No.

input → Digital system → output

Discrete → Discrete

infinite ←

إذا كان إلها احتمالية
بتكون logic
وكمان بنسميها binary

← أكبر مثال عليهم computers

Digital system

combinational logic systems — توافقي

"No state"

X=0, y=1 → [+] → Z=1

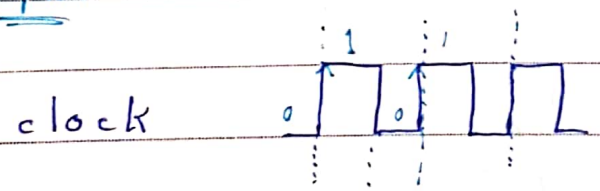
ما بنهتم بالقيمة الأولية
ما بنهتم input

sequential logic systems — تسلسلي

← لازم يكون متركز شو المدخلات

متزامن / بالوقت
Synchronous
أوقات محددة.
بتتغير بوقت معين
ما بصير فجأة تتغير

Asynchronous
أي وقت



$f = 2\,GHz \longrightarrow T = \frac{1}{f} = \frac{1}{2\times10^{9}}$ seconds

$= \frac{1}{2}$ nsec

10^{-9}

f ← frequency
السرعة

smile for life

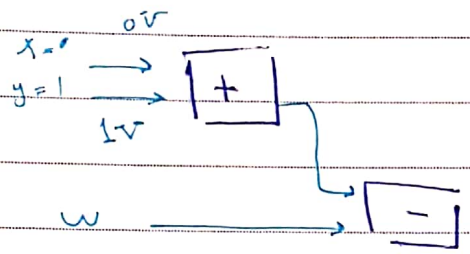
No.

clock $f = 0.25$ nsec

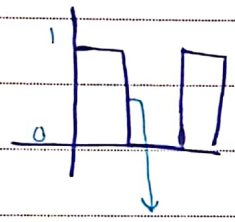
صار أصغر

$0.6\,V \rightarrow 1\,V = 1$ logic

$0.0\,V \rightarrow 0.4\,V = 0$ logic



0.5 v undifened.



0.5V undifened

لما نيجي ننقل من 0 ← 1

أو العكس

2 مطاوع * Radix → base → r

عشري decimal system

$r = 10$

$0, 1, 2, .. 9$ $r^0 = 1$ r^{-2}

Most significant digit (MSD) → The position is important

$$\underbrace{A_{n-1} \cdots A_2 A_1 A_0}_{\text{n-digit Integer}} \underset{\text{radix point}}{\bullet} \underbrace{A_{-1} A_{-2} \cdots A_{-m}}_{\text{m-digits Fraction}}$$

r^2 least LSD

No.

$r = 10$ ~~dem~~ decimal point

$r = 2$ binary point

← الفرق

أسئلة تحويل :

$0 \leq A_i < r$

$0 \leq A_i \leq r-1$

$i \rightarrow$ position

$r^i \rightarrow$ weight

* $(Number)_r = (\quad)_{10}$

* هاي الفرق لحابدنا نحول للعشرية

$$\sum_{i=m}^{n-1} A_i * r^i = \sum_{i=0}^{n-1} A_i \times r_i + \sum_{j=-m}^{-1} A_j \times r^j$$

* $r = 4$

$(233.3)_4 = (47.75)_{10}$

2 1 0 -1

radix point

$3*4^{-1} + 3*4^0 + 3*4^1 + 2*4^2$

$\frac{3}{4} + 3 + 12 + 32$

$(403)_5 = 3 + 0 + 4*5^2 = (103)_{10}$

2 1 0

$2^{16} = 2^6 * 2^{10}$

$= 64 * 2^{10}$

$= 64\,k$

$r = 12 \rightsquigarrow$ not common

$0, 1, 2, \ldots, \underset{10}{a}, \underset{11}{b}$

* $r = 2$

$\{0, 1\}$

digit $\equiv$ bit

8-bit $\equiv$ byte

انتبهي بالعمليات الحسابية

finall answer

2048 x 8 bits

2 k Byte

2 kB

~~0, 1, 2, ... 9,~~

$\rightarrow 2 \times 1024 \times 8$ bits

* $(\overset{4\,3\,2\,1\,0}{10011}.\underset{-1\,-2\,-3}{101})_2$

$1 \times 2^{-3} + 1 \times 2^{-1} + \underbrace{1 + 1 \times 2 + 1 \times 2^{4}}_{1 + 2 + 16} = (19.625)_{10}$

$0.125 + 0.5$

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

$0.5 + 7 + 16 + 64$

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

* $(A5)_{16} = (165)_{10}$

$10 * 16 + 5$

* أكبر الأرقام بالأنظمة

$r=2$	$r=8$	$r=10$	$r=16$
$11=3$	$77=63$	99	$FF=255$

* الرقم اللي بيجي بعدهم وبتتغير الخانة

$(100)_2=4$ $(100)_8=64$ $(100)_{10}$ $(100)_{16}=256$

الـ weghit بحدد كل قديش بتقلب

$r=10$

0
⋮
9
$(10)_{10}$
$(11)_{10}$
$(12)_{10}$
⋮
$(99)_{10}$ → أكبر رقم من خانتين بالعشري
زدنا خانة جديدة $(100)_{10}$

$r=2$

0
1
0
⋮
00
01
$(10)_2 \rightarrow =(2)_{10}$
$(11)_2 \rightarrow (3)_{10}$
$(100)_2 \rightarrow (4)_{10}$

كل مرتين

$(77)_8 \rightarrow (100)_8$

$(77)_8 = (7*8^1+7)_{10}$
$=(63)_{10}$ ← أكبر رقم بالتمثيل برقمين
$(64)_{10} = (100)_8$

00
01
02
⋮
$(10)_8 = (8)_{10}$
$8^1\ 8^0$
$(11)_8 = (9)_{10}$
$(12)_8 = (10)$
⋮
$(17)_8 = (15)_{10}$
$(20)_8 = 16$

0
1
2
3
⋮
F
10
11
12
⋮
1F
20

$(FF)_{16} = 15*16+15 = (255)_{10}$

$(256)_{10} = (100)_{16}$

من عشري لباقي الأنظمة.

No.

$(46.6875)_{10} = (\quad)_2$

integer

① convert Integer

	Q	R
$46 \div 2$	23	0
$23 \div 2$	11	1
$11 \div 2$	5	1
$5 \div 2$	2	1
$2 \div 2$	1	0
$1 \div 2$	0	1

الأرقام الصحيحة

LSD

ما بيطلع إلا 0, 1

لأنه بدي أحوله للنظام الثنائي

إذا طلع غير جذع بكون عندي خطأ بالقسمة

MSD

$(46)_{10} = (101110)_2$

MSD

0.6875×2	1.375
0.375×2	0.75
0.75×2	1.5
0.5×2	1.0

الأرقام العشرية

LSD

* إذا ما حدد السؤال كم رقم بدهم بعد الفاصلة المفروض نكمل ضرب لحتى يوصل الصفر

$(0.6875)_{10} = (0.1011)$

$\Rightarrow (101110.1011)_2$

آخر اشي بندمجهم

No.

$* (153.513)_{10} = (231.4\cancel{06})_8$ → 7

	Q	R → (0→7)		
① 153/8	19	1	LSD	Truncate.
19/8	2	3		
2/8	0	2	MSD	

② 0.513 x 8 — 4 . 104 (MSD)

0.104 x 8 — 0 . 835

0.835 x 8 — 6 . 656 (LSD)

0 1 2 3 — No round

4 5 6 7 — Round

0.656 x 8 (5).

بدنا نعرف هاد الرقم
عشان نقرب آخر رد

0.407(5)	0.437(5)
0.410	0.440

$* (423)_{10} = (1A7)_{16}$

	Q	R	
423/16	26	7	LSD
26/16	1	10	
1/16	0	1	MSD

No.

$\times 2^i$

$46 - 32 = 14 \qquad i = 5 \qquad 5\ 4\ 3\ 2\ 1\ 0\ .\ -1\ -2\ -3\ -4$

$14 - 8 = 6 \qquad i = 3 \qquad 1\ 0\ 1\ 1\ 1\ 0\ .\ 1\ 0\ 1\ 1$

$6 - 4 = 2 \qquad i = 2$

$2 - 2 = 0 \qquad i = 1$

$0.6875 - 0.5 = 0.1875 \qquad i = -1$

$0.1875 - 0.125 = 0.0625 \qquad i = -3$

$0.0625 - 0.0625 = 0 \qquad i = -4$

64	32	16	8	4	2	1	.	0.5	0.25	0.125	0.0625
0	1	0	1	1	1	0	.	1	0	1	1

$(108.25)_{10} = (\qquad\qquad)_2$

64	32	16	8	4	2	1	.	0.5	0.25
1	1	0	1	1	0	0	.	0	1

512	64	8	1	.
0	1	5	4	

$(\qquad)_{r_1} = (\qquad)_{r_2}$

$(\underbrace{\quad}_{\times r_2}.\underbrace{\quad}_{\times r_2})$

No.

* $\boxed{r = 2^i}$

num bits

$2^{2} = 4$, $2^{3} = 8$, $2^{4} = 16$

* Octal to binary
  hexa to binary

* $(673.12)_8 = (110\ 111\ 011\ .\ 001\ 010)_2$

بصير أقيمه

بصير أقيمه

* $(3A6.C)_{16} = (0011\ 1010\ 0110\ .\ 1100)$

* دايماً بنبدا من عند (radix point)

* $(635.177)_8$

000110 011 101 . 001 111 111000

* $(365)_r = (194)_{10}$

$3r^2 + 6r^1 + 5 * r^0 = 194$

$3r^2 + 6r + 5 - 194 = 0$

$(r-7)(r+9) = 0$

$\boxed{r = 7}$

$2^n \longrightarrow$ عدد الاحتمالات

$2^2 = 4$

00
01
10
11

7 بدي
مثلاً

$2^3 = 8$

3bits

$365 \longrightarrow 2^9 = 512$

أكبر

أصغر $2^8 =$

فما بقدر أستخدمه

* M elements

n = minimum number of bits required to represent M elements

$2^n \geq M > 2^{n-1}$

$8 \geq 7 > 4 \quad n = 3$ ✓

$16 \geq 7 > 8 \quad n = 4$ ✗

✗

$n = \lceil \log_2 M \rceil$

$= \lceil \log_2 7 \rceil = \lceil \log_2 2^{2.5} \rceil = \lceil 2.5 \log_2 2 \rceil$

$= 3$

No.

$= \lceil \log_2 10 \rceil$

$= \lceil \log_2 2^{3.5} \rceil$

$= 4$

r^n

$r^n \geq M > r^{n-1}$

$2^9 = 512$

$n = \lceil \log_r M \rceil$

$365 \rightarrow n = \lceil \log_2 365 \rceil = \lceil \log_2 2^{8.5} \rceil = 9$

$365 \rightarrow r = 8$

$r^n \rightarrow 8^3 = 512$

$n = \lceil \log_8 365 \rceil$

$n = \lceil \log_8 8^{2.5} \rceil = 3$

r Maximum "M" $\rightarrow r^n$

n

$5^2 = 25 - 5 = 20$ unused

one hot $\rightarrow$ بدنا مختلف ويكون

بس مستخدم bit واحد

لازم يكون عشري

decimal

$(13)_{10} = (0001\ 0011)_{BCD}$ $\leftarrow$ 8-bits

$(13)_{10} = (1101)_2$ $\leftarrow$ 4-bits

No.

غا احتمالين

$r = 2$ $\qquad\qquad$ $r = 10$

$(13)_{10} = (1101)$ $\qquad\qquad$ $(13)_{10} = (0001\ 0011)$

ممكن هيك يتم تخزينه

* Excess 3 ← يعني بزيد 3

	8	4	2	1
0	0	0	1	1

+3

complement

$3 \longrightarrow 9 \quad = 12$

8 , 4 , -2 , -1

$1 \rightarrow [0 \quad 1 \quad 1 \quad 1]$

$0 \rightarrow [0 \quad 0 \quad 0 \quad 0]$

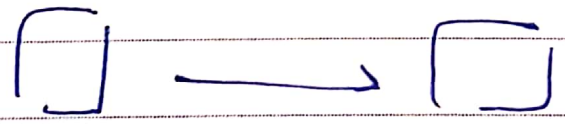
Gray code $\qquad$ الفرق يكون 1- bit

2- componants

يستهلك الطاقة

producer $\qquad$ consumer

ينتج طاقة



ASCII code chart

$A \equiv (41)_{16} \longrightarrow (100\ 0001)_2$

عمود ← صف

	0	1	2	3
3 موقع	0	1	2	3

$a \equiv (61)_{16} \equiv (110\ 0001)_2$

عمود ← صف

$A \equiv (100\ 0001)_2$

إختلاف بـ bit وحدة بين capital الكبتل وبين ال small

* uni code

2 byte (16-bits) code words.

* Parity Bit Error-Detection Codes

add → one extra bit (0 ، 1)

0001 ↳ one extra bit

* هل صار في خطأ بالداتا أو لأ

99% → بس وحدة الله بتنقلب وبتغير

odd parity: 00010

even parity: 00011

ال parity بتعرف إذا فيه bit تغيرت أو لأ مع أنه ممكن يظل محافظ على نفس القيمة

even → even

odd → odd

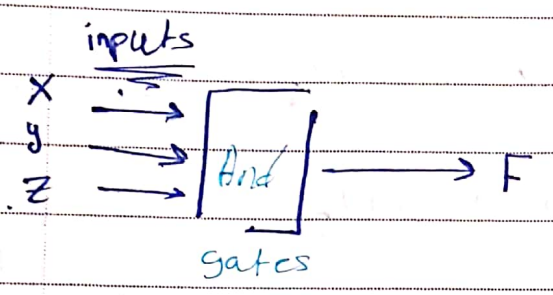
→ Error = 1 فيه error

→ Error = 0 ما في error

parity بتربط كل odd Single bit error

يكون التغير

* Combinational Logic Circuits



Arthimetich operator

logic operator

AND, OR, NOT → Binary information

gates

* boolean Algebra.

* Binary variables

True / false

On / off

yes / NO

1 / 0

* logical operators

AND ≡ . ≡ ∧ A . B , A ∧ B

OR ≡ + ≡ ∨

NOT ≡ ~ ≡ ¯ ≡ ' ~A, $\bar{A}$, A'

$Z(X, y) = X \cdot y = X \wedge y = Xy$

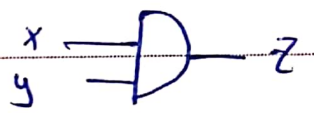
لـ هيك بعني
أنه Z بتعتمد
2-vari على

Z equals X AND y

$Z = 1$ if and only if $x == 1$ AND $y == 1$
else
$Z = 0$

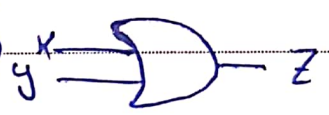
$\bar{C} = 1 \quad C = 0$

* CMOS transistors ← هاي المشهورة هلأ.



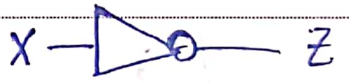
$z = X \cdot Z$

AND



$Z = x + y$

OR



$z = \bar{X}$, $\sim X$, X'

NOT

$L(A, B, C, D) = \underbrace{A \cdot D}_{4} + \underbrace{A \cdot B \cdot \bar{C}}_{2}$

$2^4 = 16$ ← احتمال.

B C لـ الناقص عندي

0 1 0 1

1 2 3 4

لـ الناقص عندي

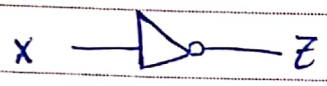
D

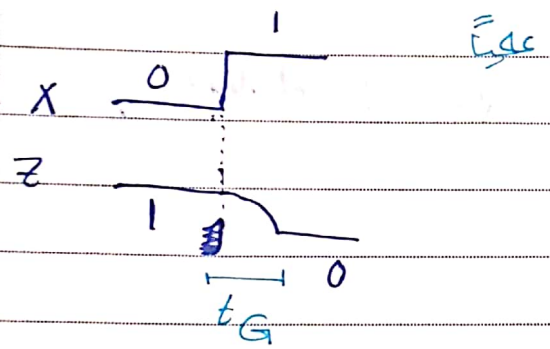
0 1

No.

* gate delay $= t_G$

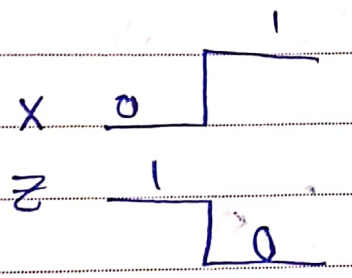
Ex:-





عكس

بس هاد المطلوب منا

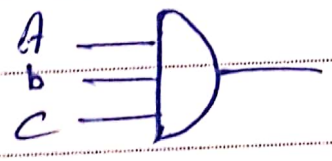


NOT (inverter) → always one input

AND, OR

→ Always one output

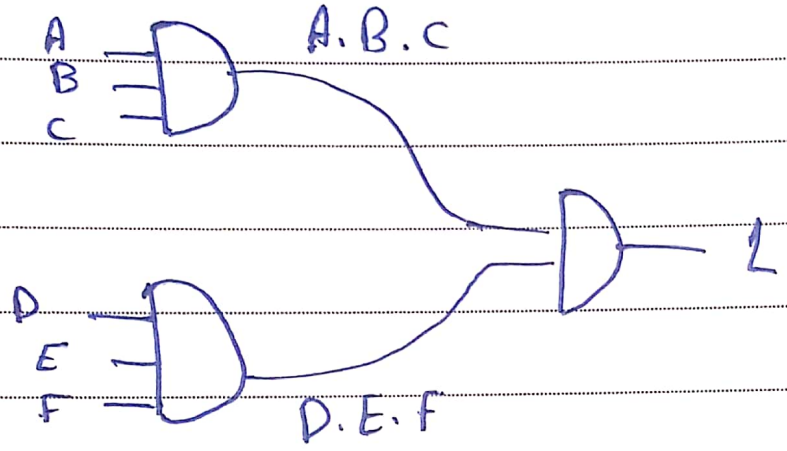
→ Two or more inputs.



$X = ABC$

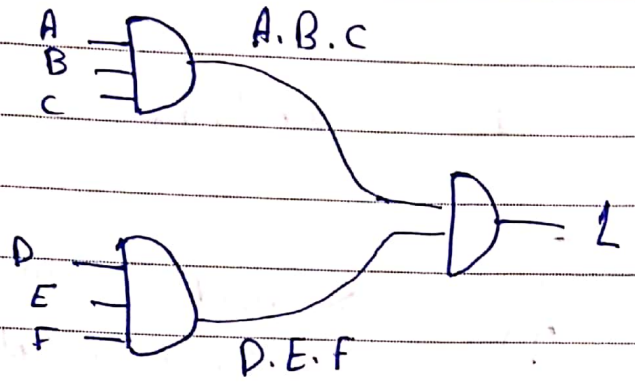
No.

$L = A.B.C.D.E.F$



No.

$L = A.B.C.D.E.F$



* Minimum → 3-digit decimal pin code ← coding

→ $n = \lceil \log_2 M \rceil$

$n = \lceil \log_2 1000 \rceil \rightarrow \boxed{n=10}$ → 10

لو octal

$r^3 \rightarrow 8^3 = 512$

← عدد ال digits = 9

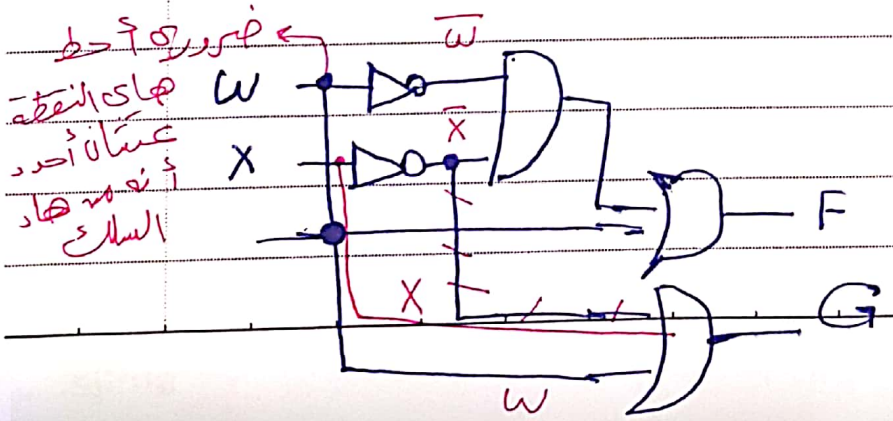
truth table always the same

أما ممكن ال gates و ال function تتغير حسب أشكالهم.

* Multiple output function

$F(w,x) = \bar{w}\bar{x} + w$

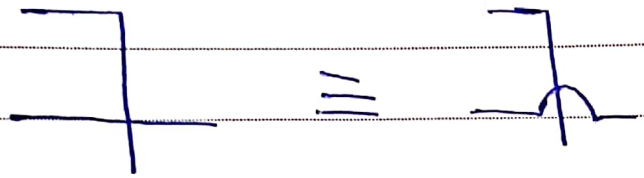
$G(w,x) = w + \bar{x}$



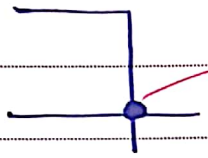
* وكمان عشان ما يبيسوا السلكين متصلين أو فوق بعض

* ضرورة وضع الأقواس .

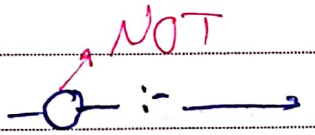
* عشان ما نلخبط نكتب حالة كل gate ال output
فيكون واضح وأسهل كتابة المعادلة



⇒ معناها أنه هدول
السلكين موصولين
مع بعض



هاي قصدنا
أنه السلكين
موصلين مع بعض



هاي تاعت الـ inverter

مفرقة عشان نميزها
عن الوصل

* كلما صارت الـ circuit أبسط بتقل مقدار الـ power
والتكلفة فعشان هيك رح نحاول نبسط الـ function
والـ gates

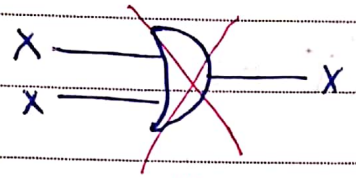
$X + 0 = X$ ($+$ → OR)

$X \cdot 1 = X$ ($\cdot$ → And)

$X + X = X$ $\qquad$ $X \cdot X = X$

$0 + 0 = 0$ $\qquad$ $0 \cdot 0 = 0$

$1 + 1 = 1$ $\qquad$ $1 \cdot 1 = 1$



AND → $X \cdot \overline{X} = \boxed{0}$

OR → $X + \overline{X} = \boxed{1}$

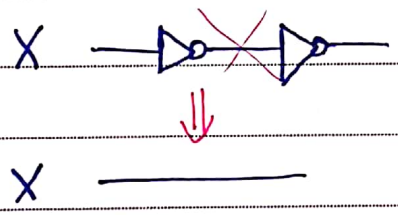
Rule

⇓

بنقدر نشيلها
ونبسط ال
design
إلي عنا

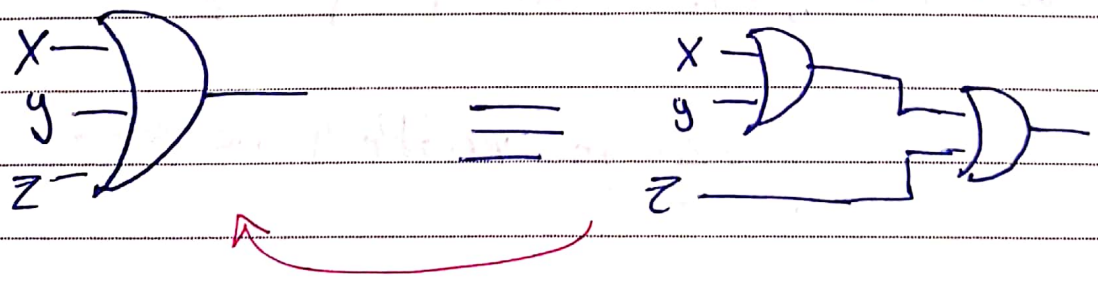
X ——

* Involution → $\overline{\overline{X}} = X$



→ ازا ما كانوا مستخدمين repeaters

* لما يكون عندي نفس ال operation بقدر أجمع
زي ما بدي لأنه الترتيب مو مهم



هيك بنبسطها

* مهم :-

- $X \cdot (y+z) = X \cdot y + X \cdot z$

- $X + y \cdot z = (X+y) \cdot (X+z)$

الأقواس هون مهمة

- $\overline{X+y} = \overline{X} \cdot \overline{y}$

- $\overline{X \cdot y} = \overline{X} + \overline{y}$

De Morgan's laws

$\Rightarrow X + 0 = X$

$(A \cdot B \cdot C) + 0 = A \cdot B \cdot C$

الـ Rules بنقدر نقيم الـ single variable

ونحط بدالها expression

$\Rightarrow \overline{AB + C\overline{D}} = \overline{AB} \cdot \overline{C\overline{D}}$

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

x y

$\Rightarrow \overline{w + x + y + z} = \overline{w} \cdot \overline{x} \cdot \overline{y} \cdot \overline{z}$

← عدد الـ veriable موضح

No.

Dual operation متبين

يعني لو أثبتنا واحدة منهم إلي قبالها
رح تكون مثبتة لحالها

$(A + \bar{B} + C) \cdot D$

$\bar{A} \cdot B \cdot \bar{C} + D$

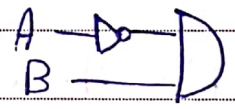
$X + 0 = X$
$X + 1 = X$] → self-Dual → كل ال possible-cases نفس ال answer

يعطوني نفس الجواب

Involution → self dual

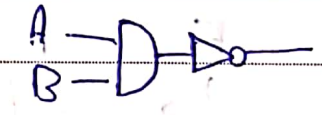
1+2 5+6 +9

$F = \sim A \cdot B = \bar{A} \cdot B$ →



$G = \sim (A \cdot B) = \overline{A \cdot B}$

→



بال duality بدنا نتنبه على الأقواس لأنه ممكن يطلع الجواب خطأ.

$\overline{X + y}$ → أول شي بنعمل ال OR بعدين بنحطلها NOT

① $F = (A + \bar{C}) \cdot B + 0$

Dual F

$$F = (A + \bar{C}) \cdot B$$
$$= \underset{②}{A \cdot B} \underset{4}{+} \underset{①}{\bar{C}} \cdot \underset{③}{B}$$

⇒ Dual

OR AND

دايماً بنبدل ① + ب ·

والعكس صحيح . ② 0 ب 1

$$F = A \cdot \bar{C} + B \cdot 1$$

F هاي على طول (دايماً)

↳ self - dual هو

② $G = X \cdot y + \overline{(W + Z)}$

$= X + y \cdot \overline{(W \cdot Z)}$ ← بنطبق الـ NOT

③ $H = AB + AC + BC$

Dual H

→ ح يطلع self - dual

أضمن طريقة لمعرفة الـ self-dual الأفضل نعمل على الـ truth table

Dual $H = (A+B)(A+C)(B+C)$

$= (A+BC)(B+C)$

$= AB + AC + B.C.B + B.C.C$

$= AB + AC + B.C + B.C$

$= AB + AC + BC$

* Useful boolean Theorms

~~$\bar{x}+y$~~ Absorption Theorm

$A + A.B = A$ $\qquad X \cdot 1 = X$

$A.1 + A.B$

$A(1+B)$ $\qquad$ Distributive law

✓ directe

$X + 1 = 1$

$A \cdot 1$

A ~~$\bar{A}$~~

$$\overset{A}{X}\,\overset{B}{\bar{y}} + \overset{\bar{A}}{\bar{X}}\,\overset{C}{\bar{w}\,z} + \overset{B}{\bar{y}}\,\overset{C}{\bar{w}\,z} = X\bar{y} + \bar{X}\bar{w}z$$

$$* \quad \bar{X} + X\bar{y} = \bar{X} + \bar{y}$$

لحاس $\rightarrow (\bar{X})$

$X + y + \bar{X}.\bar{y}$

$X + \bar{X}\bar{y} + y$

$X + \bar{y} + y$

$X + 1 = 1$

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

No.

$(x+y)\cdot\bar{x}\cdot\bar{y} \overset{?}{=} 0$

$\bar{x}\cdot\bar{y}\cdot x + \bar{x}\cdot\bar{y}\cdot y$

$0 \quad + 0$

0

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

* proof by boolean Algebraic

$(\overline{x+y})\, z + x\,\bar{y}$

Demorgan ① $\bar{x}\cdot\bar{y}\cdot z + x\,\bar{y}$

* الواحد يحاول يبسط قد ما يقدر

Distribution ② $\bar{y}\,(\bar{x}\,z + x)$

→ simple ③ $\bar{y}\,(x + \bar{x}z)$

$\bar{y}\,(x+z)$ #

(*) ~~Test~~ → Complementing functions ← (*)

$F = x'yz' + x\bar{y}\,\bar{z}$

$\overline{F} = \overline{\bar{x}\cdot y\cdot\bar{z} + x'\,\bar{y}\,\bar{z}}$

الفرق بين الـ Dual والـ Complement أنه المتغير نفسه بتنقلب إشارته (NOT)

$= \overline{\bar{x}\cdot y\cdot\bar{z}} \cdot \overline{x\,\bar{y}\,\bar{z}}$

$= (\bar{\bar{x}} + \bar{y} + \bar{\bar{z}})\cdot(\bar{x} + \bar{\bar{y}} + \bar{\bar{z}})$

$\overline{F} = (x + \bar{y} + z)\cdot(x + y + z)$

No.

* احتمال يجي بالفرست والتركيز على الأقواس

$G = (a' + bc) \cdot d' + e$

$\bar{G} = (a \cdot (\bar{b} + \bar{c}) + d) \cdot \bar{e}$ الـ trick

في السؤال هو الأقواس

← الأقواس الزيادة الصح ما بنحط عليها خط.

$F = x'yz + x'yz' + xz$

← بنحكي هاد 3-terms ما بنطلع على الـ OR ← (+)

* بنعد عدد المتغيرات ← literals

* ممكن يكون في function جوابه constant ← عادي.

* $F(x, y, z) = x \cdot \bar{y} \cdot z + \bar{x} \cdot z (x + \bar{y})$ (y missing غلط بقدر أسميه NiN...)

↳ NON standard

* $F(x, y, z) = x \cdot \bar{y} \cdot z + \bar{x} \cdot z$ AND term ≡ product term

↳ sum of product (SOP)

* $G(x, y, z) = (x + \bar{y} + z) \cdot (x + \bar{z})$ OR term ≡ sum term

product of sums (POS)

* binary ⇒ n variables x, y minterms $\equiv 2^n$ ← AND

maxterms $= 2^n$ ← OR

$\bar{x} + \bar{y}$	$\bar{x} \cdot \bar{y}$
$\bar{x} + y$	$x \cdot \bar{y}$
$x + \bar{y}$	$\bar{x} \cdot y$
$x + y$	$x \cdot y$

No.

$F(x,y) = \bar{x}y + xy$ SOP / SOM

$G(x,y) = (\bar{x}+\bar{y}) \cdot (x+\bar{y})$ POS / POM

* m ≡ minterms m(i) index

$m_0 = \bar{x}\,\bar{y}\,\bar{z}$

0 0 0

Minterms

0 → complement

$m_1 = \bar{x}\,\bar{y}\,z$

0 0 1

$m_2 = \bar{x}\,y\,\bar{z}$

010

m_3

* M = Maxterms M_i

$M_0 = x + y + z$

$M_1 = x + y + \bar{z}$

$M_2 = x + \bar{y} + z$

4 2 1

1 1 1 $\boxed{7}$ * يهتم بعدد المتغيرات يعني لو ٣ معي نهاية

8 4 2 1

1 1 1 1 $\boxed{15}$ " " ٤ "

No.

$\times\ a + b + \bar{c} = M_1$

0 0 1

$*\ b + \bar{a} + c$

0 1 0 $= M_2$ ✗ ⇐ الترتيب مختلف فلا يعتبر Max term

$*\ \bar{a} + b + c$

هاد الصح 1 0 0 $= M_4$

$*\ m_{13} = \bar{a} \cdot b \cdot \bar{c} d$

1101

complement

العلاقة

$M_{13} = \bar{a} + \bar{b} + c + d$

1101

$\ast\ m_i = \overline{M_i}$

$M_i = \overline{m_i}$

* كا يكون كل minterms موجودة بعطينا الجواب 1

* كا يكون ,, Maxterms ,, بعطينا الجواب 0

العلاقة بينهم

$*\ F(x,y) = \bar{x}\bar{y} + \bar{x}y + x\bar{y} + xy = 1$

Dual

+

complement

$*\ G(x,y) = (\bar{x}+\bar{y}) \cdot (\bar{x}+y) \cdot (x+\bar{y})(x+y) = 0$

Minimum Number of ones ←

No.

الـ Minterms :- العلاقة بالـ index

x	y	$m_0 = \bar{x}\bar{y}$	$m_1 = \bar{x}y$	$m_2 = x\bar{y}$	$m_3 = xy$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$$* \; F(x,y) = \bar{x}\bar{y} + x\bar{y}$$
$$= m_0 + m_2$$

x	y	$M_0 = x+y$	$M_1 = x+\bar{y}$	$M_2 = \bar{x}+y$	$M_3 = \bar{x}+\bar{y}$
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

$Maxterm_3 \rightarrow 3 = 0$ ←

رح يكون عندها

Maxterm :- Maximum Number of ones

$$* \; F_1 = m_1 + m_4 + m_7$$

← لازم يكون عنا عدد المتغيرات أو أسماءهم
لأنه عندي أكثر من احتمال ممكن يكونوا
3- varibles أو أكثر
مستحيل يكون أقل لأنه 2-vari
بوقف عند رقم $\boxed{4}$

No.

* عدد ال raws اللي رح تكون فيها ال functions = 1

$F(A,B,C,D,E) = m_2 + m_9 + m_{17} + m_{23}$

function = 1 → 4 مرات

* عدد ال raws تكون = صفر

$2^5 = 32 - 4 = 28$ مرة

* $f(x,y) = x + \bar{x}\bar{y}$ SOP

x	y	f			
0	0	1	→	✓	m_0
0	1	0	→		
1	0	1	→	✓	m_2
1	1	1	→	✓	m_3

بضيف المفقود

[أو]

$= x \cdot (y + \bar{y}) + (\bar{x} \cdot \bar{y})$

$= xy + x\cdot\bar{y} + \bar{x}\bar{y}$

$= m_3 + m_2 + m_0$

$= m_0 + m_2 + m_3$

$= \Sigma m(0, 2, 3)$

No.

$* F(A,B,C) = A \odot (B+\bar{B}) \odot (C+\bar{C}) + \bar{B}\, C \odot (A+\bar{A})$

← نعمل AND مع الـ missing variable

$= \Sigma_m (1, 4, 5, 6, 7)$

$* f(x,y,z) = x + \bar{x}\bar{y}$ "POM" المطلوب منا

$= (x+\bar{x}) \cdot (x+\bar{y})$ بعض الأسئلة تبدأ

$=$ ~~[illegible]~~ "POS" بـ

$= (x+\bar{y}) + z \cdot \bar{z}$ لأنه "POM" ← "POS" خاصة من

POM $= (\underset{0}{x} + \underset{1}{\bar{y}} + \underset{0}{z}) \cdot (\underset{0}{x} + \underset{1}{\bar{y}} + \underset{1}{\bar{z}})$

$M_2 \cdot M_3$

$\Pi_M (2,3)$

product

SOP → POS

POM

$* f(A,B,C) = (AC' + BC) + A'B'$

$= (AC' + BC + A') \cdot (AC' + BC + B')$

$= (A' + C' + BC) \cdot (B' + C + AC')$

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

$= (A' + B + C) \cdot (A + B' + C)$ POS, POM

$\underset{1}{A'} \; \underset{0}{B} \; \underset{0}{C} \quad \underset{0}{A} \; \underset{1}{B'} \; \underset{0}{C}$

$M_5 \cdot M_2 = \Pi_{M(2,5)}$

No.

أو $= AC' \cdot (B+B') + BC(A'+A) + A'B'(C+C')$

$= ABC' + AB'C' + A'BC + ABC + A'B'C + A'B'C'$

110 100 011 111 001 000

$= \Sigma_m (0,1,3,4,6,7)$

$= \Pi_M (2,5)$

$\bar{f}(A,B,C) = \Sigma_m (2,5)$

$= \Pi_M (0,1,3,4,6,7)$

* المعتمد والمنتشر بشكل عام هو ال minterm
لأنه هو بحكيلنا بشكل عام متى الاشي بيشتغل

$f(x,y,z) = \Sigma_m (0,1,2,3,4,5,6,7)$

كلهم موجودين

$= 1$

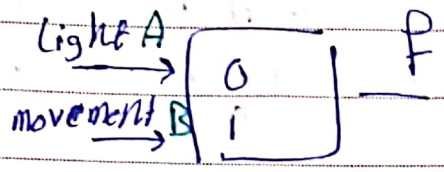
فيطلع الجواب
constant.

2-level Implementation

التسمية إجت من شكل
ال gate

SOP	POS
↓	↓
SOM	POM

the inverter is usually ignored



A	B	f
0	0	0
0	1	1
1	0	0
1	1	0

$* \; A'B'C + A$

$= A + B'C$

⇒ Gate Input Cost ⇐

$ABC + D$

ما بعتبر term

single literal

ما بنحكي عنه term

لازم يكون 2 وأكثر

الأسلاك wires

* بنحسب ال gates

$F = BD + AB'C + A'C'D'$

$G = 11$, $GN = 14$

الأدق للاستخدام

minimum GN

least cost

No.

* Cost Criteria

ممكن نرسم ال Circuit وبنحسب عدد الأعلان

$$F(A,B,C,D) = (\underbrace{\overbrace{AB}^{2}C}_{1} + D') \cdot C'$$

$D' \rightarrow$ single

$C' \rightarrow$ single

تبسيط

* ممنوع نعمله Simplification

$$F(A,B,C) = \underbrace{AB}_{1} + \underbrace{\bar{B}C}_{2} + \overbrace{C \cdot (\underbrace{\bar{A} + \bar{B}}_{3})}^{4}$$

$L = 7$

$G = L + 4 = 11$

$GN = 13$

* ممكن بالامتحان تيجي الرسمة ونحسب ↑

* Karnaugh Maps (k-map)

2^n squaers → cells

$F(x, y)$ MSD ex

$n = 2$

$2^2 = 4$

$x \backslash y$	0	1
0	$m_0 = \bar{x}\bar{y}$	$m_1 = \bar{x}y$
1	$m_3 = x\bar{y}$	$m_4 = xy$

→ $F(x, y) = \Sigma_m (1, 3)$

اللي موجودة بتراكه
هو بكون 0

	1
	1

adjecent → 1-variable اختلاف بـ

1	1

$F(x,y) = \Sigma_m(2,3)$

$= x\bar{y} + xy$

$= x(\bar{y} + y)$

$G(a,b) = a + b$

$= \Sigma_n(1,2,3)$

$= \bar{x}y + x\bar{y} + xy$

$= \bar{x}y + x(\bar{y} + y)$

$= \bar{x}y + x$

$= x + \bar{x}y$

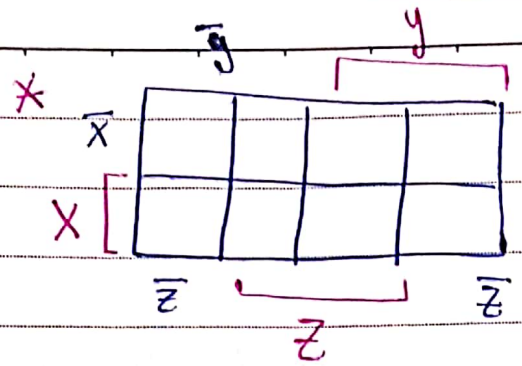
$= x + y$

a \ b	
	1
1	1

$\longrightarrow G(a,b) = a + b$

* Three Variable k-Maps → الخرائط

m_0 & m_2 → adjacent.



* ممكن من ال function نعملها squares ونعبيها.

* بقدر أجمع 2 , 4 , 8 أما 3,5,6.. لا

$2^{\square}$

power of 2

حاول تجمع أكبر عدد عشان يتبسط ال function.

* كل ما زادت عدد ال cells بيقل عدد ال variables

$F=1 \leftarrow L=0 \leftarrow$ cost بيقل.

ازا اتي جمعته من نفس النوع and

من نوعين مختلفين or

1	1	1	1
1	1		

* ال (٤) إللي على الزوايا ممكن نجمعهم مع بعض

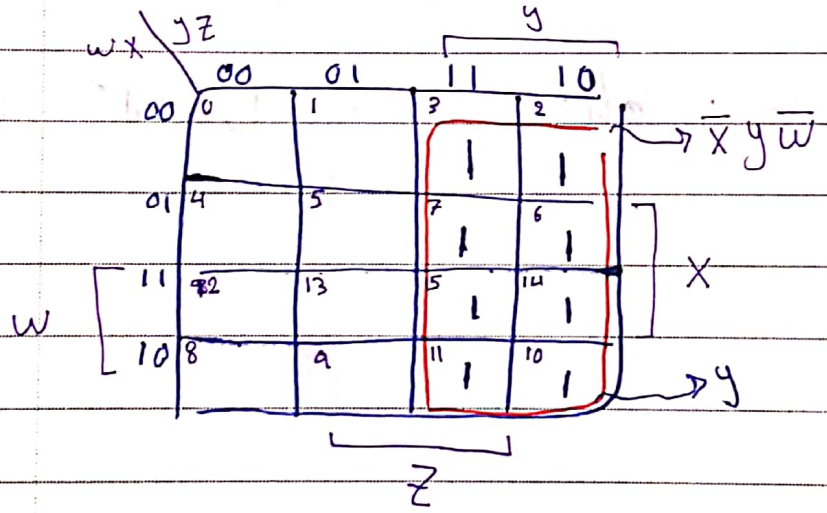
0 → 2 ✓

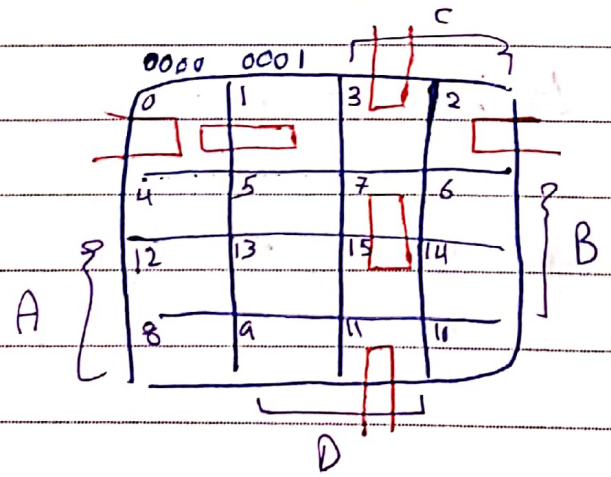
2 → 10 ✗

* بنختار ال minimum cost

* لازم نحط كل الاحتمالات إللي إلهم نفس الحجم

* F(w, x, y, z)





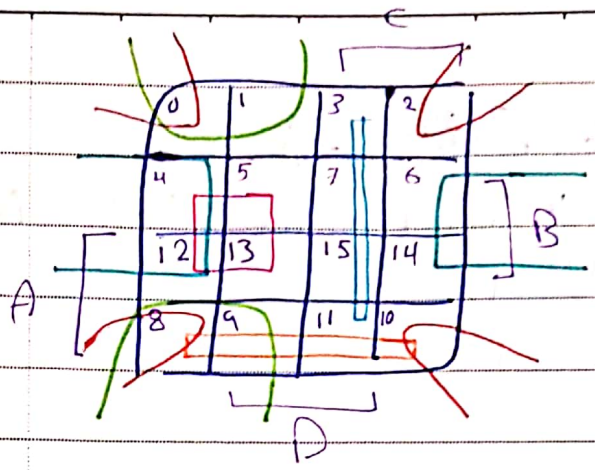
Rect(0,1) = $\bar{A}\bar{B}\bar{C}$

Rect(7,15) = BCD

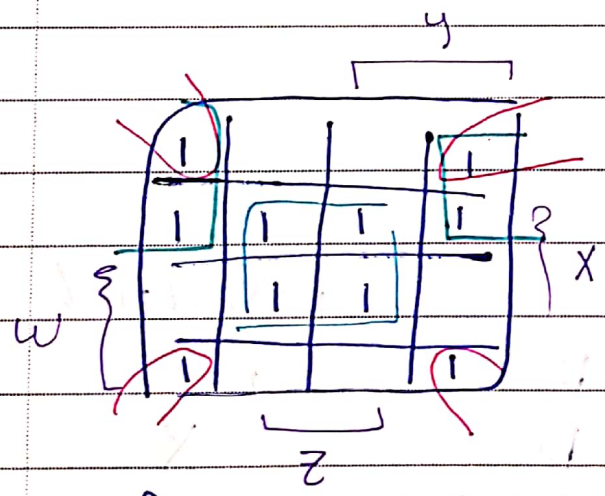
Rect(0,2) = $\bar{A}\bar{B}\bar{D}$

Rect(3,11) = $\bar{B}CD$

No.



* Rect (0, 2, 8, 10) = $\bar{B}\bar{D}$
* Rect (0, 1, 8, 9) = $\bar{B}\bar{C}$
* Rect (3, 7, 11, 15) = CD
* Rect (4, 6, 12, 14) = $B\bar{D}$
* Rect (4, 5, 12, 13) = $B\bar{C}$
* Rect (8, 9, 10, 11) = $A\bar{B}$



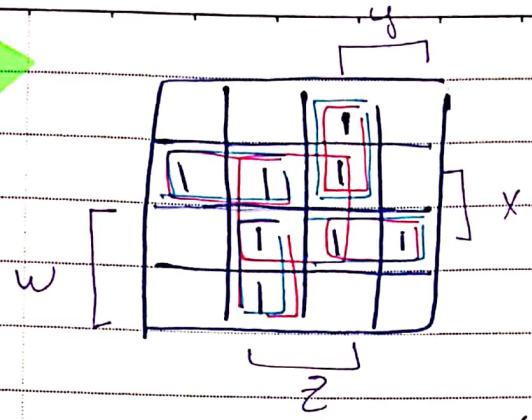
* Rect (0, 2, 8, 10) = $\bar{X}\bar{Z}$
* Rect (0, 2, 4, 6) = $\bar{W}\bar{Z}$
* Rect (5, 7, 13, 15) = XZ

* $F(W, X, Y, Z) = \bar{X}\bar{Z} + \bar{W}X + XZ$

$\rightarrow GN = 6 + 3 + 3 = 12$

$F(W, X, Y, Z) = \bar{X}\bar{Z} + \bar{W}\bar{Z} + XZ$

$\rightarrow GN = 6 + 3 + 3 = 12$



* $F(w, x, y, z) =$

$= \bar{w}x\bar{y} + \bar{w}yz + w\bar{y}z + wxy$

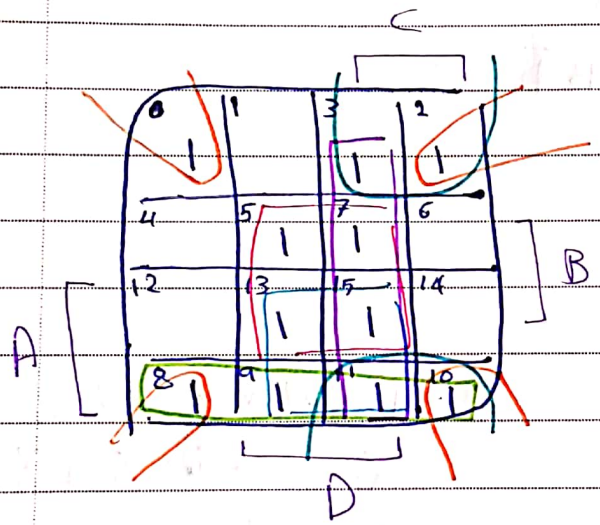
$GN = 4 + 12 + 2$

$= 18$

هون كانا ضفنا حد زيادة فالأولى أحسن cost

* $F(w, x, y, z) = xz +$

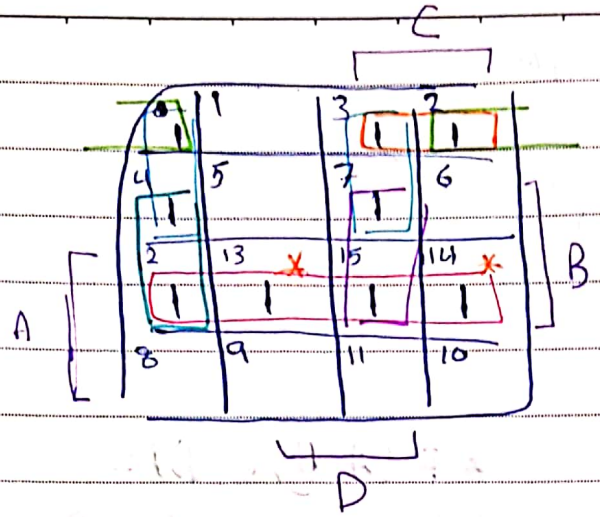
$18 + 2 = 20 \Leftarrow GN$



Essential

* Rect $(0, 2, 8, 10) = \bar{B}\bar{D}$ ✓
* Rect $(3, 7, 11, 15) = CD$ X
* Rect $(5, 7, 13, 15) = BD$ ✓
* Rect $(9, 11, 13, 15) = AD$ X
* Rect $(8, 9, 10, 11) = A\bar{B}$ X
* Rect $(2, 3, 10, 11) = \bar{B}C$ X

No.



