Lecture 4 – Digital Systems and Binary Numbers – Part III

Rami Abielmona
University of Ottawa
January 23, 2015
ITI 1100 B
Digital Systems I

Presentation Outline

- Problem Solving during Class
- Boolean Algebra
- Basic Boolean Operations
- Logic Gates
- Binary Logic and Signals
- Basic Logic Gates
- Key terms

Examples: Signed Complements 2's

	Sign-bit	
$(9)_{10}$	0	1001
$+(6)_{10}$	0	0110
	0	1111

$(9)_{10}$	0	1001
$-(6)_{10}$	1	1010
	0	0011

$(6)_{10}$	0	0110
$-(9)_{10}$	1	0111
	1	1101

Examples: Signed Complements 1's

	Sign-bit	
$(9)_{10}$	0	1001
$+(6)_{10}$	0	0110
	0	1111

$(9)_{10}$	0	1001
$-(6)_{10}$	1	1001
1	0	0010 = (0010) + (0001) = (0011)

$(6)_{10}$	0	0110
$-(9)_{10}$	1	0110
	1	1 1 0 0

Problems

Question

(a) Convert le following binary number into (i) Octal, (ii) Decimal, (iii) hexadecimal

10101101.10110

(b) Convert A= 16.25 and B = 8.25 into binary, use 7 bits to represent the integer part and 3 bits to represent the fractional part, then perform the following operations

I) C= A + B
ii) D = A – B
iii) E= A x B
vi) F = A B

Note: Compute C and D
(a) using non-signed binary numbers and without complements
(b) using signed 2's complement

(c) Convert le following number into (i)Decimal, (ii) Octal, (iii) binary

$$(FD8.C2B)_{16}$$

Problems

Answers: **10101101.10110**

i) Octal→ **(010 101 101.101 100)**

(2 5 5 . 5 4)$_8$

iii) Hexa → **(1010 1101.1011 0000)2**

(A D . B 0)16

ii) Decimal

(10101101.10110)$_2$ = $1 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2}\ 1 \times 2^{-3} + 1 \times 2^{-4} + 0 \times 2^{-5}$

= (173.6875)$_{10}$

Problems

Integer part

- Conversion de $(16)_{10}$.

```
16 / 2   = 8    remainder 0
 8 / 2   = 4    remainder 0
 4 / 2   = 2    remainder 0
 2 / 2   = 1    remainder 0
 1 / 2   = 0    remainder 1
```

- Then $(16)_{10} = (10000)_2$

Problems

Fractional part

– Converting: $(0.25)_{10}$

0.25 x 2 = 0.50
0.50 x 2 = 1.00 ↓

- then, $(.25)_{10} = (.01)_2$

and $(16.25)_{10} = (10000.01)_2$

Same as for A

→ B = 8.25: $(8.25)_{10} = (1000.01)_2$

→ Representation using 7 bits and 3 bits

A= $(16.25)_{10}$ = $(0010000.010)_2$

B= $(8.25)_{10}$ = $(0001000.010)_2$

Problems

	Non Signed Binary
A	0010000.010
+ B	0001000.010
	0 011000.100

A	0010000.010
- B	0001000.010
	0001000.000

Problems

Signed 2’ complement

A	0 010000.010
+ B	0 001000.010
	0 011000.100

A	0 010000.010
+ (-B)	1 110111.110
	0 001000.000

Problems

A 0 010000.010

x B 0 001000.010

0000000000

0010000010

0000000000

0000000000

0000000000

0000000000

0010000010

1000110.000100

Problems

A ÷ B

Dividend

10000.010

1000010

01000 0000

1000010

Divider

1000 .010

Quotient

1.1..

