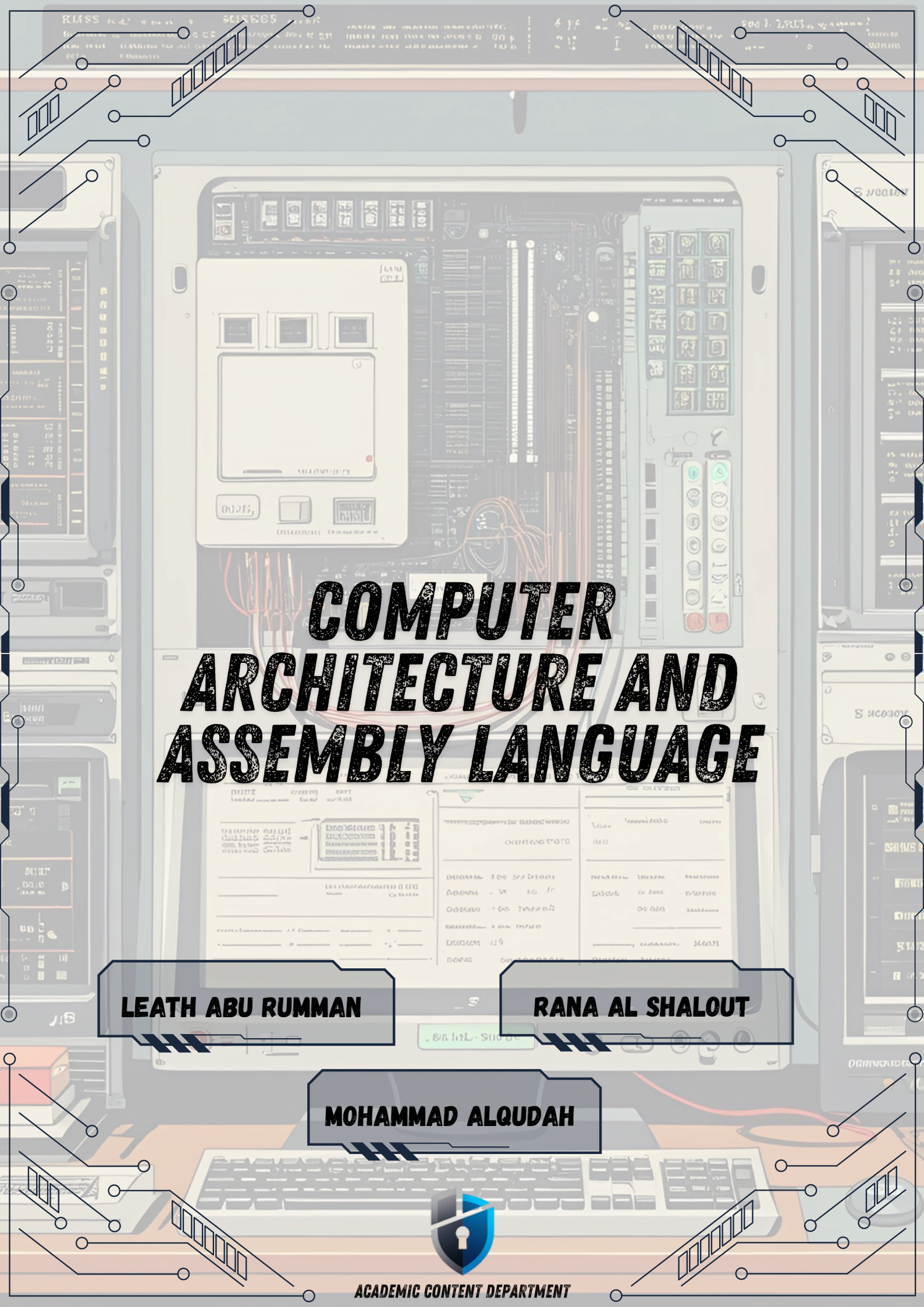


COMPUTER ARCHITECTURE AND ASSEMBLY LANGUAGE

LEATH ABU RUMMAN

RANA AL SHALOUT

MOHAMMAD ALQUDAH

ACADEMIC CONTENT DEPARTMENT

TABLE OF CONTENT

CHAPTER ONE

DIGITAL SYSTEMS AND BINARY NUMBERS

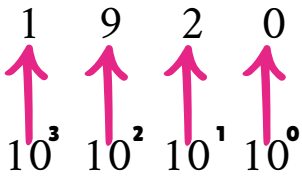
Primary Number Systems in Digital Computing :

1-Decimal number system

→ base10

Each digit may have one of 10 values (0 to 9).

ex: $(1920)_{10}$



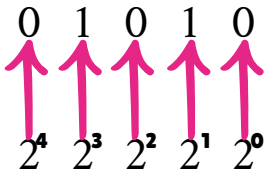
=1920

2-binary number system

→ base2

Each bit has two values: zero or one.

ex: $(01010)_2$



$=(0*2^0)+(1*2^1)+(0*2^2)+(1*2^3)+(0*2^4)=0+2+0+8+0=10$

3-Octal number system

→ base8

Each digit may have one of 8 values (0 to 7).

ex: $(274)_8$

4- system number Hexadecimal

→ base16

Each digit may have one of 16 values (0 to 15).

A = 10

B = 11

C = 12

D = 13

E = 14

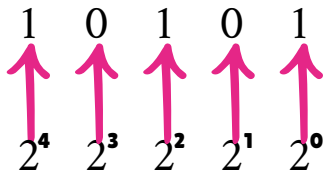
F = 15

ex: $(24E73)_{16}$

conversions

1- convert Binary to Decimal.

ex1: convert $(10101)_2$ to decimal

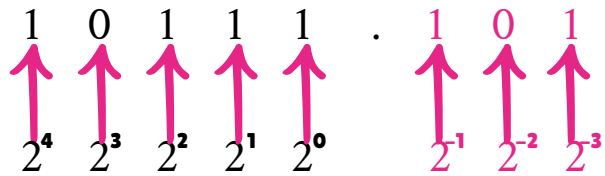


$=(1*2^0)+(0*2^1)+(1*2)+(0*2)+(1*2)= 1+4+16 =21$

$(10101)_2=(21)_{10}$

- $2^{-1}= ½ = 0.5$
- $2^{-2}= ¼ = 0.25$
- $2^{-3}= ⅛ =0.125$

ex2: (10111.101) to decimal

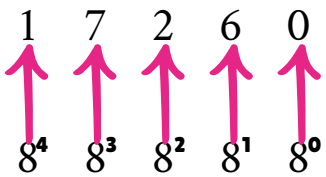


$=(1*2^0)+(1*2^1)+(1*2^2)+(0*2^3)+(1*2^4)$
$+(1*2^{-1})+(0*2^{-2})+(1*2^{-3})=1+2+4+16+$
$0.5+0.125=23.625$

$(10111.101)_2= (23.625)_{10}$

2- convert Octal to Decimal.

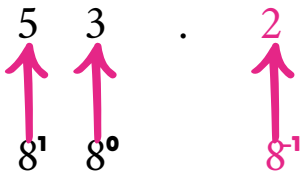
ex1: Convert $(17260)_8$ to Decimal



$=(0*8^0)+(6*8^1)+(2*8^2)+(7*8^3)+(1*8^4)=7856$

$(17260)_8=(7856)_{10}$

ex2: Convert $(53.2)_8$ to Decimal

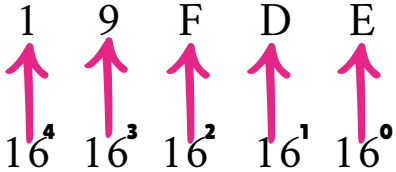


$=(3*8^0)+(5*8^1)+(2*8^{-1})= 3+40+(2*⅛)=43.25$

$(53.2)_8=(43.25)_{10}$

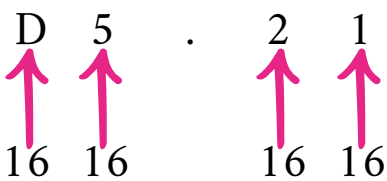
3- convert Hexadecimal to Decimal.

ex1: Convert $(19FDE)_{16}$ to Decimal



$=(E*16^0)+(D*16^1)+(F*16^2)+(9*16^3)+(1*16^4)=$
$=106462$

ex2: Convert $(D5.21)16$ to Decimal



$=(5*16^0)+(D*16^1)+(2*16^{-1})+(1*16^{-2})=213.128$

$(D5.21)_8=(213.128)_{10}$

4- convert Decimal to Binary.

ex1: Convert $(29)_{10}$ to Binary.

29

16 ✗ 32 ✓ ② ← 2 نجد اكبر رقم من مضاعفات ال ① .

هون عشان نحول من **decimal** الى **binary** عنا عده خطوات:

1. نشوف اقرب رقم من مضاعفات ال **2** على الرقم تبعنا.
2. نختار المضاعف الاكبر .
3. نفرط الرقم الى مضاعفات ال **2** (لحد ما نوصل للرقم الي اخترناه)

29-32 ✗
29-16=13 ✓
13 - 8 =5 ✓
5 - 4 =1 ✓
1 - 2 = ✗
1 - 1 =0 ✓

29

0	1	1	1	0	1	③
32	16	8	4	2	1	

$(29)_{10}=(11101)_2$

ex1: Convert $(29.625)_2$ to Binary

16 ✗ 32 ✓

29.625

0	1	1	1	0	1	.	1	0	1
32	16	8	4	2	1		½	¼	⅛
							0.5	0.25	0.125

$(29.625)_{10}=(11101.101)_2$

5-convert Decimal to Octal.

ex: Convert $(35)_{10}$ to Octal.

Step	Operation	Result	Remainder
Step1	35/8	4	3
Step2	4/8	0	4

$(35)_{10}=(43)_{8}$

6- convert Decimal to Hexadecimal.

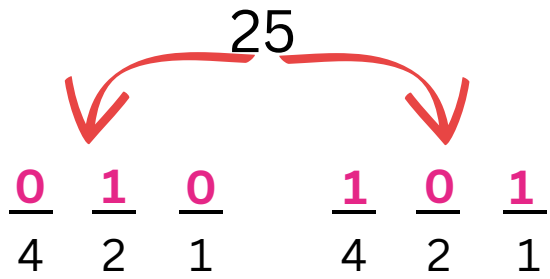
ex: Convert $(43)_{10}$ to hexadecimal

Step	Operation	Result	Remainder
Step1	43/16	2	11(B)
Step2	2/16	0	2

$(43)_{10}=(2B)_{16}$

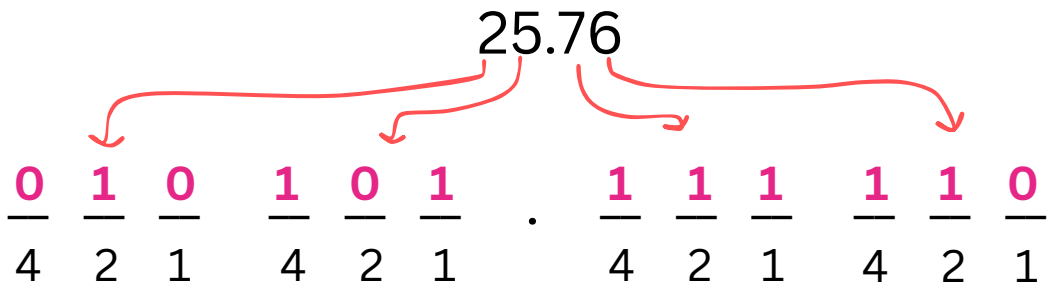
7- Convert octal to binary

ex1: Convert $(25)_8$ to binary.



$(25)_8=(010\ 101)_2$

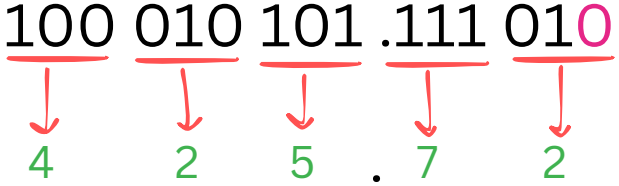
ex2: Convert $(25.76)_8$ to binary.



$(2576)_8=(010\ 101\ .\ 111\ 110)_2$

8-Convert binary to octal

ex1: Convert $(100010101.11101)_2$ to Octal.



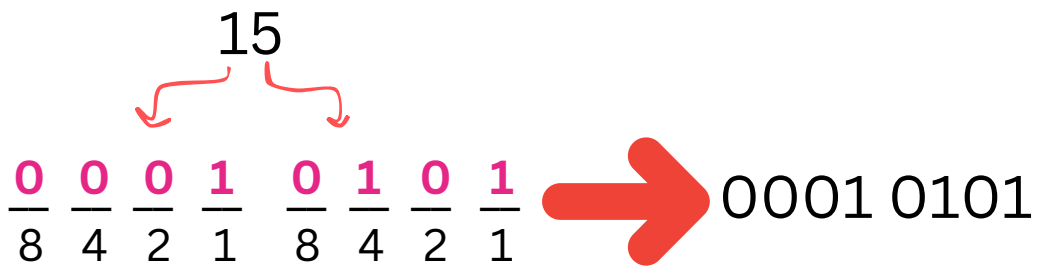
اذا نقص علينا خانات بنضيف صفر:

- على اليمين (اذا كنا يمين الفاصله)
- على اليسار (اذا كنا يسار الفاصله)

$(100\ 010\ 101\ .\ 111\ 01)_2=(425.72)_8$

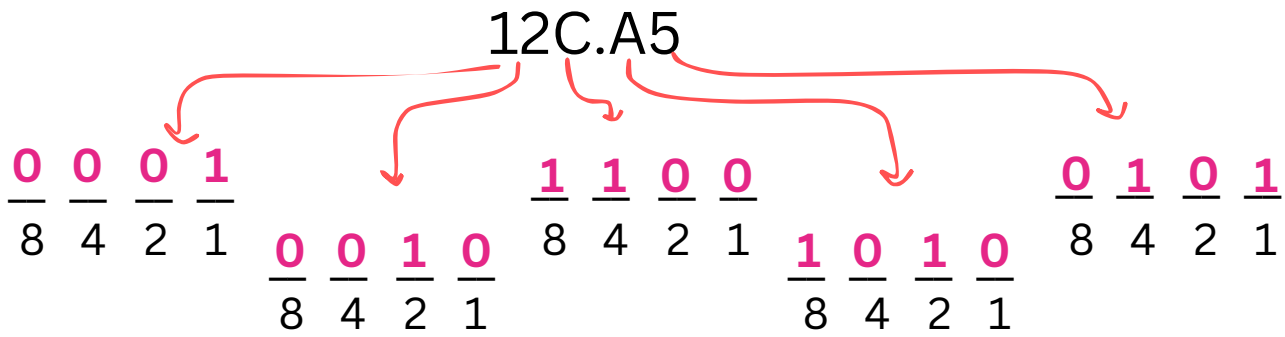
9- Convert Hexadecimal to binary

ex1: Convert $(15)_{16}$ to binary



$(15)_{16}=(00010101)_2$

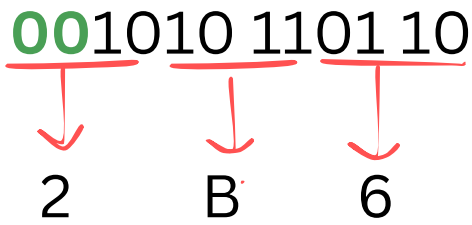
ex2: Convert $(12C.A5)_{16}$ to binary



$(12C.A5)_{16}=(0001\ 0010\ 1100\ .\ 1010\ 0101)_2$

10- Convert binary to Hexadecimal

ex1: Convert $(1010110110)_2$ to Hexadecimal



بنضيف صفرين على اليسار
اسمهم padding

$(1010110110)_2=(2B6)_{16}$

ex2: Convert $(1101011.00101)_2$ to Hexadecimal

0110 1011.0010 1000 → padding

6 B . 2 8

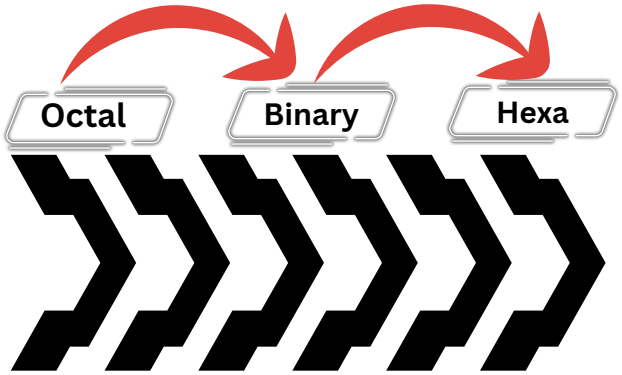
padding

اذا نقص علينا خانات بنضيف صفر:

- على اليمين (اذا كنا يمين الفاصله)
- على اليسار (اذا كنا يسار الفاصله)

$(1101011.00101)_2 = (6B.28)_{16}$

11- Convert Octal to Hexadecimal



بنحول الرقم من Octal الى binary
بعدها بنحولوا الى Hexadecimal

ex: Convert $(325.67)_8$ to Hexadecimal

325.67_8

0 1 1 0 1 0 1 0 1 . 1 1 0 1 1 1

1 $(325.67)_8 = (011010101.110111)_2$

هسا بنوخد الناتج(الي هو Binary) وبنحولوا الى Hexadecimal زي ما تعلمنا فوق

تهمل

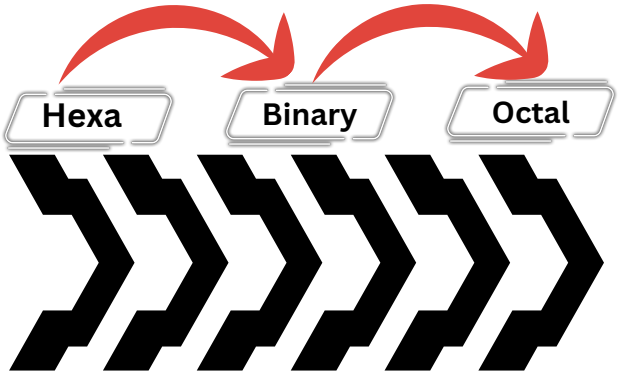
0 1101 0101.1101 1100

2

B 5 . D C

$(325.67)_8 = (B5.DC)_{16}$

12- Convert Hexadecimal to octal



بنحول الرقم من Hexa الى binary
بعدها بنحولوا الى Octal

ex: Convert $(325.67)_{16}$ to Octal

325.67_{16}

0 0 1 1 0 0 1 0 0 1 0 1 . 0 1 1 0 0 1 1 1

001 100 100 101 .011 001 110

1 4 4 5 . 3 1 6

$(325.67)_{16} = (1445.316)_8$

هيك بنكون خلصنا موضوع التحويلات بين انضمة العد.

Binary number operations

عنا عده عمليات على الBinary numbers منهم:

- **Binary addition:**

Rules:
0 + 0 = 0
0 + 1 = 1 1 + 0 =1
1 + 1 = 0 (تساوي 0 وواحد باليد)
1 + 1 +1 = 1 (تساوي 1 وواحد باليد)

Ex:

10011 + 11101=?

$$\begin{array}{r} \overset{1\,1\,1\,1}{10011} \\ 11101 \\ \hline 110000 \end{array} +$$

- **Binary subtraction:**

Rules:
0 - 0 = 0
1 - 0 = 1
0 - 1 = 1 (خذ من جارك)
1 - 1 = 0

Ex:

10011 - 01101=?

$$\begin{array}{r} \overset{0\,\not{1}\,\not{2}\,2}{\not{1}\not{0}011} \\ 01101 \\ \hline 00110 \end{array} -$$

- **Binary multiplication:**

Rules:
0 * 0 = 0
1 * 0 = 0
0 * 1 = 0
1 * 1 = 0

Ex:

1101 * 101=?

```
   1101
 *  101
 ------
 1 1101
 0000
1101
-------
1000001
```

SignedBinaryNumbers

Positive integers (including zero) can be represented as unsigned numbers. However, to represent negative integers, we need a notation for negative values

- The sign is represented with a bit placed in the leftmost position of the number. The sign bit 0 to represent positive numbers and 1 for negative.

