ENCS 2340 Summary Chapter 3

By : Malak Obaid

Ch3: Gate-Level Minimization

* Karnaugh Map (k-Map)

* كل مربع في (k-map) يمثل minterm ومنه عدد المربعات يساوي عدد ال minterms

* عدد ال minterms يساوي عدد الصفوف (rows) في ال (k-map)

* كل مربعين جنب بعض يختلفو عن بعض في رمز واحد

"Adjacent squares differ in the value of one variable"

Remember that :-

* Simplified sum-of-product expression (AND-OR circuit)

* Simplified product-of-sums expression (OR-AND circuits)

Karnaugh map

* Two variable (k-map)

* Minterms m_0, m_1 are adjacent (مجاورات لبعض)

also (m_2 and m_3) but they differ in the

value of variable y.

* Minterms (m_0 and m_2) are adjacent

also (m_1 and m_3) but they differ in the

value of variable x

$x \backslash y$	0	1
0	m_0	m_1
1	m_2	m_3

$\Rightarrow$

$x \backslash y$	0	1
0	$x'y'$	$x'y$
1	xy'	xy

الرمز المشترك بينهن هو y وقيمته صفر اذا نكتب y'
ولا ناخذ قيمة x لانها متغيرة

Ex

x	y	f
0	0	1
0	1	0
1	0	1
1	1	1

$\Longrightarrow$

$x \backslash y$	0	1
0	1 (0)	0 (1)
1	1 (2)	1 (3)

الرمز المشترك هو x
وقيمته 1 اذا يبقى x ولا ناخذ قيمة y لانها متغيرة.

* أي مربعين متجاورين فيهن رقم 1 بنقدر نوخذهم سوا

(Two adjacent cells containing 1's can be combined)

$$f = m_0 + m_2 + m_3$$

$$= x'y' + xy' + xy \quad (6 \text{ Literals})$$

$$\therefore f = x + y' \quad (2 \text{ Literals})$$

Note that :-

* $m_0 + m_2 = x'y' + xy' = (x' + x)y' = y'$ (ناتج العمود فوق)

* $m_2 + m_3 = xy' + xy = x(y' + y) = x$ (ناتج الصف فوق).

Three-Variable Karnaugh Map

* it has eight squares (for 8 minterms) numbered 0 to 7
* Each square is adjacent to three other squares.

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

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

Simplifying Three-Variable Function

* Simplify this boolean function

$f = \sum(\underline{3, 4, 5, 7})$
(x,y,z)

$f = x'yz + xy'z' + xy'z + xyz$ (12 literals)

$x \backslash yz$	00	01	11	10
0	0 (0)	0 (1)	1 (3)	0 (2)
1	1 (4)	1 (5)	1 (7)	0 (6)

xy' , yz

* نشوف طريقة نجمع فيها اكبر عدد واحدات
إما 2 أو 4 سواء جنب بعض في هنا الاشكال
نستطيع ايجاد فقط 2 متتاليتين من 1

$\therefore f = xy' + yz$

في حال كان الاقتران $f = \sum(3,4,6,7)$

$x \backslash yz$	00	01	11	10
0	0	0	1	0
1	1	0	1	1
	z'	z		z'

yz , xz'

$\therefore f = yz + xz'$

تعتبر متجاورات ...

Note :

* By combining squares , we reduce number of literals in a product term

" كلما زاد عدد المربعات المشتركة تقل عدد الرموز "

On 3-Variable K-Map

* One square represent a minterm with 3 variables

* Two adjacent squares represent a term with 2 variables

* Four " " " " " with 1 variable

* Eight " " is the constant '1' (No Variable)

Ex $f_{(A,B,C)} = A'C + A'B + AB'C + BC$

① Express the function as sum-of-minterms.

"نعيد إضافة النواقص لكل حد كما تعلمنا سابقاً"

$f = \sum(1, 2, 3, 5, 7)$

2) Find the minimal sum-of-products expression

A \ BC	B'		B	
A'	0	1	1	1
A	0	1	1	0
	C'	C		C'

$\therefore f = C + A'B$

Four-Variable Karnaugh Map

wx \ yz	00 (y')	01 (y')	11 (y)	10 (y)	
W' 00	m_0 $w'x'y'z'$	m_1 $w'x'y'z$	m_3 $w'x'yz$	m_2 $w'x'yz'$	x'
01	m_4 $w'xy'z'$	m_5 $w'xy'z$	m_7 $w'xyz$	m_6 $w'xyz'$	x
W 11	m_{12} $wxy'z'$	m_{13} $wxy'z$	m_{15} $wxyz$	m_{14} $wxyz'$	
10	m_8 $wx'y'z'$	m_9 $wx'y'z$	m_{11} $wx'yz$	m_{10} $wx'yz'$	x'
	z'	z		z'	

On 4-Variable K-Map

* One square represents a minterm with 4 variables
* Two adjacent squares represent a term with 3 variables
* Four " " " " " with 2 variables
* Eight " " " " a term with 1 variable
* Combining all 16 squares is the constant '1' (No variable)