	Essential
* Rect (0, 2) = $\bar{A}\bar{B}\bar{D}$	X
* Rect (0, 4) = $\bar{A}\bar{C}\bar{D}$	X
* Rect (3, 7) = $\bar{A}CD$	X
* Rect (12, 13, 14, 15) = AB	✓
* Rect (2, 3) = $\bar{A}\bar{B}C$	X
* Rect (7, 15) = BCD	X
* Rect (4, 12) = $B\bar{C}\bar{D}$	X

* لازم ال essential يكون مكتوب بال Function
* SOM → SOP
POS

لو بنصير نجمع ال (0s) يعني Maxterm

* قراءة المطلوب بالامتحان جيداً .

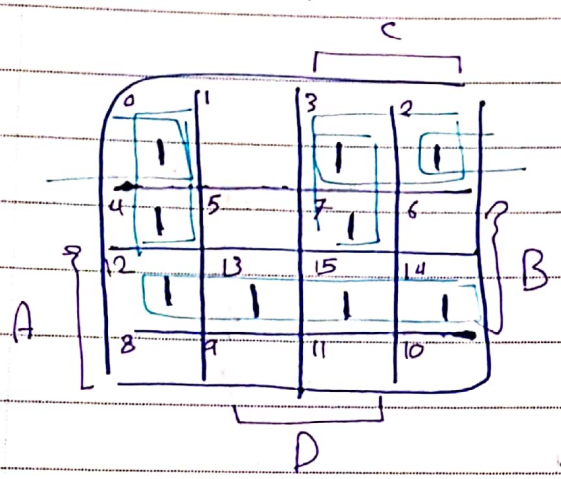
No.

* 7σ ⇒ 7 segma.

EEE OR No thing

ازا صار في ال error

سبب الخطأ



* $F(A,B,C,D) =$

$AB + \bar{A}\bar{C}\bar{D} + \bar{A}CD + \bar{A}\bar{B}C$

(12, 13, 14, 15) (0, 4) (3, 7) (2, 3)

essential

$GN = 11 + 4 + 4 = 19$

* $F(A,B,C,D) = AB + \bar{A}\bar{B}\bar{D} + \bar{A}CD + B\bar{C}\bar{D}$

(0, 2) (3, 7) (4, 12)

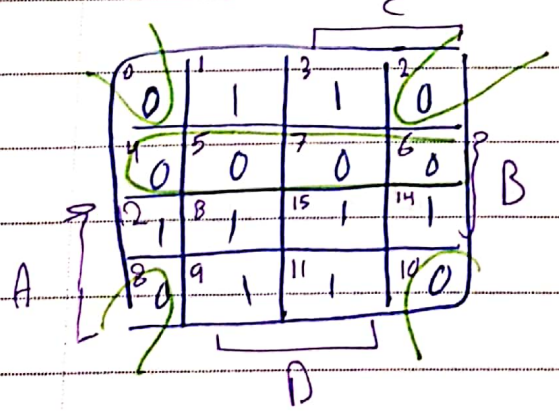
$GN = 11 + 4 + 4 = 19$

طالعلهم نفس الـ cost

بالصدفة فأي من الحلين يعتبر صح.

* POS :-

بنجمع الـ Zeros



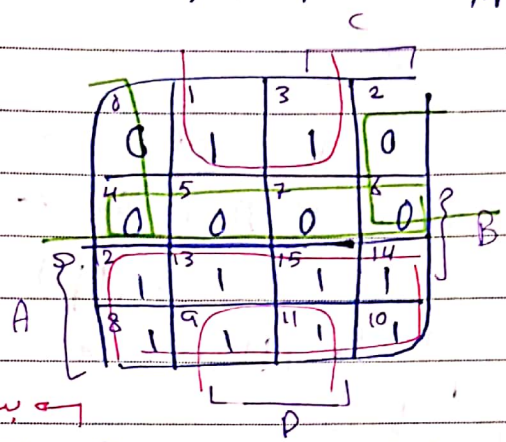
$F = \bar{A}B + \bar{B}\bar{D}$ → ازا حدا الهون كتب بياخد zero

$F = \bar{\bar{F}} = (A + \bar{B}) \cdot (B + D)$ ②

① ①

Scanned by CamScanner

$* \; F(A,B,C,D) = \prod_M (0,2,4,5,6,7)$



بدور على ال 1-s

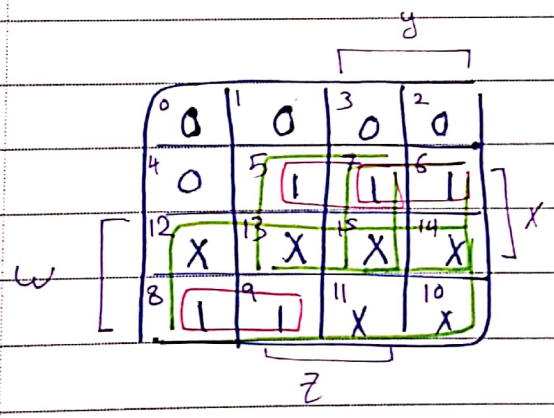
SOP: $F(A,B,C,D) = A + \bar{B}D$

① أوه اتي بكتب على الطريقة العادية بس على شرط تكون $\boxed{\bar{F}}$

بدور على 0-s ← POS: $\bar{F}(A,B,C,D) = \bar{A}\bar{D} + \bar{A}B$

② بعدين بعمل ل $\bar{F}$ complement $F = \bar{\bar{F}}$ وبطلع POS

POS: $\bar{\bar{F}} = F(A,B,C,D) = (A+D)\cdot(A+\bar{B})$



X: Don't Care

* بدي أرجع أكبر عدد من Ones فبستعين بال (X) عشان أقل ال cost

بدون الاستفادة من ال Don't care

$F(w,x,y,z) = w\bar{x}\bar{y} + \bar{w}xz + \bar{w}xy$

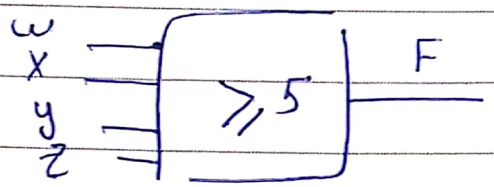
$GN = 9 + 3 + 3 = 15$

مع الاستفادة من ال Don't care

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

$GN = 5 + 2 = 7$

* فكرة عامة على المثال السابق



يعني لو كنت بدي تحت الـ (5) = 0

وفوق الـ (5) = 1

على نظام BCD بوقف الرقم

لـ [9] فأنا ما بهمني شو قيمة

ما بعد الـ [9] مثل 10, 11

فبحطهم Don't Care

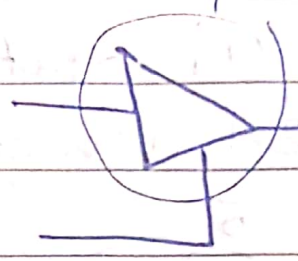
← وبفيدوني بتقليل الـ cost

Subject _____ الموضوع

Date: / / التاريخ:

Disable ← cuircit الـ

Don't care (X) ← ما يهمني شو صوته ← IN



OUT = Hi-Z

EN = 0 ←

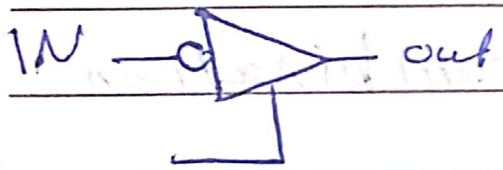
EN=1 → الـ cir بشتغل طبيعي

X buffer is enable

الـ enable is 1

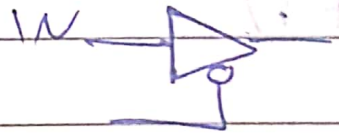
2-inputs

EN	IN	out
0	0	Hi-Z
0	1	Hi-Z
1	0	0
1	1	1

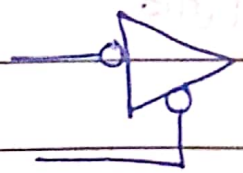


EN active high

الـ EN=0 بشتغل buffer



EN active low



active low

Subject ..... الموضوع

Date: / / التاريخ

2- tri state buffer

كم عدد الاحتمالات الي ممكن اتعامل ايها

5

2-1 2-0 Hi-Z

n- tri state buffers

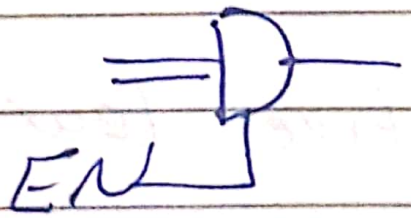
sharing ← وحدة من استخدامات في حالة

at least (n-1) buffer should be Hi-Z

وكانه لو كلهم كانوا Hi-Z

* how many valid out combination

(2n+1)



ال tri state
هو محصور بالـ buffer
بس التركيز حالة

حالة ي gate بدنا اياها.

Tri-State Buffer (3-State Buffer)

- For the symbol and truth table, IN is the **data input**, and EN is the **control input**

دلوقتي

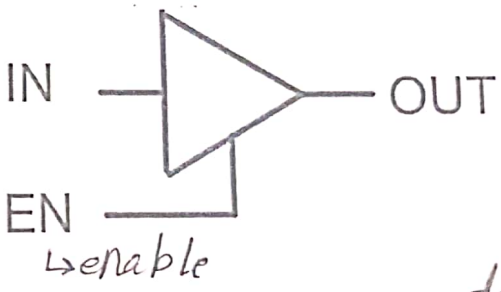
بغض النظر

- For EN = 0, regardless of the value on IN (denoted by X), the output value is Hi-Z

هي اللي بتتحكم بالـ switch

- For EN = 1, the output value follows the input value

Symbol



Truth Table

don't care

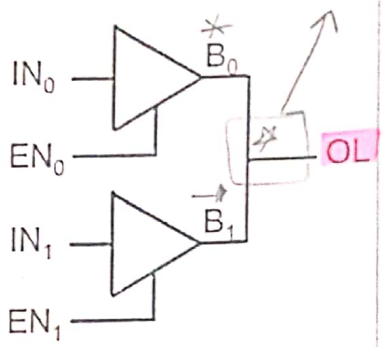
EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

Resolving 3-State Values on a Connection

- Connection of two tri-state buffer outputs, B_1 and B_0, to a wire, OL (Output Line) → Multiplexed Output

① هو في عملية معينة و لا ؟

EN_1	EN_0	IN_1	IN_0	B_1	B_0	OL
0	0	X	X	Hi-z	Hi-z	Hi-z
0	1	X	0	Hi-z	0	0
0	1	X	1	Hi-z	1	1
1	0	0	X	0	Hi-z	0
1	0	1	X	1	Hi-z	1
1	1	0	0	0	0	0
1	1	1	1	1	1	1
1	1	0	1	0	1	
1	1	1	0	1	0	



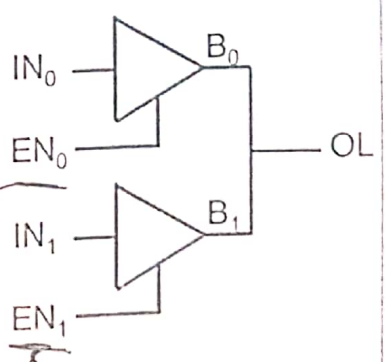
Hi-Z ~ Hi-Z → Hi-Z
Hi-Z ~ 0 → 0
Hi-Z ~ 1 → 1
0 ~ 0 → 0
1 ~ 1 → 1
0 ~ 1
1 ~ 0] → fire

Resolving 3-State Values on a Conne

- Connection of two tri-state buffer outputs, B_1 an
wire, OL (Output Line) → Multiplexed Output

truth table الحالات

EN_1	EN_0	IN_1	IN_0	B_1	B_0	OL
0	0	X	X	Hi-Z	Hi-Z	Hi-Z
0	1	X	0	Hi-Z	0	0
0	1	X	1	Hi-Z	1	1
1	0	0	X	0	Hi-Z	0
1	0	1	X	1	Hi-Z	1
1	1	0	0	0	0	0
1	1	1	1	1	1	1
1	1	0	1	0	1	Fire
1	1	1	0	1	0	Fire



ما يكونوا ال EN يشتغلوا مع بعض

يعني ما بيي يشتغلوا مع بعض

= [* at least (n-1) must be in Hi-Z
* at most one is active

Resolving 3-State Values on a Connection

output → 0
→ Hi-Z

$\boxed{2n+1}$ = عدد ال valid connection

- **Resulting Rule:** At least one buffer output value must be Hi-Z. Why?
  - Because any data combinations including (0,1) and (1,0) can occur. If one of these combinations occurs, and no buffers are Hi-Z, then high currents can occur, destroying or damaging the circuit

  لا يكونوا مفعلين طافين

- **How many valid buffer output combinations exist?**
  - 5 valid output combination

- **What is the rule for "n" tri-state buffers connected to wire OL?**

  EN = 0

  مثلاً عندي 4 buffer
  4 - 1 = 3 - Hi-Z
  0 = 3 EN
  يعني عندنا ما يمكن أكثر من واحد يشتغل مع بعض

  - At least "n-1" buffer outputs must be Hi-Z
  - **How many valid buffer output combinations exist ?** !?
    - Each of the n-buffers can have a 0 or 1 output with all others at Hi-Z. Also all buffers can be Hi-Z. So there are 2n + 1 valid combinations.

Data-Selector

* مثال بدي أعرف أنه EN_0 و EN_1 ما يكونوا active at the same time

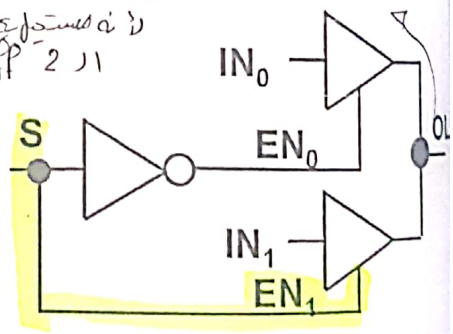
Tri-State Logic Circuit

- **Data Selection Function:** If s = 0, OL = IN_0, else OL = IN_1
- Performing data selection with tri-state buffers:

S

EN_1	EN_0	IN_1	IN_0	OL
0	1	X	0	0
0	1	X	1	1
1	0	0	X	0
1	0	1	X	1

مستحيل يطلع عنا Hi-Z
لأنه مستحيل يكونوا ال 2 off



- Since $EN_0 = \bar{s}$ and $EN_1 = s$, one of the two buffer outputs is always Hi-Z.

one of them will be active

S IN_0 IN_1

Logic Functions using Tri-State Buffers

نفس السؤال

- Implement AND gate using 3-State buffers and inverters

$$F(X,Y) = X.Y$$

- Use X as control input:
  - When $X = 0, F = 0$ regardless of the value of Y
  - When $X = 1, F = Y$

مستعمل بناءً على X

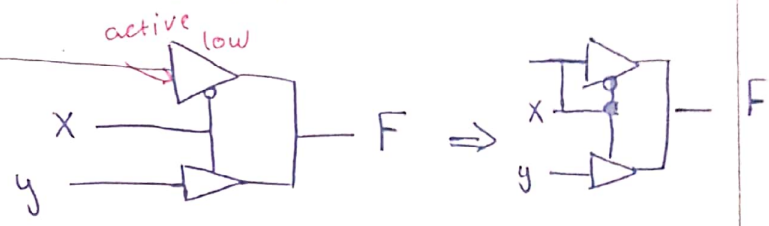
EN

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

* نستخدم الـ EN بالعادة most significant variable

إلي هو هون X

يا بشبك عليها X أو ن



دايماً منطقوا مع

بحث ②

Logic Functions using Tri-State Buffers

- Implement AND gate using 3-State buffers and inverters

$$F(X,Y) = X.Y$$

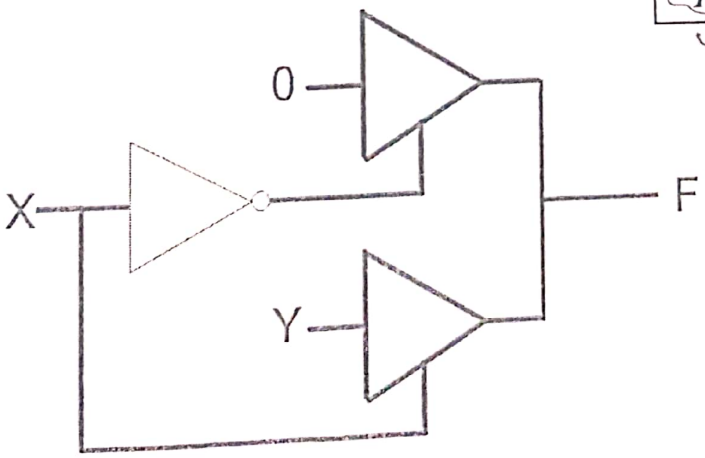
- Use X as control input:
  - When $X = 0, F = 0$ regardless of the value of Y
  - When $X = 1, F = Y$

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

F = 0
F = X

F = y

كأنها AND gate



Logic Functions using Tri-State Buffers

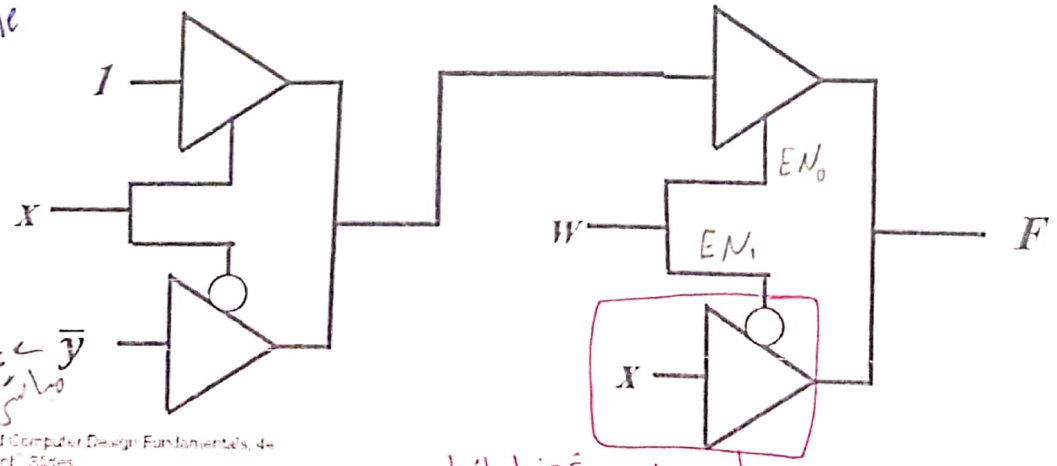
غط السؤال

- Implement the following function using 3-State buffers and inverters: $F(w,x,y) = \overline{w}x + w\overline{y} + xy$

طريقة الربط

- Use w as control input:
  - When $w = 0$, $F = x$ regardless of the value of Y
  - When $w = 1$
    - If $x = 0$, $F = \overline{y}$
    - If $x = 1$, $F = 1$

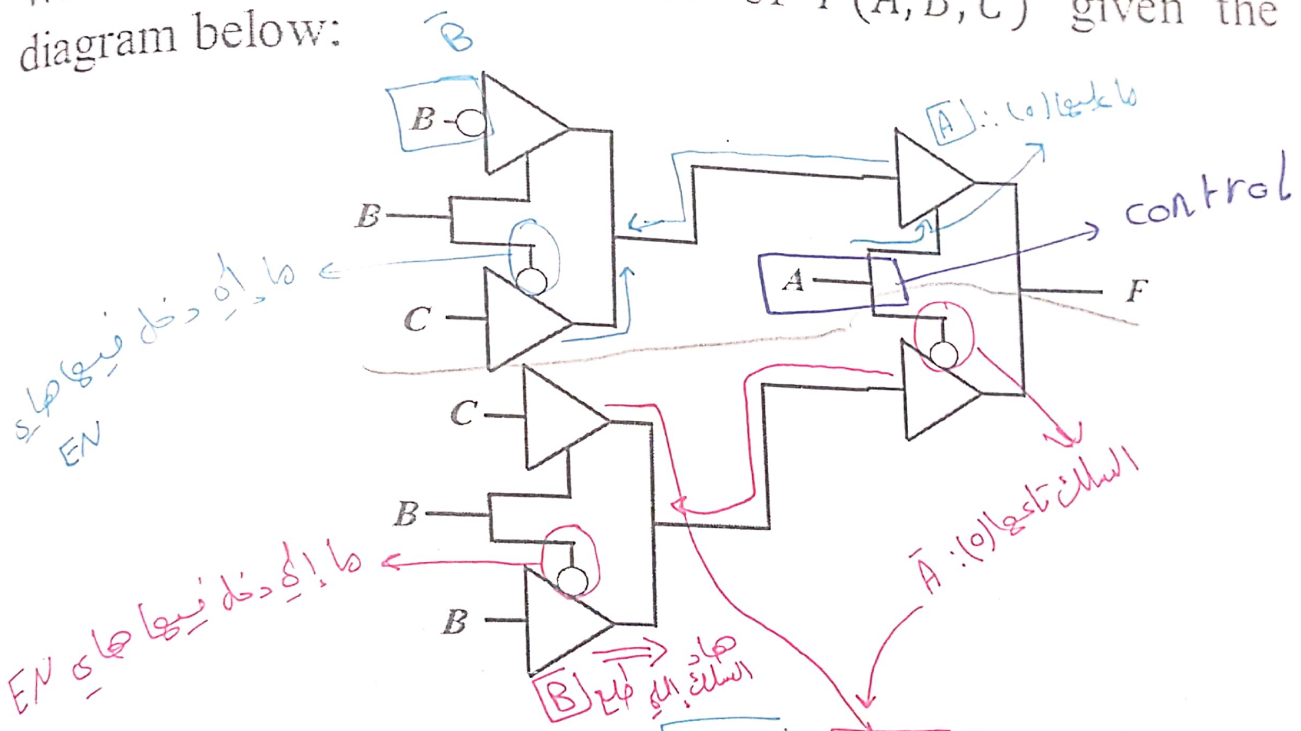
w	x	y	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



هاد بيشتغل إذا كانت قيمة $EN_1 = 1$

Logic Functions using Tri-State Buffers

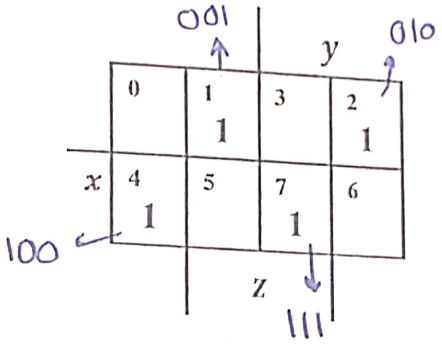
- Write the Boolean expression of $F(A,B,C)$ given the diagram below:

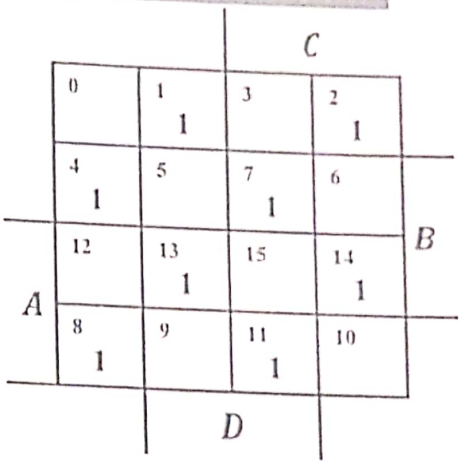


$$F(A,B,C) = \boxed{A\bar{B}C} + \boxed{\bar{A}BC}$$

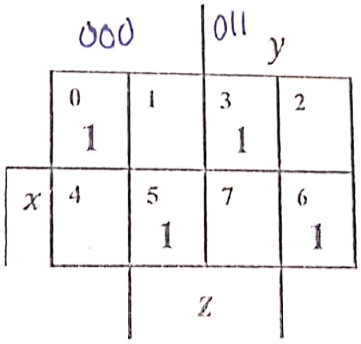
Odd and Even Functions

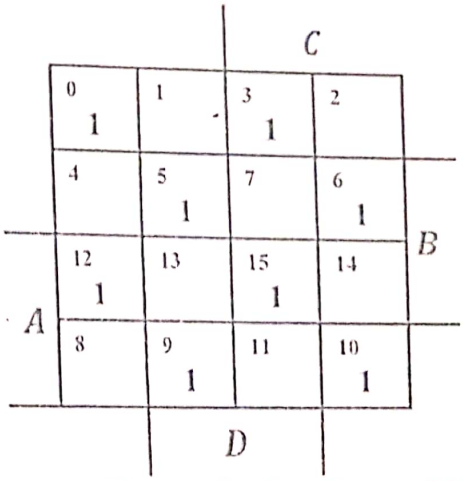
- The 1s of an ***odd function*** correspond to minterms having an index with an odd number of 1s.





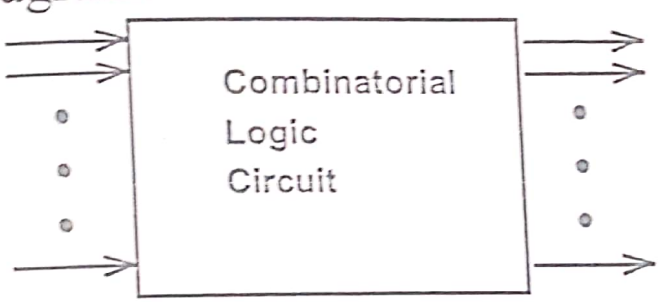
- The 1s of an ***even function*** correspond to minterms having an index with an even number of 1s.





Combinational Circuits

- A combinational logic circuit has:
  - A set of m Boolean inputs,
  - A set of n Boolean outputs, and
  - n switching functions, each mapping the 2^m input combinations to an output such that the *current output depends only on the current input values*
- A block diagram:



m Boolean Inputs n Boolean Outputs

Design Procedure

تحليل المواصفات

1. **Specification**
   - Write a specification for the circuit if one is not already available. *What does the circuit do? Including names or symbols for inputs and outputs*
2. **Formulation**
   - Derive a *truth table* or *initial Boolean equations* that define the required relationships between the inputs and outputs, if not in the specification

   imp ⇒ استنتاج
3. **Optimization**
   - Apply 2-level optimization using K-maps
   - Draw a logic diagram for the resulting circuit using ANDs, ORs, and inverters

Design Procedure

4. **Technology Mapping**
   - Map the logic diagram to the implementation technology selected

5. **Verification**
   - Verify the correctness of the final design manually or using simulation

Design Example1

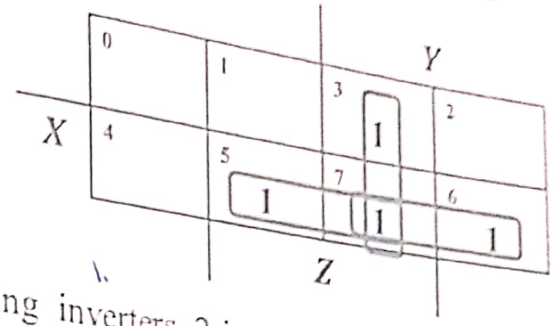
- **Specification:** Design a combinational circuit that has 3 *inputs (X, Y, Z)* and one output *F*, such that $F = 1$ when the number of 1's in the input is greater than the number of 0's (i.e. number of 1's ≥ 2)
  - This is called *majority function* (i.e. majority of inputs must be 1 for the function to be 1) أو كالأغلبية
- **Formulation:** zero تكون أكثر بيطلع

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

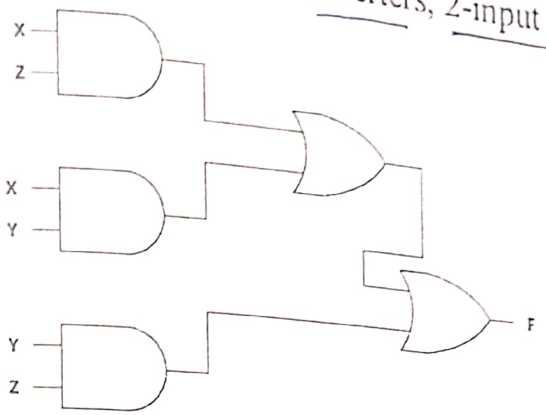
Design Example1 Cont.

- Optimization:

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



- Technology Mapping:
  - Mapping with a library containing inverters, 2-input AND, 2-input OR



Design Example2

O_0 إلي على الشمال *
O_1 إلي على اليمين

O_0 الجهة اليمنى تمثل *
O_1 " اليسار "

- **Specification:**-Design a combinational circuit that compares *2-bit* Binary number *(A, B)* and produce two outputs *(O_1, O_0)*, such that:

مجموعة الهف بيعطينا رقم

$O_1O_0 = 00$	When $A = B$ *and Both are even*
$O_1O_0 = 01$	When $A < B$
$O_1O_0 = 10$	When $A > B$
$O_1O_0 = 11$	When $A = B$ *and Both are odd*

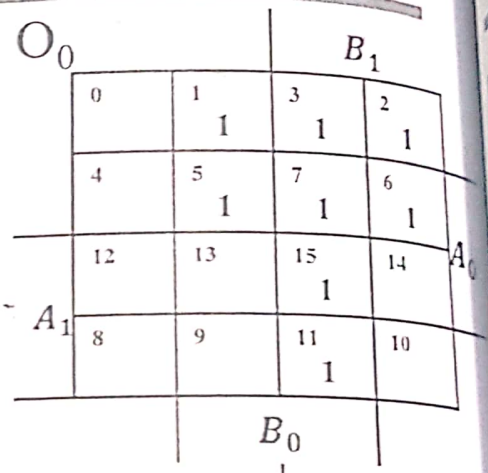
	$A(A_1A_0)$	$B(B_1B_0)$	$O(O_1O_0)$	
0	00	00	00	
1	00	01	01	B>A
2	00	10	01	
3	00	11	01	
4	01	00	10	A>B
5	01	01	11	A=B
6	01	10	01	
7	01	11	01	
8	10	00	10	
9	10	01	10	
10	10	10	00	
11	10	11	01	
12	11	00	10	
13	11	01	10	
14	11	10	10	
15	11	11	11	

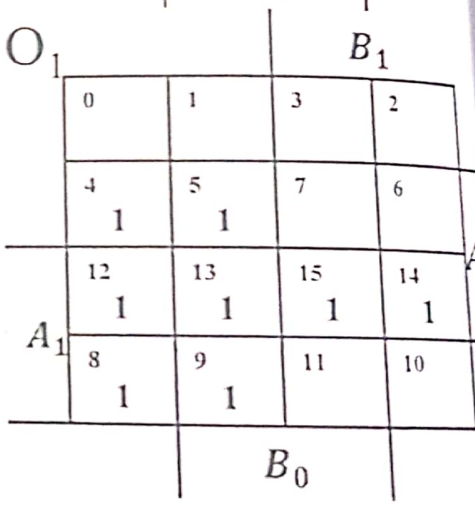
- **Formulation:**

Design Example2 Cont.

- **Optimization and Technology Mapping:**

16 * 2 = 32 → عشان هيك اعملنا
2-K map

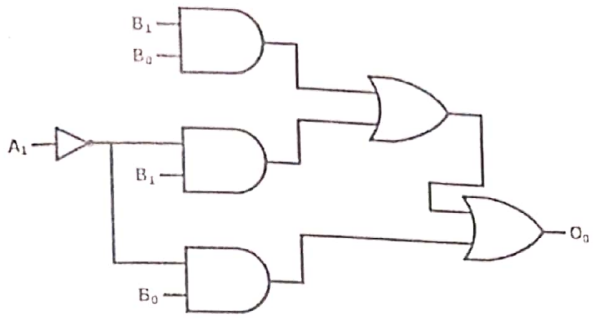


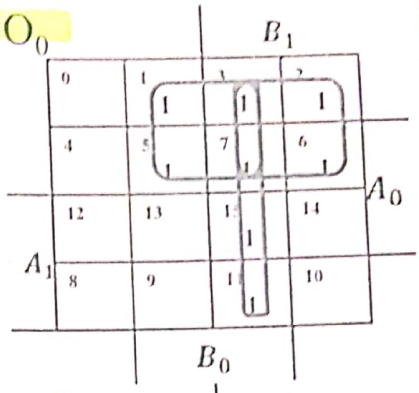


Design Example2 Cont.

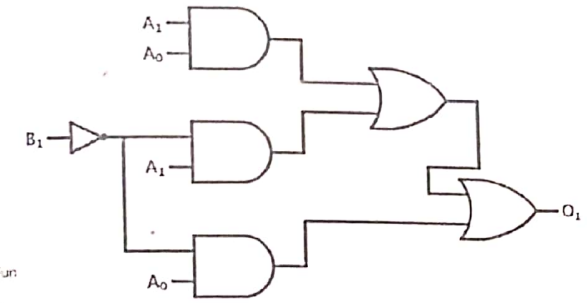
- Optimization and Technology Mapping:

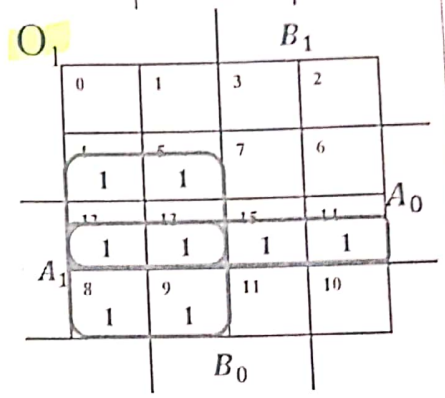
$O_0 = B_1B_0 + \overline{A_1}B_1 + \overline{A_1}B_0$





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





Design Example3

1. **Specification**
   - *BCD to Excess-3 code converter*
   - Transforms BCD code for the decimal digits to Excess-3 code for the decimal digits
   - BCD code words for digits 0 through 9: 4-bit patterns 0000 to 1001, respectively
   - Excess-3 code words for digits 0 through 9: 4-bit patterns consisting of 3 (binary 0011) added to each BCD code word
   - *BCD input is labeled A, B, C, D*
   - *Excess-3 output is labeled W, X, Y, Z*

Design Example3 Cont.

2. Formulation

ABCD	*WXYZ*
0000	0011
0001	0100
0010	0101
0011	0110
0100	0111
0101	1000
0110	1001
0111	1010
1000	1011
1001	1100
1010	XXXX
1011	XXXX
1100	XXXX
1101	XXXX
1110	XXXX
1111	XXXX

BCD = 9 1001
+3
⇒ Excess 3 = 12 1100

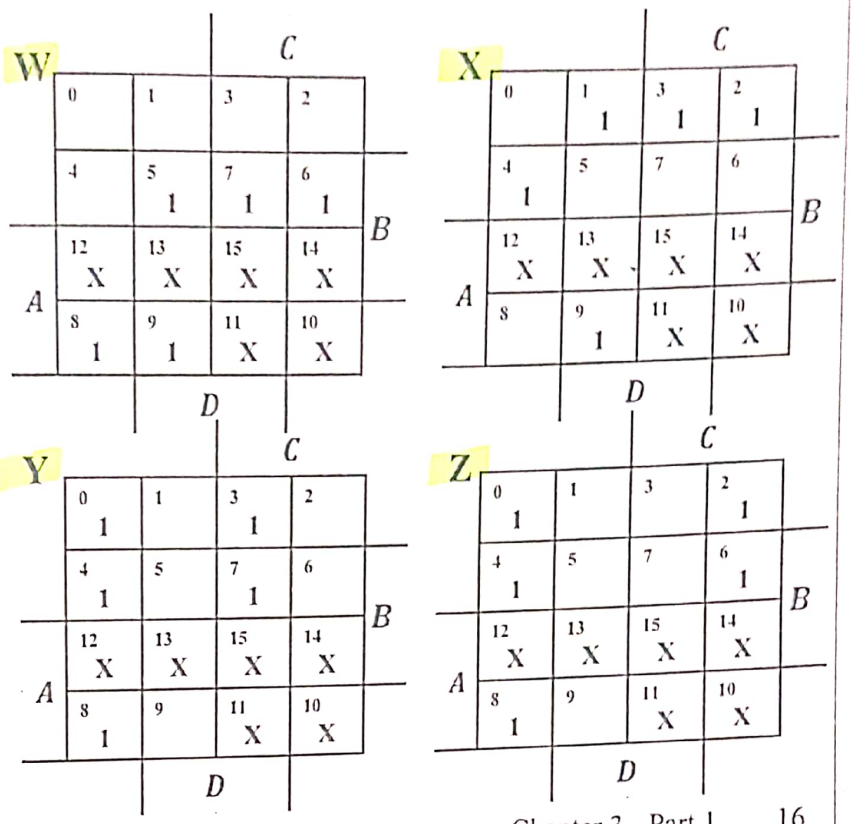
9 <

don't care

↑ 16 * 4 = 64
al bits ↓
4-Karnaugh ∴

Design Example3 Cont.

3. Optimization



Design Example3 Cont.

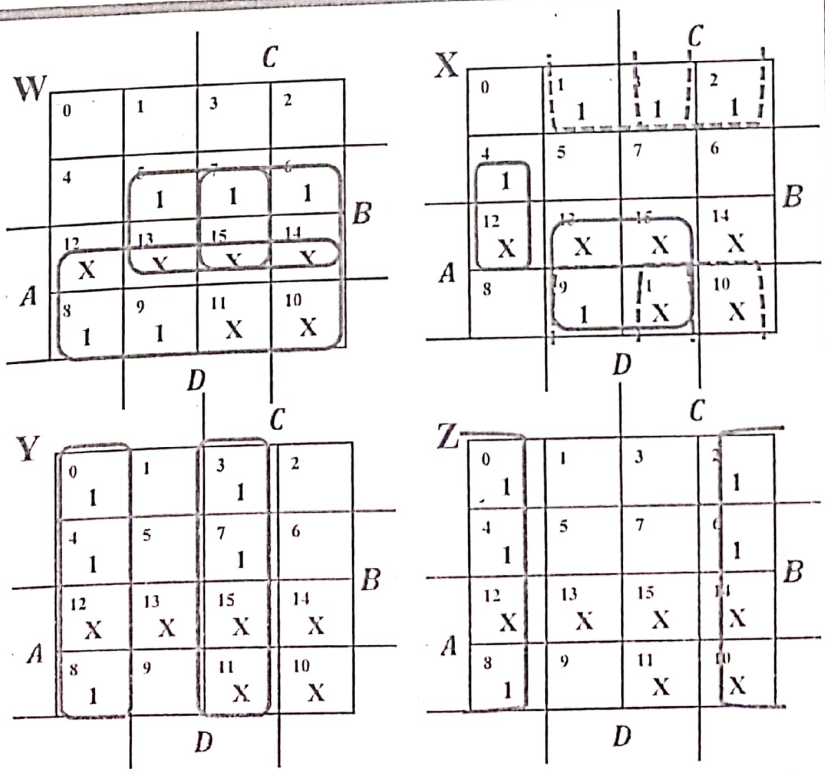
3. Optimization

$W = A + BC + BD$

$X = \bar{B}D + \bar{B}C + B\bar{C}\bar{D}$

$Y = \bar{C}\bar{D} + CD$

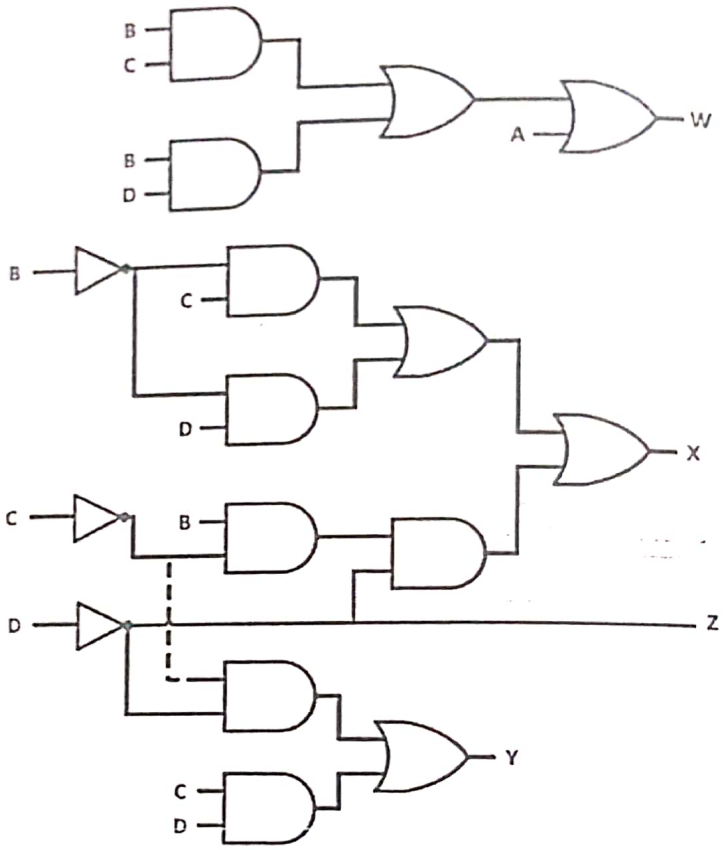
$Z = \bar{D}$



Design Example3 Cont.

4. Technology Mapping

- Mapping with a library containing inverters, 2-input AND, 2-input OR



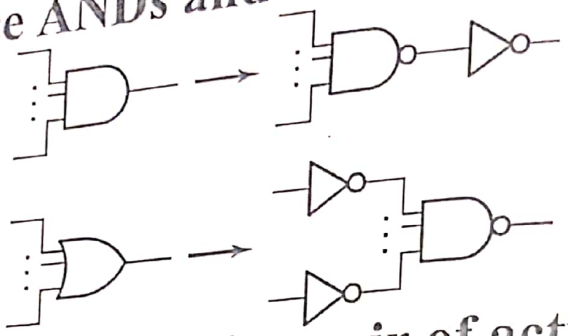
Mapping to NAND gates

- Assumptions:
  - Gate loading and delay are ignored
  - Cell library contains an inverter and n-input NAND gates, $n = 2, 3, \ldots$
  - An AND, OR, inverter schematic for the circuit is available
- The mapping is accomplished by:
  - Replacing AND and OR symbols,
  - Pushing inverters through circuit fan-out points, and
  - Canceling inverter pairs

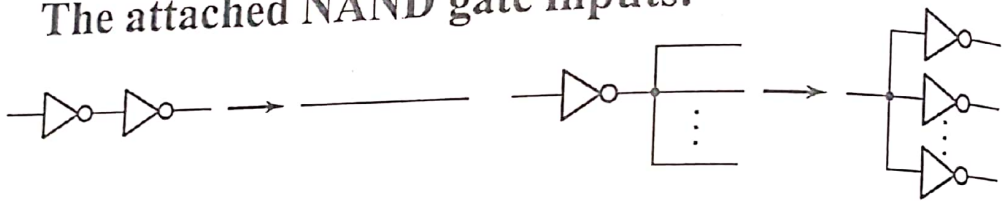
بدنا نحاول نقلل الـ inverters عشان costly

NAND Mapping Algorithm

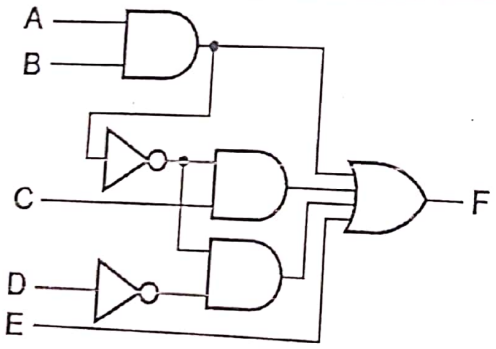
1. Replace ANDs and ORs:



2. Repeat the following pair of actions until there is at most one inverter between :
   a. A circuit input or driving NAND gate output, and
   b. The attached NAND gate inputs.

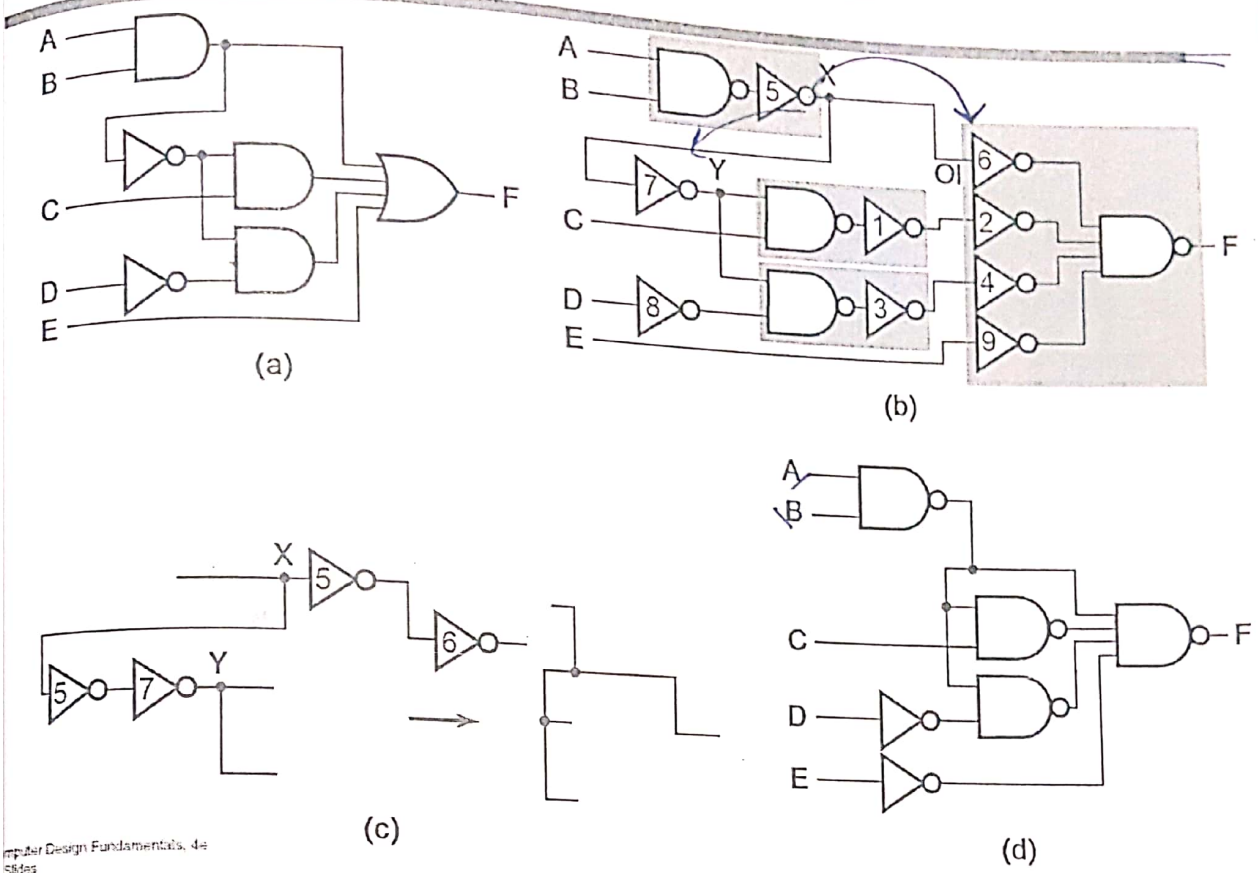


NAND Mapping Example



(a)

NAND Mapping Example

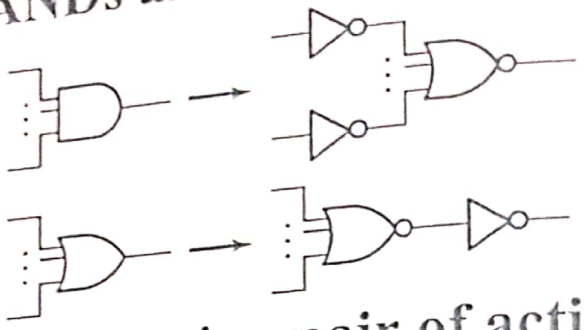


Mapping to NOR gates

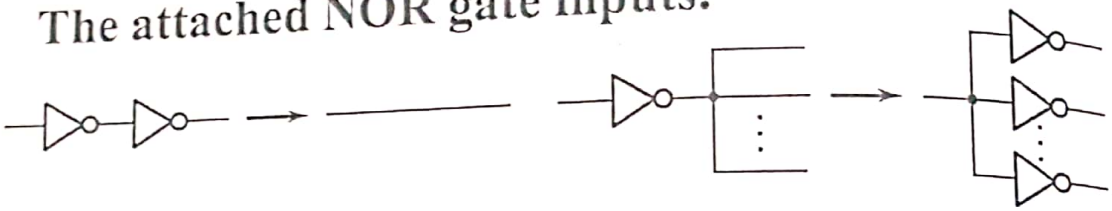
- **Assumptions:**
  - Gate loading and delay are ignored
  - Cell library contains an inverter and *n*-input NOR gates, $n = 2, 3, \ldots$
  - An AND, OR, inverter schematic for the circuit is available
- **The mapping is accomplished by:**
  - Replacing AND and OR symbols,
  - Pushing inverters through circuit fan-out points, and
  - Canceling inverter pairs

NOR Mapping Algorithm

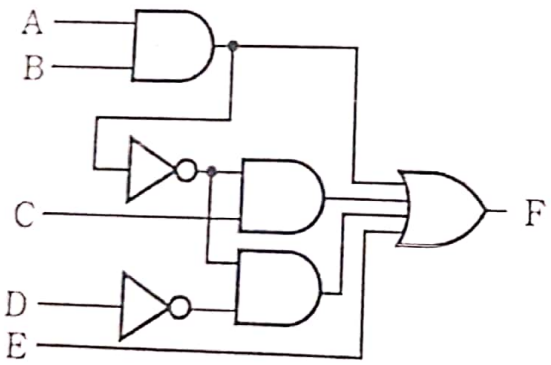
1. Replace ANDs and ORs:



2. Repeat the following pair of actions until there is at most one inverter between :
   a. A circuit input or driving NOR gate output, and
   b. The attached NOR gate inputs.

