[VLSI Digital Circuit Design], by [Oklobdzija, Yano].
ISBN 0-471-XXXXX-X

Chapter 2: Draft – NOT TO BE REPRODUCED

Advances in CMOS Circuits

Vojin G. Oklobdzija, Kazuo Yano

Introduction

Today CMOS technology is used almost exclusively to design digital circuits. But this was not the case in the past. VLSI era started with nMOS spanning a decade of 1970s. Most of the early processors were therefore designed using nMOS technology. The first microprocessor to use CMOS was RCA 1802 COSMAC which was an 8-bit processor; however there was also a CMOS implementation of PDP-8 instruction set in the Intersil IMS-6100 microprocessor chip manufactured by Harris Semiconductor. The first powerful 32-bit microprocessor buit in CMOS technology was Bellmac-32, which was also the first 32-bit microprocessor at the time. It was finished in 1982 and it not only introduced CMOS technology, but the new CMOS logic circuit called “Domino”, which will be described later in this chapter. Before about 1985 CMOS was considered to be expensive and slow technology and was used only in special devices that required either very low power or radiation hardening.

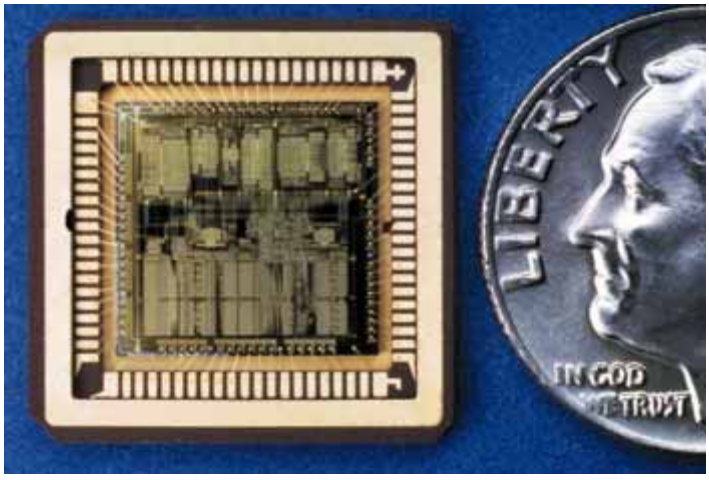

Fig. 1. The first 32-bit CMOS microprocessor designed at Bell Laboratories in 1982.

CMOS Operation

CMOS operates on the complementary switch principle. While the path to ground is enabled, the path to power supply is disabled and vice versa. So, the static power consumption of CMOS is reduced only to the leakage current and negligible, as long as we leakage currents are contained. The topology of a CMOS digital logic block is shown Fig. 1. In principle we can realize any Boolean function by mapping the function into the switching network consisting of n-MOS transistors for the pull-down part and p-MOS transistors in the pull-up part. In practice there is a limit on the number of transistors in the path, the more severe limit being imposed on the pull-up, or p-MOS transistor path. The reason for that is that the mobility of the carriers in the p-type MOS transistor is about one half of that for the n-type MOS transistor, thus making the pull-up path considerably slower. At some point, the speed of such a block becomes so slow that breaking the function into two blocks makes more practical sense. In practice, we rarely allow more than two p-MOS transistors in series in the pull-up path, while allowing three or four (but not more than four) n-MOS in pull-down path is possible.