Ex Given $f_{(a,b,c,d)} = \Sigma(0,2,4,5,6,7,8,12)$

Draw k-map and minimize f as Sum of products

$ab \backslash cd$		c'		c		
		d'	d	d	d'	
a'		1			1	b'
a'		1	1	1	1	b
a		1				b
a		1				b'

$a'd'$

$a'b$

$c'd'$

$\therefore f = a'd' + a'b + c'd'$

Ex $F = w'x'y' + x'yz' + w'xyz' + wx'y'$

① Express the function as sum-of-minterms

$F = w'x'y'(z+z') + (w+w')x'yz' + w'xyz' + wx'y'(z+z')$

$= w'x'y'z + w'x'y'z' + wx'yz' + w'x'yz' + w'xyz' + wx'y'z$

$+ wx'y'z'$

$= \Sigma(0,1,2,6,8,9,10)$

② Find the minimal sum-of-products expression

wx\yz	y'	y'	y	y	
w'	1	1		1	x'
				1	x
w					
	1	1		1	x
	z'	z	z	z'	

$\rightarrow x'z'$

$\rightarrow w'yz'$

$x'y'$

$\therefore F = x'y' + x'z' + w'yz'$

(7 literals)

Prime Implicants :-

* is a product term obtained by combining the maximum number of adjacent squares in k-Map

(يعني اكبر مربع بيقدر يشمل اكبر عدد من ال 1 المتجاورة)

* Essential prime Implicant:

يكون المربع اللي بيشمل اكبر عدد من الواحدات بحيث انه على الأقل واحد من المربعات لم يتم شمله مع المربعات الأخرى

<u>Ex</u> Find all the prime implicants and essential prime implicants for :

① $f_{(a,b,c,d)} = \sum (0,2,3,8,9,10,11,12,13,14,15)$

ab \ cd	c' d'	c' d	c d	c d'	
a' b'	1		1	1	b'
a' b					b
a b	1	1	1	1	b
a b'	1	1	1	1	b'

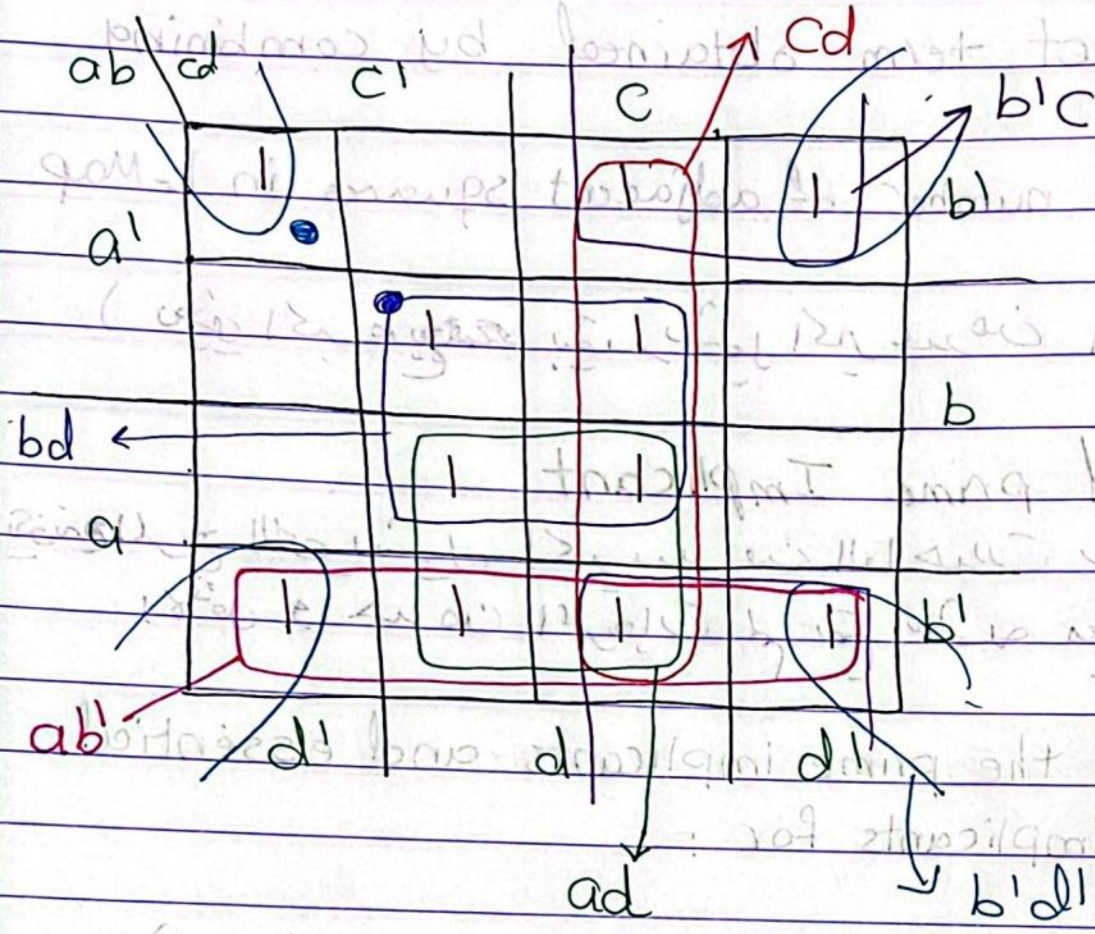
→ b'd'

