POWER UNIT

MOHAMMED ABD-ALMAJEED
LOGIC

SHYAM MAWAJDEH

POWER UNIT-JU

http : // POWER UNIT-JU.COM

Scanned by CamScanner

* Raghad Imad Jaber. *

" Shyam Khaled Mawgdeh "

Logic and Computer Design Fundamentals

Chapter 1 – Digital Systems and Information

" Shyam Mawgdeh "

Charles Kime & Thomas Kaminski

© 2008 Pearson Education, Inc.

(Hyperlinks are active in View Show mode)

Scanned by CamScanner

NUMBER SYSTEMS – Representation : (تمثيل - الأنظمة العددية)

- Positive radix (الأساس), positional number systems
- A number with *radix r* is represented by a string of digits:

$$A_{n-1}, A_{n-2}, \ldots, A_1, A_0 \bullet A_{-1}, A_{-2}, \ldots, A_{-m+1}, A_{-m}$$

(MSD) أعلى قيمة * بدون مساواة : مستحيل يوصل لنفس عدد الأساس. دائماً أقل بواحد — radix-point : فاصلة عشرية — (LSD)

in which $0 \le A_i < r$ and $\bullet$ is the *radix point* or (Decimy-point).

- i represents the position of the coefficient (المنزلة)
- r^i represents the weight by which the coefficient is multiplied (معامل) (الخانة)(وزن المنزلة)
- A_{n-1} is the most significant digit (MSD) and A_{-m} is the least significant digit (LSD)
  - أعلى قيمة خانة. (المنزلة ذات أعلى وزن)
  - أقل قيمة خانة. (المنزلة ذات أقل وزن).
- The string of digits represents the power series:

$$(Number)_r = \left(\sum_{i=0}^{n-1} A_i r^i\right) + \left(\sum_{j=-m}^{-1} A_j r^j\right)$$

$\sum_{i=-m}^{n-1} A_i r^i$ (متسلسلة) — مجموع

في حالة وجود كسور عشرية.

* radix : الأساس
* Decimal system : النظام العشري. أساسه العدد (١٠)
* radix-point : فاصلة عشرية.
* Binary-point : فاصلة النظام الثنائي.

Integer Portion الأعداد الصحيحة — **Fraction Portion** الأعداد العشرية.

Scanned by CamScanner

Number Systems – Examples

		General	Decimal (r=10) النظام العشري	Binary (r=2) النظام الثنائي
Radix (Base)		r	10	2
Digits		0 → r - 1	0 → 9	0 → 1
		$(0 \leq A_i < r) \rightleftharpoons (0 \leq A_i \leq r-1)$		
Powers of Radix	0	r^0	$10^0 = 1$	$2^0 = 1$
	1	r^1	$10^1 = 10$	$2^1 = 2$
	2	r^2	$10^2 = 100$	$2^2 = 4$
	3	r^3	$10^3 = 1000$	$2^3 = 8$
	4	r^4	$10^4 = 10,000$	$2^4 = 16$
	5	r^5	$10^5 = 100,000$	$2^5 = 32$
	-1	r^{-1}	$10^{-1} = 0.1 = \frac{1}{10}$	$2^{-1} = 0.5 = \frac{1}{2}$
	-2	r^{-2}	$10^{-2} = 0.01 = \frac{1}{100}$	$2^{-2} = 0.25 = \frac{1}{4}$
	-3	r^{-3}	$10^{-3} = 0.001 = \frac{1}{1000}$	$2^{-3} = 0.125 = \frac{1}{8}$
	-4	r^{-4}	$10^{-4} = 0.0001 = \frac{1}{10000}$	$2^{-4} = 0.0625 = \frac{1}{16}$
	-5	r^{-5}	$10^{-5} = 0.00001 = \frac{1}{100000}$	$2^{-5} = 0.03125 = \frac{1}{32}$

Scanned by CamScanner

Example

1. $(403)_5 = 4 \times 5^2 + 0 \times 5^1 + 3 \times 5^0 = (103)_{10}$

نوع النظام (خماسي) وهو نفسه الـ radix (الأساس).

digits must be: كل ما *

$0 \leq A_i \leq 4$.

or

$0 \leq A_i < 5$.

$(605)_7$ نوع النظام (سباعي)

$= (\quad)_{10}$

2. $(103)_{10} = 1 \times 10^2 + 0 \times 10^1 + 3 \times 10^0 = 103$

* Example (1): 92871.

position: 4 3 2 1 0

10^0 : (1) : آحاد : (واحد)

10^1 : (10) : عشرات : (تسعون)

10^2 : (100) : مئات : (ثمانمئة)

10^3 : (1000) : ألوف : (أربعين)

... وهكذا!

* Example (2): change from binary system → to decimal system:

binary-point ... (فاصلة النظام الثنائي)

101.10 → in binary.

$2^2 * 1 + 2^1 * 0 + 2^0 * 1 + . 2^{-1} * 1 + 2^{-2} * 0$.

$4 + 1 + 0.5 =$

5.5 → in decimal.

Scanned by CamScanner

BASE CONVERSION - Positive Powers of 2

- Useful for Base Conversion : * مفيد للتحويلات بين الأنظمة

* ما عليه نجمة (*) - يجب حفظه ضروري لتسهيل الحل ..

Exponent	Value
* 0 = 2^0:	1
* 1 = 2^1:	2
* 2 = 2^2:	4
* 3 = 2^3:	8
* 4 = 2^4:	16
* 5 = 2^5:	32
* 6 = 2^6:	64
* 7 = 2^7:	128
* 8 = 2^8:	256
* 9 = 2^9:	512
* 10 = 2^{10}:	1024

Exponent	Value
* 11 = 2^{11}:	2,048
12	4,096
13	8,192
14	16,384
15	32,768
16	65,536
17	131,072
18	262,144
19	524,288
* 20 = 2^{20}:	1,048,576
21	2,097,152

$2^{14} = \frac{2^{10}}{K} \cdot 2^4 = 16K$

* 2^{30} : 1073741824. G
* 2^{40} : $1.09 * 10^{12}$. T
* 2^{50} : $1.1 * 10^{15}$.

Scanned by CamScanner

Special Powers of 2

* memorize.

* ▪ 2^{10} (1024) is Kilo, denoted "K"

لأن الكيلو = 1000 وأقرب عدد لـ (1000) هو (1024) وهو (2^{10})

* Example: 20. KB ⇒ 20 * 1024 كيلو byte.

* ▪ 2^{20} (1,048,576) is Mega, denoted "M"

* ▪ 2^{30} (1,073, 741,824)is Giga, denoted "G"

* ▪ 2^{40} (1,099,511,627,776) is Tera, denoted "T"

* NOte: byte = 8.bite البايت = ٨ بت.

Scanned by CamScanner

Commonly Occurring Bases

Name (النظام)	Radix (الأساس)	Digits (القيم المحتملة)
Binary الثنائي	2	0,1
Octal الثماني	8	0,1,2,3,4,5,6,7
Decimal العشري	10	0,1,2,3,4,5,6,7,8,9
Hexadecimal السادس عشري	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

- The six letters A, B, C, D, E, and F represent the digits for values 10, 11, 12, 13, 14, and 15 (given in decimal), respectively, in hexadecimal. Alternatively, a, b, c, d, e, f can be used.

Scanned by CamScanner

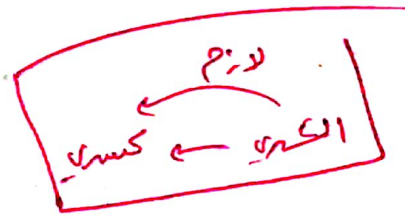
Binary System النظام الثنائي :-

- $r = 2$ الأساس
- Digits = {0, 1} أو {false, True} أو {low, high} أو {off, on}.
- Every binary digit is called a bit * هي صفة للنظام الثنائي فقط ! فكل قيمة (لا وثنية) قيمتها تسمى (بت) . (1,0)
- When a bit is equal to zero, it does not contribute to the value of the number * يمكن استثناء الـ (0) من البداية قبل حل السؤال والتحويل للنظام العشري . لأن الصفر عند ضربه بأي قيمة يكون الجواب النهائي = صفر.
- Example:
  - دون الاستثناء. $(10011.101)_2 = (1\times 2^4 + 0\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0) + (1\times 2^{-1} + 0\times 2^{-2} + 1\times 2^{-3})$
  - بالاستثناء. $(10011.101)_2 = (16 + 2 + 1) + \left(\frac{1}{2} + \frac{1}{8}\right) = (19.625)_{10}$

Scanned by CamScanner

Octal System النظام الثماني:

- $r = 8$
- Digits = {0, 1, 2, 3, 4, 5, 6, 7} : $(0 \leq A_i \leq r-1)$: الصيغة
- Every digit is represented by 3-bits → More compact than binary
  * كل قيمة رقمية في النظام الثماني تعادل (٣ قيم رقمية) في النظام الثنائي.
- Example:
  - $(127.4)_8 = (1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0) + (4 \times 8^{-1})$
    2 1 0 -1
  - $(127.4)_8 = (64 + 16 + 7) + \left(\frac{1}{2}\right) = (87.5)_{10}$



Scanned by CamScanner

Hexadecimal System النظام السادس عشري:

- $r = 16$
- Digits = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, $\underline{A}^{10}$, $\underline{B}^{11}$, $\underline{C}^{12}$, $\underline{D}^{13}$, $\underline{E}^{14}$, $\underline{F}^{15}$} $(0 \leq Ai \leq 15)$
- Every digit is represented by 4-bits
  * كل قيمة رقمية في النظام السادس عشري تعادل (٤ قيم رقمية) في النظام الثنائي.
- Example:
  - $(\underset{=32\ 10}{B}65F)_{16} = (\underline{11} \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + \underline{15} \times 16^0)$

* انتبه! : نحوّل الرموز من النظام السادس عشري لقيمتهم العددية في النظام العشري ونضربهم بوزن المقل
  - $(B65F)_{16} = (46687)_{10}$

Scanned by CamScanner

* التحويل من الأنظمة العددية المختلفة إلى النظام العشري:

Converting from any Base (r) to Decimal

$$(Number)_r = \left(\sum_{i=0}^{n-1} A_i r^i\right) + \left(\sum_{j=-m}^{-1} A_j r^j\right)$$

Integer Portion — Fraction Portion

- **Example: Convert $(11010)_2$ to $(N)_{10}$:**

* $(11010)_2 \longrightarrow (?)_{10}$

$1*2^4 + 2^3*1 + 2^2*0 + 2^1*1 + 2^0*0$

$16 + 8 + 2 = (26)_{10}$

Scanned by CamScanner

Conversion from Decimal to Base (r)

- Convert the Integer Part
- Convert the Fraction Part
- Join the two results with a radix point

(*) التقريب لأقرب خانتين او ٣ او ٤ ...
كيف !! مثلاً (,925)₁₀
بدو ايام لأقرب خانتين، نشوف الخانة الـ ٣، اذا كانت تساوي ناتج مقسوم (الاساس / 2) او اكبر منه، نحذفها ونزيد واحد للرقم اللي قبلها
(,925)₁₀ → 5 = 10/2 الاساس، اذن نزيد واحد
(0,93)₁₀ = ... اذا كان اقل مانزيد واحد.

(*) Trancate لأقرب ٢ او ٣ او ... خانات
يعني بنعد عدد الخانات المطلوبة ونحذف الباقي
(,925)₁₀ لأقرب خانتين → (,92)₁₀

* binary: $(r=2)$

MSD → ← LSD

$(000)_2 = (0)_{10}$
$(001)_2 = (1)_{10}$
$(010)_2 = (2)_{10}$
$(011)_2 = (3)_{10}$
$(100)_2 = (4)_{10}$
$(101)_2 = (5)_{10}$
$(110)_2 = (6)_{10}$
$(111)_2 = (7)_{10}$

تتغير كل وحدة — تتغير كل اثنين — تتغير كل أربعة وهكذا.

* octal: $(r=8)$

0, 1, 2, 3, 4, 5, 6, 7

$(10)_8 = (8)_{10}$
$(11)_8 = (9)_{10}$
$(12)_8 = (10)_{10}$
$(17)_8 = (15)_{10}$
$(20)_8 = (16)_{10}$

$(100)_8 = (64)_{10}$
$(77)_8 = (63)_{10}$

* Hexadecimal: $(r=16)$

0, 1, 2, 3, ..., f

$(10)_{16} = (16)_{10}$
$(11)_{16} = (17)_{10}$
$(1f)_{16}$
$(20)_{16} = (32)_{10}$

$(ff)_{16} = 255$
$(100)_{16} = 256$

Conversion Details

To Convert the Integral Part:

1. Repeatedly *divide* the number by the new radix and save the *remainders* until *the quotient is zero*
2. The digits for the new radix are the remainders in *reverse order* of their computation
3. If the new radix is > 10, then convert all remainders > 10 to digits A, B, …

A → 10
B → 11
C → 12
D → 13
E → 14
F → 15

To Convert the Fractional Part:

1. Repeatedly *multiply* the fraction by the new radix and save the *integer digits* of the results until the *fraction is zero or your reached the required number of fractional digits*
2. The digits for the new radix are the integer digits *in order* of their computation
3. If the new radix is > 10, then convert all integers > 10 to digits A, B, …

Scanned by CamScanner

Example: Convert 46.6875_{10} To Base 2

- Convert 46 to Base 2:

Division	Quotient	Remainder
46/2	23	0
23/2	11	1
11/2	5	1
5/2	2	1
2/2	1	0
1/2	0	1

LSD ... MSD

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

$32 + 8 + 4 + 2 = 32 + 14 = (46)_{10}$

- Convert 0.6875 to Base 2:

Multiplication	Answer
0.6875*2	1.3750
0.375*2	0.75
0.75*2	1.5
0.5*2	1.0

MSD ... LSD

$$(0.6875)_{10} = (0.1011)_2$$

$0.625 + 0.0625 = (0.6875)_{10}$

- Join the results together with the radix point:

$$(46.6875)_{10} = (101110.1011)_2$$

Scanned by CamScanner

Example: Convert 153.513_{10} To Base 8

- Convert 153 to Base 8:

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

Division	Quotient (الناتج)	Remainder (الباقي)	
153/8	19	1	LSD
19/8	2	3	
2/8	0	2	MSD

* can be from (0) to (7) for the octal system

- Convert 0.513 to Base 8: (*Up to 3 digits*) عدد المنازل المطلوبة بالجواب.

  - Truncate: (قطع) ترك باقي المنازل.

$$(0.513)_{10} = (0.406)_8$$

  - Round: التقريب.

$$(0.513)_{10} = (0.407)_8$$

Multiplication	Answer	
0.513*8	4.104	MSD
0.104*8	0.832	
0.832*8	6.656	
0.656*8	5.248	LSD

* إذا كان ناتج القسمة دوري ولا ينتهي يحدد لنا بالسؤال عدد المنازل في العدد الكسري.

- Join the results together with the radix point:

$$(153.513)_{10} = (231.407)_8$$

* in the octal system لا يقرب $(\frac{1}{3})$ يقرب $(\frac{5}{6})$ بينما يقرب (4)

Scanned by CamScanner

Example: Convert 423_{10} To Base 16

Division	Quotient	Remainder	
423/16	26	7	LSD
26/16	1	10 (A)	
1/16	0	1	MSD

* انتبه! يجب تحويل الأرقام من (١٠ ← 15) بالرموز المختصة في النظام السادس عشري.

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

Scanned by CamScanner

Converting Decimal to Binary: Alternative Method

يطلب بالامتحان حل السؤال باستخدامها .

الحل باستخدام مضاعفات الاثنين

1. Subtract the largest power of 2 that gives a positive remainder and record the power

نبحث عن أكبر عدد من مضاعفات الـ 2 أقل من الرقم اللي بالسؤال ونطرح اللي يطلع . نرجع نطرح اللي ظل إلى أكبر عدد من مضاعفات الـ 2 أقل منه وهكذا .

2. Repeat, subtracting from the prior remainder and recording the power, until the remainder is zero

3. Place 1's in the positions in the binary result corresponding to the powers recorded; in all other positions place 0's

eg:- $(28)_{10} \rightarrow (11100)_2$

$16 + 12$

$16 + 8 + 4$

$2^4 + 2^3 + 2^2$

5	4	3	2	1	0
	1	1	1	0	0

$(.6875)_{10} = (10110)_2$

$0.5 + 0.1875$

$0.5 + 0.125 + 0.0625$

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

-1	-2	-3	-4	-5
1	0	1	1	0

*Note: $(0.4075)_8$ octal system. لازم المنازل من (0 ← 7)

نقرّب ونكتبها كالتالي ...

$(0.410)_8$

* لأن الـ (7) لا تصبح (8) لأنه نظام ثماني لذلك نضيف (1) للمنزلة التالية ونضعها (صفر) .

Example: Convert 46.6875_{10} To Base 2 Using Alternative Method

- Convert 46 to Base 2:

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

Subtract	Remainder	Power
46-32=2^5	14	5
14-8=2^3	6	3
6-4=2^2	2	2
2-2=2^1	0	1

- Convert 0.6875 to Base 2:

$$(0.6875)_{10} = (0.1011)_2$$

Subtract	Remainder	Power
0.6875-0.500	0.1875	-1
0.1875-0.125	0.0625	-3
0.0625-0.0625	0	-4

- Join the results together with the radix point:

$$(46.6875)_{10} = (101110.1011)_2$$

- Easier way to do it:

Power	6	5	4	3	2	1	0	.	-1	-2	-3	-4
	0	1	0	1	1	1	0	.	1	0	1	1

Scanned by CamScanner

Additional Issue - Fractional Part

- Note that in this conversion, the fractional part can become 0 as a result of the repeated multiplications

- In general, it may take many bits to get this to happen or it may never happen

- Example Problem: Convert 0.65_{10} to N_2
  - $0.65 = 0.1010011001001\ldots$ الدوري
  - The fractional part begins repeating every 4 steps yielding repeating 1001 forever!

- **Solution: Specify number of bits to right of radix point and *round* or *truncate* to this number**

Scanned by CamScanner

Checking the Conversion

للتأكيد :

- To convert back, sum the digits times their respective powers of r

- From the prior conversion of 46.6875_{10}

$$101110_2 = 1*32 + 0*16 + 1*8 + 1*4 + 1*2 + 0*1$$
$$= 32 + 8 + 4 + 2$$
$$= 46$$

$$0.1011_2 = 1/2 + 1/8 + 1/16$$
$$= 0.5000 + 0.1250 + 0.0625$$
$$= 0.6875$$

Scanned by CamScanner

Octal (Hexadecimal) to Binary and Back: Method1

- Octal (Hexadecimal) to Binary:
  1. Convert octal (hexadecimal) to decimal (Slide 23)
  2. Covert decimal to binary (Slide 24 or Slide 29)

Base (r) :
or
octal (r = 8) .
or
hexadecimal (r=16) .

↓ ↑

Decimal (r=10) .

↓ ↑

Binary (r=2) .

- Binary to Octal (Hexadecimal):
  1. Convert binary to decimal (Slide 23)
  2. Covert decimal to octal (hexadecimal) (Slide 24)

* r = 8 = 2^3 . 3-bits in binary system:

4 2 1
000 : 0
001 : 1
010 : 2
011 : 3
100 : 4
101 : 5
110 : 6
111 : 7

* r = 16 = 2^4 . 4-bits in binary system:

8 4 2 1
0000 : 0
0001 : 1
0010 : 2
0011 : 3
0100 : 4
0101 : 5
0110 : 6
0111 : 7
1000 : 8
1001 : 9
1010 : A
1011 : B
1100 : C
1101 : D
1110 : E
1111 : F

Scanned by CamScanner

Octal (Hexadecimal) to Binary and Back: Method2 (Easier)

octal (r=8) to hexadecimal (r=16). ⇔ binary (r=2)

- Octal (Hexadecimal) to Binary:
  - ***Restate*** the octal (hexadecimal) as three (four) binary digits starting at the radix point and going both ways
- Binary to Octal (Hexadecimal):
  - ***Group*** the binary digits into three (four) bit groups starting at the radix point and going both ways, padding with zeros as needed
  - Convert each group of three (four) bits to an octal (hexadecimal) digit

* ملاحظة :- إذا لم يكونا النظامين المطلوب التحويل بينهما حوّة للأساس (r) نقوم بالتحويل للنظام العشري (decimal) ومن ثم نحوّل للنظام المطلوب.

Octal	0	1	2	3	4	5	6	7
Binary	000	001	010	011	100	101	110	111

Hexadecimal	0	1	2	3	4	5	6	7
Binary	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal	8	9	A (10)	B (11)	C (12)	D (13)	E (14)	F (15)
Binary	1000	1001	1010	1011	1100	1101	1110	1111

Scanned by CamScanner

Examples

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

- من 2 الى 8 نتوقف 3 منازل
- من 2 الى 16 نتوقف 4 منازل

- $(3A6.C)_{16} = (0011\ 1010\ 0110\ .\ 1100)_2$

- $(101100011011.111100000001)_2 = (\ ?\ )_8$

* نبدأ التقسيم من الفاصلة ذهاباً لليمين و اليسار

$(10/110/001/101/011.111/100/000/1)_2 = (26153.7404\ )_8$

- $(101100011011.111100000001)_2 = (\ ?\ )_{16}$

$(10/1100/0110/1011.1111/0000/01)_2 = (2C6B.F04\ )_{16}$

Scanned by CamScanner

Octal to Hexadecimal via Binary

- Convert octal to binary
- Use groups of four bits and convert to hexadecimal digits
- Example: Octal to Binary to Hexadecimal

$(635.177)_8$

octal: $(110\ 011\ 101\ .\ 001\ 111\ 111)_2$

binary: $(0001/1001/1101\ .\ 0011/1111/1000)_2$

hexadecimal: $(19D.3F8)_{16}$

Scanned by CamScanner

One last Conversion Example

- Given that $(365)_r = (194)_{10}$, compute the value of r?

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

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

$$3r^2 + 6r - 189 = 0$$

$$r^2 + 2r - 63 = 0$$

* عبارة تربيعية ---- تحلّل

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

* نأخذ الـ (Radix) الموجب فقط و نهمل السالب.

$$r = 7$$

Scanned by CamScanner

Binary Numbers and Binary Coding

⊛ نلاحظ : دائمًا كتابة الـ (codes) باستخدام الـ (binary system).

- Flexibility of representation
  - Within constraints below, can assign any binary combination (called a *code word*) to any data as long as data is uniquely encoded

* لمعرفة عدد الـ (codes) اللازمة :

- Information Types

n-bits . ⟶ 2^n . (using binary system, r=2)

  - *Numeric* . رقميّة / . عدديّة
    - Must represent range of data needed
    - Very desirable to represent data such that simple, straightforward computation for common arithmetic operations permitted
    - Tight relation to binary numbers
  - *Non-numeric* . غير عدديّة / . غير رقميّة
    - Greater flexibility since arithmetic operations not applied
    - Not tied to binary numbers

Scanned by CamScanner

Non-numeric Binary Codes

- Given *n* binary digits (called bits), a binary code is a mapping from a set of represented elements to a subset of the 2^n binary numbers.

عدد الـ digits

[n-bits. → 2^n]

* لمعرفة عدد الـ (Codes) اللازمة :

اذا اي اساس آخر (عن النظام العشري)

- Example: A binary code for the seven colors of the rainbow
- Code 100 is not used * متاحة ولكن لم تستخدم عادي

* عندي ≠ خيارات
يختار اي رقم لدي ايام عادي

Color	Binary Number
Red	000
Orange	001
Yellow	010
Green	011
Blue	101
Indigo	110
Violet	111

Scanned by CamScanner

Number of Bits Required

base = 2. (binary)
n = عدد الـ digits = 2.
M = عدد الـ elements = عدد الـ codes

(Number of elements).

- Given M elements to be represented by a binary code, the minimum number of bits, n, needed, satisfies the following relationships:

عدد الحالات

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

if n=3 . $2^3 \geq M > 2^2$
(8 ≥ 7 > 4).

$n = \lceil \log_2 M \rceil$, where $\lceil x \rceil$ is called the *ceiling function*, is the integer greater than or equal to x.

digits عدد

تقريب الجواب لأعلى رقم صحيح أكبر منه

عدد البت المطلوبة:

- Example: How many bits are required to represent decimal digits with a binary code?

radix = 2

عدد الـ digits للنظام العشري

$M = 10$

$n = \lceil \log_2 10 \rceil = \lceil 3.33 \rceil = 4$

تقريب للعدد الأكبر

* if: r=2 . n=7.
M (maximum possible number of elements) $= 2^7 = 128$.

* also: r = ⊛
$M = ⊛^n$. (radix) تمثل الـ

Scanned by CamScanner

Number of Elements Represented

- Given n digits in radix r, there are r^n distinct elements that can be represented.
- But, you can represent m elements, $m \leq r^n$

Examples:

- You can represent 4 elements in radix $r = 2$ with $n = 2$ digits: (00, 01, 10, 11).
- You can represent 4 elements in radix $r = 2$ with $n = 4$ digits: (0001, 0010, 0100, 1000).
- This second code is called a "one hot" code.

1000
0100
0010
0001

كل مرة يكون العدد (1) في منزلة معينة والباقي أصفار

* if we have elements and we want to represent them using minimum number of digits of base "r", we use the form:

$n = \lceil \log_r M \rceil$.

Ex: $n = \lceil \log_8 7 \rceil = \lceil \log_8 8^{0-} \rceil =$

$\lceil 0- \rceil = 1$

* n: minimum number of bits to represent (M).

Ex: $n = \lceil \log_2 365 \rceil = \lceil \log_2 2^{8-} \rceil = \lceil 8- \rceil =$ 9

* يمكن أيضاً معرفة أكبر قيمة (value) نقدر نمثلها:

if $r = 10$, $n = 2$ } then $r^n - 1 \Rightarrow$

$r^{10} - 1 = 100 - 1 = 99$. the biggest value.

* دائماً نقرب لأكبر بعدد واحد. (+1)

© 2008 Pearson Education, Inc.

Scanned by CamScanner

DECIMAL CODES - Binary Codes for Decimal Digits

(BCD).

- There are over 8,000 ways that you can chose 10 elements from the 16 binary numbers of 4 bits. A few are useful.

Decimal	8, 4, 2, 1	Excess 3	8, 4, -2, -1	Gray
0	0000	0011	0000	0000
1	0001	0100	0111	0001
2	0010	0101	0110	0011
3	0011	0110	0101	0010
4	0100	0111	0100	0110
5	0101	1000	1011	1110
6	0110	1001	1010	1010
7	0111	1010	1001	1011
8	1000	1011	1000	1001
9	1001	1100	1111	1000

(6) codes will be unused or available.

0000 ← 0
0001 ← 1
0010 ← 2
0011 ← 3
⋮
1001 ← 9

these are used codes but codes from (1010 → 1111) will not be used.

* Ex: 1 : 0 1 1 1 : 4-3 = 1
... وهكذا ... 9 : 1 1 1 1 : 12-3 = 9

Binary coded decimal

Scanned by CamScanner

Binary Coded Decimal (BCD)

- Numeric code
- The BCD code is the 8, 4, 2, 1 code
- 8, 4, 2, and 1 are weights → BCD is a *weighted* code
- This code is the simplest, most intuitive binary code for decimal digits and uses the same powers of 2 as a binary number, ***but only encodes the first ten values from 0 to 9***
- Example: 1001 (9) = 1000 (8) + 0001 (1)
- How many "invalid" code words are there?
  - Answer: 6
- What are the "invalid" code words?
  - Answer: 1010, 1011, 1100, 1101, 1110, 1111

Scanned by CamScanner

Warning: Conversion or Coding?

التحويل. و الترميز

- Do NOT mix up ***conversion*** of a decimal number to a binary number with ***coding*** a decimal number with a BINARY CODE.

- $13_{10} = 1101_2$ (This is **conversion**) تحويل بين الأنظمة

8 4 2 1
12+1 = (13) كقيمة.

- $13 \Leftrightarrow 0001|0011$ (This is **coding**) ترميز.

(BCD)
Decimal فيستخدم عادة لما يكون العدد

Excess 3

8 4 2 1 | 8 4 2 1
(1) (3)
كأعداد

Excess 3 Code and 8, 4, –2, –1 Code

- What interesting property is common to these two codes?
  - Answer: Both codes have the property that the codes for 0 and 9, 1 and 8, etc. can be obtained from each other by replacing the 0's with the 1's and vice-versa. Such a code is sometimes called a *complement code*.

Decimal	Excess 3	8, 4, –2, –1
0	0011	0000
1	0100	0111
2	0101	0110
3	0110	0101
4	0111	0100
5	1000	1011
6	1001	1010
7	1010	1001
8	1011	1000
9	1100	1111

Scanned by CamScanner

ALPHANUMERIC CODES - ASCII Character Codes

- Non-numeric code

⊛ ASCII code:-

if: $n=7$, $r=2$ → then: maximum number of elements $= 2^7 = 128$.

7 bits

- ASCII stands for American Standard Code for Information Interchange (Refer to Table 1-5 in the text)

- This code is a popular code used to represent information sent as character-based data. It uses 7-bits (i.e. 128 characters) to represent: $2^7 = 128$.
  - 95 Graphic printing characters
  - 33 Non-printing characters

Scanned by CamScanner

ASCII Code Table (c++)

معلومات

Least Significant

ASCII Code Chart

Most Significant

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

small

$W = (77)_{16} = (0111\ 0111)_2$

capital

$W = (57)_{16} = (0101\ 0111)_2$.

row صف — column عمود — الاختلاف

الاختلاف بنفس الـ (bit)

$A = (41)_{16} = (0100\ 0001)_2$.

$a = (61)_{16} = (0110\ 0001)_2$

الاختلاف

Scanned by CamScanner

ASCII Character Codes

- Graphic printing characters
  - 26 upper case letters (A-Z) Capital
  - 26 lower case letters (a-z) Small
  - 10 numerals (0-9)
  - 33 special characters (e.g. %, @, $) signs.

- Non-printing characters
  - Format effectors: used for text format (e.g. BS = Backspace, CR = carriage return)
  - Information separators: used to separate the data into paragraphs and pages (e.g. RS = record separator, FS = file separator)
  - Communication control characters (e.g. STX and ETX start and end text areas).

Scanned by CamScanner

ASCII Properties

- ASCII has some interesting properties:
  - Digits 0 to 9 span Hexadecimal values 30_{16} to 39_{16}
  - Upper case A-Z span 41_{16} to $5A_{16}$
  - Lower case a-z span 61_{16} to $7A_{16}$
  - Lower to upper case translation (and vice versa) occurs by flipping bit 6 (0 1 0 0 0 1 1 1)

Scanned by CamScanner

UNICODE

- UNICODE extends (توسيع) ASCII to 65,536 universal (عالمي) characters codes:
  - Non-numeric (غير رقمي)
  - For encoding characters in world languages
  - Available in many modern applications
  - 2 byte (16-bit) code words

* if: $n = 16$.

$r = 2$.

$2^{16} = 65,000$.

Scanned by CamScanner

PARITY BIT Error-Detection Codes

↳ (ECC) ↲

* تبحث عن الأخطاء الفردية فقط و تجد الخطأ فقط لا تقوم بتصليحه. (تحدّد مكان الخطأ فقط)

detect odd number of errors.
(التحدّد)

- Non-numeric
- ***Redundancy*** (e.g. extra information), in the form of extra bits, can be incorporated into binary code words to detect and correct errors

- A simple form of redundancy is ***parity***, an extra bit appended onto the code word to make the number of 1's odd or even. Parity can detect all single-bit errors and some multiple-bit errors
- A code word has ***even parity*** if the number of 1's in the code word is even
- A code word has ***odd parity*** if the number of 1's in the code word is odd

Scanned by CamScanner

4-Bit Parity Code Example

- Fill in the even and odd parity bits:

Even Parity Message	Odd Parity Message
0000 ← parity bit.	0001
0011	0010
0101	0100
0110	0111
1001	1000
1010	1011
1100	1101
1111	1110

* if we have: 0011
parity bit نضيف
parity=0 → even parity → 00110 (عدد الواحدات زوجي لذلك parity زوجية)
parity=1 → odd parity → 00111 (عدد الواحدات فردي لذلك parity فردية)

*Ex: 0001
even parity → 00011 (لكي يصبح عدد (1) زوجي.)
odd parity → 00010 (لكي يبقى عدد (1) فردي.)

- The code word "1111" has even parity and the code word "1110" has odd parity. Both can be used to represent the same 3-bit data

* Note to know:
write the gray codes for:
0 → 00
1 → 01
2 → 11
3 → 10
يجب اختلاف بين بت وبت واحد فقط بين كل code و الأخرى

Scanned by CamScanner

Logic and Computer Design Fundamentals

Chapter 2 – Combinational Logic Circuits

Part 1 – Gate Circuits and Boolean Equations

Charles Kime & Thomas Kaminski

© 2008 Pearson Education, Inc.

(Hyperlinks are active in View Show mode)

Scanned by CamScanner

Combinational Logic Circuits

- Digital (logic) circuits are hardware components that manipulate binary information.

- Integrated circuits: transistors and interconnections.
  - Basic circuits is referred to as ***logic gates***
  - The outputs of gates are applied to the inputs of other gates to form a digital circuit

- Combinational? Later…

© 2008 Pearson Education, Inc.

Scanned by CamScanner

Overview

- **Part 1 – Gate Circuits and Boolean Equations**
  - Binary Logic and Gates
  - Boolean Algebra
  - Standard Forms
- **Part 2 – Circuit Optimization**
  - Two-Level Optimization
  - Map Manipulation
  - Practical Optimization (Espresso)
  - Multi-Level Circuit Optimization
- ***Part 3 – Additional Gates and Circuits***
  - *Other Gate Types*
  - *Exclusive-OR Operator and Gates*
  - *High-Impedance Outputs*

Scanned by CamScanner

Binary Logic and Gates

- ***Binary variables*** take on one of two values
- ***Logical operators*** operate on binary values and binary variables
- Basic logical operators are the logic functions ***AND***, ***OR*** and ***NOT***

  ⑧ Basic operator gates:
  ① AND : ·, ∧ ③ NOT : ∼ / ′ / −
  ② OR : +, ∨
- ***Logic gates*** implement logic functions
- ***Boolean Algebra***: a useful mathematical system for specifying and transforming logic functions
- We study Boolean algebra as a foundation for designing and analyzing digital systems!

© 2008 Pearson Education, Inc.

Scanned by CamScanner

Binary Variables

- Recall that the two binary values have different names:
  - True/False
  - On/Off
  - Yes/No
  - 1/0
- We use 1 and 0 to denote the two values
- Variable identifier examples:
  - A, B, y, z, or X_1 for now
  - *RESET*, START_IT, or ADD1 later

Scanned by CamScanner

Logical Operations

- The three basic logical operations are:
  - AND
  - OR
  - NOT

- AND is denoted by a dot (·) or (∧)
- OR is denoted by a plus (+) or (∨)
- NOT is denoted by an over-bar (¯), a single quote mark (') after, or (~) before the variable

Scanned by CamScanner

Scanned by CamScanner

Notation Examples

- Examples:
  - $Z = X \cdot Y = XY = X \wedge Y$: is read "Z is equal to X AND Y"
    - Z = 1 if and only if X = 1 and Y = 1; otherwise, Z = 0
  - $Z = X + Y = X \vee Y$: is read "Z is equal to X OR Y"
    - Z = 1 if (only X = 1) or if (only Y = 1) or if (X = 1 and Y = 1)
  - $Z = \bar{X} = X' = \sim X$: is read "Z is equal to NOT X"
    - Z = 1 if X = 0; otherwise, Z = 0
- Notice the difference between arithmetic addition and logical OR:
  - The statement:

    1 + 1 = 2 (read "one plus one equals two")

    is not the same as

    1 + 1 = 1 (read "1 or 1 equals 1")

Operator Definitions

- Operations are defined on the values "0" and "1" for each operator:

معرّف الاحتمالات.

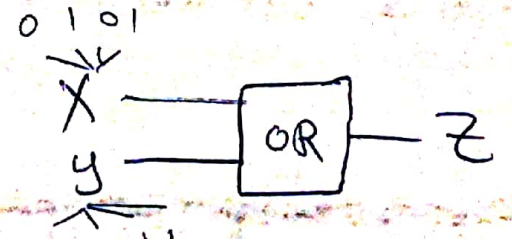
AND
0 . 0 = 0
0 . 1 = 0
1 . 0 = 0
1 . 1 = 1

OR
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1

NOT
$\overline{0} = 1$
$\overline{1} = 0$

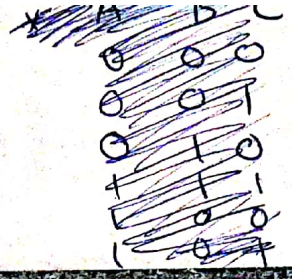
r = 2
n = 2 } → then $2^2 = 4$

احتمالات.



Scanned by CamScanner

Truth Tables



جدول الحقيقة

يشمل كل الاحتمالات.

- ***Truth table*** - a tabular listing of the values of a function for all possible combinations of values on its arguments

- Example: Truth tables for the basic logic operations:

AND		
Inputs		Output
X	Y	Z = X . Y
0	0	0
0	1	0
1	0	0
1	1	1

0 ← 1 ← 2 ← 3 ←

العد بالترتيب

عدد المدخلات 2

$2^2 = 4$ احتمالات

OR		
Inputs		Output
X	Y	Z = X + Y
0	0	0
0	1	1
1	0	1
1	1	1

$2^2 = 4$ احتمالات

NOT	
Inputs	Output
X	$Z = \bar{X}$
0	1
1	0

عدد المدخلات = 1

$2^1 = 2$ احتمال فقط.

most significant. ↑ MSbit

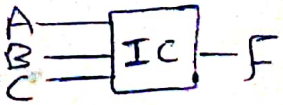
least significant ↑ LSbit

$2^3 = 8$

inputs:

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

Draw:-

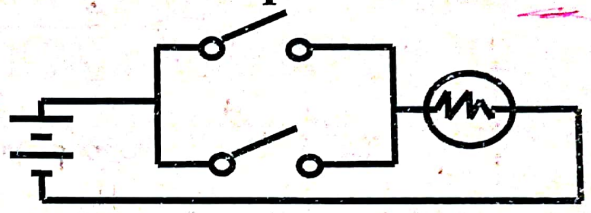


Scanned by CamScanner

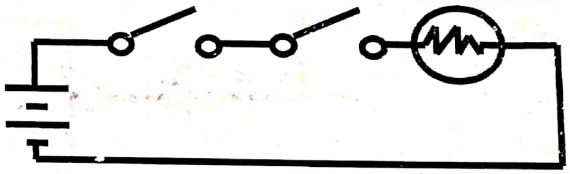
Logic Function Implementation

- Using Switches
  - For inputs:
    - logic 1 is switch closed
    - logic 0 is switch open
  - For outputs:
    - logic 1 is light on
    - logic 0 is light off
  - NOT uses a switch such that:
    - logic 1 is switch open
    - logic 0 is switch closed

Switches in parallel => OR

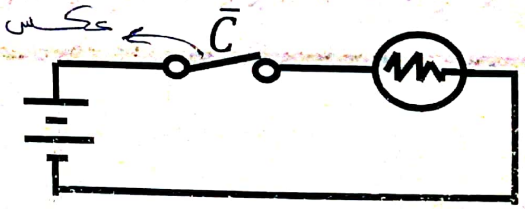


Switches in series => AND



Normally-closed switch => NO

عكس القيمة.



