvantages are in terms of area, since we are adding FFs, then also power consumption and latency, because in the previous case after one ck period the signal was at the output (two clock rises), while in this case after one clock period we have available only the result of the sum of A and B. It doesn't take more time, but more clock cycles for the signal to reach the output.

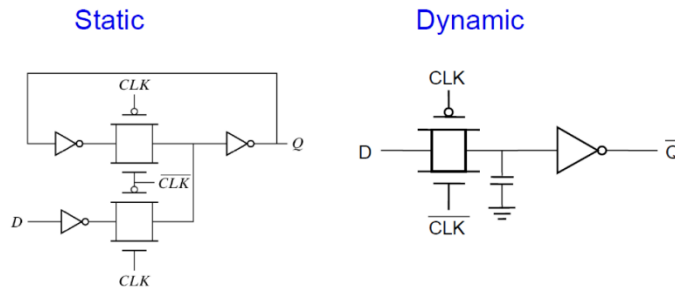
However, this last circuit has a bigger throughput, we have more bit per second.

PHYSICAL IMPLEMENTATION OF A FLIP-FLOP

It can be built combining two latches in cascade, so we need to implement a latch first.

We can implement either a static or a dynamic latch; the former is based on a bistable element, which are two inverters in cascade.

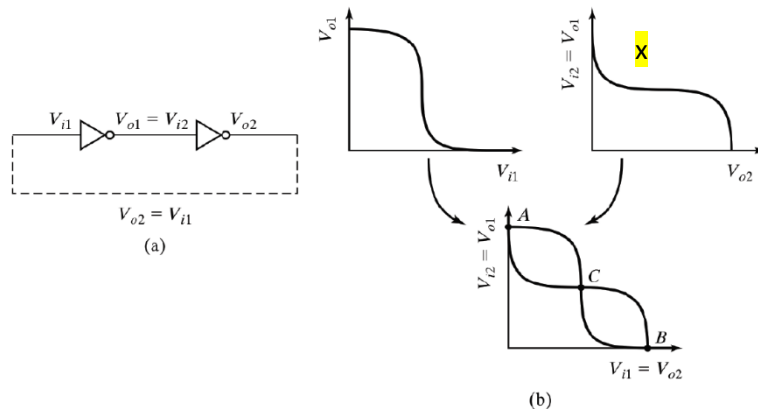
Instead, dynamic implementation relies on storing a bit charging or discharging a capacitance by means of a switch.



- ✓ Static mechanism is based on positive loop (bistable element)
- ✓ Dynamic mechanism is based on charge stored on a capacitance

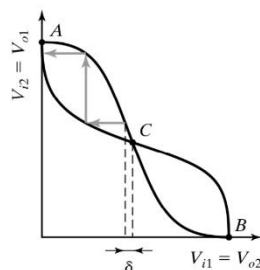
Bistable element

It is the mechanism used in the static latch; two inverters in cascade represent a positive feedback loop. X is the inverse of the static characteristic of the second inverter.



Then we put the two characteristics on the same plot. We have 3 working points, which are the possible bias point. For sure, point A corresponds to $V_{i2} = V_{o1} = V_{dd}$, and $V_{o2} = 0V$. Point B corresponds to have $V_{i1} = V_{dd}$, $V_{o1} = 0V$ and $V_{o2} = 0V$. A and B are two stable points; if we apply a small variations of these points, the circuit is able somehow to regenerate the circuit and go back to the original point.

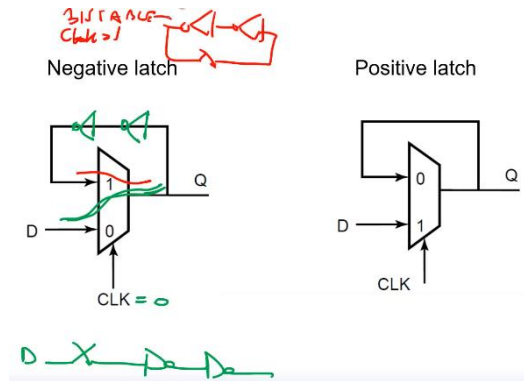
For point C we have V_m everywhere, so each inverter is biased at the switching threshold. Now, if the variation of threshold for V_{i1} is slightly positive, V_{o1} goes to $0V$, and $V_{o2} = V_{dd}$; consequently, $V_{i1} = V_{dd}$. Hence working point C collapses towards V_B . Viceversa if the variation of V_m is negative, we move towards A. So the condition with all the inverters biased at V_m is unstable, it is a **metastable solution**.



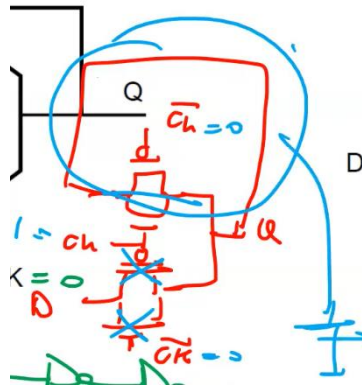
Gain (in absolute value) should be greater than 1 in the transition region of the inverters to avoid metastability

MULTIPLEXER-BASED LATCH

Let's use the bistable element to implement a latch. Let's add a multiplexer to the two inverters in cascade; the multiplexer allows the bistable element to be transparent or opaque. If we look at the image, when $ck = 0$, the bottom path that connects D is on, and the loop is cut. Instead, for $ck = 1$ we have a loop, so two inverters in feedback loop (positive feedback). In this latter case we have a bistable element. So the multiplexer allows to cut the loop, making the latch transparent, or closing the loop, making the latch in the hold phase (opaque), because the loop is not connected to the input and any variation of the input is not sensed.

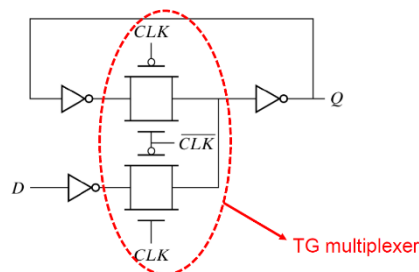


For the negative latch, during the hold phase $ck = 1$, in pass transistor logic implementation without inverters we have, in an electrical model we have that the blue circle is like a capacitor. So this is a circuit that is not reliable, because we don't have a low impedance path to Vdd or ground, so with digital noise present the circuit can fail. So to the reason to use two inverters is to have a static gate.



Real implementation

In this case we recognize a positive static latch, because the circuit is transparent when $ck = 1$ and we have the two inverters. The multiplexer is based on TG implementation.

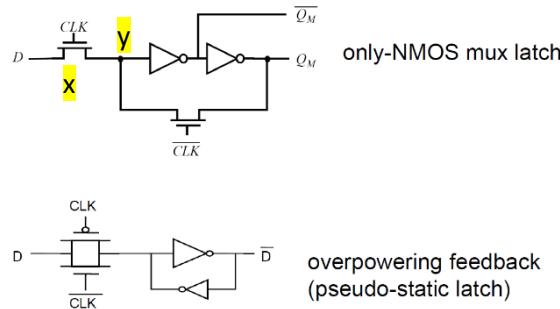


The inverter after D is not needed, but it acts as a buffer to read the voltage at D as high impedance. It is not mandatory for the correct work of the gate, but needed to avoid an increase of the delay that is quadratically increasing with the number of transistors.

For a single latch we are using 3 inverters and 2 TGs, so 10 transistors. So large area and power consumption.

Moreover, in this case, we have 4 transistors connected to the clock, plus the inverter to generate $\overline{ck_bar}$. So to ck and ck_bar we have 6 transistors connected. ck and ck_bar transition every clock cycle by definition, so with switching activity of 1, so the capacitances connected to them have a particular importance in terms of power consumption. They represent a figure of merit of the latch.

Alternative implementations

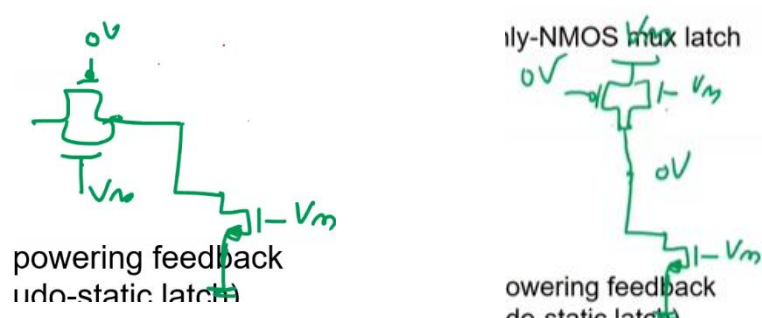


In the first case, the difference relies in the multiplexer, which is made of only nMOS transistors (the two in the image). This leads to some benefits, such as smaller area and power consumption. There are however some issues. In fact, we have only nMOS transistors, and the PU is compromised.

Let's suppose to have V_{dd} in input D , ck high (so x is closed), while the other nMOS is open. At node y we have a capacitance and it can charge up to $V_{dd} - V_{th}^*$ affected by Body effect. The output of the first inverter goes to $0V$ and the output to V_{dd} , if y is above threshold.

So the bit in the latch is correct, but we have cross conduction current in the first inverter, so I reduced the dynamic power consumption but increased the cross conduction current. Moreover, the PU is slow, so we have problems in terms of propagation delay (T_{cq} is worsened).

As for the other bottom implementation, we have the bistable element but only one switch. If the switch is closed we have a bistable element (two inverters in feedback loop). The problem is the writing of the latch in the transparent phase. We have a negative latch (switch is on if $ck = 0$). We have a problem because after the switch we have both a low impedance and high impedance path. Let's suppose we wrote a 0 and we want to write 1. At the beginning we have the following situation.



From V_{dd} to ground we have a voltage divider, actually. The $0V$ in the middle must be pulled up up until the switching threshold of the inverter of the forward path.

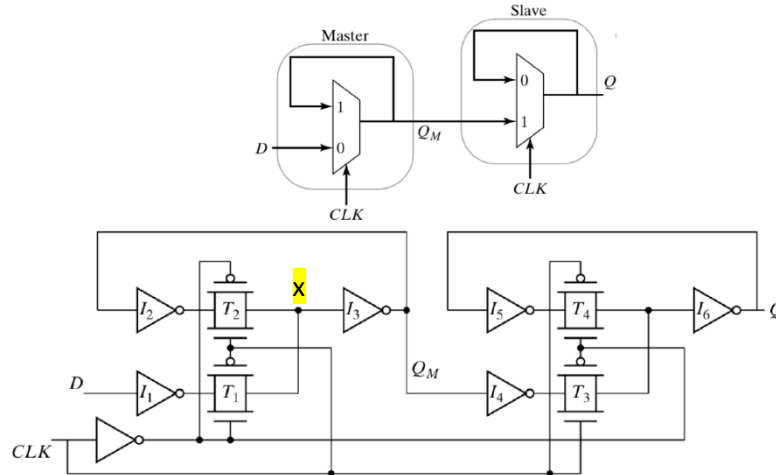
So the two transistor of the PTL must be stronger than the PD to win the PD and bring the voltage in the middle up to the switching threshold of the inverter. So it is a **ratioed solution**.

So the switch (and eventually the inverter before it to decouple the impedance) must be stronger than the feedback loop to write the latch.

The bottom topology is powerful in terms of number of transistors connected to ck and ck_bar , only 2 (like in the one at the top). Moreover, it is very similar to what we will see in SRAM cells.

MASTER-SLAVE STATIC FLIP-FLOP

We put in cascade two latches; in this case the master is a negative latch and the slave is a positive latch. We get a positive edge triggered FF.



The slave establishes the type of FF. Actually, to implement the bistable element we need two inverters in the loop of the latches, as seen before. Then we also need buffers before the TG switches, and another one to implement ck_bar . In the image the MUX have 0 and 1 inverted, it is like having them the same and driving with ck and ck_bar one and the other. This is in the end a FF, able to store a single bit, and it is made of 22 transistors (a lot), so FF are used only for foreground memory, not background memory, to store only a limited amount of bits.

Let's now assess the characteristic times of the FF, setup, hold and propagation delay.

As a first remark, it is very difficult to assess them because:

- Definition of setup time T_{su} is complicated, too generic.
- Once we have transistors in cascade, the propagation delay is difficult to be assessed.

Let's suppose to have blocks with a propagation delay that is T_{tg} (for the transmission gates) or T_{inv} (for the inverters), for the sake of simplicity.

Setup time T_{su} and T_{cq}

Let's start from the setup time. Since we have different inverters with different loads we need to distinguish between one inverter and the other (T_{i1} , T_{i2} , ...), and the same for the gate.

The setup time is the time the signal needs to remain stable in order to acquire correctly the signal. The sensitive edge of the ck signal is the rising edge.

Before the first edge, the first latch L1, being negative, is transparent. When $ck = 1$, the L1 becomes opaque, and the opposite for latch 2 (L2).

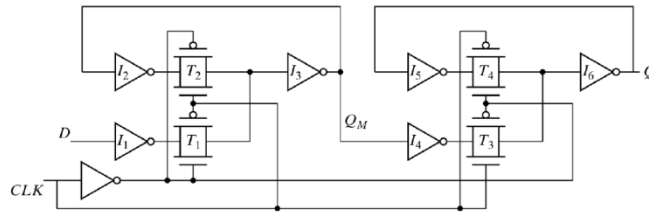
When transparent, switch T1 is closed, and T2 is open. Let's now suppose that 1 is stored at x, and D transitions from 1 to 0 before the rising edge of the ck signal.

How much close can we put the D change to the rising edge of the clock?

It's a matter of assessing the propagation delay of the circuit. In fact, we want that when the ck signal transitions from 0 to 1, all the signals in the master latch are transitioned, so that when the rising edge of the clock happens the loop has been written correctly.

I need to guarantee that before T2 switches on, the signal from the input D has reached the output of the inverter 2. So T_{su} is equal (better greater) to the delay of the inverter 1, the TG1 and the inverters 3 and 2.

If not, and the edge is too close to the edge of the clock, we are not able to correctly toggle the inverters and we are not sure about the logic level we are storing.



$$T_{set-up} = T_{I1} + T_{T1} + T_{I3} + T_{I2}$$

$$T_{hold} = -T_{I1}$$

$$T_{CQ} = T_{T3} + T_{I6} \quad (\text{if } I_2 \text{ and } I_4 \text{ have the same delay})$$

If the previously mentioned condition is fulfilled, once T2 closes the data is stored correctly and the output of I3 can reach the output, because the second latch (slave) becomes transparent.

Now we can assess the propagation delay starting from the edge of the clock for L2, because the signal in input to L2 is stable if the constraint is fulfilled.

From when the clock transitions from low to high to when also Q does this, the delay is considering the $T_{i4} + T_{tg} + T_{i6}$, but if T_{su} is very large, so i_4 is very fast because $T_{i4} < T_{i2}$, T_{i4} can be neglected, because we have already Vdd at the input of the TG after i_4 .

So if $T_{i4} < T_{i2}$, the propagation delay is made just of 2 terms.

In the end, what counts is the sum of T_{su} and T_{cq} . This is a very reliable FF but also very slow, because the delay is very large due to the lot of elements in cascade.

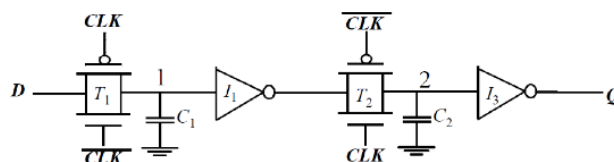
Thold

T_{hold} is how much I have to keep the data stable at the input after the rising edge of the clock. When ck transitions from 0 to 1, T_1 turns off. Once this happens, for how much do I have to keep the data stable at the input? Since we have inv_1 , the hold time is negative, because the input can change before the transition of the clock because the inverter is not transitioning up until a time T_{i1} , so the latch is insensitive because it takes a delay T_{i1} to reach the input of the TG1.

If we remove inv_1 , the $T_{hold} = 0$.

If we violate the T_{su} or T_{hold} condition, either the signal is not acquired even if the signal transitions before the sensitive edge (if T_{su} is violated) or the signal is acquired but it transitions very slowly, so the propagation delay explodes.

DYNAMIC FLIP-FLOP



- ✓ Smaller area than static flip-flop
- ✓ Faster ($T_{set-up} + T_{CQ}$) than static counterpart
- ✓ Smaller power consumption (fewer transistors connected to CLK and CLK')

The first one is a negative dynamic latch, the second one a positive one. The inverter I1 acts as a buffer, so both the latches are inverting.

Let's focus on the first latch. It is like a switch, a capacitance and a buffer; the basic idea is to store the bit as a charge across the capacitance. The advantage is that we have 4 transistors per latch, and moreover ck and ck_bar have to feed only 4 transistors. The second latch is positive because the switch closes when $ck = 1$ (latch becomes transparent). So the greatest advantage is the smaller area than static FFs.

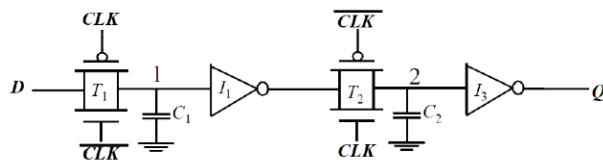
Furthermore, this implementation is much faster than the static FF, in terms of sequential overhead (i.e. the sum of T_{su} and T_{cq}).

It's called dynamic because since its working principle is the charge storage over a capacitor, which must be saved when the switch is off, when we have a floating capacitance. But since there are no ideal capacitances and switches, the capacitor is a little discharged with a RC discharge and leakage current. The leakage current is the drain-substrate diode leakage current, together with the transistor subthreshold leakage current.

$C1$ is a parasitic capacitance, that in our technology (0.25um) is 4 fF (2 fF for the inverter and 2 fF for the TG). Obviously, the larger the capacitance the larger the time it requires to be discharged by the leakage current. It is called dynamic because **the capacitor has to be refreshed continuously**. If we read the FF, we also refresh the content of the capacitor. It is a concept that will hold also for the DRAM. So ck has to be provided to this circuit to refresh them, otherwise the content is lost.

There is another drawback. There is no low impedance path to Vdd or GND at every time instant, and so a floating capacitance can be aggressed by an aggressor, e.g. a nearby transitioning wire.

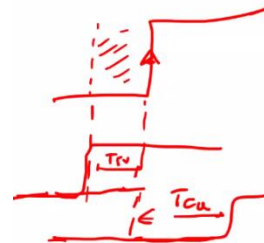
Let's assess the three characteristic times for the positive latch.



$$T_{set-up} = T_{T1}$$

$$T_{hold} = 0$$

$$T_{CQ} = T_{I1} + T_{T2} + T_{I3}$$

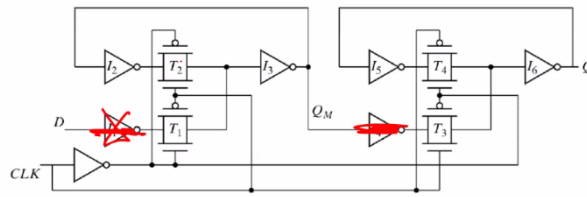


For T_{su} we need to grant that the charge is stored in the capacitor, so it is the delay of the TG.

Instead, T_{hold} is 0 because once $T1$ is open, as soon as it opens, the input D can change but we don't care because the switch is off and nothing happens in the circuit.

In this case the overall delay includes two TGs and two inverters, only 4 contributions with respect to the 6 of the static FFs. But this is not a fair comparison because in the dynamic implementation we are missing two inverters (which are not mandatory for the correct working of the gate but helpful).

So let's consider the setup condition in case of ideal D input signal and no buffers $I1$ and $I4$ in the static FF. $I4$ delay was already absent in the formula. So in any case, even if we remove the buffers, we have a larger sequential overhead in the case of static FF, because of the working principle of the latch in this implementation.



$$T_{set-up} = T_{I_2} + T_{T_1} + T_{I_3} + T_{I_2}$$

$$T_{hold} = -T_{I_1}$$

$$T_{CQ} = T_{T_3} + T_{I_6} \quad (\text{if } I_2 \text{ and } I_4 \text{ have the same delay})$$

Now, coming back to the dynamic FF, which is the function of I1 and I3? Why not remove them to spare area and power consumption?

Without them, the situation would be the following.



In this case, charge sharing occurs and at the end of the day on the last capacitance I have $V_{dd}/2$ if the first capacitance is to V_{dd} and the last one to $0V$. And this is not working.

Even if the first capacitance is $100 \cdot C$ and the second is C , after charge sharing we have V_{dd} on the second capacitor, so it is good, but if we have other stages it is a nightmare. So the buffer is needed for sure.

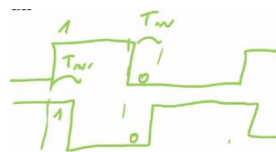
What happens if the signal transitions very far from the edge?

The T_{su} condition is not violated, but T_{cq} is not the same as before. Not only we reach C_1 , but there is also time to reach the output of I1. Then, on the rising edge of the clock T_2 closes and we need to travel only across T_2 and I_3 , so T_{cq} is smaller because the T_{i1} has to be considered negligible, because the transition of D has occurred very far from the edge, so there is time for the signal to reach the point at the output of I1.

Problems

In general, for both static and dynamic FFs, we have some problems. For instance, ck_bar is generated via an inverter, which however is generated with an inverter with delay. So there is a time where ck and ck_bar are both equal to 1 or 0.

So ck and ck_bar are not synchronous, there is always a delay.



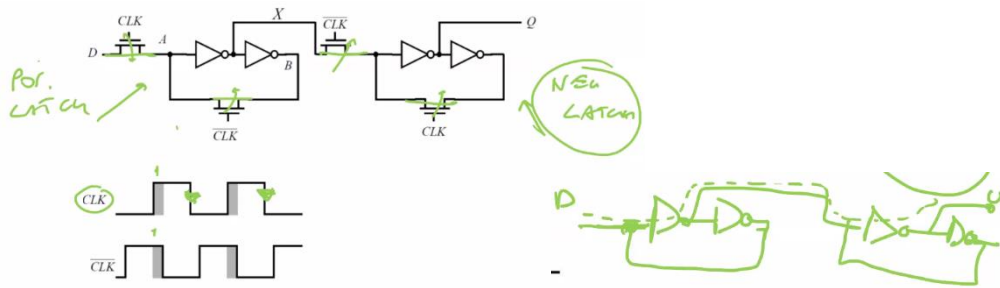
Clock overlap issue

Static FF

The first one is a positive latch (transparent when $ck = 1$) and the second is a negative latch, so we have a negative edge-triggered FF.

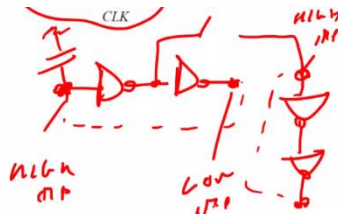
Due to the overlap (1-1 in the first case), we have the following behavioural standpoint, with all the transistor closed.

The main problem is that there is a direct connection, if we neglect the feedback because it is e.g. not too strong, between the input and the output. This is like sampling the input of the non-sensitive edge of the FF, which is a very bad misbehaviour.



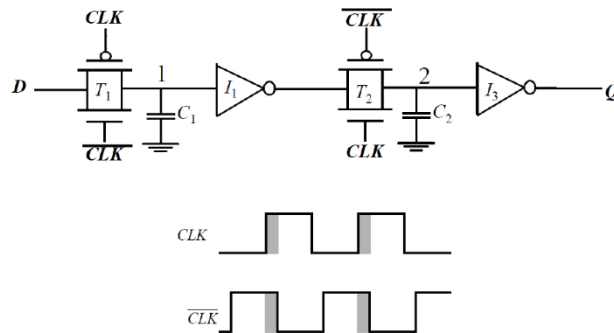
- Overlap of CLK and CLK' can lead to critical race
- Input signal can be sampled on the non-sensitive edge

Instead, in the case of overlap 0-0, since the MUX is made of only nMOS transistor, we don't have bistable elements in cascade, because the feedback is off because all the switches are open. Hence we are violating an essential condition for static gates, since there is no low impedance path that connects the output to either Vdd or GND. So during the overlap 0-0 we have a dynamic gate.

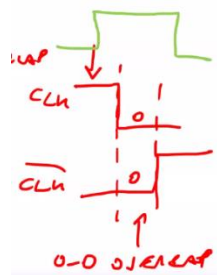


If we had clocks without the 1-1 overlap (the 0-0 is not a big issue, so it can be present), we are good, and this signal can work only for this FF with nMOS in the MUX.

Dynamic FF



Negative latch and positive latch, so it is a positive edge triggered FF. In this FF, 0-0 overlap can be a problem. In this case, the TGs have the nMOS off and pMOS on, so I have a path from the input to the output. If the overlap 0-0 is large enough, we sample the input signal in correspondence of the negative edge. Let's sketch the overlap and a signal that transitions across the overlap.



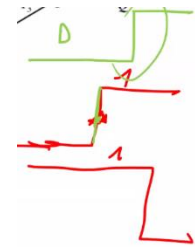
If the FF works correctly (without 0-0 overlap), a transition in the middle between two rising edges is not sensed. In this case, before the 0-0 overlap, when $ck = 1$ and $ck_bar = 0$, T1 is off and T2 is on, so the FF is working correctly.

During the overlap, $ck = 0$ and $ck_bar = 0$, so the T1 has the nMOS still off, but the pMOS is on, while for T2 the nMOS is off and pMOS is on \rightarrow direct connection between D and Q.

Now, if the signal D is able to reach the second latch on C2 (during the overlap duration), so if $T_{00} > T_{t1} + T_{i1} + T_{t2}$ (T_{t1} and T_{t2} are the delays of only the pMOS transistors, not of the TG), the signal on C2 is sampled.

However, the voltage has been sampled on C2, and so the signal is free to travel to the output. So the output was 0, and it goes to 1. So the signal D is sampled, and this is completely wrong, because we are sampling it in correspondence of the non-sensitive edge of the FF.

Moreover, this FF is subject to a misbehaviour also in the case of the 1-1 overlap. Let's consider a signal that transitions immediately after the positive edge of the ck. It is not sampled if we have a correctly working FF, but in our case we can suppose to have a large overlap.



When $ck = 0$ and $ck_bar = 1$, T1 is on and T2 is off. After the ck rising edge, $ck = 1$ and $ck_bar = 1$, so the nMOS of the TGs are on, the pMOS are off.

During the overlap, D transitions. Then, in the last period of time when $ck_bar = 0$, the T2 is on and T1 off, so the second latch becomes transparent in both the pMOS and nMOS transistors.

It is enough that we reach the point on C1 and the misbehaviour occurs, because even if the first latch becomes opaque, the value on C1 is transmitted along the chain to the output. Hence the condition is more stringent. So if $T_{11} > T_{t1}$ we have a misbehaviour.

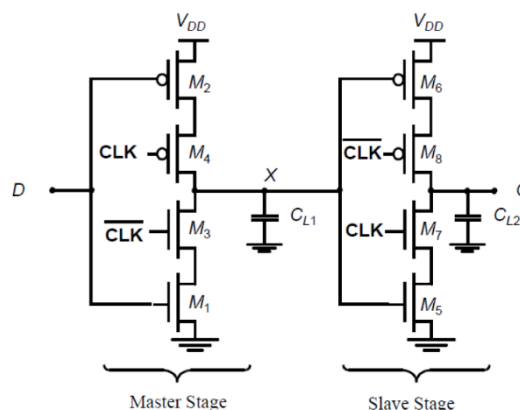
To solve this issue, we can set a T_{hold} for this FF, saying that $T_{hold} = T_{11}$, to ensure that the signal is not transitioning in the overlap period (for positive edge triggered FFs).

However, we cannot do anything for the 0-0 overlap. So we need to implement a circuit that is insensitive to the clock overlap. We have two alternatives:

- **C²MOS**: the problem is solved with a different disposition of the transistors, so it is a dynamic implementation with 8 nMOS transistors with a different arrangement.
- **True single phase clock FF (TSPC)**: we use just a single clock, not ck_bar . Also this one is a dynamic implementation.

C²MOS FLIP-FLOP

The implementation is always master-slave and it is based on tristate inverters. The first one is a classical inverter if $ck = 0$, so it is a negative latch. Conversely, the second one is a positive latch, since it is an inverter if $ck = 1$.

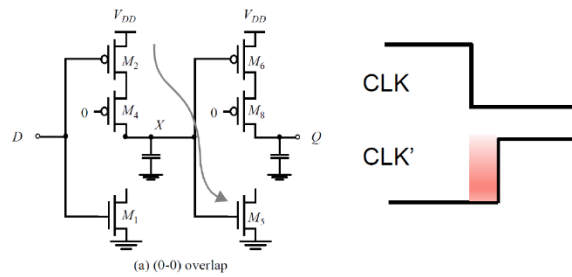


For the master stage, with $ck = 1$ it is in hold phase, viceversa the slave stage for $ck = 0$.

Let's now verify that the C2MOS FF isn't suffering from clock overlap.

0-0 overlap

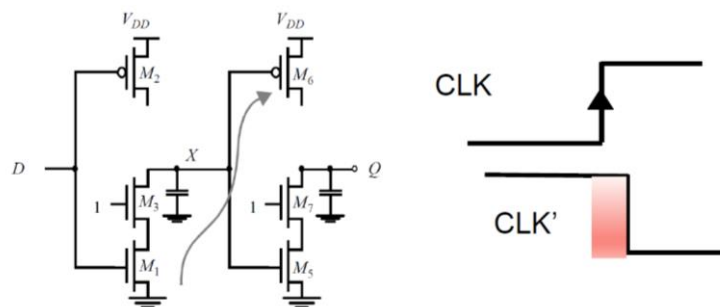
So we have a positive edge triggered FF, so the most problematic overlap is this one, since the sensitive edge is the rising one.



