Notes for ECE 27000 - Introduction to Digital System Design

Zeke Ulrich

February 12, 2025

Contents

Course Description

An introduction to digital system design, with an emphasis on practical design techniques and circuit implementation.

Verilog

SystemVerilog is a hardware description and verification language (HDVL) that extends Verilog by adding high-level programming constructs. It is widely used for modeling, simulating, and verifying digital systems.

The listing below is an example of a NAND gate expressed in SystemVerilog.

Listing 1: NAND

```
module nand_gate (
    input  logic a,
    input  logic b,
    output logic y
);
    assign y = ~(a & b);
endmodule
```

In SystemVerilog, numbers are written in format `[size]'[base][number]`, for example:

- 4'b1001 (binary, 9 in decimal, bit width 4 bits)
- 8'hf1 (hex, equals 421, bit width 8 bits)
- 3'o3 (octal, 3, bit width 3 bits)
- 32'b1001_1101_0101_1111 (binary, 40255, bit width 32 bits)

Number Systems

In daily life, we primarily interact with the familiar base-10 numbers. However, when interaction with digital systems, we must also concern ourselves with base-2, base-8, base-16, and other bases which are friendly to binary states. Unless completely unambiguous, the base of a number is written as a right subscript such as 144_{10} for base-10 or 1001_2 for base-2.

For binary numbers, each digit represents a power of two. To convert a binary number to decimal, you sum the products of each binary digit with its corresponding power of two. For example, the binary number 1001 is calculated as

$$\begin{aligned} 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 0 + 2^0 \times 1 &= 8 + 0 + 0 + 1 && (1) \\ &= 9_{10} && (2) \end{aligned}$$

To convert from hexadecimal to decimal, each digit represents a power of sixteen. For instance, the hexadecimal number f1 is calculated as

$$\begin{aligned} 15 \times 161 + 1 \times 160 &= 240 + 1 && (3) \\ &= 241_{10} && (4) \end{aligned}$$

where f represents the decimal value 15. When converting to another base, reverse the process by dividing the decimal number by the target base, recording the remainder, and repeating with the quotient until it reaches zero. The remainders give you the digits of the number in the new base, read in reverse order.

To convert a decimal number into binary, for example, you repeatedly divide the number by 2 and record the remainders. For the decimal number 9, dividing by 2 gives a quotient of 4 and a remainder of 1. Dividing 4 by 2 gives a quotient of 2 and a remainder of 0. Dividing 2 by 2 gives a quotient of 1 and a remainder of 0, and finally, dividing 1 by 2 gives a quotient of 0 and a remainder of 1. Reading the remainders from bottom to top, the binary representation of 9 is 1001.

Boolean Algebra

Computers operate in binary. To represent the state of a computer we require a suitable mathematical framework, provided by boolean algebra. In boolean algebra, variables can only take on two values: 0 and 1.

Rule	**Expression**
Commutativity	$X + Y = Y + X$ $X \cdot Y = Y \cdot X$
Associativity	$(X + Y) + Z = X + (Y + Z)$ $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$
Distributivity	$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$ $(X + Y) \cdot (X + Z) = X + Y \cdot Z$
Covering	$X + X \cdot Y = X$ $X \cdot (X + Y) = X$
Combining	$X \cdot Y + X \cdot Y = X$ $(X + Y) \cdot (X + Y) = X$
Consensus	$X \cdot Y + X \cdot Z + Y \cdot Z = X \cdot Y + X' \cdot Z$ $(X + Y) \cdot (X + Z) \cdot (Y + Z) = (X + Y) \cdot (X + Z)$
Generalized Idempotency	$X + X + \cdots + X = X$ $X \cdot X \cdot \cdots \cdot X = X$
DeMorgan's Theorems	$(X_1 \cdot X_2 \cdot \cdots \cdot X_n)' = X_1' + X_2' + \cdots + X_n'$ $(X_1 + X_2 + \cdots + X_n)' = X_1' \cdot X_2' \cdot \cdots \cdot X_n'$
Generalized DeMorgan's	$F(X_1, X_2, \ldots, X_n, +, \cdot) = F(X_1, X_2, \ldots, X_n, \cdot, +)'$
Shannon's Expansion	$F(X_1, X_2, \ldots, X_n) = X_1 \cdot F(1, X_2, \ldots, X_n) + X_1' \cdot F(0, X_2, \ldots, X_n)$ $F(X_1, X_2, \ldots, X_n) = [X_1 + F(0, X_2, \ldots, X_n)] \cdot [X_1' + F(1, X_2, \ldots, X_n)]$

Table 1: Boolean Algebra

An interesting and useful property in boolean algebra is "duality", where replacing all ANDs with ORs and all 1s with 0s gives a valid and equivalent theorem. For instance,

X AND 0 = 0	X OR 1 = 1
X OR 0 = X	X AND 1 = X

Table 2: Boolean Duality

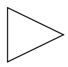
Any logic can be implemented using just the following:

- AND, OR, and NOT gates
- NAND gates
- NOR gates

Logic Gates

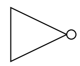
Buffer

A	Output
0	0
1	1



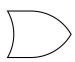
NOT

A	Output
0	1
1	0



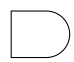
OR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1



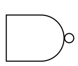
AND

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



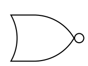
NAND

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0



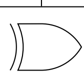
NOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0



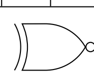
XOR

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0



XNOR

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1



CMOS

Complementary Metal-Oxide-Semiconductor (CMOS) technology is the dominant semiconductor technology for modern integrated circuits. CMOS combines both n-type (NMOS) and p-type (PMOS) metal-oxide-semiconductor field-effect transistors (MOSFETs) to create the familiar logic gates such as AND, NOT, XOR, etc.