مثل ما عنا ارقام موجبه وسالبه بال decimal numbers برضو عنا ارقام موجبه وسالبه بال Binary numbers... بس كيف بنميزهم؟
شايف اخر bit بالرقم من اليسار؟

10001001

هاظ الbit لو كان صفر ← بكون الرقم موجب
ولو كان 1 بكون الرقم سالب

The user determines whether the number is signed or unsigned, If the binary number is signed, then the leftmost bit represents the sign and the rest of the bits represent the number i.e., 11001 represents -9.
If the binary number is assumed to be unsigned, then the leftmost bit is the most significant bit of the number i.e., 11001 represents 25.

Ex: To represent the number -9:

رقم واحد هون يعني ان الرقم سالب

Magnitude representation: 1 0 0 0 1 0 0 1

9

1’s-complement representation:

بحول الرقم السالب الى موجب -9 ← +9
بعدين بمثل الرقم
هسا بعكس كل 1 الى 0
وكل 0 الى 1

00001001
↓
11110110 ➡ 1’s-complement representation

2’s-complement representation:

نفس خطوات ال 1’s-complement بس زياده عليها خطوه وهي انو نجمع للناتج 1

$$\begin{array}{rl} & 11110110 \rightarrow \text{1’s-complement representation:} \\ + & 00000001 \rightarrow \text{اضافة واحد} \\ \hline & 11110111 \rightarrow \text{2’s-complement representation:} \end{array}$$

- **The 2’s-complement is the most used representation**

2’s complement Addition

Ex:

6 + 13=?

```
  11
  00000110
+ 00001101
----------
  00010011  → 19
```

-6 - (-13)=? -6 + 13 =?

نستخدم 2’s complement

+6 → 00000110 (1’s complement) → 11111001 (2’s complement) → 11111010

```
  11111010
+ 00001101
----------
 100000111  → 7
```

شطبنا الواحد عشان ما نتعدى ال8 خانات

+6 −13 = ? +6 + (-13) = ?

```
  00000110
+ 11110011   ← -13 in 2’s-complement
----------
  11111001   ← -7
```

−6 −13 = ? −6 + (−13) = ?

```
  11111010   ← -6 in 2’s-complement
+ 11110011   ← -13 in 2’s-complement
----------
 111101101   ← -19
```

Digital system are used in everywhere in our life such as communication, traffic control, medical treatment, weather monitoring, the Internet, and many other areas. They use just two discrete values and are therefore said to be binary.

The general-purpose digital computer is the best-known example of a digital system.

تُستخدم الأنظمة الرقمية في مختلف جوانب حياتنا اليومية، مثل الاتصالات، والتحكم في حركة المرور، والعلاج الطبي، ورصد الأحوال الجوية، والإنترنت، وغيرها من المجالات. وتعتمد هذه الأنظمة على قيمتين منفصلتين فقط، ولهذا تُعرف بأنها أنظمة ثنائية (Binary).

ويُعد الحاسوب الرقمي متعدد الأغراض المثال الأبرز والأكثر شيوعًا على الأنظمة الرقمية

• Bit: A binary digit which has two numerical values: 0 and 1.

• Binary codes: Discrete elements of information are represented with groups of bits called binary codes, such as numbers and symbols.

The major parts of a computer are:

1) **Memory unit**, which stores programs as well as input, output, and intermediate data.

2) **Central processing unit**, which performs arithmetic and other data processing operations as specified by the program.

3) **Input–output units**: The program and data prepared by a user are transferred into memory by means of an input device such as a keyboard and an output device, such as a printer, receives the results of the computations, and the printed results are presented to the user.

Data size:

2^{10} is referred to as K (kilo)
2^{20} is referred as M (mega)
2^{30} is referred as G (giga)
2^{40} is referred as T (tera)

Ex:

$4K = 4*2^{10} = 2^{2}*2^{10} = 2^{12} = 4,096$

$16M = 16*2^{20} = 2^{4} * 2^{20} = 2^{24} = 16,777,216$

Computer memory capacity are usually given in bytes which is equal to eight bits.
For example, a computer hard disk with four gigabytes of storage has a capacity of $4G = 2^{32}$ bytes (approximately 4 billion bytes).

Binary-Coded Decimal Code (BCD) and ASCII code

BCD requires **four-bit** code for each decimal digit.
Decimal 396 is represented in BCD with 12 bits as 0011 1001 0110, with each group of four bits representing one decimal digit.

- It is used when we care about the performance, not the storage.

يعني بنوخذ كل رقم لحال وبنحوله للنظام الثنائي(binary)، وهاي الطريقه اسهل واسرع بس بتوخذ ذاكره اكبر.

Many applications of digital computers require the handling not only of numbers but also of other characters or symbols, such as the letters of the alphabet. The standard binary code for the alphanumeric characters is the American Standard Code for Information Interchange (ASCII), which uses seven bits to code 128 characters. For example, the letter A, is represented in ASCII as **1000001**.

CHAPTER TWO

BOOLEAN ALGEBRA AND LOGIC GATES

Boolean Algebra – Definition

A two-valued Boolean algebra is defined on a set of two elements
B = {0, 1} with rules for the two binary operators (·) => and ,(+) => or .
The binary operators are closed, commutative, and associative .

- 0 is the identity element for the (+) operator
- 1 is the identity element of the (·) operator

شو يعني هاد الحكي ؟
لما يكون عندي صفر (+),or أي متغير سواء 1 او 0 الجواب رح يكون المتغير نفسه
اما لما يكون عندي واحد (.),and أي متغير الجواب رح يكون المتغير نفسه

Truth Table

x	y	x . y
0	0	0
0	1	0
1	0	0
1	1	1

and operator

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

or operator

x	x'
0	1
1	0

prime (يعني عكس الرقم)

Boolean Algebra – Axioms and Theorems البديهيات والنظريات

x + 0 = x	-------	اي متغير or الصفر رح يعطينا المتغير نفسه
x . 0 = x	-------	اي متغير and الرقم 1 رح يعطينا المتغير نفسه
x + x' = 1	-------	اي متغير or الرقم الي عسكه رح يعطينا 1
x . x'= 0	-------	اي متغير and الرقم الي عكسه رح يعطينا 0
x + x = x	-------	اي متغير or الرقم نفسه رح يعطينا نفس المتغير
x . x = x	-------	اي متغير and الرقم نفسه رح يعطينا نفس المتغير

$x + 1 = 1$	-------	اي متغير or الرقم 1 رح يعطينا الرقم 1
$x \cdot 0 = 0$	-------	اي متغير and الرقم 0 رح يعطينا 0
$(x')' = x$	involution	$(1')' = 0' = 1$
$x + y = y + x$ $x.y = y.x$	commutative	
$x + y.z = (x + y)(x + z)$ $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	distributive	بنوزع ال x على كل المتغيرات بالطرف الثاني
$(x + y)' = x'.y'$ $(x. y)' = x' + y'$	DeMorgan	اذا شفت نفي لمجموعة داخل أقواس : بدل العملية (OR ⇄ AND) وانفِ كل متغير لحاله
$x + xy = x$	absorption	$x + xy = x.1 + xy$ $= x(1 + y)$ $= x$
$x.(x + y) = x$	duality	

distributive => التوزيع
Involution => نفي النفي برجع الاصل
commutative => علاقة تبادلية
Associative => علاقة ترابطية

Boolean Functions

_ A Boolean function described by an algebraic expression consists of binary variables, the constants 0 and 1, and the logic operation symbols

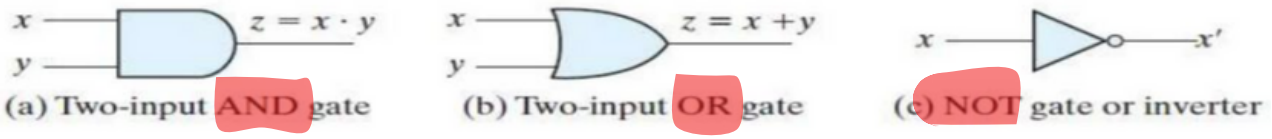
_ For a given value of the binary variables, the function can be equal to either 1 or 0.

*** Example, F = x + y′. z**

_ A Boolean function can be represented in a truth table

Boolean Function هي معادلة تستخدم:
- متغيرات ثنائية (0 أو 1)
-الثوابت 0 و 1
- العمليات المنطقية زي(NOT=> ′ ، AND => · ، OR => +)
* لما نعوّض بقيم للمتغيرات الناتج بكون اما 1 او 0

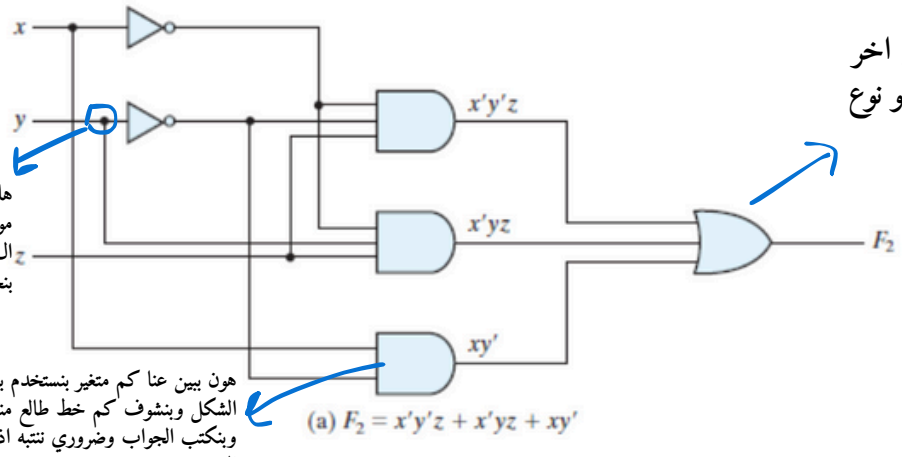
Logic Gates

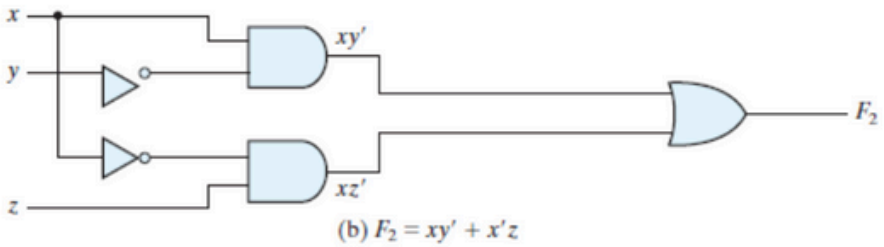


احنا دايما بنتطلع على اخر gate عشان نعرف شو نوع العملية بين ال terms(الحدود)

هاي معناها انه في المتغير نفسه موجود بأكثر من جهة اذا كان قبل ال not نستخدمه زي ما هو اذا بعد بنحطله ` زي ما رح نشوف بالأمثلة

هون بيين عنا كم متغير بنستخدم بكل term بنتتبع الشكل وبنشوف كم خط طالع منه ولأي متغير بوديني وبنكتب الجواب وضروري نتتبه اذا الخط قبل او بعد ال not gate





Simplify the following Boolean expressions to a minimum number of literals.

$x(x`+ y)$:

$x`.x + x.y$

$= 0 + xy$

$= xy$

اول اشي وزعنا ال x على القوس
وهون حكينا انه المتغير and عسكه بعطينا صفر

$x + x`y$:

$= (x+x')\ (x+y)$

$= 1.(x+y)$

$= x + y$

$(x + y)(x + y`)$:

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

$=x+xy'+xy+0$

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

$=x(1)$

$=x$

اول اشي وزعنا المتغيرات بالاقواس على بعض
بعدين بسطنا حسب الجدول الموجود بصفحة 1

$(x`+ yz`)+ x`y`$:

$= xy + x'z + yz(x + x')$

$= xy + x'z + xyz + x'yz$

$= xy(1 + z) + x'z(1 + y)$

$= xy + x'z$

هاد المثال شوي غير لانه بدنا نضرب ب 1
الي هو $(x+x`)$ عشان نعوض نقص
المتغيرات بواحد من الاطراف وهو ما بأثر لانه
زي كأنه ضربنا بواحد

Minterms and Maxterms

_ Minterm is obtained from an AND term of the n variables, with each variable being primed if the corresponding bit of the binary number is a 0 and unprimed if a 1.

اول اشي ال minterm بتكون الاشارة بين المتغيرات and (.) وال 0 بمثل المتغير معه نفي ` وال 1 بمثل المتغير زي ما هو زي ما رح نشوف ال truth table

لنفرض عنا 3 متغيرات A,B,C
لو كانت ال bits = 1,0,1,
معناته ال m = A.B`.C

_ Maxterm is obtained from an OR term of the n variables, with each variable being primed if the corresponding bit of the binary number is a 1 and unprimed if a 0.

بالنسبة لل maxterm الاشارة بين المتغيرات بتكون or وال 0 بمثل المتغير نفسه وال 1 بمثل المتغير منفي يعني عكس ال minterm

لنفرض عنا 3 متغيرات A,B,C
لو كانت ال bits = 1,0,1,
معناته ال m = A`+B+C`

_ Each maxterm is the complement of its corresponding minterm and vice versa

كل maxterm هو نفي لل minterm بنفس الرقم والعكس صحيح !

			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

Canonical Forms

Sum of Minterms (SoM) :

لما تكون ال F=1 وبكون عندي and بين المتغيرات و or بين الحدود
مثال : A.B+ A`.B وهاي اشارتها ∑

Product of Maxterms (PoM):

لما تكون ال F=0 وبكون عندي or بين المتغيرات و and بين الحدود
مثال : (A`+B) . (A+B) وهاي اشارتها ∏

Example SOM

	A	B	C	F
m_0	0	0	0	0
m_1	0	0	1	1
m_2	0	1	0	0
m_3	0	1	1	0
m_4	1	0	0	1
m_5	1	0	1	0
m_6	1	1	0	1
m_7	1	1	1	0

★ Consider the 1s of the function
★ AND the input variables to produce 1

$0 \cdot 0 \cdot 1 \Rightarrow \bar{0} \cdot \bar{0} \cdot 1 \Rightarrow \bar{A} \cdot \bar{B} \cdot C$

+

$1 \cdot 0 \cdot 0 \Rightarrow 1 \cdot \bar{0} \cdot \bar{0} \Rightarrow A \cdot \bar{B} \cdot \bar{C}$

+

$1 \cdot 1 \cdot 0 \Rightarrow 1 \cdot 1 \cdot \bar{0} \Rightarrow A \cdot B \cdot \bar{C}$

$F(A,B,C) = A'B'C + AB'C' + ABC'$

$F(A,B,C) = m_1 + m_4 + m_6$

$F(A,B,C) = \sum (1,4,6)$

Example

★ Derive the truth table, SoM logic expression and logic diagram of

$F(x,y,z) = \sum (0,3,5,6)$

	x	y	z	F
m_0	0	0	0	1
m_1	0	0	1	0
m_2	0	1	0	0
m_3	0	1	1	1
m_4	1	0	0	0
m_5	1	0	1	0
m_6	1	1	0	1
m_7	1	1	1	0

$F(x,y,z) = m_0, m_3, m_5, m_6$

$F(x,y,z) = x'y'z' + x'yz + xy'z + xyz'$

هون بناء على ال 3 متغيرات الي عندي بدي اعمل truth table لل 3 متغيرات وعند الارقام الي معطيني اياهم رح احط 1 عند ال F وبعدين زي ما قلنا فوق عند ال 0 رح نحط المتغير منفي وعند ال 1 رح نحط المتغير مثل ما هو :)

Using Algebra to find SOM

★ Use algebra to write the SOM expression for

$F(A,B,C,D) = A(BC + BC') + A'B'C$

$F(A,B,C,D) = ABC + ABC` + A`B`C$

$F(A,B,C,D) = (ABC.1) + (ABC`.1) + (A`B`C.1)$

$F(A,B,C,D) = (ABC.(D+D`)) + (ABC`.(D+D`)) + (A`B`C.(D+D`))$

$F(A,B,C,D) = ABCD + ABCD` + ABC`D + ABC`D` + A`B`CD + A`B`CD`$

$F(A,B,C,D) = m_{15} + m_{14} + m_{13} + m_{12} + m_3 + m_2$

$F(A,B,C,D) = \sum 15, 14, 13, 12, 3, 2.$

$F(A,B,C,D) = \sum 2, 3, 12, 13, 14, 15.$

او بطلب السؤال زي هيك هون اول اشي رح نوزع واذا عندي متغير ناقص بواحد من الحدود بضيفه عن طريق انه بضربه ب (D+D`) الي هو يساوي 1 يعني ما بأثر على الحل وطبعا حسب المتغير الي ناقص يعني مش شرط D

Product of Maxterms (PoM)

★ Consider the 0s of the function

★ OR the input variables to produce 0

	x	y	F(x,y)
M_0	0	0	0
M_1	0	1	1
M_2	1	0	0
M_3	1	1	1

$0 + 0 \Rightarrow 0 + 0 \Rightarrow x + y$

$1 + 0 \Rightarrow \overline{1} + 0 \Rightarrow \overline{x} + y$

$F(x,y) = (x + y) \bullet (x' + y)$ *Product of Maxterms (POM)*

$F(x,y) = M_0 \bullet M_2$

$F(x,y) = \prod (0,2)$

maxterm
Sum (*OR*) term that contains all variables

Example

★ Derive the truth table, PoM logic expression and logic diagram of

$F(x,y,z) = \prod (1,7)$

	x	y	z	F
M_0	0	0	0	1
M_1	0	0	1	0
M_2	0	1	0	1
M_3	0	1	1	1
M_4	1	0	0	1
M_5	1	0	1	1
M_6	1	1	0	1
M_7	1	1	1	0

$F(x,y,z)=M_1 ,M_7$

$F(x,y,z) = (x+y+z')(x'+y'+z')$

★ If $F(w,x,y,z) = \sum (1,2,4,9,12,13,15)$, then

- $F'(w,x,y,z) = \sum (0,3,5,6,7,8,10,11,14)$
- $F(w,x,y,z) = \prod (0,3,5,6,7,8,10,11,14)$
- $F'(w,x,y,z) = \prod (1,2,4,9,12,13,15)$

هون طلب البرايم يعني نكتب عكس الارقام(يعني الارقام الي مش موجودة)

هون نفس الاشي بس بصيغة ثانية

Note

- The minterms of F are the maxterms for F'
- The maxterms of F are the minterms for F'

هون زي كأنه بحكيلي عكس العكس يعني الارقام الي موجودة نفسها لانه احنا عارفين ال POM هو عكس ال SOM وبرضه طلبه prime

- **Find a product of maxterms expression for**

$F(x, y, z) = \sum(1, 2, 3, 5, 7)$

$F = \Pi(0, 4, 6)$ $F = (x+y+z)(x'+y+z)(x'+y'+z)$

- **Find a sum of minterms expression for**

$F = \Pi\ (1, 3, 4, 6).$

$F(x, y, z) = \Sigma(0, 2, 5, 7) = x'y'z' + x'yz' + xy'z + xyz$

Express each of the following Boolean functions as a sum of minterms and as a product of maxterms :

1- $F(A, B, C) = A + B'C$

The function has three variables: A, B, and C

$A = A(B + B') = AB + AB'$

$A = AB(C + C') + AB'(C + C') = ABC + ABC' + AB'C + AB'C'$

$B'C = B'C(A + A') = AB'C + A'B'C$ Your paragraph text

the final answer => $ABC + ABC' + AB'C + AB'C' + A'B'C$

$m7 + m6 + m5 + m4 + m1$ ← هون حولنا

زي ما احنا ملاحظين اول حد ناقص متغيرين b,c لازم نضيفهم وثاني حد ناقص متغير A لازم نضيفه كمان عشان نعرف نحل ,وبعدين حولناه لعشري , كيف حولناه ؟ عنا ABC واحنا عافين بالباينري الارقام هيك بتكون 1 2 4 جمعناهم طلعو 7 واذا كان معه ` زي ABC` بطلع الجواب 6 لانه C` ما بنحسبها وهكذا

