Lecture 4
Signed Binary Numbers, BCD, and Gray Code

Hussein Alsheakh, PhD

Quiz 2

Perform the subtraction $X - Y$

$X = 1010100$

$Y = 1000011$

Signed Binary Numbers

- To represent negative integers, we need a notation for negative values.
- It is customary to represent the sign with a bit placed in the leftmost position of the number since binary digits.
- The convention is to make the sign bit 0 for positive and 1 for negative.
- Example:

Signed-magnitude representation:	10001001
Signed-1's-complement representation:	11110110
Signed-2's-complement representation:	11110111

- All possible four-bit signed binary numbers in the three representations.

Signed Binary Numbers

Table 1.3
Signed Binary Numbers

Decimal	Signed-2's Complement	Signed-1's Complement	Signed Magnitude
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000		—

Signed Binary Numbers

- Arithmetic addition
  - The addition of two numbers in the signed-magnitude system follows the rules of ordinary arithmetic.
  - **If the signs are the same**, we add the two magnitudes and give the sum the common sign.
  - **If the signs are different**, we subtract the smaller magnitude from the **larger** and **give the difference the sign if the larger magnitude**.
  - The addition of two signed binary numbers with **negative numbers** represented in **signed-2's-complement** form is obtained from the **addition of** the two numbers, **including their sign bits**.
  - A carry out of the sign-bit position is discarded.
- Example:

+ 6	00000110	− 6	11111010
+13	00001101	+13	00001101
+ 19	00010011	+ 7	00000111
+ 6	00000110	− 6	11111010
−13	11110011	−13	11110011
− 7	11111001	−19	11101101

Signed Binary Numbers

- Arithmetic Subtraction
  - 1. Take the 2's complement of the subtrahend (including the sign bit) and add it to the minuend (including sign bit).
    2. A carry out of sign-bit position is discarded.

$$(\pm A)-(+B)=(\pm A)+(-B)$$
$$(\pm A)-(-B)=(\pm A)+(+B)$$

- Example:

$(-6)-(-13)$ → $(11111010-11110011)$

→ $(11111010+00001101)$

→ $00000111\ (+7)$

Binary Codes Decimal (BCD)

Why BCD?

Binary-coded decimal is useful in digital displays, it can be difficult to manipulate or display large numbers. Since binary-coded decimal treats each digit as a separate subcircuit, data becomes easier.

Thus BCD is suitable for Computer applications, digital communications.

- Decimal 396 is represented in BCD with 12bits as 0011 1001 0110, with each group of 4 bits representing one decimal digit.
- A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9.
- The binary combinations 1010 through 1111 are not used and have no meaning in BCD.

Table 1.4
Binary-Coded Decimal (BCD)

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Binary Code

- Example: (*check if the sum exceeds 9 or has a carry)
  - Consider decimal 185 and its corresponding value in BCD and binary:

$$(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2$$

- BCD addition

4	0100	4	0100	8	1000
+5	+0101	+8	+1000	+9	+1001
9	1001	12	1100	17	10001
			+0110		+0110
			10010		10111

If sum >9 or any Carry
Add 6 = 0110

Hex	Binary
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

Binary Code

- Example: (*check if the sum exceeds 9 or has a carry)
  - Consider the addition of 184 + 576 = 760 in BCD:

BCD	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

- Decimal Arithmetic: (+375) + (-240) = +135

Hex	Binary
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

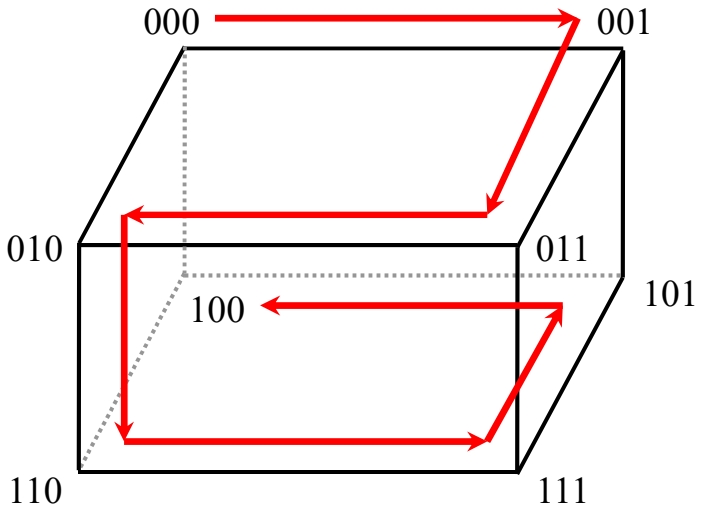
BCD Conversion

Decimal Digit	(B C D) 8 4 2 1	Excess-3 BCD+0011
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