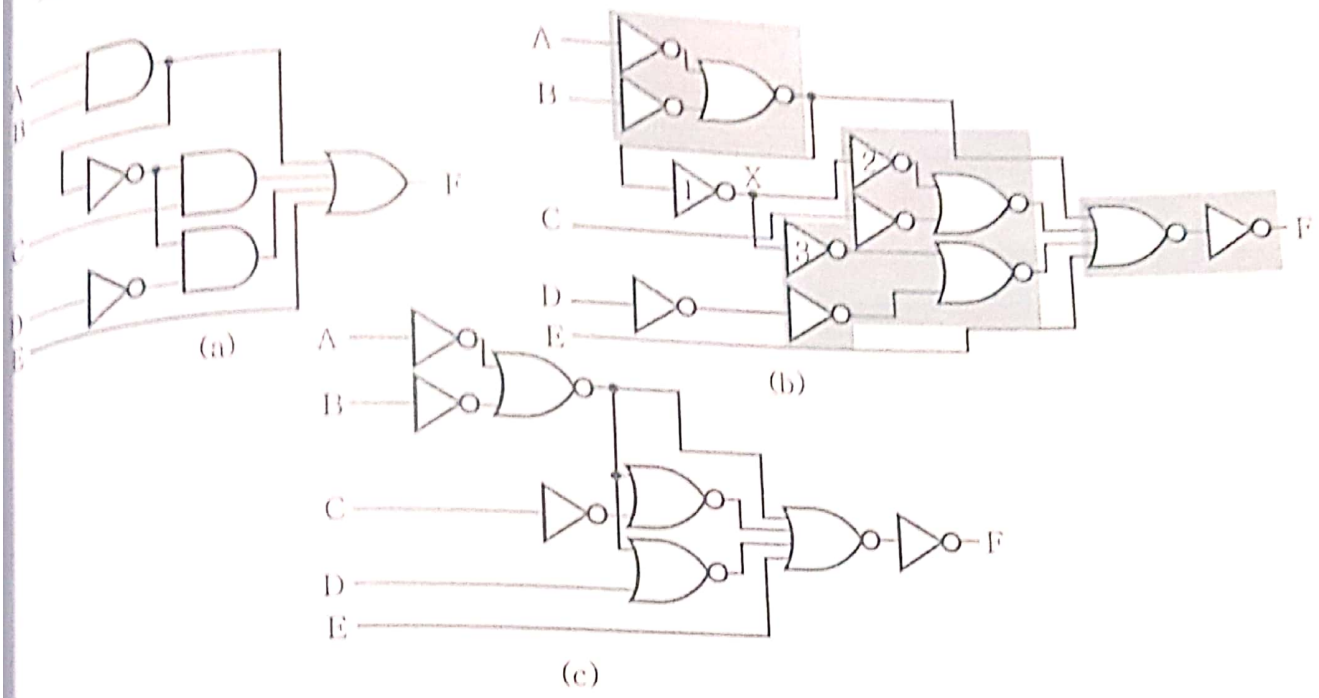


NOR Mapping Example

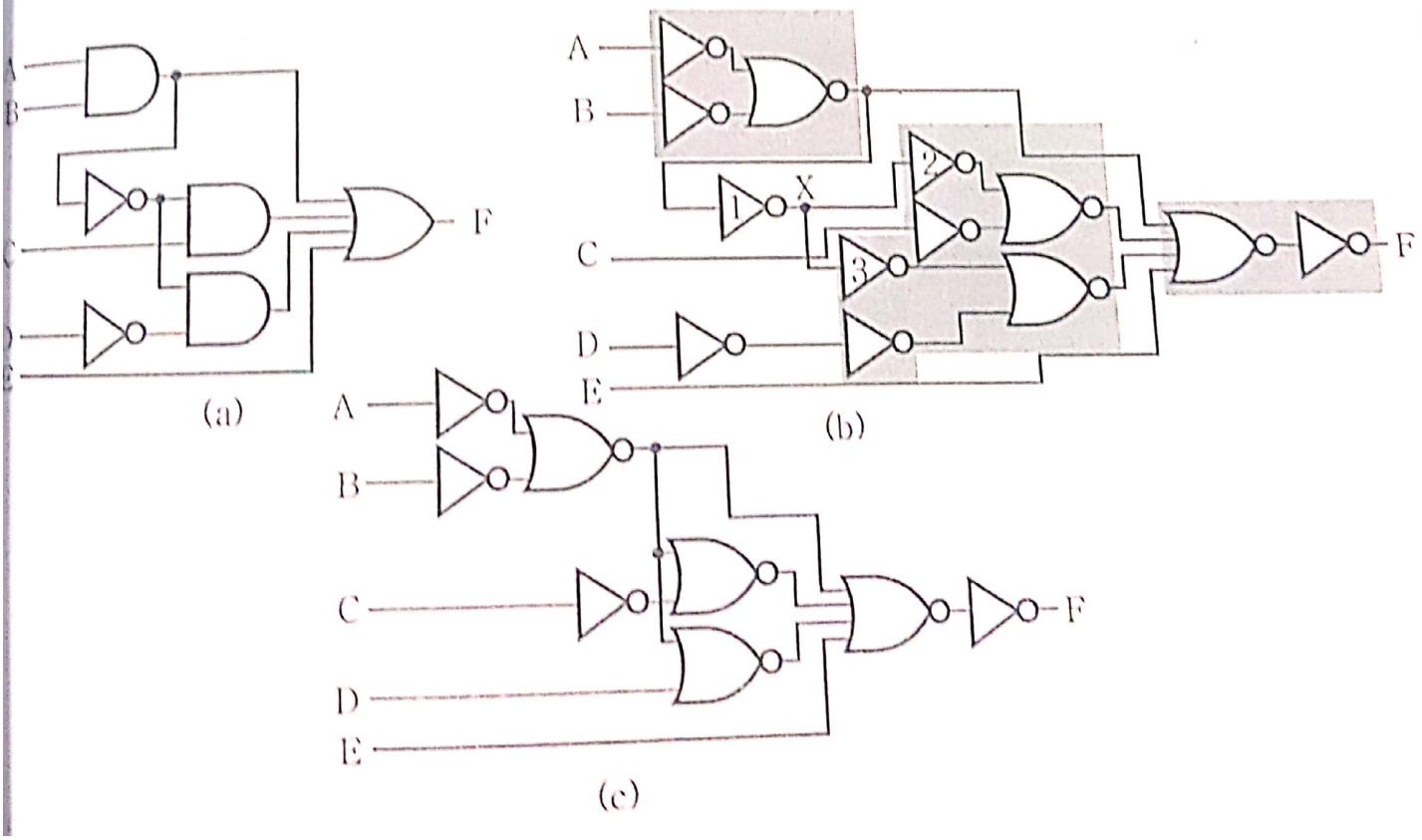


(a)

NOR Mapping Example



NOR Mapping Example



Functions and Functional Blocks

- The functions considered are those found to be very useful in design
- Corresponding to each of the functions is a combinational circuit implementation called a *functional block*
- In the past, functional blocks were packaged as small-scale-integrated (SSI), medium-scale-integrated (MSI), and large-scale-integrated (LSI) circuits
- Today, they are often simply implemented within a very-large-scale-integrated (VLSI) circuit

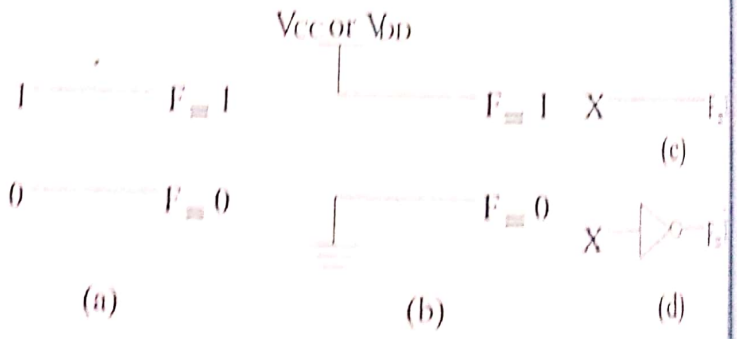
Rudimentary Logic Functions

- Functions of a single variable X → OR zero
- Can be used on the inputs to functional blocks to implement other than the block's intended function

Functions of One Variable				
X	$F = 0$	$F = 1$	$F = X$	$F = \bar{X}$
0	0	1	0	1
1	0	1	1	0

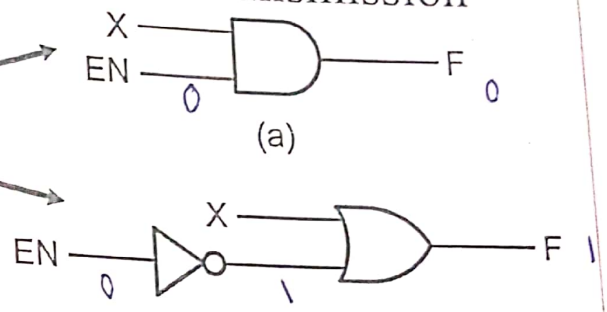
value fixing

- ***Value fixing*** *: a, b*
- ***Transferring*** *: c*
- ***Inverting*** *: d*
- ***Enabling*** *: next slide*



Enabling Function

- ***Enabling*** permits an input signal to pass through to an output
- ***Disabling*** blocks an input signal from passing through to an output, replacing it with a fixed value
- The value on the output when it is disable can be ***Hi-Z*** (as for three-state buffers and transmission gates), 0 , or 1
- When disabled, *0 output*
- When disabled, *1 output*



Decoding

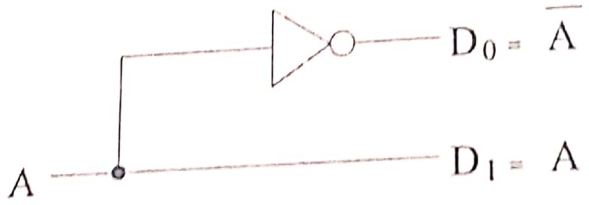
- ***Decoding:*** the conversion of an *n-bit* input code to an *m-bit* output code with $n \le m \le 2^n$ such that each valid code word produces a unique output code
- Circuits that perform decoding are called *decoders*
- Functional blocks for decoding are
  - called *n-to-m line decoders*, where $m \le 2^n$, and
  - generate 2^n (or fewer) minterms for the *n* input variables

1-to-2 Line Decoder

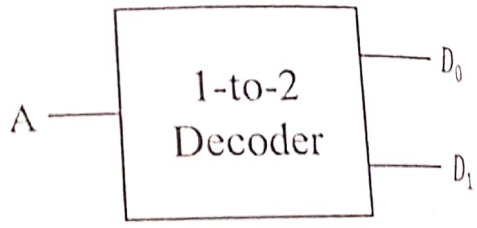
- When the decimal value of A equals the subscript of D_i, that D_i will be 1 and all others will be 0's
- Only one output is active at a time

A	D_0	D_1
0	1	0
1	0	1

(a)



(b)



(c)

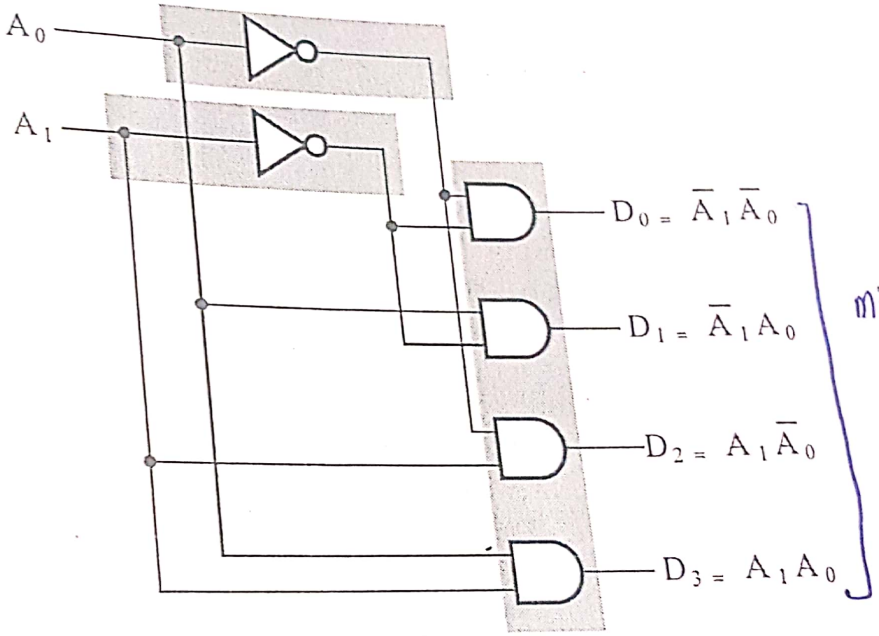
- Decoders are used to control multiple circuits by enabling only one of them at a time

2-to-4 Line Decoder

2-to-3 لو

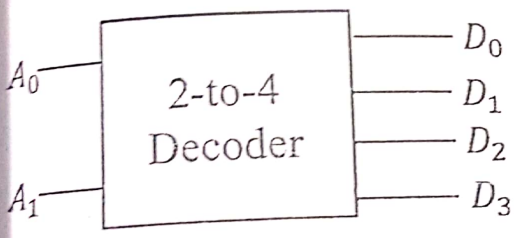
A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)



(b)

مشتقوا minterms



(c)

- No more optimization is possible
- *Note that the 2-to-4 line decoder is made up of* two 1-to-2- *line decoders and 4 AND gates*

Decoder Expansion

توسيع

n: inputs
2^n: outputs

- General procedure given in book for any decoder with *n inputs and 2^n outputs*

إجراء

- This procedure builds a decoder backward from the outputs using

1. Let k = n

2. We need 2^k 2-input AND gates driven as follows:
   - If k is even, drive the gates using two k/2-to-$2^{k/2}$ decoders
   - If k is odd, drive the gates using one (k+1)/2-to-$2^{(k+1)/2}$ decoder and one (k-1)/2-to-$2^{(k-1)/2}$ decoder

3. For each decoder resulting from step2, repeat step2 until k = 1. For k = 1, use 1-to-2 decoder

Decoder Expansion - Example 1

$2^3 = 8$

- 3-to-8-line decoder
  - $k = n = 3$
  - We need $2^3(8)$ 2-input AND gates driven as follows:
  - k is odd, so split to:
    - 2-to-4-line decoder
    - 1-to-2-line decoder

$k=3$

  - 2-to-4-line decoder → $k = n = 2$
    - We need $2^2(4)$ 2-input AND gates driven as follows:
    - k is even, so split to:
      - Two 1-to-2-line decoder

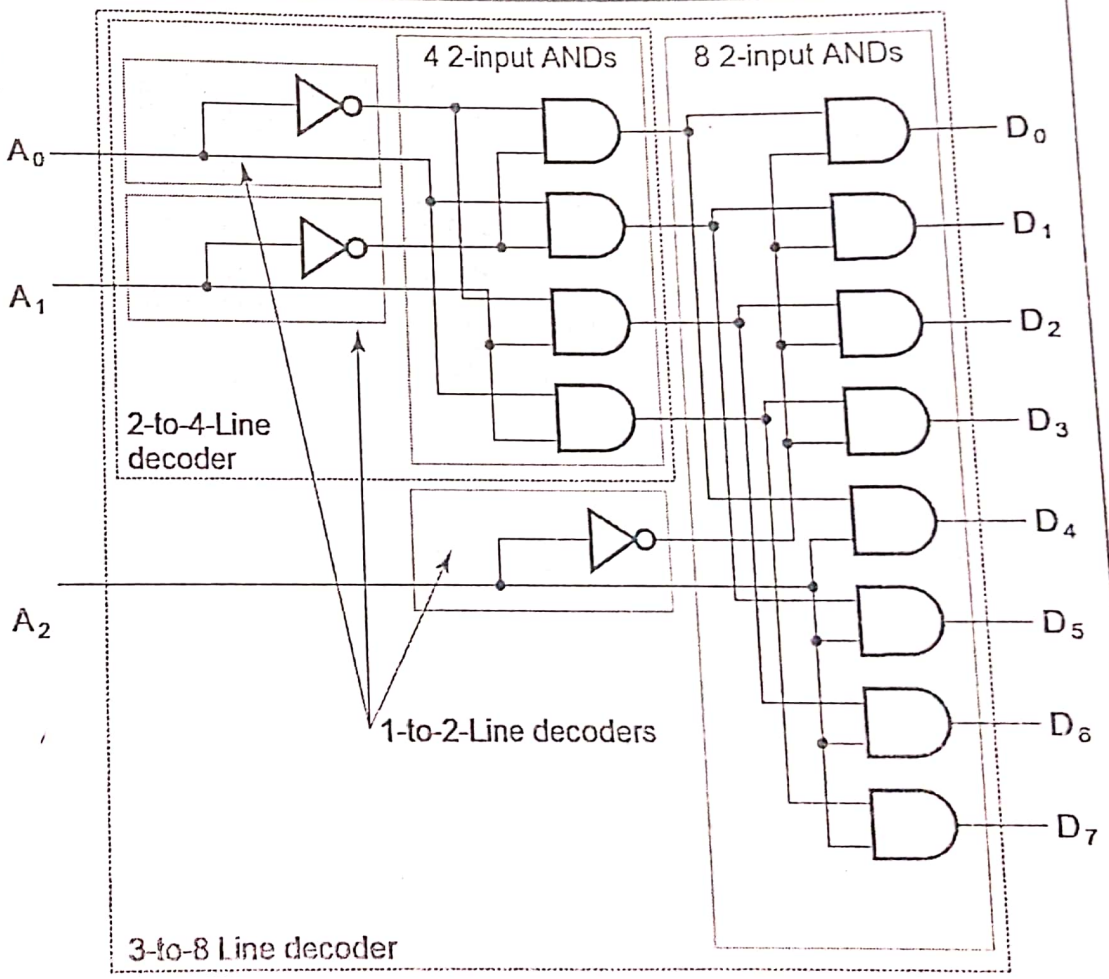
- See next slide for result

Decoder Expansion - Example 1

$GN = 8 \times 2 + 4 \times 2 + 3$

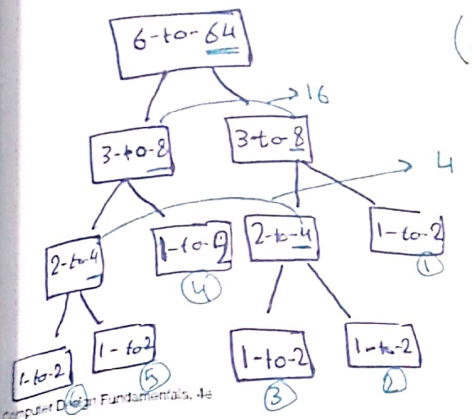
$GN = 27$

Straight forward design has the same GN cost



Decoder Expansion - Example 2

- 6-to-64-line decoder $2^6 = 64$
  - $k = n = 6$
  - We need 2^6(64) 2-input AND gates driven as follows:
  - k is even, so split to:
    - Two 3-to-8-line decoders
  - Each 3-to-8-line decoder is designed as shown in Example 1



$(64 \times 2) + (16 \times 2) + (4 \times 2) + 6$

$128 + 32 + 8 + 6 = 182$

Decoder Expansion - Example 2

$GN = 64 \times 2 + 16 \times 2 + 8 \times 2 + 6$

$GN = 182$

Straight forward design has GN cost of 390

IC. 1

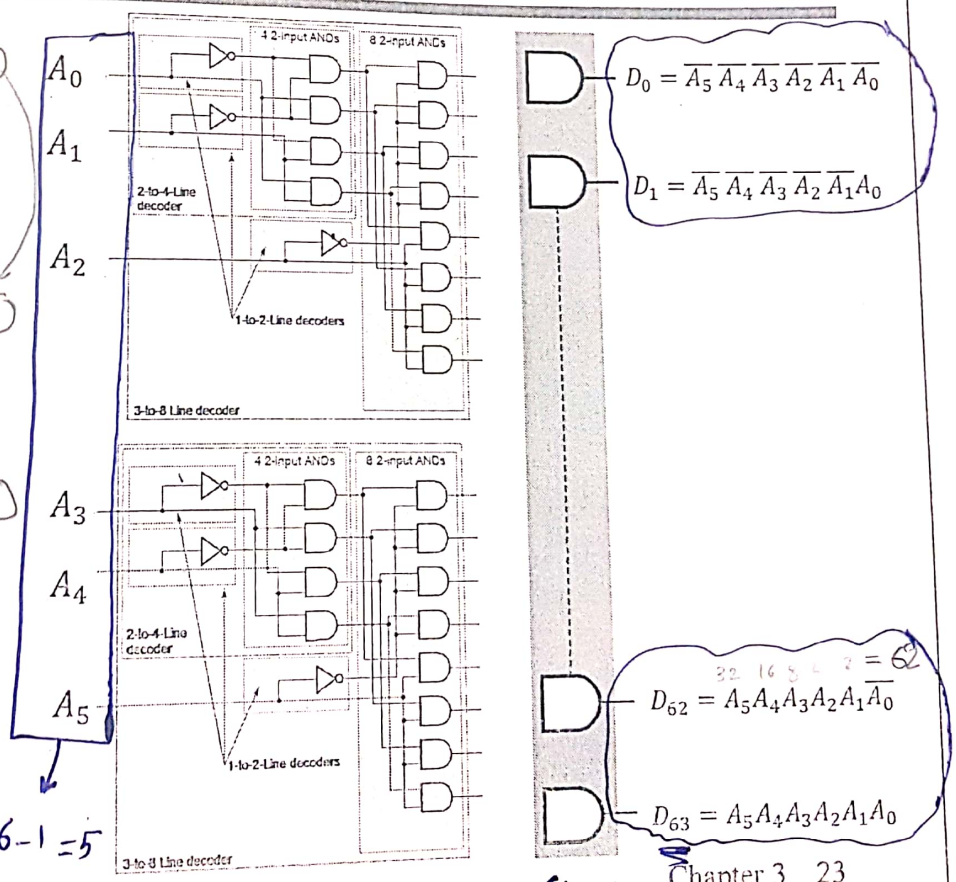
1-to-2 لبيته 2

2 بنظرية

يعني بدل بنحجز عادي

3 هلأ صرلك 6 هي

IC inverter



$6 - 1 = 5$

$64 - 1 = 63$

Decoder Expansion - Example 3

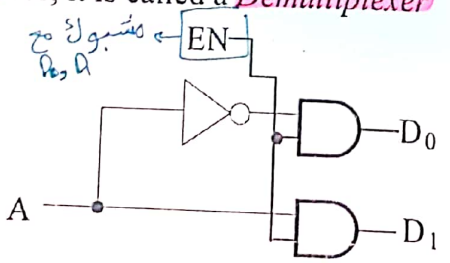
- 7-to-128-line decoder
  - $k = n = 7$
  - We need 2^7 (128) 2-input AND gates driven as follows:
  - k is odd, so split to:
    - 4-to-16-line decoder
    - 3-to-8-line decoder
  - 4-to-16-line decoder
    - $k = n = 4$
    - We need 2^4 (16) 2-input AND gates driven as follows:
    - k is even, so split to:
      - Two 2-to-4-line decoders
  - Complete using known 3-8 and 2-to-4 line decoders
- $GN = 128 \times 2 + 16 \times 2 + 8 \times 2 + 12 \times 2 + 7 = \boxed{335}$
- Compare to straight forward design with GN cost of 903

Building Larger Decoders

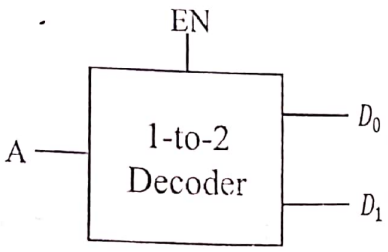
- Method_1: Decoder Expansion
- Method_2: Using Small *Decoders with Enable input*
- Example: 1-to-2 line decoder with enable → A
  - In general, attach *m-enabling* circuits to the outputs
  - See truth table below for function
    - Note use of X's to denote both 0 and 1
    - Combination containing two X's represent two binary combinations
- Alternatively, can be viewed as distributing value of signal EN to 1 of 2 outputs
  - In this case, it is called a *Demultiplexer*

EN	A	D_0	D_1
0	X	0	0
1	0	1	0
1	1	0	1

(a)



(b)



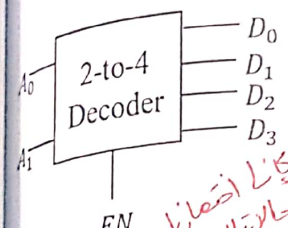
(c)

A1 A0

Decimal لـ Binary من تحول *

2-to-4 Line Decoder with Enable

- Attach *4-enabling* circuits to the outputs
- See truth table below for function
  - Combination containing two X's represent four binary combinations
- Alternatively, can be viewed as distributing value of signal EN to 1 of 4 outputs
  - In this case, it is called a *Demultiplexer*



(c)

EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(a)

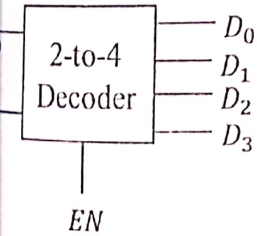
كل اختيار لـ 2 الحالات المعروفة بيه

٤ حالات عنا للواحد

2-to-4 Line Decoder with Enable

- Attach *4-enabling* circuits to the outputs
- See truth table below for function
  - Combination containing two X's represent four binary combinations
- Alternatively, can be viewed as distributing value of signal EN to 1 of 4 outputs
  - In this case, it is called a *Demultiplexer*

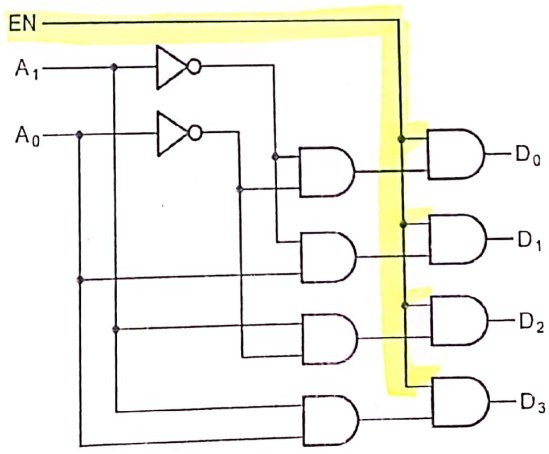
* الـ EN مشبوك مباشرة مع الـ output عشان لما يكون لا يسمح للقيم تظهر عليهم .



(c)

EN	A_1	A_0	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

(a)



(b)

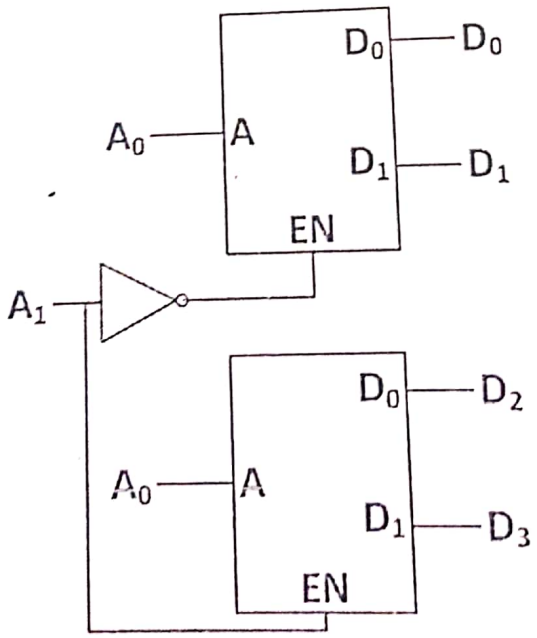
Computer Design Fundamentals, 4e
Slides
Pearson Education, Inc

2-to-4 Decoder using 1-to-2 Decoders with enable and Inverters

	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	1	0	0	0
1	0	1	0	1	0	0
2	1	0	0	0	1	0
3	1	1	0	0	0	1

1st 1-to-2 Decoder

2nd 1-to-2 Decoder

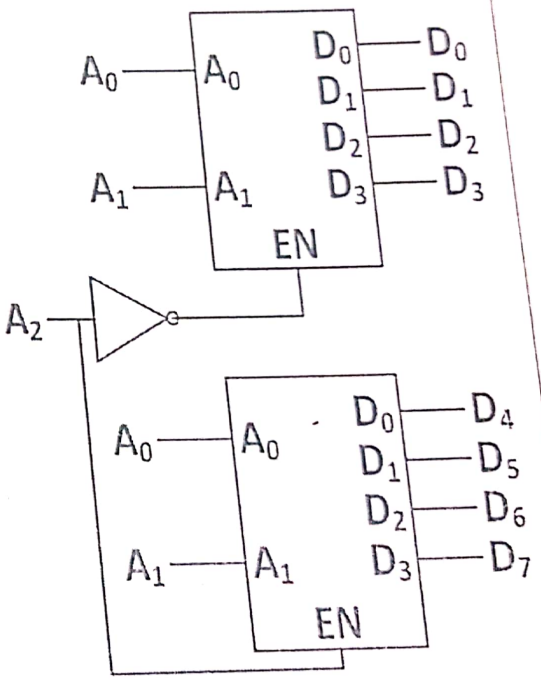


with enable

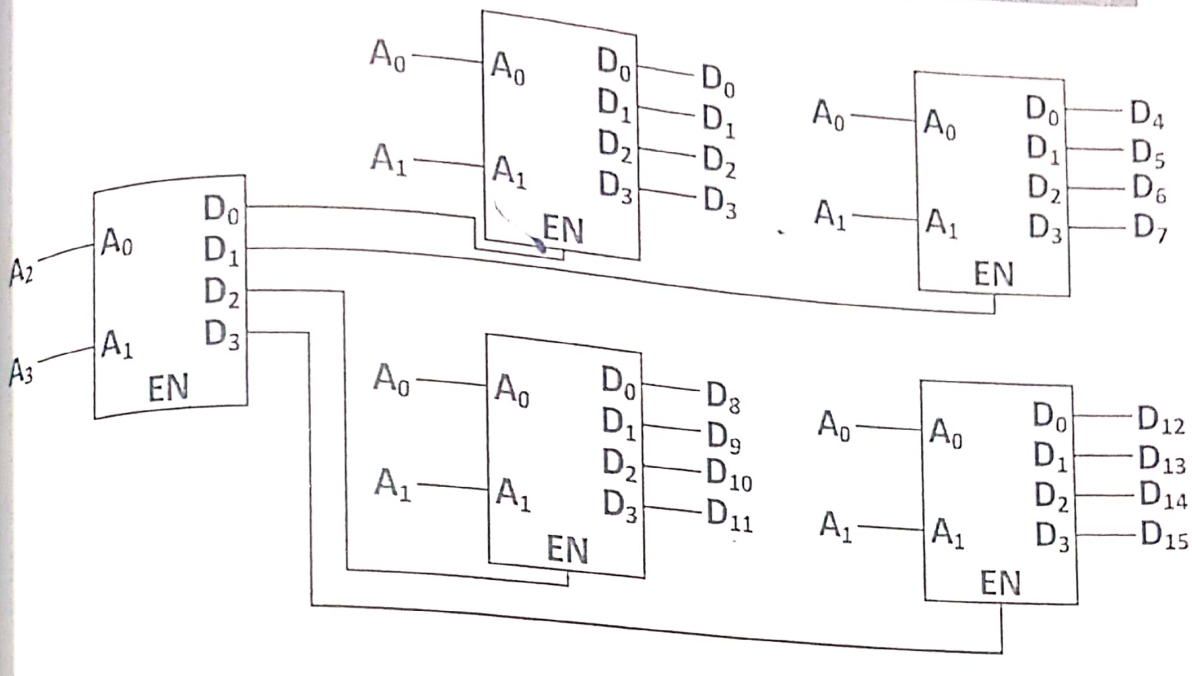
3-to-8 Decoder using 2-to-4 Decoders and Inverters

A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0
1	1	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	0	1

1st 2-to4 Decoder 2nd 2-to4 Decoder



4-to-16 Decoder using Only 2-to-4 Decoders



Combinational Logic Implementation - Decoder and OR Gates

- Implement m functions of n variables with:
  - Sum-of-minterms expressions SOM
  - One n-to-2^n-line decoder
  - m OR gates, one for each function
  - For each function, the OR gate has k inputs, where k is the number of minterms in the function
- **Approach 1:**
  - Find the truth table for the functions
  - Make a connection to the corresponding OR from the corresponding decoder output wherever a 1 appears in the truth table
- **Approach 2**
  - Find the minterms for each output function
  - OR the minterms together

Example1

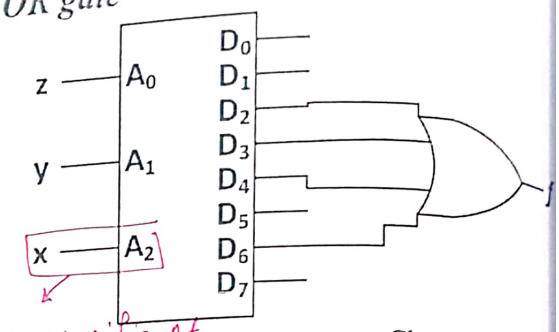
- Implement function f using decoder and OR gate:

1 function $f(x, y, z) = x\bar{z} + \bar{x}y$ 3 variables

- $n = 3$ variables → *3-to-8 decoder*
- One function → *One OR gate*
- Solution: Convert حول f to SOM format
  - $f = x\bar{z}(y + \bar{y}) + \bar{x}y(z + \bar{z}) = xy\bar{z} + x\bar{y}\bar{z} + \bar{x}yz + \bar{x}y\bar{z}$
  - $f(x, y, z) = \sum_m(2,3,4,6)$ → *4-input OR gate*

the most significant

- ***Decoder is a Minterm Generator***



most significant

Example2

- Implement function f using decoder and OR gate:

$$f(w, x, y, z) = \sum_m (0, 4, 8, 11, 12, 14, 15)$$

الدكتور بيفضل التقسيم أربعات

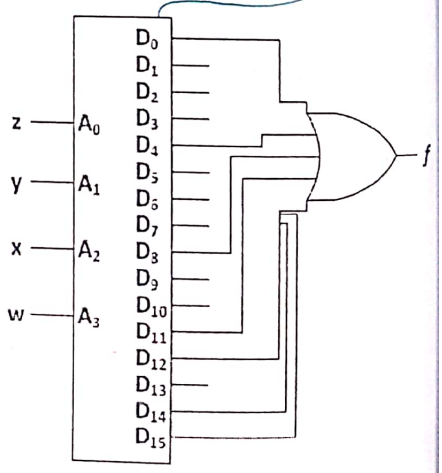
- $n = 4$ variables → *4-to-16 decoder*
- One function with 7 minterms → *One 7-input OR gate*

- ***If number of minterms is greater than $\frac{2^n}{2}$, then design for complement F ($\bar{F}$) and use NOR gate instead of OR to generate F***

عدد المنتجات

$\frac{16}{2} = 8$

$7 \leq 8$



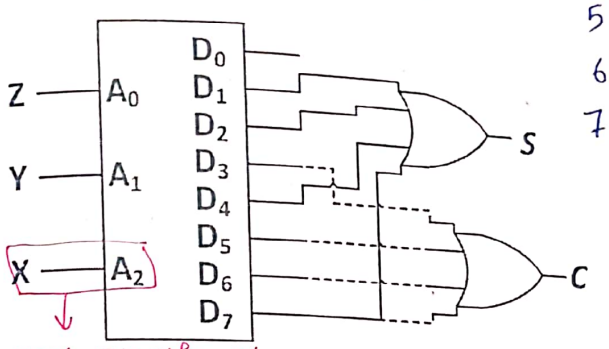
Example3

- Implement functions C and S using decoder and OR gates:
  - $n = 3$ variables → 3-to-8 decoder
  - Two function → Two OR gates
  - Solution:

٣. بدور على ال Min لكل فنكشن
لحال (C و S) ← فايعطي الناتج = ١

  - $C = \Sigma_m(3,5,6,7)$ → 4-input OR gate
  - $S = \Sigma_m(1,2,4,7)$ → 4-input OR gate

	X	Y	Z	C	S
0	0	0	0	0	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	1	1



most significant.

Example4

- Implement the following set of odd parity functions of (A_7, A_6, A_5, A_4)

$P_1 = A_7 \oplus A_5 \oplus A_4$

$P_2 = A_7 \oplus A_6 \oplus A_4$

$P_3 = A_7 \oplus A_6 \oplus A_5$

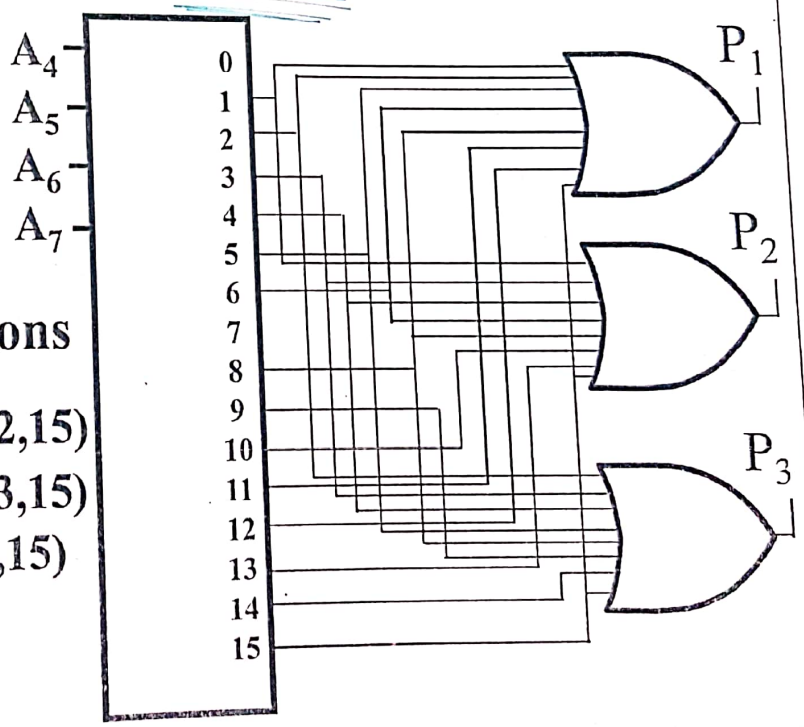
- Finding sum of minterms expressions

$P_1 = \Sigma_m(1,2,5,6,8,11,12,15)$

$P_2 = \Sigma_m(1,3,4,6,8,10,13,15)$

$P_3 = \Sigma_m(2,3,4,5,8,9,14,15)$

- Find circuit
- Is this a good idea?



* A_7, A_6, A_5, A_4

$$P_1 = A_7 \oplus A_5 \oplus A_4$$

$$= \bar{A}_7 \bar{A}_5 A_4 + \bar{A}_7 A_5 \bar{A}_4 + A_7 \bar{A}_5 \bar{A}_4 + A_7 A_5 A_4$$

$(A_6 + \bar{A}_6)$ $(A_6 + \bar{A}_6)$ $(A_6 + \bar{A}_6)$ $(A_6 + \bar{A}_6)$

* أخيراً ليش وزعنا ال (comp) مثل هيك لأنه بالسؤال حاكيلي أنه بدو إياها (odd parity) فأنا لازم أضمن أنه يكون عدد ال (1's) فردي

* بعدين بعمل ANDing مع العنصر المفقود من ال function مثل $(A_6 + \bar{A}_6)$ لأنه متى رح تأثر على ال function كأني زدت أو اعلت AND مع 1 فما بتأثر القيمة

* كل حد رح يصير حدين فأنا عندي 4 حدود رح يصيروا 8

$$\underset{0\,0\,0\,1}{\bar{A}_7 \bar{A}_6 \bar{A}_5 A_4}\,(1) + \underset{0\,1\,0\,1}{\bar{A}_7 A_6 \bar{A}_5 A_4}\,(5) + \underset{0\,0\,1\,0}{\bar{A}_7 \bar{A}_6 A_5 \bar{A}_4}\,(2) + \underset{0\,1\,1\,0}{\bar{A}_7 A_6 A_5 \bar{A}_4}\,(6)$$

$$\underset{1\,0\,0\,0}{A_7 \bar{A}_6 \bar{A}_5 \bar{A}_4}\,(8) + \underset{1\,1\,0\,0}{A_7 A_6 \bar{A}_5 \bar{A}_4}\,(12) + \underset{1\,0\,1\,1}{A_7 \bar{A}_6 A_5 A_4}\,(11) + \underset{1\,1\,1\,1}{A_7 A_6 A_5 A_4}\,(15)$$

$$\Rightarrow P_1 = \Sigma m(1, 2, 5, 6, 8, 11, 12, 15)$$

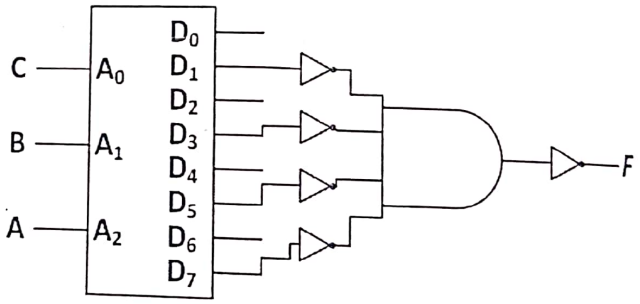
والباقي نفس خطوات الحل

Example5

- Implement function F using 3-to-8 decoder, AND gate and inverters: $F(A,B,C) = \sum_m(1,3,5,7)$

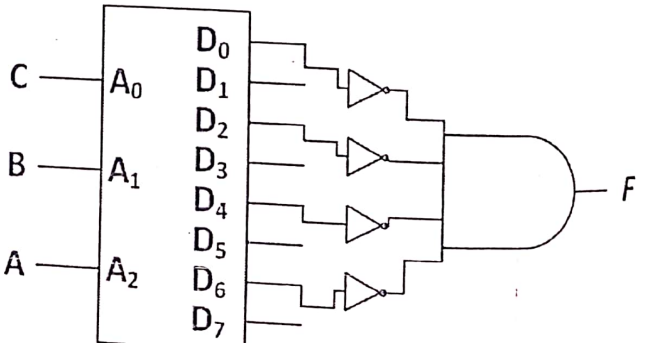
① ② ③

⇒ ▪ Solution with 5 inverters:



⇒ ▪ Solution with 4 inverters:

- $F(A,B,C) = \prod_M(0,2,4,6)$



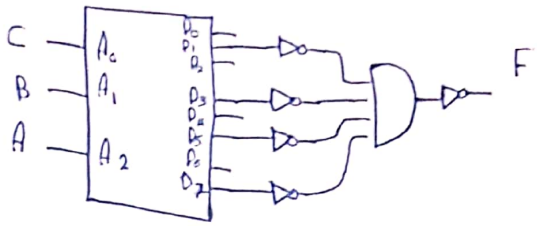
$F(A,B,C) = \Sigma m(1,3,5,7)$

- $n = 3$ variables
  3-to-8 decoder
- One function → One OR gate

طريقة حل الـ Decoder العادية

لـ بس هو طالب منى AND وما بحلى استخدام inverter

* طريقة التحويل من OR لـ AND



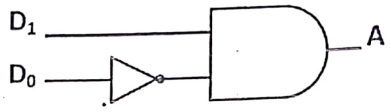
Encoding

- ***Encoding***: the opposite of decoding - the conversion of an m-bit input code to a n-bit output code with $n \leq m \leq 2^n$ such that each valid code word produces a unique output code
- Circuits that perform encoding are called ***encoders***
- An encoder has 2^n (or fewer) input lines and n output lines which ***generate the binary code corresponding to the input values***
- Typically, an encoder converts a code containing exactly one bit that is 1 to a binary code corresponding to the position in which the 1 appears

2-to-1 Encoder & 4-to-2 Encoder

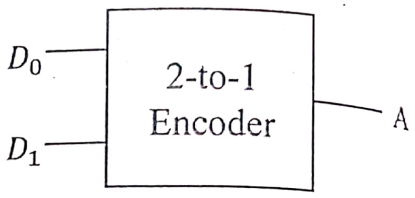
D_1	D_0	A
0	0	Invalid Input
0	1	0
1	0	1
1	1	Invalid Input

(a)



$A = D_1.\overline{D_0} = D_1$

(b)



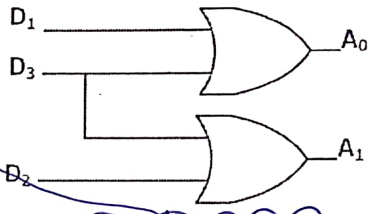
(c)

متغير

ثابتة

D_3	D_2	D_1	D_0	A_1	A_0	
0	0	0	1	0	0	0
0	0	1	0	0	1	1
0	1	0	0	1	0	2
1	0	0	0	1	1	3

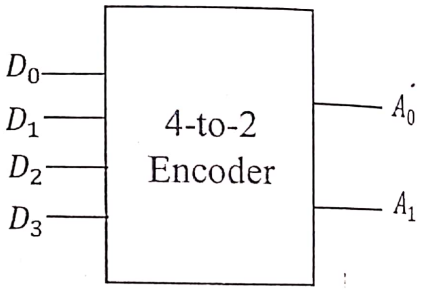
(a)



$A_0 = D_1 + D_3$

$A_1 = D_2 + D_3$

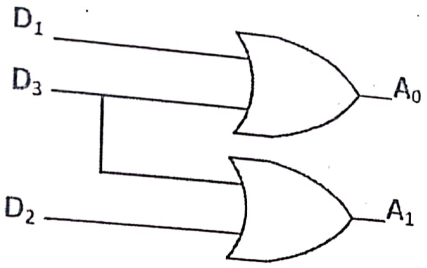
(b)



(c)

D_3	D_2	D_1	D_0	A_1	A_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

(a)

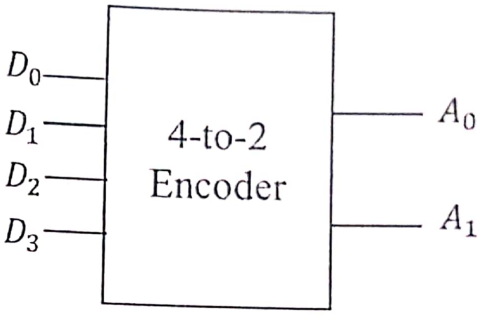


$$A_0 = D_1 + D_3$$

$$A_1 = D_2 + D_3$$

(b)

(c)

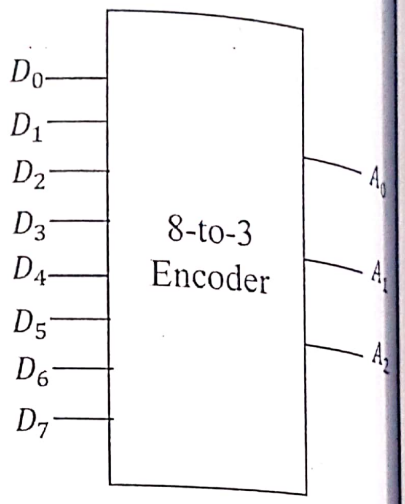


(c)

8-to-3 Encoder (Octal-to-Binary Encoder)

	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	1	0	0	0	1
2	0	0	0	0	0	1	0	0	0	1	0
3	0	0	0	0	1	0	0	0	0	1	1
4	0	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	0	0	0	0	1	0	1
6	0	1	0	0	0	0	0	0	1	1	0
7	1	0	0	0	0	0	0	0	1	1	1

(a)



(c)

$$1 = A_0 = D_1 + D_3 + D_5 + D_7$$
$$1 = A_1 = D_2 + D_3 + D_6 + D_7$$
$$1 = A_2 = D_4 + D_5 + D_6 + D_7$$

(b)

Decimal-to-BCD Encoder

- ***Inputs:*** 10 bits corresponding to decimal digits 0 through 9, (D_0, …, D_9)
- ***Outputs:*** 4 bits with BCD codes (A_3, A_2, A_1, A_0)
- ***Function:*** If input bit D_i is a 1, then the output is the BCD code for i
- The truth table could be formed, but alternatively, the equations for each of the four outputs can be obtained directly

Decimal-to-BCD Encoder Cont.