$F(A, B, C) = \Sigma(1, 4, 5, 6, 7) \rightarrow$ minterms

$F(A, B, C) = \Pi(0, 2, 3) \rightarrow$ maxterms

- $F(x, y, z) = xy + x'z$

$xy = xy(z+z') = xyz + xyz'$

$x'z = x'z(y+y') = x'yz + x'y'z$

Combining all the terms we finally obtain:

$F = x'y'z + x'yz + xyz' + xyz = m_1 + m_3 + m_6 + m_7$

$F(x, y, z) = \Sigma(1, 3, 6, 7)$

$F(x, y, z) = \Pi(0, 2, 4, 5)$

$$F(A,B,C,D) = \sum(A+B'+C)(B'+C+D)(A+C+D')$$

F= (A+B '+ C + (DD')) (B '+C+D+(AA')) (A+C+D'+(BB')) ظفنا المتغيرات الناقصة للحدود

F= (A+B'+C+D) (A+B'+C+D') (A+B'+C+D) (A'+B'+C+D) (A+B+C+D') (A+B'+C+D') شلنا الحدود المكررة

0100 1100 0001 0101

M4 M12 M1 M5

لأنه قاعدين بنحل عال maxterm ضروري كثير ننتبه لما بكون المتغير ما عليه ` بكون 0 ولما يكون عليه ` بكون 1 بالباينري

F (A, B, C, D) =Π{1,4,5,12}

F (A, B, C, D) = ∑{0,2,3,6,7,8,9,10,11,13,14,15}

Sum of Products

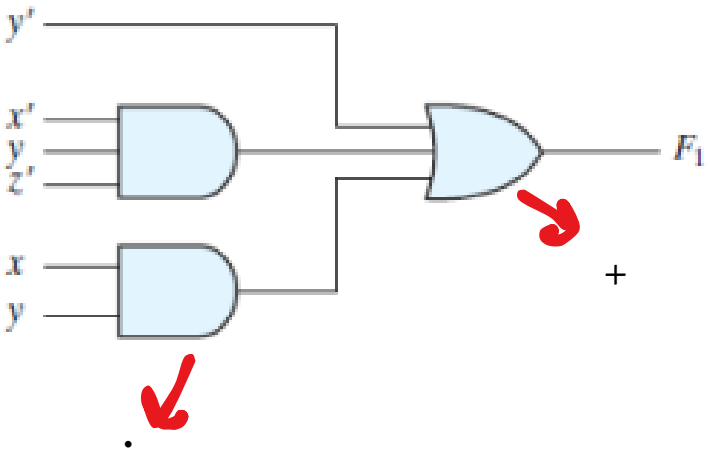
_ The sum of products is a Boolean expression containing AND terms, called product terms, with one or more literals each. The sum denotes the ORing of these terms.

_ An example of a function expressed as a sum of products is

F1 = y` + xy + x`yz`

_ This standard type of expression results in a two-level structure of gates

بين الحدود بتكون الاشارة (+) وبين المتغيرات نفسهم بتكون (.)



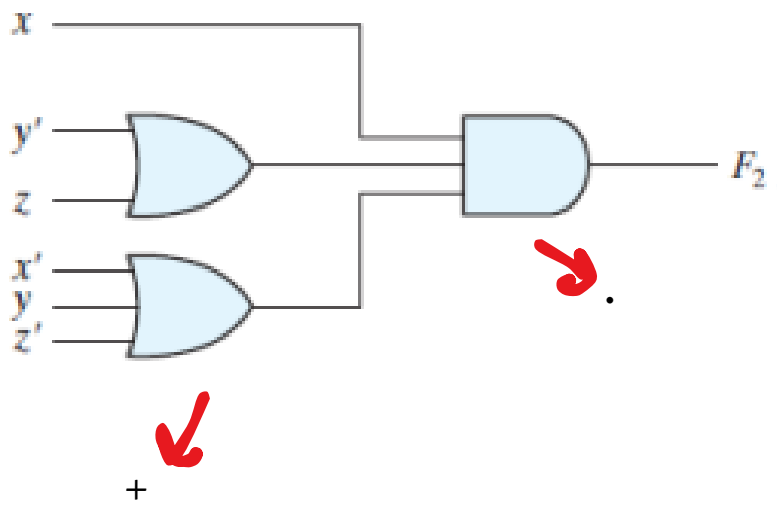
Product of Sums

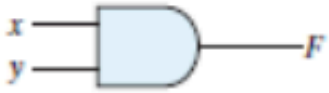
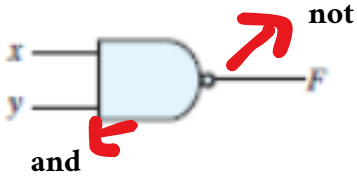
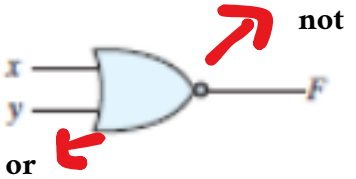
_ A product of sums is a Boolean expression containing OR terms, called sum terms. Each term may have any number of literals. The product denotes the ANDing of these terms

_ An example of a function expressed as a product of sums is F2 = $x(y`+ z)(x` + y + z `)$

_ This standard type of expression results in a two-level structure of gates

بين الحدود بتكون الاشارة (.) وبين المتغيرات نفسهم بتكون (+)



Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 1$
OR		$F = x + y$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 1$
Inverter		$F = x'$	$x \mid F$ $0 \mid 1$ $1 \mid 0$
Buffer		$F = x$	$x \mid F$ $0 \mid 0$ $1 \mid 1$
NAND		$F = (xy)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 0$
NOR		$F = (x + y)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 0$
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 0$
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 1$

CHAPTER THREE

GATE-LEVEL MINIMIZATION

K-map Method

_The map method provides a simple, straightforward procedure for minimizing Boolean functions.
_The map method is also known as the Karnaugh map or K-map method.
_ A K-map is a diagram made up of squares, with each square representing one minterm of the function that is to be minimized.
_The map presents a visual diagram of all possible ways a function may be expressed in standard form, from which the simplest can be selected

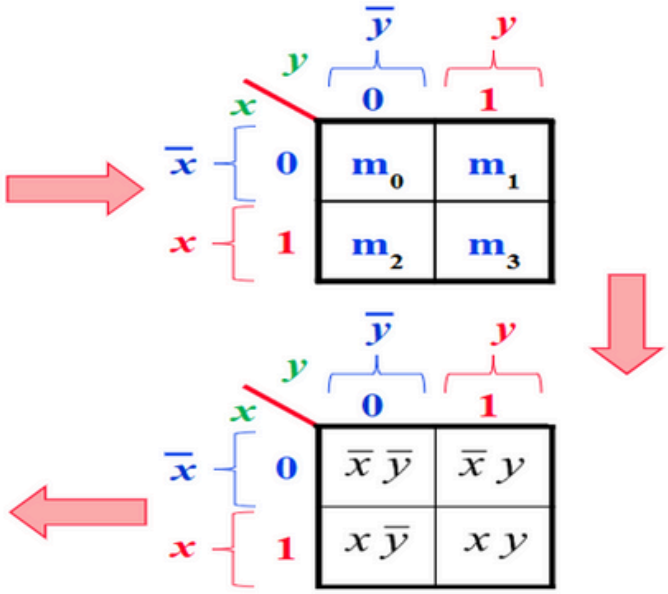
* الهدف من ال k-map نختار ابسط شكل ممكن لل function

* ال k-map عبارة عن مربعات كل مربع بمثل minterm يعني كل احتمال من الاجوبة الي رح تطلع معنا رح يكون اله مربع.

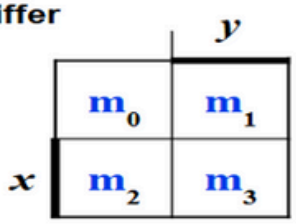
Two-Variable K-Map

هون عنا متغيرين x,y عملنالهم truth table

	x y	Minterm	
0	0 0	m_0	$\bar{x}\,\bar{y}$
1	0 1	m_1	$\bar{x}\,y$
2	1 0	m_2	$x\,\bar{y}$
3	1 1	m_3	$x\,y$



Note: minterms in adjacent squares differ in one variable!



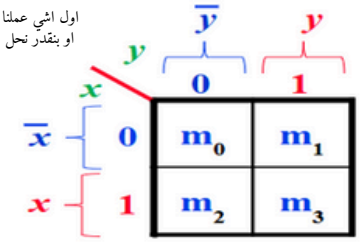
ال k-map مكونة من 4 مربعات لانه عنا متغيرين 2^n يعني $2^2 = 4$
بنوزع المتغيرات حسب الصفوف والاعمدة:
*الصفوف بتمثل ال x سواء كانت (x or `x) بنبلش من ال `x
*الاعمدة بتمثل ال y سواء كانت (y or `y) بنبلش من ال `y

Example 1

★ $F(x,y)= \sum (0,1)$

اول اشي عملنا truth table وطلعنا المتغيرات
او بتقدر تحل عطول وتحط محل الارقام الموجودة بالسؤال 1 جوا الجدول

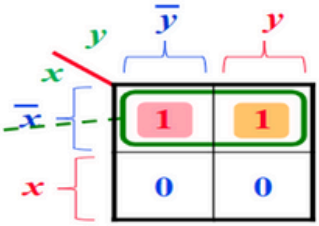
	x y	F	Minterm	
0	0 0	1	m_0	$\bar{x}\,\bar{y}$
1	0 1	1	m_1	$\bar{x}\,y$
2	1 0	0	m_2	$x\,\bar{y}$
3	1 1	0	m_3	$x\,y$



F(x,y) = x'y' + x'y

F(x,y) = x' (y' + y)

F(x,y) = x'

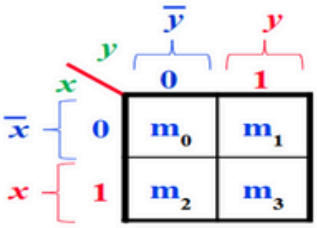


بنحط رقم 1 محل ما بتتقاطع عندي المتغيرات الي طلعناهم من ال truth table يعني اول اشي رقم 0 اعطانا x`y` , بعدين بنشوف شو المتغير المشترك بالسطر الي اخذناه بطلع x` وهو الجواب

Example 2

★ $F(x,y)= \sum (1,3)$

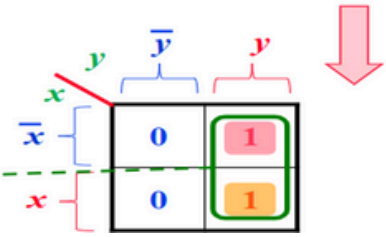
	x y	F	Minterm	
0	0 0	0	m_0	$\bar{x}\,\bar{y}$
1	0 1	1	m_1	$\bar{x}\,y$
2	1 0	0	m_2	$x\,\bar{y}$
3	1 1	1	m_3	$x\,y$



F(x,y) = x'y + xy

F(x,y) = y (x' + x)

F(x,y) = y

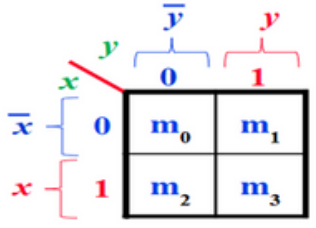


وهون نفس الاشي بس المتغير المشترك هو y

Example 3

★ $F(x,y)= \sum (1,2,3)$

	x y	F	Minterm	
0	0 0	0	m_0	$\bar{x}\,\bar{y}$
1	0 1	1	m_1	$\bar{x}\,y$
2	1 0	1	m_2	$x\,\bar{y}$
3	1 1	1	m_3	$x\,y$

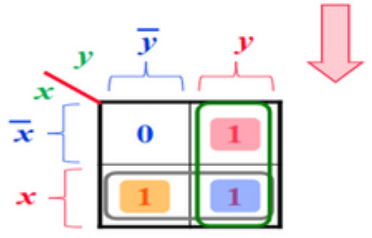


F(x,y) = x'y + xy'+ xy

F(x,y) = x'y + xy'+ xy + xy

F(x,y) = y (x'+x) + x (y'+ y)

F(x,y) = y + x

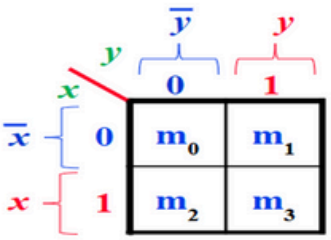


لازم ناخد كل ال 1 الي موجودة بالجدول , اخذنا اول اشي الي المربع الاخضر بعدين بضل عندي ال 1 بالاصفر فبناخدها مع الي جنبها عشان بدنا نلم اكبر عدد من ال 1 بنفس المربع عشان هدفنا التبسيط بعدين بنشوف شو المشترك بين كل مربع اخذناه لحال وبنطلع الجواب

Example 4

★ $F(x,y)= \sum (1,2)$

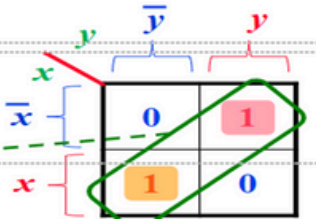
	x	y	F	Minterm	
0	0	0	0	m_0	$\bar{x}\,\bar{y}$
1	0	1	1	m_1	$\bar{x}\,y$
2	1	0	1	m_2	$x\,\bar{y}$
3	1	1	0	m_3	$x\,y$



F(x,y) = x'y + xy'

F(x,y) = x'y + xy'

??



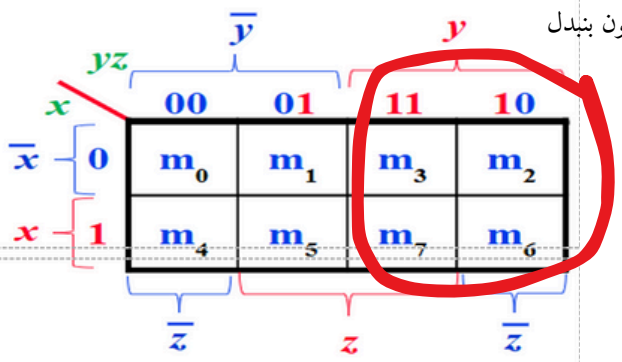
بهاد المثال ما عنا اشي مشترك يعني ما بنقدر نبسط اكثر من هيك وممنوع ناخذ ال 1 بشكل قطري

Three – Variable K-Map

هون بكون عندي 3 متغيرات x,y,z وعدد المربعات =8 لأنه $2^3 = 8$

	x	y	z	Minterm	
0	0	0	0	m_0	$\bar{x}\,\bar{y}\,\bar{z}$
1	0	0	1	m_1	$\bar{x}\,\bar{y}\,z$
2	0	1	0	m_2	$\bar{x}\,y\,\bar{z}$
3	0	1	1	m_3	$\bar{x}\,y\,z$
4	1	0	0	m_4	$x\,\bar{y}\,\bar{z}$
5	1	0	1	m_5	$x\,\bar{y}\,z$
6	1	1	0	m_6	$x\,y\,\bar{z}$
7	1	1	1	m_7	$x\,y\,z$

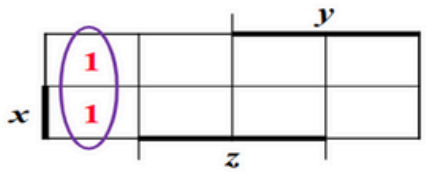
لازم ننتبه انه هون بنبدل



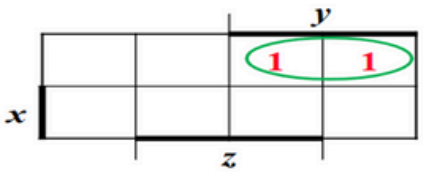
كيف طريقة الحل؟

احنا بنحاول بجمع اكبر عدد ممكن من الواحدات (1s)عشان هدفنا التبسيط فبنبلش نشوف اذا عندي ال 8 مربعات فيهم 1 بناخدهم مع بعض اذا ما في بننتقل للاصغر الي هو 4 بشوف اذا عندي 4 مربعات فيهم 1 سواء كانو بصف او بعامود بناخدهم واذا ما في بننتقل للأصغر 2 بشوف اذا عندي واحدتين بصف او بعامود وبرضه الاطراف بقدر اخدهم مع بعض وهسا رح نشوف امثلة :)

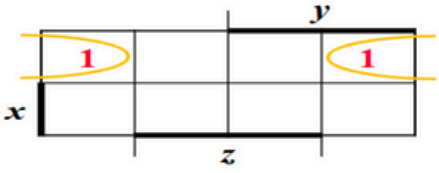
Groups of Two



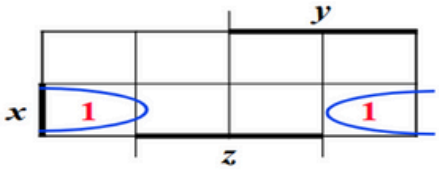
F = y'z'



F = x'y



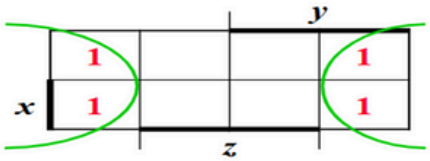
F = x'z'



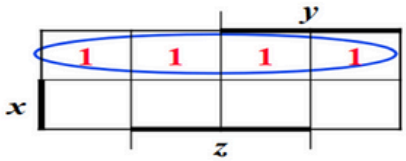
F = x z'

هدول أمثلة على كيف نختار كل 2 من الـ1s في الـk-map ونبسطهم لمعادلات أسهل

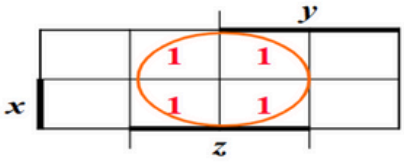
Groups of Four



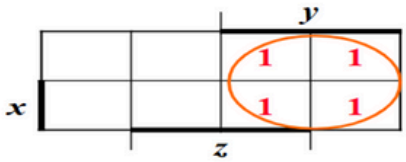
F = z'



F = x'



F = z



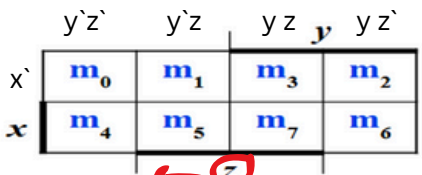
F = y

وهدول كيف نختار كل 4 من الـ1s في الـk-map ونبسطهم لمعادلات أسهل

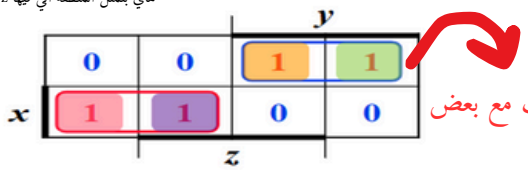
Example 1 :

f(x,y,z) = Σ(2,3,4,5)

	x	y	z	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0



هاي بتمثل المنطقة الي فيها z وال y نفس الاشي



بهاد المثال اخذنا الواحدات اللي بصف مع بعض

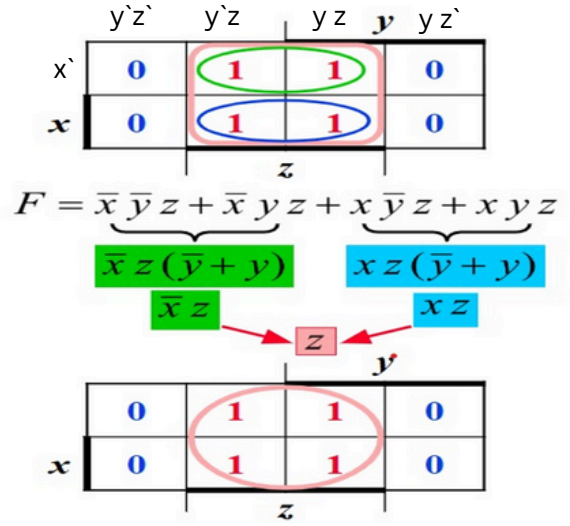
F(x,y,z) = x'yz' + x'yz + xy'z' + xy'z

F(x,y,z) = x'y(z +z') + xy'(z'+z)

F(x,y,z) = x'y + xy'

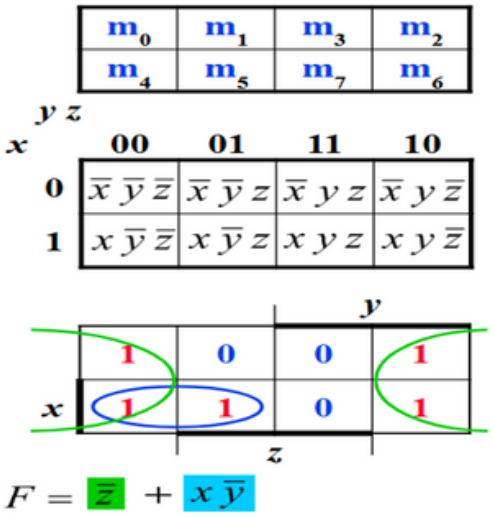
Example 2 : f(x,y,z) = Σ(1,3,5,7)

	x	y	z	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1



Example 3 : f(x,y,z) = Σ(0,2,4,5,6)

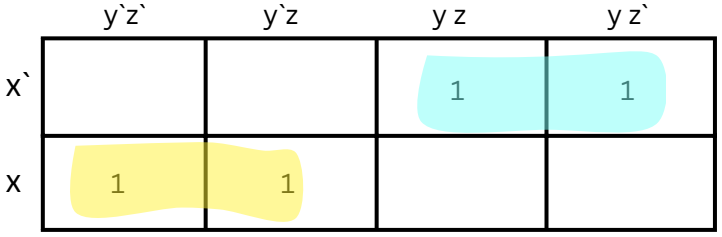
	x	y	z	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



Simplify each of the following Boolean functions into sum-of-products form.

