Digital Design

Dr. Amer Abu-Jassar
Faculty of Science and Information Technology

Textbook

Digital Design,

Fourth Edition,

M. Morris Mano

Grading Policy

- **First Exam:** **20%**
- **Second Exam:** **20%**
- **Final Exam:** **50%**

Digital Design

- Ch1: Digital Systems and Binary Numbers
- Ch2: Boolean algebra and Logic Gates
- Ch3: Gate-level Minimization
- Ch4: Combinational Logic
- Ch5: Synchronous Sequential Logic
- Ch6: Registers and Counters

Lecture1

Ch1: Digital Systems and Binary Numbers

الانظمة الرقمية والأعداد الثنائية

Outline

- Digital Systems
- Binary Numbers
- Number-Base Conversion
- Octal and Hexadecimal Number
- Signed Binary Numbers
- Binary Codes
- Binary Storage and Registers
- Binary Logic

Digital Systems

الانظمة الرقمية

- Discrete information - processing systems
- Manipulate discrete elements of information represented in binary form (digits)

- أنظمة تشغيلِ البيانات المنفصلة
- تعالج العناصر المنفصلة من المعلومات الممثلة بالشكل الثنائي 0, 1

Digital Systems are used in:

- Communication
- Business transactions
- Traffic control
- Space guidance
- Medical treatment
- Weather monitoring
- Internet
- Commercial
- Industrial
- Scientific enterprises

- الاتصالات
- المعاملات التجارية
- السيطرة على حركة المرور
- توجيه الفضاء
- العلاج الطبي
- رصد الأحوال الجوية
- الإنترنت
- التجارية الصناعية
- المؤسسات العلمية

Digital systems examples

- Telephone switching exchanges
- Digital camera
- Digital versatile discs (DVD), digital information representing video, audio
- Digital TV

Digital computers

- General purposes , أغراض العامة
- Many scientific, industrial and commercial applications.

العديد من التطبيقات العلمية و الصناعية و التجارية

A Digital Computer Example

- Digital computer is the best example of digital system
  - ✓ Follow a sequence of instructions (program)

    اتباع سلسلة من التعليمات (البرنامج)
  - ✓ Operates of given data ,

    تعمل من البيانات الواردة

Digital computers …cont.

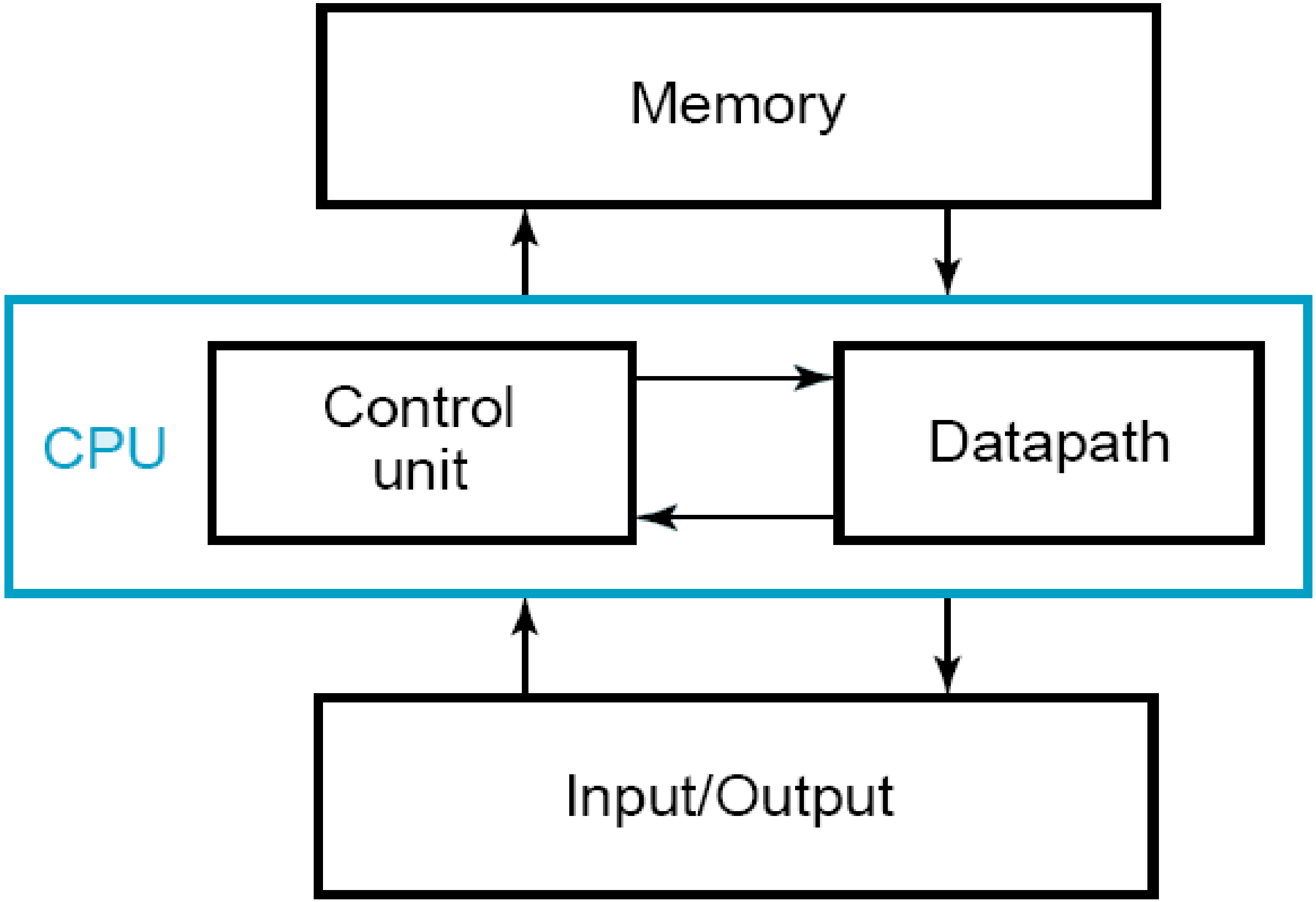
- Commutation unit:

  it Provides interaction with other users through internet.

- Memory unit:

  it stores programs and information.

- CPU:
  - is an integrated circuit IC consists of millions of transistors to produce complex functions.
  - Manipulate discrete elements of information (digits)
  - It performs the arithmetic computations and logical operations.



Synchronous or Asynchronous?

Signal

- Signal: an information variable represented by physical quantity (Voltage).
  - الجهد، والمعروف باسم فرق الجهد الكهربائي (تقاس في فولت) هو الفرق في الجهد الكهربائي بين نقطتين (هو خاصية الكائن التي يمكن قياسها بأداة القياس مثل قياس شدة التيار)

- Discrete elements represented in a digital system by physical quantities called <u>signals</u>

- In digital systems:

  The variable takes on digital values:
  - Two levels,
  - or binary values are the most prevalent values

Binary values

- Binary values are represented abstractly by:
  - digits 0 and 1
  - words (symbols) False (F) and True (T)
  - words (symbols) Low (L) and High (H)
  - words On and Off.

- The signals in digital system use two discrete values called binary. (0, 1)

- A binary digit called bit, has two values: 0 and 1

- Discrete elements are represented with groups of bits called binary codes.

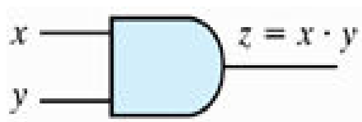
0	1	0	0	0	1	1	0

Binary values ...cont

- The decimal digits 0 through 9 are represented in a digital system with a code of 4-bit.

- Example, the number 5 is represented by 0101.

Logic gates

- Logic gates are electric circuits that operate on one or more input signals to produce an output signal.
- Example: 

- Electrical signals such as voltages exist as analog signals.
- Example of binary signals
- - Logic Level Ranges of Voltage for a Digital Circuit
- - Each voltage level has an acceptable range

Why Binary System?

- Computers are made of a series of switches
- Each switch has two states: ON or OFF
- Each state can be represented by a number,
- 1 for “ON” and 0 for “OFF”
- Computers use binary numbers internally because storage devices like memory and disk are made to store 0s and 1s.
- A number or a text inside a computer is stored as a sequence of 0s and 1s.

- Example: When you write a number like *30* in a program, it is assumed to be a decimal number. Internally, computer software is used to convert decimal numbers into binary numbers.

Types of number systems

أنواع أنظمةِ العدد

- Decimal
- Binary
- Octal
- Hexadecimal

Number System	Base	Digits use
Binary	Base 2	0, 1
Octal	Base 8	0, 1, 2, 3, 4, 5, 6, 7
Decimal	Base 10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Hexadecimal	Base 16	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Binary Numbers

- The binary number system has two possible values: 0 and 1, and the positions value are multiplied by powers of 2.

$2^1 = 2$	$2^8 = 256$
$2^2 = 4$	$2^9 = 512$
$2^3 = 8$	$2^{10} = 1024$
$2^4 = 16$	$2^{11} = 2048$
$2^5 = 32$	$2^{12} = 4096$
$2^6 = 64$	$2^{13} = 8192$
$2^7 = 128$	$2^{14} = 16384$

Decimal number system

- The Decimal Number System uses base **10**.
- It includes the digits {**0, 1, 2, 3, 4, 5, 6, 7, 8, 9**}.
- The weighted values for each position are:

10^4	10^3	10^2	10^1	10^0	10^-1	10^-2	10^-3
10000	1000	100	10	1	0.1	0.01	0.001

- Example,
- The digits 7, 4, 2, and 3 in decimal number 7423 represent 7000, 400, 20, and 3 units, respectively, as shown below:
- The number 7423 $= 7 \times 10^3 +$

$$4 \times 10^2 +$$

$$2 \times 10^1 +$$

$$3 \times 10^0$$

7423 = 7000 + 400 + 20 + 3

- يسمى الرقم على اقصى اليمين بالرقم الاقل اهمية LSD أما الرقم على اقصى اليمين فيسمى الرقم الاكثر أهمية MSD

Decimal number

$$\cdots a_5 a_4 a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3} \cdots$$

Decimal point

$$a_j$$

Base or radix

Power

$$\cdots + 10^5 a_5 + 10^4 a_4 + 10^3 a_3 + 10^2 a_2 + 10^1 a_1 + 10^0 a_0 + 10^{-1} a_{-1} + 10^{-2} a_{-2} + 10^{-3} a_{-3} + \cdots$$

Example:

$$7{,}329 = 7 \times 10^3 + 3 \times 10^2 + 2 \times 10^1 + 9 \times 10^0$$

General form of base-r system

$$a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \cdots + a_2 \cdot r^2 + a_1 \cdot r^1 + a_0 + a_{-1} \cdot r^{-1} + a_{-2} \cdot r^{-2} + \cdots + a_{-m} \cdot r^{-m}$$

Coefficient: $a_j = 0$ to $r - 1$

Binary Conversion

التحويل الثنائي

Converting numbers from any numbering system to decimal

❑ Base 2 (Binary) To Decimals

- To convert binary to decimal, use decimal arithmetic to form the Sum: Ex: (3) → (0011) = $0\times2^3+0\times2^2+1\times2^1+1\times2^0=3$
- (Digit × respective power of 2).

❑ Base 8 (Octal) to Decimal $(532)_8 \rightarrow (346)_{10}$

- The coefficient value for base 8 (0, 1, 2, 3, 4, 5, 6, 7)

❑ Base 16 (hexadecimal) to Decimal $(A31)_{16} \rightarrow (2609)_{10}$

- The coefficient value for base 16 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, F)

- عملية التحويل من اي نظام الى النظام العشري تعتمد على قانون التمثيل الموضعي للاعداد

Example: Base-2 number

$$(11010.11)_2 = (26.75)_{10}$$
$$= 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2}$$

Example: Base-5 number

$$(4021.2)_5$$
$$= 4\times 5^3 + 0\times 5^2 + 2\times 5^1 + 1\times 5^0 + 2\times 5^{-1} = (511.4)_{10}$$

Example: Base-8 number

$$(127.4)_8$$
$$= 1\times 8^2 + 2\times 8^1 + 7\times 8^0 + 4\times 8^{-1} = (87.5)_{10}$$

Example: Base-16 number

$$(B65F)_{16} = 11\times 16^3 + 6\times 16^2 + 5\times 16^1 + 15\times 16^0 = (46,687)_{10}$$

Example: Base-2 number

$$(110101)_2 = 32 + 16 + 4 + 1 = (53)_{10}$$

- Exercise: Convert $(120)_3 \rightarrow (15)_{10}$

- The **digit** in binary number called **bits**.

- When a bit is equal to 0. It does not contribute to the sum during the conversion.

- Example: $(11\mathbf{0}1\mathbf{0}1)2 = 32 + 16 + 4 + 1 = (53)_{10}$

- Four 1's in the binary number contribute to the sum

Table 1.1
Powers of Two

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

- 1 byte = 8 bits

Name	Abbr.	Size
Kilo	K	2^10 = 1,024
Mega	M	2^20 = 1,048,576
Giga	G	2^30 = 1,073,741,824
Tera	T	2^40 = 1,099,511,627,776
Peta	P	2^50 = 1,125,899,906,842,624
Exa	E	2^60 = 1,152,921,504,606,846,976
Zetta	Z	2^70 = 1,180,591,620,717,411,303,424
Yotta	Y	2^80 = 1,208,925,819,614,629,174,706,176

1 Byte		8 Bits
1024 Bytes	2^10	1 Kilobyte
1024 Kilobytes	2^20	1 Megabyte
1024 Megabytes	2^30	1 Gigabyte
1024 Gigabytes	2^40	1 Terabyte
1024 Terabytes	2^50	1 Petabyte
1024 Petabytes	2^60	1 Exabyte
1024 Exabytes	2^70	1 Zettabyte
1024 Zettabytes	2^80	1 Yottabyte
1024 Yottabytes	2^90	1 Brontobyte
1024 Brontobytes	2^100	1 GeopByte

Lecture-2

Arithmetic Operation

العملية الحسابية

- Arithmetic operations with number in base ***r*** follow the same rules as for decimal numbers.

- The sum of two binary numbers is calculated by the same rules as in decimal, but the sum is return to binary codes.

- Example: $5 \quad + \quad 6 \quad = 11$

$$0101 + 0110 = 1011$$

Addition	
Augend:	101101
Addend:	+100111
Sum:	1010100

Subtraction	
Minuend:	101101
Subtrahend:	–100111
Difference:	000110

Multiplication

Multiplicand	1011
Multiplier	× 101
Partial Products	1011
	0000 -
	1011 - -
Product	110111

Augend	المجموع عليه
Addend	المجموع اليه
minuend	المطروح منه
Subtrahend	المطروح
Multiplicand	المضروب
Multiplier	المضاعفة

Addition

Performing Basic Arithmetic Operation is the r-th Numbering System

1001101 + 1011010 = 10100111

11011+ 001101 = 101000

0101+ 0011 = 1000

Subtraction

$$\begin{array}{r} 110110010 \\ -\ \underline{100010101} \end{array}$$

Octal and Hexadecimal Numbers (Numbers with different bases)

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

$(8)_{10} = (\qquad)_8$

Number-Base Conversions

- The conversion of a decimal integer to a number in base r

تحويل العدد الصحيح العشري إلى عدد في القاعدةِ r

- If a number include a radix point
- Separate the number into an
  - Integer part
  - Fraction part

إذا كان العدد يتضمن النقطة (.) Radix point افصل العدد إلى جزء صحيح وجزء كسر

- Converting numbers from decimal to any other numbering system

- The conversion of a decimal integer to any other base r is done by dividing the decimal number and all successive results (النَتائِج المتعاقبة) by radix r and accumulating the remainder

- يستخدم في عملية التحويل طريقة remainder method حسب خوارزمية معينه.

Decimals to Binary

$(\qquad)_{10} = (\qquad)_2$

- Integer part

Example: convert decimal $(41)_{10}$ to binary $(\qquad)2$

	Integer	Remainder
/	**41**	
2	20	1 (LSB)
2	10	0
2	5	0
2	2	1
2	1	0
2	0	1 (MSB)

- The process is continued until the integer quotient becomes 0
- $(41)_{10} = (101001)_2$

- اقسم العدد واحصل على الباقي
- اكتب الناتج من الاسفل (البت الاكثر تأثيرMSB) الى الاعلى (البت الاقل تاثيرLSB)
- التحويل من العشري الى اي base بنفس الطريقة ولكن يتم القسمة على **r** base بدل 2

Decimals to Binary …cont.

- Divide the number and extract the remainder.
- Rewrite the solution from MSD to LSD
- The MSD in a number is the digit that has the greatest effect on that number.
- The LSD in a number is the digit that has the least effect on that number

Decimals to Octal

$(\qquad\qquad)_{10} = (\qquad\qquad)_8$

- Integer part
- Example: Convert decimal 153 to octal.

  The required base *r* is 8.
- $(153)_{10} = (\qquad\qquad)_8$

/	153	Remainder
8	19	1 (LSB)
8	2	3
8	0	2 (MSB)

$(153)_{10} = (231)_8$

ان التحويل في حالة العدد الصحيح { من العشري الى اي أساس اخر يتم بالقسمة على الاساس الجديد للحصول على عدد صحيح وكسر ونكرر العملية حتى يصبح العدد الصحيح صفر}

- Convert decimal numbers to the following bases

بالقسمة على اساس العدد الجديد

$(153)_{10}$	$=$	$(\ \ 231\ \ )_8$
$(231)_{10}$	$=$	$(\qquad)_8$
$(153)_{10}$	$=$	$(\qquad)_{16}$
$(23)_{10}$	$=$	$(\ \ 43\ \ )_5$
$(153)_{10}$	$=$	$(\qquad)_{12}$

Decimals to Binary

$(\qquad)_{10} = (\qquad)_2$

- Fraction part

Example: Convert $(0.625)_{10} = (\ 0.101\ )_2$

×	0.625	product	Integer
2	0.25	1.25	1 (MSB)
2	0.50	0.50	0
2	0.00	1.00	1 (LSB)

$(0.625)_{10} = (\ 0.101\ )_2$

$(0.625)_{10} = (\ 0.30\ )_5$

- اضرب الكسر في الاساس r وهو في هذا المثال 2 لتحصل على عدد صحيح وكسر
- اضرب الكسر الجديد في 2 لتحصل على عدد صحيح وكسر
- كرر العملية حتى يصبح الكسر صفر او دقة كافية

- التحول من الكسر العشري decimal fraction الى اي عدد في base r تستخدم نفس الطريق وذلك بضرب الكسر في r بدلا من 2

Decimals to Binary

$$(\qquad)_{10} = (\qquad)_2$$

- In fraction part …cont.
- Multiply the fraction part by 2 to give an integer and a fraction
- The new fraction is multiplied by 2 to give a new integer and a new fraction
- The process continued until the fraction becomes 0 or sufficient accuracy (دقة كافية)

Decimals to base r

$$(\qquad\qquad)_{10} = (\qquad\qquad)_r$$

Fraction part …cont.

- To convert a decimal fraction to a number expressed in base r, a similar procedure is used.

- However, a multiplication by r instead of 2, and the coefficients found from the integers may range in value from 0 to $r - 1$ instead of 0 and 1.

- Example:
- Convert the following decimal number with both integer and fraction parts:
- $(41.6875)_{10} = (101001.1011)_2$

- للتحويل من النظام العشري الى اي نظام في حالة العدد الصحيح استخدم القسمة على اساس العدد الذي يتم التحويل اليه مع تكرار العملية و العدد الكسري .

Convert $(0.6875)_{10} = (\qquad\qquad)_2$ طريقة الضرب

Decimals to base *r*

$$(\qquad\qquad)_{10} \rightarrow (\qquad\qquad)_r$$

Fraction part …cont.

Example:

×	0.6875	product	Integer
2	0.375	1.375	1
2	0. 75	0.75	0
2	0.50	1.50	1
2	0.00	1.00	1

إن التحويل من النظام العشري الى اي نظام r للعدد الكسري بالضرب في أساس العدد الذي يتم التحويل اليه مع تكرار العملية حتى النهاية وتجميع الايجابه للجزء الصحيح الناتج عن ضرب كل جزء من الاكبر تاثير الى الاقل تاثير

Conversion… cont.

Examples:

$(.20)_{10} = (\qquad)_8$

×	0.20	product	Integer
8	0.6	1.6	1
8	0.8	4.8	4
8	0.4	6.4	6
8	0.2	3.2	3

Answer (0.1463)

$(.12)_{10} = (\qquad)_{16}$ بالضرب

×	**0.12**	product	Integer
16	0.92	1.92	1
16	0.72	14.72	14
16	0.52	11.52	11
16	0.32	8.32	8
16	0.12	5.12	5

Answer = (0.1EB85)

Quiz: $(12)_{10} = (\qquad\qquad)_{16}$

Decimals → Octal

$(\qquad)_{10} \rightarrow (\qquad)_8$

- Fraction part

Convert $(0.513)_{10} = (\qquad)_8$

×	0.513	product	Integer
8	0.104	4.104	4
8	0.832	0.832	0
8	0.656	6.656	6
8	0.248	5.248	5
8	0.984	1.984	1
8	0.875	7.872	7

$(0.513)_{10} = (0.406517\ldots.)_8$

يتم التحويل للعدد العشري الصحيح والعدد العشري الكسري بشكل منفصل الصحيح بالقسمة على الاساس والكسري بالضرب في الاساس وتجمع نتيجة الطرفين

Conversion...cont.

- Conversion from binary to octal can be done by positioning the binary number into groups of three digits each, starting from the binary point and proceeding to the left and to the right.

(10	110	001	101	011	·	111	100	000	$110)_2 = (26153.7406)_8$
2	6	1	5	3		7	4	0	6

- Conversion from binary to hexadecimal is similar, except that the binary number is divided into groups of four digits:

(10	1100	0110	1011	·	1111	$0010)_2 = (2C6B.F2)_{16}$
2	C	6	B		F	2

- Conversion from octal or hexadecimal to binary is done by reversing the preceding procedure.

$(673.124)_8 =$ (110	111	011	·	001	010	$100)_2$
6	7	3		1	2	4

$(306.D)_{16} =$ (0011	0000	0110	·	$1101)_2$
3	0	6		D

Quiz: Convert (68BE) to binary and from binary to octal

Exercises

- $(175)_{10}$ = $(10101111 \quad)_2$
- $(235)_8$ = $(10\ 011\ 101\ )_2$
- $(AF)_{16}$ = $(1010 \quad 1111\ )_2$
- $(0.4)_{10}$ = $(0.011001 \quad)_8$
- $(25.125)_{10}$ = $(11001.001 \quad)_2$

Lecture-3

Complements

- Used in digital system to simplify the subtraction operation and for logical manipulation لتبسيط الطرح وللمعالجة المنطقية

- There are two types of complements for each base-r system:
  - radix (r) complement
  - diminished (r-1) radix complement

- 1's & 2's complement for binary numbers
- 9's &10's complement for decimal numbers

Subtraction with Complements
الطرح بالتكملاتِ

❑ Subtraction with Complements for Decimal numbers

- Unsigned numbers
- Signed numbers

❑ Subtraction with Complements for binary numbers

- Unsigned numbers
- Signed numbers

Signed Binary Numbers

- There are two representations of the sign numbers
- Signed magnitude system used in ordinary arithmetic that negates the number by changing its sign النفي بتغير اشارة المقدار

  Ex: -9 to +9 (negates the number by changing its sign)
- Signed-complement system (negates the number by taking its complement (يَنْفي العددُ بأَخْذ المكمل)
- Ex: 0110 the 2's complement is 1010 (negate the number)

♣ 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.
♣ The convention is to make the sign bit 0 for positive and 1 for negative.

Signed Binary Numbers...cont.

- Example: Signed magnitude representation of the number -9 by using 8-digits.

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

في الأعداد الثنائية:

- الصفر على يسار Binary codes يمثل الاشارة الموجبه +
- الواحد يمثل الاشارة السالبه -

♣ **Table 3** lists all possible four-bit signed binary numbers in the three representations.

No change

change

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	—	—

Lecture-4

Arithmetic Addition

- Signed magnitude system
- Signed-complement system

- The addition of two numbers in the signed magnitude system follows the rule of ordinary arithmetic

جمع عددين في نظامِ المقدارِ يعمل بنفس قاعدةُ الحسابِ العاديِ

- If the signs are the same, we add the two magnitudes and give the common sign to the sum.

إذا كانت إشارة العددين في السالب ضع إشارة سالب للمجموع و إذا كانت الإشارة للعددين جمع ضع إشارة الجمع للمجموع

- Example: +6 +13 = +19

  - 6 – 13 = - 19

Signed magnitude system...cont.

- If the signs are different, subtract the smaller magnitude from the larger.
- Give the difference and put the sign to the larger magnitude.
- Example: → $-6 + 13 = +7$

  $-12 + 2 = -10$

إذا الإشاراتِ مختلفة، نَطْرحُ المقدار الأصغرَ مِنْ الأكبرِ ويَعطي الإشارةَ للمقدارِ الأكبرِ .

Signed-complement system

- 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.

Signed-complement system…cont.

Example: using binary addition

1

+ 6	00000110
+13	00001101
+ 19	00010011

2

+ 6	00000110
−13	11110011
− 7	11111001

3

− 6	11111010
+13	00001101
+ 7	00000111

4

− 6	11111010
−13	11110011
− 19	11101101

Example: (-6) + (-13) = -19

• In binary, take the 2's complement of 6 and 13 and then add them to get the result

The 2’s C of 6 = -6 using 8 digits

6 = 00000110 -6 = 11111010

-6 = 2's C of 6	00000110	-6 = 11111010
-13 = 2's C of 13	00001101	-13 = 11110011 +
(-6) + (-13) =		-19 = 1 11101101

The carry 1 is discarded

Arithmetic subtraction

In 2's-complement form:

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) $\Rightarrow$ (11111010 – 11110011)

$\Rightarrow$ (11111010 + 00001101)

$\Rightarrow$ 00000111 (+ 7)

Example:

Use the arithmetic subtraction to subtract +6 -13

= (+6) + (-13) = 00000110 + 2's C of 13

= 00000110 + 11110011= 11111001= -7 → 2's C of 7

BCD Code

- A number with ***k*** decimal digits will require 4*k* bits in BCD.
- Example: 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.
- A BCD number greater than 10 looks different from its equivalent binary number, even though both contain 1's and 0's.
- Moreover, the binary combinations 10 =1010 through 15 =1111 are not used and have no meaning in BCD.
- واجب : بحث

BCD Code...cont.

- Table 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

- تحويل BCD هو نفس Decimal من 0-9

Lecture-5

Binary Storage and Registers

- Registers are very important components in the computer
- تعتبر المسجلات من المكونات الهامة جدا في الحاسوب
- Used in all units and work as links between all units.
- تستخدم في جميع وحداته وتعمل كحلقات وصل بين جميع الوحدات.

❑ Registers functions:

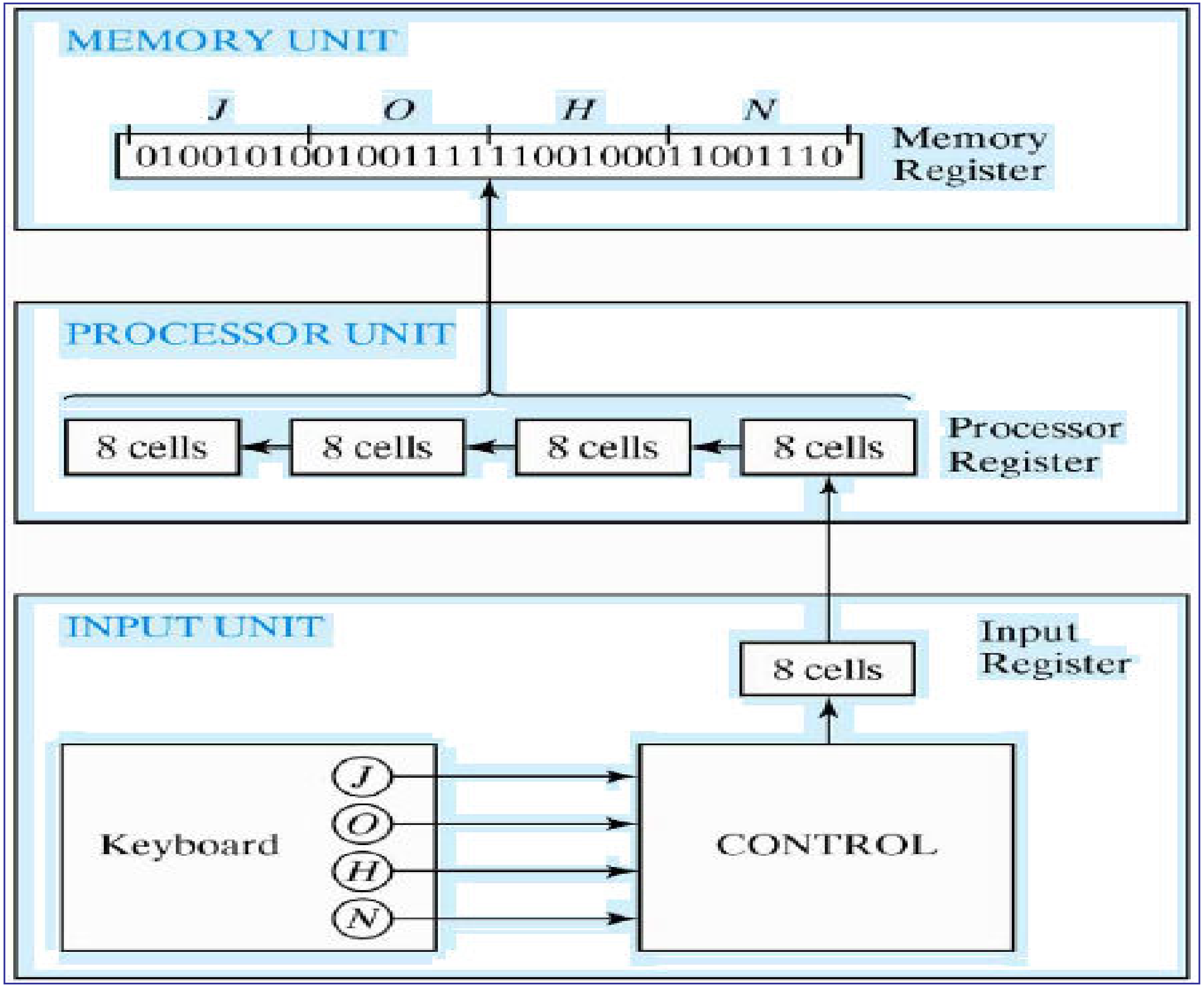
» Information Storage

» Perform some mathematical and logical operations

» خزن المعلومات وإجراء بعض العمليات الحسابية والمنطقية

- Registers have two types:
- Storage registers (E.g. : main memory registers)
- Operation registers: Perform some mathematical and logical operations
- E.g. ALU registers

Transfer of information



Binary Logic

- Binary logic consists of:
  - binary variables.
  - 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.

- There are three basic logical operations: AND, OR, and NOT. العمليات المنطقية الأساسية

Three basic logical operations

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.

- The truth tables for AND, OR, and NOT are given in Table 1.8.

Table 1.8
Truth Tables of Logical Operations

AND

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

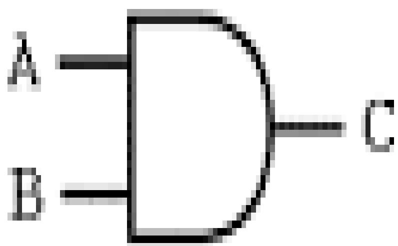
x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

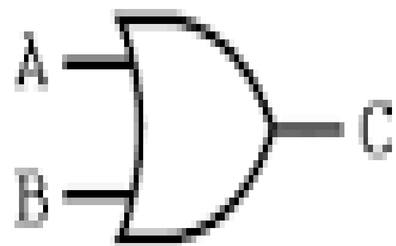
x	x'
0	1
1	0

Logic gates
البوابات المنطقية الأساسية

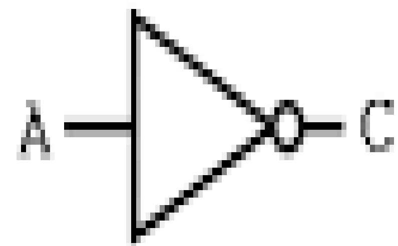
AND



OR



NOT



Inputs		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Inputs		Output
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Input	Output
A	C
0	1
1	0

Digital Logic Design

Chapter 2

Dr. Amer Abu-Jassar
Faculty of Science and Information Technology

Lecture-6

Boolean algebra and Logic Gates

Outlines

- Basic Definitions of Algebra
- Axiomatic Definitions of Boolean Algebra
- Basic Theorems and Properties of Boolean Algebra
- Boolean Function
- Canonical and Standard Forms
- Other logic Operations
- Digital Logic Gates

Boolean algebra

- Boolean algebra is a mathematical system for the manipulation of variables that can have one of two values.

- نظام رياضي يستخدم البوابات المنطقية في معالجة المتغيرات للحصول على قرار منطقي

- Boolean algebra is the basis for the design of logic circuits that make up the computer
- الجبر البولي هو الاساس في تصميم الدوائر المنطقية التي يتكون منها الحاسوب

- يسمى المتغير بوليا (او منطقيا) اذا اتخذ احد الحالتين
- الحالة الصحيحة (true)
- الحالة الخاطئة (false)

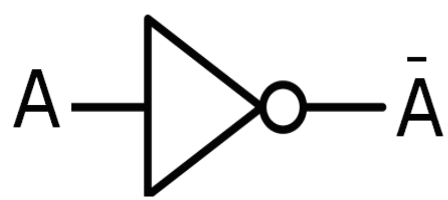
- The Boolean algebra may be defined with a:
- Set of elements (مجموعة من العناصر)

  E.g. a set S = { 1, 2, 3, 4 } , means that the numbers 1, 2, 3, and 4 are elements of the set S
- E.g. N ={1, 2, 3, ...}
- Set of operations (e.g. **+, - , *,..**) (مجموعة من العمليات)
  - a binary operator on a set S of elements is a rule that assigned, to each pair of elements from s.
  - E.g. : a * b = c, indicate that * is a binary operator, if a, b, c ∈ S,
  - if a, b, c لا تنتمي to S, the * is not a binary operator

A Boolean function: (الاقتران البولي)

- A Boolean function has:

- At least one Boolean variable,
- At least one Boolean operator,
- At least one input from the set B{0, 1}.
- It produces an output that is also a member of the set B{0, 1}.
- e.g. , بوابه NOT has one variable (A), one operator (NOT), and has one input from the set B{0, 1} to represent the variable A , and produces an output that is also a member of the set B{0, 1}.