→ a

b'c

three prime implicants & all of them are essential

② $f_{(a,b,c,d)} = \Sigma(0,2,3,5,7,8,9,10,11,13,15)$



* Six prime implicants
  bd, b'd', ab', ad, cd, b'c

Only two Prime implicants are essential

bd and b'd'

Simplification Procedure Using K-Map

1) Find All the essential prime implicants

2) Add prime implicant to cover the function

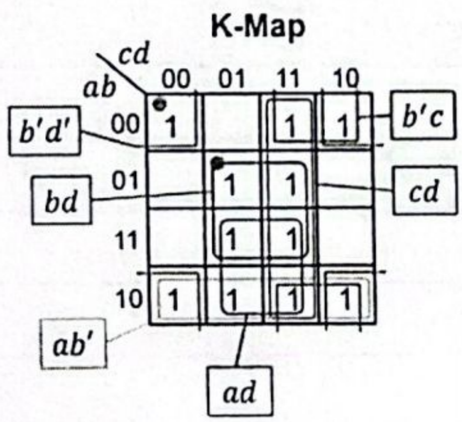
* في حال أن ال essential prime... غطّت كل الواحدات في الجدول نكتفي بأول خطوة فوق أما إذا لم تشملهن نضيف ال prime implicant

- We choose the minimal subset of prime implicants that covers the remaining 1'S
all

* يمكن التعبير عن الفنكشن بأكثر من صورة لكن نفس عدد الرموز

Consider again: $f(a,b,c,d) = \sum(0,2,3,5,7,8,9,10,11,13,15)$

Obtain all minimal sum-of-products (SOP) expressions



Two essential Prime Implicants: bd and $b'd'$

Four possible solutions:

$f = bd + b'd' + cd + ad$

$f = bd + b'd' + cd + ab'$

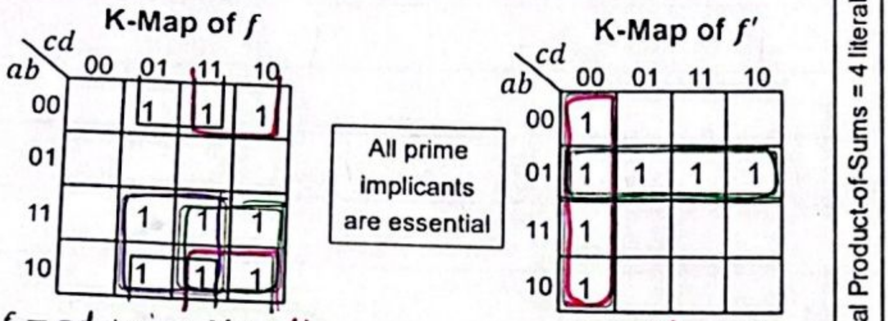
$f = bd + b'd' + b'c + ab'$

$f = bd + b'd' + b'c + ad$

the essential prime implicants نعتبرهن ثم نقوم باختيار اي 2 من ال prime implicants

Product-of-Sums (POS) Simplification

❖ Example: $f(a,b,c,d) = \sum(1,2,3,9,10,11,13,14,15)$



$f = ad + ac + b'd + b'c$

Minimal Sum-of-Products = 8 literals

$f' = c'd' + a'b$

$f = (c+d)(a+b')$ → Minimal Product-of-Sums = 4 literals

* حتى نحول الصيغة لأي اقتران على شكل POS

أولا نجد ال k-map للاقتران ك SOM's (minterms) ثم نجد

ال k-map ل f' بأن نضع 1 بالأماكن الفارغة

كما في الصورة ونجد الصيغة له ك Sum of product's

ثم نحوله ل product of sum عن طريق أخذ ال prime.

للجواب كما تم توضيحه بالمثال.

Simplification procedure

1) Draw K-Map for the function f

(نكتب الفنكشن f بأقل عدد من الرموز العكسية كما تعلمنا سابقاً (as sum of product expression)

2) Draw K-Map for f' (we replace the 0's of f with 1's in f')

3) Obtain a minimal sum of product expression for f'

(يعني نكتب f' بأقل عدد من الرموز العكسية من الواحدات)

4) Use DeMorgan theorem to obtain f = (f')'

(يعني نرجع ناخذ كومبلمنت لـ f' حتى نحصل على f)

the result is a minimal Product of sum expression for f (كما في المثال الاتي)

5) Compare the cost of the minimal SOP and POS expression

يعني بنشوف أي عدد رموز من f أقل وناخذه

SOP : Sum of product
POS : Product of Sum

Ex : Express the boolean function in Standard form using the minimal number of literals

$$F_{(A,B,C,D)} = \prod(3,4,6,7,11,12,13,14,15)$$

* حتى نجد الجواب بأقل عدد من الرموز نجد الاقتران
في ال k-map مرة بصيغة POS ومرة بشكل SOP
ونقارن ونأخذ "الأقل عدداً" من الرموز

