EECS 16B   Designing Information Devices and Systems II
Summer 2020   Note 0

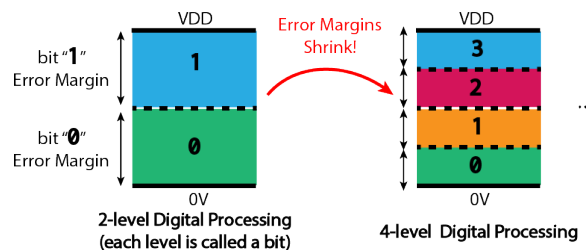# 1   Digital Information Processing

Electrical circuits manipulate voltages (V) and currents (I) in order to:

   (a) Process information (Arithmetics, Storage, Data Transfer, etc.)

   (b) Transduce energy to and from their environment (Sense, Actuate, Harvest energy, etc.)

Because the universe appears **analog** (continuous in time and quantity) at our scale, we usually use **analog** circuits to interface with anything that interfaces with the outside world (i.e. sensors, actuators, antennas, etc.). We can also use analog circuits to perform a function such as divider circuits you saw in EE16A. However, processing information in the analog world is extremely challenging and analog information processing is shrinking! **Why?** You already experienced all the errors, variations and fluctuations exist in the real world in the EE16a labs. There are many sources for these errors in practice; for example, any two metal wires close enough together form a capacitor and their voltages affect each other (this effect is known as "Cross-talk")! With billions of transistors on a centimeter-scale chip areas and wires with nanometer spacings, this effect happens very frequently and causes errors. This is one of multiple noise and error sources limiting the analog information processing performance and complexity. In this lecture, we are interested to see how we can build processors robust to noise and scalable to perform complex operations.

In order to make the processing immune to the errors, we should leave some margin between the voltages we plan to use by segmenting the voltage domain. These margins are called **Noise Margins**. Let's assume a circuit that contains a VDD volt independent voltage supply and a bunch of passive components. We know that all voltage nodes in the steady state are between $0V$ and VDD since the components just form some sort of dividers! For analog processing any voltage from $0V$ to VDD can show up at the input/output ports in the under the error-free conditions. Now let's see how a "2-level" digital processing format (binary), where only $0V$ and VDD are meant to show up at the ports under the error-free conditions. These kinda circuits are called **digital**, where voltages at the ports should accept only certain discontinuous values under error free conditions. Although, you may heard already about binary digital processing, the number of level can be any arbitrary number beside 2 as well. Next figure shows noise margins for 2- and 4-level digital processing. *Notice binary digital processing provides the largest noise margins and makes the processing most immune to noise and error perturbations compared to the higher order processing formats. This is why most of digital systems are binary.*

2-level Digital Processing
(each level is called a bit)

4-level Digital Processing

Bits and Logic Gates

To perform digital information processing, we have to give meaning to these voltages to be able to do useful processing and operations. Assume we assign $0V$ to a state namely bit zero (**0**) and VDD to the other state bit one (**1**). Since we have only two states (**0/1**), we use number representation in base 2, where each digit can have only two states similarly. For example:
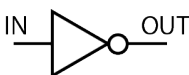
$$3 = (11)_2 \quad 16 = (10000)_2 \quad 1234 = (10011010010)_2$$

Thus, assuming the data (for instance captured from a sensor) is in a fixed unit and represented in decimal digits, we can map it in a format with two possible states for each digit similar to the voltages in our digital circuit voltages. Now let's see how we can process these bits. One of the most useful arithmetics is the summation. These binary numbers can be added together as follows:

Carries:
$$C_2 \ C_1 \ C_0$$
$$a_2 \ a_1 \ a_0$$
$$+ \ b_2 \ b_1 \ b_0$$

Example:
$$1 \ 1 \ 1$$
$$1 \ 0 \ 1$$
$$+ \ 0 \ 1 \ 1$$

Sum:
$$S_3 \ S_2 \ S_1 \ S_0$$
$$1 \ 0 \ 0 \ 0$$

Binary Summation: Similar to decimal addition, you start with adding columns from the right side and when the summation result at each step overflows (in this case goes over 1 instead of 9 in the decimal case), the resultant bit will be the addition result reminder and pass a carry bit to the next column. Once all the columns are summed up, the result is the summation of two binary numbers.