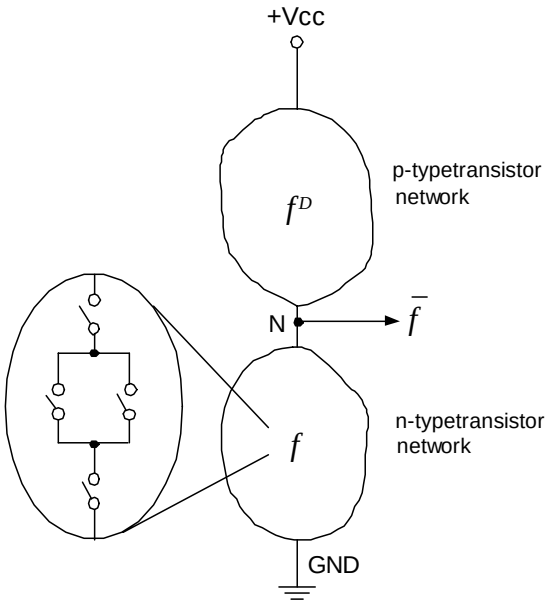

Fig. 1. Principle of CMOS circuit operation

Creating the circuit topology for a CMOS circuit is straight forward and could be derived directly from the Karnaugh map. An example implementing CMOS NAND gate is shown in Fig. 2. By circling ones the pull-up switching network topology (p-MOS transistor path) is determined, while circling zeros will yield the pull-down path (switching topology consisting of n-MOS transistors). More complex functions are possible, thus the notion of gate is loosing its meaning in CMOS VLSI digital circuits. We should get used to the fact that we are operating with CMOS blocks (functions) implementing a part of a given function, rather than simple Boolean primitives such as NAND, NOR etc.

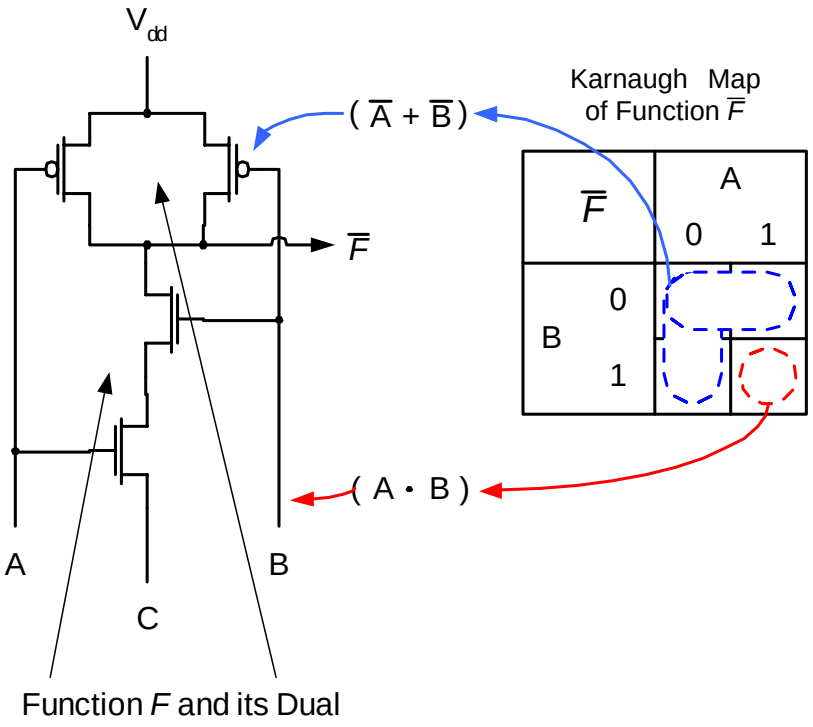

Fig. 2. Determining CMOS topology from a Karnaough map. An example of a NAND gate is given.

An example of producing a more complex functional block is shown in Fig. 3. In the case shown, the p-MOS transistor path contains three p-MOS transistors in series, thus this block will most likely have to be broken into two. A better realization of this circuit, which minimizes the number of p-MOS transistors in series, is shown in Fig. 3.b. It contains only one transistor in the p-path but it does allow for three n-MOS transistors in series to be connected in n-path. It also requires both polarities of the inputs A,B,C and D.

Sometimes the relationship between the topology of the p-path and the n-path is not obvious from the equation. And example of the function with symmetric n-path and p-path containing bridge circuit is shown in Fig. 4.

CMOS circuit could not implement a fast XOR and XNOR function efficiently. Therefore some pass-transistor alternatives were proposed. They are shown in Fig. 5.