① (sum-of-product) SOP نبدأ بال
لكن ننتبه هو اعطانا الاقتران بال Π يعني POS فنضع
1 في الأماكن (0,1,2,5,8,9,10)

$A'C'D$ $B'D'$

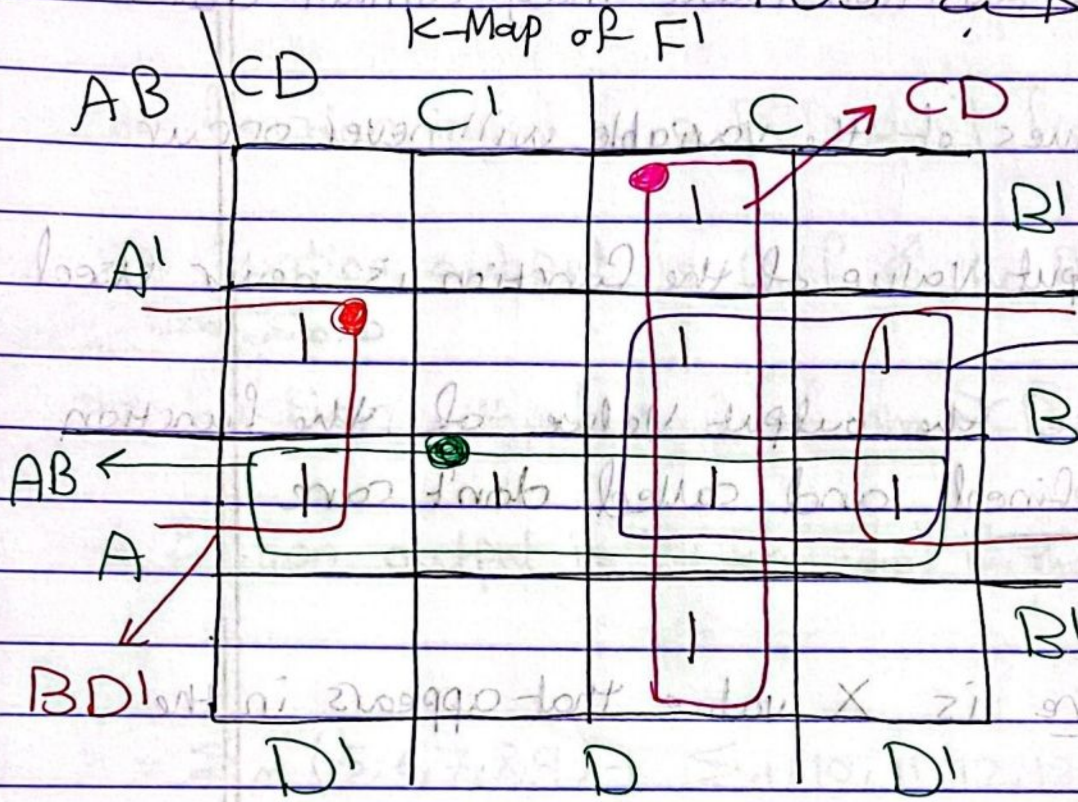
AB \ CD	C'	C'	C	C	
A'	1	1		1	B'
A'		1			B
A					B
A	1	1		1	B'
	D'	D	D	D'	

$B'C'$

$\therefore F = B'D' + B'C' + A'C'D$

$\therefore$ Minimal Sum-of-product = 7 literals

* الآن نكتب 1 مكان الارقام في السؤال حتى نجد الاقترانات حسب POS



ثم نأخذه لان تم 1 تغطيتهن من ال essential ولكن نريد اقل عدد من الرموز

$F' = CD + BD' + AB$ 6 Literals

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

↓

العدد اقل من ال SOP

Don't Cares

* Sometimes a function table may contain entries which
- the input values of the variable will never occur
or the output value of the function is never used

* In this case the output value of the function is not defined and called don't care

* A don't care is X value that appears in the function table

* The X value can be later chosen to be 1 or 0 to minimize the function implementation

Example of a function with Don't Cares

* BCD

- the function input is a BCD digit from 0 – 9

- the function output is 0 if the BCD input 0 to 4

هذه تكون معطيات من السؤال

- the function output is 1 if the BCD input is 5 to 9

- the function output is X (don't care) if the input is (10 to 15) (Not BCD)

$$\therefore f = \underbrace{\Sigma_m (5,6,7,8,9)}_{\text{Minterms}} + \underbrace{\Sigma_d (10,11,12,13,14,15)}_{\text{don't Care}}$$

لـ بنفع نحط قيم الـ Maxterms

Minimize the function with Dont cares

Consider $f = \sum_m (5,6,7,8,9) + \sum_d (10,11,12,13,14,15)$

ab \ cd	$c'd'$	$c'd$	cd	cd'	
a'	0	0	0	0	b'
a'	0	1	1	1	b
a	X	X	X	X	b
a	1	1	X	X	b'
	d'	d	d	d'	

1* if the dont cares were treated as 0's (zero's)

we get: $f = a'bd + a'bc + ab'c'$ (9 literals)