- **Ex** Convert (13) 10 to Binary , BCD code?
  Sol:
  - (13) 10 = (1 1 0 1) 2
  - (13) 10 = (0 0 0 1 0 0 1 1) BCD
  Ex Decode the following BCD numbers?
  1 – (1 0 0 0 1 1 1 1 0 0 0 1 0 0 1 0 1 0 1) BCD
  2 – (1 1 0 1 1 0 . 0 1 1 0 1) BCD
  Sol :
  1 – (0 1 0 0 0 1 1 1 1 0 0 0 1 0 0 1 0 1 0 1) BCD = (4 7 8 9 5) 10
  Ex Encode the following numbers into BCD code?
  1 – (1 1 1 0 0 1 1) 2 2 – (7 6 4 8) 12
  Solution
  1 – 1 1 1 0 0 1 1 = 1 x 2^0 + 1 x 2^1 + 0 x 2^2 + 0 x 2^3 + 1 x 2^4 + 1 x 2^5 + 1 x 2^ 6
  = 1 + 2 + 0 + 0 + 16 + 32 + 64
  = (115) 10
  (115) 10 = (0001 0001 0101) BCD
  Ex What are the Ex-3 of the following numbers?
  1 – (3 6 5) 10 2 – (1 1 0 1 0 1 0 1 0) 2
  Sol: 3 6 5
  +3 +3 +3
  6 9 8 = (0110 1001 1000)

Gray Code

- The advantage is that only bit in the code group changes in going from one number to the next.
  - Error detection.
  - Representation of analog data.
  - Low power design.



ray ode	Decimal Equivalent
000	0
001	1
011	2
010	3
110	4
111	5
101	6
100	7

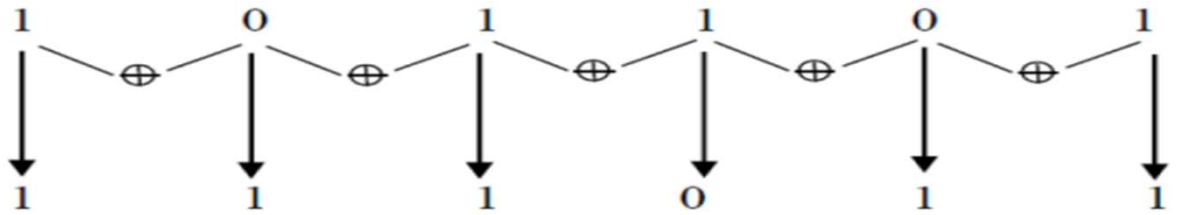
Gray Codes

- To convert a number from binary to Gray code the relations must be known

$$0 \oplus 0 = 0 \quad 0 \oplus 1 = 1 \quad 1 \oplus 1 = 0 \quad 1 \oplus 0 = 1$$

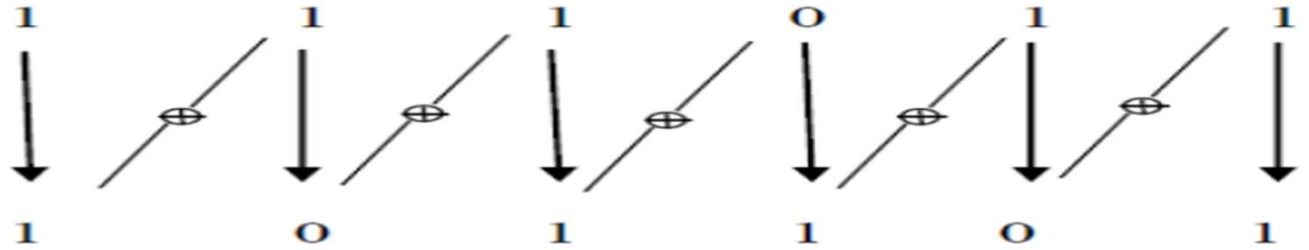
Ex Convert the binary number (1 0 1 1 0 1)2 into Gray Code?

Solution:



$(1\ 0\ 1\ 1\ 0\ 1)_2 = (1\ 1\ 1\ 0\ 1\ 1)_G$

لأرجاع الزقم|من صيغة Gray الى ال Binary يتم انزال اول bit من جهة اليسار وتطبيق العلاقات بين الناتج والزقم الذي يليه



Character (Alphanumeric) Codes

- Alphanumeric codes are the codes that represent numbers and alphabetic characters. The following three alphanumeric codes are very commonly used for the data representation:

  - American Standard Code for Information Interchange (ASCII).
  - Extended Binary Coded Decimal Interchange Code (EBCDIC).
  - Five bit Baudot Code.

ASCII Character Codes

- American Standard Code for Information Interchange
- A popular code used to represent information sent as character-based data.
- It uses 7-bits to represent:
  - 94 Graphic printing characters.
  - 34 Non-printing characters.
- Some non-printing characters are used for text format (e.g. BS = Backspace, CR = carriage return).
- Other non-printing characters are used for record marking and flow control (e.g. STX and ETX start and end text areas).

ASCII Properties

- ASCII has some interesting properties:
  - Digits 0 to 9 span Hexadecimal values 30_{16} to 39_{16}
  - Upper case A-Z span 41_{16} to $5A_{16}$
  - Lower case a-z span 61_{16} to $7A_{16}$
    - Lower to upper case translation (and vice versa) occurs by **flipping bit 6**.

- Complement bit 6

A: 100 0001

a: 110 0001

Ex: What are the characters corresponding to the ASCII code?