* عندما يكون المفتاح مغلق يعني الدارة (0)
* عندما يكون المفتاح مفتوح يعني الدارة (1)

(0 ← 1)
(1 ← 0)

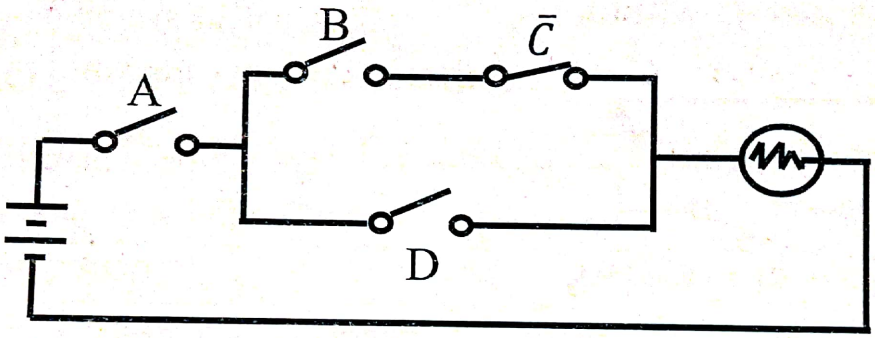
عكس الواقع، لأن البوابة (not)

© 2009 Pearson Education, Inc.

Scanned by CamScanner

Logic Function Implementation (Continued)

- Example: Logic Using Switches



- Light is *ON* (L = 1) for $L\,(A, B, C, D) = A\,.\,(B\bar{C} + D) = AB\bar{C} + AD$ and *OFF* (L = 0), otherwise.

- Useful model for relay circuits and for CMOS gate circuits, the foundation of current digital logic technology

Scanned by CamScanner

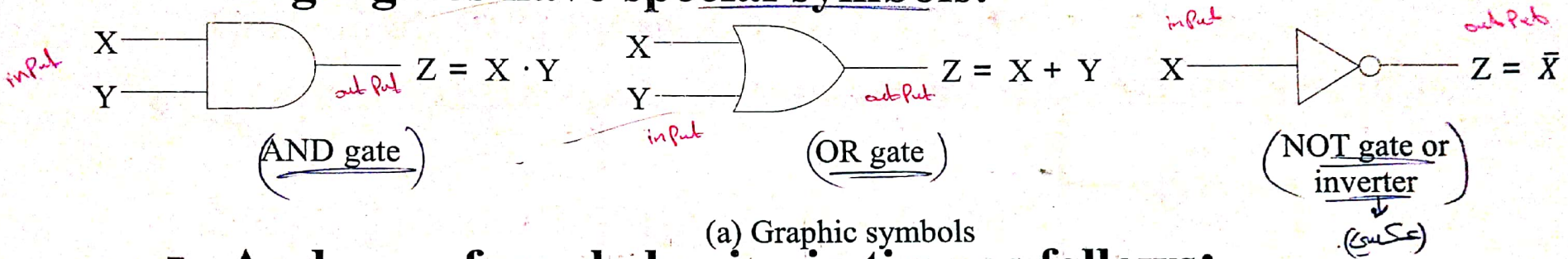
Logic Gates

- In the earliest computers, switches were opened and closed by magnetic fields produced by energizing coils in *relays*. The switches in turn opened and closed the current paths
- Later, *vacuum tubes* that open and close current paths electronically replaced relays
- Today, *transistors* are used as electronic switches that open and close current paths
- Optional: Chapter 6 – Part 1: The Design Space

Scanned by CamScanner

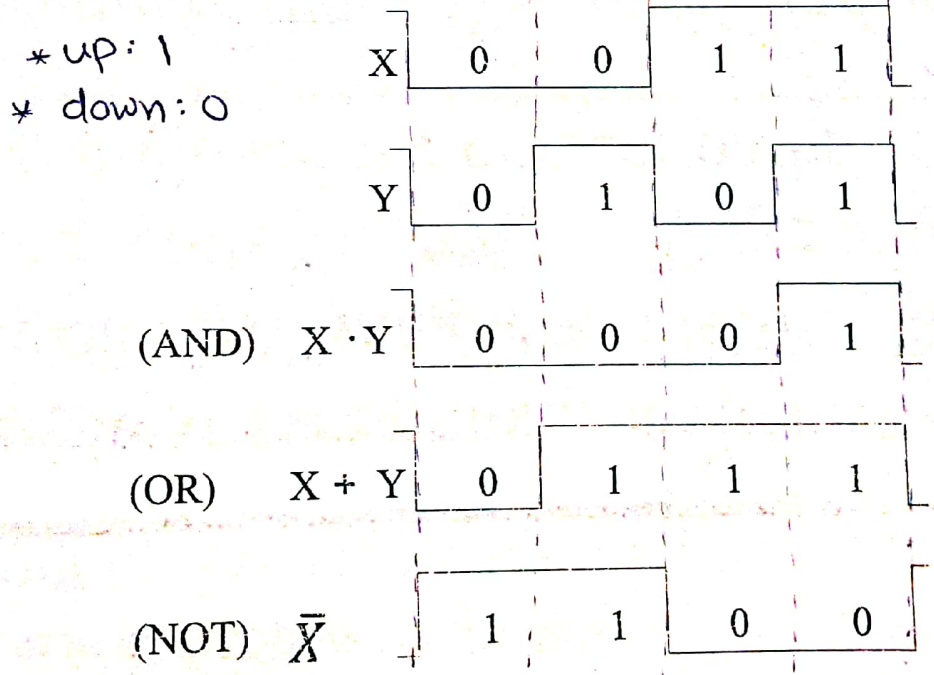
Logic Gate Symbols and Behavior

- **Logic gates have special symbols:** أشكال خاصة



(a) Graphic symbols

- **And waveform behavior in time as follows:**



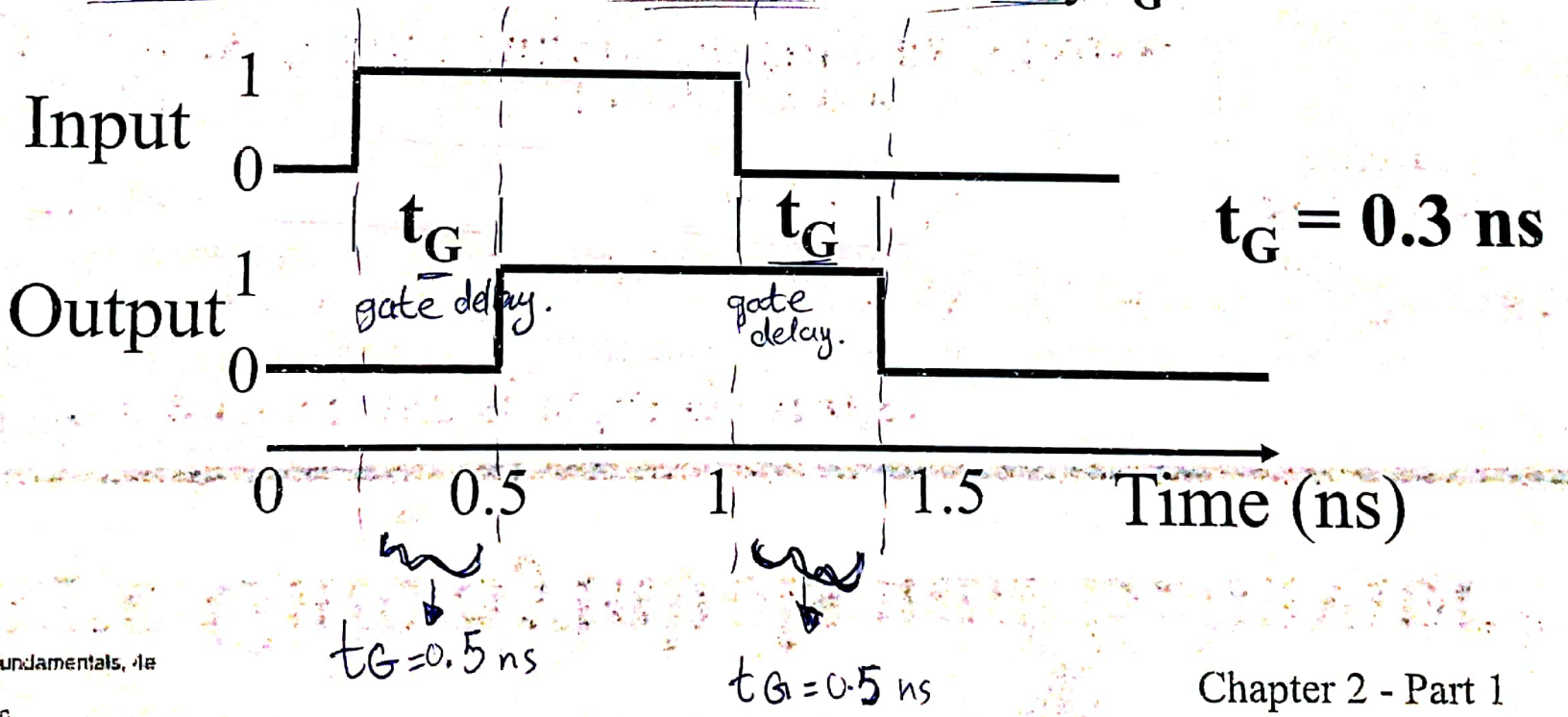
(b) Timing diagram

Scanned by CamScanner

Gate Delay (التأخير)

- In actual physical gates, if one or more input changes causes the output to change, the output change does not occur instantaneously (لحظيًا)

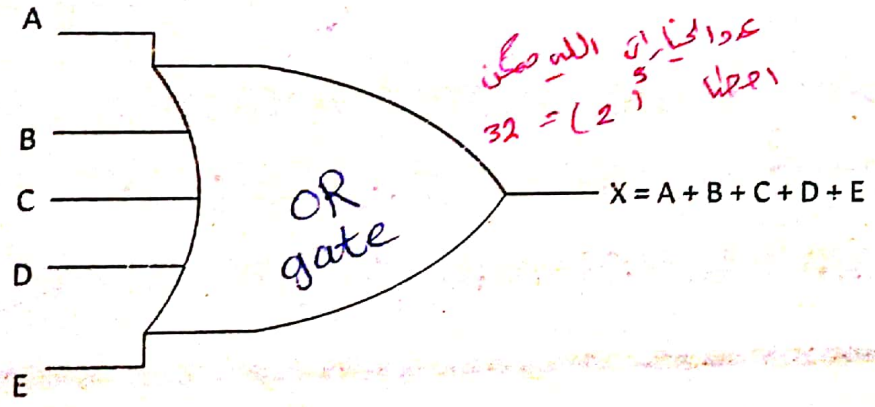
- The delay between an input change(s) and the resulting output change is the ***gate delay*** denoted by t_G:

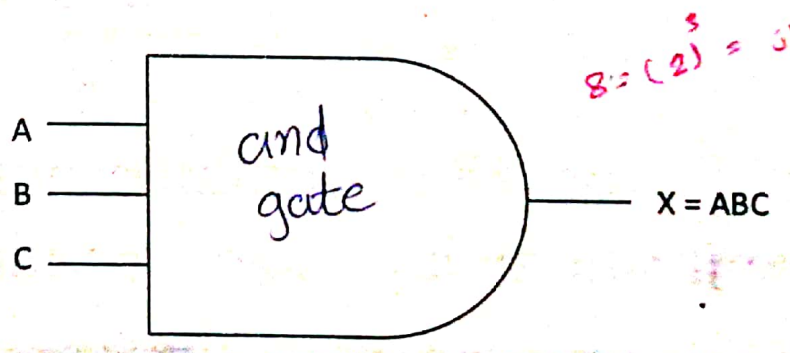


Scanned by CamScanner

Logic Gates: Inputs and Outputs

- NOT (inverter)
  - Always one input and one output
- AND and OR gates
  - Always one output
  - Two or more inputs





Scanned by CamScanner

Boolean Algebra

تعبير رياضي منطقي

- An algebra dealing with binary variables and logic operations
  - Variables are designated by letters of the alphabet
  - Basic logic operations: AND, OR, and NOT

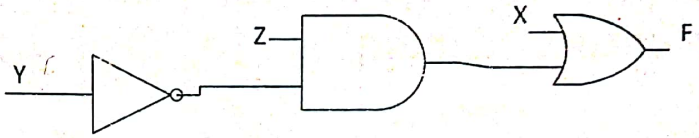
- ***A Boolean expression*** is an algebraic expression formed by using binary variables, constants 0 and 1, the logic operation symbols, and parentheses (أقواس)
  - E.g.: X . 1, A + B + C, (A + B)(C + D)

- ***A Boolean function*** consists of a binary variable identifying the function followed by equals sign and a Boolean expression
  - E.g.: $F = A + B + C$, $L(D, X, A) = DX + \bar{A}$

Scanned by CamScanner

Logic Diagrams and Expressions

1. Equation: $F = X + \bar{Y}Z$
2. Logic Diagram:



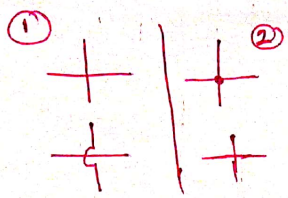
3. Truth Table:

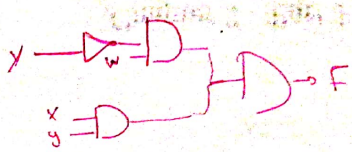
X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- Boolean equations, truth tables and logic diagrams describe the *same* function!
- Truth tables are *unique*; expressions and logic diagrams are not. This gives flexibility in implementing functions.

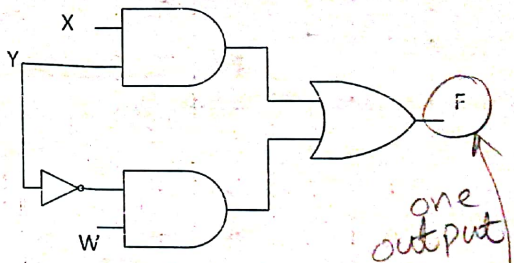
Scanned by CamScanner

Example





- Draw the logic diagram and the truth table of the following Boolean function: $F(W,X,Y) = XY + W\bar{Y}$
- Logic Diagram:



- Truth Table:

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

عدد المتغيرات ٣ = عدد الاحتمالات

$٢^{٣} = ٨ = \frac{٤}{٤} = \frac{٢}{٢}$ ..

نوزّع القيم بالأول ٤ ← أصفار. ٤ ← واحدات.

ثم $\frac{٤}{٢} = ٢$ ← صفر. ٢ ← واحد. وهكذا

- This example represents a ***Single Output Function***

Scanned by CamScanner

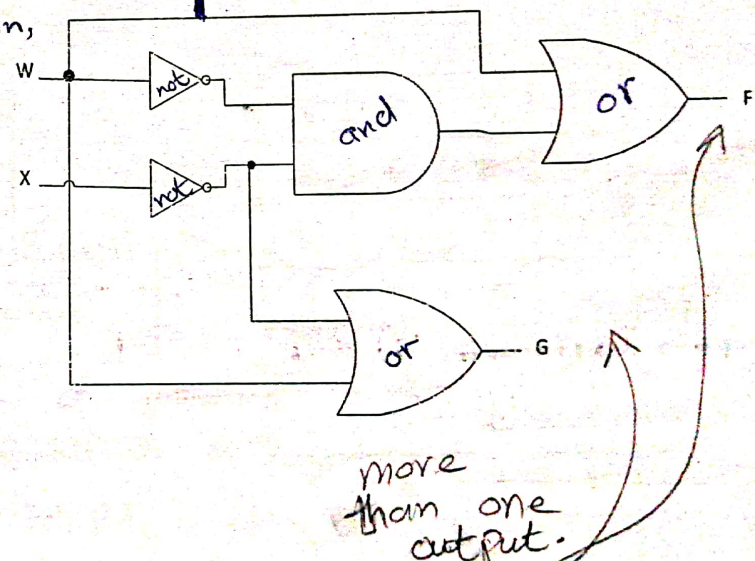
Scanned by CamScanner

Example

- Draw the logic diagram and the truth table of the following Boolean functions: $F(W,X) = \bar{W}\bar{X} + W$, $G(W,X) = W + \bar{X}$

as a function, of (w, y)

- Logic Diagram:



more than one output.

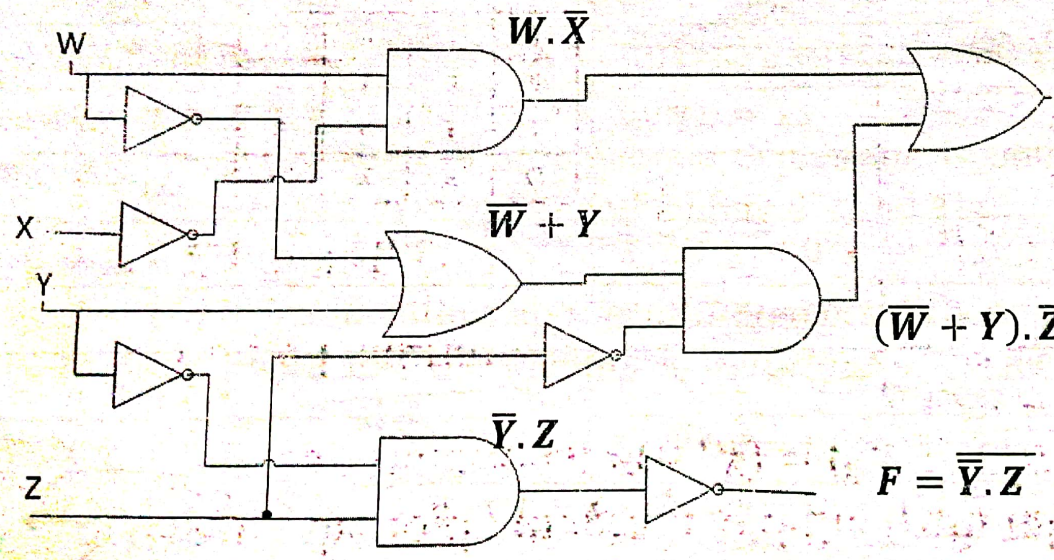
- Truth Table:

W	X	F	G
0	0	1	1
0	1	0	0
1	0	1	1
1	1	1	1

- This example represents a ***Multiple Output Function***

Example:

- Given the following logic diagram, write the corresponding Boolean equation:



- Logic circuits of this type are called combinational logic circuits since the variables are combined by logical operations.

Scanned by CamScanner

Basic Identities of Boolean Algebra

* The distributive law even though more than one variable

*Ex: $AB.(x+y+z) = AB.x + AB.y + AB.z$

* $0.0 = 0$, $0.1 = 0$, $(A+B+C).0 = 0$ — صفر النظم عنده

* Complement = not = bar = $\bar{x}$ (نفس الطرق)

* $X.1 = X$ (نفسه) , $0.1 = 0$, $1.1 = 1$ (نفسه)

1. $X + 0 = X$	2. $X \cdot 1 = X$	*Existence of 0 and 1*
3. $X + 1 = 1$	4. $X \cdot 0 = 0$ ($1.0=0$, $0.0=0$)	
5. $X + X = X$	6. $X \cdot X = X$	*Idempotence*
7. $X + \bar{X} = 1$	8. $X \cdot \bar{X} = 0$	*Existence of complement*
9. $\bar{\bar{X}} = X$		*Involution*
10. $X + Y = Y + X$	11. $XY = YX$	*Commutative Laws*
12. $(X + Y) + Z = X + (Y + Z)$	13. $(XY)Z = X(YZ)$	*Associative Laws*
14. $X(Y + Z) = XY + XZ$	15. $X + (YZ) = (X + Y)(X + Z)$	*Distributive Laws*
16. $\overline{X + Y} = \bar{X}\,\bar{Y}$	17. $\overline{XY} = \bar{X} + \bar{Y}$	*DeMorgan's Laws*

Enjoy Khaled Mawaydeh

قانون الترتيب للمتغيرات (تبادلي العملية)

ترتيب المتغيرات عند تساوي العملية بينهم لا يهم.

شرط هذا القانون أن تكون العملية بين المتغيرات نفسها

Even though more than two variables

*Ex: $\overline{x+y+z} = \bar{x}.\bar{y}.\bar{z}$

توزيع ما خارج القوس على القوس وعمل العملية نفسها بين المتغيرات.

توزّع الـ (not) وتعكس العملية (and ⟷ or).

* كل الاشارات اللي تحت (—) بتعكسه

Scanned by CamScanner

Some Properties of Identities & the Algebra

- If the meaning is unambiguous, we leave out the symbol "·"
  - * AB = A·B = A AND B
- The identities above are organized into pairs
  - The ***dual*** of an algebraic expression is obtained by interchanging (+) and (·) and interchanging **0**'s and **1**'s
  - The identities appear in ***dual*** pairs. When there is only one identity on a line the identity is ***self-dual***, i. e., the dual expression = the original expression.

#) $\overline{X+Y} = \bar{X} \cdot \bar{Y}$

* Note:- $X + Y \cdot Z = \bar{X} \cdot Y \cdot Z \rightarrow \bar{X} \cdot (\bar{Y} + \bar{Z})$

© 2008 Pearson Education, Inc.

Scanned by CamScanner

Some Properties of Identities & the Algebra (Continued)

- Unless it happens to be self-dual, the dual of an expression does not equal the expression itself
- Examples:
  - $F = (A + \bar{C}) \cdot B + 0$
    - $Dual\ F = [(A \cdot \bar{C}) + B] \cdot 1 = A \cdot \bar{C} + B$
  - $G = XY + (\overline{W + Z})$
    - $Dual\ G = (X + Y) \cdot \overline{W \cdot Z} = (X + Y) \cdot (\bar{W} + \bar{Z})$
  - $H = AB + AC + BC$
    - $Dual\ H = (A + B)(A + C)(B + C) = (A + BC)(B + C)$
      $= AB + AC + BC$
- Are any of these functions self-dual?
  - Yes, H is self-dual

X	Y	Z	G	Dual G
0	0	0	1	0
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Scanned by CamScanner

Boolean Operator Precedence

- The order of evaluation in a Boolean expression is: أولويات العمليات
  1. Parentheses الأقواس
  2. NOT بوابة not
  3. AND بوابة and
  4. OR بوابة or

- Consequence: Parentheses appear around OR expressions

* لكي يتم تنفيذه قبل and أو not حسب الحاجة. *

- Examples:
  - $F = A(B + C)(C + \bar{D})$
  - $F = \sim\bar{A}B = \bar{A}B$
  - $F = AB + C$
  - $F = A(B + C)$ $= A \cdot B + A \cdot C$

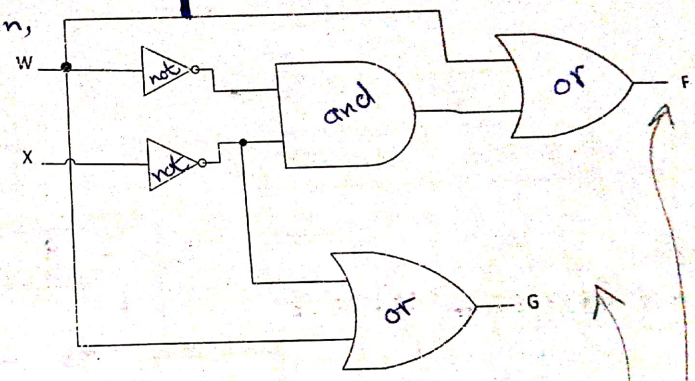
Scanned by CamScanner

Example

- Draw the logic diagram and the truth table of the following Boolean functions: $F(W,X) = \bar{W}\bar{X} + W$, $G(W,X) = W + \bar{X}$

as a function, of(w,y)

- Logic Diagram:



- Truth Table:

W	X	F	G
0	0	1	1
0	1	0	0
1	0	1	1
1	1	1	1

more than one output.

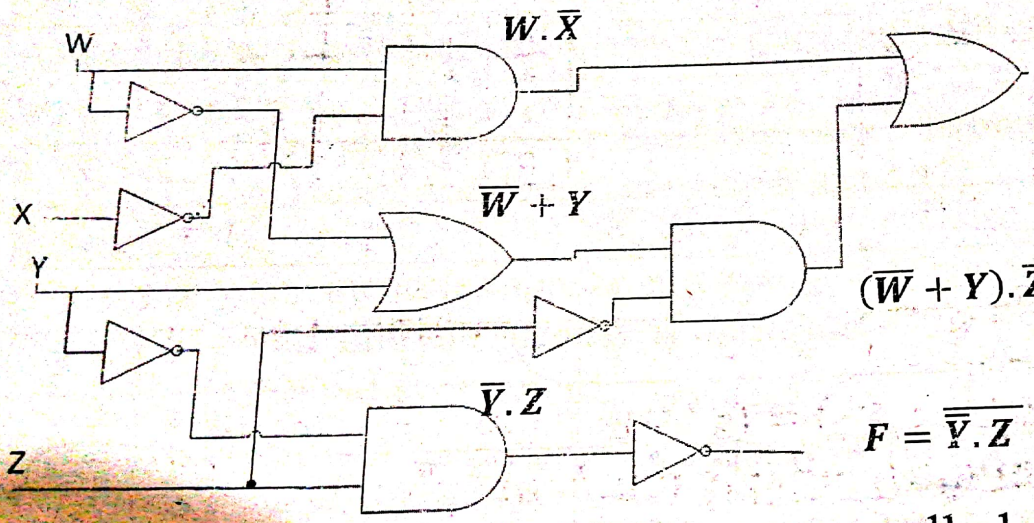
- This example represents a ***Multiple Output Function***

Scanned by CamScanner

Example:

- Given the following logic diagram, write the corresponding Boolean equation:

المعادلة المكافئة التي تمثل الشكل التالي.



- Logic circuits of this type are called combinational logic circuits since the variables are combined by logical operations

why ?

answer :-

Scanned by CamScanner

Basic Identities of Boolean Algebra

* The distributive law even though more than one variable
*Ex: AB.(x+y+z) = AB.X + AB.y + AB.Z

* 0.0 = 0
0.1 = 0
(A+B+C).0 = 0
بغض النظر عنه .

* complement = not = bar = $\bar{X}$. (نفس المعنى)

* X.1 = X , 0.1 = 0 , 1.1 = 1

1. $X + 0 = X$	2. $X . 1 = X$	Existence of 0 and 1
3. $X + 1 = 1$	4. $X . 0 = 0$ (1.0 = 0, 0.0 = 0)	
5. $X + X = X$	6. $X . X = X$	Idempotence
7. $X + \bar{X} = 1$	8. $X . \bar{X} = 0$	Existence of complement
9. $\bar{\bar{X}} = X$		Involution
10. $X + Y = Y + X$	11. $XY = YX$	Commutative Laws
12. $(X + Y) + Z = X + (Y + Z)$	13. $(XY)Z = X(YZ)$	Associative Laws
14. $X(Y + Z) = XY + XZ$	15. $X + (YZ) = (X + Y)(X + Z)$	Distributive Laws
16. $\overline{X + Y} = \bar{X} . \bar{Y}$	17. $\overline{X . Y} = \bar{X} + \bar{Y}$	DeMorgan's Laws

Shyam Khaled Mawaydeh

قانون الترتيب للمتغيرات (تبادل العملية).

ترتيب المتغيرات عند تساوي العملية بينهم لا يهم.

شرط هنا القانون أن تكون العملية بين المتغيرات نفسها

even though more than two variables
*Ex: $\overline{X+y+z} = \bar{X}.\bar{y}.\bar{z}$

توزيع ما خارج القوس على القوس وعمل العملية نفسها بين المتغيرات.

توزع الـ (not) وتعكس العملية (and ⇄ or).

(*) لحل الاختيار اللي تحت (—) نعكسه

Scanned by CamScanner

Some Properties of Identities & the Algebra

- If the meaning is unambiguous, we leave out the symbol "·"

  * AB = A.B = A AND B.

- The identities above are organized into pairs
  - The ***dual*** of an algebraic expression is obtained by interchanging (+) and (·) and interchanging **0**'s and **1**'s
  - The identities appear in ***dual*** pairs. When there is only one identity on a line the identity is ***self-dual***, i. e., the dual expression = the original expression.

عبارة صحيحة : $X + 0 = X$

عبارة صحيحة أخرى : $X \cdot 1 = X$

* Note:- $X + (Y \cdot Z) = \bar{X} \cdot Y \cdot Z$ ←

$\bar{X} \cdot (\bar{Y} + \bar{Z})$,

Scanned by CamScanner

Some Properties of Identities & the Algebra (Continued)

- Unless it happens to be self-dual, the dual of an expression does not equal the expression itself
- Examples:
  - $F = (A + \bar{C}) \cdot B + 0$
    - $Dual\ F = [(A \cdot \bar{C}) + B] \cdot 1 = A \cdot \bar{C} + B$
  - $G = XY + (\overline{W + Z})$
    - $Dual\ G = (X + Y) \cdot \overline{WZ} = (X + Y) \cdot (\bar{W} + \bar{Z})$
  - $H = AB + AC + BC$
    - $Dual\ H = (A + B)(A + C)(B + C) = (A + BC)(B + C)$
      $= AB + AC + BC$
- Are any of these functions self-dual?
  - Yes, H is self-dual

X	Y	Z	G	Dual G
0	0	0	1	0
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Scanned by CamScanner

Boolean Operator Precedence

- The order of evaluation in a Boolean expression is: أولويات العمليات.
  1. Parentheses الأقواس
  2. NOT بوابة not.
  3. AND بوابة and.
  4. OR بوابة or.

- Consequence: Parentheses appear around OR expressions

* لكي يتم تنفيذه قبل and أو not حسب الحاجة. *

- Examples:
  - $F = A(B + C)(C + \overline{D})$
  - $F = \sim AB = \overline{A}B$
  - $F = AB + C$
  - $F = A(B + C)$ $= A \cdot B + A \cdot C$

Scanned by CamScanner

Useful Boolean Theorems

$X + \bar{X}yz = (X + \bar{X})(X + yz)$

$X + \bar{X}y = X + y$ → simplification.

$X \cdot y + \bar{X} \cdot y = y$ → minimization.

$X + X \cdot y = X$ → Absorption

Theorem	Dual	Name
1. $x.y + \bar{x}.y = y$	$(x + y)(\bar{x} + y) = y$	Minimization
2. $x + x.y = x$	$x.(x + y) = x$	Absorption
3. $x + \bar{x}.y = x + y$	$x.(\bar{x} + y) = x.y$	Simplification
4. $x.y + \bar{x}.z + y.z = x.y + \bar{x}.z$		Consensus
$(x + y)(\bar{x} + z)(y + z) = (x + y)(\bar{x} + z)$		

$* F(A,B) = \sim A.B$ → not A ① , A and B ②

بينما $* F(A,B) = \sim(A.B)$ → A and B ① , not(A and B) ②

Scanned by CamScanner

Example 1: Boolean Algebraic Proof

- $A + A \cdot B = A$ (Absorption Theorem)

عاملاً (A) فنأخذ مشتركًا.

كأنها (A·1)

$A + A \cdot B$		dual:-
$= A \cdot 1 + A \cdot B$	$X = X \cdot 1$	$A \cdot (A+B) = A$
$= A \cdot (1 + B)$	*Distributive Law*	$A \cdot A + A \cdot B$
$= A \cdot 1$	$1 + X = 1$	$A + AB$
$= A$	$X \cdot 1 = X$	$A(1+B)$
	كأنها متغير واحد	$A \cdot 1 = A$

*even though: $X + X \cdot (ABCDE) = X$

- Our primary reason for doing proofs is to learn:
  - Careful and efficient use of the identities and theorems of Boolean algebra
  - How to choose the appropriate identity or theorem to apply to make forward progress, irrespective of the application

Scanned by CamScanner

Example 2: Boolean Algebraic Proofs

even though more than one variables- $X.yz + \bar{X}.w + wyz = xyz + \bar{X}.w$.

- $AB + \bar{A}C + BC = AB + \bar{A}C$ (Consensus Theorem)

كأن (BC) غير موجودة أي لا تؤثر.

* قاعدة: $A + \bar{A} = 1$ دائماً.

$AB + \bar{A}C + BC$	كأن BC هي (BC.1)
$= AB + \bar{A}C + 1.BC$	نضرب ×1 — $1.X = X$
$= AB + \bar{A}C + (A + \bar{A}).BC$	$X + \bar{X} = 1$
$= AB + \bar{A}C + ABC + \bar{A}BC$	*Distributive Law* توزيع القوس:
$= AB + ABC + \bar{A}C + \bar{A}BC$	*Commutative Law* ترتيب الحدود
$= AB.1 + AB.C + \bar{A}C.1 + \bar{A}C.B$	$X.1 = X$ *and Commutative Law*
$= AB(1 + C) + \bar{A}C(1 + B)$	*Distributive Law* إخراج عامل مشترك
$= AB.1 + \bar{A}C.1$	$1 + X = 1$
$= AB + \bar{A}C$	$X.1 = X$

Scanned by CamScanner

Proof of Simplification

- $A + \bar{A}.B = A + B$ (Simplification Theorem)

$A + \bar{A}.B$:	توزيع على العوس.
$= (A + \bar{A})(A + B)$	*Distributive Law*
$= 1.(A + B)$	$X + \bar{X} = 1$
$= A + B$	$X.1 = X$

- $A.(\bar{A} + B) = AB$ (Simplification Theorem)

$A.(\bar{A} + B)$	
$= (A.\bar{A}) + (A.B)$	*Distributive Law*
$= 0 + AB$	$X.\bar{X} = 0$
$= AB$	$X + 0 = X$

Scanned by CamScanner

Proof of Minimization

البرهان

- $A.B + \bar{A}.B = B$ (Minimization Theorem) نظرية الاختزال (التقليص).

$A.B + \bar{A}.B$	B: عامل مشترك.
$= B(A + \bar{A})$	*Distributive Law*
$= B.1$	$X + \bar{X} = 1$
$= B$	$X.1 = X$

* حقيقة: $A + \bar{A} = 1$

* حقيقة: $A \cdot \bar{A} = 0$.

- $(A + B).(\bar{A} + B) = B$ (Minimization Theorem)

$(A + B)(\bar{A} + B)$	B: عامل مشترك
$= B + (A.\bar{A})$	*Distributive Law*
$= B + 0$	$X.\bar{X} = 0$
$= B$	$X + 0 = X$

Scanned by CamScanner

Proof of DeMorgan's Laws (1)

- $\overline{X+Y} = \bar{X}.\bar{Y}$ (DeMorgan's Law)
  - We will show that, $\bar{X}.\bar{Y}$, satisfies the definition of the complement of $(X+Y)$, defined as $\overline{X+Y}$ by DeMorgan's Law.
  - To show this, we need to show that $A + A' = 1$ and $A.A' = 0$ with $A = X + Y$ and $A' = X'.Y'$. This proves that $X'.Y' = \overline{X+Y}$.

- Part 1: Show $X + Y + X'.Y' = 1$

$(X+Y) + X'.Y'$	
$= (X + Y + X')(X + Y + Y')$	*Distributive Law*
$= (1 + Y)(X + 1)$	$X + \bar{X} = 1$
$= 1.1$	$X + 1 = 1$
$= 1$	$X.1 = X$

Scanned by CamScanner

Proof of DeMorgan's Laws (2)

- Part 2: Show $(X + Y).X'.Y' = 0$

$(X + Y).X'.Y'$	
$= (X.X'.Y') + (Y.X'.Y')$	*Distributive Law*
$= (0.Y') + (X'.0)$	$X.\bar{X} = 0$
$= 0 + 0$	$X.0 = 0$
$= 0$	$\bar{X} + 0 = X$

- Based on the above two parts, $X'.Y' = \overline{X + Y}$
- The second DeMorgans' law is proved by duality
- Note that DeMorgan's law, given as an identity is not an axiom in the sense that it can be proved using the other identities.

Scanned by CamScanner

Example 3: Boolean Algebraic Proofs

- $\overline{(X+Y)}Z + X\bar{Y} = \bar{Y}(X+Z)$

$\overline{(X+Y)}Z + X\bar{Y}$	
$= X'.Y'.Z + X.Y'$	*DeMorgan's law* (not) توزيع
$= Y'(X'Z + X)$	*Distributive law* عامل مشترك
$= Y'(X + X'Z)$	*Commutative law* تبديل الحدود
$= Y'(X + Z)$	*Simplification Theorem* $A + \bar{A}.B = A+B$.

Scanned by CamScanner

Boolean Function Evaluation

- $F_1 = xy\bar{z}$
- $F_2 = x + \bar{y}z$
- $F_3 = \bar{x}\bar{y}\bar{z} + \bar{x}yz + x\bar{y}$
- $F_4 = x\bar{y} + \bar{x}z$

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	1	0
0	0	1	0	1	0	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

* في مثل هذه الأسئلة ، نحدّد متى يكون قيمة الناتج النهائي (١) أو (٥) حسب العبارة وحالة المسألة أفضل من حل جميع العبارات لجميع القيم وذلك للسرعة و لتسهيل الحلّ.

