

# *Notes for ECE 29595PD - Principles of Digital System Design*

*Shubham Saluja Kumar Agarwal*

*January 12, 2024*

These are lecture notes for spring 2024 ECE 29595PD at Purdue. Modify, use, and distribute as you please.

## *Contents*

<i>Course Introduction</i>	1
<i>Introduction</i>	2
<i>Digital Logic Signals</i>	2
<i>Logic Circuits</i>	3
<i>Gates</i>	4
<i>Timing</i>	4
<i>Software Aspects of Digital Design</i>	5
<i>Integrated Circuits</i>	5
<i>Logic Families</i>	5
<i>CMOS Logic Circuits</i>	5

## *Course Introduction*

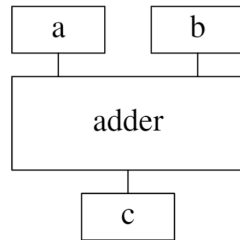
This course serves as an introduction to digital system design, with an emphasis on principles of digital hardware and embedded system design. It is an alternate class to ECE 27000.

Learning Outcomes:

1. Ability to analyze and design combinational logic circuits.
2. Ability to analyze and design sequential logic circuits.
3. Ability to analyze and design computer logic circuits.
4. Ability to realize, test, and debug practical digital circuits.

## Introduction

Digital design entails creating hardware that can conduct an operation or set of operations within a computer system. For example, adding two numbers.



This hardware can add two numbers, that is, conduct the operation  $c = a + b$ . It could also perform  $f = d + e$  or  $i = g - h = g + (-h)$  as it is not restricted to the sole values of  $a$  and  $b$  as inputs. This process fits into the logic design and switching algebra portions of chip manufacturing.

The creation of systems like these is based on the fact that voltage and current are time-varying and can assume any value in a continuous range of real numbers, but are mapped to only two values.

## Digital Logic Signals

A digital signal is modeled as assuming, at anytime, only one of two discrete values, which represent:

0	1
LOW	HIGH
FALSE	TRUE

This is called positive logic. This maps the infinite values of voltage and current to the two values. An example of this is CMOS 2-Volt logic:

0	1
0-0.5V	1.5-2.0V

These completely separated ranges of values allow for 0 and 1 to be completely separate, with noise and other possible errors being ignored. In this way, all physical values can be partitioned into the two values, though, for intents and purposes of digital system design, voltage is the most relevant.

Additionally, circuits known as buffer circuits can be used to restore logic values. That is, if a voltage is not sufficiently close to the values

of 0 or 2 (in the case of the CMOS), they can push these values far closer to the desired values, reducing the possibility of error.

Digital circuits have replaced analog circuits because they are far easier to design. They can be made using Hardware Description Languages (HDLs), softwares that are similar to programming languages. A logic value is called a binary digit, or bit. A set of  $n$  bits, represent  $2^n$  values. For example:

$b_0$	
0	
1	
$b_0$	$b_1$
0	0
1	0
0	1
1	1

### Logic Circuits

At the highest level, a logic circuit is a *black box* with  $n$  input and  $m$  outputs. Only zeros and ones are required to represent inputs and outputs.

Combinational circuits are circuits that have outputs that solely depend on the inputs. It can be represented by a truth table.

An adder can be represented by a truth table in this manner.

$x_1$	$x_2$	$x_3$	$z_2$	$z_1$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

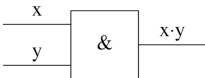
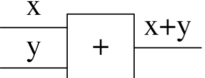
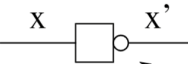
On the other hand, a sequential circuit has memory. That is, the output is dependent on both the inputs and the current state of the circuit itself. This is representable through a state table.

input	present state	next state	output
0	00	00	0
1	00	01	0
0	01	01	0
1	01	10	0
0	10	10	0
1	10	00	1

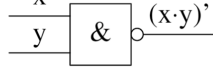
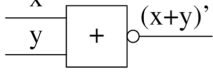
## Gates

Basic digital devices are called gates. These implement basic logic functions. These are AND, OR, NOT. However, AND, NOT and OR, NOT are sufficient to make any combination, as the third gate can be formed as a combination of the other two.

Each of the gates is represented symbolically like the following, and have their truth tables shown below:

AND			OR			NOT	
							
x	y	$x \wedge y$	x	y	$x \vee y$	x	$\neg x$
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

This can be done for more inputs, by combining multiple gates. There are two more often used gates that can be created through a combination of these two.