Logic Operations

- The Boolean operators consist of: تقسم العمليات البولية الى:

  - Basic logic operations [AND, OR, and NOT] .
  - Other logic operations .

البوابات البولية الأساسية

  - AND, OR, and NOT

- تسمى العملية (AND و OR) عمليتان ثنائيتان (Binary Operations) لان الاقتران له متغيران وعمليه واحدة

- Example: X AND Y

- بينما البوابه او العملية NOT تعتبر عملية احادية (Unary) لان لها متغير واحد او مدخلا واحدا .

التعبير عن العمليات الثلاثة بالنظام الثنائي

- A Boolean operator can be completely described using a truth table
- يمكن وصف العملية البولية باستخدام جدول الحقيقة

X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

NOT X = X'

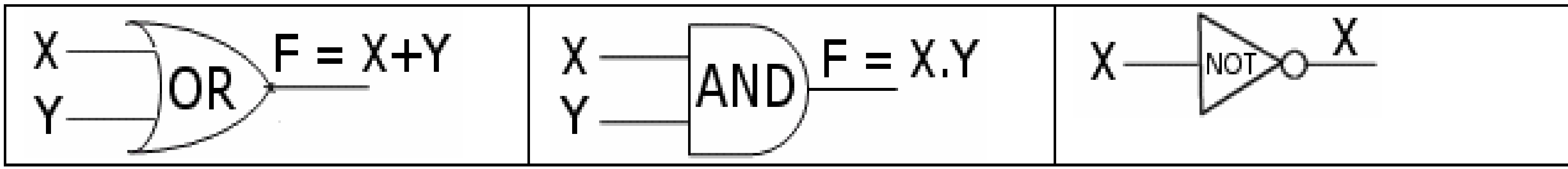
X	$\overline{X}$
0	1
1	0

- The AND operator is also known as a Boolean product → X .Y
- The OR operator is the Boolean → sum X+Y.
- The NOT operator is the Boolean → inverter or complement.
- It is sometimes indicated by a prime mark (')

- عملية NOT هي المكمل وتعكس حالة المتغير من 0 إلى 1 و العكس

Graphic Symbol

• التعبير عن العمليات الثلاثية الأساسية بالرموز Symbols



طرق تمثيل البوابات المنطقية الرئيسية

البوابات البولية المشتقة Other logic Operations

وقد اشتقت هذه البوابات من البوابات المنطقية الرئيسية وهي:

1	عملية NAND Operation	وتسمى (NOT AND)
2	عملية NOR Operation	وتسمى (NOT OR)
3	عملية XOR Operation	وتسمى (Exclusive OR)
4	عملية XNOR Operation	وتسمى (Exclusive NOR)

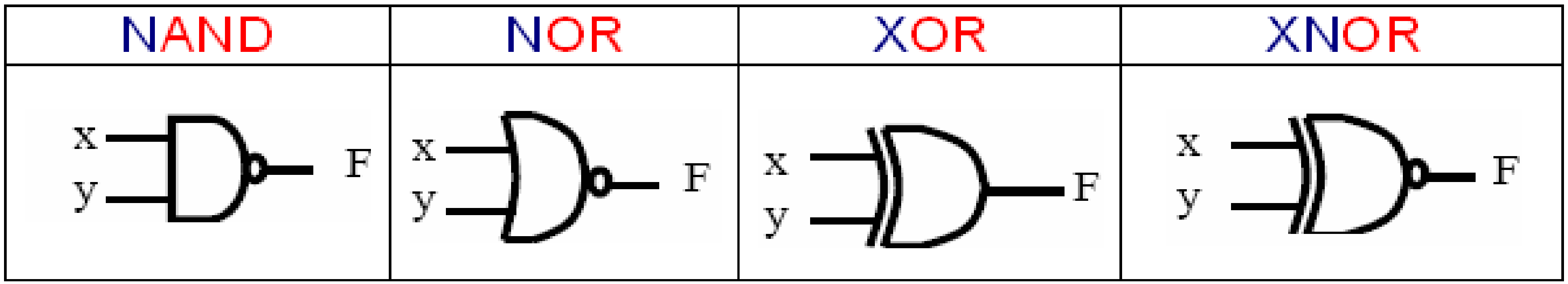
Standard Form البوابات المنطقية المعيارية

- The gate NOR and NAND are the most important logic gates because by using any of them can build any other logic gate, and thus can be used either in the construction of many logical circuits.

- تعتبر بوابة NOR و NAND من أهم البوابات المنطقية وذلك لأنه باستخدام أي منهما يمكن بناء أية بوابه منطقية أخرى، وبالتالي يمكن استخدام أي منهما في بناء دوائر منطقية كثيرة.

Other Logic Operations

The graphic symbol of the other logic operations is



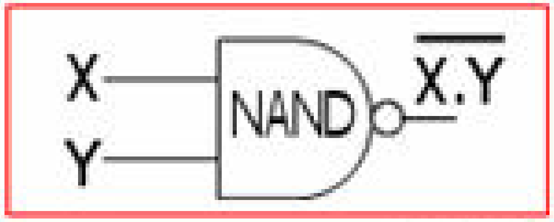
جدول الحقيقة (Truth table) يوضح هذه العمليات لكي تعرف العلاقة بين المتغيرات ونتائج العمليات المشتقة من العمليات الأساسية السالفة الذكر

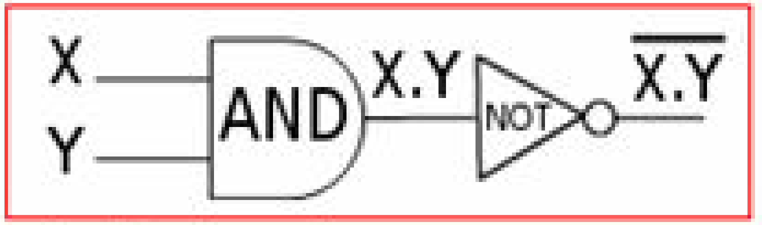
Variables		NAND	NOR	XOR	XNOR
X	Y	F= (XY)'	F= (X+Y)'	$F = X \oplus Y$	$F = X \odot Y$
0	0	1	1	0	1
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1

جدول يمثل العمليات البولية المشتقة

Other Logic Operations…cont.

- بوابة NAND Gate: هي بوابة AND وتليها بوابة NOT كما هي موضحة في الشكل التالي:





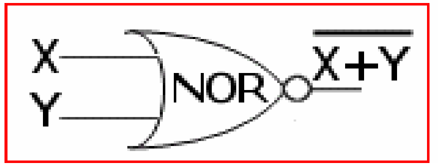
الرمز المنطقي لبوابة NAND

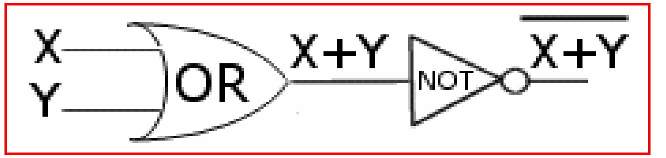
- أن بوابة NAND تعمل عكس عمل بوابة AND. كما في جدول الحقيقة التالي

X	Y	AND	NAND → not AND (X.Y)'
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Other Logic Operations…cont

- إن بوابة NOR وهي عبارة عن بوابة OR تليها بوابة NOT كما هي موضحة في الشكل



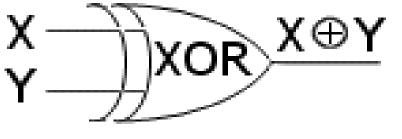


الرمز المنطقي لبوابة NOR

- أن بوابة NOR تعمل عكس عمل بوابة OR. كما في جدول الحقيقة التالي

X	Y	OR	NOR → not OR (X+Y)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

- بوابة: XOR وهي بوابة تعطي ناتجاً في الحالة الصحيحة إذا كان مدخلاها مختلفين، وتعطي ناتجا في الحالة الخاطئة إذا كان المدخلان متشابهين، والرمز الرياضي لها هو دائرة صغيرة بداخلها علامة الزائد، وفي ما يلي الرمز المنطقي لها.

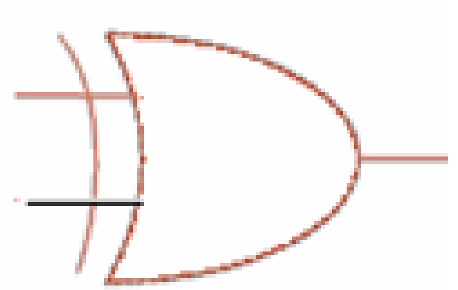


الرمز المنطقي لبوابة Exclusive OR

- وجدول الحقيقة التالي يمثل عمل هذه البوابة XOR

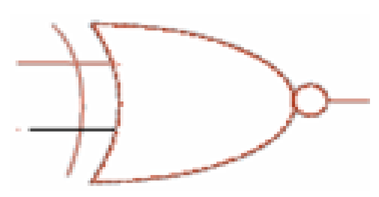
X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

- إن بوابة XOR تعطي الحالة 1 إذا كانت مدخلات الدائرة مختلفة

			X	Y	F
XOR		$F = x \oplus y = XY'+X'Y$ المدخلان المختلفان تكون النتيجة 1	0	0	0
			0	1	1
			1	0	1
			1	1	0

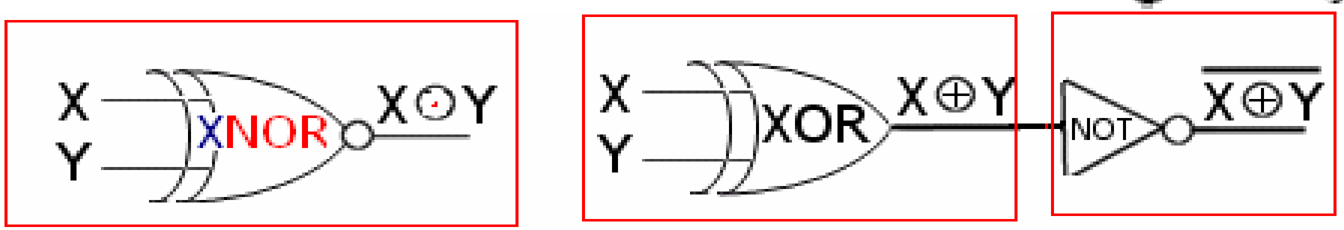
- بوابة XNOR تعطي نتيجة 1 إذا كانت مدخلاتها متشابهة

X	Y	$\overline{X\oplus Y} = X \odot Y$
0	0	1
0	1	0
1	0	0
1	1	1

XNOR		$F = (x\oplus y)' = XY+X'Y'$ $= X\odot Y$ المدخلان المختلفان تكون النتيجة 0	

X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1

- XNOR تعمل عكس عمل بوابة XOR، وهي عبارة عن بوابة XOR تليها بوابة NOT كما هي موضحة في الشكل



XNOR = XOR + NOT

الرمز المنطقي لبوابة XNOR

- هذه العمليات اشتقت منها قوانين هامة لتستخدم في بناء الدوائر المنطقية

GRAMMARS

OR	AND
$X + 1 = 1$	$X.1 = X$
$X + X' = 1$	$X.X' = 0$
$X + X = X$	$X.X = X$
$X + 0 = X$	$X.0=0$
$(X')' = X$	$(X')' = X$
$X + Y = Y + X$	$X.Y = Y.X$
$X + (Y+Z)= (X+Y) + Z$	$X.(Y.Z)= (X.Y).Z$
$X. (Y+Z) = X.Y + X.Z$	$X+Y.Z = (X+Y).(X+Z)$
$(X + Y)' = X'.Y'$	$(X.Y)' = X' + Y'$
$X+ (X.Y) = X$	$X. (X+Y)= X$

Basic Definition of Boolean Algebra

- The most common postulates used to formulate various algebraic structures are:

- المسلّمات الأكثر شيوعاً تُستعملُ لصيَاْغَة التراكيبِ الجبريةِ المُخْتَلِفةِ:

1. Closure : خاصية الانغلاق

- A set *S* is closed with respect to a <u>binary operator</u> if, for every pair of elements of *S*, the binary operator specifies a rule for obtaining a unique element of *S*.

- نسمي مجموعة S مُغلقةٌ على العملية الثنائية + أو * ، إذا حددت العملية الثنائية قانون مثل الجمع لكُلّ زوج من العناصرِ للحُصُول على العنصر الفريد للمجموعةٍ S .

Example: the set of natural numbers N = {1, 2, 3,4, …}is closed with respect to the binary operator plus (+) by the rule of arithmetic addition, since for any $a, b \in$ N we obtain a unique $c \in$ N by the operation $a + b =$ c.

- الإعداد الطبيعية للمجموعة N مغلقة على العملية الثنائية + حسب قانون الجمع الرياضي، فأي زوج من عناصر N ينتمي إلى المجموعة فان حاصل جمعهما أيضا ينتمي إلى N حسب خاصة الانغلاق للمجموعة.

2. Associative Law (القانون التِرابُطي أو قانون الاقتران)

- A binary operator * on a set *S* is said to be associative whenever, (X * Y) * Z = X * (Y * Z) for all X, Y, Z ∈ S

- في القانون الترابطي الأقواس لا توثر على العملية * والعملية +

3. Commutative Law (القانون التبادلي)

- A binary operator * on a set *S* is said to be commutative whenever,
- $X * Y = Y * X$

4. Identity Element (العنصر المحايد)

- A set *S* is said to have an identity element with respect to a binary operation * on S if there exists an element $e \in S$ with the property $\mathbf{e * x = x * e = x}$

Example, $0 + X = X + 0 = X$

- 0 is an identity element with respect to operator +.
- The additive identity is 0 and the multiplicative identity is 1

العنصر المحايد الجمعي على مجموعة الإعداد الصحيحة هو 0 والعنصر المحايد في عملية الضرب هو 1

5. Inverse (المعكوس) تبديل الحالة Complement

- A set B having an identity element e with respect to a binary operator * is said to have an inverse whenever, for every $x \in B$, there exist an element $y \in B$ such as: y = complement of x

- x * y (inverse) = e (كل عنصر في المجموعة له مكمل)
- Example, : $x . x' = 0$ $x + x' = 1$ (complement)
  $0 * 1 = 0$ $1 + 0 = 1$

6. Distributive Law (القانون ألتوزيعي)

- If (* and .) are two binary operators on a set S,
- * is said to be distributive over . whenever : $X * (Y . Z) = (X * Y) . (X * Z)$

Summarize the basic meanings to the some of the basic algebraic properties

تلخيص بعض المعاني للخصائص الجبرية الأساسية

- العملية الثنائية + تعرف بعملية الجمع و الممثلة بالدائرة المنطقية OR

- العنصر المحايد على عملية الجمع OR هو الصفر 0
- Example: $X + 0 = X$ →
- But $X + 1 = 1$ in binary system

- المضاف المعكوس يُعرّفُ بعملية الطرحً و المقصود به المكمل اي جمع عدد مع المكمل له في العملية الثنائية =1
- Example: $x + x' = 1$
- العملية الثنائية (.) تعرف بالضرب
- العنصر المحايد ألضربي و الممثل بالبوابة المنطقية AND هو 1
- Example: $X . 1 = X$

Lecture-7

Rules of Boolean Algebra

❑ Axiomatic Definition of Boolean algebra

- Most Boolean identities have an AND and OR
- We give the identities using both forms: AND & OR
- Boolean Algebra defined by a set of elements *B(0,1)* and two binary operators (+) and (.) and has the following postulates:

• الجبر المنطقي عرّفَ على مجموعة العناصرِ B و على العمليات الثنائية (+) و (.) ولهما المسلّماتُ أو الفرضيات التاليةُ:

Postulations and Theorems of Boolean algebra

- Postulations

	Postulates	(OR) form	(AND) form
Postulate 1	Closure خاصية الانغلاق	The structure is closed with respect to the operator (+).	The structure is closed with respect to the operator (.)
Postulate 2	Identity element العنصر المحايد	$x + 0 = x$	$x \cdot 1 = x$
Postulate 3	Commutative خاصية التبادل	$x + y = y + x$	$x.y = y.x$
Postulate 4	Distributive خاصية التوزيع	$x.(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$
Postulate 5	Complement خاصية المكمل	$x + x' = 1$	$x \cdot x' = 0$

☒ التوضيح للجدول السابق

Postulate 1: الانغلاق

(a) The structure is closed with respect to the operator (+).

(b) The structure is closed with respect to the operator (.).

Postulate 2: العنصر المحايد

(a) The element 0 is an identity element with respect to +.

$0 + x = x + 0 = x$

(b) The element 1 is an identity element with respect to (.).

$1 . x = x . 1 = x$

Postulate 3: التبادلي

(a) The structure is communicative with respect to +.

$X + y = y + x$

(b) The structure is communicative with respect to (.)

$x.y = y.x$

Postulate 4: التوزيع

(a) The operator (.) is distributive over +:

$x. (y + z) = (x. y) + (x. z)$

(b) The operator + is distributive over (.)

$x + (y. z) = (x + y). (x + z)$

Postulate 5: المكمل

(a) For every element $x \in B$, there exists an element $x' \in B$ (Complement of x) such that $x+x'=1$ and $x.x'=0$.

Postulate 6:

There exist at least two elements $x, y \in B$ such as $x \neq y$.

Difference with ordinary algebra

- The distributive law of + over (.) is valid for: Boolean algebra, but not valid for ordinary algebra.
- Example: $X + (Y . Z) = (X + Y) . (X + Z)$
- Example: 4 + (6 . 5) != (4+6) . (4+5)
- Boolean algebra does not subtraction or division operations.
- لا يوجد قسمة ولا طرح على مجموعة Binary numbers
- Complement is valid for Boolean algebra, but not for ordinary algebra.

❑Two-Valued Boolean Algebra

- A two-valued Boolean algebra is defined on a set of two elements, {0, 1}, with rules for the two binary operators (+ and .) As shown in the following operator tables. The rules of operations

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

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

x	x'
0	1
1	0

Postulate 1: الانغلاق

The structure is closed with respect to the two operators is obvious from the tables since the result of each operation is either (1 or 0) and {1, 0} تنتمي *B*.

Postulate 2: المحايد from the table, we see that

0 + 0 = 0	0 + 1 = 1 + 0 = 1
1 .1 =1	1. 0 = 0 .1 = 0

Two identity elements 0 & 1

Postulate 3: التبادلي

The communicative laws are obvious from the symmetry التناظر of the binary operator tables.

$$1.0 = 0.1$$

$$0 + 1 = 1 + 0$$

Postulate 4: التوزيع

- The distributive law can be shown to hold from the truth table of all possible values of x, y, and z.

ملخص لفرضيات وقوانين الجبر البولي

Postulates a and b

Postulates	a	b
Postulate 2 المحايد	$x + 0 = x$	$x \cdot 1 = x$
Postulate 3, Commutative	$x + y = y + x$	$xy = yx$
Postulate 4, Distributive	$x(y + z) = xy + xz$	$x + yz = (x + y)(x + z)$
Postulate 5 المكمل	$x + x' = 1$	$x \cdot x' = 0$

Theorems a) over OR b) over AND

Theorems	a	b
Theorem 1 قانون التماثل	$x + x = x$	$x \cdot x = x$
Theorem 2 قانون عمليات الواحد Null element	$x + 1 = 1$	$x \cdot 0 = 0$
Theorem 3, Involution قانون النفي المزدوج	$(x')' = x$	
Theorem 4, Associative الترابطي	$x + (y + z) = (x + y) + z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Theorem 5, DeMorgan	$(x + y)' = x'y'$	$(x \cdot y)' = x' + y'$
Theorem 6, Absorption الاختزال	$x + xy = x$	$x(x + y) = x$

DeMorgan'sTheorems

(X+Y)' = X'. Y‘

(X.Y)' = X' + Y‘

x	y	$x+y$	$(x+y)'$	x'	y'	$x'y'$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Operator Precedence
أولوية العمليات

- تستخدم الاولويات من اجل تقييم التعبير البولي
- These operators are used to evaluate the Boolean expressions as follows:
- Parenthesis الأقواس أولا
- NOT complement or inverter النفي
- AND العملية و
- OR العملية أو
- Example1: $(x + y)' = x' . y'$
- Example2: $(ABCD)' = A' + B' + C' + D'$
- تقييم ما بداخل القوس أولا X+Y ثم (النفي) المكمل يأتي رقم 2 (X+Y)' ثم عملية AND ثم OR حسب الاقتران المعطى

Example: Consider the truth table for one of the Demorgan's theorem, the left side of the expression is (X+Y)'

x	y	$x+y$	$(x+y)'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

- The expression inside the parentheses is evaluated first
- The result the complemented.

Lecture-8

Boolean Functions
البوابات المنطقية

- استخدمت القوانين المنطقية السابقة لبناء الدوائر المنطقية الرقمية والتي تتكون من مجموعة من البوابات المنطقية المصممة هندسيا

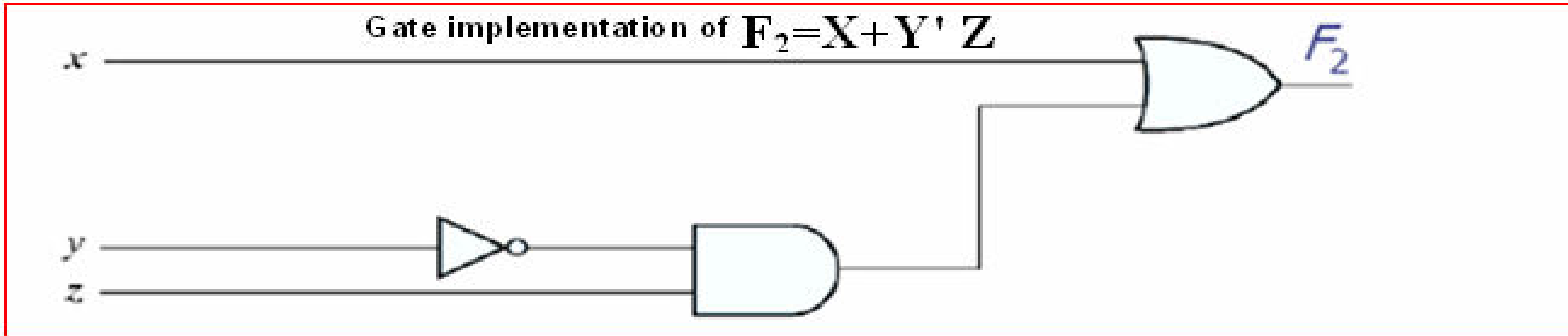
- ذكرنا ثلاثة بوابات أساسيات رئيسية (AND OR NOT)

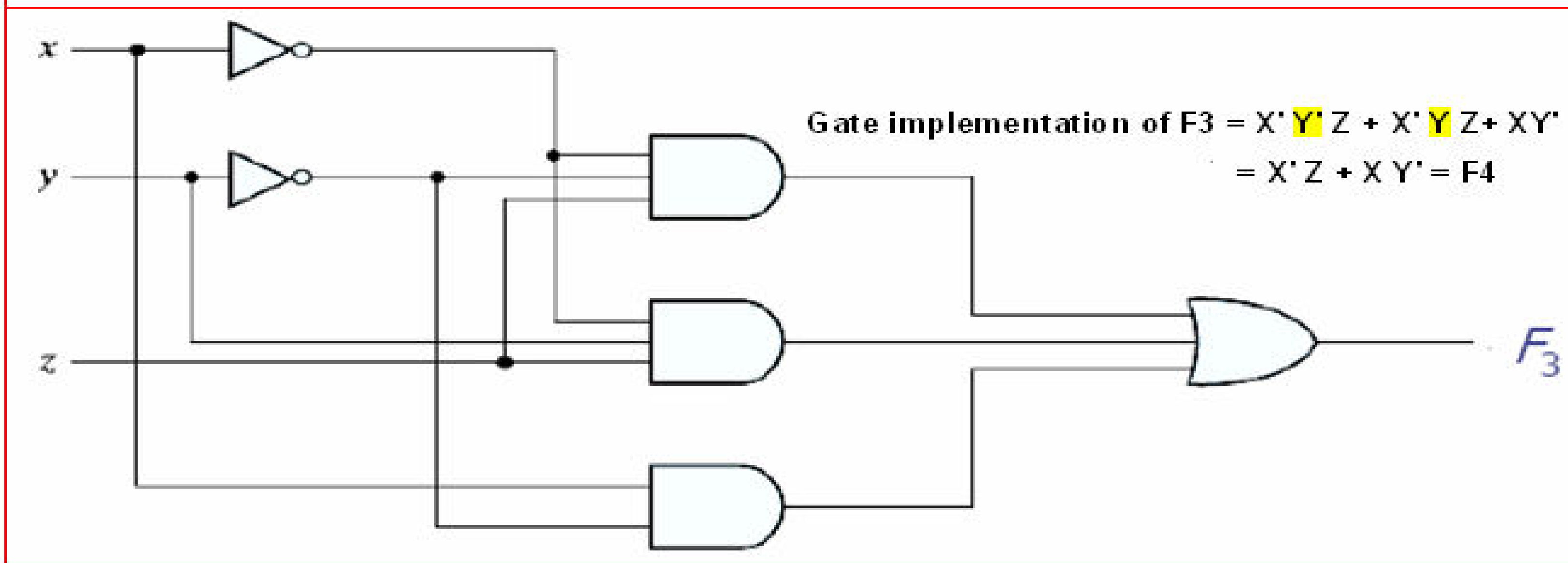
A Boolean function

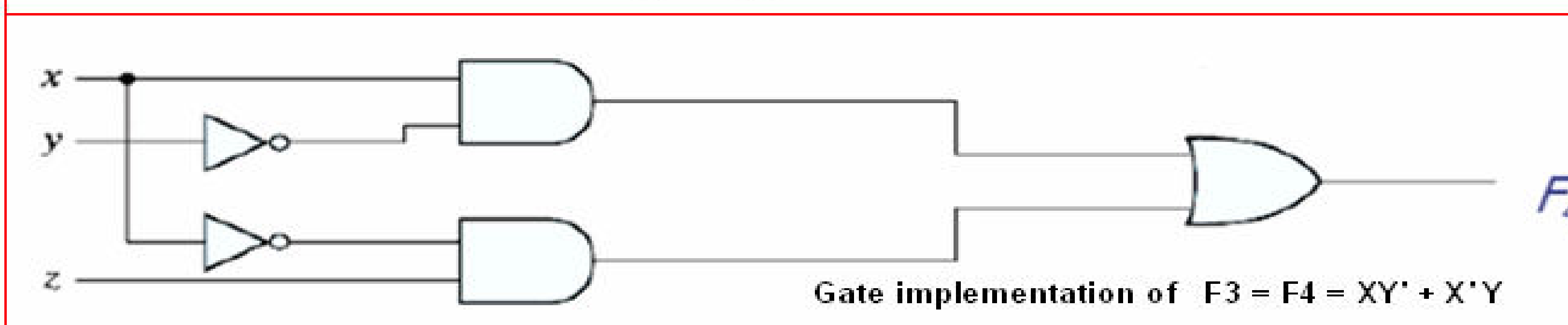
- ❑ There are 3 kinds of representations for Boolean functions
- Boolean Algebra consist of:
- Function: for example: F= (X + Y') . Z consist of:
  - Binary variables such as x, y, z ….
  - Binary operators OR and AND.
  - Unary operator NOT → prime '
  - parentheses ( )
- Truth table
- Circuit diagram

Implementation with logic gates

بناء الدوائر المنطقية





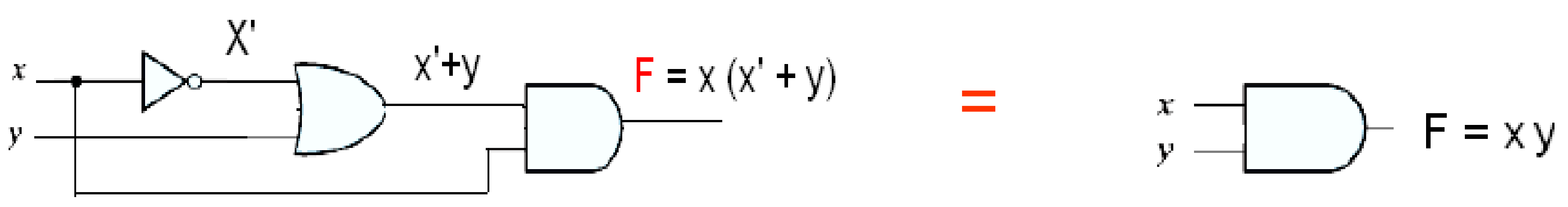


Algebraic manipulation
معالجة المعادلات الجبرية

- الجبر البولي (الجبر المنطقي) هو الأساس في تصميم الدوائر المنطقية التي يتكون منها الحاسوب لأنه يتخذ دائما إحدى الحالتين True أو False
- إن الهدف من معالجة المعادلات الجبرية هو اختصار الدالة باستخدام قوانين الجبر البولي ومن ثم بناء الدائرة لصيغة الدالة المختصرة من اجل التوفير وسرعة المعالجة وتقليل تكاليف الدوائر المطلوبة وتقليل تعقيد بناءها

- Example, the logic diagram of the following expression
- $F = X(X' + Y)$ is:

- يمكن اختصار هذه الدالة إلى الدائرة المنطقية التالية



$$F = X(X'+Y) = XX' + XY \quad = \quad 0 + XY = XY$$

To minimize Boolean Expressions

Boolean algebra can be used to simplify circuit design

الاختصار للاقتران بهدف تبسيط الدوائر المنطقية

- EX1: $F= X(X'+Y) = XX' + XY = 0+XY= XY$
- EX2: $F= X+X'Y = (X+X') . (X+Y) = 1.(X+Y)=X+Y$
- EX3: $F= X (X' + Y) = X.X' + X.Y = 0 + XY = XY$
- EX4: $F= C + (B C)'$

  $= C + B' + C'$

  $= (C + C') + B'$

  $= 1 + B' = 1$
- EX5: $F= X (X' + Y) + XY'$

  $= X.X' + X.Y + X.Y'$
- $= 0 + X + (Y + Y')$
- $= X$

- Example: Simplify $F = ABC + ABC' + A'C$

$$= AB\,(C + C') + A'C$$

$$= AB \cdot 1 + A'C$$

How many gates are there in this function → 4 gates

كم عدد الدوائر المنطقية قبل وبعد الاختصار

- Example: Consider the implementation of the function
  - $A \cdot B + A \cdot C$
  - This function requires 2 AND Gates and OR gate
  - By distributive law
  - $A \cdot B + A \cdot C = A \cdot (B + C)$
  - This can be implemented using 1 AND gates and 1 OR gate

Minimize Boolean Expressions…cont.

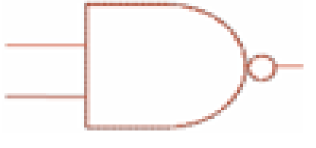
- Simplify the following Boolean expressions to a minimum number of literals
- Example: $X + X'Y = X+Y$
- $= (X+X').(X+Y)=1.(X+Y)$
- $= X+Y$

Standard Logic Gates (NAND & NOR)

البوابات المنطقية المعيارية

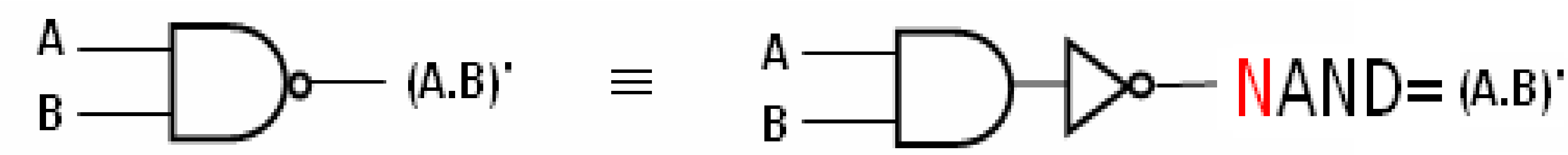
- البوابتان NAND & NOR من أهم البوابات المنطقية وتسمى معياريتان لأنه باستخدام أي منهما يمكن بناء أية بوابة منطقية أخرى
- بوابه NAND gate: بوابة منطقية لها مدخلان على الأقل ومخرج واحد على الأقل وتستخدم لبناء الدوائر المنطقية الأساسية

- **NAND** gate is self-sufficient (can build any logic circuit with it).
- It can be used to implement AND/OR/NOT.