Binary Logic

- Binary logic deals with

1 - **Variables** that can take on **two** discrete **values**

→*Values can be called **True**, **False**, **yes**, **no**, etc.*

2 - **Operations** that assume **LOGICAL** Meaning

→ *Binary logic is equivalent to **Boolean algebra***

Boolean Algebra

•Basic mathematics required for the description of digital circuits

• used to describe the different interconnections of digital circuits

• the variable used in the Boolean algebra are called Boolean variables

→ We will study two-valued Boolean algebra and functions with simplifications using basic Boolean Identities

Two-valued Boolean Algebra

- It consists of

1- Boolean Variables

- Designated by letters of the alphabet such as A, B, C, x, y, z etc.

- Each variable can have two and only two distinct values: 1 and 0 (True, False)

- Can be a Function of some other Boolean variables (F=ABC)

2- Boolean Operations

-There are three Basic logical operations:

AND, OR, and NOT

Basic Boolean Operations- *AND operation*

- Represented by a dot or by the absence of an operator

Example: $x.y = or \quad xy=z$

read: x *AND* y *is equal to* z

Interpretation: $Z = 1$ *if and only if* $x= 1$ *AND* $y= 1$

Otherwise $z = 0$

Truth table:

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

Don't confuse this with binary multiplication operation

Truth table gives the value of z for all possible values of x and y

Basic Boolean Operations- *OR operation*

- *Represented by a plus sign (+)*

Example: $x + y = z$

read: *x OR y is equal to z*

Interpretation: z = 1 if x= 1 or if y= 1 or if both x =1 and y =1. z = 0 if x = 0 and y=1

Don't confuse this with binary addition operation

Truth table:

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

Truth table gives the value of z for all possible values for x and y

Basic Boolean Operations- *NOT operation*

- *Represented by a prime or an overbar (also called complement)*

Example: $x' = z$ *(or* $\bar{x} = z$*)*

read: *Not x is equal to z*

Interpretation: *z = "what x is not"*

x= 1 then z=0; x= 0 then z=1

Truth table:

x	x'
0	1
1	0

Truth table gives the value of z for all possible values for x

Binary Logic and Binary Signals

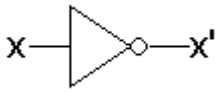
• For simplicity, we often still write digits instead:

  – 1 is true
  – 0 is false

• We will use this interpretation along with special operations to *design functions* and *logic circuits* for doing arbitrary computations.

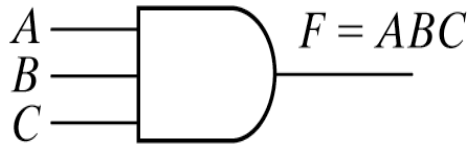
Logic Gates

- **Logic gates are electronic circuits that operate on one or more input signal to produce an output signal**
- Basic operations can be implemented in hardware using a Basic logic gate.
  - Symbols for each of the logic gates are shown below.
  - These gates output the product, sum or complement of their inputs

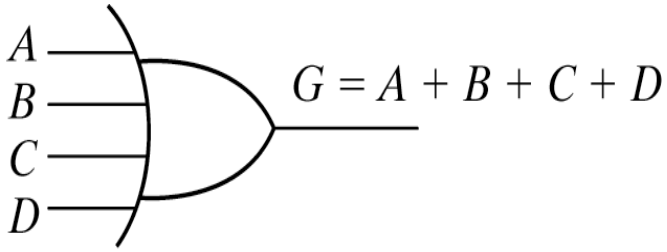
Logic Operation:	AND (product) of two inputs	OR (sum) of two inputs	NOT (complement) With one input
Representation:	x.y, or xy	x + y	x’
Logic gate:			

Gates with Multiple Inputs

- AND and OR Gates may have more than 2 input signals



(a) Three-input AND gate



(b) Four-input OR gate

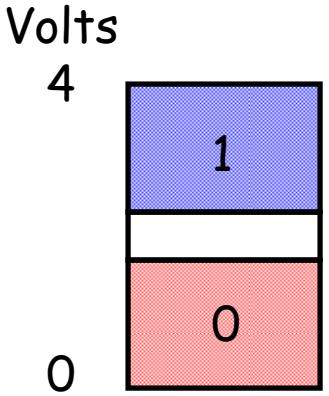
Binary Signals

- Computers use voltages to represent information.
- Two voltage levels are used to represent a binary value