✓ 0-0 overlap is solved because of 2 pull-up devices in cascade that cannot be ever on at the same time

Before the transition to 0, M4 and M3 and M7 are off. the first latch is opaque, so even with a transition of D nothing happens, and this is the normal behaviour. Then we enter in the 0-0 overlap. D in the meantime has transition from 1 to 0. The situation is the one depicted in the image above. Signal D cannot reach Q because X can only be pulled up, since we have a PUN attached to it. But if X is pulled up, the second PUN switches off, so Q remains floating. Since we have to PUNs in cascade we cannot have a cascade from input to output, because two PUNs cannot be turned on at the same time. Hence the signal D is not sampled by the output Q. If D instead transitions from 0 to 1 we switch off the first PUN.

1-1 overlap



✓ 1-1 overlap seems solved because of 2 pull-down devices in cascade that cannot be on at the same time

✓ But as CLK' is pulled down, M₈ turns on. If X has been discharged during the overlap, it propagates to the output causing the sampling of the input signal after the sensitive edge

✓ The only solution is to set a hold time, $T_{hold} > T_{overlap}$ 33

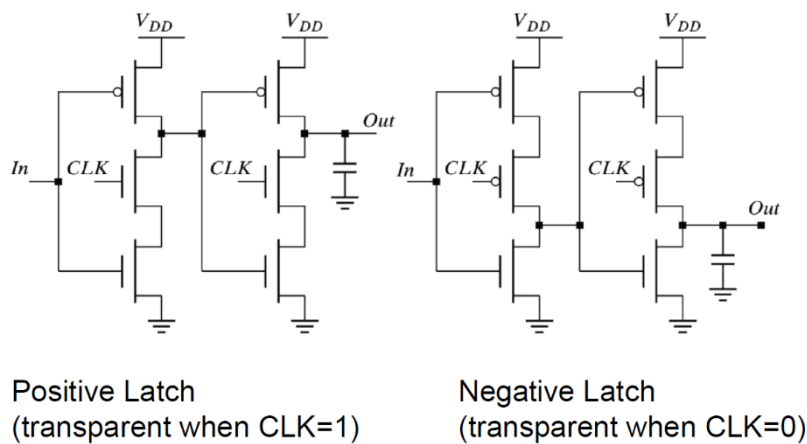
The two PDNs are on depending on the signal. If D transitions from 1 to 0, M1 is off and we are not sampling D, so X remains floating and nothing occurs. If instead D transitions from 0 to 1, the first PDN is on and X is pulled down. If so, the second PDN is deactivated, so Q is not changing.

So it seems that the issue has been solved. X has been pulled down during the overlap, because the overlap is large enough, so Q is not transitioning. After the overlap, ck_bar = 0 and ck = 1. Consequently, M8 turns on, M6 is on and we are writing a 1 in the output, a 1 related to the D transition.

So we have solved the 0-0 overlap but not the 1-1 overlap when the input transitions from 0 to 1. The only solution we can exploit is to set a Thold to be Thold > Toverlap₁₁.

Hence the overlap issue is solved completely for the 0-0 overlap, the 1-1 only setting a hold constraint, not only due to the design of the FF.

TRUE SINGLE PHASE CLOCK LATCH (TSPC)



Positive latch

When $ck = 1$, the two nMOS transistors are on, so I have the combination of 2 inverters in cascade, so the input is sampled on the output capacitance. When $ck = 0$, the nMOS are off; if $In = V_{dd}$, the pMOS is off, but since the transistor in the middle is off it's like if the latch is opaque, so nothing is transferred to the output. In case of $In = 0V$, it is the nMOS at the bottom to be off and the internal node is pulled to V_{dd} . On the second part, the nMOS is on but the intermediate nMOS is off, hence the output is a floating capacitor.

Negative latch

For $ck = 0$ we have two inverters in cascade and the input is sampled on the output node.

For $ck = 1$ we are in the opaque phase; the two pMOS transistors are off. For $In = V_{dd}$, the pMOS on top is off, the nMOS on the bottom is on, the internal node is pulled to $0V$ and consequently the second stage has the pMOS on and the nMOS off, so the output is floating.

For $In = 0V$, the output is not directly connected to the input since the nMOS is off and the pMOS that is on is not directly attached to the intermediate node, and consequently the output.

Let's try to change the connection. From a logical standpoint this circuit is working, because for $ck = 1$ it is a transparent latch. For $ck = 0$ the two transistors in the middle are off; if $In = V_{dd}$, the output is connected to two transistors that are off, so it is in the opaque phase.

However, the previous topology is much better than this one because the intermediate node in the second case is not full swing.

In fact, if $ck = 1$ and we are in the transparent phase, let's consider $0V$ as input.

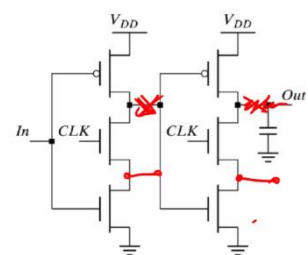
The nMOS at the bottom is off and the pMOS and nMOS top and middle are on, and the intermediate node reaches $V_{dd} - V_{tn}^*$ that is $1.8V$ more or less.

The output voltage, actually, is close to $0V$; moreover, also the pMOS at the top of the second stage and the middle nMOS are on, so we have a cross-conduction current (static power consumption).

Moreover, since the intermediate node is not fully driven, we have worsen reliability and the propagation delay of the latch is compromised when pulling down the output node because the output stage is not fully driven.

In this circuit (if we connect them in master-slave configuration) the clock overlap problem is solved. Furthermore, we have more transistors than the C2MOS topology which, however, is not solving the clock overlap issue.

Let's assess the characteristic times.



Positive Latch
(transparent when CLK=1)

The storage capacitance in the image are parasitic capacitances, and we need to store the signal on them to have the correct working of the gate.

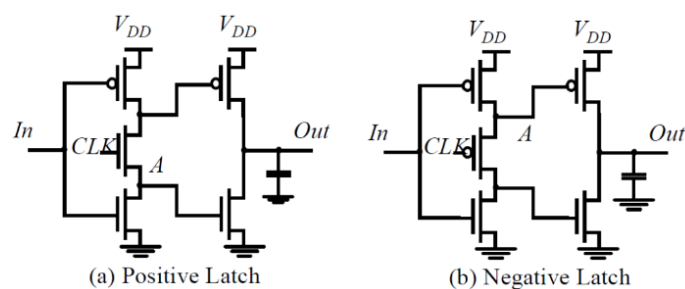
The setup time is the propagation delay of the two inverters of the positive latch, because we must have a stable signal stored on the intermediate capacitance (output node of the first latch) before the falling edge. The worst case is with 0V in input, not Vdd.

As for T_{hold} , for sure it cannot be greater than 0, because there is no sense in keeping the value stable after the falling edge. $T_{hold} = 0$ in the worst case, which is the one with $In = V_{dd}$.

As for T_{cq} , we have just to reach the output from the middle position, so we have the delay of the second latch.

Can we spare some transistors in this topology? Yes.

TSPC split output latch



- ✓ Reduced number of transistors connected to CLK
- ✓ Reduced swing of node A
- ✓ Decreased noise margin, increased delay

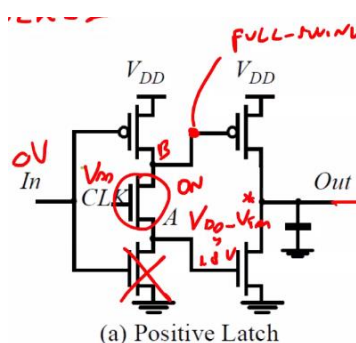
We have 10 transistors to implement a FF, and only 2 connected to the synchronization signal (clock), which is an important figure of merit in terms of power consumption.

Being a latch, we have always to consider the two phases:

- $Ck = 1$. The positive latch has two inverters in cascade and it works, we are in the transparent phase
- $Ck = 0$. We are in the opaque phase. We have two PUN in cascade and two PDN in cascade (not in series). The cascade of 2 nMOS transistors cannot have both the transistors on at the same time, so one has to be off, and it's ok.

The drawbacks are in the phase where $ck = 1$, that is the transparent phase. When the input is 0V, the intermediate nMOS is on, node B is able to reach Vdd (full swing), but A reaches $V_{dd} - V_{tn}^* = 1.8$, so it is not full swing. When instead the input is Vdd, node A is at 0V, and the voltage on B is pulled down to 0V, because we have two nMOS in the pull down path (pMOS transistor is off).

So the problem is in the PU, when A is not able to reach Vdd because we have an nMOS in the middle.



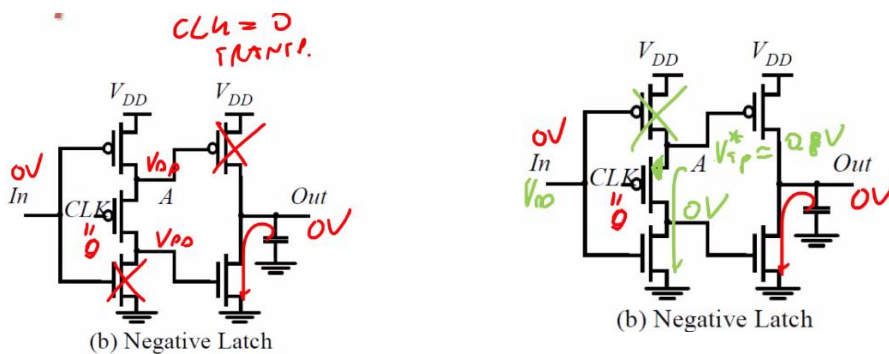
Since 1.8V is above the threshold of the inverter, the output is exactly 0V because the top pMOS is off, and since there cannot be current in the series, the output reaches exactly 0V.

In any case, we have reliability concerns and dynamic performances worsened, because the PD is performed by a nMOS that is not fully driven (large equivalent resistance).

Since the delay is increasing, we are worsening the setup time and Tcq (for a different phase of the input signal).

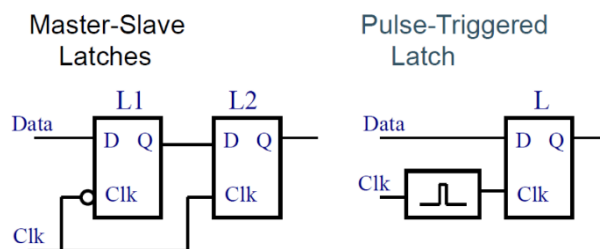
The negative latch has a similar behaviour than the positive one, and when $ck = 0$ we have similar problem. In this second stage it is the PU that is compromised. Let's consider the transparent phase for the second latch, so $ck = 0$, and $In = 0V$.

The second stage is fully driven and the output node is pulled down to 0V, so perfect behaviour. The problem arises when we have Vdd at the input, because A is not able to reach 0V since we have a connection to ground via a pMOS and nMOS, so it stops at V_{tp}^* . So it is the pull-up that is compromised.



PULSED FLIP-FLOP

So far we have seen FFs implemented with latches in cascade in a Master-Slave configuration, but it is not the only way to implement them.



The signal I connect to the latch is the output of a pulse generator, not the classical square wave, and during the pulse the latch is transparent. So we are not combining latches with different polarities. In correspondence of the rising edges I have small pulses at the input of the clock for the latch, and the latch is transparent only during the short pulse, and opaque in the rest of the period.

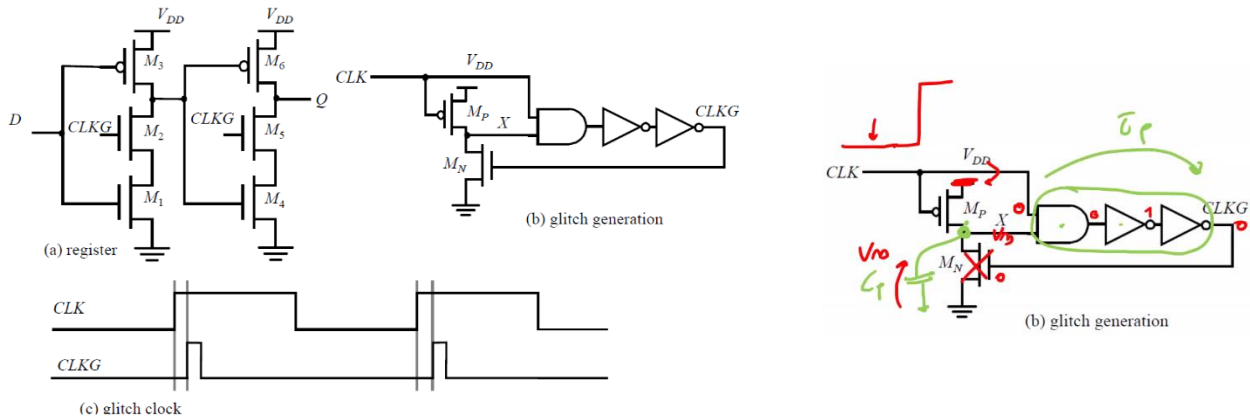
I can have a lot of latches in the circuit and I can generate the pulsed clock that is then delivered to all the latches. This allows a great saving of area, because it is like if a single FF is implemented with just one latch, since the glitch is shared.

The latch can be e.g. implemented with a TSPC latch (on the left of the image below, that is a positive TSPC), and the clock with a **glitch generation**. The input is clk and the output clkg, pulsed clock we deliver to the latch. The input is a squarewave signal, the output a glitch signal. There is a delay because it is a real circuit.

Glitch generation

So, the NAND and the inverters are used to implement a delay τ_p , and let's consider the capacitance C_p at node X as negligible (fF).

Let's consider ck transitioning from 0 to 1. When $ck = 0$, I have 0 at the input of the AND gate, so $clk_g = 0$, so the nMOS on the bottom is off, the pMOS is on and C_p is charged to V_{DD} . This is at steady state.



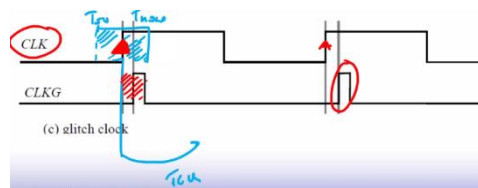
On the rising edge, the clk_g transitions to 1 after a delay τ_p with respect to the rising edge of clk . The nMOS turns on and discharges C_p , so X goes to 0, it's pulled down. After another delay τ_p , $clk_g = 0$ again and it happens as before, so C_p is charged to $0V$, with both the pMOS and nMOS off.

The situation changes when clk gets back to 1 (after being reset to 0).

Pro and cons

The FF is implemented with just one single latch, because the glitch generator can be shared. Since it is shared, a possible drawback is that it is indeed a short pulse. If the number of latches changes, the pulse is not exactly a pulse; because of the short duration, the delay counts a lot. If the load changes we could have something that is no more a pulse, so it is harder to generate a short pulse.

The rising edge that counts is the one of the clock, so all the times have to be assessed starting from the clk rising edge, not clk_g . So also in this case there is a T_{su} before the rising edge of the clock, a hold time after it in which the data must be stable to be correctly acquired, and the data is presented at the output after T_{cq} .



How much is T_{su} , referring to the rising edge of the clock?

It is 0 because when $ck = 0$, also $clk_g = 0$, so there is no reason to keep the data stable, because the data is not influencing the output. In reality, $T_{su} = -\tau_p$ because we can have a transition since the latch is not transparent between when the clk rises and when clk_g rises, so there the latch is opaque and not transparent. So there is a minimum amount of τ_p which must be at least $\tau_p \geq T_1 + T_2$.

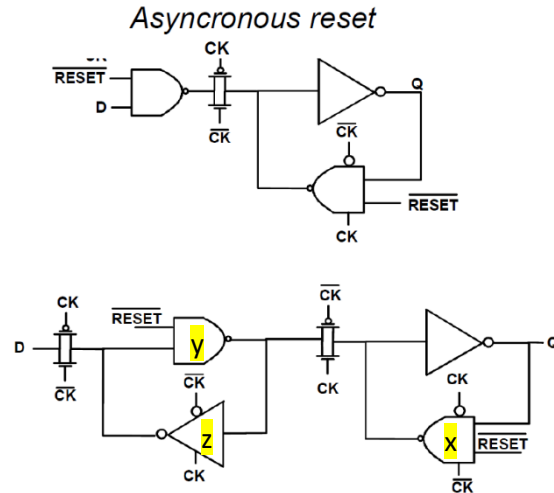
How much is T_{hold} ?

$2 \cdot \tau_p$, because the latch becomes transparent in correspondence of the pulse, so in the red region.

How much is T_{cq} ?

$T_{cq} = \tau_p + T_1 + T_2$, where T_1 and T_2 are the delays of the two 'inverters' in the latch.

LATCH/FF WITH RESET



Let's focus on the FF on the bottom.

Z is a tri-state inverter, that is an inverter followed by a transmission gate. Also x is a NAND followed by a transmission gate.

The second difference is that, instead of implementing the feedback path with two inverters to implement the bistable element, we use an inverter as a NAND gate driven by the signal that circulates inside the loop and the other input connected to the asynchronous reset. The NAND gate behaves as an inverter if one of the two inputs is set to 1.

Let's verify we can bring the output to zero applying a short pulse to the reset_bar input.

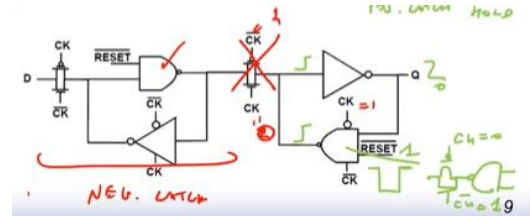
We need 2 NANDs connected to reset_bar to have the output set to 0 because the FF has two states, since made by two latches, negative and positive, one transparent when clock = 1 and the other one when clock = 0. This means that we need 2 circuits to be sure that the output goes to 0 because if the second latch is transparent, the NAND x is off and so I have just the upper path (when $\text{ck} = 1$), so the circuit in the feedback path is not useful to bring the output to zero, so I need the NAND y that is connected directly to the forward path.

In the opposite case, when clock = 0, the second TG is off, and the second latch is in hold phase, hence NAND y is useless. But in this case the feedback through NAND x is on, and so the output is brought to 0.

Ck = 0

The feedback path of the second latch is on. If I apply a small negative pulse on the reset_bar, the output of the NAND x goes to 1, whatever the other input of the NAND. If so, the output of the inverter goes to 0, which is expected. When reset_bar gets back to 1, the other input has transitioned to 0, so the output Q remains to 0.

In this framework, the minimum duration of the pulse is the duration that allows the signal reset_bar to transition after the NAND and the inverter to reach Q. so it is $T_{\text{nand}} + T_{\text{tg}} + T_{\text{inv}}$.

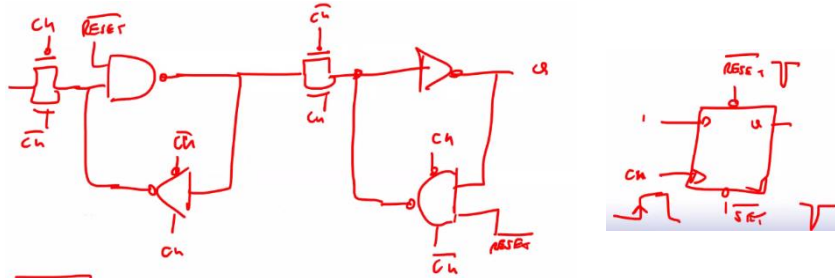


The approximation we are making is that the NAND and the transmission gate have independent propagation delays.

Ck = 1

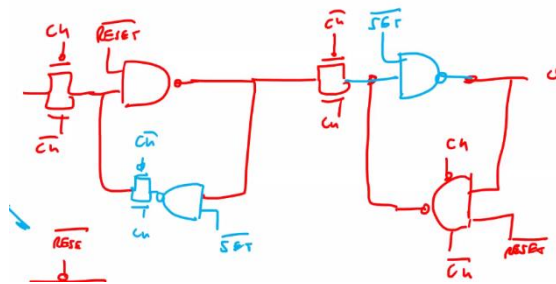
The first latch is in hold phase (first TG is off), the second latch (positive) is transparent, so the x NAND is off and the y NAND is on. If on the reset_bar of the y NAND I apply a negative pulse, the output of the NAND goes to 1, the 1 is sampled by the transparent latch and Q goes to 0. Also in this case, reset_bar has to be low for a minimum amount of time, that is the one needed to complete the loop of the first latch.

How can we implement a set input, or both an asynchronous set and asynchronous reset?



This is a FF with asynchronous reset, and I want to add a SET_bar input, able to bring the output to 1 whenever I want, whatever the state of the FF. We need asynchronous inputs for reset or startup purposes.

Now we want to add a set input and verify that the set is working in the two cases.

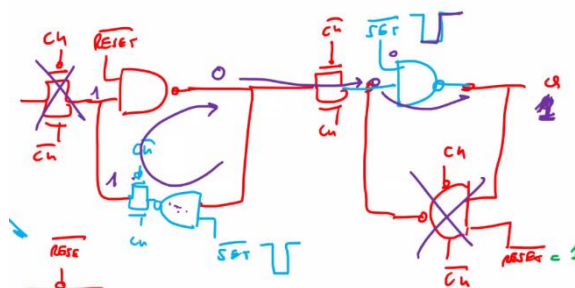


Ck = 0

L2 is in hold phase, so the small pulse is applied on the set, which goes to 0, so the output of the NAND goes to 1 (second blue NAND), the 1 travels and 0 returns in input to the input of the second blue NAND, having the output to 1 fixed.

Ck = 1

Now it's the master that is working and active, L1 is hold and L2 is transparent. When SET_bar of the second blue NAND goes to 0, Q = 1, but it is not fixed to 1 because there is no feedback path. This is the reason why the master must work, to fix Q = 1.



TIMINGS

CLOCK NON-IDEALITIES

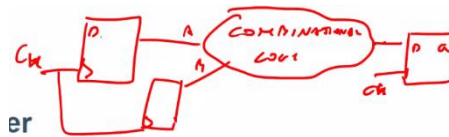
□ Clock skew

- Spatial variation in temporally equivalent clock edges; deterministic + random, t_{SK}

□ Clock jitter

- Temporal variations in consecutive edges of the clock signal; modulation + random noise
- Cycle-to-cycle (short-term) t_{JS}

Let's suppose to have a sequential circuit made of a combinational logic with more than 1 input and the output that is received by a FF. all the FF are connected to a ck line.

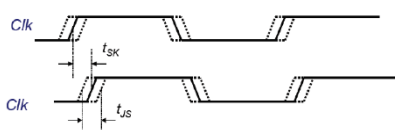


So far we assumed the ck to be ideal. However, ck is subjected to 2 non idealities:

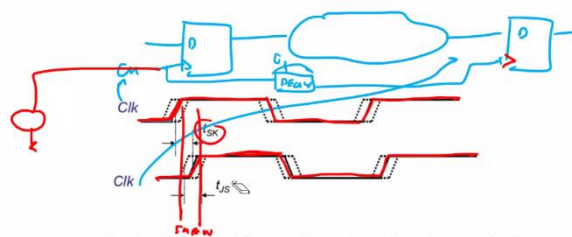
- **Clock skew:** spatial variation in the clock edges. For instance, let's suppose we have two FF on one side of the chip and the receiving FF on another side, far away. In the middle there is a long wire, so a delay. So it is like if different FF in the same chip receive a clock signal with a non-perfectly aligned edges. This delay is called clock skew. There can be either a positive or a negative skew. It is positive skew if the clock travels in the same direction of the data, otherwise it is negative. Skew can be intentionally used to speed up a pipelined circuit.
- **Clock jitter:** it is random noise. The clock period has a mean value and a statistical variation around this mean value.

SKEW AND JITTER

The bold line represents the input ck. Due to the delay of the wire, the receiving FF senses a clock that is a delayed version. The difference is the skew, t_{sk} .



- Both skew and jitter affect the effective cycle time
- Only skew affects the race margin (usually)



The skew relates to the fact that the ck period we measure is sometimes large, sometimes smaller. So the delay is deterministic, while the jitter is stochastic, due to noise.

Both skew and jitter are effective in determining the max delay constraint. The skew affects only the min delay constraint (hold constraint).

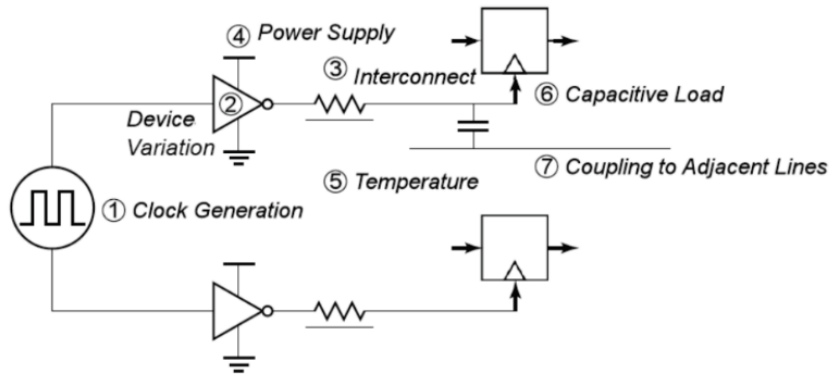
Origin of the skew

In this case the clock is considered as generated in the middle, so ideally it reaches the two FF at the same time if the paths are equal.

The ck generates a square wave that passes through an inverter and then wire and to the FF. The differences can be in:

1. Different interconnect, i.e. different wires.

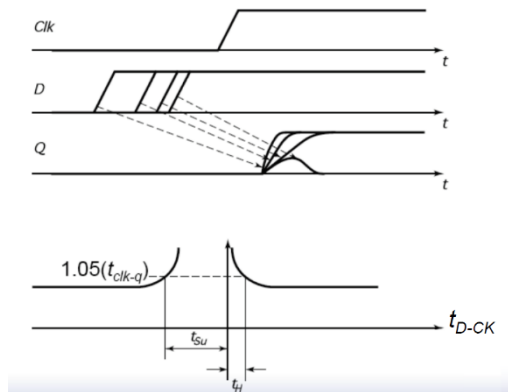
2. Different inverters used to deliver the signal to the FF. Nominally they are equal, but due to mismatches inside an IC (tolerances in the photolithographic process).
3. Temperature differences.
4. Coupling to adjacent lines. We might have parasitic capacitive coupling in one FF and not in the other.
5. Power supply, which is nominally the same but it is distributed through interconnections. If a digital circuit has different power supply voltages, the equivalent resistance changes, that is smaller when the PS is smaller.



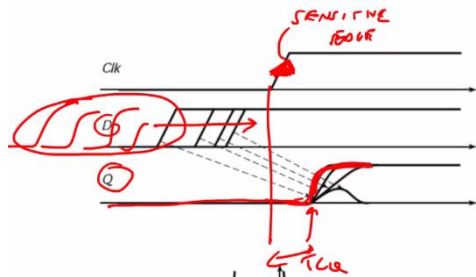
The main responsible for the jitter is the clock generation, that is the oscillator closed in a feedback loop (PLL).

Precise Setup-hold time definition

We can consider different transitions of the input D with respect to the rising edge of the clock. Once I'm very close to the sensitive edge I'm violating the setup constraint.



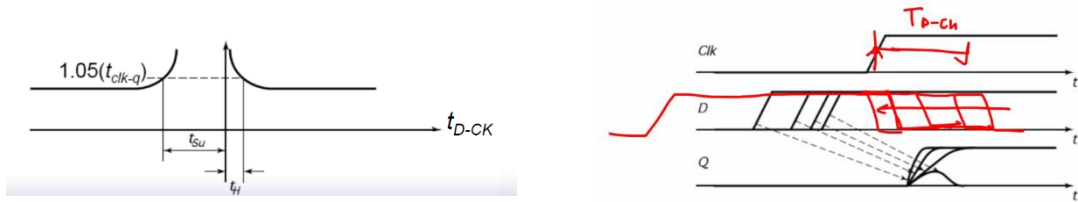
When we are very far from the edge, we expect the output to transition in the first time stamp, after T_{cq} .



On the bottom we are representing T_{cq} on the y axis, for different value of t_{D-CK} on the x axis, which is the time between the D transition and the rising edge of the clock. When we are very far, the delay is constant; then we get closer and closer and at a certain point we violate the setup constraint. When we are too close, Q is not transitioning to 1, it goes up and then immediately 0. This is explained by T_{cq} increasing up to diverging (infinite delay to transition).

In literature, T_{su} is assessed as the time where T_{cq} is above 5% the nominal T_{cq} value.

Instead, the right axis of the following plot corresponds to have D transitioning very far from the rising edge, and then going back to 0 closer and closer to the rising edge, violating the Thold constraint.

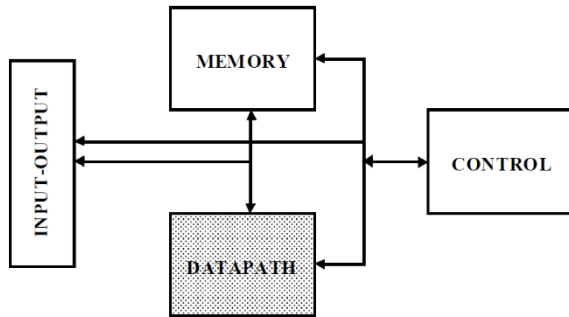


Also in this case, T_{hold} is assessed as 5% of T_{cq} value.

(see illustration on slides)

ARITHMETIC CIRCUITS

In the data path block (ALU) we can find arithmetic circuits such as adders, multipliers, shifters, digital comparators. They are all based on adders. In a digital processor we also have a memory, i/o interfaces and the control block.



Datapath (ALU)

- Arithmetic and logic operators (adder, multiplier, shifter, comparator, etc.)