- Input D_i is a term in equation A_j if bit A_j is 1 in the binary value for i
- Equations:

$$A_3 = D_8 + D_9$$
$$A_2 = D_4 + D_5 + D_6 + D_7$$
$$A_1 = D_2 + D_3 + D_6 + D_7$$
$$A_0 = D_1 + D_3 + D_5 + D_7 + D_9$$

1 at the same time

- What happens if two inputs are high simultaneously?
  - For example if D_3 and D_6 are high, then the output is 0111 which indicates that only D_7 is high ???
  - ***Solution: Establish input priority***

Priority Encoder

- If more than one input value is 1, then the encoder just designed does not work
- One encoder that can accept all possible combinations of input values and produce a meaningful result is a *priority encoder*
- Among the 1s that appear, it selects the most significant input position (or the least significant input position) containing a 1 and responds with the corresponding binary code for that position
  - *High priority encoder*: gives priority for the input whose value is 1 and has the highest subscript
  - *low priority encoder*: gives priority for the input whose value is 1 and has the lowest subscript

الدكتور يلي بحددهم بالامتحان

- If all inputs are 0's, what happens?
  - Define an output (V) to encode whether the input is valid or not
  - When all inputs are 0's, V is set to 0 indicating that the input is invalid, otherwise V is set to 1

(c)

4-to-2 Low Priority Encoder

# of Minterms/ Rows	D_3	D_2	D_1	D_0	A_1	A_0	V	
1	0	0	0	0	X	X	0	
8	X	X	X	1	0	0	1	0
4	X	X	1	0	0	1	1	1
2	X	1	0	0	1	0	1	2
1	1	0	0	0	1	1	1	3

(a)

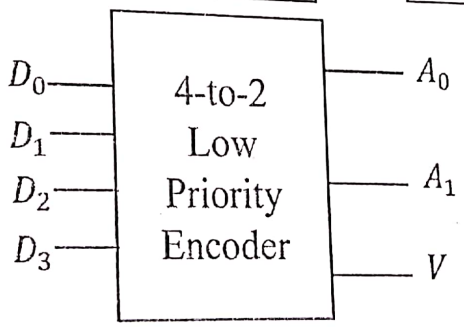
$$A_0 = D_1\overline{D_0} + D_3\overline{D_2}\ \overline{D_1}\ \overline{D_0}$$
$$A_0 = \overline{D_0}(D_1 + D_3\overline{D_2}\ \overline{D_1})$$
$$A_0 = \overline{D_0}(D_1 + D_3\overline{D_2})$$
$$A_0 = D_1\overline{D_0} + D_3\overline{D_2}\ \overline{D_0}$$

$$A_1 = D_2\overline{D_1}\ \overline{D_0} + D_3\overline{D_2}\ \overline{D_1}\ \overline{D_0}$$
$$A_1 = \overline{D_1}\ \overline{D_0}(D_2 + D_3\overline{D_2})$$
$$A_1 = \overline{D_1}\ \overline{D_0}(D_2 + D_3)$$
$$A_1 = D_2\overline{D_1}\ \overline{D_0} + D_3\overline{D_1}\ \overline{D_0}$$

$$V = D_3 + D_2 + D_1 + D_0$$

(b)

Number of Minterms per Row $= 2^{\#\ of\ don't\ cares}$



(c)

4-to-2 High Priority Encoder

#_of_Minterms/ Rows	D_3	D_2	D_1	D_0	A_1	A_0	V
1	0	0	0	0	X	X	0
1	0	0	0	1	0	0	1
2	0	0	1	X	0	1	1
4	0	1	X	X	1	0	1
8	1	X	X	X	1	1	1

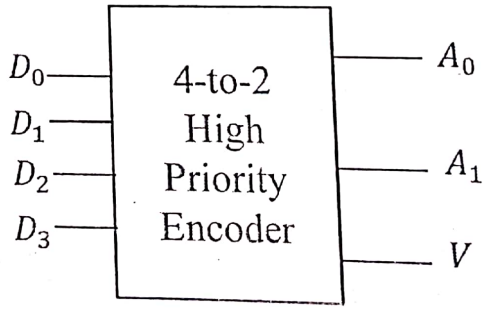
(a)

$$A_0 = D_3 + \overline{D_3}\,\overline{D_2}D_1$$
$$A_0 = D_3 + \overline{D_2}D_1$$

$$A_1 = D_3 + \overline{D_3}D_2$$
$$A_1 = D_3 + D_2$$

$$V = D_3 + D_2 + D_1 + D_0$$

(b)



(c)

5-input Priority Encoder

- Priority encoder with 5 inputs (D_4, D_3, D_2, D_1, D_0) - highest priority to most significant 1 present - Code outputs A_2, A_1, A_0 and V where V indicates at least one 1 present

No. of Min-terms/Row	Inputs					Outputs				
	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0	V	
1	0	0	0	0	0	X	X	X	0	
1	0	0	0	0	1	0	0	0	1	0
2	0	0	0	1	X	0	0	1	1	1
4	0	0	1	X	X	0	1	0	1	2
8	0	1	X	X	X	0	1	1	1	3
16	1	X	X	X	X	1	0	0	1	4

- X's in input part of table represent 0 or 1; thus table entries correspond to product terms instead of minterms. The column on the left shows that all 32 minterms are present in the product terms in the table

5-input Priority Encoder Cont.

- Could use a K-map to get equations, but can be read directly from table and manually optimized if careful:

$$A_2 = D_4$$

$$A_1 = \overline{D}_4 D_3 + \overline{D}_4 \overline{D}_3 D_2 = \overline{D}_4 (D_3 + D_2)$$

$$A_1 = \overline{D}_4 D_3 + \overline{D}_4 D_2$$

$$A_0 = \overline{D}_4 D_3 + \overline{D}_4 \overline{D}_3 \overline{D}_2 D_1 = \overline{D}_4 (D_3 + \overline{D}_2 D1)$$

$$A_0 = \overline{D}_4 D_3 + \overline{D}_4 \overline{D}_2 D_1$$

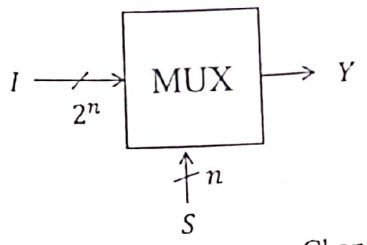
$$V = D_4 + D_3 + D_2 + D_1 + D_0$$

Selecting

- Selecting of data or information is a critical function in digital systems and computers
- Circuits that perform selecting have:
  - A set of information inputs from which the selection is made
  - A single output
  - A set of control lines for making the selection
- Logic circuits that perform selecting are called *multiplexers*
- Selecting can also be done by three-state logic

Multiplexers (MUX) (Data Selectors)

- A multiplexer selects information from an input line and directs the information to an output line
- A typical multiplexer has n control inputs ($S_{n-1}, \ldots S_0$) called *selection inputs*, 2^n information inputs ($I_{2^n-1}, \ldots I_0$), and one output Y
- A multiplexer can be designed to have m information inputs with $m < 2^n$ as well as n selection inputs
- Multiplexers allow sharing of resources and reduce the cost by reducing the number of wires



2-to-1-Line MUX

control inputs

* تعبئة هذا الجدول كالنمط المعتاد $2^3=8$

one output

S = 0 يسمح لقيم I_0 تظهر

S = 1 يسمح لقيم I_1 تظهر

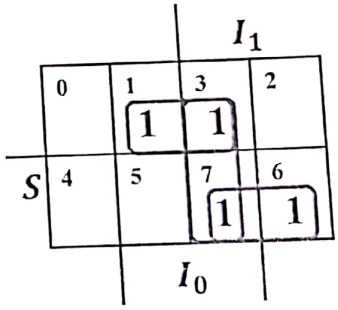
- Since $2 = 2^1$, $n = 1$
- The single selection variable S has two values:
  - S = 0 selects input I_0
  - S = 1 selects input I_1
- The equation:

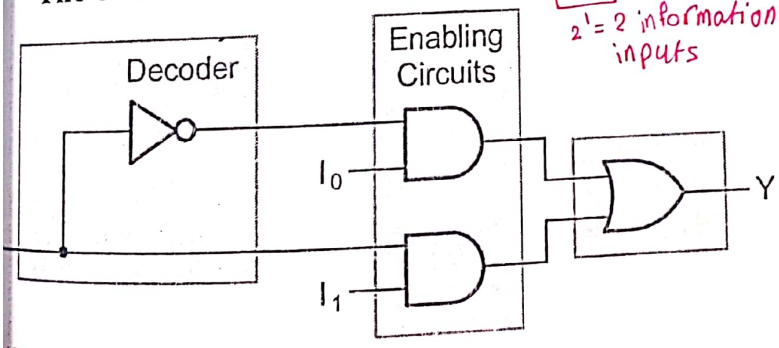
  $Y = \bar{S}I_0 + SI_1$
- The circuit:

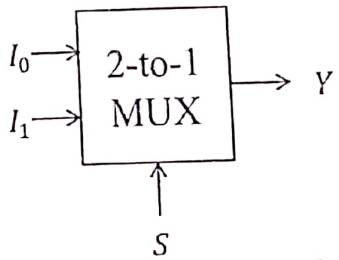
S	I_1	I_0	Y	
0	0	0	0	$Y = I_0$
0	0	1	1	
0	1	0	0	
0	1	1	1	
1	0	0	0	$Y = I_1$
1	0	1	0	
1	1	0	1	
1	1	1	1	

$2^1 = 2$ information inputs

S	Y
0	I_0
1	I_1





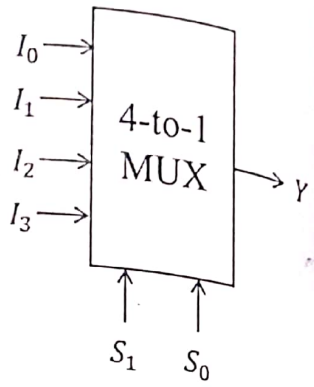


4-to-1-Line MUX

- Since $4 = 2^2$, n = 2
- There are two selection variables (S_1S_0) and they have four values: كانها بتمثل رقمها
  - $S_1S_0 = 00$ selects input I_0
  - $S_1S_0 = 01$ selects input I_1
  - $S_1S_0 = 10$ selects input I_2
  - $S_1S_0 = 11$ selects input I_3
- The equation:

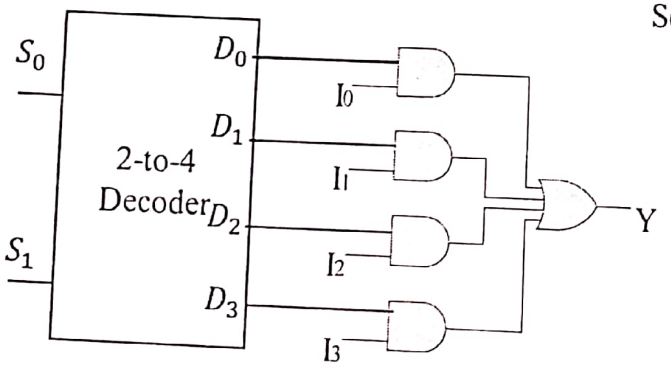
$$Y = \overline{S_1}\,\overline{S_0}I_0 + \overline{S_1}\,S_0I_1 + S_1\overline{S_0}\,I_2 + S_1S_0I_3$$

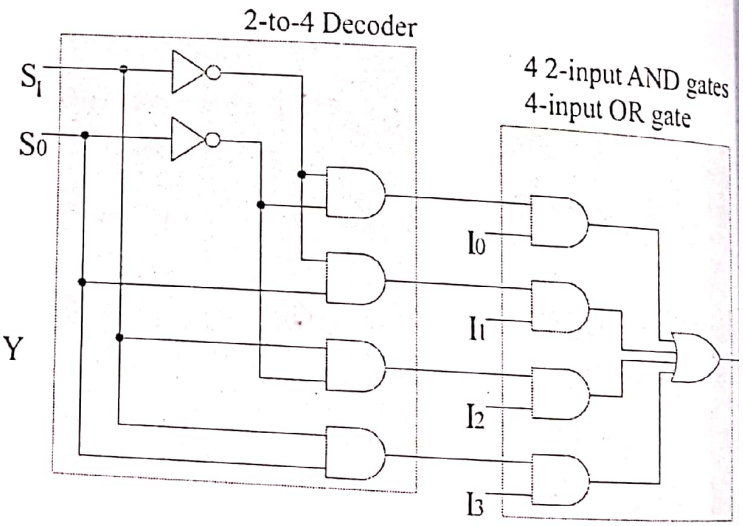
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



4-to-1-line MUX Cont.

- 2-to-4-line decoder
- 4 2-input AND gates
- 4-input OR gate





2-to-1-Line MUX Cont.

- Note the regions of the multiplexer circuit shown:
  - 1-to-2-line Decoder
  - 2 Enabling circuits
  - 2-input OR gate

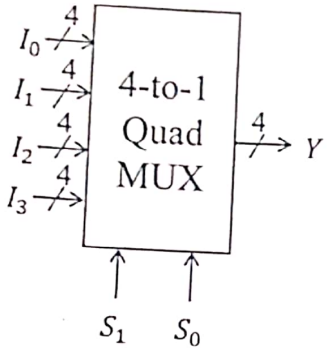
- *In general, for an 2^n-to-1-line multiplexer:*
  - *n-to-2^n-line decoder*
  - *2^n 2-input AND gate*
  - *One 2^n-input OR gate*

Homework

- **Implement 8-to-1-Line MUX and 64-to-1 MUX:**
  - **How many select lines are needed?**
  - **Decoder size?**
  - **How many 2-input AND gates are needed?**
  - **What is the size of the OR gate?**

Multiplexer Width Expansion

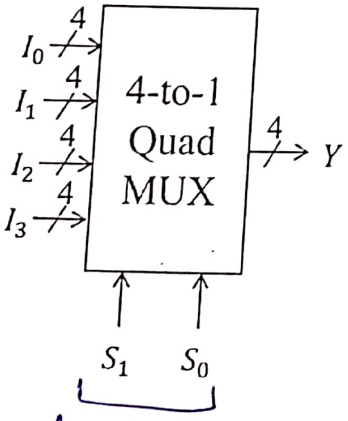
- Select "vectors of bits" instead of "bits"
- Example: *4-to-1-line quad multiplexer*



?

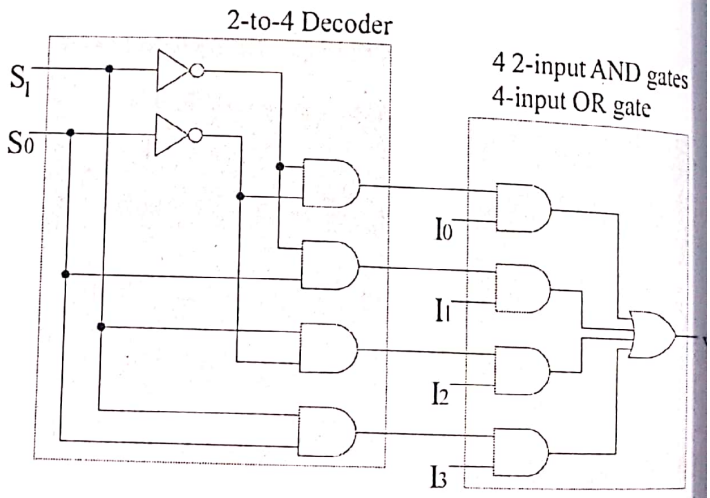
Multiplexer Width Expansion

- Select "vectors of bits" instead of "bits"
- Example: *4-to-1-line quad multiplexer*



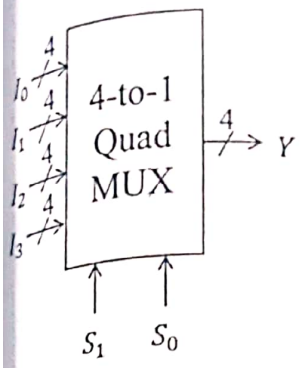
select line

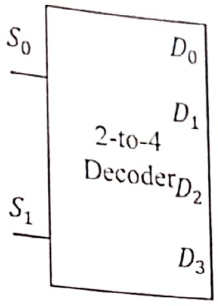
?

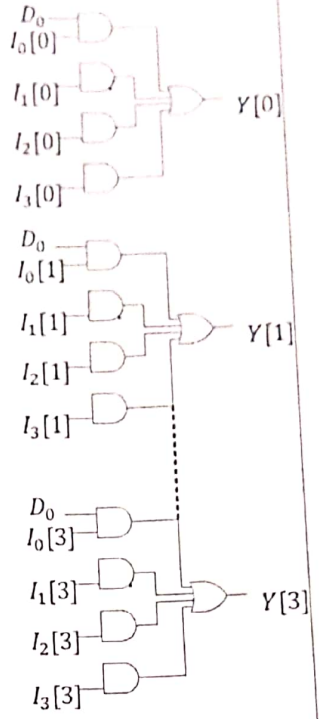


Multiplexer Width Expansion

- Select "vectors of bits" instead of "bits"
- Example: *4-to-1-line quad multiplexer*

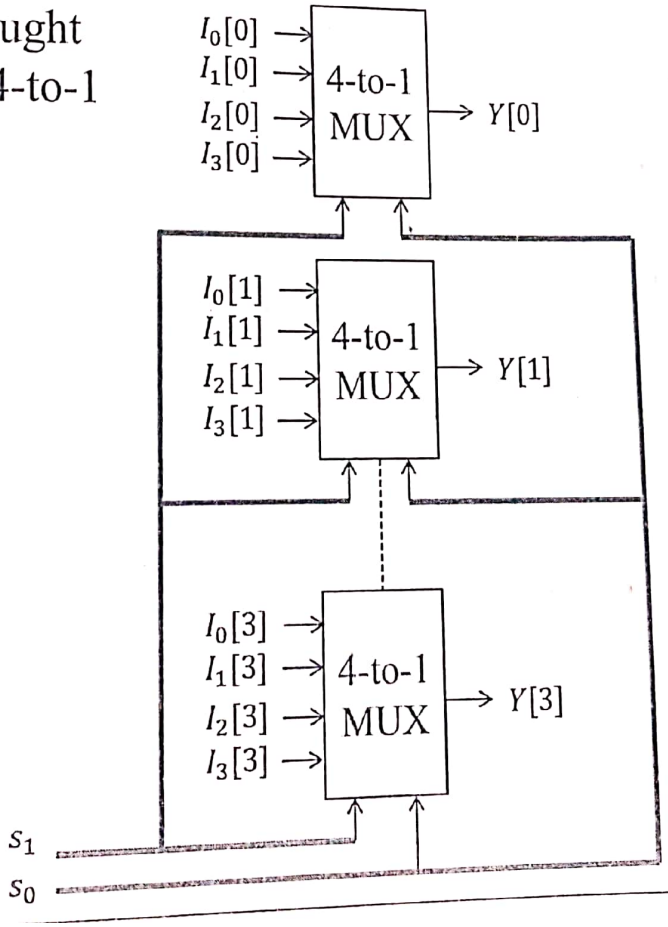






Multiplexer Width Expansion Cont.

- Can be thought of as four 4-to-1 MUXes:

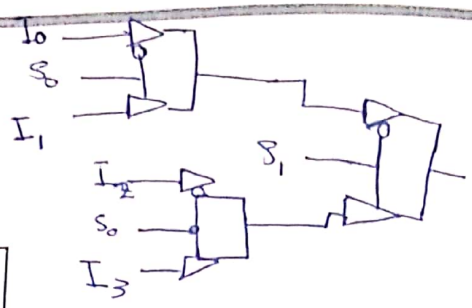


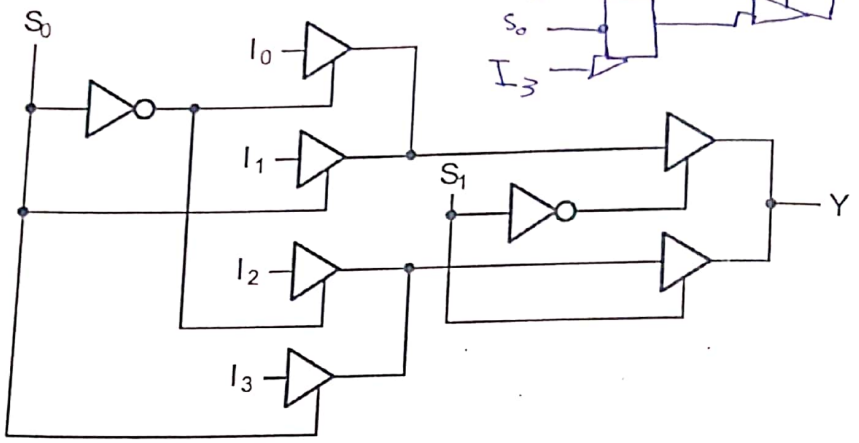
selectline الـ
بالعربي

Other Selection Implementations

لو كان Real بيكون نفس المشكل بس أربع مرات
y[0] : y[1]

- **Three-state logic**





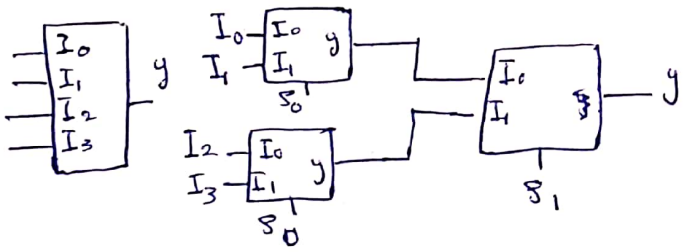
التريك بعملية الـ labels

Building Large MUXes from Smaller Ones

لازم نحطلها الـ label

- **4-to-1 MUX using three 2-to-1 MUXes**

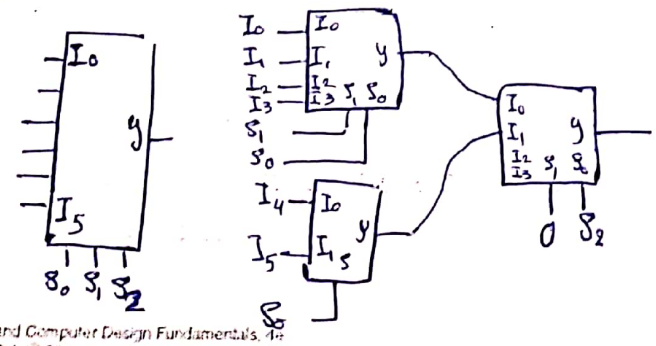
جاد الحل الصح



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

- **6-to-1 MUX using two 4-to-1 MUXes and one 2-to-1 MUX**

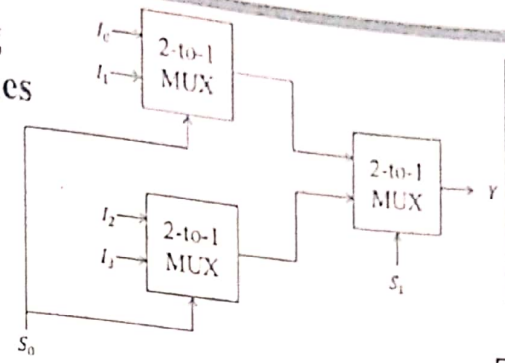
بالامتحان * نسأل الدكتور



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	X
1	1	1	X

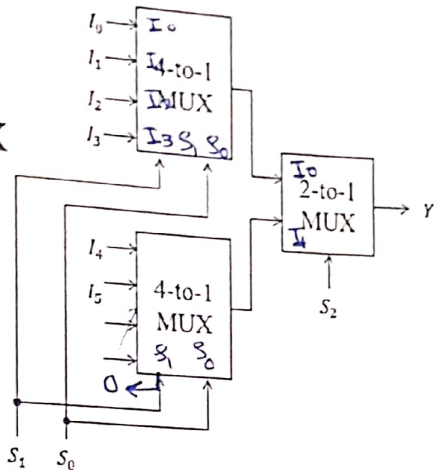
Building Large MUXes from Smaller Ones

- 4-to-1 MUX using three 2-to-1 MUXes



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

- 6-to-1 MUX using two 4-to-1 MUXes and one 2-to-1 MUX



S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	X
1	1	1	X

Homework

- **Build an 8-to-1 MUX using:**
  - **Two 4-to-1 MUX and one 2-to-1 MUX**

ممكن يجي سؤال ضمن اختر.

قديما بدنا نستخدم منها

  - **One 4-to-1 MUX and multiple 2-to-1 MUXes** → minimum 3
  - **Only 2-to-1 MUXes (How many MUXes are need?)**

m function
n variable] → (m-bit) 2^n-to-1 MUX

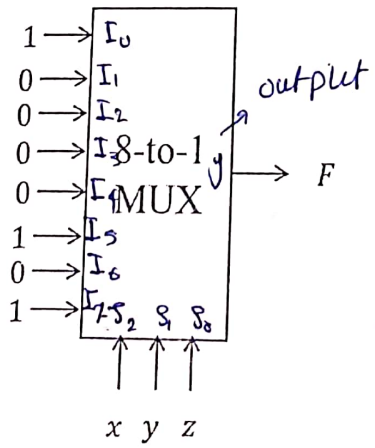
Combinational Logic Implementation - Multiplexer Approach 1

- Implement m functions of n variables with:
  - Sum-of-minterms expressions
  - An m-wide 2^n-to-1-line multiplexer
- Design:
  - Find the truth table for the functions
  - In the order they appear in the truth table:
    - Apply the *function input variables* to the multiplexer *select inputs* $S_{n-1}, \ldots, S_0$
    - Label the outputs of the multiplexer with the output variables
  - Value-fix the information inputs to the multiplexer using the values from the truth table (for don't cares, apply either 0 or 1)

Example1

- Implement the following function using a single MUX based on Approach1 : $F(x, y, z) = \sum_m(0, 5, 7)$
- Solution:
  - Single function → m = 1
  - 3 variables → n = 3 → 8-to-1 MUX
  - Fill the truth table of F

8 inputs



	x	y	z	F
0	0	0	0	1
	0	0	1	0
	0	1	0	0
	0	1	1	0
	1	0	0	0
5	1	0	1	1
	1	1	0	0
7	1	1	1	1

Example2: Gray to Binary Code

- Design a circuit to convert a 3-bit Gray code to a binary code
- The formulation gives the truth table on the right

ترتيب بهاي الطريقة عشان
نعرف انه
gray
code
لازم
يكون
كمه
اختلاف
1-bit-3

	Gray Code ABC	Binary Code XYZ
0	000	000
1	001	001
3	011	010
2	010	011
6	110	100
7	111	101
5	101	110
4	100	111

$\boxed{X = A}$

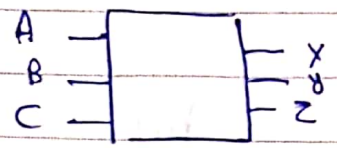
Gray to Binary Code Cont.

- Rearrange the table so that the input combinations are in counting order
- It is obvious from this table that $\boxed{X = A.}$ However, Y and Z are more complex
- Two functions (Y and Z) → m = 2
- 3 variables (A, B, and C) → n = 3
- Functions Y and Z can be implemented using a dual 8-to-1-line multiplexer by:
  - connecting A, B, and C to the multiplexer select inputs
  - placing Y and Z on the two multiplexer outputs
  - connecting their respective truth table values to the inputs

	Gray Code ABC	Binary Code XYZ
0	000	000
1	001	001
2	010	011
3	011	010
4	100	111
5	101	110
6	110	100
7	111	101

* Example 2

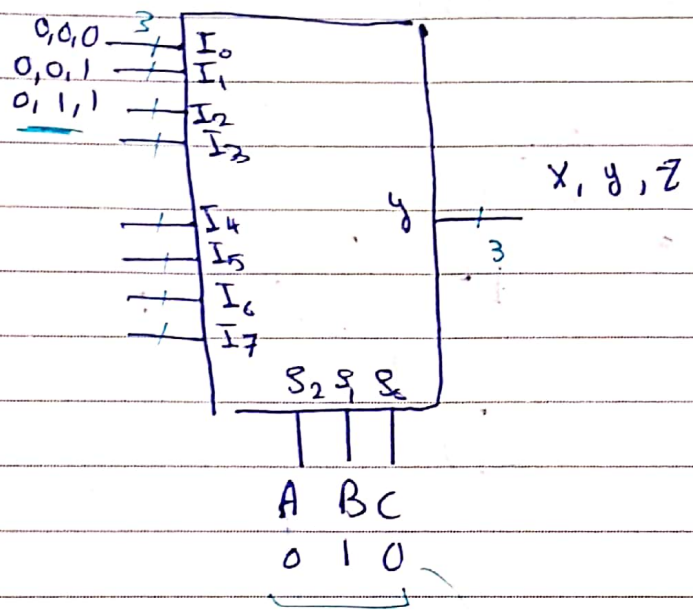
طريقة حل الدكتور



$m = 3$

$n = 3$

8-to-1 MUX



Gray Code ≅ 2

	A	B	C	X	y	Z
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	1
5	1	0	1	1	1	0
6	1	1	0	1	0	0
7	1	1	1	1	0	1

أو

No.

متى يكون الـ 1 =

$$X(A, B, C) = \sum_m (4, 5, 6, 7)$$
$$y(A, B, C) = \sum_m (2, 3, 4, 5)$$
$$Z(A, B, C) = \sum_m (1, 2, 4, 7)$$

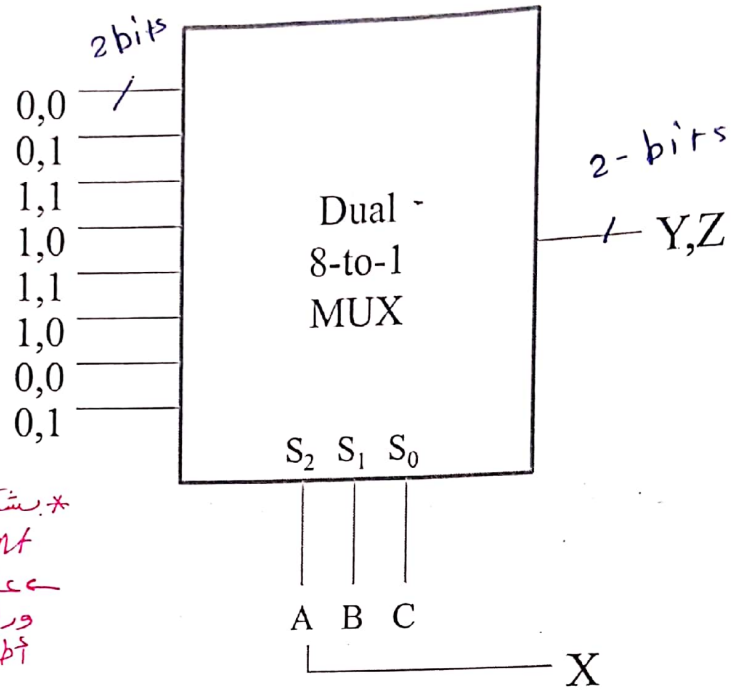
Gray to Binary Code Cont.

- Rearrange the table so that the input combinations are in counting order
- It is obvious from this table that X = A. However, Y and Z are more complex
- Two functions (Y and Z) → m = 2
- 3 variables (A, B, and C) → n = 3
- Functions Y and Z can be implemented using a dual 8-to-1-line multiplexer by:
  - connecting A, B, and C to the multiplexer select inputs
  - placing Y and Z on the two multiplexer outputs
  - connecting their respective truth table values to the inputs

	Gray Code ABC	Binary Code XYZ
0	000	000
1	001	001
2	010	011
3	011	010
4	100	111
5	101	110
6	110	100
7	111	101

Gray to Binary Code Cont.

* قاعدة :
لازم ال inputs
والـ outputs
يكونوا نفس ال bits



* بشكل عام برتب حسب ال
Most significant
عشان يطلعوا الحالات المتشابهة
ورا بعض ويكون أسهل
أطلع العلاقة بدون خطأ

Combinational Logic Implementation - Multiplexer Approach 2

تنفيذ

* ▪ **Implement any *m* functions of *n* variables by using:**
  - An m-wide $2^{(n-1)}$-to-1-line multiplexer — بتقلل المساحة
  - A single inverter if needed — + بتقلل التكلفة
- **Design:**
  - Find the truth table for the functions
  - Based on the values of the most significant *(n-1)* variables, separate the truth table rows into pairs
  - For each pair and output, define a rudimentary function of the
  - * least significant variable $(0, 1, X, \overline{X})$
  - Connect the most significant *(n-1)* variables to the select lines of the MUX, value-fix the information inputs to the multiplexer with the corresponding rudimentary functions
  - Use the inverter to generate the rudimentary function $\overline{X}$

Example1

most significant ١. بعبيهم عادي وبرتبهم حسب ال
٢. بشوف هو متى معطى الـ Function = 0 ومتى 1
٣. بقسمهم أزواج عشان أشوف العلاقة مع ناتج ال Function
٤. العلاقة هو بوظهر مع least significant يعني بعاد الاحتمال إما

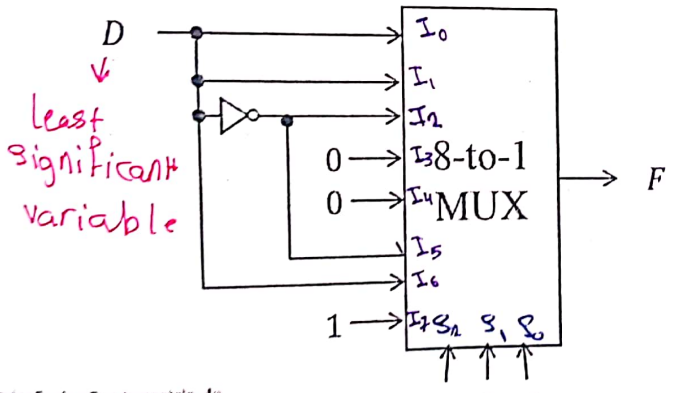
$\bar{D} = F$

$2^4 = 16$

- Implement the following function using a single MUX and an inverter (if needed) based on Approach2 :

$$F(A,B,C,D) = \sum_m (1,3,4,10,13,14,15)$$

- Solution:
  - Single function → m = 1
  - 4 variables → n = 4 → 8-to-1 MUX ($2^{4-1} = 8$)
  - Fill the truth table of F



A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = \bar{D}$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	1	$F = \bar{D}$
1	0	1	1	0	
1	1	0	0	0	$F = D$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	

0 1 2 3 4 5 6 7

Most significant variable (n-1)
$S_2 \Leftarrow 2 = n-1$ بحديث n صارت 3

* الأسهل يعمل ال truth table بحديث كل مجموعة
ABC=1
ABC=2
⋮ → بشوف علاقتهم مع F, D

Example2: Gray to Binary Code

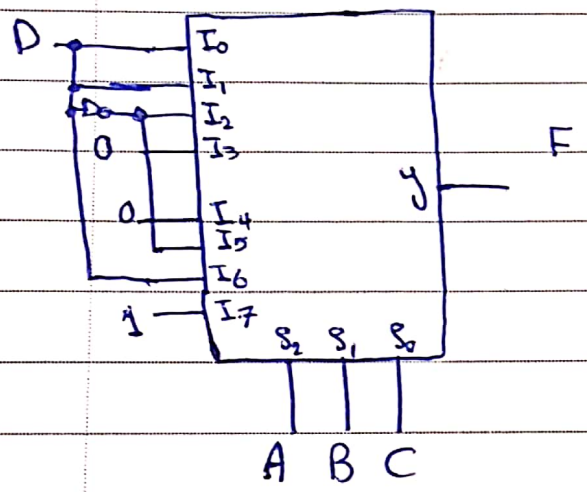
الـ Single inverter بصغر الـ MUX وبقلل التكلفة دايماً.

m functions $m = 1$

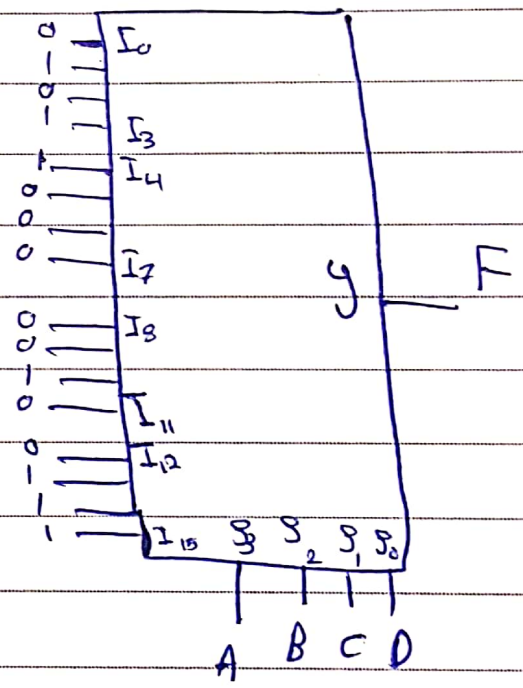
n variables $n = 4$

m-bit 2^{n-1}-to-1 MUX + inverter

* Example 1



* كيف بالعادة بنرسمها قبل هاد الدرس



لما نستخدم الـ inverter بتصغر (بتقل مساحتها) + بتقل التكلفة

Example2: Gray to Binary Code

$A'BC = ?$ → بشوف علاقتهم مع F, D

Gray Code ABC	Binary Code XYZ	Rudimentary Functions of C for Y	Rudimentary Functions of C for Z
0 000	000	$Y = 0$	$Z = C$
1 001	001		
2 010	011	$Y = 1$	$Z = \overline{C}$
3 011	010		
4 100	111	$Y = 1$	$Z = \overline{C}$
5 101	110		
6 110	100	$Y = 0$	$Z = C$
7 111	101		

$2^3 = 8$

$2^2 = 4$

0

1

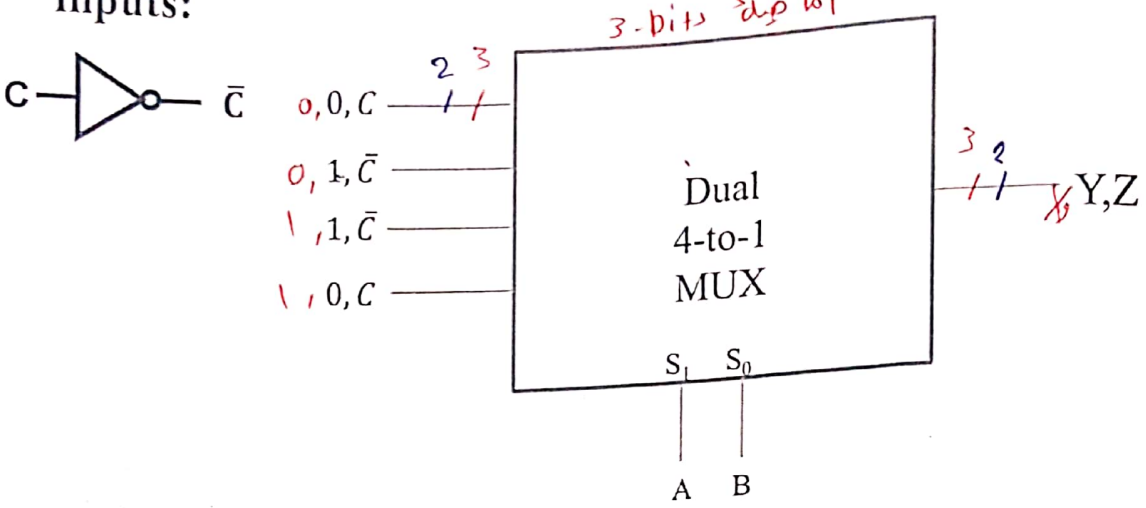
2

3

Gray to Binary Code Cont.