(0100101010011111001000100111001000001000100100001011010110)$_{\text{ASCII}}$

Sol

(1001010 1001111 1001000 1001110 0100000 1000100 1000101 1010110) $_{\text{ASCII}}$

74=J 79= O 72=H 78= N 32=space 68=D 69=E 86=V

= JOHN DEV

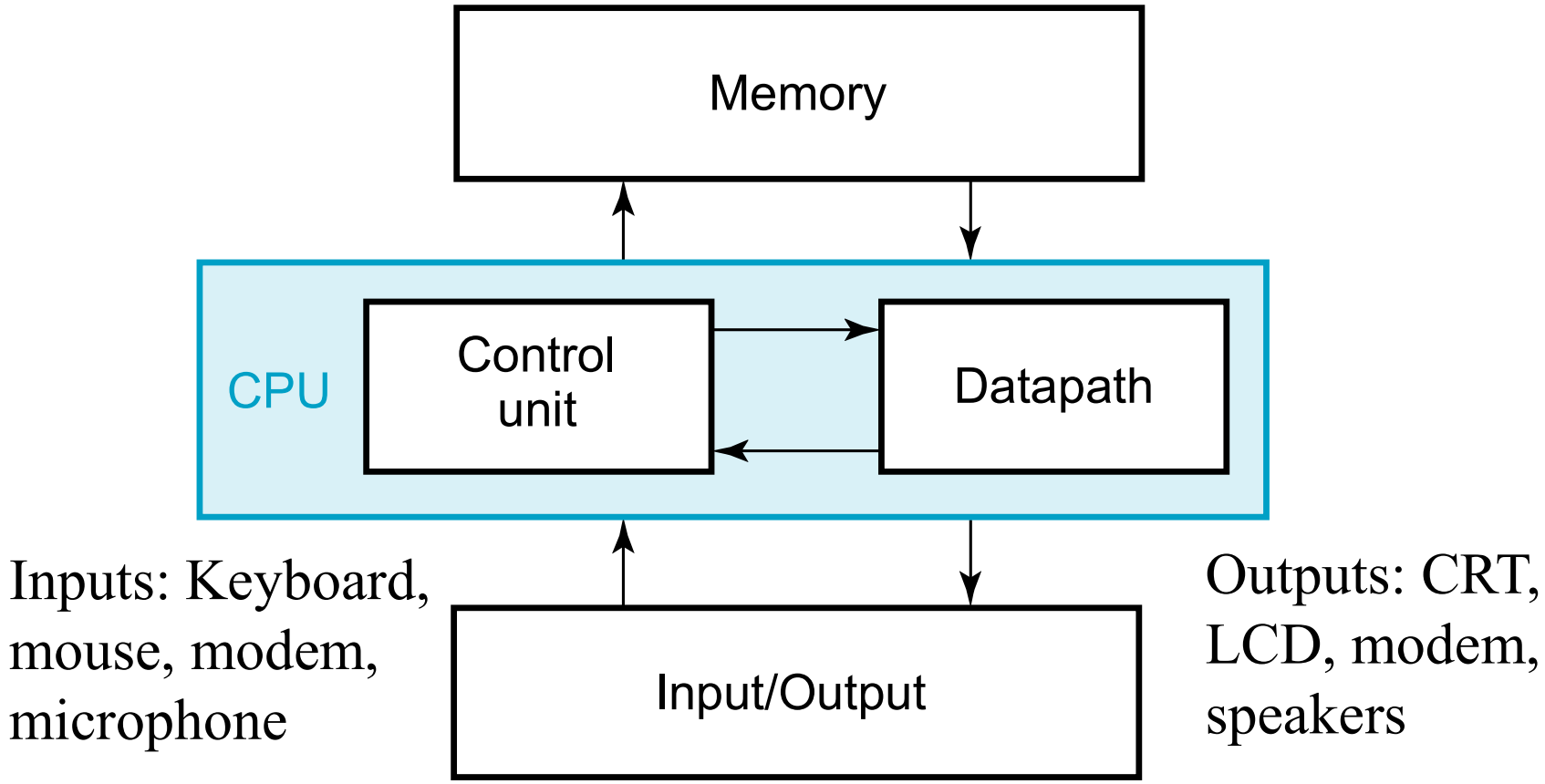
Binary Storage and Registers

- Registers
  - A binary cell is a device that possesses two stable states and is capable of storing one of the two states.
  - A register is a group of binary cells. A register with *n* cells can store any discrete quantity of information that contains *n* bits.