How can we build a circuit that takes these 3-bit binary numbers in this example and produce the summation result at the output? In order to find the answer to this question, you need to learn about the basic bit-wise binary operators (a.k.a **logic gates**) first. There are 3 basic logic gates shown and defined below:

Operation:	NOT	AND	OR

Symbol:

IN ⊳o OUT
inverter

A ⊐ OUT
B

A ⊐ OUT
B

Truth Table:
(Outcome for every possible combination of inputs)

IN	OUT
0	1
1	0

A	B	OUT
0	0	0
0	1	0
1	0	0
1	1	1

A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	1

Notation:

$$\text{OUT} = \overline{\text{IN}}$$

$$\text{OUT} = A \cdot B$$

$$\text{OUT} = A + B$$

It turns out that any binary operator can be implemented by combining these 3 logic gates. For instance to implement the summation output described above, we can first breakdown the 4-bit output into the single bits and formulate each one in terms of the input bits. For example:

a_0	b_0	S_0	C_0
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

=>
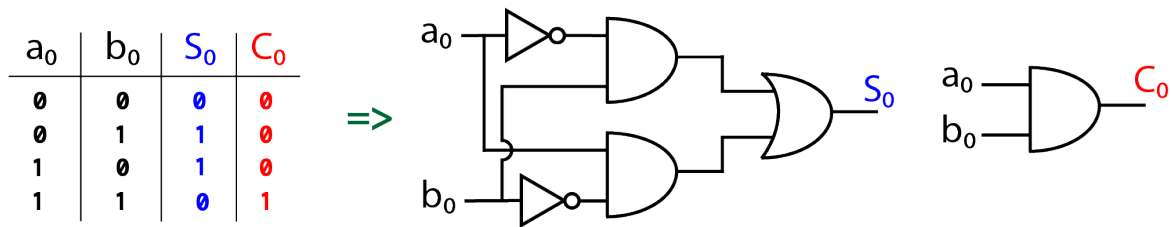
In this course we are not focusing on how to implement arbitrary logic operations using the 3 basic logic gates (covered in 61C course). Instead, we will go over the details for building these basic 3 logic gates and see how they are implemented in the physical world. Surprisingly, knowing these very basics can help us to understand what's limiting the speed of state-of-the-art electronics (laptops, smart-phones, etc.) and why playing games with your cell phone cause them to ran out of battery quickly!

Starting from the first logic gate (NOT), an <u>inverter</u>, one idea to build such a block is to use an opamp:

inverter (INV)
V_{in}/V_{out} relation

implementation
using OpAmp!
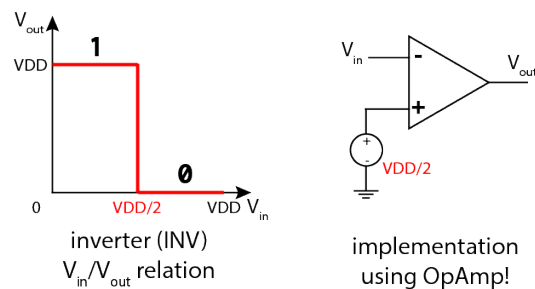
Although this works fine, we need around 10-20 transistors (explained later) to build a single opamp which is not affordable in terms of area and energy in practice! You will see how to build this gate using only two transistors!

To do so, we first see how switches can be used to build these gates and later discuss how the switches are built in the real world using devices called transistors.

2   Switch based Logic

You already used switches in the 16A course for instance in the charge sharing capacitor circuits for touch-screen sensing. Although, we represented them as a 2-terminal device previously, there should be another terminal to electrically control the state of the switch as well! Let's call this control terminal the Gate and whenever the Gate voltage is high enough the switch is closed and while the Gate voltage is low enough the switch is open. In fact this control terminal existed in 16A labs but they were set by the Launch-Pad or Arduino kits. Although, this definition sounds enough to know the state of switch by knowing the Gate voltage, there's still something missing! First of all, the voltage of a node should be referenced to another node and also we need to define a threshold voltage (V_{th}) to know what we meant by "high/low voltages". Thus the complete switch symbol and definition can be explained as:
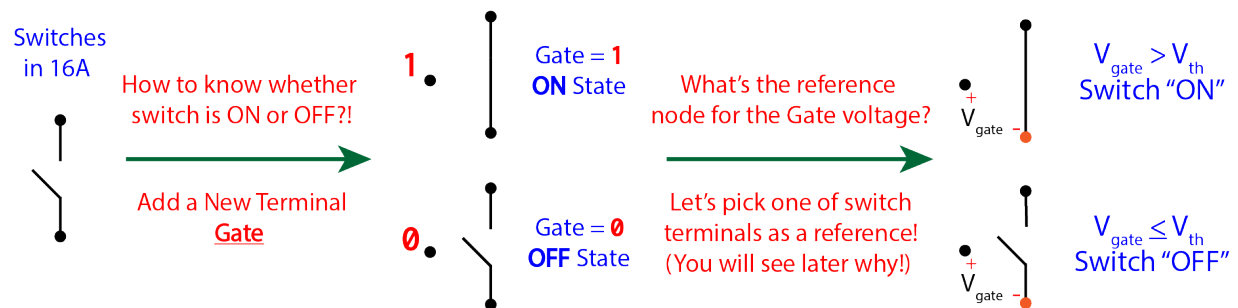