- Assign the variables and functions to the multiplexer inputs:

2-bits مدخل
3-bits مدخل أما

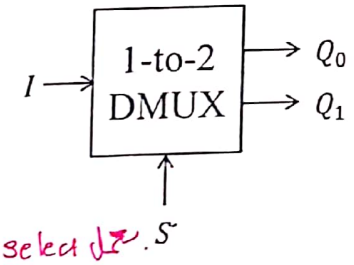


- *Note that Approach2 reduces the cost by almost half compared to Approach1*

* دائماً يبدأ بـ 1 أما الـ MUX دائماً بتنتهي بـ 1

Demultiplexer (DMUX)

- Opposite of multiplexer
- Receives **one** input and directs it to one from 2^n outputs based on **n-select** lines
- Example: 1-to-2 DMUX

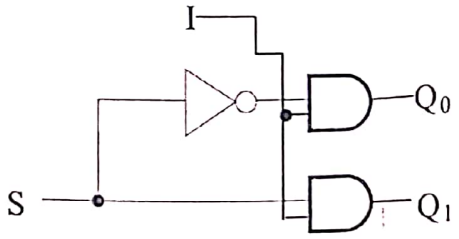


output / select عدد

$Q_0 = \bar{S}I$
$Q_1 = SI$

S	I	Q_1	Q_0
0	0	0	0
0	1	0	1
1	0	0	0
1	1	1	0

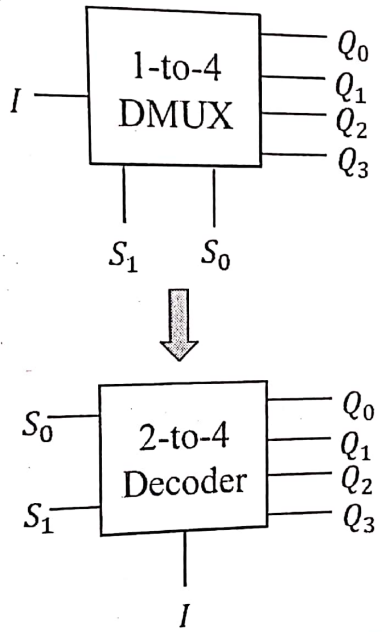
- *DMUX ≡ Decoder with Enable*

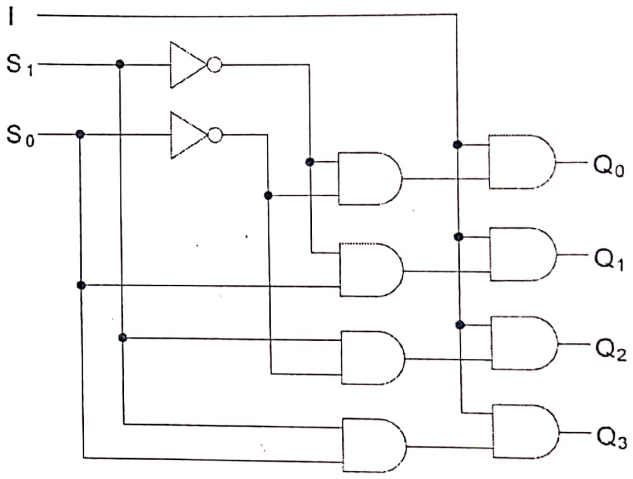


1-to-4 DMUX

- $Q_0 = \overline{S_1}\,\overline{S_0}I$
- $Q_1 = \overline{S_1}S_0I$
- $Q_2 = S_1\overline{S_0}I$
- $Q_3 = S_1S_0I$

S_1	S_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0





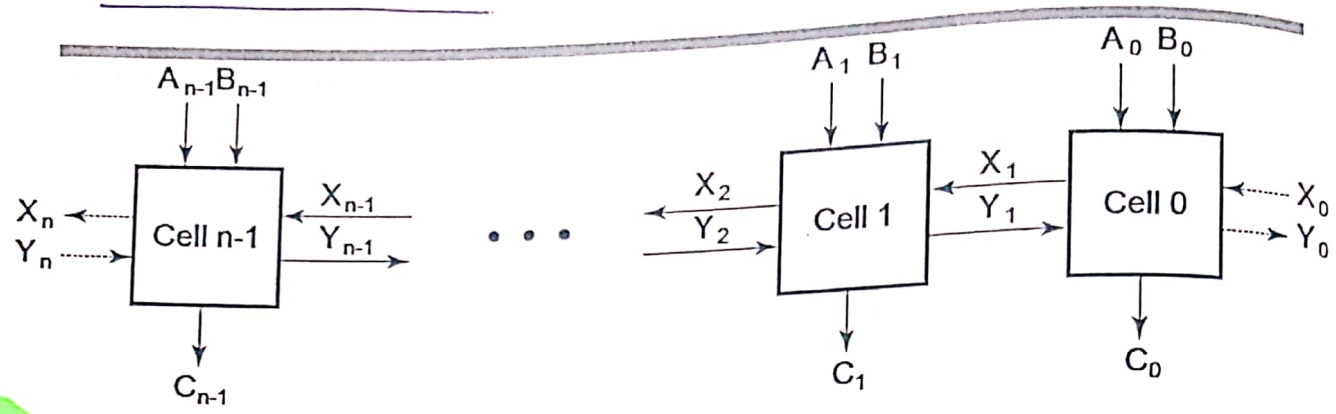
Iterative Combinational Circuits

- Arithmetic functions
  - Operate on binary vectors
  - Use the same sub-function in each bit position
- Can design functional block for the sub-function and repeat to obtain functional block for overall function
- ***Cell:*** sub-function block
- ***Iterative array:*** array of interconnected cells

ANDing هو المسؤول عن عملية ال

ترابطي

Block Diagram of an Iterative Array



- Example: n = 32
  - Number of inputs = 32*2 + 1 + 1 = 66
  - Truth table rows = 2^{66}
  - Equations with up to 66 input variables
  - Equations with huge number of terms
  - Design impractical!
- Iterative array takes advantage of the regularity to make design feasible

Functional Blocks: Addition

- Binary addition used frequently
- Addition Development:

ANDing المسؤول عن عملية block diagrame

  - ***Half-Adder (HA)***: a 2-input bit-wise addition functional block
  - ***Full-Adder (FA)***: a 3-input bit-wise addition functional block
  - ***Ripple Carry Adder***: an iterative array to perform vector binary addition

Functional Block: Half-Adder

- A 2-input, 1-bit width binary adder that performs the following computations:

X	0	0	1	1
+Y	+0	+1	+0	+1
CS	00	01	01	10

- A half adder adds two bits to produce a two-bit sum
- The sum is expressed as a *sum bit (S)* and a *carry bit (C)*
- The half adder can be specified as a truth table for S and C ⇒

carry out → sum

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

Logic Simplification and Implementation: Half-Adder

- The K-Map for S, C is:

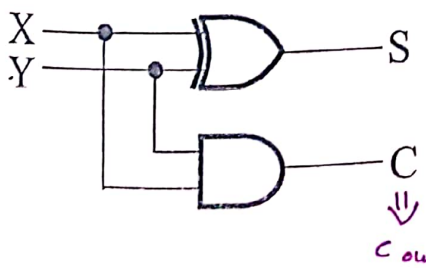
$$S = X \cdot \overline{Y} + \overline{X} \cdot Y = X \oplus Y$$

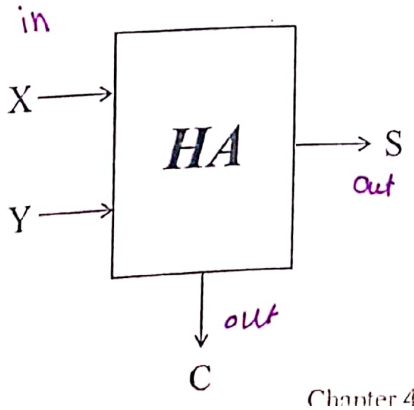
$$C = X \cdot Y$$

S	$\overline{Y}$	Y
$\overline{X}$	0	1
X	1	3

C		Y
	0	1
X	2	1

- The most common half adder implementation is:





*بيفرق الـ full عن الـ half في أنه 3-bits,
carry in كمان الـ full الـ 2-bits أنه

3-input ≡ 3-bits

Functional Block: Full-Adder

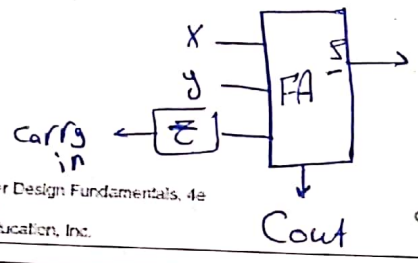
- A full adder is similar to a half adder, but includes a carry-in bit from lower stages. Like the half-adder, it computes a *sum bit (S)* and a *carry bit (C)*

- For a carry-in (Z) of 0, it is the same as the half-adder:

Z	0	0	0	0
X	0	0	1	1
+Y	+0	+1	+0	+1
CS	00	01	01	10

- For a carry- in (Z) of 1:

Z	1	1	1	1
X	0	0	1	1
+Y	+0	+1	+0	+1
CS	01	10	10	11



لازم أكتبها على العينة؟
full آه

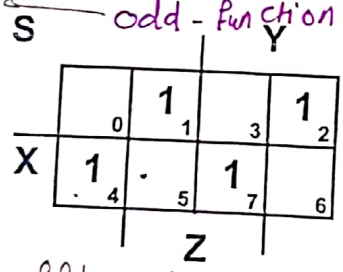
Logic Optimization: Full-Adder

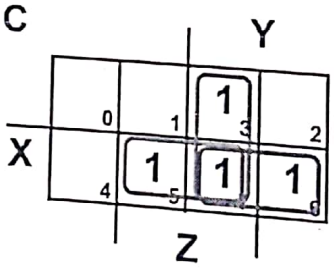
- Full-Adder Truth Table:

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

- Full-Adder K-Map:

odd - function ≡ XOR





$$S = \overset{001}{\bar{X}\bar{Y}Z} + \overset{010}{\bar{X}Y\bar{Z}} + \overset{100}{X\bar{Y}\bar{Z}} + \overset{111}{XYZ} \qquad C = XZ + XY + YZ$$

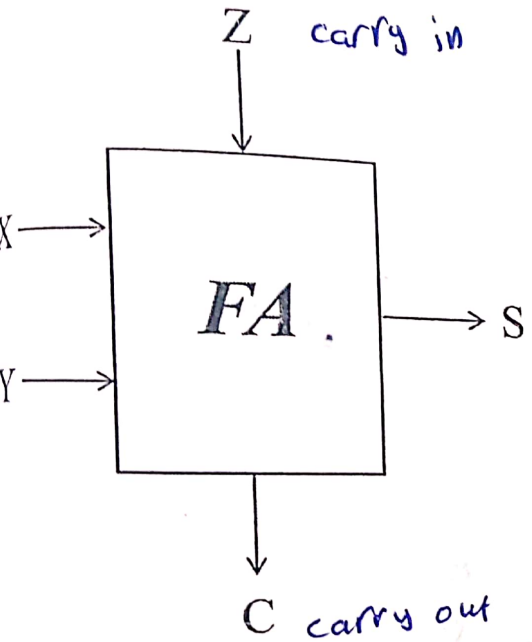
1 = على الأقل يكون 2inputs تعريف majority function

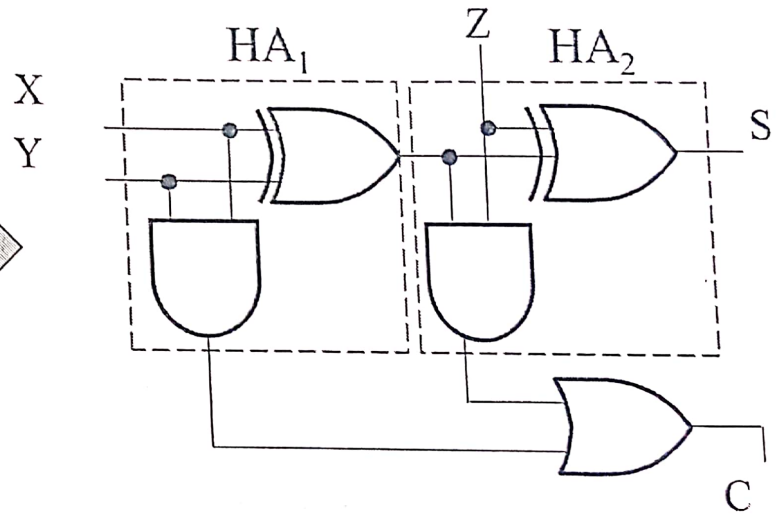
- The S function is the three-bit XOR function (Odd Function):
  - $S = X \oplus Y \oplus Z$

- The Carry bit C is 1 if both X and Y are 1 (the sum is 2), or if the sum is 1 and a carry-in (Z) occurs. Thus C can be re-written as:
  - $C = XY + (X \oplus Y)Z$

Implementation: Full Adder

Common أكثر
1-gate لأنه بتوفر





Binary Adders

- To add multiple operands, we "bundle" logical signals together into vectors and use functional blocks that operate on the vectors
- Example: *4-bit ripple carry adder* adds input vectors A(3:0) and B(3:0) to get a sum vector S(3:0)
- Note: carry-out of *cell i* becomes carry-in of *cell i + 1*

Description	Subscript 3 2 1 0	Name
Carry In	0 1 1 0	C_i
Augend	1 0 1 1	A_i
Addend	0 0 1 1	B_i
Sum	1 1 1 0	S_i
Carry out	0 0 1 1	C_{i+1}

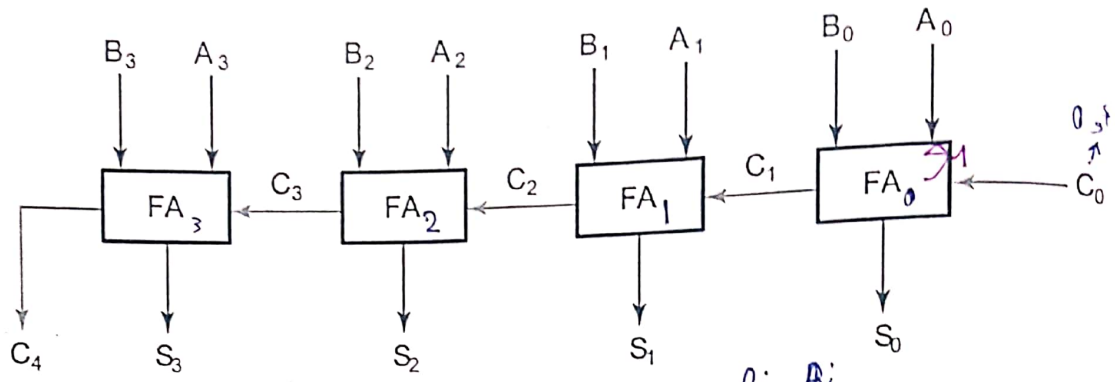
A_i B_i

Full Adder

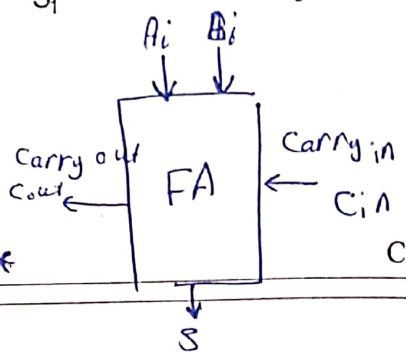
RCA

4-bit Ripple-Carry Binary Adder

- A four-bit Ripple Carry Adder made from four 1-bit Full Adders:



* Carry out of cell $\boxed{i}$ is the carry in of cell $\boxed{i+1}$

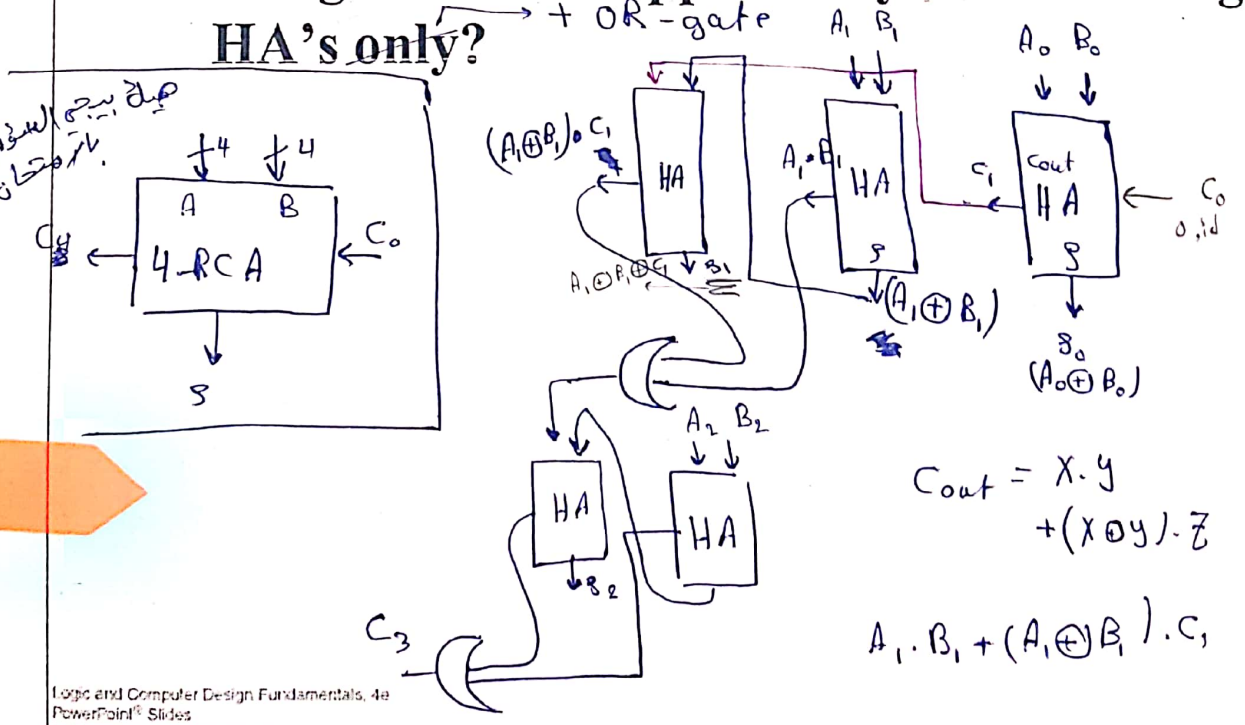


Homework

* لازم دائماً نعتمد على الـ equation عشان نعرف حلنا خطأ أو لا

~~half Adder 7 / 8~~ ممكن أحله بـ

- **Design a 4-bit ripple-carry adder using HA's only?** → + OR-gate



$C_{out} = X \cdot y + (X \oplus y) \cdot Z$

$A_1 \cdot B_1 + (A_1 \oplus B_1) \cdot C_1$

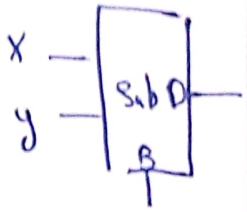
Unsigned Subtraction

- When we subtract one bit from another, two bits are produced: *difference bit (D)* and *borrow bit (B)*

X	0_0	1_0	0_1	0_1
− Y	− 0	− 1	− 0	− 1
B D	0 0	1 1	0 1	0 0

B=0 ∴ B عادية

B=1 ∴ B لا تعاد



- **Algorithm:**
  - Subtract the *subtrahend (N)* from the *minuend (M)*
  - If no end borrow occurs, then M ≥ N and the result is a non-negative number and correct
  - If an end borrow occurs, then N > M and the difference ($M - N + 2^n$) is subtracted from 2^n, and a minus sign is appended to the result

$$\begin{array}{r} 1_0 \\ -\ 1 \\ \hline 1 \end{array} \qquad n=1 \qquad M-N+2^1 = 0-1+2 = 1$$

→ Magnitude بس ال بدون إشارة $2^n - (M-N+2^n)$

Magnitude بس الـ

Unsigned Subtraction

بدون إشارة
كلهم positive
يخص

$2^n - (M - N + 2^n)$

$2^n - M + N - 2^n$

$\boxed{N - M}$

- **Examples:**

$4 - 7 + 2^4 = 13$

$\boxed{1\ 0000} \rightarrow 2^4 = 16$

$-\ 1101 \rightarrow$ 4-bits

0011

* العمليتين إذا طلع معاه borrow لازم أكمل على الطريقة

0	1	1	0	1
1001 9	0100 M=4	10011	10010110	01100100
−0111 7	−0111 N=7	−11110	−01100100	−10010110
0010	1101	10101	00110010	11001110
	13			
	10000	100000		100000000
	−1101	−10101		−11001110
	(−) 0011	(−) 01011		(−) 00110010

خطأ ← يكون لازم أكمل

بالامتحان بحط (−) ؟!

لا

I_0 $S(\text{Subtract}/\overline{\text{Add}}) = 1$ 0

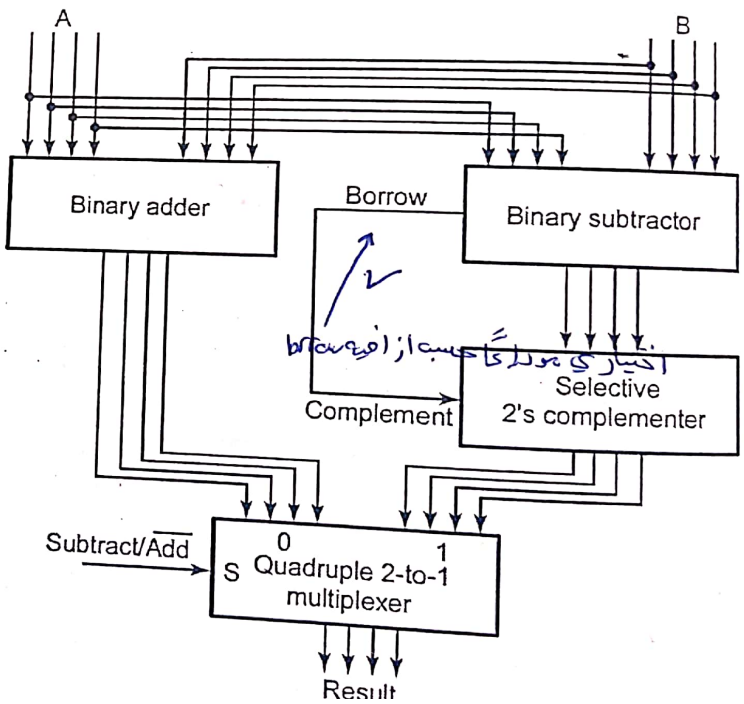
I_1

Unsigned Subtraction (continued)

- The subtraction, $2^n - D$, is taking the *2's complement of D*
- To do both unsigned addition and unsigned subtraction requires:
  - Addition and Subtraction are performed in parallel and Subtract/$\overline{Add}$ chooses between them

S

- Quite complex!
- *Goal:* Shared simpler logic for both addition and subtraction
- Introduce complements as an approach



Complements

- For a number system with radix (r), there are two complements:
  - *Diminished Radix Complement*
    - Famously known as *(r – 1)'s complement*
    - Examples:
      - **1's complement for radix 2**
      - **9's complement for radix 10**
    - For a number (N) with n-digits, the diminished radix complement is defined as:
      - $(r^n - 1) - N$ → هاد دايماً بطلع عنده بالحسبي ولازم أحوله للصيغة المطلوبة
  - *Radix Complement*
    - Famously known as *r's complement* for radix r
    - Examples:
      - **2's complement in binary**
      - **10's complement in decimal**
    - For a number (N) with n-digits, r's complement is defined as:
      - → $r^n - N$, *when* $N \neq 0$
      - → 0, *when* $N = 0$

Diminished Radix Complement ≡ (r-1)'s complement

radix

- If N is a number of n-digits with radix (r), then
  - $N + (r-1)'s\ complement\ of\ N = \underbrace{(r-1)(r-1)(r-1)\ldots(r-1)}_{\text{n-digits}}$
  - *The (r – 1)'s complement can be computed by subtracting each digit from (*

* يعني لو طلب منّي التحويل
4's complement of $(21)_5$
بعكس
$\cancel{55}\ 44 - 21 = (34)_5$

→ ▪ Example: Find 1's complement of $(1011)_2$ — المطلوب
- r = 2, n = 4 → the number of digits
- Answer is $(2^4 - 1) - (1011)_2 = (0100)_2$ → $(15)_{10}$ → بحولها من Decimal binary
- Notice that $(1011)_2 + (0100)_2 = (1111)_2$ which is $\underbrace{(2-1)(2-1)(2-1)(2-1)}_{\text{4-digits}}$

Decimal
→ ▪ Example: Find 9's complement of $(45)_{10}$
- r = 10, n = 2
- Answer is $(10^2 - 1) - (45)_{10} = (54)_{10}$
- Notice that $(45)_{10} + (54)_{10} = (99)_{10}$ which is $\underbrace{(10-1)(10-1)}_{\text{2-digits}}$

* N + (r-1)'s complement of N
$= \underbrace{(r-1)(r-1)(r-1)\ldots}_{\text{n-times}}$

$(45)_{10} + (54)_{10} = (99)_{10}$

→ ▪ Example: Find 7's complement of $(671)_8$
- r = 8, n = 3 — لأنه أحولها octal
- Answer is $(8^3 - 1) - (671)_8 = (106)_8$ ✓
- Notice that $(671)_8 + (106)_8 = (777)_8$ which is $\underbrace{(8-1)(8-1)(8-1)}_{\text{3-digits}}$

$(511)_{10} \rightarrow (777)_8$

$(777)_8 - (671)_8$

$\cancel{FFFFF}$
$N = (3AD56)_{16}$

15's complement of N $= (C52A9)_{16}$

Binary 1's Complement

- For r = 2, N = 01110011_2, n = 8 (8 digits):

  $(r^n - 1) = 256 - 1 = 255_{10}$ or 11111111_2

- The 1's complement of 01110011_2 is then:

لو ما بدي أحسبه بطلع جواب تاني

لازم أرجع بعد الدفع إلى مضيفة لحياهم

$$\begin{array}{rr} & 11111111 \\ - & \underline{01110011} \\ & 10001100 \end{array}$$

كأنو عملت complement في حالة binary

بس بالثنائي Binary

- Since the $2^n - 1$ factor consists of all 1's and since $1 - 0 = 1$ and $1 - 1 = 0$, the one's complement is obtained *by complementing each individual bit (bitwise NOT)*

Radix Complement

- For number N with n-digit and radix
  - If N ≠ 0, r's complement of N = $r^n - N$
    - *r's complement = (r-1)'s complement +*
  - If N = 0, r's complement of N = 0
- Example: Find 10's complement of $(92)_{10}$
  - r = 10, n = 2
  - Answer is $10^2 - (92)_{10} = (8)_{10}$
  - Notice that 9's complement of $(92)_{10}$ is $(7)_{10}$
  - ***10's complement = 9's complement + 1***
- Example: Find 16's complement of $(3AE7)_{16}$
  - r = 16, n = 4
  - Answer is $16^4 - (3AE7)_{16} = (10000)_{16} - (3AE7)_{16} = (C519)_{16}$
  - 15's complement = $(C518)_{16}$ → 16's complement = $(C518)_{16} + 1 = (C519)_{16}$

Binary 2's Complement

- For r = 2, N = 01110011_2, n = 8 (8 digits), we have:
  - $(r^n) = 256_{10}$ or 100000000_2
- The 2's complement of 01110011 is then:

$$\begin{array}{r} 100000000 \\ -\ \underline{01110011} \\ 10001101 \end{array}$$

- *Note the result is the 1's complement plus 1, a fact that can be used in designing hardware*
- *Remember the 2's complement of $(000..00)_2$ is $(000..00)_2$*
- *Complement of a complement restores the number to its original value:*
  - *The Complement of complement* $N = 2^n - (2^n - N) = N$

Alternate 2's Complement Method

بس لل Binary

- Given: an n-bit binary number, beginning at the least significant bit and proceeding upward:
  - **Copy all least significant 0's**
  - **Copy the first 1**
  - **Complement all bits thereafter**

بدور على أول digit قيمتها 1 بعدين بعكسه اللي بعدها

أول رقم 1

1001 0100

0110 1100

بعكسهم

نزل زي ما هو

- 2's Complement Example:

  10010<u>100</u>

  - **Copy underlined bits:**

    <u>100</u>

  - **and complement bits to the left:**

    <u>01101</u>100

Subtraction with 2's Complement

- **For n-digit, <u>unsigned</u> numbers M and N, find M – N in base 2:**
  - Add the 2's complement of the subtrahend N to the minuend M:
    - $\mathbf{M - N \Longrightarrow M + (2^n - N) = M - N + 2^n}$

end carry
$M \geq N$
$(M-N)$ is correct

no end carry / $N > M$
$N - M$
$2^n - (M - N + 2^n) = N - M$

- If $M \geq N$, the ***sum*** produces end carry 2^n which is discarded; and from above, M – N remains
- If $M < N$, the ***sum*** does not produce end carry, and from above, is equal to $2^n - (N - M)$ which is the 2's complement of (N – M)
- To obtain the result – (N – M) , take the 2's complement of the sum and place a "–" to its left

Unsigned 2's Complement Subtraction Example: (M > N)

- Find $01010100_2 - 01000011_2$

	01010100		1 01010100 → 8-bits
–	01000011	2's comp →	+ 10111101 → 8-bits
		end carry	1 00010001

ازا طلع عندي آخر carry ∴ 8 بتجاهل القيمة الأخيرة وباخذ الرقم لحالهم بتجاهلها

- The carry of 1 indicates that no correction of the result is required

*ازا طلع عندي 1 آخر شي على الشمال بعتبر أنه الجواب صح لا بده تعمله و بحذف آخر رقم يلي هو الـ 1

Unsigned 2's Complement Subtraction Example: (M < N)

- Find $01000011_2 - 01010100_2$

	01000011		0 01000011 → 8-bits
–	01010100	2's comp →	+ 10101100 → 8-bits
			11101111 ← 8-bits ∴ ما في carry ∴ الجواب correct لازم it's not 2's comp عالـ
		2's comp →	00010001

- The carry of 0 indicates that a correction of the result is required

- Result = – (00010001)

2 option
MUX وبدي أختار واحدة منهم

Addition I_0
Subtraction I_0 أو I_1

2's Complement Adder/Subtractor for Unsigned Numbers

- *Subtraction can be done by addition of the 2's Complement*
  1. Complement each bit (1's Complement)
  2. Add 1 to the result

$X \oplus 0 = X$

$X \oplus 1 = \overline{X}$

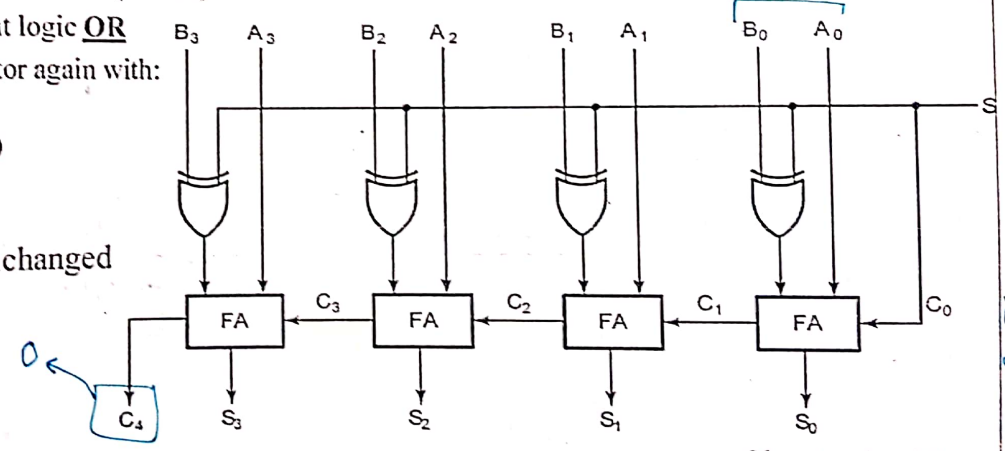
- The circuit shown computes A + B and A – B:
- Subtract (S = 1): $A - B = A + (2^n - B) = A + \overline{B} + 1$
  - The 2's complement of B is formed by using XORs to form the 1's complement and adding the 1 applied to C_0
  - If $C_4 = 1$ ($A \geq B$): correct result
  - If $C_4 = 0$ ($A < B$): result $= 2^n - (B - A)$
    - Use 2's complement logic **OR**
    - Use Adder/Subtractor again with:
      - $A = 0$
      - $B = 2^n - (B - A)$
- Add (S = 0): A + B
  - B is passed through unchanged

$\boxed{C_0 = 0}$ $A + B$

$\boxed{C_0 = 1}$ $A + \overline{B} + 1$ $(A - B)$

C_0 → 1, 0

input Subtraction Addition



input
اذا إلي بنتحكم فيه

Signed Integers

- ***Positive numbers and zero*** can be represented by unsigned ***n-digit, radix r*** numbers. ***We need a representation for negative numbers***
- To represent a sign (+ or –) we need exactly one more bit of information (1 binary digit gives $2^1 = 2$ elements which is exactly what is needed).
- Since computers use binary numbers, by convention, ***the most significant bit is interpreted as a sign bit***:

Sign

(-) (+) 0 1

$$s\ a_{n-2} \ldots a_2 a_1 a_0$$

where:

s = 0 for Positive numbers

s = 1 for Negative numbers

and a_i = 0 or 1 represent the magnitude in some form

2's complement $-(2)^3\ 2^2\ 2^1\ 2^0$ 1101 $= -8+4+0+1 = -3$

بعد بالنزبط

Signed Integer Representations

- ***Signed-Magnitude:*** here the (n – 1) digits are interpreted as a positive magnitude
  - Max = + (2^{n-1} – 1)
  - Min = – (2^{n-1} – 1)
  - Two representation for zero (i.e. ± 0)

1011 → −3 0011 → +3

- ***Signed-Complement:*** here the digits are interpreted as the rest of the complement of the number. There are two possibilities here:
  - ***Signed 1's Complement:*** Uses 1's Complement Arithmetic
    - Max = + (2^{n-1} – 1)
    - Min = – (2^{n-1} – 1)
    - Two representation for zero (i.e. ± 0)

n-bit $+(2^{n-1}-1)$ +7 (0111)

4-bit $-(2^{n-1}-1)$ −7 (1111)

  - ***Signed 2's Complement:*** Uses 2's Complement Arithmetic
    - Max = + (2^{n-1} – 1)
    - Min = – 2^{n-1}
    - Single representation for zero

حل مشكلة الـ zero

Range $(2^{n-1}-1), -2^{n-1}$

1's complement

0011 → 3 1100 → −3

وهاي بس الاشارة

0100 → +4 1011 → −4

$\mp(2^{n-1}-1)$

Signed Integer Representation Example

- **r = 2, n = 3**

بعد عكس الاشارة

Number	Signed-Magnitude	1's Complement	2's Complement
+3	011	011	011
+2	010	010	010
+1	001	001	001
+0	000	000	000
-0	100	111	----
-1	101	110	111
-2	110	101	110
-3	111	100	101
-4	----	----	100

*Only in 2's complement

$-(2)^2\ 2^1\ 2^0$ 111 $= -4+2+1 = -1$

$-(2)^3\ 2^2\ 2^1\ 2^0$ 1111 $= -8+4+2+1 = -1$

كل

- **Represent the number -9 using 8-bits**
  - Sign-Magnitude = 10001001
  - 1's complement = 11110110
  - 2's complement = 11110111

more Negative

0000 1001

$(-9)_{10}$

دائماً بنبدا من اليمين

2's Complement Signed Numbers

- Signed 2's complement is the most common representation for signed numbers
  - Focus of the course

- For any n-bit 2's complement signed number ($b_{n-1}b_{n-2}b_{n-3} \ldots b_2b_1b_0$), the decimal value is given by

$$Value = (-2^{n-1} \times b_{n-1}) + \sum_{i=0}^{n-2} 2^i \times b_i$$

- Example: What is value of the 2's complement number $(100111)_2$?

$$Value = -2^5 \times 1 + 7 = -25$$

$-(2)^5$ 2^4 2^3 2^2 2^1 2^0

1 0 0 1 1 1 $= -32 + 4 + 2 + 1 = \boxed{-25}$

Signed-2's Complement Arithmetic

- **Addition:**
  - **Add the numbers including the sign bits**
  - **Discard the carry out of the sign bits**

- **Subtraction:**
  - **Form the complement of the number you are subtracting**
  - **Follow the same rules for addition**
  - $A - B = A + (-B) = A + (\overline{B} + 1)$

    Negation $\hookrightarrow (2^n - B)$

Signed 2's Complement Addition

```
(+6)   00000110
+      +
(+13)  00001101
       00010011 (+19)
```

```
(-6)   11111010
+      +
(+13)  00001101
       100000111 (+7)
```

Carry-out is ignored

```
(+6)   00000110
+      +
(-13)  11110011 → 2'comp
       11111001 (-7)
```

```
(-6)   11111010
+      +
(-13)  11110011
       111101101 (-19)
```

Carry-out is ignored

Signed 2's Complement Subtraction

```
(+6)   00000110
-      +
(+13)  11110011
       11111001 (-7)
```

```
(-6)   11111010
-      +
(+13)  11110011
       111101101 (-19)
```

Carry-out is ignored

① بحول الى سالب
② بحسب 2's complement

00001 11
-7 ∴ 7

```
(+6)   00000110
-      +
(-13)  00001101
       00010011 (+19)
```

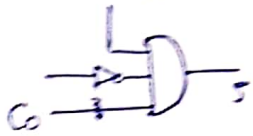
```
(-6)   11111010
-      +
(-13)  00001101
       100000111 (+7)
```

Carry-out is ignored

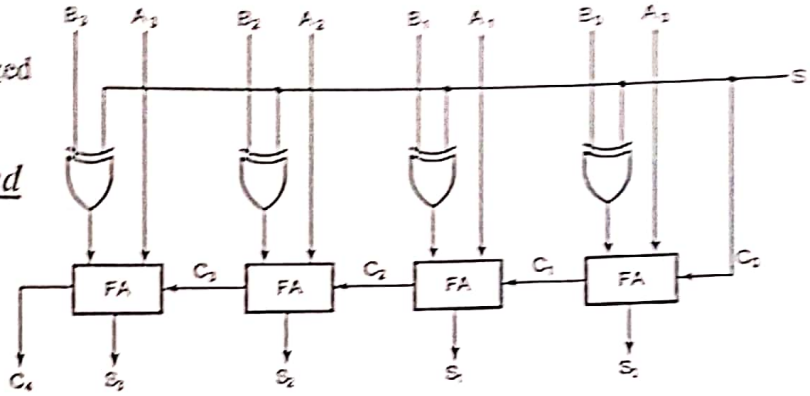
2's Complement Adder/Subtractor for Signed Numbers

signed / unsigned

- *Subtraction can be done* ***by addition of the 2's Complement***
  1. Complement each bit (1's Complement)
  2. Add 1 to the result
- The circuit shown computes A + B and A – B:



- Subtract (S = 1): $A - B = A + (2^n - B) = A + \bar{B} + 1$
  - The 2's complement of B is formed by using XORs to form the 1's complement and adding the 1 applied to C_0
- Add (S = 0): A + B
  - B is passed through unchanged
- ***Same Hardware for Signed and Unsigned numbers***



Overflow Detection

* result
n+1 n-bit

- In computers, the number of bits is fixed
- *Overflow* occurs if $n + 1$ bits are required to contain the result from an n-bit addition or subtraction

[c_1 ,

- *Unsigned number overflow* is detected from the **end carry-out** when *adding* two unsigned numbers
  - **Overflow is *impossible* for unsigned subtraction**

unsigned

$$\begin{array}{r} 1000 \\ +\ 1000 \\ \hline \text{carry } \boxed{1}0000 \end{array}$$

OV = 1

OV = C_n

(8)	1000
+	+
(12)	1100
	$\boxed{1}$0100 (4)

Carry-out = 1 → Overflow

- *Signed number overflow* can occur for:
  - **Addition of two operands with the same sign**
  - **Subtraction of operands with different signs**

Signed-number Overflow Detection

- Signed number cases with carries C_n and C_{n-1} shown for correct result signs:

0 0	0 0	1 1	1 1
0	0	1	1
+ 0	− 1	− 0	+ 1
0	0	10	11

Addition

0 111 (+7)
0 111 (+7)
0 1110

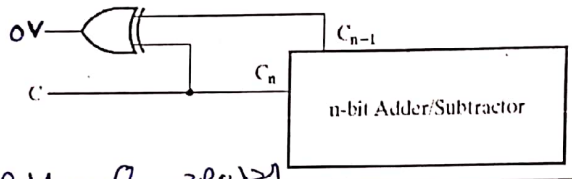
- Signed number cases with carries shown for erroneous result signs (indicating overflow):

0 1	0 1	1 0	1 0
0	0	1	1
+ 0	− 1	−0	+ 1
1	1	0	0

1000 (-8)
1000 (-8)
$\boxed{1}$0000

0111 (+7)
1000 (-8)
$\boxed{-1}$ 1111

- *Simplest way to implement signed overflow is:* $OV = C_n \oplus C_{n-1}$



OV = 0 الجواب صح
OV = 1 الجواب خطأ

Signed-number Overflow Examples

$n = 8$, $2^n - 1 = 2^8 - 1$ ← 0-to-255 ← لو كانت unsigned

- 8-bit signed number range between: -128 to +127

في طريقتين اخرى آخرى فيه OV

(+70) 01000110	(-70) 10111010
+ +	+ +
(+80) 01010000	(-80) 10110000
10010110 (-106)	101101010 (+106)
$V = C_7 \oplus C_8 = 1 \oplus 0 = 1$	$V = C_7 \oplus C_8 = 0 \oplus 1 = 1$

موجب
موجب
OV : سالب
150 = carry
2

(+70) 01000110	(-70) 10111010
$\boxed{-}$ +	+ $\boxed{-}$
(-80) 01010000	(+80) 10110000
10010110 (-106)	101101010 (+106)
$V = C_7 \oplus C_8 = 1 \oplus 0 = 1$	$V = C_7 \oplus C_8 = 0 \oplus 1 = 1$

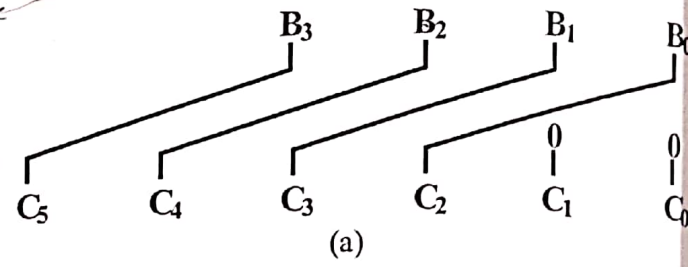
بس انتاج الاجابة خاطئة

30
40
70 → Range لأنه خرج عن ال

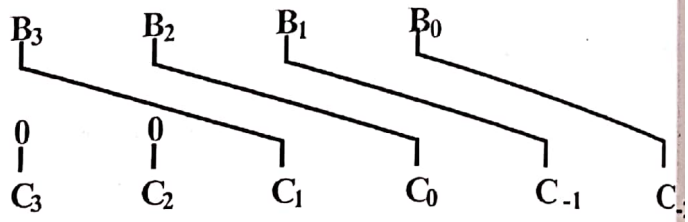
ولا OV

Multiplication/Division by 2^n

- Multiplication by 2^n: Shift left by n
  - Add n-zeros on the left
- Division by 2^n: Shift right by n
  - Add n-zeros on the right
- Multiplication by $(100)_2$
  - Shift left by 2
- Division by $(100)_2$
  - Shift right by 2
  - Quotient = $C_3C_2C_1C_0$
  - Remainder = $C_{-1}C_{-2}$



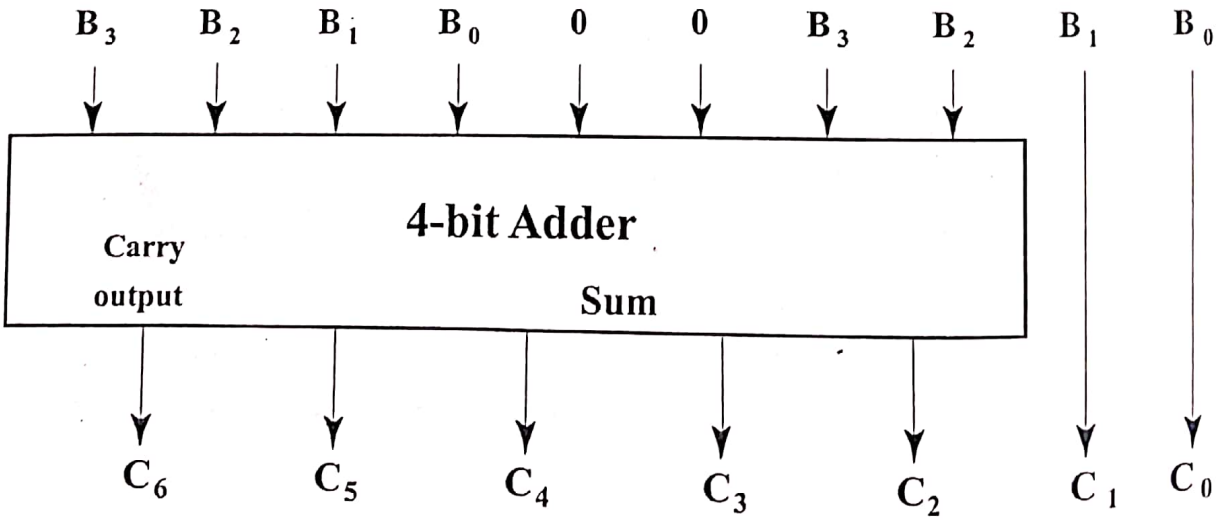
(a)



(b)

Multiplication by a Constant

- Multiplication of B(3:0) by 101
- See text Figure 4-10 in page 171 for contraction



Zero Fill

unsigned numbers

- ***Zero fill:*** filling an m-bit operand with 0s to become an n-bit operand with $n > m$
- Filling usually is applied to the MSB end of the operand, but can also be done on the LSB end
- **Example: 11110101 filled to 16 bits**
  - **MSB end: 0000000011110101** (Zero Extension)
  - **LSB end: 1111010100000000**

لأنها ما غيرت بقيمة الرقم

على اليسار

signed

1001 00001001

extension -7 → +9

Extension

1111 1001 00000111

باخذ الاشارة وبكررها 7

- ***Extension:*** increase in the number of bits at the MSB end of an operand by using a complement representation
  - **Copies the MSB of the operand into the new positions**
  - **Positive operand example - 01110101 extended to 16 bits:**

    0000000001110101
  - **Negative operand example - 11110101 extended to 16 bits:**

    11111111/11110101

Multiplication/Division by 2^n

بالكتاب
171

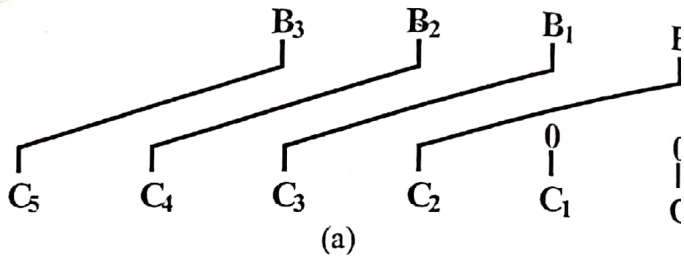
بالضرب بقوى الرقم

- Multiplication by 2^n: Shift left by n
  - Add n-zeros on the left
- Division by 2^n: Shift right by n
  - Add n-zeros on the right

بالقسمة بنصغر الرقم

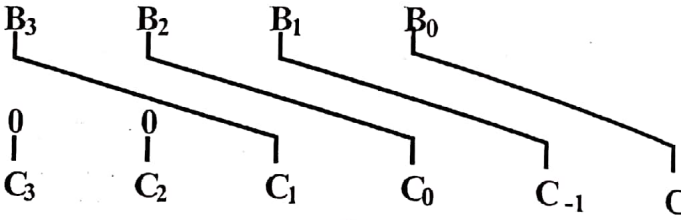
- Multiplication by $(100)_2$
  - Shift left by 2

421
$=4$
$=2^2$



(a)

- Division by $(100)_2$
  - Shift right by 2
  - Quotient = $C_3C_2C_1C_0$
  - Remainder = $C_{-1}C_{-2}$

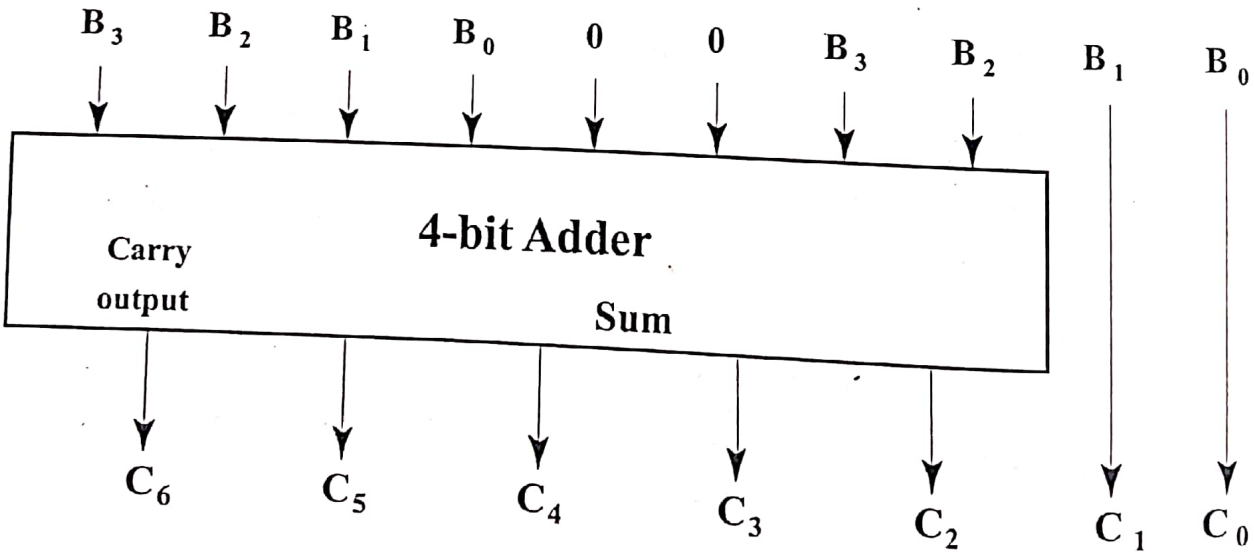


(b)

Decimal point

Multiplication by a Constant

- Multiplication of B(3:0) by 101
- See text Figure 4-10 in page 171 for contraction



Zero Fill unsigned numbers

- *Zero fill:* filling an m-bit operand with 0s to become an n-bit operand with $n > m$
- Filling usually is applied to the MSB end of the operand, but can also be done on the LSB end
- **Example: 11110101 filled to 16 bits**
  - **MSB end: 0000000011110101** (Zero Extension) لانها ما غيرت بقيمة الرقم
  - **LSB end: 1111010100000000**

علا اليسار

signed

1001	00001001
extension -7 →	+9

Extension

1111 1001 باخذ الاشارة وبكررها

00000111
7

- *Extension:* increase in the number of bits at the MSB end of an operand by using a complement representation
  - **Copies the MSB of the operand into the new positions**
  - **Positive operand example - 01110101 extended to 16 bits:**

    0000000001110101
  - **Negative operand example - 11110101 extended to 16 bits:**

    11111111/11110101

Comparator A, B 4 bit
A < B A - B < 0
A = B ⇐ A - B = 0

Hexadecimal, Octal, BCD Addition

A > B A - B > 0

A + B + 1 = A - B

- *Hexadecimal and Octal Addition:*
  - Add each digit then take modulus (r)

$C_4 = 0$ A < B
$C_4 = 1$ A > B

unsigned $A_4 A_3 A_2 A_1$ $B_4 B_3 B_2 B_1$

$(59F)_{16}$	$(762)_8$
+	+
$(E46)_{16}$	$(345)_8$
$(13E5)_{16}$	$(1327)_8$

less (A < B) unsigned → 0

sign bit
less (A < B) signed
unsigned = C_4 → 1 Equal (A = B)
signed = S_3

- *BCD Addition:*
  - Add each 4-bit together
    - If the binary sum is greater than 1001
      - Add 0110 to the result

$(448)_{10}$	$(0100\ 0100\ 1000)_B$
+	+
$(489)_{10}$	$(0100\ 1000\ 1001)_B$
$(937)_{10}$	1001 1101 0001
	+ 0110 0110
	1001 0011 0111

A = 1000 (−8)
B = 0011 (+3)

A − B
1000
+ 1101
1 0101
C_4 C_3

sign num ← OV = $C_4 \oplus C_3$
= 1 ⊕ 0
= 1

Terms of Use

signed num

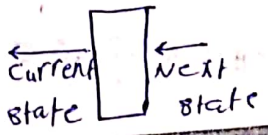
S_3	OV	A < B
0	0	0
0	1	1
1	0	1

→ $S_3 \oplus$ OV

C_4
C_3 → OV
S_3
less A < B signed

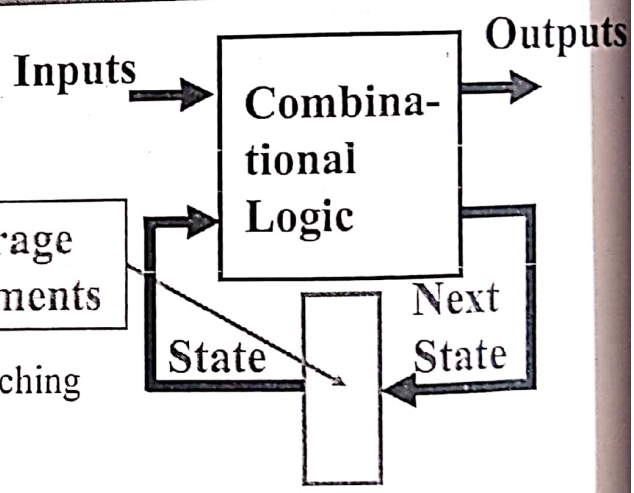
greater (A > B) signed

Overview



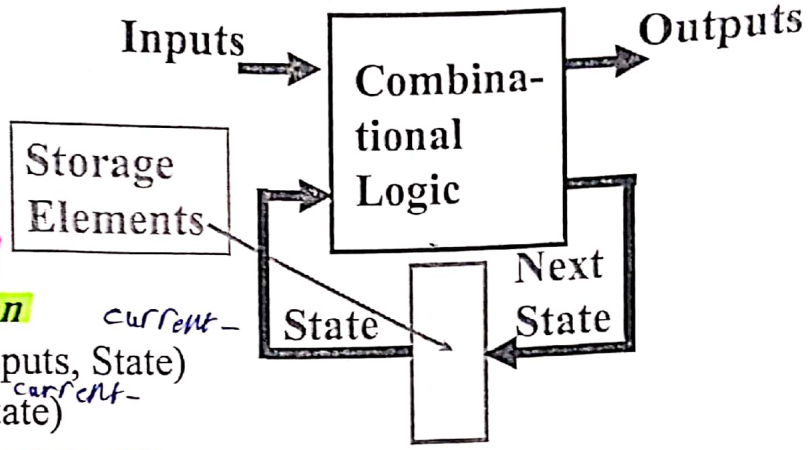
- Part 1 - Storage Elements and Analysis
  - Introduction to sequential circuits
  - Types of sequential circuits
  - Storage elements
    - Latches
    - Flip-flops
  - Sequential circuit analysis
    - State tables
    - State diagrams
    - Equivalent states
    - Moore and Mealy Models
- Part 2 - Sequential Circuit Design

Introduction to Sequential Circuits



- A Sequential circuit contains:
  - ***Storage elements:***
    - Latches or Flip-Flops
  - ***Combinational Logic:***
    - Implements a multiple-output switching function
    - Inputs are signals from the outside
    - Outputs are signals to the outside
    - Other inputs, State or Present State, are signals from storage elements
    - The remaining outputs, Next State are inputs to storage elements

Introduction to Sequential Circuits



- **Combinatorial Logic**
  - ***Next state function***
    Next State = f(Inputs, State)
    OR Next State = f(State)
  - ***Output function (Mealy)***
    Outputs = g(Inputs, State)
  - ***Output function (Moore)***
    Outputs = g(State)
- **Output function type depends on specification and affects the design significantly**

Types of Sequential Circuits

- **Depends on the times at which:**
  - storage elements observe their inputs, and
  - storage elements change their state
- **Synchronous**
  - Behavior defined from knowledge of its signals at *discrete* instances of time
  - **Storage elements observe inputs and can change state only in relation to a timing signal (*clock pulses* from a *clock*)**
  - *Simple to design but slow*
- **Asynchronous**
  - Behavior defined from knowledge of inputs at any instant of time and the order in *continuous* time in which inputs change
  - *Complex to design but fast*

Storage Elements

- Any storage element can maintain a binary state indefinitely (as long as the power is on) until directed by the input signals to switch
- Storage elements: ***Latches*** and ***Flip-flops (FFs)***
- Latches and FFs differ in:
  - Number of inputs
  - Manner in which the inputs affect the binary state
- Latch:
  - Asynchronous
  - Although difficult to design, we discuss latches first because they are the building blocks for flip-flops

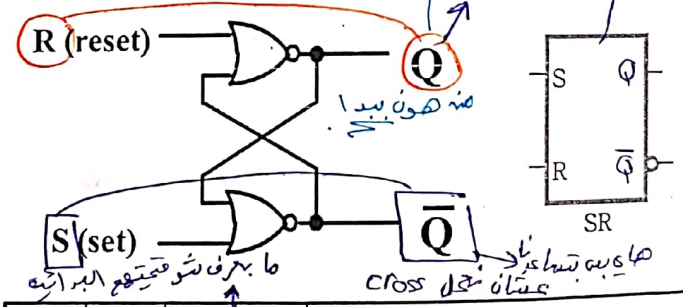
2 inputs → 2-outputs

Basic (NOR) SR Latch

Q, $\bar{Q}$ are always oppisite

- **Cross-coupling** two NOR gates Feed back loop

R	S	Q	$\bar{Q}$	Comment
0	0	Q	$\bar{Q}$	Hold, no change
0	1	1	0	Set
1	0	0	1	Reset
1	1	0	0	Not allowed

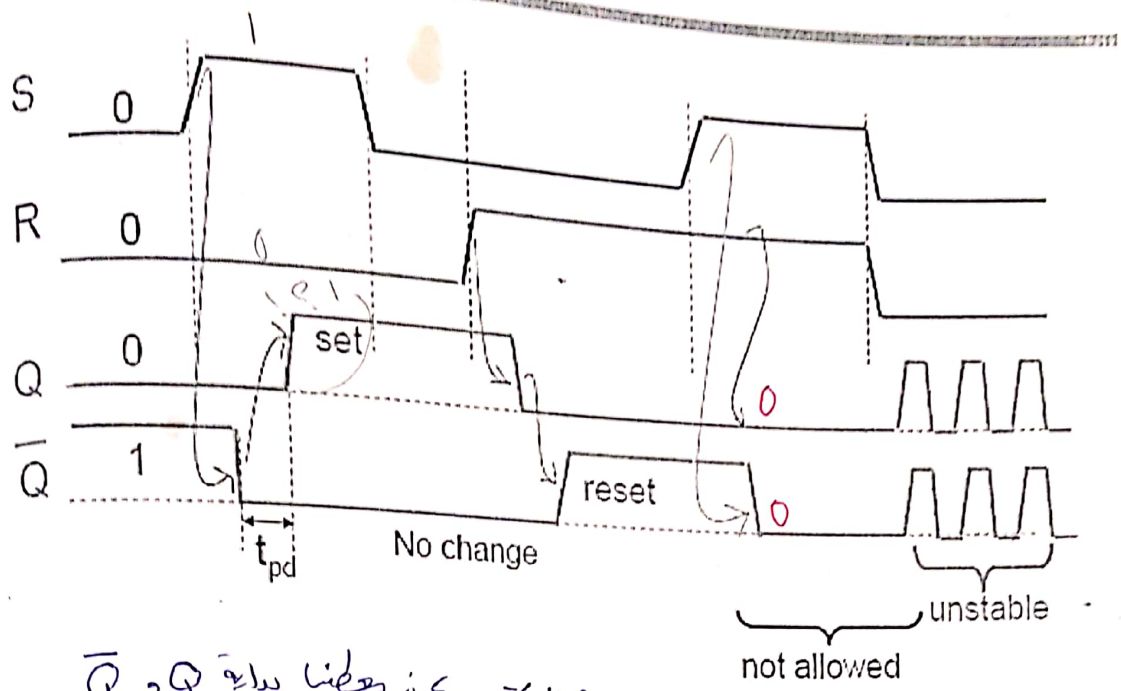


- **Time sequence behavior:**

Time	R	S	Q	$\bar{Q}$	Comment
	0	0	?	?	Stored state unknown
	0	1	1	0	"Set" Q to 1
	0	0	1	0	Now Q "remembers" 1
	1	0	0	1	"Reset" Q to 0
	0	0	0	1	Now Q "remembers" 0
	1	1	0	0	Both go low
	0	0	?	?	Unstable!