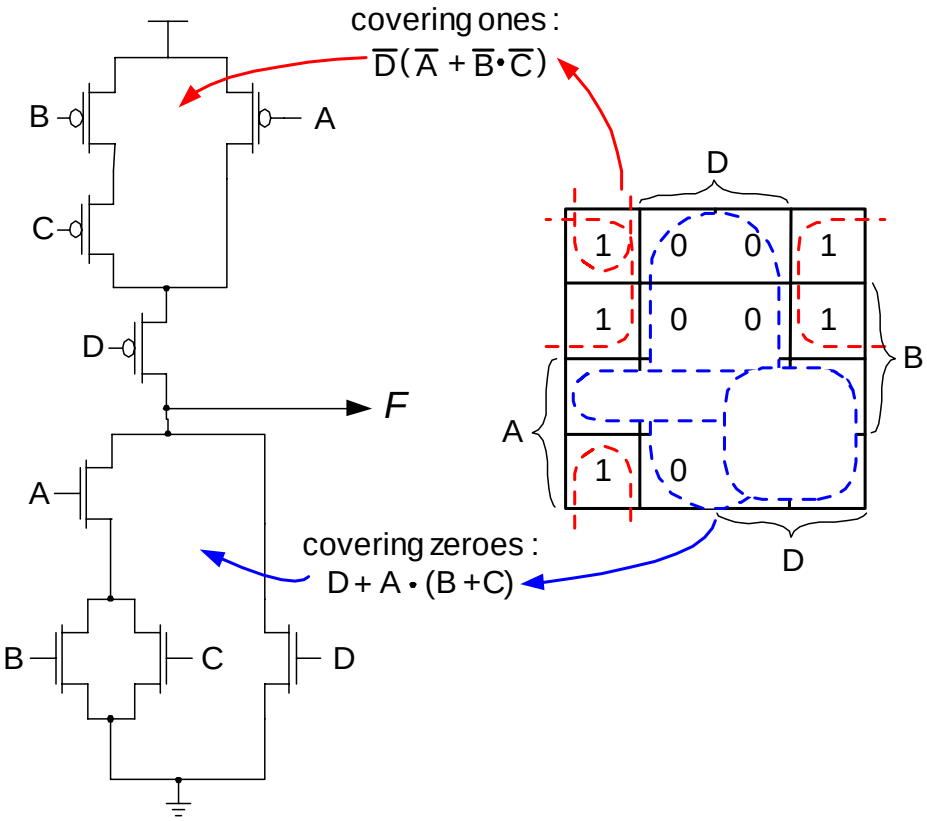

Fig. 3. Derivation of the CMOS functional block from the Karnaugh Map. Logic representation is given (b)