  “1” and “0”

- Some digital systems for example may define that:
  - Binary ‘0” is equal to 0 Volt
  - Binary “1” is equal to 4 Volt

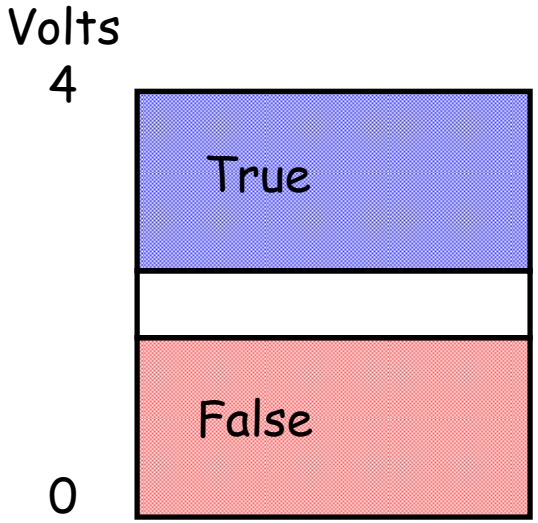
→ *It’s convenient for us to translate these voltages into values 1 and 0.*



Binary Logic and Binary Signals

•It's also possible to think of voltages as representing two *logical* values, ***true*** and ***false***.

→ These logical values are called Boolean values



Logic Gates - Signals

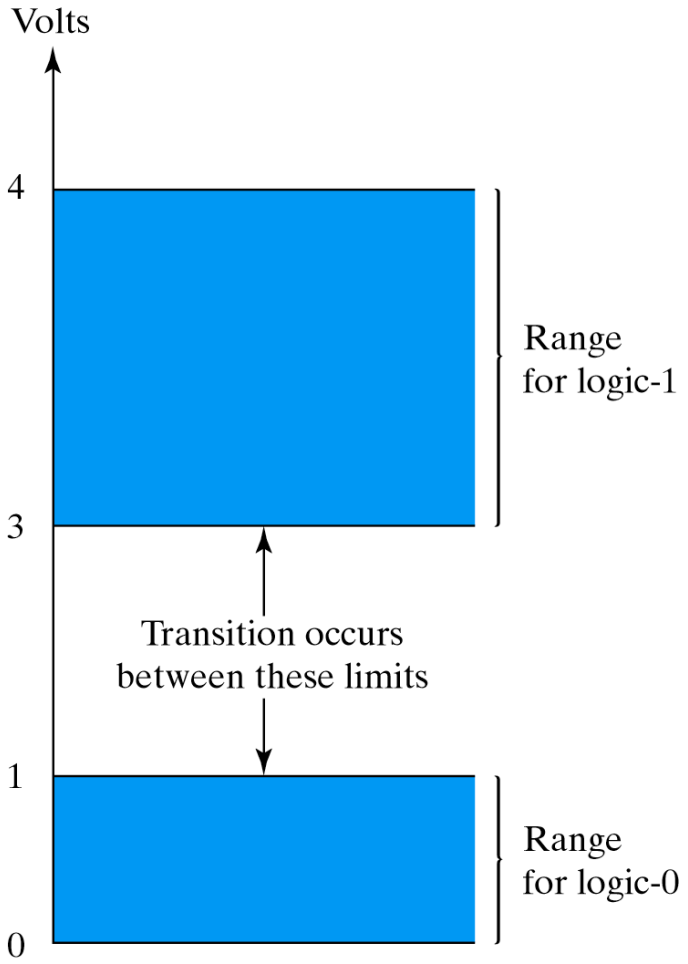
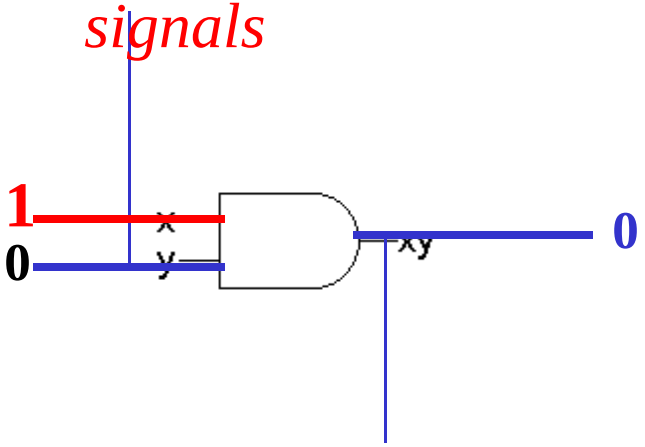