Scanned by CamScanner

Expression Simplification

- An application of Boolean algebra
- Simplify to contain the smallest number of literals (complemented and uncomplemented variables). have (not) gate.
- Example: Simplify the following Boolean expression
  - $AB + A'CD + A'BD + A'CD' + ABCD$

$AB + A'CD + A'BD + A'CD' + ABCD$	
$= AB + ABCD + A'CD + A'CD' + A'BD$	*Commutative law*
$= AB(1 + CD) + A'C(D + D') + A'BD$	*Distributive law*
$= AB.1 + A'C.1 + A'BD$	$1 + X = 1$ *and* $X + X' = 1$
$= AB + A'C + A'BD$	$X.1 = X$
$= AB + A'BD + A'C$	*Commutative law*
$= B(A + A'D) + A'C$	*Distributive law*
$= B(A + D) + A'C \rightarrow$ *5 Literals*	*Simplification Theorem*

Scanned by CamScanner

Complementing Functions (not) وضع نفي المتغير

- Use DeMorgan's Theorem to complement a function:
  1. Interchange AND and OR operators
  2. Complement each constant value and literal

- Example: Complement $F = x'yz' + xy'z'$

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

- Example: Complement $G = (a' + bc)d' + e$

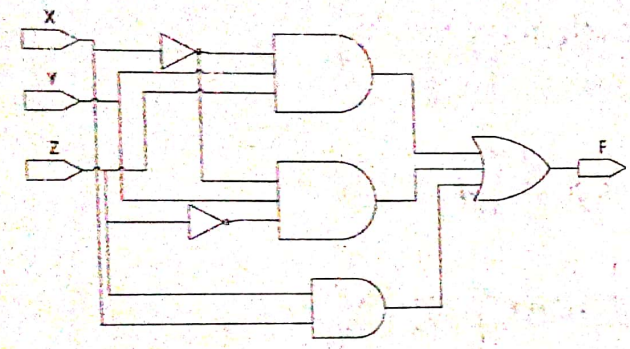
$$G' = (a(b' + c') + d).e'$$

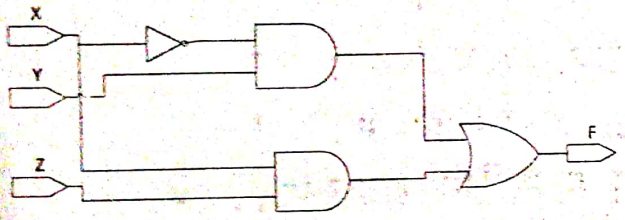
Scanned by CamScanner

Example

- Simplify the following:
  - $F = X'YZ + X'YZ' + XZ$

$X'YZ + X'YZ' + XZ$	
$= X'Y(Z + Z') + XZ$	*Distributive law*
$= X'Y.1 + XZ$	$X + X' = 1$
$= X'Y + XZ$	$X.1 = X$





x	y	z	$X'YZ + X'YZ' + XZ$	$X'Y + XZ$
0	0	0	0	0
0	0	1	0	0
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1
			3 terms and 8 literals	2 terms and 4 literals

Scanned by CamScanner

Example

- Show that $F = x'y' + xy' + x'y + xy = 1$
  - Solution1: Truth Table

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

* Note: عند وجود جميع احتمالات المتغير خلال الاقتران هذا يعني بأن الجواب دائماً (1).
الاحتمالات جميعها هي:
$\bar{X}\bar{Y}$
$\bar{X}Y$
$X\bar{Y}$
XY

  - Solution2: Boolean Algebra

$x'y' + xy' + x'y + xy$	
$= y'(x' + x) + y(x' + x)$	*Distributive law*
$= y'.1 + y.1$	$X + X' = 1$
$= y' + y$	$X.1 = X$
$= 1$	$X + X' = 1$

Scanned by CamScanner

Examples

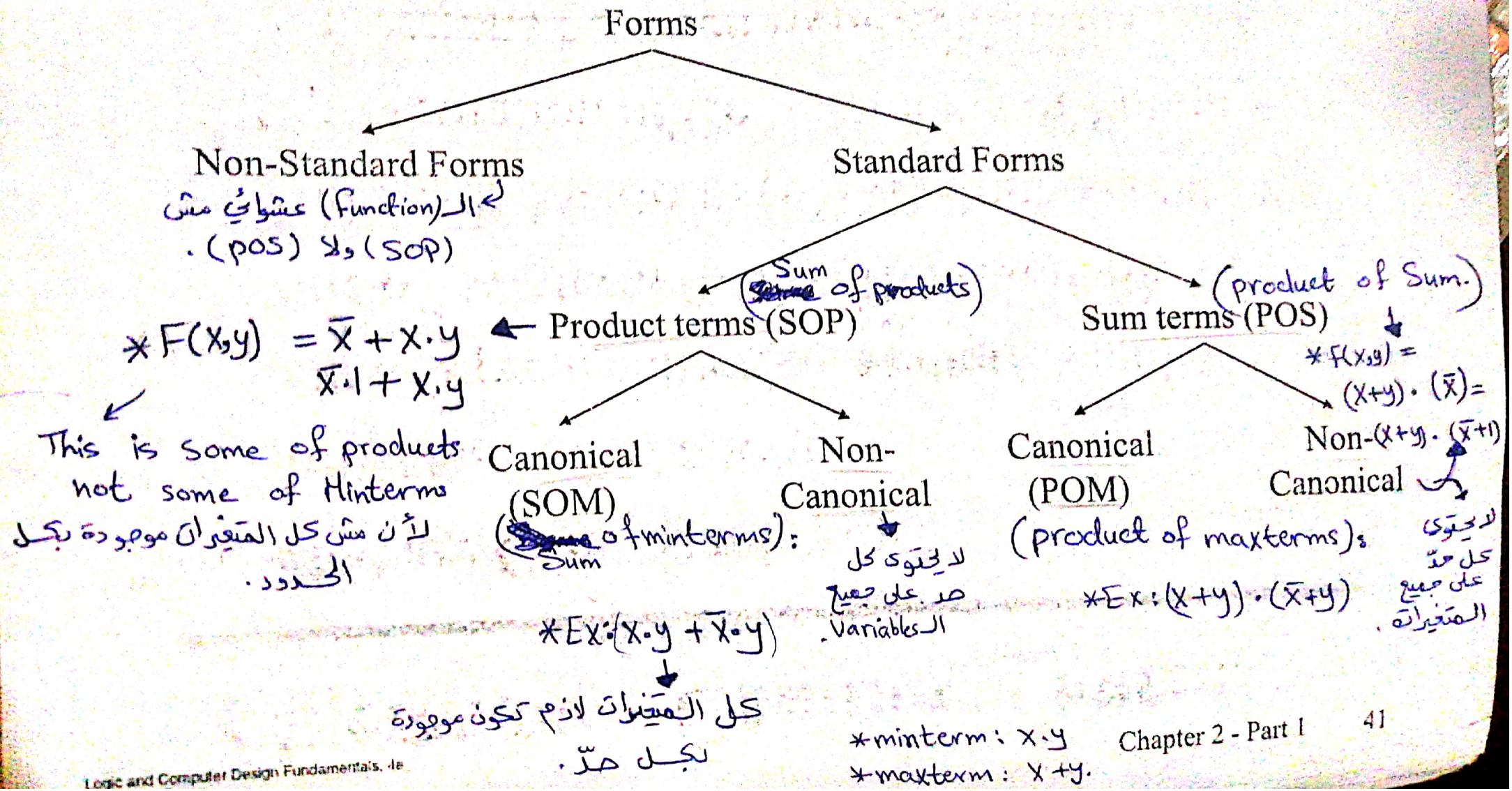
- Show that $ABC + A'C' + AC' = AB + C'$ using Boolean algebra.

$ABC + A'C' + AC'$	
$= ABC + C'(A' + A)$	*Distributive law*
$= ABC + C'.1$	$X + X' = 1$
$= ABC + C'$	$X.1 = X$
$= (AB + C')(C + C')$	*Distributive law*
$= (AB + C').1$	$X + X' = 1$
$= AB + C'$	$X.1 = X$

- Find the dual and the complement of $f = wx + y'z.0 + w'z$
  - $Dual(f) = (w + x)(y' + z + 1)(w' + z)$
  - $f' = (w' + x')(y + z' + 1)(w + z')$

Scanned by CamScanner

Boolean Representation Forms



Scanned by CamScanner

Canonical Forms

- It is useful to specify Boolean functions in a form that:
  - Allows comparison for equality
  - Has a correspondence to the truth tables
  - Facilitates simplification
- Canonical Forms in common usage:
  - Sum of Minterms (SOM) $xy + x\bar{y} + \bar{x}y \cdots$
  - Product of Maxterms (POM) $(x+y)\cdot(\bar{x}+\bar{y})\cdots$

Scanned by CamScanner

Minterms

- ***Minterms*** are AND terms with ***every variable*** present in either true or complemented form
  (X) إيجاب (X̄) منفية
- Given that each binary variable may appear normal (e.g., x) or complemented (e.g., $\bar{x}$), there are 2^n minterms for n variables
- Example: Two variables (X and Y) produce $2^2 = 4$ combinations:
  1. $X.Y$ (both normal)
  2. $X.\bar{Y}$ (X normal, Y complemented)
  3. $\bar{X}.Y$ (X complemented, Y normal)
  4. $\bar{X}.\bar{Y}$ (both complemented)

	x	y	z	minterm
0	0	0	0	$\bar{x}\bar{y}\bar{z}$
1	0	0	1	$\bar{x}\bar{y}z$
2	0	1	0	$\bar{x}y\bar{z}$
3	0	1	1	$\bar{x}yz$

- Thus there are ***four minterms*** of two variables

$$f(x,y) = x.y + x.\bar{y} + \bar{x}.y + \bar{x}.\bar{y}.$$ (some of minterms).

Scanned by CamScanner

Maxterms

- ***Maxterms*** are OR terms with ***every variable*** in true or complemented form
- Given that each binary variable may appear normal (e.g., x) or complemented (e.g., $\bar{x}$), there are 2^n maxterms for n variables
- Example: Two variables (X and Y) produce $2^2 = 4$ combinations:

① $X + Y$ (both normal)

② $X + \bar{Y}$ (X normal, Y complemented)

③ $\bar{X} + Y$ (X complemented, Y normal)

④ $\bar{X} + \bar{Y}$ (both complemented)

Scanned by CamScanner

Maxterms and Minterms

- Examples: Three variable (X, Y, Z) minterms and maxterms

Index	Minterm (m)	Maxterm (M)
0	$\bar{X}\bar{Y}\bar{Z}$	$X + Y + Z$
1	$\bar{X}\bar{Y}Z$	$X + Y + \bar{Z}$
2	$\bar{X}Y\bar{Z}$	$X + \bar{Y} + Z$
3	$\bar{X}YZ$	$X + \bar{Y} + \bar{Z}$
4	$X\bar{Y}\bar{Z}$	$\bar{X} + Y + Z$
5	$X\bar{Y}Z$	$\bar{X} + Y + \bar{Z}$
6	$XY\bar{Z}$	$\bar{X} + \bar{Y} + Z$
7	XYZ	$\bar{X} + \bar{Y} + \bar{Z}$

- **The *index* above is important for describing which variables in the terms are true and which are complemented**

Scanned by CamScanner

Standard Order

- Minterms and maxterms are designated with a subscript
- The subscript is a number, corresponding to a binary pattern
- The bits in the pattern represent the complemented or normal state of each variable listed in a standard order $m_0 / m_1 / m_2 \ldots$ $M_0 / M_1 / M_2 \ldots$
- All variables will be present in a minterm or maxterm and will be listed in the ***same order (usually alphabetically)*** ترتيب أبجدي
- **Example: For variables a, b, c:**
  - **Maxterms:** $(a + b + \bar{c})$, $(a + b + c)$
  - **Terms:** $(b + a + c)$, $a\bar{c}b$, **and** $(c + b + a)$ **are NOT in standard order.** → مرتبين أبجدياً (عادي يجوز) و لكن يكون (term) ليس (maxterm).
  - **Minterms:** $\bar{a}\bar{b}c$, $ab\bar{c}$, $\bar{a}bc$ → must contain all variables & must be in the standerd order.
  - **Terms:** $(a + c)$, $\bar{b}c$, **and** $(\bar{a} + b)$ **do not contain all variables**

* الـ minterm : هو تعبير يحتوي جميع الـ Variables (متغيرات) في السؤال ويكون فيه جميع المتغيرات بالترتيب الأبجدي.

Scanned by CamScanner

Purpose of the Index

- The ***index*** for the minterm or maxterm, expressed as a binary number, is used to determine whether the variable is shown in the true form or complemented form

*Ex :- $m_0 = \bar{x}\bar{y}\bar{z} = 000$
$m_1 = \bar{x}\bar{y}z = 001$
(minterms):

- **For Minterms:**
  - **"0" means the variable is "Complemented"**
  - **"1" means the variable is "Not Complemented"**

*Ex :- $M_0 = x + y + z = 000$
$M_1 = x + y + \bar{z} = 001$
(maxterms):

عكس بعض دائماً

- **For Maxterms:**
  - **"0" means the variable is "Not Complemented"**
  - **"1" means the variable is "Complemented"**

Scanned by CamScanner

Index Example: Three Variables

* for minterm: 0 → complemented. 1 → true.

* for maxterm: 0 → true. 1 → complemented.

Index (Decimal)	Index (Binary) n = 3 Variables	Minterm (m)	Maxterm (M)
0	000	$m_0 = \bar{X}\bar{Y}\bar{Z}$	$M_0 = X + Y + Z$
1	001	$m_1 = \bar{X}\bar{Y}Z$	$M_1 = X + Y + \bar{Z}$
2	010	$m_2 = \bar{X}Y\bar{Z}$	$M_2 = X + \bar{Y} + Z$
3	011	$m_3 = \bar{X}YZ$	$M_3 = X + \bar{Y} + \bar{Z}$
4	100	$m_4 = X\bar{Y}\bar{Z}$	$M_4 = \bar{X} + Y + Z$
5	101	$m_5 = X\bar{Y}Z$	$M_5 = \bar{X} + Y + \bar{Z}$
6	110	$m_6 = XY\bar{Z}$	$M_6 = \bar{X} + \bar{Y} + Z$
7	111	$m_7 = XYZ$	$M_7 = \bar{X} + \bar{Y} + \bar{Z}$

Scanned by CamScanner

Index Example: Four Variables

* الفرق فقط أنه تم زيادة عدد المتغيرات ← $2^4 = 16$.

i (Decimal)	i (Binary) n = 4 Variables	الحد تكون (-) m_i	الحد تكون (+) M_i
0	0000	$\bar{a}\bar{b}\bar{c}\bar{d}$	$a+b+c+d$
1	0001	$\bar{a}\bar{b}\bar{c}d$	$a+b+c+\bar{d}$
3	0011	$\bar{a}\bar{b}cd$	$a+b+\bar{c}+\bar{d}$
5	0101	$\bar{a}b\bar{c}d$	$a+\bar{b}+c+\bar{d}$
7	0111	$\bar{a}bcd$	$a+\bar{b}+\bar{c}+\bar{d}$
10	1010	$a\bar{b}c\bar{d}$	$\bar{a}+b+\bar{c}+d$
13	1101	$ab\bar{c}d$	$\bar{a}+\bar{b}+c+\bar{d}$
15	1111	$abcd$	$\bar{a}+\bar{b}+\bar{c}+\bar{d}$

* بنا. $(\ldots, m_6, m_4, m_2)$ ← عوا بزم كـ (function) جز.

Scanned by CamScanner

Minterm and Maxterm Relationship

- Review: DeMorgan's Theorem
  - $\overline{x.y} = \bar{x} + \bar{y}$ and $\overline{x + y} = \bar{x}.\bar{y}$
- Two-variable example:
  - $M_2 = \bar{x} + y$ and $m_2 = x.\bar{y}$
  - Using DeMorgan's Theorem → $\overline{\bar{x} + y} = \bar{\bar{x}}.\bar{y} = x.\bar{y}$
  - Using DeMorgan's Theorem → $\overline{x.\bar{y}} = \bar{x} + \bar{\bar{y}} = \bar{x}.y$
  - Thus, M_2 is the complement of m_2 and vice-versa
- Since DeMorgan's Theorem holds for n variables, the above holds for terms of n variables:

$$M_i = \overline{m_i} \text{ and } m_i = \overline{M_i}$$

- Thus, M_i is the complement of m_i and vice-versa

Scanned by CamScanner

Function Tables for Both

- Minterms of 2 variables:

* كل صفّ بكون الرقم (1) موجود لحالة وحدة والباقي أصفار (0).
* نلاحظ بأنّ الرقم (1) قليل كثيرًا لذلك minterm.

xy	m_0	m_1	m_2	m_3
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

- Maxterms of 2 variables:

* كل عمود أو سطر في جدول الـ (minterm) يكون نفي لنفس العمود أو السطر في الـ (maxterm). لأنّ $(M_i = \overline{m_i})$ وكذلك $(m_i = \overline{M_i})$
* كما قلنا سابقًا بأنّ $M_i = \overline{m_i}$ حيث أنّهم عكس بعض.
* نلاحظ بأن الرقم (1) سائد كثيرًا لذلك maxterm.
* الرقم هنا موجود بحالة واحدة

xy	M_0	M_1	M_2	M_3
00	0	1	1	1
01	1	0	1	1
10	1	1	0	1
11	1	1	1	0

عكس بعض تمامًا

- Each column in the maxterm function table is the complement of the column in the minterm function table since M_i is the complement of m_i. ⊛ $M_i = \overline{m_i}$ ⊛ $M_i = m_i$

Scanned by CamScanner

Observations

- In the function tables:
  - Each ***minterm*** has one and only one 1 present in the 2^n terms (a minimum of 1s). All other entries are 0.
  - Each ***maxterm*** has one and only one 0 present in the 2^n terms All other entries are 1 (a maximum of 1s).

- We can implement any function by
  - "ORing" the minterms corresponding to "1" entries in the function table. These are called the minterms of the function.
  - "ANDing" the maxterms corresponding to "0" entries in the function table. These are called the maxterms of the function.

- This gives us two canonical forms for stating any Boolean function:
  - ***Sum of Minterms (SOM)*** → $f(x,y) = x\bar{y} + xy$. (ORing)
  - ***Product of Maxterms (POM)*** → $f(x,y) = (x+\bar{y}) \cdot (x+y)$. (ANDing)

Scanned by CamScanner

Minterm Function Example

- **Example: Find** $F_1 = m_1 + m_4 + m_7$
- $F_1(x,y,z) = x'y'z + xy'z' + xyz$ → الجدول الذي يكون جوابهم (1) بال minterm وجوابهم (0) بال maxterm (Some of minterm) (SOM)

$$\underbrace{x'y'z}_{001 \to m_1} + \underbrace{xy'z'}_{100 \to m_4} + \underbrace{xyz}_{111 \to m_7}$$

$2^3 = 8$ squares in the K-map

xyz	Index	$m_1 + m_4 + m_7 = F_1$
000	0	$0 + 0 + 0 = 0$
001	1	$1 + 0 + 0 = 1$
010	2	$0 + 0 + 0 = 0$
011	3	$0 + 0 + 0 = 0$
100	4	$0 + 1 + 0 = 1$
101	5	$0 + 0 + 0 = 0$
110	6	$0 + 0 + 0 = 0$
111	7	$0 + 0 + 1 = 1$

Scanned by CamScanner

Minterm Function Example

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

أرقام الـ(minterm) الذين قيمتهم =(صفر) وهم (٢، ٩، ١٧، ٢٣).

- $F(A, B, C, D, E) = A'B'C'DE' + A'BC'D'E + AB'C'D'E + AB'CDE$

* ملاحظة: أرقام الـ(maxterms) تخبرنا عن الأماكن التي يكون فيها الـ (function) = (صفر).

وكذلك أرقام الـ (minterms) تخبرنا عن الأماكن التي يكون فيها الـ (function) = (واحد).

Scanned by CamScanner

Maxterm Function Example

- **Example: Implement F1 in maxterms:**
- $F_1 = M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$ (0,2,3,5,6) هي أماكن الأصفار
- $F_1 = (x+y+z)\cdot(x+y'+z)\cdot(x+y'+z')\cdot(x'+y+z')\cdot(x'+y'+z)$ maxterms للـ truth table لأنه

xyz	Index	$M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 = F_1$
000	0	0 . 1 . 1 . 1 . 1 = 0
001	1	1 . 1 . 1 . 1 . 1 = 1
010	2	1 . 0 . 1 . 1 . 1 = 0
011	3	1 . 1 . 0 . 1 . 1 = 0
100	4	1 . 1 . 1 . 1 . 1 = 1
101	5	1 . 1 . 1 . 0 . 1 = 0
110	6	1 . 1 . 1 . 1 . 0 = 0
111	7	1 . 1 . 1 . 1 . 1 = 1

Scanned by CamScanner

Maxterm Function Example

- $F(A, B, C, D) = M_3 \cdot M_8 \cdot M_{11} \cdot M_{14}$

- $F(A, B, C, D)$ [product of sums]
  $= (A + B + C' + D') \cdot (A' + B + C + D) \cdot$
  $(A' + B + C' + D') \cdot (A' + B' + C' + D)$

Scanned by CamScanner

Canonical Sum of Minterms

From (SOP) → (SOM)

- Any Boolean function can be expressed as a Sum of Minterms (SOM):
  - For the function table, the minterms used are the terms corresponding to the 1's
  - For expressions, expand all terms first to explicitly list all minterms. Do this by "ANDing" any term missing a variable v with a term $(v + \bar{v})$

- Example: Implement $f_{(x,y)} = x + \bar{x}\bar{y}$ as a SOM?

not (SOM) but (SOP): (Sum of products)

1. Expand terms → $f_{(x,y)} = x(y + \bar{y}) + \bar{x}\bar{y}$
2. Distributive law → $f = xy + x\bar{y} + \bar{x}\bar{y}$
3. Express as SOM → $f = m_3 + m_2 + m_0 = m_0 + m_2 + m_3$

ترتيبهم تصاعدي ليس شرط!

Scanned by CamScanner

Another SOM Example

- Example: $F = A + \bar{B}C$
- There are three variables: A, B, and C which we take to be the standard order
- Expanding the terms with missing variables:
  - $F = A(B + \bar{B})(C + \bar{C}) + (A + \bar{A})\bar{B}C$
- Distributive law:
  - $F = ABC + A\bar{B}C + AB\bar{C} + A\bar{B}\bar{C} + A\bar{B}C + \bar{A}\bar{B}C$
- Collect terms (removing all but one of duplicate terms):
  - $F = ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}\bar{B}C$
- Express as SOM:
  - $F = m_7 + m_6 + m_5 + m_4 + m_1$
  - $F = m_1 + m_4 + m_5 + m_6 + m_7$

Scanned by CamScanner

Shorthand SOM Form

- **From the previous example, we started with:**
  - $F = A + \bar{B}C$
- **We ended up with:**
  - $F = m_1 + m_4 + m_5 + m_6 + m_7$
- **This can be denoted in the *formal shorthand*:**
  - $F(A, B, C) = \sum_m (1,4,5,6,7)$
- **Note that we explicitly show the standard variables in order and drop the "m" designators.**

Scanned by CamScanner

Canonical Product of Maxterms

From (POS) → (POM)

- Any Boolean Function can be expressed as a Product of Maxterms (POM):
  - For the function table, the maxterms used are the terms corresponding to the 0's
  - For an expression, expand all terms first to explicitly list all maxterms. Do this by first applying the second distributive law , "ORing" terms missing variable v with $(v \cdot \bar{v})$ and then applying the distributive law again
- Example: Convert $f(x, y, z) = x + \bar{x}\bar{y}$ to POM?
  - Distributive law → $f = (x + \bar{x}) \cdot (x + \bar{y}) = x + \bar{y}$
  - ORing with missing variable (z) → $f = x + \bar{y} + (z \cdot \bar{z})$
  - Distributive law → $f = (x + \bar{y} + z) \cdot (x + \bar{y} + \bar{z})$
  - Express as POS → $f = M_2 \cdot M_3$

*For converting s-
from (SOP) to (SOM)
$X \cdot (v + \bar{v}) = X$
$X + (v \cdot \bar{v}) = X$
from (POS) to (POM)

* في الجدول رتبة الـ (maxterm) هو يمثل قيمة (صفر) في الـ truth table

is (POS)

POM

Scanned by CamScanner

Another POM Example

- Convert $f(A,B,C) = AC' + BC + A'B'$ to POM?
- Use $x + yz = (x + y) \,.\, (x + z)$, assuming $x = AC' + BC$ and $y = A'$ and $z = B'$
  - $f(A,B,C) = (AC' + BC + A') \,.\, (AC' + BC + B')$
- Use Simplification theorem to get:
  - $f(A,B,C) = (BC + A' + C') \,.\, (AC' + B' + C)$
- Use Simplification theorem again to get:
  - $f(A,B,C) = (A' + B + C') \,.\, (A + B' + C) = M_5 \,.\, M_2$
  - $f(A,B,C) = M_2 \,.\, M_5 = \prod_M(2,5)$ → ***Shorthand POM form***

Scanned by CamScanner

Function Complements

- The complement of a function expressed as a sum of minterms is constructed by selecting the minterms missing in the sum-of-minterms canonical forms.
- Alternatively, the complement of a function expressed by a sum of minterms form is simply the Product of Maxterms with the same indices.
- Example: Given $F(x,y,z) = \sum_m(1,3,5,7)$, find complement F as SOM and POM?
  - $\bar{F}(x,y,z) = \sum_m(0,2,4,6)$ (SOM)
  - $\bar{F}(x,y,z) = \prod_M(1,3,5,7)$ (POM)

$F(x,y,z) = m_1 + m_3 + m_5 + m_7.$

$F(x,y,z) = \sum_m(1,3,5,7) = \pi_M(0,2,4,6).$

$\bar{F}(x,y,z) = \sum_m(0,2,4,6) = \pi_M(1,3,5,7).$

Scanned by CamScanner

Conversion Between Forms

- To convert between sum-of-minterms and product-of-maxterms form (or vice-versa) we follow these steps:
  1. Find the function complement by swapping terms in the list with terms not in the list.
  2. Change from products to sums, or vice versa.

- **Example:** Given F as before: $F(x, y, z) = \sum_m(1,3,5,7)$
  - Form the Complement:

$$\bar{F}(x, y, z) = \sum_m(0,2,4,6)$$

  - Then use the other form with the same indices – this forms the complement again, giving the other form of the original function:

$$F(x, y, z) = \prod_M(0,2,4,6)$$

Scanned by CamScanner

Important Properties of Minterms

- Maxterms are seldom (نادرة) used directly to express Boolean functions *Minterms are used more than Maxterms.

* في (function) معيّن الـ(complement) له هو نفسه الـ(maxterm) لذلك الـ(function) (كمتمم)

- Minterms properties:

* إذا كان كل الـ(minterms) موجودين بالـ(function) معناه قيمته (1)

* إذا كان كل الـ(maxterms) موجودين بالـ(function) معناه قيمته (0)

  - For n Boolean variables, there are 2^n minterms (0 to 2^n -1)
  - Any Boolean function can be represented as a logical sum of minterms (SOM)

    $F(1,3,7)$ Minterm $\rightarrow \bar{F}(2,4,5,6)$ Maxterm.

  - The complement of a function contains those minterms not included in the original function.

    * النفي للاقتران يحتوي على الحدود الغير موجودة بالاقتران نفسه (مكمّلة لها).

  - A function that include all the 2^n minterms is equal to 1

$f(x,y) = \Sigma m(0,1,2,3) \Rightarrow$ The function contains all (2^n) then it equals to (1).

* عندما تحتوي الـ(minterm) على جميع عناصر الـ(function) يكون لا يوجد (maxterms) ويكون جواب الـ function = (1) والـ(complement) للـ(function) هو (0) دائماً.

Scanned by CamScanner

Standard Forms

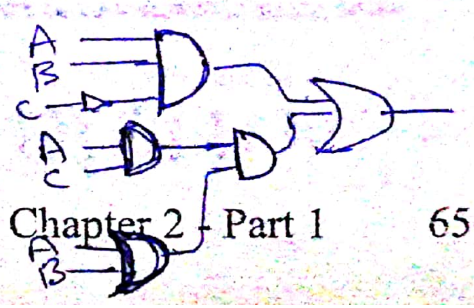
- Standard Sum-of-Products (SOP) form: equations are written as an OR of AND terms $F(x,y) = (x \cdot y) + (\bar{x} \cdot y) + (x \cdot \bar{y}) + (\bar{x} \cdot \bar{y})$ (SOM)

  * لا يشترط فيه احتواء جميع الحدود على جميع المتغيرات.

- Standard Product-of-Sums (POS) form: equations are written as an AND of OR terms $F(x,y) = (x+y) \cdot (\bar{x}+y) \cdot (x+\bar{y}) \cdot (\bar{x}+\bar{y})$ (POM)

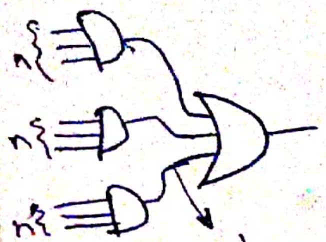
- Examples:
  - SOP: $ABC + \bar{A}\bar{B}C + B$ → not (SOM)
  - POS: $(A + B) \cdot (A + \bar{B} + \bar{C}) \cdot C$ → not (POM)

- These "mixed" forms are neither SOP nor POS
  - $(AB + C)(A + C)$
  - $AB\bar{C} + AC(A + B)$



Standard Sum-of-Products (SOP)

- A sum of minterms form for n variables can be written down directly from a truth table
- Implementation of this form is a two-level network of gates such that:
  - The first level consists of n-input AND gates, and
  - The second level is a single OR gate (with fewer than 2^n inputs)
- This form often can be simplified so that the corresponding circuit is simpler



Scanned by CamScanner

Standard Sum-of-Products (SOP)

- A Simplification Example: $F(A,B,C) = \sum m(1,4,5,6,7)$
- Writing the minterm expression:
  - $F(A,B,C) = A'B'C + AB'C' + AB'C + ABC' + ABC$
- Simplifying using boolean Algebra:

$A'B'C + AB'C' + AB'C + ABC' + ABC$	$m1 + m4 + m5 + m6 + m7$
$= A'B'C + AB'(C' + C) + AB(C' + C)$	*Distributive law*
$= A'B'C + AB' + AB$	$X + X' = 1$
$= A'B'C + A(B' + B)$	*Distributive law*
$= A'B'C + A.1$	*Simplification Theorem*
$= A + B'C$	

- Simplified F contains 3 literals compared to 15 in minterm F

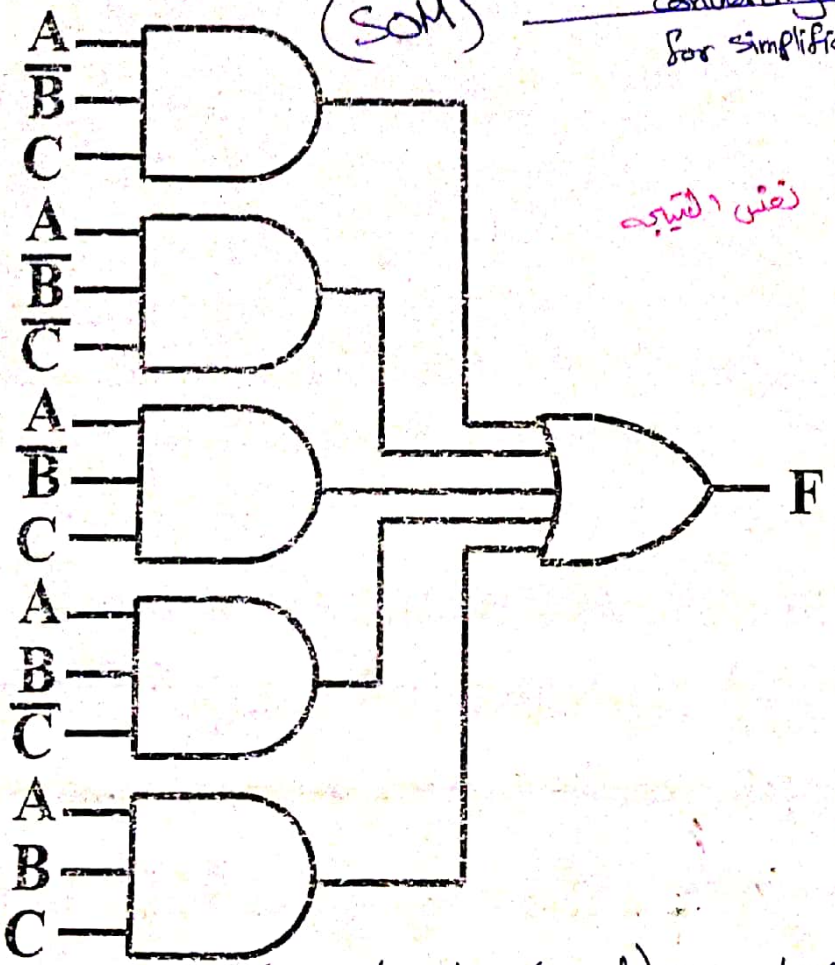
Scanned by CamScanner

AND/OR Two-level Implementation of SOP Expression

- **The two implementations for F are shown below -- it is quite apparent which is simpler!**

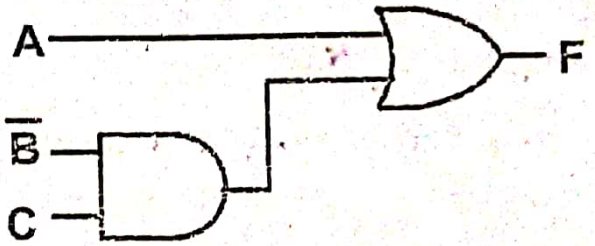
(SOM) converting for simplification (SOP)

* لتبسيط شكل الـ (circuit)



two levels (and gate) then (OR gate)

نفس النتيجة



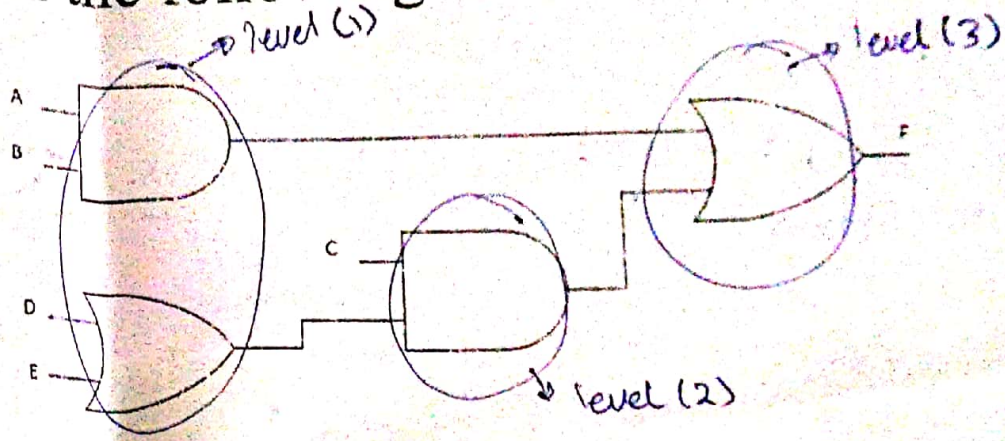
two levels (and gate) then (OR gate)

ولكن هذه مبسّطة أكثر.
وعدد البوابات أقل لتقليل الـ (cost)

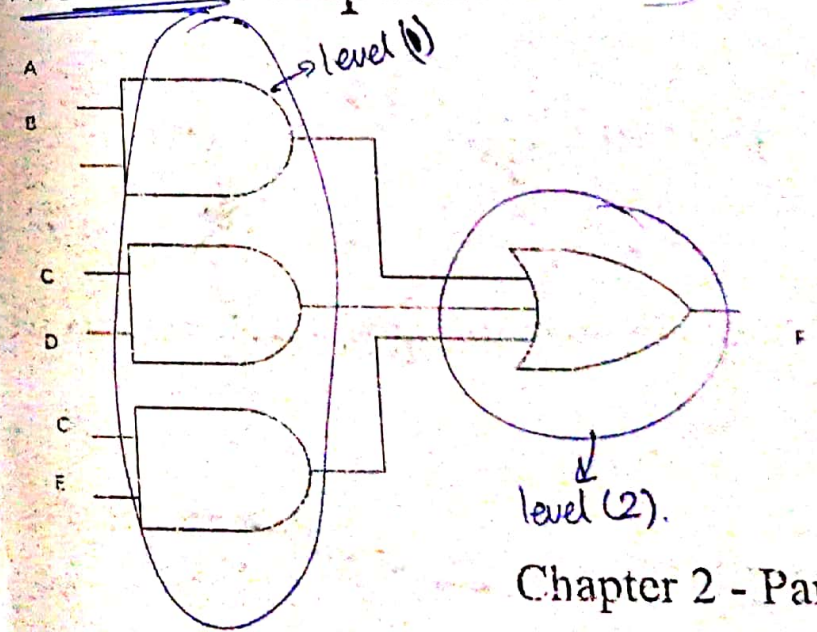
Scanned by CamScanner

Two-level Implementation

- Draw the logic diagram of the following boolean function:
  - $f = AB + C(D + E)$



- Represent the function using two-level implementation:
  - $f = AB + CD + CE$ → SOP



Scanned by CamScanner

SOP and POS Observations

- *The previous examples show that:*
  - Canonical Forms (Sum-of-minterms, Product-of-Maxterms), or other standard forms (SOP, POS) differ in complexity
  - *Boolean algebra* can be used to manipulate (اختصار) equations into simpler forms.
  - Simpler equations lead to simpler two-level implementations

* عند التحويل من: SOP ⇆ SOM
POS ⇆ POM
نقلل الـ (cost).

- Questions:
  - How can we attain a "simplest" expression?
  - Is there only one minimum cost circuit?
  - The next part will deal with these issues.

Scanned by CamScanner

Literal Cost (L)

- ***Literal:*** a variable or its complement أي متغير بحسبه
- ***Literal cost (L):*** the number of literal appearances in a Boolean expression corresponding to the logic circuit diagram
- Examples:
  - $F = BD + AB'C + AC'D'$
    - $L = 8$ (Minimum cost → Best solution)
  - $F = BD + AB'C + AB'D' + ABC'$
    - $L = 11$
  - $F = (A + B)(A + D)(B + C + D')(B' + C' + D)$
    - $L = 10$

*L :- عدد الأسلاك هو نفسه عدد الـ Variables

Scanned by CamScanner

Gate Input Cost

- ***Gate input cost (G):*** the number of inputs to the gates in the implementation corresponding exactly to the given equation or equations. (***G: inverters not counted, GN: inverters counted***)
- For SOP and POS equations, it can be found from the equation(s) by finding the sum of:
  - **All literal appearances (L)**
  - **The number of terms excluding single literal terms,(G) and**
  - **optionally, the number of distinct complemented single literals (GN).**
- Examples:
  - $F = BD + AB'C + AC'D'$ (number of terms = 3, L = 11; number of variables complemented (inverters) = 3)
    - $G = 11, GN = 14$ (Minimum cost → Best solution)
  - $F = BD + AB'C + AB'D' + ABC'$
    - $G = 15, GN = 18$
  - $F = (A + B)(A + D)(B + C + D')(B' + C' + D)$
    - $G = 14, GN = 17$

Scanned by CamScanner

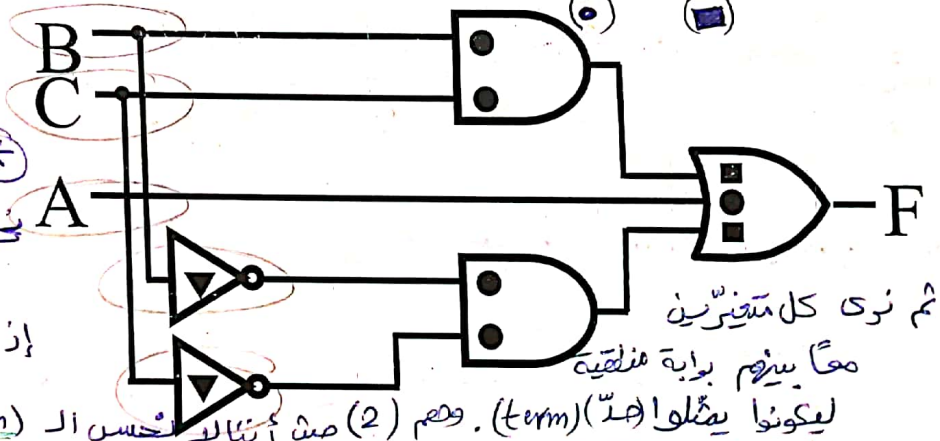
Cost Criteria (continued)

- **Example 1:**
- $F = A + B\bar{C} + \bar{B}C$

$GN = G + 2 = 9$

$L = 5$

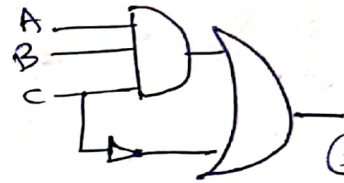
$G = L + 2 = 7$



- **L (literal count) counts the AND inputs and the single literal OR input.**
- **G (gate input count) adds the remaining OR gate inputs**
- **GN(gate input count with NOTs) adds the inverter inputs**

*Example:-

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



① $L = 4$ (ABC) هو الحدّ (term) (C̄) لا نعتبر term

② $G = 5 = 4 + 1$

③ $GN = 6$ $5+1$ (C̄) منفية

Scanned by CamScanner

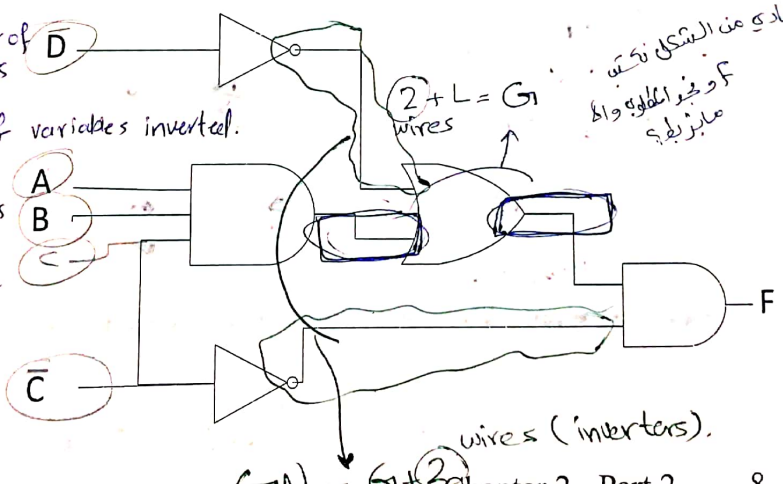
Cost Criteria (continued)

- **Example 2:**
- $F = (A, B, C, D) = (ABC + D').C'$
  - $L = 5$
  - $G = 5 + 2 = 7$ → because distributive law: $AB\bar{C}C + \bar{D}\bar{C}$. (term① term②)
  - $GN = 7 + 2 = 9$

L = variables = inputs = number of

GN = G + number of variables inverted.