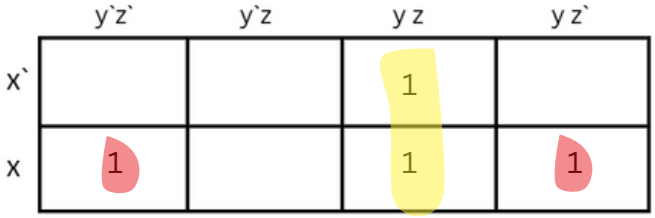
- $F = x'yz + x'yz' + xy'z' + xy'z$

F (x,y,z) = Σ (2,3,4,5)



الجواب : x`y + xy`

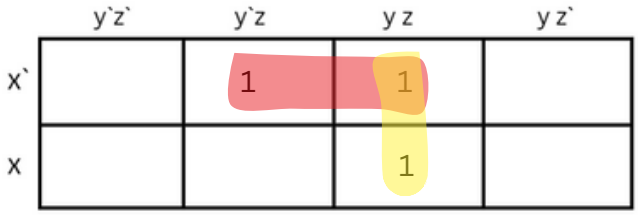
- $F(x, y, z) = \sum(3, 4, 6, 7)$



الجواب : yz + xz`

- $F(x, y, z) = \prod(0, 2, 4, 5, 6)$

F (x,y,z) = Σ (1, 3, 7)

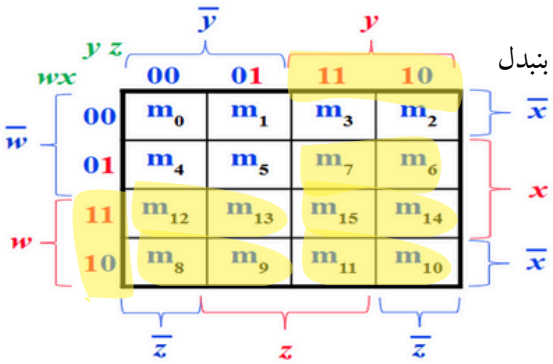


الجواب : x`z+ yz

Four – Variable K-Map

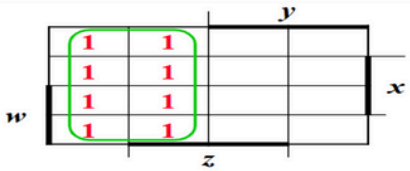
هون بكون عندي 4 متغيرات w,x,y,z وعدد المربعات = 16 لأنه $2^4 = 16$

	w x y z	Minterm	
0	0 0 0 0	m_0	$\bar{w}\bar{x}\bar{y}\bar{z}$
1	0 0 0 1	m_1	$\bar{w}\bar{x}\bar{y}z$
2	0 0 1 0	m_2	$\bar{w}\bar{x}y\bar{z}$
3	0 0 1 1	m_3	$\bar{w}\bar{x}yz$
4	0 1 0 0	m_4	$\bar{w}x\bar{y}\bar{z}$
5	0 1 0 1	m_5	$\bar{w}x\bar{y}z$
6	0 1 1 0	m_6	$\bar{w}xy\bar{z}$
7	0 1 1 1	m_7	$\bar{w}xyz$
8	1 0 0 0	m_8	$w\bar{x}\bar{y}\bar{z}$
9	1 0 0 1	m_9	$w\bar{x}\bar{y}z$
10	1 0 1 0	m_{10}	$w\bar{x}y\bar{z}$
11	1 0 1 1	m_{11}	$w\bar{x}yz$
12	1 1 0 0	m_{12}	$wx\bar{y}\bar{z}$
13	1 1 0 1	m_{13}	$wx\bar{y}z$
14	1 1 1 0	m_{14}	$wxy\bar{z}$
15	1 1 1 1	m_{15}	$wxyz$

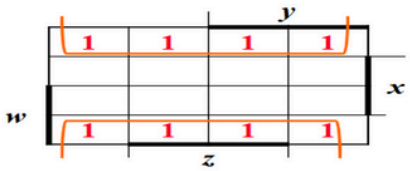


لازم ننتبه انه بنبدل

Groups of Eight



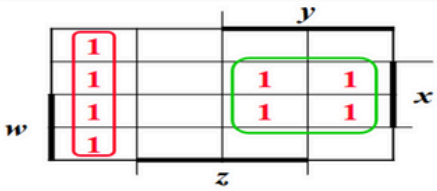
F = y'



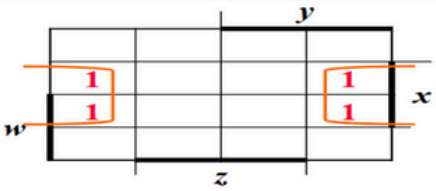
F = x'

هدول أمثلة على كيف نختار كل 8 من الـ1s في الـk-map ونبسطهم لمعادلات أسهل

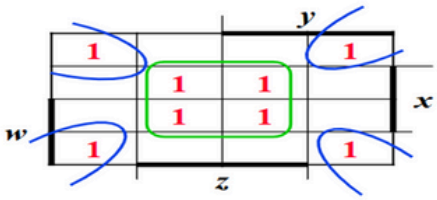
Groups of Four



F= y'z' + xy



F = xz'

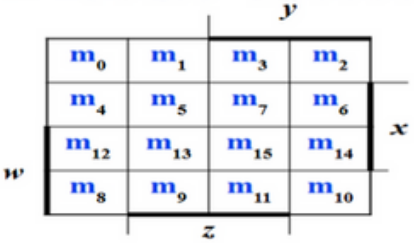


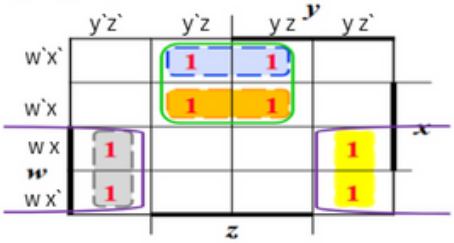
F = x'z' + xz

هدول أمثلة على كيف نختار كل 4 من الـ1s في الـk-map ونبسطهم لمعادلات أسهل

Example 1 :

★ $G(w,x,y,z) = \sum(1,3,5,7,8,10,12,14)$





F = w'x'y'z+w'x'yz+w'xy'z+w'xyz+wx'y'z'+wx'yz'+wxy'z'+wxyz'

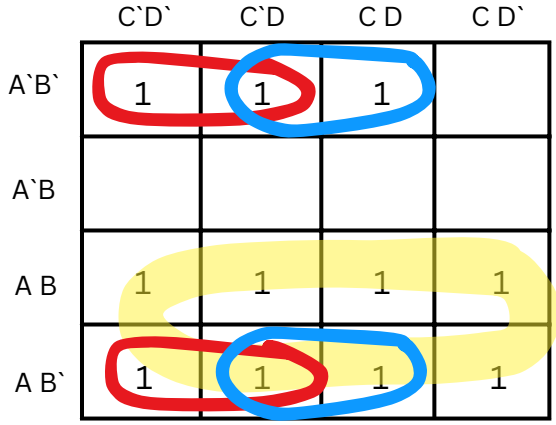
F = w'x'y'z+w'x'yz+w'xy'z+w'xyz+wx'y'z'+wx'yz'+wxy'z'+wxyz'

F = w'x'z + w'xz + wy'z'+ wyz'

F = w'z + wz'

Simplify each of the following Boolean functions into sum-of-products form.

- $F(A, B, C, D) = \sum(0, 1, 3, 8, 9, 10, 11, 12, 13, 14, 15)$

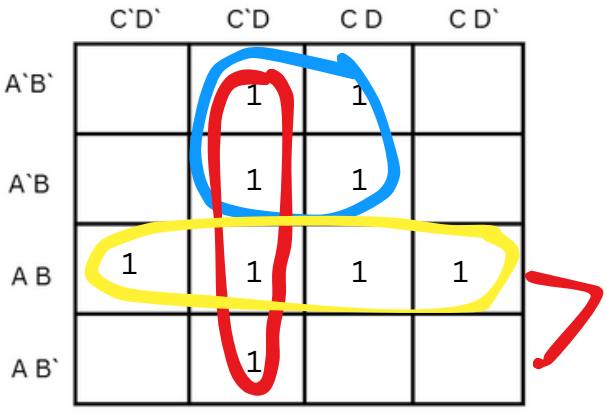


اول اشي اخذنا الـ8 الي بالاصفر بعدين اخذنا الـ 4 الي بالاحمر وبالازرق عشان نبسط قد ما بنقدر

بطلع الجواب : A + B'C' + B'D

- $F(A, B, C, D) = \prod(0, 2, 4, 6, 8, 10, 11)$

F(A, B, C, D) = Σ (1,3,5,7,9,12,13,14,15) بنحوله زي هيك عشان نسهل الحل



الجواب : AB + C`D + A`D

Don't-Care Conditions

_ In practice, in some applications the function is not specified for certain combinations of the variables

_ As an example, the four-bit binary code

_ A don't-care minterm is a combination of variables whose logical value is not specified.

_ The variable X is used to represent a don't care term.

شو يعني don't-care؟

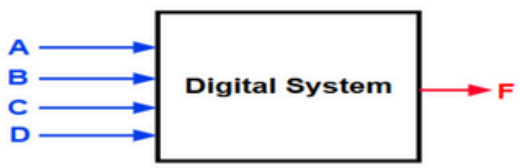
ما بتفرق معنا لو كانت النتيجة 1 أو 0، لأنه مش رح نستخدم هاي الحالة

وهذا الاشي مفيد في التبسيط باستخدام K-map

لأنه بنقدر نستغل الـ (don't-care) X لتكوين مجموعات أكبر وبالتالي تبسيط أفضل

Example 1 :

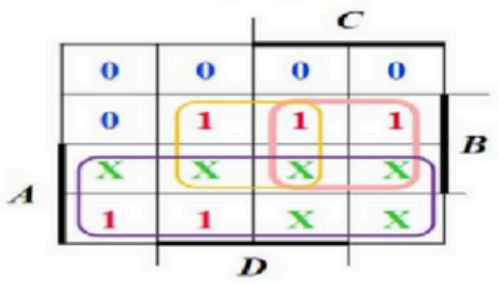
★ A digital circuit accepts 4 inputs; A, B, C and D, which represent a single BCD digit and outputs 1 only when the input value is greater than 4



A	B	C	D	F(A,B,C,D)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

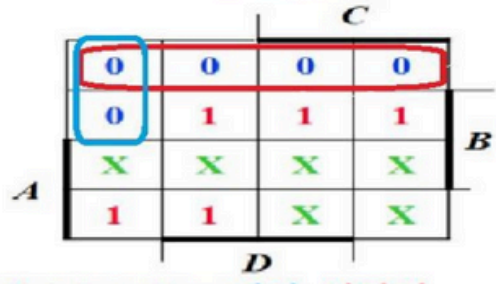
Example 1 continued :

SoP



F(A,B,C,D) = BC + BD + A

PoS

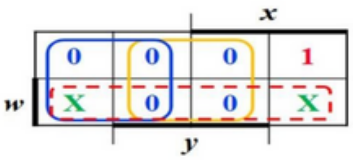


F'(A,B,C,D) = A`B`+A`B`D`

F(A,B,C,D) = (A+B)(A+B+C)

Example 2 :

★ **Simplify F(w,x,y) = ∏(0,1,3,5,7) + d(4,6) as POS**



F'(w,x,y) = x`+y SOP

F(w,x,y) = x.y` POS

بنمثّل الـ minterms (اللي قيمتهم 1) والـ don't-care terms (X) على خريطة كارنو (K-map)،
وبنبلش نرسم الدوائر (المجموعات) بهدف التبسيط.
الـ don't-care (X) بتساعدنا نعمل مجموعات أكبر وأسهل،
لأنه ممكن نعتبرها زي الـ 1 إذا بتفيدنا في التبسيط.
بس مش ضروري نستخدم كل الـ don't-care.
إحنا بنستخدمهم بس إذا بساعدوا بضم الـ 1s وتكبير الدوائر
الهدف النهائي هو تغطية كل الواحدات بمجموعات مبسطة بمساعدة الـ X إذا احتجناهم

Simplify each of the following Boolean functions into sum-of-products form.

- $F(A, B, C, D) = \sum(1, 3, 7, 11, 15)$, $d(A, B, C, D) = (0, 2, 5)$

رمز ال dont care

	C`D`	C`D	C D	C D`
A`B`	X	1	1	X
A`B		X	1	
A B			1	
A B`			1	

زي ما حكينا مش شرط نوخد كل ال x بس بنوخد الي بساعدوني اجمع اكبر عدد ممكن من ال 1 بنفس المربع

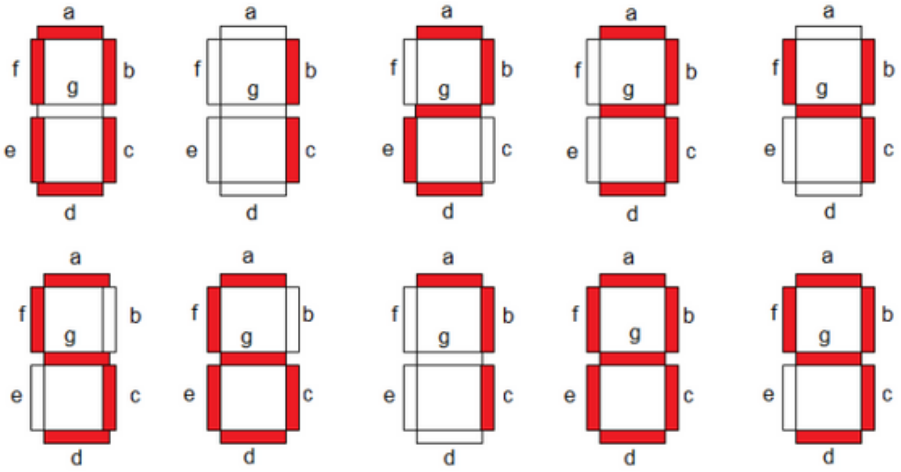
الجواب : A`B` + CD

- $F(A, B, C, D) = \sum(4, 5, 6, 7, 12)$, $d(A, B, C, D) = (0, 8, 13)$

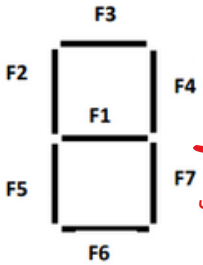
	C`D`	C`D	C D	C D`
A`B`	X			
A`B	1	1	1	1
A B	1	X		
A B`	X			

الجواب : C`D` + A`B

Logic circuit to represent numbers



إحنا بدنا نمثل الأرقام من 0 إلى 9 باستخدام دائرة منطقية
لتمثيل الأرقام من 0 إلى 9 باستخدام النظام الثنائي (binary) بنحتاج إلى 4 بتات (bits)
ليه؟ لأنه:
$2^4=16$ احتمالات (من0000إلى1111)
مع إنه رح نستخدم بس 10 احتمالات منهم (من 0000 لـ 1001) لأنهم بيمثلوا الأرقام من 0 إلى 9،
وبنخلي باقي الاحتمالات مهملة أو "don't care"



طريقة ترتيب المدخلات

عنا 3 خطوات للحل:
1- Truth table
2- k-map
3- draw the logic circuit

Step 1:

A	B	C	D	F1	F2	F3	F4	F5	F6	F7
0	0	0	0	0	1	1	1	1	1	1
0	0	0	1	0	0	0	1	0	0	1
0	0	1	0	1	0	1	1	1	1	0
0	0	1	1	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	0	0	1
0	1	0	1	1	1	1	0	0	1	1
0	1	1	0	1	1	1	0	1	1	1
0	1	1	1	0	0	1	1	0	0	1
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	0	1							
1	1	1	1							

زي ما شفنا بطريقة ترتيب المدخلات
عشان امثل الرقم صفر لازم اضوي
الكل واطفي ال F1

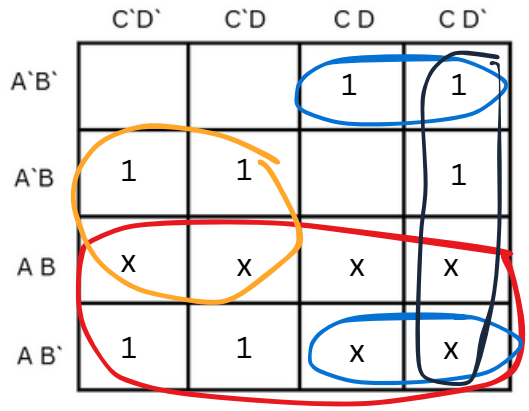
_ بعد ال 9 الارقام بتبطل مهمة عندي لانه اكبر عدد رح
تضويه الoutput هو 9

_ هسا رح نعمل k-map لكل عامود F عشان نقدر نرسم
الfunction الخاص فيه

Step 2 :