Memory

- RAM, ROM, Shift registers

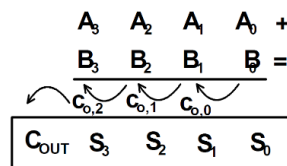
Control

- Finite state machines (PLA, random logic)
- Counters

Interconnect (Input/Output)

- Switches
- Bus

ADDERS



Cout is the carryout and it can be also 0. We are adding two ‘words’ of 4 bits. We note that the result is a 5-bit word. So if we sum two words of N bits, the result is made of N+1 bits.

Maybe we also have a carry-in (Cin) coming from another circuit when summing two words.

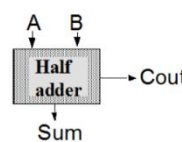
In terms of HW, to implement the sum we need a circuit that is fed by A₀ and B₀ and with two outputs, S₁ and Co,1. This is the **half adder**. In the remaining bits we have to use a **full adder**. So an half adder receives in input two bits and the result is two bits. A full adder has 3 bits in input and two output bits. The worst case is when A, B and C are all 1, because the result is 1 and 1 in output.

So half adder and full adder are the building blocks to implement an adder.

HALF ADDER

It sums two bits with the same weight, and the result is made by LSB, that is the sum, and MSB, that is the carryout. The sum goes to 1 when one of the two input bits is 1 → EXOR function.

Since instead the carryout goes to 1 when bot the inputs are 1, the logic function that represents it is the AND.



A	B	SUM	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

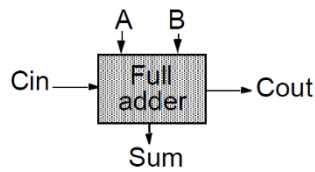
Truth table of the half-adder

The half adder can be considered a full adder with the carry-in = 0.

$$S = A \oplus B$$

$$C_{OUT} = AB$$

FULL-ADDER



A	B	C _i	SUM	Cout	Carry status
0	0	0	0	0	delete
0	0	1	1	0	delete
0	1	0	1	0	propagate
0	1	1	0	1	propagate
1	0	0	1	0	propagate
1	0	1	0	1	propagate
1	1	0	0	1	generate
1	1	1	1	1	generate

S is the "odd" function, C_o is the majority function of the inputs

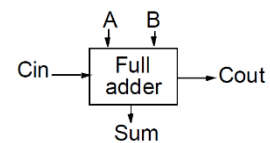
The outputs are as for the half adder. If we consider the three cases when only one bit out of 3 is 1, the LSB is always 1, since the sum of the three bits is with the same weight. Then there are other three equivalent cases when two bits are 1. A third case is the one with all 1.

We can recognize that the sum goes to 1 if we have an odd number of 1 amongst the inputs → **sum is an odd function**.

As for the carryout, it goes to 1 in 4 cases when at least two bits are 1 → **carryout is a majority function of the inputs**, i.e. when we have more ones than 0 in input.

Last column is the carry status. If we look at the first two rows, whatever the value of the carry in, the Cout = 0, so we are in the **delete condition**, also known as kill condition. Then we have 4 intermediate cases characterized from the fact that A ≠ B. In these cases, Cin = Cout → **propagation state**, since the Cin is propagated in the Cout. In the last two, whatever the Cin value, Cout = 1, so they are called **generate status**, since we generate a Cout = 1 whatever the Cin value.

As for the Boolean function for the full adder, the sum is the odd function of 3 inputs. In fact, each of the 3 terms in the odd function goes to 1 when one of the three inputs is 1. E.g. the first term is 1 if A = 1 and B = Cin = 0. So we have the EXOR of three inputs in the end.



As for Cout, it is the majority function; it goes to 1 when we have a dominance of 1 amongst the inputs. So if we have at least two 1 in input, Cout = 1.

$$\begin{aligned}
 \text{SUM} &= A \oplus B \oplus C_i \\
 &= A\bar{B}\bar{C}_i + \bar{A}B\bar{C}_i + \bar{A}\bar{B}C_i + ABC_i \\
 \text{Cout} &= AB + BC_i + AC_i
 \end{aligned}$$

Express Sum and Carry as a function of P, G and D

If G = 1, Cout = 1. If P = 1, which means that A ≠ B, Cout = Cin. If D = 1, Cout = 0

P, G and D are mutually exclusive, they cannot all be 1. Once one is 1, the other two must be 0.

$$C_{out} = AB + C_{in}(A+B) = G + C_{in} \cdot P$$

$$Sum = (A \oplus B) \oplus C_{in} = P \oplus C_{in}$$

Define 3 new variables which only depend on A and B

Generate (G) = AB

Propagate (P) = A ⊕ B

Delete (D) = $\overline{A} \overline{B}$

G, P and D: only one function can be 1 at a time.

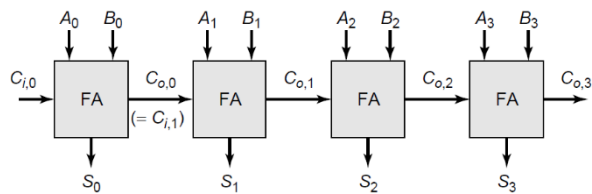
$$C_o(G, P) = G + PC_i$$

$$S(G, P) = P \oplus C_i$$

Once we have a half adder and a full adder, we can implement an adder.

THE RIPPLE-CARRY ADDER

The LSB is on the left, the MSB on the right.



t_c = propagation delay of CARRY-OUT signal
 t_s = propagation delay of SUM signal

Worst case delay linear with the number of bits: $t_{adder} = (N-1)t_c + t_s$

Goal: make the fastest possible carry path circuit

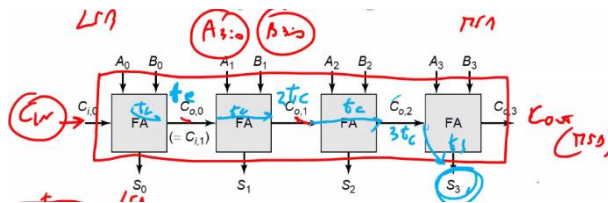
If we consider the circuit as a black box, we notice that $C_{i,0}$ is the LSB, and $C_{o,3}$ is the MSB. The intermediate carries are intermediate signals we need to exploit to compute the sum, but they are not at the output.

Propagation delay

In a full adder we can define two different propagation delays:

- t_c is the propagation delay of the carryout signal.
- t_s is the propagation delay of the sum signal.

The worst case delay is when the carry $C_{o,0}$ has to propagate down to the last block up to C_{out} and S_3 .

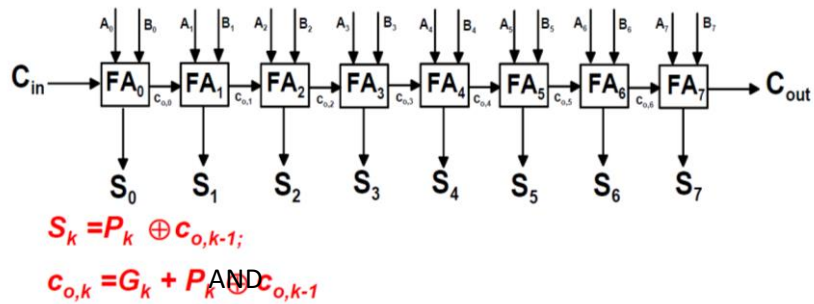


The worst condition is that FA1 and FA2 are both in propagate ($P = 1$), because we need to propagate the carry in the full adders. The operation takes $3 \cdot t_c + t_s$.

If $t_c > t_s$ and N is very big, $t_{conv} = N \cdot t_c$, but this is not a good adder.

It is called ripple carry adder because it refers to the fact that $C_{o,0}$ has to propagate up to the output. It is better to optimize t_c because we have $N \cdot t_c$ in the formula, so the delay can explode if it is big.

Example of 8-bit ripple-carry adder



Which is the "worst case" for the adder propagation delay?

- Worst-case condition is due to $c_{o,0}$ that propagates along the chain down to the last FA
- Last signal to transition can be either S_7 or C_{out} depending on t_s and t_c

If we consider a generic sum bit, with k that spans from 0 to 7, we have the red formulas in the image. In the expression of the sum bit and carry bit there are 3 signals involved, P_i , G_i and $C_{o,i-1}$. We know that $P_i = A_i \text{ EXOR } B_i$, and $G_i = A_i \cdot B_i$, and these are fast. The problem is the carry $C_{o,i-1}$, that comes from the previous block. For instance:

$$S_7 = P_7 \oplus C_{o,6}$$

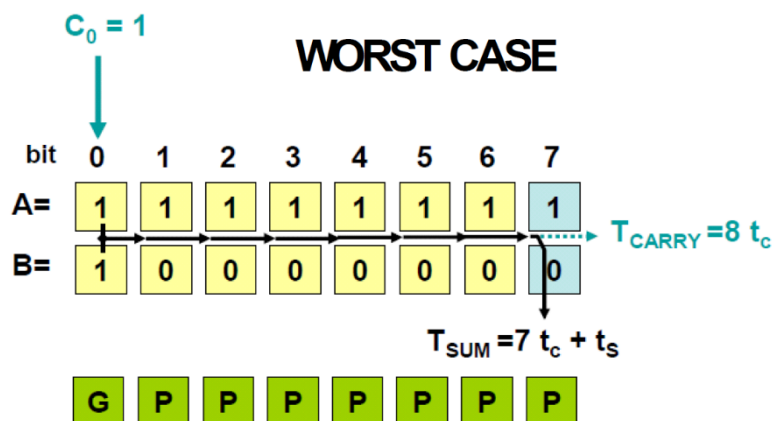
$$C_{out} = G_7 + P_7 \oplus C_{o,6}$$

P_7 and G_7 are very fast, available after a gate delay, the problem is $C_{o,6}$, that in the worst case depend on the propagation of $C_{o,0}$.

$$t_s > t_c \quad S_7 \quad t_{S_7} = 7t_c + t_s$$

$$t_c > t_s \quad C_{out} \quad t_{C_{out}} = 8t_c$$

In the blue case $P_{1:7} = 1$, while in the green case $P_{1:6} = 1$. So the problem is when we have to propagate.



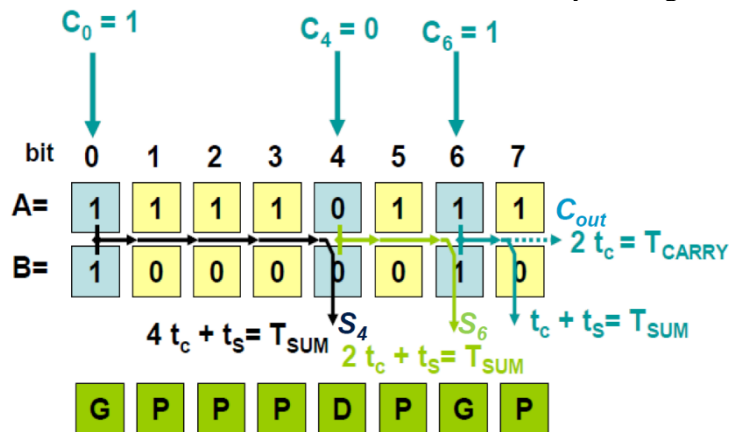
Inside the blocks of the full adders we have the values of the bits. FA from 1 to 7 are in propagate, while the FA0 is in generate carry, and in fact $A_0 = B_0 = 1$.

This means that $C_{0,0}$ goes to 1 after t_c but then it's propagated all the way down to the end of the chain with t_c between each FA, and so C_{out} requires $8 \cdot t_c$. The result of the full adder S_7 has to wait for the C_{in} to generate, which is $7 \cdot t_c$ and then I have to compute the sum, so the delay of S_7 is $7 \cdot t_c + t_s$. This is a typical worst case.

The first FA0 can be in G, D or P, but the worst delay is not changing, whatever the condition of the first FA, in any case $C_{0,0}$.

Non-critical condition

In the following case, it is not as before. In this case it is not true that the last signal to transition is C_{out} or S_7 . For this combination of the inputs, C_{out} and S_7 are very fast, because there is a D condition and a G one in the intermediate FA, we don't have propagate in all the inner FA. C_{out} is very fast, because it's $G_7 + P_7 \cdot C_{0,6}$. As for $C_{0,6} = G_6 + P_6 \cdot C_{0,5}$, and $G_6 = 1$ since $A_6 = B_6 = 1$, so we are generating $C_{0,6}$. There is no reason to wait for the propagation of the carry generated by the first FA. This generation of $C_{0,6}$ operation takes t_c , G_6 goes to 1 after a delay of t_c . Then once we have $C_{0,6}$, it is a matter of propagate it, and C_{out} is available after $2 \cdot t_c$. So C_{out} and S_7 are very fast signal.



The worst case signal must be searched in some full adders that are propagating the carrier. So S_4 can be a good candidate, since $C_{0,3}$ is the propagation, back in the chain, of $C_{0,0}$.

This is an example of non-critical condition, because of the presence of D and G in the middle.

Moreover, S_5 is very fast, it requires $t_c + t_s$ because the result of the FA5 is: $S_5 = P_5 \text{ EXOR } C_{0,4}$, but $C_{0,4}$ is very fast since it is deleted after t_c , goes to 0. **The problem is when the previous full adders are propagating the carriers, in this case we have to wait for the propagation.**

ADDER IMPLEMENTATION

Let's consider a generic FA and the expressions for the C_{out} and S . I want to implement it at transistor level. Before resorting to transistors, let's implement the De Morgan theorem, since the C_{out} and S are difficult to implement because they are not inverting, so it is difficult to implement them in FC-CMOS.

So we apply the De Morgan theorem to express them in an inverting form.

$$C_{OUT} = AB + AC_{IN} + BC_{IN} = AB + C_{IN}(A+B)$$

$$S = A \oplus B \oplus C_{IN}$$

Applying the De Morgan theorem....

$$\overline{C_{OUT}} = \overline{AB + AC_{IN} + BC_{IN}} = \overline{AB} \cdot \overline{(A+B)C_{IN}} = \dots = \overline{AB + C_{IN}} \cdot \overline{(A+B)}$$

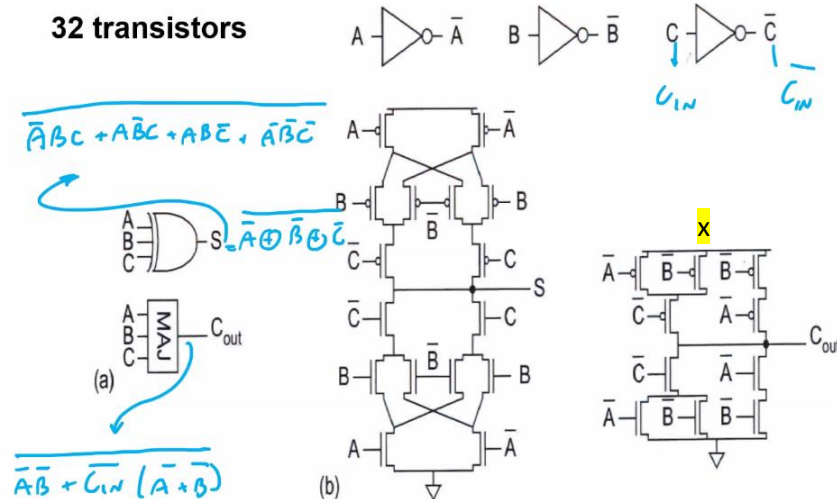
$$\begin{aligned} \overline{S} &= \overline{ABC_{IN} + \overline{ABC_{IN}} + \overline{ABC_{IN}} + ABC_{IN}} = \dots = \overline{ABC_{IN}} + \overline{\overline{ABC_{IN}}} + \overline{ABC_{IN}} + \overline{\overline{ABC_{IN}}} \\ &= \overline{A} \oplus \overline{B} \oplus \overline{C_{IN}} \end{aligned}$$

Inversion Property:

Inverting the input signals, the outputs (both sum and carry) are inverted

Cout_bar is like Cout if I complement the inputs. So **complementing the inputs of a FA provides the inverted function at the output (inversion property)**. It is just a property of the full adder. Now that I have the inverted functions I can implement the FA in FC-CMOS logic.

FULL ADDER IN FC-CMOS LOGIC

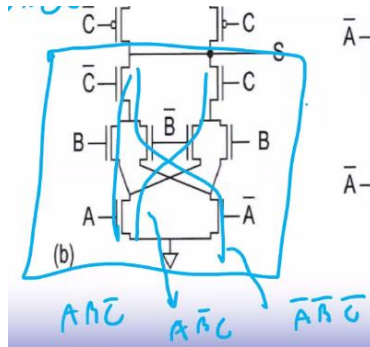


If we look at image x, C (C_{in}) is placed at the top because it is a critical signal and also to reduce the intrinsic capacitance of the gate.

To implement the PUN corresponding to the PDN, we should invert parallel and series, and it should not be the one in the image. However, from a logical standpoint they are the same. Moreover, the one in the slide is better because sizes are smaller.

Starting from the PDN, we can design the PUN in the image, and we notice it is exactly the same, just swapping nMOS and pMOS transistors. We can design similar PUN and PDN because of the inversion property. In fact, if we complement the input we complement the output for the full adder, so we can do this. In image x, it is indeed like having the same networks but with complemented inputs. So we have also simplified the network.

If we look at the circuit in the center, it seems complex but it is a way to condense the usage of transistors.



The critical path, looking at circuit x, is the one from Cin (C) to Cout. We can verify that once the Cout circuit is generating or deleting the carry, there is no reason to wait for the Cin. In fact, we can have a couple of situations. A = B = 1, which is generate, or A = B = 0, which is delete. For the G condition, means A_bar = B_bar = 0, and we don't care about C_bar, since the output is pulled up after tc. In the other D condition, correspondingly the output node is pulled down regardless the Cin value.

This is not the best solution because the propagation delay is too large. We want to improve this solution to arrive at the so-called **mirror adder**.

EXPLOITING Cout TO GENERATE S

S		AB			
		00	01	11	10
C _{in} C _{out}	00	0	1	X	1
	01	X	X	0	X
	11	X	0	1	0
	10	1	X	X	X

S		AB			
		00	01	11	10
C _{in} C _{out}	00	0	1	X	1
	01	X	X	0	X
	11	X	0	1	0
	10	1	X	X	X

MIRROR ADDER

A=B=1
C_{in}=0
C_{out}=0

$$S = ABC_{in} + \overline{C_{out}} \cdot (A + B + C_{in})$$

$$C_{out} = AB + C_{in} \cdot (A + B)$$

The sum is not a function of Cout in reality, but let's express a k-map where S is expressed as a function of Cout. There are a lot of 'don't care' slots because the function is a 3 bit function (8 values). For instance, we can have the following don't care condition, that is never reached, because when A = B = 1, Cout = 1, so Cout cannot be 0.

Let's synthesize some groups of '1', considering also the 'don't care' as 1.

S		AB			
		00	01	11	10
C _{in} C _{out}	00	0	1	X	1
	01	X	X	0	X
	11	X	0	1	0
	10	1	X	X	X

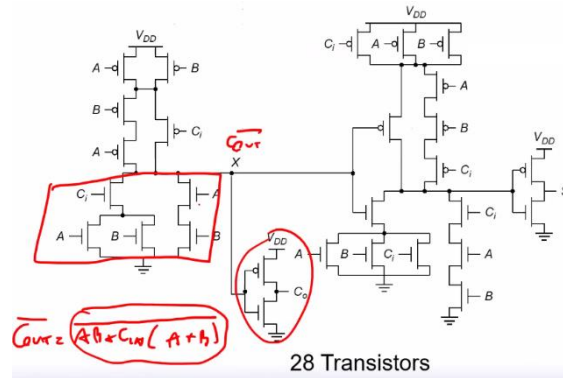
BC_{out}

A_{out}

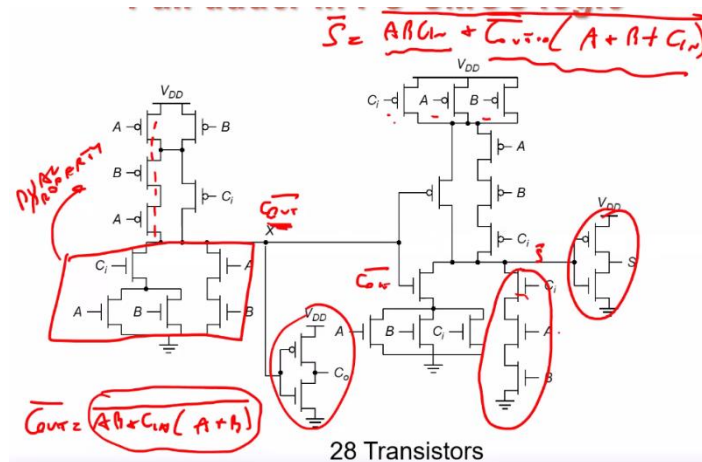
C_{in} C_{out}

ABC_{in}

We get the expressions x, to which I apply the De Morgan theorem.



Then in order to get Cout I take Cout_bar and invert it with an inverter. In the image the PUN is implemented with the dual property, not with the inversion property. Also S_bar can be implemented in FC-CMOS, and then we get S with an inverter.



Also for S_bar we implement the PDN with the FC-CMOS theory and then the PUN with the dual property.

We have a huge amount of transistors, 28, which is still an improvement with respect to the initial 32.

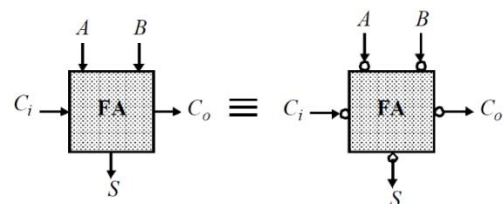
As for the carry, if we look at the propagation, we have Cin that is Ci and it has to propagate to x (Cout_bar) and then an inverter. So two stages and the first one is complex. So with respect to the FC-CMOS gate the situation is not so improved.

To further improve, we can exploit the inversion property, at transistor level, while the second improvement is at behavioural level (architecture). In both improvements I want to exploit the inversion property.

Architectural level – inversion property

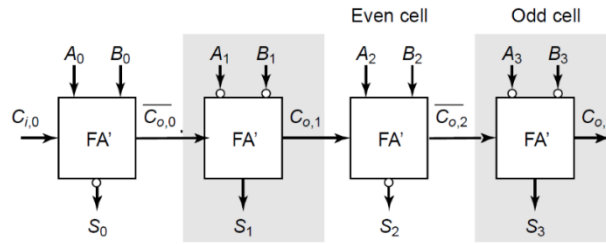
We complement the inputs to get the complemented outputs.

Let's suppose now that we want to implement a 4 bit adder. The blocks in the image below are inverted full adders (FA'), that are the previously seen stages without the inverter at the output, so we have S_bar and Cout_bar.



$$\bar{S}(A, B, C_i) = S(\bar{A}, \bar{B}, \bar{C}_i)$$

$$\bar{C}_o(A, B, C_i) = C_o(\bar{A}, \bar{B}, \bar{C}_i)$$

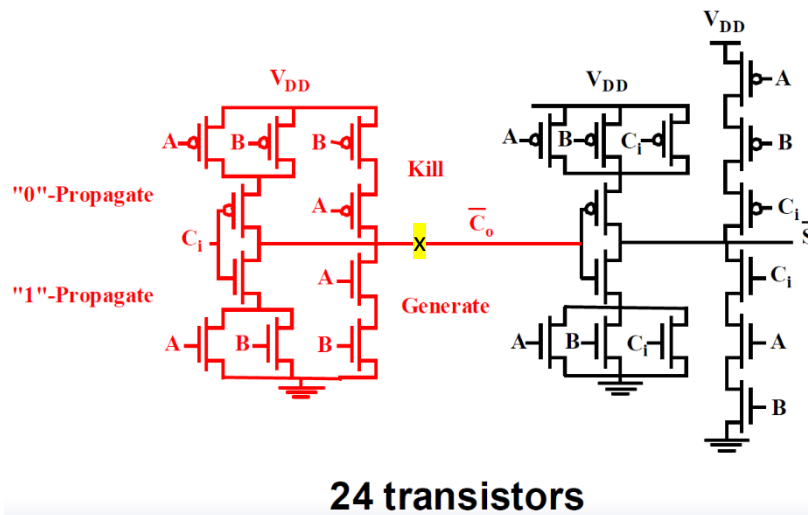


FA' is the full-adder in FC-CMOS logic without final inverters

If we look at the first FA', at the output I have $C_{o,0_bar}$, and to get S at the output I add an inverter, which can be done because it is not a critical path. Basically I removed the inverters on the critical path of the carry. Now the second FA' receives at the input $A1_bar$ and $B1_bar$ and $C_{o,0_bar}$. Since it is inverting, at the output I have $C_{o,1}$.

In terms of critical path it is an improvement because each full adder has a single stage and not the inverter, but there is also an improvement in terms of transistors (overall 108).

THE MIRROR ADDER

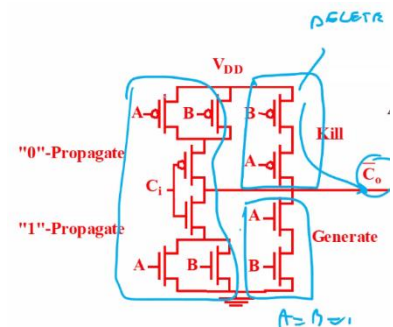


The one in the image is the inverting full adder. The PUN is created mirroring the PDN. Now in the critical path I have 2 transistors in series in the carry path, and only 3 in the S path (previously it was 3 and 4).

We can recognize 3 parts, that correspond to D, G and P. In terms of carry-in if the B pMOS is on and A nMOS is on we have an inversion for C_i . So in the upper part I have the propagate corresponding to 0 and in the lower part the propagate of 1 in terms of carry-in.

The red on is the critical part because in an adder what counts is the propagation of the carrier.

The sum S (black) is not a critical part.

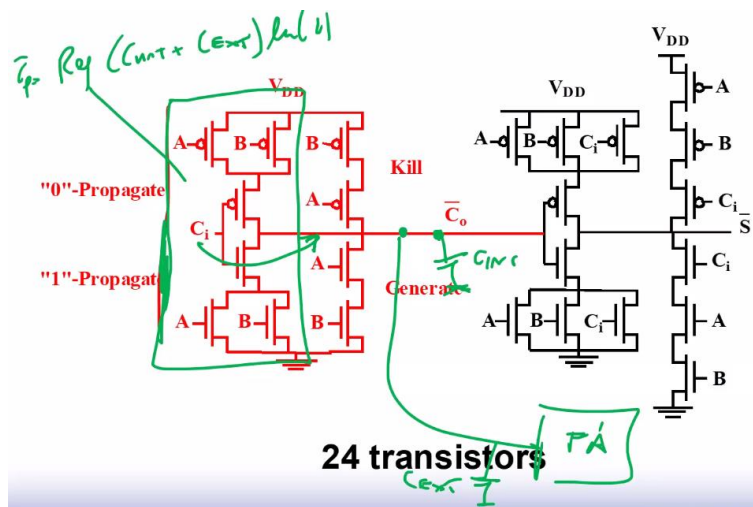


In which part of the circuit can I use minimum size transistors to spare capacitance?

The idea is to use a $(W/L)_n = 1$ and $(W/L)_p = 2$. The objective is to reduce the intrinsic capacitance at C_{out_bar} , and what comes into play is the sum circuit, so we can use minimum size circuits in the sum part. What about the parts related to kill and generate, in terms of resistance? Do I have to care about their sizing? Or I don't care about the kill and generate delay? Since the critical condition is the propagation, I don't care. In fact, the critical path is from propagate, hence the transistors of kill and generate can be very small.

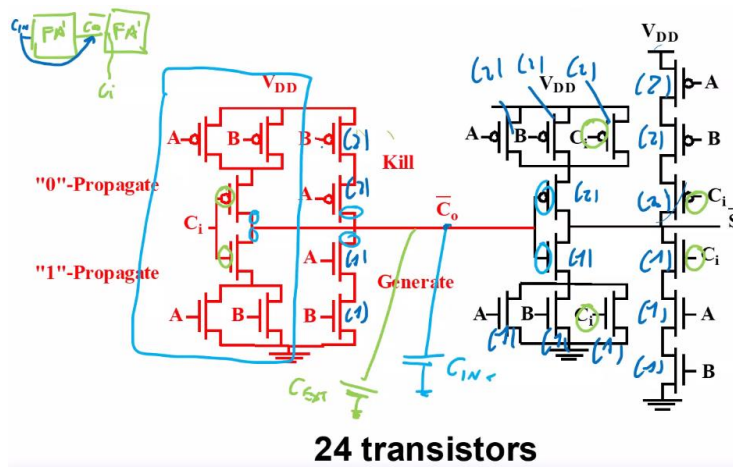
Instead, in the propagation region I cannot choose minimum size transistors, otherwise I struggle a lot, because that node C_{out_bar} is connected to the C_{in} of another full adder.

So in node x I have an intrinsic contribution and an extrinsic contribution, in terms of capacitance, due to the next FA'.



Hence the green highlighted part is the most critical part of the circuit. Let's assess the capacitance C_{o_bar} , which is the one that must be somehow minimized, since the critical delay is from the input C_i to C_{o_bar} . It is the delay to be minimized.

Let's step back to the transistors' sizing. In the non-critical path I can size them with size (1) (minimum size). So (1) for the nMOS and (2) for the pMOS, even if the propagation delays of PUN and PDN are not equalized, but it is needed to minimize the intrinsic capacitance. We don't use size (1) for the pMOS because it would be a too poor switch, the propagation delay of the PUN would explode.



Also in the carry circuit we have some parts that are non-critical and can have a size of (1) and (2).