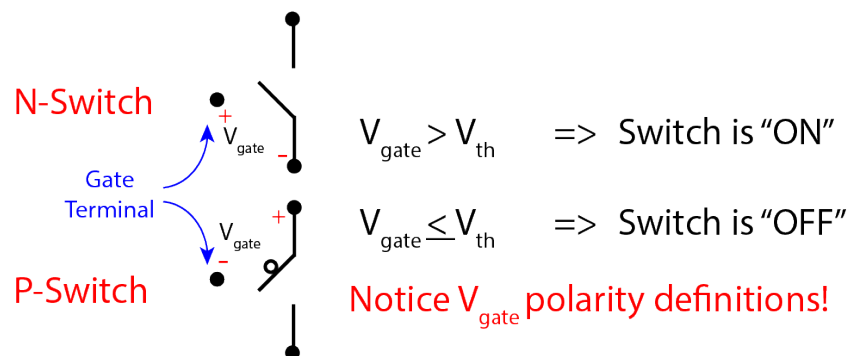Now, let's see how we can build an inverter by just using this switch and a resistor:
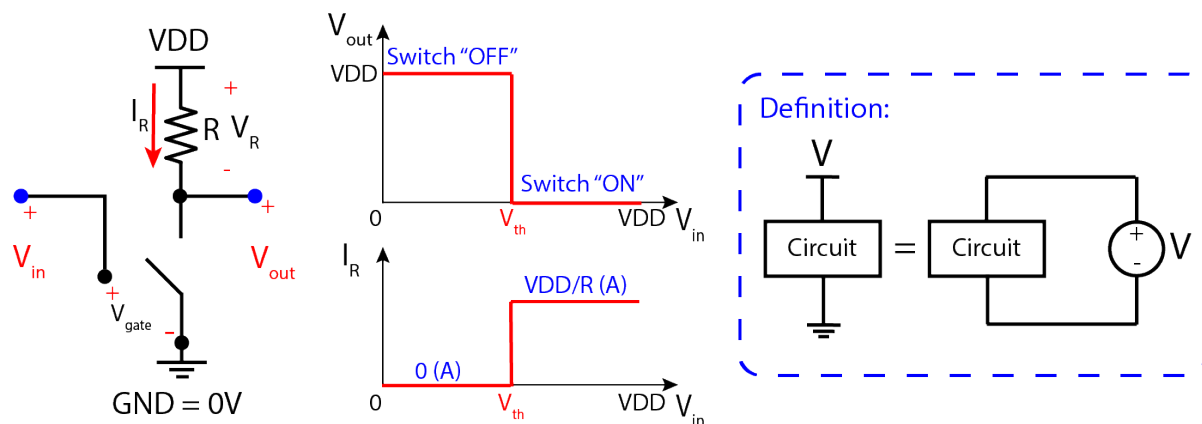
Analysis: Notice here $V_{gate} = V_{in}$. Hence, if $V_{in} < V_{th}$, switch is OFF and we can write KVL equation as: $V_{DD} = V_{out} + V_R$, and using ohm's law, we will have $V_{DD} = V_{out} + I_R R$. Since $I_R = 0$, thus $V_{out} = V_{DD}$. For the other case, where $V_{in} > V_{th}$, switch is ON and consequently $V_{out} = GND = 0V$. So this circuit performs the NOT operation on the input voltage. But there's a huge problem with this circuit! Look at the drawn current from the supply source providing V_{DD} in this circuit. The current is equal to the I_R by the KCL law and shown above. So while the input is at V_{DD}, there's a static current passing through the supply and dissipating energy!!! Having billions of these circuits on a chip makes it impractical due to energy and power limitations. Also, you may wonder why not make resistors extremely large!? The answer is not only large resistors occupy large areas, they will cause circuit's to be slow (you'll see in next lectures).

Recall the main motivation for inventing digital circuit is noise tolerance. How does this inverter circuit can tolerate noise levels below noise margin? Next figure shows how the noise levels below V_{th} will be filtered at the output, while large noise amplitudes can certainly cause errors at the output. Notice, here we assumed the only noise source is the noisy input and circuit component are all ideal.
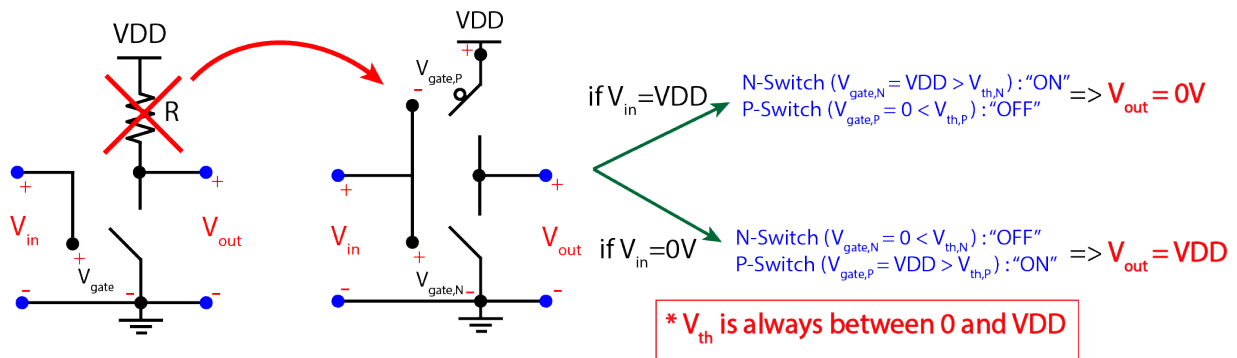
Question: How can we increase the noise margin this circuit? What are the advantages and disadvantages for each solution?