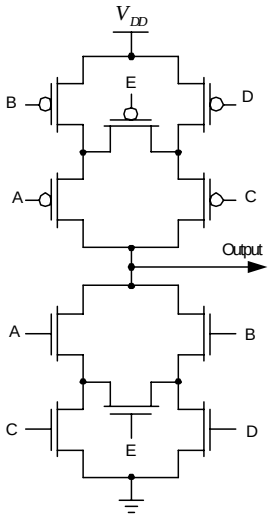

Fig. 4. CMOS block exploiting the symmetry between n-path and p-path

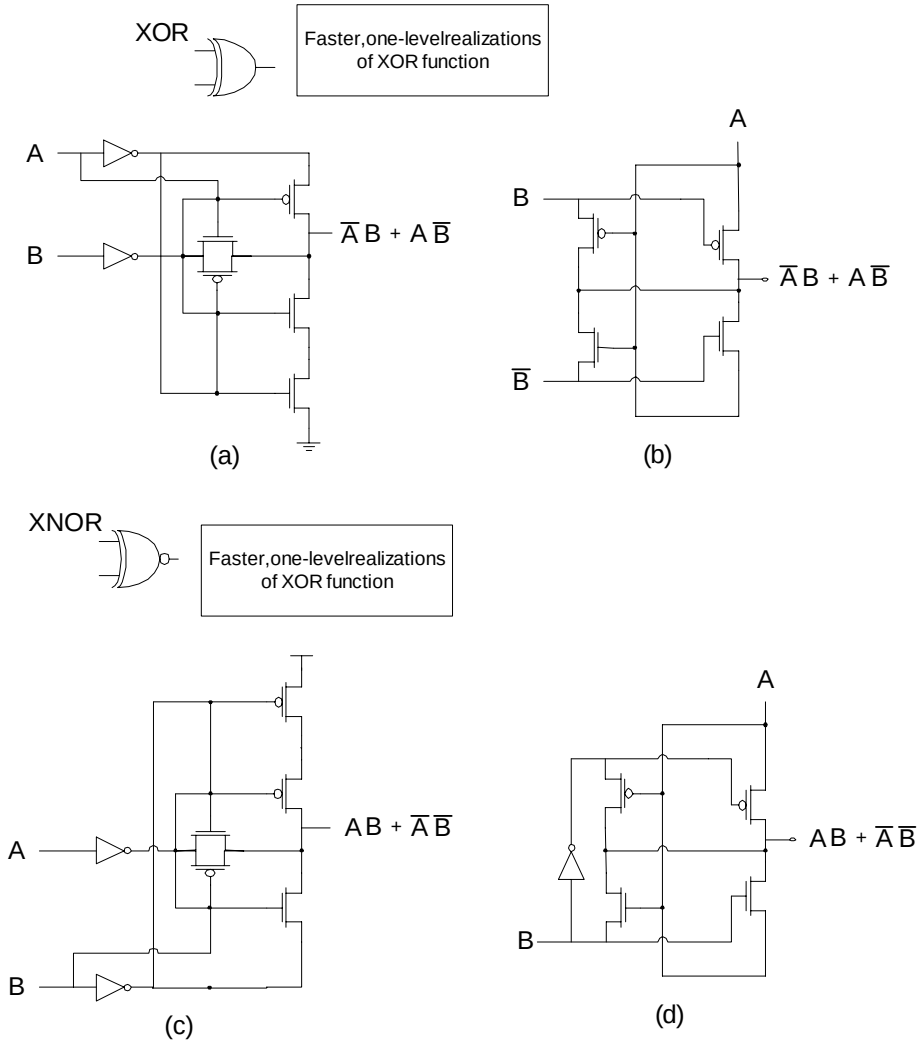

Fig. 5. Efficient way of implementing XOR, XNOR function

Inverter Transfer function

Transfer function of a CMOS inverter is shown in Fig. 6.

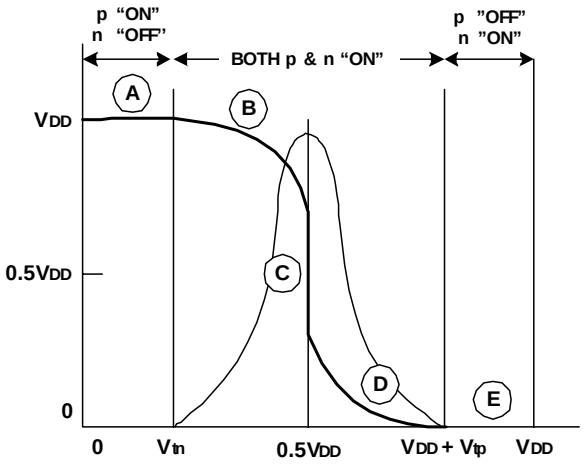

Fig. 6. CMOS inverter transfer characteristic

Show how the switching points changes depending on the p/n size ratio.

Show current on the transfer characteristic. How the current does exist during the transition.

Noise Margins of CMOS circuits

CMOS has better noise margins than nMOS because of the full voltage swing on its output. The best way to determine noise margins is to plot an inverter transfer function and flip it by interchanging the X and Y axes.