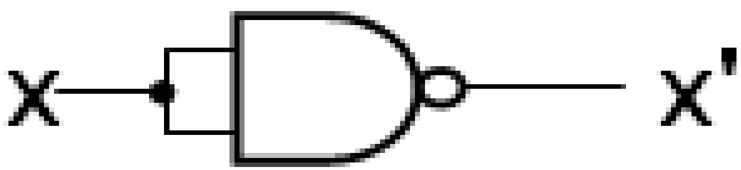
			X	Y	F
NAND		X NAND Y $F(X, Y) = (X.Y)' = X' + Y'$	0	0	1
			0	1	1
			1	0	1
			1	1	0

بناء بوابة NOT باستخدام NAND

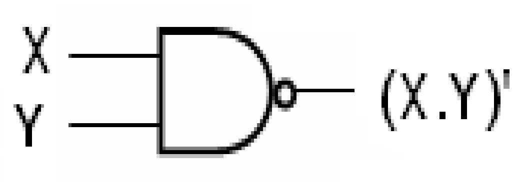
- (AND)' = NAND → NOT AND
- (NAND)'' = NAND → involution قانون النفي المزدوج



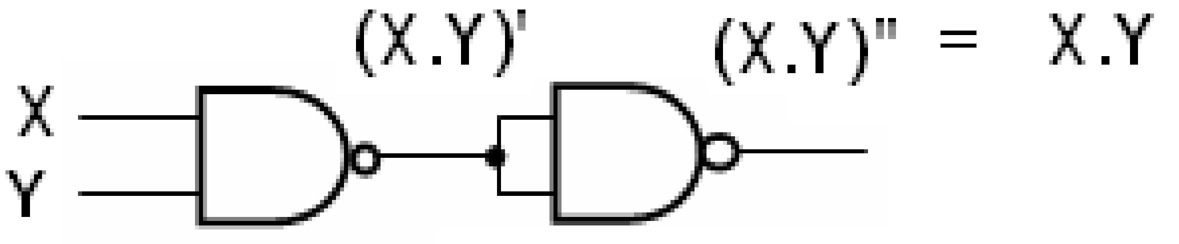
- (X.Y)’ = X’+ Y’ حسب ديمورقان
- بناء بوابة NOT باستخدام NAND
- Implement an inverter using NAND gate: المكمل باستخدام بوابة ناند



بناء بوابة AND باستخدام بوابة NAND

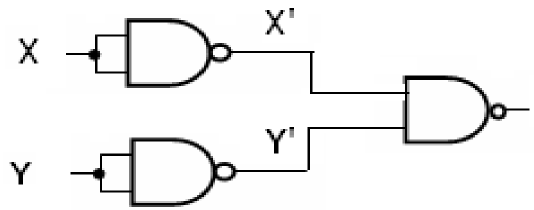


- ولو أضفنا بوابة NOT بعد بوابه NAND تصبح الدائرة المنطقية تمثل بوابة AND = X.Y
- قم بتنفيذ قانون النفي المزدوج involution على بوابة AND تحصل على بناء البوابة الجديدة:



بناء بوابة OR باستخدام بوابة NAND

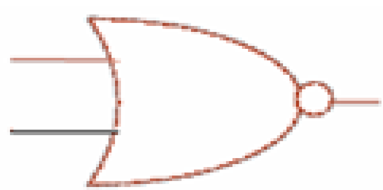
- قم بتنفيذ قانون النفي المزدوج involution على بوابة OR تحصل على بناء البوابة الجديدة:
- $X+Y= (X+Y)'' = (X'.Y')' = X'' + Y'' = X+Y$



$F = (X'.Y')' = X+Y$

بوابه NOR gate

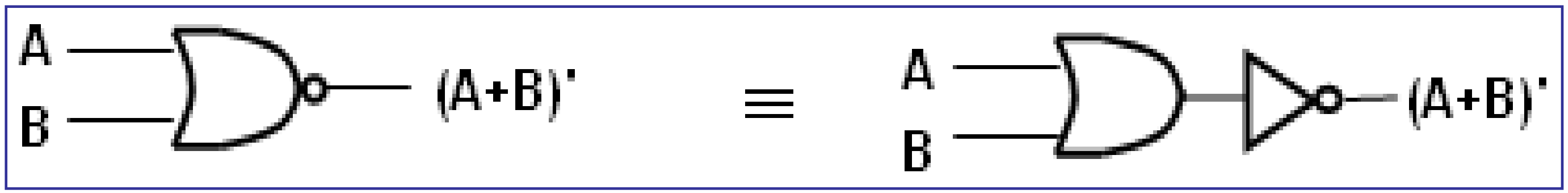
NOR gate: بوابة منطقية لها مدخلان على الأقل ومخرج واحد على الأقل وتستخدم لبناء الدوائر المنطقية الأساسية

NOR		X NOR Y $F = (x + y)' = x' . y'$

X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0

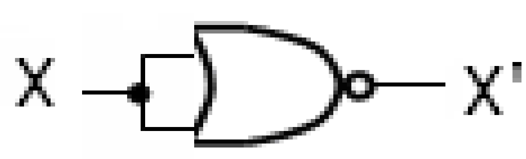
Standard Logic Gates (NAND & NOR)...cont.

- NOR gate is also self-sufficient (can build any logic circuit with it). It can be used to implement AND/OR/NOT.



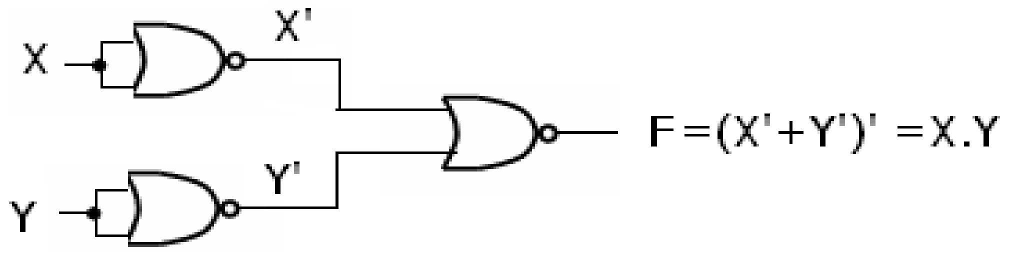
- Implementing an inverter using NOR gate:

بناء بوابة NOT باستخدام NOR



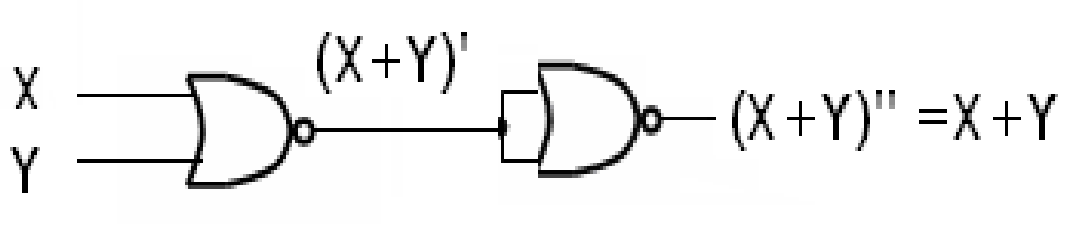
بناء بوابة AND باستخدام بوابة NOR

- $X.Y = (X.Y)'' = (X'+Y')' = X''.Y''=X.Y$



بناء بوابة OR باستخدام بوابة NOR

- $(X+Y)' \rightarrow (X+Y)''$



بناء الدوائر المنطقية

استنتاج الدوائر المنطقية من جدول الحقيقة

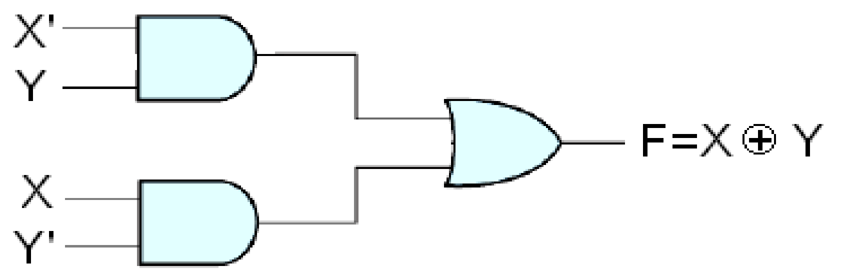
X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0

- استنتج الدالة F من جدول الحقيقة التالي ثم ارسمها ؟
- تكون النتيجة True عندما تكون المدخلات مختلفة

- F = XOR = X' Y + X Y' = X⊕Y

  → Exclusive OR

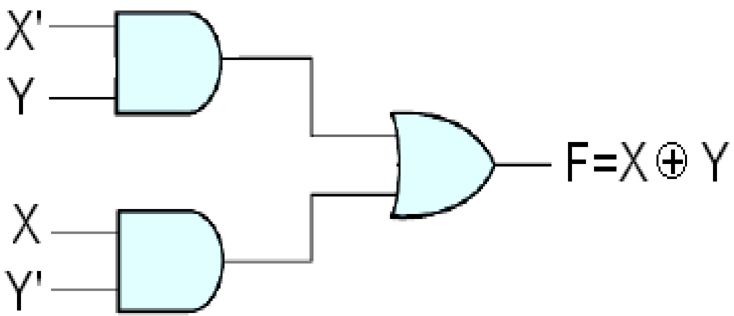
- Draw logic diagram

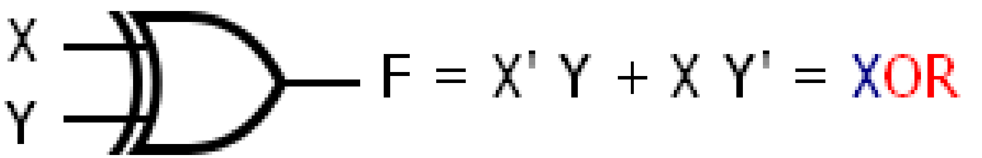


Exclusive OR
XOR

• وبسبب استخدام هذه الدائرة المتكرر في الحاسوب فقد تم تجميعها في دائرة مكافئة كما في الشكل التالي:

الدائرة 2 تكافئ الدائرة 1





الدائرة 2	الدائرة 1

Lecture-9

Complement of a Function

- The complement of F is F′
- Examples: Find the complement of the function F
- $F = (X'YZ' + X'Y'Z)$
- $F' = (X'YZ' + X'Y'Z)' = (X'YZ')' . (X'Y'Z)'$

$$= (X+Y'+Z)\ (X+Y+Z')$$

- Find the complement of the function F
- $F = [X.(Y'Z' + YZ)]$
- $F' = [X.(Y'Z' + YZ)]'$

$$= X' + (Y'Z' + YZ)'$$

$$= X' + (Y'Z')' . (YZ)' = X' + (Y+Z)\ (Y'+Z')$$

Example, Find the complement of the function F = X'YZ'+X'Y'Z

F	=	X'	.	Y	.	Z'	+	X'	.	Y'	.	Z
F'	=	(X	+	Y'	+	Z)	.	(X	+	Y	+	Z')

Standard Form of Logic Function

الإشكال المعياري للدالة المنطقية

إن أية دالة بولية تأخذ احد الشكلين المعياريين التاليين:

- ❑ Sum of product
- ❑ Product of sum

Example, F (X,Y,Z) = XYZ + X'Y'Z'

- وتسمى هذه العمليات مجموع حاصل ضرب المتغيرات

• مجموعة عمليات AND تجمعها عملية OR كما في الصيغة التالية:

• F= XYZ + X'Y'Z'

- وكل حد من هذه الحدود يسمى حد اصغر Minterm والحدود الصغرى تسمى Minterms وكل حد

منها يحتوي على كل المتغيرات في الدالة أو قد تكون بشكلها غير المعياري إذا كان احد الحدود لا يحتوي على كل المتغيرات

Sum of product

- يرمز عادة لصيغة Sum of product جمع المضاريب كما يأتي
- $F = \Sigma$ (m0, m3, m4…) مجموع حواصل ضرب المتغيرات
- Or $F = \Sigma$ (0, 3, 4…) الارقام ترمز الى ترتيب حدود الدالة

- مثال الحد 0 يعني 000 أو X'Y'Z'

- الحد 3 يرمز إلى 011 أو X'YZ وهكذا

Product of sum

- مجموعة عمليات OR تجمعها عملية AND كما في الصيغة التالية:
- Example, F=(X, Y, Z) = (X+Y'+Z) . (X'+Y'+Z) . (X+Y'+Z')

- وكل حد من هذه الحدود يسمى Maxterm وقد تكون بالشكل المعياري أو في غير المعياري ويرمز ل product of sum بالصيغة

- F(X, Y, Z) = Π (M2, M3, M6)
- هذا الرمز Π يرمز لصيغة حاصل ضرب المجاميع product of sum

- M تعني Maxtem والحدود بداخل الاقواس تدل على ترتيب الحدود المكونة للدالة مثلا الحد X+Y'+Z يدل على 0+1+0 وهكذا

Canonical and Standard Form

❑ Minterms

- A minterm (also called a standard product): an AND term consists of all literals in their normal form or in their complement form
- Example: the truth table of two binary variables *X* and *Y*,

x	y	F
0	0	X'Y'
0	1	X'Y
1	0	XY'
1	1	XY

F= X'Y'‘X'Y, XY', XY,
00, 01, 10, 11

- $F(X, Y) = X'Y' + X'Y + XY' + XY$

+ OR المضاريب تجمع حدودها بواسطة

- Sum of product → Standard product → minterms = same meaning
  - 2 Variables has $2^2 = 4$ minterms
  - n variables has 2^n minterms
- Example: of two binary variables x and y,
- The standard product (minterms) are : XY, XY', X'Y, X'Y'

Maxterms

- product of sum → standard sum → maxterm = same meaning
- n variables has 2^n maxterms
- Example: of two binary variables x and y,
- The standard sum (maxterms) are : X+Y, X+Y', X'+Y, X'+Y'

Example: the truth table of Minterm and maxterm for two variables

variable		Minterm Sum of product		Maxterm product of sum	
X	Y	Term	Designation	Term	Designation
0	0	X'Y'	*mo*	X+Y	M_0
0	1	X'Y	m_1	X+Y'	M_1
1	0	X Y'	m_2	X'+ Y	M_2
1	1	X Y	m_3	X'+Y'	M_3

- Each maxterm is the complement of its corresponding minterm, and vice versa
- Example: The minterm X'Y' represents the complement of the maxterm X+Y

Example: The truth table of Minterm and maxterm for three variables

			Minterms		Maxterms	
x	y	z	Term	Designation	Term	Designation
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

- minterms represent the sum of product → standard product
- maxterm represents the product of sum → standard sum

Canonical Forms

- يمكن التعبير عن اقتران الجبر البولي بواسطة
- Any Boolean function can be expressed as:
  - Truth table
  - A sum of minterms
  - A product of maxterms

Canonical Forms

• Example: Find the minterms that produce a 1 of the functions F1 & F2

F1= X'Y'Z+ XY'Z' + XYZ

001 100 111

= m1 + m4 + m7

= Σ (1, 4, 7)

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	$\boxed{1}$	0
0	1	0	0	0
0	1	1	0	$\boxed{1}$
1	0	0	$\boxed{1}$	0
1	0	1	0	$\boxed{1}$
1	1	0	0	$\boxed{1}$
1	1	1	$\boxed{1}$	$\boxed{1}$

F2 = X'YZ+ XY'Z+ XYZ'+XYZ

011 101 110 111

= Σ(m3 + m5 + m6 + m7)

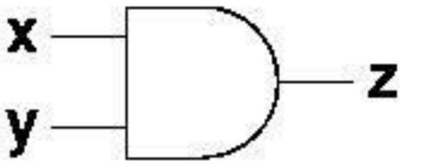
= Σ (3, 5, 6, 7)

• The truth table of three variables has 8 possible combinations (states)

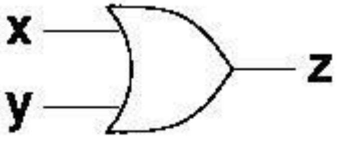
Lecture 10

Basic gates

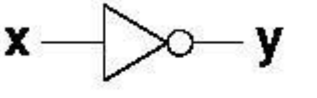
- The *and*-gate



- The *or*-gate

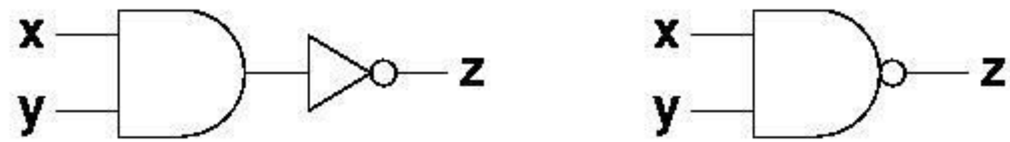


- The *inverter*

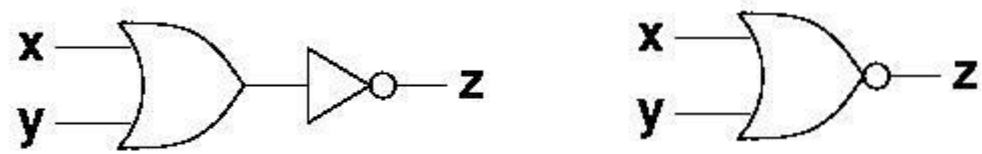


Combined gates

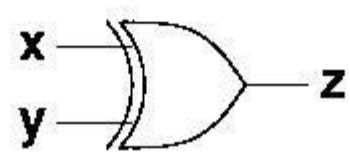
- The *nand*-gate



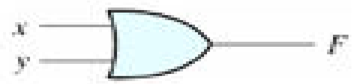
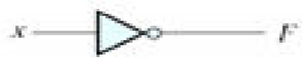
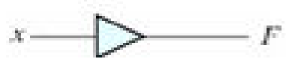
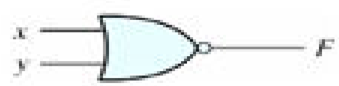
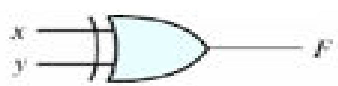
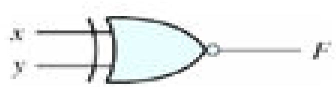
- The *nor*-gate



- The *exclusive-or*-gate



Digital Logic Gates

Name	Graphic Symbol	Algorithm function	Truth table
AND		F= X.Y	x y F 0 0 0 0 1 0 1 0 0 1 1 1
OR		F= X+Y	x y F 0 0 0 0 1 1 1 0 1 1 1 1
Inverter		F = X'	x F 0 1 1 0
Buffer		F = X	x F 0 0 1 1
NAND		F = (X.Y)'	x y F 0 0 1 0 1 1 1 0 1 1 1 0
NOR		F = (X+Y)'	x y F 0 0 1 0 1 0 1 0 0 1 1 0
XOR		F = XY'+X'Y = X ⊕ Y	x y F 0 0 0 0 1 1 1 0 1 1 1 0
XNOR		F = XY+X'Y' = (X ⊕ Y)'	x y F 0 0 1 0 1 0 1 0 0 1 1 1

Digital Logic Design

Gate-level Minimization

Chapter 3

Dr. Amer Abu-Jassar
Faculty of Science and Information Technology

Lecture 11

Introduction

- في الفصل السابق تم تبسيط واختصار الاقترانات بواسطة فرضيات وقوانين الجبر البولي

- إن الهدف من Boolean algebra هو تصميم الدوائر المنطقية وان أي اختصار للدالة البولية ما هو إلا اختصار للدائرة المنطقية التي سوف يتم بناءها

- تبسيط الدوال المنطقيةِ (عملية الاختصار) هو تقليل تكاليف بناء الدوائر المطلوبة وتقليل تعقيد هذه الدوائر ويستعمل بشكل رئيسي لتَخفيض عدد البوابات المنطقية لتصميم أفضل.

- ويوجد طرق اخرى للاختصار يتم شرحها وتفصيلها في هذا الفصل (الثالث)

- الدوال المنطقية أيضاً يُمْكِنُ أَنْ تُبسّطَ بطريقةِ الخريطةِ كخريطة كارنو

Introduction

- The Boolean functions also can be simplified by:

- map method as Karnaugh map or K-map.

- Simplification of Boolean functions is mainly used to reduce the gate count of a design.

- The map is made up of squares, with each square representing one minterm of the function.

- This produces a circuit diagram with a minimum number of gates and the minimum number of inputs to the gate.

- إنّ الخريطةَ تتُكَوّنُ من المربعاتِ، بكُلّ مربع يُمثّلُ حد واحد minterm من الدالة

- هذا يُنتجُ تخطيط دائرةِ منطقية بعدد اقل مِنْ البوابات gates وبعددِ أقل من المدخلات (المتغيرات) إلى البوابة المنطقية.

- Less number of gates means less power consumption, sometimes the circuit works <u>faster</u> and also when number of gates is reduced, cost also comes down.

- We will learn how to **simplify** Boolean functions to achieve <u>economical</u> (simpler) gate implementation.

- أقل عدد من gates يَعْني أقل استهلاكا للطاقة، أحياناً الدائرة تَعْملُ أسرعَ وأيضاً عندما يكون عددَ البوابات gates اقل، تكون الكلفةَ أيضاً اقل

- **تبسيط** الاقتران المنطقي مهم في تنفيذ البوابات المنطقية بطريقة اقتصادية سهلة وسريعة.

Simplification of the logical function

- There are many ways (methods) to simplify a logic design; some of them are:

- هناك عدّة أشكال لتَبْسيط تصميم الدوائر المنطقية والبعض مِنْها:

❑ Boolean algebra (الطرق البولية الجبرية)

❑ Karnaugh or K-Maps (طريقة كارنو ماب المربعات)

❑ Don't care condition

❑ Tabular methods (طريقة الجداول)

Karnaugh Maps (K Map)

- كارنو ماب Karnaugh Maps طريقة أو تقنية لتبسيط الدوال البولية وسنتعرف على إشكالها وطرق استخدامها في المنطق الرقمي

- Limited to no more than **6 variables.**

Standard Representation of Logic Functions

التمثيل القياسي للاقتران المنطقي

- الجدول التالي يبين العلاقة بين **الحدود الصغرى والكبرى** لدالة ذات 3 متغيرات:

- This table illustrates the relationship between:
  minterms SOP & maxterms POS)

			Minterms		Maxterms	
x	**y**	**z**	**Term**	**Designation**	**Term**	**Designation**
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Simplification of the number of adjacent squares

تبسيط عددِ المربعاتِ المتجاورةِ

- A larger number of adjacent squares are combined; we obtain a product term with fewer literals as follows: نتائج اختصارات المربعات

- 1 square = 1 minterm = three literals.
  - Example, X’Y’Z’ = three literals
- 2 adjacent squares = 1 term = two literals.
  - Example, X'Y'Z' + X'Y'Z = **X'Y'** = two literals
- 4 adjacent squares = one literal.
  - Example, X'Y'Z'+X'Y'Z+X'YZ'+X'YZ = **X'**
- 8 adjacent squares encompass the entire map and produce a function that is always equal to 1.

The Map Method

❑ تمثيل الدوال البولية باستعمال خارطة كارنو

- تستعمل خارطة كارنو لتبسيط الدوال المنطقية إلى اقل عدد ممكن من الحدود
- و هذا الاختصار يودي إلى تقليل الكلفة الاقتصادية للدوائر المنطقية
- تصميم وبناء الدوائر المنطقية بشكل فعال
- تصبح هذه الطريقة غير دقيقة و غير فعالة إذا زادت عدد المتغيرات عن 6 متغيرات
- وبهذه الحالة نبحث عن طرق أخرى للاختصار
- طريقة الخارطة تقدم طريقة سهلة وعمليات واضحة في اختصار الدوال البولية وتسمى ب كارنو ماب
- وكارنو ماب هو تخطيط مكون من مربعات
- كل مربع يمثل حد اسمه minterm من الاقتران (الدالة) الذي ننوي اختصاره
- لذلك أي اقتران بولي يمكن أن يكون:

Sum of minterms

- Sum of products (or product of sum) in the simplest form.
- A minimum number of terms يمكن اختصاره إلى اقل عدد من الحدود
- a minimum number of literals يمكن اختصاره إلى اقل عدد من المتغيرات
- The simplified expression may not be unique (يمكن وجود اكثر من اختصار)
- الاقتران المبسط ليس فريدا وانه من الممكن ايجاد تعبير أخر يرضي معيار الاختصار (طرق أخرى مثل don't care
- Gate-level minimization refers to the design task of finding an optimal gate-level implementation of Boolean functions describing a digital circuit.
- الاختصار يشير إلى مهمة التصميمَ لإيجاد تطبيقَ (تنفيذ) بوابة gate مثاليةِ المستويِ للدوال المنطقيةِ لتصِفُ الدائرة الرقمية.

Logic minimization منطق الاختصار

- Algebraic approach: lack specific rules
  - طريقة الاختصار الجبري تعتمد على الفرضيات وتفتقر إلى قواعد أخرى

❑ Karnaugh map approach (طريقة كارنو)

- A simple straight forward procedure (إجراء سهل وبسيط)
- A pictorial form of a truth table (يعتبر شكلا تصوري لجدول الحقيقة)
- A diagram made up of squares, each square represents one minterm
  - التخطيط مكون من مربعات وكل مربع يحتوي إلى حد واحد من الحدود الممثلة بجدول الحقيقة

Minimize function by K-map

اختصار الدالة بواسطة خارطة كارنو(شكل خارطة كارنو)

- تعتبر شكلا أخر لجدول الحقيقة ولكن بترتيب **جديد**
- تتعامل مع مربعات بعدد احتمالات جدول الحقيقة (عدد الحدود)
- إذا كانت الدالة ذات متغيرين فان الخارطة تحتوي 4 مربعات كل منها يمثل حالة من حالات جدول الحقيقة (حدا من حدود الاقتران)
- إذا كانت الدالة ذات 3 متغيرات فان الخارطة تحتوي 8 مربعات = 2^3 كل منها يمثل حالة من حالات جدول الحقيقة (حدا من حدود الاقتران)
- طريقة استخدام الخارطة لاختصار الدوال المنطقية وتصميم الدوائر المنطقية بعد الاختصار

Two-Variable Map of (X, Y)

http://www.google.jo/search?hl=ar&rlz=1T4ADFA_enJO435JO436&sa=X&ei=qyUoTvHzLZGAhQfb0q3hCQ&ved=0CBYQvwUoAQ&q=one+General+form+for+a+karnaugh+Map&spell=1&biw=1024&bih=494

خارطة كارنو لمتغيرين

- There are 4 minterms for 2 variable = 2^2
- X' = row 0; X= row 1 → توضيح نتائج الحدود بمربع من متغيرين
- A truth table in square diagram (مخطط المربعات يمثل جدول الحقيقة)
- The Map consists of 4 squares, (one for each minterm)
- من خلال الخارطة نجد **العلاقة بين المربعات** التي تحتوي على متغيرات الاقتران

- Two-variable has four minterms, and consists of four squares.

- Example: simplify the following Boolean functions using k-map?

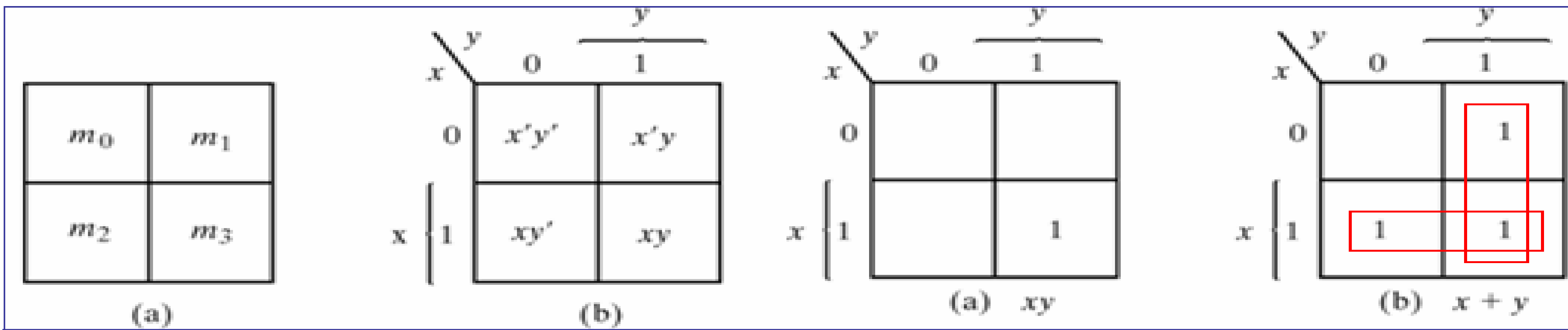
Truth table

- F(X, Y) = X+Y → OR gate
- F(X, Y) = X.Y → AND gate

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

- تكون الحالة 1 فقط عندما x و y =1 في عملية AND
- تكون الحالة 1 في كل الحالات عند العملية OR ما عدا عندما تكون المتغيرات اصفار
- Minterms: m0+m1 + m2 + m3 = X'Y', X'Y, XY', XY



- الصف الأول الافقي يمثل 'X والثاني يمثل X والعمود الأول يمثل 'Y والثاني يمثل Y

Exercises

$F = (X, Y) = X'Y + XY = Y$

$X'Y + XY = Y(X' + X) = Y$

$F = (X, Y) = XY' + X'Y' = Y'$

$F = (X, Y, Z) = \sum (2, 3, 6, 7) = XYZ + XYZ' + X'YZ + X'YZ' = Y$

Lecture 12

Three-Variable Map

Three-Variable Map → F(x, y, z)

- Eight minterms (8 squares)

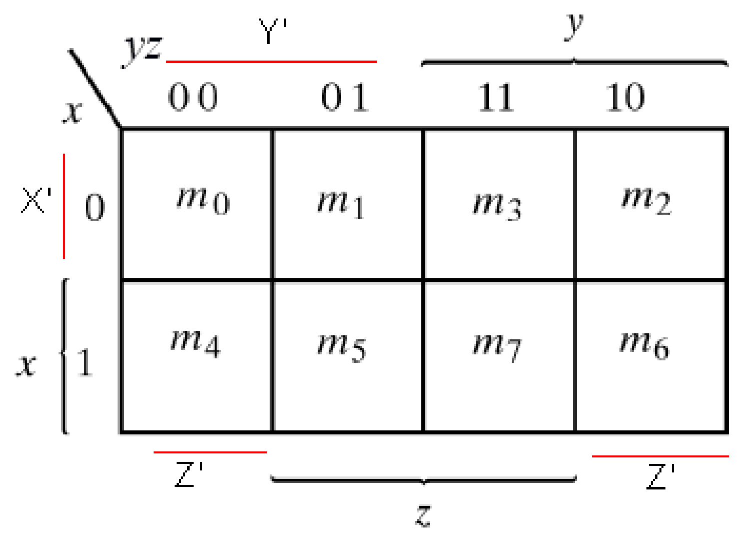
- Any two adjacent squares in the map differ by only one variable
  - في حالة 3 متغيرات فان الحدود تكون في 8 مربعات ممثلة على خارطة
  - وان أي مربعين متجاورين في الخارطة يختلفان فقط بمتغير واحد مختلفات في الحالة
  - يعني من الحالة الصحيحة إلى الحالة الخاطئة

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

$x \backslash yz$	00	01	11 (y)	10 (y)
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
x 1	$xy'z'$	$xy'z$	xyz	xyz'
		z	z	

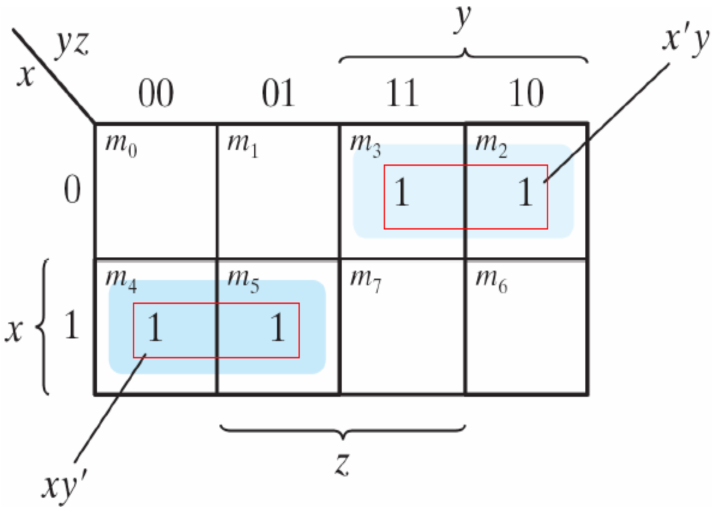
- Example: m5 and m7 can be simplified
- m5 + m7 = XY'Z+ XYZ = XZ(Y'+Y) = XZ

- باستخدام قانون التوزيع مباشره أو استخدام فرضية المحايد مع فرضية التماثل ليصبح الجواب = xz
- لاحظ بان الصف الأول يمثل X' والصف الثاني يمثل X
- العمودين على اليمين يمثلان المتغير Y والعمودين على اليسار يمثلان Y' بينما يمثل العمودان الأوسطان المتغير الثالث Z والعمودين على الإطراف يمثلان Z‘



- كما ويمكن التعبير عن هذه الحدود كما يلي:

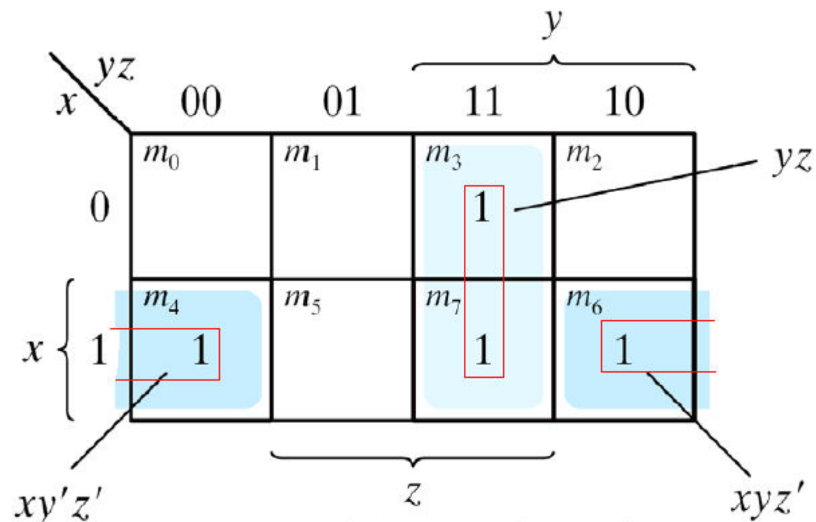
Exemple1: $F(X, Y, Z) = \Sigma (2, 3, 4,5) = X'Y + XY' = x \oplus y$



- Two squares in the map are considered to be adjacent even though the do not touch each other.
- **الحواف** تعتبر متجاورة لان الحدود مختلفة في حالة واحدة فقط كما في:
- $m0+m2= X' Y' Z' + X' Y Z' = X'Z' (Y'+Y)= X'Z'$
- $m4+m6= X Y' Z' + X Y Z'= XZ' + (Y'+Y)= XZ'$
- الاقتران قبل الاختصار:
- $F = m3+m2+m4+m5 = X'YZ+ X'YZ' + XY'Z' + XY'Z =$
- بعد اختصار المربعات المتجاورة للاقتران يصبح الاقتران يساوي:
- $F= X' Y + XY' = XOR \rightarrow X \oplus Y$ كما مر معك سابقا

- Exemple2: simplify the Boolean function:
- F(X, Y, Z) = Σ(3, 4, 6, 7)
- Draw the map

كل حد من حدود الدالة يمثل الحالة ا الصحيحة



Note: $xy'z' + xyz' = xz'$

$F(X, Y, Z) = \Sigma(3, 4, 6, 7) = X'YZ+XY'Z'+XYZ'+XYZ = YZ+XZ'$

$= m3 + m4 + m6 + m7$

- المربعات المتجاورة في الخارطة m4+m6 و m3+m7
- m3+m7= X'YZ + XYZ = YZ نحذف المتغير الذي تبدلت حالته
- m4+m6 = XY'Z'+XYZ' = XZ' نحذف المتغير الذي تبدلت حالته

F = YZ + XZ'

- Exemple3: simplify the Boolean function:
- $F(X, Y, Z) = \Sigma\,(0, 2, 4, 5,6) = Z' + XY'$
- $m0 = X'Y'Z'$ in row 0 and column 00 and so on.