رح نعمل k-map لأول عامود F1

- **F1 (A,B,C,D) = Σ (2,3,4,5,6,8,9) d (10,11,12,13,14,15)**



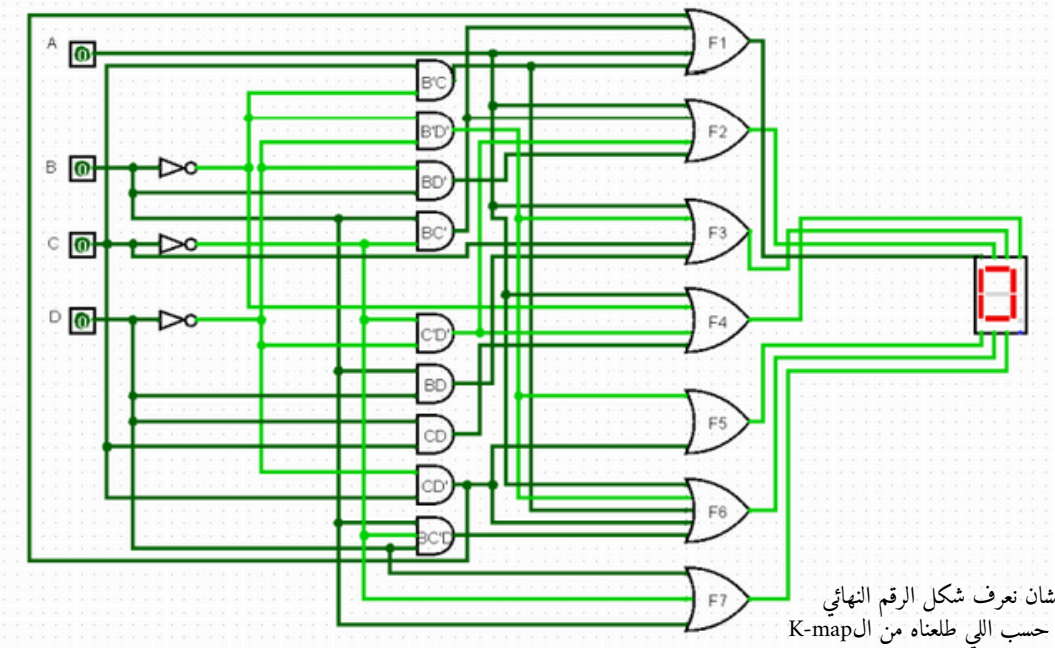
الجواب بطلع : A + BC`+ B`C+ CD`

كملو بنفس الطريقة على F2,F3,F4,F5,F6,F7 بطلع معكم الجواب :

F1 = A + CD′ + BC′ + B′C
F2 = A + BC′ + C′D′ + BD′
F3 = A + C + BD + B′D′
F4 = A + C′D′ + CD + B′
F5 = B′D′ + CD′
F6 = A + B′D′ + B′C + CD′ + BC′D
F7 = B + C′ + D

Step 3 :

بنرسم المعادلات كلهم وبنراعي الاشارة بين المتغيرات بتكون and (.) وبين الحدود بتكون or (+)



ملاحظات مهمة لما نرسم الدائرة :

_ بنبدأ من شكل الحرف (زي A) عشان نعرف شكل الرقم النهائي
_ عدد ال(inputs) لكل gate بكون حسب اللي طلعناه من الK-map
_ بنقدر نتحكم بالبوابة ونكبس عليها لو كنا بنستخدم برنامج Logisim
_ شكل ال(output) بوصل على 7Segment Display عشان يفرجينا الرقم الي طلع معنا
_ ال OR gate لما توصل أكثر من خط لل output خليه منظم:

- F1 أول وحدة من اليسار
- آخر سلك يوصل فيها لل output النهائي

_ الشكل ممكن يختلف من شخص لثاني بس المهم إنه الرقم يطلع صح من 0 لـ 9

CHAPTER FOUR

COMBINATIONAL_LOGIC

Chapter 4

Introduction

Logic circuits for digital systems may be **combinational or sequential.**

A logic circuit is combinational if its outputs at any time are a function of only the present inputs.

A logic circuit is sequential if the outputs are a function of the inputs and the state of the storage elements.

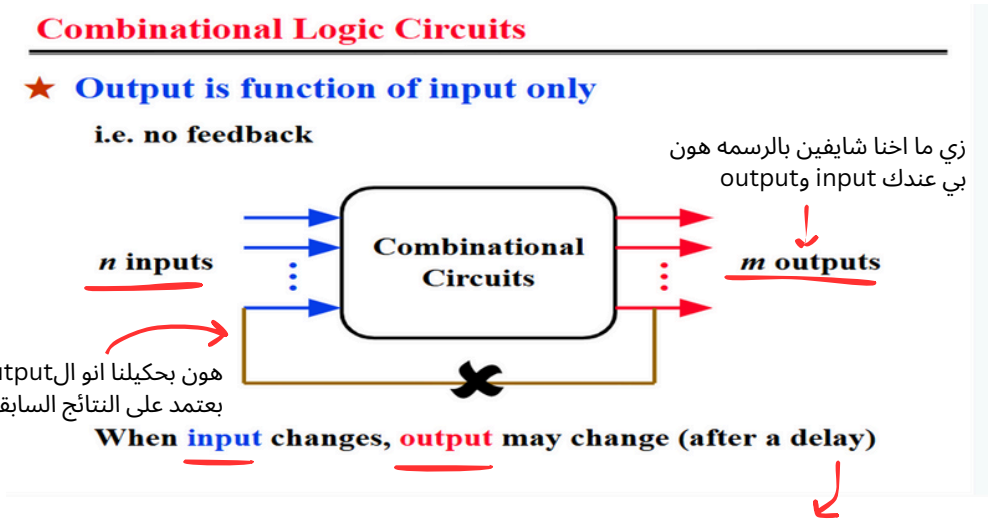
The map presents a visual diagram of all possible ways a function may be expressed in standard form, from which the simplest can be selected.

Combinational circuit and **sequential** circuit عنا نوعين من الدوائر المنطقية

ورح ندرس في هاظ الشابتر ال **Combinational circuit**

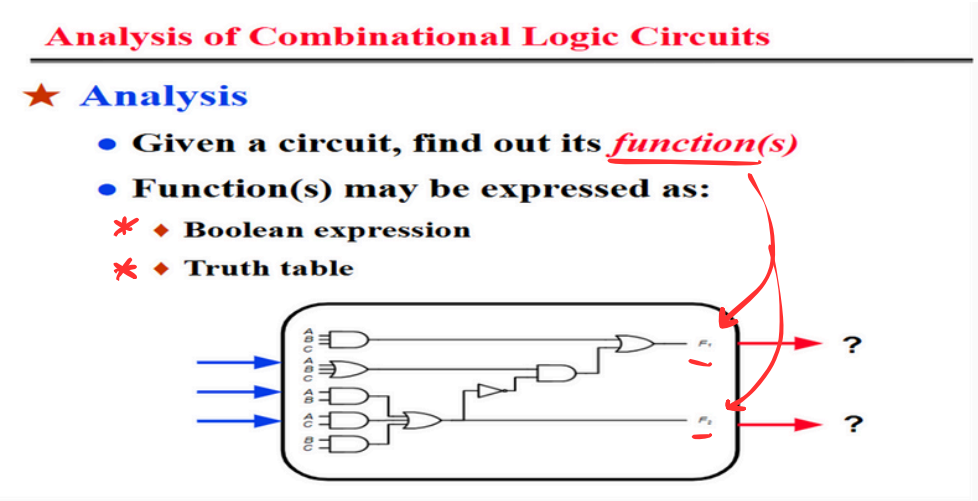
Combinational circuit

هي دوائر منطقية، مخرجاتها بتعتمد فقط على المدخلات الحالية. يعني ما في "ذاكرة" للأشياء اللي صارت قبل. زي لما بتكبس على كبسة الضوء: الضوء بضوي أو بطفي بناءً على كبستك الحالية، مش على كم مرة كبست قبل.



رح نحكي بالبداية عن موضوع ال **Analysis**

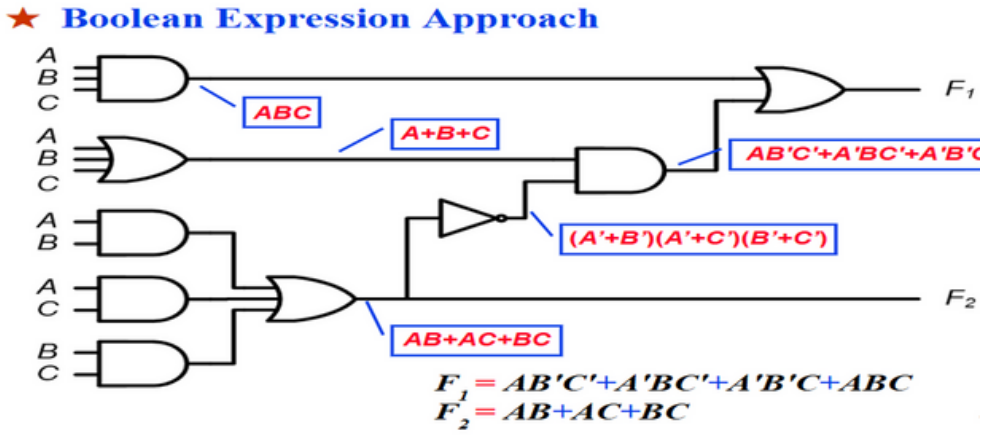
اخذنا بالشباتر السابقة موضوع ال Truth table وال k-map وتعمقنا فيهم وهسا في موضوع ال Analysis
رح نحتاجهم حتى نقدر نحل



بالمثال فوق بحكيلنا نعمل Analysis لهاي ال circuts
عنا طريقتين زي ما هو موضح بالمثال

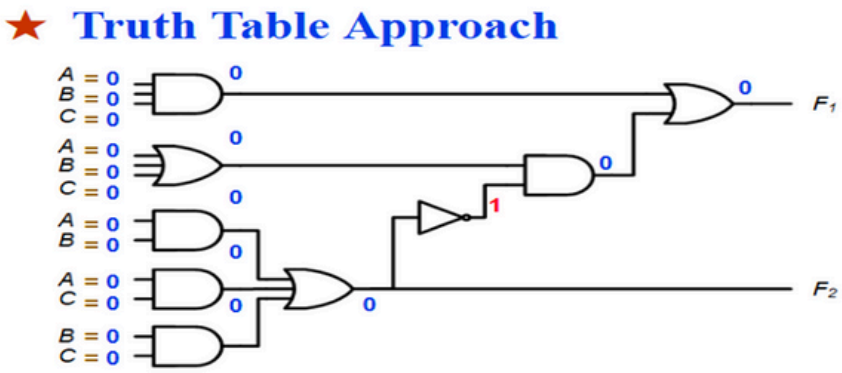
1- Bollean expression (اسهل واسرع)
2- Truth table

1-Bollean expression



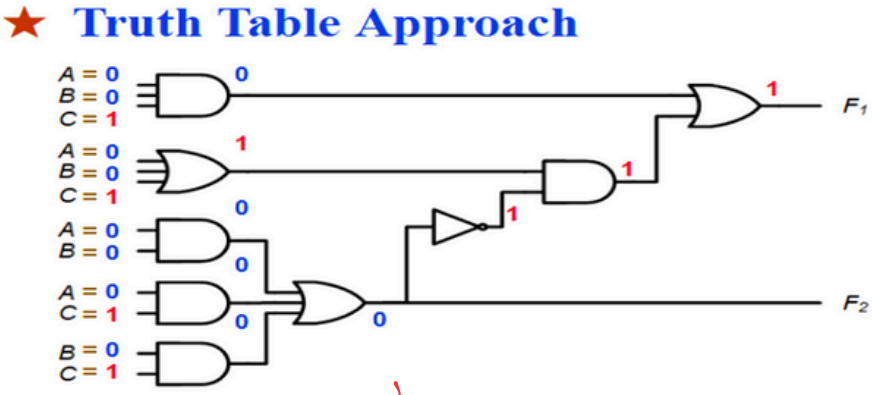
هون عنا الطريقة الاولى بنمشي على الرسمة خطوة خطوة وال output الي بطلع معنا هو الجواب

2- Truth table

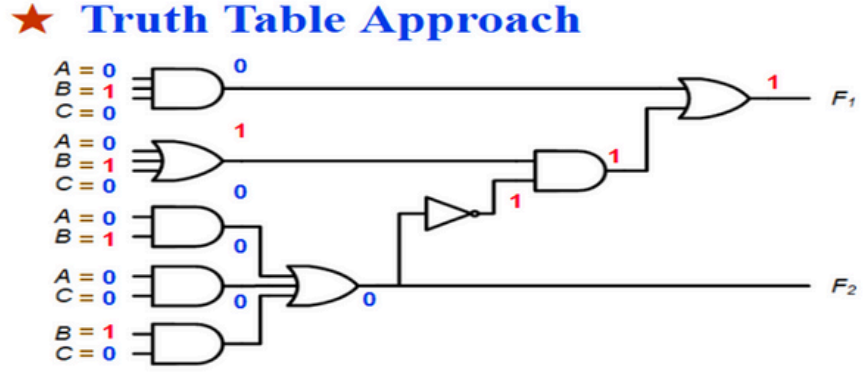


A	B	C	F_1	F_2
0	0	0	0	0

هون الطريقة الثانية بنمشي على الرسمة وبنعوض فيها قيم ال Truth table
يعني بنعبي ال Truth table كامل بعدها بنطلع ال Functions عن طريق ال K-map

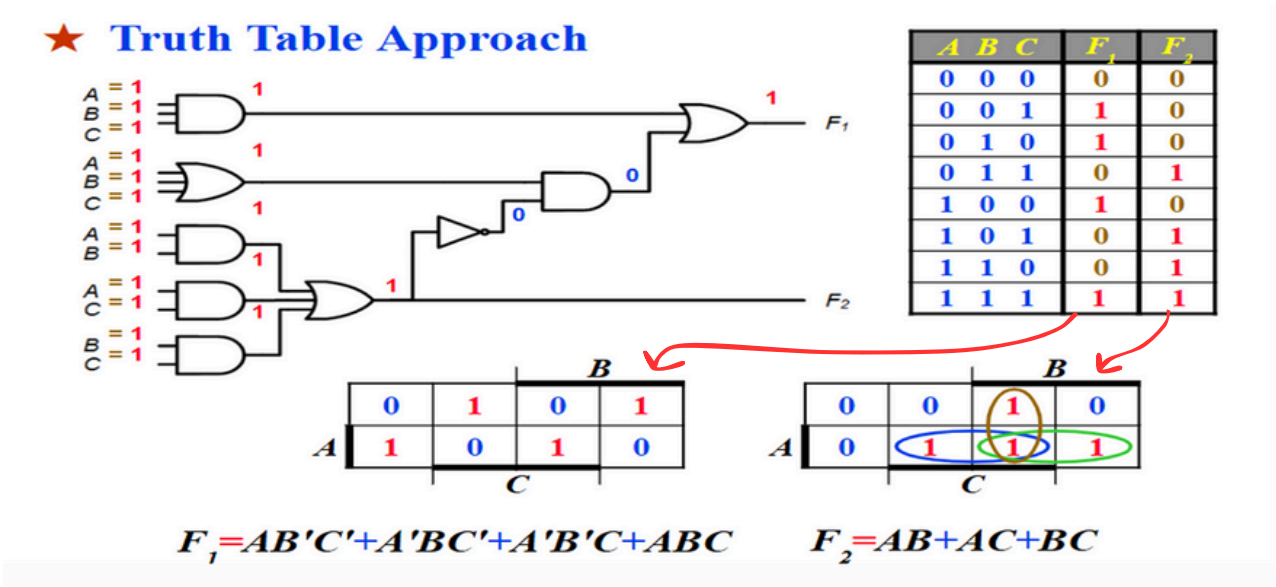


A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0



A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0

بنكمل مع جميع الاحتمالات لل Truth table لحد ما نخلص كامل الاحتمالات

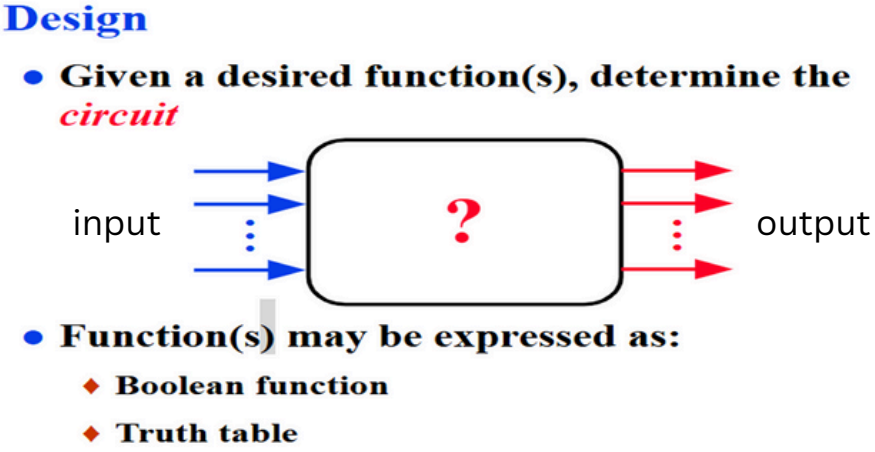


بالاخر بنطلع الحل عن طريق اخذ ال K-map للل Functions وهيك بنكون عملنا Analysis للل circuit عن طريق ال Truth table Approach

وهيك بنكون خلصنا موضوع ال Analysis وهسا رح ننتقل للل Design

Design of Combinational circuit

هسا رح نحكي عن موضوع ال Design وهاظ الموضوع جدا مهم وكثير سهل وإذا فهمتوا موضوع ال Analysis فرح يكون ال Design جدا سهل عشان الموضوعين ببساطة عكس بعض



هسا هاي الرسمة بتشبه رسمة ال Analysis لكن زي ما قلنا فوق هي عكسها عندك input وعندك output لكن الفرق هون انك ما بدك تطلع المعادله لا انت هون بدك تطلع ال logic diagram

Ex1:

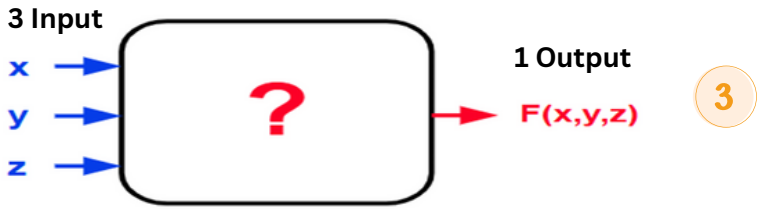
Design a logic circuit that has 3 inputs; **x, y, and z**, and one output; **F(x,y,z)**. The output is 1 if the number of 1s in the input is greater than the number of 0s. Otherwise, the output is zero. **(Majority Circuit)**

خطوات الحل:

1. **نحدد ال input**
2. **نحدد ال output**
3. **نحدد ال Function**
4. **نرسم ال Truth table**
5. **بعدها بنرسم ال logic diagram بناء على ال K-map**

Number of inputs = 3 → x, y, z (1)

Number of outputs = 1 → F(x,y,z) (2)



شو هو ال Function؟

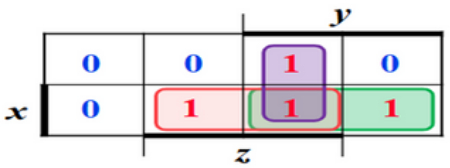
بقلك بكل بساطة لو كان عدد ال 1 اكبر من عدد ال 0 حط الجواب 1.... والعكس صحيح

	x	y	z	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

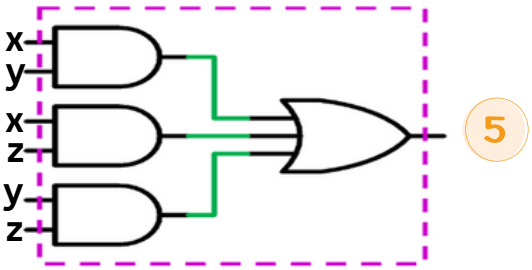
هون كان عدد ال 1 اكثر عشان هيك الجواب (1)

4

K-map



$F(x,y,z) = xy + xz + yz$



Ex2:

Design a logic circuit that compares **two 2-bit numbers** ; A and B, and outputs two signals; O_1 and O_0 such that:

$$O_1O_0 = \begin{cases} 00, A==B \text{ and both are even} \\ 01, A < \\ 10, A > \\ 11, A==B \text{ and both are odd} \end{cases}$$

3

هاظ هو ال Function

solution:

1. Number of inputs = 4 → A_1, A_0, B_1, B_0
2. Number of outputs = 2 → O_1 and O_0

A_1	A_0	B_1	B_0	O_1	O_0
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	0	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	1