G = L + number of terms that contains more than one variable.

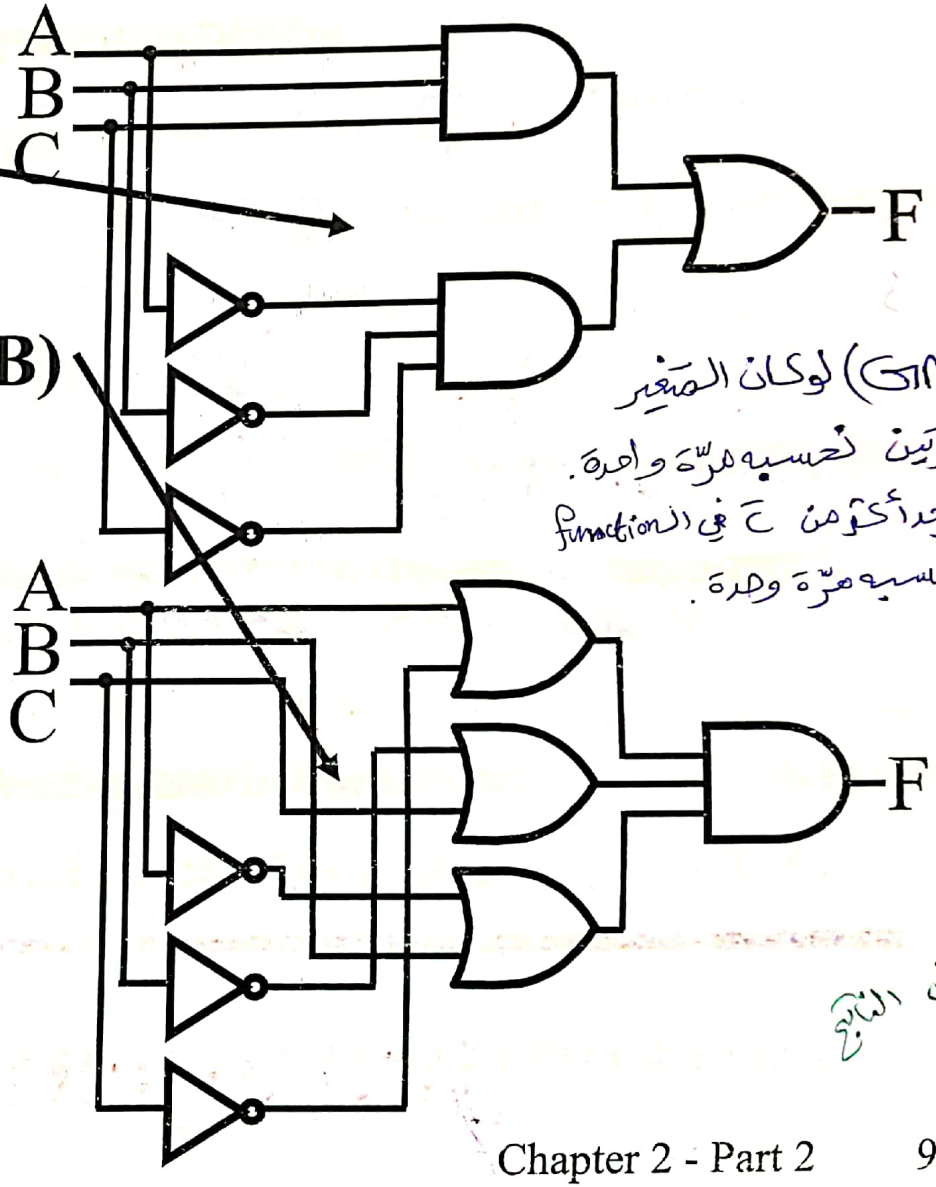


GN = G + 2

Scanned by CamScanner

Cost Criteria (continued)

- **Example 3:**
- **$F = A\,B\,C + \bar{A}\,\bar{B}\,\bar{C}$**
- **L = 6, G = 8, GN = 11**
- **$F = (A + \bar{C})(\bar{B} + C)(\bar{A} + B)$**
- **L = 6 , G = 9, GN = 12**
- **Same function and same literal cost** (L). same variables number.
- **But first circuit has better gate input count** (G) **and better gate input count with NOTs** (GN)
- **Select it!**



* في (GN) لو كان المتغير منفي مرتين نحسبه مرّة واحدة. يوجد أكثر من C في الـ function نحسبه مرّة واحدة.

الرسمتين نفس الناتج

© 2008 Pearson Education, Inc.

Scanned by CamScanner

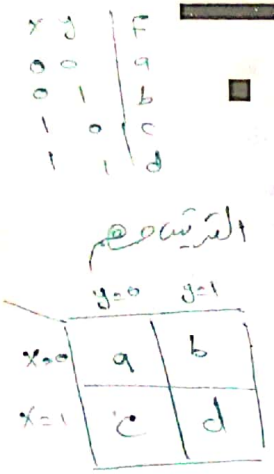
Boolean Function Optimization

- Minimizing the gate input (or literal) cost of a (a set of) Boolean equation(s) reduces circuit cost
- We choose gate input cost
- Boolean Algebra and graphical techniques are tools to minimize cost criteria values
- Some important questions:
  - When do we stop trying to reduce the cost?
  - Do we know when we have a minimum cost?
- *Treat optimum or near-optimum cost functions for two-level (SOP and POS) circuits*
- *Introduce a graphical technique using Karnaugh maps (K-maps, for short)*

Scanned by CamScanner

Karnaugh Maps (K-map)

- A K-map is a collection of squares
  - Graphical representation of the truth table
  - Each square represents a minterm, or a maxterm, or a row in the truth table
  - For n-variable, there are 2^n squares
  - The collection of squares is a graphical representation of a Boolean function
  - Adjacent squares differ in the value of one variable
  - Alternative algebraic expressions for the same function are derived by recognizing patterns of squares



Scanned by CamScanner

Some Uses of K-Maps

- Finding optimum or near optimum
  - SOP and POS standard forms, and
  - two-level AND/OR and OR/AND circuit implementations

  for functions with small numbers of variables
- Visualizing concepts related to manipulating Boolean expressions, and
- Demonstrating concepts used by computer-aided design programs to simplify large circuits

Scanned by CamScanner

Two Variable Maps

- **A 2-variable Karnaugh Map:**
  - Note that minterm m_0 and minterm m_1 are "adjacent" and differ in the value of the variable y
  - Similarly, minterm m_0 and minterm m_2 differ in the x variable
  - Also, m_1 and m_3 differ in the x variable as well
  - Finally, m_2 and m_3 differ in the value of the variable y

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

Scanned by CamScanner

K-Map and Truth Tables

2^n = عدد المربعات → 2^n = عدد الـ rows and columns.

- The K-Map is just a different form of the truth table
- Example: Two variable function
  - We choose a,b,c and d from the set {0,1} to implement a particular function, $F(x, y)$

implementation of a function: Truth table. → K-maps.

Input Values (x, y)	F(x, y)
0 0	a
0 1	b
1 0	c
1 1	d

Truth Table

x \ y	y = 0	y = 1
x = 0	a	b
x = 1	c	d

K-Map

(y) احتمالات — (x) احتمالات

Same implementation for the function.

Scanned by CamScanner

K-Map Function Representation

- Example: $F(x, y) = x$

$F(x, y) = x$	y = 0	y = 1
x = 0	0	0
x = 1	1	1

- For function $F(x, y)$, the two adjacent cells containing 1's can be combined using the Minimization Theorem:

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

© 2008 Pearson Education, Inc.

Scanned by CamScanner

K-Map Function Representation

- Example: $G(x, y) = x + y$

$f(x,y) = \Sigma_m(1,2,3)$ →

$\bar{x}y + x\bar{y} + xy$ →

$\bar{x}y + x(y+\bar{y})$ (1) →

$x + \bar{x}y = x + y$ (simplification theorem).

$G(x, y) = x + y$	y = 0	y = 1
x = 0	0	1
x = 1	1	1

المشترك بينهم هو (y)

المشترك بينهم هو (x)

* لجميع الـ (1) يكون هو جواب الـ function يعني : جواب هذا الـ function هو (x+y)

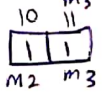
- For $G(x, y)$, two pairs of adjacent cells containing 1's can be combined using the Minimization Theorem:

يتم دمجهم.

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

$$G(x, y) = x + y$$

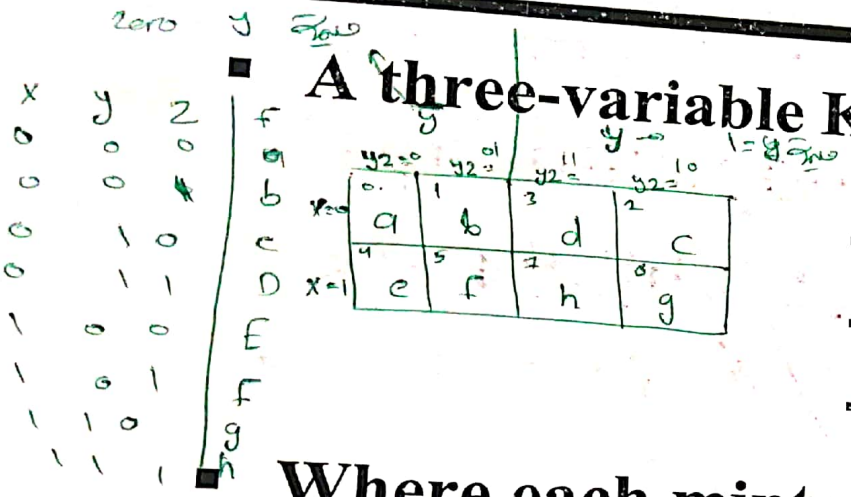
: $\bar{x}y + xy \rightarrow y(\bar{x}+x) = y.1 =$ (y)



: $x\bar{y} + xy \rightarrow x(\bar{y}+y) = x.1 =$ (x)

Scanned by CamScanner

Three Variable Maps



- **A three-variable K-map:**

	yz = 00	yz = 01	yz = 11	yz = 10
x = 0	m_0	m_1	m_3	m_2
x = 1	m_4	m_5	m_7	m_6

- **Where each minterm corresponds to the product terms:**

	yz = 00	yz = 01	yz = 11	yz = 10
x = 0	$\bar{x}\bar{y}\bar{z}$	$\bar{x}\bar{y}z$	$\bar{x}yz$	$\bar{x}y\bar{z}$
x = 1	$x\bar{y}\bar{z}$	$x\bar{y}z$	xyz	$xy\bar{z}$

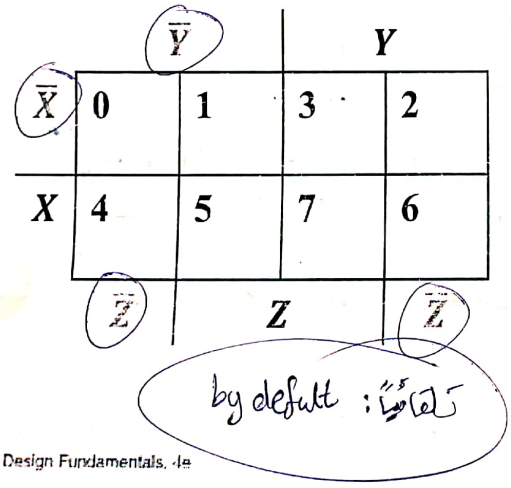
- ***Note that if the binary value for an index differs in one bit position, the minterms are adjacent on the K-Map***

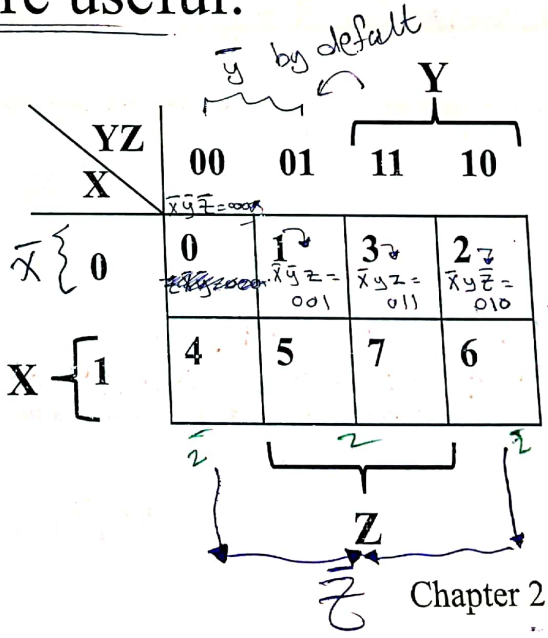
Scanned by CamScanner

Alternative Map Labeling

- Map use largely involves:
  - Entering values into the map, and
  - Reading off product terms from the map
- Alternate labelings are useful:

* نذكر الرموز المشتركة فيهم فقط.





Scanned by CamScanner

Example Functions

- By convention, we represent the minterms of F by a "1" in the map and leave the minterms of $\bar{F}$ blank
- Example:
  - $F(x, y, z) = \sum_m(2,3,4,5)$

	$\bar{y}$		Y	
$\bar{x}$	0	1	3	2
	0	0	1	1
X	4	5	7	6
	1	1	0	0
	$\bar{z}$		Z	$\bar{z}$

most significant = X

least significant = Z

- Example:
  - $G(a, b, c) = \sum_m(3,4,6,7)$

	$\bar{b}$		b	
$\bar{a}$	0	1	3	2
	0	0	1	0
a	4	5	7	6
	1	0	1	1
	$\bar{c}$		c	$\bar{c}$

$F = bc + ab + a\bar{c}$

$2^3 =$ 3 variables.

- Learn the locations of the 8 indices based on the variable order shown (X, most significant and Z, least significant) on the map boundaries

© 2008 Pearson Education, Inc.

Scanned by CamScanner

Steps for using K-Maps to Simplify Boolean Functions

- Enter the function on the K-Map
  - Function can be given in truth table, shorthand notation, SOP,…etc
  - Example:
    - $F(x,y) = \bar{x} + xy$
    - $F(x,y) = \sum m(0,1,3)$

x	y	F(x,y)
0	0	1
0	1	1
1	0	0
1	1	1

		y
	0: 1	1: 1
x	2:	3: 1

- Combining squares for simplification
  - Rectangles that include power of 2 squares {1, 2, 4, 8, …}
  - Goal: Fewest rectangles that cover all 1's → as large as possible
- Determine if any rectangle is not needed (not essential)
- Read-off the SOP terms

Scanned by CamScanner

Combining Squares

- By combining squares, we reduce number of literals in a product term, reducing the literal cost, thereby reducing the other two cost criteria

- On a 2-variable K-Map:
  - One square represents a minterm with two variables
  - Two adjacent squares represent a product term with one variable
  - Four "adjacent" terms is the function of all ones (no variables) = 1.

- On a 3-variable K-Map:
  - One square represents a minterm with three variables
  - Two adjacent squares represent a product term with two variables
  - Four "adjacent" terms represent a product term with one variable
  - Eight "adjacent" terms is the function of all ones (no variables) = 1.

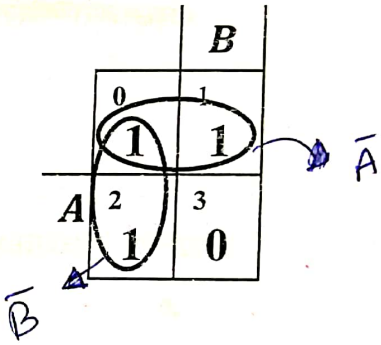
© 2008 Pearson Education, Inc.

Scanned by CamScanner

Example: Combining Squares

- Example: $F(A,B) = \sum_m(0,1,2)$

$$F(A,B) = \overset{m_0}{\bar{A}\bar{B}} + \overset{m_1}{\bar{A}B} + \overset{m_2}{A\bar{B}}$$



- Using Distributive law
  - $F(A,B) = \bar{A} + A\bar{B}$
- Using simplification theorem
  - $F(A,B) = \bar{A} + \bar{B}$

- **Thus, every two adjacent terms that form a 2×1 rectangle correspond to a product term with one variable**

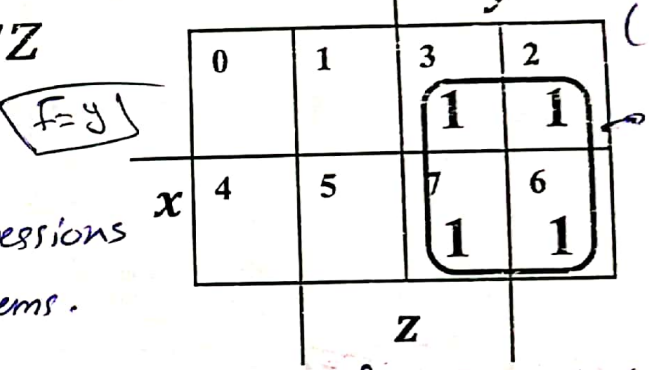
Scanned by CamScanner

Example: Combining Squares

- Example: $F(x, y, z) = \sum_m(2,3,6,7)$

$F(x,y,z) = \sum_m(1,3,5,7) = \prod_M(0,2,4,6)$

$\bar{F}(x,y,z) = \sum_m(0,2,4,6) = \prod_M(1,3,5,7)$



- $F(x, y, z) = \bar{x}y\bar{z} + \bar{x}yz + xy\bar{z} + xyz$
- Using Distributive law
  - $F(x, y, z) = \bar{x}y + xy$ ← By Boolean Expressions using theorems.

* when we have a function with (2 variables)

① عندما يكون لدينا (1) ولا تجمّع مع (1) غيرها ← يكون تمثيله بمتغيّرين. (2 variables)

② عندما يكون لدينا (1) مجمّعة مع (1) أخرى. ← يكون تمثيله بمتغيّر واحد. (1 variable)

③ عندما يكون لدينا (1) مجمّعة مع (4 واحدات) أخرى. ← يكون تمثيله برقم ثابت (عدد) أي (0 variables)

- Using Distributive law again.
  - $F(x, y, z) = y$ (by the k-map with combining the (ones together))
- **Thus, the four adjacent terms that form a 2×2 square correspond to the term "y"**

* عند تجميع الـ (ones) معًا يقلّ كل مجموعة عدد الـ variables بمقدار واحد. (1)

Scanned by CamScanner

Three-Variable Maps

- Reduced literal product terms for SOP standard forms correspond to rectangles on K-maps containing cell counts that are powers of 2

* عدد الـ (ones) الـ
يتم تجميعهم بحيث أن يكون 2^n حول العدد (2)

- Rectangles of 2 cells (Squares) represent 2 adjacent minterms

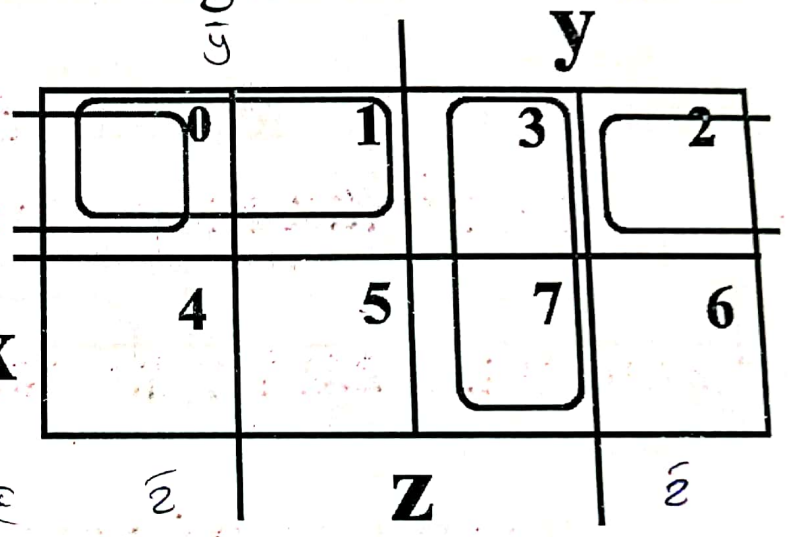
- Rectangles of 4 cells represent 4 minterms that form a "pairwise adjacent" ring

- Rectangles can contain non-adjacent cells as illustrated by the "pairwise adjacent" ring above

Scanned by CamScanner

Three-Variable Maps

- Example shapes of 2-cell rectangles:

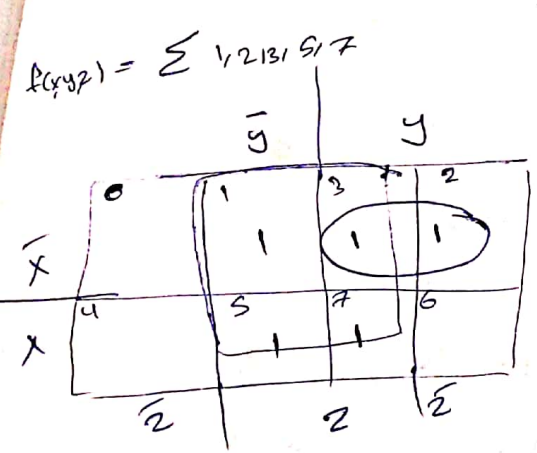


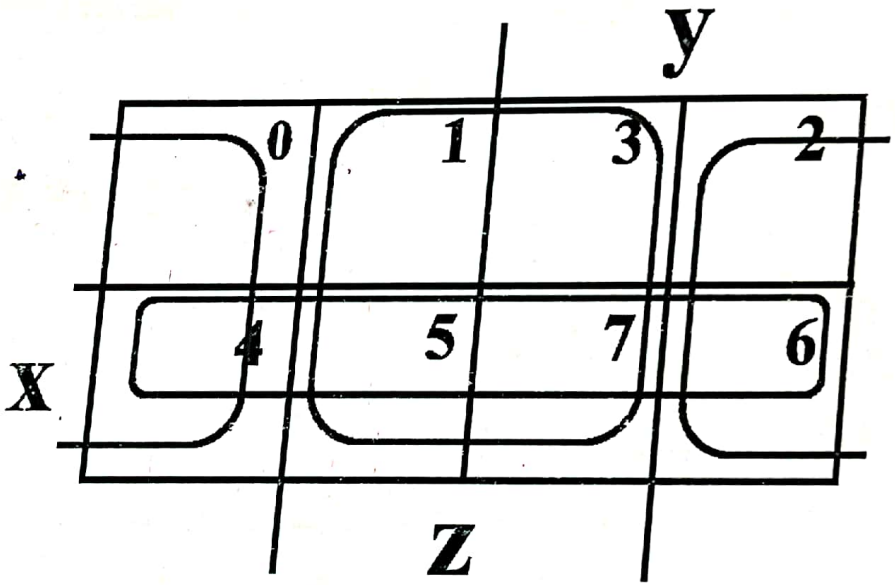
- Read-off the product terms for the rectangles shown:
  - $Rect(0,1) = \bar{X}\bar{Y}$
  - $Rect(0,2) = \bar{X}\bar{Z}$
  - $Rect(3,7) = YZ$

Scanned by CamScanner

Three-Variable Maps

- *Example shapes of 4-cell Rectangles:*



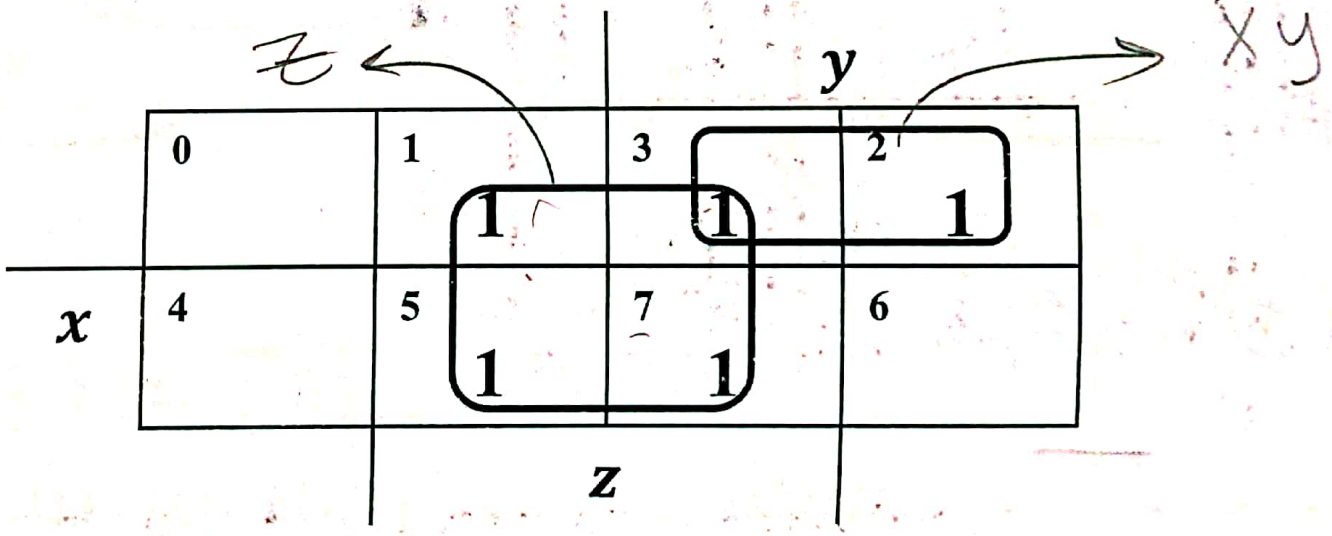


- *Read off the product terms for the rectangles shown:*
  - $Rect(1,3,5,7) = Z$
  - $Rect(0,2,4,6) = \bar{Z}$
  - $Rect(4,5,6,7) = X$

Scanned by CamScanner

Three Variable Maps

- K-maps can be used to simplify Boolean functions by systematic methods. Terms are selected to cover the "1s"in the map.
- Example: Simplify $F(x, y, z) = \sum_m(1,2,3,5,7)$

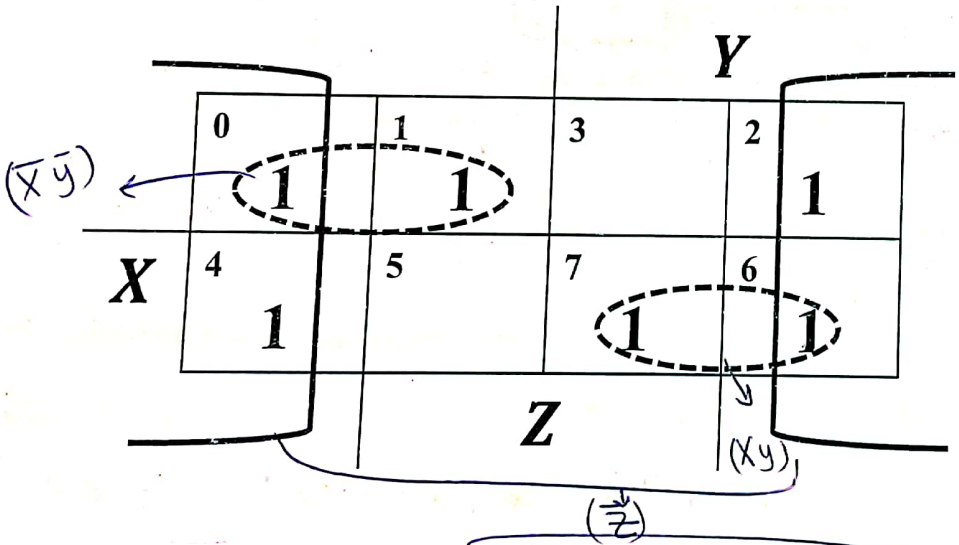


$$F(x, y, z) = z + \bar{x}y$$

Scanned by CamScanner

Three-Variable Map Simplification

- Use a K-map to find an optimum SOP equation for $F(X,Y,Z) = \sum_m(0,1,2,4,6,7)$



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

$$\bar{F} = \bar{\bar{Z}} \cdot \overline{\bar{X}\bar{Y}} \cdot X\bar{Y}$$

$$\bar{F} = Z \cdot (X+Y) \cdot (\bar{X}+\bar{Y})$$

Scanned by CamScanner

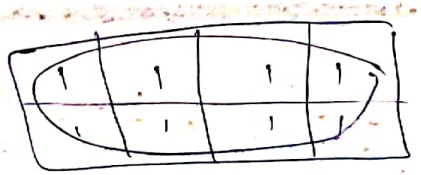
Four Variable Maps

- Map and location of minterms $F(W, X, Y, Z)$:

WX \ YZ	00	01	11	10	
00	0 $\bar{w}\bar{x}\bar{y}\bar{z}$ 0000	1 $\bar{w}\bar{x}\bar{y}z$ 0001	3 $\bar{w}\bar{x}yz$ 0011	2 $\bar{w}\bar{x}y\bar{z}$ 0010	$\bar{X}$
01	4	5	7	6	
11	12	13	15	14	X
10	8	9	11	10	X
		Z			

W: 00 = $\bar{W}$, 01

Y: 11, 10 = $\bar{Y}$



Scanned by CamScanner

Four Variable Terms

- Four variable maps can have rectangles corresponding to:
  - A single 1: 4 variables (i.e. Minterm) — which means: $F = (4)$ variables.
  - Two 1's: 3 variables — $F = (3)$ variables.
  - Four 1's: 2 variables — $F = (2)$ variables
  - Eight 1's: 1 variable — $F = (1)$ variable
  - Sixteen 1's: zero variables (function of all ones) $F = 1$.

* كل ما يقل متغير بقل عدد التمثيل بمقدار (1) (متغير واحد).

* دائماً يتم تجميع الـ (1) بـ power of (2)

$2^4 / 2^3 / 2^2 / 2^1 / 2^0$

16 8 4 2 1

عدد الـ cells أو الـ rectangles.

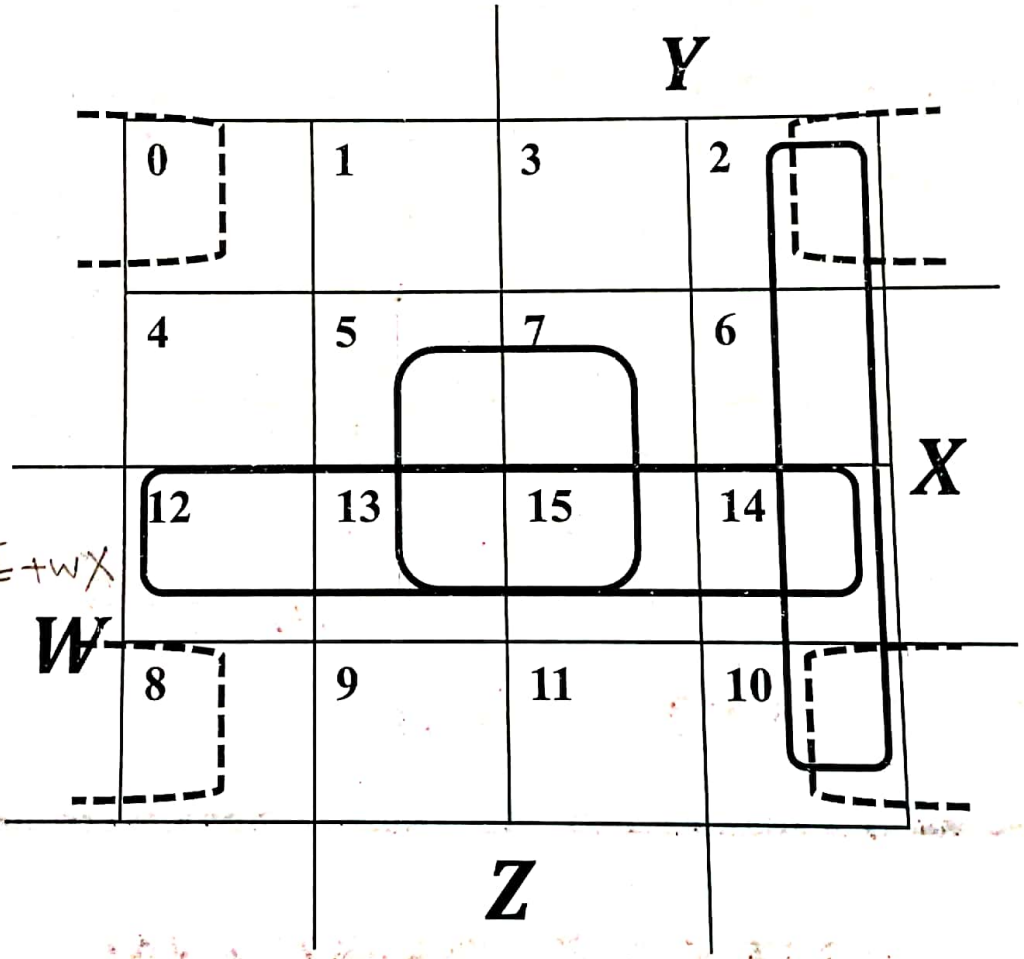
Scanned by CamScanner

Four-Variable Maps

- Example shapes of 4-cell rectangles:

rec(5,7,13,15) : XZ.
rec(0,2,8,10) : $\bar{X}\bar{Z}$.
rec(2,6,10,14) : $y\bar{Z}$.
rec(12,13,14,15) : WX.

$f(w,x,y,z) = XZ + \bar{X}\bar{Z} + y\bar{Z} + wX$

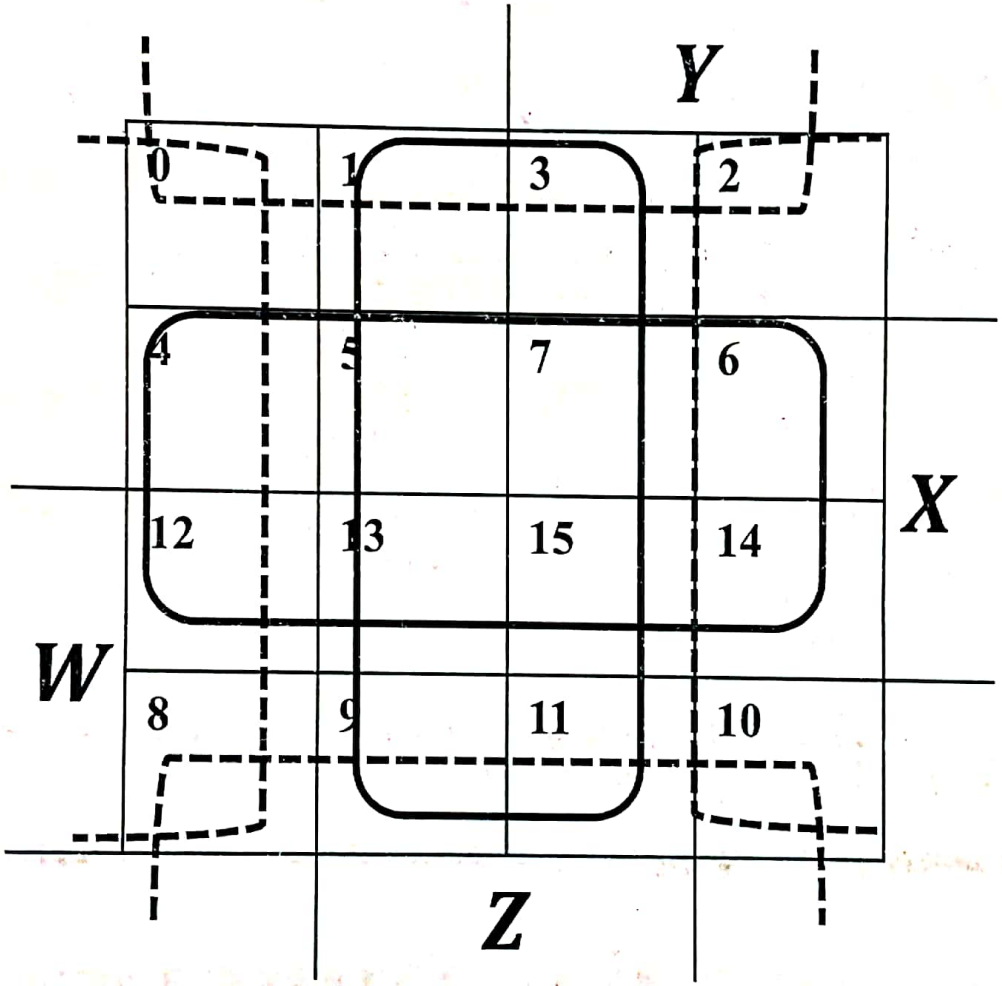


© 2008 Pearson Education, Inc.

Scanned by CamScanner

Four-Variable Maps

- Example shapes of 8-cell rectangles:

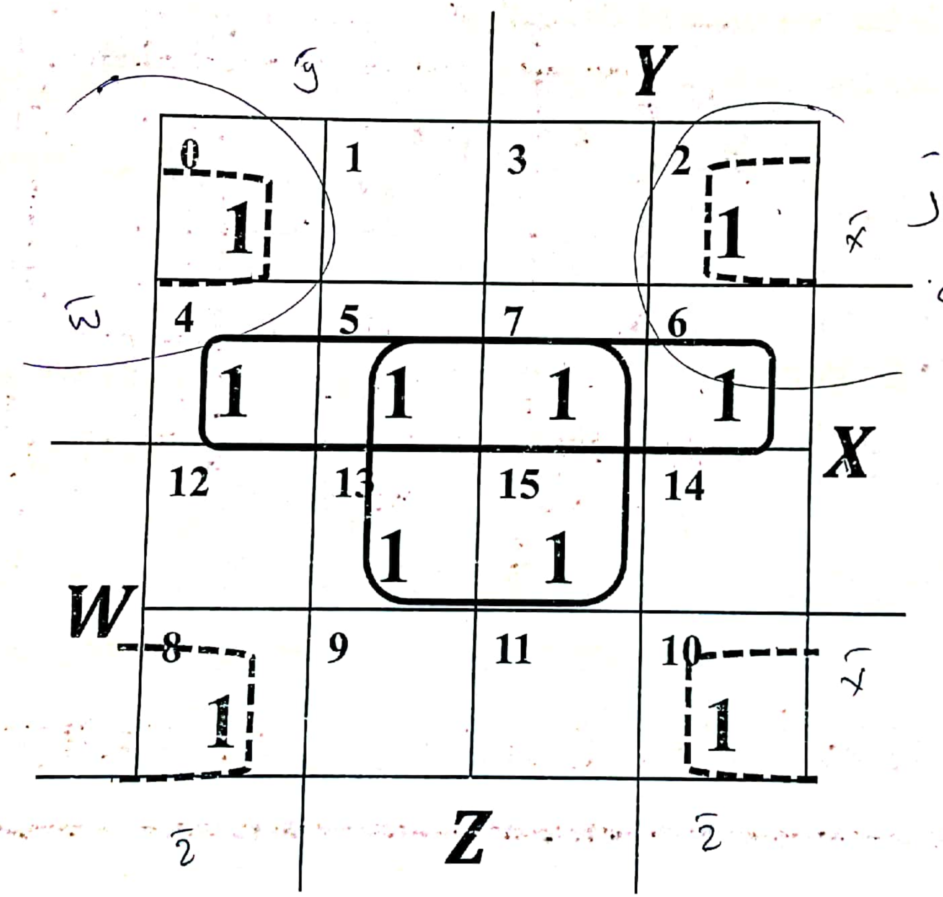


Scanned by CamScanner

Four-Variable Map Simplification

- $F(W,X,Y,Z) = \Sigma_m(0,2,4,5,6,7,8,10,13,15)$

min لازم نغطي كل رقم بأقل عدد من المجموعات



* الـ (prime implicant) :- هو أن نجمع أكبر عدد من الـ (ones) معًا بأكبر مربع.