- Exemple4: Simplify a Boolean function:
- $F(X, Y, Z) = X'\,Y\,Z' + XYZ + X\,Y'\,Z = \sum(2, 5, 7)$

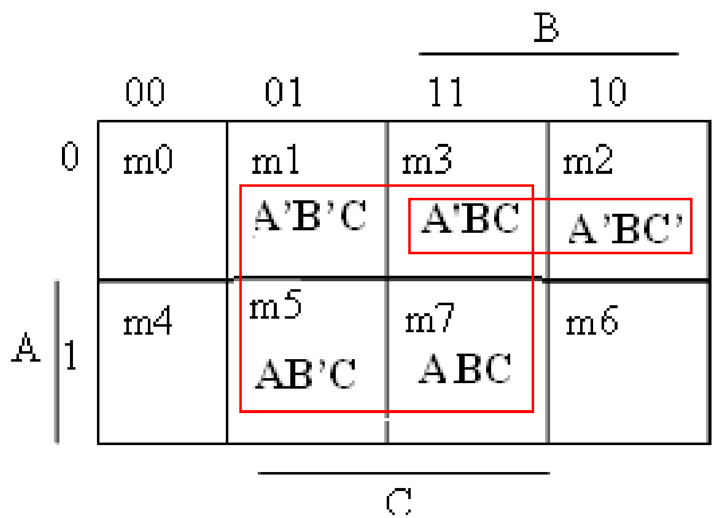
- Exemple5: Simplify a Boolean function:
- $F(A, B, C) = A'\,B\,C + A'\,B\,C' + A\,B'\,C = A'B + AB'C$

- في هذا المثال 4 مربعات متجاورة و هي m0m4m2m6 وينتج عنهم حرف واحد
- ويوجد أيضا مربعان متجاوران و هما m4m5 وينتج عنهم حرفان
- $m0+m4+m2+m6= \cancel{X'}\cancel{Y'}Z' + \cancel{X}\cancel{Y'}Z' + \cancel{X'}\cancel{Y}Z' + \cancel{X}\cancel{Y}Z'$
- احذف المتغير الذي تبدلت حالته فنحصل على حرف واحد بقانون التماثل = Z‘
- <u>في المثال أعلاه يجب علينا جمع الحدود لأختصار الدالة قدر المستطاع .</u>
- $m4+ m5 = XY'Z' + XY'Z= XY'$ نفس الطريقة السابقة
- ألان عملية تجميع الحدود المختصرة فينتج عن جمع مضاريب الدالة التالية
- $F(X, Y, Z) = \Sigma (0, 2, 4, 5,6) = Z' + XY'$
- Draw the circuit?

- Exemple6 : Simplify a Boolean function
- F = A'C + A'B + AB'C + BC
- Express it in sum of minterms → $\Sigma(1,2,3,5,7)$

- Exemple7 : Simplify a Boolean function :

- $F = \sum (0, 2, 3, 5, 7)$ using K-Map..



- Remember that, 4 adjacent squares = 1 term = one literal.
- بعد الاختصار ينتج الدالة التالية
- F = A'C + A'B + AB'C + BC = C + A'B
- m1+m3+m5+m7 = C ينتج حرف واحد من اختصار الأربع حدود
- m3+m2 = A'B ينتج حرفين من اختصار الحدين

Simplify جمع الأطراف

$F = (X, Y, Z) = \sum(0, 2,3,4,7) = Y' Z' + YZ + X' Z'$

$F= (X,Y,Z) = \sum(3,4,6,7) = X Z'+ YZ$

$F= (X,Y,Z) = \sum(0,2,5,7) = XZ + X' Z'$

$F= (X,Y,Z) = \sum (0,2,4,6) = Y'Z' + YZ' = Z' (Y+Y')= Z'$

$F= (X,Y,Z) = \sum(0,1,3,5,7) = X' Y' + Z$

Lecture 13

Four-Variable Map

Four-Variable Map

خارطة كارنو لأربع متغيرات

wx \ yz	00	01	11	10
00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
01	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
11	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

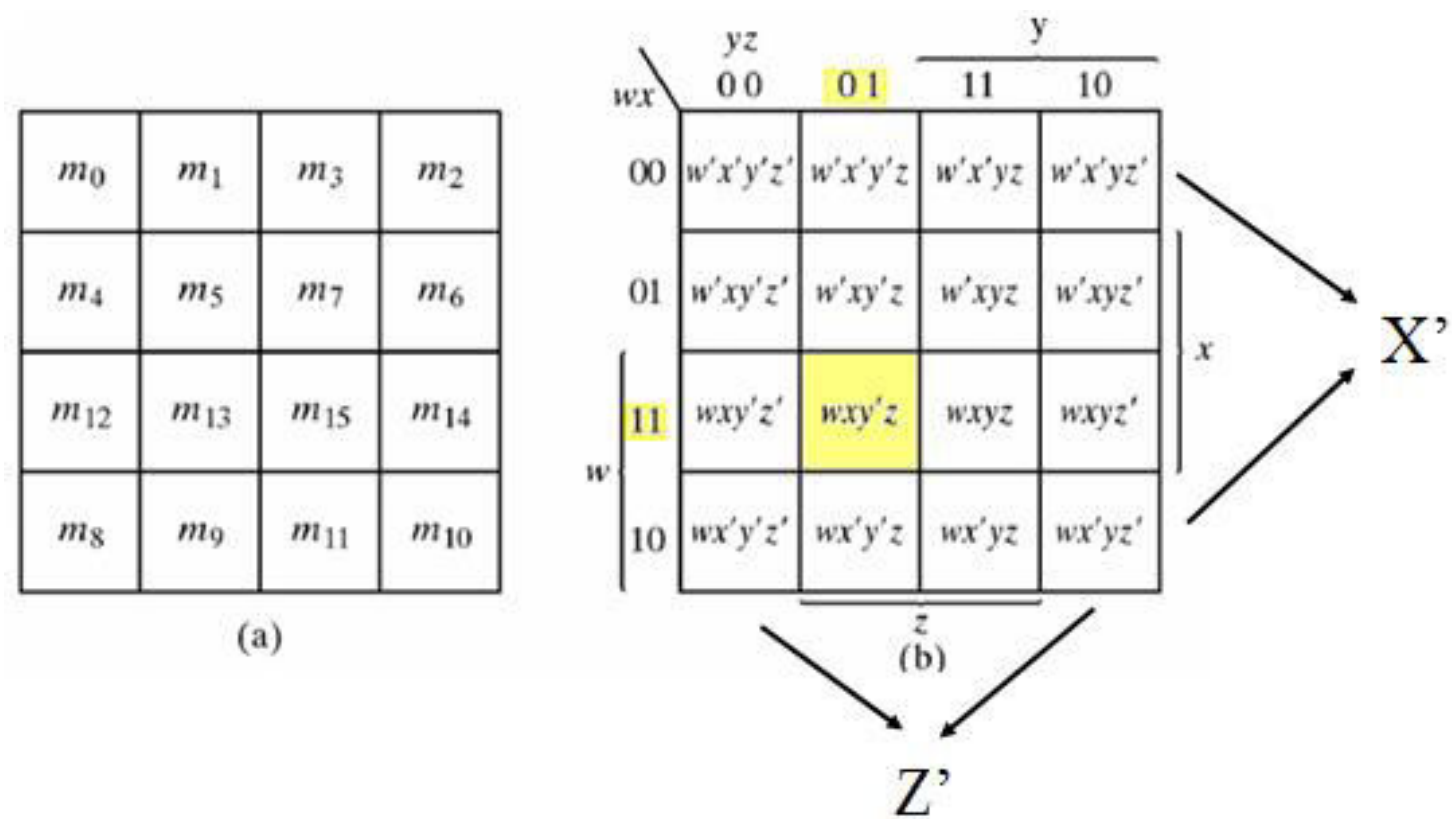
0	1	3	2
4	5	7	6
12	13	15	14
8	9	11	10

16 minterms
Combinations of 2, 4, 8, and 16 adjacent squares

Four-Variable Map

خارطة كارنو لأربع متغيرات

- 16 minterms
- Combinations of 2, 4, 8, and 16 adjacent squires



- The numbers of the third row is 11 and second column is 01, when concatenated, give the binary number 1101
- The binary equivalent to decimal 13 represents m_{13} in the third row and second column:
- خارطة كارنو لأربع متغيرات تنتج ما يلي:
  - 1 square = 1 minterm = 4 literals
  - 2 adjacent squares = 1 term = 3 literals
  - 4 adjacent squares = 1 term = 2 literals
  - 8 adjacent squares = 1 term = 1 literal
  - 16 adjacent squares = 1 → number 1

- Example: Simplify the Boolean Function:
- $F(X,Y,Z,W)=\Sigma(5,6,7,11,14,15)= X'YW + XZW+YZ$
- لاحظ المربعات المتجاورة واختصر حدودها فينتج:

- اختصر الدالة حسب الأكثرية المتجاورة من المربعات

- Example: Simplify the Boolean Function:
- $F= (X,Y,Z,W) = \sum (0,2,3,9,11,12,14,15)$
- $F= X'Y'W' + X'Y'Z+ XYW' + XYZ+ XY'W$

- Quiz: Simplify the Boolean Function using 4 variable?
- $F = A'B'C' + B'CD' + A'BCD' + AB'C' =$

- $F = (X,Y,Z,W) = \sum (4,5,6,9,11,12,13)$

Lecture 14

Gate Implementation

Gate implementation

استخدام خارطة كارنو لتصميم الدوائر المنطقية

- F (A, B, C, D) $=\sum(0,1,2,5,8,9,10) = \mathbf{B'D' + B'C' + A'C'D}$
- The implementation of the simplified expression obtained in SOP & POS is:

- **SOP** $\rightarrow$ minterms of 1's is as follows:
- $\mathbf{F}=\sum(0,1,2,5,8,9,10) = \mathbf{B'D' + B'C' + A'C'D}$

- Quiz:

- Simplify the following expression

  $F(A, B, C, D) = \sum(0,1,2,3,8,9,10,11,14,15) =$

  in SOP by using k-map and then draw the Gate implementation of the function?

Don't-Care Conditions

- There are many ways (methods) to simplify a logic design; one of them is Don't-Care Conditions
- It is a combination of variables whose logical value is not specified, used X
- Example: in BCD code; the code [1010 ……...1111], this means that, from 10 to 15 known as don't car conditions.
- The don't care conditions can be utilized in logic minimization for further simplification of the Boolean expression.
- It can be implemented as 0 or 1

- تستعمل هذه الميزة من اجل تبسيط أكثر للاقتران ويمكن تمثيل الاقتران باكثر من اختصار وجميع الحالات صحيحة.

- لا تكون قيمتها مخصصة في الخارطة بل تمثل 1 او 0 حسب الحالة المطلوب معالجتها سواءا حالة جمع المضاريب تمثل x بـ 1 أو في حالة ضرب المجاميع تمثل x بـ 0

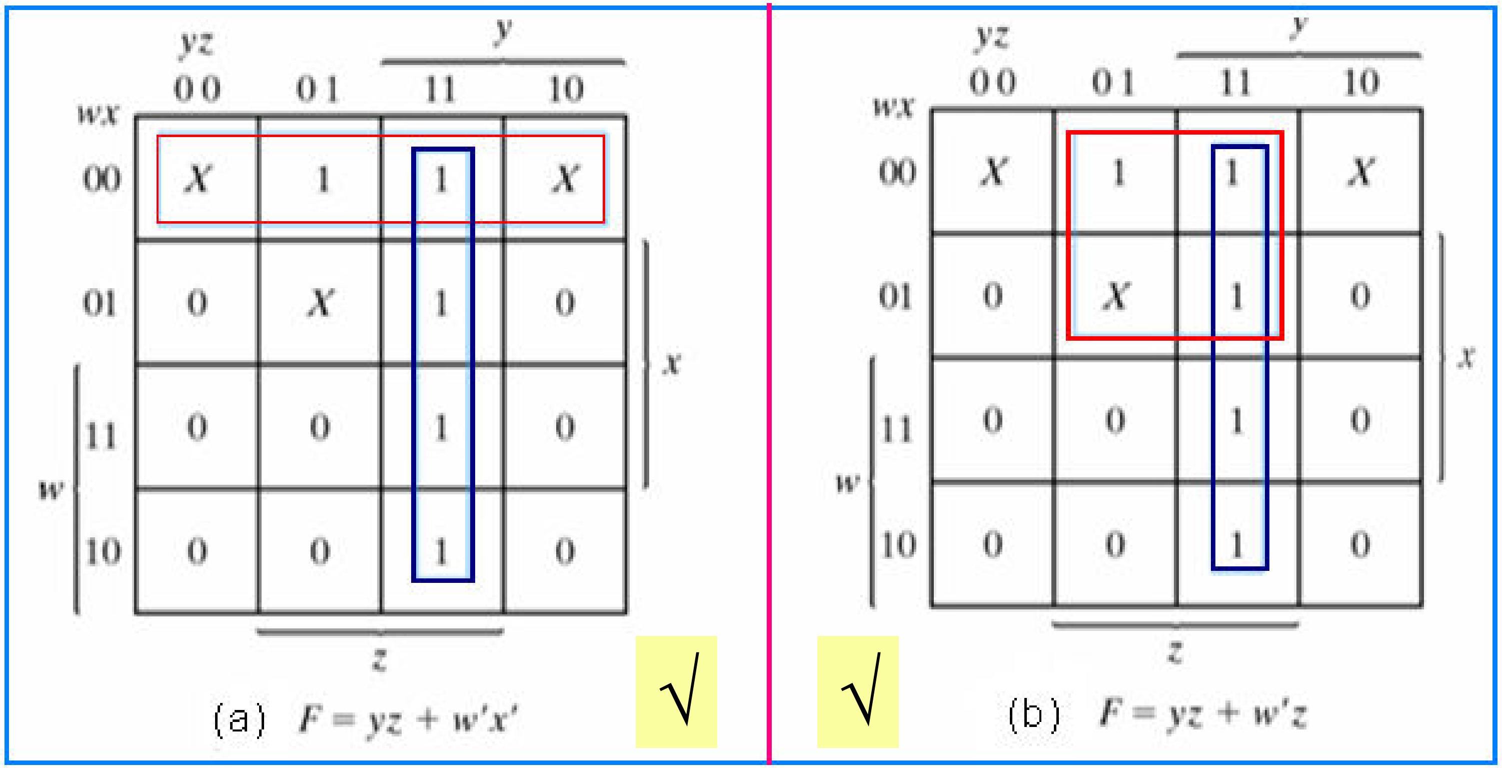
- لتمييز حالة don't care عن حالة ال 0 وال 1 نضع في المربع X

- X تشير انه ليس مهم حالة X في أن تكون في المربع المخصص لها يمثل سوءا 1 أو 0

- عند تبسيط الاقتران لك الحرية ان تختار حالة X ليكون 0 أو 1 للمربعات المتجاورة من اجل المساعدة في تبسيط الاقتران كما بالمثال التالي

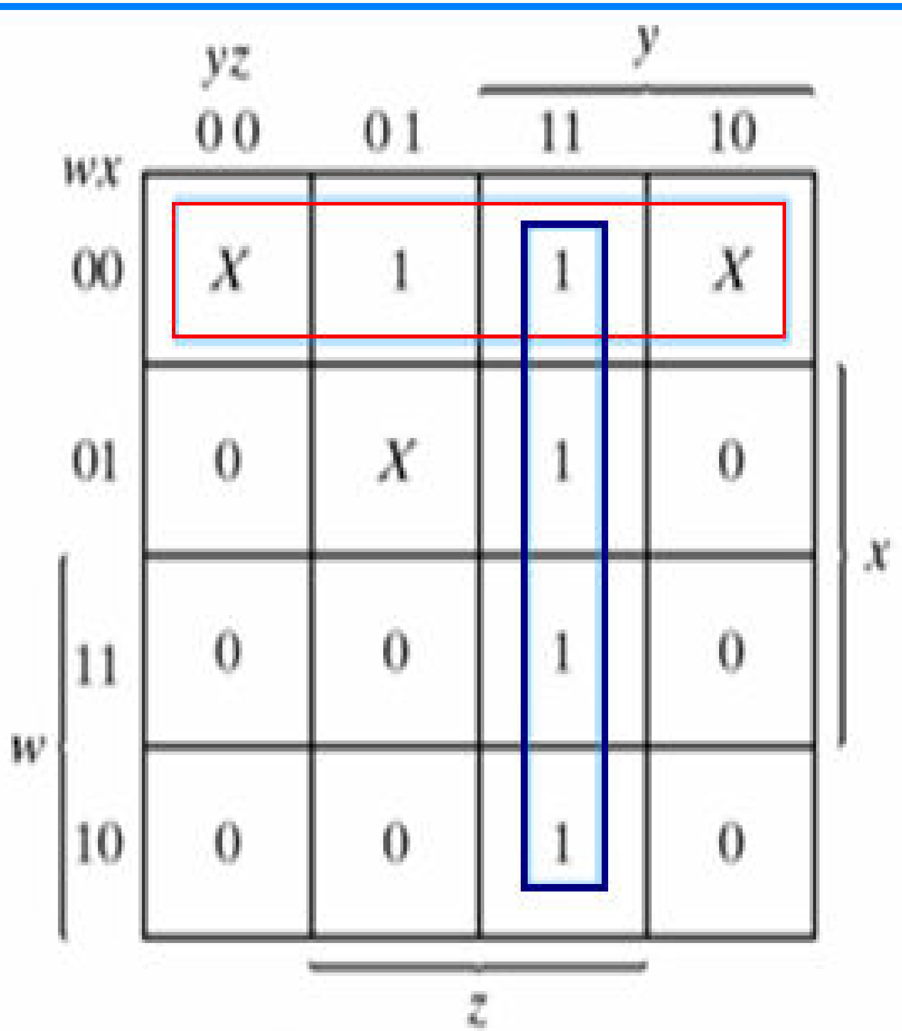
Don't-Care Conditions

- Example: Simplify the Boolean function
- $F(W, X, Y, Z) = \sum(1, 3, 7, 11, 15)$ which has the don't care conditions: $d(w, x, y, z) = \sum(0, 2, 5)$

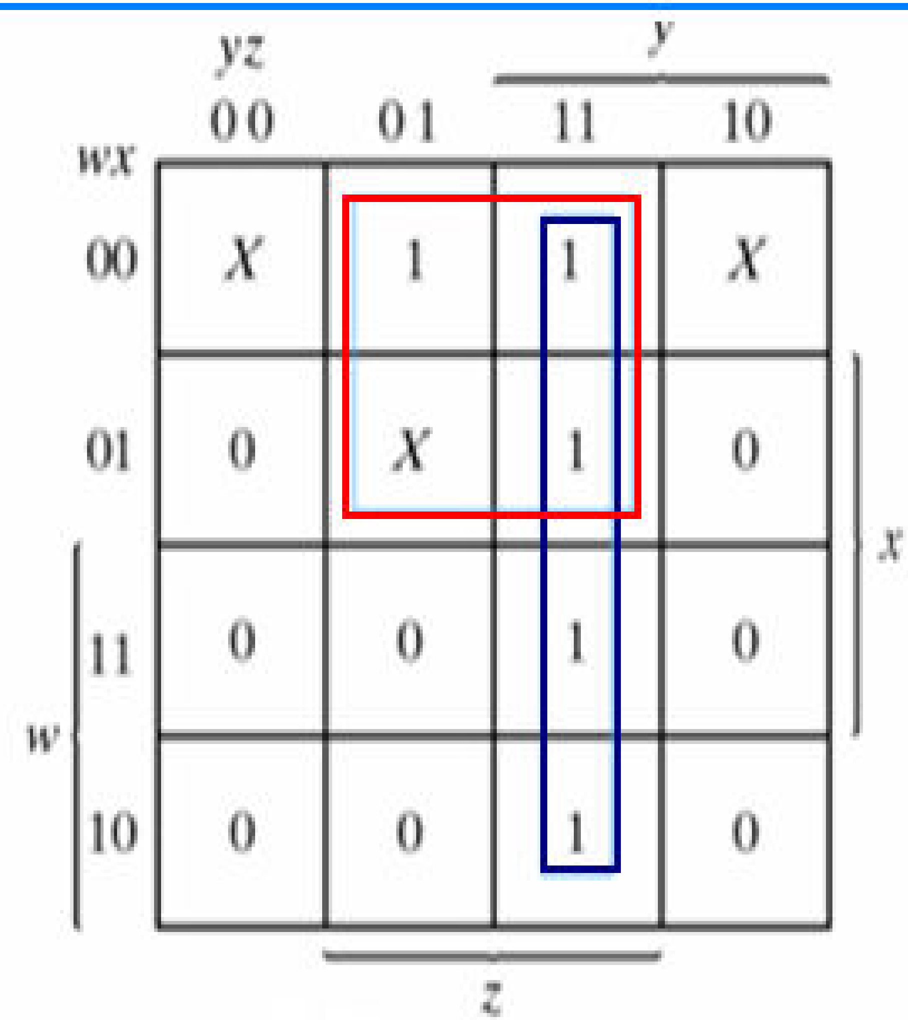


(a) $F = yz + w'x'$

(b) $F = yz + w'z$



(a) $F = yz + w'x'$



(b) $F = yz + w'z$

- In part (a) with minterms 0000 = x and 0010 =x
- الحالة الأولى كان don't care يمثل المربع 0 و 2 وبعد الاختصار حصلنا على الاقتران التالي:
- F (X, Y,Z, W) = YZ+W'X' → SOP
- In part (b) with minterm 5
- الحالة الثانية الdon't care يمثل المربع 5 (0101) وبعد الاختصار أصبح الاقتران
- F= YZ + W'Z → SOP

- Example: Simplify the Boolean function
- F(W, X, Y, Z) = $\sum$(**0, 2, 4, 6,8**) which has the don't care conditions: d(w, x, y, z) = $\sum$(**10, 12, 14**)
- F= Z' after minimization

wx \ yz	00	01	11	10
00	1			1
01	1			1
11	x			x
10	1			x

- **Question**: Simplify the Boolean function: F= Σ (1, 2, 3, 4, 5,7) witch has the don’t care condition d= $\sum$(0,6)

Questions

3.1) Simplify the following Boolean functions, using three-variable maps?

a) $F(X, Y, Z) = \sum(0, 2, 6, 4)$

d) $F(X, Y, Z) = \sum(3, 5, 6, 7)$

3.2) Simplify the following Boolean functions, using three-variable maps?

a) $F(X, Y, Z) = \sum(0, 1, 5, 7)$

f) $F(X, Y, Z) = \sum(1, 4, 5, 6, 7)$

3.3) Simplify the following Boolean expressions, using three-variable maps?

a) $F(X, Y, Z) = XY+X'Y'Z'+X'YZ'$

d) $F(X, Y, Z) = XYZ+ X'Y'Z+XY'Z'$

3.4) Simplify the following Boolean functions, using Karnaugh maps?

b) $F(A, B, C, D) =\sum(4, 6, 7, 15)$

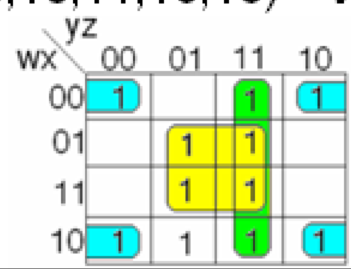
e) $F(A, B, C, D) =\sum(1, 4, 5, 6, 7, 13)$

3.5) Simplify the following Boolean functions, using 4-variable maps?

a) $F(W, X, Y, Z) = \sum(1, 4, 5, 6, 12, 14, 15)$

c) $F(W, X, Y, Z) = \sum(0, 1, 4, 5, 6, 7, 8, 9)$

F) $F(W,X,Y,Z) = (0,2,3,5,7,8,9,10,11,13,15) = WX' + YZ + XZ + X'Z'$



Questions

3.14) **Simplify** the following Boolean expressions

$F = B'C'D' + AB'CD' + BC'D + A'BCD$ to

1) sum of products

2) product of sum using Karnough map:

3.15) Simplify the following Boolean functions F, together with the **don't care condition** d, and the express the simplified function in sum of minterms form?

a) $F(X, Y, Z) = \sum(2, 3, 4, 6, 7)$ and $d(x, y, z) = \sum(0, 1, 5)$

b) $F(A, B, C, D) = \sum(0, 6, 8, 13, 14)$

with $d(A, B, C, D) = \sum(2,4,10)$

Lecture 15

NAND & NOR Implementation

NAND & NOR Implementation

NAND Circuits

- AND-invert = NAND → invert-OR = NAND
- OR-invert = NOR → invert-AND = NOR

- NAND and NOR gates are easier to fabricate with electronic components and are the basic gates used in all IC digital logic families.
- NAND gate is a universal gate because any digital system can be implemented with it.
- NAND gate can be used to express the basic gates, NOT, AND, and OR.
- The logic operation of AND, OR, and complement (NOT) can be obtained with NAND gates alone.

The graphic symbol of the other logic operations is

NAND

NOR

XOR

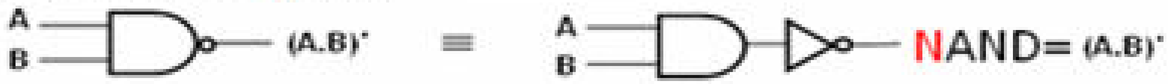
XNOR

جدول الحقيقة (Truth table) يوضح هذه العمليات لكي تعرف العلاقة بين المتغيرات ونتائج العمليات المشتقة من العمليات الأساسية

Variables X	Variables Y	NAND $F=(XY)'$	NOR $F=(X+Y)'$	XOR $F=X \oplus Y$	XNOR $F=X \odot Y$
0	0	1	1	0	1
0	1	1	0	1	0
1	0	1	0	1	0
1	1	0	0	0	1

NAND gate is self-sufficient (can build any logic circuit with it). It can be used to implement AND/OR/NOT.

- الدوائر المنطقية الأساسية البديلة التي تتركب باستخدام NAND gate
- (AND)' = NAND

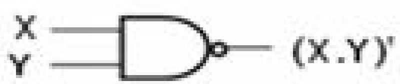


1. بناء بوابة NOT باستخدام NAND كما في الشكل
المكمل باستخدام بوابة

- Implementing an inverter using NAND gate:

2. بناء بوابة AND باستخدام بوابة NAND

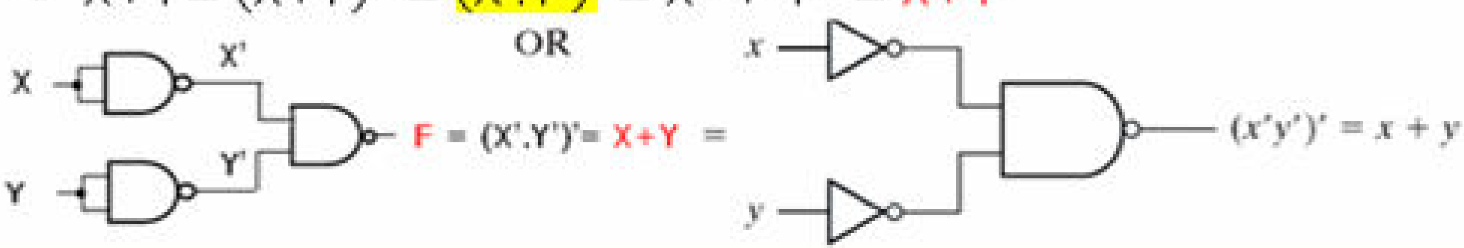
NAND



- ولو أضفنا بوابة NOT بعد بوابة NAND تصبح الدائرة المنطقية تمثل بوابة AND = X.Y

3. بناء بوابة OR باستخدام بوابة NAND

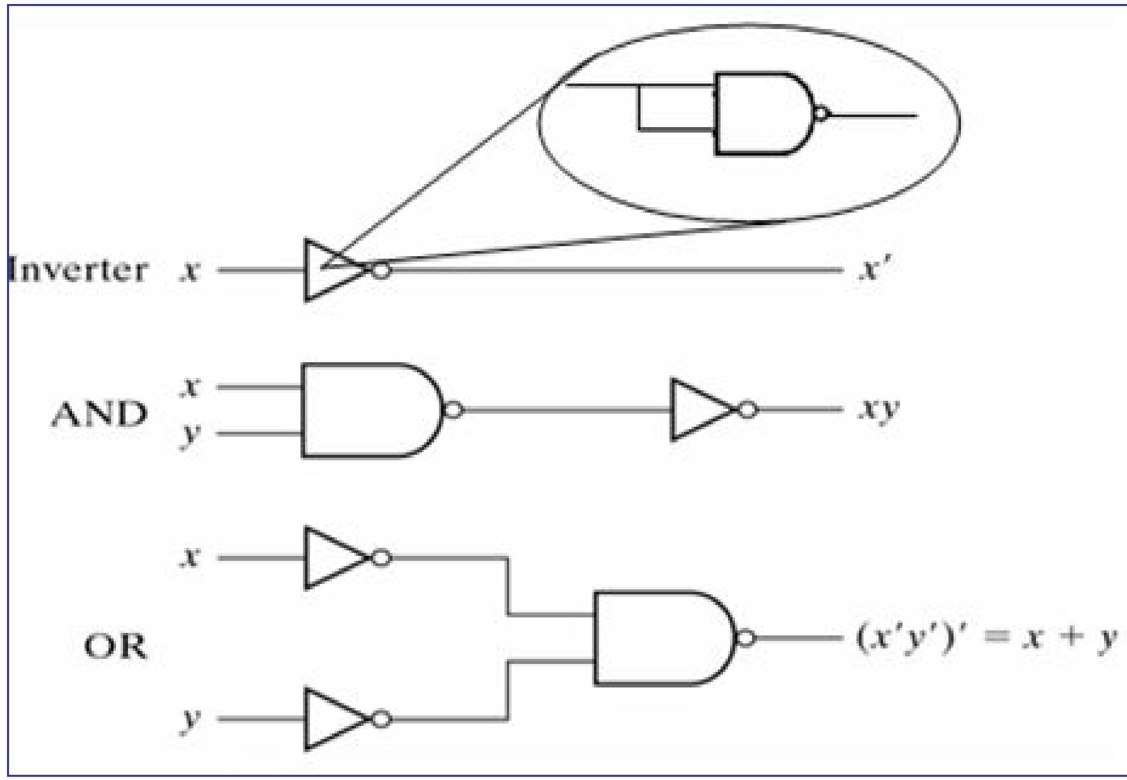
- X+Y= (X+Y)'' = (X'.Y')' = X'' + Y'' = X+Y



NAND Implementation

- logical operations of AND, OR, and complement (NOT)
- The logical operations of AND, OR, and complement can be obtained with NAND gate alone and NAND gate can implement any digital system.

بتطبيق قانون ديمورقان تحصل على الدوائر المنطقية الأساسية المنفذة باستخدام NAND



NAND Implementation…cont.

- The complement operation is obtained from a one input NAND gate that behaves exactly like as inverter.

- AND operation requires → two NAND gates. The first produces the NAND operation and the second inverts the logical sense of the signal.

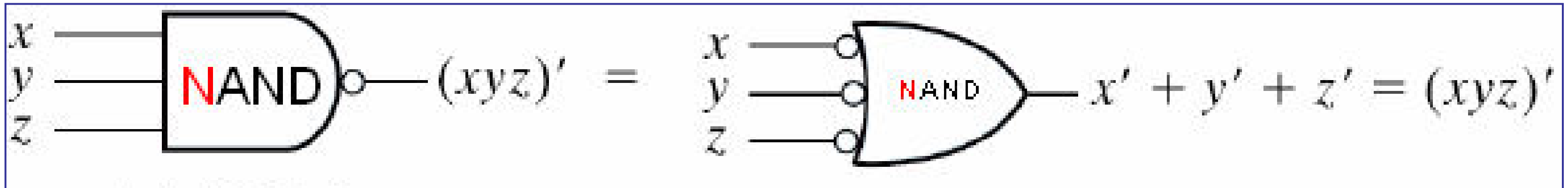
- The OR operation is achieved through a NAND gate with additional inverters in each input.