- **S = 1, R = 1 is forbidden as input pattern**

Timing Waveform of NOR SR Latch



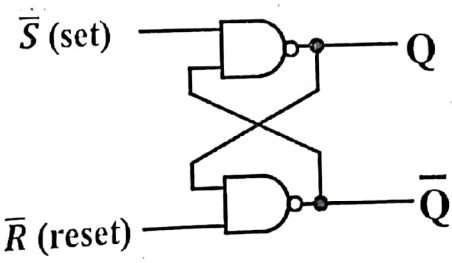
لما تيجينا رسمة بالامتحان لازم الدكتور يكون معطينا بداية Q و $\overline{Q}$
at the same time
مو معطيه منا ال delay بنعمله

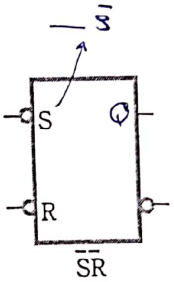
لازم أميز بالامتحان من الرسمة

Basic (NAND) $\overline{S}\overline{R}$ Latch

كأنها

- Cross-coupling two NAND gates
- *Active low inputs*





$\overline{R}$	$\overline{S}$	Q	$\overline{Q}$	Comment
0	0	1	1	Not allowed
0	1	0	1	Reset
1	0	1	0	Set
1	1	Q	$\overline{Q}$	Hold, no change

- Time sequence behavior:

Time ↓

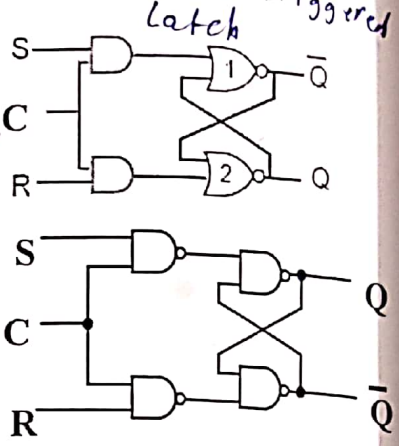
$\overline{R}$	$\overline{S}$	Q	$\overline{Q}$	Comment
1	1	?	?	Stored state unknown
1	0	1	0	"Set" Q to 1
1	1	1	0	Now Q "remembers" 1
0	1	0	1	"Reset" Q to 0
1	1	0	1	Now Q "remembers" 0
0	0	1	1	Both go high
1	1	?	?	Unstable!

- $\overline{S} = 0, \overline{R} = 0$ is forbidden as

Clocked SR Latch (Pulse-triggered Latch)

- The operation of the basic NOR and basic NAND latches can be modified by a providing a control input (C) that determines when the state of the latch can be changed
  - **Adding two AND gates to basic SR latch *OR***
  - **Adding two NAND gates to *SR* basic latch**

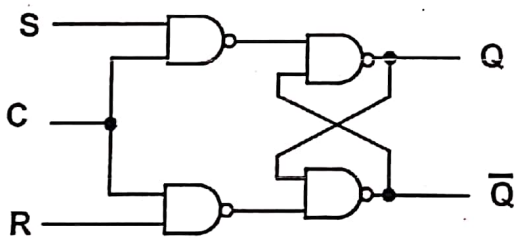
C	R	S	Q	$\bar{Q}$	Comment
0	x	x	Q	$\bar{Q}$	Hold, no change
1	0	0	Q	$\bar{Q}$	Hold, no change
1	0	1	1	0	Set
1	1	0	0	1	Reset
1	1	1			Not allowed



- Has a time sequence behavior similar to the basic S-R latch except that the S and R inputs are only observed when the line C is high
- C means "control" or "clock"

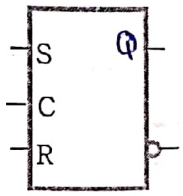
Clocked SR Latch (continued)

- The Clocked SR Latch can be described by a table:



Q(t)	S	R	Q(t+1)	Comment
0	0	0	0	No change
0	0	1	0	Clear Q
0	1	0	1	Set Q
0	1	1	???	Indeterminate
1	0	0	1	No change
1	0	1	0	Clear Q
1	1	0	1	Set Q
1	1	1	???	Indeterminate

- The table describes what happens after the clock [at time (t+1)] based on:
  - current inputs (S,R) and
  - current state Q(t)



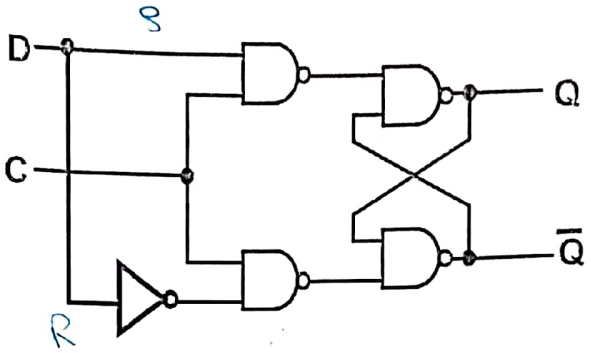
Clocked SR

Not Allowed

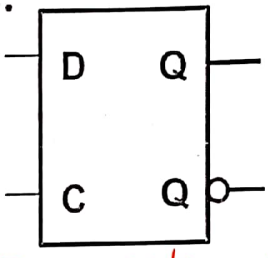
D Latch

هون مستحيل يصير ال
وهاد الهدف منه
بربط على ال NAND
وعلى ال NOR

- **Adding an inverter to the S-R Latch, gives the D Latch:**
- **Note that there are no "indeterminate" states!** الهدف منه



The graphic symbol for a D Latch is:



C	D	Q	$\bar{Q}$	Comment
0	x	Q	$\bar{Q}$	Hold, no change
1	0	0	1	Reset
1	1	1	0	Set

→ S=0 R=1

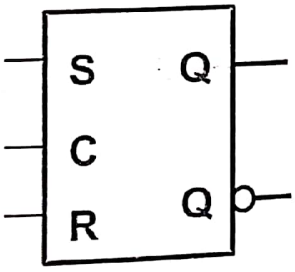
→ S=1 R=0

*ال Q نفس قيمة ال D

وبنفس الوقت بقدر أعمل D-Latch [+ve pulse-triggered
-ve pulse-triggered
بس بنحط bubble

Variations of Clocked SR and D Latches

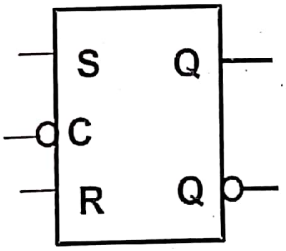
* بطلع على
C → clk
قط بتشتغل ومتى لا حسب ال bubble



+ve pulse-triggered SR latch

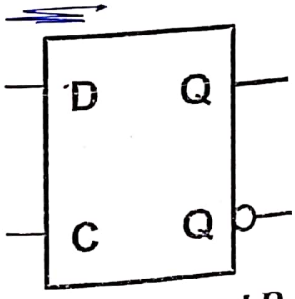
C = 0 → Hold

C = 1 → Change هون بشتغل

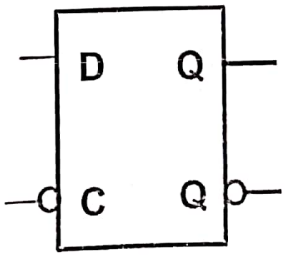


-ve pulse-triggered SR latch *C = 0 → Change* هون بشتغل

C = 1 → Hold



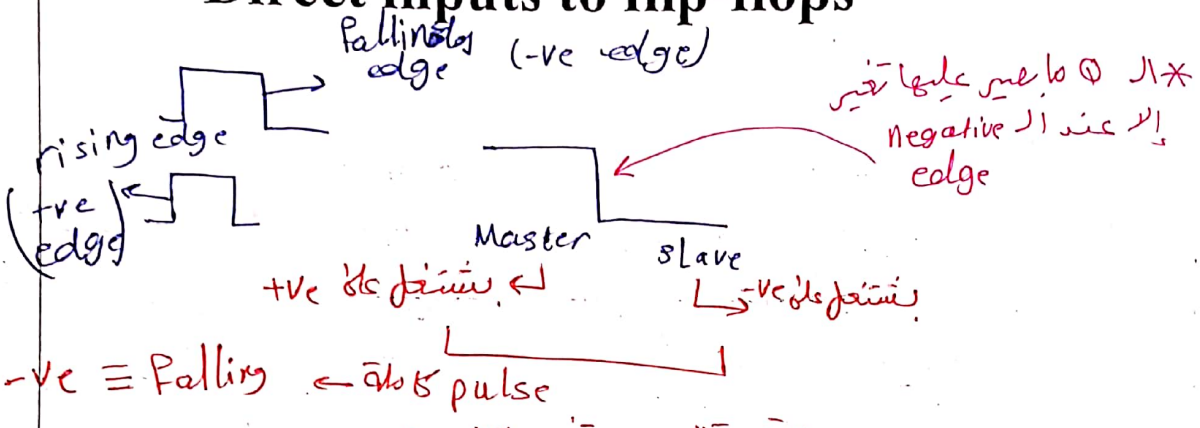
+ve pulse-triggered D latch



-ve pulse-triggered D latch

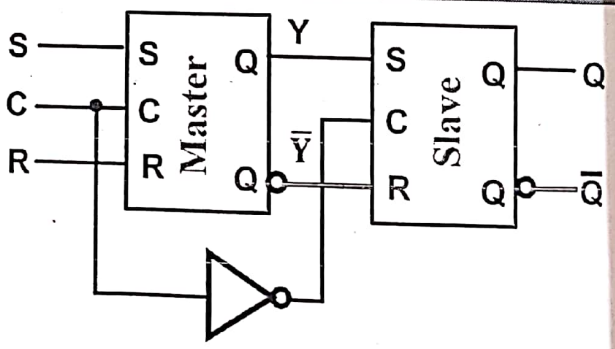
Flip-Flops

- Master-slave flip-flop
- Edge-triggered flip-flop
- Standard symbols for storage elements
- Direct inputs to flip-flops



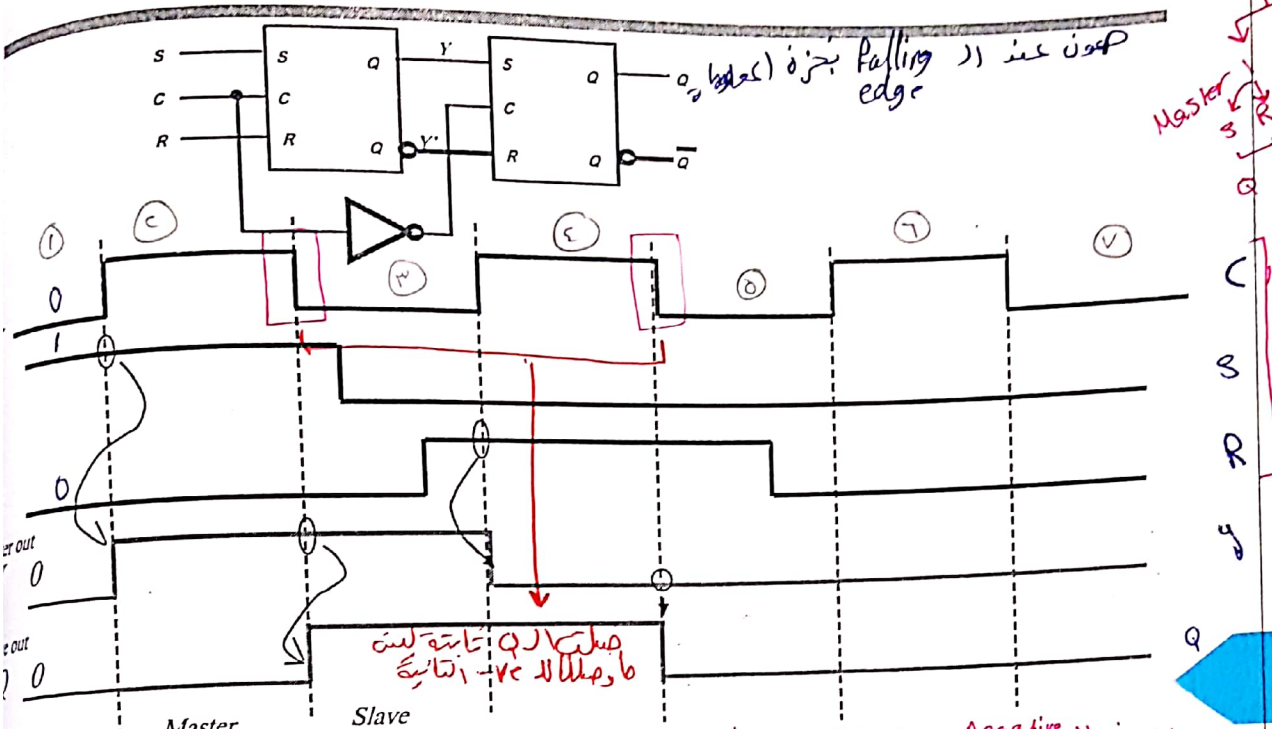
SR Master-Slave Flip-Flop

- Consists of two clocked SR latches in series with the clock on the second latch inverted



- The input is observed by the first latch with C = 1
- The output is changed by the second latch with C = 0
- The path from input to output is broken by the difference in clocking values (C = 1 and C = 0)

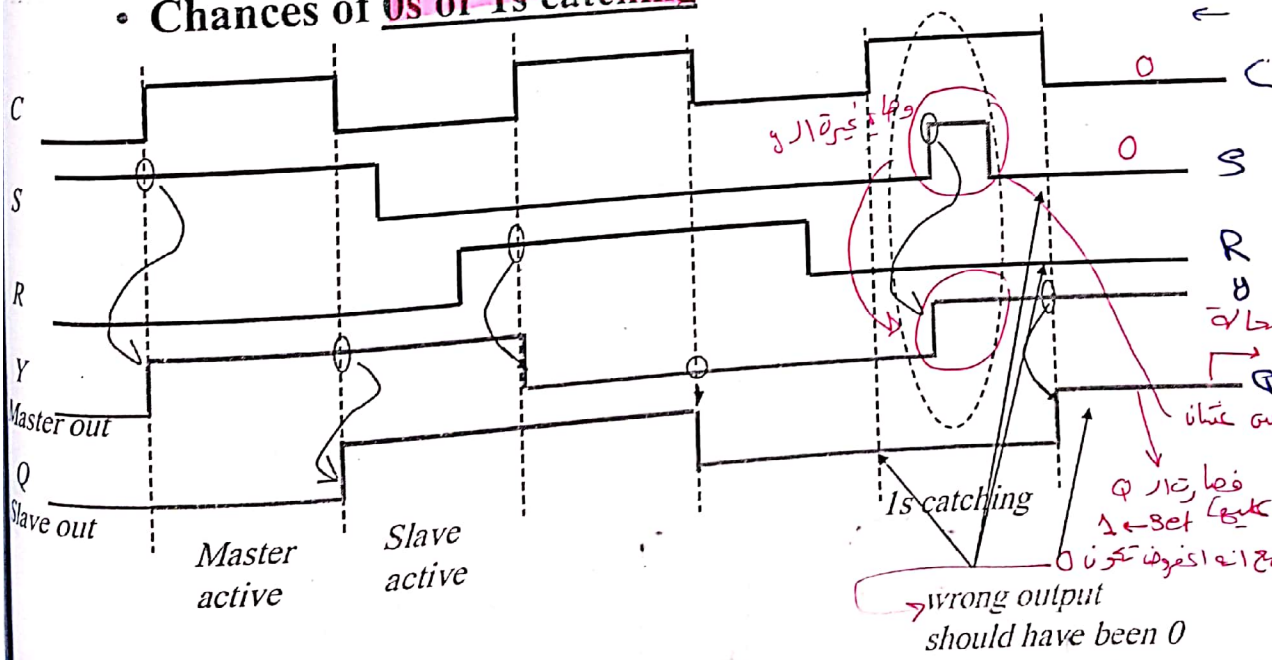
Timing diagram for SR Master-Slave Flip-Flop



Chapter 5 - Part 1 17

Master-Slave Flip-Flop Problem

- S and/or R are permitted to change while C = 1
  - Chances of 0s or 1s catching



Chapter 5 - Part 1 18

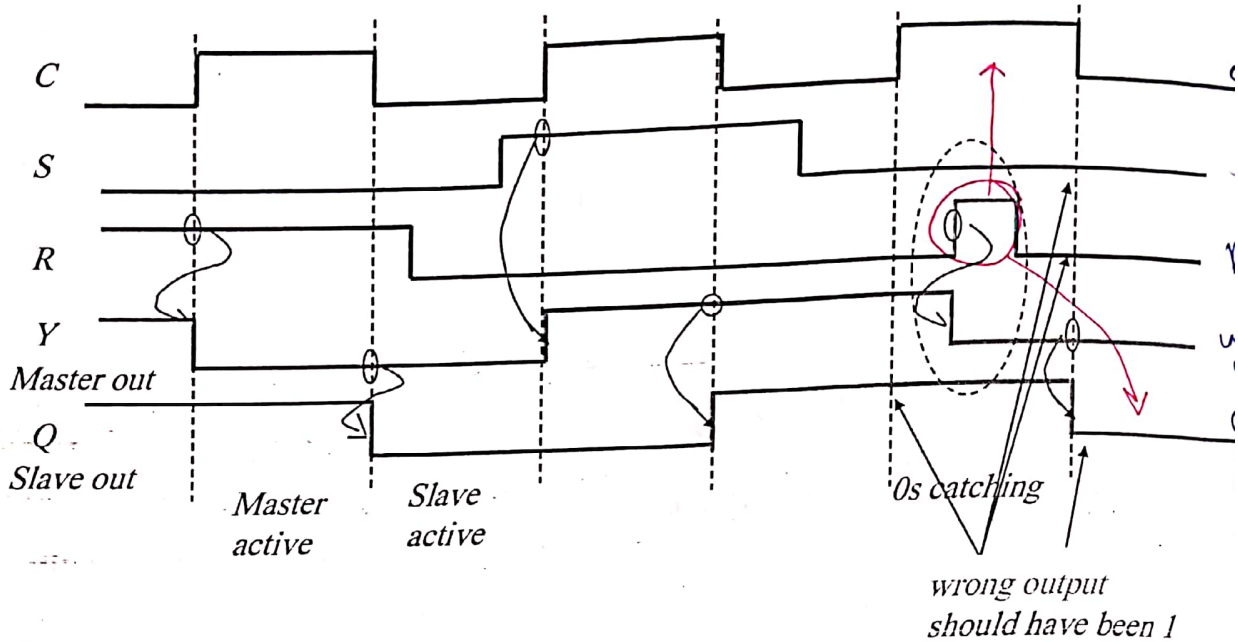
+ve pulse R تتغير بال ← بتصير لما (0's catching)

0s Catching

* بالامتحان لما تيجيني ال wave form بتطلع دايماً

على ال (negative edge) وبشوف قيمة S و R

وبعمل الرسمة أو بحكيك إنه صار فيها catching ← 1's OR 0's



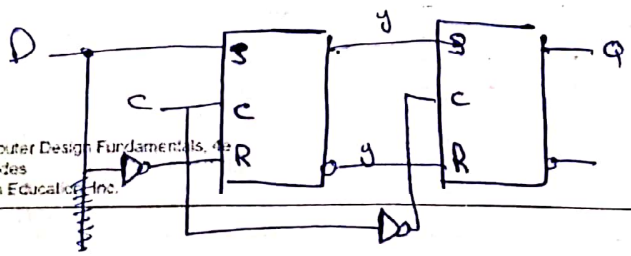
يخزن 1-bit (0 أو 1)

at discreat time وقت محدد

Flip-Flop Solution

ما بتغير الـ output إلا على ال edge triggered ما بتتغير خلال ال pulse

- Use ***edge-triggering*** instead of master-slave
- An ***edge-triggered*** flip-flop ignores the pulse while it is at a constant level and triggers only during a ***transition*** of the clock signal
- Edge-triggered flip-flops can be built directly at the electronic circuit level, or
- A ***master-slave D flip-flop*** which also exhibits ***edge-triggered behavior*** can be used



نفس هدا ال design

لأنه القيم تاعت Q بس بتتغير عند الـ edges

مو ضروري احسب y

Edge-Triggered D Flip-Flop

Negative

بقدر أعرف قيمة الـ Q من قيمة الـ D

+ve pulse

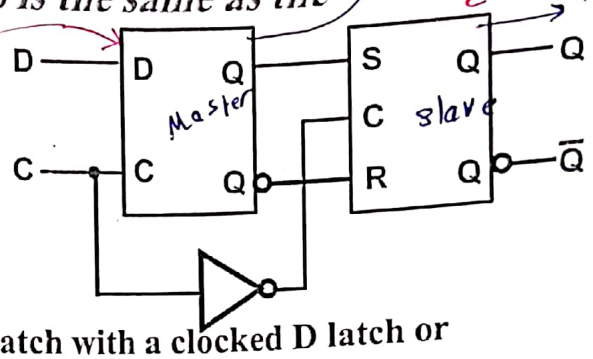
يشتغل على

SR-latch

- ve pulse

- *The edge-triggered D flip-flop is the same as the master-slave D flip-flop*

D latch



- It can be formed by:
  - **Replacing the first clocked SR latch with a clocked D latch or**
  - **Adding a D input and inverter to a master-slave SR flip-flop**
- The 1s and 0s catching behaviors are not present with D replacing S and R inputs
- The change of the D flip-flop output is associated with the negative edge at the end of the pulse
- ***It is called a negative-edge triggered flip-flop***

$D = 0 \rightarrow Q = 0$

$D = 1 \rightarrow Q = 1$

(negative edge)

As one black box

× input D

output Q

No 1s catching in the edge-triggered D Flip-Flops

* بالامتحان الدكتور بيعطينا الـ input (D, C) وبنطلعه الـ output اللي هو Q

* الـ D-FF ما في داعي نحسب الـ y

* الـ Mas بتشتغل لما C +ve

* الـ Slave بتشتغل لما C -ve

* الـ input للـ Slave هو y.

* الـ initial value كمان الدكتور بيحددها.

* الـ Master بتعتمد على الـ D

* بتطلع على الـ D لما يكون C +ve

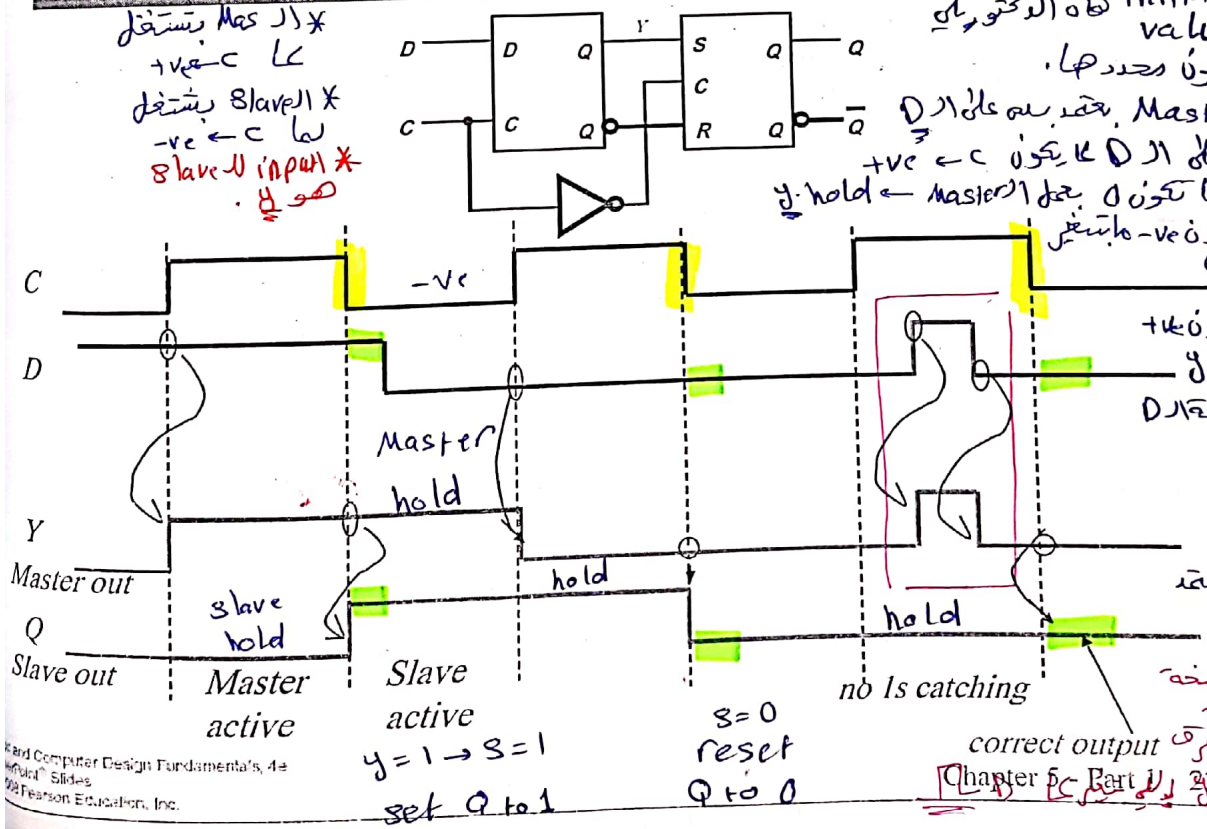
الـ D لما تكون 0 بعمل الـ Master hold ← y

* لما C تكون -ve ما بتتغير قيمة الـ y

* لما C تكون +ve على طول الـ y بتلحق قيمة الـ D

عا بصيرها 1's أو 0's catching

* الـ y بتعتمد على الـ D



$y = 1 \rightarrow S = 1$ set Q to 1

$S = 0$ reset Q to 0

* الـ Q نسخة متأخرة من الـ y بس الفرق أنه بس الـ y بيتغير لما

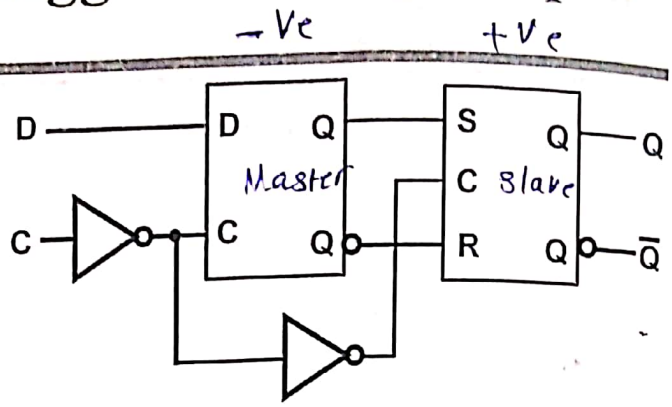
raising edge triggered

* لازم أحط 2-inverter عشان أخلي ال delay

Positive-Edge Triggered D Flip-Flop

- Formed by adding inverter to clock input

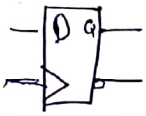
لازم ال C توصل لل 2 بنفس الوقت
يعني يصير التغير عليهم



- Q changes to the value on D applied at the positive clock edge

- Our choice as the **standard flip-flop** for most sequential circuits

-single flip-flop
[edge triggered

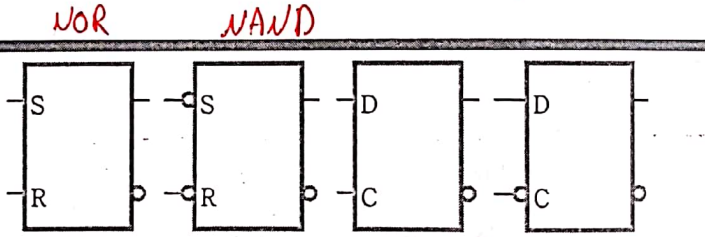


Standard Symbols for Storage Elements

✓ → ✓
* → *
بعادلوا بعض

- **Latches:** كل ما يتغير ال input بتتغير القيمة المخزنة داخلهم

SR - Flip - flop

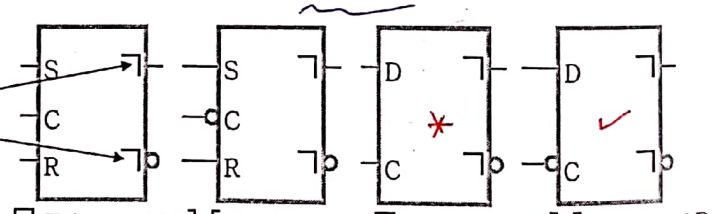


(a) Latches

- **Master-Slave:** Postponed output indicators

يعني 2-latches

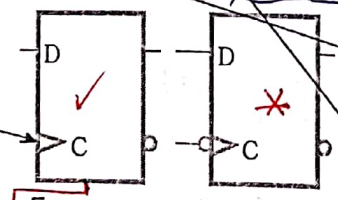
بيجي عند ال Master



(b) Master-Slave Flip-Flops

- **Edge-Triggered:** Dynamic indicator

* أكثر شي بيجي الامتحان مهم



(c) Edge-Triggered Flip-Flops

Master active when C = 1
Slave active when C = 0

Master active when C = 0
Slave active when C = 1

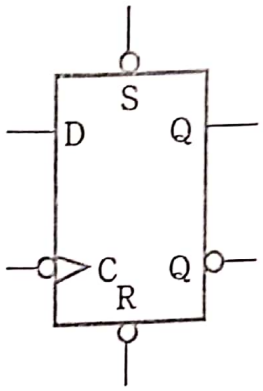
most common

+ve edge triggered D-FF

S, R* ← asynchronously ما إلهم علاقة بالـ clk
فحتا نه هيك سميناهم Direct
* بس وحدة منطقع (S,R) مصوصلها تكون
شغالة ataline
* (S,R) أقوى من الـ clk signals ←

Direct Inputs

- At power up or at reset, all or part of a sequential circuit usually is initialized to a known state before it begins operation
- This initialization is often done outside of the clocked behavior of the circuit, i.e., asynchronously

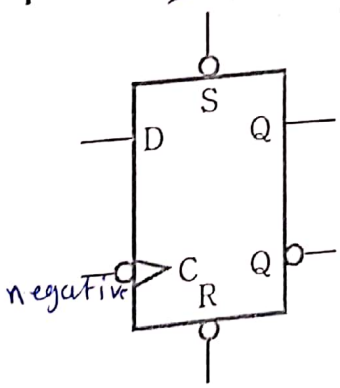


- Direct R and/or S inputs that control the state of the latches within the flip-flops are used for this initialization
- For the example flip-flop shown
  - 0 applied to R resets the flip-flop to the 0 state
  - 0 applied to S sets the flip-flop to the 1 state

مهم *

Direct inputs

- D flip-flop with active-low direct inputs :



Direct inputs

S	R	C	D	Q	Q'
0	1	x	x	1	0
1	0	x	x	0	1
1	1	↓	0	0	1
1	1	↓	1	1	0

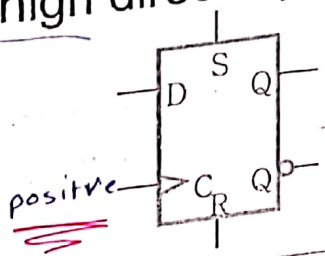
Q̄

high

low

-ve edge

- Active high direct inputs:



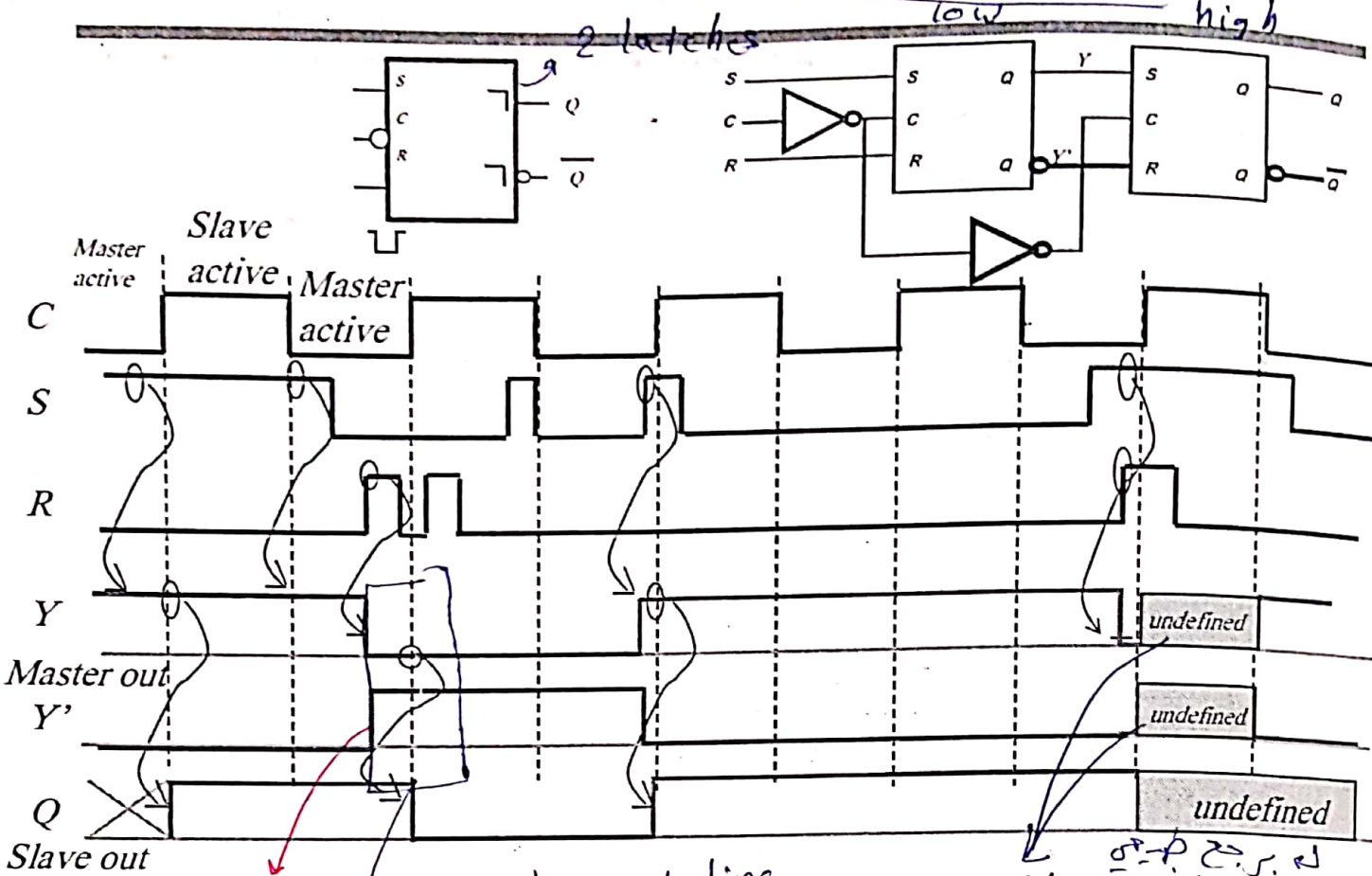
S	R	C	D	Q	Q'
0	1	x	x	0	1
1	0	x	x	1	0
0	0	↑	0	0	1
0	0	↑	1	1	0

S, R active highly

+ve

Master slave +ve edge tri

Timing diagram of A SR Master-Slave Flip-Flo



لا يرجع لحالته لين ما يجي Reset أو Set

27

هون المفروض كان يحل hold لأنه 0 ، 0 يعني ما تغير قيمة ال y يعني كان Q تخليها ال 0 فصار 0's catching

* إذا سألنا هل 0's أو 1's catching بتنعرف على S,R و Q
← وهاي الحالة بتطلع على (+ve edge)

2 - FFs → 4 states

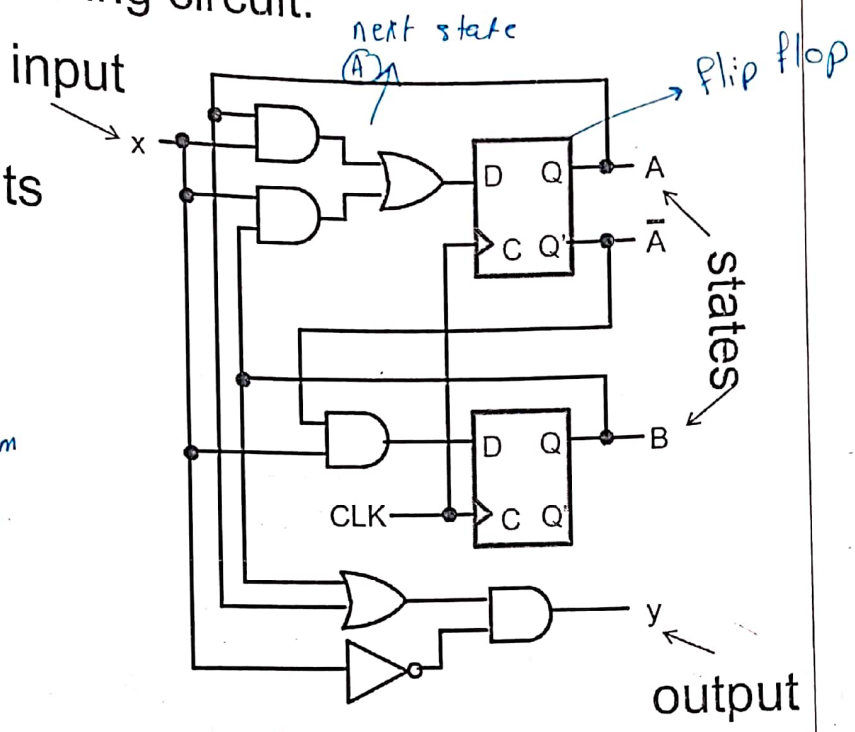
5-4 Sequential Circuit Analysis

- Consider the following circuit:

2-bits flip flop

1-bit ⇐ flip flop

- What does it do?
- How do the outputs change when an input arrives?



* maximum num of states $= 2^m$

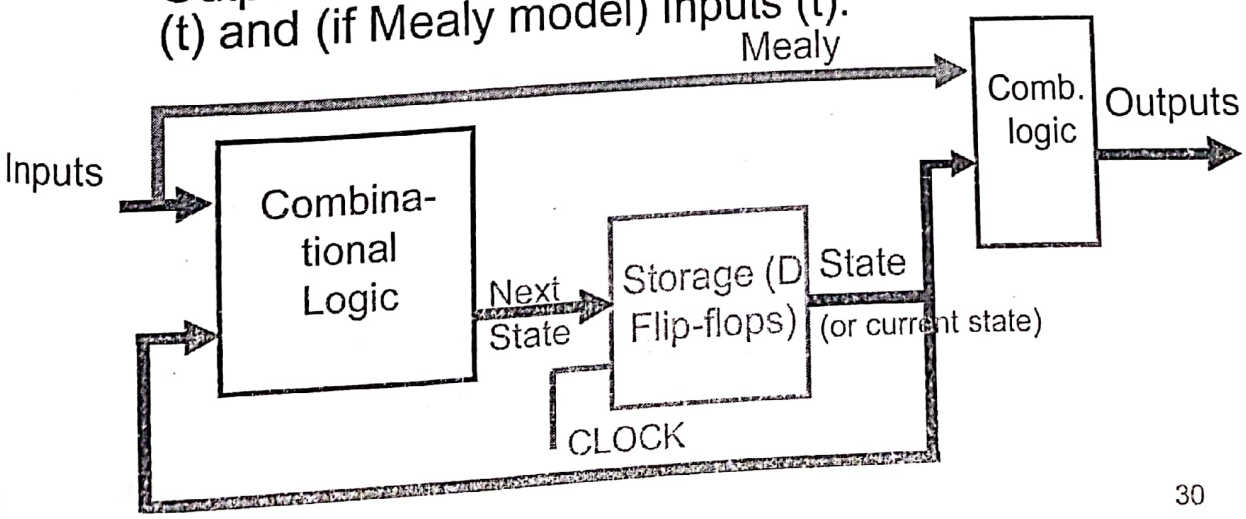
m = of FF's

→ 5-4

→ 5-5

Sequential Circuit Model

- General Model
  - Current or Present State at time (t) is stored in an array of flip-flops.
  - Next State is a Boolean function of State and Inputs.
  - Outputs at time (t) are a Boolean function of State (t) and (if Mealy model) Inputs (t).



present state A , B
current " A(t), B(t)
state

Next State D_A D_B } equations Flip Flop ds *
A(t+1), B(t+1)
$D_A(t)$, $D_B(t)$

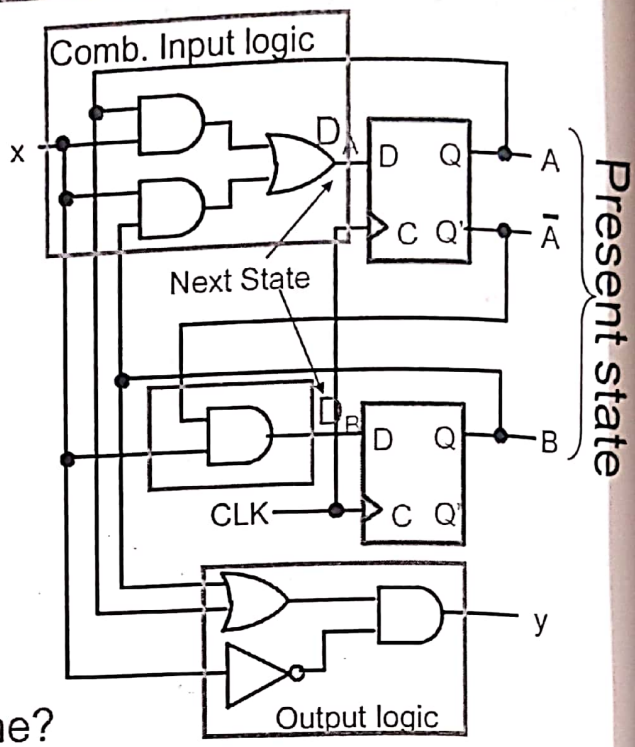
Previous Example (from Fig. 5-15)

- **Input:** X only one
- **Output:** Y
- **State:** (A(t), B(t))

  Example: (AB)= (01), (10)
- **Next State:**

  $(D_A(t), D_B(t))$

  = (A(t+1), B(t+1))



Is this a Moore or Mealy machine?