نفس الرقم لكن مقسم الى 2-Bits

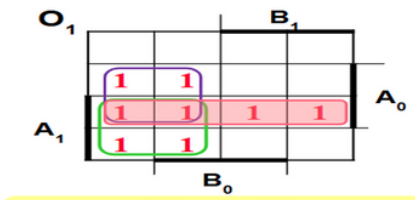
ليش في عنا A_1, A_0, B_1, B_0 من وين اجو هظول؟
من كلمة two 2-bit numbers
يعني لما يحكيلي bit-2 بقسم كل حرف ل 2Bits
كذلك لو قال 3Bits برضو بقسم الحرف ل 3bits

4

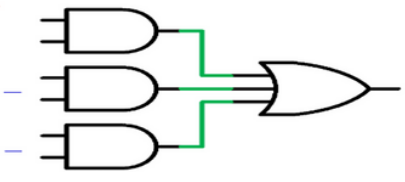
$$O_1O_0 = \begin{cases} 00, A==B \text{ and both are even} \\ 01, A < \\ 10, A > \\ 11, A==B \text{ and both are odd} \end{cases}$$

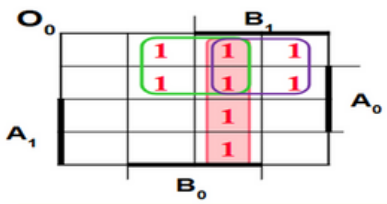
بنشتغل على هاظ الجدول

بنعمل K-map لكل output عشان نطلع ال logic diagram الهم

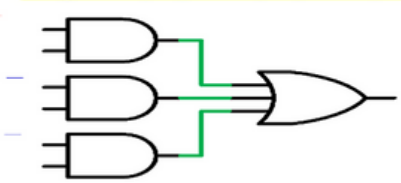


$$O_1 = A_1\overline{B_1} + A_0\overline{B_1} + A_1A_0$$

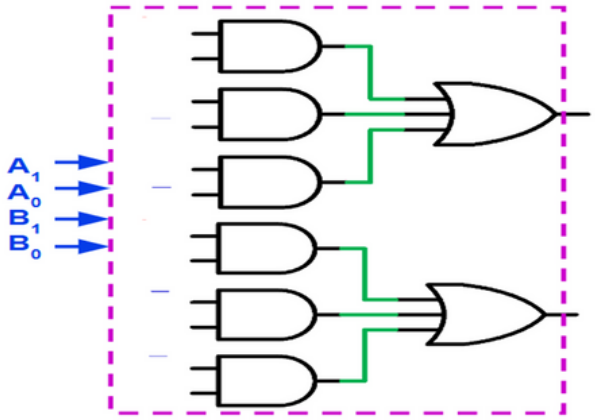




$$O_0 = \overline{A_1}B_0 + \overline{A_1}B_1 + B_1B_0$$



5



وهيك بكون الشكل النهائي لل logic diagram
بعد دمجهم

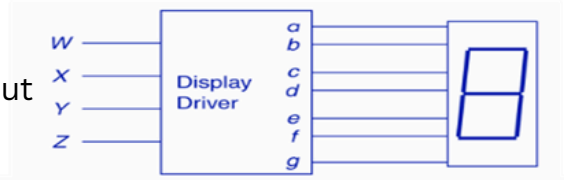
Ex3:

Design a BCD to 7-segment decoder.

What is a 7-Segment Display?

كثيييييير مهم
حفظ مع الترتيب

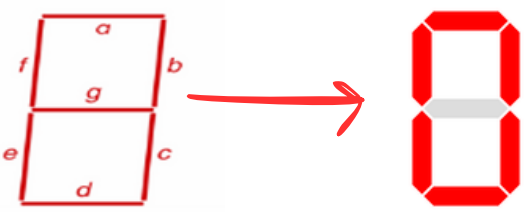
4 Input



7 Output

قبل ما نبدأ بحل المثال
شو هو ال BCD ؟ وشو هو ال 7-segment Display ؟
ال BCD هي طريقة تستخدم عشان نمثل كل رقم عشري(0,9) الى بتات Binary
اما بالنسبة ل 7-segment Display ؟ فهي شاشة مكونة من 7 أجزاء (مثل LEDs) يمكن إضاءة كل جزء منها بشكل مستقل لتشكيل الأرقام.

بتوخذ هاي الشاشة 4 Input وبتطلع 7 Output



مثلا هون لو ضوينا a,b,c,d,e,f
رح يطلع معنا رقم 0

Digit	W	X	Y	Z	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	X	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	X	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	X	0	1	1
–	1	0	1	0	X	X	X	X	X	X	X
–	1	0	1	1	X	X	X	X	X	X	X
–	1	1	0	0	X	X	X	X	X	X	X
–	1	1	0	1	X	X	X	X	X	X	X
–	1	1	1	0	X	X	X	X	X	X	X
–	1	1	1	1	X	X	X	X	X	X	X

Don't care are marked with X

عشان الشاشه ما بظهر فيها ارقام اكبر من 9

هون رح نشتغل نفس الامثلة فوق بنس الخطوات ورح نعمل K-map لكل output ...

a=W+Y+XZ+X`Z`
b=Y`+Z`X`
c=Y+X`+Z
.....

بنكمل باقي ال Output وبعدها بنرسم ال logic diagram مثل ما تعودنا.

Ex4:

Design a logic circuit that converts a single BCD digit into Excess-3 code

Single BCD digit → values [0,9] → 4 bits
Number of inputs = 4 → A, B, C, D

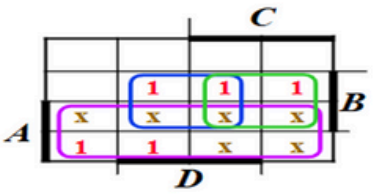
Excess-3 = BCD + 3 → Values [3,12] → 4 bits
Number of outputs = 4 → w, x, y, z

كل الي رح نعمله هو انو رح نزيد رينج الارقام بمقدار 3
يعني كل رقم رح يزداد بمقدار 3 على قيمتة الاصلية... رقم 2 مثلا رح يصير 5
ورقم 6 رح يصير 9 وهكذا

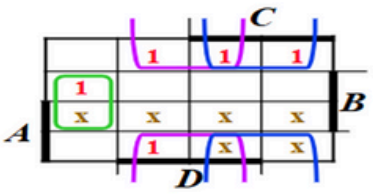
BCD-to-Excess3 Converter

Excess-3 = BCD + 3

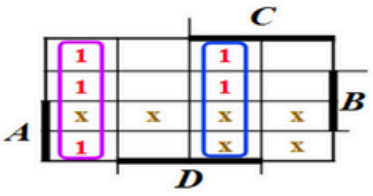
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x



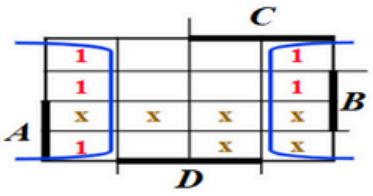
$w = A+BC+BD$



$x = B'C+B'D+BC'D'$

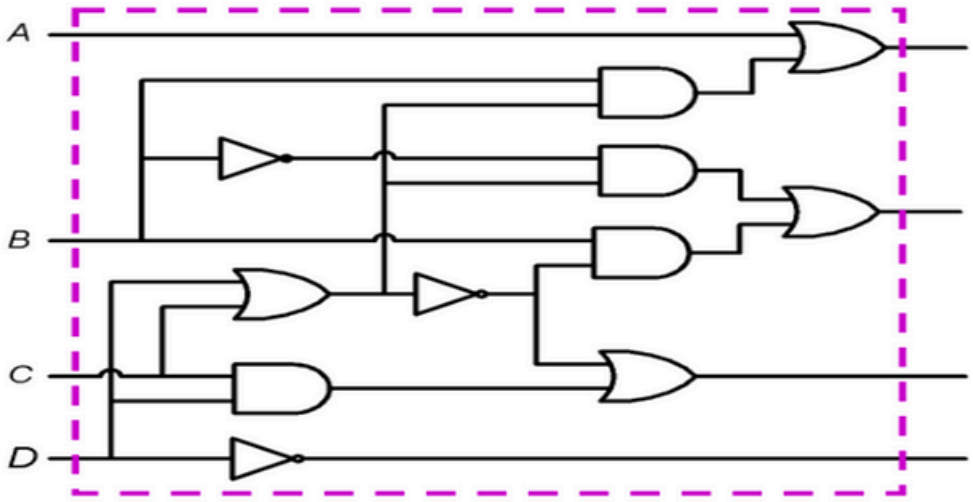


$y = C'D'+CD$



$z = D'$

هون عملنا الTruth table بناء على الي شرحناه فوق وبعدها رسمنا ال K-map



$w = A + B(C+D)$

$y = C'D' + CD$

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

$z = D'$

Half Adder

Used for the addition of just two binary inputs, the half adder outputs both the **Sum** of those bits and **Carry** bit

1 2

Ex:

Carry Sum

x y	c s
0 0	0 0
0 1	0 1
1 0	0 1
1 1	1 0

K-map

$$\begin{array}{r} x \\ + \quad y \\ \hline C \quad S \end{array}$$

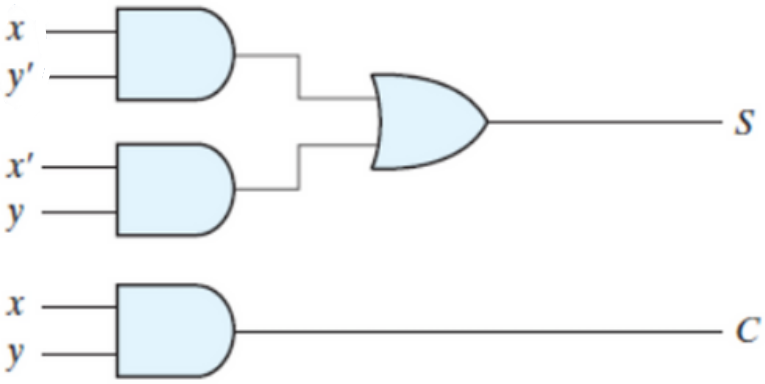
Half Adder

x' \ y' y	y'	y
x'		
x		1

C = xy

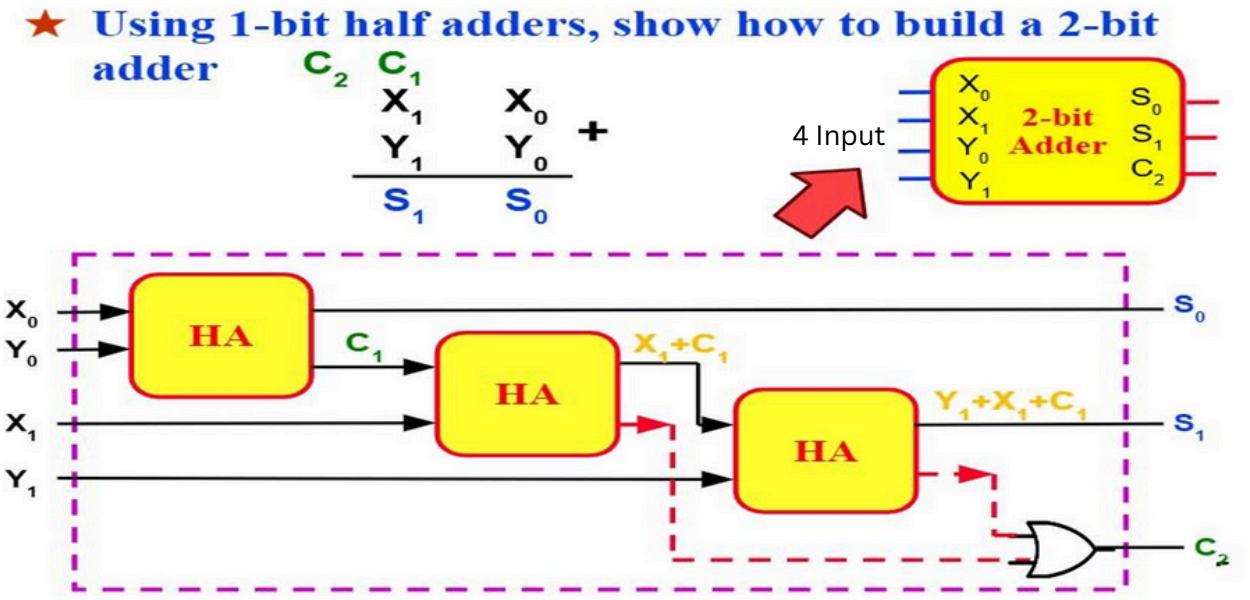
x' \ y' y	y'	y
x'		1
x	1	

S = xy' + x'y



$S = xy' + x'y$
$C = xy$

كلنا بنعرف انو ال Half Adder ما بتوخذ الا Tow Input فكيف بدنا نخليها توخذ اكثر؟ او كيف بدنا نخليها توخذ four Input؟ عن طريق بناء هاي ال circuit



خلينا هسا نشرح شو هاي؟

هسا مبدأيا بدنا نعمل circuit توخذ four input وتطلع three output . بس كيف؟

اول اشي جبنا Half Adder دخلنا فيه X_0 و Y_0 وجمعناهم. هسا هاي ال HA رح تطلعلنا two output

اول output هو S_0 وهاظ بدنا اياه, والثاني هو C_1,هسا C_1 رح تدخل معنا برضو ك input مع X_1 للHA

الثاني وناتج هاي ال HA رح يكون $X_1 + C_1$ وبرضو هاظ الناتج رح يكون input للHA الثالثة مع Y_1

وبالنهاية ناتج ال HA الاخيرة رح يكون $Y_1 + X_1 + C_1$ والي هو S_1 وبرضو هاي بدنا اياها

بضل عنا C_2 وهاي زي ما هو موضح بالرسمة فوق رح تنتج من ال output تبع ال HA الثانية والثالثة

(مش دايما بكون عنا C_2)

Full Adder

It use when we have 3 binary inputs and we want to add them together which results two outputs.
where the third input represents the **carry** from the previous lower significant position.

Ex:

3 input

Full Adder

- Adds 1-bit plus 1-bit plus 1-bit
- Produces Sum and Carry

$$\begin{array}{r} x \\ + \; y \\ + \; C_{in} \\ \hline C_{out} \; S \end{array}$$

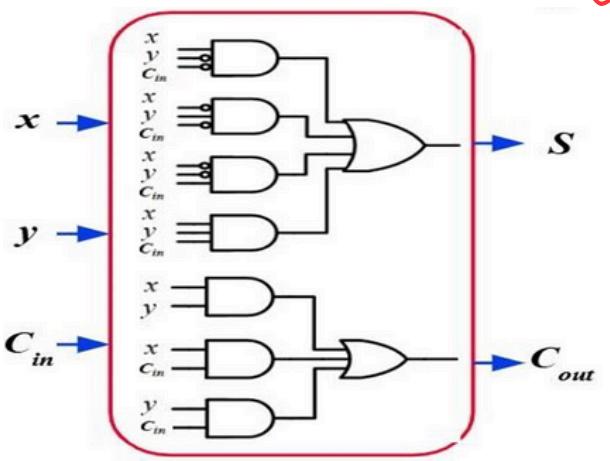
x	y	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

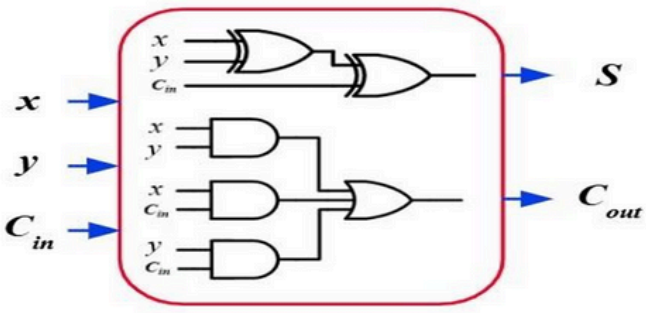
x \ y, C_{in}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$S = x'y'C_{in} + x'yC_{in}' + xy'C_{in}' + xyC_{in} = x \oplus y \oplus C_{in}$$

x \ y, C_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$C_{out} = xy + xC_{in} + yC_{in}$$





سوف يتم شرح موضوع ال **Adder-Subtractor** في مقطع فيديو لتسهيل شرحه.

CHAPTER FIVE

SEQUENTIAL LOGIC

chapter 5

Sequential Circuits

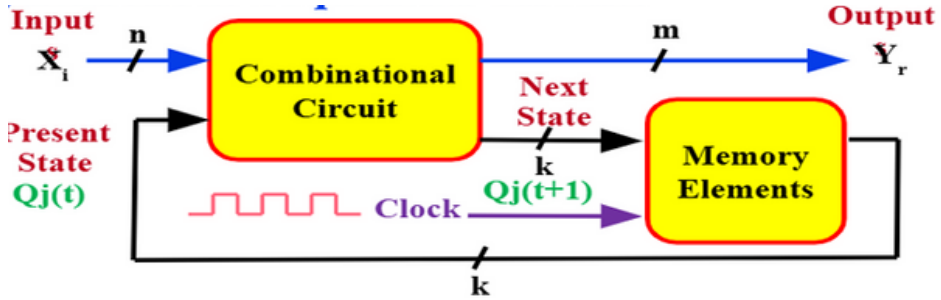
Sequential Logic Circuit

- **A logic circuit is sequential (Asynchronous)if the output depends on both inputs and the state (memory).=LATCHES**
- **A synchronous sequential circuit updates the state only at specific times, controlled by a clock signal.=FLIP-FLOPS**

دائرة المنطق التسلسلي

- الدائرة المنطقية تُعتبر تسلسلية إذا كانت المخرجات تعتمد على المدخلات والحالة (الذاكرة).
- الدائرة التسلسلية المتزامنة تحدث بها التغييرات فقط في لحظات محددة يتم تحديدها بواسطة إشارة الساعة (Clock).

synchronous Sequential Circuits:



الفرق الأساسي

كل الدوائر المتزامنة هي تسلسلية، لكن ليس كل الدوائر التسلسلية متزامنة. الفرق في توقيت تغيير الحالة:

- في المتزامنة: التغيير مضبوط بالساعة
- في غير المتزامنة (Asynchronous): التغيير يحدث مباشرة عند تغير المدخلات.

كيف تعمل الدائرة:

في لحظة زمنية t:

الدخل Xi والحالة الحالية Qj(t) تدخل إلى الدائرة التوافقية.

يتم حساب الخرج Yr والحالة التالية Qj(t+1).

عند نبضة الساعة القادمة:

يتم تخزين Qj(t+1) في الذاكرة لتصبح الحالة الجديدة.

خلاصه

في الدوائر التسلسلية المتزامنة، يتم التحكم في التغيرات بالحالة (الانتقال من الحالة الحالية إلى التالية) بواسطة نبضات الساعة، بحيث لا يحدث أي تغيير إلا في لحظة محددة زمنياً.

تعتمد المخرجات على كل من المدخلات الحالية والحالة السابقة المخزنة في عناصر الذاكرة، مما يمنح هذه الدوائر القدرة على "تذكر" التسلسل الزمني للأحداث، وهذا ما يميزها عن الدوائر التوافقية.

Storage Element

A storage element is a logic circuit that stores one bit of data as long as power is available.

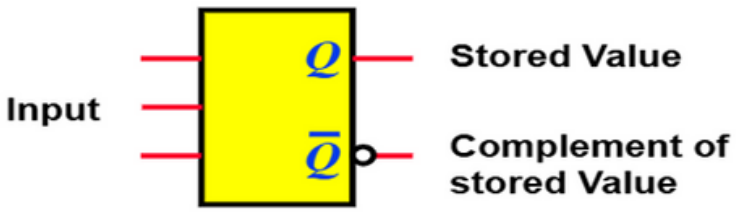
.The input determines how and when the stored value changes

.The output represents the stored value (Q) and its complement ($\bar{Q}$)

العنصر التخزيني هو دارة منطقية تُستخدم لتخزين 1 بت من البيانات طالما أن الجهاز موصول بالطاقة.