Errors @output due to intolerable noises!
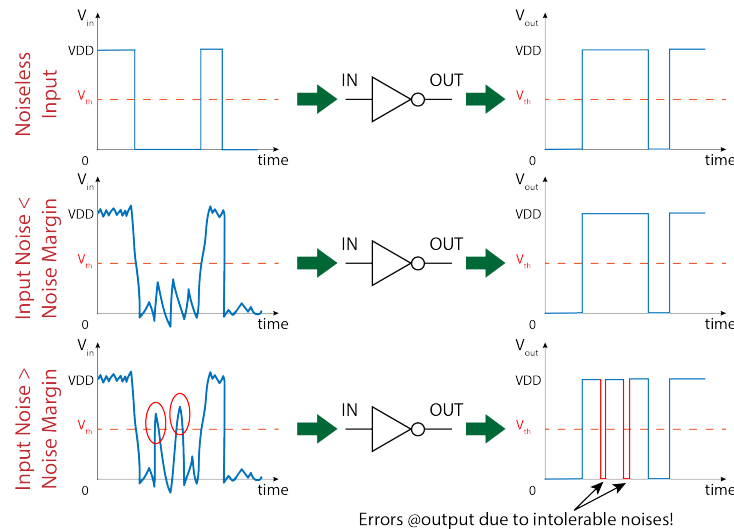
In order to solve the static current issue, we need to introduce a new type of switches and replace the resistor with this switch which turns OFF whenever the first switch in the circuit is ON and vice versa. This can be easily done, by swapping the Gate voltage polarity. We call this new switch device as "P-Switch" :

N-Switch

Gate
Terminal

P-Switch

$V_{gate} > V_{th}$ => Switch is "ON"

$V_{gate} \leq V_{th}$ => Switch is "OFF"
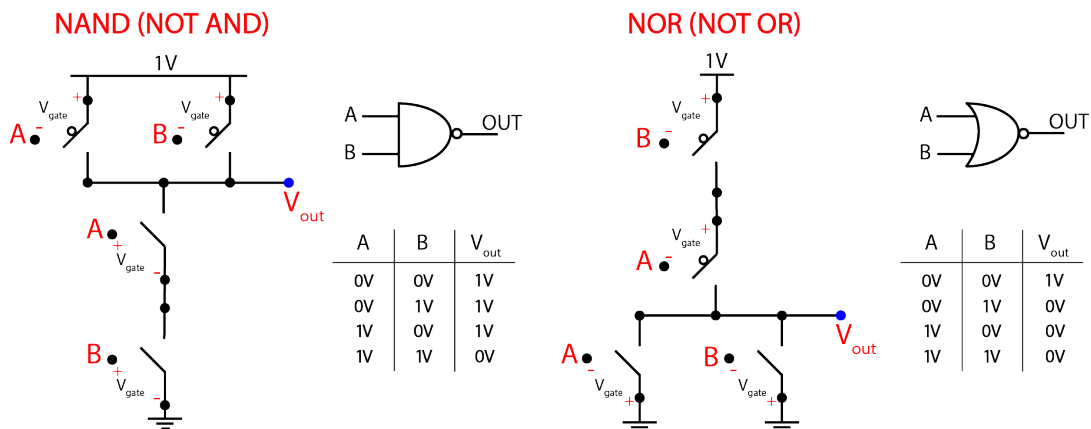
Notice V_{gate} polarity definitions!

Replacing the resistor with this new switch should fix the problem (Threshold voltage of N-Switch and P-Switch is denoted by $V_{th,N}$ and $V_{th,P}$, respectively):

if V_{in}=VDD: N-Switch ($V_{gate,N} = VDD > V_{th,N}$) : "ON" => $V_{out} = 0V$
P-Switch ($V_{gate,P} = 0 < V_{th,P}$) : "OFF"

if V_{in}=0V: N-Switch ($V_{gate,N} = 0 < V_{th,N}$) : "OFF" => $V_{out} = VDD$
P-Switch ($V_{gate,P} = VDD > V_{th,P}$) : "ON"

* V_{th} is always between 0 and VDD

Thus, with input voltage at either of interest values there will not be any current drawn from the supply :)

> You may have noticed that for some input voltages ($V_{th,N} < V_{in} < V_{DD} - V_{th,P}$) both switches will be ON! These voltages may show up temporarily in practice as input voltage is transitioning between two states (Notice input is set by the preceding logic gates). Studying the output behavior during the transitions require more elaborated switch models covered in (EE151/141) courses and are skipped here.