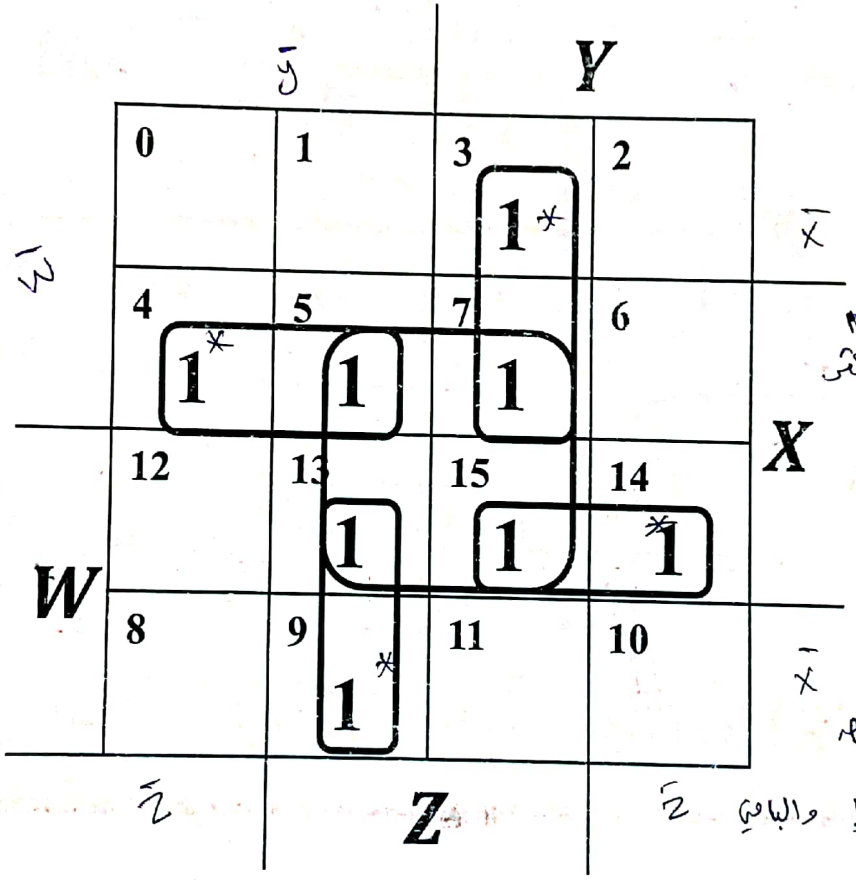
ممكن نأخذ 0,4,2,6 و $\bar{W}\bar{Z}$

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

Scanned by CamScanner

Four-Variable Map Simplification

- $F(W,X,Y,Z) = \sum_m(3,4,5,7,9,13,14,15)$



* Essential Implicants:-

* rec(9,13)
* rec (3,7)
* rec (4,5)
* rec (14,15)

* Non - Essential :-
rec(5, 7, 13, 15) لأنه مكرر والـ (ones) فيه مغطيين بأكثر من مربع آخر.

5 - prime implicants.

rec(5,7) → لا يكون prime implicant لأنه يوجد rec(5,7,13,15) وهي أكبر لذلك تعتبر prime implicant

* لو كان عنا الجدول فاضي ومعطيني function ولازم أعبي الجدول 1,0 نحط 1 مكان كل تقاطع من الكد الواحد يعني W̄ȲZ وين تتقاطعوا نحط 1 والباقي اصفار وهكذا

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

* ما أخذنا الدائرة الكبيرة لأنها مش essential (نفقر لوخذ الـ 1 اللي وضعنا مع دوائر أخرى)

Scanned by CamScanner

Systematic Simplification

* Implicant ≡ term.

Prime Implicant: is a product term obtained by combining the **maximum** possible number of adjacent squares in the map into a rectangle with the number of squares a power of 2

* the maximum possible number of adjacent terms.

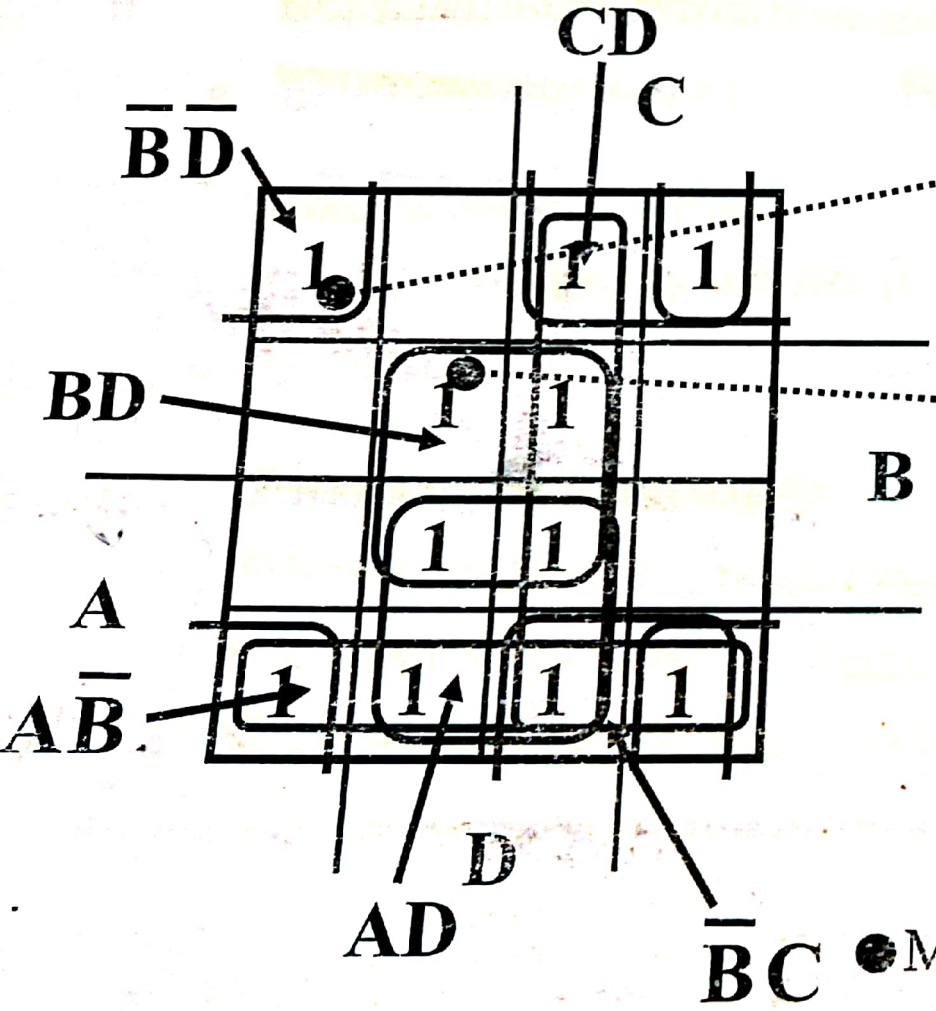
- A prime implicant is called an ***Essential Prime Implicant*** if it is the **only** prime implicant that covers (includes) one or more minterms
- Prime Implicants and Essential Prime Implicants can be determined by inspection of a K-Map
- A set of prime implicants *"covers all minterms"* if, for each minterm of the function, at least one prime implicant in the set of prime implicants includes the minterm

© 2008 Pearson Education, Inc.

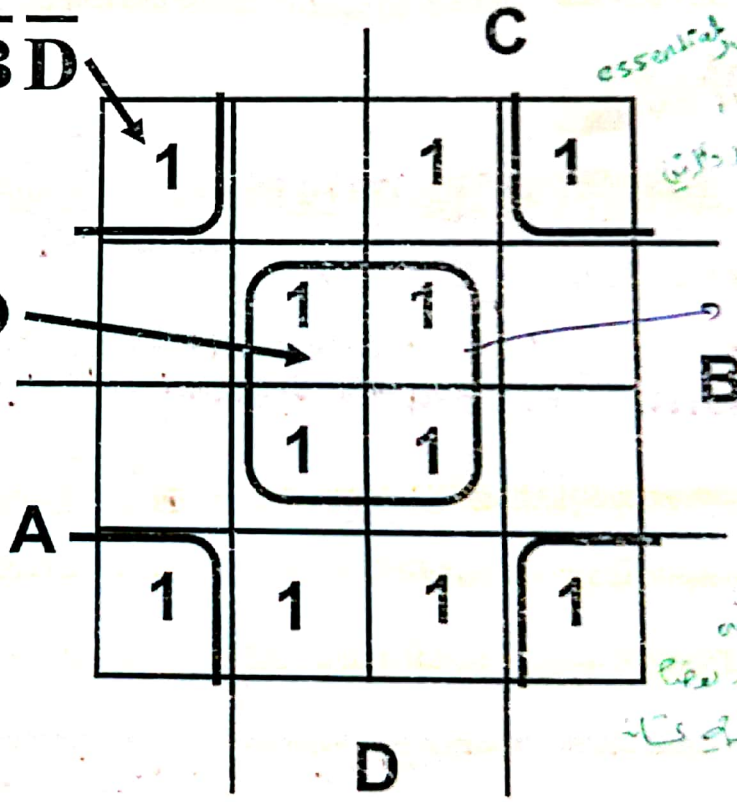
Scanned by CamScanner

Example of Prime Implicants

- **Find ALL Prime Implicants**



ESSENTIAL Prime Implicants



● Minterms covered by single prime implicant

Scanned by CamScanner

Prime Implicant Practice

- Find all prime implicants for:

$$F(A,B,C,D) = \sum_{m}(0,2,3,8,9,10,11,12,13,14,15)$$

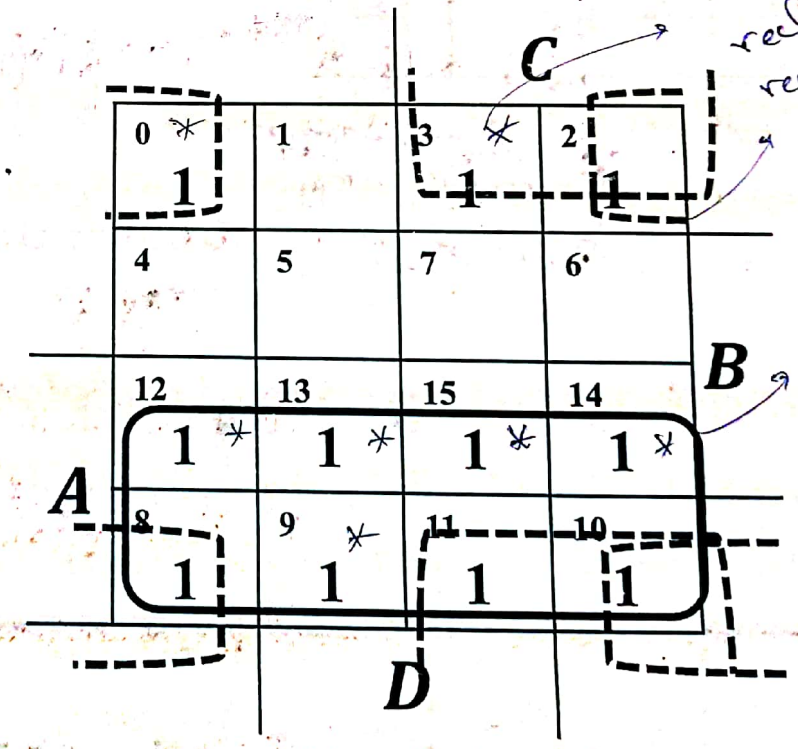
- Prime Implicants: [* Essentials].
  - A
  - $\bar{B}C$
  - $\bar{B}\bar{D}$

essential

essential

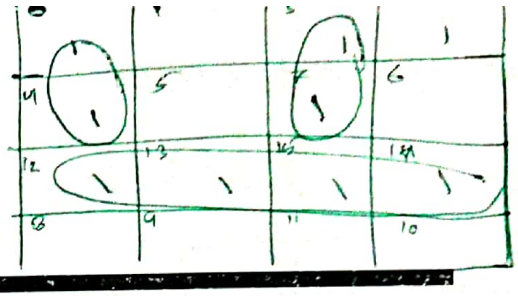
essential

(or any one)



Scanned by CamScanner

Another Example



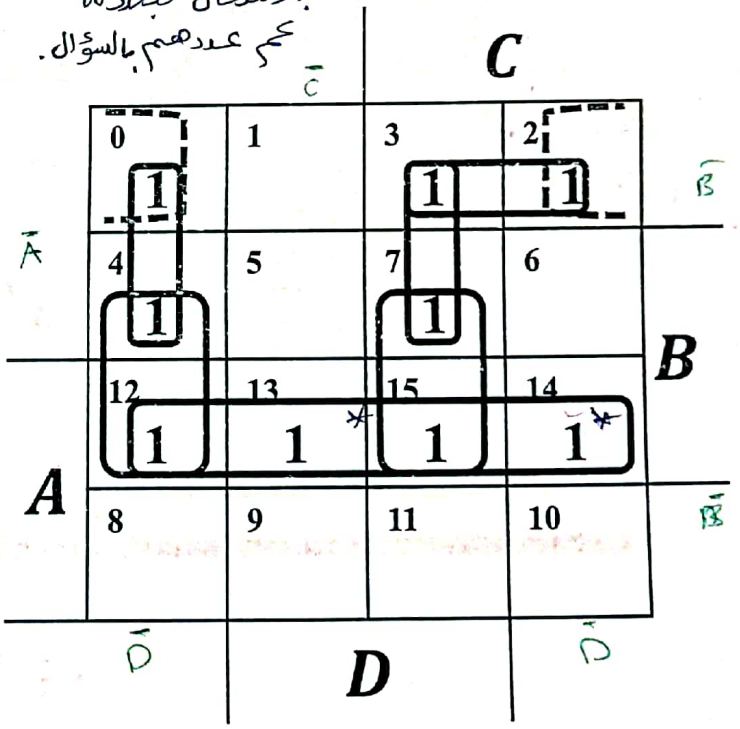
- Find all prime implicants for:

$$G(A,B,C,D) = \sum_m (0,2,3,4,7,12,13,14,15)$$

- Hint: There are seven prime implicants! لأن عددهم كثير بالامتحان فيحدّد لنا كم عددهم بالسؤال.

- Prime Implicants:
  - AB → (12, 14) The only one (Essential).
  - BCD (7,15)
  - $B\bar{C}\bar{D}$ (4,12)
  - $\bar{A}CD$ (3,7)
  - $\bar{A}\bar{C}\bar{D}$ (0,4)
  - $\bar{A}\bar{B}C$ (2,3)
  - $\bar{A}\bar{B}\bar{D}$ (0,2)

* كل مُغطّى (one) بأكثر من مرّة لذلك يكون (non-essential) نوعًا



Scanned by CamScanner

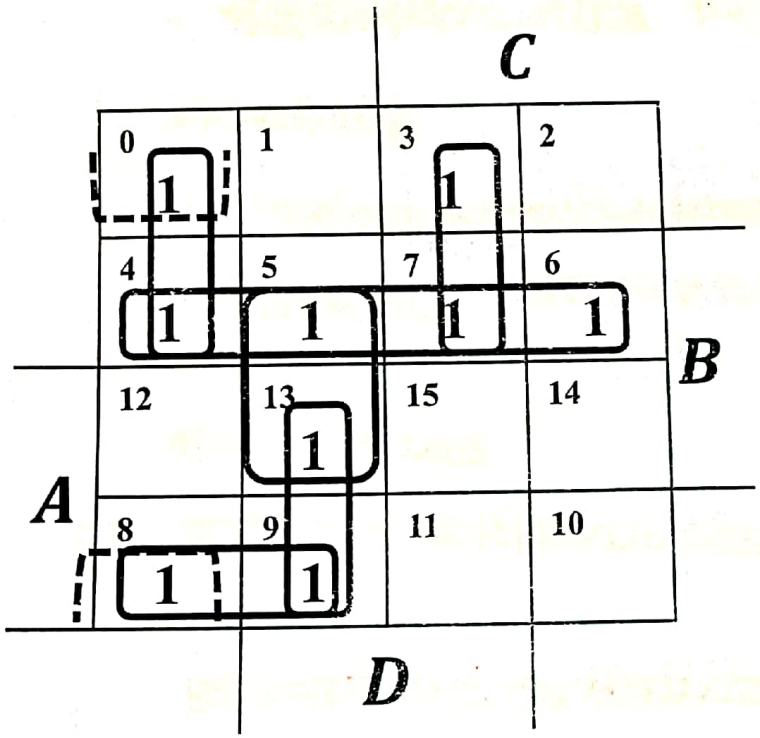
Optimization Algorithm

1. Find all prime implicants
2. Include all essential prime implicants in the solution
3. Select a ***minimum cost*** set of non-essential prime implicants to cover all minterms not yet covered
   - Selection Rule: Minimize the overlap among prime implicants as much as possible. In particular, in the final solution, make sure that each prime implicant selected includes at least one minterm not included in any other prime implicant selected

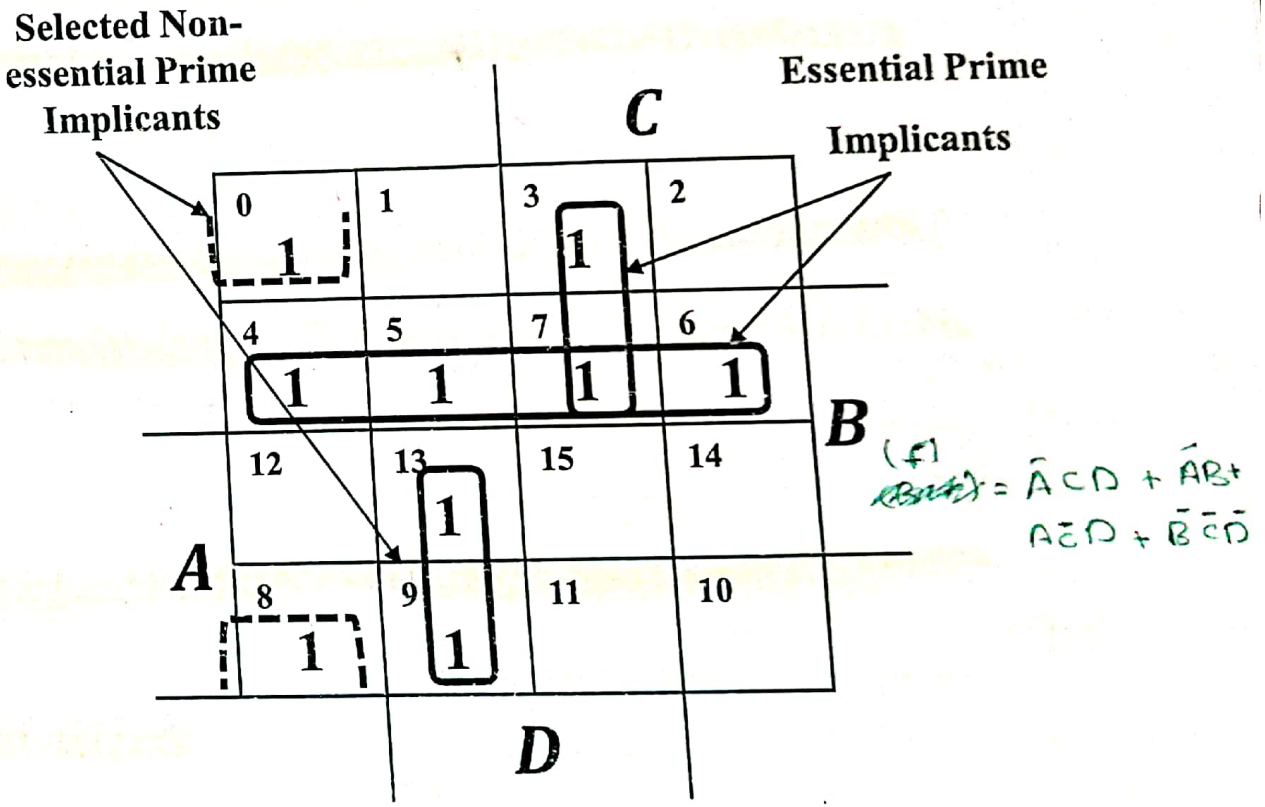
Scanned by CamScanner

Selection Rule Example

- Simplify F(A, B, C, D) given on the K-map



Prime Implicants



Essential and Selected Non-essential Prime Implicants

(f) = $\bar{A}CD + A\bar{B}\bar{C} + A\bar{C}D + \bar{B}\bar{C}\bar{D}$

الأفضل أن يكون عدد الـ(ones) أقل (prime implicants) *

→ not prime implicant.

→ a prime implicant.

Scanned by CamScanner

Product of Sums Example

- Find the optimum POS solution for:

$$F(A,B,C,D) = \sum_m (1,3,9,11,12,13,14,15)$$

Solution:

1. Find optimized SOP for $\bar{F}$ by combining 0's in K-Map of F
2. Complement $\bar{F}$ to obtain optimized POS for F

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

- Using Demorgan's Law:

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

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

Scanned by CamScanner

Example

- Find the optimum POS and SOP solution for:

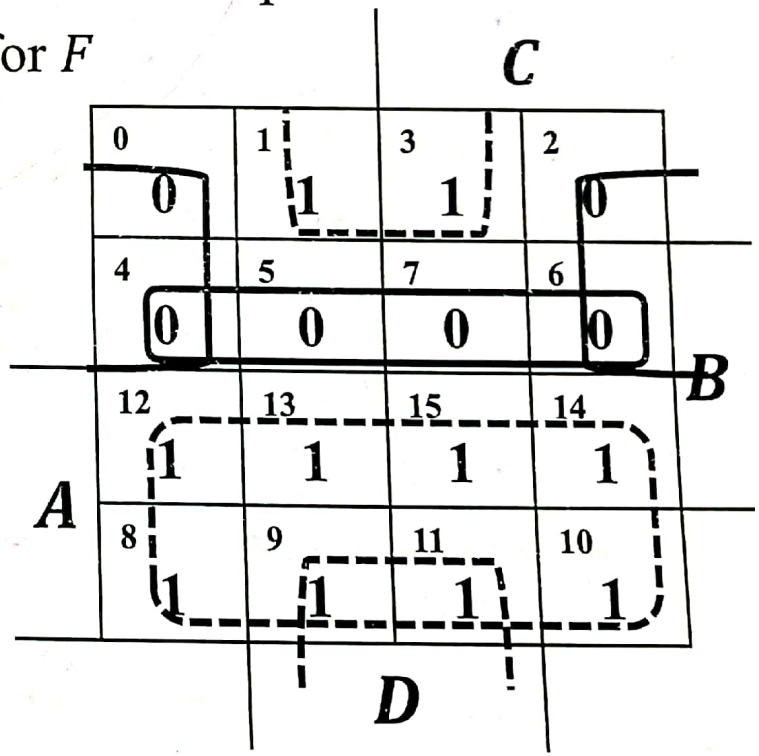
$$F(A,B,C,D) = \prod(0,2,4,5,6,7)$$

- POS solution (Red):
  - Find optimized SOP for $\bar{F}$ by combining 0's in K-Map of F
  - Complement $\bar{F}$ to obtain optimized POS for F

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

- SOP solution (Blue):
  - Combining 1's in K-Map of F

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



Scanned by CamScanner

Don't Cares in K-Maps

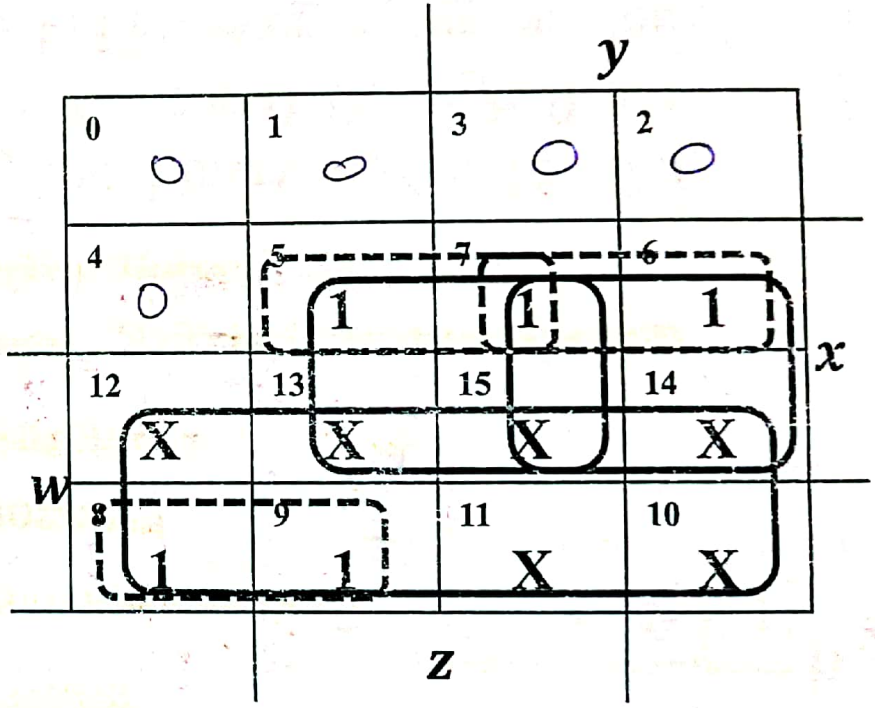
- Incompletely specified functions: Sometimes a function table or map contains entries for which it is known:
  - the input values for the minterm will never occur, or
  - The output value for the minterm is not used
- In these cases, the output value is defined as a "don't care"
- By placing "don't cares" (an "x" entry) in the function table or map, the cost of the logic circuit may be lowered
- **Example:** A logic function having the binary codes for the BCD digits as its inputs. Only the codes for 0 through 9 are used. The six codes, 1010 through 1111 never occur, so the output values for these codes are "x" to represent "don't cares".
- ***"Don't care" minterms cannot be replaced with 1's or 0's because that would require the function to be always 1 or 0 for the associated input combination***

© 2008 Pearson Education, Inc.

Scanned by CamScanner

Example: BCD "5 or More"

- The map below gives a function $F(w, x, y, z)$ which is defined as "5 or more" over BCD inputs. With the don't cares used for the 6 non-BCD combinations:



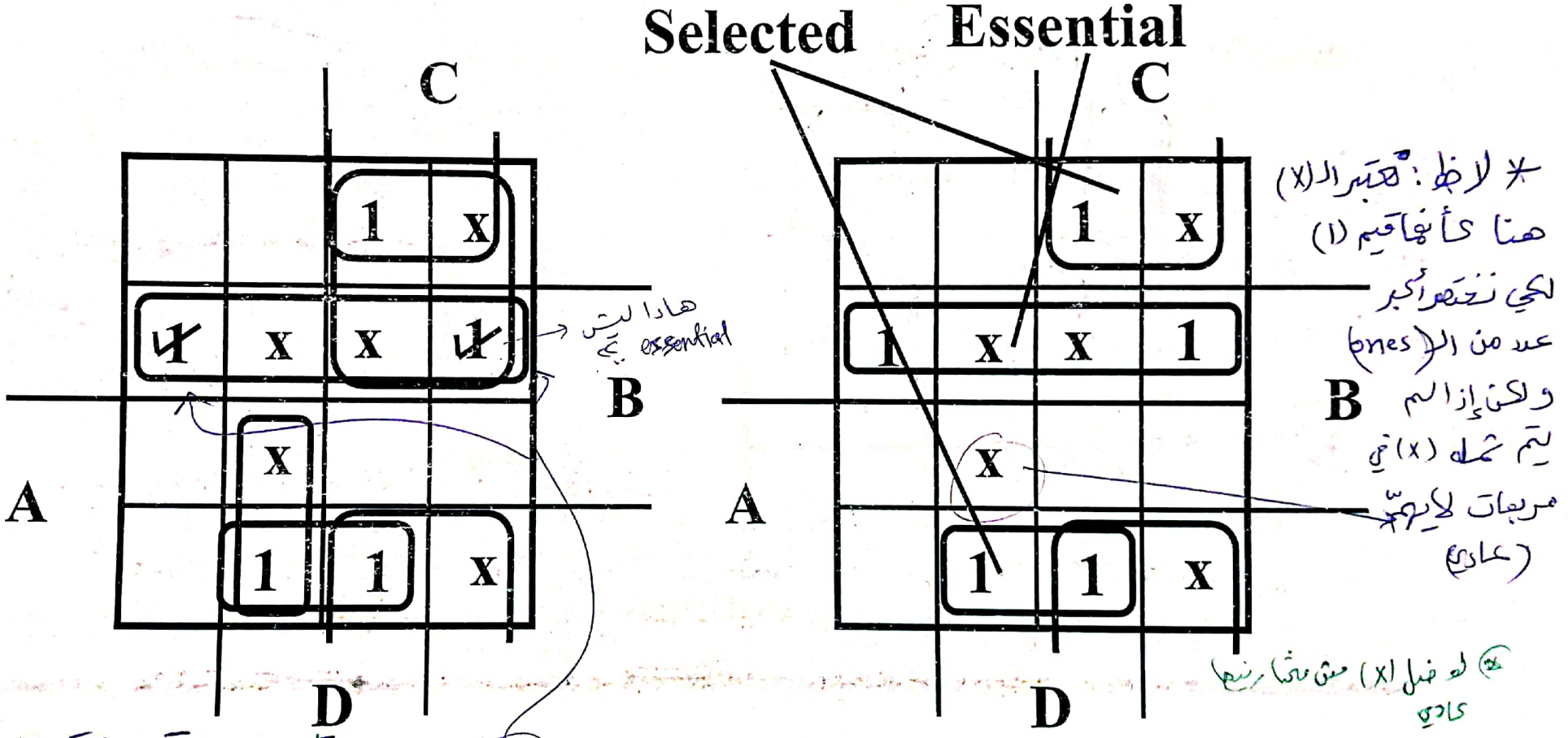
- If don't cares are treated as 1's (Red):
- $F_1(w, x, y, z) = w + xy + xz$
  - $G = 7$
- If don't cares are treated as 0's (Blue):
- $F_2(w, x, y, z) = \bar{w}xz + \bar{w}xy + w\bar{x}\bar{y}$
  - $G = 12$

- ***For this particular function, cost G for the POS solution for $F(w, x, y, z)$ is not changed by using the don't cares***
  - ***Choose the one less inverters (i.e. less GN)***

Scanned by CamScanner

Selection Rule Example with Don't Cares

- Simplify F(A, B, C, D) given on the K-map.



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

GL = 10 . GN = 12.

✓ Minterms covered by essential prime implicants

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

GL = 10 . GN = 12 . → (inverters) أقل لأنه أقل عدد

© 2008 Pearson Education, Inc.

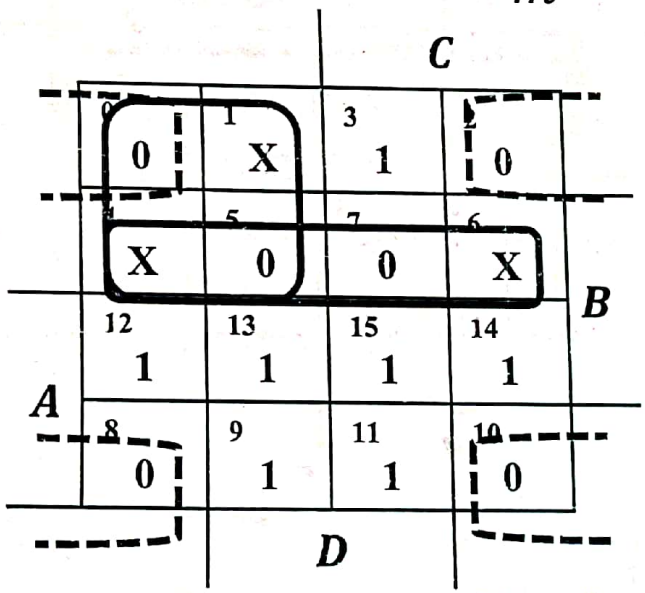
Scanned by CamScanner

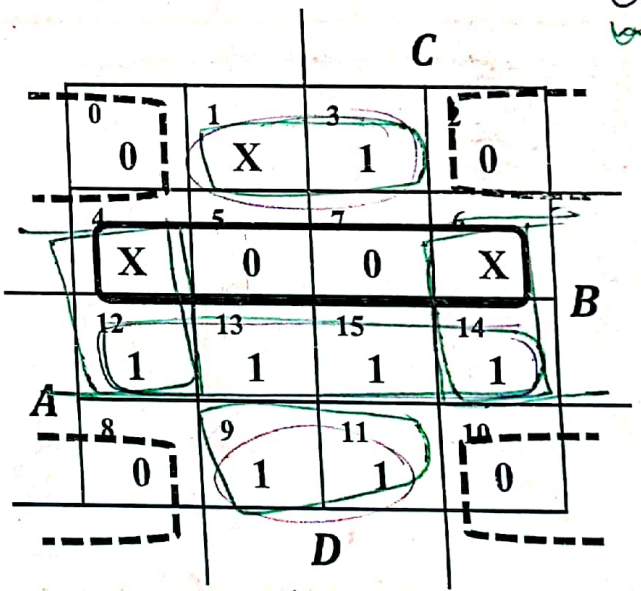
Product of Sums with Don't Care Example

- Find the optimum **POS** solution for:

$$F(A,B,C,D) = \sum_m (3,9,11,12,13,14,15) + \sum_d (1,4,6)$$

→ don't cares (X)





$F_{SOP} = AB + \bar{B}D$ → (ones)

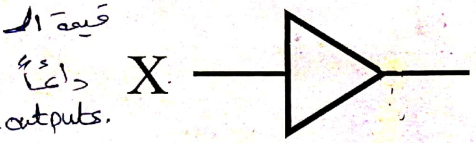
$$\bar{F}(A,B,C,D) = \bar{A}B + \bar{B}\bar{D}$$

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

Scanned by CamScanner

Buffer

- A ***buffer*** is a gate with the function $F = X$:



X	F
0	0
1	1

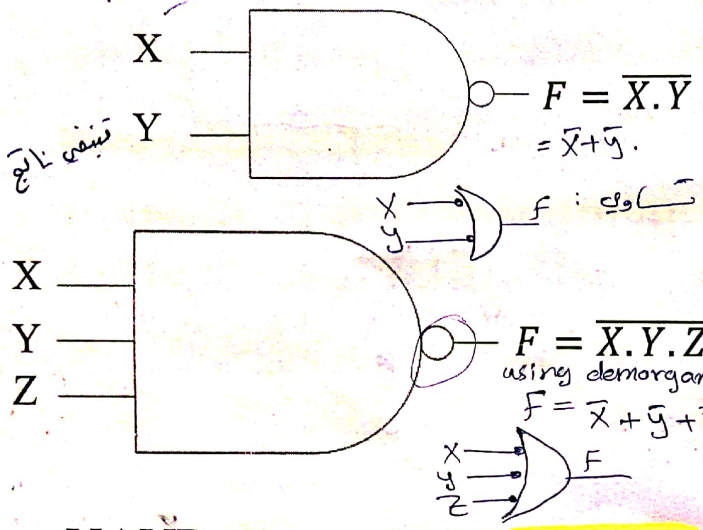
- In terms of Boolean function, a buffer is the same as a connection!
- **So why use it?**
  - A buffer is an electronic amplifier used to improve circuit voltage levels and increase the speed of circuit operation
  - Protection and isolation between circuits

© 2008 Pearson Education, Inc.

Scanned by CamScanner

NAND Gate

- The NAND gate has the following symbol and truth table:



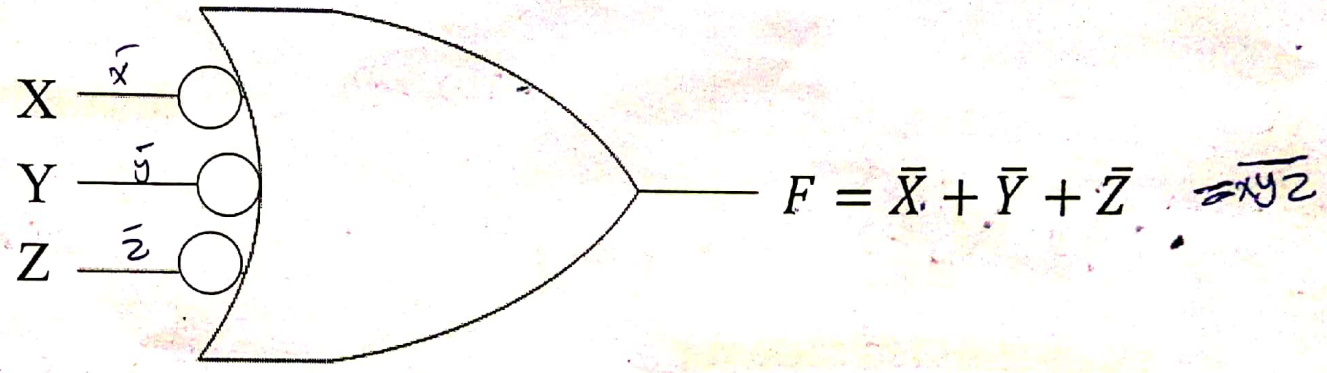
X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0

- **NAND** represents **NOT-AND**, i.e., the AND function with a NOT applied. The symbol shown is an **AND-Invert**. The small circle ("bubble") represents the invert function

Scanned by CamScanner

NAND Gates (continued)

- Applying DeMorgan's Law gives **Invert-OR** (NAND)



- This NAND symbol is called **Invert-OR**, since inputs are inverted and then ORed together

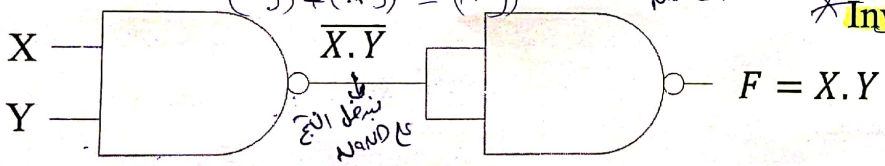
- **AND-Invert** and **Invert-OR** both represent the NAND gate. Having both makes visualization of circuit function easier

© 2008 Pearson Education, Inc.