PMOS

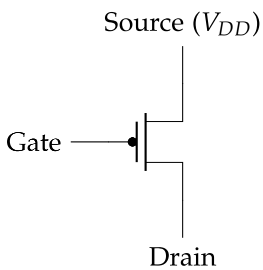


Figure 1: PMOS transistor circuit symbol

PMOS transistors consist of:

- p+ source and drain regions
- n-type substrate (body)
- SiO_2 gate dielectric
- Polysilicon gate electrode

PMOS operates with negative gate-to-source voltage (V_{GS}):

- **Cut-off Region** ($V_{GS} > V_{th,p}$):

$$I_D = 0$$

- **Linear Region** ($V_{GS} \leq V_{th,p}$ and $V_{DS} \geq V_{GS} - V_{th,p}$):

$$I_D = -\mu_p C_{ox} \frac{W}{L} \left[(V_{GS} - V_{th,p}) V_{DS} - \frac{V_{DS}^2}{2} \right]$$

- **Saturation Region** ($V_{GS} \leq V_{th,p}$ and $V_{DS} < V_{GS} - V_{th,p}$):

$$I_D = -\frac{1}{2} \mu_p C_{ox} \frac{W}{L} (V_{GS} - V_{th,p})^2$$

NMOS

NMOS transistors consist of:

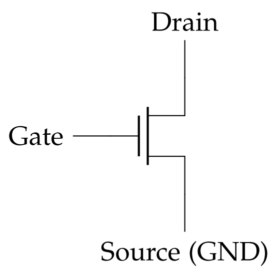


Figure 2: NMOS transistor circuit symbol

- n+ source and drain regions
- p-type substrate (body)
- SiO_2 gate dielectric
- Polysilicon gate electrode

NMOS operates with positive gate-to-source voltage (V_{GS}):

- **Cut-off Region** ($V_{GS} < V_{th,n}$):

$$I_D = 0$$

- **Linear Region** ($V_{GS} \geq V_{th,n}$ and $V_{DS} \leq V_{GS} - V_{th,n}$):

$$I_D = \mu_n C_{ox} \frac{W}{L} \left[(V_{GS} - V_{th,n}) V_{DS} - \frac{V_{DS}^2}{2} \right]$$

- **Saturation Region** ($V_{GS} \geq V_{th,n}$ and $V_{DS} > V_{GS} - V_{th,n}$):

$$I_D = \frac{1}{2} \mu_n C_{ox} \frac{W}{L} (V_{GS} - V_{th,n})^2$$

Parameter	**PMOS**	**NMOS**
Majority Carrier	Holes	Electrons
Substrate Type	n-type	p-type
Threshold Voltage	Negative	Positive
Mobility (μ)	Lower ($\approx 150 \frac{cm^2}{V_s}$)	Higher ($\approx 400 \frac{cm^2}{V_s}$)
Speed	Slower	Faster

Table 3: PMOS vs NMOS characteristics

Propagation Delays and Transition Times

Propagation Delay

The propagation delay T_P is defined as the time delay between the 50% crossing of the input and the corresponding 50% crossing of the output. There are two kinds:

t_{pHL}: The time between an input change and the corresponding output change when the output is changing from HIGH to LOW.

t_{pLH}: The time between an input change and the corresponding output change when the output is changing from LOW to HIGH.

