Lecture 1 - Notes

Rahul Arya

January 2019

1 Overview

In EE16A, we focused on analog circuits, often performing mathematical operations. However, analog circuits suffer from a high degree of susceptibility to noise. Consider, for instance, an analog circuit adding two 64 bit integers. Even if the levels of noise were as low 1 nV, our circuit would have to run with voltages of the order of

$$2^{64} \cdot (1\,\text{nV}) = 18\,\text{GV},$$

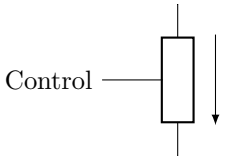
in order to handle the full range of the input, a clearly impractical value.

The use of *digital circuits* allows us to mitigate the problem of noise quite considerably, by adopting a discrete / quantized interpretation of a continuous signal. At its most basic level, a signal V in digital logic corresponds to two different logical values: 0 or 1. Signals very close to 0 correspond to logical 0, while signals very close to 1 (or some other maximum value) correspond to logical 1. The errors allowed before a signal is incorrectly interpreted are known as the *noise margins*, as shown below:

A key principle in digital logic is in avoiding the accumulation of noise across computations. Otherwise, even the most generous error bounds and the most noise-free computations will inevitably lead to error, as small errors accumulate over millions of computations. We wish to obtain a guarantee that if the inputs to our circuit lie within our noise margins, that the output signals will as well.

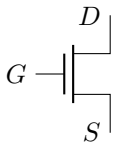
2 CMOS

To achieve this, we use what is known as CMOS logic (CMOS stands for "complementary metal oxide semiconductors"). CMOS consists of two related components (hence "complementary"): NMOS and PMOS. Both these components have the following function: they control a switch between two terminals, based on the signal supplied at a third terminal:

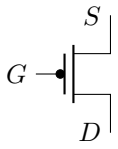


We call the control terminal the *gate*, with the controlled current (as labelled) between the *drain* and the *source*.

Typically, in circuit diagrams, these components are drawn as follows, with G representing the gate, D representing the drain, and S representing the source:



(a) NMOS

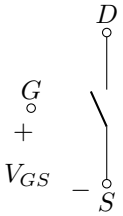


(b) PMOS

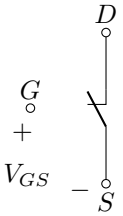
Of course, voltages are only meaningful when measured between two points. Both NMOS and PMOS devices respond to the voltage between the gate and the source, denoted as V_{GS}. However, from the diagram above, the drain and the source appear to be symmetrical, so it is not apparent which terminal is which. For NMOS devices, the source terminal is the terminal at a lower voltage, while for PMOS devices, the source terminal is the terminal at a higher voltage.

Now, we will discuss the precise behavior of these components in more detail. For NMOS devices, a voltage V_{GS} that is near 0 (more precisely, when $V_{GS} < V_{th,N}$ for some threshold voltage $V_{th,N}$) will cause current to be blocked between the drain and the source, while a voltage V_{GS} that is near some peak voltage V_{dd} ($V_{GS} > V_{th,P}$) will allow current to flow.

This behavior is summarized in the below figures, modelling NMOS and PMOS devices using switches:



(a) $V_{GS} < V_{th,N}$



(b) $V_{GS} > V_{th,N}$

The behavior of an NMOS device.

(a) $V_{GS} < V_{th,P}$ (b) $V_{GS} > V_{th,P}$

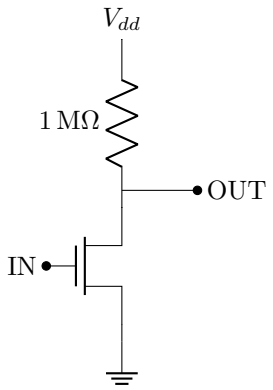
The behavior of a PMOS device.

3 Simple Inverters

Now, we will try to build circuits that do something interesting, using these new CMOS components. The first circuit we will build is an inverter - given a signal corresponding to a logical 0, it will output a signal corresponding to logical 1, and given a signal corresponding to a logical 1, it will output a signal corresponding to logical 0. We can represent the behavior of this circuit usign a truth table, as follows:

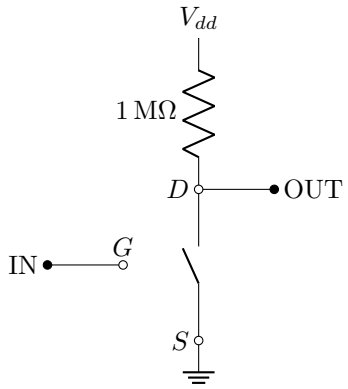
IN	OUT
0	1
1	0

The following circuit is one possible implementation of an inverter, using a single NMOS device:



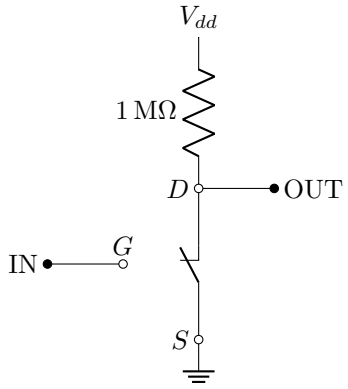
Let's see why this circuit behaves like an inverter. To do so, we should consider the two possible logical values of the input: logical 0, and logical 1.