Two graphic symbols for a NAND gate

- NAND can represented as AND- invert or invert-OR
- تذكر قانون ديمورقان عند التحويل من AND-invert إلى invert-OR

Theorem 5, DeMorgan	$(X + Y)' = X'Y'$	$(X \cdot Y)' = X' + Y'$

- In part (b), we can place a bubble (NOT) in each input and apply the DeMorgan's theorem, then get a Boolean function in NAND type.
- When the both symbols are mixed in the same diagram, the circuit is said to be in mixed notation (التَرقيم المُخْتَلَط).
- بإتباع قانون Demorgan theory تم عمل الدائرة (b) بعمل invert بمدخل ال OR للمتغيرات و هذا التمثيل مهم في تحليل وتصميم دوائر NAND = (AND-INVERT) وعندما تتوفر هذه الرموز a,b في التصميم لنفس التخطيط يسمى بالترقيم المختلط

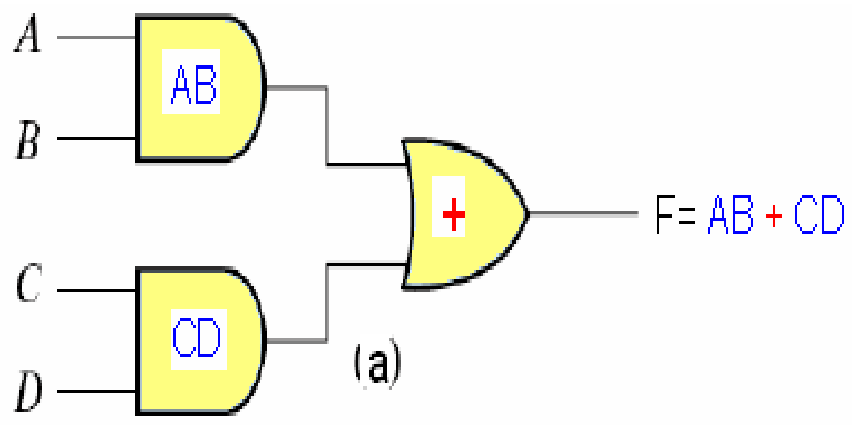


(a) AND-invert (b) Invert-OR

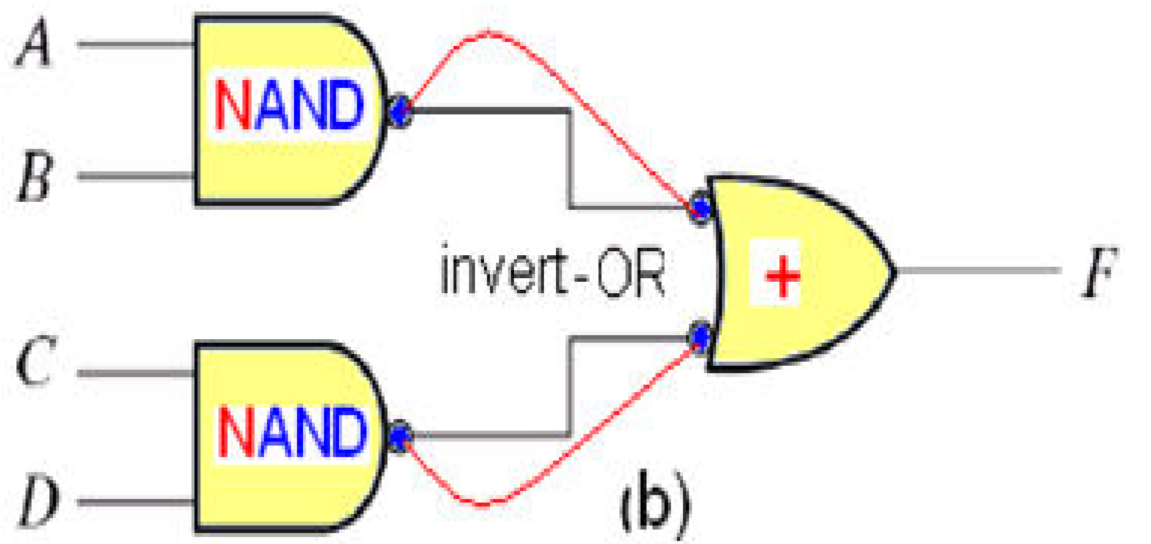
Two-level Implementation

- التنفيذ في حالة المستوى الأول والثاني باستخدام NAND
- The implementation of Boolean function with NAND gates requires that the function be in sum-of-product form SOP (This means that)
- لرؤية العلاقة بين حدود جمع المضاريب وتنفيذ دائرة NAND المكافئة نأخذ المثال التالي:

- **Example1**: Implement the expression F = AB+ CD with NAND gate:
- There are three ways to implement F with NAND gate. The following diagrams are equivalent and implemented the function F
- The function is implemented with
  - The AND and OR gates in **sop**

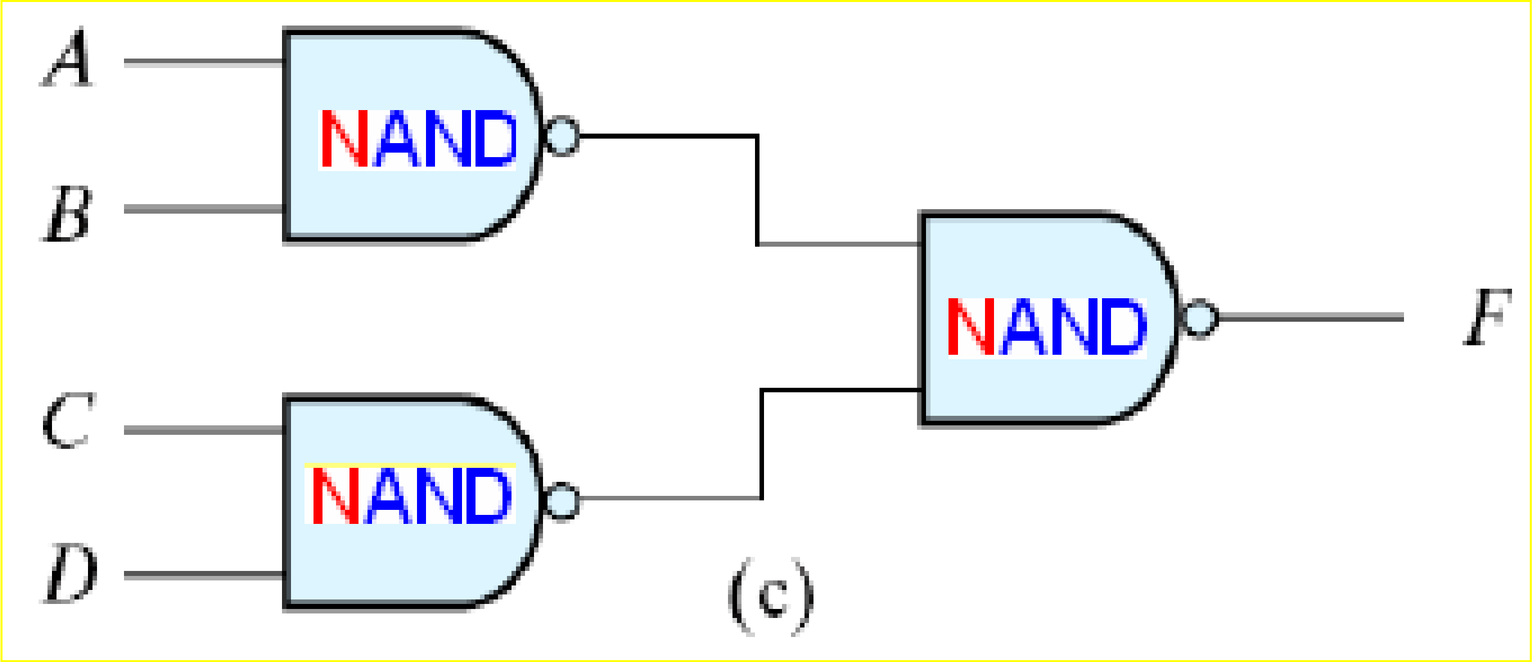


- The AND gates are replaced be NAND gates and the OR gate is replaced by a NAND gate with an invert-OR graphic symbol.



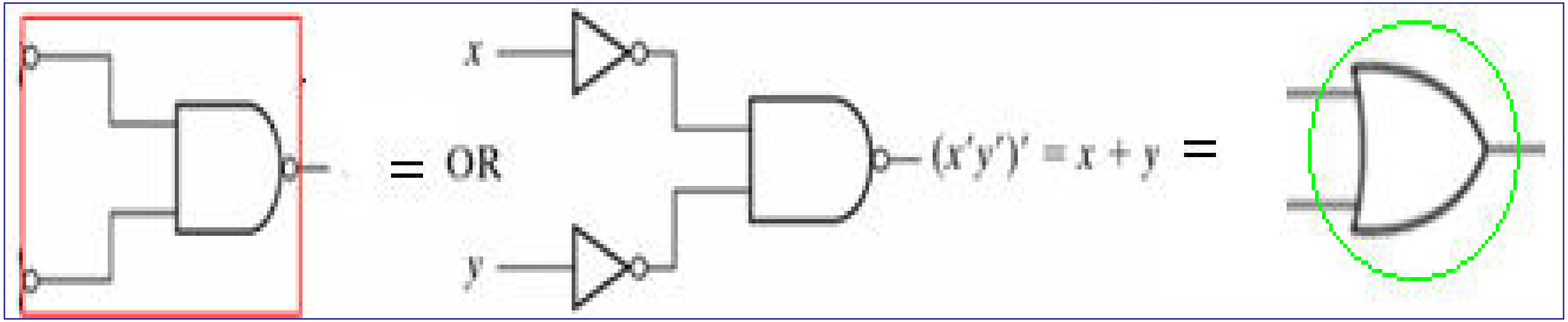
F = [(AND)‘]' + [(AND)‘]' = [(AB)‘]' + [(CD)‘]' = AB+CD

- The output NAND gate is redrawn with the AND-invert graphic symbol (to change invert-OR into AND-invert gate) as follows:



- NAND = (AND)‘ → F = [(AND)'. (AND)'] ' = [(AB)' . (CD)']' = [AND. AND]' F= ((AB)' (CD)')' = (AB)'' + (CD)'' = AB+CD حسب قانون ديمورقان

- لاحظ في الشكل (b) أن النفي المزدوج يُلغى، وان OR في الشكل (a) = الشكل المشار إليه في (c) بتطبيق DeMorgan theorem كما في الشكل التالي

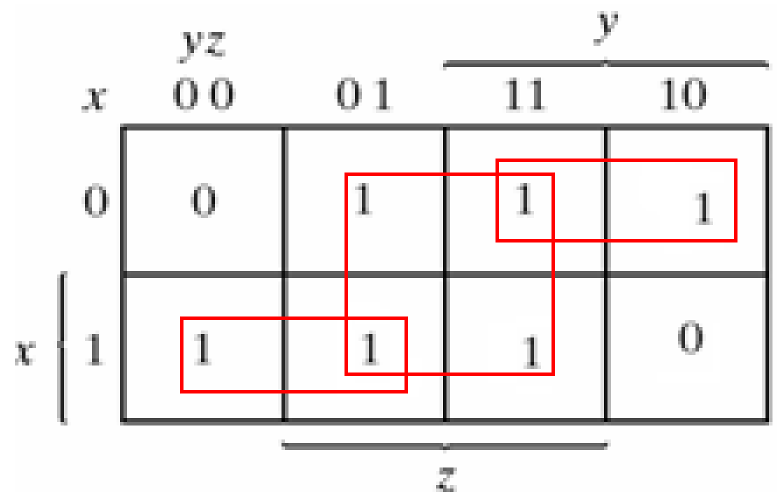


(X' Y')' = X" + Y" = X+Y

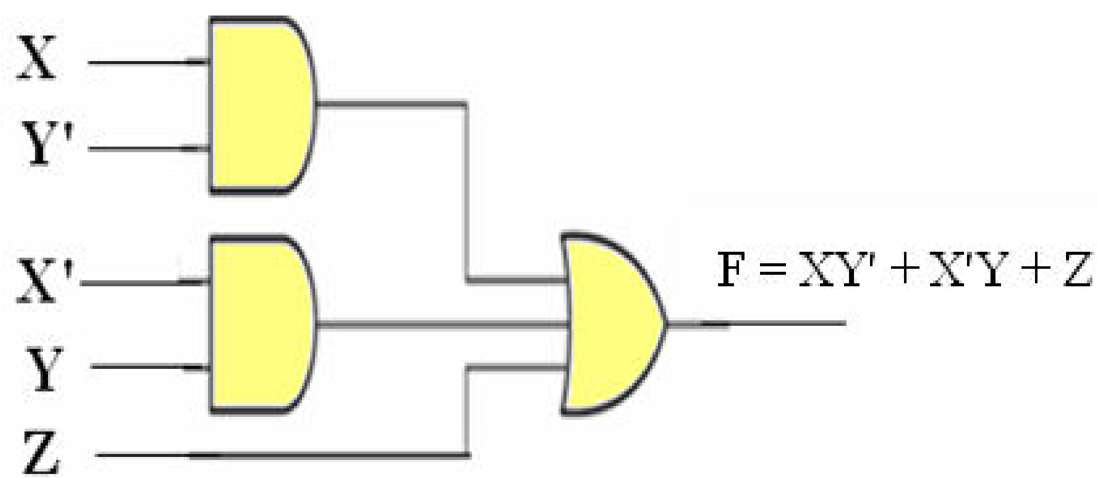
Example2: implement the following Boolean function

F(X, Y, Z) = Σ (1, 2, 3, 4, 5,7) with two level using NAND gate The procedure for obtaining the logic diagram from a Boolean function is: (اتبع الخطوات التالية في تنفيذ الاقتران باستخدام NAND)

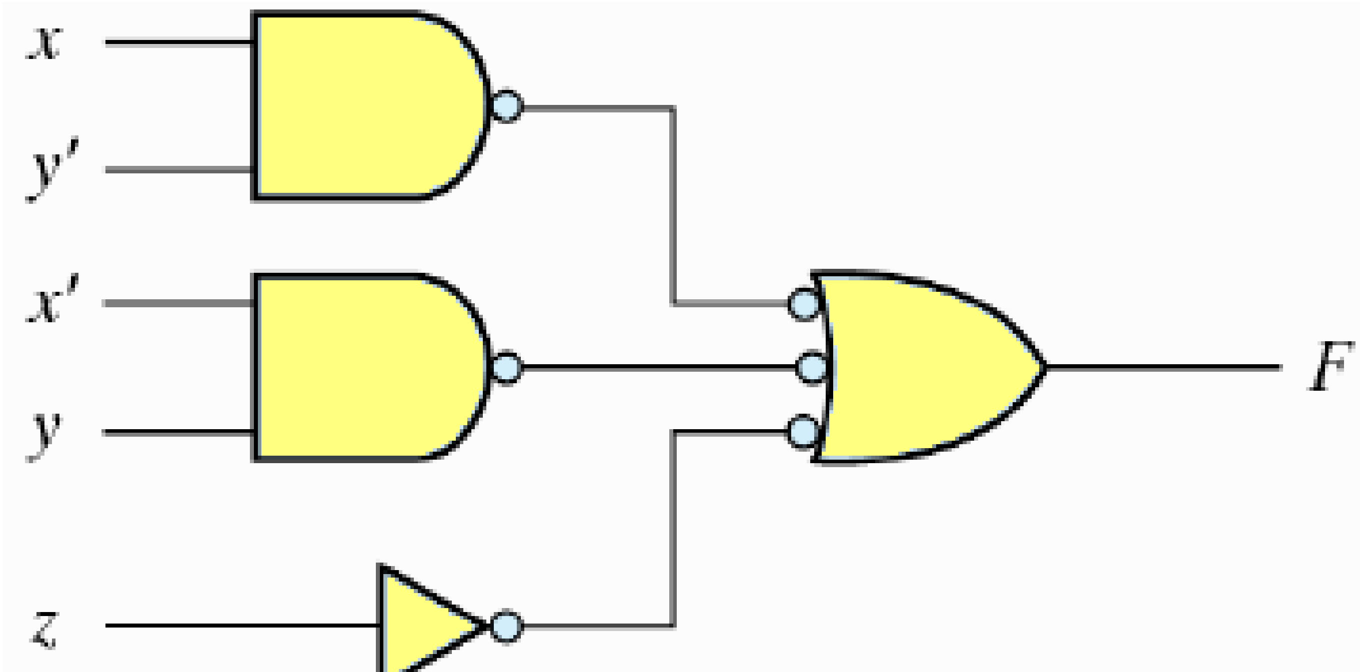
- Example3: Simplify the function and express it in sum-of-product using Karnaugh map by representing the following: F= Σ (1, 2, 3, 4, 5,7)



- F(X, Y, Z) = XY' + X'Y + Z = (X XOR Y)+Z

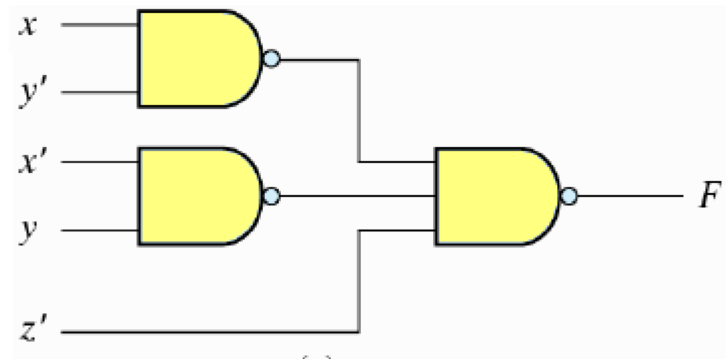


- Draw a single gate using the AND-inverter or the inverter-OR graphic symbol in the second level
- with inputs coming from outputs of the first-level gates.
- In (b) represents the mixed notation in (inverter-OR)
- استخدمنا الرمز invert-OR لتمثيل ال NAND مع الدائرة في حالة
- mixed notation



(b)

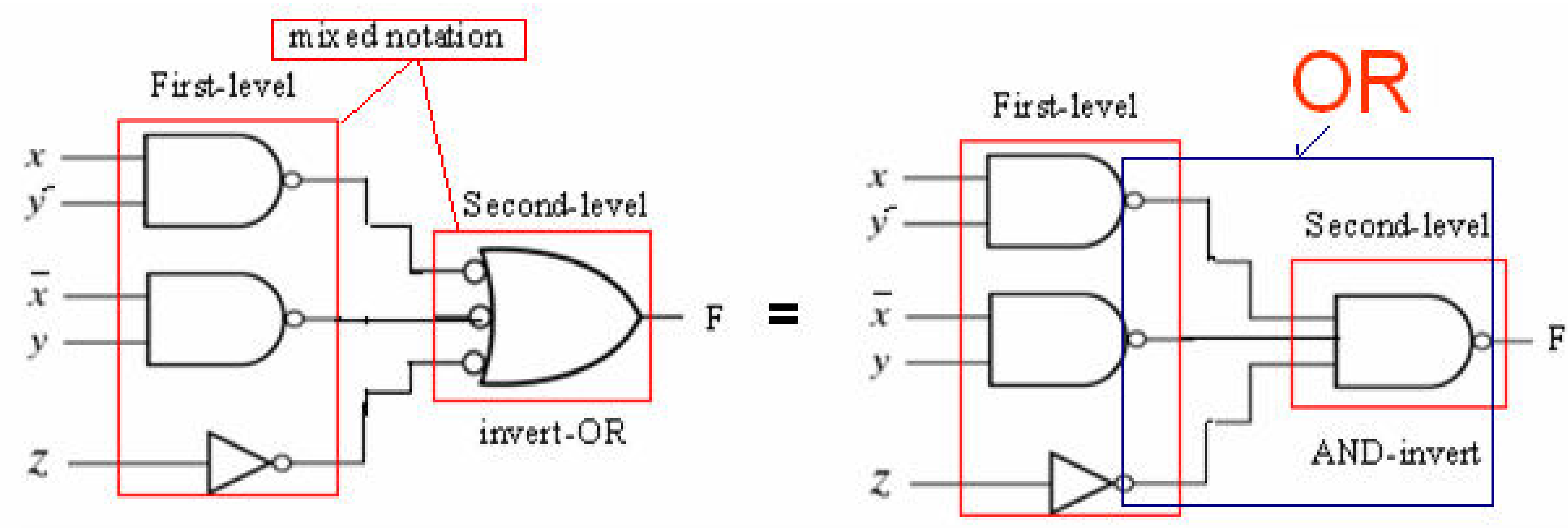
- The function implemented with two-level NAND gate as follows: AND-inverter
- بدل الرمز invert-OR بـالرمز AND-invert لتحصل على الدائرة التالية
- F = XY' + X'Y + Z



(c)

- A term z with a single literal requires an inverter in the first level, if the single literal is complemented; it can be connected directly to an input of the second-level NAND gate.
- إذا كان المتغير وحيد مثل Z يحتاج إلى inverter في المستوى الأول أما إذا كان Z' أي له مكمل يمكن ربطه مباشرة كمدخل إلى المستوى الثاني

- The standard form resulted in two-level implementation:
- This diagram is for more explanation?



Invert OR invert → AND

Invert AND invert → OR

Lecture 16

Nondegenerate forms

Nondegenerate Forms

The forms considered thus far are not the only ways in which a Boolean expression can be implemented in two levels with the four types of gates: AND, OR, NAND, and NOR. In fact, if we assign one type of gate to the first level and one type of gate to the second level, there are 16 possible 2-level combinations because the same type of gate can be used at both levels (as in the NAND-NAND or in the NOR-NOR implementation). But eight of these 16 forms are **degenerate** since they reduce to a single operation. For example, the AND-AND and OR-OR forms degenerate to just AND and OR, respectively.

The remaining eight **nondegenerate** forms are listed in Figure 5.5. The gate listed first in each of these forms constitutes one of the gates in the first level of the implementation, whereas the gate listed second constitutes a single gate in the second level. Any two forms listed on the same line in Figure 5.5 are duals of each other. The arrows indicate which forms can be obtained from others by successive applications of DeMorgan's laws.

The eight nondegenerate forms are separated into two groups of four related forms each. We call one group the AND-OR family since it contains that form for expressing a function. By the same token, the second group is called the OR-AND family. Conversion between forms in the same family is relatively simple, but interfamily conversion is not as easy. One way that a family-to-family conversion can be accomplished is to go back to the truth table (or K-map) of the original expression. The following example illustrates how we can convert to different nondegenerate forms using DeMorgan's laws.

Nondegenerate forms

- 16 possible combinations of two-level forms
  - Eight of them: degenerate forms = a single operation
    - AND-AND, AND-NAND, OR-OR, OR-NOR, NAND-OR, NAND-NOR, NOR-AND, NOR-NAND.
  - The eight non-degenerate forms
    - AND-OR, OR-AND, NAND-NAND, NOR-NOR, NOR-OR, NAND-AND, OR-NAND, AND-NOR.
    - AND-OR and NAND-NAND = sum of products.
    - OR-AND and NOR-NOR = product of sums.
    - NOR-OR, NAND-AND, OR-NAND, AND-NOR = ?

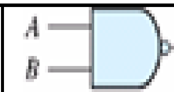
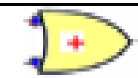
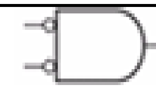
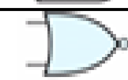
Nondegenerate forms

هذا الـ form لاستنتاج كم عدد الدوائر الممكن اتحادها في المستوى الأول والثاني

- We consider four types of gates:
- AND, OR, NAND, and NOR.
- These gates will have 16 combinations of two-level forms (16 functions).
- Eight of these combinations are said to be degenerate forms, because they degenerate to a single operation. تنتج عملية مفرده في المستوى الأول والثاني
  - ومثال ذلك AND-AND حيث يكون AND gates في المستوى الأول و AND gate في المستوى الثاني وفي هذه الحالة تكون المخرجات هي اقتران AND لكل المدخلات في الدوائر الأساسية AND-AND أو OR-OR

- Example: F(A, B, C, D) = AB.CD
- [Degenerate forms = single operation], for example, a dot (.)
- The other eight nondegenerate forms produce an SOPs or POSs as follows:
- يتوفر بها أكثر من عملية في الاقتران للمستوى الأول و الثاني
- Figure 5.5

L1 L2	L1 L2
AND-OR	OR -AND
NAND-NAND	NOR -NOR
NOR-OR	NAND -AND
OR-NAND	AND -NOR

AND-invert	NAND	
Invert-OR	NAND	
Invert-AND	NOR	
OR-invert	NOR	

- الدائرة الأولى (باللون الأحمر) تشكل المستوى الأول في التخطيط diagram والثانية (الزرقاء) تشكل المستوى الثاني في التخطيط و هكذا العمود الثاني
- The AND-OR and OR-AND forms are the basic two level forms

- الأشكال ذات المستويان الأساسيان هما :

- AND-OR
- OR-AND
- Example: AND-OR → F=A'B'E' + BD'E' + ACE (SOP form)
- AND في المستوى الأول والـ OR في المستوى الثاني والعمود الثاني رقم 2 في المربع السابق يُنفذ تماما مثل العمود الأول ولكن بالثنائية By duality of each other
- Example: AND-OR → F= A.(B+C)=(A.B)+(A.C)

(law of distributivity of the AND operator)

By duality

- Example: OR-AND → F= A+(B.C)=(A+B).(A+C)

(law of distributivity of the AND operator)

AND-OR-INVERT implementation ملخص

- Do the following using AND- OR- INVERT implementation by using the two equivalent two nondegenerate forms:
  - ✓ AND-NOR
  - ✓ NAND-AND

Of the following function: **F=AB + CD + E**

1. يجب ان يكون في sop
2. المعادلة حققت AND-OR نريد ان تحقق AND-OR-INVERT فنأخذ لها F’
3. **F=(AB + CD + E)’** حققت AND-OR-INVERT
4. وبهذا تكون حققت الـ nondegenerate الاولى وهي المكافئة AND-NOR
5. ولتحيق NAND-AND نحصل على الاقتران التالي من F’ بواسطة ديموروقان law
6. F= (AB)’ (CD)’ E’
7. Draw the diagram?

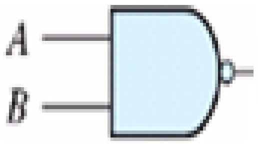
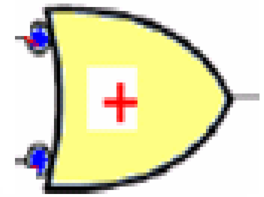
OR-AND-INVERT implementation ملخص

- Do the following OR - AND - INVERT Implementation by using the two equivalent two nondegenerate forms:
  - ✓ OR-NOR
  - ✓ NOR - OR

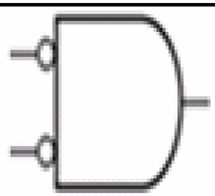
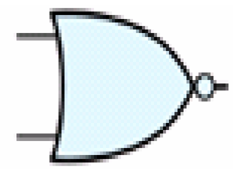
Of the following function: **F=AB + CD + E**

1. Must be in POS F= (A’ + B’) (C’+D) E’ → OR-NOR
2. نريد ان تحقق OR - AND - INVERT فنأخذ لها F’
3. F= [(A’ + B’) (C’+D) E’]’ → OR-NOR
4. وبهذا تكون حققت الـ nondegenerate الاولى وهي المكافئة OR-NOR
5. ولتحقيق NOR-OR نحصل على الاقتران التالي من F’ بواسطة ديمورقان law
6. F= (A’+B’)’ + (C’+D’)’ + E
7. Draw the diagram?

Note: AND-invert = NAND = invert-OR

 (AB)'= AND-invert	= NAND =	 A'+B' = invert OR
DeMorgan theorem		

Note: invert-AND = NOR = OR-invert

 (A'B')= invert-AND	= NOR =	 (A+B)' = OR-invert
DeMorgan theorem		

AND-OR-INVERT implementation

- The two forms [NAND-AND and AND-NOR] are equivalent forms and can be treated together.
- NAND-AND تشبه AND-NOR
- نريد تنفيذ AND- OR- INVERT باستخدام nondegenerat form المكافئة التالية:
- إن NAND-AND تكافئ AND-NOR ويعملان نفس عمل
- Implement the following nonDegenerate forms:

❑ AND-OR-INVERT :

✓ [AND-NOR] is equivalent to

✓ [NAND-AND]

Example: implement the function $F=(AB + CD + E)'$ by using AND-OR-INVERT implementation

AND-OR-INVERT

(AB + CD + E)'

- nondegenerat form باستخدام AND- OR- INVERT نريد تنفيذ المكافئة التالية: NAND-AND تكافئ AND-NOR
- ويعملان نفس عمل

❑ AND-OR-INVERT

1. AND-NOR → $(AB + CD + E)'$ must be in SOP
2. NAND-AND → $(AB)' (CD)' E'$

1 & 2 are equivalent and can be treated together.

For more declaration:

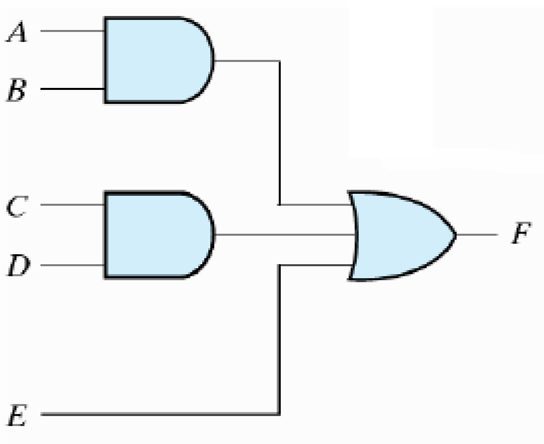
- Implement the following nonDegenerate forms:

❑ AND-OR-INVERT : [AND-NOR] [NAND-AND]

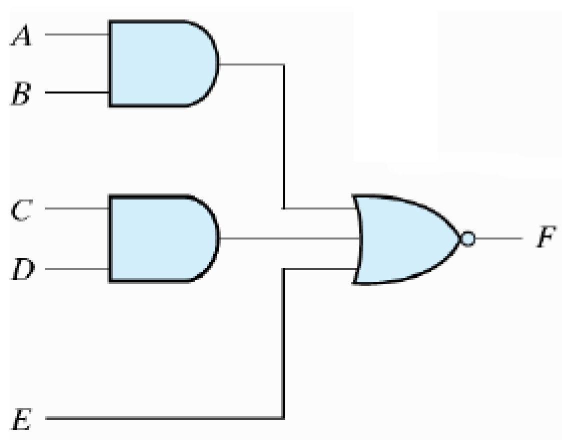
By Using the following function:

Example: implement the function

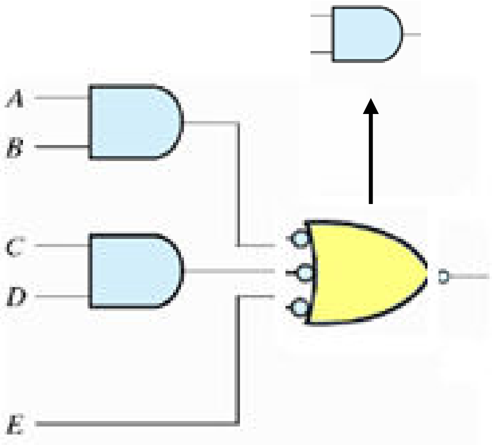
F=(AB + CD + E)   F=(AB + CD + E)’



(a) AND- OR



AND–NOR

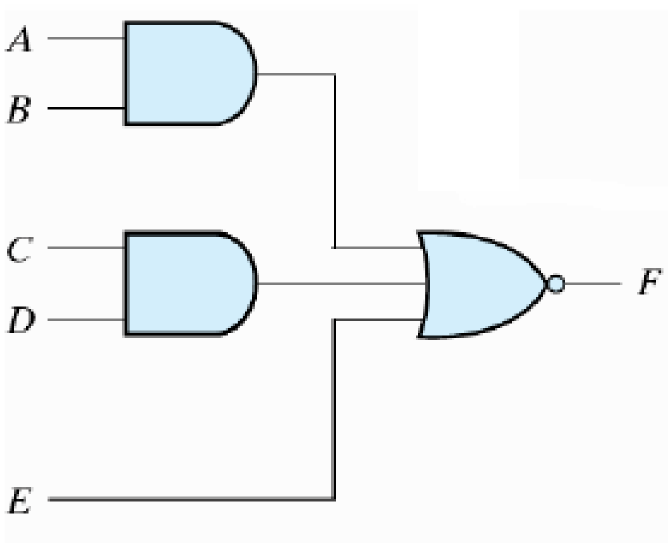


NAND-AND

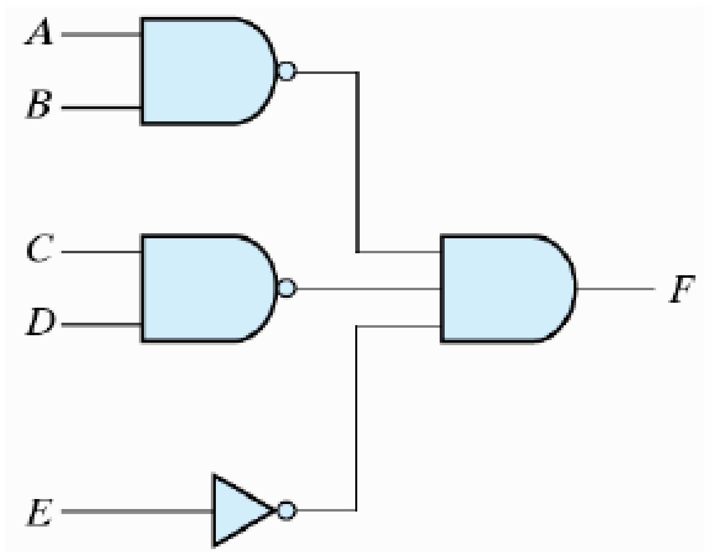
❑AND-OR-INVERT : [AND-NOR] [NAND-AND]

F=(AB + CD + E)’

F = (AB)’ . (CD)’ . E’

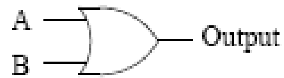


AND–NOR

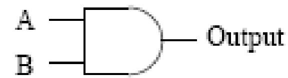


NAND–AND

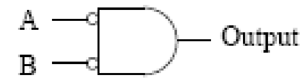
Exercise: Identify each of the following logic gates by name, and complete their respective truth tables:



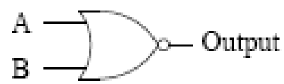
A	B	Output
0	0	
0	1	
1	0	
1	1	



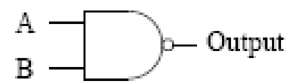
A	B	Output
0	0	
0	1	
1	0	
1	1	



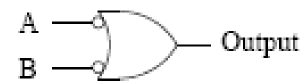
A	B	Output
0	0	
0	1	
1	0	
1	1	



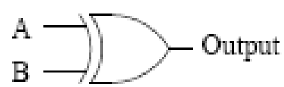
A	B	Output
0	0	
0	1	
1	0	
1	1	



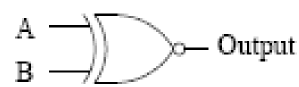
A	B	Output
0	0	
0	1	
1	0	
1	1	



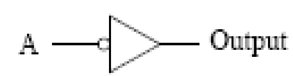
A	B	Output
0	0	
0	1	
1	0	
1	1	



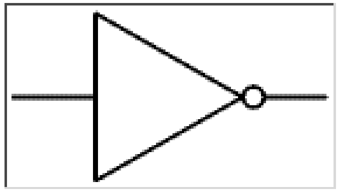
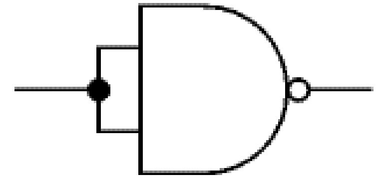
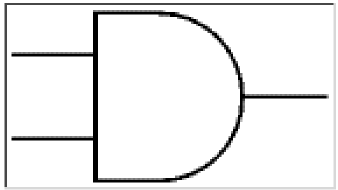
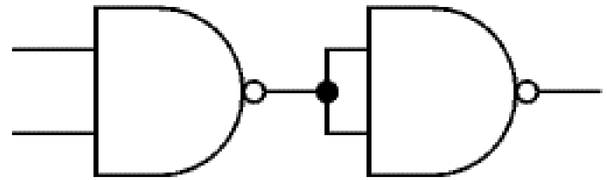
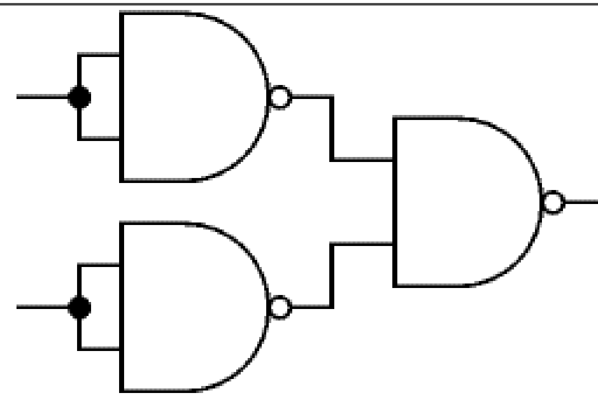
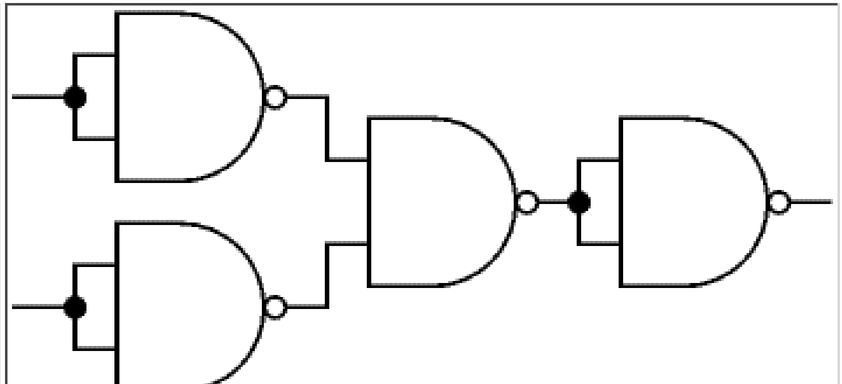
A	B	Output
0	0	
0	1	
1	0	
1	1	



