

---

**EE40**  
**Lecture 35**  
**Prof. Chang-Hasnain**

12/5/07

Reading: Ch 7, Supplementary  
Reader

---

**Week 15**

- **OUTLINE**
  - Need for Input Controlled Pull-Up
  - CMOS Inverter Analysis
  - CMOS Voltage Transfer Characteristic
  - Combinatorial logic circuits
  - Logic
  - Binary representations
  - Combinatorial logic circuits
- **Reading**
  - Chap 7-7.5
  - Supplementary Notes Chapter 4

## Digital Circuits – Introduction

---

- Analog: signal amplitude is continuous with time.
- Digital: signal amplitude is represented by a restricted set of discrete numbers.
  - Binary: only two values are allowed to represent the signal: High or low (i.e. logic 1 or 0).
- Digital word:
  - Each binary digit is called a bit
  - A series of bits form a word
    - Byte is a word consisting of 8-bits
- Advantages of digital signal
  - Digital signal is more resilient to noise  $\diamond$  can more easily differentiate high (1) and low (0)
- Transmission
  - Parallel transmission over a bus containing n wires.
    - Faster but short distance (internal to a computer or chip)
  - Serial transmission (transmit bits sequentially)
    - Longer distance

## Analog vs. Digital Signals

---

- **Most** (but not all) **observables are analog**

*think of analog vs. digital watches*

**but the most convenient way to represent & transmit information electronically is to use digital signals**

*think of telephony*

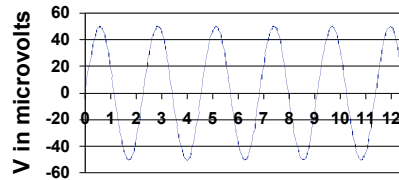
$\diamond$  Analog-to-digital (A/D) & digital-to-analog (D/A) conversion is essential (and nothing new)

*think of a piano keyboard*

## Analog Signal Example: Microphone Voltage

Voltage with normal piano key stroke

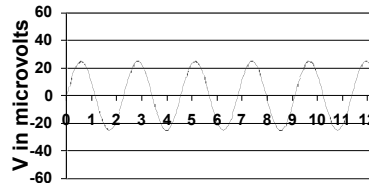
50 microvolt 440 Hz signal



t in milliseconds

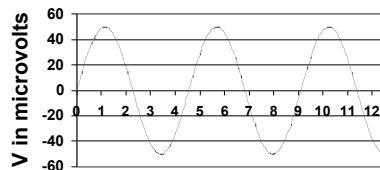
Voltage with soft pedal applied

25 microvolt 440 Hz signal



t in milliseconds

50 microvolt 220 Hz signal



t in milliseconds

← Analog signal representing piano key A, below middle C (220 Hz)

## Digital Signal Representations

**Binary numbers can be used to represent any quantity.**

We generally have to agree on some sort of “code”, and the dynamic range of the signal in order to know the form and the number of binary digits (“bits”) required.

**Example 1:** Voltage signal with maximum value 2 Volts

- Binary two (**10**) could represent a 2 Volt signal.
- To encode the signal to an accuracy of 1 part in 64 (1.5% precision), 6 binary digits (“bits”) are needed

**Example 2:** Sine wave signal of known frequency and maximum amplitude  $50 \mu\text{V}$ ;  $1 \mu\text{V}$  “resolution” needed.

## Decimal Numbers: Base 10

Digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Example:

$$3271 = (3 \times 10^3) + (2 \times 10^2) + (7 \times 10^1) + (1 \times 10^0)$$

This is a four-digit number. The left hand most number (3 in this example) is often referred as the most significant number and the right most the least significant number (1 in this example).

## Numbers: positional notation

- Number Base B  $\Rightarrow$  B symbols per digit:

–Base 10 (Decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

–Base 2 (Binary): 0, 1

- Number representation:

– $d_{31}d_{30} \dots d_1d_0$  is a 32 digit number

–value =  $d_{31} \times B^{31} + d_{30} \times B^{30} + \dots + d_1 \times B^1 + d_0 \times B^0$

- Binary: 0,1 (In binary digits called “bits”)

$$\begin{aligned} 11010 &= 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \\ &= 16 + 8 + 2 \\ &= 26 \end{aligned}$$

–Here 5 digit binary # turns into a 2 digit decimal #

## Hexadecimal Numbers: Base 16

---

- Hexadecimal:  
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F  
–Normal digits + 6 more from the alphabet
- Conversion: Binary ↔ Hex  
–1 hex digit represents 16 decimal values  
–4 binary digits represent 16 decimal values  
⇒1 hex digit replaces 4 binary digits

## Digital Signal Representations

---

**Binary numbers can be used to represent any quantity.**

We generally have to agree on some sort of “code”, and the dynamic range of the signal in order to know the form and the number of binary digits (“bits”) required.