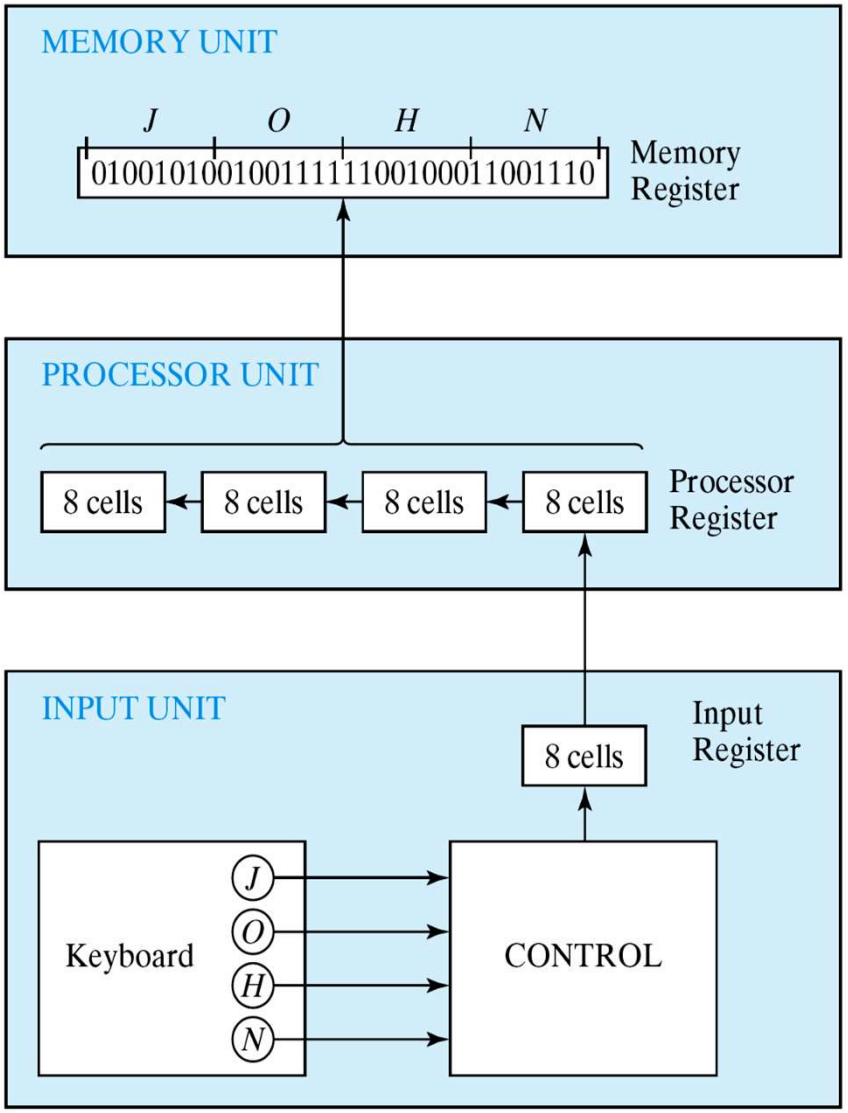
    n cells → 2^n possible states

- A binary cell
  - Two stable state
  - Store one bit of information
  - Examples: flip-flop circuits, ferrite cores, capacitor
- A register
  - A group of binary cells (8 bit)
- Register Transfer
  - A transfer of the information stored in one register to another.
  - One of the major operations in digital system.

A Digital Computer Example



Transfer of information



Transfer of information among register

Binary Logic

- Definition of Binary Logic
  - Binary logic consists of binary variables and a set of logical operations.
  - The variables are designated by letters of the alphabet, such as *A*, *B*, *C*, *x*, *y*, *z*, etc, with each variable having two and only two distinct possible values: 1 and 0,
  - Three basic logical operations: AND, OR, and NOT.

1. AND: This operation is represented by a dot or by the absence of an operator. For example, $x \cdot y = z$ or $xy = z$ is read "x AND y is equal to z," The logical operation AND is interpreted to mean that $z = 1$ if only $x = 1$ and $y = 1$; otherwise $z = 0$. (Remember that x, y, and z are binary variables and can be equal either to 1 or 0, and nothing else.)
2. OR: This operation is represented by a plus sign. For example, $x + y = z$ is read "x OR y is equal to z," meaning that $z = 1$ if $x = 1$ or $y = 1$ or if both $x = 1$ and $y = 1$. If both $x = 0$ and $y = 0$, then $z = 0$.
3. NOT: This operation is represented by a prime (sometimes by an overbar). For example, x' = z (or $\bar{x} = z$) is read "not x is equal to z," meaning that z is what z is not. In other words, if $x = 1$, then $z = 0$, but if $x = 0$, then $z = 1$, The NOT operation is also referred to as the complement operation, since it changes a 1 to 0 and a 0 to 1.