2* if the dont cares were treated as 1's

we get: $f = a + bd + bc$ (5 literals)

Note that: the don't care values can be selected to be either 0 or 1 to produce a minimal expression

Simplification Procedure with Don't Cares

1- Find all the essential prime implicants

- Covering maximum number (power of two) of 1's & X's (don't care)
- Mark the 1's that make the prime implicants essential

2- Add prime implicants to cover the function

- Choose a minimal subset of prime implicants that cover all the remaining 1's
- make sure to cover all 1's not covered by the essensial prime implicants
- minimize the overlap among the additional prime implicants
- You need to cover all the don't cares (some can remain uncovered)

☆ Sometimes a function has multiple simplified expressions

Minimal product-of-sums with Don't Cares

* Simplify $g = \sum m(1,3,7,11,15) + \sum_d (0,2,5)$

obtain a minimal product of sums expression

لكن نوجده بطريقة (POS) نوجد K-Map لـ g' ثم

نحوله لـ g باستخدام deMorgan

$g' = \sum m(4,6,8,9,10,12,13,14) + \sum_d (0,2,5)$

لم نكتب 0,2,5 لأنه يحتمل don't care

تبقى كما هي

ab \ cd		c'		c		
a'	X	0	0	X	b'
	1	X	0	1	b
a	1	1	0	1	
	1	1	0	1	b'
	d'	d		d'		

نبدأ بتجميع اكبر عدد من الواحدات من أول 1

K-Map of g'

$\therefore$ Minimal $g' = d + ac'$

Minimal product of sum:-

$g = d(a' + c)$ (3 literals)

* ليس شرطاً ان نشمل كل ال X في المربعات

Five-variable Karnaugh Map :

$2^5 = 32$ squares → numbered 0 to 31

* بنعمل مربعين كل واحد 16 مربع (لِ 4 رموز)

f(a, b, c, d, e)

أما رمز a مرة نحطه صفر ومرة 1 فوق كل مربع كامل هكذا

a=0

bc\de				
	m_0	m_1	m_3	m_2
	m_4	m_5	m_7	m_6
	m_{12}	m_{13}	m_{15}	m_{14}
	m_8	m_9	m_{11}	m_{10}

a=1

bc\de				
	m_{16}	m_{17}	m_{19}	m_{18}
	m_{20}	m_{21}	m_{23}	m_{22}
	m_{28}	m_{29}	m_{31}	m_{30}
	m_{24}	m_{25}	m_{27}	m_{26}

layer 1 layer 2

* كل مربع له 5 مربعات تعتبر مجاورة (adjacent) لَهُ

* كل اللي بالهايلايت الازرق تعتبر adjacent لِ m_{21}

* ⇐ أي مربع بِ layer 2 الهاطلا adjacent square في layer 1 (يكونو مقابلات لبعض في نفس الموقع)

أي

m_0 adjacent to m_{16}

m_1 " " 17

m_4 " " m_{20}

Example of Five-Variable K-map

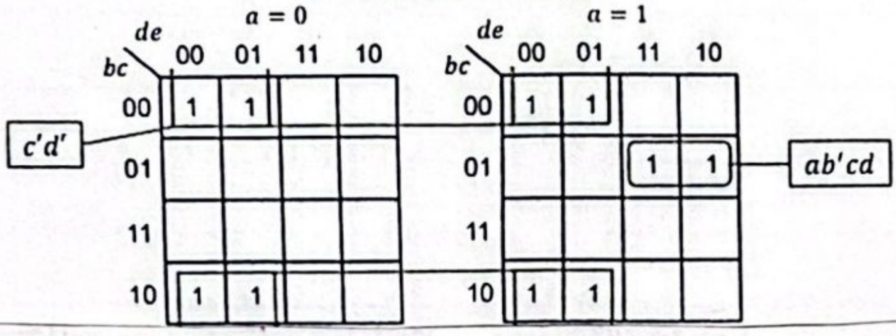
Given: $f(a,b,c,d,e) = \sum(0,1,8,9,16,17,22,23,24,25)$

Draw the 5-Variable K-Map

Obtain a minimal Sum-of-Products expression for f

Solution: $f = c'd' + ab'cd$ (6 literals)

5-Variable K-Map



Example of Five-Variable K-Map with Don't Cares

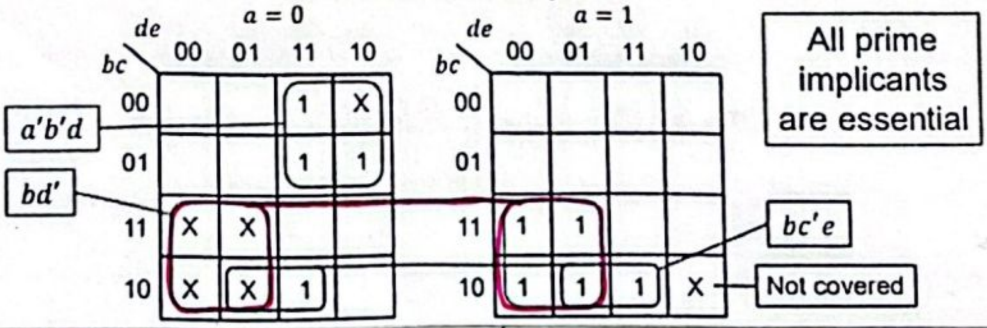
$g(a,b,c,d,e) = \sum_m(3,6,7,11,24,25,27,28,29) + \sum_d(2,8,9,12,13,26)$