لـ ما بيعتمد على الـ input ↳ يعتمد على الـ input

Steps for Analyzing a Sequential Circuit

1. Find the input equations (D_A, D_B) to the flip-flops (next state equations) and the output equation.
2. Derive the State Table (describes the behavior of a sequential circuit).
3. Draw the State Diagram (graphical description of the behavior of the sequential circuit).
4. Simulation

لأنه لازم أكتبها بأبسط صورة

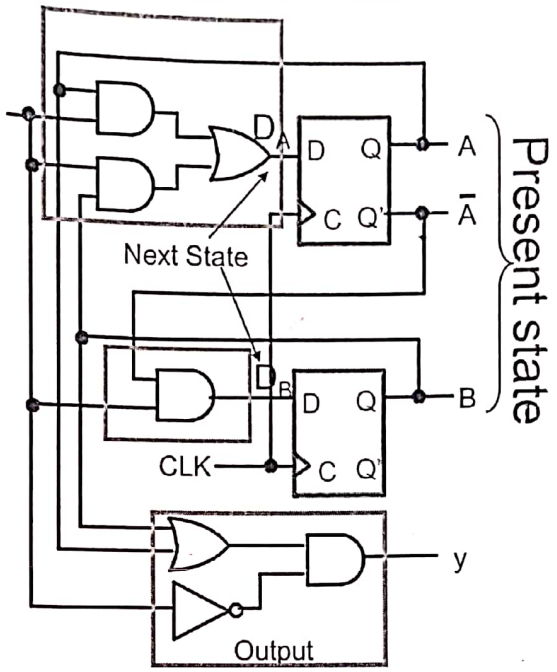
$y = Ax + A$
$= A(x+1)$ ← لا تستخدمها حتى أفكر بها mealy

Moore ← $y = A$

Step 1: Input and output equations

- Boolean equations for the inputs to the flip flops:
  - $D_A = AX + BX$ = $D_{(A)}(t) = A(t+1)$ X
  - $D_B = \overline{A}X$
- Output Y output الـ ← حتى الـ input الـ إذا
  - $Y = \overline{X}(A + B)$ mealy ∴

  بأبسط صورة لازم تكون
- Also can be written as
  - $A(t+1) = D_A = A(t)X + B(t)X$
  - $B(t+1) = D_B = \overline{A(t)}X$
  - $Y = \overline{X}(A(t) + B(t))$



* لو كان عندي 2 outputs لازم أتأكد أنه كل الحالات ما بتعتمد على الـ outputs

Step 2: State Table

- The state table: shows what the *next state* and the *output* will be as a function of the present state and the input:

Inputs of the combinational circuit — Outputs of the table

Present State	*Input*	Next State	Output

- The State Table can be considered a truth table defining the combinational circuits:
  - the inputs are *Present State* and *Input*,
  - and the outputs are *Next State* and *Output*

State Table For The Example

- For the example: $A(t+1) = A(t)\,x + B(t)\,x$

  $B(t+1) = A'(t)\,x$

  $Y(t) = X'\,(B(t) + A(t))$

1-Dimension

	Present State	Input	Next State	Output
	A(t) B(t)	X	A(t+1) B(t+1)	Y
0	0 0	0	0 0	0
1	0 0	1	0 1	0
2	0 1	0	0 0	1
3	0 1	1	1 1	0
4	1 0	0	0 0	1
5	1 0	1	1 0	0
6	1 1	0	0 0	1
7	1 1	1	1 0	0

Inputs of the table — Outputs of the table

2^3 rows (2^{m+n}) rows

m: no. of flip-flops
n: no. of inputs

Alternate State Table

- The previous (1-dimensional table) can become quite lengthy with 2^{m+n} rows (m=no. of flip-flops; n=no. of inputs)
- Alternatively, a 2-dimensional table has the present state in the left column and inputs across the top row
  - $A(t+1) = A(t)\,X + B(t)\,X$
  - $B(t+1) = A'(t)\,X$
  - $Y = X'\,(B(t) + A(t))$

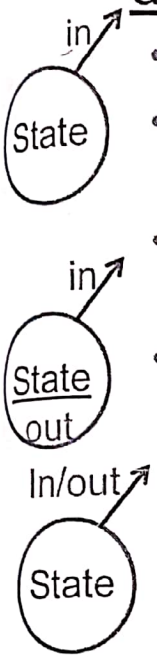
2-D

Present State A(t) B(t)	Next State X = 0 A(t+1) B(t+1)	Next State X = 1 A(t+1) B(t+1)	Output X=0 Y	Output X=1 Y
0 0	0 0	0 1	0	0
0 1	0 0	1 1	1	0
1 0	0 0	1 0	1	0
1 1	0 0	1 0	1	0

2^m

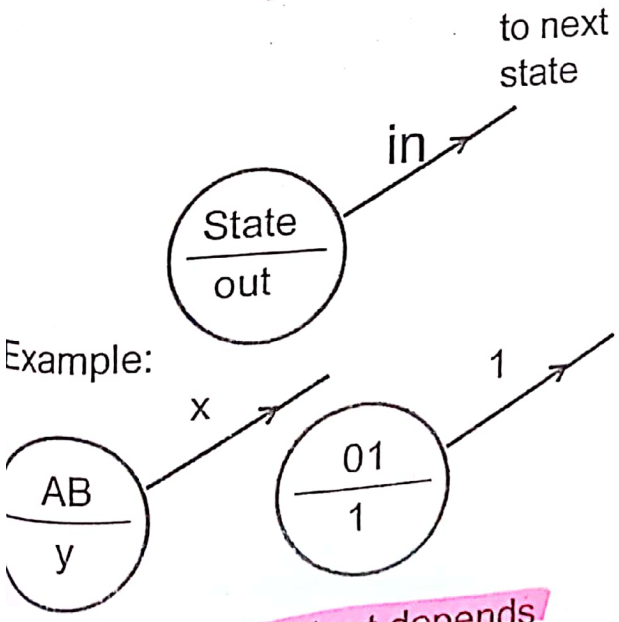
Step 3: State Diagrams

- The sequential circuit function can be represented in graphical form as a state diagram with the following components:
  - A circle with the state name in it for each state
  - A directed arc from the Present State to the Next State for each state transition
  - A label on each directed arc with the Input values which causes the state transition, and
  - A label:
    - In each circle with the output value produced, or
    - On each directed arc with the output value produced.



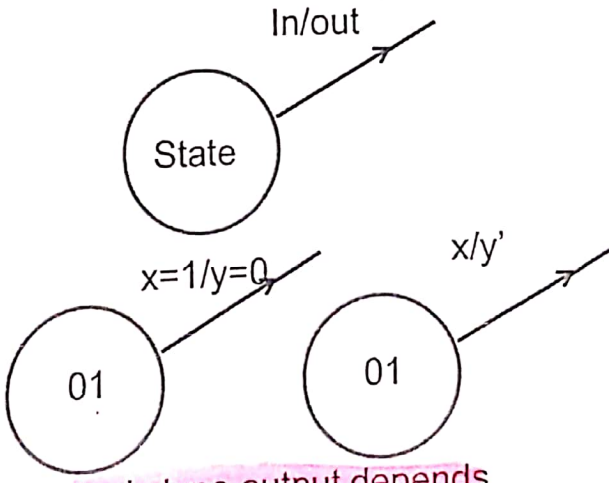
State Diagram Convention

Moore Machine:



Moore type output depends only on state

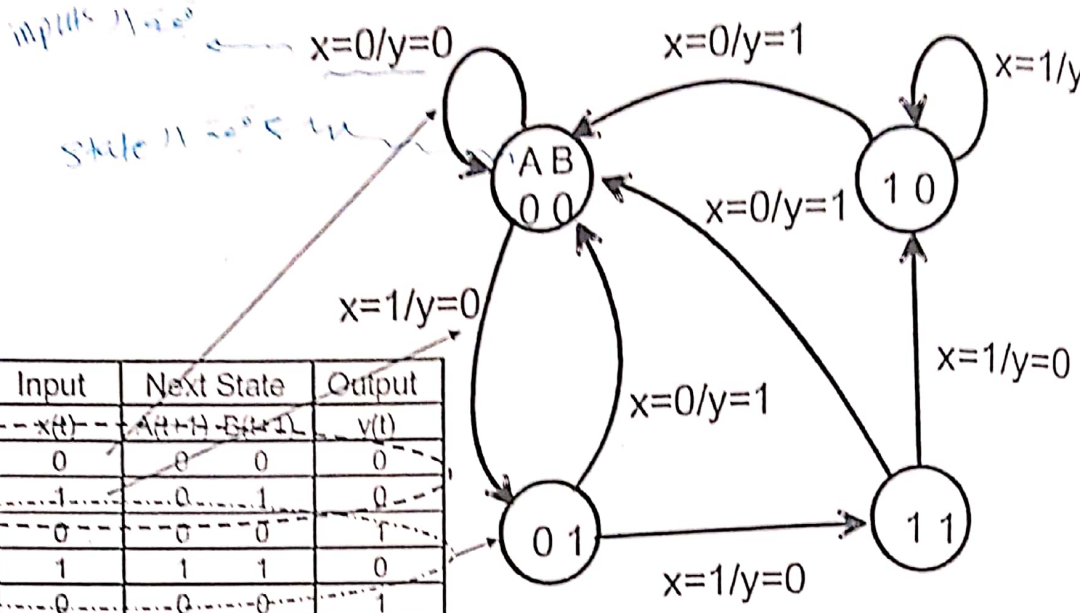
Mealy Machine:



Mealy type output depends on state and input

State Diagram For The Example

- Graphical representation of the state table:



Present State		Input	Next State		Output
A(t)	B(t)	x(t)	A(t+1)	B(t+1)	y(t)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Step 4: Simulation

- Two types:
  - Functional simulation: objective is to verify the functionality of the circuit
  - Timing simulation: objective is to perform a more realistic testing (with gate delays counted)
- More about this step in the lab (CPE0907234)

Example2

m ← 1 - FF $2^{m+n} = 2^3 = 8$ states

n ← 2-inputs

Moore

- Derive the state table and state diagram for the sequential circuit.

Inputs :-
x, y

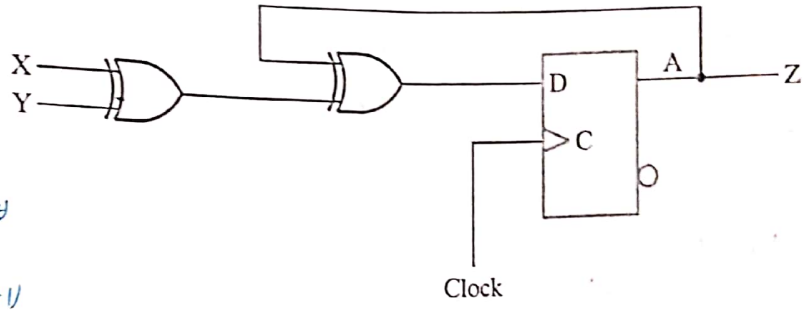
Out puts :-
z

state :- $A \equiv A(t)$

Next state :-
$D(A), D_A(t), A(t+1)$

* Inputs Equations
$D_A = D_A(t) = A(t+1) = X \oplus Y \oplus A$

* out-puts Eqs
$Z = A$ Moore



Example2 Cont.

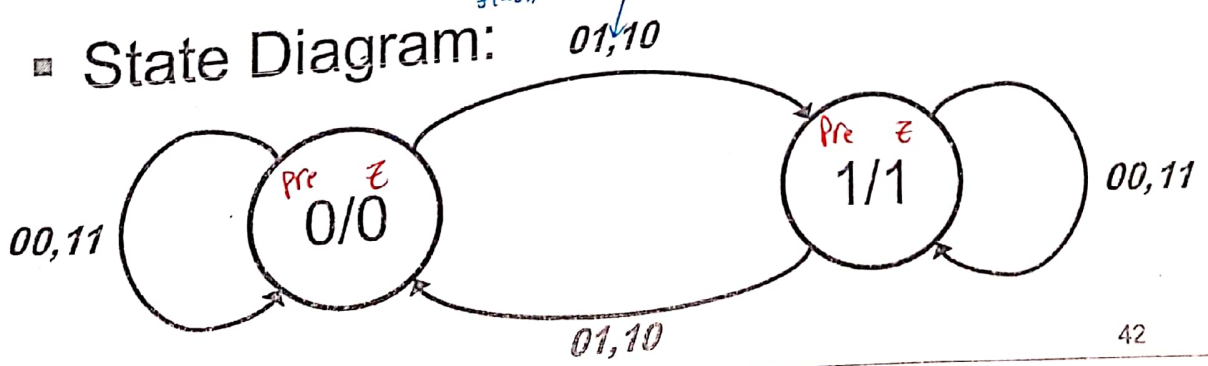
- State Table:

pre / Z
ما يهمنى الحالات إلا بالنص

Preset State A(t)	Next State XY = 00 A(t+1)	XY = 01 A(t+1)	XY = 10 A(t+1)	XY = 11 A(t+1)	Z
0	0	1	1	0	0
1	1	0	0	1	1

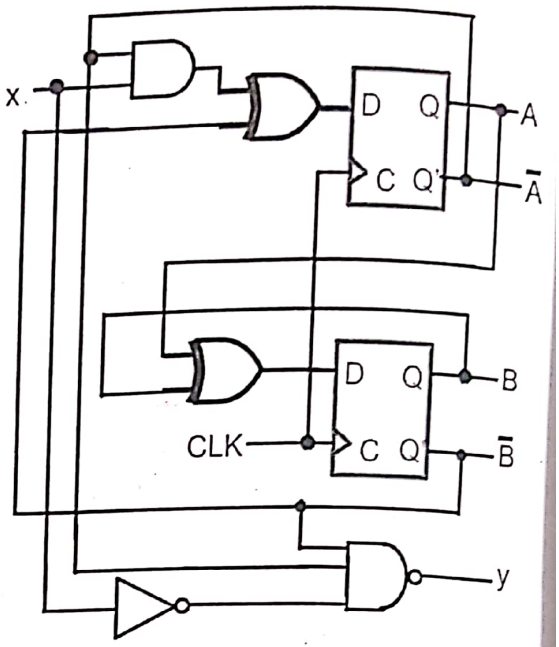
back slash سلاش

- State Diagram:



Example3

- Derive the state table and state diagram for the sequential circuit:



* States :- A, B
* Next states :- D_A, D_B
  $A(t+1)$, $B(t+1)$
* Inputs :- x
* Outputs :- y

Input equation :- Next state

$D_A = (\bar{A} \cdot x) \oplus \bar{B}$

$D_B = A \oplus B$

Output equation :-

$y = \overline{\bar{B} \cdot \bar{A} \cdot \bar{x}}$

$y = B + A + x$ → "Mealy"

Example3 Cont.

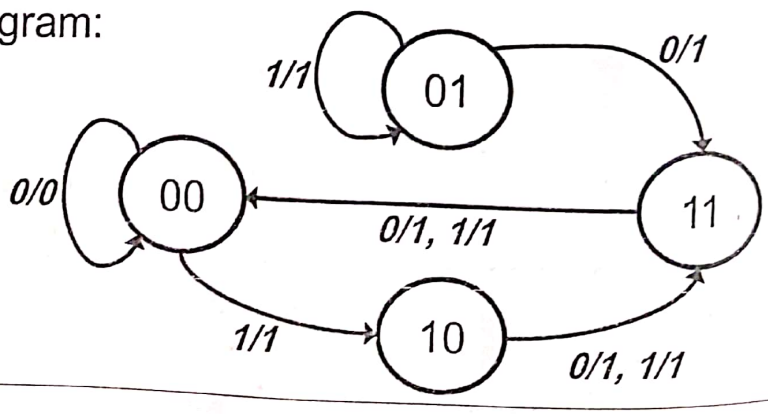
- State Table:

A B x
0 0 1

$A(t+1) = (\bar{A} \cdot x) \oplus \bar{B}$
$= 1 \oplus 1 = 0$

Preset State A(t) B(t)	Next State X = 0 A(t+1) B(t+1)	Next State X = 1 A(t+1) B(t+1)	Output X = 0 Y	Output X = 1 Y
0 0	0 0	1 0	0	1
0 1	1 1	0 1	1	1
1 0	1 1	1 1	1	1
1 1	0 0	0 0	1	1

- State Diagram:

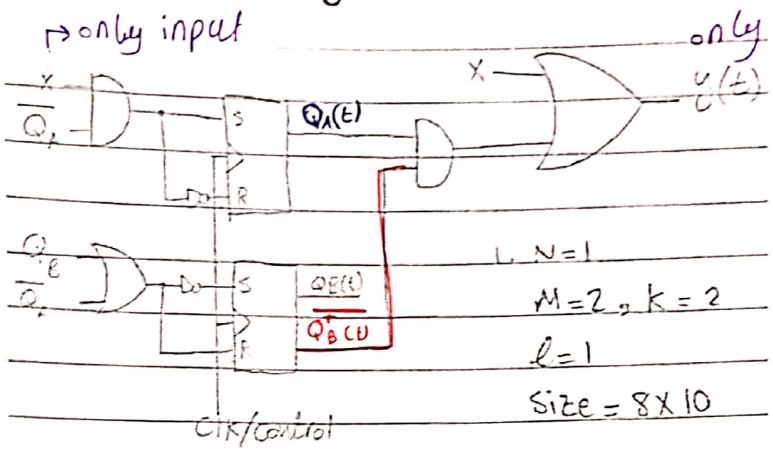


Example4

SR Flip flop ← postive

- Derive the state table and state diagram for the sequential circuit:

* Present State:-
  $Q_A(t) \equiv Q_A$
  $Q_B(t) = Q_B$
* Next States
  $Q_A(t+1) \equiv Q_A^+$
  $Q_B(t+1) \equiv Q_B^+$
* Inputs: X
* Outputs y(t)



Input Eq:-

$S_{QA} = X \cdot \overline{QA}$

$R_{QA} = \overline{X + QA}$

$S_{QB} = \overline{QB + \overline{QA}} = \overline{QB} \cdot QA$

$R_{QB} = QB + \overline{QA}$

Output Eq:-

$y(t) = X + (QA \cdot \overline{QB})$

Example4 Cont.

S_A عكس R_A لازم ننتبه عشان سرعة الحل

0,0 ← بتفرج على ال equations اللي كتبناها. وبنثبتها ال current state

flip-flop بتترك فراغ على ما يكون

* وظيفتهم انه نعرف ال next state خلالهم

- State Table

أحيانا بناعدونا بالرسمة

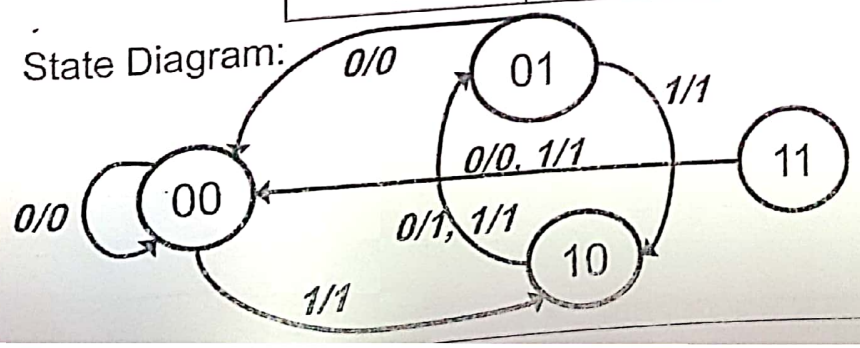
ما يتطلع عليهم أبدا خلال الرسمة

Present State Q_A Q_B	Input X	S_A R_A	S_B R_B	Next State $Q_A(t+1)$ $Q_B(t+1)$ (S_A S_B)	Output Y
0 0	0	0 1	0 1	0 0	0
0 0	1	1 0	0 1	1 0	1
0 1	0	0 1	0 1	0 0	0
0 1	1	1 0	0 1	1 0	1
1 0	0	0 1	1 0	0 1	1
1 0	1	0 1	1 0	0 1	1
1 1	0	0 1	0 1	0 0	0
1 1	1	0 1	0 1	0 0	1

هي القيمة المخزنة في a

$S_A = Q_A^+$

- State Diagram:



Exercise: Derive the state diagram of the following Circuit

هاي الرسمة ما فيها input

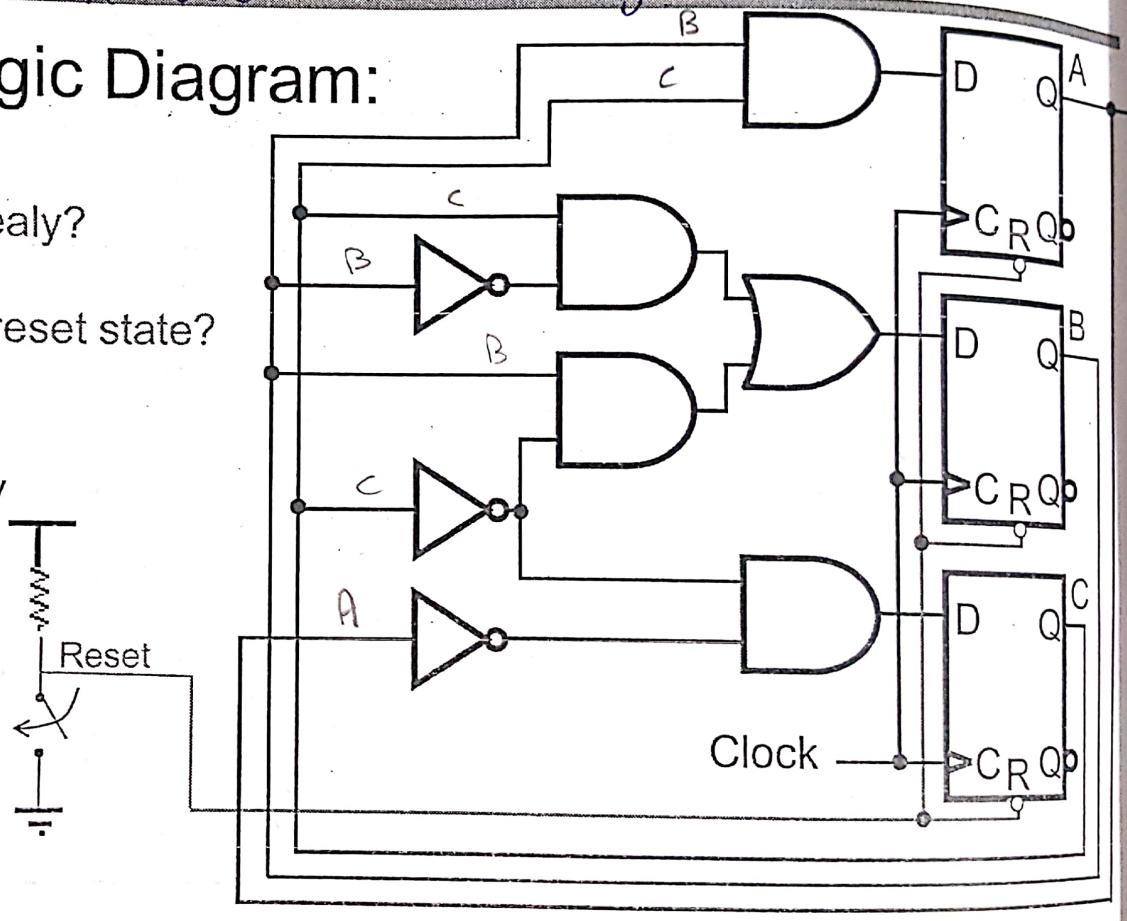
initial state 000 moore Design.

- Logic Diagram:

Moore or Mealy?

What is the reset state?

input ← لا يعتبر



Step1: Flip-Flop Input Equations

- Variables
  - Inputs: None
  - Outputs: Z
  - State Variables: A, B, C
- Initialization: Reset to (0,0,0)
- Equations
  - $A(t+1) = BC$ $\qquad Z = A$
  - $B(t+1) = B'C + BC' = B \oplus C$
  - $C(t+1) = A'C'$

Step 2: State Table

تغيير الـ state بيتم على الـ positive edge

A B C	$A^+ B^+ C^+$	Z
0 0 0	0 0 1	0
0 0 1	0 1 0	0
0 1 0	0 1 1	0
0 1 1	1 0 0	0
1 0 0	0 0 0	1
1 0 1	0 1 0	1
1 1 0	0 1 0	1
1 1 1	1 0 0	1

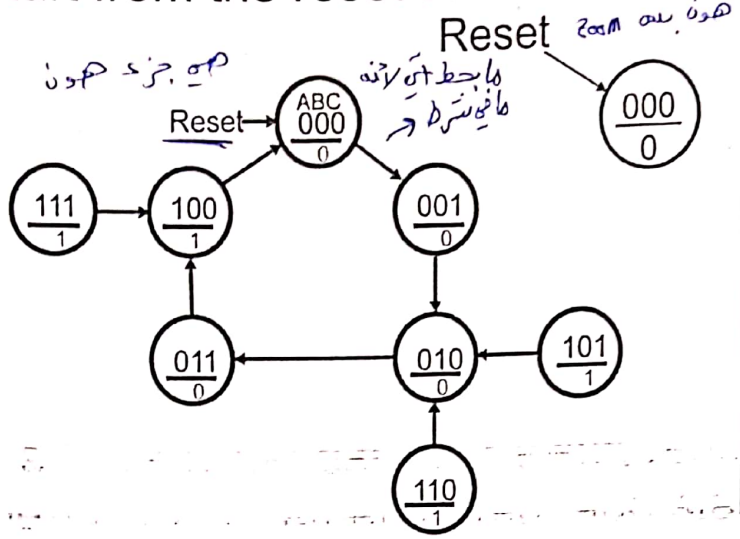
$A(t+1) = BC$ $\qquad Z = A$

$B(t+1) = B'C + BC' = B \oplus C$

$C(t+1) = A'C'$

Step 3: State Diagram

Start from the reset state



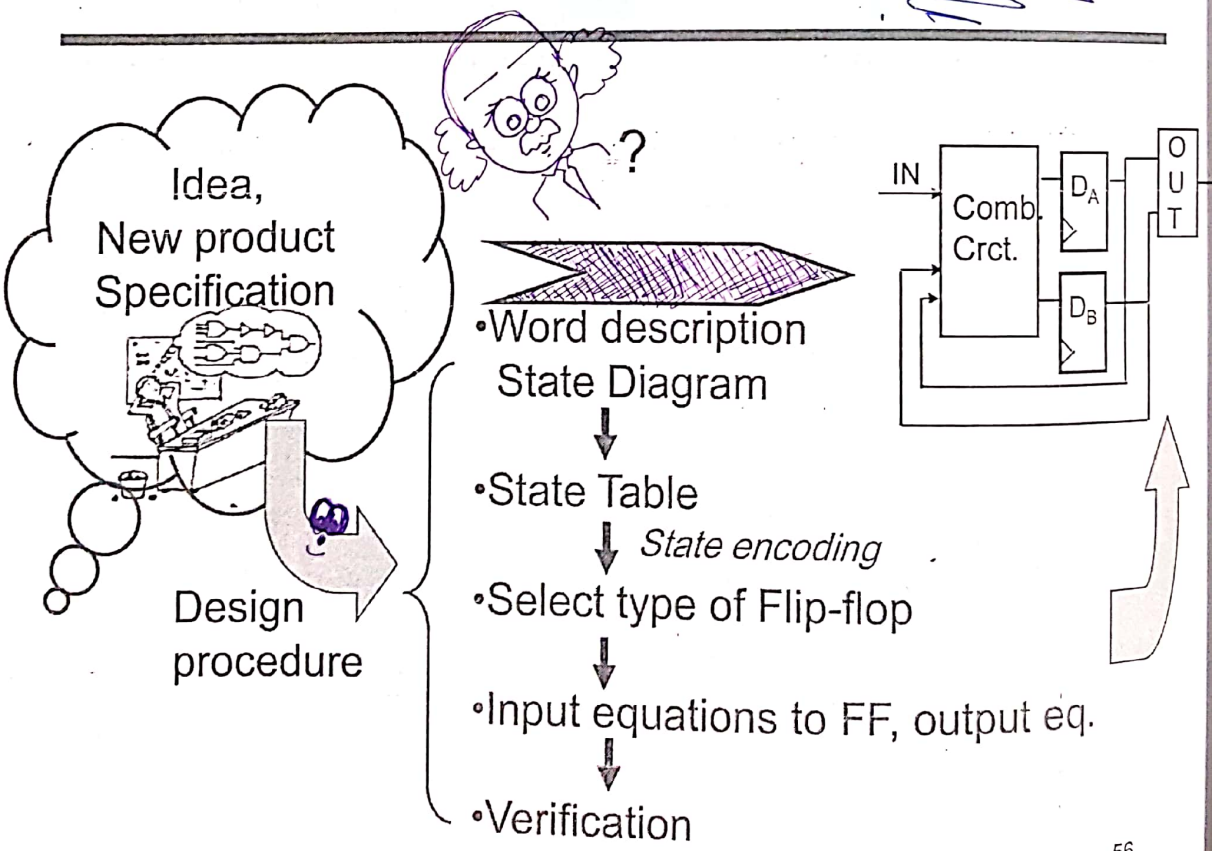
A B C	$A^+B^+C^+$	Z
0 0 0	0 0 1	0
0 0 1	0 1 0	0
0 1 0	0 1 1	0
0 1 1	1 0 0	0
1 0 0	0 0 0	1
1 0 1	0 1 0	1
1 1 0	0 1 0	1
1 1 1	1 0 0	1

- Are all states used? Which ones? NO

unused stated 5, 6, 7

synchronouns

5-5 Sequential Circuit Design



Specification

- Component Forms of Specification
  - Written description
  - Mathematical description
  - Hardware description language
  - Tabular description
  - Equation description
  - Diagram describing operation (not just structure)

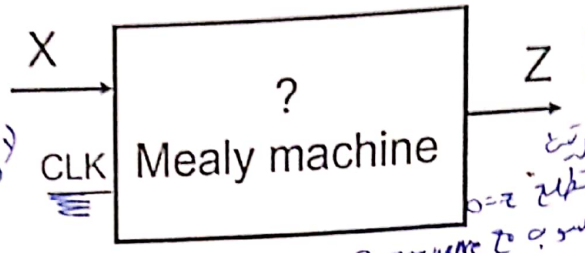
Formulation: Finding a State Diagram

- In specifying a circuit, we use states to remember meaningful properties of past input sequences that are essential to predicting future output values.
- As an example, a sequence recognizer is a sequential circuit that produces a distinct output value whenever a prescribed pattern of input symbols occur in sequence, i.e, recognizes an input sequence occurrence.
- Next, the state diagram, will be converted to a state table from which the circuit will be designed.

Sequence Detector Example: 1101

X 1101101
Z 0001001

لأنه مسموح الـ over lap



لازم أحطها

فطلعت الـ Z = 1 مرتين
أما إذا ممنوع بتطلع Z=0
لأنه في elem محسوبة 2-sequence

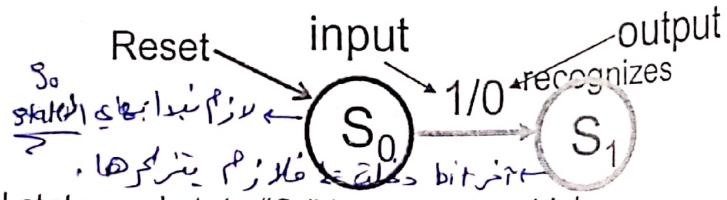
Input X:	0011100110101101101001111 0111
Output Z:	0000000000100001001000000100

Overlapping sequences are allowed

ممكن يكون الـ over lapp
بـ 2-bits

Step2: Finding A State Diagram

- Define states for the sequence to be recognized:
  - assuming it starts with first symbol X=1,
  - continues through the right sequence to be recognized, and
  - uses output 1* to mean the full sequence has occurred,
  - with output 0 otherwise.
- Starting in the initial state (named "S_0"):
  - Add a state that the first "1."



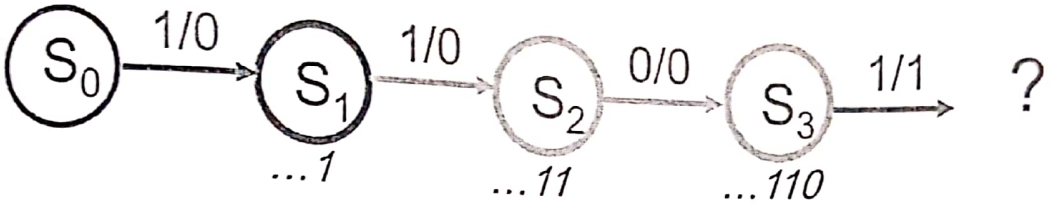
S_0
لازم نبدأ بهاي الحالة
آخر bit وصلت له فلازم نتركها.

  - State "S_0" is the initial state, and state "S_1" is the state which represents the fact that the "first" one in the input subsequence has occurred. The first "1" occurred while being in state S_0 during the clock edge.

Finding a State Diagram(cont.)

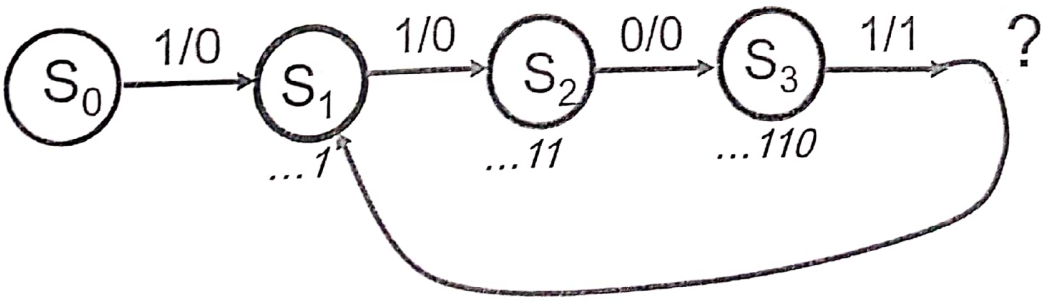
1 1 1
فضلي نفس الحالة

- Assume that the 2nd 1 arrives of the sequence 1101: needs to be remembered: add a state S_2



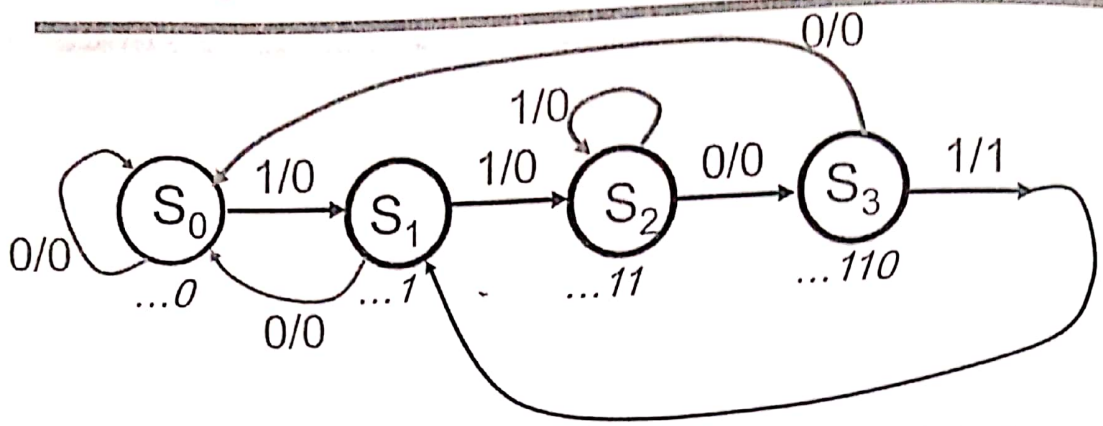
- Next, a "0" arrives: part of the sequence 1101 that needs to be remembered; add state S_3
- The next input is "1" which is part of the right sequence 1101; now output Z=1

Completing The State Diagram



- Where does the final arrow go to:
  - The final 1 of the sequence 1101 can be the beginning of another sequence; thus the arrow should go to state S_1

Completing The State Diagram

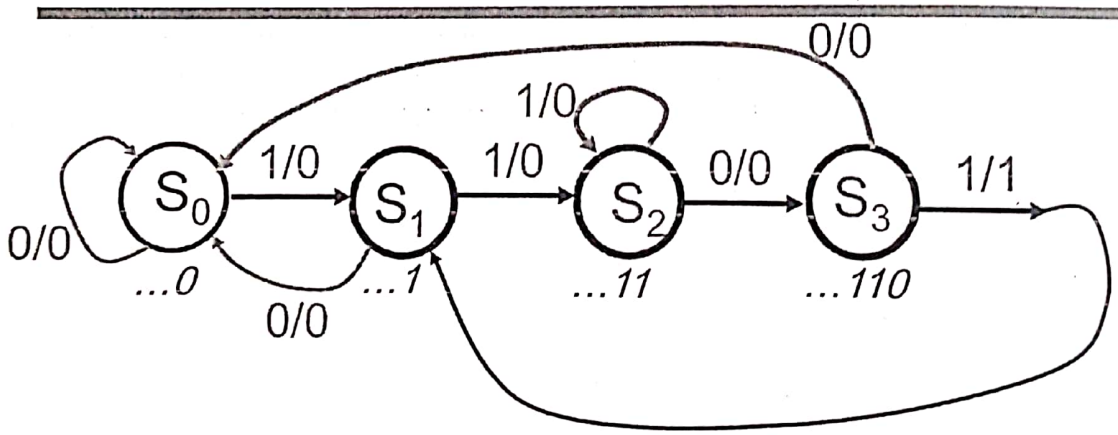


- Start is state S_0: assume an input X=0 arrives; what is the next state?
- Next, consider state S_1: input X=0; next state?
- Next state S_2 and S_3: completes the diagram
- Each state should have two arrows leaving

number of FF's $\Rightarrow 2^m = 5$ $\boxed{m=2}$

Max. number of states

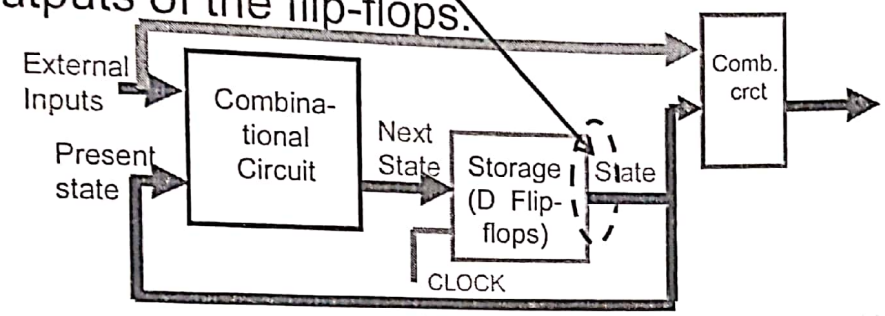
Deriving State Table



Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_3	S_2	0	0
S_3	S_0	S_1	0	1

Step 3: State Assignment

- Right now States have names such as S_0, S_1, S_2 and S_3
- In actuality these state need to be represented by the outputs of the flip-flops.



- We need to assign each state to a certain output combination AB of the flip-flops:
  - e.g. State S_0=00, S_1=01, S_2=10, S_3=11
  - Other combinations are possible: S_0=00, S_1=10, S_2=11, S_3=01

Popular State Assignments

- 1. Counting order assignment:
  - 00, 01, 10, 11
- 2. Gray code assignment:
  - 00, 01, 11, 10
- 3. One-hot state assignment
  - 0001, 0010, 0100, 1000
- Does state assignment make a difference in cost?

State Assignment: Counting order

"Counting Order" Assignment:

لازم أحدد
أنا شو عملت

$S_0 = 0\ 0$
$S_1 = 0\ 1$
$S_2 = 1\ 0$
$S_3 = 1\ 1$

State Table:

Present State	Next State x=0	Next State x=1	Output x=0	Output x=1
S_0	S_0	S_1	0	0
S_1	S_0	S_2	0	0
S_2	S_3	S_2	0	0
S_3	S_0	S_1	0	1

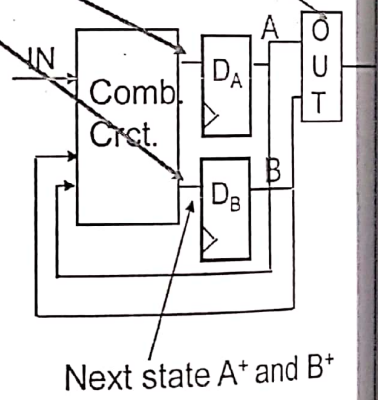
Resulting coded state table:

Present State A B	Next State x = 0 $A^+ B^+$	Next State x = 1 $A^+ B^+$	Output x = 0 Z	Output x = 1 Z
0 0	0 0	0 1	0	0
0 1	0 0	1 0	0	0
1 0	1 1	1 0	0	0
1 1	0 0	0 1	0	1

Step 4: Find Flip-Flop Input and Output Equations

* بطلبوا optimization عشان نعمل على الـ k-map

size of k-map → input, present state

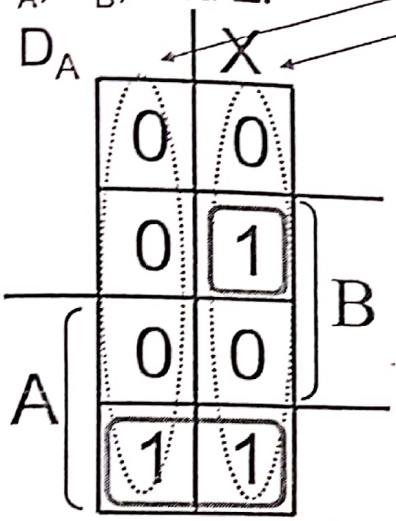


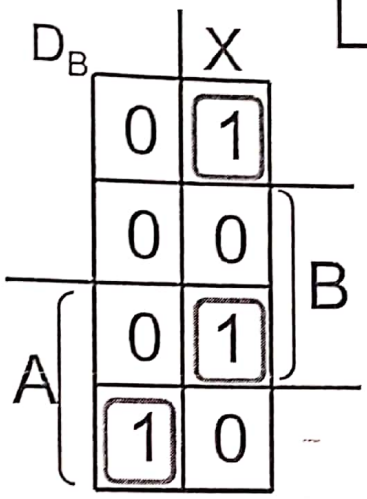
- State Diagram
- State Table
  - *State encoding*
- Select type of Flip-flop
- Input equations to FF, output eq.
- Verification

Find Flip-Flop Input and Output Equations: Example – Counting Order Assignment

- Using D flip-flops: thus $D_A=A^+$, $D_B=B^+$(the state table is the truth table for D_A and D_B).
- Interchange the bottom two rows of the state table, to obtain K-maps for D_A, D_B, and Z:

Present State	Next State x = 0	Next State x = 1	Output x = 0	Output x = 1
AB	A^+B^+	A^+B^+	Z	Z
0 0	0 0	0 1	0	0
0 1	0 0	1 0	0	0
1 0	1 1	1 0	0	0
1 1	0 0	0 1	0	1





$Z = XAB$

Gate Input Cost = 22

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

$D_B = \overline{X}\overline{A}\overline{B} + XAB + \overline{X}A\overline{B}$

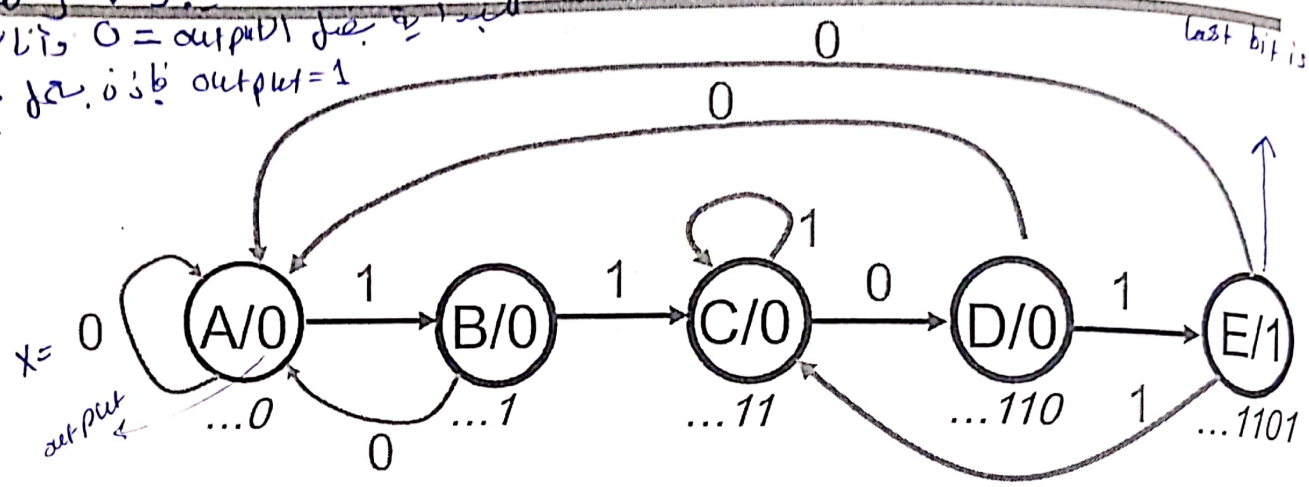
Step 5: Verification

- We will learn software tools for verifying the functionality of sequential circuits in the lab (CPE0907234)

بشكل عام بسبب state زيادة

Moore model for Sequence Recognizer "1101"

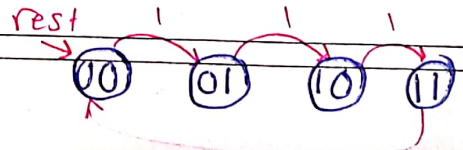
x ضفنا كمان state لأنه بوصل لحلقة ارجعت للبداية بفضل الـ output = 0 وأنا لازم يصير عندي state جديدة بحكم output = 1



- State Assignment:
  - Counting order (3 Flip-flops):
    - A = 000, B = 001, C = 010, D = 011, E = 100
  - Gray code (3 Flip-flops):
    - A = 000, B = 001, C = 011, D = 010, E = 110
  - One hot (5 Flip-flops):
    - A = 00001, B = 00010, C = 00100, D = 01000, E = 10000

Present State	Next State X = 0	Next State X = 1	Output
A	A	B	0
B	A	C	0
C	D	C	0
D	A	E	0
E	A	C	1

Free running counter



Exercise

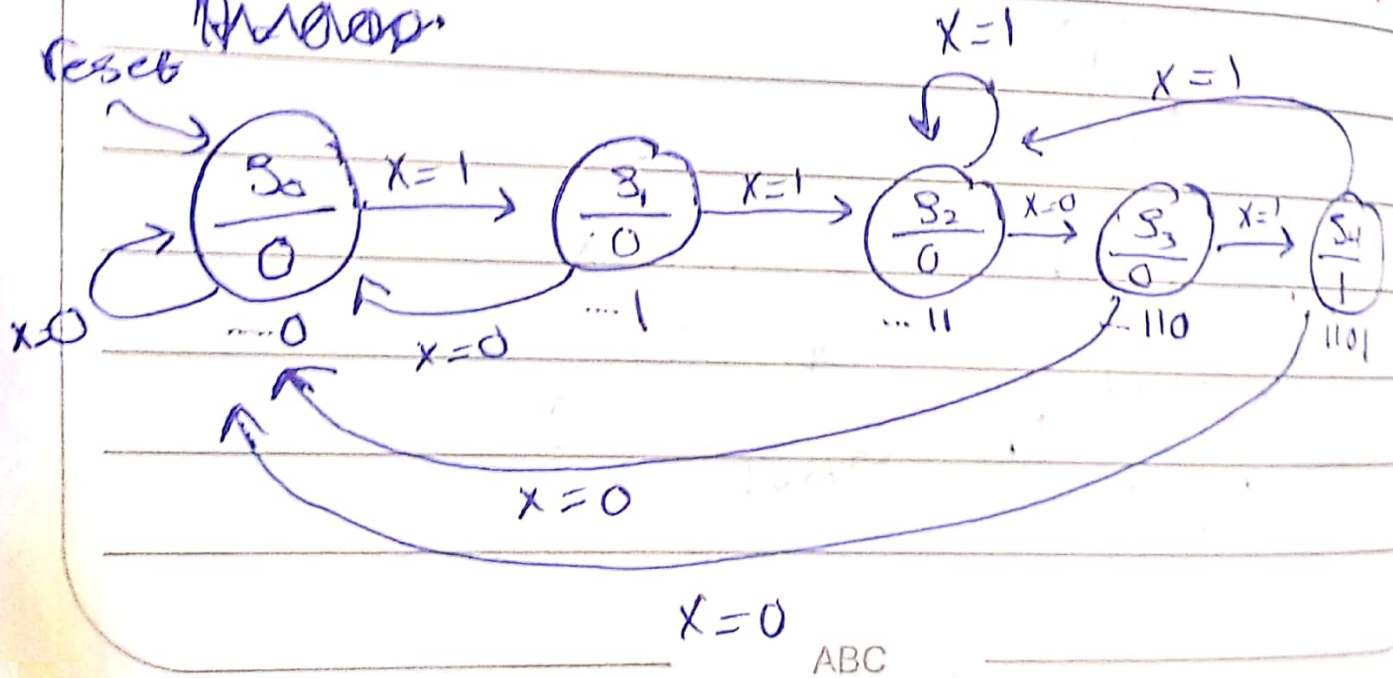
2-input

1101.