**Example 1:** Voltage signal with maximum value 2 V and minimum of 0 V.

- Binary two (**10**) could represent a 2 Volt signal.
- To encode the signal to an accuracy of 1 part in 64 (1.5% precision), 6 binary digits (“bits”) are needed

**Example 2:** Sine wave signal of known frequency and maximum amplitude 50  $\mu\text{V}$ ; 1  $\mu\text{V}$  “resolution” needed.

## Resolution

---

- The size of the smallest element that can be separated from neighboring elements. The term is used to describe imaging systems, the frequency separation achieved by spectrometers, and so on.

## Decimal-Binary Conversion

---

- Decimal to Binary
  - Repeated Division By 2
    - Consider the number 2671.
  - Subtraction – if you know your  $2^N$  values by heart.

- Binary to Decimal conversion

$$\begin{aligned} 110001_2 &= 1x2^5 + 1x2^4 + 0x2^3 + 0x2^2 + 0x2^1 + 1x2^0 \\ &= 32_{10} + 16_{10} + 1_{10} \\ &= 49_{10} \\ &= 4x10^1 + 9x10^0 \end{aligned}$$

## Example 2 (continued)

Possible digital representation for the sine wave signal:

Analog representation: Amplitude in $\mu\text{V}$	Digital representation: Binary number
1	000001
2	000010
3	000011
4	000100
5	000101
8	001000
16	010000
32	100000
<b>50</b>	<b>110010</b>
63	111111

## Binary Representation

- N bit can represent  $2^N$  values: typically from 0 to  $2^N-1$ 
  - 3-bit word can represent 8 values: e.g. 0, 1, 2, 3, 4, 5, 6, 7
- Conversion
  - Integer to binary
  - Fraction to binary ( $13.5_{10}=1101.1_2$  and  $0.392_{10}=0.011001_2$ )
- Octal and hexadecimal

- 
- Logic gates
    - Combine several logic variable inputs to produce a logic variable output
  - Memory
    - Memoryless: output at a given instant depends the input values of that instant.
    - Momory: output depends on previous and present input values.

## Boolean algebras

---

- Algebraic structures
  - "capture the essence" of the logical operations AND, OR and NOT
  - corresponding set for theoretic operations intersection, union and complement
  - named after George Boole, an English mathematician at University College Cork, who first defined them as part of a system of logic in the mid 19th century.
  - Boolean algebra was an attempt to use algebraic techniques to deal with expressions in the propositional calculus.
  - Today, Boolean algebras find many applications in electronic design. They were first applied to switching by Claude Shannon in the 20th century.

## Boolean algebras

---

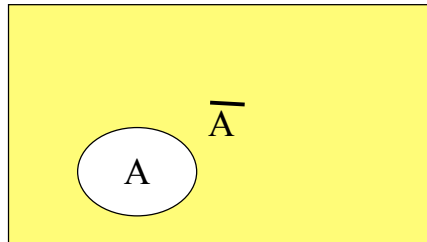
- The operators of Boolean algebra may be represented in various ways. Often they are simply written as AND, OR and NOT.
- In describing circuits, NAND (NOT AND), NOR (NOT OR) and XOR (eXclusive OR) may also be used.
- Mathematicians often use + for OR and · for AND (since in some ways those operations are analogous to addition and multiplication in other algebraic structures) and represent NOT by a line drawn above the expression being negated.