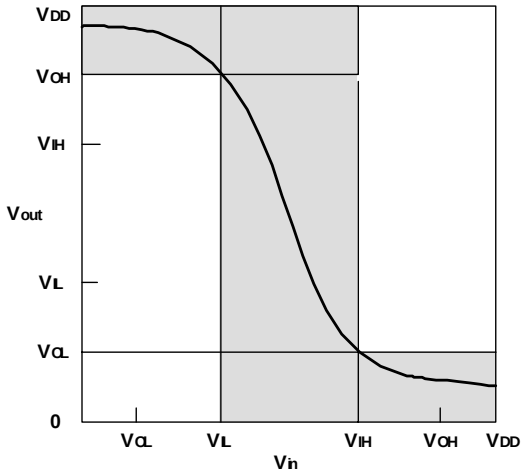

Fig. 7. Vout to Vin Transfer characteristic

Power Consumption in CMOS

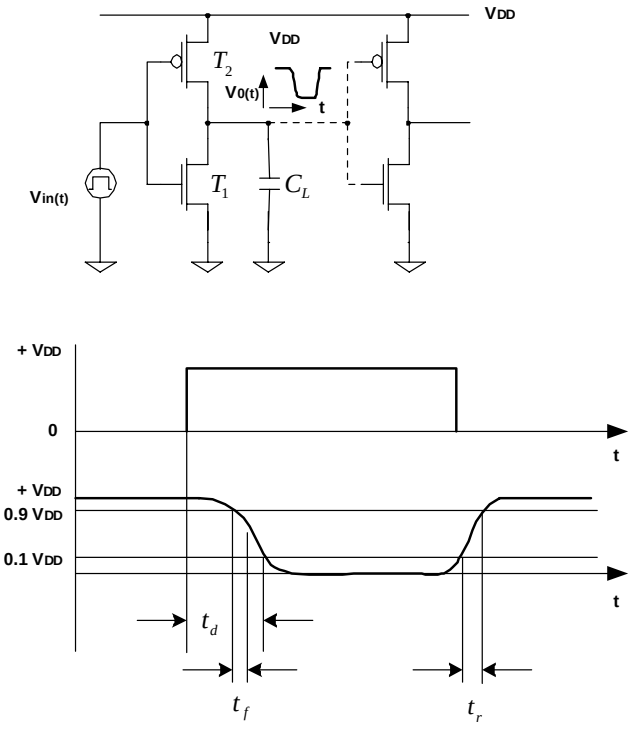

Fig. 8. Power consumption

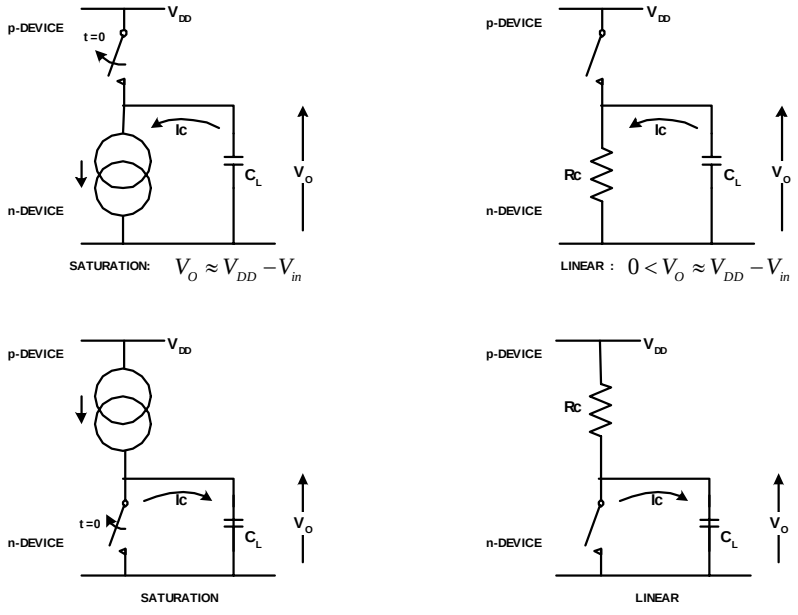

Fig. 9. Charging and discharging during the transition

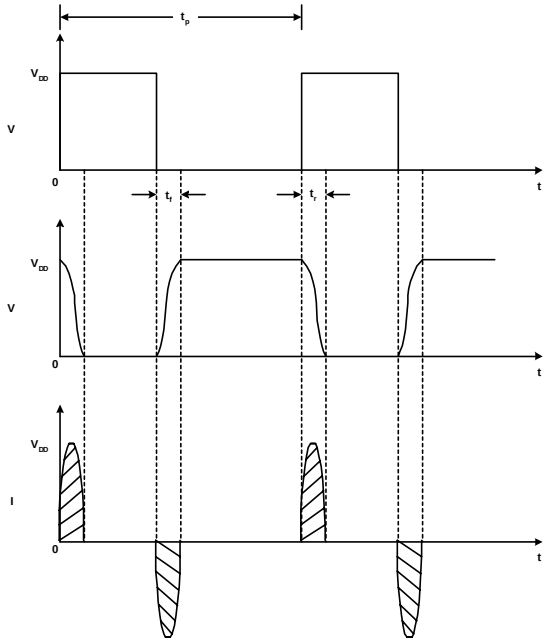

Fig. 10. Power

$$P_{CMOS}=kC_LV^2_{DD}f_o$$

This is an E=mc2 of low-power design

There are three ways to control power:

- Reducing Power-Supply Voltage (most effective !!)
- Reducing the switching activity *k* (various ways)
- Reducing ***CL*** (technology scaling etc.)
- Reducing the required frequency of operation (?)

Speed of CMOS Circuits

It is not easy and straight forward to estimate the speed of a signal path in CMOS. This is because the speed is dependent on many factors, including the size of the transistors. Having fewer levels of logic, or fewer CMOS block in the signal path does not always guarantee the faster path.