As for the part in the light blue box, it must be accurately sized. So size $2s$ for the nMOS and $4s$ for the pMOS of this part. Hence it is a circuit of size s , so it has an equivalent resistance given by $R_{eq}^{(1)}/s = R_{eq}^{(s)}$.

Now we have to assess C_{int} and C_{ext} . The capacitance in a node is proportional to the sum of the widths of the transistors connected to that node. Since all the transistors have minimum length, we can say that the capacitance is proportional to the aspect ratios.

Cint assessment

It is like the intrinsic capacitance of the first stage. Then I have another contribution C_{ext} that I'm trying to decouple. C_{int} is proportional to the sum of the aspect ratios; we have 3 nMOS transistors attached to the node, and also 3 pMOS (1 nMOS and 1 pMOS also from the black stage).

So C_{int} is proportional to $(2s + 1 + 1 + 4s + 2 + 2)$, considering nMOS and pMOS.

Cext assessment

We have to think about the next inverting full adder, so we have to consider the green circles of the previous image. Overall, the sum of aspect ratios is $(2s + 4s + 1 + 1 + 2 + 2)$.

Propagation delay assessment

It is the propagation delay of the propagate part of the carry circuit. It is proportional to the equivalent resistance of the driving circuit times the sum of the aspect ratios.

$$\tau_p \approx R_{eq}^{(s)} (C_{int} + C_{ext}) = \frac{R_{eq}^{(s)}}{V_{DD}} [12s + 12] C_1$$

This is the propagation delay of the carry. We want to find the size to minimize the delay from the input to the output, and the size is infinite theoretically, and the delay cannot be smaller than $R_{eq}^{(1)} * 12$.

However, this is not the best choice to have an infinite size, because of area and power consumption. A reasonable choice is to have the second term almost negligible but not actually 0.

So it is like if the sum $12*s + 12$ must be dominated by the first term, so having a $s = 3$ or 4 .

A reasonable choice for the propagate circuit of the mirror adder is $s = 3$ or 4 .

We are not using the Sutherland theory of the minimization of gates because there is no load to drive at the end of the chain.

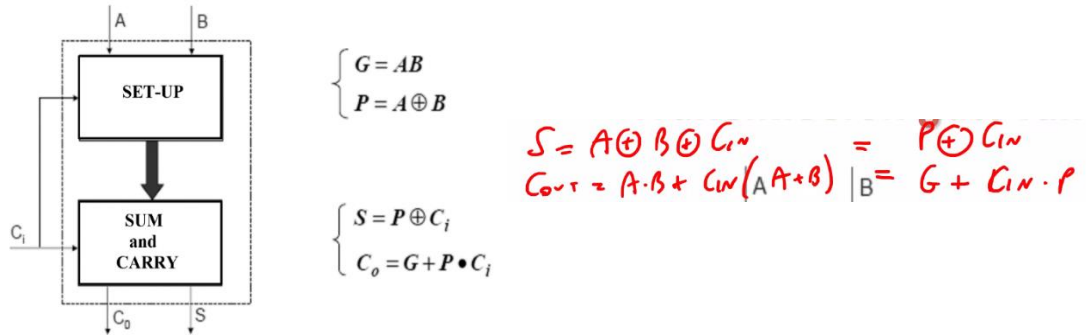
Some considerations

- The NMOS and PMOS chains are **completely symmetrical**. A maximum of two series transistors can be observed in the carry-generation circuitry.
- The most critical issue is the minimization of the capacitance at node \overline{C}_o . The reduction of the diffusion capacitances is particularly important.
- The capacitance at node \overline{C}_o is composed of four diffusion capacitances, two internal gate capacitances, and six gate capacitances in the connecting adder cell.
- The transistors connected to C_{in} are placed closest to the output node.
- Only the transistors in the carry stage (propagation) have to be optimized for optimal speed. All the transistors in the sum stage can be minimal. The carry stage can be sized with a size of 3-4. This size is needed to optimally drive the large capacitance corresponding to \overline{C}_o .

Having a maximum of 2 transistors to Vdd or GND is a very good improvement with respect to the FC-CMOS gate. Moreover, to have the minimum delay from C_{in} to C_{out} we have to minimize the capacitance, as we have done, selecting also a minimum size for the non-critical transistors.

Then, transistors connected to C_{in} are placed close to the output node. This is something that is always a good solution, because transistor connected to the slowest signals must be connected closer to the output node.

TRANSMISSION GATE FULL-ADDER



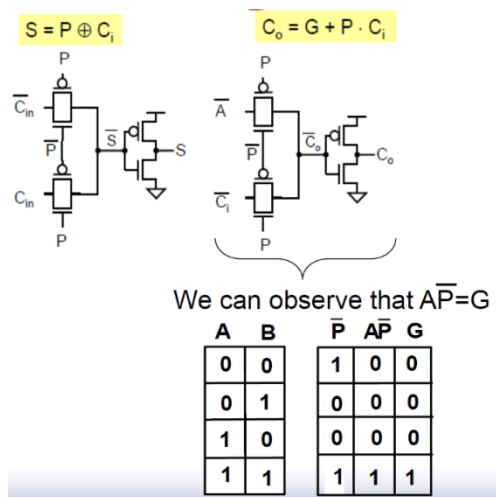
Here we have EXOR and AND functions. So why not implementing the single bit FA with transmission gates? Maybe we can spare even more transistors.

Let's split the single bit FA in a setup part that is responsible for the P and G production at the output, and the sum and carry part for the production of C_o and S. To be honest, the G signal is not needed, it can be avoided and only the signal P is needed.

Propagate is A EXOR B, which means $A \cdot \bar{B} + \bar{A} \cdot B$.

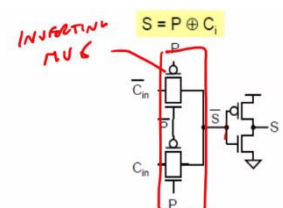
The critical part is from C_i to C_o . The P generation is not the critical part.

Sum and Carry block



We don't need G because if e.g. we consider the truth tables in the image, we notice that $G = 1$ iff $A = B = 1$. So G can be substituted by $A \cdot \bar{P}$. $A = B$ corresponds to having the kill case if $A = 0$, the generate case if $A = 1$, so if $A = B = 1$ we have the generate.

Now in the setup block we can focus only on P production, not on G → simpler block.

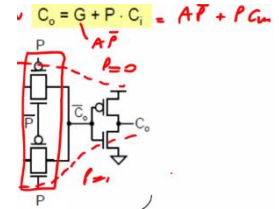


Let's now suppose to have P and P_bar available. If I have P I can implement S with an inverting mux followed by an inverter. We use an inverting mux and an inverter because the inverter is a good solution after a TG to avoid a cascade of a lot of transistors.

NB: $P \text{ EXOR } C_{in} = P \cdot C_{in_bar} + P_bar \cdot C_{in}$.

For $P = 1$ the lower path is on, for $P = 0$ the upper part is. If $P = 0$, $S = C_{in}$, while if $P = 1$, $S = C_{in_bar}$.

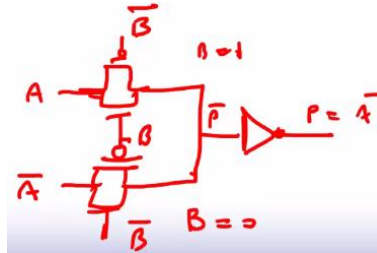
Let's now focus on the C_o circuit. G can be substituted by $A \cdot P_bar$. So we can implement an inverting multiplexer followed by an inverter. For $P = 0$ the bottom path is on.



As we can see there is no reason to generate $A \cdot P_bar$, with this trick to express C_o it is already generated by the multiplexer implementation. So we can spare a lot of hardware also.

Now we need to implement the circuit that outputs P and P_bar. Since $P = A \text{ EXOR } B = A \cdot B_bar + A_bar \cdot B$, I can output P using an inverting multiplexer.

This is correct from a logical standpoint but it is not the correct solution.



The problem is that P and P_bar in this implementation have different delays, so there is a period of time when P and P_bar are on, since both paths of the multiplexers are on at the same time.

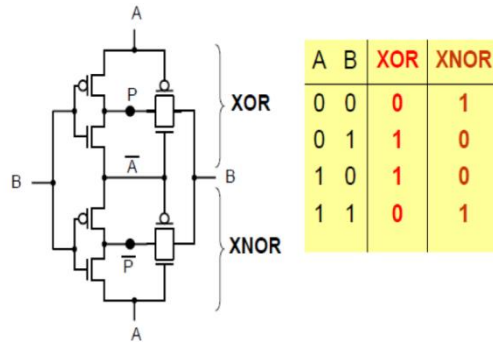
This leads to a couple of problems:

- If both paths are on we have a direct connection between Vdd and GND, so cross conduction current and power consumption.
- At the output we can have glitches, which correspond to a logical mistake.

The real solution for the transmission gate full adder is the following.

Correct TG-FA

The advantage of this solution is that it avoids glitches, because the circuit is more symmetric. The TG full adder exploits this solution (one of the two parts, the upper XOR or lower XNOR).

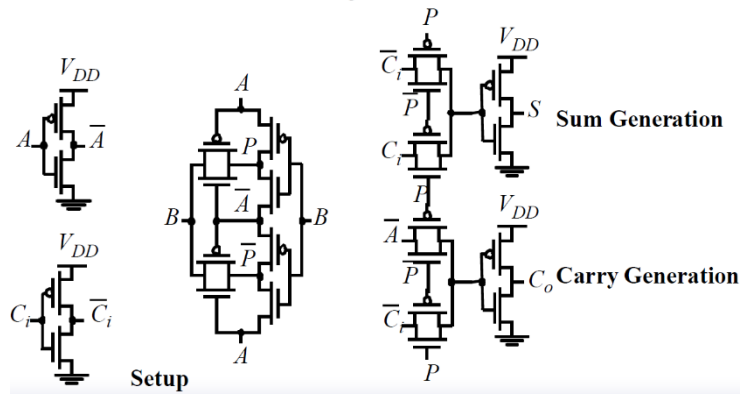


With this solution P and \bar{P} are generated with the same delay, avoiding glitches in the adder

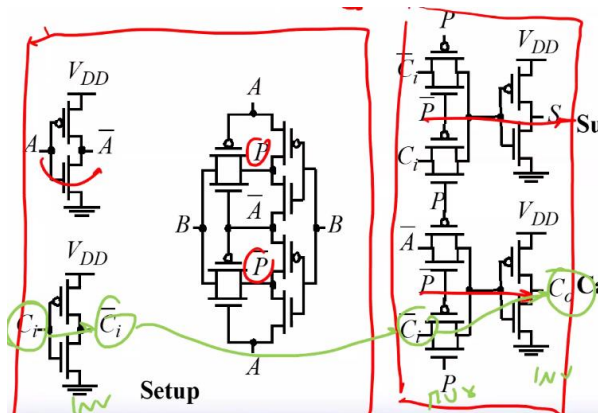
The advantage of this solution is to avoid glitches in the adder.

So we have this building block above, and in the setup block we need also two inverters for A_bar and Ci_bar generation. In this configuration S and Co have the same delay.

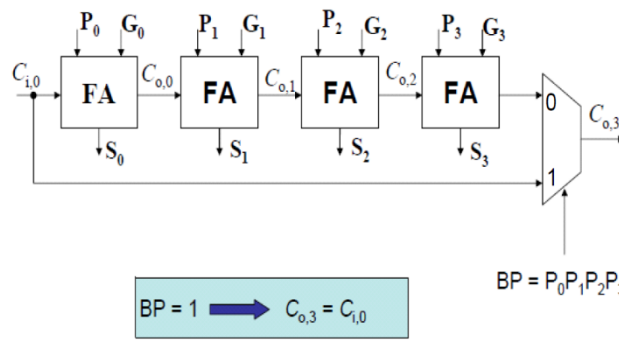
- 24 transistors
- Same delay for S and Co (interesting in multipliers)



Let's now shift to a n-bit adder, cascading the circuit n times. The critical path is always the propagation of the carrier. From Ci to Co we have three stages in cascade, inverter, inverting multiplexer and inverter. They seem a lot for a simple one bit FA, but they are very simple stages.



CARRY BYPASS (CARRY-SKIP) ADDER



The idea is to “skip” the cells if the carry has to propagate

The most important figure of merit was previously the propagation delay, and the worst path is from C_{in} to C_{out} , and the worst case condition is the one that corresponds to $C_{i,0}$ propagated down to $C_{o,2}$ (or $C_{o,3}$), so when the carry generated, deleted or propagated by the first FA has to be propagated up to the last point and then is needed to compute the sum.

If we cascade N single bit full adder, the problem is that the propagation delay in the worst case is proportional to N .

At architectural level, we want to implement an efficient adder whose delay doesn't increase linearly with N .

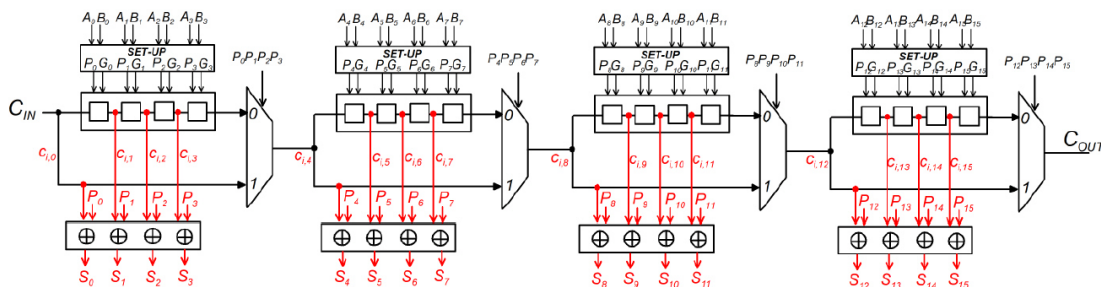
Let's suppose we need to implement a very large N adder. We put in cascade FAs and the critical path is the same as before, if the mux was not present and all the circuits were in propagate. This leads to the largest delay.

Then we add a **2-bits multiplexer**. Then we use a function called **bypass propagate (BP)** that, if 1, delivers $C_{i,0}$ to the output. So we skip the FAs, which are complex circuits and slow, and we bring $C_{i,0}$ to the output directly. Of course we skip only when needed, that is when all the FAs are in propagate, since $C_{i,0}$ should be propagated from the input to the output.

In this circuit, if we consider it as a standalone circuit, the critical condition is when $P_0 = 0$, so the mux is on in the '0' path, but all the other FAs are in propagate, so $C_{o,3}$ has to wait for $C_{o,0}$, which is a carry that is generated or deleted and then has to be propagated.

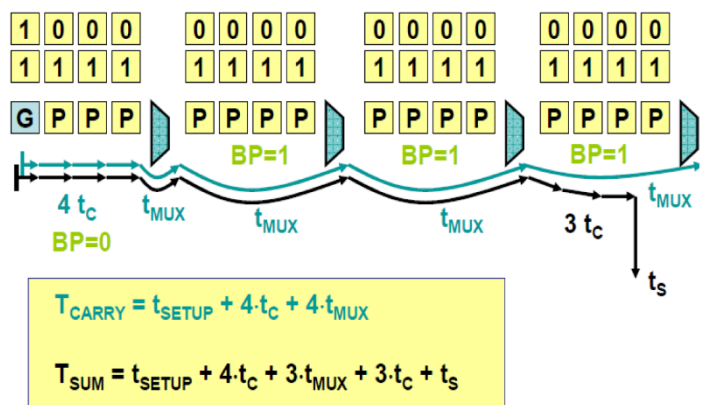
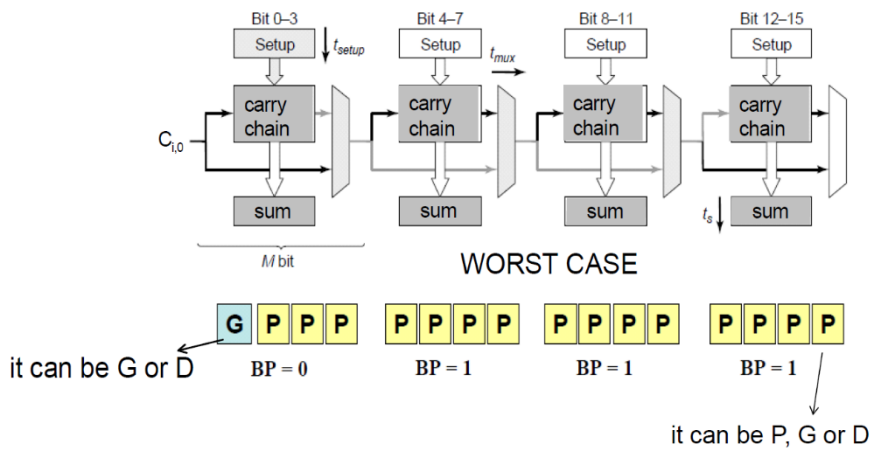
Example of 16-bit carry-skip adder

We are placing in cascade 4 circuits as the one above.



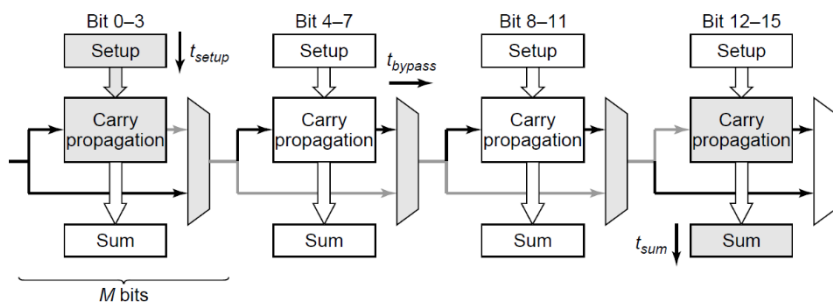
On the top of each block we have a setup block responsible for creating the P and G signals. Then we have the carry circuits in cascade (white squares), so each block computes $C_{o,k} = G_k + P_k * C_{o,k-1}$, where $C_{o,k-1} = C_{in,k}$.

Then we have EXOR gate to produce the sum, since $S = P_k \text{ EXOR } C_{i,k}$.



Let's try to formalize this improvement.

Worst case delay



N: overall number of bits
M: number of bits per block
N/M: number of multiplexers

$$t_{adder} = t_{setup} + M t_{carry} + (N/M - 1) t_{bypass} + (M - 1) t_{carry} + t_{sum}$$

We have N overall bits with M bits per block, so overall N/M multiplexers.

The worst case condition is with **P0 = 0** and **P1:N-2 = 1**, that is when $C_{i,1}$ is propagated up to $C_{i,N}$ and then $C_{i,N-1}$ is used to generate S_{N-1} .

The overall delay in the worst case condition is written in the slide. We have to wait $M \cdot T_{carry}$ ($T_{carry} = T_c$), then we have a term with $N/M - 1$ because all the multiplexers except for the last one. Then in the last carry chain I have to pass in $M - 1$ circuits.

Which is the optimum M value?

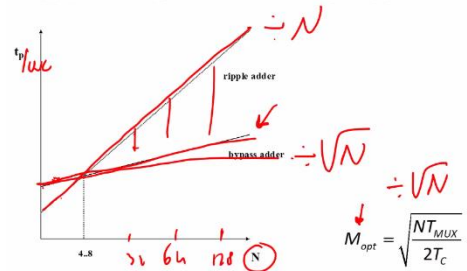
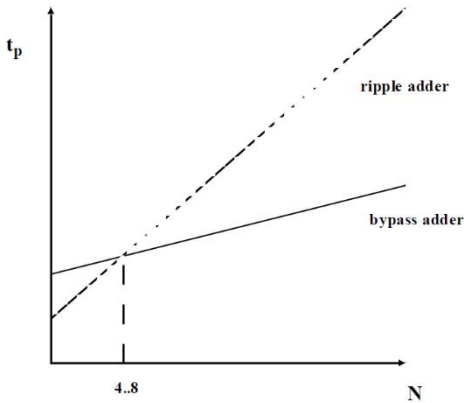
I have to take the derivative.

$$0 = T_c - \frac{N}{2M^2} T_{mux}$$

$$t_{sum} \quad M_{opt} = \sqrt{\frac{NT_{mux}}{2T_c}}$$

For $T_{mux} = 0$, M_{opt} tends to 0. Moreover, what counts is the dependence on N.

RIPPLE-CARRY VS CARRY-SKIP

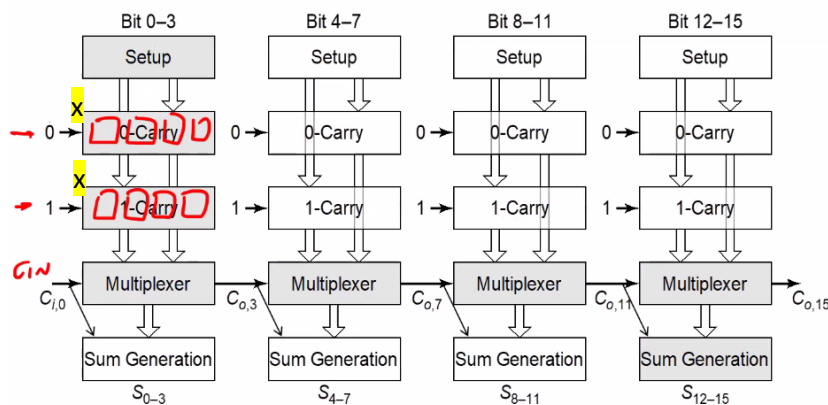


$$M_{opt} = \sqrt{\frac{NT_{MUX}}{2T_c}}$$

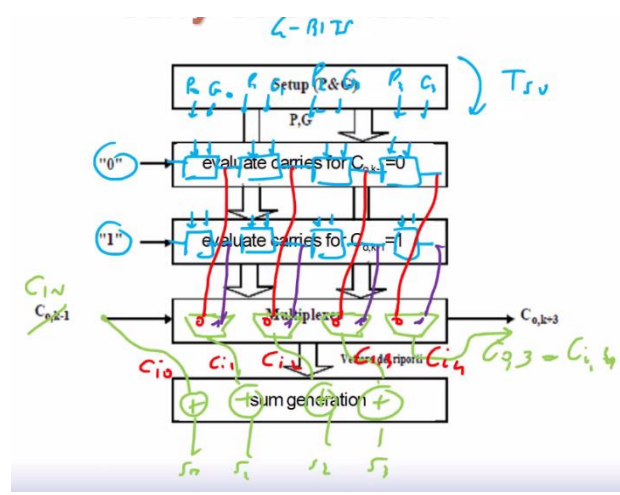
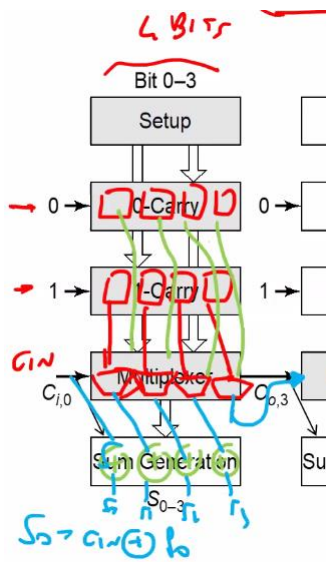
There is a region where the bypass adder is worse than the ripple carry adder, for small value of N, and the reason is the overhead, that is increased by the delay of the multiplexer, and if we have a small amount of bits there is no reason in increasing the delay because of the increase in the overhead.

CARRY-SELECT ADDER

It exploits **parallelism**. In the image we have a 16 bit adder split in two parts. The idea is to compute in advance the carry inside the circuit hypothesizing a 0 or a 1 at the input. So I have replicated the carry circuit and supposed a 0 at the input or a 1. I'm computing carries in both x circuits, but the real carry can be coming from the upper or lower block depending on the $C_{i,0}$.

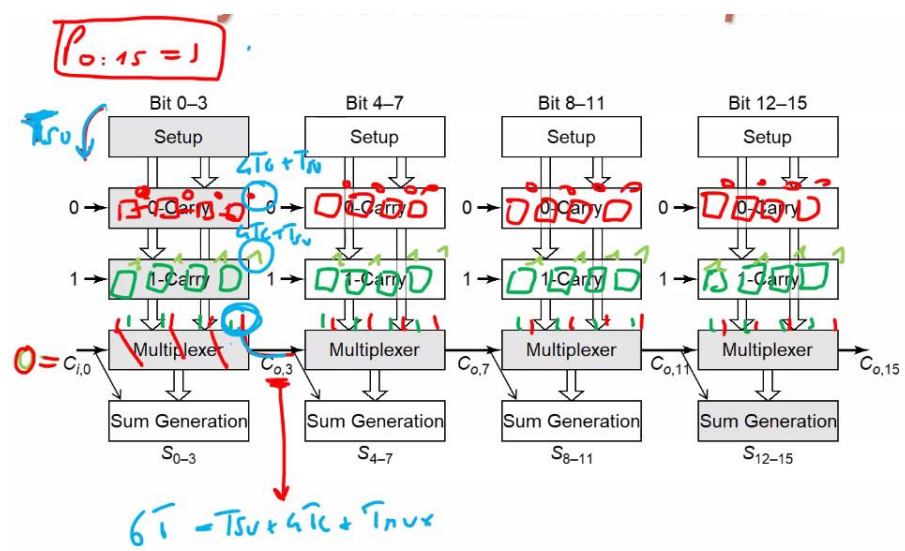


Then we have multiplexers whose inputs are chosen depending on the C_{in} value from the 1-carry or 0-carry. The first EXOR gate then receives directly C_{in} , and the last carry is used as select signal for the next multiplexer. This leads to a great advantage in terms of time, but we are worsening the area. However, this is the only way to improve the delay, computing in advance the carries.



Critical path

We have 4 bits for each block. Let's suppose we have $P = 1$ from 0 to 15, so we are propagating all the carries from the input to the output. So each carry circuit is propagating the carry. The red carries are 0 and the green ones are 1.



If we suppose $C_{in} = 0$, we select the red ones that are passed through the multiplexer. We have to wait for the transit of the carriers, regardless being a 0 or 1, and then we have to pass through the mux. The last signal to arrive at the mux is the select signal, not the real signal, because it takes $6T$. The last signals to arrive at the mux are the blue circled ones, they take $T_{su} + 4 \cdot T_c$. So it seems that the select signal arrives with a certain delay with respect to the real signal. Then also for bits 4-7 the delay is the same for the bits 0-3. Also $C_{0,7}$ has a delay that is $T_{su} + 4 \cdot T_c + 2 \cdot T_{mux} = 7T$, because I have to wait for $C_{0,3}$ and then add another T_{mux} delay. $C_{0,11}$ has a delay of $8T$ ($3 \cdot T_{mux}$ inside).

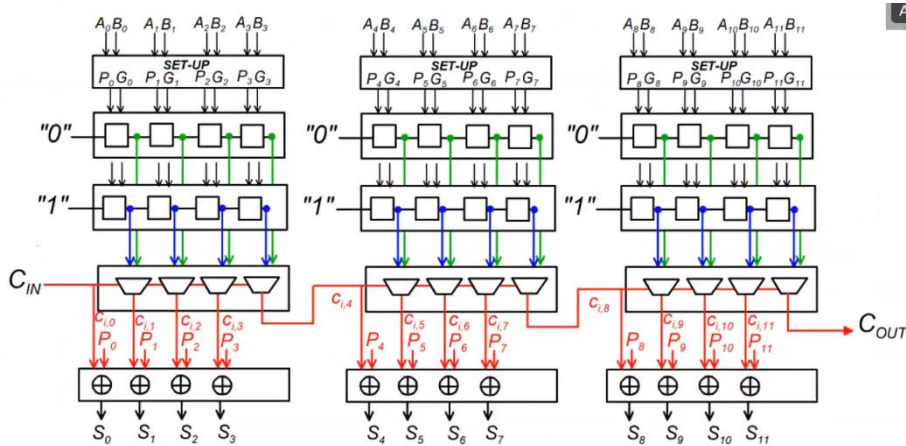
Is $C_{0,15}$ the last signal to transition or one of the sum bits?
 C_{out} has a delay of $9T$ to transition, because I have to wait for $C_{0,11}$, which is the select signal of the mux, because all the inputs are already available.

$S_{12} = C_{i,11} \text{ EXOR } P_{12}$, so the delay of S_{12} is the delay of $C_{o,11}$ plus the delay of a sum, so $9T$. similarly, $S_{13} = C_{i,13} \text{ EXOR } P_{13}$, which is $9T$ still, as well as for S_{14} . This because the carry in the multiplexer arrive at the same time, and their delay is $8T$, that is the one of $C_{o,11}$, plus T_{mux} .

$$10T = T_{mux} + 4T_c + 4T_{mux} + T_s$$

$$\left. \begin{aligned} P_{13} &= C_{i,11} \oplus P_{13} \\ S_{14} &= C_{i,13} \oplus P_{14} \\ S_{15} &= C_{i,15} \oplus P_{15} \end{aligned} \right\} 9T$$

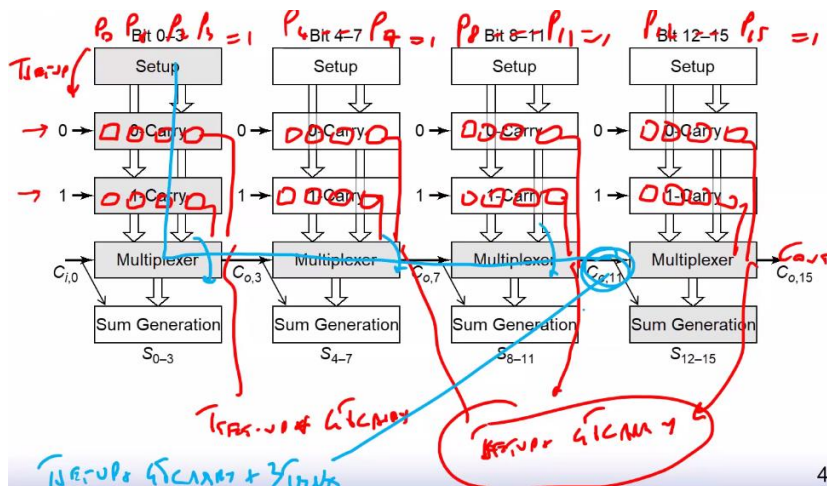
Overall we have $10T$, because I have to compute the final sum. This is an improved solution with respect to the previous adder ($12T$).