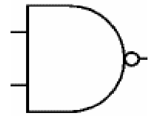
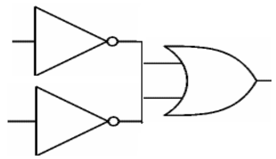
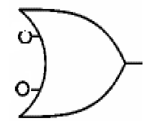
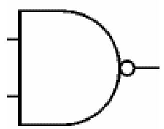
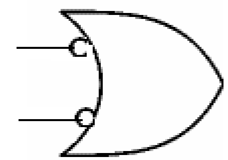
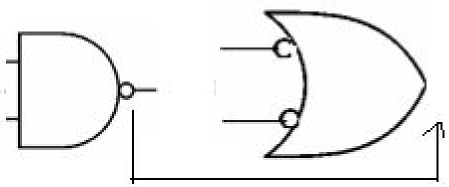
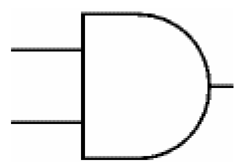
A	B	Output
0	0	
0	1	
1	0	
1	1	



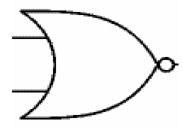
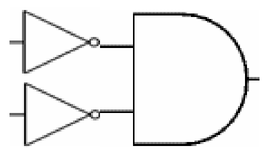
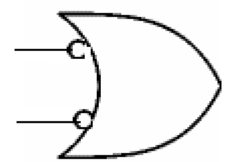
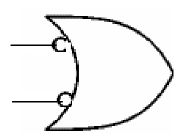
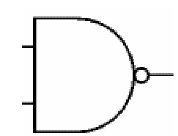
A	Output
0	
1	

Gate		Equivalent in NAND gates
NOT		
AND		
OR		
NOR		

Equivalent in NAND gates

- NAND→NOT AND→  → AND- invert → (X.Y)'
- (X.Y)'= X'+Y'→  →  → (invert-OR)
- Invert – AND equivalent to OR-invert = NAND
- The symbol  is equivalent to 
-  → 
- (Invert- OR)-invert equivalent to AND

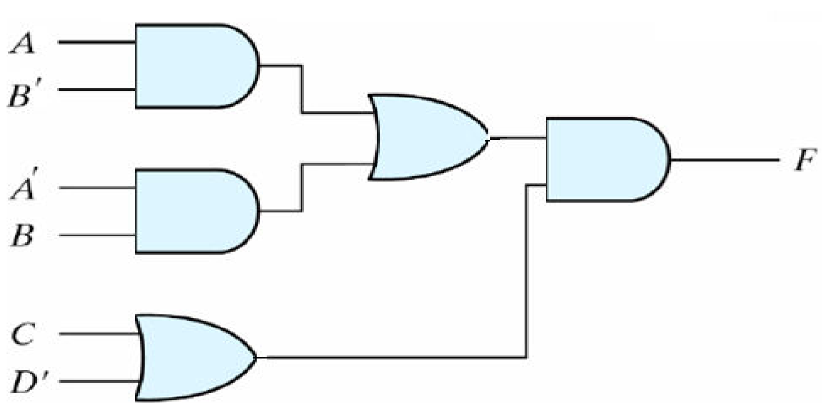
Equivalent in NOR gates

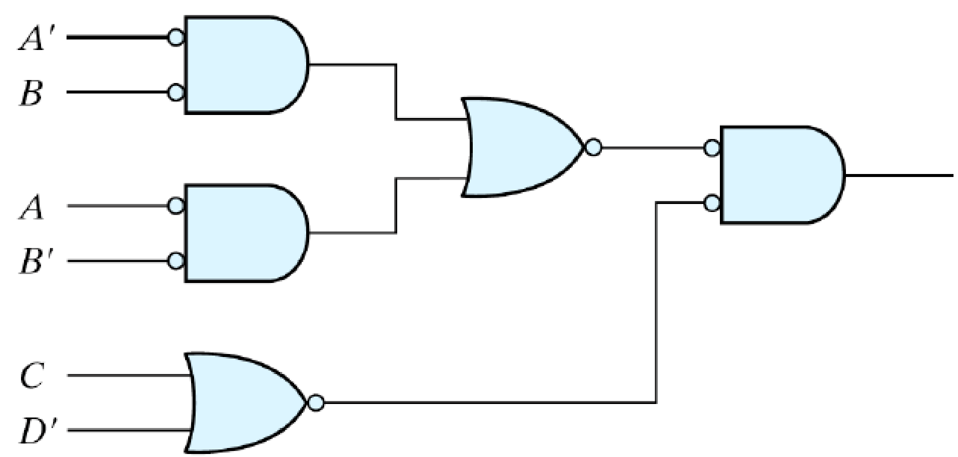
- NOR→NOT OR→  → OR- invert → (X+Y)’
- (X+Y)’= X’.Y’→  →  → (invert-OR)
- Invert –OR  equivalent to AND-invert 
- = NOR

Multilevel NOR circuits

- **Example**: Implementing F= (AB’ + A’B)(C + D’) with NOR gates

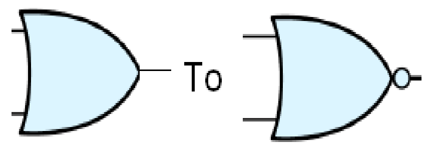
- تحويل تخطيط AND-OR الى تخطيط NOR

- 

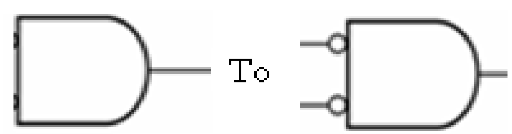


- عملية تحويل تخطيط multilevel AND-OR إلى تخطيط NOR مشابهة تماما كما في عملية التحويل السابقة في NAND gates.

- For the NOR case, we must convert each OR gate to an OR-invert



- Convert each AND gate to an invert-AND



- Any bubble that is not compensated by another bubble along the same line needs an inverter, or the complementation of the input literal.

Lecture 20

Combinational Logic

Chapter 4

Dr. Amer Abu-Jassar
Faculty of Science and Information Technology

Outlines

- Combinational Circuits
- Analysis Procedure
- Design Procedure
- Binary Adder-Subtractor
- Decimal Adder
- Binary Multiplier
- Decoders
- Encoders
- Multiplexers

❑ Introduction

- The purpose of this chapter is to use the knowledge acquired in previous chapters to formulate systematic analysis and design procedures for combinational circuits.

- الغرض من هذا الفصل هو استخدام المعارف المكتسبة في الفصول السابقة من اجل:
- صياغة التحليل المنهجي واجراءات التصميم للدوائر الجمعية (اي تحليل الدوال وإيجاد مخرجات الدوائر)

- وكذلك نتطرق إلى تصميم Boolean functions وحل مسائل التصميم.

- نتعرف على أنواع وإشكال الدوائر المعيارية الجمعية التي تودي وظيفة محددة في الدائرة المتكاملة ومنها:-

- Adder, Subtractors, Decoders, Encoders, and Multiplexers.

- Logic circuits for digital systems may be either combinational or sequential.

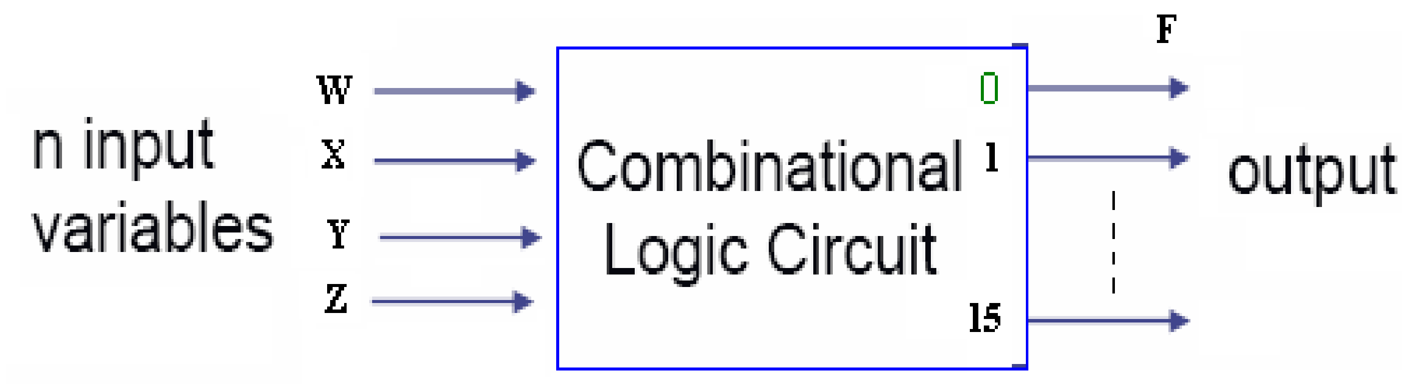
- A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs.

- In digital system there are two types of logic circuits :

  - Combinational logic circuits دوائر المنطق التوافقية
  - Sequential logic circuits دوائر المنطق التسلسلية

- Combinational Logic:
  - Output only depends on current input
  - تعتمد مخرجات دائرة المنطق الجمعية على المدخلات الحالية للدائرة (متغيرات الدالة هي مدخلات الدائرة فإذا كانت المتغيرات 4 تكون مدخلات الدائرة المحتملة 2^4
  - Perform an operation specified logically by a set of Boolean function
  - تودي وظيفة محددة منطقيا من قبل مجموعة من الدوال البولية

2^4

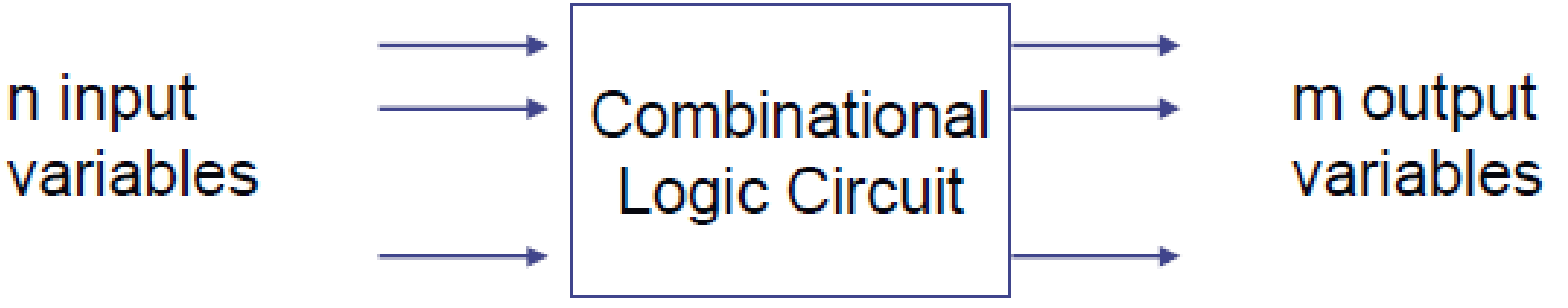


Block diagram

المخطط الرمزي للدائرة المنطقية التتابعية

A combinational circuits

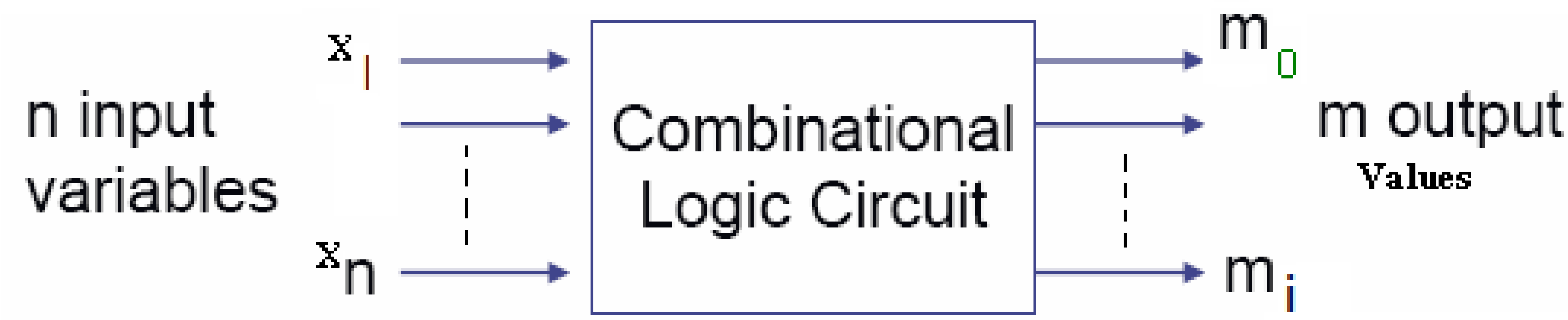
- 2^n possible combinations of input values



- Sequential Logic:

  - Output depends on current (storage element = function of the previous inputs) and past inputs

- تعتمد مخرجات الدائرة التسلسلية على المدخلات الحالية وعلى مخرجات الدوال لتصبح مدخلات للدائرة من جديد لأنها تكون مخزنه في مسجلات لتستخدم كمدخلات من جديد.
- و تشمل البوابات المنطقية و عناصر الذاكرة, و بسبب وجود عناصر الذاكرة فإن خرج دوائر المنطق التسلسلي لا يعتمد فقط على قيم المدخلات الحالية فقط و إنما أيضاً على القيم السابقة للمدخلات.



- The output functions specified in the truth table give the exact definition of the combinational circuit.
- Also the combinational circuits can be described by **m** Boolean functions, one for each output variables.
- Each output function is expressed in terms of the **n** input variables.
- It means that : $M_i = F(x_1,x_2,x_3,\ldots,x_n)$

- ان اي من مخرجات الدائره المنطقية التتابعية Mi هو دالة من مجموعة من مدخلات الدائره (او كلها)

- Output only depends on current input such as 000 001 010 011…
- المخرجات تعتمد على مدخلات الاقتران الحالي
- Perform an operation specified logically by a set of Boolean function
- تودي الدائرة الجمعية وظيفة محددة منطقيا بواسطة مجموعة من الدوال البولية
- Has ***n*** variables come from an external source , the ***m*** values of functions are produced by the internal combinational logic circuit and go to external destination
- Each input and output variable exist physically as an analog signal, and the values interpreted to represent logic 0,1
- كل متغيرات المدخلات والمخرجات موجودة فعليا على شكل إشارات تناظرية ، والقيم تفسر على أنها تمثل منطق 0 ، 1

Combinational Logic Consist of

- ✓ input variable
- ✓ logic gates
- ✓ output variable – function for each input variable
- The Combinational Logic circuits employed in the design of digital system and available as an integrated circuits (IC's) to perform specific digital function
- الدوائر الجمعية المنطقية مستخدمة في تصميم النظام الرقمي ومتوفرة كدوائر متكاملة لأداء وظيفة رقمية محددة
- The important standard forms of the Combinational circuits are: -Adder -Subractors -

  Decoders –Encoders & multiplexers.

- To obtain the truth table directly from the logic diagram with-out going through the derivations of the Boolean functions. Proceed as follows:
  - للحصول على جدول الحقيقة من التخطيط مباشرة دون أن تشتق الاقتران البولي من التخطيط اتبع الخطوات التالية:

- Determine the number of input variables.. حدد عدد المتغيرات في الدائرة
- مثال: اذا كان عدد المدخلات 4 فيكون possible input combination (تركيبة المدخلات الممكنة) هو $2^n = 16$ فتكون قائمة المدخلات الثنائية في جدول الحقيقة تساوي 2^n ‎-1 أي من 0 إلى 15

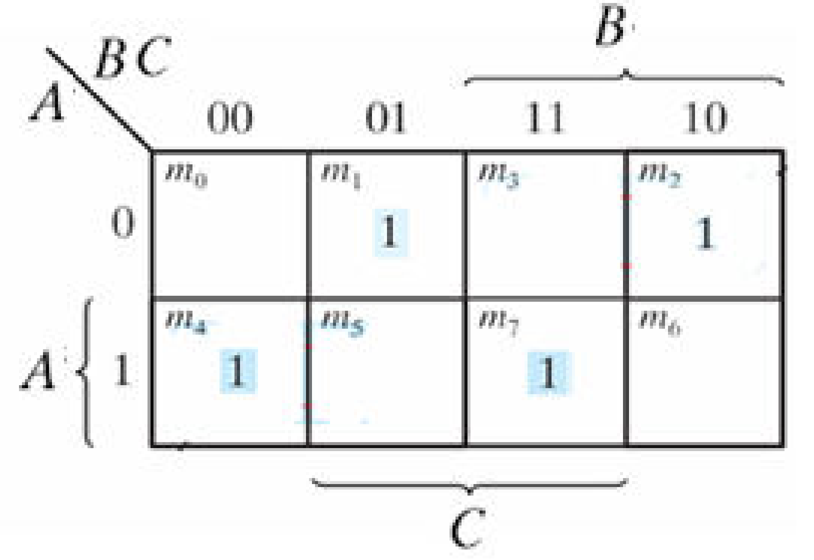
- Label the outputs of selected gates with arbitrary symbols.

- Obtain the truth table for the outputs of those gates which are a function of the input variable only.

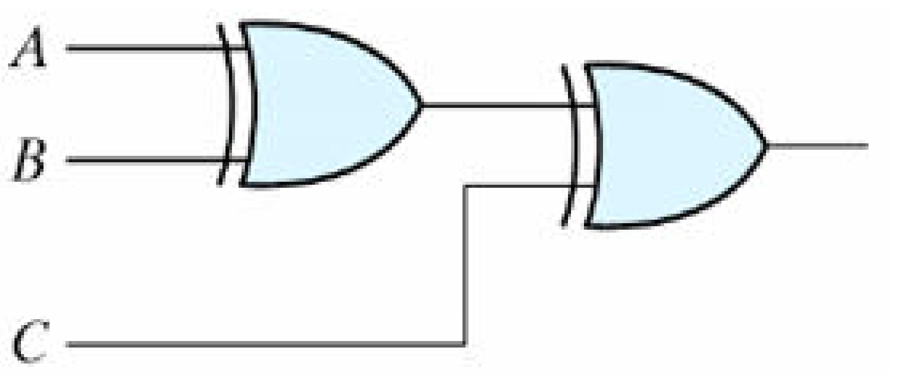
- Determine the standard function for all previous output by using k-map.

Truth table for the above logic diagram

A	B	C	F_2	F'_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1



- The Boolean function of the map is
- F = A'BC'+A'B'C+AB'C'+ABC
- F=A'(BC'+ B'C) +A(B'C'+ BC)
- F=A ⊕ B ⊕ C
- F=A'BC+A'B'C+AB'C'+ABC
- F = 010 + 001 + 100 + 111
- Represent an Odd function



(a) 3-input odd function

Lecture 21

إجراء التصميم

- تصميم الدائره التتابعية يعني بناء الدائره المنطقية التي تقوم بوظيفة معينة.
- يتم تحديد الوظيفة للدائره بواسطة جدول الحقيقة او بالمعادلة الجبرية

❑ **لتصميم الدائره التتابعية:**

- حدد وظيفة الدائره
- حدد عدد مدخلات ومخرجات الدائره
- تسمية المدخلات والمخرجات
- اوجد العلاقة بين المدخلات والمخرجات من خلال جدول الحقيقة.
- احصل على الدالة الجبرية لكل من مخرجات الدائره
- اختصر الدالة الجبرية بالطرق المتاحة
- ارسم الدائرة

- **مثال** : صمم الدائره المنطقية لثلاثة مدخلات ومخرجات الدائرة تكون في الحالة الصحيحة **1** على ان تكون اغلب المدخلات تمثل 1؟
- **الحل:** نسمي المدخلات A B C والمخرجات بـ F
- نحصل على جدول الحقيقة:

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

- نختصر الدالة باستخدام خارطة كارنو

F	B'C'	B'C	BC	BC'
A'	0	0	1	0
A	0	1	1	1

$$F=AB+AC+BC$$

- ارسم الدائرة المنطقية المطلوبة!

Binary Adder-Subtractor

- Computers perform a variety of information-processing task. Among the functions encountered are the various operations.

- تودي أجهزة الكمبيوتر مجموعة متنوعة لمعالجة المهمات المتعلقة بالمعلومات المهمة. الدوال التي تواجهنا مكونة من عمليات مختلفة

- The most basic arithmetic operation is the addition of two binary digits.
- The simple addition operation consist of 4 possible elementary operations:

- ابسط عمليات الجمع الاساسية الممكنة

- 0+0=0, 0+1=1, 1+0=1, 1+1=10.

- Binary adder-subtractor is *combinational circuit* that performs the arithmetic operations of addition and subtraction with binary numbers.

Half Adder

- A combinational circuit that performs the addition of two bits is called a *half adder*.

- For example: 0+0=0, 0+1=1, 1+0=1, 1+1=10

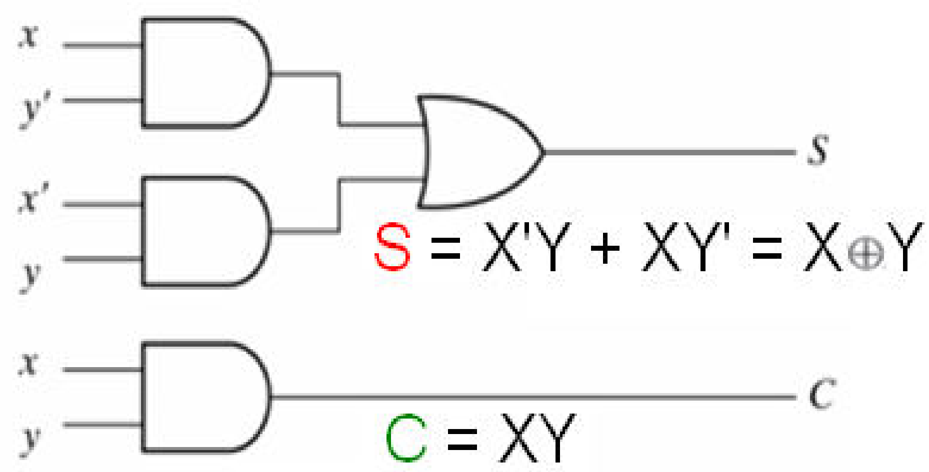
- Half Adder Design: The circuit in half adder needs:
  - 2 inputs such as symbols x and y
  - 2 outputs to produce the carry (c) and sum (s)
  - Truth table
  - Obtain the functions for the outputs s and c from the truth table directly.
  - The simplified sum-of-products expressions are :

    $S = X'Y + Y'X = X \oplus Y$ & $C = XY$
  - Draw the logic diagram of the half adder by implemented the functions.

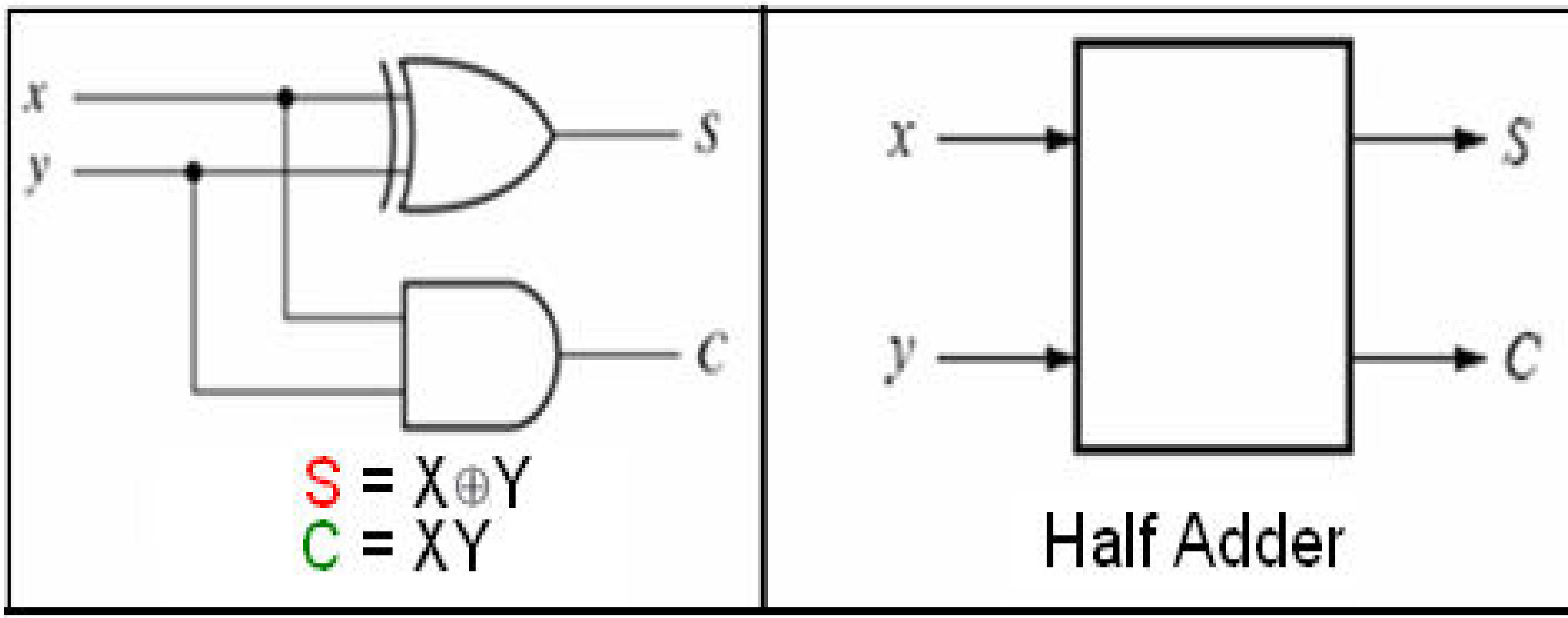
The half-adder implementation

- Derive the truth table
- Half Adder functions from the table اشتق الدالة من الجدول فورا أو استخدم الخارطة
- S = X'Y + XY' = X⊕Y → SOP form
- S = 01 + 10
- C = XY → 1.1
- Half-adder Circuit implementation (logic diagram of the half adder)
  - Implementation of half adder in sum of products form

X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



- Implementation a half adder with an exclusive-OR and an AND gate



Full Adder

- A combinational circuit that performs the arithmetic addition of *three* bits.

- Two half adders can be employed to implement a full adder.

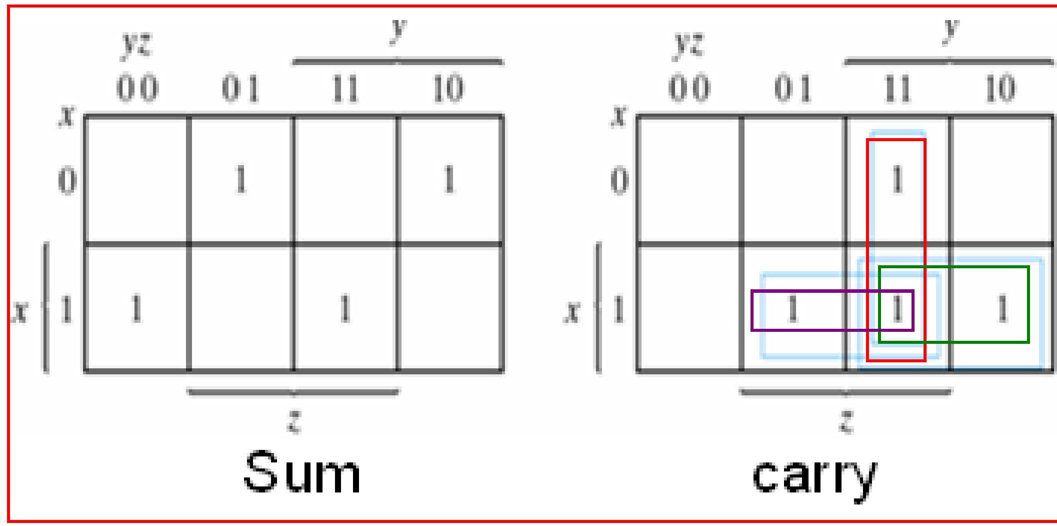
- Full Adder Design: The circuit in full-adder needs:
  - – 3 inputs such as symbols X, Y, and Z
  - – 2 outputs to produce the carry (C) and sum (S)
  - – Truth table
  - – Simplified functions of the output

- The Full-adder implementation: CXCXCXCXCX
- *Truth table* designate all possible combinations of the three variables as follows:

Truth table

X	Y	Z	C	S
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

The full-adder *map*



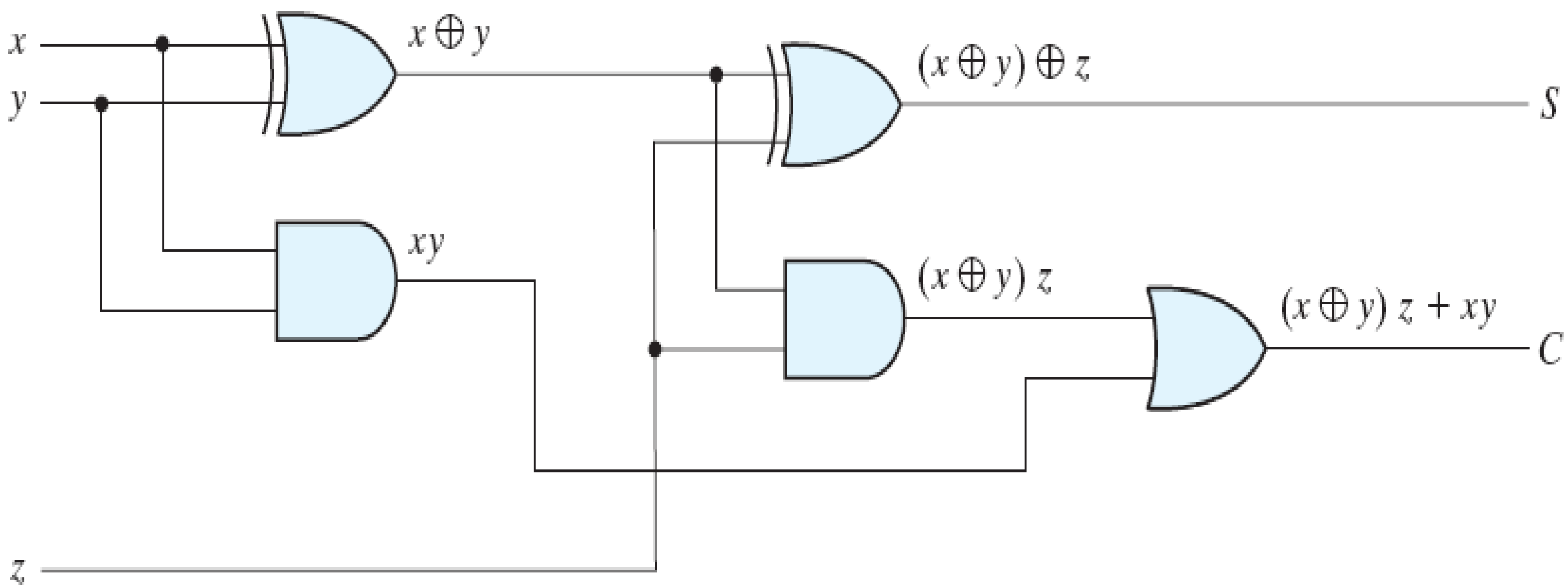
Full Adder *functions*

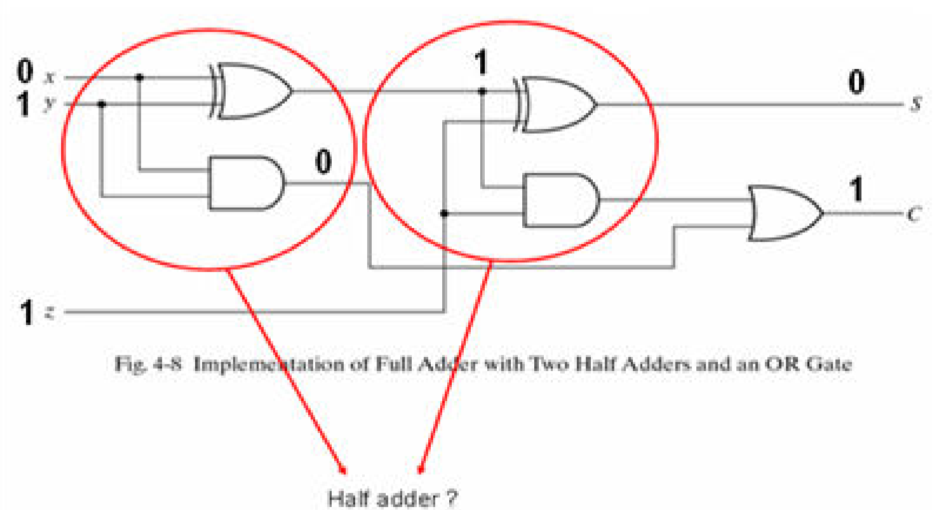
$S = \sum(1,2,4,7)$

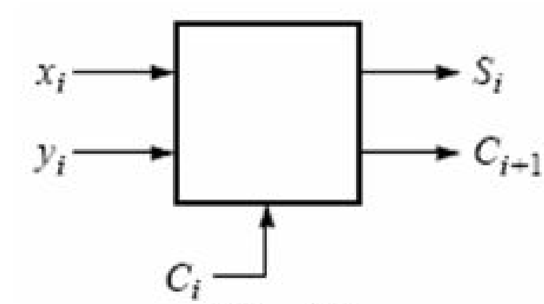
$C = \sum(3,5,6,7)$

- **Sum = X′ Y′ Z + X′ Y Z′ + X Y′ Z′ + X Y Z**
- **Carry = XY + XZ + YZ**

Implement a full adder with two half adders and an OR gate







Full adder

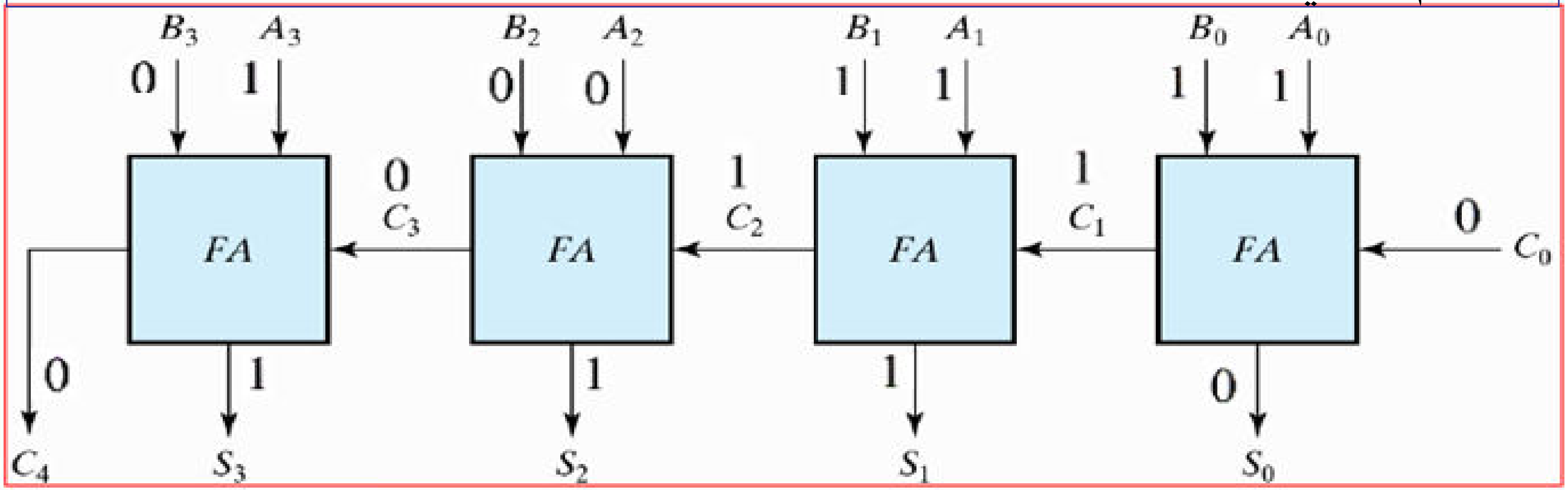
Design a 4-bit full adder circuit?