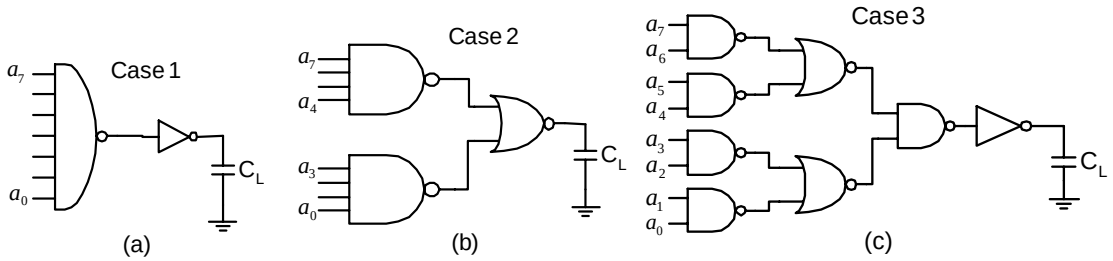

Fig. 11. 8-way NAND design

For example, let us take a case of a large 8-input AND gate. We could implement it as an 8-input NAND gate followed by an inverter, as shown in Fig. 11.a. Alternatively, this could be done in two levels using two 4-input NAND gates and one 2-input NOR gate, as shown in Fig.11.b. Finally, we could use all 2-input NOR and NAND gates followed by an inverter, and implement it in the way shown in Fig. 11.c. The example given in Fig. 11.c. has four stages of inversion in the signal path, while the case a. and b. have two. Never the less, the case c. produces the fastest path. How do we know that ? The answer is given in transistor sizing theory known as "Logical Effort" described in the next chapter.

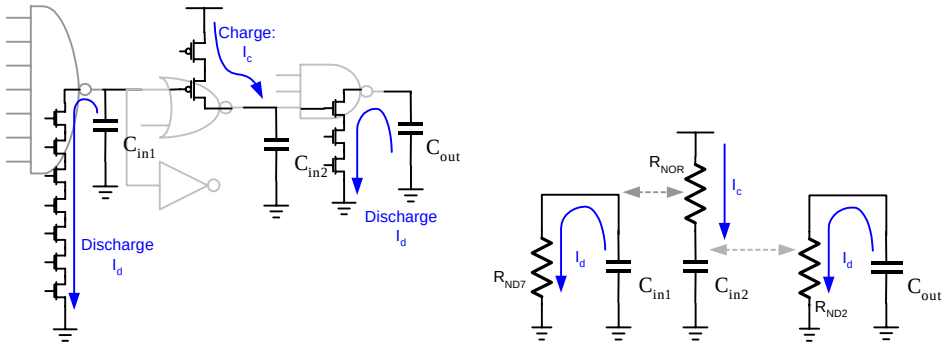

Fig. 12. RC delay in the chain of gates.

Delay can be approximated with:

$$\mathbf{R_{ND7}C_{in1}+R_{NOR}C_{in2}+R_{ND2}C_{out}}$$