Scanned by CamScanner

NAND Gates (continued)

- ***Universal gate***: a gate type that can implement any Boolean function. **The NAND gate is a universal gate:**



AND using NAND

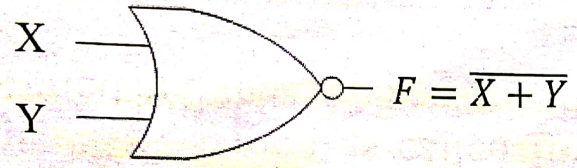
OR using NAND

Inverter using NAND

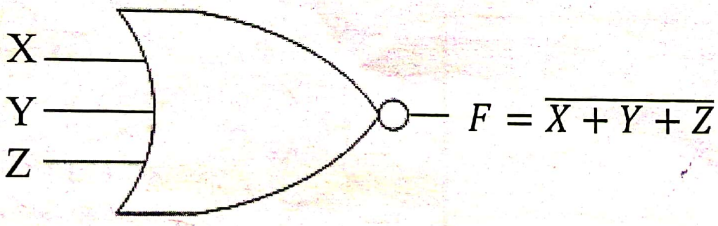
Scanned by CamScanner

NOR Gate

- The NOR gate has the following symbol and truth table:



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



- **NOR** represents **NOT-OR**, i.e., the OR function with a NOT applied. The symbol shown is an **OR-Invert**. The small circle ("bubble") represents the invert function

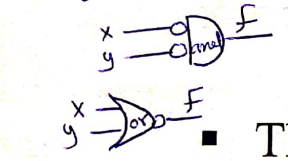
© 2008 Pearson Education, Inc.

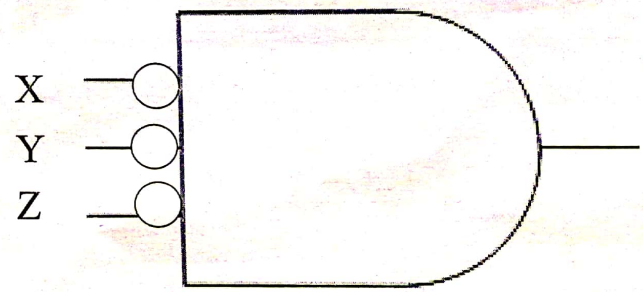
Scanned by CamScanner

NOR Gates (continued)

- Applying DeMorgan's Law gives **Invert-AND** (NOR)

* عندما نغيّر مكان ال inverter من ال input إلى ال output نعكس نوع ال gate.





$F = \overline{X+Y+Z} \rightarrow \bar{X}.\bar{Y}.\bar{Z}$

$F = \bar{X}.\bar{Y}.\bar{Z}$

- This NOR symbol is called **Invert-AND**, since inputs are inverted and then ANDed together

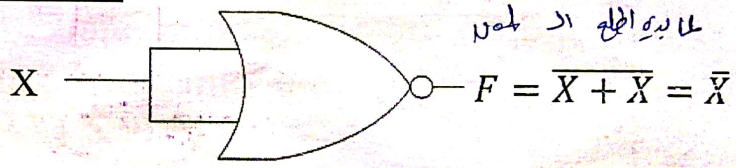
- **OR-Invert** and **Invert-AND** both represent the NOR gate. Having both makes visualization of circuit function easier

تصوّر

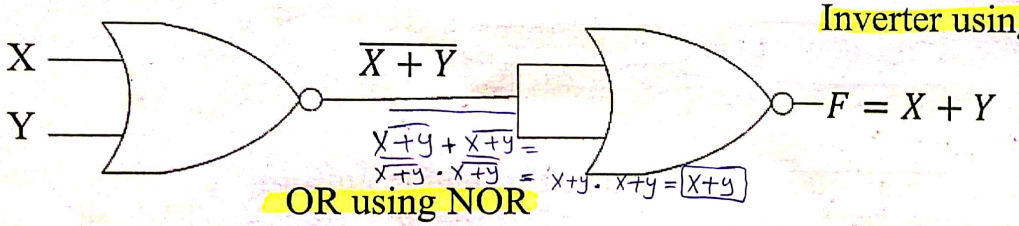
Scanned by CamScanner

NOR Gates (continued)

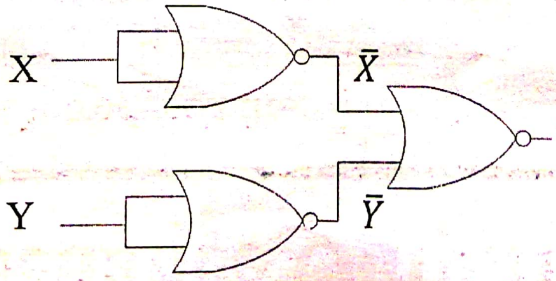
- **The NOR gate is a universal gate:**



Inverter using NOR



OR using NOR



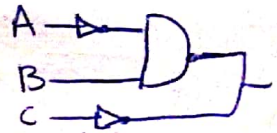
AND using NOR

© 2008 Pearson Education, Inc.

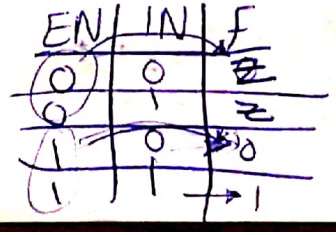
Scanned by CamScanner

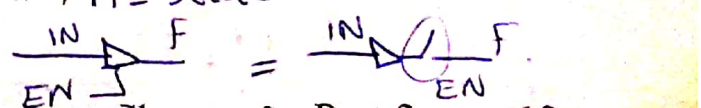
Hi-Impedance Outputs

- Logic gates introduced thus far
  - have 1 and 0 output values,
  - cannot have their outputs connected together, and
  - transmit signals on connections in only one direction



- Three-state logic adds a third logic value, **Hi-Impedance (Hi-Z)**, giving three states: 0, 1, and Hi-Z on the outputs.
- ***Hi-Z can be also denoted as Z or z***
- The presence of a Hi-Z state makes a gate output as described above behave quite differently:
  - "1 and 0" become "1, 0, and Hi-Z"
  - "cannot" becomes "can," and
  - "only one" becomes "two"



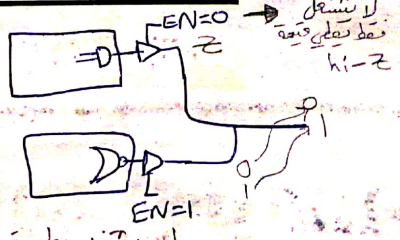


Scanned by CamScanner

Hi-Impedance Outputs (continued)

- **What is a Hi-Z value?**
  - The Hi-Z value behaves as an open circuit
  - This means that, looking back into the circuit, the output appears to be disconnected
  - It is as if a switch (مفتاح) between the internal circuitry and the output has been opened

- Hi-Z may appear on the output of any gate, but we restrict gates to ***3-state buffer***

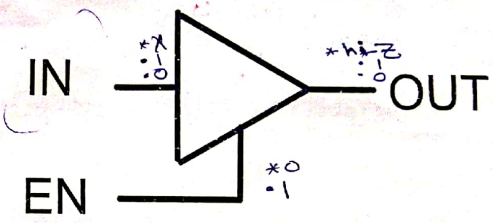


Scanned by CamScanner

Tri-State Buffer (3-State Buffer)

- For the symbol and truth table, **IN** (input) is the **data input**, and **EN** is the **control input** (Enable)
- For **EN = 0**, regardless of the value on IN (denoted by X), the output value is Hi-Z
- For **EN = 1**, the output value follows the input value

Symbol



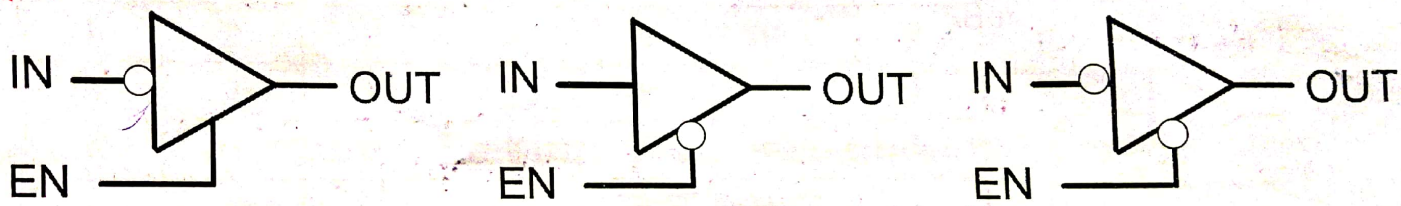
Truth Table

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

Scanned by CamScanner

Tri-State Buffer Variations

- By adding "bubbles" to signals:
  - Data input, IN, can be inverted (by bubbles)
  - Control input, EN, can be inverted (by bubble).



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

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

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

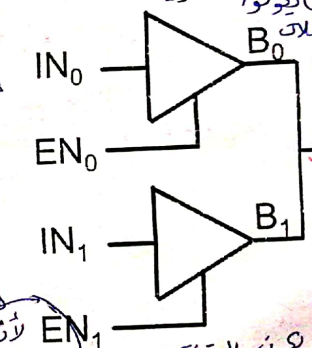
© 2008 Pearson Education, Inc.

Scanned by CamScanner

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

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	Error
1	1	1	0	1	0	Error



Scanned by CamScanner

Resolving 3-State Values on a Connection

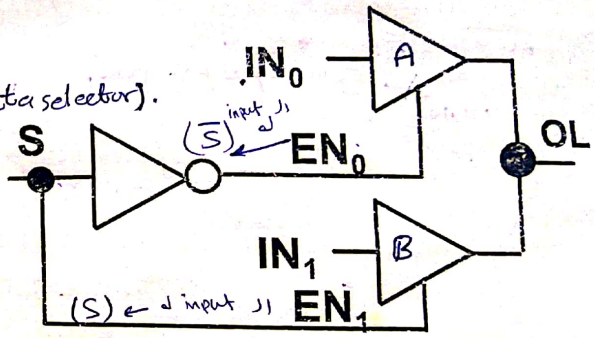
- **Resulting Rule: At least one buffer output value must be Hi-Z. Why?**
  - Because any data combinations including (0,1) and (1,0) can occur. If one of these combinations occurs, and no buffers are Hi-Z, then high currents can occur, destroying or damaging the circuit
- **How many valid buffer output combinations exist?**
  - 5 valid output combination
- **What is the rule for "*n*" tri-state buffers connected to wire, OL?**
  - 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.

Scanned by CamScanner

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	0	1	X	0	0
0	0	1	X	1	1
1	1	0	0	X	0
1	1	0	1	X	1



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

Logic Functions using Tri-State Buffers

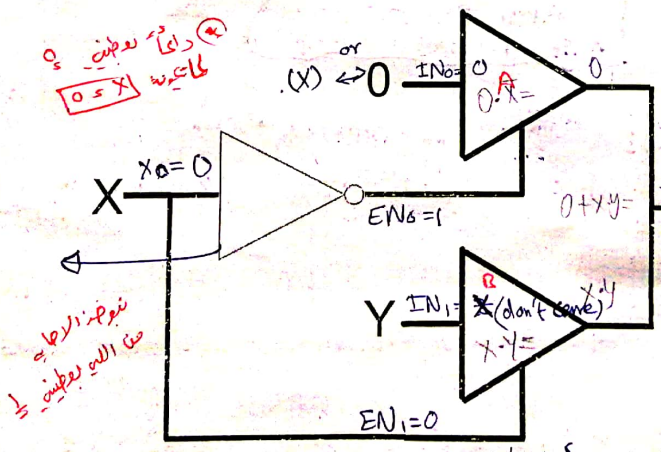
- Implement AND gate using 3-State buffers and inverters

Tri state buffers + inverters to implement any gate we want.

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

- Use X as control input: (* 2-input AND gate) *this implementation is:
  - When $X = 0$, $F = 0$ regardless of the value of Y
  - When $X = 1$, $F = Y$ → or $F = X$

X	Y	F	
0	0	0	F = constant. F = 0 أو A
0	1	0	F = X
1	0	0	F = y أو B
1	1	1	F ≠ constant



* ملاحظة: دائماً أن الـ outputs مشبوكين معاً في سلك واحد لابد

كاملة ← يجوز بدل ⊳ دائرة تحتوي على bubble (o) وضع على buffer بحيث لا يلزم كتابة إشارة الـ inverter كاملة بل فقط الـ bubble على الـ buffer

دائماً تحتوي على X = 0 كالكتابة

نوع الإخراج من اللي يحتوي ؟

* الخطوات لتمثيل أي gate باستخدام inverters + Tri-state buffer

① نرسم الـ (Truth table) للـ (function) المراد تمثيله

② نقسم الـ (truth table) من اليمين بناء على الـ most significant variable (X)

③ بعد ما نقسم الجدول نحاول نطلع الـ function بدلالة X أو Y أو constant (0 أو 1)

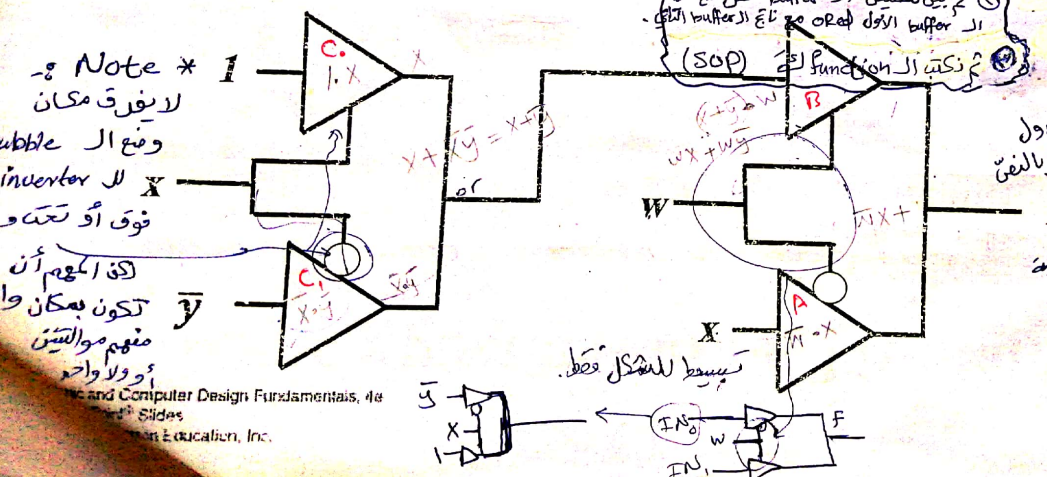
©2008 Pearson Education, Inc.

Scanned by CamScanner

Logic Functions using Tri-State Buffers

- Implement the following function using 3-State buffers and inverters: $F(w,x,y) = \bar{w}x + w\bar{y} + xy$
- Use w as control input:
  - When $w = 0$, $F = x$ regardless of the value of Y
  - When $w = 1$
    - If $x = 0$, $F = \bar{y}$
    - 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



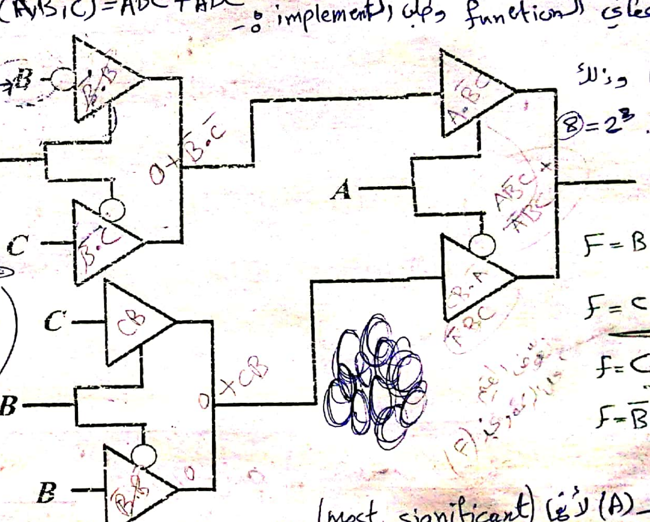
Scanned by CamScanner

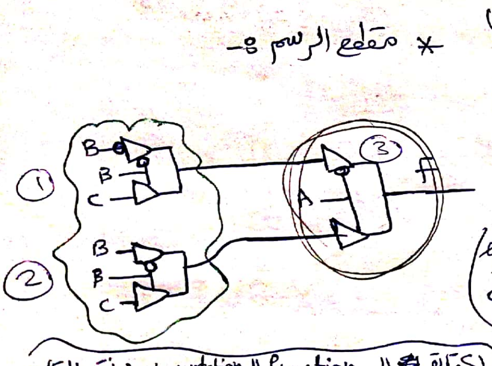
Logic Functions using Tri-State Buffers

* مهم جدًا أكيد جاي عليه سؤال !

- Write the Boolean expression of $F(A,B,C)$ given the diagram below: $F(A,B,C)=A\bar{B}C+\bar{A}BC$

* لحل هذا السؤال إذا أعطاني الـ function وطلب الـ implement :





① نرسم الـ (truth table) وذلك : 2^n : n : عدد المتغيرات . $(8)=2^3$

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

F = B, F = C, F = C, F = B̄

* لكتابة الـ function للـ implementation نتبع التالي :
① $\bar{B}\cdot B$ ، $\bar{B}\cdot C$ → $B\bar{B}+\bar{B}.C = \bar{B}.C$
② $C.B$ ، $B.\bar{B}$ → $C.B + B.\bar{B}(0) = C.B$
③ $A.\bar{B}.C + \bar{A}.B.C$ وهو المطلوب .

② نقسم الجدول بالنصف بالنسبة للـ (A) لأنها (most significant variable)

نحاول كتابة النصف العلوي بدلالة F = X أو F = B أو F = B̄ أو F = 0 و V ولكنه لا يرتبط لذلك نقسم الجدول مرة أخرى بالنسبة لـ (B) لأنها the next most significant variable .

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

③ نبدأ الرسمة من الـ output (F) ثم نرجع للوراء ونرسم (2-buffers) ونضع (A) هي الـ (ENable) لها ووحدة منهم عليها (bubble) .

نبدأ نحدد الـ (inputs) للـ (buffers) وهم بناءً على (ENable (B)) كالتالي نرسم (2 buffers) لكل (input) .

يكون (B) هو الـ (ENable) لهما ونحدد الـ (inputs) حسب ما تم كتابة الـ function بدلالة : أي $\bar{B} = F$ وهكذا

The complex gatess

Exclusive OR/ Exclusive NOR

- The ***eXclusive OR (XOR)*** function is an important Boolean function used extensively in logic circuits
- The XOR function may be:
  - implemented directly as an electronic circuit (truly a gate) or
  - implemented by interconnecting other gate types (used as a convenient representation)
- The ***eXclusive NOR (XNOR)*** function is the complement of the XOR function
- By our definition, XOR and XNOR gates are ***complex gates***

Scanned by CamScanner

Exclusive OR/ Exclusive NOR

- Uses for the XOR and XNORs gate include:
  - Adders/subtractors/multipliers
  - Counters/incrementers/decrementers
  - Parity generators/checkers

- Definitions
  - The XOR function is: $X \oplus Y = \bar{X}Y + X\bar{Y}$
  - The XNOR function is: $X \odot Y = \overline{X \oplus Y} = XY + \bar{X}\bar{Y}$

- Strictly speaking, XOR and XNOR gates ***do no exist for more than two inputs***. Instead, they are replaced by odd and even functions

© 2008 Pearson Education, Inc.

Scanned by CamScanner

Proof: XNOR is the complement of XOR

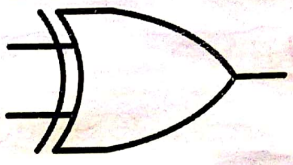
- $\overline{X \oplus Y} = \overline{\overline{X}Y \oplus X\overline{Y}}$ (معقدة)(XOR) (بسيطة)(OR)
- $\overline{X \oplus Y} = \overline{\overline{X}Y} \cdot \overline{X\overline{Y}}$
- $\overline{X \oplus Y} = (X + \overline{Y})(\overline{X} + Y)$ distributive law.
- $\overline{X \oplus Y} = X\overline{X} + XY + \overline{X}\overline{Y} + Y\overline{Y}$ (0) (0)
- $X \odot Y = \overline{X \oplus Y} = XY + \overline{X}\overline{Y}$

* انه لا يجوز استخدام (demorgan's law) بل يجب تحويل (complex gates) إلى (primitive gates)

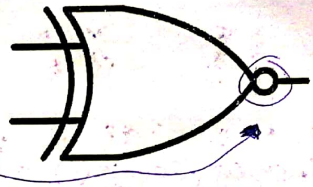
Scanned by CamScanner

Symbols For XOR and XNOR

- XOR symbol:



- XNOR symbol:

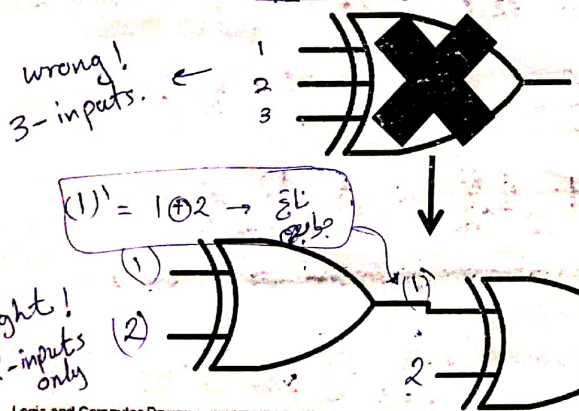


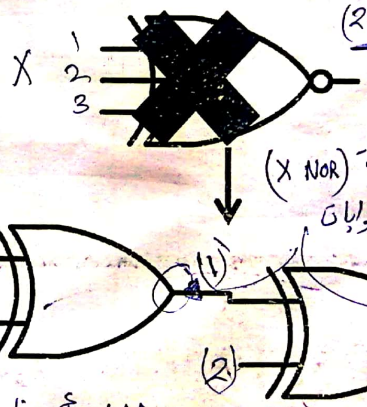
لا يسمح بأكثر من 2 مدخل

- ***Shaped symbols exist only for two inputs***

* مهم جداً : فقط

يسمح لهم بـ (2-IN)





* (انتبه!: عند عمل بوابة (X NOR) لأكثر من (2-inputs) نعمل بوابات (XOR) قبل آخر بوابة وفقط البوابة الأخيرة تكون (X NOR) والباقي (XOR) كما تم تمثيله

* إذا أردنا زيادة عدد الـ (inputs) عن (two) نقوم بإضافة (gates) أخرى.

© 2008 Pearson Education, Inc.

Scanned by CamScanner

Truth Tables for XOR/XNOR

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

X	Y	$X \odot Y (X \equiv Y)$
0	0	1
0	1	0
1	0	0
1	1	1

equation الـ من $(\bar{x}y + x\bar{y})$

(1) Function الـ ناتج الـ لكون مختلفين (x,y) يكونوا أن يجب *

complement دليل أنهم to each other.

(1) الناتج الـ لكون متشابهين (x,y) يكونوا أن يجب *

equation الـ من $(\bar{x}\bar{y} + xy)$.

- The XOR function means: ***X OR Y, but NOT BOTH***

- Why is the XNOR function also known as the ***equivalence*** function, denoted by the operator ≡?
  - Because the function equals 1 if and only if $\boldsymbol{X = Y}$

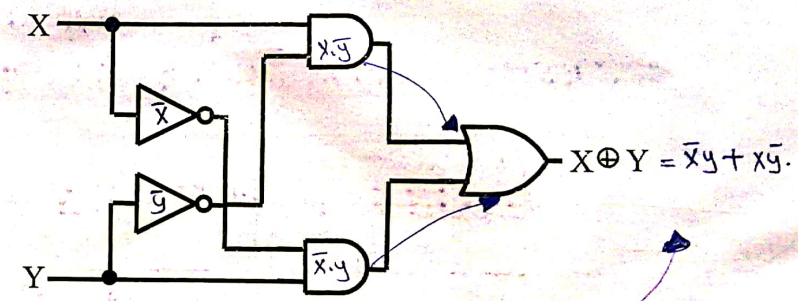
(XNOR Gate)

لأنها تتأكد من أن (x,y) قيمتين متساويتين.

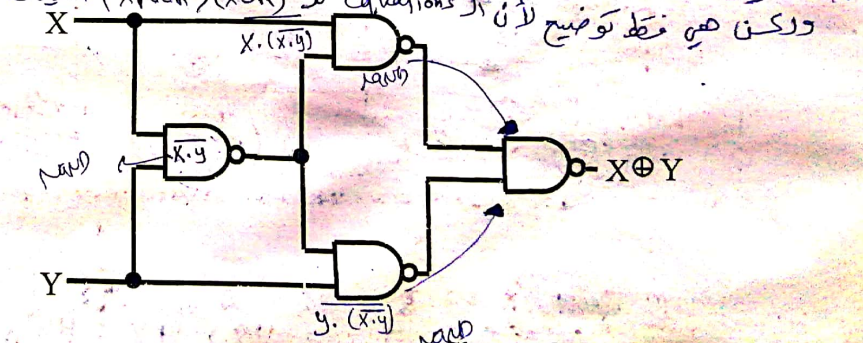
Scanned by CamScanner

XOR Implementations

- **The simple SOP implementation uses the following structure:**



- **A NAND only implementation is:**



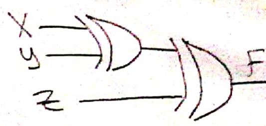
© 2008 Pearson Education, Inc.

Scanned by CamScanner

XOR

- The XOR identities:

$X \oplus 0 = X$	$X \oplus 1 = \overline{X}$
$X \oplus X = 0$	$X \oplus \overline{X} = 1$
$X \oplus \overline{Y} = \overline{X \oplus Y}$	$\overline{X} \oplus Y = \overline{X \oplus Y}$
$X \oplus Y = Y \oplus X$	
$(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z) = X \oplus Y \oplus Z$	



- The XOR function can be extended to 3 or more variables. For more than 2 variables, it is called an ***odd function*** or ***modulo 2 sum*** (***Mod 2 sum***), not an XOR:

$$X \oplus Y \oplus Z = \overline{X}\,\overline{Y}Z + \overline{X}Y\overline{Z} + X\overline{Y}\,\overline{Z} + XYZ \text{ (Odd \# of 1's)}$$

Scanned by CamScanner

XNOR

* ليس شرط فقط هذه الـ (Identites) حتى لو عوّضنا القيم (X,Y) بالامتحان نستطيع أن نجد الجواب بسهولة .

- The XNOR identities:

(1) $X \odot 0 = \bar{X}$	(2) $X \odot 1 = X$
(3) $X \odot X = 1$	(4) $X \odot \bar{X} = 0$
$X \odot Y = Y \odot X$ (خاصية تبديلية)	
$X \odot Y \odot Z = (X \oplus Y) \odot Z = X \odot (Y \oplus Z)$ (خاصية تجميعية)	

(1) 1⊙0 = 0 ← مختلفين
0⊙0 = 1 ← متشابهين

(2) 1⊙1 = 1 ← متشابهين
0⊙1 = 0 ← مختلفين

(3) 0⊙0 = 1
1⊙1 = 1
دائماً متشابهين إذاً الناتج دائماً (1)

1⊙0 = 0
0⊙1 = 0
دائماً مختلفين إذاً الناتج دائماً (0)

- The XNOR function can be extended to 3 or more variables. For more than 2 variables, it is called an ***even function***, not an XNOR:

* إذا كان عدد الـ (ones) في الـ (input) تبعي يكون ناتج الـ function (0) في (XOR) و (1) في (XNOR)

$$X \odot Y \odot Z = \bar{X}\bar{Y}\bar{Z} + X\bar{Y}Z + XY\bar{Z} + XYZ \quad \text{(Even \# of 1's)}$$

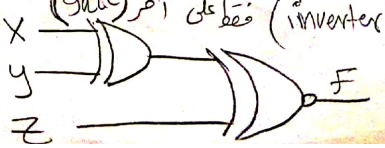
(0 1 1) + (1 0 1) + (1 1 0) + (0 0 0)

* ملاحظة: عندما يكون الحدّ (minterm):
000 يكون عدد الـ (1's) زوجي لأنه لا يوجد (1's) إذاً Number of ones = even number of ones.

- ***The even function is the complement of the odd function***

Reason:

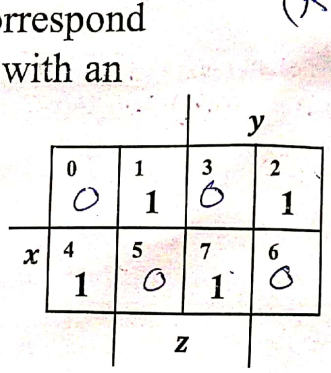
* انتبه: نضع (inverter) فقط على آخر (gate)



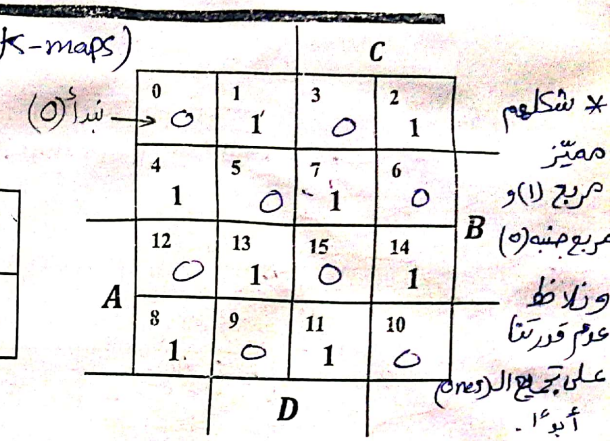
Scanned by CamScanner

Odd and Even Functions

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

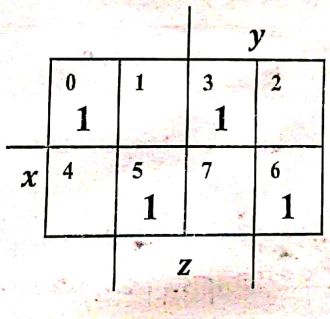


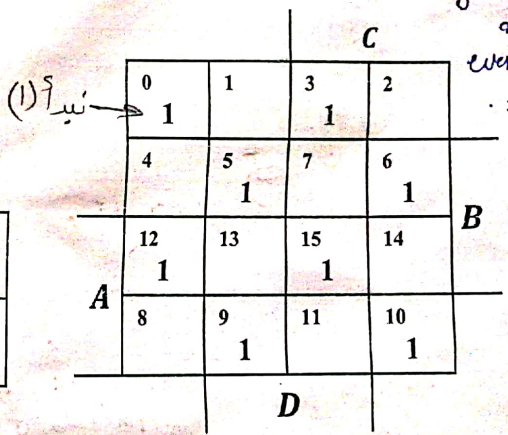
(K-maps)



* شكلهم مميّز مربع (1) و مربع جنبه (0) و ولاحظ عدم قدرتنا على تجميع (ones) أبدًا. ولكن من الشكل نعرف أنه even / odd function.

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

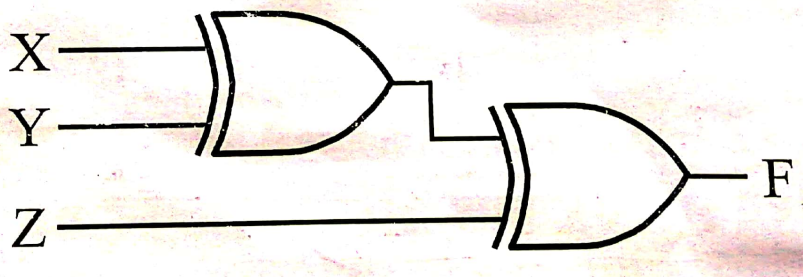




Scanned by CamScanner

Example: Odd Function Implementation

- Design a 3-input odd function $F = X \oplus Y \oplus Z$ with 2-input XOR gates
- Factoring, $F = (X \oplus Y) \oplus Z$
- The circuit:

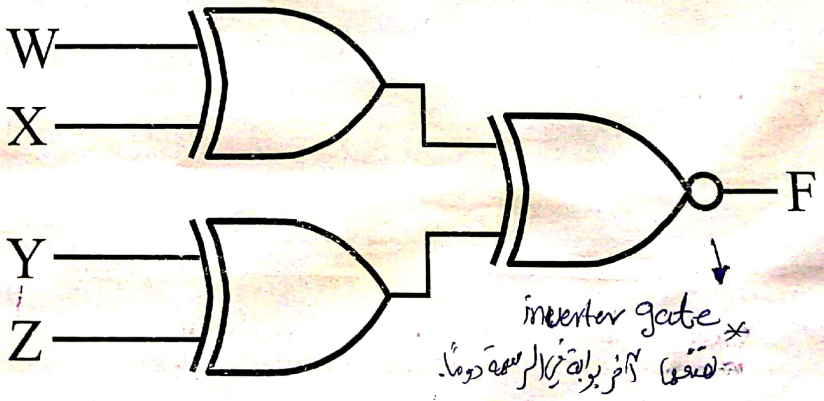


© 2008 Pearson Education, Inc.

Scanned by CamScanner

Example: Even Function Implementation

- **Design 4-input even function $\overline{F = W \oplus X \oplus Y \oplus Z}$ with 2-input XOR and XNOR gates**
- **Factoring, $F = \overline{(W \oplus X) \oplus (Y \oplus Z)}$**
- **The circuit:**



Scanned by CamScanner

Overview

- **Part 1 – Design Procedure**
  - **Steps**
    - **Specification**
    - **Formulation**
    - **Optimization**
    - **Technology Mapping**
    - **Verification**

* نريد استخدام التالي في تصميم دوائر كهربائية تقوم بوظيفة معينة.

- **Technology Mapping** using: **AND, OR, and NOT to NAND or NOR**

© 2008 Pearson Education, Inc.

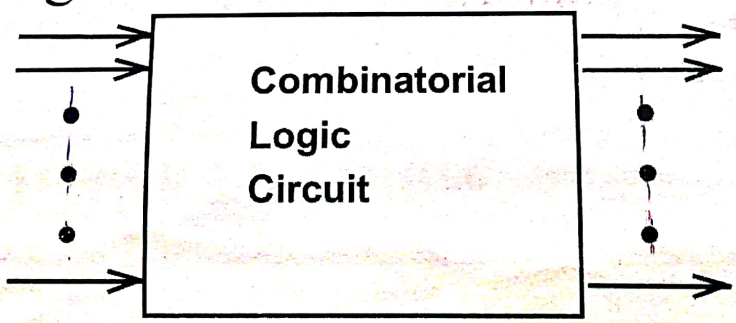
Scanned by CamScanner

Combinational Circuits (doesn't have memory).

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

* يكون هناك أكثر من input وبالتالي أكثر من output ولكن كل output يعتمد على ال input الذي يخصه فقط.

- A block diagram:



Scanned by CamScanner

Design Procedure

خطوات لكي نقلل من احتمال الخطأ في تصميم الـ (circuit):

1. **Specification** (الوصف)
   - 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** (التمثيل)
   - Derive a ***truth table*** or ***initial Boolean equations*** that define the required relationships between the inputs and outputs, if not in the specification
3. **Optimization** (الاختصار / التقليص)
   - Apply 2-level optimization using K-maps
   - Draw a logic diagram for the resulting circuit using ANDs, ORs, and inverters (basic gates)

Scanned by CamScanner

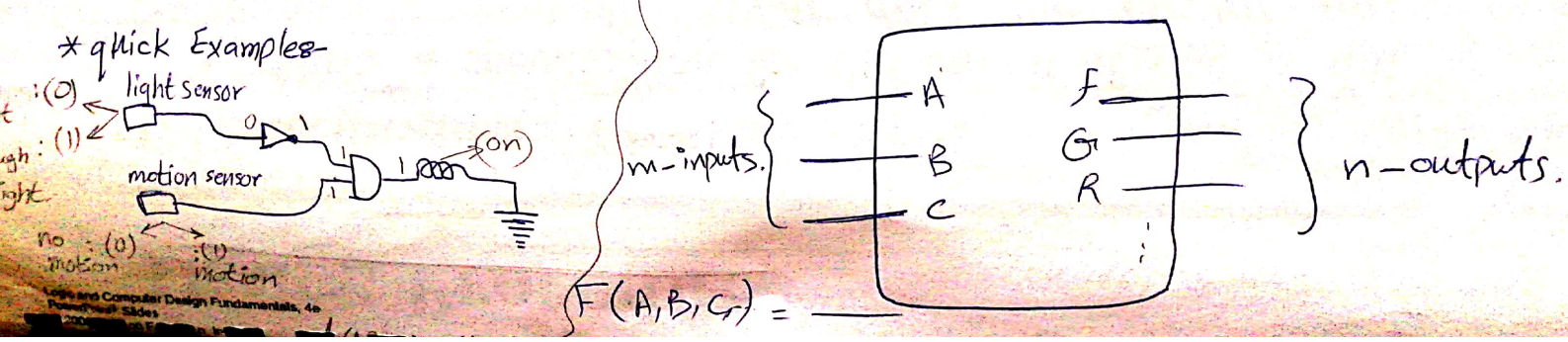
Design Procedure

4. Technology Mapping (التمثيل بالرسم)

- Map the logic diagram to the implementation technology selected

5. Verification (using lab logic).

- Verify the correctness of the final design ***manually*** or using ***simulation*** → (محاكاة)



Scanned by CamScanner

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 $\geq$ 2)
  - This is called ***majority function*** (i.e. majority of inputs must be 1 for the function to be 1)
- **Formulation:**

	X	Y	Z	F
m_0	0	0	0	0
m_1	0	0	1	0
m_2	0	1	0	0
m_3	0	1	1	1
m_4	1	0	0	0
m_5	1	0	1	1
m_6	1	1	0	1
m_7	1	1	1	1

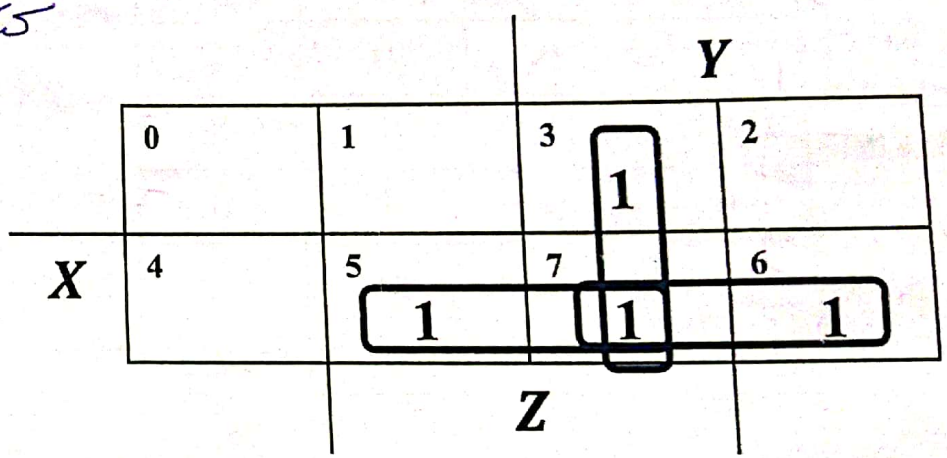
Scanned by CamScanner

Design Example1 Cont.