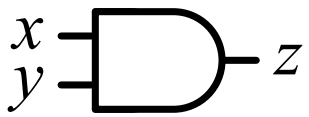
Binary Logic

- Truth Tables, Boolean Expressions, and Logic Gates

AND

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

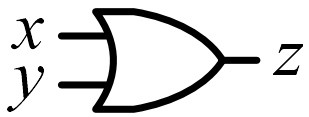
$$z = x \bullet y = x\,y$$



OR

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

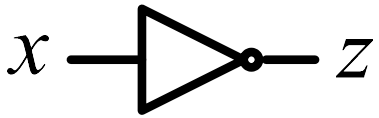
$$z = x + y$$



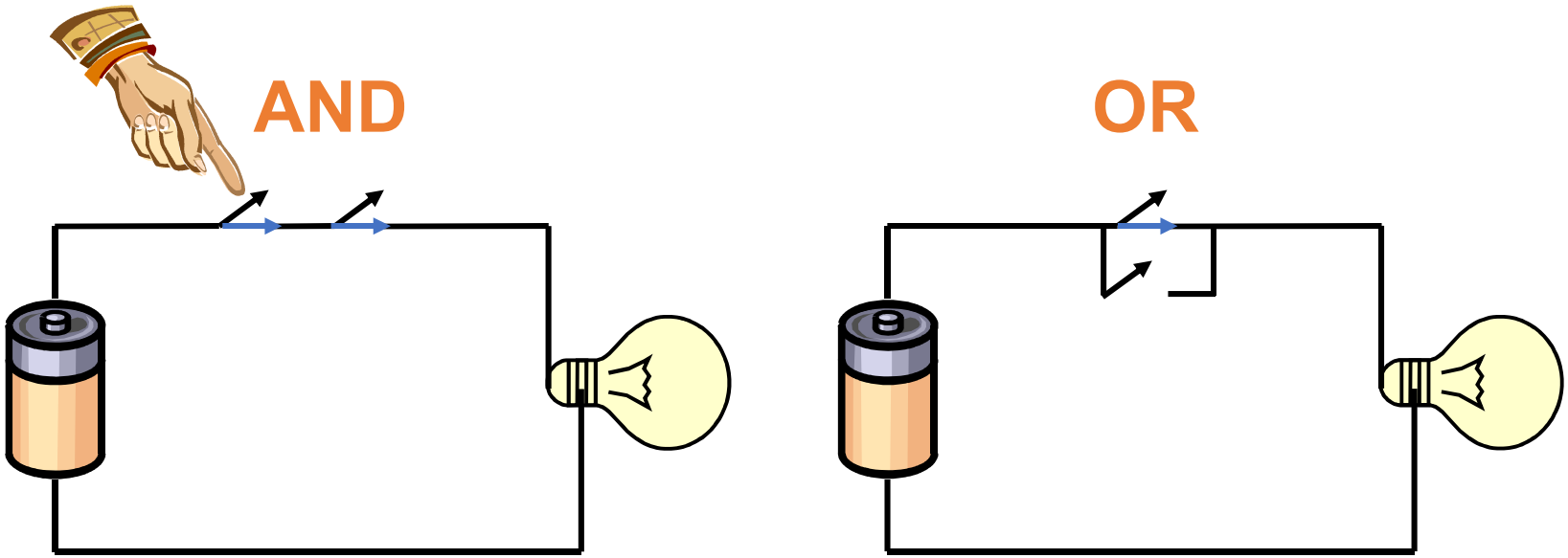
NOT

x	z
0	1
1	0

$$z = \overline{x} = x'$$

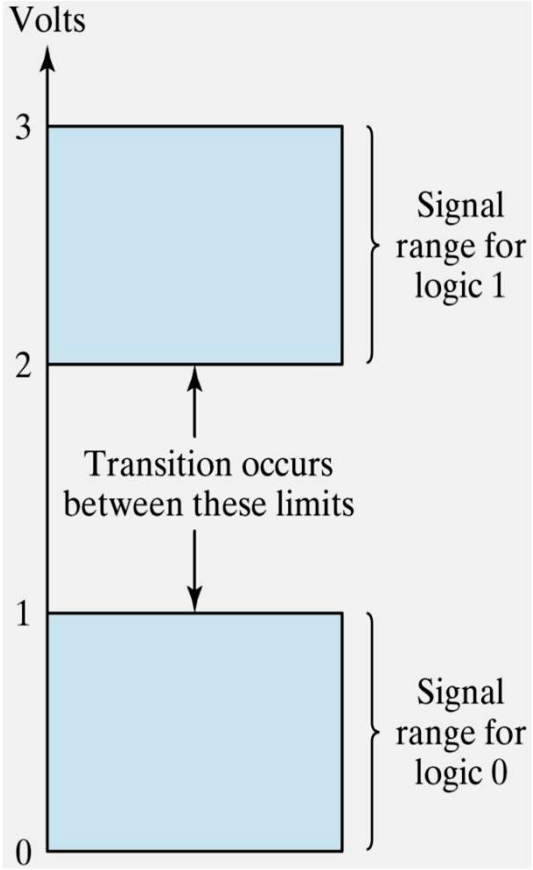


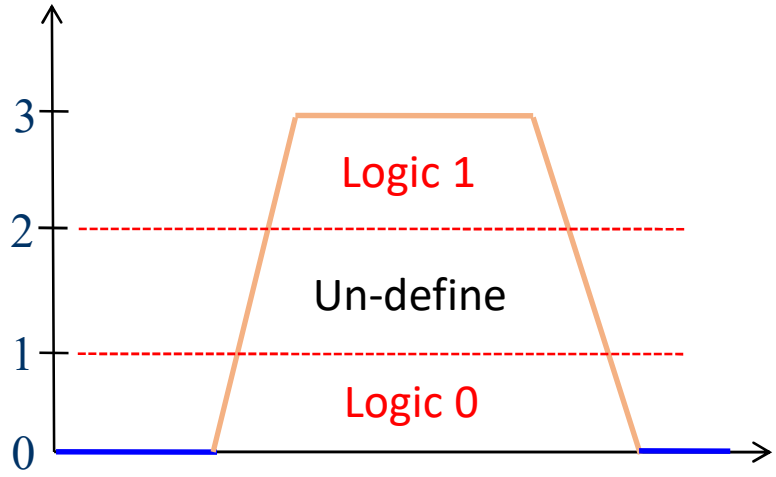