- المطلوب تصميما لـ 4-bit full adder لإجراء عملية الجمع للأرقام الثنائية في المثال التالي:

Example: 0011+ 1011

- Input carry = C_i and Output carry = C_{i+1}
- The input carry to the adder is $C_0 = 0$ (least significant position must be 0)

- بمعنى أن أول carry في بداية عملية الجمع هو C_0 ويساوي دائما صفر
- المطلوب عملية جمع للأعداد الثنائية التالية المذكورة في الجدول
- الرقم الأول A3 A2 A1 A0
- الرقم الثاني B3 B2 B1 B0



Lecture 22

Subtractor

- Subtracter circuits take two binary numbers as input and subtract one binary number input from the other binary number input. Similar to adders, it gives out two outputs, difference and borrow (carry-in the case of Adder).

- There are two types of subtracters.
    - Half Subtracter.
    - Full Subtracter.

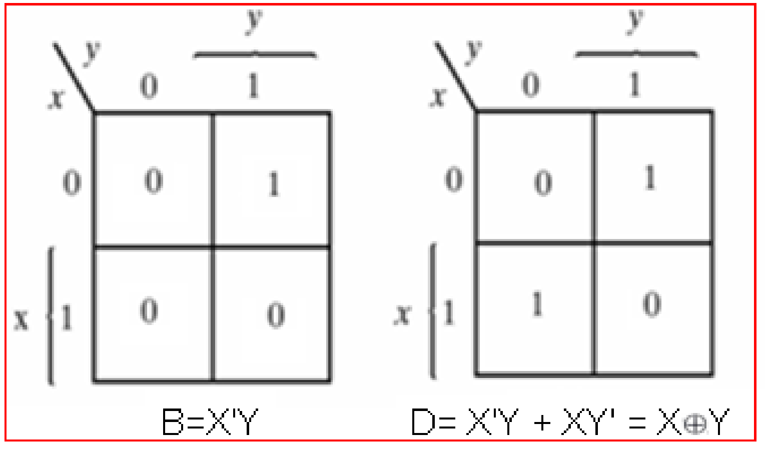
Half Subtracter

- The half-subtracter is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, X (minuend) and Y (subtrahend) and two outputs D (difference) and B (borrow).

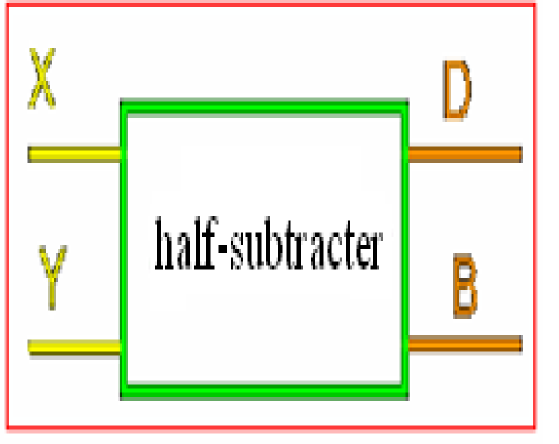
- Truth table

X	Y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

- K-map



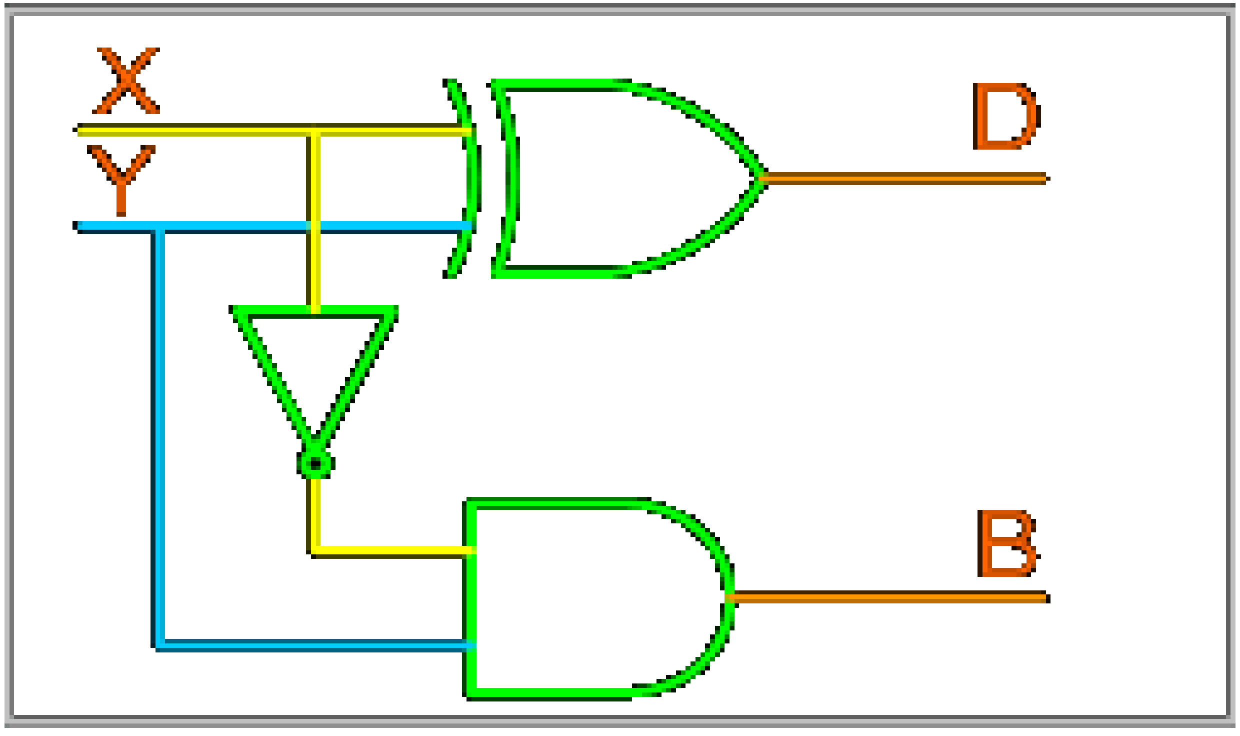
- Logic Symbol



- Half subtractor *functions*

  B=X'Y   D= X'Y + XY' → $D = X \oplus Y$

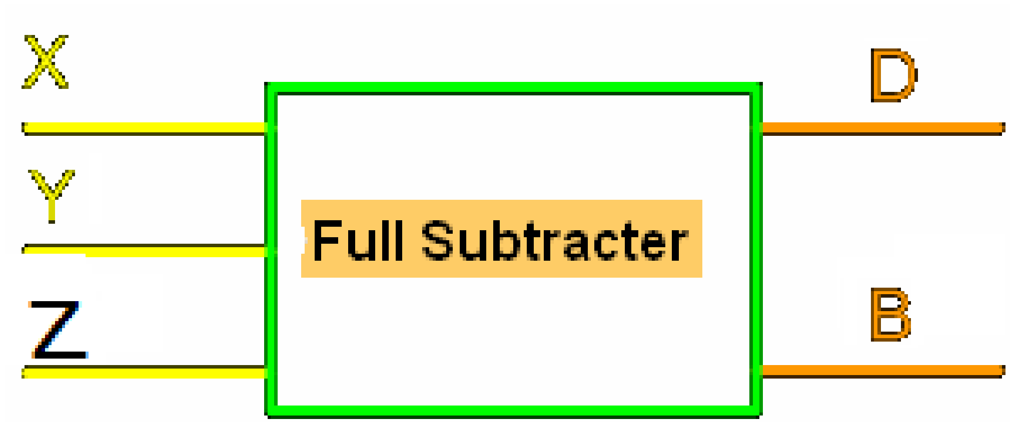
- Half subtractor *Circuit* implementation



D=∑(1,2), B= ∑(1,)

Full subtractor

- A full subtracter is a combinational circuit that performs subtraction involving three bits, namely minuend, subtrahend, and borrow-in.
- The logic symbol

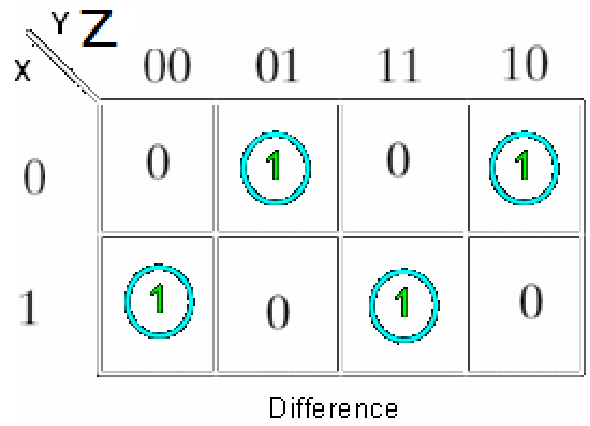


Truth table

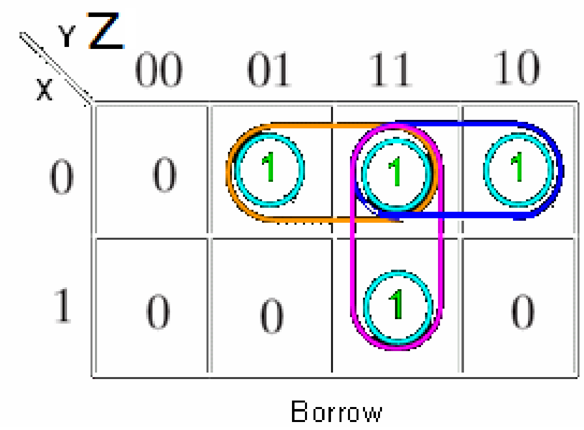
X	Y	Z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$D=\sum(1,2,4,7)$, $B=\sum(1,2,3,7)$

K-map

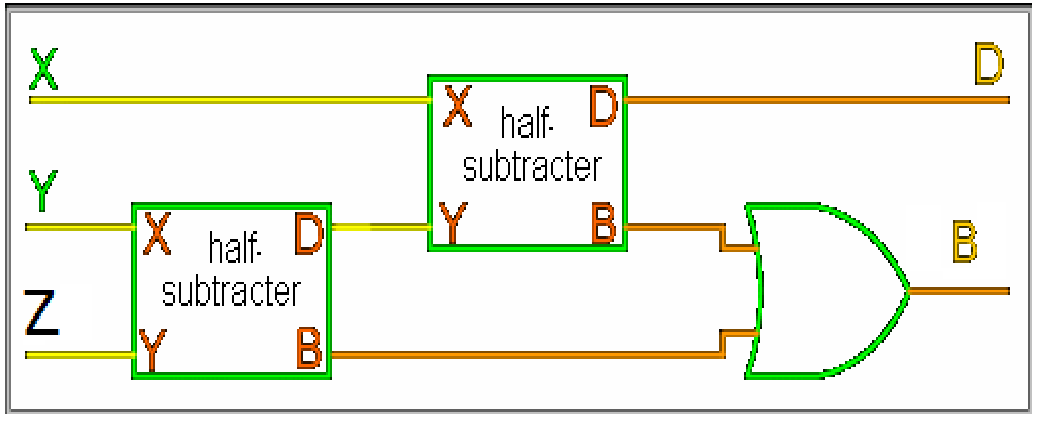


Difference



Borrow

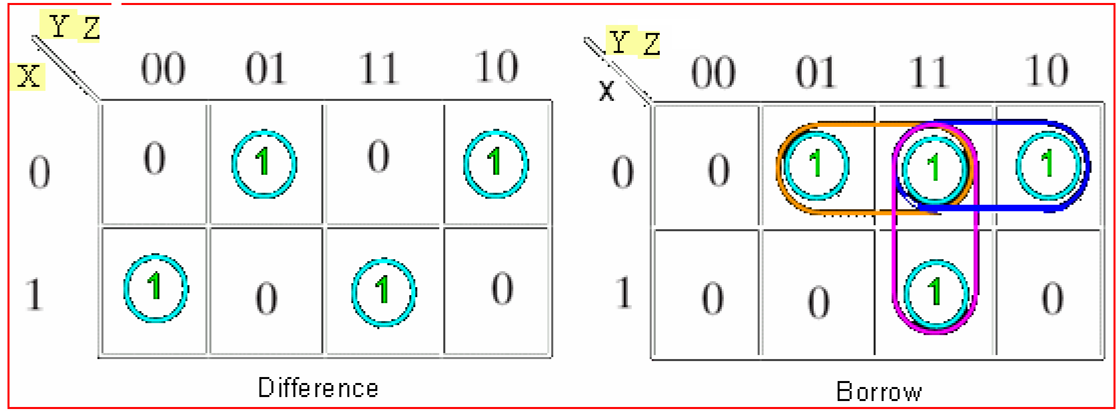
Logic diagram



Full subtractor

Truth table

X	Y	Z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

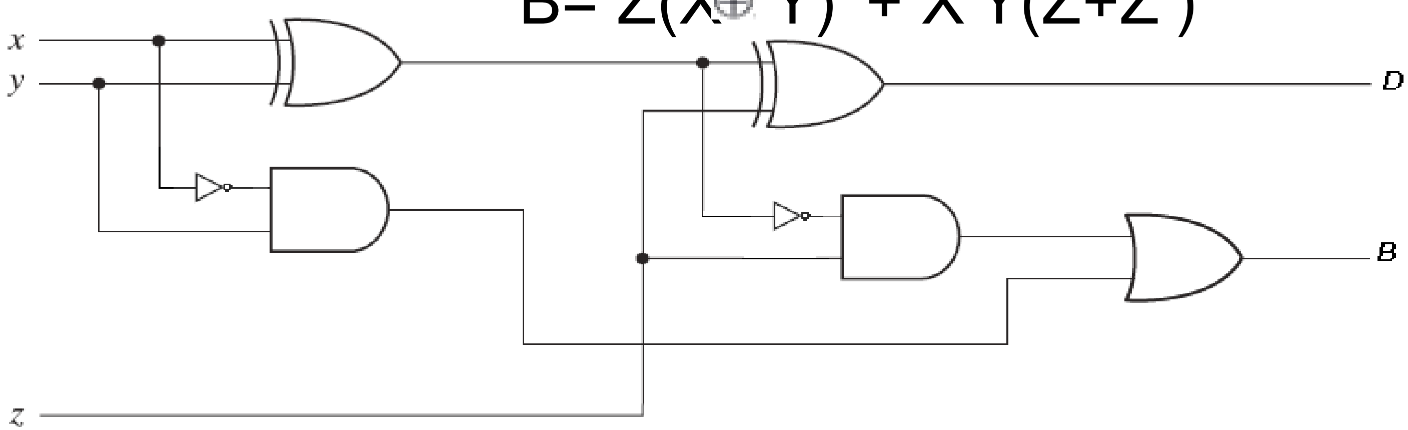


Full Subtractor function:

$D = X \oplus Y \oplus Z$ $\quad$ B = $X'.Y + X'.Z + Y.Z$

$B = X'Y'Z + X'YZ + X'YZ' + XYZ$

$B = Z(X \oplus Y)' + X'Y(Z+Z')$

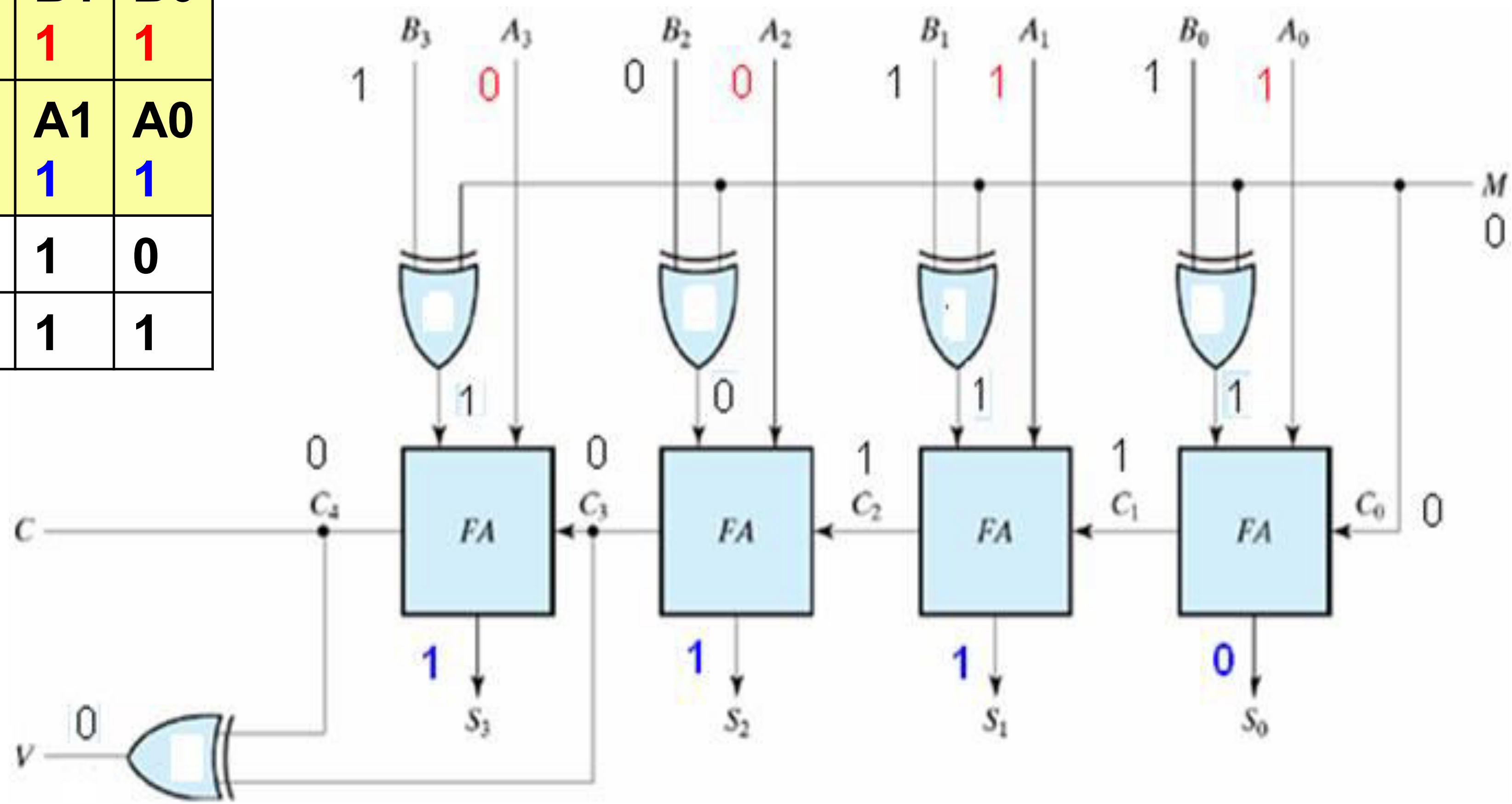


Example: use 4-bit adder to add 11+3=14 (add operation)

With control M?

C_i	0	1	1	0
B_i	B3 1	B2 0	B1 1	B0 1
A_i	A3 0	A2 0	A1 1	A0 1
Sum	1	1	1	0
C_i+1	0	0	1	1

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0



Lecture 23

Decoders

- محلل التعليمة ويسمى بدائرة فك (تحليل) الرموز

- يسمى decoder بدائرة فك (تحليل) الرموز و هى عبارة عن دائرة جمعية منطقية لها n مدخل و m مخرج

- التعليمة في الحاسوب تمر في مراحل ومنها مرحلة البحث عن التعليمة ومرحلة تنفيذ التعليمة.

- للتعليمة دورة تنفيذ في الحاسوب فيتم استقبال التعليمة وتخزينها في الذاكرة كأرقام ثنائية بهدف تحليلها وتحديد نوع العملية المطلوب إجراؤها لهذه التعليمة من قبل وحدة التحكم.

- **مثال**: لدينا تعليمة مخزنة كأرقام ثنائية في الذاكرة ونوع العملية لها هو إيجاد المكمل: Complement

- إيجاد المكمل لمحتوى التعليمة 00100011 حيث يتم تحريك المحتوى للمسجل **B** ليصبح 11011100 وهكذا لحين انتهاء العملية.

- وكل مخرج من مخارج هذه الدائرة هو حد أصغر (minterm) . وبهذا فانه عند إعطاء قيم المدخلات فان احد المخرجات **m** سيقع في الحالة الصحيحة **1** وبقية المخرجات ستقع في الحالة الخاطئة (0)

- **Decoders** are simply a collection of logic gates represented in digital system by binary codes.

- A **decoder** is a combinational circuit that converts binary information from ***n*** input lines to a maximum of $\mathbf{2^n}$ output lines.

- A binary code of ***n*** bits = 2^n distinct information. This means that, a decoder provides 2^n minterms of *n* input variable.
- Only one output can be active (high) at any time

- أنواع وإشكال دائرة فك الرموز decoder تعتمد على المدخلات

- **Example**: consider a **two to four line decoder**
- A two input variables are decoded into four outputs.

- It produces **minterms** of its input variables (one line at the output for each possible input).

- كل دالة تمثل minterm منفصل ولكن يتم ربط كل منها بخط من المتغيرات التي تمثل هذه الدوال الممكنة

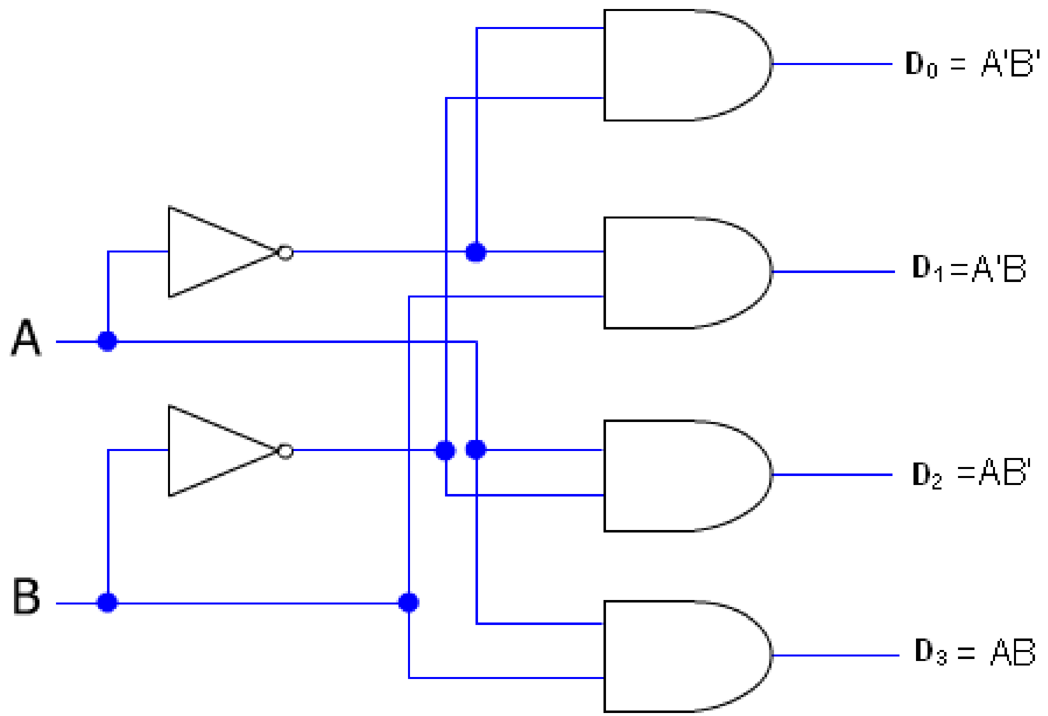
- The truth table for the considered example can be shown as follows.

- نضع الناتج 1 في المخرجات لكل الحدود حسب ما يمثله رقم المدخلات binary code مثلا D2 يمثله 10 فيكون تقاطعهما هي مخرجات الحد الذي يمثله D2 وكذلك D3=AB وبالتالي جميعا تمثل الحدود الصغرى

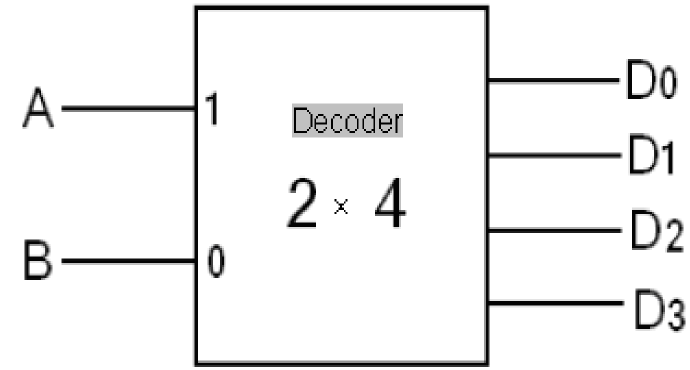
	A	B	D_0	D_1	D_2	D_3
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1

The circuit that represents a **two to four** line decoder is as follows:

	A	B	D_0	D_1	D_2	D_3
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1



Graphic Symbol



Two to four line decoder

All minterms of an input are generated from decoders

Three-to-Eight Line Decoder

- Example: consider the **3-to-8**-line decoder circuit.
- A 3 inputs are decoded into 8 outputs.
- Each represents one minterm of the three input variables.
- The three inverters provide the complement of the inputs.
- Each one of the AND gates generates one of the minterms.
- A particular application of the decoder is binary to octal conversion.
- The truth table of a 3×8 line decoder can be shown as follows:

	Inputs			Outputs							
	x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

D0 = X'Y'Z' الحد الاصغر

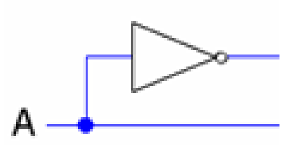
D1= X'Y'Z ....and so on?

- The output whose value is equal to 1 represents the minterm equivalent to the binary number currently available in the input line.

- كل دالة من دوال المخرجات تمثل حد من حدود المدخلات minterm

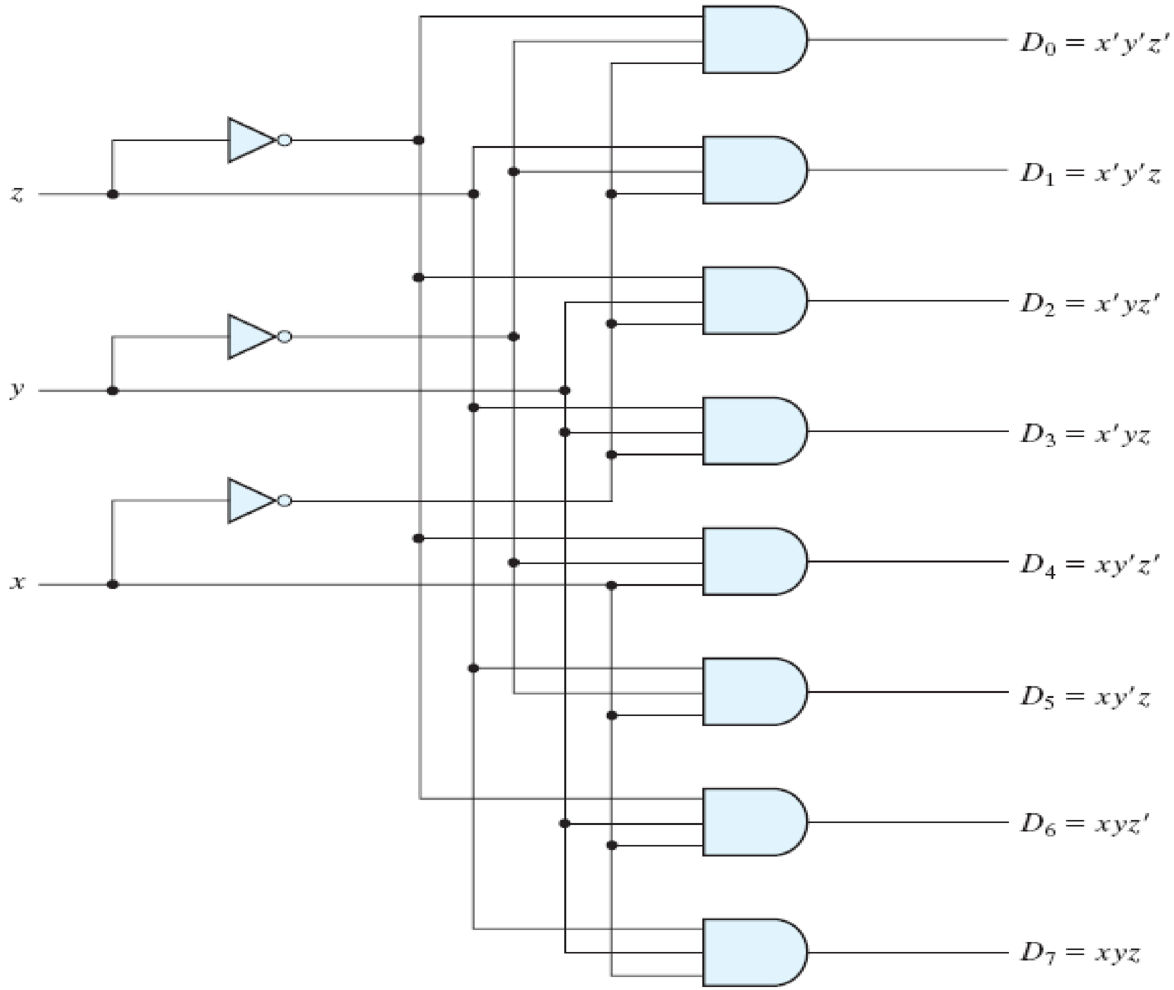
- The circuit that represents a 3 × 8 line decoder is as follows:

- لتمثيل جميع قيم المدخلات الممكنة الناتجة عن المتغيرات الثلاثة تضع خط للمتغير وخط المكمل لنفس المتغير و هكذا لبقية المتغيرات مثال :

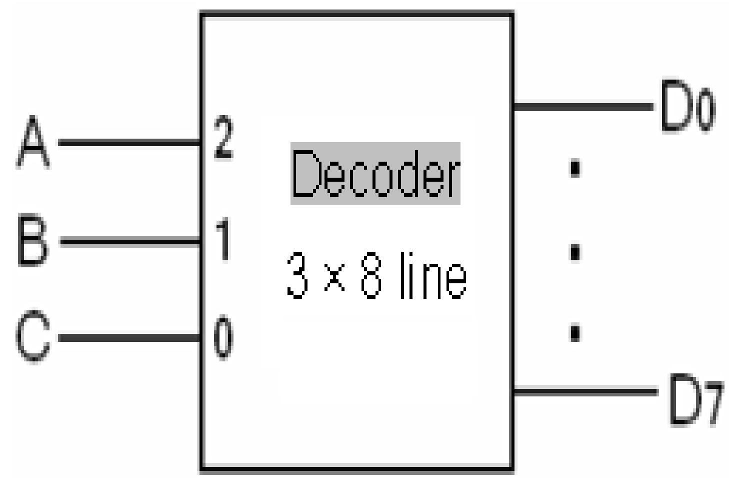


- The circuit that represents a **3 × 8** line decoder is as follows:

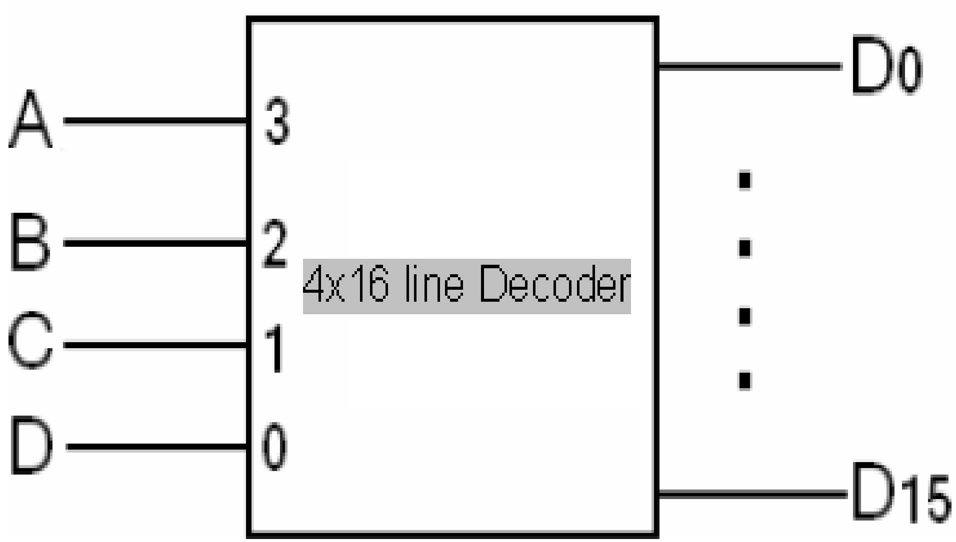
- المتغير و المكمل له في الشكل لكي يمثل جميع المدخلات الممكنة الناتجة عن المتغيرات الثلاثة



3 × 8 line decoder



3 × 8 line decoder



- Write the truth table and find out all minterms that represents the D0 to D15.
- Draw the circuit that represents a 4 × 16 line decoder in detail? D_0=A'B'C'D'……….D_{15}=ABCD

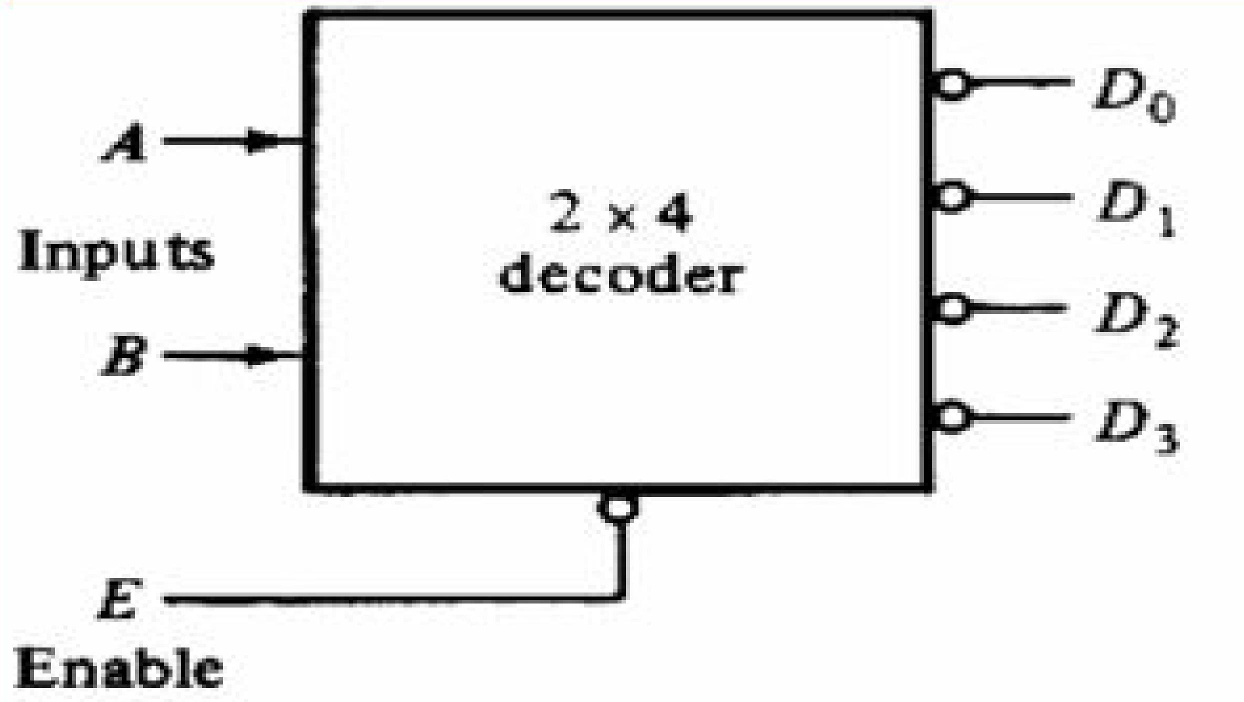
Decoder with Enable /Demultiplexers

- يمكن جمع أكثر من decoder في وظيفة مماثلة بنفس الدائرة بحيث تكون محكومة بعنصر تحكم Enable.
- يستخدم إل Enable كعنصر تحكم في إل decoder لكي يعمل أو لا يعمل وقد تكون الدائرة تحتوي أكثر من decoder فيستخدم إل enable لغرض الجمع بينهما في وظيفة مماثلة مع مزيد من المدخلات والمخرجات كما في المثال القادم.
- عنصر التحكم يمثل مفتاح الدالة كأول عنصر بالمتغيرات لتكون الاصفار مساوية للواحدات؟
- فإذا كان enable=0 فان احد إل decoders لا يعمل
- وإذا كان enable=1 فان احد إل decoders يعمل
- الجدول التالي يمثل (**2×4 decoder**) محكوم بمفتاح تحكم **E**.