يتم استخدام المدخل لتحديد كيفية ومتى تتغير القيمة المخزنة.

أما المخرجات، فهي تمثل القيمة المخزنة نفسها (Q) والمتممة لها ($\bar{Q}$).



Types of storage Element

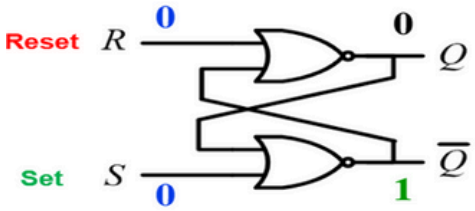
1. latches
2. flip-flops
3. controlled/clocked latches

Operations performed by the input could be:
Set (change stored value to 1) •
Reset (change stored value to 0) •
Hold (no change) •
Toggle (complement the stored value) •

latches

Latches

★ SR Latch



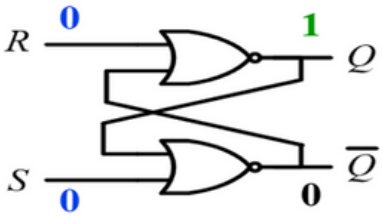
S	R	Q	Q+	Q+'
0	0	0	0	1

$Q+ = Q$

Initial Value

Q is Q(t); the present state
Q+ is Q(t+1); the next state

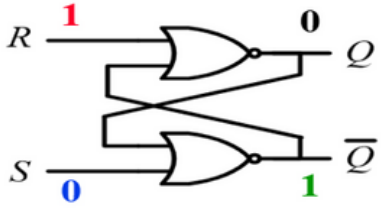
★ SR Latch



S	R	Q	Q+	Q+'
0	0	0	0	1
0	0	1	1	0

$Q+ = Q$
$Q+ = Q$

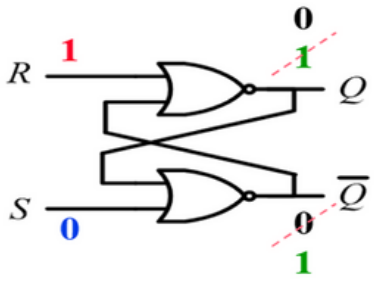
★ SR Latch



S	R	Q	Q+	Q+'
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1

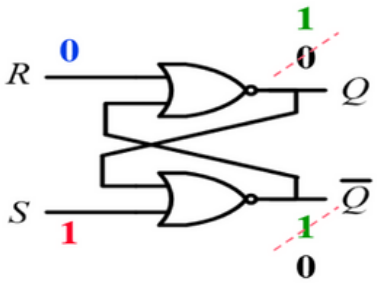
$Q+ = Q$
$Q+ = 0$

★ SR Latch



S	R	Q	Q+	Q+'	
0	0	0	0	1	Q+ = Q
0	0	1	1	0	
0	1	0	0	1	Q+ = 0
0	1	1	0	1	Q+ = 0

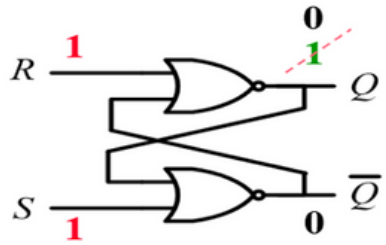
★ SR Latch



S	R	Q	Q+	Q+'	
0	0	0	0	1	Q+ = Q
0	0	1	1	0	
0	1	0	0	1	Q+ = 0
0	1	1	0	1	
1	0	0	1	0	Q+ = 1

اختصار خطوات....

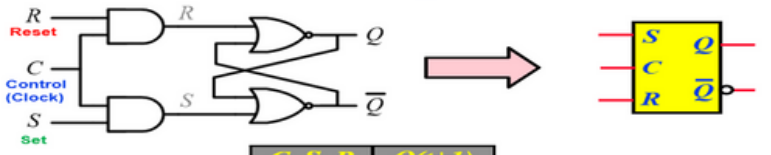
★ SR Latch



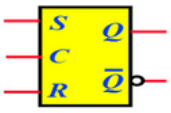
S	R	Q	Q+	Q+'	
0	0	0	0	1	Q+ = Q
0	0	1	1	0	
0	1	0	0	1	Q+ = 0
0	1	1	0	1	
1	0	0	1	0	Q+ = 1
1	0	1	1	0	
1	1	0	0	0	Q+ = Q+'
1	1	1	0	0	Q+ = Q+'

controlled/clocked latches

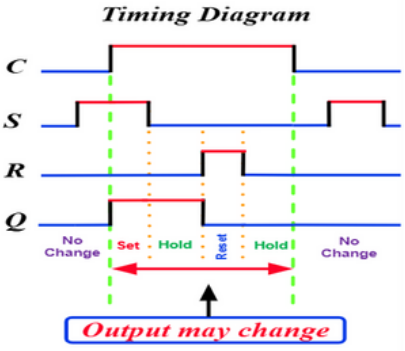
★ SR Latch with Control Input



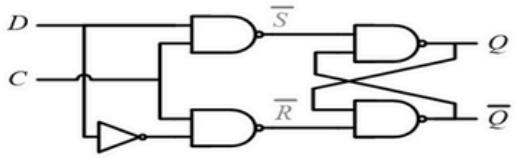
C	S	R	Q(t+1)	
0	x	x	Q(t)	No change
1	0	0	Q(t)	No change
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	Q=Q'	Invalid

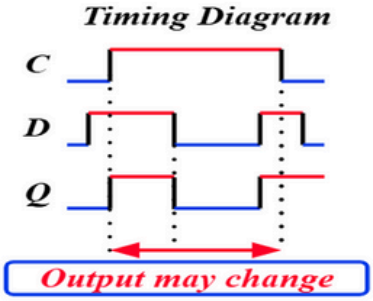


C	S	R	Q(t+1)	
0	x	x	Q(t)	No change
1	0	0	Q(t)	Hold
1	0	1	0	Reset
1	1	0	1	Set
1	1	1	Q=Q'	Invalid

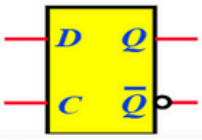


★ D Latch (D = Data)

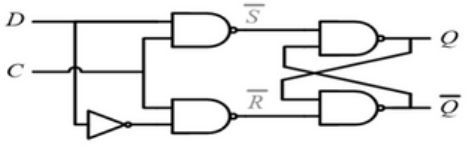


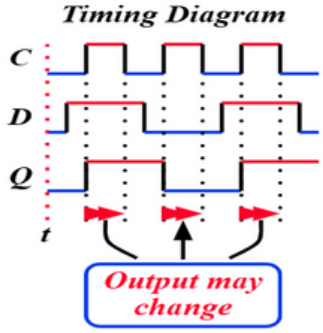


C	D	Q(t+1)	
0	x	Q(t)	No change
1	0	0	Reset
1	1	1	Set



★ D Latch (D = Data)

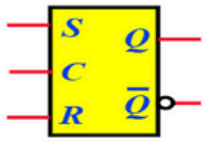




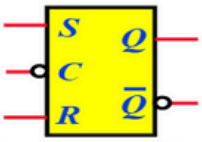
C	D	Q(t+1)	
0	x	Q(t)	No change
1	0	0	Reset
1	1	1	Set

الخلاصه

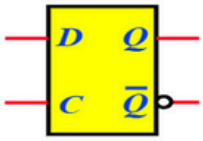
Latches Summary



+ve pulse-triggered SR latch

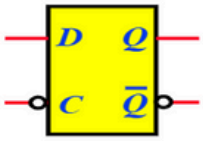


-ve pulse-triggered SR latch



+ve pulse-triggered D latch

C = 0 → Hold
C = 1 → Change



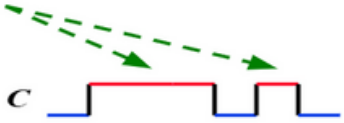
-ve pulse-triggered D latch

C = 1 → Hold
C = 0 → Change

Controlled Latches

★ In controlled latches, the stored value may change *as long as the latch is enabled*

★ In other words, we say that controlled latches are *level-triggered*



★ What if we need to have more control such that the stored value is allowed to change *at specific time*?

★ Flip-flops!

IMPORTANT

الـ Controlled Latch هو عنصر يُستخدم لتخزين قيمة رقمية ويتغير محتواه فقط عندما تكون إشارة التحكم مفعّلة.

يُسمى هذا النوع بـ Level-Triggered لأنه يسمح بتغيير القيمة طالما أن إشارة التمكين في حالة معينة (مثل High).

هذا يعني أن القيمة قد تتغير خلال فترة التفعيل، وليس في لحظة واحدة فقط يعني مش في لحضه ارتفاع او نزول الحضه الذي يحصل فيها الحدث.

لكن إذا أردنا التحكم بحيث تتغير القيمة في لحظة محددة فقط، فنحتاج إلى نوع آخر من العناصر.

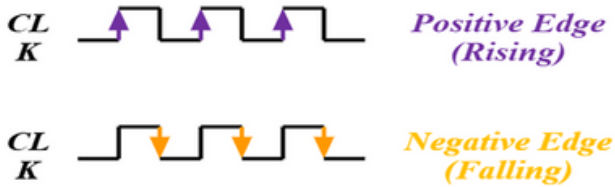
الحل هو استخدام Flip-Flops، وهي تخزن القيمة عند لحظة انتقال محددة من إشارة الساعة فقط.

Flip-Flops

★ Flip-Flops are *edge-triggered* storage elements

★ They allow the input to affect the stored value *at the edge of the clock only*

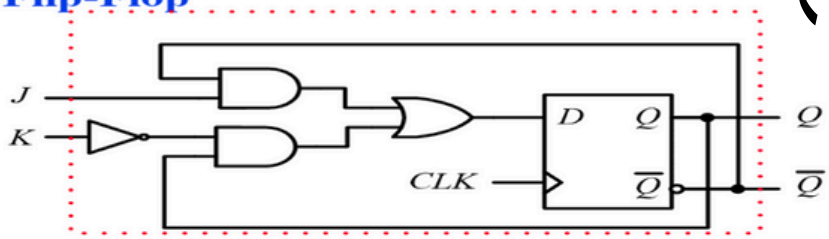
- *positive*
- *negative*

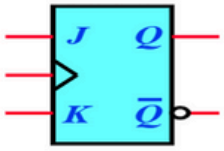


هون في الفليب بفرق عن latches كل ما صار ارتفاع او نزول في قيمه الclk بتقدر تغير القيمه ويا بتختار ارتفاع او نزول ما بصير الاثنتين مع بعض

امثله عليهم

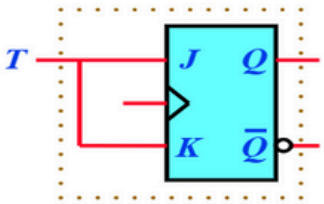
★ *JK* Flip-Flop

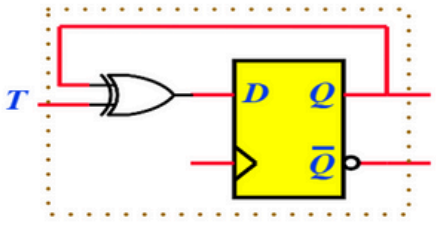


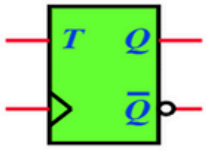


J	K	Q(t+1)	
0	0	Q(t)	Hold
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Toggle

★ *T* Flip-Flop

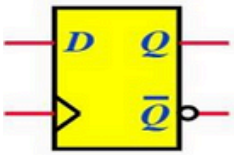




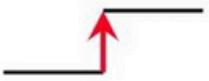


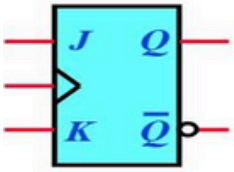
T	Q(t+1)	
0	Q(t)	Hold
1	Q'(t)	Toggle

Flip-Flop Characteristic Tables (For Analysis)

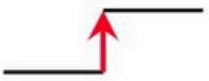


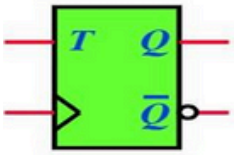
D	Q(t+1)	
0	0	Reset
1	1	Set



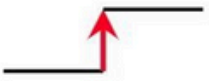


J	K	Q(t+1)	
0	0	Q(t)	Hold
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Toggle

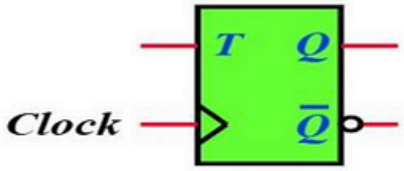




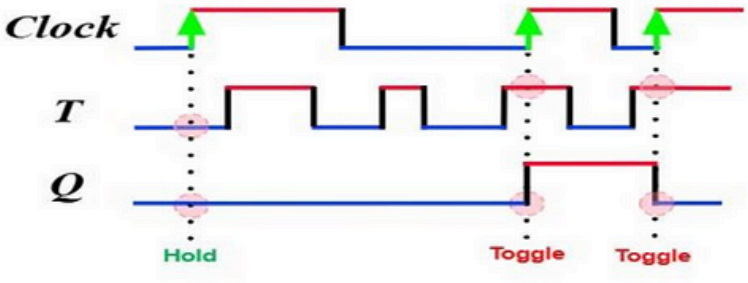
T	Q(t+1)	
0	Q(t)	Hold
1	Q'(t)	Toggle



مثال توضيحي مع الكل وعند كل
ارتفاع في قيمه الكلك يحدث
تغير في القيمه

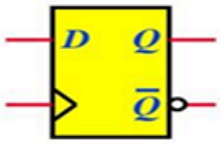


T	Q(t+1)	
0	Q(t)	Hold
1	Q'(t)	Toggle



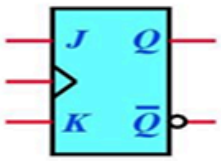
معادله لكل نوع

Flip-Flop Characteristic Equations (For Analysis)



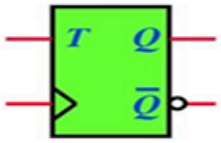
D	Q(t+1)
0	0
1	1

$$Q(t+1) = D$$



J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Q'(t)

$$Q(t+1) = JQ' + K'Q$$

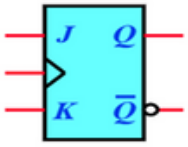


T	Q(t+1)
0	Q(t)
1	Q'(t)

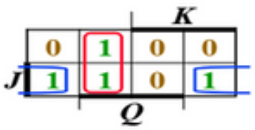
$$Q(t+1) = T \oplus Q$$

Leath Abu Rumman

★ Analysis / Der



J	K	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



$$Q(t+1) = JQ' + K'Q$$

كيف نطلع معادله كل فليب اول اشي لازم نعوض في TABLE بعدين بنعمل K-MAP

CHAPTER SEX

REGISTERS

chapter 6

Registers

register:

Is a group of flip-flops, each one of which shares a common clock and is capable of storing one bit of information

هي مجموعة من العمليات، كل واحدة منها تتشارك في ساعة زمنية مشتركة وقادرة على تخزين بت واحد من المعلومات.

register

وهو مكون في المعالجات والأنظمة الرقمية يُستخدم لتخزين البيانات مؤقتًا. يتكون من مجموعة من flip-flops، وكل flip-flop يخزن 1 bit.

Types of registers:

1. Register with Parallel Load
2. Shift Registers

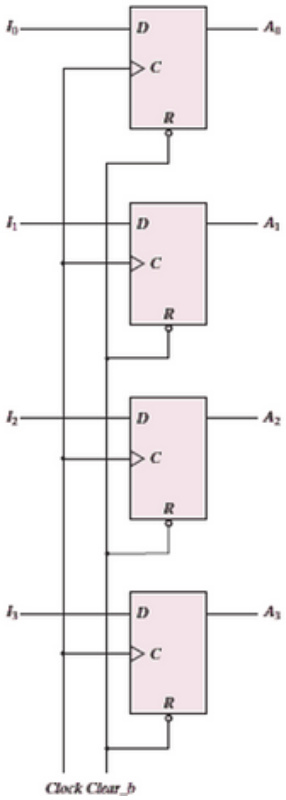
Register with Parallel Load

التحميل المتوازي (Parallel Load):

- يعني أن كل Flip-Flop داخل السجل يستقبل قيمة جديدة في نفس اللحظة (نفس نبضة الساعة).
- يُستخدم عندما نحتاج إلى إدخال عدد من البتات (مثل 4 أو 8 أو 16 بت) دفعة واحدة.
- يتكون السجل من Flip-Flops من نوع D، حيث كل Flip-Flop يخزن بت واحد.
- يوجد أمام كل Flip-Flop Multiplexer (MUX):
  - لاختيار ما إذا كنا نريد تحميل قيمة جديدة أو الاحتفاظ بالقيمة الحالية.
- خط التحكم بالتحميل (Load Control Line – LOAD):
  - إذا كانت قيمة LOAD = 1: يتم تحميل البيانات الجديدة في السجل.
  - إذا كانت قيمة LOAD = 0: يحتفظ السجل بالقيم المخزنة حاليًا.

:Parallel Load

- Means that every Flip-Flop in the register receives a new value simultaneously (at the same clock pulse).
- Used when you need to input multiple bits at once (like 4, 8, or 16 bits).
- The register consists of D Flip-Flops, where each Flip-Flop stores one bit.
- There is a Multiplexer (MUX) in front of each Flip-Flop:
  - To select whether to load a new value or keep the current value.
- Load Control Line (LOAD):
  - If LOAD = 1: new data is loaded into the register.
  - If LOAD = 0: the register retains its current stored values.

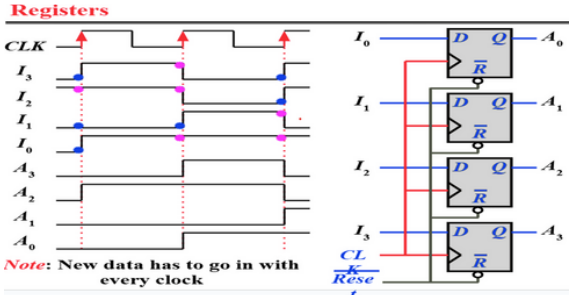


هاي الرسمه بتوضح طريقه عمله عنا اربعه فليب وبدخل عليهم اربع قيم مع بعض وبخرج اربع قيم برضو مع بعض وعنا اهم اشي CLOCK بتحكم بوقت ادخال القيم واله قيمتين يا اما 1 او 0

IMPORTANT

الـ Clock زي نبضات القلب بتتحكم في توقيت شغل الدوائر، كل نبضة (طلعة أو نزلة) بتحدد إمتى العناصر زي الفليب-فلوب تقرأ أو تخزن البيانات.

أغلب الدوائر بتشتغل على الطلعة (rising edge)، يعني أول ما ترتفع الساعة من 0 إلى 1، مش لما تصير 0.وفي هار النوع من الرجستر بشتغل على rising edge يعني لما يصير في ارتفاع في قيمه الكلك لحضتها بدخل القيم الجديده مع بعض



هاي الرسمه بتوضح كيف بشتغل الكلك ركز في الرسم البيان موضح النا قيم كل من المدخلات I0 I1 I2 I3 وقيم المخرجات A0 A1 A2 A3 وقيمه الكلك لاحظ تتغير قيم المخرجات فقط عند حصول زياده في قيمه الكلك وهاض ما يسمى RISING EDGE باختصار شو ما اغير بقيم المدخلات ما رح تتأثر المخرجات غير لما يصير ارتفاع في قيمه الكلك .واذا ضلت قيمه الكلك ثابت معصفر او واحر بدون ارتفاع رح تضل قيم المخرجات نفسها ما رح تتغير

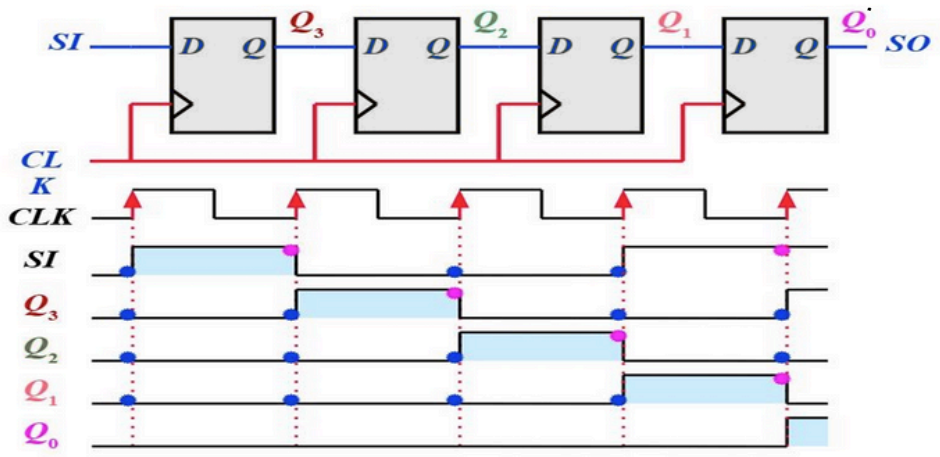
Shift Registers

It is a circuit that receives data serially (one bit per clock pulse), and outputs it in the same serial manner from the other end after passing through several flip-flops.