Fig. 1-3 Example of binary signals

Example

two input signals



one output signal

Logic Gates - Signals

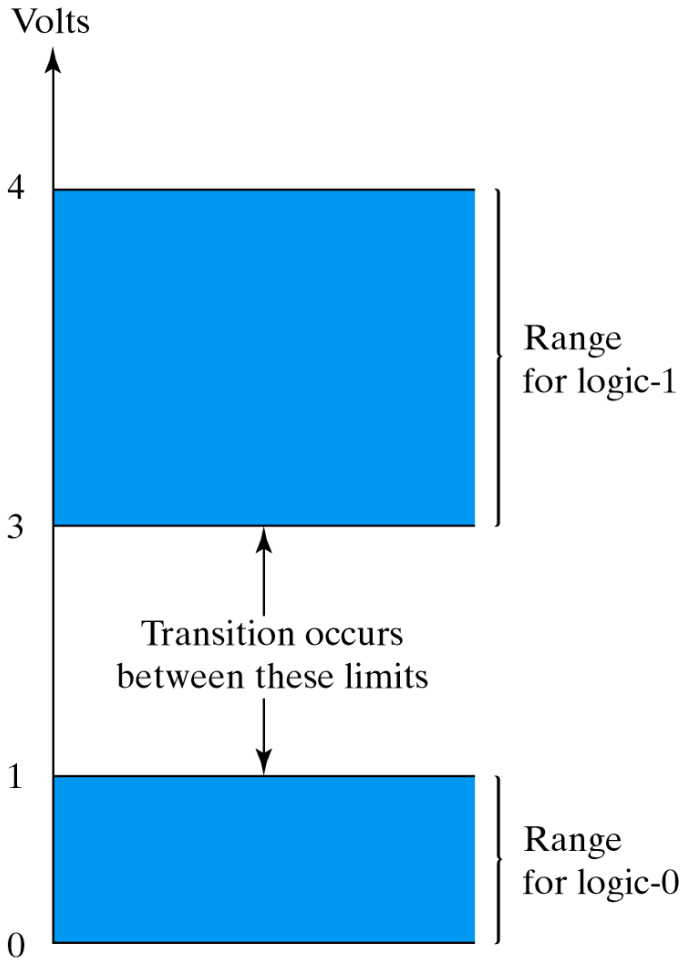
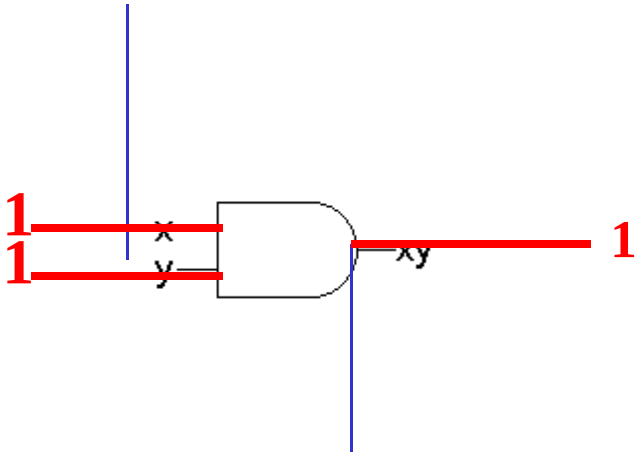