Draw the 5-Variable K-Map

Obtain a minimal Sum-of-Products expression for g

Solution: $g = bd' + a'b'd + bc'e$ (8 literals)

5-Variable K-Map



* ليس شرط ان نشمل كل ال don't Cares

Six-Variable Karnaugh Map

cd \ ef	ab = 00: 00	01	11	10	ab = 01: 00	01	11	10	ab = 11: 00	01	11	10	ab = 10: 00	01	11	10
00	m_0	m_1	m_3	m_2	m_{16}	m_{17}	m_{19}	m_{18}	m_{48}	m_{49}	m_{51}	m_{50}	m_{32}	m_{33}	m_{35}	m_{34}
01	m_4	m_5	m_7	m_6	m_{20}	m_{21}	m_{23}	m_{22}	m_{52}	m_{53}	m_{55}	m_{54}	m_{36}	m_{37}	m_{39}	m_{38}
11	m_{12}	m_{13}	m_{15}	m_{14}	m_{28}	m_{29}	m_{31}	m_{30}	m_{60}	m_{61}	m_{63}	m_{62}	m_{44}	m_{45}	m_{47}	m_{46}
10	m_8	m_9	m_{11}	m_{10}	m_{24}	m_{25}	m_{27}	m_{26}	m_{56}	m_{57}	m_{59}	m_{58}	m_{40}	m_{41}	m_{43}	m_{42}

adjacents to m_{37} كل اللي بالهايلايت الأخضر

* Example of Six-Variable K-map

$h(a,b,c,d,e,f) = \sum(2,10,11,18,21,23,29,31,34,41,50,53,55,61,63)$

Draw the 6-Variable K-Map

Obtain a minimal Sum-of-Products expression for h

Solution: $h = c'd'ef' + b\,d\,f + a'b'c\,d'e + a\,b'\,c\,d'e'f$ (18 literals)

cd \ ef	ab = 00: 00	01	11	10	ab = 01: 00	01	11	10	ab = 11: 00	01	11	10	ab = 10: 00	01	11	10
00				1				1				1				1
01						1	1			1	1					
11						1	1			1	1					
10			1	1										1		

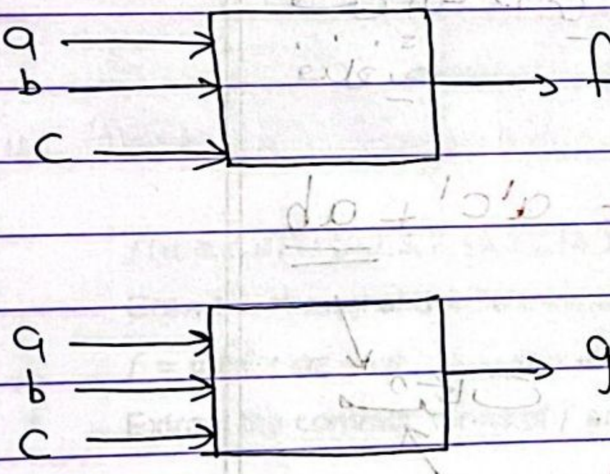
$c'd'ef'$ bdf $a'b'c\,d'e$ $a\,b'\,c\,d'e'f$

Multiple Outputs

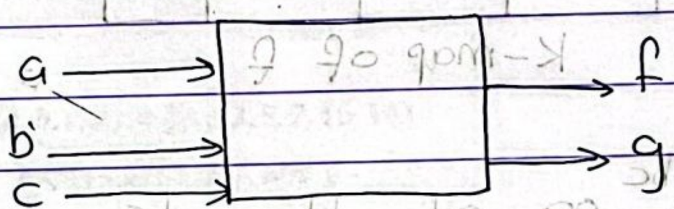
- Suppose that we have 2 functions $f_{(a,b,c)}$ and $g_{(a,b,c)}$ and they have same inputs (a, b, c) but 2 outputs