$N = 16$ and $P_i = 1$

If we have all the blocks in $P = 1$, after a T_{setup} we start propagate the carries. After a $T_{setup} + 4T_c$, the last carry is available at the multiplexer input. This for each block of 4 bits.

The worst case path, since the inputs of the multiplexer are available after $T_{setup} + 4 * T_c$ (small time), corresponds to the propagation of the last carry of the first block (blue path). Last bits to transition are S_{13} , S_{14} and S_{15} .

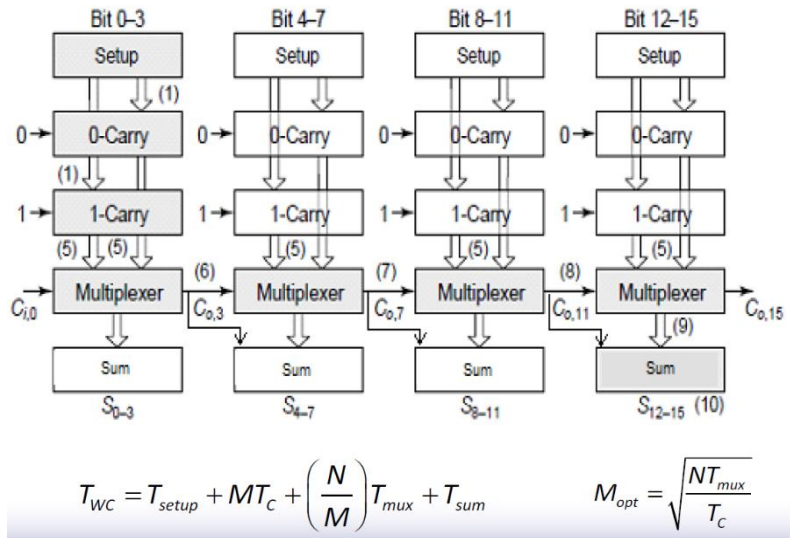


Now we want to assess the worst case delay in the generic case.

WORST CASE DELAY

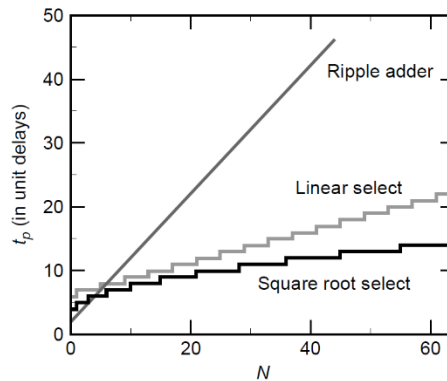
Let's consider $T_{setup} = T_c = T_{mux} = T_{su} = T_{sum}$, N bits and M bits per block, so N/M multiplexers. The worst case path is the grey one. We have $P = 1$ in all the bits. T_{setup} is needed for the generation of the carry, then T_c for the arrival at the multiplexer, and then we have to pass through the N/M multiplexers and in the end through the sum generation circuit (T_{sum}).

The expression in the image has a derivative in some point, since one term increases and another decreases with M . Also in this case, the optimal delay corresponds to have a M proportional to \sqrt{N} , like in the carry-skip adder, the expression is very similar apart from a factor 2.



This is still not the best topology for the adder.

Comparison between adder delays



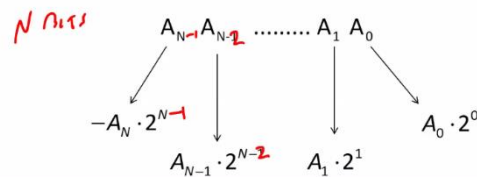
If we don't optimize the linear select adder, the increase with respect to N in the delay is linear. But if we optimize it, the linear dependence becomes a square root one. In any case, for large N there is an advantage in using the linear adder. The cost is that the area we have to use is much bigger.

SIGNED NUMBERS

So far we consider unsigned numbers, so only positive numbers, and now we want to code also negative numbers to implement a subtractor. We want to implement negative numbers so that we can use adders to implement a subtractor. This way is the **2's complement form**.

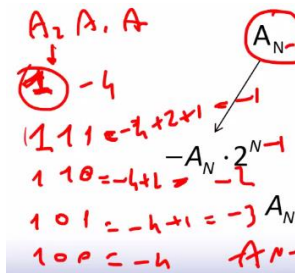
2'S COMPLEMENT REPRESENTATION

- We need a representation of negative numbers to use adders also for subtraction;
- 2's complement representation adopts a fixed number of bits: MSB as bit sign. If MSB=1 the number is negative, if MSB=0 the number is positive;
- MSB has a negative binary weight.



It adopts a fixed number of bits, which is decided a priori. If we add two words of N bits, we get a word of N+1 bits. The idea is to use the MSB as a sign. $A_{N-1} = 1$ means that the number is negative, if it is 0, the number is positive. Moreover, the MSB has also a negative binary weight. It is not just a sign bit, it has a negative weight.

For instance, if we consider a 3 bit number with the third bit that is the sign. We have the following.



So we can code from +3 (when $A_2 = 0$) to -4, because the MSB is not just a sign bit, but it has a negative binary weight. All the remaining bits have a positive weight.

The most important feature of this representation, is that if we consider a positive number nothing is changing from what seen so far from unsigned numbers, but we have also negative numbers.

In order to represent a number, we have to start from the absolute value of the number and, once the number of bits is selected, we represent the bit code for that specific number, e.g. if +2 it's 010 in a 3 bits representation. To move to -2, we have to complement all the bits, also considering the sign bit $\rightarrow 101$. As a second thing we add a +1, so the final result is 110.

In this way we can use an adder to implement a subtractor.

In order to implement a negative number with a fixed number of bits follow the next steps:

1. Implement the positive number (MSB=0 is the sign bit)
2. Complement the number bit by bit (also the sign bit)
3. Add 1

Example with 4-bit words.
If we want to implement -3:

1. 0011 (equal to +3, the first bit, MSB, is the sign bit)
2. 1100 (all bits are complemented)
3. 1101 (added 1)

Let's check: $-2^3 + 2^2 + 2^0 = -8 + 4 + 1 = -3$

Let's consider now $N = 4$ and let's suppose we want to assess $+3 - 4 = -1$. So we should do $+3 + (-4)$. It is $0011 + (0100 (+4)) \rightarrow 1011 \rightarrow 1100 (-4) = 1111$.

In 2's complement, once we fix the number N of bits, we can represent the numbers from $2^{(N-1)} - 1$ to $-2^{(N-1)}$.

- In 2's complement, adopting words of N bits, we can implement numbers ranging from $2^{N-1} - 1$ to -2^{N-1}
- Example of 4-bit words:

0111	→ +7	1111	→ -1
0110	→ +6	1110	→ -2
0101	→ +5	1101	→ -3
0100	→ +4	1100	→ -4
0011	→ +3	1011	→ -5
0010	→ +2	1010	→ -6
0001	→ +1	1001	→ -7
0000	→ +0	1000	→ -8

- To implement $A-B$ we can add A to the 2's complement of B
- Having fixed the number of bits, this representation can lead to **OVERFLOW**, since the carry-out does not matter

Since we fixed the number of bits and the result might be $N+1$ bits, we might have an overflow. E.g. let's try to do $7 + 1 = 0111 + 0001 = 1000 = -8$, but it should be 8. It is like if in the table adding a 7 leads to an overflow, but this is a mistake.

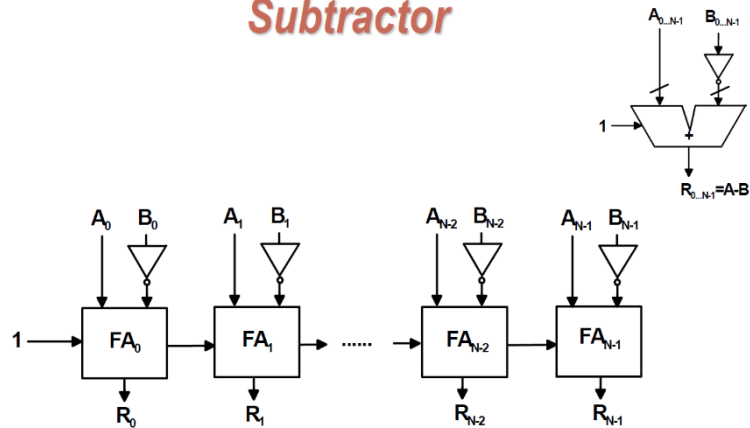
Let's consider another example, for instance $-6 - 7 = 1010 + 1001 = 0011 = 3$.

SUBTRACTOR

We are using an N bit adder with B (the negative number) that is complemented bit a bit. Then we have to add a 1. The result is made just of N bits, not $N+1$, **the last carryout is discarded**, and this can lead to an overflow problem.

In the upper left of the image we have the way to represent a subtractor in digital books, adding a 1.

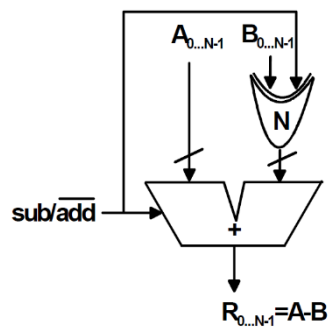
Subtractor



The subtractor is an adder where one of the two input words is in 2's complement format

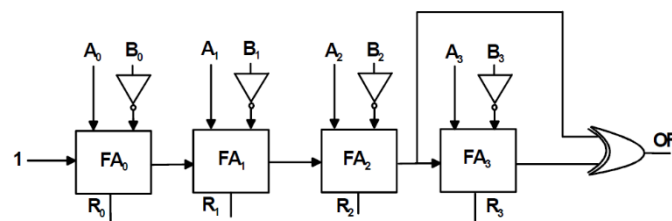
48

If instead of inverters we use XOR gates, we can implement the following powerful circuit. In fact, a XOR with B and zero in input gives B in output, so a buffer behaviour, while with B and 1 in input gives B_bar in output, so we have an inverter behaviour. So depending on the value of sub/add_bar input we have either a buffer or an inverter attached to B.



Sub=1 -> XOR gate works as an inverter and B is complemented
Sub=0 -> XOR gate works as a buffer

Detecting the overflow

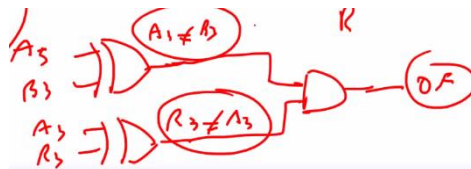


We simply need to add an EXOR gate which has in input the Cin and Cout of the last FA. If its output is 1 we have an overflow, otherwise we don't have it.

In a subtractor, the cases that leads to an overflow are:

- We perform a subtraction and $A_3 \neq B_3$ in a 4 bit operation with different signs.
- We perform the subtraction and $R_3 \neq A_3$, where R_3 is the result of $A - B$.

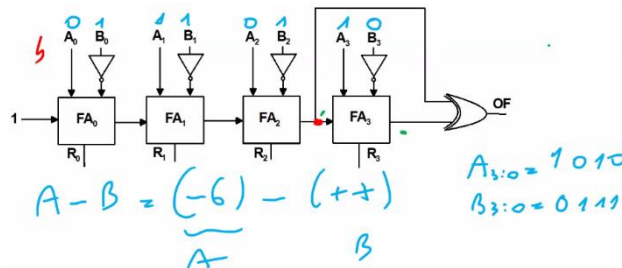
These two conditions must be fulfilled at the same time.



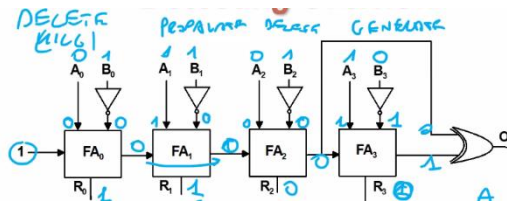
It is a very complex hardware, we can do the same with a simple EXOR gate between the carries.
 Firstly, $A_3 \neq B_3$ means that at the inputs of FA3 we have to inputs that are equal. Let Cin and Cout be the carry-in and carry-out of the last FA.
 By definition of $Cout = A_3 * B_3_bar + Cin * (A_3 + B_3_bar)$. If $A_3 \neq B_3$, means that A_3 and B_3_bar have the same sign $\rightarrow Cout = A_3 + Cin * (A_3) = A_3 * (1 + Cin) = A_3$, since $1 + Cin = 1$ in Boolean algebra. So $Cout = A_3$ in the end.

R_3 is then the sum bit of the sign FA, so $R_3 = A_3 \text{ EXOR } B_3_bar \text{ EXOR } Cin = Cin$ because A_3 and B_3_bar are equal, so their EXOR is 0. So in the end we can EXOR A_3 and Cin.

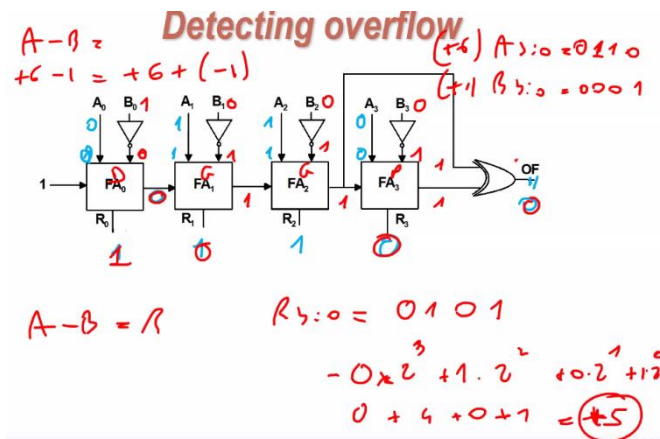
Let's implement $-6 - 7 = -6 + (-7)$, that in the subtractor is $-6 - (+7)$.



Let's verify that is correct at HW level.
 The first FA is in delete, the second one is in propagate, the third is in delete again and the last one is in generate, so the $OF = 1$, and it's correct.



Let's try with $6-1$, which should not overflow.



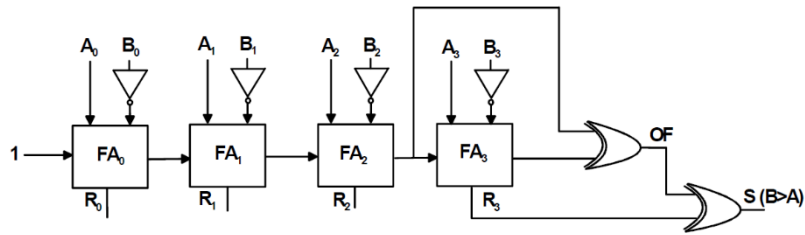
MAGNITUDE COMPARATOR FOR SIGNED NUMBERS

It is a subtractor able to assess if a number is greater or smaller than another number. S is an output that goes to 1 if $B > A$, and B and A are signed numbers.

The result S should go to 1 if $B > A$, so in the case there is no overflow detector, the result R3, which is the sign bit in case of no overflow, is 1. But if there is overflow, the R3 is 0 and so sign_bar is R3_bar if the overflow is present.

So with $OF = 0$ R3 is the sign, but if $OF = 1$, we have R3_bar.

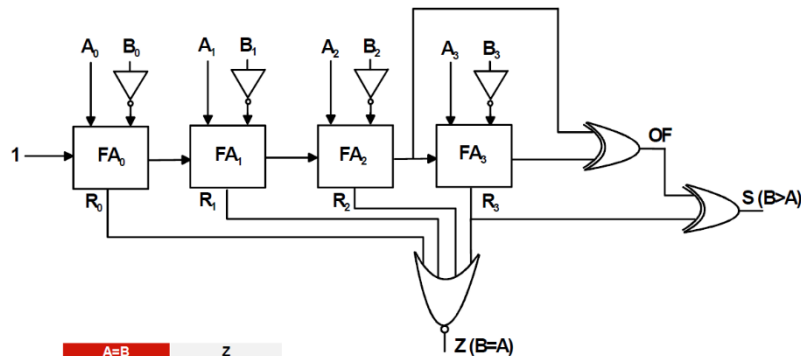
Basically I have to implement a subtractor and an OF detector and the OF result and the sign result are put in EXOR in the final gate to get S.



The actual sign of the difference A-B is $OF \oplus R_3$

1. If $OF=0$, then $OF \oplus R_3 = R_3$
2. If $OF=1$, then $OF \oplus R_3 = \overline{R_3}$

Let's add some circuitry to get different conditions as in the table below.



A=B	Z
A≠B	Z'
A>B	(S+Z)'
A<B	S
A≥B	S'
A≤B	S+Z

It is a magnitude comparator with a NOR gate. S is a flag that goes to 1 if $B > A$ as seen. Z is a flag that is 1 if $B = A$. If so, $A - B = 0$, so all the bits in output should be 0.

Now we can combine Z and S.

BINARY MULTIPLICATION

Let's consider two digital words, X of N bits and Y of M bits

The product $Z=XY$ is a (N+M)-bit word:

$$Z = \underbrace{\left(\sum_{i=0}^{N-1} x_i 2^i \right)}_X \cdot \underbrace{\left(\sum_{j=0}^{M-1} y_j 2^j \right)}_Y = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_i y_j 2^{i+j}$$

The multiplication involves the AND of one word with all the inputs of the other word.

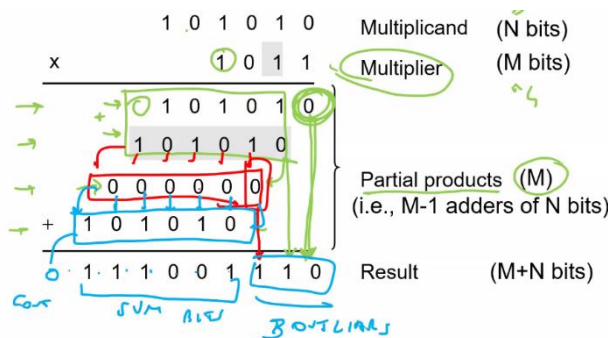
Example

	1 0 1 0 1 0	Multiplicand (N bits)
x	1 0 1 1	Multiplier (M bits)
	1 0 1 0 1 0	}
	1 0 1 0 1 0	
	0 0 0 0 0 0	
+	1 0 1 0 1 0	
	1 1 1 0 0 1 1 1 0	Result (M+N bits)

The first partial product is equal to the multiplicand because the LSB of the multiplier is 1. The same for the second partial product, but shifted. Then the third partial product is 0 shifted since the third digit of the multiplier is 0.

In a multiplier we have M partial products, where M is the number of bits of the multiplier, each made of N bits, the number of bits of the multiplicand. Then the result is the sum of the partial products. So the multiplier is made of M-1 adders of N bits.

In the end we have 3 outliers, the sum bits and the Cout. The result is made of 10 bits. This is the way to implement the array multiplier, summing the partial products.



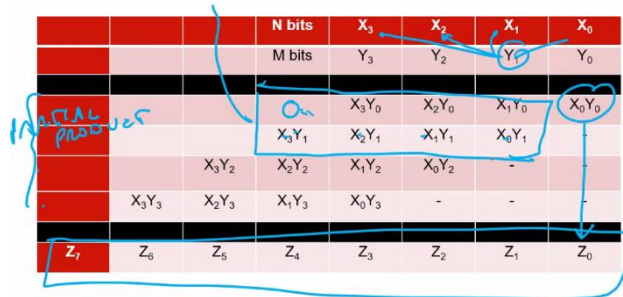
4x4 BINARY MULTIPLIER

			N bits	X_3	X_2	X_1	X_0
			M bits	Y_3	Y_2	Y_1	Y_0
				X_3Y_0	X_2Y_0	X_1Y_0	X_0Y_0 X
				X_3Y_1	X_2Y_1	X_1Y_1	-
		X_3Y_2	X_2Y_2	X_1Y_2	X_0Y_2	-	-
	X_3Y_3	X_2Y_3	X_1Y_3	X_0Y_3	-	-	-
Z_7	Z_6	Z_5	Z_4	Z_3	Z_2	Z_1	Z_0

The multiplication is just a AND gate. Each term inside the table corresponds to an AND gate, aside from the adders.

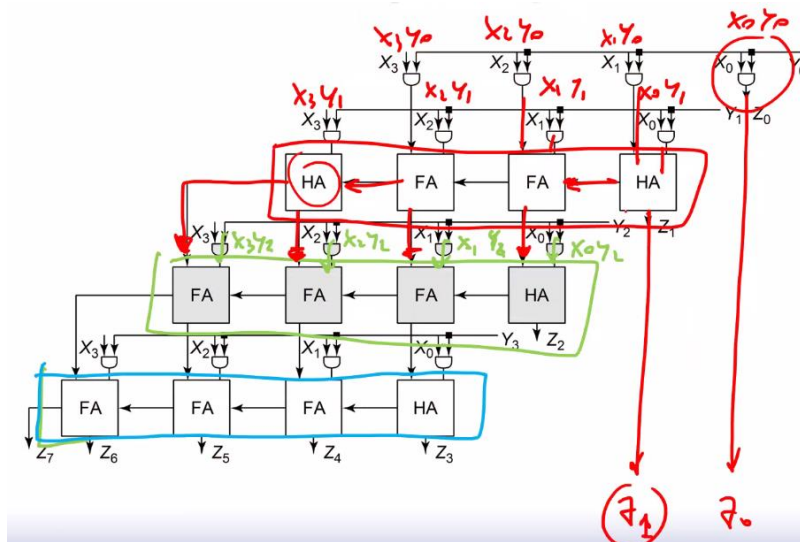
As for the second row, aside from the shift, we have 4 terms, and the same for the 3rd and 4th partial products. Then we need to sum them, and the result is made of 8 bits.

X is an outlier, so goes directly to the output. Then we need to assess the sum of the two words, and in each column we have bits with the same weight. It is the first adder, which is an adder of 4 bits.



The result is of 5 bits. The LSB of the sum goes directly to the output, and we have a carry that is propagate to the next column. So the result of the previous adder must be summed with the 3rd row, and we have the second adder, which receives in input 5 bits (a carry and 4 digits). Then the result of the second adder is fed to the last adder.

We need to implement M-1 adders of N bits. Aside from the first outlier, then we have words that must be summed together. We want to use AND gates, since all the terms are AND functions (so we need 16 ANDs) and 3 adders of 4 bits. The implementation is the following.



We have the three adders (the three groups of FA), and we can recognize also the AND gates. Then on the left we see also the outlier, X0Y0, which corresponds directly to Z0. The red adder has as first input of the half adder X1*Y0 and X0*Y1. Then the carry is propagated to the next FA which receives in input X2*Y0 and X1*Y1 plus the previous carry. Then we go on completing the table.

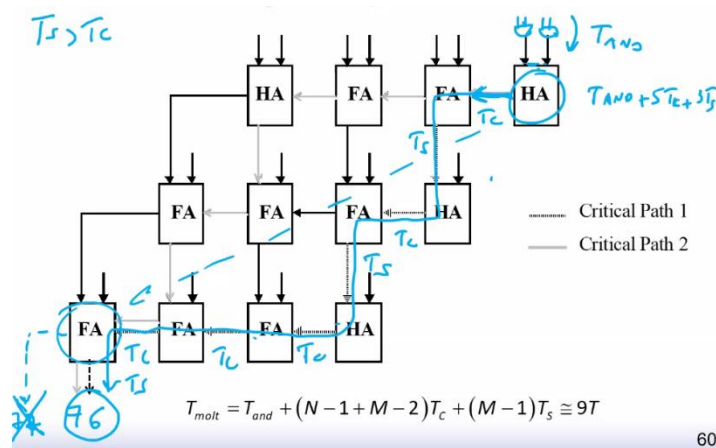
The only 'outlier' is the last HA, which must sum X3*Y1 and 0 and the last carry. So there is no reason to use a FA with a 0 in input, we can use a HA with X3*Y1 and the last carry in input. Then the results of the red adder are Z1, directly at the output, and other 4, which are then fed to the next adder in green.

So in the end we need 16 AND gates, 4 HA and 8 FA.

Critical path

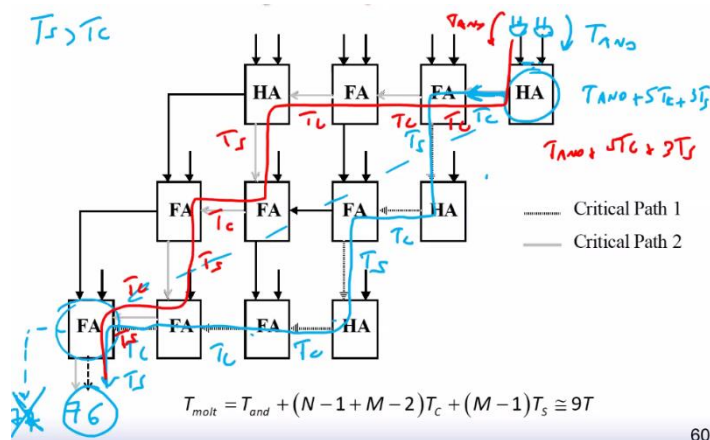
The worst case delay corresponds always to the MSB of the result, so Z7 or Z6. The output of the last bottom left FA are the worst case signal, and it corresponds to something that happens in the first HA and has to propagate down to the last FA.

Let's suppose that $T_s > T_c$. Typically it's Z6 the last signal to transition in this case, and we reach this critical condition, as said previously, when something happening in the first HA has to propagate down to it (blue path).



We have to account for the delay of the AND gates, then for the T_c and T_s for the carry and the sum. Every time I move to the left I have to consider a T_c , down a T_s .

So in the end I have: $T_{and} + 5 \cdot T_c + 3 \cdot T_s$.



There is, however, another critical path, the red one, which is $T_{and} + 5 \cdot T_c + 3 \cdot T_s$.

Since this array multiplier is done implementing $M-1$ N bit adders, we can write the generic expression (for the array multiplier) in the bottom of the image.

Instead, in the case $T_s < T_c$, the last signal to transition is Z_7 , but the critical paths are always the same. The relationship is the same but we have to swap the T_{sum} and T_{carry} .

This is different from the ripple carry adder, because the carry and sum delay play more or less with the same weight, it is not dominating the T_c . So for this adder the TG full adder is a good choice because it allows to implement the carry and sum circuit in a lean way, since both delays must be accounted for.

SEMICONDUCTOR MEMORIES

Whatever the type of memory, the architecture is typically the same. Aside from the element that stores the bit, also the auxiliary circuits are the same and are 3, mandatory for the correct working of the memory:

- Row decoder
- Column decoder
- Sense amplifier: needed to speed up the read phase of the bit in case we are using minimum size transistors to store a single bit, because they have a very small driving capability.

The most important figure of merit is not the power consumption or the propagation delay, but the **cost per bit**, so how much area we need to store a single bit. Every effort has to be done in order to decrease the area to store a single bit.

OVERVIEW

- Semiconductor memories are circuits able to store a large amount of data.
- Global memory chip market is expected to reach about \$250 billion by 2023
- Usually, the size is expressed in terms of bytes (1 byte=8 bits).
- The most important figure-of-merit is the **integration density**, which determines the cost per bit.
- Different types of classification:
 - A. VOLATILE / NON-VOLATILE
 - B. READ ONLY / READ-WRITE
 - C. RANDOM ACCESS / NON-RANDOM ACCESS

We cannot use FFs to store a large amount of bits since already a FF employs 20 transistors, and so we would need too many transistors. Semiconductor memories are dedicated to store a large amount of data using the smallest possible area.

Foreground memories are the memories done with latches and FF to implement counters and dividers seen so far, but we also have background memory, dedicated to store a large amount of data.

Every effort has to be done to improve the integration density, that is the cost per bit.

Moreover, a volatile memory is a memory that is not able to retain the data if the PS is switched off. Then we can distinguish ROM and R/W memory, that can also be written. An example of ROM is the BIOS, set of instructions to power on the computer.

Instead, write and read memory are SRAM (used as cache memory) and DRAM, that is a non-volatile memory. Then the access can be random access, i.e. we can read each bit in the memory, or non-random access memories. Most of the memories are random access.

SEMICONDUCTOR MEMORY CLASSIFICATION

On the left I have read/write memories, while on the right we have ROM memories. ROM memories are non-volatile, since they can only be read.

The ROM can be mask-programmed (done by the manufacturer) or PROM, programmed by the user with fuses.