Switching Circuits



Binary Logic

- Logic gates
  - Example of binary signals

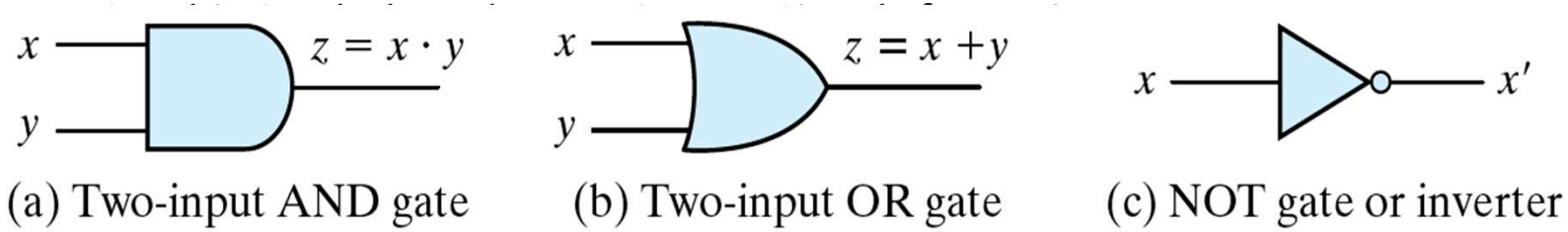




Example of binary signals

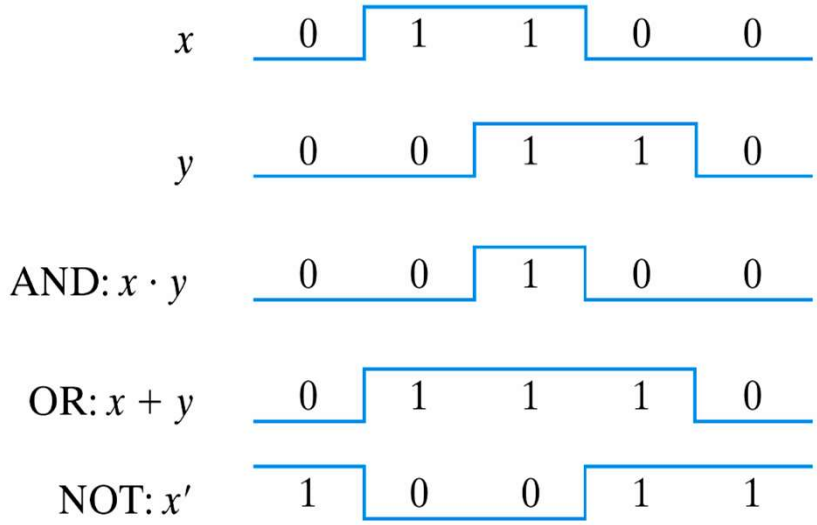
Binary Logic

- Logic gates



(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter

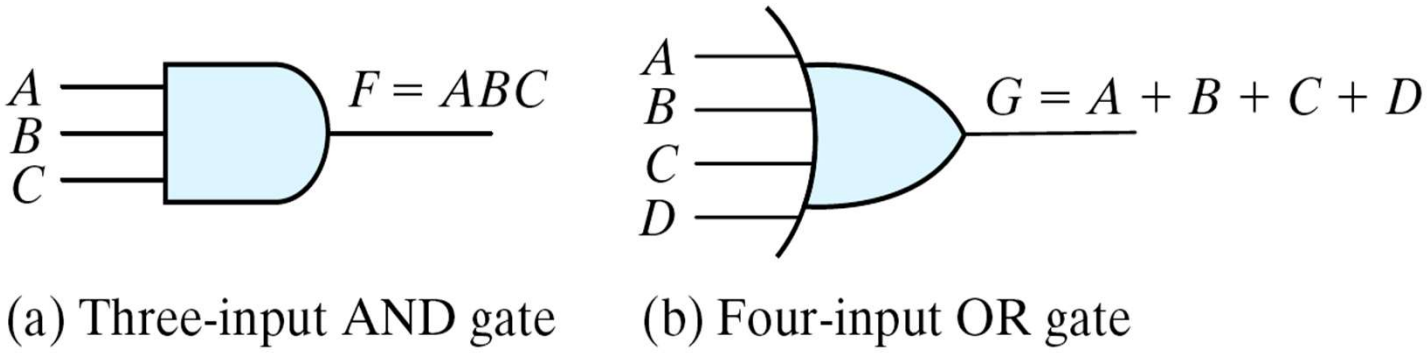
Fig. 1.4 Symbols for digital logic circuits



Input-Output signals for gates

Binary Logic

- Logic gates
  - Graphic Symbols and Input-Output Signals for Logic gates:



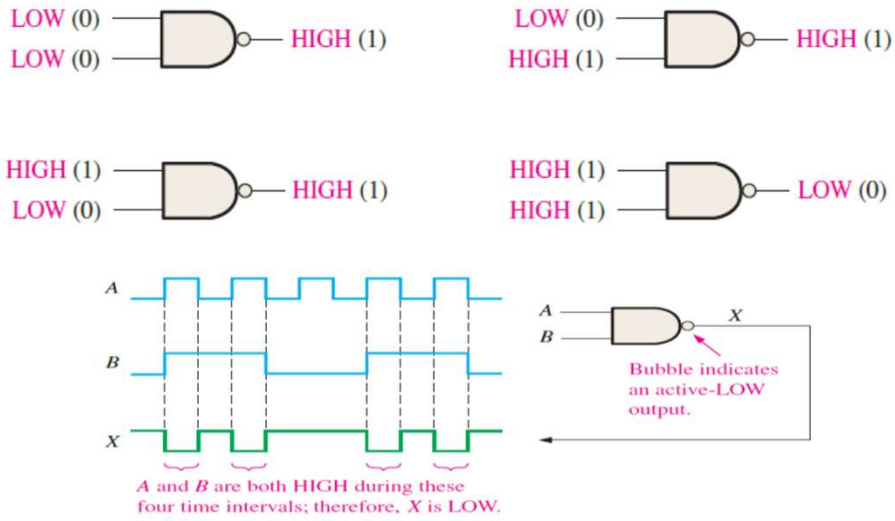
(a) Three-input AND gate (b) Four-input OR gate

Gates with multiple inputs

The NAND Gate

- The NAND gate is a popular logic element because it can be used as a **Universal Gate**; that is, NAND gates can be used in combination to perform the AND, OR, and inverter operations.

- The term *NAND* is a contraction of NOT-AND and implies an AND function with a complemented (inverted) output.



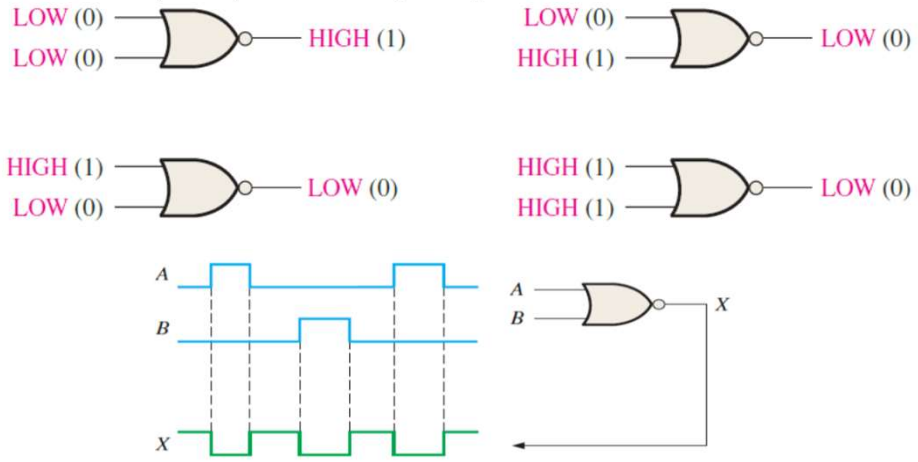
Inputs		Output
A	*B*	*X*
0	0	1
0	1	1
1	0	1
1	1	0

1 = HIGH, 0 = LOW.

The NOR Gate

- The NOR gate, like the NAND gate, is a useful logic element because it can also be used as a universal gate; that is, NOR gates can be used in combination to perform the AND, OR, and inverter operations.

- The term NOR is a contraction of NOT-OR and implies an OR function with an inverted (complemented) output.

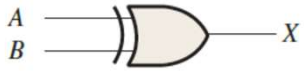


Inputs		Output
A	*B*	*X*
0	0	1
0	1	0
1	0	0
1	1	0

1 = HIGH, 0 = LOW.

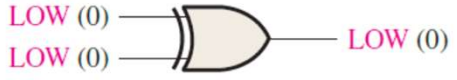
The Exclusive-OR Gate

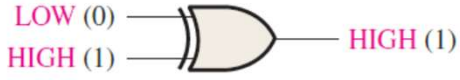
- The exclusive-OR gate performs modulo-2 addition. Standard symbols for an exclusive-OR (XOR for short) gate and The Boolean expression for the output of a 2-input XOR gate can be written as:

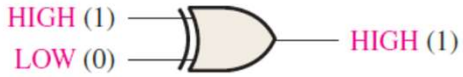


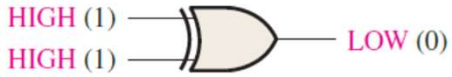
$$X=\overline{A}B + A\overline{B} = A \oplus B$$

- The XOR gate has only two inputs. The four possible input combinations and the resulting outputs for an XOR gate. The operation of an XOR gate is summarized in the truth table shown in the Table below.





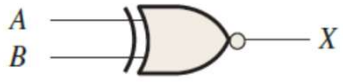




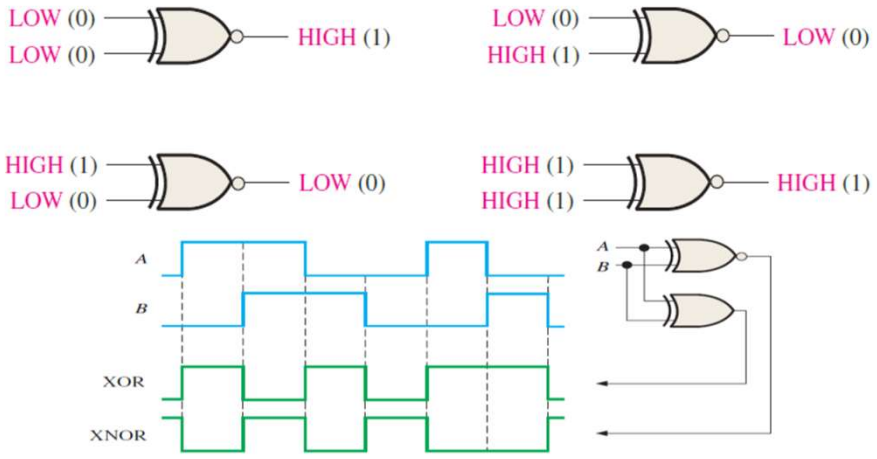
Inputs		Output
A	***B***	***X***
0	0	0
0	1	1
1	0	1
1	1	0

