Notes for ECE 29595PD - Principles of Digital System Design

Shubham Saluja Kumar Agarwal

January 10, 2024

Contents

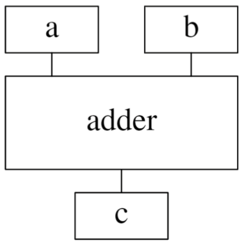
Course Introduction

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 teh possibility of error.

Digital circuits have replaced analog circuits becuse 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

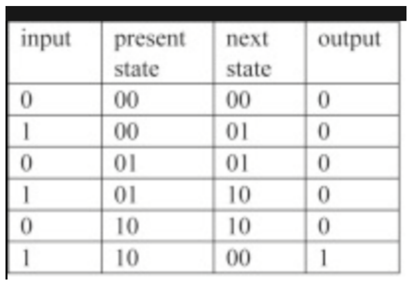
b_0	b_1
0	0
1	0
0	1
1	1

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.

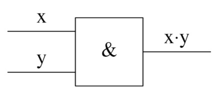
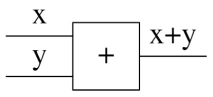
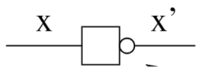


input	present state	next state	output
0	00	00	0
1	00	01	0
0	01	01	0
1	01	10	0
0	10	10	0
1	10	00	1

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:

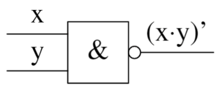
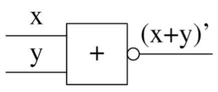
AND	OR	NOT
		

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

x	$\neg x$
0	1
1	0

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	$\neg(x \wedge y)$
0	0	1
0	1	0
1	0	0
1	1	0

x	y	$\neg(x \vee y)$
0	0	1
0	1	1
1	0	1
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.