READ-WRITE		ROM	
Read-Write Memory (Volatile memory)		Read-Write Memory (Non-Volatile)	Read-Only Memory (Non-Volatile)
Random Access	Non-Random Access	EPROM E ² PROM FLASH	Mask-Programmed (by foundry) PROM (by user with fuses)
SRAM DRAM	FIFO LIFO Shift Register CAM		

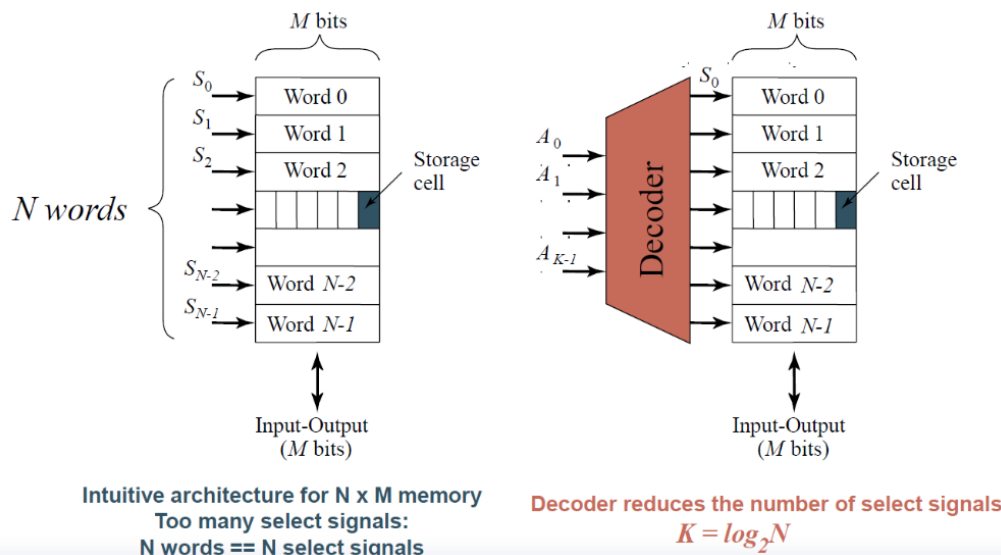
R/W memory can be non-random access, like in the case of the FIFO or LIFO. SRAM and DRAM are volatile R/W memories, with random access. Also FLASH and mask-programmed memories are random access, so the SRAM should be called SRWM.

Aside from the non-random access memories, all the remaining memories are organized with the same architecture.

Memory architecture - Decoders

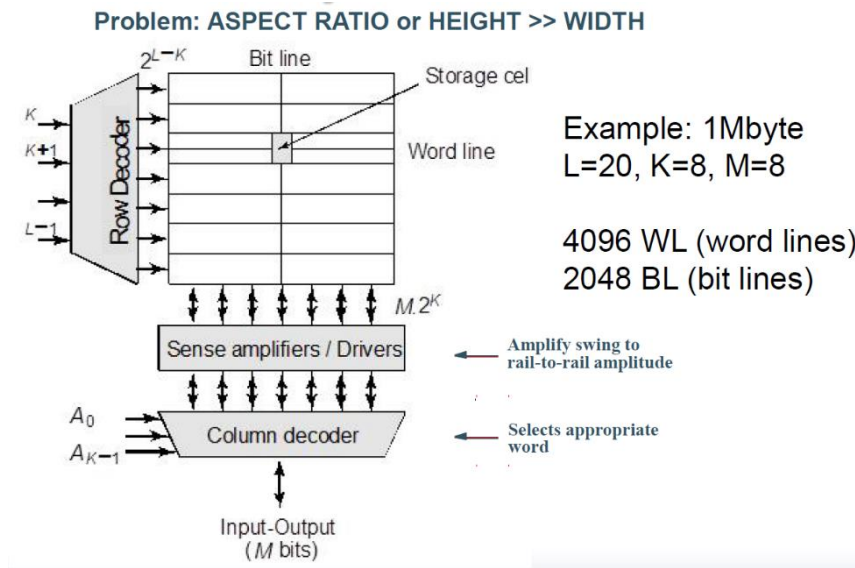
We have a memory organized in N words, where a word is made of 8 bits (one byte), and the words are like 'stacked'. In this architecture that is organized in words and columns, every bit corresponding to the same column is connected to the same bit line (that is nothing more than a wire). But then we also have different rows, and we can see S₀, S₁ etc... as select signal to select the specific row.

The problem is that we have to handle a large amount of select signals for Mb memories. So we use a row decoder, that receives a certain number of bits in input and, with k inputs, outputs 2^k outputs. In this way we reduce the amount of select signals.



So for each S₀, S₁ and so on we have to implement a gate that goes to 1 for a particular code in input. It should be a gate whose output is 0 for any other combination than the correct one. So we use 2²⁰ (1M) AND gates with k bits in input. Since the AND is not implementable in FC-CMOS, we could use a NOR gate.

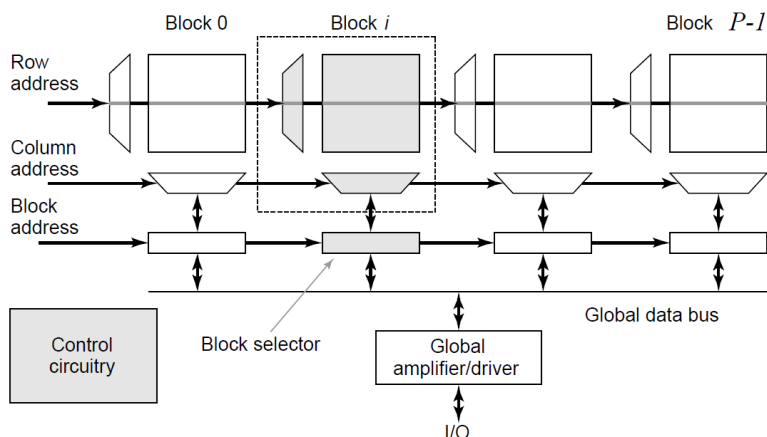
The other issue is the aspect ratio of the memory, which is that the memory is very tall but not large, so we can change the architecture, re-arranging the words so that the aspect ratio is more or less a square. Inside the same line we have more words of 8 bits.



In this case we need to add, if we are not considering the sense amplifiers so far, also a column decoder. Per row, we have 2^8 words (256). So the column decoder, which is a MUX even if it is called decoder, can be implemented with 8 select signals. Now, the row decoder is made with a smaller amount of bits, that is 12. In the end I still have to have 2^{20} select signals.

Hierarchical memory architecture

Sometimes we have also a **block decoder**. In the previous implementation, the bit line is a very long wire, as well as for the word line. Typically, a large memory can occupy cm in both directions; a wire that is long few cm introduces a large delay, because it is a combination of parasitic distributed resistances and capacitances. To avoid the large delay corresponding to a long wire, the memory is split in blocks.



Advantages:

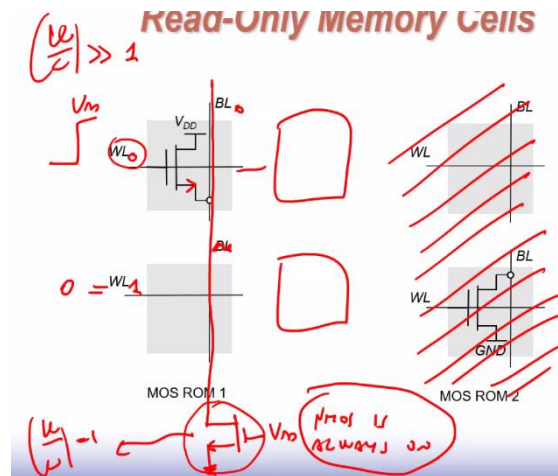
1. Shorter wires within blocks
2. Block address activates only 1 block => power savings

Each block as a row and column address, but also a block address as a select signal. So we split the overall addressing in 3 parts, and this has the advantage of saving power consumption and delay.

ROM CELLS

We store a bit with just one single transistor. If we consider a bit line connecting all the bits, we can add an nMOS connected to ground. It is a pull down device that pulls the line to ground. It can be considered always on.

The WL (word lines) are activation signals, BL is the bit line. We are focusing on BL0, in common to all the cells belonging to the first column, and it is kept to GND by a minimum size transistor (aspect ratio of 1).



The grey corresponds to a cell where we store a single bit. Let's suppose that the first nMOS is turned on, and it has a large (W/L). Of course, WL1 is 0 because we can read one word at a time. The BL0 is pulled up because the nMOS has a large size, so it is raised to a logical 1.

If we want to read the second location, we pull up WL1 and set WL0. There is no transistor in WL1, nor an intersection with BL0. So BL0 after a transient is 0, so since there is no transistor in the cell, the BL0 remains 0. So to create a ROM we implement or not a PU device.

To work properly, only one WL can be pulled up at a time, so we can only read one world at a time.

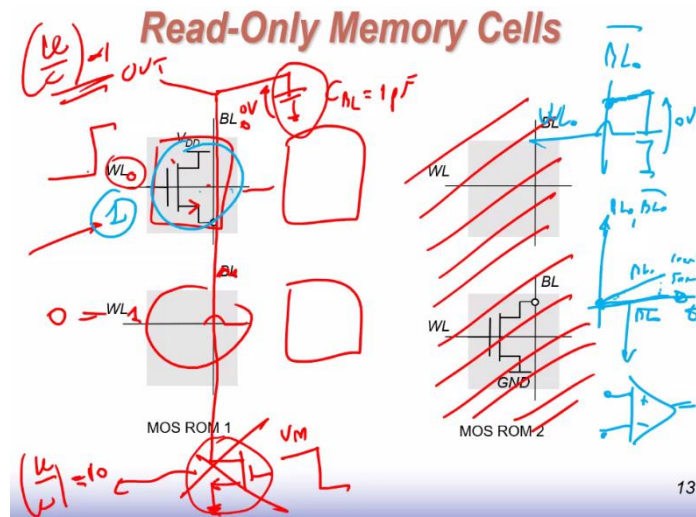
We have some issues. Firstly, the PDN nMOS is not minimum size (10) and dynamic, so not always on, while the ones in the cells are small, minimum size. So in theory the PU is not effective, but it happens since the PD nMOS is dynamic.

Before reading the memory, all the WLs are 0, and the BL has an overall capacitance of $C_{bl} = 1pF$ that is discharged since the PD nMOS is on. Then the PD nMOS is switched off and the capacitance C_{bl} remains floating, keeping 0V across its terminals. Then we want to read the WL1, so we pull up the WL and we read 0 since the C_{bl} is not changing. If instead we read WL0, of course it takes a lot of time since the transistor is minimum size, but after a while the output is $V_{dd} - V_t$.

This is typically an architecture that is differential, so every cell has another cell that is storing the complementary bit. We have always this differential architecture, so to store a bit we always need 2 cells. So when $WL_0 = 1$, we have that C_{bl} is pulled up slowly, but we will also have the counter-cell that will be without the transistor implemented, so its capacitance will remain 0. If we plot BL_0 and BL_0_{bar} , they both start from 0, and one increases while the other remains 0.

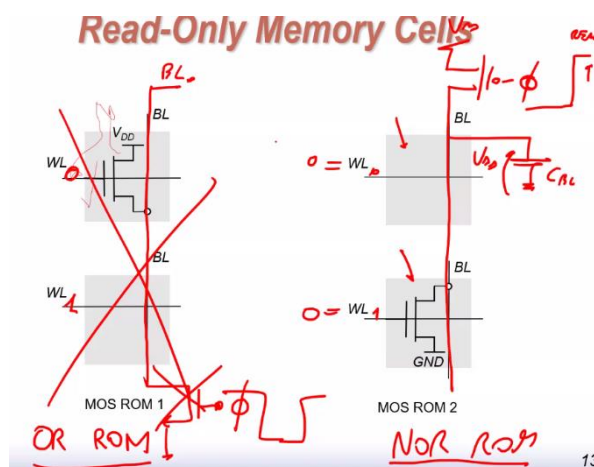
The voltage difference between the two is sent to an amplifier that is turned on and the output is quickly regenerated.

In fact, if we wait for the nMOS to pull up BL_0 , it would require too much time, and I can shorten the time using a **sense amplifier** and a differential signal. This also explains how we can use minimum size transistors inside the memory.



This is known as **OR ROM**, since it resembles an OR gate. The PD device nMOS is connected to a signal that is Vdd before the reading phase, and 0 during the reading phase. It is typically not used, despite the same area taken with respect to the NOR ROM, because of high propagation delay, i.e. access time (read time), because nMOS transistor is not so prone to pull a node.

On the right we have instead the **NOR ROM**, that is the one typically used. Now we have a PU device connected to Vdd, and the signal is 0 and released to Vdd during the reading phase. Cbl, before the reading phase starts, it's charged to Vdd. Then inside the cell we can have nothing or a transistor device to ground, so it's the opposite than before.



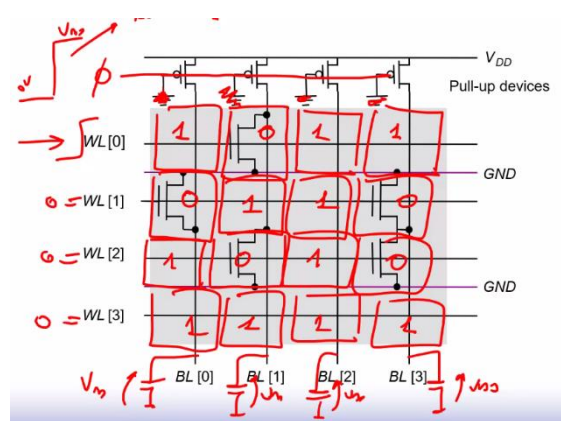
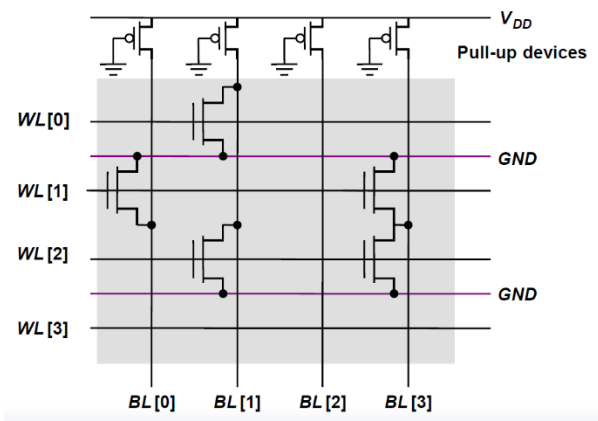
If we want to read WL0, we pull it up and we read a 1, while if we read WL1 the nMOS is activated and we read a 0. Also in this case the architecture is not single-ended, we have the complementary counterpart.

This gate resembles a NOR gate because if one input goes to 1 the output is discharged to ground.

MOS NOR ROM

On the top we have the PU devices, connected to GND (in reality to a signal phi that is 0 before the read phase and Vdd during the read phase, so it's a dynamic PU). We have 4 word lines and 4 bit lines in this example.

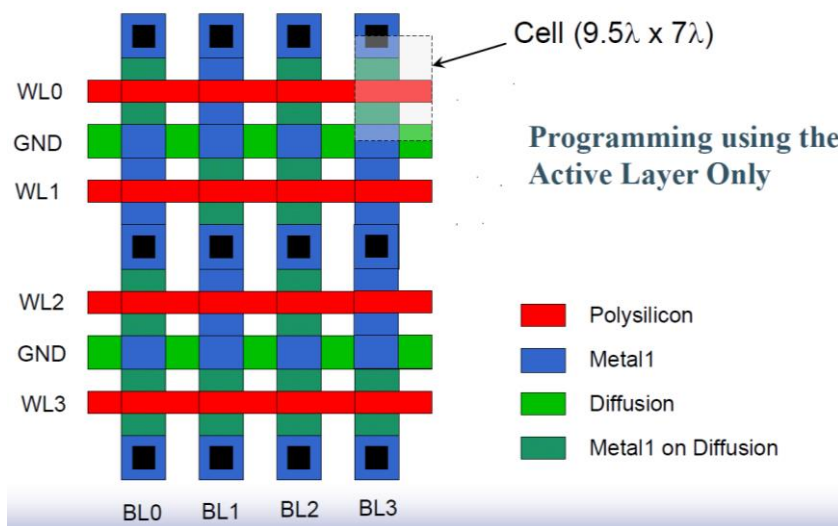
Before the reading phase, all the BL capacitances are charged to Vdd. Then let's suppose we want to read the bit corresponding to WL0. All the other WL are at 0. In the first location we read a 1 because there is no transistor, in the second cell we have a nMOS so from BL1 we read 0 and so on, reading 1011.



Remark

If we look at the ground connection, it is brought via the purple wires, and it is in common between two rows, so rows are folded to share the same ground. So bit lines are implemented with metal wires, word lines are implemented in polysilicon and ground wire is implemented through diffusion. This is to spare area.

Layout



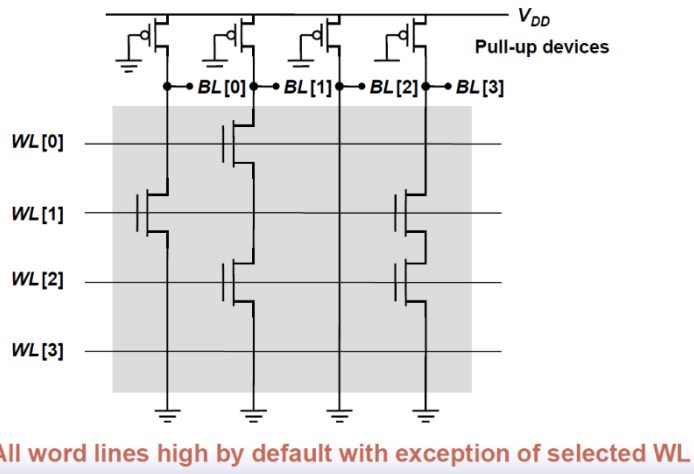
MOS NAND ROM

In the NOR ROM, only one WL is on at a time, and the transistors are 'connected in parallel' for each BL, since they share the same drain, and the sources are connected to ground. The difference is that the gate terminals are different.

In the NAND ROM, transistors are connected in series. The pMOS are connected to a signal phi that is normally 0, and Vdd during the reading phase.

As for the architecture, in this case, the bottom of the BLs are connected to ground, and the transistors implemented in the cells are in series. To work properly, all the WL has to be 1, and the one we want to read must be 0.

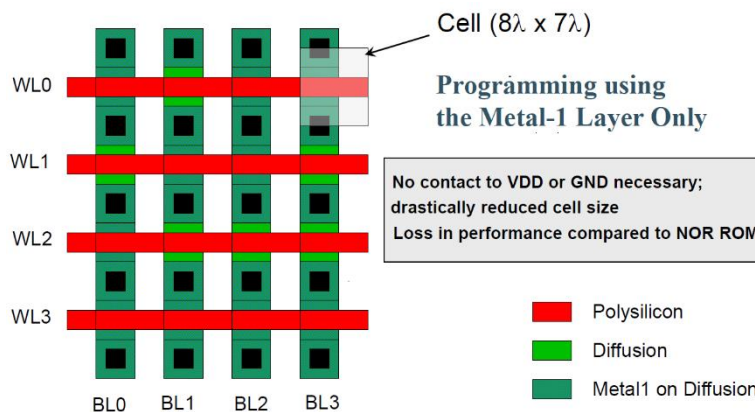
Let's focus on the 00 cell. The transistor in WL1 on BL0 is on, since connected to Vdd. So BL0 is 0, since the BL capacitance is discharged to ground. Then to read WL1, it has to be 0 and so the nMOS is off, but all the other WL are 1. So BL0 remains at 1, since it has been charged between the reading phase begins. So the absence of a transistor means a 0, the presence a 1 in this architecture.



If we look at a single column, it resembles a NAND gate with transistors in series, this is the reason for the name.

The main difference with respect to the NOR architecture is in the fact that there is no ground wire signal, so in this case we have smaller area per bit.

Layout



So the area is improved in this architecture, but another figure of merit is worsened, and it's the delay. In fact, for the same C_{bl} in the NAND ROM architecture we have minimum size transistors in series, which means a large resistance to pull down the capacitance.

EQUIVALENT TRANSIENT MODEL FOR MOS NOR ROM

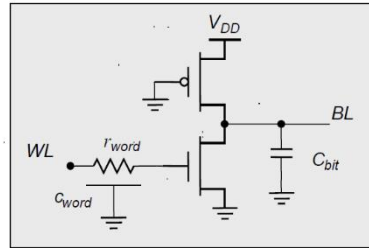
A memory architecture is organized in WLS and BLs. I give a signal on the WL and I want to read the content of a BL. There are two times to consider, the propagation delay of the WL signal and the time it takes for the BL to be discharged, in the case of the NOR ROM.

This is why in the model we have highlighted the WL as a combination of RC cells (r_{word} , c_{word} , the specific resistance and capacitance, and the capacitances are due to gates and polysilicon).

Then we reach the farthest transistor on the right side which is minimum size ($R = 13 \text{ k}\Omega$) and has to discharge the C_{bl} . The BL is metal wire and very conductive and can be considered as just a capacitance, since the resistance is not dominant (metal).

This C_{bl} is made of two contributions: the metal capacitance and the drains capacitances. So the access time of this type of memory is divided in two contributions.

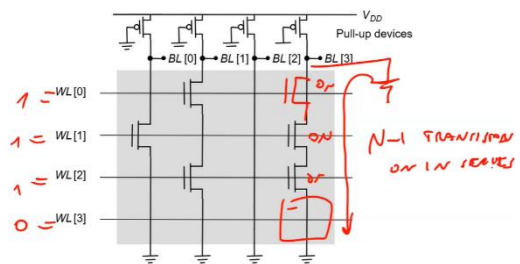
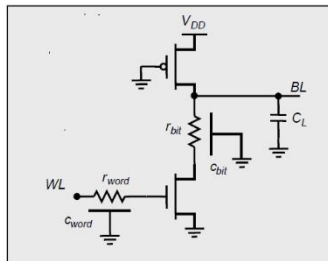
Model for NOR ROM



- Word line parasitics
 - Wire capacitance and gate capacitance
 - Wire resistance (polysilicon)
- Bit line parasitics
 - Resistance not dominant (metal)
 - Drain and Gate-Drain capacitance

EQUIVALENT TRANSIENT MODEL FOR MOS NAND ROM

Model for NAND ROM

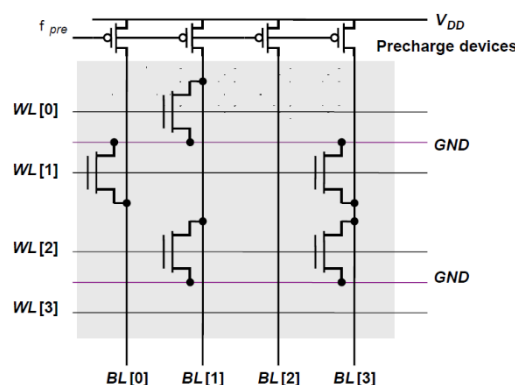


- Word line parasitics
 - Similar to NOR ROM
- Bit line parasitics
 - Resistance of cascaded transistors dominates
 - Drain/Source and complete gate capacitance

Now the WL is pulled down when we want to read and the signal travels from left to right, and the worst case is when the signal has to reach the BL on the right side and we have to pull down the Cbl through all transistors, so N-1 transistors on in series, where N is the number of WLs.

The combination of transistors is represented in the model with a wire resistance and capacitance.

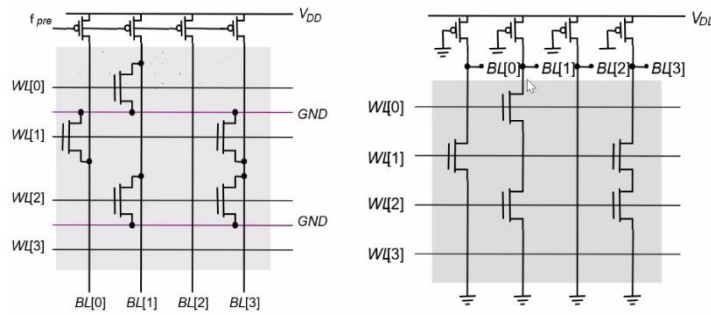
PRE-CHARGED MOS NOR ROM



PMOS precharge device can be made as large as necessary, but clock driver becomes harder to design.

In the NOR or NAND architecture, the PU device (or PD) doesn't work as static device, always on, but switched off when the reading phase starts, and we have just capacitances charged to Vdd or GND.

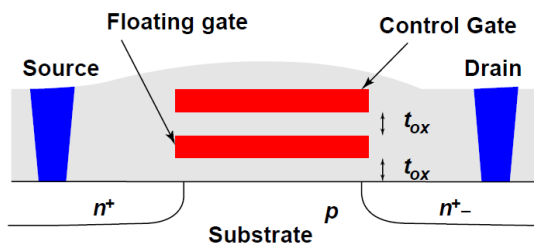
NOR AND NAND ROM



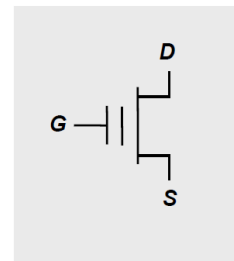
This memory is programmed by the foundry. Let's suppose that, instead of the presence or absence of the transistor, we have a transistor in every cell in the NOR case. And let's suppose that the transistor can be adapted, so we can dynamically change its threshold voltage so that it's always on or off. This is the basic idea behind the FLASH memory. The same thing can be done for the NAND ROM. This transistor is called **floating gate transistor (FAMOS)**.

FAMOS

In the oxide between the control gate and the substrate we insert a floating gate of PolySi, which behaves as a tank for charges, it's able to store or remove electrons. In this way we can change the threshold.



Device cross-section



Schematic symbol

$$V_T = V_{FB} + 2\phi_F + \frac{\sqrt{2\epsilon_s q N_a (2\phi_F + V_{SB})}}{C_{ox}}$$

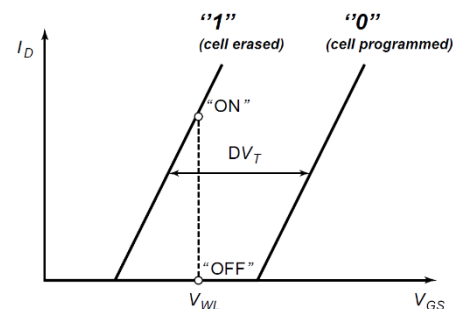
$$V_{FB} = \Phi_{MS} - \frac{Q_f}{C_{ox}} - \frac{1}{C_{ox}} \int_0^x \frac{x}{x_{ox}} \rho_{ox}(x) dx$$

Let's consider the floating gate full of e-. If we increase the gate voltage, the e- act as a barrier, so it is more difficult to invert the channel and create an inversion channel, so the V_T is increasing. So **we cannot drive the floating gate but we can create e- inside it to increase the threshold voltage with respect to the nominal value, or remove them.**

When we remove electrons we are doing an **erasing**, while if we are adding them a **programming**.

The one on the left is the classical characteristic, while the one on the right is the one with the increase in threshold. We can move from one characteristic to the other acting on the floating gate e- content. So they can either act as an open circuit or a normal transistor depending on the threshold voltage.

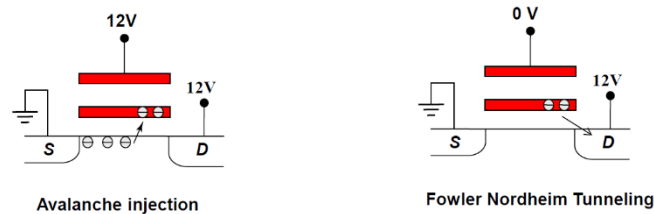
This is a way to implement FLASH memory.



FAMOS programming and erasing

On the left we have a high V_t , on the right a nominal or low V_t , since erased, and it behaves as a normal transistor.

How to write (program) and erase a FLASH cell?



The programming phase is made with **avalanche injection**. We apply a very high voltage to the control gate pin (12V, so the technology must withstand it) and this means a layer of electrons in the substrate, since we are inverting the channel (in the nMOS case). But we apply also a high voltage at the drain side, with the source to ground. We will have a high electric field that accelerates e^- which become **hot electrons** with high energy and are able to jump across the oxide and reach the floating gate, where they remain trapped.

To erase the cell we apply 0V to the control gate, like if we are switching off the transistor, and a high voltage at the drain side. In this case there are no free e^- in the channel, and due to the high electric field we remove electrons from the floating gate, since they are attracted by the drain \rightarrow **Fowler Nordheim Tunneling**.

So to program a memory firstly we erase all the cells, and then we program the cells we want to program, but this is a multi-phase procedure. We apply the 12V, we check the threshold and if the threshold is not the one needed the operation has to be repeated up to the desired value. So the write of a flash memory take some time, hundreds of us. So the problem is the time taken for the write phase to write the memory.

This is a very non-volatile memory in the sense that the e^- remain trapped in the layer once created, even if we switch off the power supply. It is a R/W memory non-volatile. It is not used everywhere but only in substitution of the hard disk drives because it is slow, the write phase is slow (hundreds of us).

There is also another problem, and it's the power consumption. Indeed, we have to apply a lot of V to write it, and also we have a large current in the transistor.

Then, as a last issue, because we can write and erase the memory only for some times, like 1 mln times. At a certain point, the oxide is damaged and the e^- that are trapped are immediately released, so we are not writing correctly the memory \rightarrow **leakage, retention of data**. SRAM and DRAM don't have this last problem.

SRAM

Difference between SRAM and DRAM

Both SRAM and DRAM are volatile, in the sense that the memory loses the stored data if we switch off the power supply. The SRAM is based on a bistable element (static latch) and a couple of transistors to implement switches, so we have 6 transistors per cell. Moreover, the SRAM is differential and fast with respect to the DRAM, in terms of access time.

On the other side, the DRAM requires a periodic refresh, and it is based on a transistor and a capacitor (3 transistors per cell is a very old architecture). The storage capacitance is a physical capacitance implemented in the cell. Between two consecutive reads we need to refresh the content of the cell.

Moreover, intrinsically it's single-ended, but it is nevertheless implemented in a differential architecture composing cells in a certain way.

Once we read the DRAM cell, the voltage swing is not so big, so we have to distinguish a 0 and a 1 between two values that are not so far in terms of voltages. Because of this the differential architecture is needed so that we can in a way remove a sort of digital noise.