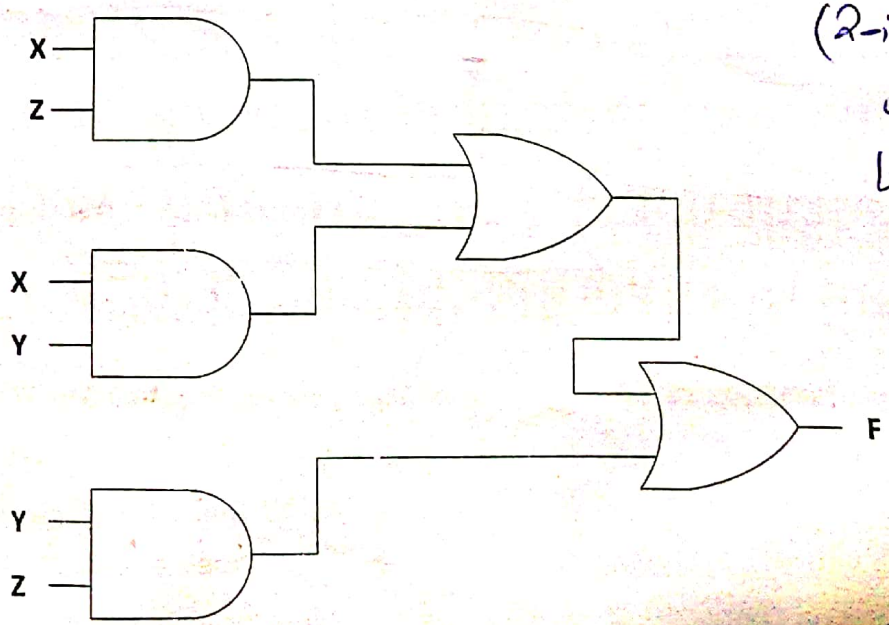
- **Optimization:** (cost) لتقليل ال ← كتابة ال function بأبسط صورة using (K-maps)

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

as a sum of minterms: $\sum m(3,5,6,7)$.



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



* شرط فقط (2-inputs) لذلك نعمل أكثر من بوابة لكي يربط معنا ذلك.

Scanned by CamScanner

Design Example2

* لماذا؟ (2 binary numbers and each number consists of 2 bits.)

- **Specification:** Design a combinational circuit that compares *2-bit* Binary number *(A, B)* and produce two outputs *(O_1, O_0)*, such that: *(one two bits output) decimal لـ: even (0)*

$O_1O_0 = 00$	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

$A(A_1A_0)$	$A=B$	$B(B_1B_0)$	$O(O_1O_0)$
(even) 00	$A<B$	00	00
00	$<$	01	01
00		10	01
00		11	01
01	$A>B$	00	10
(odd) 01	$A=B$	01	11
01		10	01
01		11	01
10		00	10
10		01	10
10		10	00
10		11	01
11		00	10
11		01	10
11		10	10
11		11	11

لكل bit output لـ (kmap) نعمل لـ.

لـ odd decimal (1)

decimal لـ odd (3)

(0,1) ← (kmap) لـ

(2,3) لـ (kmap)

- **Formulation:** (truth table) تعبئة الـ (decimal لـ B أكبر من A)

(decimal لـ A أكبر من B)

Note * نعتبر الجدول كأنه (4-variable inputs) يعني لدخل لأن (2 bits) كمتغير لها ← A نعتبرها متغيرين.

© 2009 Pearson Education, Inc.

Scanned by CamScanner

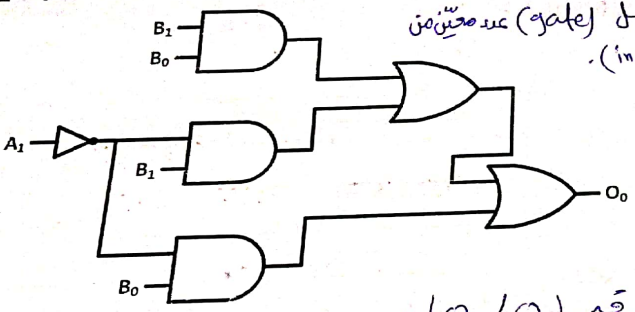
Design Example2 Cont.

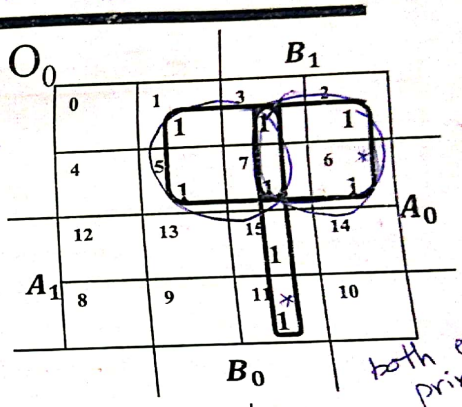
* لكل (output) نعمل (k-map) منفصلة . وحجمها (16 مربع) لأن $2^4=16$. 4: متغيرات على اعتبار B_0, B_1, A_0, A_1 (هي 4 متغيرات).

- **Optimization and Technology Mapping:**

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

* نأخذ عدد أكبر من ال (gates) لأن لكل (gate) عدد معيّن من ال (inputs).

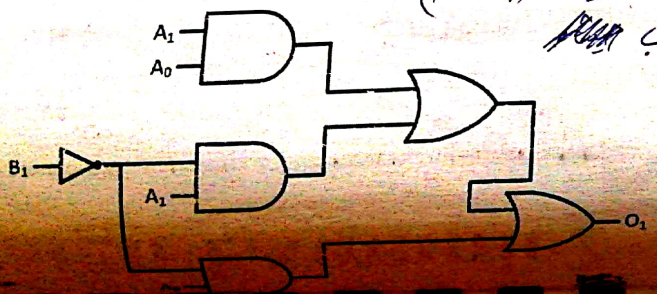


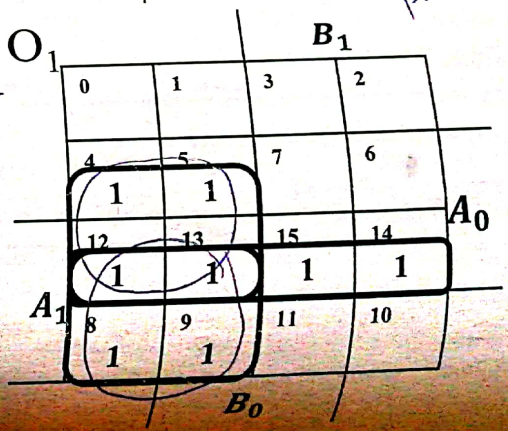


both essential prime implicants

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

* يتم أخذ قيم (O_1 / O_0) من الجدول ووضعها في ال (k-map) حسب الترتيب





Scanned by CamScanner

Design Example3

1. Specification

- ***BCD to Excess-3 code converter*** (Excess 3) ← (BCD) تحويل من (circuit) اعمل
  (4-bits) (4-bits) (+3)
- 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 BCD : (0 → 9)
- Excess-3 code words for digits 0 through 9: 4-bit patterns consisting of 3 (binary 0011) added to each BCD code word (12) هي (Excess3) الـ كود تكون فيه. حيث (9) تغطي لغاية
- ***BCD input is labeled A, B, C, D***
- ***Excess-3 output is labeled W, X, Y, Z***

Scanned by CamScanner

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

don't care.

* لم يخبرني بالسؤال ماذا نفعل
بال (illegal outputs) وهم من (10 ← 15) حيث
(BCD) أنهم غير موجودين بال
لذلك من (0 ← 9) فقط.

Scanned by CamScanner

Design Example3 Cont.

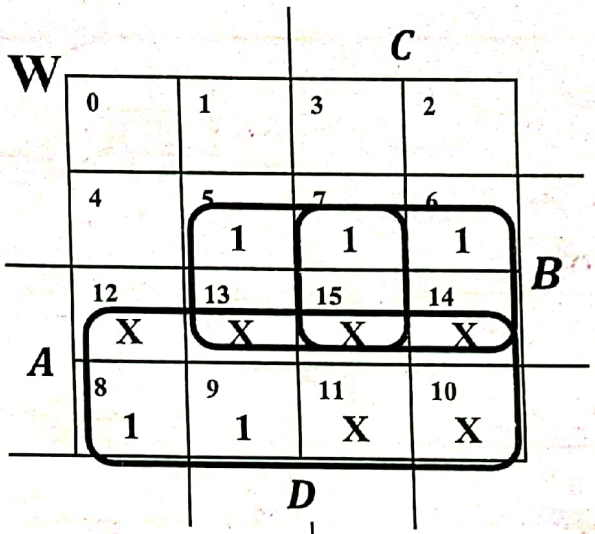
3. Optimization

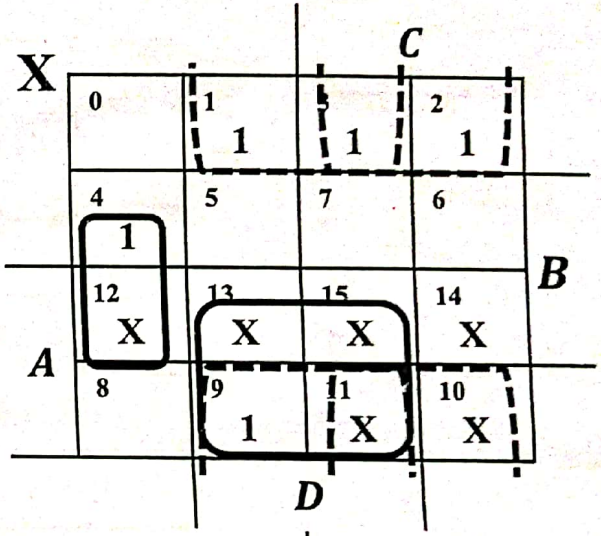
$W = A + BC + BD$

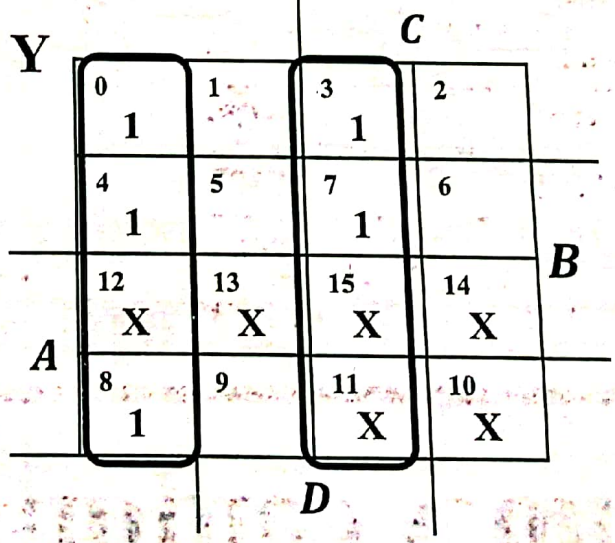
$X = \bar{B}D + \bar{B}C + B\bar{C}\bar{D}$

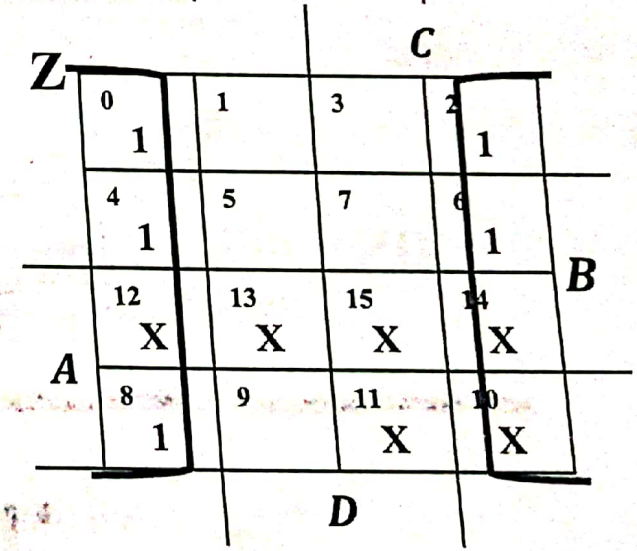
$Y = \bar{C}\bar{D} + CD$

$Z = \bar{D}$









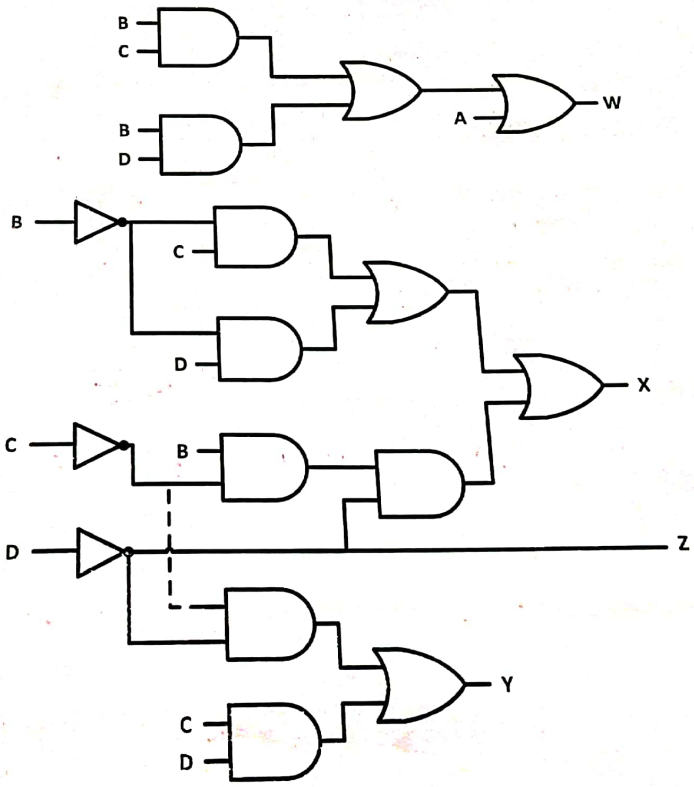
© 2008 Pearson Education, Inc.

Scanned by CamScanner

Design Example3 Cont.

4. Technology Mapping

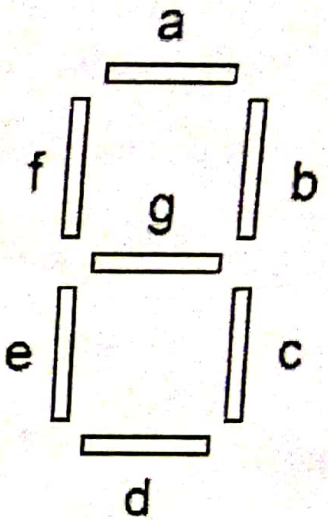
- **Mapping with a library containing inverters, 2-input AND, 2-input OR**



Scanned by CamScanner

Homework: BCD to 7-Segment

- **Specification:**
  - **Inputs:** ***(A, B, C, D)*** BCD code from 0000-to-1001
  - **Outputs:** ***(g, f, e, d, c, b, a)***



- **Formulation:**

ABCD	*gfedcba*
(0) *0000*	*0111111*
(1) *0001*	*0000110*
(2)	
1001	*1100111*
1010	*0000000*
1111	*0000000*

- **Optimization:**
  - **How many K-maps?**

Logic and Computer Design Fundamentals, 4e
PowerPoint® Slides
© 2008 Pearson Education, Inc.

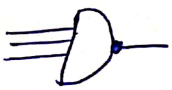
Scanned by CamScanner

Technology Mapping

- **Mapping Procedures**
  - **To NAND gates**
  - **To NOR gates**

* better to make a circuit only with one gate type. (using one type universal gate).

* NAND Gate :-



⊛ NOT :-

$\overline{A \cdot A} = \overline{A} = A$ — $\overline{A}$

⊛ AND :-

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

A, B — $= A \cdot B$ — $\overline{A \cdot B}$

⊛ OR :-

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

$A + B$

A, B

* NOR Gate :-

⊛ NOT :-

$\overline{B + B} = \overline{B} = B$ — $\overline{B}$

⊛ OR :-

$\overline{A + B}$

A, B — $= A + B$ — $A + B$.

⊛ AND :-

A, B — $A \cdot B$

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

Scanned by CamScanner

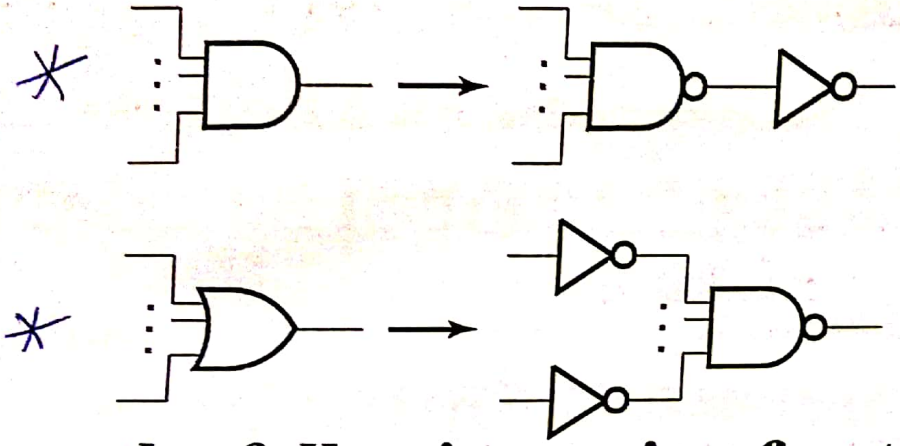
Mapping to NAND gates

- **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

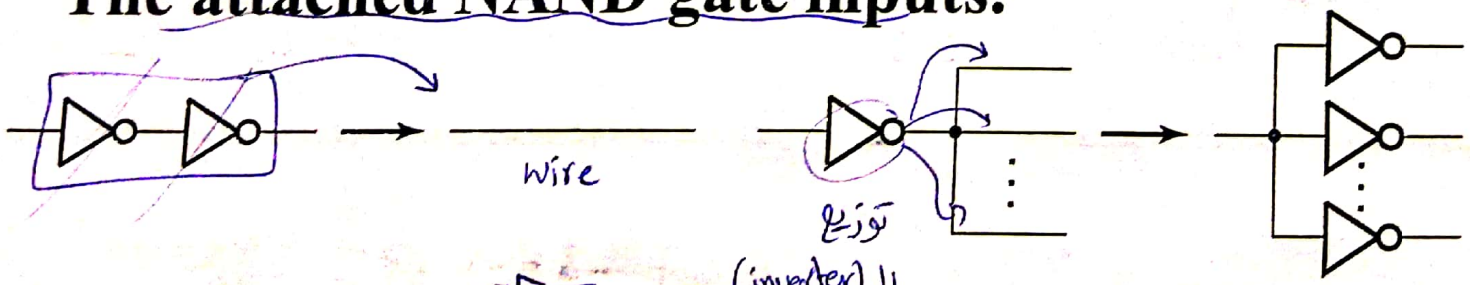
Scanned by CamScanner

NAND Mapping Algorithm

1. **Replace ANDs and ORs:**

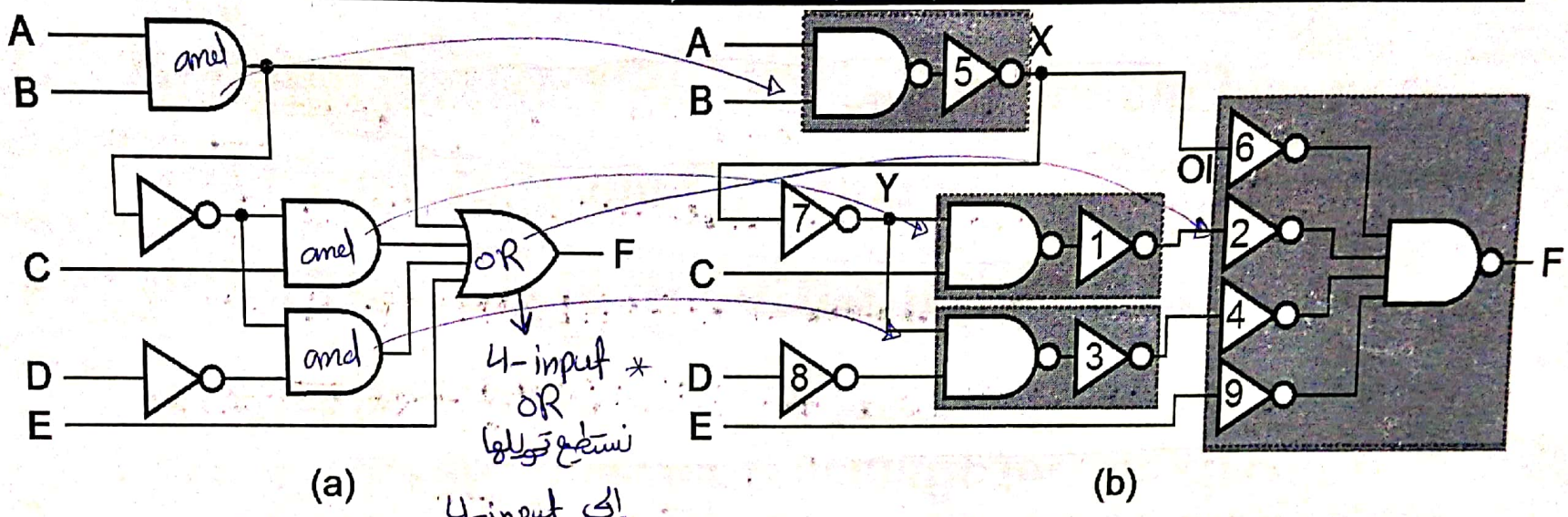


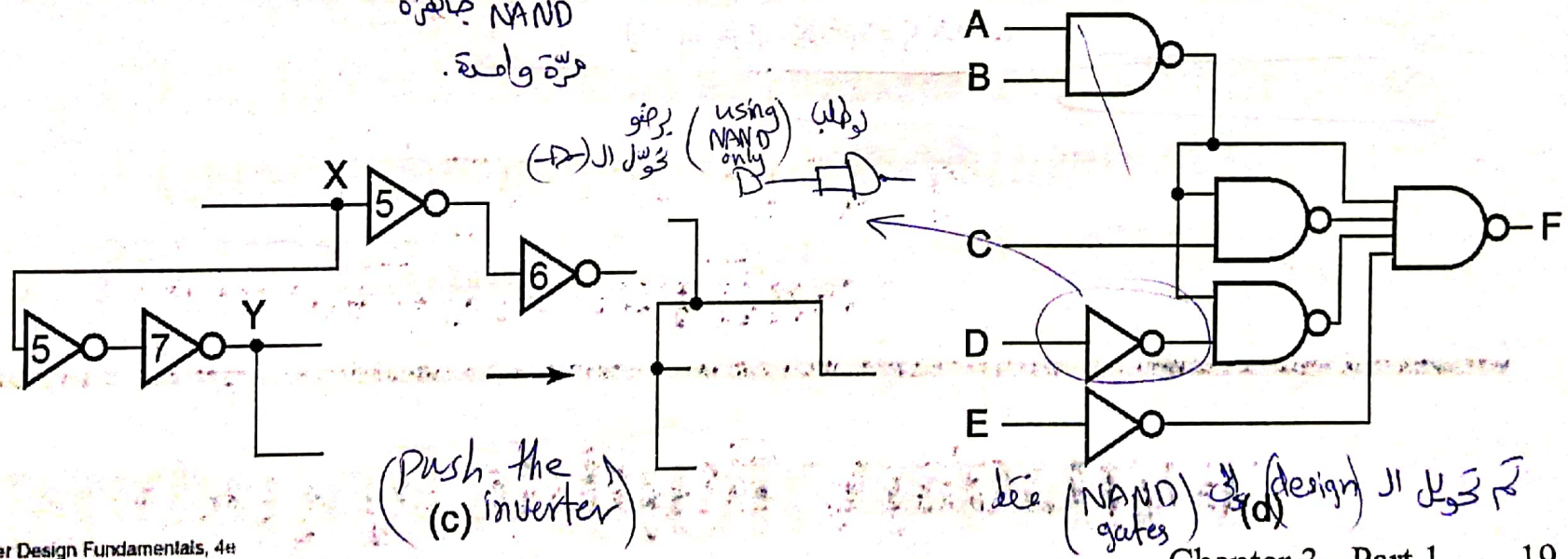
2. **Repeat the following pair of actions until there is at most one inverter between :**
   a. **A circuit input or driving NAND gate output, and**
   b. **The attached NAND gate inputs.**



Scanned by CamScanner

NAND Mapping Example





*to reduce the number of inverters.

© 2008 Pearson Education, Inc.

Scanned by CamScanner

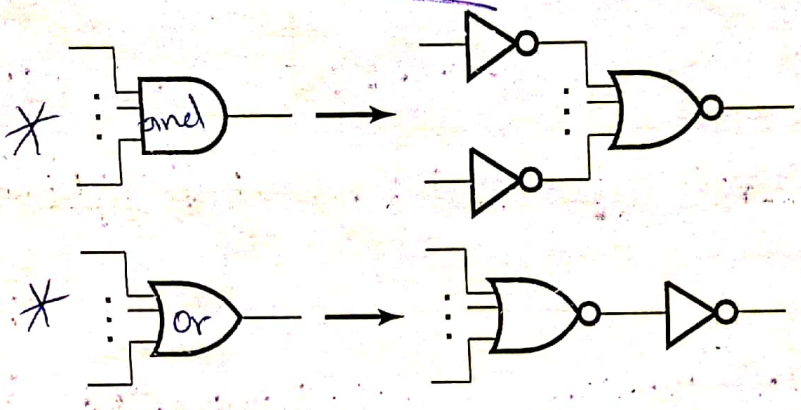
Mapping to NOR gates (نفس الـ NAND)

- **Assumptions:**
  - Gate loading and delay are ignored
  - Cell library contains an inverter and *n*-input NOR gates, *n* = 2, 3, … (لا يوجد داعي لتوفير الـ inverter فقط الـ AND/OR gates)
  - 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

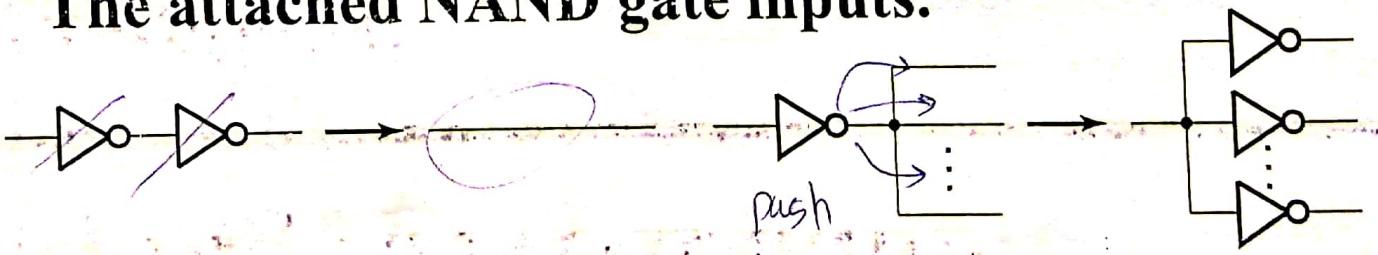
Scanned by CamScanner

NOR Mapping Algorithm

1. **Replace ANDs and ORs:**

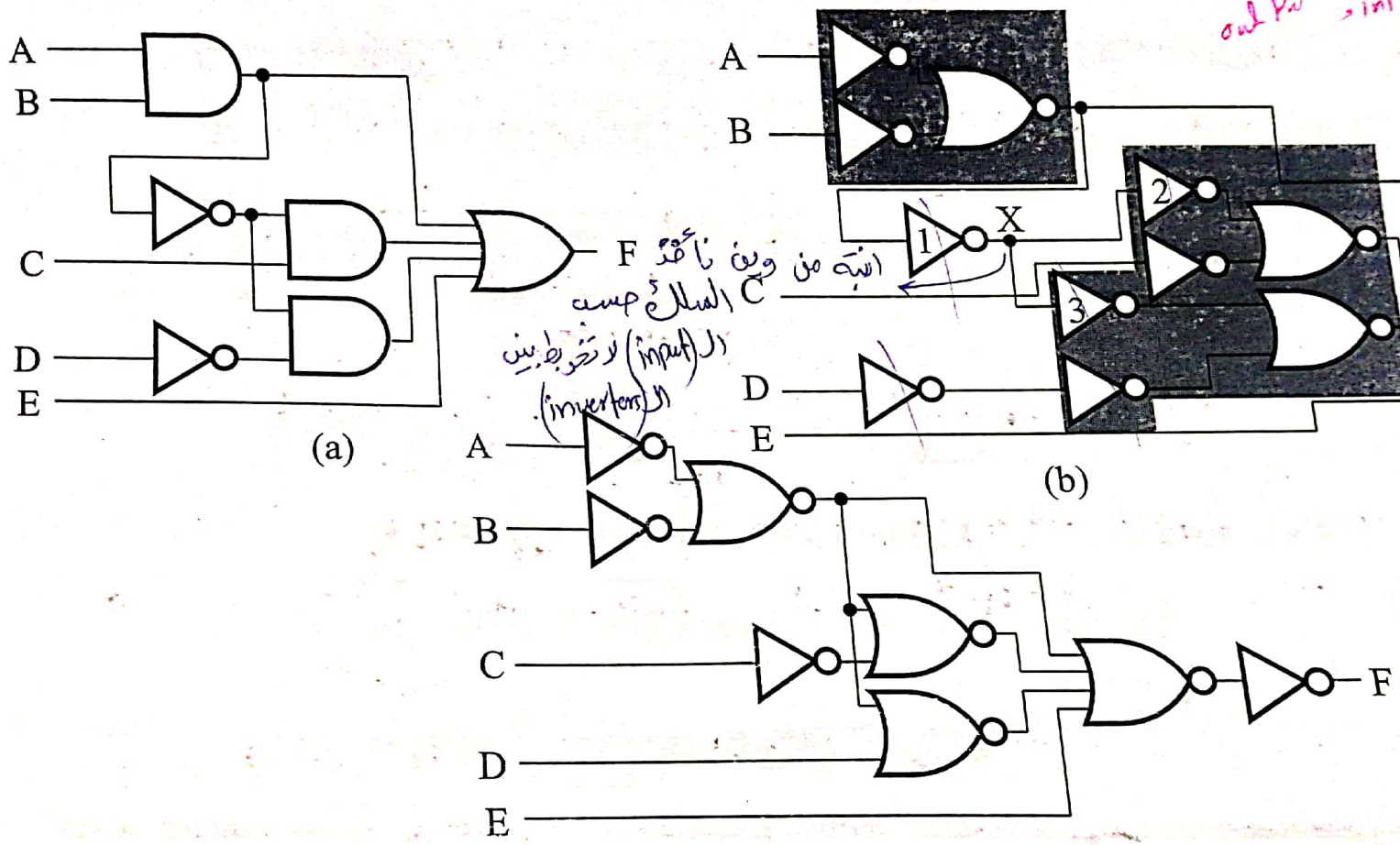


2. **Repeat the following pair of actions until there is at most one inverter between :**
   a. **A circuit input or driving NAND gate output, and**
   b. **The attached NAND gate inputs.**



Scanned by CamScanner

NOR Mapping Example



Scanned by CamScanner

Terms of Use

- All (or portions) of this material © 2008 by Pearson Education, Inc.
- Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.
- These materials or adaptations thereof are not to be sold or otherwise offered for consideration.
- This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.

Scanned by CamScanner

Overview

- **Part 2 – Combinational Logic** (without memory). input gives output.
  - Functions and functional blocks
  - Rudimentary logic functions
  - Decoding using Decoders (blockes).
    - Implementing Combinational Functions with Decoders
  - Encoding using Encoders (blockes)
  - Selecting using Multiplexers
    - Implementing Combinational Functions with Multiplexers

Scanned by CamScanner

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

Scanned by CamScanner

Rudimentary Logic Functions

Basic Functions :-

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

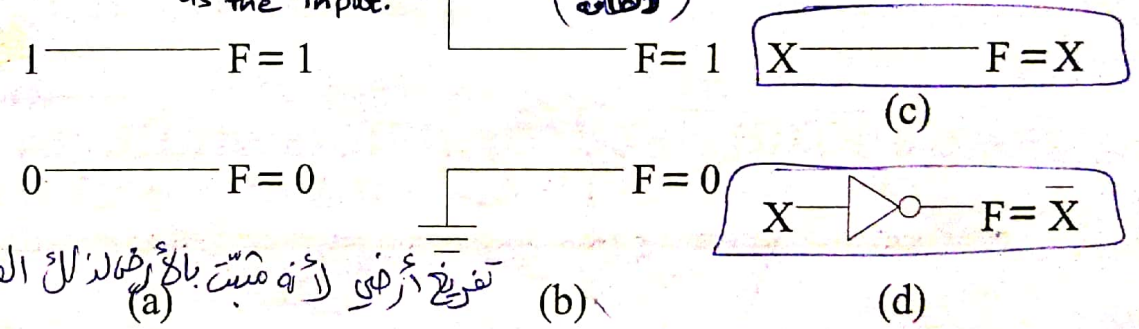
Functions of One Variable				
X	**F = 0**	**F = 1**	**F = X**	**F = $\bar{X}$**
0	0	1	0	1
1	0	1	1	0

Value fixing. transferring. inverting

- ***Value fixing* : a, b** (output) يأخذ وصف ال (F) ويحاله وخرج
- ***Transferring* : c** (same data). → the output is as same as the input. (نقل)
- ***Inverting* : d** (invert the data)
- ***Enabling* : next slide**

VDD ⇒ 1 (logic high)
= ⇒ 0
يأخذ قيمة ثابتة ومعينة حسب المفهوم (الحالة).

Vcc or VDD (مصدر الطاقة)

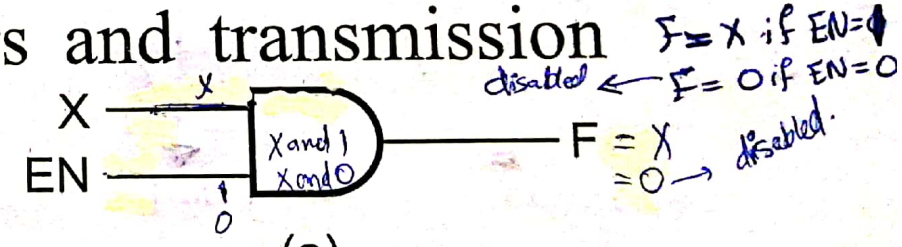


تفريغ أرضي لأنه مثبت بالأرض فيحدد لنا القيمة لـ (F) دائماً (0)

Scanned by CamScanner

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*** (using AND)
- When disabled, ***1 output*** (using OR)



Scanned by CamScanner

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

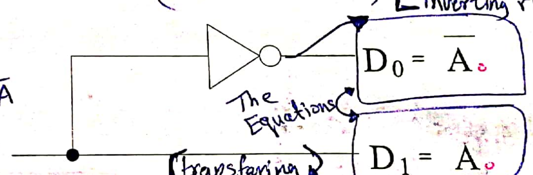
Scanned by CamScanner

1-to-2 Line Decoder

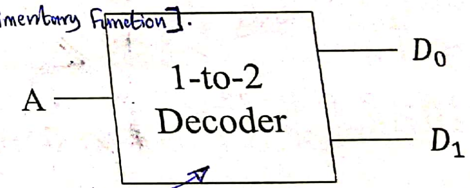
- When the decimal value of A equals the subscript of D_i, that D_i will be 1 and all others will be 0's
- Only one output is active at a time

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

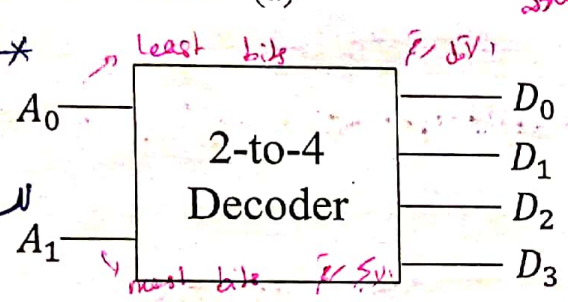
Scanned by CamScanner

2-to-4 Line Decoder

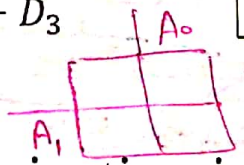
When $n = 2$

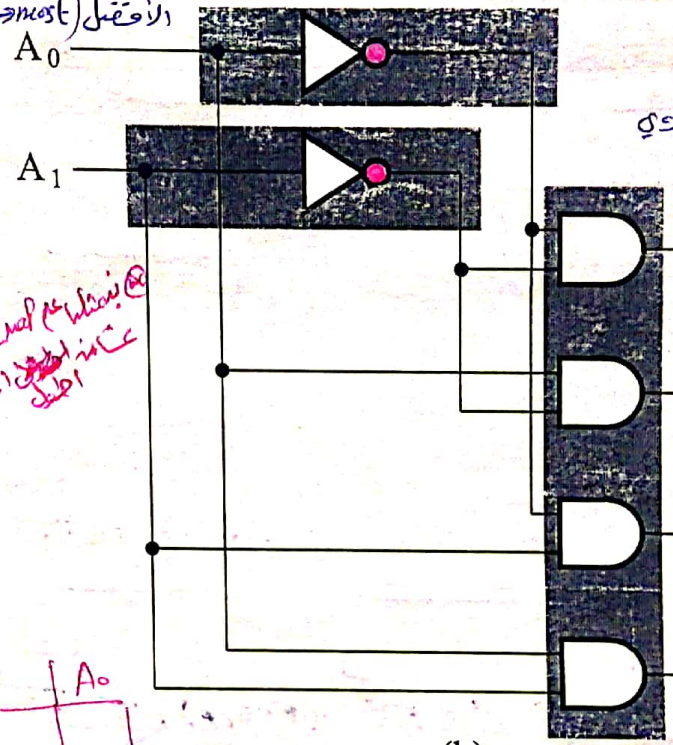
A_1	A_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

(a)



(c)





(b)

- 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***

Scanned by CamScanner

Decoder Expansion (less cost).

- General procedure given in book for any decoder with ***n inputs and 2^n outputs*** (نبدأ من آخر الشكل (الخلف للأمام)
- 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**

(loop) until we reach the simplest decoder

Scanned by CamScanner

Decoder Expansion - Example 1

- 3-to-8-line decoder
  - $k = n = 3$ = number of inputs.
  - 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
  - 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

Scanned by CamScanner

Decoder Expansion - Example 1

- $GN = 8 \times 2 + 4 \times 2 + 3$ (AND gates — كل وحدة 2-inputs; AND gates — كل وحدة 2-inputs; عدد الـ inverters)
- $GN = 27$
- Straight forward design has the same GN cost

* من ناحية (cost) متساويات ولكن من ناحية (delay) يختلفوا.