So we can minimize each function separately

or minimize f and g as one ciruit with 2 outputs



2 separate circuits



One Circuit with two outputs

Ex Given $f_{(a,b,c)} = \Sigma(0,2,6,7)$

and $g_{(a,b,c)} = \Sigma(1,3,6,7)$

① First, we minimize each function separately

② Second, we minimize both functions as one circuit

a \ bc	00	01	11	10
0	1	0	0	1
1	0	0	1	1

X → بتقع هاد لكن فش الو داعي نلفه

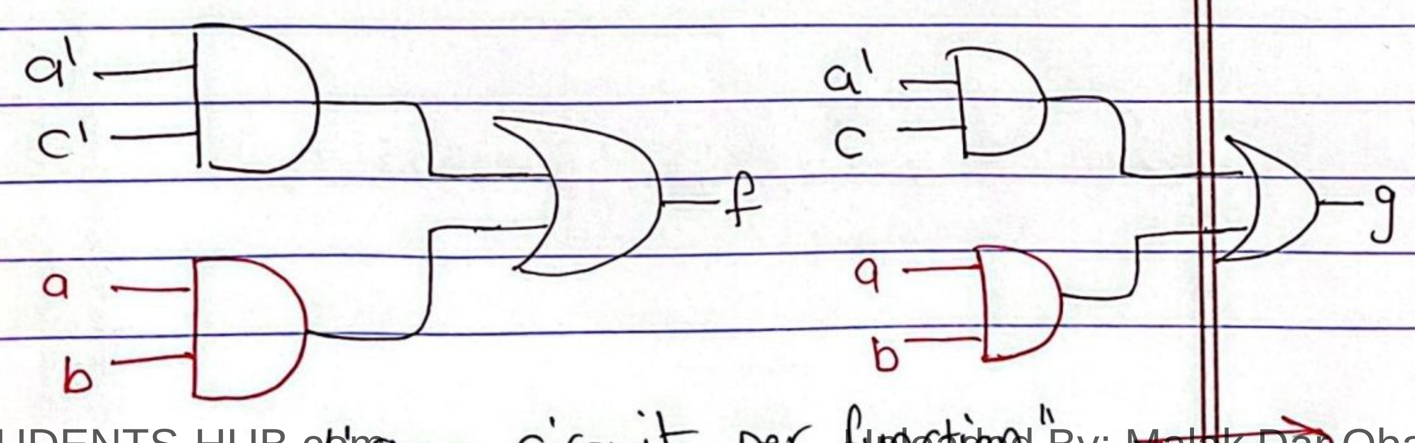
K-map of f ← $f = a'c' + ab$

مشتركات

a \ bc	00	01	11	10
0	0	1	1	0
1	0	0	1	1

$g = a'c + ab$

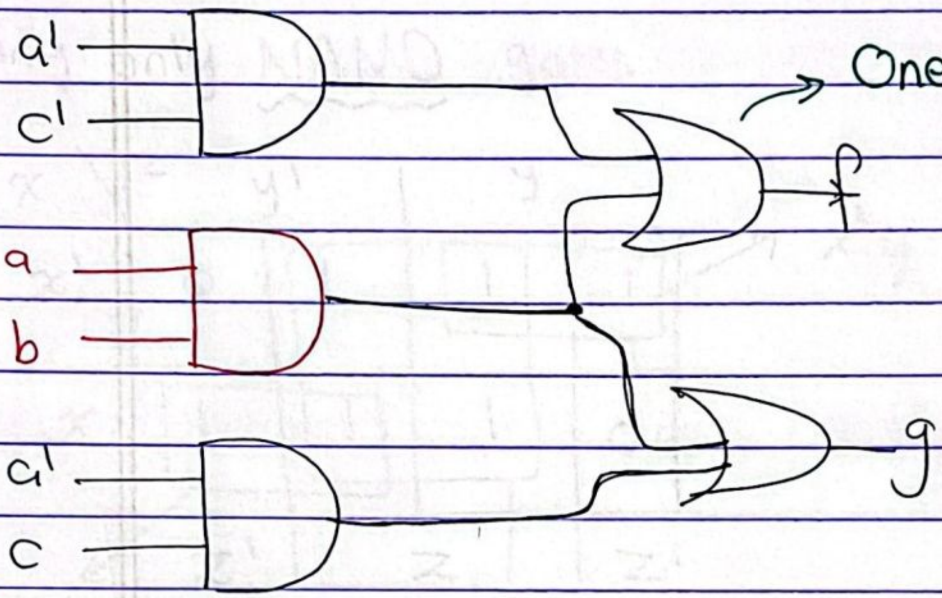
K-map of g



"One circuit per function" →

* بما أنه لدينا term مشترك فنقدر نعمله مشترك

في دارة واحدة للفنكشنين هكذا



One circuit with two outputs

* Another Example :-

$f(a,b,c,d) = \sum(3,5,7,10,11,14,15)$, $g(a,b,c,d) = \sum(1,3,5,7,10,14)$

Draw the K-map and write minimal SOP expressions of f and g

$f = a'bd + ac + cd$ $\qquad$ $g = a'd + acd'$

Extract the common terms of f and g

K-Map of f

ab \ cd	00	01	11	10
00			1	
01		1	1	
11			1	1
10			1	1

K-Map of g

ab \ cd	00	01	11	10
00		1	1	