Counting order

Moore Model page:

State Diagr



Subject

Date: / /

الموضوع

التاريخ

$5 \times 2 = 10$ rows

2 State table

Pre Sta ($Q_A Q_B Q_C$)	Input (X)		Nex. St ($Q_A^+ Q_B^+ Q_C^+$)	out Z
S_0 000	0	0	S_0 000	0
S_0 000	1	1	S_1 001	0
S_1 001	0	2	S_0 000	0
S_1 001	1	3	S_2 010	0
S_2 010	0	4	S_3 011	0
S_2 010	1	5	S_2 010	0
S_3 011	0	6	S_0 000	0
S_3 011	1	7	S_4 100	0
S_4 100	0	8	S_0 000	1
S_4 100	1	9	S_2 010	1

3 State encoding → كل واحد بيميزها binary bit

→ counting order

at leaste 3 → عشان يطلع 5-states

$S_0 = 000$

$S_1 = 001$

$S_2 = 010$

$S_3 = 011$

$S_4 = 100$

4) input output equation must be optimized so we will ues k-maps

* 3-input equation because we have 3-FF's → (3-bits)

* 1-output eq for Z

* 4-k-maps

* Size 16 $Q_A\ Q_B\ Q_C\ X \equiv 2^4 = 16$

Q_A^+

	Q_C			
	0	1	3	2
	0	0	0	0
	4	5	7	6
	0	0	$\boxed{1}$	0
Q_A	12	13	15	14
	0	0	0	0
	8	9	11	10
	0	0	0	0

Q_B

X

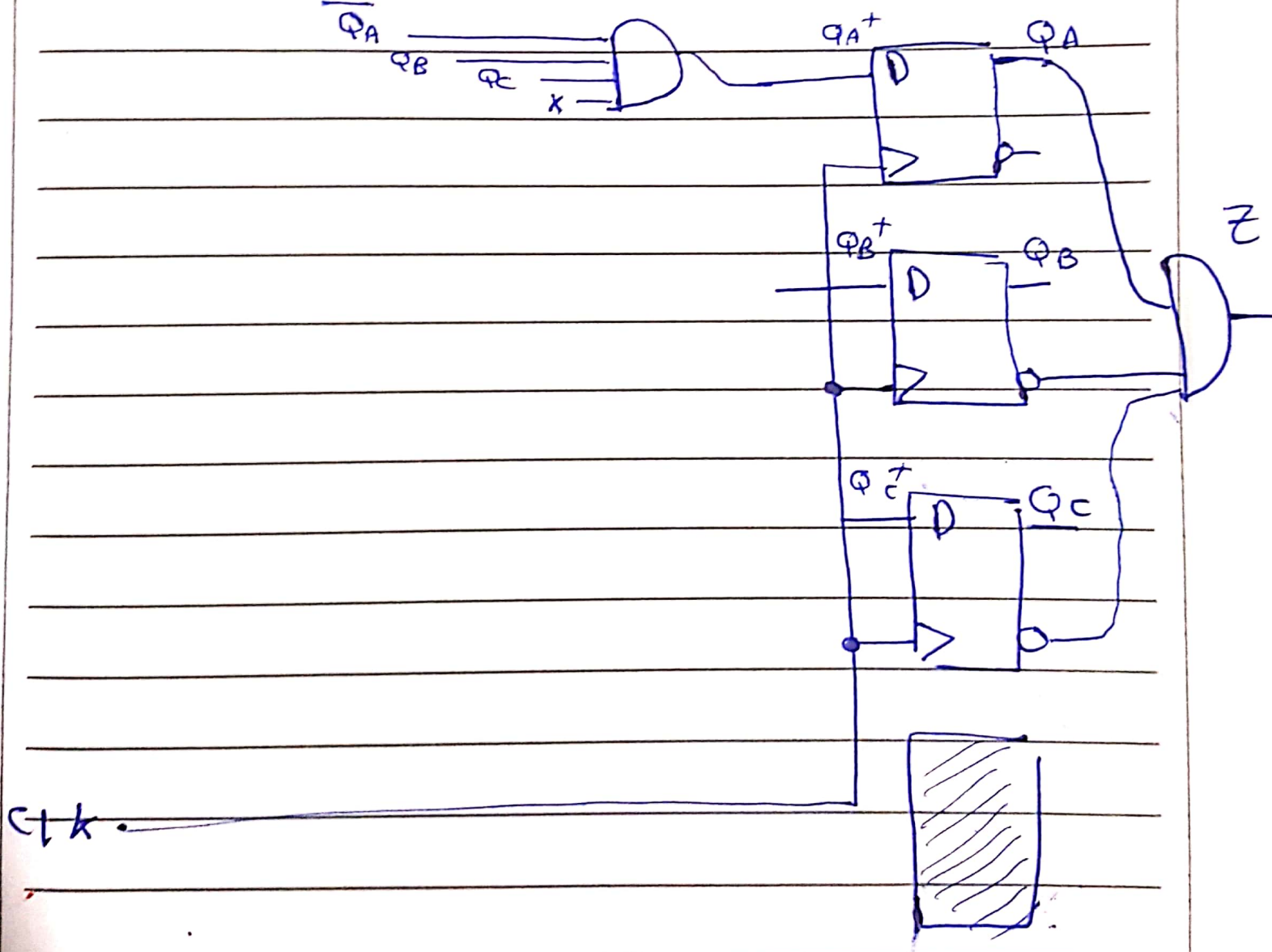
الباقي الي مو موجودة بتعتبرهم 0

$Q_A^+ = \bar{Q}_A\ Q_B\ Q_C\ X$

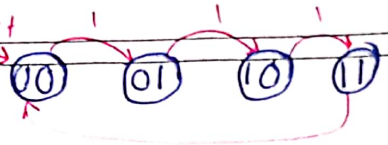
← الـ K-map لـ Z لأنها moore وما بتعتمد
على الـ input بقدر أكتبها بس 8

$Z = Q_A \overline{Q_B}\,\overline{Q_C}$ ← state 4

[5] Design 3-FF's



Free running counter



Exercise

2-input

8 بدل 16 ← K-map ←

فيرجع لأول State

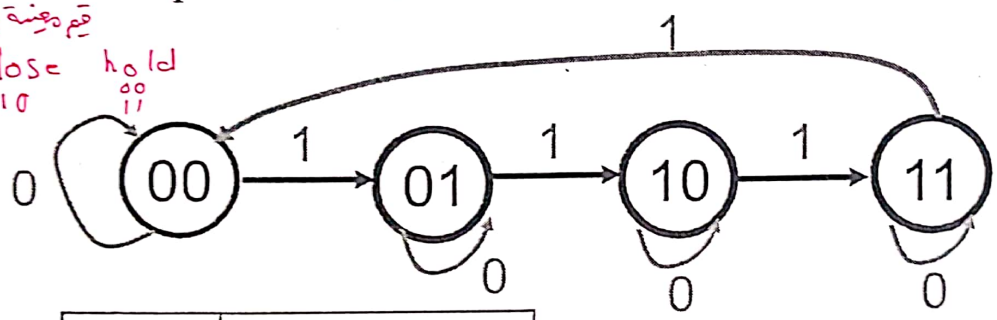
sequential

- Use D Flip-Flops design a counter that counts 00,01,10,11,00,01,10,11, ..etc. ← 2-bit counter Max value = 11 = 3
- The counter also has an input x such that the counter pauses if x=0 and proceeds to the next state if x=1.

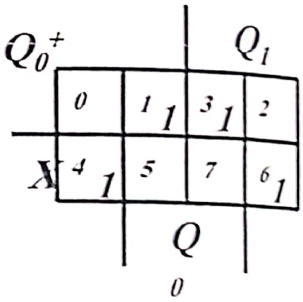
2-input يعني x y

إذا بدي أحدد قيم معينة يشتغلها

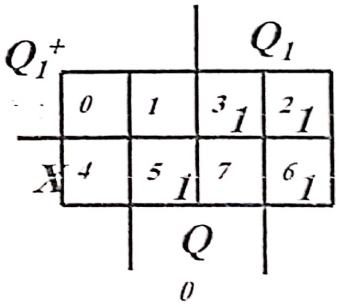
up 01 close 10 hold 00 11



Present State Q_1Q_0	Next State X = 0 $Q_1^+Q_0^+$	Next State X = 1 $Q_1^+Q_0^+$
00	0 00	1 01
01	2 01	3 10
10	4 10	5 11
11	6 11	7 00



$$Q_0^+ = X \oplus Q_0$$



$$Q_1^+ = \overline{X}Q_1 + Q_1\overline{Q_0} + X\overline{Q_1}Q_0$$

Unused States in Sequential Circuits Design

- Unused states are states which the system <u>cannot</u> enter under normal operation.
- The system can enter an unused state due to:
  - Outside interference OR
  - Malfunction
- Three ways to accommodate unused states:
  - Assume the next state for the unused state to be don't care
  - Force the next state for the unused state to be one of the used states
  - * Include a special output to indicate that the present state is unused. This output can change the state asynchronously through direct inputs of the state flip-flops

بنفرضها
انها 0

Exercise

size of k-map is 16 → 2^4 ← A, B, C, D
3-bits → 3 FF's

- Use D-FFs to design the sequential circuit that implements the following state table. Note that there are three unused states (000, 110 and 111).

	Present State			Input	Next State		
	A	B	C	X	A^+	B^+	C^+
1	0	0	1	0	0	0	1
	0	0	1	1	0	1	0
2	0	1	0	0	0	1	1
	0	1	0	1	1	0	0
3	0	1	1	0	0	0	1
	0	1	1	1	1	0	0
4	1	0	0	0	1	0	1
	1	0	0	1	1	0	0
5	1	0	1	0	0	0	1
	1	0	1	1	1	0	0

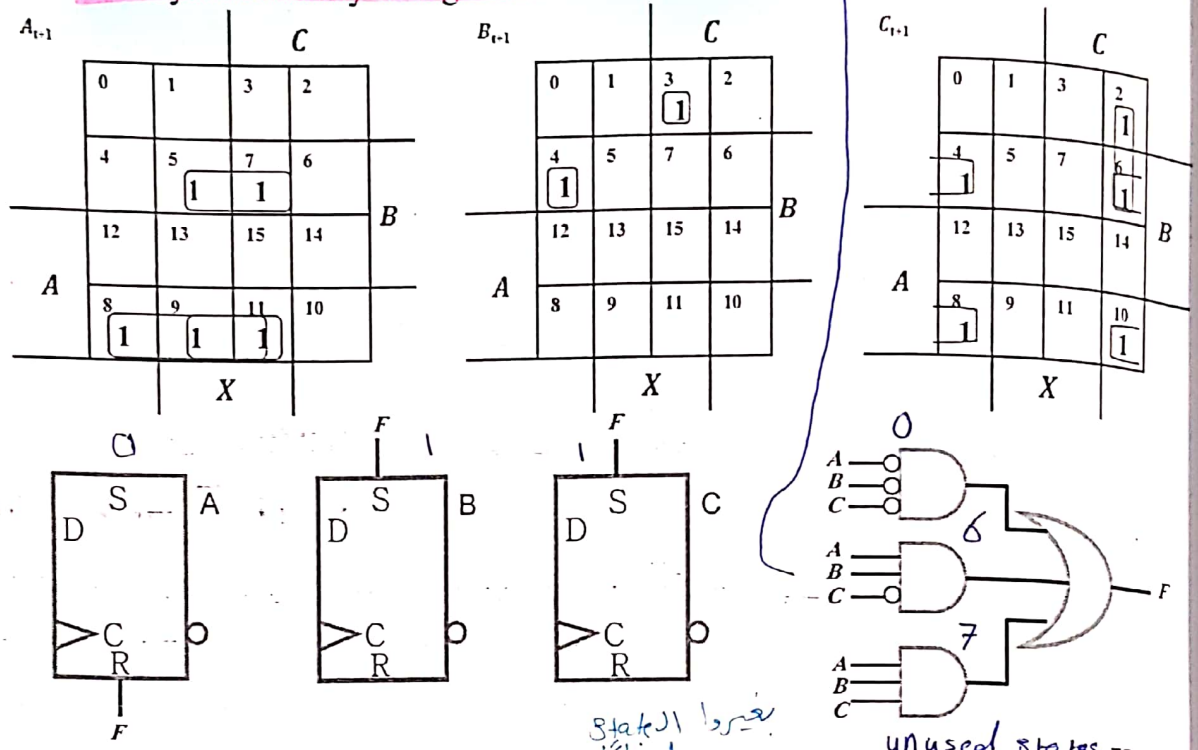
0 6 7

← رتبتهم بهاي الطريقة لأنه ما عندي (0,1) وإلي بعده أكبر هو 0

Solution

$A\,B\,\bar{C}$
$1\,1\,0 = 6$

- Asynchronously change the state to "011"



بغيروا الـ State لوحدها

لا بنقدر نلزم مبرمج الـ FF ال designed يخليها دايماً تروح على حالة ع
فستبلش الـ S=1 على F و الـ R=0 على F عشان يعطي م كا يكون بالـ unused

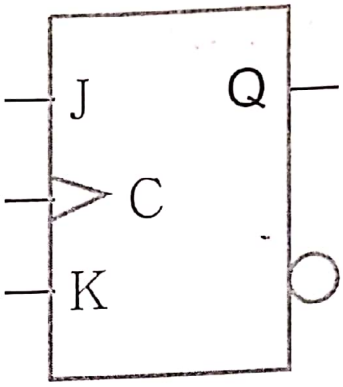
5-6 Other Flip-Flop Types

- J-K and T flip-flops
  - Behavior
  - Implementation
- Basic descriptors for understanding and using different flip-flop types
  - Characteristic tables
    - Defines the next state as a function of the present state and input
  - Characteristic equations
  - Excitation tables

J-K Flip-flop

- Behavior of JK flip-flop:
  - Same as S-R flip-flop with J analogous to S and K analogous to R
  - Except that J = K = 1 is allowed, and
  - For J = K = 1, the flip-flop changes to the *opposite state* (toggle)
- Behavior described by the characteristic table (function table):

عملي ال input
بحطلها كل الاحتمالات وبعرف بطلعلها ال next state



J	K	Q(t+1)	
0	0	Q(t)	no change
0	1	0	reset
1	0	1	set
1	1	$\overline{Q}(t)$	toggle →

بنعكس ال Q(t)

Design of an edge-triggered J-K Flip-Flop

$Q(t+1) = Q(t)\bar{J}\bar{K} + 1 \cdot J\bar{K} + \overline{Q(t)}\,J\,K$

State table of a JK FF:

	Present state Q	Inputs J	K	Next state Q(t+1)
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	0
6	1	1	0	1
7	1	1	1	0

$Q(t+1)=D_A$

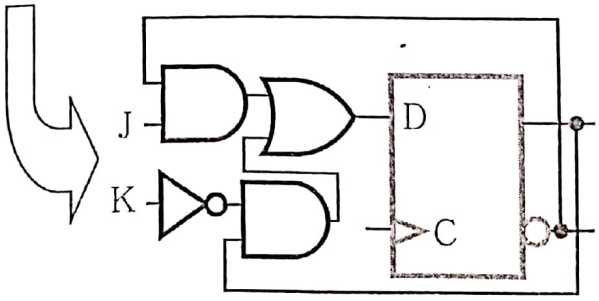
Q \ J	0	1
0	0	1
1	0	1

K

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

Called the characteristic equation

نفسها



بس نتبهوا انه نحط كل الاحتمالات .

J-K Flip-Flop Excitation Table

~~Exciting~~ Excitation ← عيز وين ال input ب

← هون نتفرج على Q(t) و Q(t+1) وبناءً عليه بحدد شو لازم يكون ال input

Q(t)	Q(t +1)	J (S)	K (R)	Operation
0	0	0	X	No change
0	1	1	X	Set
1	0	X	1	Reset
1	1	X	0	No Change

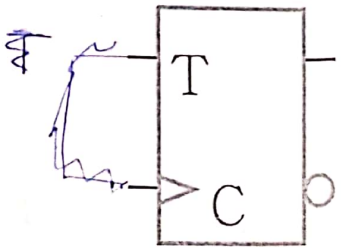
T Flip-Flop

- Behavior described by its characteristic table:
  - Has a single input T
    - For T = 0, no change to state
    - For T = 1, changes to opposite state
- Same as a J-K flip-flop with J = K = T

T	Q(t+1)
0	Q(t) no change
1	$\overline{Q}(t)$ complement

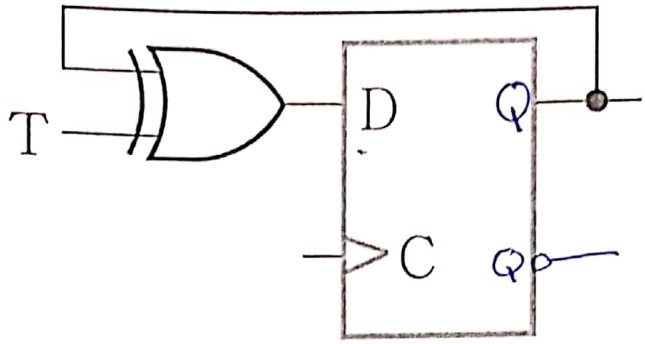
Characteristic equation:

$$Q(t+1)=\overline{T}Q(t) + T\overline{Q}(t)$$
$$= T \oplus Q(t)$$



T Flip-Flop Realization

- Using a D Flip-flop: $D=T\oplus Q(t)$



- Cannot be initialized to a known state using the T input
  - Reset (asynchronous or synchronous) essential

ال set
و ال Reset
لازم تكون من خلال direct

T Flip-Flop Excitation Table

Next state

Q(t +1)	T	Operation
$Q(t)$	0	No change
$\overline{Q}(t)$	1	Complement

Flip-Flops Characteristics

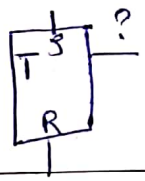
For analysis

- *Characteristic table* - defines the next state of the flip-flop in terms of flip-flop inputs and current state
- *Characteristic equation* - defines the next state of the flip-flop as a Boolean function of the flip-flop inputs and the current state.

For design

- *Excitation table* - defines the flip-flop input variable values as function of the current state and next state. In other words, the table tells us what input is needed to cause a transition from the current state to a specific next state.

* لازم أعرف input S و R عشان T ما عندها قيمة ابتدائية .



D Flip-Flop Descriptors

- Characteristic Table

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

- Characteristic Equation

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

- Excitation Table

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

S-R Flip-Flop Descriptors

- Characteristic Table

S	R	Q(t+1)	Operation
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	?	Undefined ← not included

$Q(t+1) = Q(t)\,S\bar{R} + 0 \cdots + 1\,S\bar{R}$

$= Q(t)\,S\bar{R} + S\bar{R}$

$= S + \bar{R}\,Q(t)$

- Characteristic Equation

$$Q(t+1) = S + \bar{R}\,Q,\ S \cdot R = 0$$

- Excitation Table

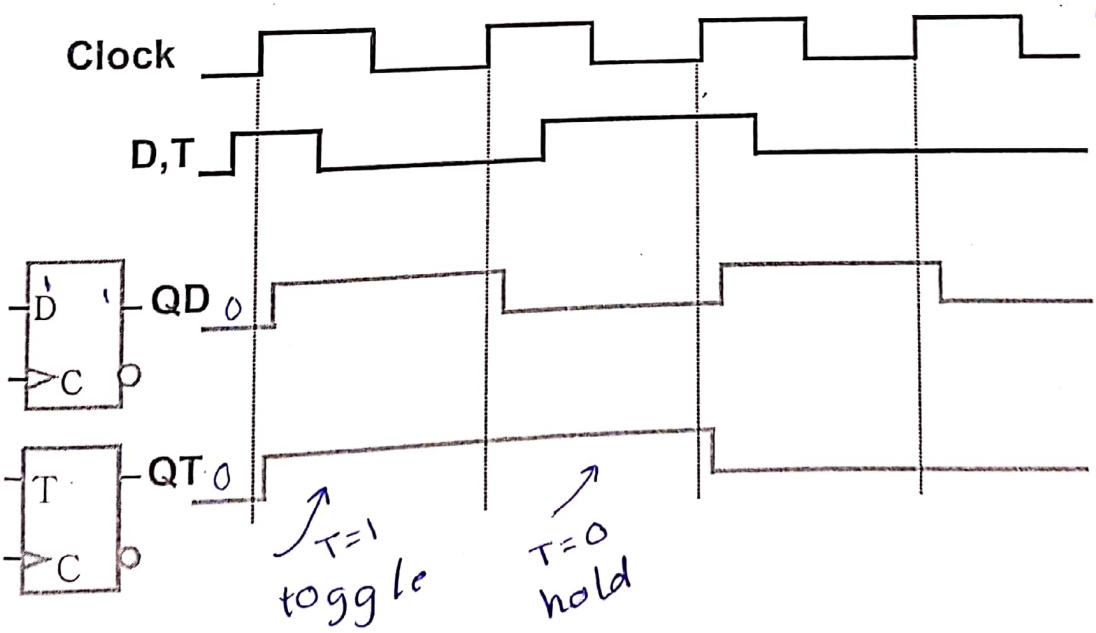
Q(t)	Q(t+1)	S	R	Operation
0	0	0	X	No change
0	1	1	0	Set
1	0	0	1	Reset
1	1	X	0	No change ←

Flip-flop Behavior Example

T=0 hold
T=1 toggle
D=0 $Q^+=0$
D=1 $Q^+=1$

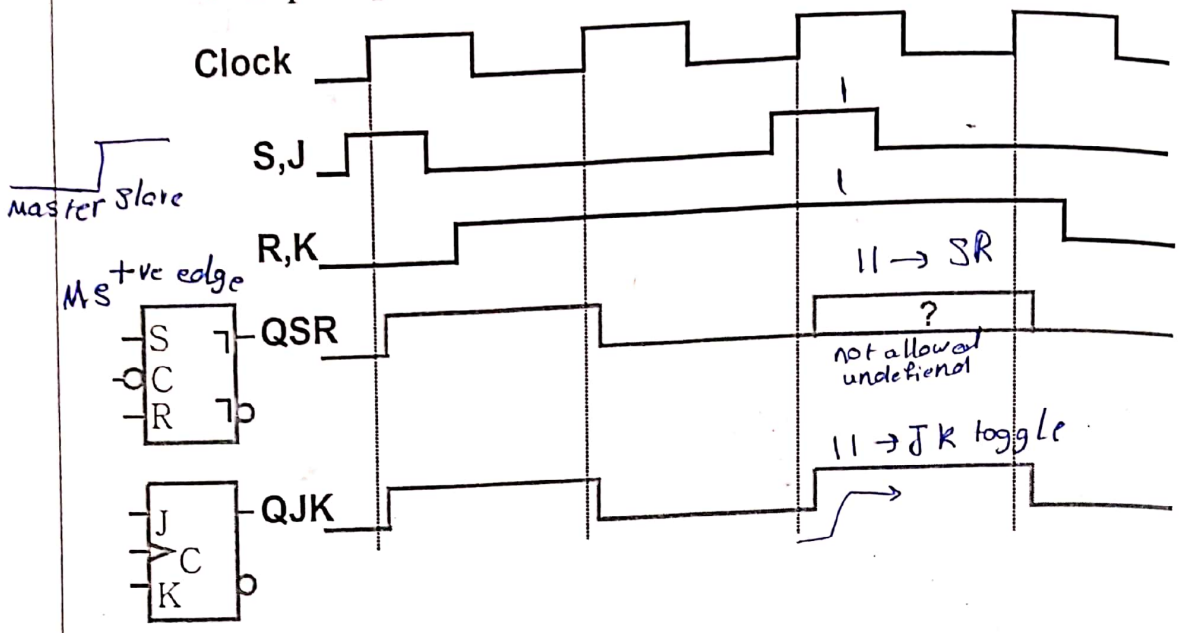
- Use the characteristic tables to find the output waveforms for the flip-flops shown:

X تتغير عند positive edge
هون لأنه الـ clk
not inverted



Flip-Flop Behavior Example (continued)

- Use the characteristic tables to find the output waveforms for the flip-flops shown:



هي موجودة ⊕ next state بالبيانات

Moore → لأنها ما تعتمد على input ال

Exercise: Find State Diagram

3-TFF

Analysis Data

* خطوات حل السؤال :-

[1] Input & output equations

no input

one output (y)

$T_A = \bar{C}$

$T_B = A$

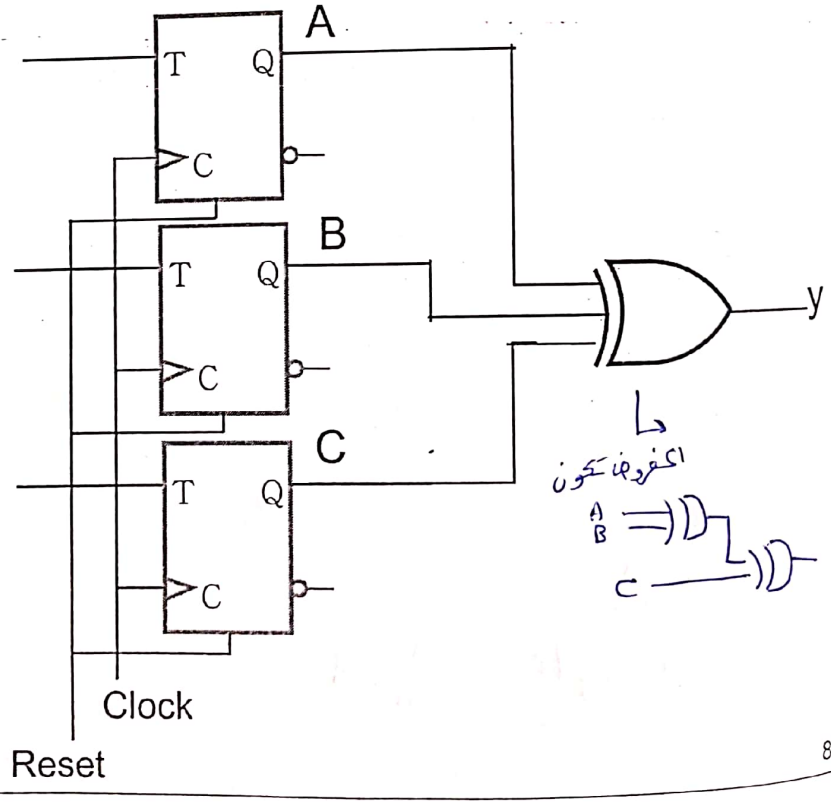
$T_C = \bar{B}$

input

$y = A \oplus B \oplus C$ output

[2] State Table

← بشوف ال (T) وبناءً عليه نحدد toggle أو Nochange



الغرض تكون

1 ← Toggle
0 ← hold

4-k-map

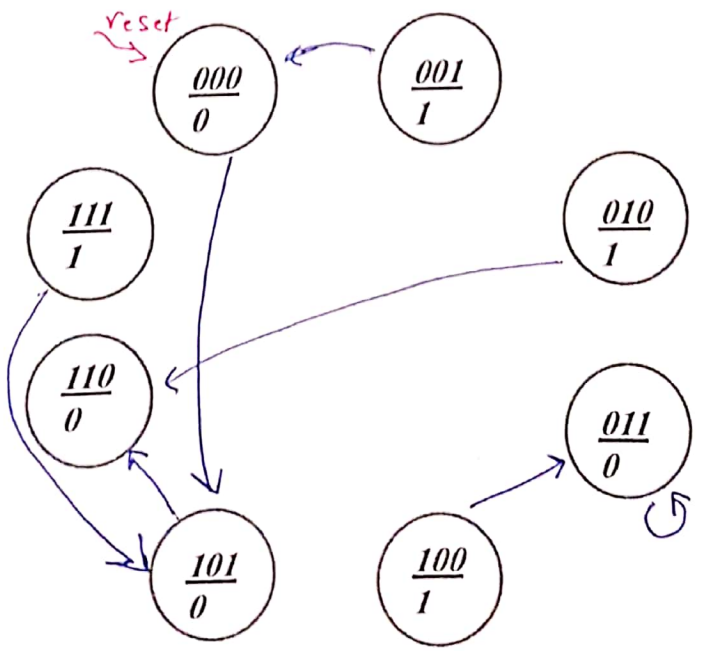
$Q(t+1) = Q(t) \oplus T$

Analysis → charasta

	Present state						Next State			Y
	A	B	C	T_A	T_B	T_C	A+	B+	C+	y
0	0	0	0	1	0	1	1 toggle	0	1	0
1	0	0	1	0	0	1	0 hold	0	0	1
2	0	1	0	1	0	0	1	1	0	1
3	0	1	1	0	0	0	0	1	1	0
4	1	0	0	1	1	1	0	1	1	1
5	1	0	1	0	1	1	1	1	0	0
6	1	1	0	1	1	0	0	0	0	0
7	1	1	1	0	1	0	1	0	1	1

بنبدا من عند الـ reset

0 → 5 → 6



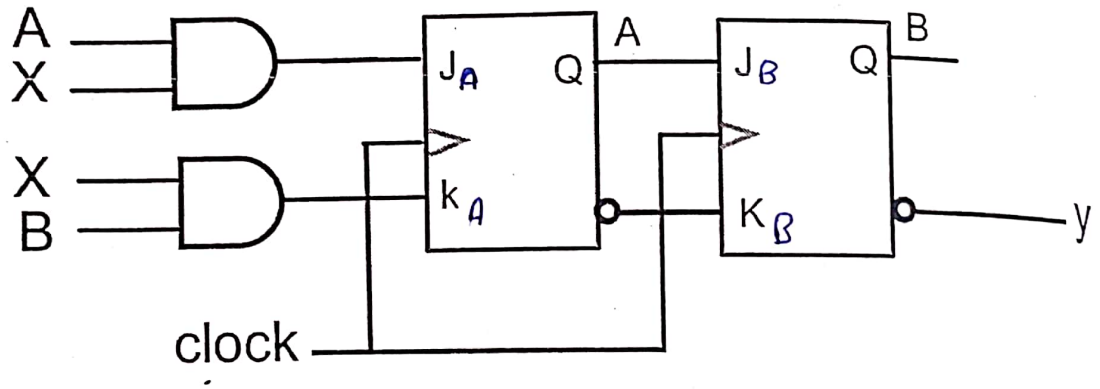
Exercise: Find State Diagram

Input: X

State: A B

Next state: A^+ B^+

output = y



* Input Eq:

$J_A = A \cdot X$

$J_B = A$

$k_A = B \cdot X$

$k_B = \overline{A}$

output

$y = \overline{B}$

Moore

J K
0 0 → no change
0 1 → reset
1 0 → set
1 1 → toggle

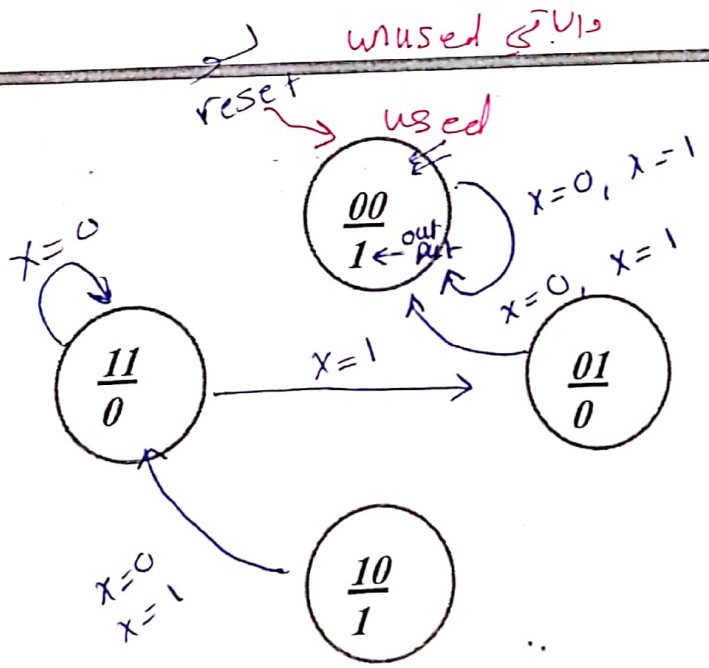
Analysis ⟹ charac...

Moore

Present state		Input					Next State		Y
A	B	X	J_A	K_A	J_B	K_B	A+	B+	y
0	0	0	0	0	0	1	0	0	1
0	0	1	0	0	0	1	0	0	1
0	1	0	0	0	0	1	0	0	0
0	1	1	0	1	0	1	0	0	0
1	0	0	0	0	1	0	1	1	1
1	0	1	1	0	1	0	1	1	1
1	1	0	0	0	1	0	1	1	0
1	1	1	1	1	1	0	0	1	0

reset

$\bar{B}$ a



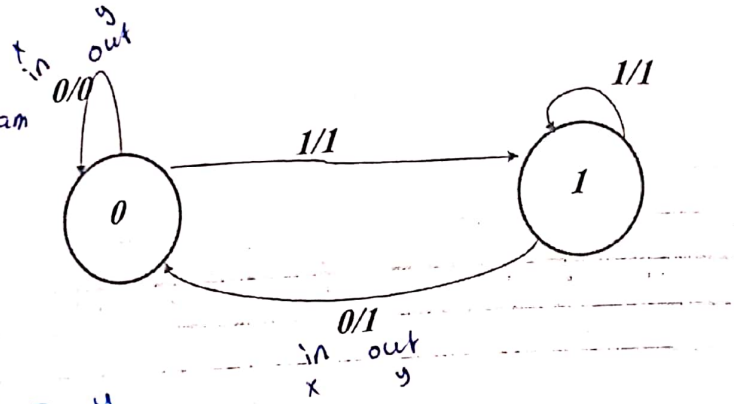
* بنعالج ال TA بدل ال input & present state
* إذا كانت D على طول من ال next state
* هون لازم أعملها جدولها
* هون، ح نستخدم ال excition table
* ال TA مو نفسها A^+

Mealy

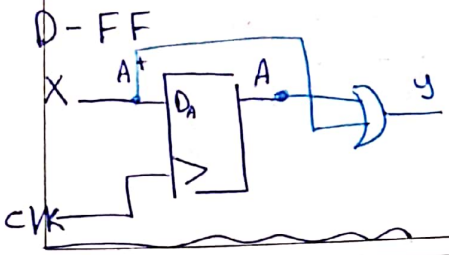
1- flip flop → 1-bit

- Given the following state diagram design the sequential circuit that implements it. the sequential circuit that implements it. Compare the design when TFF & DFF is used.

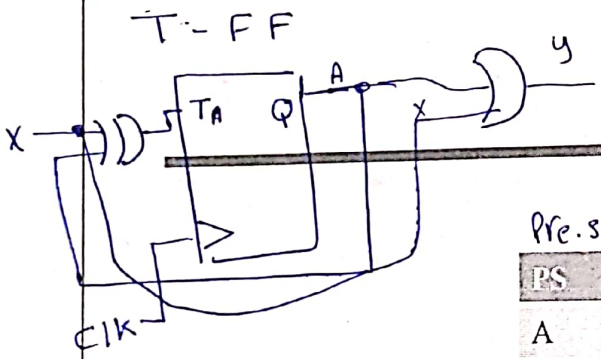
* خطوات حل السؤال :-
[1] Derive state diagram
[2] State table



D-FF



T-FF



$y = A + X$

Pre. state PS	input	Next. sta NS	out	TA
A	X	A^+	y	
0	0	0	0	0
0	1	1	~~0~~ 1	1
1	0	0	~~0~~ 1	1
1	1	1	1	0

$\boxed{1 = A^+}$ متى ال

$A^+ = \bar{A}X + AX = X$

*Excitation table for FF-T

$T_A = A \oplus X$

Q(t)	Q(t+1)	TA
0	0	0
0	1	1
1	0	1
1	1	0

Exercise

* مو ضروري الـ 2 يكونوا نفس النوع

* عدد الـ FF ← 2
عشان الـ 2-bits

Design the sequential circuit that implements the following state table using

- JK Flip-Flops
- T Flip-Flops
- SR Flip-Flops
- D Flip-Flops

2-flip flop.

Present State		Input	Next State		Output
A(t)	B(t)	X	A(t+1)	B(t+1)	Y
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	0
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	0	0	1

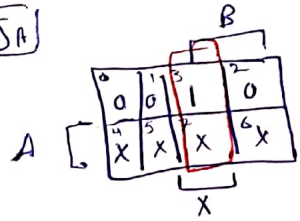
$2^3 = 8$ ← A, B, X size of k-map

num. of k-maps = 5 ⇒ J_A K_A J_B K_B y

Using JK FF

	Present state							Next State		Y
	A	B	X	J_A	K_A	J_B	K_B	A+	B+	y
0	0	0	0	0	X	0	X	0	0	0
1	0	0	1	0	X	1	X	0	1	1
2	0	1	0	0	X	X	0	0	1	0
3	0	1	1	1	X	X	1	1	0	1
4	1	0	0	X	0	0	X	1	0	1
5	1	0	1	X	0	1	X	1	1	1
6	1	1	0	X	0	X	0	1	1	1
7	1	1	1	X	1	X	1	0	0	1

J_A



$J_A = BX$

Subject

Date: / /

$\boxed{J_A}$

0: 0	1: 0	3: 1	2: 0
4: X	5: X	7: X	6: X

$J_A = B X$

$\boxed{J_B}$

0: 0	1: 1	3: X	2: X
4: 0	5: 1	7: X	6: X

$J_B = X$

$\boxed{K_A}$

0: X	1: X	3: X	2: X
4: 0	5: 0	7: 1	6: 0

$K_A = X B$

$\boxed{K_B}$

0: X	1: X	3: 1	2: 0
4: X	5: X	7: 1	6: 0

$K_B = X$

الموضوع

Subject

التاريخ: / /

Date: / /

y)

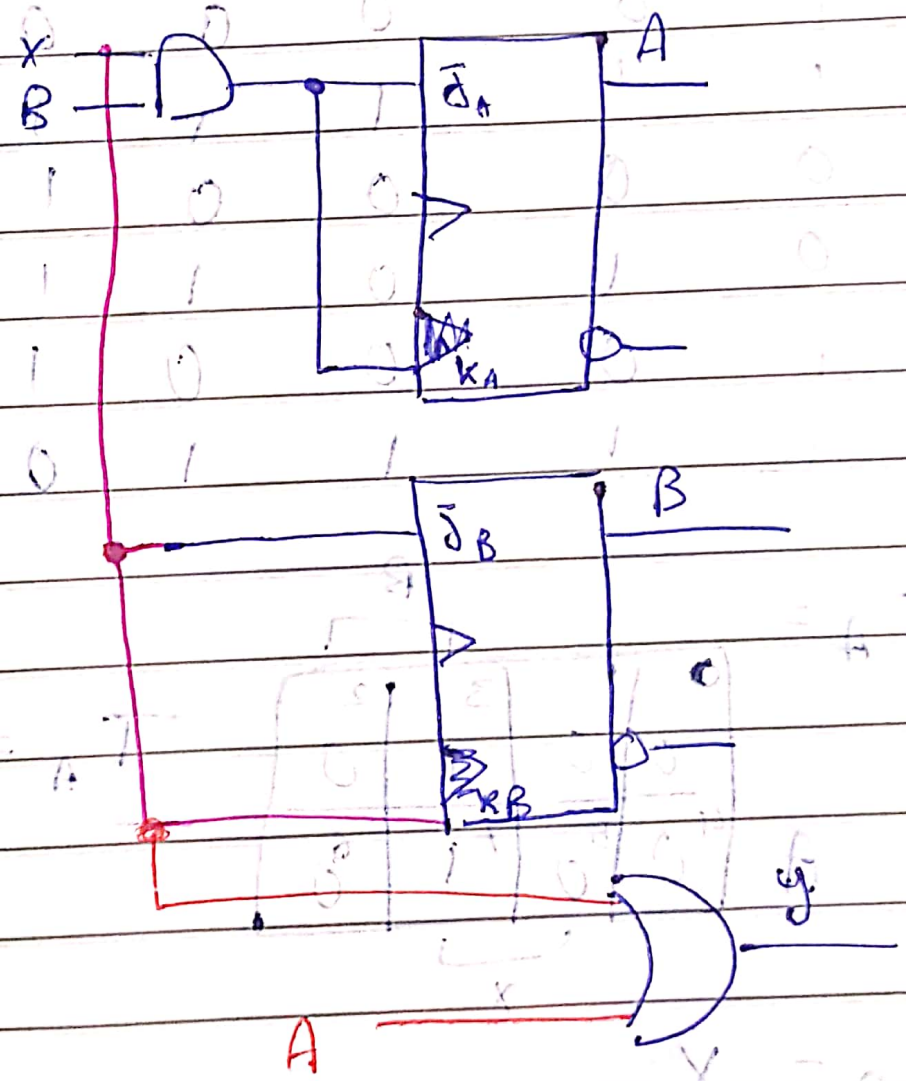
0	1	3	2
0	1	1	0
4	5	7	6
1	1	1	1

B

A

X

$y = A + X$



2- FF

Using TFF

	Present state					Next State		Y
	A	B	X	T_A	T_B	A+	B+	y
0	0	0	0	0	0	0	0	0
1	0	0	1	0	1	0	1	1
2	0	1	0	0	0	0	1	0
3	0	1	1	1	1	1	0	1
4	1	0	0	0	0	1	0	1
5	1	0	1	0	1	1	1	1
6	1	1	0	0	0	1	1	1
7	1	1	1	1	1	0	0	1

2- FF

Using SR FF

	Present state							Next State		Y
	A	B	X	S_A	R_A	S_B	R_B	A+	B+	y
0	0	0	0	0	X	0	X	0	0	0
1	0	0	1	0	X	1	0	0	1	1
2	0	1	0	0	X	X	0	0	1	0
3	0	1	1	1	0	0	1	1	0	1
4	1	0	0	X	0	0	X	1	0	1
5	1	0	1	X	0	1	0	1	1	1
6	1	1	0	X	0	X	0	1	1	1
7	1	1	1	0	1	0	1	0	0	1

$T_A =$

		B		
	0	1	3	2
	0	0	1	0
A	4	5	7	6
	0	0	1	0
		X		

$T_A = BX$

$T_B = X$

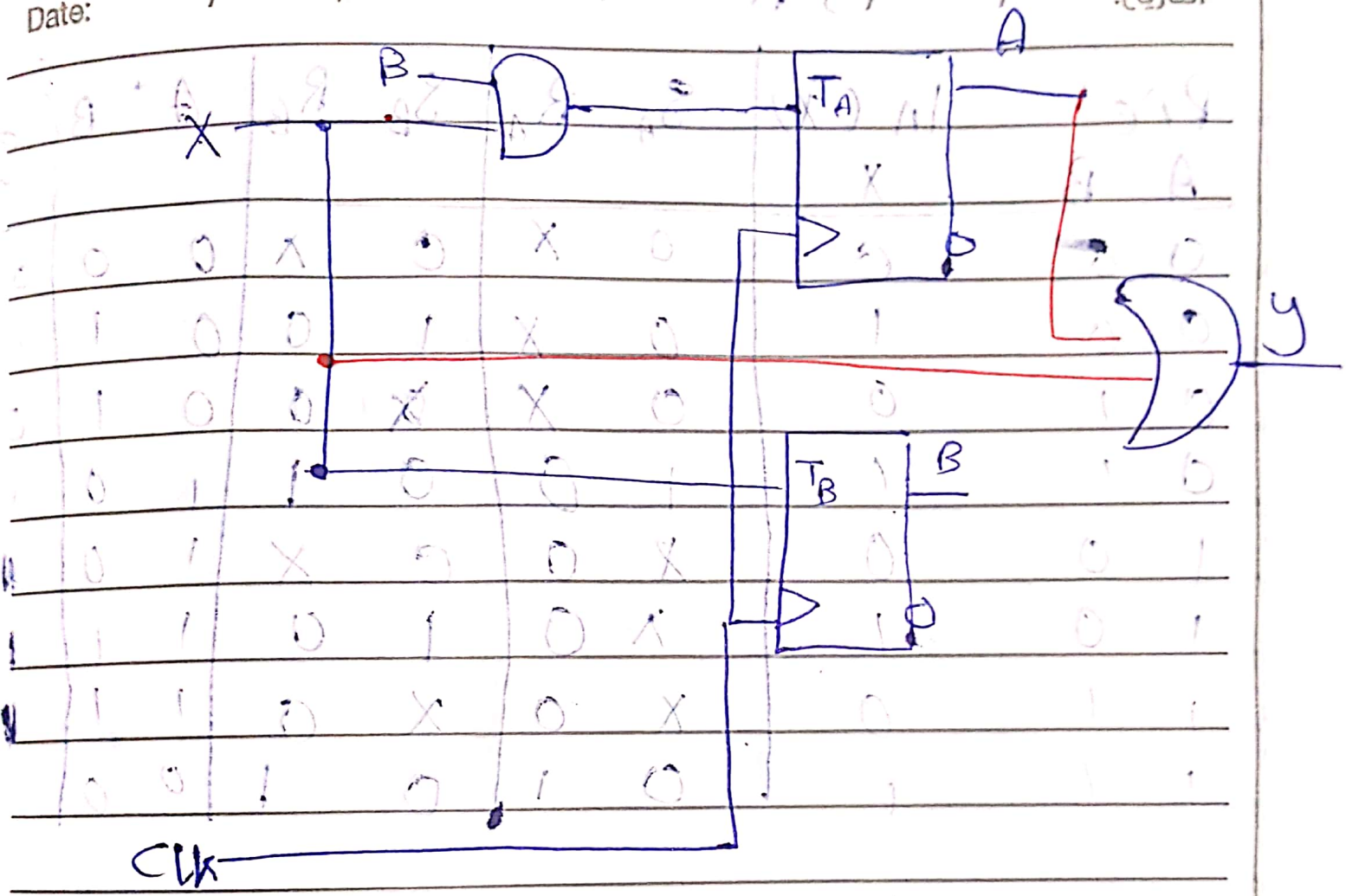
		B		
y	0	1	3	2
	0	1	1	0
A	4	5	7	6
	1	1	1	1

$y = A + X$

ABC

Subject

Date: / / التاريخ: / /



Exercise

Design the sequential circuit that implements the following state table using

- JK Flip-Flops
- T Flip-Flops
- SR Flip-Flops
- D Flip-Flops

Present State		Input	Next State		Output
A(t)	B(t)	X	A(t+1)	B(t+1)	Y
0	0	0	0	1	1
0	0	1	0	0	1
0	1	0	1	0	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	1	0	0	1
1	1	0	1	1	0
1	1	1	1	0	0