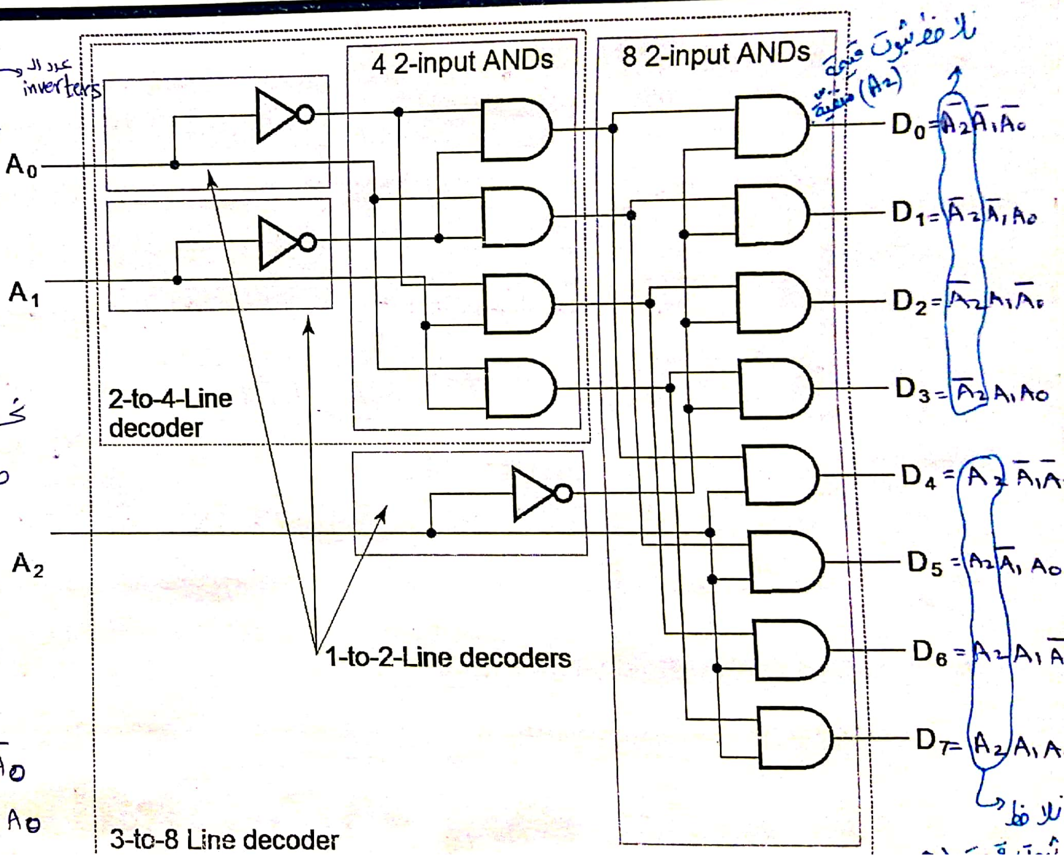
* لكي نسهّل الرسم، نبدأ من اليمين ثم نحدد شكل الـ output لكل (and gate) من حيث

Do ← هي $\bar{A}_2\bar{A}_1\bar{A}_0$ وهكذا لكل الحزمات

* يجوز تبديل (A2 بـ A0) أو غيرها أي أنّه أي ترتيب لا يهم والمهم فقط أن ترتيب الـ outputs يكون عليه صح.

$0 \equiv \bar{A}_2\bar{A}_1\bar{A}_0$

$1 \equiv \bar{A}_2\bar{A}_1 A_0$



Scanned by CamScanner

Decoder Expansion - Example 2

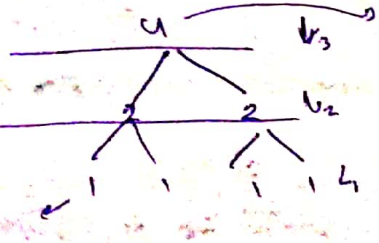
- 6-to-64-line decoder = 182
  - $k = n = 6$ → The number of inputs.
  - We need 2^6 (64) 2-input AND gates driven as follows: (2^n = number of AND gates).
  - k is even, so split to:
    - Two 3-to-8-line decoders
  - Each 3-to-8-line decoder is designed as shown in Example 1

4 to 2 = 4
2 to 2 = 2
8 2inPut ang = 16

or continue the loop!:

$k=n=3$ → odd → (1 to 2) decoder.
(2 to 4) decoder. → continue the loop! → $n=k=2$ even → 1 to 2 decoder.
1 to 2 decoder.

4 to 16 line decodar



1 to 3

24 anelgate) 1 2S
52 = Cost

© 2008 Pearson Education, Inc.

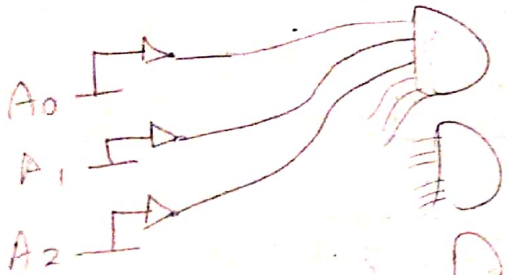
Scanned by CamScanner

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

($\rightarrow 2^6$ and gates. $\rightarrow 2^2$ and gates. $\rightarrow 2^4$ and gates. $\rightarrow$ (inverters))

Traditional

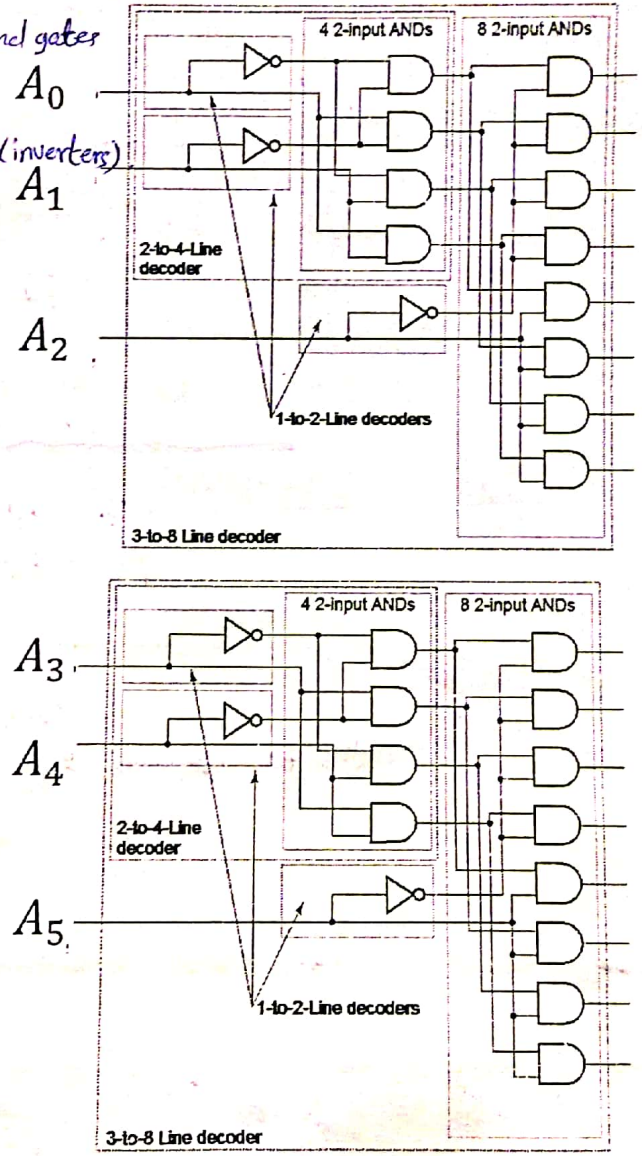


(and) ؟؟ ؟؟

(6-inverters) ؟؟ (6-inputs)

64+6+6 = ؟؟؟

390



$D_0 = \overline{A_5}\,\overline{A_4}\,\overline{A_3}\,\overline{A_2}\,\overline{A_1}\,\overline{A_0}$

$D_1 = \overline{A_5}\,\overline{A_4}\,\overline{A_3}\,\overline{A_2}\,\overline{A_1}A_0$

For the cost decoder Expansion is better.

But for the delay the straight forward design is better because it has one level implementation

$D_{62} = A_5A_4A_3A_2A_1\overline{A_0}$

$D_{63} = A_5A_4A_3A_2A_1A_0$

Scanned by CamScanner

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:
    - ① 4-to-16-line decoder
    - ② 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
- $GN = 128 \times 2 + 16 \times 2 + 8 \times 2 + 12 \times 2 + 7 = 335$
- Compare to straight forward design with GN cost of 903

* write the procedure for the Decoder Expansion:

© 2008 Pearson Education, Inc.

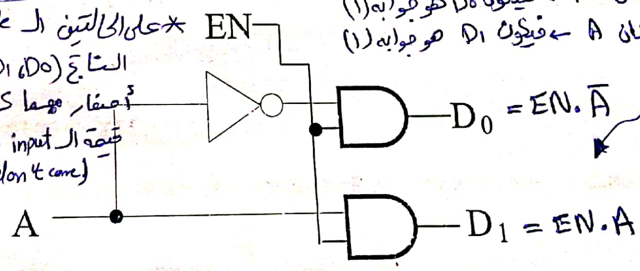
Scanned by CamScanner

Building Larger Decoders

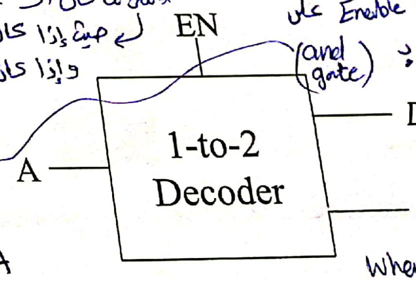
- **Method_1:** Decoder Expansion
- **Method_2:** Using Small ***Decoders with Enable input***
- Example: 1-to-2 line decoder with enable
  - In general, attach ***m-enabling*** circuits to the outputs
  - See truth table below for function
    - Note use of X's to denote both 0 and 1
    - Combination 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)

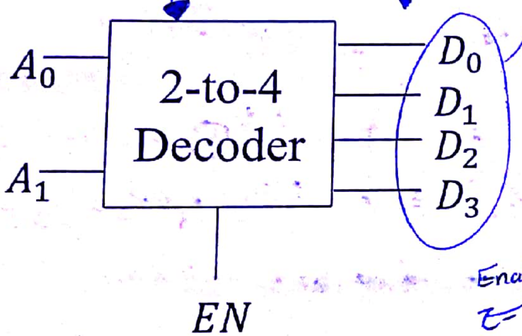


(c)

Scanned by CamScanner

2-to-4 Line Decoder with Enable

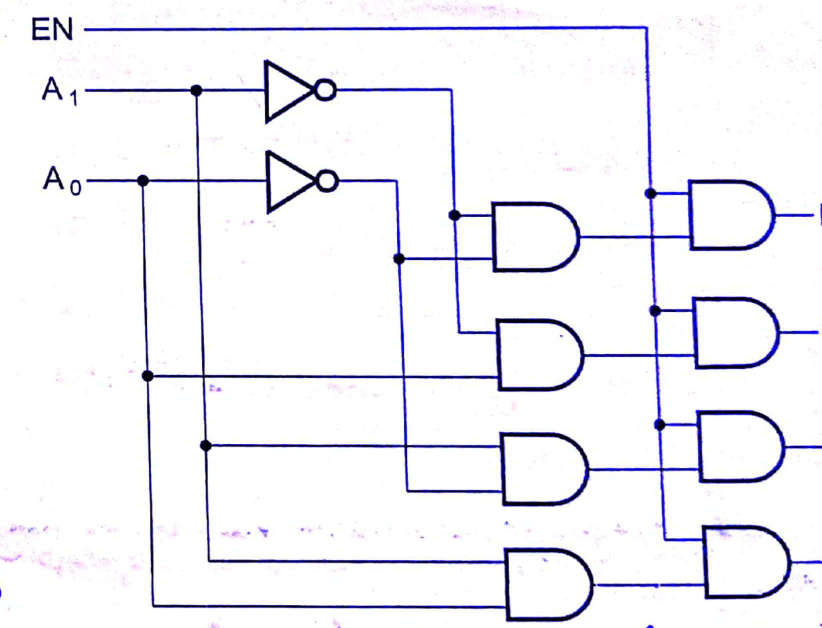
- Attach ***4-enabling*** circuits to the outputs
- See truth table below for function
  - Combination containing two X's represent four binary combinations
- Alternatively, can be viewed as distributing value of signal EN to 1 of 4 outputs
  - In this case, it is called a *Demultiplexer*



(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)



(b)

Logic and Computer Design Fundamentals, 4e
PowerPoint® Slides
© 2008 Pearson Education, Inc.

Scanned by CamScanner

$n = 2$ = inputs

$m = 2^n$ = outputs = number of 2-input AND gates.

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

(two 1-2 decoder and one inverter) ← طريقة الأسئلة بالامتحان

٢) ثم نحدد الـ output كيف يتم تمثيله برموز الـ inputs ونكمل توصيل للأسلاك.

* معرفة كيفية حل السؤال: ١) يجب علينا رسم T.T (Truth table) للـ decoder المطلوب اختصاره بـ decoder أبسط منه. ٢) نقسم الجدول ونحدد الـ Enable حسب الـ most significant digit.

A_1	A_0	Decimal	D_0	D_1	D_2	D_3
0	0	(0)	1	0	0	0
0	1	(1)	0	1	0	0
1	0	(2)	0	0	1	0
1	1	(3)	0	0	0	1

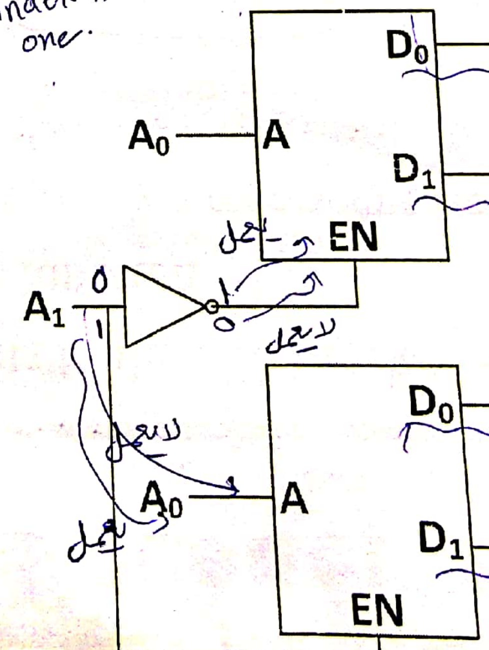
1st 1-to-2 Decoder (columns D_0, D_1) — 2nd 1-to-2 Decoder (columns D_2, D_3)

the index is one.

* فقط (decoder) واحد يشتغل ويكون جوابه (1) والباقي الـ (decoders) يكون جوابهم (0).

* وجود الـ Enable هو فقط لجعل واحد من الـ (decoders) يعمل (1) أي شغال والثاني لا يعمل لكي لا يطلع قيم، فيكون الـ output أكثر من قيمة وتتعارض.

* لاحظ: غالباً الـ (Enable) هو الـ (most significant digit.)



$D_0 = EN \cdot \bar{A_0} \rightarrow \bar{A_1}\bar{A_0}$

$D_1 = EN \cdot A_0 \rightarrow \bar{A_1} A_0$

D_2: $EN \cdot \bar{A_0} = A_1 \cdot \bar{A_0}$

D_3

* نأخذ سطر كامل:

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

* لاحظ أن: A_1 هو الـ Enable نجعله invert لكي يشتغل وعندما يعمل يعطي قيمة (1) عند الـ (D) التي رتبتها (0) وهي D_0 والباقي أصفار.

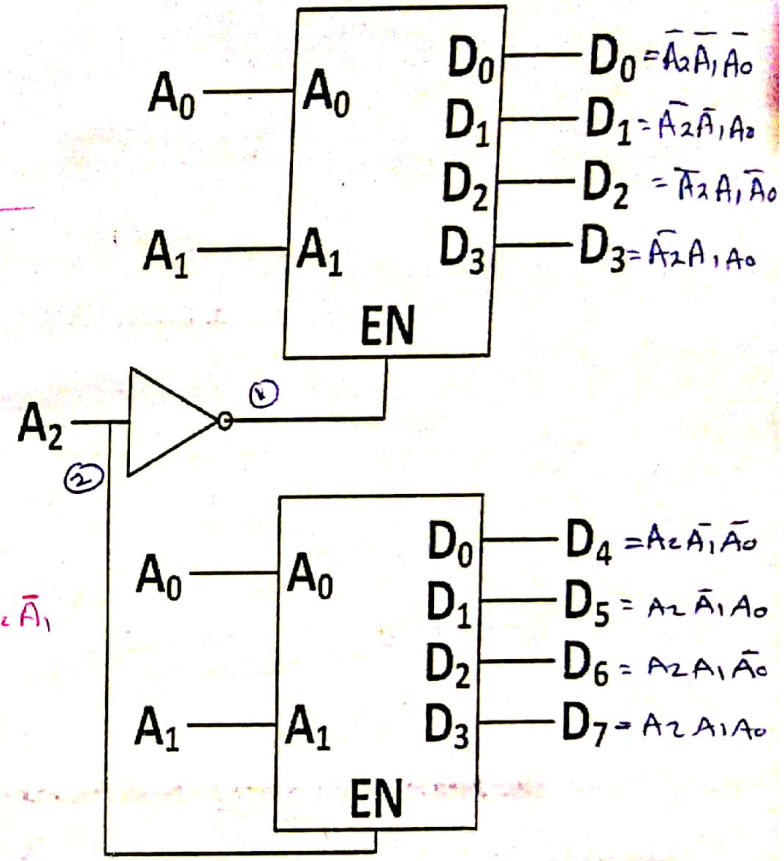
3-to-8 Decoder using 2-to-4 Decoders and Inverters

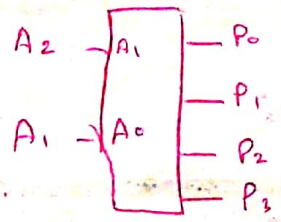
$n = 3$ = inputs.

$m = 8$ = outputs $= 2^n$ = number of 2-input AND gates.

A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

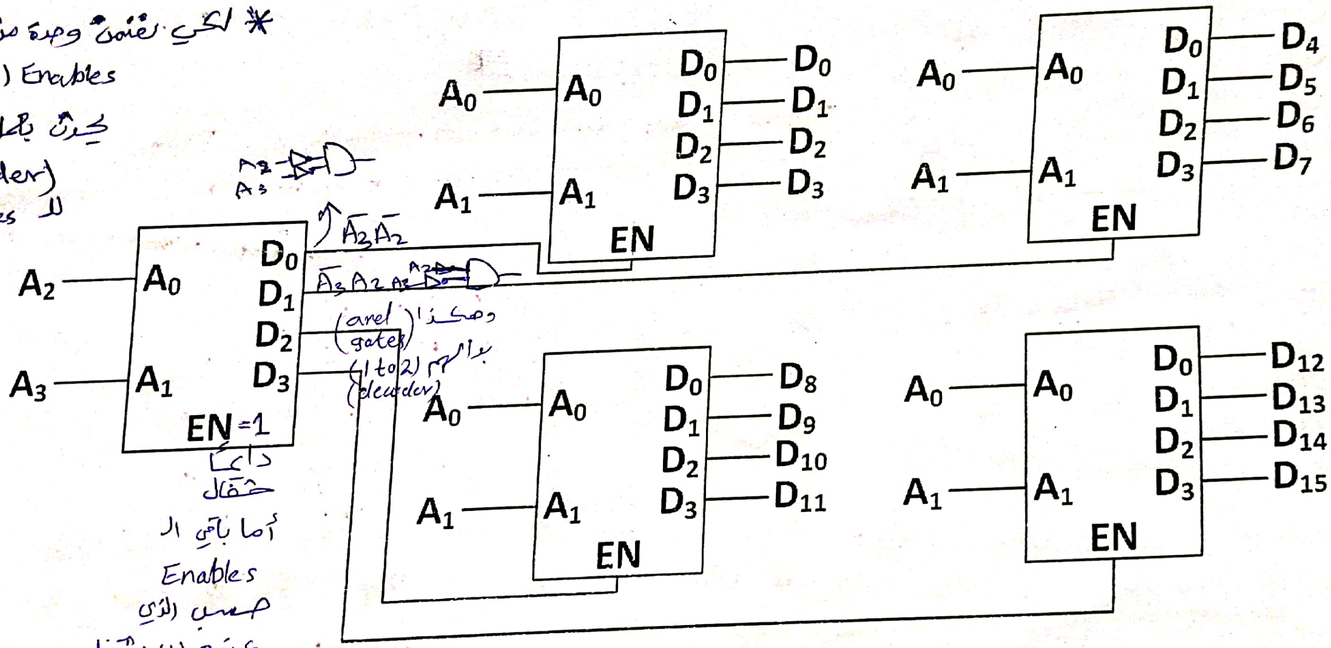
1st 2-to4 Decoder — 2nd 2-to4 Decoder





Scanned by CamScanner

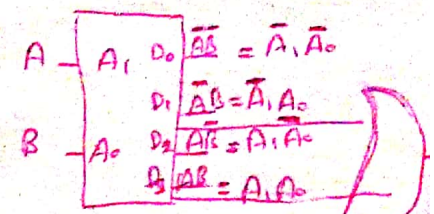
4-to-16 Decoder using Only 2-to-4 Decoders



Scanned by CamScanner

Combinational Logic Implementation - Decoder and OR Gates

- Implement ***m*** functions of ***n*** variables with:
  - Sum-of-minterms expressions
  - 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

© 2008 Pearson Education, Inc.

Scanned by CamScanner

Example1

- Implement function f using decoder and OR gate:

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

→ from Boolean Equation to Sum of minterms.

m = 1 · number of functions.

- $n = 3$ variables → ***3-to-8 decoder***

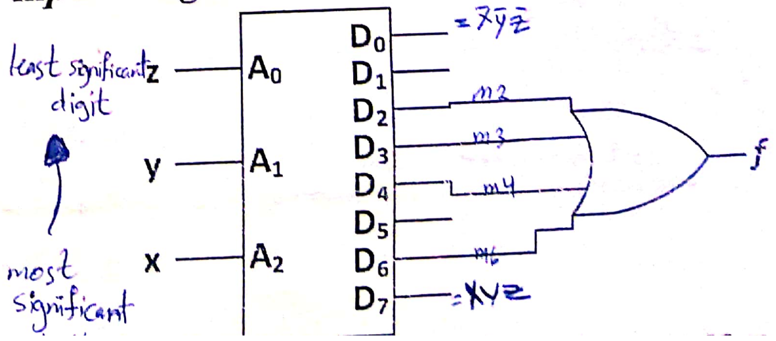
or using truth table.
$x\bar{z} \rightarrow xy\bar{z}$ or $x\bar{y}\bar{z}$
110 . 100

- One function → ***One OR gate***
- Solution: Convert 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***

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



Scanned by CamScanner

Example2

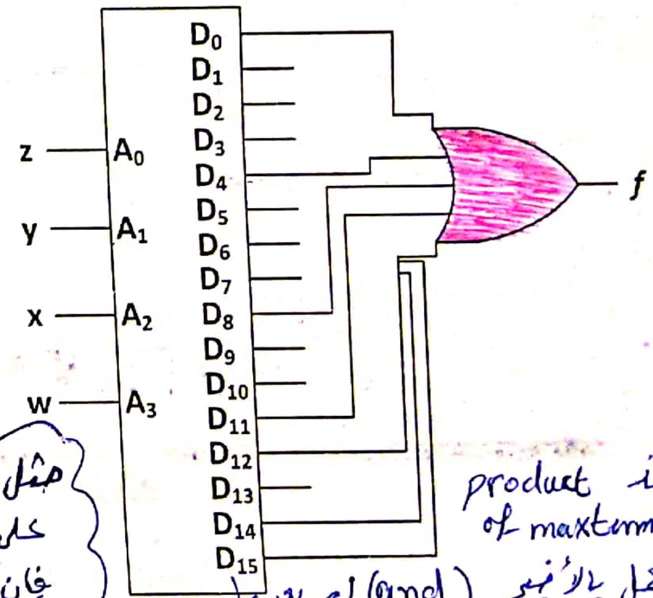
- Implement function f using decoder and OR gate:

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

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

one (OR) gate

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



* أحيانًا قد يكون الـ Function يحتوي على أكثر من نصف الـ (minterms) أي أن "عددهم كبير" ولا يجوز جعلهم جميعًا (inputs) لـ (OR gate) لذلك نأخذ الـ ($\overline{F}$) ونجعل بآخر شكل الـ (OR gate) (inverter) فتصبح (NOR gate)، حيث يكون "أقل عدد" الـ (inputs)

مثل أن F تحتوي على (minterm : ١٤) فإن $\overline{F}$ تحتوي على (minterm : ٢) وذلك أفضل

أو نأخذ product of maxterms وذلك نعمل بالأخير (and) لهم بعدها نعمل لكل (maxterm) inverter

"مثلاً"

13, 15

© 2008 Pearson Education, Inc.

Scanned by CamScanner

* لو كان عندنا (2functions) وبختلفوا في عدد ال inputs

Example3

نعمل لل (function الناقص) الحرف الناقص [ ] (W+W̄) & and

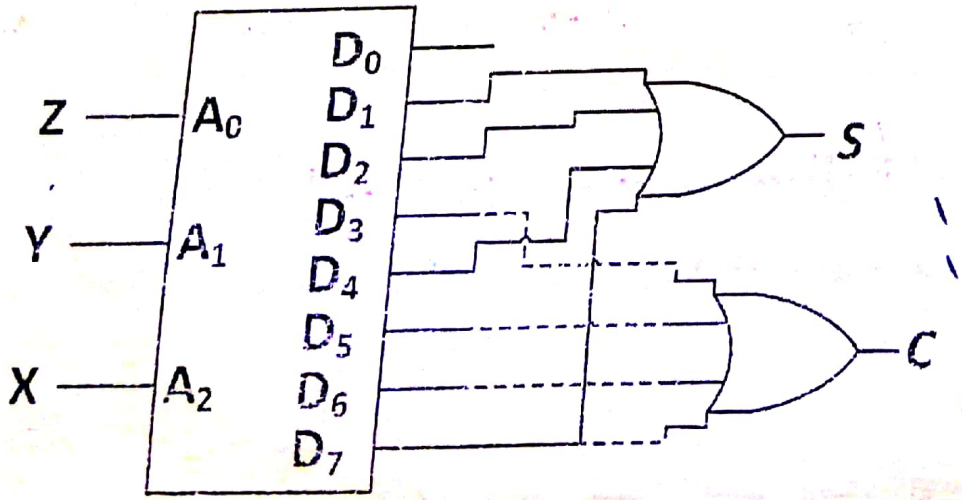
ونكتب ال (minterms) لكل الأرقام الثنائية بدلالة العدد الأكبر من المتغيرات . حيث أن (W) لا تؤ (nterms) قيمة ال (nction) ال الناقص

- Implement functions *C and S* using decoder and OR gates:

السؤال ← طلب بأي طريقة بالممكن.

- $n = 3$ variables → ***3-to-8 decoder***
- Two function → ***Two OR gates***
- Solution:
  - $C = \sum_m(3,5,6,7)$ → ***4-input OR gate***
  - $S = \sum_m(1,2,4,7)$ → ***4-input OR gate***

X	Y	Z	C	S	
0	0	0	0	0	m0
0	0	1	0	1	m1
0	1	0	0	1	m2
0	1	1	1	0	m3
1	0	0	0	1	m4
1	0	1	1	0	m5
1	1	0	1	0	m6
1	1	1	1	1	m7



* نرى متى يكون ال Function = 1 ← C
= 1 ← S

ونكتبه كشكل مجموع minterms
ونوصله بـ (OR gate)

Scanned by CamScanner

Example4

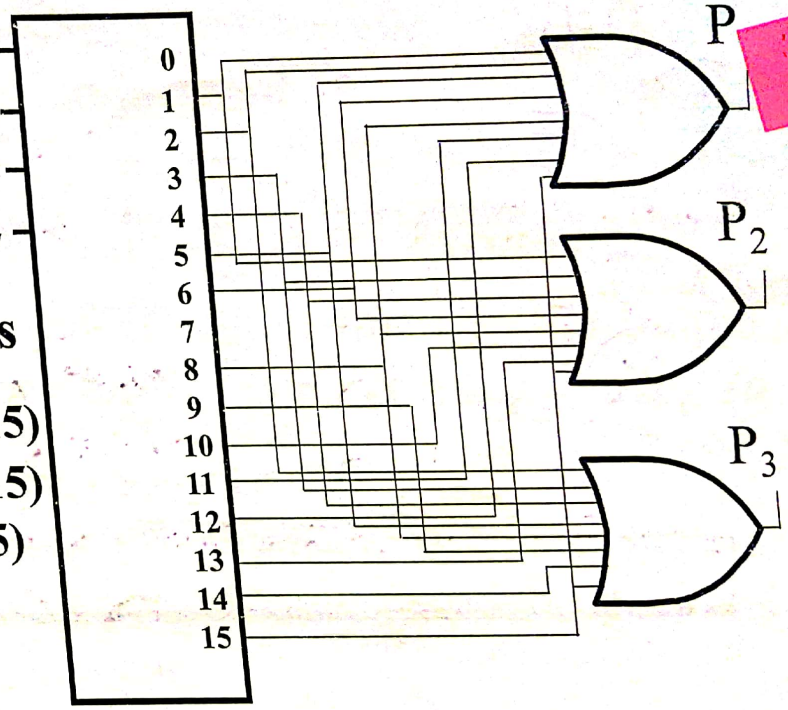
- Implement the following set of odd parity functions of (A_7, A_6, A_5, A_4)

$$*P_1 = A_7 \oplus A_5 \oplus A_4$$
$$*P_2 = A_7 \oplus A_6 \oplus A_4$$
$$*P_3 = A_7 \oplus A_6 \oplus A_5$$

- 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?



© 2008 Pearson Education, Inc.

Scanned by CamScanner

Example5

- Implement function F using 3-to-8 decoder, AND gate and (4) inverters: $F(A,B,C) = \sum_m(1,3,5,7)$ ⇄ 4 minterms + 4 maxterms.

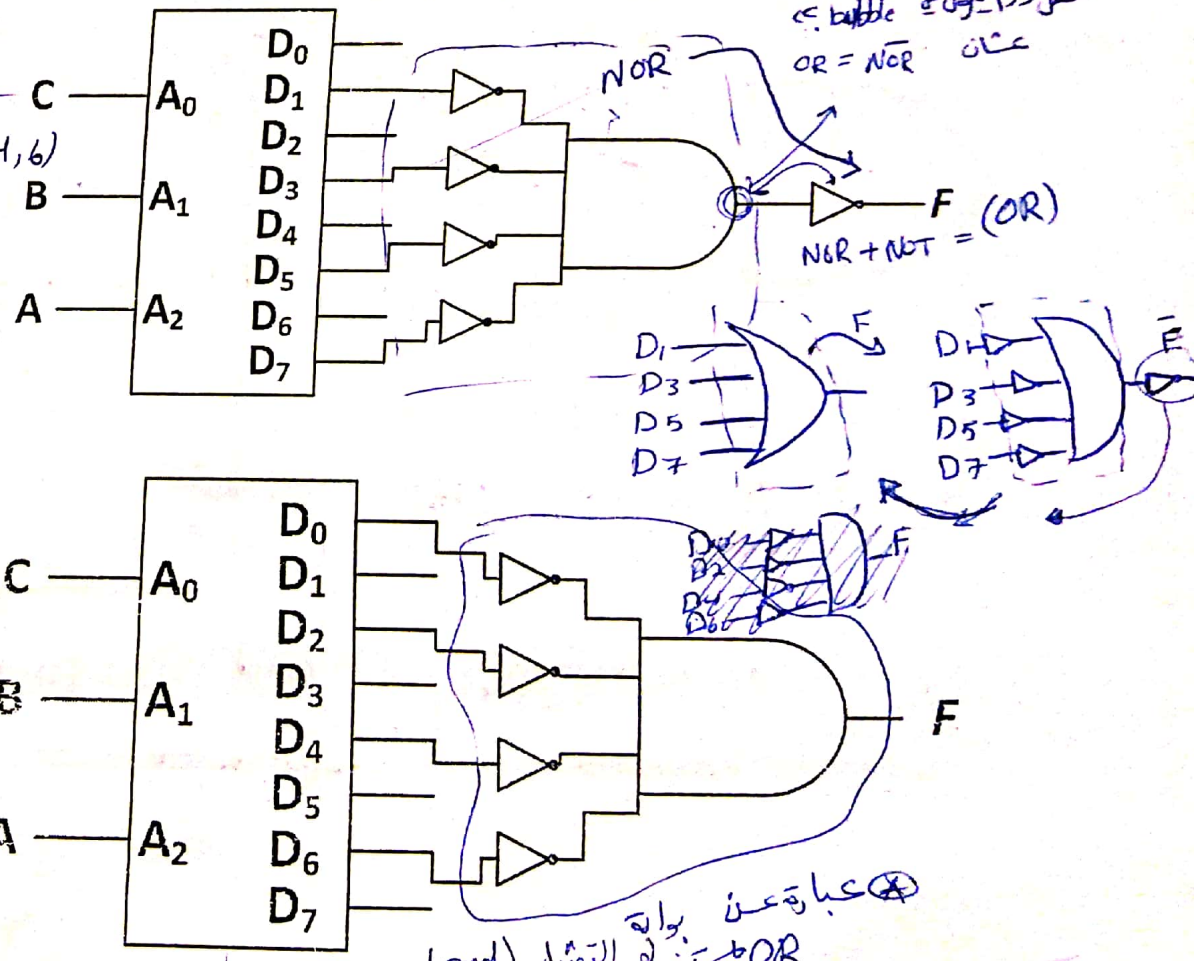
- Solution with 5 inverters:

$\Sigma m F$ مثل $\Pi_M F$

$\Sigma m F = \overline{\Pi_M F}$

$\Pi_M F = \Sigma m \overline{F}$

$(\overline{F}) = \Sigma_m(0,2,4,6)$ as a sum of minterms



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

$F \rightarrow \Pi_M(0,2,4,6)$

عبارة عن بوابة التمثيل (and) تستخدم OR

Scanned by CamScanner

Encoding

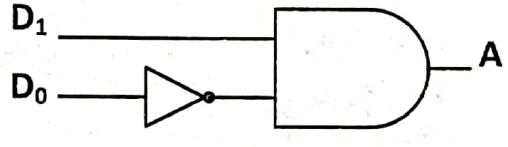
- ***Encoding:*** the opposite of decoding - the conversion of an m-bit input code to a n-bit output code with $n \leq m \leq 2^n$ such that each valid code word produces a unique output code
  - Circuits that perform encoding are called ***encoders***
  - An encoder has 2^n (or fewer) input lines and 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

Scanned by CamScanner

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

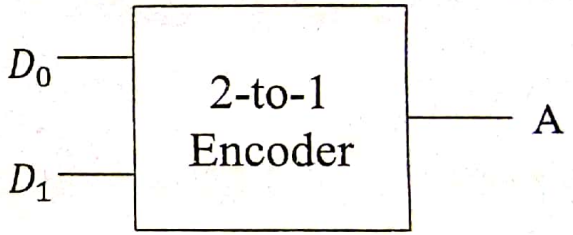
D_1	D_0	A
0	0	Invalid Input
0	1	0 (decimal)
1	0	1 (decimal)
1	1	Invalid Input

(a)



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

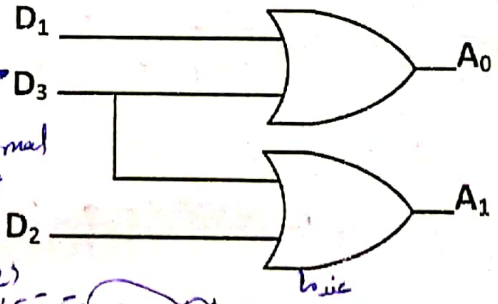
(b)



(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

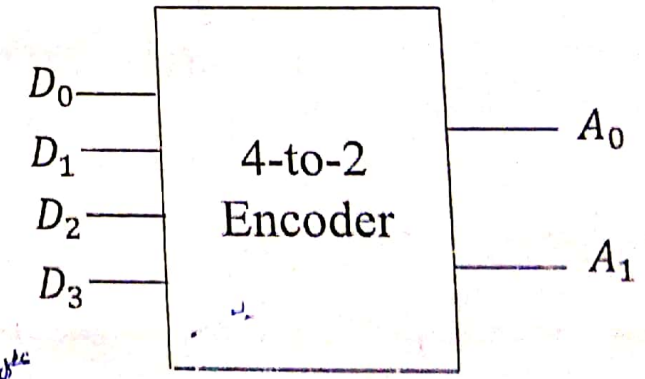
(a)



$$A_0 = D_1 + D_3$$

$$A_1 = D_2 + D_3$$

(b)



(c)

Scanned by CamScanner

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

كأنّه تحويل من octal ← binary

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	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

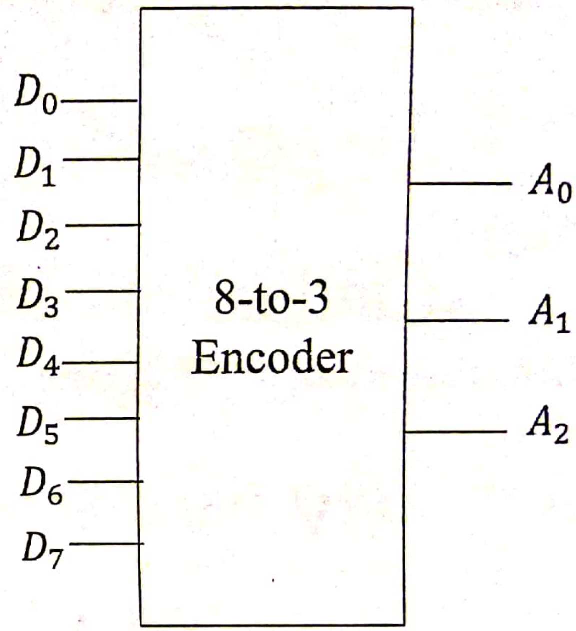
(a)

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

(b)

* لمّا ال bit التي ترتيبها نفس ال index تاع (A) يكون هو (1)

متى يكون (A2) فيه (1) أضعها بدل ما أكتب truth table



(c)

© 2008 Pearson Education, Inc.

Scanned by CamScanner

10 - inputs.. 4 outputs (digits) $2^4 \rightarrow 16$ ✓ (10) يغطي $2^3 \rightarrow 8$ × (10) لا يغطي

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

Scanned by CamScanner

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:

$$A_3 = D_8 + D_9$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_0 = D_1 + D_3 + D_5 + D_7 + D_9$$

- 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***

© 2008 Pearson Education, Inc.

Scanned by CamScanner

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

- 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

Scanned by CamScanner

4-to-2 Low Priority Encoder

# of Minterms/ Rows	D_3	D_2	D_1	D_0	A_1	A_0	V
$2^0=1$	0	0	0	0	X	X	0 invalid
$2^3=8$	X	X	X	1	0	0	1 valid
$2^2=4$	X	X	1	0	0	1	1 valid
$2^1=2$	X	1	0	0	1	0	1 valid
$2^0=1$	1	0	0	0	1	1	1 valid

(a)

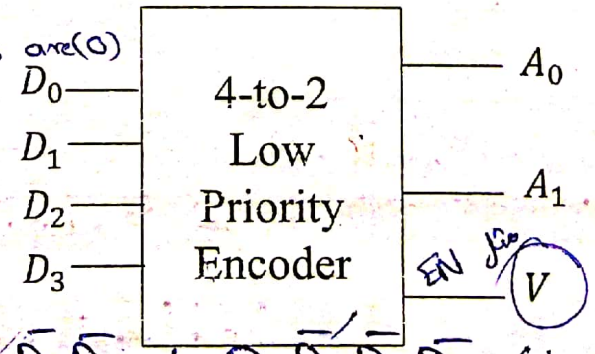
Number of Minterms per Row = 2^k of don't cares

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

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

$$V = D_3 + D_2 + D_1 + D_0$$ (OR gates)

(b)



* valid bit = 0 . when all inputs are(0)
* valid bit = 1 . when one or two or more than input = (1)

$$A_2 = D_2\overline{D_1}\,\overline{D_0} + D_3\overline{D_2}\,\overline{D_1}\,\overline{D_0}$$ (simplification theorem).

$$A_2 = D_2\overline{D_1}\,\overline{D_0} + D_3\overline{D_1}\,\overline{D_0}$$ (c)

© 2008 Pearson Education, Inc.

Scanned by CamScanner