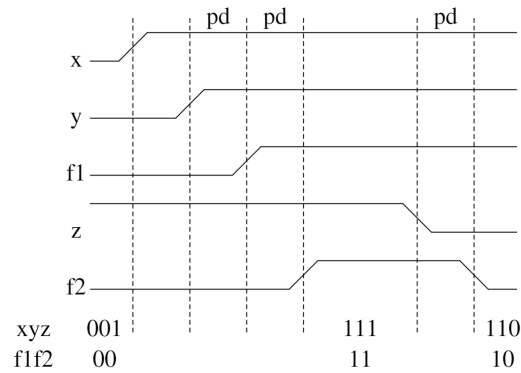
NAND			NOR		
					
x	y	$\neg(x \wedge y)$	x	y	$\neg(x \vee y)$
0	0	1	0	0	1
0	1	1	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0

It is a good exercise to see how two of the basic gates (AND, NOT or OR, NOT) can be used to create the rest. It is possible to connect gates to form complex circuits and thus, complex functions. By convention, signal flows from left to right. Thus, output comes out the right and input goes in at left.

## Timing

When the values of an input changes, it takes time for the outputs to change as well, which is known as *propagation delay*, which can be represented in a timing diagram. However, for most cases, it is possible to ignore these, as the behavior is well defined regardless of delay.

The following is a timing diagram:



Propagation delay, represented by  $pd$  in the timing diagram, is irrelevant if there are no changes, regardless of the fact that there is a new set of inputs. The values at the bottom of the diagram are the values that each of the elements hold after all the necessary transitions have happened for that set of inputs.

Transition time is the time it takes for a signal to change from 0 to 1, and is represented by " $rt$ ".

### *Software Aspects of Digital Design*

#### *Integrated Circuits*

#### *Logic Families*

There are many different ways of implementing logic circuits. Connectable logic circuits are of the same logic family. So, the technology controlling them is different. Chips need to be from the same family to be connected to each other. If they are from different families, they cannot be connected to each other.

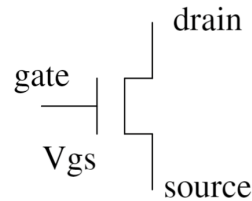
For example TTL (Transistor-Transistor Logic) is a logic family based on bipolar junction transistors.

CMOS (Complementary Metal-Oxide Semiconductor) is based on MOSFETs (MOS Field-Effect Transistors). These are more commonly used, and the ones we will be analyzing in this class.

#### *CMOS Logic Circuits*

A MOS transistor is modelable as a voltage controlled resistor. This means that the input voltage controls the resistance of the MOSFET, allowing it to act as a kind of switch. This is because it has only two effective states: very high resistance and very low resistance.

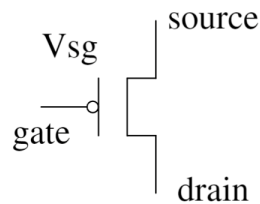
This is an n-channel (NMOS) transistor:



When  $V_{gs} = 0$ , the resistance is very high.

When  $V_{gs}$  is increased, the resistance is very low.

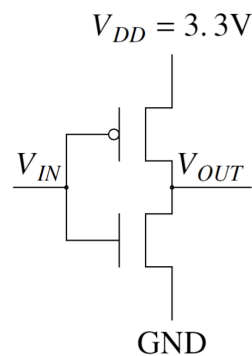
On the other hand, this is the PMOS transistor:



When  $V_{sg} = 0$ , the resistance is very high.

When  $V_{sg}$  is increased, the resistance is very low.

NMOS and PMOS together allow us to form CMOS logic. For example, this is a CMOS inverter, in which the output will be the opposite of the input.

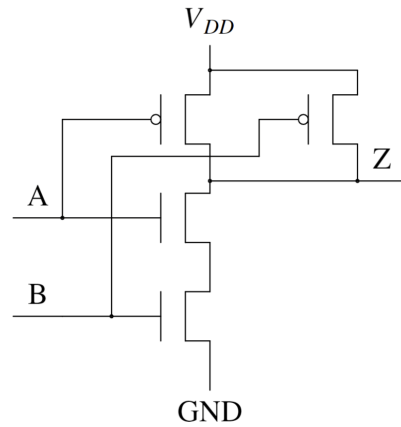


If  $V_{in} = 3.3V$ , or high, the PMOS transistor be off, and the NMOS would be on. This would mean that the output and ground are shorted, causing the output to be low.

If  $V_{in} = 0V$ , or low, the PMOS transistor be on, and the NMOS would be off. This would mean that the output and  $V_{DD}$  are shorted, causing the output to be high.

This allows this to be act as an inverter.

Now we will observe the CMOS arrangement of a NAND gate.



If A and B are both HIGH, both PMOS are HIGH, and thus  $V_{DD}$  is shorted to Z.

If they are both LOW, the exact opposite happens, with Z shorting to ground.

If A or B are HIGH, than one of the PMOS will be high, and one of the NMOS will be LOW, causing the output to be HIGH.

The NOR gate can be constructed easily from the same thought process, but NMOS in parallel, and PMOS in series.

These can be expanded to more inputs by adding more transistors.