□ STATIC (SRAM)

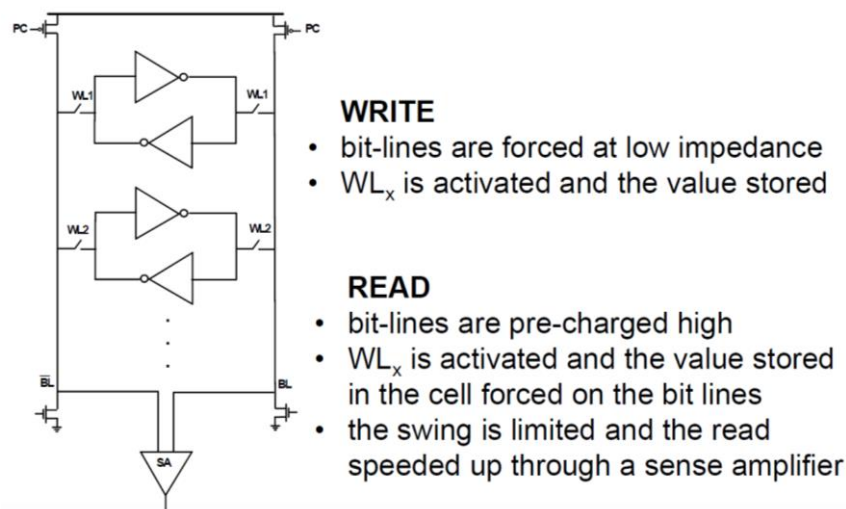
Data stored as long as supply is applied
Large (6 transistors/cell)
Fast
Differential

□ DYNAMIC (DRAM)

Periodic refresh required
Small (1-3 transistors/cell)
Slow
Intrinsically single-ended

SRAM ARCHITECTURE

We still have a WL and BL organization. Vertically we have BLs, horizontally WLs, but we also have a differential structure, so BL0 and BL0_bar, for each bitline we have the complementary one. In the image, on the left we have BL_bar, on the right BL. Each bistable element and two switches represent the memory cell. Each cell is made of a bistable element and two nMOS that behave as pass-transistors that have to isolate the bistable element or connect it to the BL to write or read the cell.

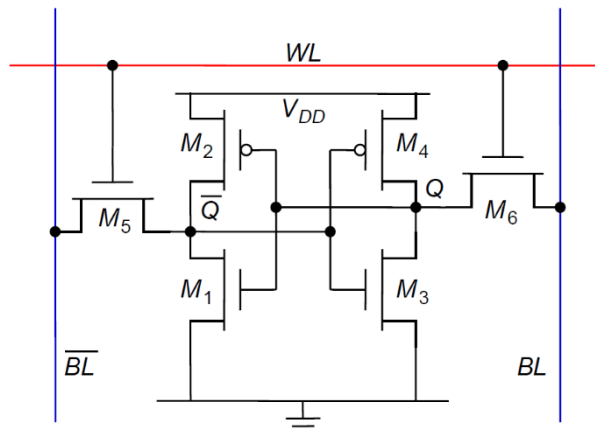


We need also some auxiliary transistors. On the top we have two pMOS that behave as **precharge devices**. It means that, before the read phase begins, all the WL are 0, so the cells are disconnected from the BLs and the BL and BL_bar are pulled up since the pMOS are on. BL and BL_bar lines behave as capacitors

(fF) and they are pre-charged. Then during the read phase we turn on one of the WL and we create a differential signal across the two BLs, BL and BL_bar. So the PU devices are mandatory for the correct working (read) of the circuit.

On the bottom we have also other devices used to write the cells belonging to the column. To write a cell, we need to PU the corresponding WL and then we force the BL at low impedance, one to Vdd and the other to GND.

6-transistor CMOS SRAM single cell



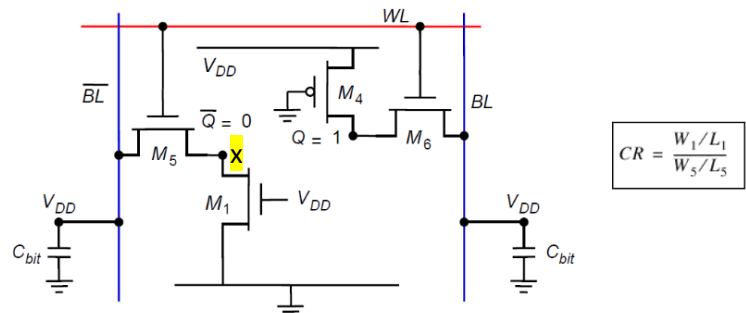
In the middle we have the bistable element. Then M5 and M6 are the pass transistors, driven by the WL signals (red line), while the blue lines are BL. The WL is implemented typically in PolySi and it has to connect all the gates, while the BL is typically implemented in metal1 and all the BLs run in parallel. Q is the true value, Q_bar is the complement value. Let's suppose that Q = 1, so the cell has been written with a 1. So on the right side the pMOS is on and the nMOS is off, and the opposite in the left inverter.

This situation is depicted in the following image, where transistors are deleted to simplify the scheme.

READ PHASE

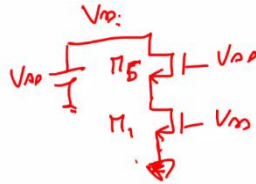
Only one WL is on at a time, so we read all the cells belonging to the same row, since the WL signal is in common. Three operations are performed:

1. **Precharge:** WL = 0V, while PC (precharge) is connected to 0 and we precharge BL and BL_bar. They are metal wires, so the resistance is negligible but the capacitance is not. The capacitance is the metal1 capacitance plus all the contributions of the transistors (drain contributions) connected to the BL. The PC capacitances are charged at Vdd on BL and BL_bar.
2. **Activation of WL:** WL goes to 1, that is Vdd, that is 2.5V, so the pass transistors are turned on. Once we pull up the WL, the situation is the one in the image aside, at t = 0. At the gate of M5 we have Vdd, so there is an implausibility if we have 0V at node x. I have two nMOS in series connected to ground, it is like the following.



$$k_{n,M5} \left((V_{DD} - \Delta V - V_{Tn}) V_{DSATn} - \frac{V_{DSATn}^2}{2} \right) = k_{n,M1} \left((V_{DD} - V_{Tn}) \Delta V - \frac{\Delta V^2}{2} \right)$$

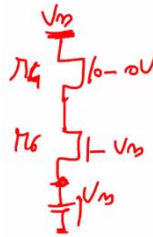
$$\Delta V = \frac{V_{DSATn} + CR(V_{DD} - V_{Tn}) - \sqrt{V_{DSATn}^2(1 + CR) + CR^2(V_{DD} - V_{Tn})^2}}{CR}$$



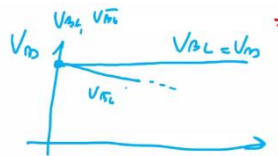
So we have a current from Vdd to ground. So the voltage in the intermediate node cannot be 0, but it rises to a value $\Delta(V)$ which depends on the relative size of the two transistors.

Returning to the SRAM, BL_bar voltage that was at Vdd drops. At the beginning it drops linearly since we have a capacitance and a constant current in the two transistors.

If we focus on the right side, we have the following situation. The pMOS is on and also the nMOS. But there cannot be current because we are between Vdd and Vdd.



Of course the situation is the opposite one if we consider a cell written with a 0. So BL_bar is discharged, BL remains the same. Let's plot on a graph V_{BL} and V_{BL_bar} .



If we wait for a time long enough, the final value reached by V_{BL_bar} is 0V, because we discharge the capacitance and then we are between 0V and 0V with the two transistors. However, we don't reach 0V because after few ns after the linear trend, after a time fixed a priori, we activate the sense amplifier.

3. **Sense amplifier enable:** its activation must be fixed a priori to speed up the read phase.

Read upset

There is however an issue, the **read upset**.

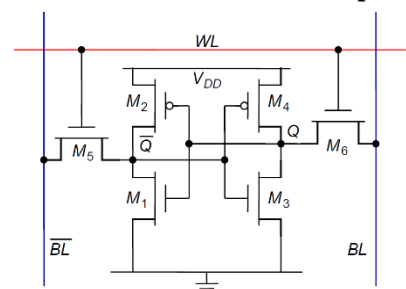
We said that the x point goes to $\Delta(V)$, which is a value function of the aspect ratio of the transistors. If M1 is very large and M5 is very small, $\Delta(V)$ is close to ground, because M1 is a smaller resistance than M5. If $\Delta(V)$ is very large, we see that node x (Q_bar) is the input of the other inverter of the stable element. If $\Delta(V)$ is greater than the threshold voltage of the inverter, the output of the inverter toggles to 0V in output, and if so also the inverter on the left side has 0V in input and it toggles to Vdd in output. So node Q_bar reaches Vdd → during the read phase we have written the cell and stored a 0 in the cell.

To avoid read upset, we need to be sure that $\Delta(V)$ is not too large. This means that we have to respect the following condition: **$\Delta(V) < V_m$** .

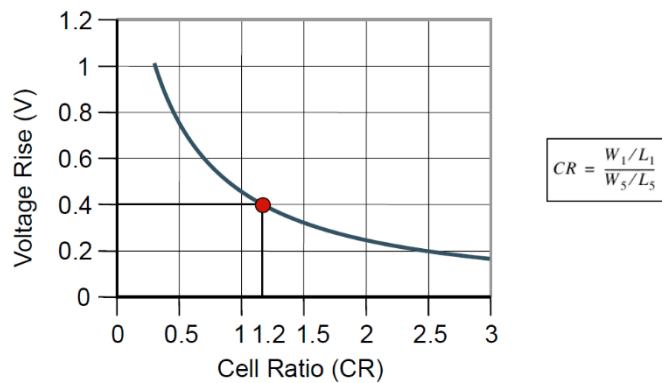
Typically, digital designers require a more stringent condition: $\Delta(V) < V_{tn} = 0.43V$. This more stringent requirement means that M3, which is off, is not turned on.

We have to size accurately M5 and M1 hence.

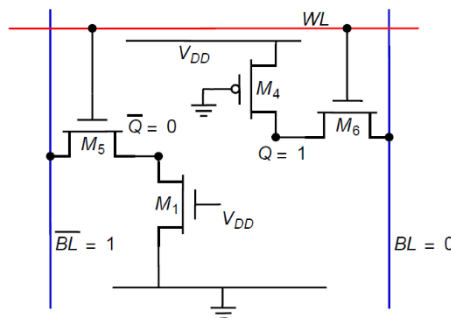
Let's write the currents in the two transistors at the beginning when we activate the WL. M1 is in ohmic, while M5 is in velocity saturation.



Simulation results



WRITE PHASE



$$PR = \frac{(W/L)_4}{(W/L)_6} \quad V_Q \cong \frac{k'_{p4} V_{DASTP} \left(V_{DD} - V_{TP} - \frac{V_{DASTP}}{2} \right)}{k'_{n6} (V_{DD} - V_{TN})} PR$$

Let's suppose to have a bistable element written with a 1. To change the content, we need to connect the right side to 0 and the left side to 1, as in the image above.

On the left side we have M5 and M1 between Vdd and GND, so there is a current, but since M1 is larger than M5 to avoid read upset, the write phase is not effective on the left side. As for the right side, we have a current since Vdd and GND between M4 and M6, so the voltage V_Q drops from Vdd down. We can rewrite the situation as follows. Which is the voltage V_Q that enables the cell to toggle?

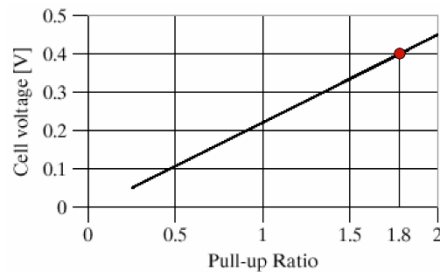
V_Q is the input of the inverter on the left side, so it has to be smaller than V_m, switching threshold of the inverter. However, for safety reasons we have a more stringent condition: **V_Q < V_{tn}**, a condition that allows to turn off M1.

$$I_{p4} = \frac{k_p}{2} \left(\frac{W}{L} \right)_4 V_{DSATP} \left(V_m - V_{TP} - \frac{V_{DSATP}}{2} \right) = I_{n6} = k_n \left(\frac{W}{L} \right)_6 V_Q \left(V_m - V_{TN} - \frac{V_Q}{2} \right)$$

Let's write the expressions for the current. First of all, we assume V_Q close to ground, so M4 is in velocity saturation, M6 is in ohmic. Again, we neglect the channel modulation effect. V_Q/2 can be neglected because it is very small with respect to Vdd.

$$V_Q \approx \frac{k'_p}{k'_n} \cdot \frac{V_{DD} - V_{TP} - \frac{V_{DD} - V_{TP}}{2}}{(V_{DD} - V_{TN})} \cdot \frac{(W/L)_4}{(W/L)_6}$$

$(W/L)_4/(W/L)_6$ is called **pull-up ratio (PR)**. The result is that **PR < 2**, so the pMOS has to be very resistive with respect to the nMOS. The following is the result of the simulations.



Overall reasonable sizing

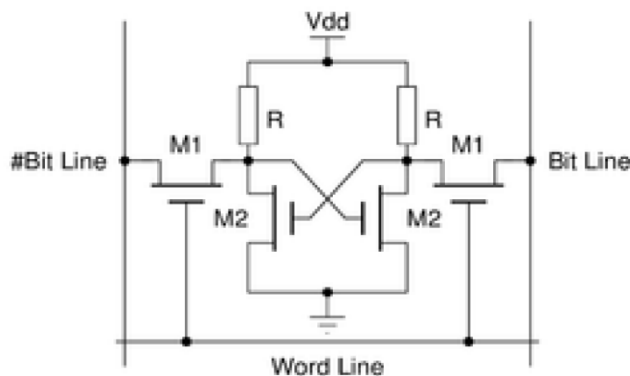
CR > 1.2 and PR < 2 (for 0.25 um CMOS from INTEL). So if M5 (1), M1 must be at least (1.2). Then we can size M2 with (1) because of the PR. Of course, the smaller the aspect ratio, the better, because we reduce the area occupation.

Once we have the sizes of all the transistors, if we consider the read phase it is just a matter of fixing the activation time of the sense amplifier. Let's suppose Cbl and Cbl_bar are known, the two capacitances of BL and BL_bar. The activation time must be set to be sure that between V_BL and V_BL_bar there is a reasonable difference, e.g. 100 mV. To set the activation time we can use the following formula.

$$\Delta T = \frac{V \cdot C}{I}$$

Delta(T) is the time after the activation of the WL where we have to activate the sense amplifier.

4-TRANSISTORS SRAM



Again we have the bistable element and the pass transistors, but the inverter is implemented with a resistive load.

The benefit is that in this case we avoid two transistors that are substituted with two resistors. This allows to greatly reduce the area of the cell.

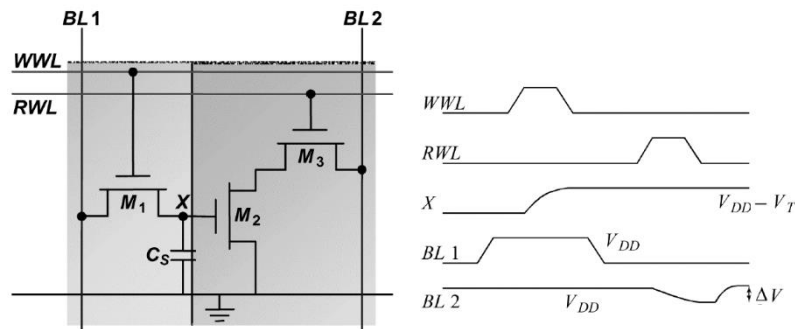
The resistance must be within minimum and maximum allowed values. Let's consider the cell is storing a 1. On the left side we have static power consumption because Q_bar is close to ground if R >> r_on of M2 on the left, so we have a current Vdd/R, and we have a static power consumption Vdd^2/R.

Let's suppose we want to have a power consumption smaller than 1mW. In this way we get the Rmin.

As for the maximum value of the resistance, we cannot just put an open circuit because if we switch off the transistor M2 on the right we have a leakage current which also flows in the resistance, and this might generate a huge voltage drop if the resistance is very large.

DRAM

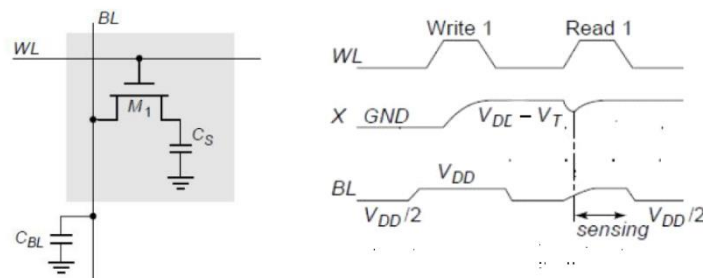
1-TRANSISTOR DRAM CELL



- ✓ No constraints on device ratios
- ✓ Reads are non-destructive
- ✓ Value stored at node X when writing a "1" = $V_{DD} - V_T$
- ✓ Inverting cell
- ✓ To improve speed, WWL can be boosted to $V_{DD} + V_T$

Write phase

WL is pulled up and BL is pulled to GND or Vdd depending on the value we want to store. If we force BL to 0V, the capacitance is discharged. If we want to write a 1, the nMOS is on as for storing a 0 and the Cs is charged to $V_{DD} - V_{tn}^*$. From now on let's neglect the body effect.



Write: C_S is charged or discharged by asserting WL and BL.
Read: Charge redistribution takes places between bit line and storage capacitance

$$\Delta V = V_{BL} - V_{PRE} = (V_X - V_{PRE}) \frac{C_S}{C_S + C_{BL}}$$

Voltage swing is small; typically lower than 100 mV.

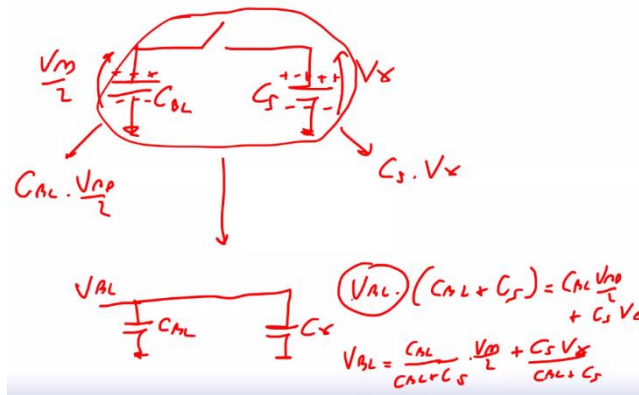
Read phase

We have the following steps:

1. Precharge of the bitline to $V_{DD}/2$.
2. $WL = 1$ and we put in contact the Cbl (1pF) and the storage capacitance C_S . So the switch (nMOS) is closed. Since we are connecting two capacitances, we have **charge sharing**. In the expression of the image above, $V_{pre} = V_{DD}/2$.

Let's call V_X the voltage across C_S . It can be either 0V or $V_{DD} - V_t$ depending on the stored value.

Charge sharing works as follow. Before the charge sharing occurs, on the left we have a charge $C_{bl} \cdot V_{dd}/2$, on the right $C_s \cdot V_x$. Then we have the switch closed and the voltage is shared and the same. Now the charge is $V_{bl} \cdot (C_{bl} + C_x)$ and it must be equal to the charge on the two capacitances before the activation of the WL. Our unknown is V_{bl} .



Instead of V_{bl} , let's write the voltage difference $\Delta(V) = V_{bl} - V_{DD}/2$. If there is a 0 stored in the cell I expect this voltage to decrease.

$$\frac{C_{AL}}{C_{AL} + C_S} \frac{V_{DD}}{2} - \frac{V_{DD}}{2} + \frac{C_S V_x}{C_{AL} + C_S}$$

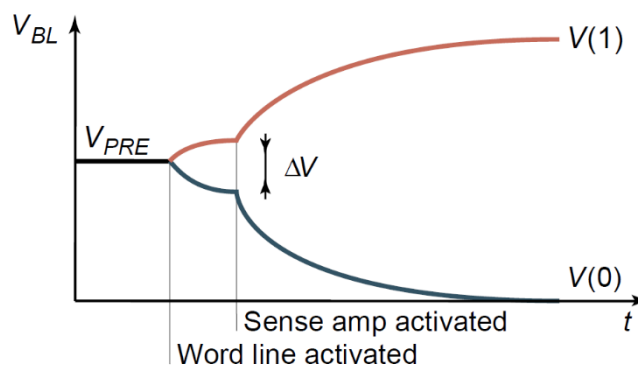
$$\frac{C_{AL} \frac{V_{DD}}{2} - C_{AL} \frac{V_{DD}}{2} - C_S \frac{V_{DD}}{2} + C_S V_x}{C_{AL} + C_S} = \frac{C_S V_x - C_S \frac{V_{DD}}{2}}{C_{AL} + C_S} = \frac{C_S V_x - C_S \frac{V_{DD}}{2}}{C_{AL} + C_S}$$

$V_{k=0} = \frac{C_S V_x - C_S \frac{V_{DD}}{2}}{C_{AL} + C_S} = -30 \text{ mV}$
 $V_{k=1} = \frac{C_S V_x - C_S \frac{V_{DD}}{2}}{C_{AL} + C_S} = +25 \text{ mV}$

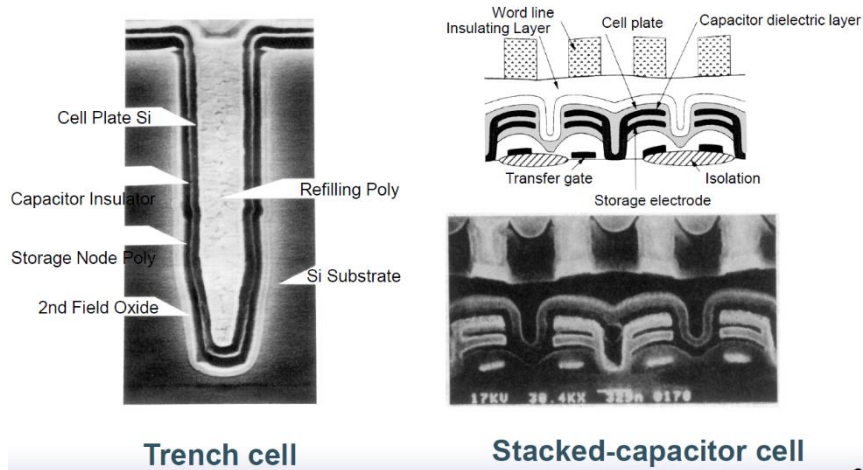
So with respect to the precharge value the voltage difference we can sense is very limited, from -30mV to +25mV. Aside from this, which makes the use of the sense amplifier mandatory, there is another problem.

In C_s we store either a 0 or a 1, and during the read phase the voltage across it is no more $V_{DD} - V_t$ or 0, but close to $V_{DD}/2$. This means that we have destroyed the content of the cell, so we need to refresh the content of the cell, since the read phase as it is destructive → every time we read the memory we have to rewrite its content, so we need a particular kind of sense amplifier that is able to do so.

Sense amplifier operation



We need to precharge at $V_{DD}/2 = V_{pre}$, and then we have a very small swing, a delta of 56mV. However, at a certain time stamp we turn on the sense amplifier (which is a static latch for the DRAM) which refreshes the content of the cell and brings V_{BL} to $V_{DD} - V_t$ or GND. So it has a double role with respect to the sense amplifier of the SRAM.



COMPARISON

SRAM	DRAM
Stores data till the power is supplied	Stores data only for few milliseconds even when power is supplied
Uses an array of 6 transistors for each memory cell	Uses a single transistor and capacitor for each memory cell
Does not refreshes the memory Cell	Needs to refresh the memory cell after each reading of the capacitor
Data access is faster	Data access is slower
Consume more power	Consume less power
Low density/less memory per chip	High density/more memory per chip
Cost per bit is high	Cost per bit is low

In DRAM, since the switch is implemented with a transistor, and we store a logical 1, we have a leakage current that slowly destroys the content of the memory. So DRAM requires also a refresh, once in a while we need to read all the cells and refresh their content, otherwise the content is lost. The leakage current is the one of the junction diode or the subthreshold current.

The refresh is not required in SRAM memory. Due to the refresh lack, the data access is faster in the SRAM, typically in the order of 10ns, while in DRAM it's 50-100ns.

Moreover, SRAM can waste more power because it can reach higher speeds, but at the same speed the DRAM is more power hungry because we need to refresh the whole array.

PERIPHERY

- Row/column decoders
- Sense amplifiers
- I/O buffers
- Control/timing circuits

In case of DRAM is mandatory to have the sense amplifier between the column decoder and the cells, and we must have one per each bit line, it's mandatory. In case of SRAM there is no reason to have a sense amplifier per each BL, but only for each bit of the word we want to read, regardless the amount of words we have in each WL. This is also a reason why the DRAM is more power hungry.

ROW DECODER

It has N bits in input and outputs 2^N bits. Let's start from the row decoder for the NOR ROM and SRAM. We need a circuit with N bits in input and 2^N outputs, and the outputs are so that only one output is a logical 1 at a time, all the remaining are 0. In these two architectures, only one WL is on at a time.

Let's consider the case of N = 10. Each AND gate is fed by 10 bits (NOR = AND). So we have all complement inputs at the input of the AND. Only when all the inputs are 0 the WL0 goes to 1. For N = 10 we have $1024 = 2^{10}$ WLs.

The last AND gate will correspond to all the inputs 'true': $WL_{1023} = A_0 * A_1 * A_2 * \dots * A_9$.

Collection of 2^M complex logic gates Organized in regular and dense fashion

AND Decoder

$$WL_0 = \bar{A}_0 \bar{A}_1 \bar{A}_2 \bar{A}_3 \bar{A}_4 \bar{A}_5 \bar{A}_6 \bar{A}_7 \bar{A}_8 \bar{A}_9$$

$$WL_{511} = \bar{A}_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9$$

NOR Decoder

$$WL_0 = \overline{A_0 + A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8 + A_9}$$

$$WL_{511} = \overline{A_0 + \bar{A}_1 + \bar{A}_2 + \bar{A}_3 + \bar{A}_4 + \bar{A}_5 + \bar{A}_6 + \bar{A}_7 + \bar{A}_8 + \bar{A}_9}$$

Of course, in FC-CMOS logic the AND is difficult to be implemented, so we can substitute the AND with the NOR, which has almost the same behaviour of the AND. So we apply the De Morgan theorem putting a double bar on the AND functions and we can implement the row decoder using NOR gates.

NAND ROM

In this architecture transistors are in series, so it requires all the WL = 1 except for the one we want to read. So the AND decoder can be substituted with a OR decoder, that translates to NAND in FC-CMOS logic implementation.

In this case (OR decoder), $WL_0 = A_0 + A_1 + \dots + A_9$. $WL_{1023} = \bar{A}_0 + \bar{A}_1 + \dots + \bar{A}_9$.

Translating into a NAND decoder we have $WL_{1023} = (\bar{A}_0 * \bar{A}_1 * \dots * \bar{A}_9)$

However, we have some issues. In a NAND decoder, the main issue is that we have a gate with a lot of transistors in series, so the delay increases quadratically with the number of transistors. And this is a problem because the access time to the memory, once the address to row and column are given, is made of 3 main contributions: row decoder propagation delay (some time for the WL to be pulled up), cell access delay, delay of the sense amplifier activation (negligible) and delay of the column decoder.

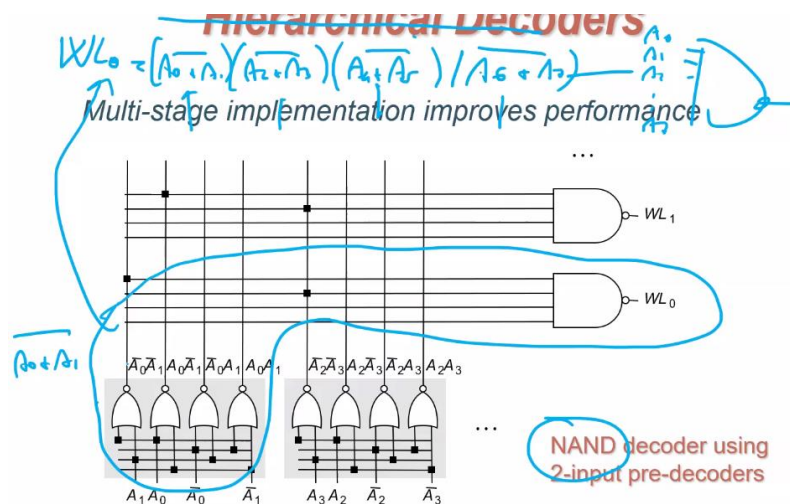
In this case, per each WL we have to implement a NOR or NAND gate with a lot of transistors, and the delay increases.

NAND decoder – hierarchical decoders

The idea is to split the decoder in two parts: a pre-decoder and a real decoder. Let's consider a case with 8 bits.

We know that $WL_0 = (\bar{A}_0 * \bar{A}_1 * \dots * \bar{A}_7)$, and to reduce the complexity of this NAND gate we can split it using the associative property of the NAND function, coupling $A_0 * A_1$, $A_2 * A_3$, $A_4 * A_5$ and $A_6 * A_7$. Then for each of these terms let's apply the De Morgan theorem, applying a couple of bars per each couple.

The inner terms are NOR gates, whose implementation is the one encircled.



So the initial NAND gate with a fan-in of 8 has now a fan-in of 4. The grey box is called **pre-decoder** because it can be used also by other NAND gates that need the same output as inputs.

So we reduce the complexity of the decoders, we are using gates with smaller fan-in, and we are also reducing the number of transistors because **the pre-decoder can be shared among different gates**.