Example: (2×4 decoder) controlled with key E

E	A	B	D0	D1	D2	D3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$D0=EA'B'$ $D1=EA'B$ $D2=EAB'$ $D3=EAB$



(a) Decoder with enable

- فإذا كان enable=0 فان decoder الأول لا يعمل وتكون جميع قيم المخرجات ل Decoder = 0
- وإذا كان enable=1 فان decoder الثاني يعمل وتظهر نتيجة عملة على المخرجات كما يلي:
- D_0=EA’B’ D_1=EA’B D_2=EAB’ D_3=EAB

- كون **الصفوف الأربعة** الأولى لها نفس المخرجات لذا يمكن اختصارها في الجدول إلى صف **واحد** كما يأتي:

E	A	B	D0	D1	D2	D3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

E	A	B	D0	D1	D2	D3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

- إن الحرف **x** يدل إلى اختصار المدخلات ولا يعتبر don't care لأنه لا يوجد don't care في المدخلات
- ويمكن اعتبار **x** مرة يساوي **1 ومرة يساوي صفر** حسب قيم المتغيرات أو اكتب بدل x في الجدول A, B

Exercise 4-23

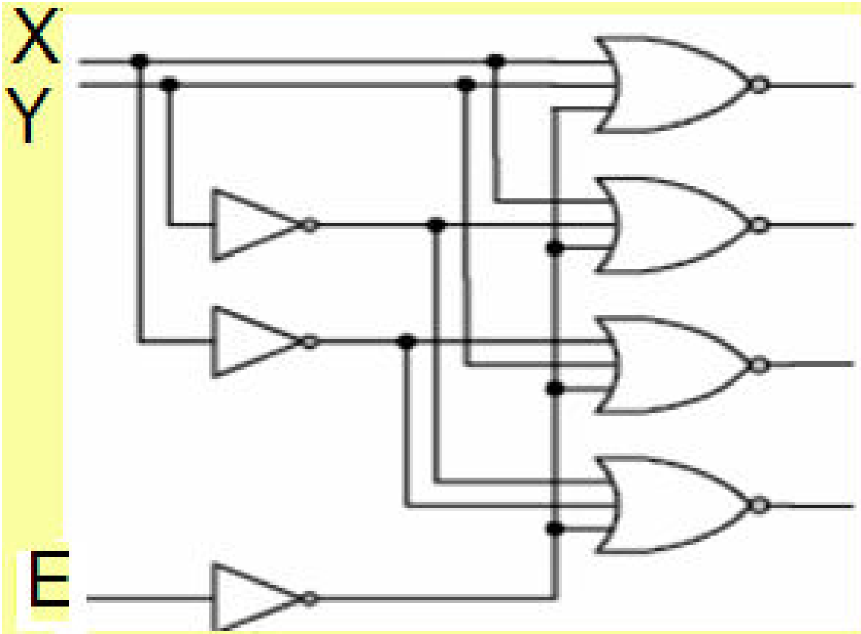
Exercise: Draw the logic diagram of a 2-to-4 line decoder using NOR gates only Include an enable input?

a) Find out the truth table for the circuit?

b) Find out the *functions using NOR gates and then*

c) *Draw* the logic diagram?

E	A	B	D0	D1	D2	D3
0	x	x	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1



$$D0 = EX'Y' = (E'+X+Y)'$$

$$D1 = EX'Y = (E'+X+Y')'$$

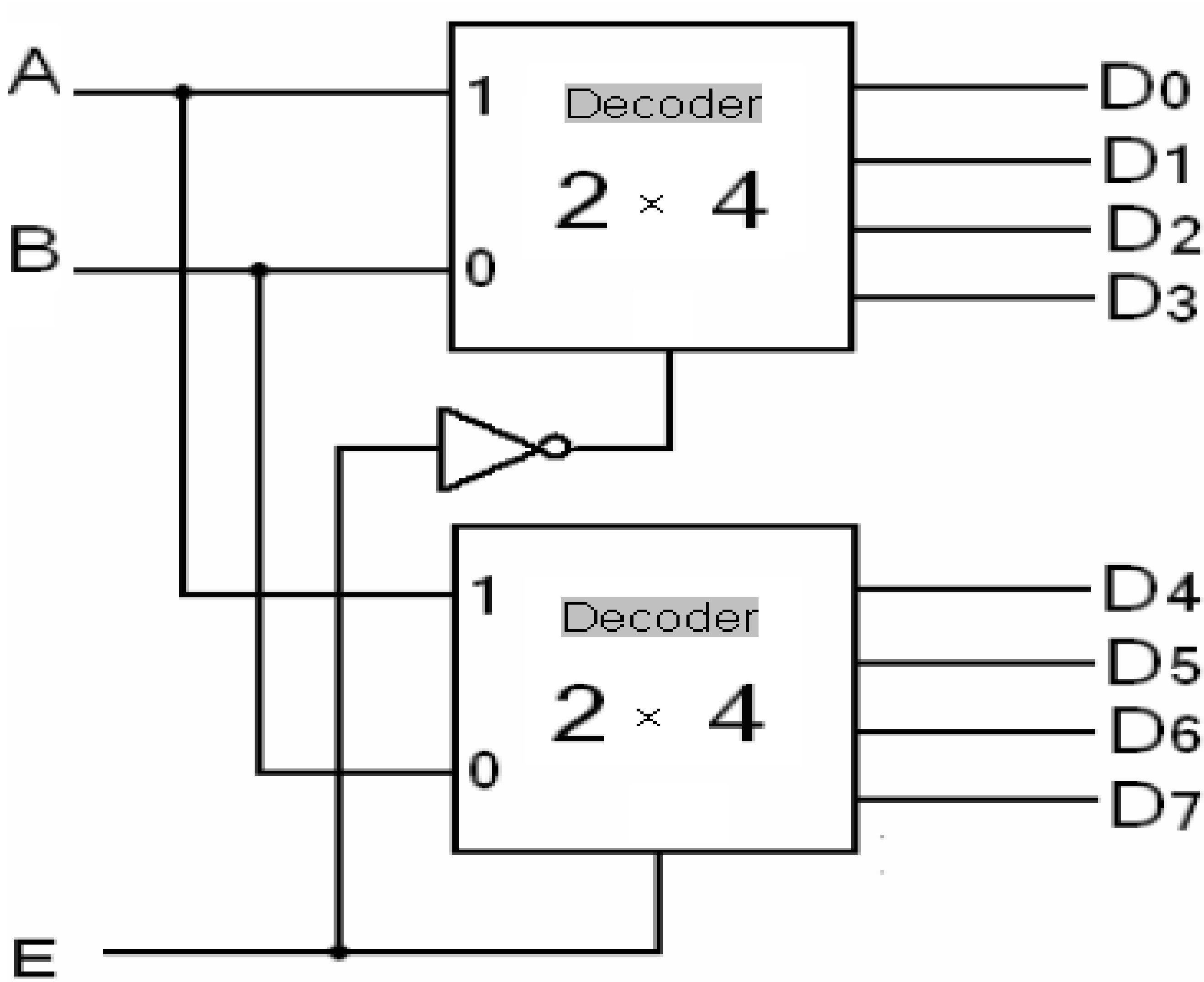
$$D2 = EXY' = (E'+X'+Y)'$$

$$D3 = EXY = (E'+X'+Y')'$$

Example: Design a decoder (3×8) by using a decoder (2×4) with enable. Truth table الجدول يصف نتائج العملية في الحالتين

E	A	B	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

The circuit that represents a two to four line decoder is as follows:



A Decoder (3×8) by using a decoder (2×4)

- إل E تمثل عنصر التحكم وكأنها أول متغير يعمل كمفتاح للدالة لتكون مرة في
- حالة الصفر ومرة في حالة الواحد.
- فإذا كان **E =0** يعمل إل Decoder الأعلى وتظهر علية قيم المخرجات
- وكذلك يعمل Decoder الأسفل في حالة **E=1** وتظهر علية قيم المخرجات

Decoder with Enable /Demultiplexer يعمل عمل

- Some decoders are constructed with **NAND** gates.
- More economical to generate the decoder minterms in their **complemented** form.
- Decoders include one or more *enable* inputs to control the circuit operation.

Lecture 24

Encoders

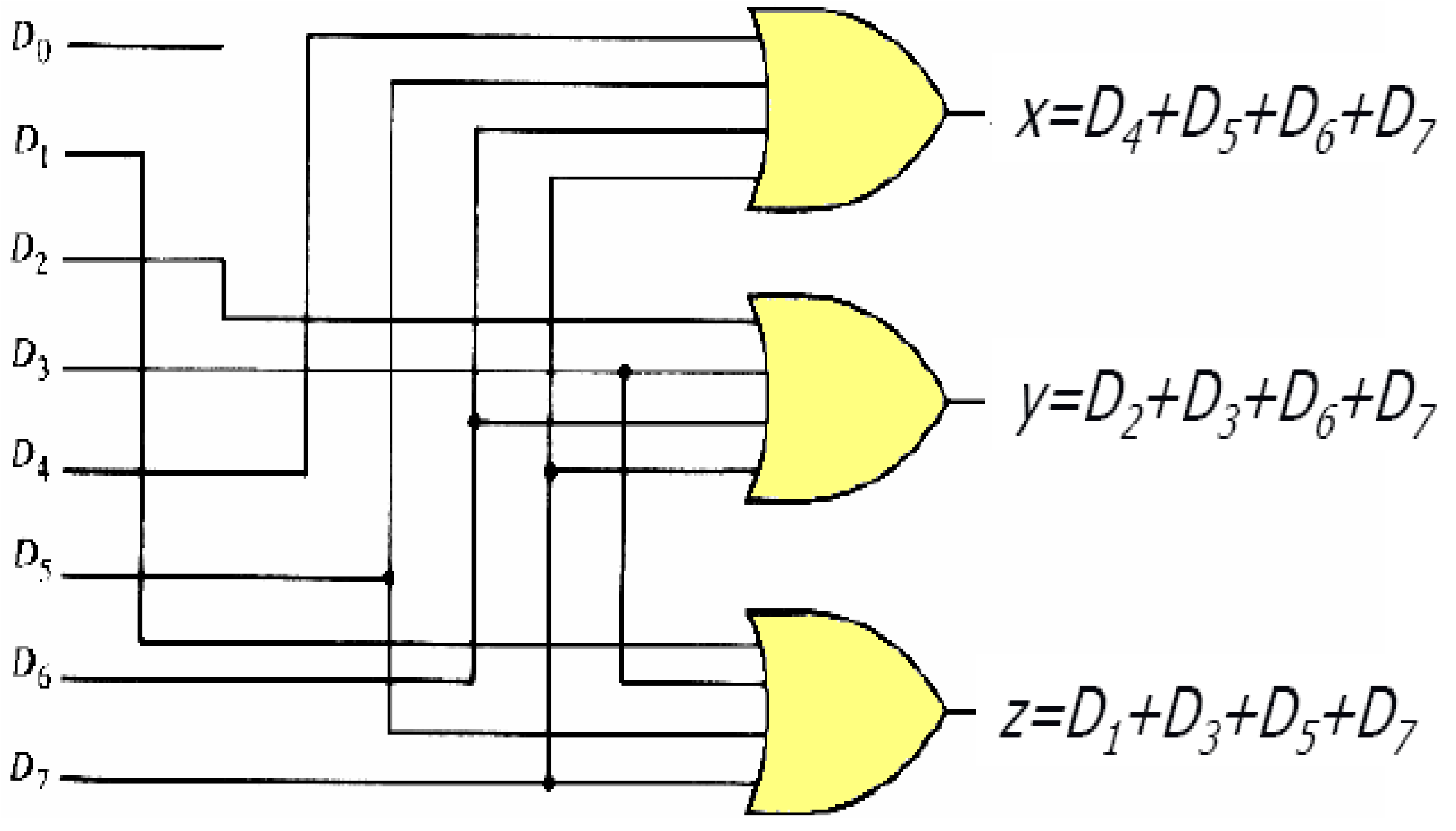
- An encoder is a digital circuit that performs the inverse operation of a decoder.
- Example:
- Truth table of an Octal-to-Binary Encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

- عندما يكون عدد المخرجات n فان عدد مدخلات encoder تساوي 2^n أي يعمل عكس عمل decoder
- $X = D4+D5+D6+D7$
- $Y = D2+D3+D6+D7$
- $Z = D1+D3+D5+D7$

- The encoder can be implemented with three OR gates as follows:

An implementation

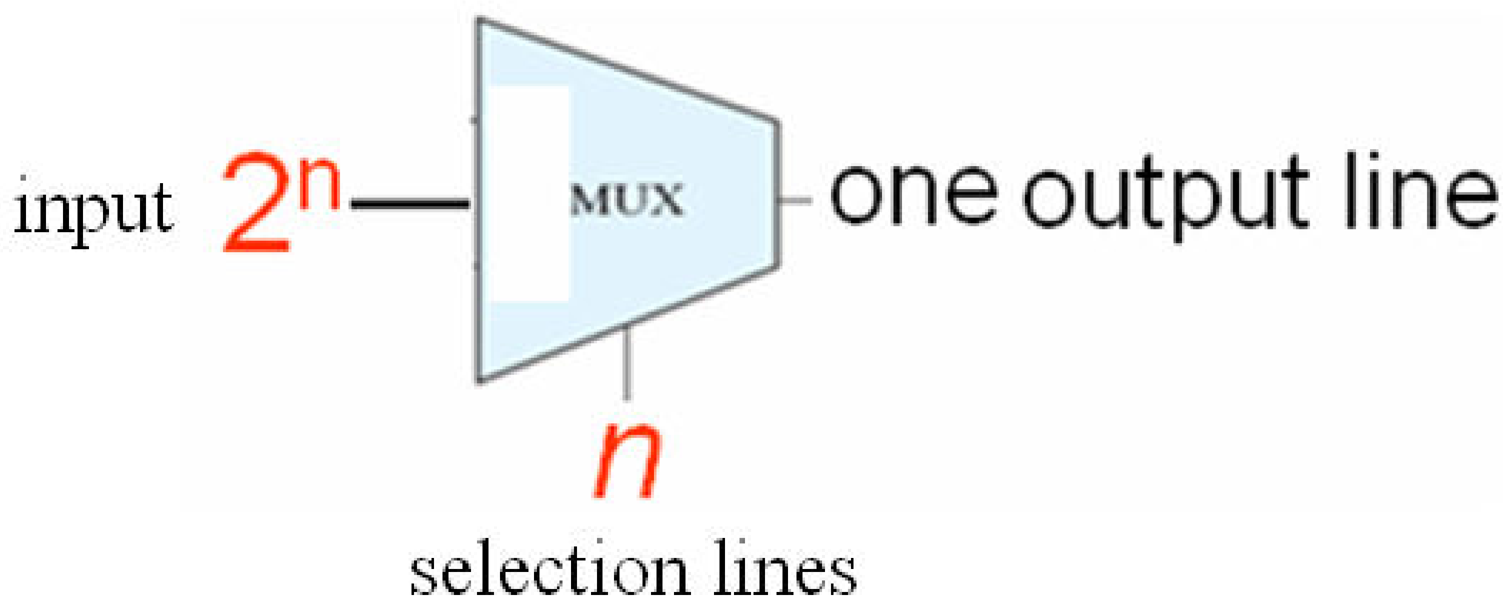


Multiplexers

- MUX: a decoder + OR gate
- A multiplexer is a combinational circuit that **selects** binary information from one of many input lines and direct it to a **single output line.**

- هو أن يختار معلومة واحدة من المعلومات (المدخلات) الثنائية من العديد من خطوط الإدخال وتوجيهها إلى خط واحد كمخرجات (تكون المخرجات فقط واحد من عناصر المدخلات) **محكومة** بخط اختيار لأحد عناصر المدخلات

- The selection of a particular input line is controlled by a set of selection lines.
  - اختيار خط محدد واحد من خطوط المدخلات محكوم بمجموعه خطوط اختيار(selection).



- There are 2^n input lines, **n** selection lines and **one** output line

- مدخلات دالة إل 2^n multiplexer ومخرجاتها **1** وعدد الاختيارات هو **n** selection lines

- مثال لو كان عدد المدخلات هو $4=2^2$ فان عدد المخرجات **1** وعدد اختيارات الدالة هو **2**

- و لو كان عدد المدخلات هو $8=2^3$ فان عدد المخرجات **1** وعدد اختيارات الدالة هو **3**

- **Example**: 2-to-1-line multiplexer
  - يعني أن دائرة Multiplexer لها خطين من المدخلات ليتم اختيار احدهما بناءا على selection واحد يمثل حالتين 0 و 1
  - فاذا كان **S=0** فان مخرجات الدالة تكون **B** والا مخرجات الدالة تكون **A**.
- A Multiplexer has **2^1=2** inputs, **1** output and **1** selection
- **Truth table**

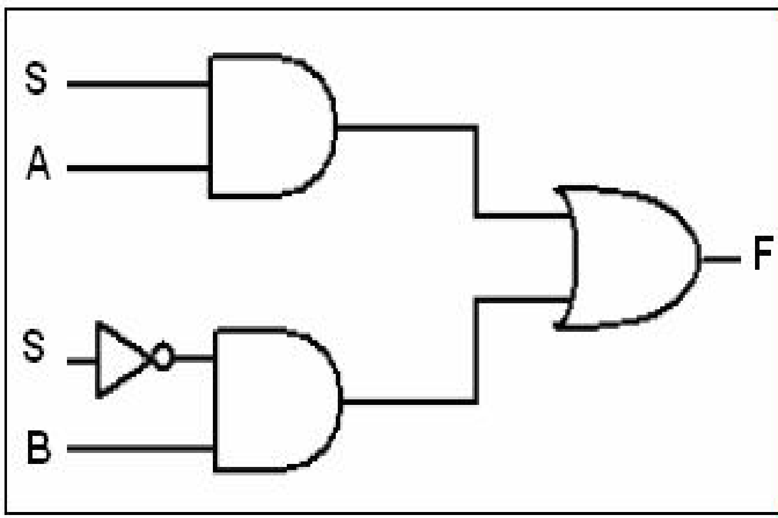
Selection	Variables		Output
S	A	B	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

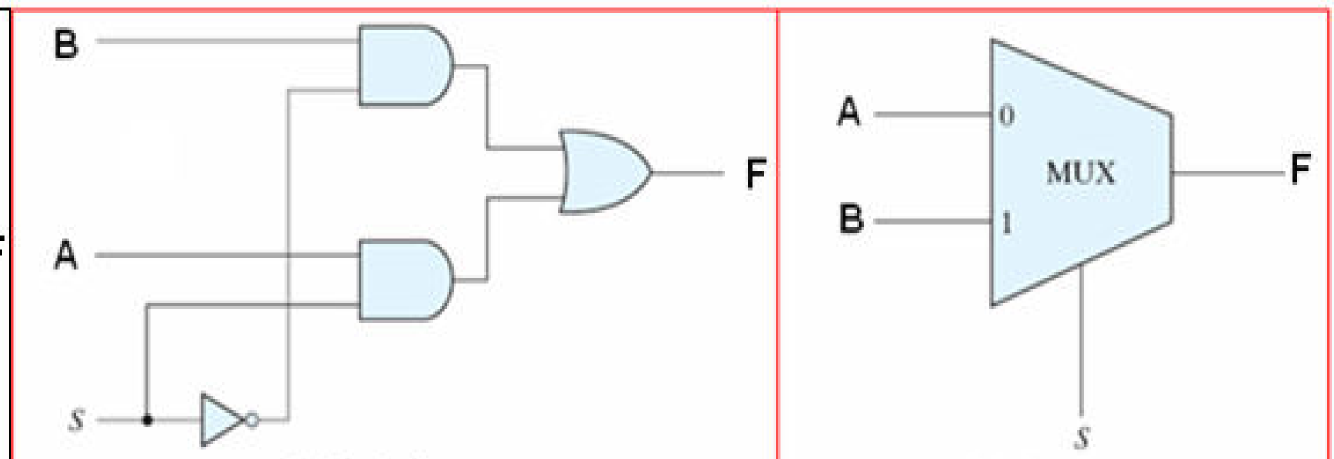
- يكون خط واحد من خطوط المدخلات هو مخرجات الدائرة بناءا على خط الاختيار المحدد .

- **map**

- The **Function** of MUX is **F = S’B+SA**

Logic and Block Diagrams





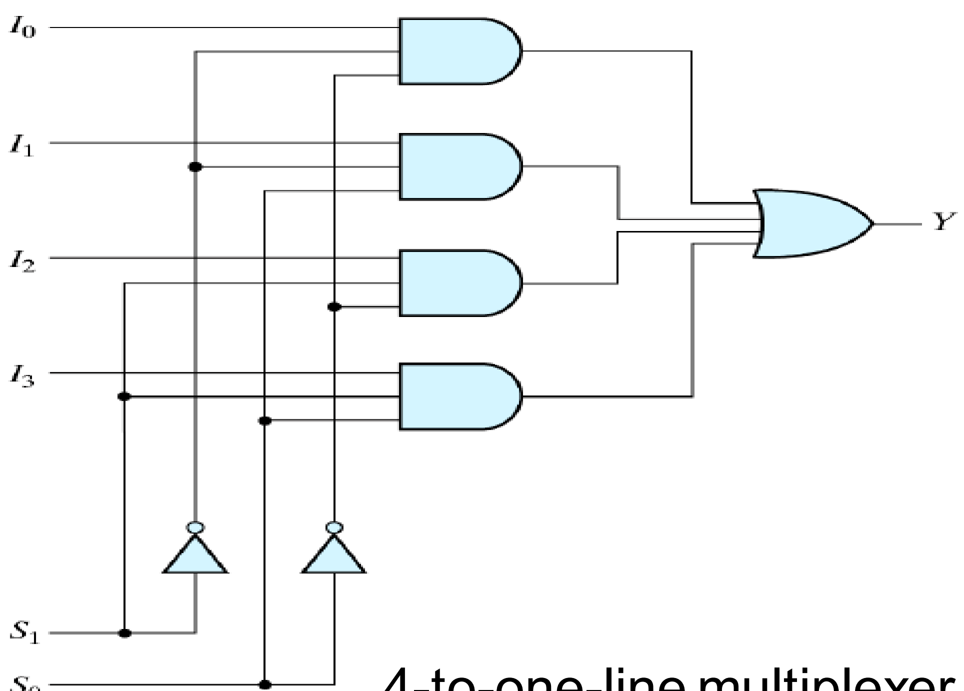
(a) Logic diagram

(b) Block diagram

- وصف نتائج العمل:
- إذا كان **S=0** فان قيم F =B وإذا كانت **S=1** فان قيمة F=A.

4-to-1-Line Multiplexer

- عدد مخرجات الدائرة **1** وعدد المدخلات **4** $=2^2$ يتم اختيار واحده منها كمخرجات بناءا على مجموعة الاختيار selections وهي 2. $S_0\ S_1$ عمل جدول للمتغيرات يبدو غير مجدي ولكن يمكن عمله في هذه الحالات لأنه مهما كان حال المتغيرات وكانت $S_1\ \&\ S_0 = 0$ فان دائما $Y=I_0$ وهكذا لبقية الحالات لذلك يتم اختصار truth table ب function table للاختصار؟



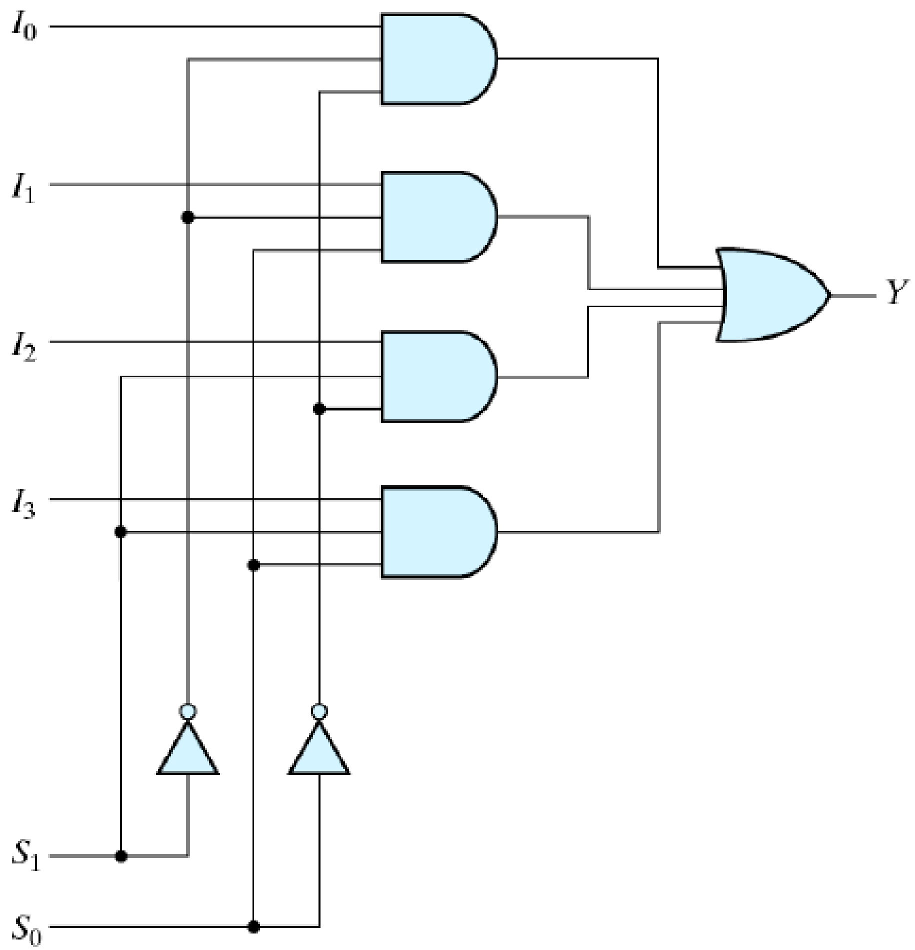
4-to-one-line multiplexer

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Function table

S_1	S_0	I0	I1	I2	I3
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1
0	1	الحالة الثانية			
1	0	الحالة الثالثة			
1	1	الحالة الرابع حتى 64 صف			

$$Y = I_0 S'_1 S'_0 + S'_1 S_0 I_1 + S_1 S'_0 I_2 + S_1 S_0 I_3$$



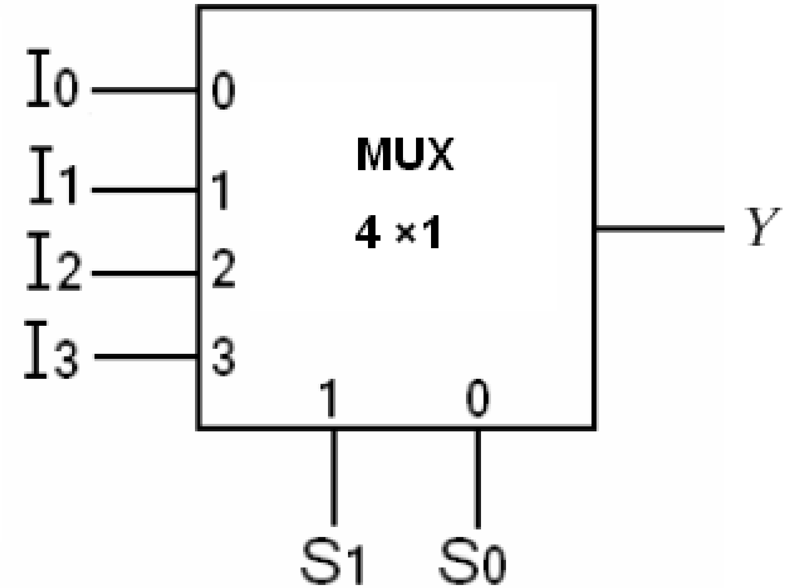
4-to-one-line multiplexer

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Function table

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Function table



In this case **$Y=I_2$**

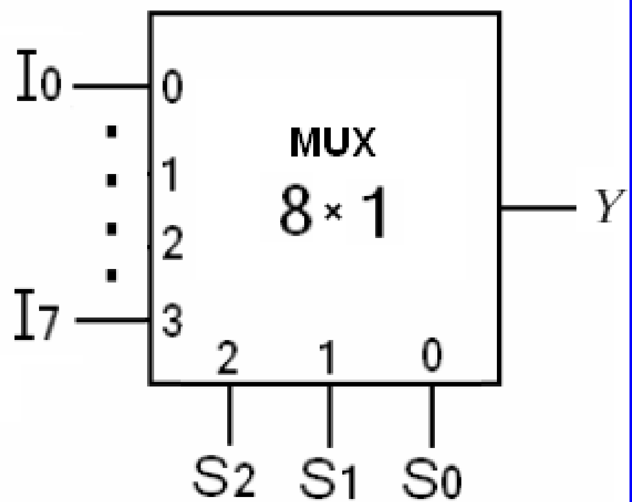
- If $S_1=0$ and $S_0=0$ the output is $y=I_0$
- If $S_1=0$ and $S_0=1$ the output is $y=I_1$
- If $S_1=1$ and $S_0=0$ the output is $y=I_2$
- If $S_1=1$ and $S_0=1$ the output is $y=I_3$
- في حالة أن يكون $S_1=0$ و $S_0=0$ فان النتيجة $y=I_0$ لان بقية الخطوط تكون في حالة صفر و هكذا

8-to-1-Line Multiplexer

- عدد مخرجات الدائرة **1**
- وعدد المدخلات **8** يتم اختيار واحده منها كمخرجات بناءا على مجموعة **الاختيار**
- مجموعة الاختيارات selections تساوي **3** وهي $\mathbf{S_0\ S_2\ S_1}$

S_2	S_1	S_0	Y
0	0	0	*I0*
0	0	1	*I1*
0	1	0	*I2*
0	1	1	*I3*
1	0	0	*I4*
1	0	1	*I5*
1	1	0	*I6*
1	1	1	*I7*

Function table



- Y= I_0 if S_0=0 S_2=0 S_1=0
- Y= I_1 if S_0=0 S_2=0 S_1=1 *and so on*.