Fig. 1-3 Example of binary signals

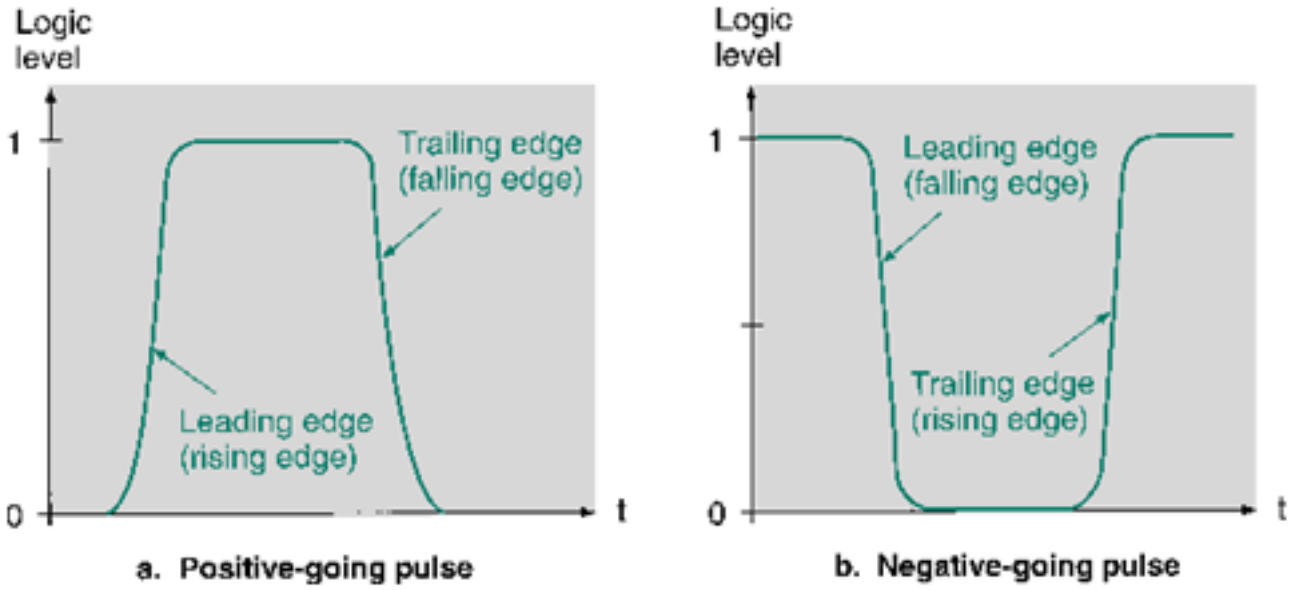
Example

2 input signals



1 output signal

Timing Diagram –Input and output signals



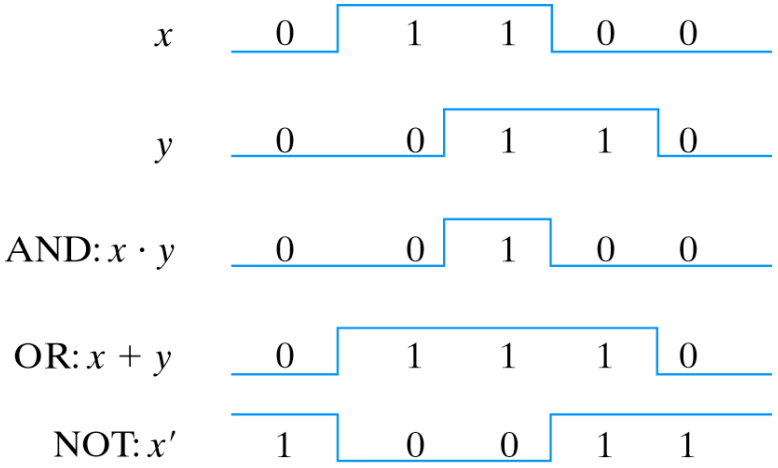


Fig. 1-5 Input-output signals for gates

Basic and Other Logic gates

- **Basic Logic gate**

  - AND
  - OR
  - NOT

  } These are called “fundamental logic gates” as all other gates and digital Circuits can be created from these gates.