## Boolean Algebra

---

- NOT operation (inverter)  $A \cdot \bar{A} = 0$
- AND operation  $A + \bar{A} = 1$   
 $A \cdot A = A$   
 $A \cdot 1 = A$   
 $A \cdot 0 = 0$   
 $A \cdot B = B \cdot A$
- OR operation  $(A \cdot B) \cdot C = A \cdot (B \cdot C)$   
 $A + A = A$   
 $A + 1 = 1$   
 $A + 0 = A$   
 $A + B = B + A$   
 $(A + B) + C = A + (B + C)$

## Graphic Representation



$$A \cdot \bar{A} = 0$$

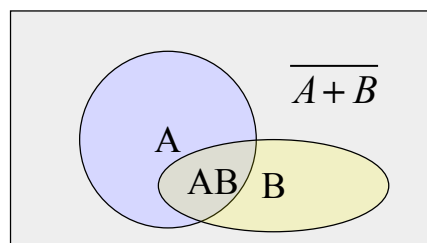
$$A + \bar{A} = 1$$

Full square = complete set = 1

Yellow part = NOT(A) =  $\bar{A}$

White circle = A

## Graphic Representation



$$A \oplus B = \bar{A}B + A\bar{B} = (A+B) \cdot \overline{(A+B)} = \overline{A \cdot B + \bar{A} \cdot \bar{B}}$$

Exclusive OR = yellow and blue part –

intersection/overlap part

= exactly when only one of the input is true

## Boolean Algebra

---

- Distributive Property

$$A \cdot (B + C) = A \cdot B + A \cdot C$$

$$(A + B) \cdot C = (A + B) \cdot (A + C)$$

- De Morgan's laws

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$

- An excellent web site to visit

– [http://en.wikipedia.org/wiki/Boolean\\_algebra](http://en.wikipedia.org/wiki/Boolean_algebra)

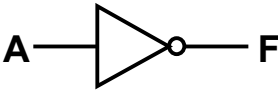
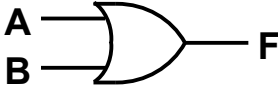

## Examples

---

$$F = A \cdot \overline{B} \cdot C + A \cdot B \cdot C + (C + D) \cdot (\overline{D} + E)$$

$$F = C \cdot (A + \overline{D} + E) + D \cdot E$$

## Logic Functions, Symbols, & Notation



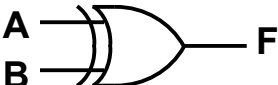
NAME	SYMBOL	NOTATION	TRUTH TABLE															
“NOT”		$F = \bar{A}$	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	
“OR”		$F = A+B$	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
“AND”		$F = A \cdot B$	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																

EE40 Fall

Slide 23

Prof. Chang-Hasnain

## Logic Functions, Symbols, & Notation 2

“NOR”		$F = \overline{A+B}$	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	F	0	0	1	0	1	0	1	0	0	1	1	0
A	B	F																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
“NAND”		$F = \overline{A \cdot B}$	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	F	0	0	1	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
“XOR” (exclusive OR)		$F = A \oplus B$	<table border="1" style="margin: auto;"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	0
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	0																

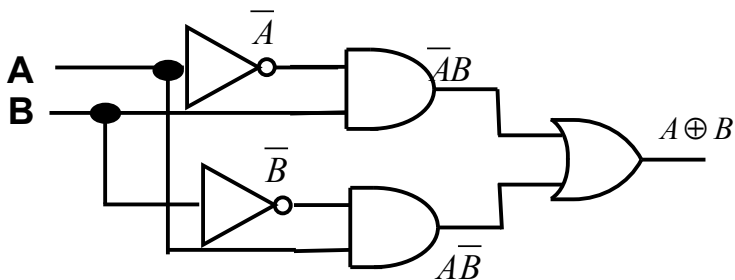
EE40 Fall

Slide 24

Prof. Chang-Hasnain

## Circuit Realization

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



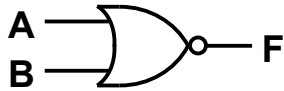
## Logic Functions, Symbols, & Notation

<u>NAME</u>	<u>SYMBOL</u>	<u>NOTATION</u>	<u>TRUTH TABLE</u>															
“NOT”		$F = \overline{A}$	<table border="1"> <thead> <tr> <th>A</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> </tr> </tbody> </table>	A	F	0	1	1	0									
A	F																	
0	1																	
1	0																	
“OR”		$F = A+B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	F	0	0	0	0	1	1	1	0	1	1	1	1
A	B	F																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
“AND”		$F = A \cdot B$	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>F</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	F	0	0	0	0	1	0	1	0	0	1	1	1
A	B	F																
0	0	0																
0	1	0																
1	0	0																
1	1	1																