The Exclusive-NOR Gate

- The bubble on the output of the XNOR symbol indicates that its output is opposite that of the XOR gate. Standard symbols for an exclusive-NOR (XNOR) gate and The Boolean expression for the output of a 2-input XNOR gate can be written as:



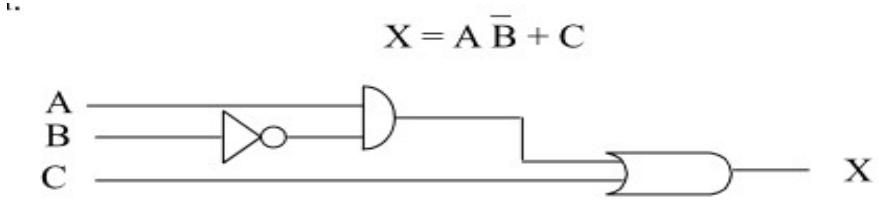
$$X = \overline{A}\,\overline{B} + AB = \overline{A + B}$$



Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Networks and Expressions

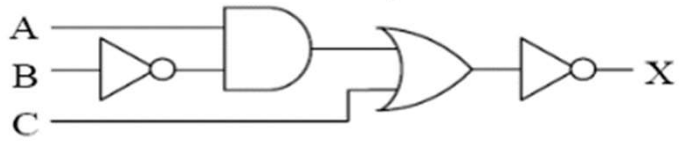
- Combining AND gates, OR gates and inverters, we can construct a logic network representing a given Boolean expression. The logic network for the Boolean expression:



- Home work: Draw the logic circuit represented by each of the following expressions:
- 1- A + B + C
- 2- ABC
- 3- AB + CD
- 4- AB(C + D)

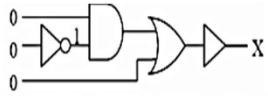
Truth Tables for Combinational Logic

- Not all digital circuits lend themselves to quick conversion to a truth table. For example, input B in the digital circuit shown in Figure passes through four logic gates before its effect is seen at the

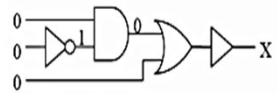


- So, how do we convert this circuit to a truth table?
- One method is to go through each pattern of ones and zeros at the input and fill in the resulting output in the truth table row by row. Figure takes A=0, B=0, and C=0 through each gate to determine its corresponding output.

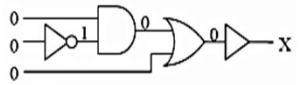
a.) A 0 is input to the first inverter which outputs a 1.



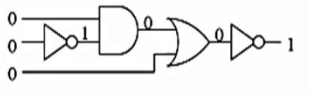
b.) The 1 coming from the inverter is combined with a 0 in the AND gate to output a 0.

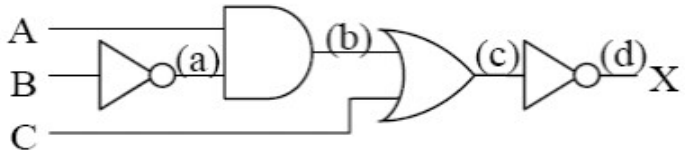


c.) The OR gate receives a 0 from the AND and a 0 from the inputs which makes it output a 0.



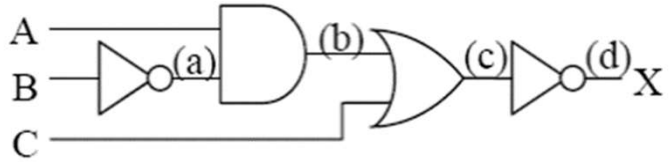
d.) The 0 output from the OR gate passes through the inverter output a 1.





Truth Tables for Combinational Logic Cont.

- Figure represents only one row of a truth table with eight rows.
- Add another input and the truth table doubles in size to sixteen rows.
- There is another way to determine the truth table.
- In Figure, we took the inputs through a sequence of steps
- first through the inverter connected to the input B,
- then through the AND gate,
- then through the OR gate,



- and lastly through the inverter connected to the output of the OR gate.
- These steps are labeled (a), (b), (c), and (d) in Figure 2.8.

Truth Tables for Combinational Logic Cont.

- Begin by creating the input columns of the truth table listing all of the possible combinations of ones and zeros for the inputs. In the case of our sample circuit, that gives us a truth table with eight rows.
- Next, add a column for each layer of logic.

A	B	C	(a)=Not of B	(b)=(a) AND A	(c)=(b)OR C	X=(d)=NOT C
0	0	0	1	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	1	1	1	0
1	0	1	1	1	1	0
1	1	0	0	0	0	1
1	1	1	0	0	1	0

Home Work:

- **Develop the truth table of the combinational logic circuit shown below?**