- **Other Logic gates**

  - NAND
  - NOR

  } These are called “Universal logic gates” as any digital circuit can be designed by just using these gates

  - XOR
  - XNOR

The NAND & NOR Gates

• We can use a NAND and NOR gates to implement all three of the *basic operations* (AND,OR,NOT).

→ They are said to be **functionally complete**

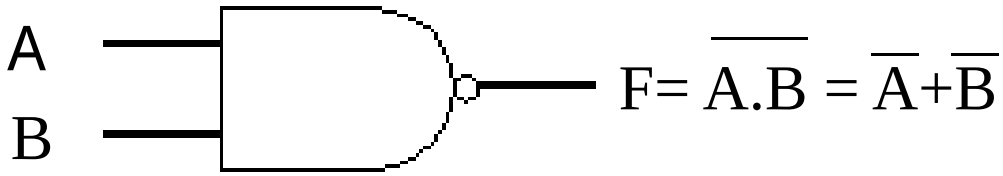
→ Both NAND and NOR gates are very valuable as any design can be realized using either one.

•It is easier to build digital circuits using all NAND or NOR gates than to combine AND,OR, and NOT gates.

•NAND/NOR gates are typically faster and cheaper to produce.

The NAND Gate

- The NAND gate is a combination of an AND gate followed by an inverter (NOT gate).
- We can use a NAND gate to implement all three of the *basic operations* (AND,OR,NOT).
- Such a gate is said to be **functionally complete**.

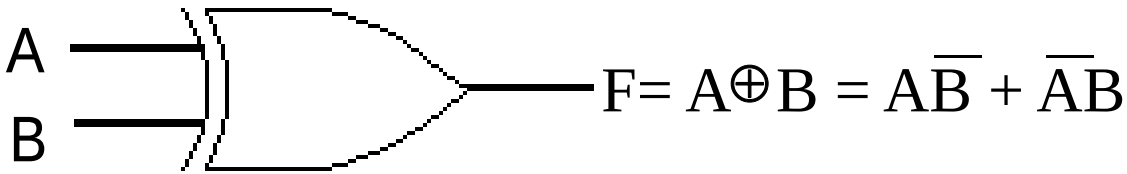


A	**B**	**F**
0	0	1
0	1	1
1	0	1
1	1	0

The XOR Gate (Exclusive-OR)

- This is a XOR gate.
- XOR gates assert their output when exactly one of the inputs is asserted, hence the name.
- The operator symbol for this operation is $\oplus$

  $1 \oplus 1 = 0$ and $1 \oplus 0 = 1$.

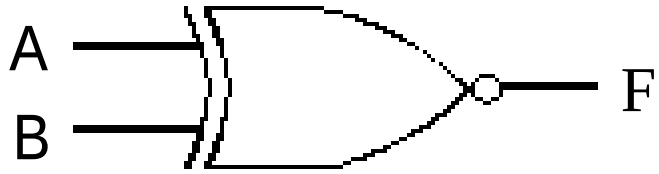


A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

The XNOR Gate

- This functions as an exclusive-NOR gate, or simply the complement of the XOR gate.
- The symbol for this operation is ⊙

  1 ⊙ 1 = 1 and 1 ⊙ 0 = 0.



$$F = \overline{A \oplus B} = (AB) + (\overline{A} \cdot \overline{B}) = AB + A'B'$$

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Key Terms and Review Points

- And Operation/Logic Gate
- Binary Logic
- Boolean Algebra
- Boolean Operation
- Boolean Variable
- Complement of a Variable/Gate
- Functional Completeness
- Logic Gate
- Nand Operation/Logic Gate
- Not Operation/Logic Gate
- Nor Operation/Logic Gate
- Or Operation/Logic Gate
- Xnor Operation/Logic Gate
- Xor Operation/Logic Gate

- References: Dr. Karmouch ITI 1100 Slides