Extending the same design methodology, we can build NAND (NOT+AND) and NOR (NOT+OR) gates as well:

NAND (NOT AND)

A	B	V_{out}
0V	0V	1V
0V	1V	1V
1V	0V	1V
1V	1V	0V

NOR (NOT OR)

A	B	V_{out}
0V	0V	1V
0V	1V	0V
1V	0V	0V
1V	1V	0V

Eventually cascading these gates with inverters we already built, enable us to implement AND and OR gates too:
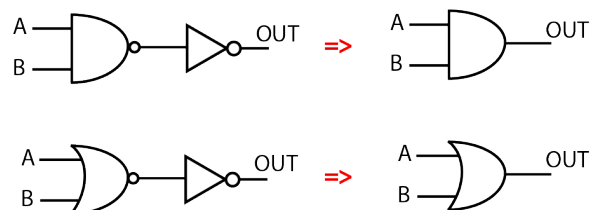
Question: Why we couldn't build AND/OR gates at the beginning directly without the help of NAND/NOR?
Hint: N-Switches are always placed on the path to the GND.
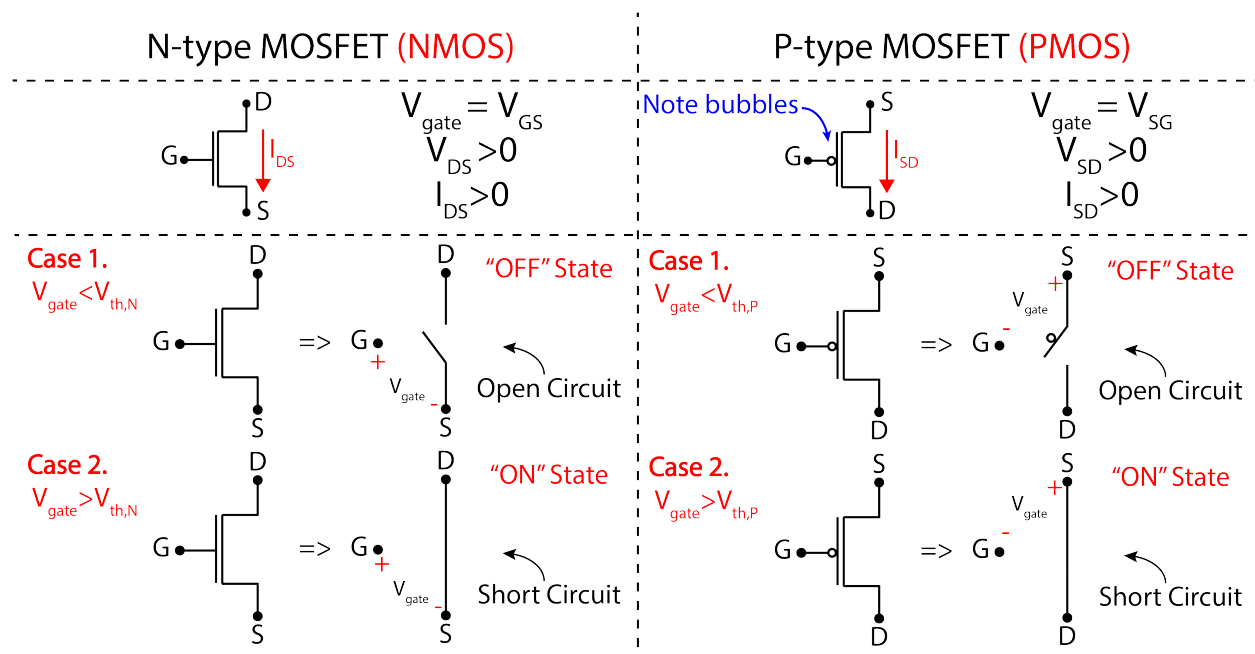
3 Transistors

So far we learned how to build logics using two types of switches and the fact that more complicated digital operators to perform useful functions such as summation, etc. can be built from them. In this section, we elaborate on implementing these switches and learn how they look like in reality.

The key enabler of all complex electronics around you such as smart phones, laptops, TV, etc. are electronic devices called **Transistors**. They are used in both analog and digital circuits. In digital circuits, they can be modeled as switches. There are many families of transistors, differentiated by different physics. In this class we will use the: Metal Oxide Semiconductor Field Effect Transistors (MOSFET).There are many FET variants, MOSFET is of one them. (For fun search JFET, TFET, HFET, etc.). MOSFETs in this class will be treated as 3 terminal devices known as: Gate (G), Drain (D), and Source (S). We can define some useful voltages and currents as shown below:
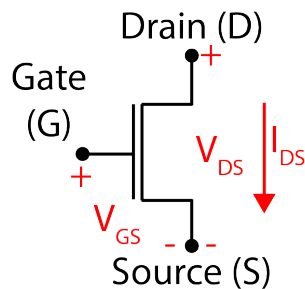
- I_{DS} is the current that flows from D to S.