## Logic Functions, Symbols, & Notation 2

---

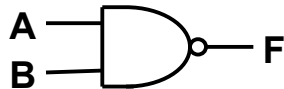
“NOR”



$$F = \overline{A+B}$$

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

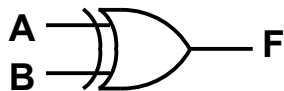
“NAND”



$$F = \overline{A \cdot B}$$

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

“XOR”  
(exclusive OR)



$$F = A \oplus B$$

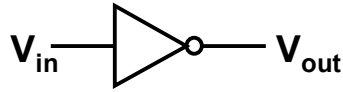
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

## Fan in/Fan out

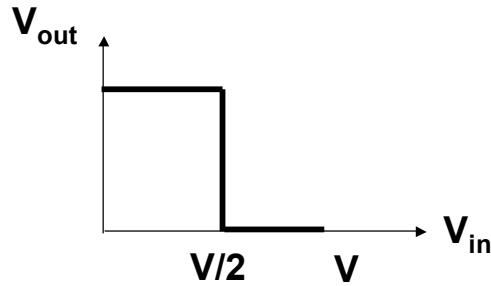
---

- Complex digital operations are formed with a variety of gates interconnected to yield the desired logic function.
- Sometimes a number of inputs are connected to one gate input and output of a gate may be connected to a number of gates.
- Fan-in: the maximum number of logic gates that can be connected at the input of a gate **without altering its performance**.
- Fan-out: the maximum number of logic gates that can be connected to the output of a gate **without altering its performance**.
- Typical fan-in and fan-out numbers are 3.

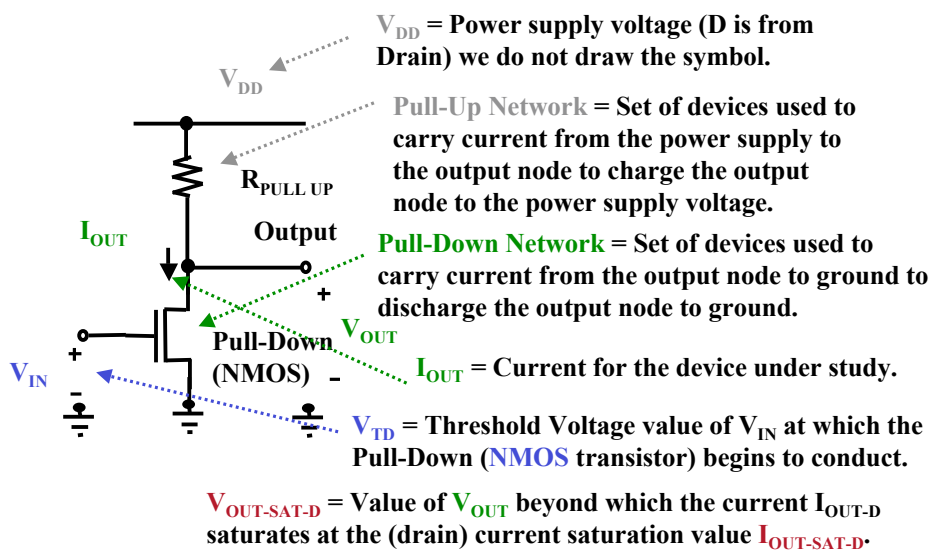
## Inverter = NOT Gate



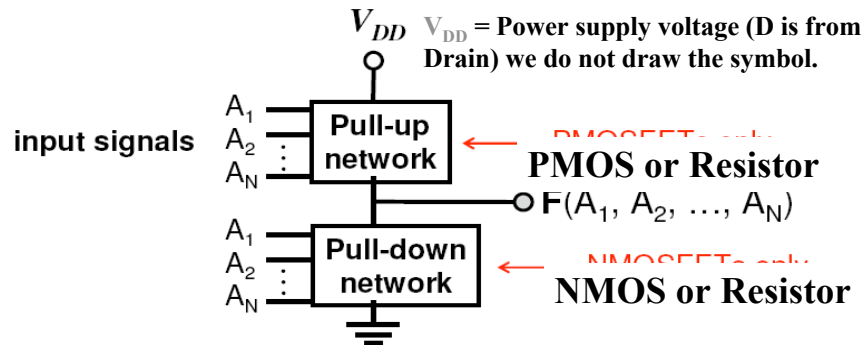
Ideal Transfer Characteristics



## Terminology for a Logic Circuit



## Pull-Up and Pull-Down



Pull-up current	$V_F$	logic state of F	Pull-down current	$V_F$	logic state of F
0	$V_{DD}$	1	Non-zero	Some value	1
Non-zero	Some value	0	0	GND	0