هي دائرة تستقبل البيانات بتسلسل (بت واحد في كل نبضة ساعة)، وتُخرجها بنفس الطريقة من الطرف الآخر بعد مرورها من خلال عدة فليب-فلوبات (Flip-Flops).عكس النوع الي قبله لان هديك بتوخذ اربع قيم مع بعض هاي بتوخذ قيمه قيمه وحده وحده وهدفها الاساسي نقل البينات وهديك استقبال البينات

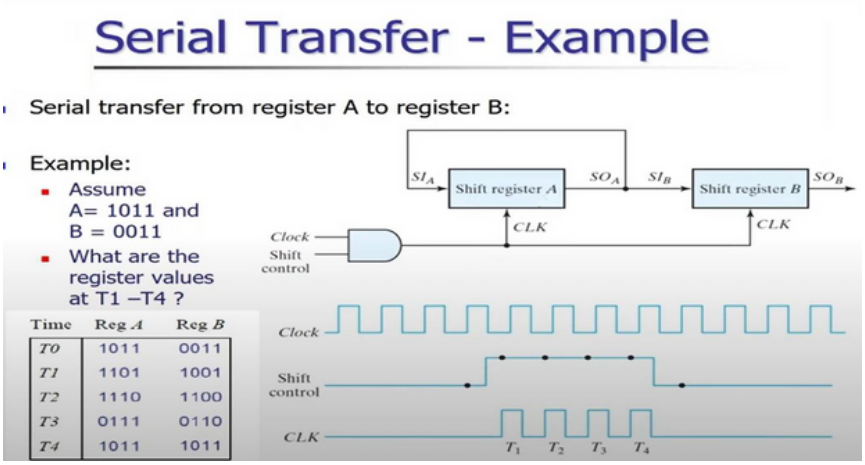
1. SI (Serial Input):
2. This is the serial data input, where a bit (0 or 1) enters the first flip-flop on each clock pulse.
3. D Flip-Flops (4 in total):
4. Each flip-flop holds the bit received from the previous stage and passes it to the next one on every clock pulse.
5. CLK (Clock):
6. These are the clock pulses that control the timing of data movement between the flip-flops in the register.
7. SO (Serial Output):
8. This is the serial output, where the bit exits after shifting through all the flip-flops.

1. SI (Serial Input):
2. هذا هو مدخل البيانات التسلسلي، حيث يدخل البتّ (0 أو 1) إلى أول فليب-فلوب في كل نبضة ساعة (Clock pulse).
3. D Flip-Flops (4 عددهم):
4. كل فليب-فلوب يحتفظ بالبت القادم من الفليب-فلوب اللي قبله، وينقله للي بعده مع كل نبضة ساعة.
5. CLK (Clock):
6. هي نبضات الساعة التي تتحكم في توقيت انتقال البيانات من مرحلة إلى مرحلة داخل المسجل.
7. SO (Serial Output):
8. هذا هو الخرج التسلسلي، يتم فيه إخراج البت بعد مروره خلال جميع الفليب-فلوبات.



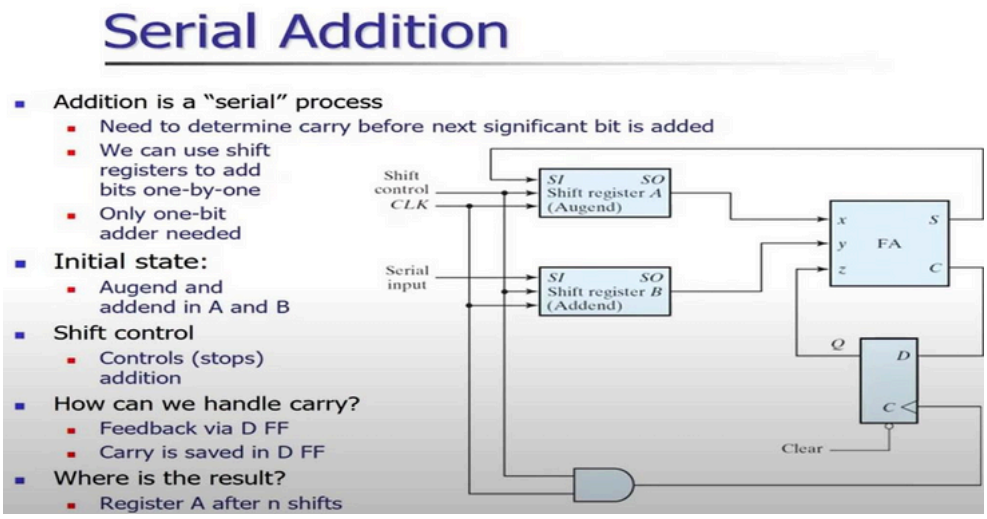
هون برضو عنا اربعه فليب لكن بنوخذ قيمه وحده وبنطلع قيمه وحده هاي القيمه متى بدخلها برضو عن طريق الكلك كل ما يرتفع بندخا قيمه والقيم الي بتكون في الفليب بتروح على الفليب الي بعده وهنكذا مثلا كان عنا في الفليب الثاني قيمه 1 وارتفعت قيمه الكلك رح تدخل قيمه على الفليب الاول وقيمه الاول تروح لثاني وقيمه الثاني الي كانت 1 بتروح لثالث وقيمه الثالث لرابع وقيمه الرابع بتطلع برا هيك مبداء الشفت

Serial Transfer



في الصوره بوضح النا عمليه نقل معلومات من رجستر الى رجستر ثاني هون بنستخدم شيفت رجستر وكل رجستر زي ما تعلمنا بتكون من اربع فليب يعني عنا هون ثمنيه فليب ورح ننقل من A الى B ومعطينا قيمهم الاثنتين.بس في شغله كثير مهمه عنا هون كلوك خارجي بدخل القيمه نفسها على الرجسترين وهيك بنفهم انه كل ما ترتفع قيمه الرجستر بتنتقل قيمه من A الى B وعنا اربع انتقالات T1 T2 T3 T4

Serial Addition



لجمع التسلسلي هو طريقة لإجراء عملية جمع رقمين ثنائيين بتًا ببت (bit by bit)، باستخدام دارة بسيطة تتكوّن من:

- مسجلين للإزاحة (Shift Registers): يحتويان على الرقمين المراد جمعهما.
- جامع كامل (Full Adder): يجمع كل بت من الرقمين مع بت الحمل (carry).
- Flip-Flop (نوع D): يخزن قيمة الحمل (Carry) للجولة التالية.

IMPORTANT

لو نركز فصوره FA بطلع منه اوت بوت ثنتين S C الي هما الكري والناتج الكري بروح بتخزن في الفليب D والناتج بضل بتخزن في الرجستر A ولانه نوعه SHIFT REGISTERS مثال لتوضيح لو كانت القيمه داخل A =00
وطلع الناتج 1 رح يتخزن في A وبنعمل شفت فبصير 10 وبنرجع نعمل للبت الاخير الي هو صفر بطلع معنا ناتج مثلا 1فبصير قيمه الرجستر A هي الناتج =11

CHAPTER SEX

MEMORY

chapter 7

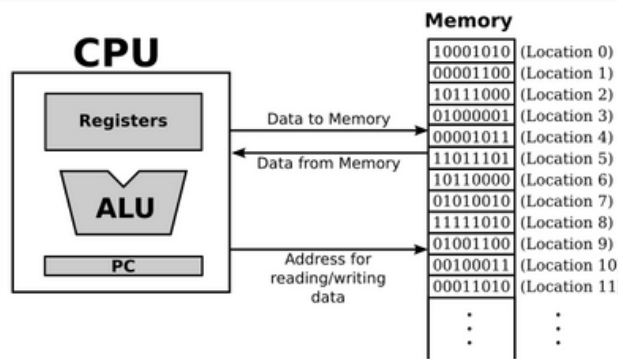
Memory

Memory unit :

The device to which binary information is transferred for storage and from which information is retrieved when needed for processing

الجهاز الذي يُنقل إليه المعلومات الرقمية (الثنائية) للتخزين، ويُسترد منها عند الحاجة للمعالجة

▶When data processing takes place, information from memory is transferred to selected registers in the processing unit



▶There are two types of memories that are used in digital systems: random-access memory (RAM) and read-only memory (ROM)

هناك نوعان من الذاكرة المستخدمة في الأنظمة الرقمية: الذاكرة العشوائية (RAM) وذاكرة القراءة فقط (ROM)

Contentsofa 1024x16 memory

Memory address		Memory content
Binary	Decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

- تحتوي على 1024 عنوان ذاكرة (من 0 إلى 1023) ,كل عنوان ذاكرة يتسع لـ 16 bit من البيانات (ما يعادل 2 Byte)
- يتم تمثيل العناوين بنظام ثنائي (10 bit) وعشري الثنائي فقط لتمثيل العشري مثلا 1 عشري = 0000000001 وتكون سعته 10 bit
- البتات الكلية = 1024 عنوان × 16 بت = 16,384 bit. السعة بالبايت = 16,384 ÷ 8 = 2,048 byte 2) كيلوبايت)

IMPORTANT

اهم نقطه لازم نعرفها عشان نعرف كم بحتاج خط عشان امثل موقع في الذاكره
نستعمل هاي المعادله $1024=2^x$ اكس هي عدد الخطوط
وكيف نمثل القيمه داخل الموقع بنحط خطوط بعدد السعه هون عنا في الصوره 16 bit

ROM

Read-Only Memory (ROM):

- A memory device that stores permanent binary data, retaining information even when power is off.
- Features k inputs (address selection) and n outputs (data reading).

Key Characteristics:

- Non-volatile (data persists without power).
- Used for firmware and essential instructions storage.

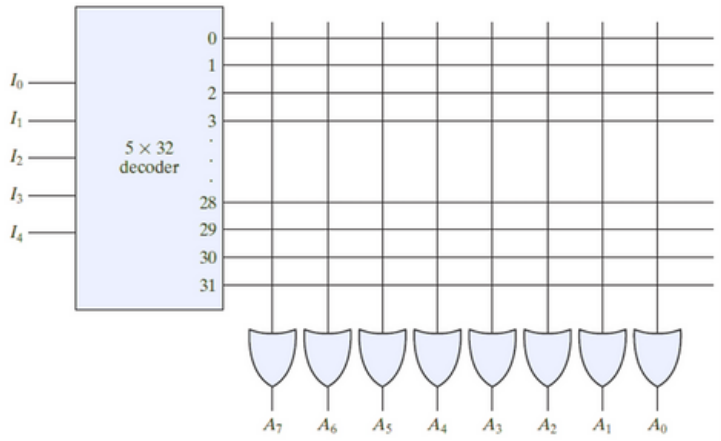
ذاكرة القراءة فقط (ROM):

- ذاكرة تخزن بيانات ثنائية دائمة تبقى محفوظة حتى بعد انقطاع التيار الكهربائي.
- تحتوي على مداخل (k) لاختيار العناوين ومخارج (n) لقراءة البيانات.

الخصائص الرئيسية:

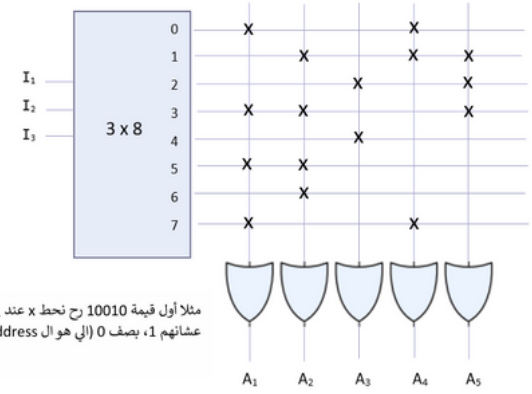
- غير متطايرة (لا تفقد البيانات عند انقطاع الطاقة).
- تُستخدم لتخزين البرامج الثابتة والتعليمات الأساسية

ROM Example



يكون السوال معطيك معلومه 32x16. زي ما حكينا فوق عدد الخطوط للموقع محدد هي هون عرفنا من 2^5 =32 اذا عدد الخطوط خمسه I0 I1 I2 I3 I4
هي خطوط القيمه المخزنه راخل الموقع وهي هنا8 Bit ...A0 A1 A2 A3

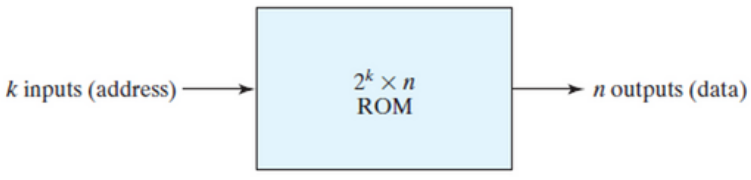
هون طبقنا اكثر من رقم وكل اكس بدل على ان القيمه عنده كانت 1 مش 0 مثلا اول خط ادخلنا
input=000
ووالقيمه داخل هذا الموقع 10010
يعني باستخدام الديكودر دخلنا في موقع معين في الذاكره وطلعنا القيمه الي بداخله اذا في ان بوت وفي اوت بوت



مثلا أول قيمة 10010 رح نحط x عند A_1 و A_4 عشانهم 1، بصف 0 (الي هو ال address لهاي القيمة)

لازم تدربه على النومين لان يمكن يجيبلك الرسمه ويطلبك مثلا عند 001 كم كانت القيمه في المموري او يمكن يجيبلك الجدول واانت ترسم الرسمه

Address	Content
I_1 I_2 I_3	A_1 A_2 A_3 A_4 A_5
0 0 0	1 0 0 1 0
0 0 1	0 1 0 1 1
0 1 0	0 0 1 0 1
0 1 1	1 1 0 0 1
1 0 0	0 0 1 0 0
1 0 1	1 1 0 0 0
1 1 0	0 1 0 0 0
1 1 1	1 0 0 1 0



هاي الرسمه مهمه جدا للفهم k هي الان بوت هي عدد الخطوط لتحديد الموقع ومنها بجيب الاوت بوت n عن طريق القيمه داخل الموقع الذي حدتته داخل الذاكره

RAM

Random Access Memory (RAM):

- Volatile storage for temporary data and active programs, connected via data/address/control lines.
- Performs write (store) and read (retrieve) operations with instant access to any memory location.
- Essential for computer main memory and real-time data processing.

ذاكرة الوصول العشوائي (RAM):

- وحدة تخزين مؤقتة للبيانات والبرامج النشطة، متصلة عبر خطوط البيانات/العناوين/التحكم.
- تقوم بعمليتي الكتابة (تخزين) والقراءة (استرجاع) مع إمكانية الوصول الفوري لأي موقع تخزين.
- أساسية للذاكرة الرئيسية في الحواسيب ومعالجة البيانات في الزمن الحقيقي.

Types of RAM Memories

Static RAM (SRAM):

- Consists of internal latches that store binary information
- Faster access time
- More expensive than DRAM
- Used for Cache memory

Dynamic RAM (DRAM):

- Stores binary information as electric charges on capacitors
- Slower performance compared to SRAM
- Less expensive than SRAM
- Used in main memory systems

RAM Cell

RAM في
بنستخدمflip-flop

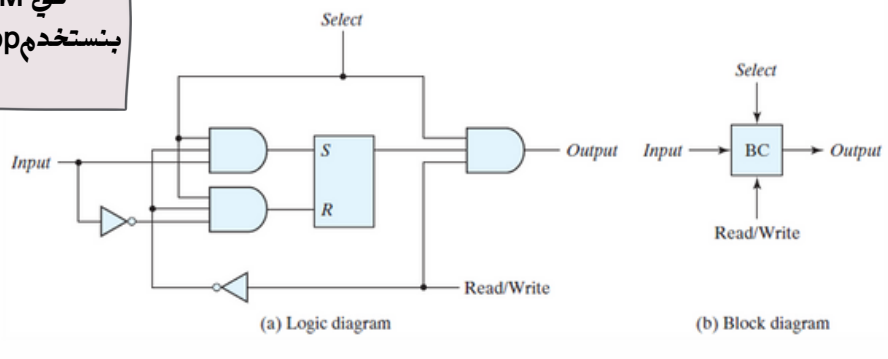
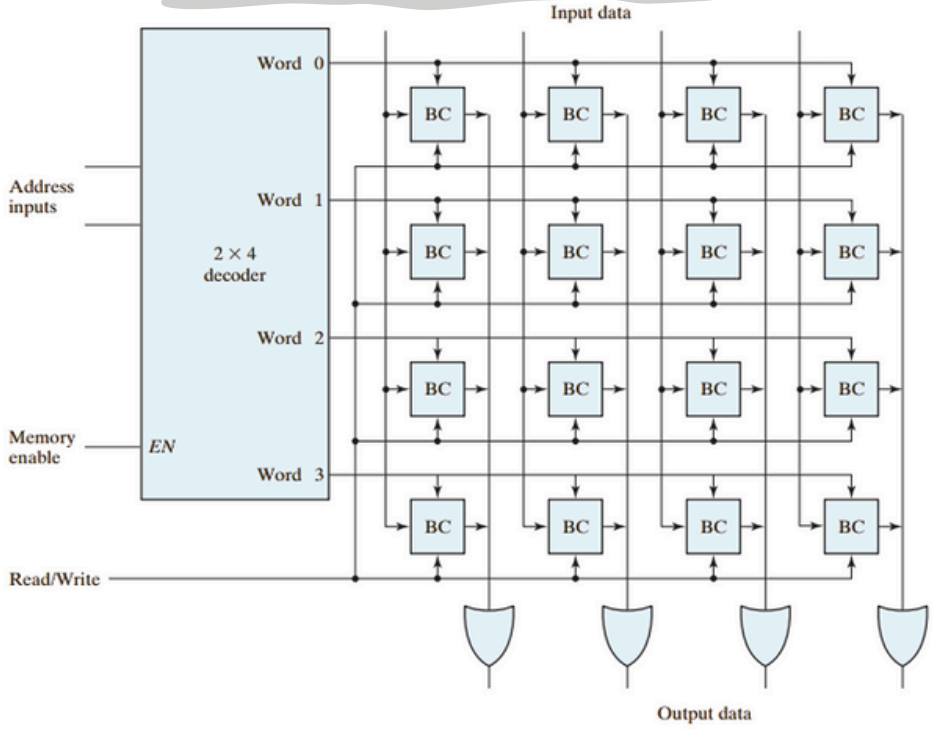


Diagram of a 4X 4 RAM



THE END

"OUR GRATITUDE TO EVERYONE WHO MADE THIS WORK SUCCESSFUL.

WISHING YOU THE VERY BEST IN ALL YOUR FUTURE ENDEAVORS."

Academic Content Department

SEE MORE OF OUR WORK


ERROR!
CATCHERS
WE CATCH WHAT OTHERS MISS !
MORE ABOUT US !