- V_{GS} is the potential applied between G and S nodes.

- V_{DS} is the voltage arising between D and S.

Just like switches, we can build two types of MOSFETs in reality as well:

Important Points:

(a) Note there is always an open circuit between Gate (G) and Source (S) in the ideal model.

(b) The V_{th} is the threshold voltage and is an internal property of the transistors which is between $0V$ abd VDD. (i.e. we will tell you what V_{th} is).

(c) NMOS acts as a N-Switch and PMOS acts as a P-Switch.

(d) For NMOS transistors, the source is always the lowest potential, and for PMOS transistors, the source is always the highest potential[1].

Note: In many text books and possibly in future circuits classes, PMOS transistors are defined with the polarities of everything (gate voltage, threshold voltage, current) flipped, but we just do not like dealing with negative numbers in this class!

Transistor Physics

This section explains why these transistors are called MOSFET. The figure below shows a simplistic cross-section of a NMOS device. The semiconductor region is made of silicon normally. The conductivity of these materials can be manipulated and that's why they are called "semiconductor". More details on this subject will be covered in EE105 and EE230s.

The way this device works is as follows: the Metal-Oxide-Semiconductor sandwich forms a capacitor. By applying positive voltage this capacitor starts charging and floating electrons concentrate near the surface. If the voltage is high enough such that enough electrons rise and concentrate near the surface, an "electronic bridge" will form which connects source and drain regions together and electrons can travel in between them i. e. there can be an electrical current between S and D and transistor switch is ON. These devices are designed such that without enough electron concentration ("electron bridge"), source and drain are not connected and no electrical current can flow in between. Similarly, in the PMOS devices where positive charges are floating in the semiconductor, a strong enough negative voltage can form a bridge and turn the switch ON.

Notice in reality these devices are symmetric in a way that source and drain are exact similar structures and their voltages can be used to define them as a source or drain.
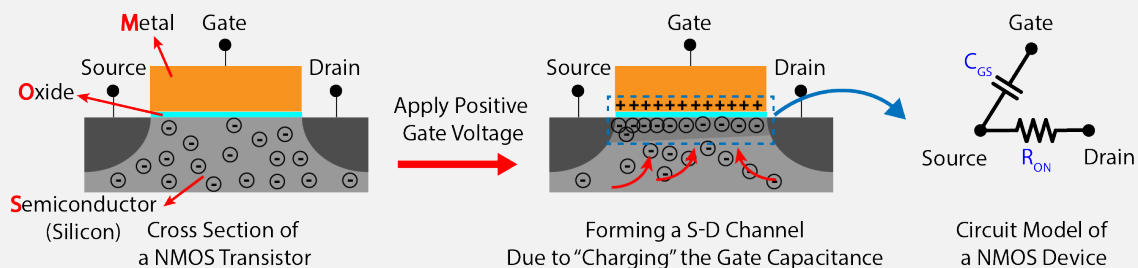

A better model of MSOFETs are also shown above; the "electron bridge" between S and D can be modeled as a resistor, R_{ON} when the device is ON. In this course we assumed, the resistance is infinity large when switch is OFF. Although, there is no conductive path between G and S/D, the gate capacitance exist between G/S and G/D, however, in this course for simplicity we only consider the capacitance between G and S (C_{GS}). All other sort of parasitic elements are ignored through the rest of this course as well.

---

[1]This is the reason why we use NMOS devices on the path to the GND and PMOSs on the path to VDD in CMOS logic gates.