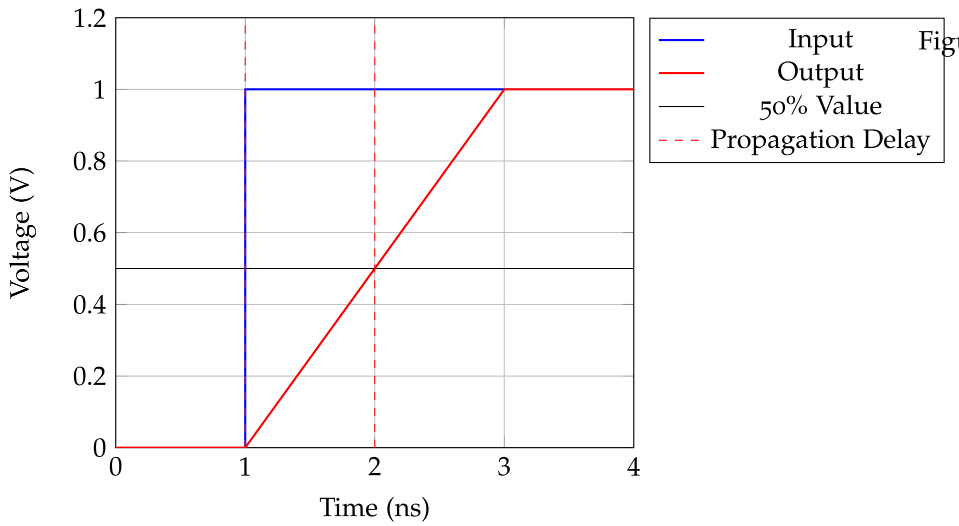


Figure 3: Propagation Delay

Transition Time

The amount of time that the output of a logic circuit takes to change from one state to another is called the transition time. An output takes a certain time, called the rise time (t_r), to change from LOW to HIGH, and a possibly different time, called the fall time (t_f), to change from HIGH to LOW. The rise time and fall time of the output signal are defined as the time required for the voltage to change from its 10% level to its 90% level or vice versa.

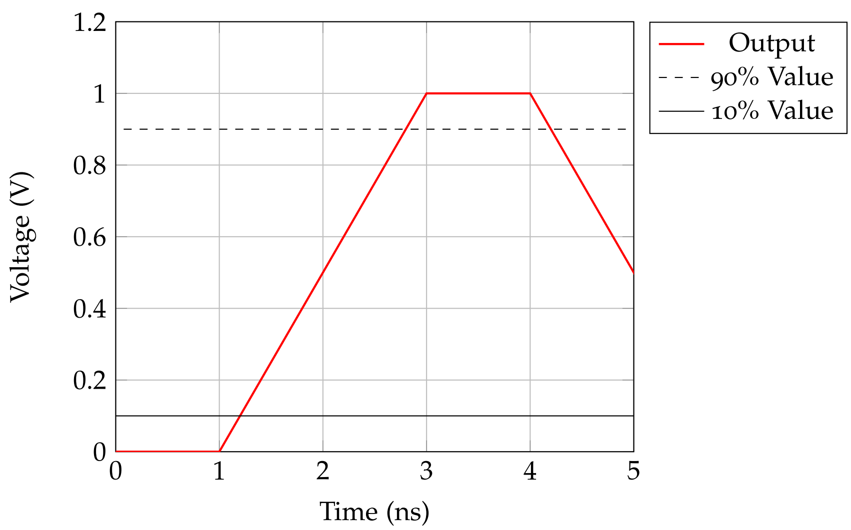


Figure 4: fig:risefalltime

Power Consumption

The power consumption of a CMOS circuit whose output is not changing is called *static power dissipation*. The power a CMOS circuit consumes during signal transitions is called *dynamic power dissipation*.

In general dynamic power dissipation is much larger than static. One significant source of dissipation is output transitions. The power consumed as the voltage transitions is given by

$$P_T = C_{PD} V_{CC}^2 f \tag{5}$$

where

- P_T is circuit's internal power dissipation due to output transitions
- C_{PD} is power-dissipation capacitance, normally specified by the device manufacturer
- V_{CC} is power supply voltage
- f is transition frequency of the output signal

A second (and often more significant) source of CMOS power consumption is the capacitive load (C_L) on the output.

Noise

$$N_{MH} = V_{OH} - V_{IH} \tag{6}$$

$$N_{ML} = V_{IL} - V_{OL} \tag{7}$$

$$N_{MH}(A \to B) = V_{OHA} - V_{IHB} \tag{8}$$

$$N_{ML}(A \to B) = V_{ILB} - V_{OLA} \tag{9}$$

K-Maps

A Karnaugh Map (K-Map) is a visual representation used to simplify Boolean expressions and minimize logic functions. It is a method to perform Boolean algebra simplifications by organizing truth table values into a grid format, allowing easy identification of common terms. K-Maps are structured as grids where each cell corresponds to a specific combination of input variables. The number of cells in a K-Map depends on the number of variables:

- 2-variable K-Map: $2^2 = 4$ cells
- 3-variable K-Map: $2^3 = 8$ cells
- 4-variable K-Map: $2^4 = 16$ cells

For a function with two variables (A and B), the K-Map is a 2×2 grid:

AB	0	1
0	F(0,0)	F(0,1)
1	F(1,0)	F(1,1)

For three variables (A, B, C), the K-Map has $2^3 = 8$ cells:

AB \C	0	1
00	F(0,0,0)	F(0,0,1)
01	F(0,1,0)	F(0,1,1)
11	F(1,1,0)	F(1,1,1)
10	F(1,0,0)	F(1,0,1)

The goal of a K-Map is to group adjacent cells containing 1s into power-of-two groups (1, 2, 4, etc.), forming simplified expressions. Groups can wrap around the edges of the K-Map.

An example would be illustrative. Consider the K-Map in figure 5.

A \BC	00	01	11	10
0	0	1	1	0
1	0	0	1	1

Figure 5: K-Map Example

There are two groups, the two 1s in the first row and the two 1s in the second. From the top group we get the term $A'B'C + A'BC = A'C$. From the bottom group we get the term $ABC + ABC' = AB$, and summing these product terms we get the final expression $A'C + AB$.