When the input is at logical 0, the source and gate of the NMOS transistor are at equal voltages, so $V_{GS} < V_{th,N}$. Thus, the NMOS transistor's switch will open, so we obtain the following equivalent circuit:



When no current is drawn from the output terminal, no voltage drops along the resistor, so the output terminal remains at V_{dd}, corresponding to a logical 1, as we expect.

When the input is at logical 1, the gate of the NMOS transistor will be at voltage $V_{dd} > V_{th,N}$. Thus the NMOS transistor's switch will close, so we obtain the following equivalent circuit:



As can be seen, the output terminal is connected directly to ground, so the output of this circuit will be a logical 0. Thus, its behavior matches its truth table for both possible inputs, so this circuit is an accurate model of an inverter.

Observe also that small levels of noise at the input do not affect the output signal at all, so this circuit does not allow noise to accumulate across chained circuits, as we had hoped.

4 Efficient Inverters

However, this inverter has a number of drawbacks. Most notably, observe that when the input is at logical 1, current continually passes through the resistor. If we assume $V_{dd} = 1\,\text{V}$, a realistic value, then we find that the static power

consumption of this transistor is

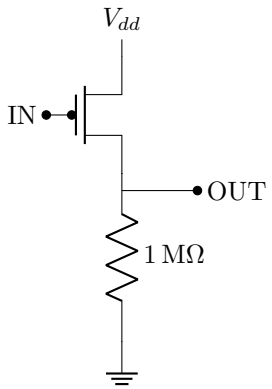
$$P = \frac{V^2}{R} = 1\,\mu\text{W}.$$

This doesn't seem like very much. But when we consider that a modern electronic processor has upwards of $N = 10^9$ transistors, we find that this power consumption adds up to a total of

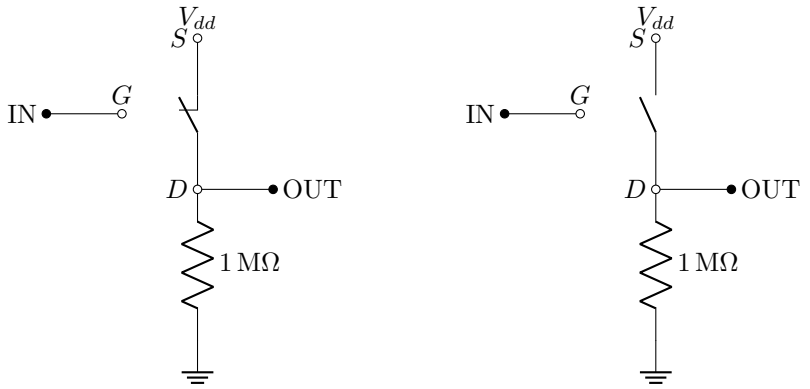
$$P_{device} = PN = 1\,\text{kW},$$

which is an unreasonably large quantity, especially for devices like smartphones or laptops.

It turns out that, by combining NMOS and PMOS devices, we can produce an inverter that has no static power consumption in either state. To do so, we need to first consider the opposite of the previous device, by building an inverter using a single PMOS device. We can do so as follows:



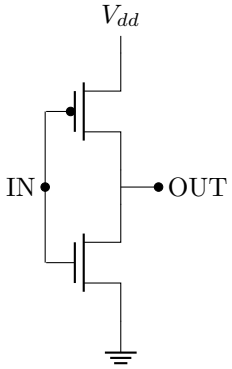
The analysis of this circuit is extremely similar to that of the NMOS inverter. We again consider both possible logical states of the input signal, determine the position of the switch in our model of the transistor, and draw the equivalent circuits, as follows:



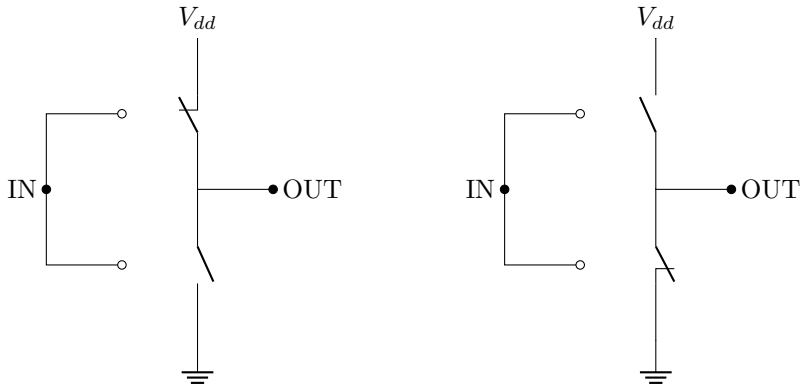
(a) Input at logical 0. (b) Input at logical 1.

Here, we see that the output signal is again what we expect. However, when the input is at logical 1, no power is consumed, whereas when the input is at logical 0, there is some static power. Looking at our previous NMOS inverter, we see the opposite behavior, with power consumed when the input is at logical 1 but not a logical 0.