3.1   Transistor model with resistors

The transistor does not look like a perfect conductor between D and S, actually a slightly better model is:
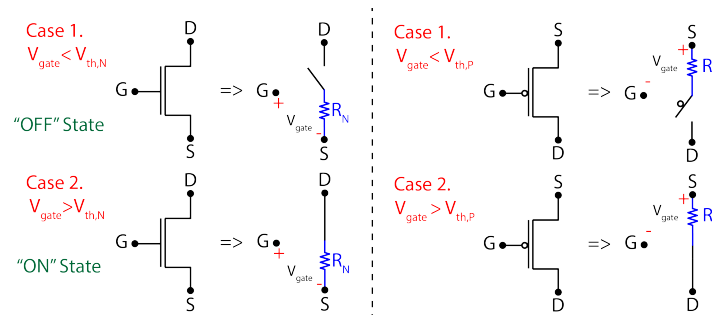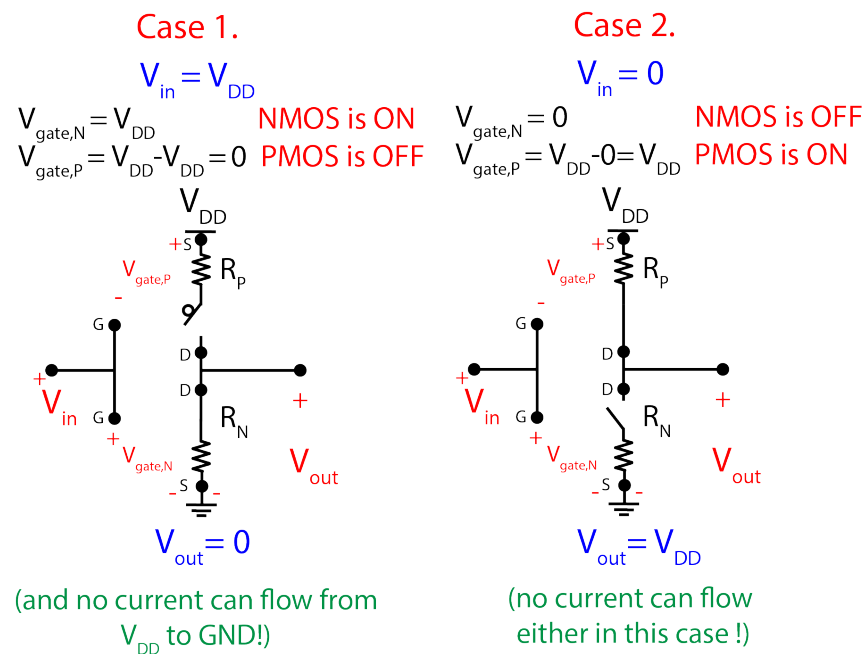

3.2   The CMOS Inverter

Now we introduce the Complimentary MOS (CMOS) Inverter. The PMOS and NMOS transistors act in a complimentary way just like P-type and N-type switches and this is why they are called CMOS! The vest bulk of the digital circuit "chips" in your devices are built from CMOS logic gates.
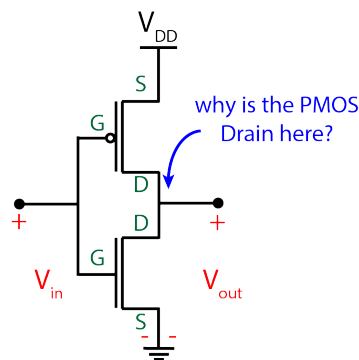
This circuit is directly derived from the switch-based inverter schematic. Pay close attention to the G, D, and S labels. Note $V_{gate,N} = V_{in}$ and $V_{gate,P} = V_{DD} - V_{in}$ and $V_{out} = V_{DS,N}$. Before we analyze the CMOS inverter, let's look at the first inverter we built using N-switch and resistance (called pseudo-NMOS inverter) shown below:
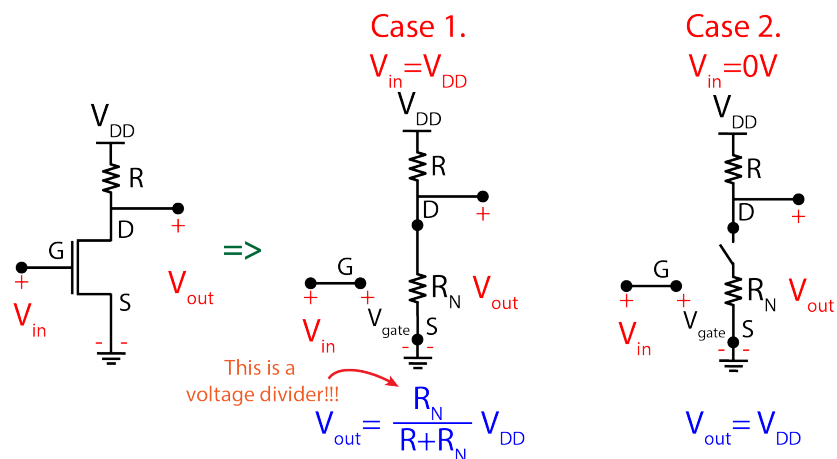
This figure revealed another issue of using only NMOS devices in addition to the static current in the ON state. The problem is that using transistor models with resistors, the output voltage is not $0V$ and it is a function of resistances and VDD. So for V_{out} to be low enough, R should be large again! However, in the CMOS inverter this issue does not exist:


### 3.2.1 Simple CMOS logic gates

The trick to build other logic gates are similar to what we did using switches. Notice that the CMOS inverter is built from two blocks:
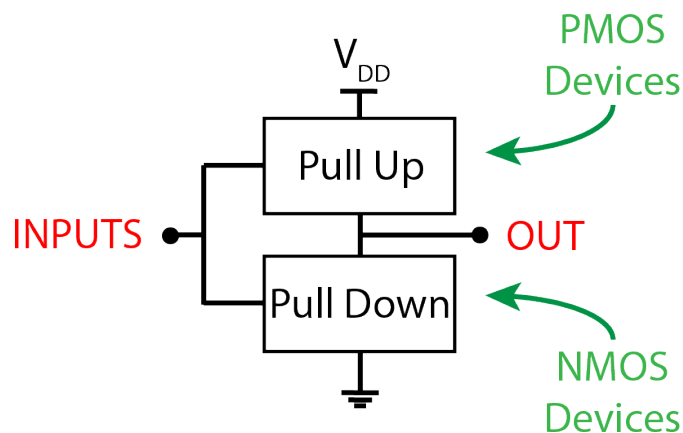
The job of the NMOS was to "pull down" the output to GND. Likewise, the job of the PMOS was to "pull up" the output to V_{DD}. They <u>never</u> do this at the same time! This is the underlying concept behind the logic gates we built from P- and N-switches as well. The following is the CMOS NOR gate:
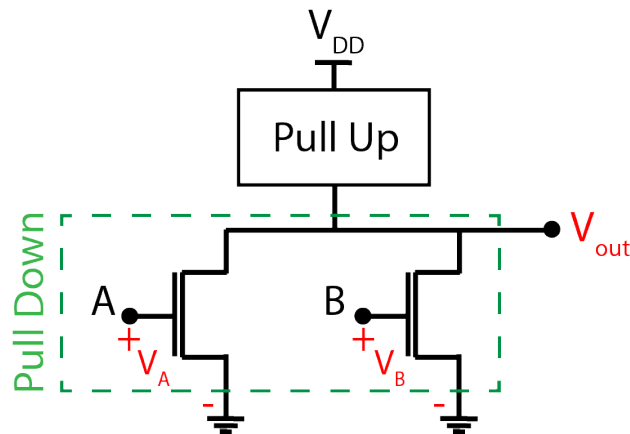
The pull down network (PDN) wants to short the output to GND if either $V_A = V_{DD}$ or $V_B = V_{DD}$ or both (check the condition of the NMOSs and see for yourself). We never want the Pull up network (PUN) to pull up the V_{out} to V_{DD} for any input combination that would pull V_{out} down to GND. Try this:
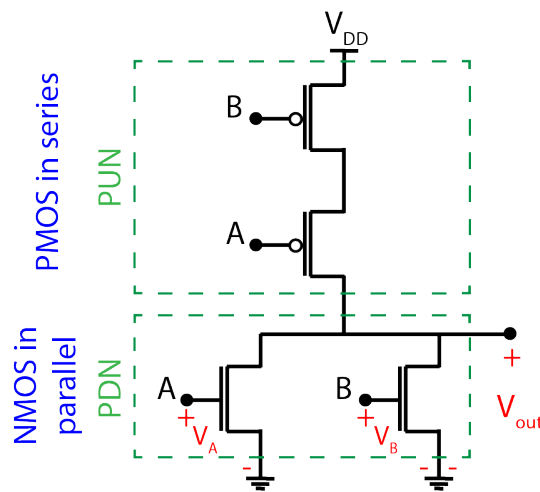
Here's the table of MOS status and logic:

V_A	V_B	PMOS A	PMOS B	NMOS A	NMOS B	V_{out}
0	0	ON	ON	OFF	OFF	V_{DD}
0	V_{DD}	ON	OFF	OFF	ON	0
V_{DD}	0	OFF	ON	ON	OFF	0
V_{DD}	V_{DD}	OFF	OFF	ON	ON	0

Hence, the circuit is a CMOS NOR gate. And finally you can verify yourself that the next schematic is a CMOS NAND gate:
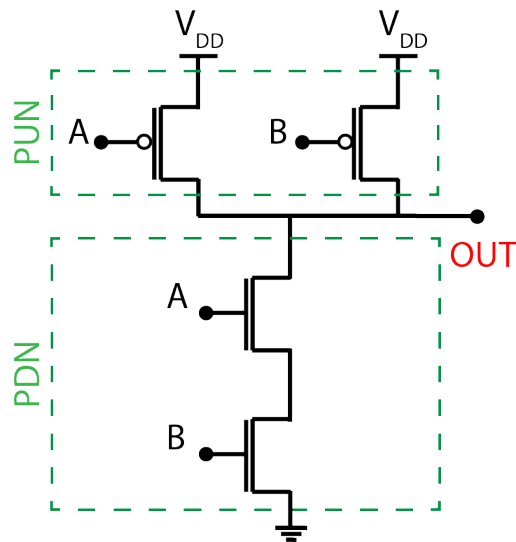
Summary

- Digital processing can tolerate noise and errors to certain levels and this makes them favorable to build complex processors.

- Binary digital circuits have the largest noise immunity compared with higher order multi-level digital systems.

- Any binary operator can be implemented by 3 fundamental logic gates: NOT, OR, AND.

- We showed how to build these basic gates using P-switches and N-switches.

- It turns out NMOS and PMOS transistors act as N- and P- type switches, respectively.

- All the modern electronics around us are designed using CMOS gates and in the next lecture we will study the speed/power limitations of these circuits ...

Contributors:

- Nikhil Shinde.

- Anant Sahai.

- Aditya Arun.