Area sparing

Let's consider a NAND decoder. In classical FC-CMOS we would need $2^8 = 256$ NAND gates with a fan-in of 8, so made with 16 transistors. So we would require $256 \cdot 16 = 4096$ transistors. If we use a hierarchical decoder, in any case we have 256 NAND gates but with a smaller fan-in, so made of 8 transistors. Moreover, we have $4 \cdot 4$ NOR gates, each made with 4 transistors, so overall 64 transistors for the pre-decoder. In the end we have $256 \cdot 8 + 64 = 2116$ transistors, so we have spared almost 50% of the transistors.

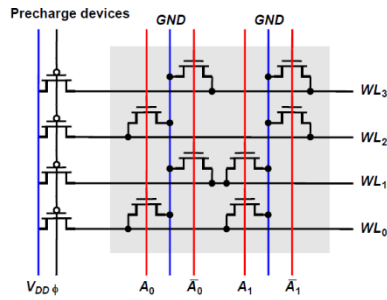
Hence there is a huge improvement in terms both of area and delay, because of the smaller fan-in.

In the other implementation, that is the NOR decoder, we have NOR gates and NAND pre-decoders.

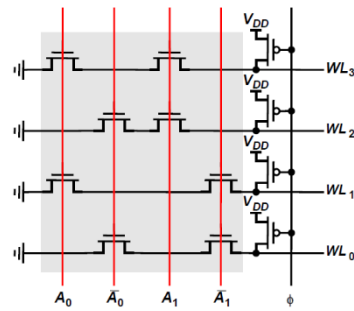
DYNAMIC DECODERS

To further spare transistors, at transistors level instead of implementing FC-CMOS gate we can implement pseudo-nMOS decoder with a PU device with a pMOS and PDNs. It's an improvement in terms of transistors, we are removing the cumbersome PUN using only pMOS which are dynamic transistors.

A gate like this is not properly a pseudo-nMOS gate, because the pMOS should be always with the gate connected to GND and very weak in a real pseudo-nMOS. In reality the pMOS is connected to a signal that is 0 at the beginning, so we precharge the node as in SRAM memories, and then we go to 1 to turn the pMOS off and leave a floating capacitance charged to Vdd. If at least on nMOS is to 1, the capacitance is discharged.



2-input NOR decoder



2-input NAND decoder

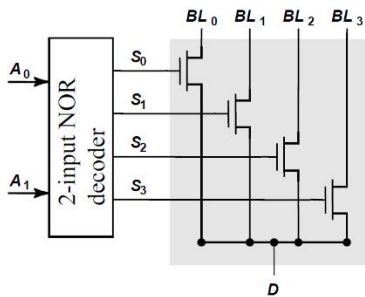
COLUMN DECODER

It is not properly a decoder, but a multiplexer.

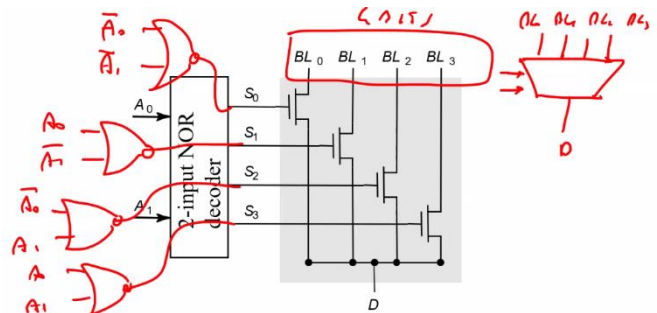
4-INPUT PASS-TRANSISTOR COLUMN DECODER

Let's suppose we have 4 bits (4 BLs) and we want to output only 1 bit (D). We have to implement a multiplexer, so we need 2 select signals.

To implement this multiplexer we can use e.g. a NOR decoder. So the multiplexer is made only of 4 transistors, and only one at a time is turned on for a particular combination of the select signal bits A0 and A1. So only one of the 4 select signals S0, S1 etc. is to 1 depending on the input bits.



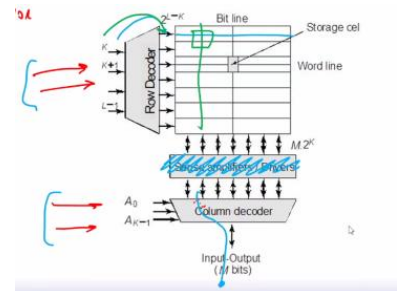
Advantages: speed (t_{pd} does not add to overall memory access time)
 Only one extra transistor in signal path
 Disadvantage: Large transistor count



Advantages: speed (t_{pd} does not add to overall memory access time)
 Only one extra transistor in signal path
 Disadvantage: Large transistor count

Basically we are implemented a NOR decoder. This implementation requires a lot of transistors, because we have 4 NOR gates and 4 pass transistors (PT), so overall we have $4 \cdot 4 + 4$ transistors, so overall 20 transistors.

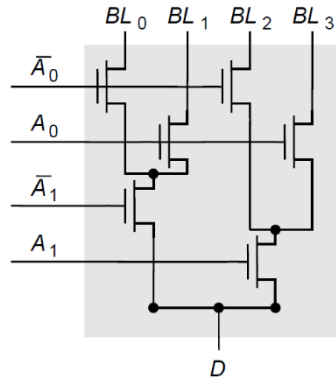
The advantage is that the speed at which it can operate is higher. In fact, in the worst case path at the beginning at $t = 0$ we have two addresses given at the input to the memory. Then the row decoder processes the input data and pulls up a word line. Then once the WL is selected, all the cells connected to the WL write their content on the BL, and then the signal travels along the BL, arrives at the column decoder to reach the output node. The worst case path is the one highlighted in green. So we have the propagation delay of the row decoder, the delay of the cell (that is the time required to discharge the bit lines), and the delay of the column decoder.



4-to-1 TREE BASED COLUMN DECODER

We can implement the column decoder with a **tree multiplexer**. The codes of the column decoder are directly fed to the transistors, and only one path is on at a time.

In this implementation we have only 6 transistors, and this is a huge difference in terms of area.



Number of devices drastically reduced
 Delay increases quadratically with # of sections; prohibitive for large decoders
 Solutions: buffers
 progressive sizing
 combination of tree and pass transistor approaches

In this implementation we have two transistors in series (8 if we consider 8 inputs), and the number of transistors in series is equal to the number of inputs. In the previous implementation we have only 1 transistor in the series, and this is beneficial.

So in this second implementation we have less transistors but problems in terms of propagation delay.

SENSE AMPLIFIER

Whatever the memory we implement, e.g. in case of SRAM we have a bistable element and a couple of PT, the signal swing at the BL node is very limited. This because of the big capacitance Cbl and minimum size transistors, so large equivalent resistance and small current available.

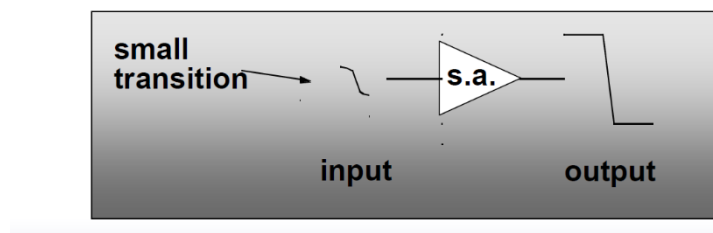
The sense amplifier is enabled after a fixed amount of time and the difference is amplified.

We need to assess the delay t_p with which the sense amplifier comes into action.

$$t_p = \frac{C \cdot \Delta V}{I_{av}}$$

Annotations: ΔV is labeled "make ΔV as small as possible"; I_{av} is labeled "small"; t_p is labeled "large".

Idea: use of a sense amplifier



Delta(V) should be 2.5V, but if we are not willing to wait, we can set a smaller delta(V) e.g. 50mV and we activate the sense amplifier.

So to cope with a small propagation delay we need to reduce as much as possible the voltage difference between the two BL and BL_bar lines.

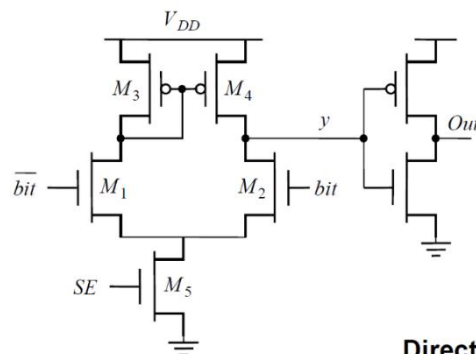
So instead of waiting for the signals BL and BL_bar to reach the full swing, we amplify them after a while to reach the full swing, so we need a sense amplifier.

DIFFERENTIAL SENSE AMPLIFIER FOR SRAMs

We have a differential stage followed by a single ended amplifier, that is an inverter. The differential amplifiers takes the signals BL and BL_bar and amplifies the difference. To be honest, M5 is not a current generator but a switch, driven by the signal SE (sense-amplifier enable).

M5 is 0 before the read phase starts, and then logical 1 when we want to amplify the voltage difference between the two lines. So it is a switch, and not a current generator, because we don't want to increase the CMRR, but we just want to amplify the voltage difference and output a full swing signal. We are not interested in implementing a good analog amplifier.

We just need something to amplify the difference and something other to have a full-swing signal at the output.



Directly applicable to SRAMs

Why do we need to activate the sense amplifier at around 50mV or 100mV and not before? Because we have to cope with noise, which could be so large that the output is random due to noise.

We have also other non-idealities in a differential amplifier, like the offset; every real circuit has an offset because of mismatches between transistors. So the input is imbalanced and we need to win it to amplify the correct signal. Bit and bit_bar in a SRAM configuration are precharged high.

If we precharge the inputs to $V_{dd}/2$, once we connect the storage element to the bitline capacitance C_{bl} , the voltage on the inputs slightly increases or decreases with respect to $V_{dd}/2$.

For these reasons we have to wait that the difference between BL and BL_bar is large enough. The good thing is that this analysis can be done a priori during the simulation phase of the circuit, and so we can know a priori the activation time of the sense amplifier.

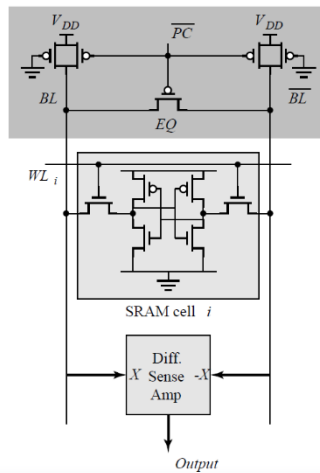
Why not designing an amplifier with the pMOS input pair? It is not a matter of driving capabilities of the transistor, but the fact that both inputs are close to V_{dd} , and so the pMOS are off when the signal is high. Since the SRAM works with a precharge to V_{dd} , we need to use a sense amplifier made of nMOS transistors.

Differential sensing

We have the whole circuitry connected to BL and BL_bar. We have the sense amplifier, the cell and at the top the precharge circuitry, driven by an active-low signal. We recognize 2 pMOS attached to precharge signal (PC) which, when 1, turns on the pMOS and BL and BL_bar are pre-charged to V_{dd} .

Then we have also other auxiliary transistors, the EQ, called **equalization transistor**. It is like connecting the two BLs by means of a resistance, to equalize the voltage between the two BLs. In fact, during the charge phase, it may happen that BL is already at Vdd, and BL_bar is charging to Vdd, because during a readout one of the two BL or BL_bar can reach 0.

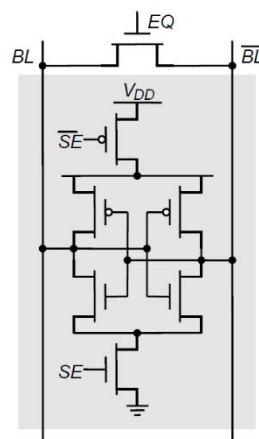
BL_bar charges exponentially and reaches Vdd at an infinite time, so we don't reach exactly Vdd. Since we are coping with a small difference between BL and BL_bar and BL_bar might not be Vdd exactly, it's important to have the transistor in the middle to equalize the two voltages.



Furthermore, there are two pMOS transistors always attached to GND. As a remark, the pMOS connected to the PC signal are large transistors, not minimum size, because we need a fast PU. The pMOS connected always to GND are minimum size instead, so they are not responsible for the PU. However, we need them and always on because during the read phase one of the two BL remains at Vdd, the other one has to decrease toward 0. If the large pMOS are off, the two capacitances are floating. So the capacitance is subjected to a leakage current, and it can be aggressed by an aggressor (noise).

Because of this we implement the small pMOS transistors, also called **bleeders**. They serve the purpose of reducing the impedance of the wires (10 kOhm resistance), they are a weak connection to Vdd. In this way the capacitance is immune to noise and the leakage issue is no longer an issue because there is a connection to Vdd.

SENSE AMPLIFIER FOR DRAM – LATCH BASED SENSE AMPLIFIER



Initialized in its meta-stable point with EQ
 Once adequate voltage gap created, sense amp enabled with SE
 Positive feedback quickly forces output to a stable operating point.

If we forget for a moment the EQ transistor we can recognize a differential structure with BL and BL_bar, so we have created a differential architecture with two bit lines. In the middle we have a sense amplifier made of a latch, a bistable element. So **the sense amplifier for DRAM is a bistable element** plus a couple

of transistors on top or bottom to enable it and enable the amplification, driven by two complementary signals, so that we can activate both of them for SE = 1. If SE = 0, the bistable element is not active and not amplifying, because we are not powering it and it's floating.

How can a bistable element operate as an amplifier?

A bistable element has 3 stable points, out of which one is metastable, corresponding to $V_q = V_{dd}/2$ and $V_{q_bar} = V_{dd}/2$ considering inverters with $V_m = V_{dd}/2$ ((3.5) – (1) inverter). However, thermal noise is enough to move away from this bias point.

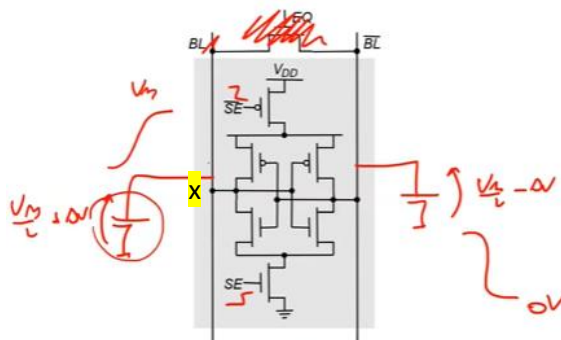
Once the inverter is biased at V_m , the inverter is not a digital element, but it works as an amplifier, it's the combination of two common sources. So it's enough little imbalance between Q and Q_bar and the inverters generate a signal either to Vdd or GND. Basically we are using the positive loop to amplify the voltage.

Let's consider a situation with $V_{dd}/2 + \delta(V)$ on BL and $V_{dd}/2 - \delta(V)$ on BL_bar. At a certain point we turn on the bistable element with SE = 1.

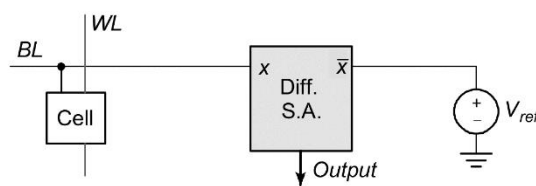
NB: BL and BL_bar are floating, there is no low impedance path towards Vdd or GND.

When the bistable element is powered, it amplifies the voltage difference on BL and BL_bar, and the signal on BL goes to Vdd, the other to GND.

If we suppose that the node x is connected to a DRAM cell, we have also refreshed the content of the cell (storage capacitor).



This is a very good amplifier solution with the input node corresponding to the output node but we have to cope with the fact that the DRAM cell is single-ended, so we need to create a differential structure to use this amplifier.



How to make a good V_{ref} ?

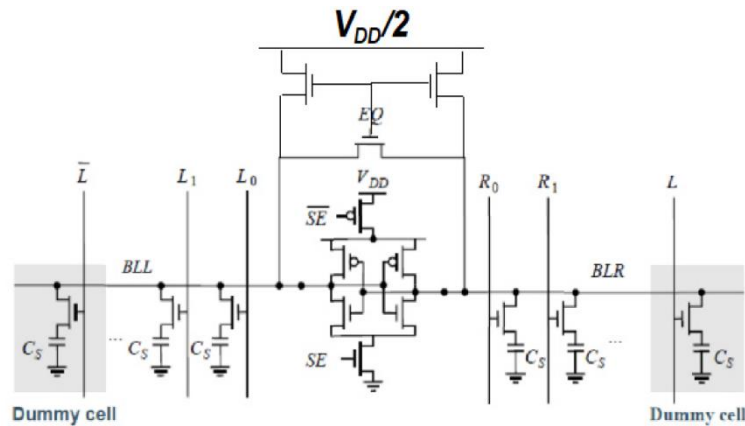
The simplest way to create a differential architecture is to double the memory. On the left we put the true array, and on the right the complement array (BL_bar instead of Vref, with a complement value).

So on one side we have an increase of $V_{dd}/2 + \delta(V)$, on the other $V_{dd}/2 - \delta(V)$.

The main drawback of this solution is that we are doubling the area, we are storing a bit using two transistors and two capacitors, so we are halvening the integration density. Since this drawback is too critical, we need another solution.

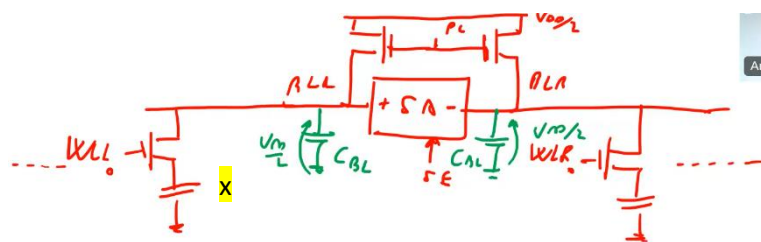
OPEN BITLINE ARCHITECTURE WITH DUMMY CELLS

It's the most adopted architecture. Dummy cells are the ones in the gray boxes, and in the middle we can recognize the sense amplifier. On the top we have the precharge circuitry, done at $V_{DD}/2$ with nMOS transistors, so we can use them. In the middle we have an equalization transistor that works as a resistance.



If we neglect the dummy cells, we have the bit line left cell (BLL0), and on the right BLR. We have to imagine that these are a lot of bit lines stacked, so this structure is replicated vertically. Lines that run vertically are the WL, called $L_0, L_1, R_0, R_1, \dots$. So we have split the array in two halves, and in the middle there is the sense amplifier and the equalization circuit. But they are not complementary, we have bundle of cells on the left and on the right, but not complementary cells. Dummy cells are not mandatory for the correct working for the gate, but useful. In an ideal environment they are useless, but since we work in an environment with a lot of noise, they are needed.

Let's rewrite the circuit. SA is the sense amplifier, and on top we have the precharge circuit at $V_{DD}/2$. To simplify the analysis, let's consider only two cells, one on the left and one on the right. The one on the left is connected to WLL_0 and the one on the right to WLR_0 .



The first step is to precharge to $V_{DD}/2$ BLL and BLR, with all the $WL = 0$. So we precharge just the parasitic capacitances C_{bl} , that are in the order of pF, at $V_{DD}/2$, with all the cells disconnected from the BLs, otherwise we overwrite all the cells to $V_{DD}/2$.

The second step is the activation of the WL of interest, one at a time, so e.g. $WLL_0 = 1$. The BL that changes in terms of voltage is BLL, since the other is not connected because the SA behaves as an open circuit since it's not active. So the activation of WLL_0 makes the voltage on capacitance x change. The voltage can increase or decrease depending on the content of the cell. Let's suppose that is discharged to $V_{DD} - V_{t^*}$, affected by body effect.

Once we raise WLL_0 to BLL we have charge sharing between the two capacitances and voltage on BLL increases to $V_{DD}/2 + \Delta(V)$.

The third step is the activation of the sense amplifier, at which input we have $V_{dd}/2 + \Delta V$ on the left and $V_{dd}/2$ on the right. The sense amplifier amplifies the voltage difference, the voltage on the left goes to V_{dd} and the one on the right to $0V$. During this phase, $WLL0$ remains at logical 1, so that the amplification allows the voltage to reach V_{dd} . In this way we also refresh the content of the cell.

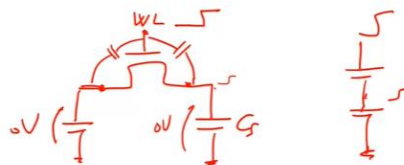
During phase 2, the voltage across the storage capacitance C_x is no more $V_{dd} - V_{t^*}$, but it goes to $V_{dd}/2 + \Delta V$ because of charge sharing, we have the same voltage on the BL capacitance and on the storage capacitance. So it's like if we have destroyed the content of the cell.

However, if $WLL0$ remains to 1 during the third phase, both the voltages on BLL and C_x are brought to V_{dd} . However, in reality the capacitance C_x is not charged to V_{dd} . Actually, it's $V_{dd} - V_{t^*}$ because we have the nMOS transistor. The nMOS turns off even if $WLL0$ is a logical 1, because the V_{gs} goes below V_t . So during the third phase we have **amplification** and **regeneration**.

If we want to read a cell on the right we have to repeat the same procedure, but instead of $WLL0$ we have to consider $WLR0$.

This is however an ideal behaviour, considering ideal transistors that act as switches. Let's consider that the switches are real circuits and that, once we have a real switch, we have also parasitic capacitances.

Let's consider a real switch with the storage capacitance C_s (C_x). Once we turn to 1 the WL signal, we have **clock feedthrough**. We have in fact 2 capacitances in series and, due to the voltage divider, the voltage in the middle increases.



There is also **charge sharing**. In fact, every time we have a transistor we have a couple of problems that occur at the same time: feedthrough due to the parasitic capacitances and charge sharing.

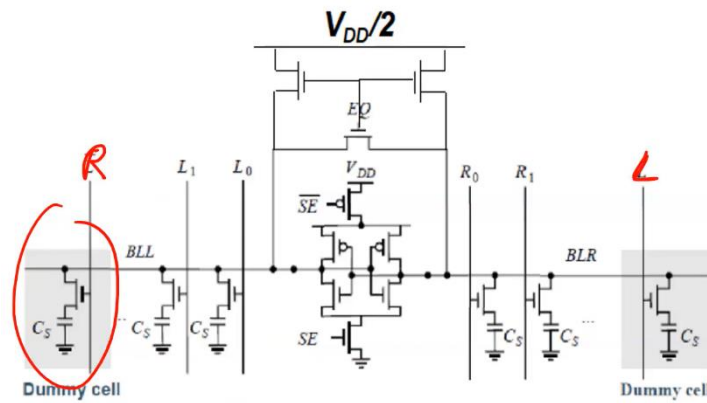
Once we turn on the device, we have to create a layer of electrons to activate the transistor, and these electrons come from the 'surrounding world', that are the n+ regions, so from the parasitic capacitances.

These two phenomena are correlated, and the final effect is that the voltage across the two terminals increases. The most important point is that the two junctions at source and drain are floating nodes at $0V$ before turning on the transistor. The effect of the feedthrough and charge sharing is the same, the voltage at the source and drain increases.

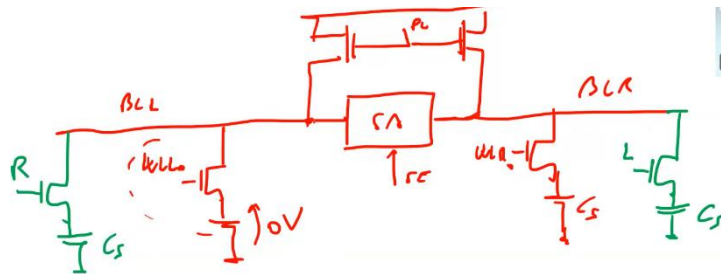
If we had an ideal switch, the voltage was 0 at the beginning on both the C_s capacitances and it remains that, it cannot change. These problems happen every time we have transistors in a circuit, like in the S&H circuit and in switched capacitors.

Moreover, in our case we also have to deal with small signals, drops of $31mV$ and increase of $25mV$. So these two mechanisms can alter the voltage on the bit lines. This is the reason why dummy cells are needed.

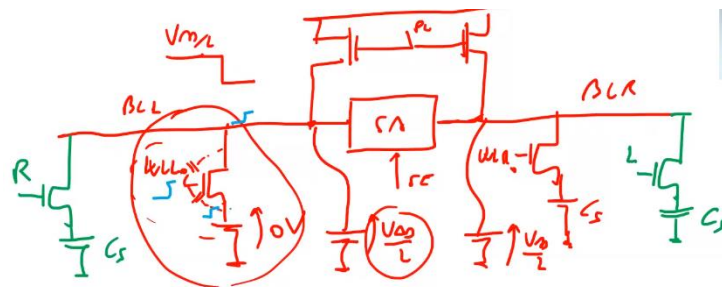
We have a dummy cell for each bit line. This is not a great overhead, because the increase of area is negligible. In the image, the dummy cell on the left is connected to the signal L_bar , that we rename R , the one on the right to L .



Let's see the behaviour in case of digital noise, that is charge sharing, capacitive coupling, feedthrough, and how the dummy cells are able to make the architecture immune to digital noise. Again, let's redesign the circuit. Let's consider only one cell, with WLL0 and WLR0. Let's suppose a zero is stored on the left cell and I want to read it. In green we have the dummy cell, connected to another WL signal that is R or L. It is completely equivalent to another cell, so we have a storage capacitance.



Let's do the same steps done previously and see what happens in presence of real signals. As a first step we precharge, so the parasitic capacitances BLL and BLR are charged to $V_{DD}/2$. The second step is the activation of the WL, so we pull up WLL0 to V_{DD} , the others remain at 0. If we don't use the dummy cells, we have the parasitic capacitances, a voltage $V_{DD}/2$ on BLL which drops to $V_{DD}/2$ minus something. But since there are the parasitic elements, we have the effect of the feedthrough (blue), which means a voltage rise on BLL which is opposite to the drop on the BLL, which is the one we want to read.



If the capacitive coupling is large enough, the digital noise can cancel the real signal we want to read, this is the real problem.

To make the memory immune to this kind of noise we use the dummy cells. During the precharge phase, also the signal R and L are pulled up to 1, so that we precharge also the storage capacitance inside the dummy cell, not only we precharge BLL and BLR.

Once we precharged BLL and BLR, the voltage across the dummy cells is $V_{DD}/2$, not a logical 1 or 0.

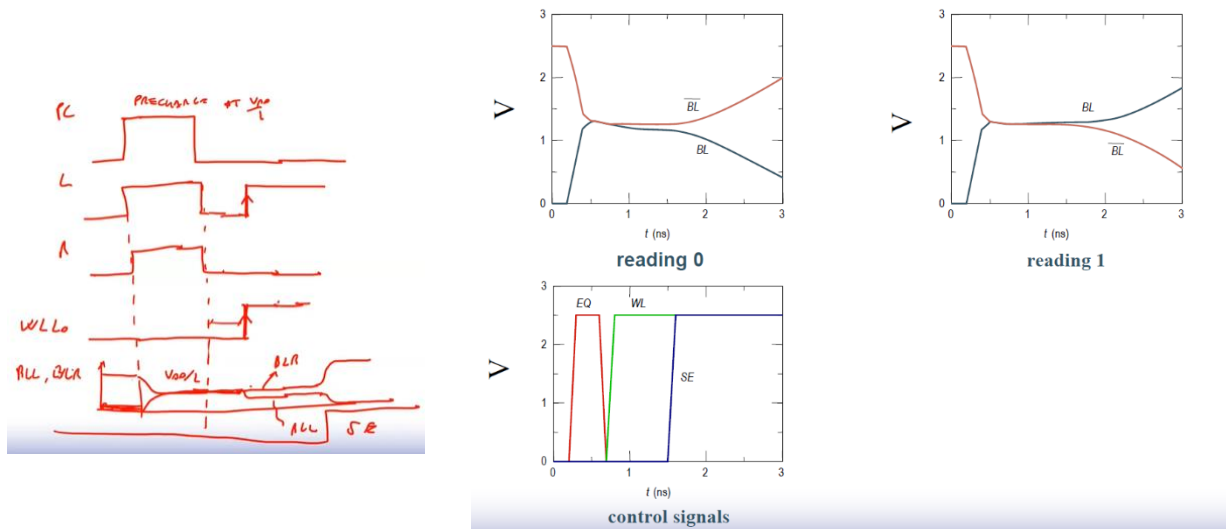
Then we need to introduce an intermediate step. After the precharge of BLL and BLR and of the dummy cell, PC signal is pulled down, so we have written the cell but then deactivated it.

Now we want to read WLL0. It is pulled up to 1, and at the same time we pull up the dummy word line on the opposite side, that in this case is L.

The third step is the activation of the sense amplifier.

Graphs of all the waveforms

Main signals involved are PC, L and R, and the last two follow PC at the beginning. Then we want to read WLL0, which at the beginning is 0 and then we rise it. Once we want to read WLL0, we activate also the L signal. As for BLL and BLLR, one starts from 0 and one 1 (let's suppose), and both are both to $V_{dd}/2$ (equalized value) during the precharge phase. Then they remain floating; to be honest, the activation of the WLL0 happens just after the PC is ended, so time has been exaggerated on the plots.



Since in the left cell there was a 0, from an ideal standpoint BLL decreases a little and BLR remains at $V_{dd}/2$, constant. Then somewhere I have to activate the sense amplifier to amplify the difference. Once I do this, BLR goes to V_{dd} and BLL to 0V. Then we have the refresh of the memory.

However, this is an ideal behaviour without considering noise. The activation of the dummy cell is able to cancel the noise because we are creating noise on both sides (left and right), **we create a common mode noise thanks to the dummy cell**, and the sense amplifier is capable of reading a differential signal.

So dummy cells are not needed if the switches were ideal.

x

x