It turns out that, by combining these two circuits, we can generate a so-called CMOS inverter that combines the best aspects of both of these circuits, and removes the need for a 1 MΩ resistor. We can do so as follows:



Let's see why this works. As before, we have to consider two cases: the input signal of logical 0, and logical 1. Applying our understanding of the behavior of NMOS and PMOS devices as modeled as switches, we obtain the following two equivalent circuits in these two cases:



(a) Input at logical 0. (b) Input at logical 1.

5 NAND and NOR gates

Thus, we clearly see that this new CMOS inverter behaves as we would expect, with the output signal always the logical opposite of the logical value of the input signal, and unaffected by any small levels of noise in the input signal. In addition, however, notice, when no load is connected to the output terminal, that no static power is drawn in either case, making this new inverter superior to either of the two we previously considered.

Now, we will consider some more complex gates - specifically, the NAND and NOR gates. These gates take two inputs, not one, but still produce a single output. The truth table of NOR is outlined below:

A	B	NOR(A, B)
0	0	1
1	0	0
0	1	0
1	1	0

Essentially, this gate produces an output of logical 1 only when both of the inputs are 0, and outputs logical 0 in all other cases.

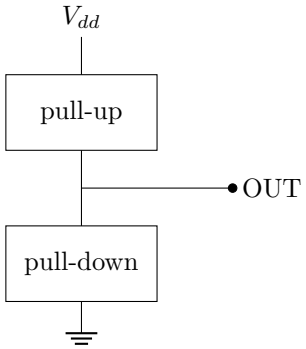
What is important about the NAND and NOR gates is that they are *universal* gates. This means that all other boolean functions can be implemented by creating a circuit consisting solely of that one type of gate. This is very useful, since it means that if we know how to produce NAND or NOR gates using whatever components we have, then we know how to use these components to produce any other digital function.

To produce a NOR gate (or any other boolean function) using CMOS, we typically divide the problem into two parts: “pull-up” and “pull-down”. The pull-down portion of a gate consists of those components that force the output to

be logical 0, and the pull-up portion consists of those that force the output to be logical 1. For any possible input, we want exactly one of the pull-up and pull-down portions to be active - if both are active, then we may short our power rails, while if neither are active, then our output may be disconnected entirely.

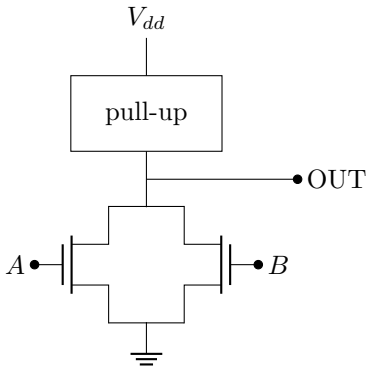
In the CMOS inverter we constructed previously, the single NMOS transistor connected to ground formed the pull-down portion of the circuit, and the single PMOS transistor connected to V_{dd} formed the pull-up portion of the circuit. In general, we always use NMOS transistors to pull the output down, and always use PMOS transistors to pull it up.

Thus, we can draw an arbitrary digital circuit modelling a single gate as follows:



Now, let's try constructing the pull-down portion of our NOR gate. The NOR gate should be pulled down to logical zero whenever either input signal is logical 1. Thus, we should place two NMOS transistors in parallel linking the output to ground, with each transistor corresponding to one of the input signals, so that if either signal is 1, then its corresponding transistor will pull the OUTPUT to ground.

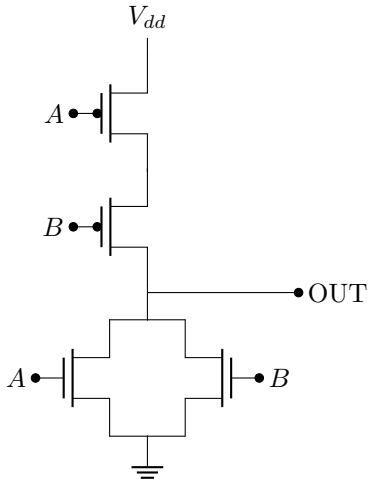
We now have a circuit that looks like this:



Now, when either A or B is logical 1, the output signal is pulled down to logical 0. However, when both A and B are set to logical 0, the switches in both transistors open, and the output node is left floating. To address this, we will now aim to construct a pull-up network that acts exactly in this case.

Incidentally, it can be shown (and will be shown in CS 61C) that any pull-down network has a corresponding pull-up network that ensures that the voltage sources are never shorted to ground, and that the output node is never disconnected from both rails. However, we do not need this proof for this specific example.

Our pull-up network requires both A and B to be set to logical 0, for a connection to be made between V_{dd} and the output. Clearly, we should place two PMOS transistors in series, connecting one transistor to each input signal. Thus, only when both signals are logical 0 will the switches for both transistors close, and a connection made to the output. Constructing this circuit, we obtain:



Similar techniques may be used to construct a NAND gate.