01		1	1	
11				1
10				1

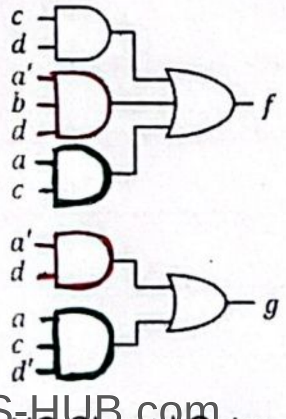
Common Terms

$T_1 = a'd$ and $T_2 = ac$

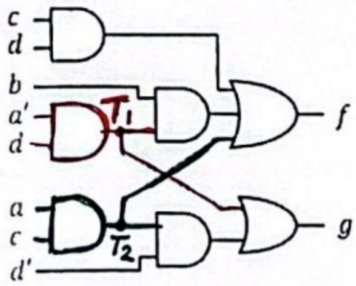
Minimal f and g

$f = T_1 b + T_2 + cd$

$g = T_1 + T_2 d'$



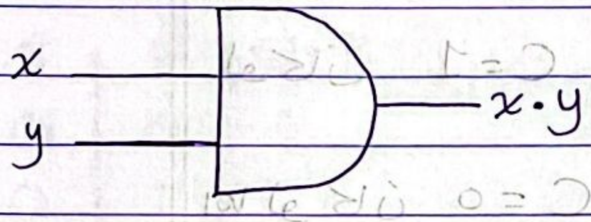
NO Shared Gates



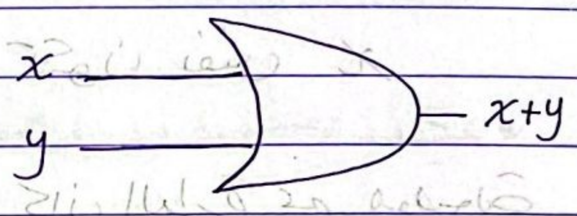
One Circuit
Two Shared Gates

Additional Logic Gates and symbols :

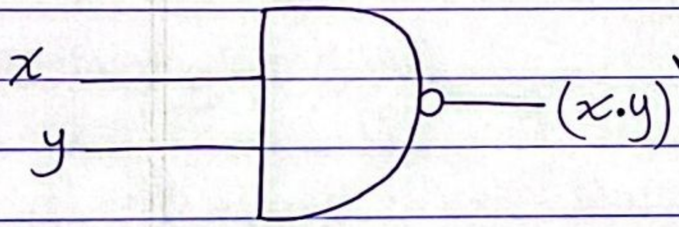
* لماذا كُنّا مهتمين أو نستخدمهن ؟
* التكلفة أقل
* يمكننا ننشئ دائرات بعدد رموز أقل .



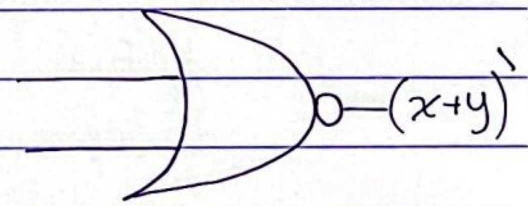
AND gate



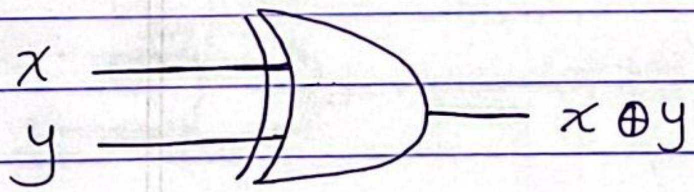
OR gate



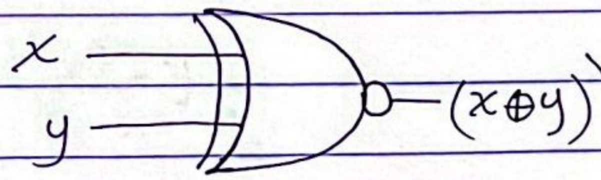
NAND gate



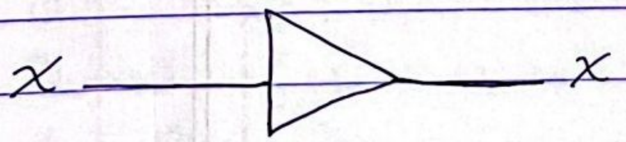
NOR gate



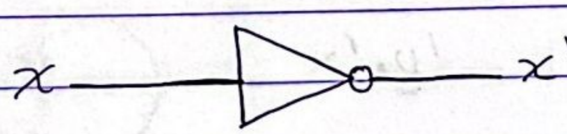
XOR gate



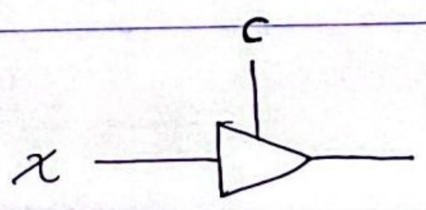
XNOR gate



Buffer

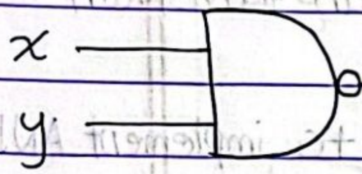


NOT gate (Inverter)



⇒ 3-state gate

NAND Gate (NOT AND)

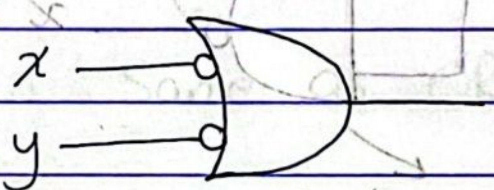


$(x \cdot y)' = x' + y'$

x	y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

عكس نتائج ال AND

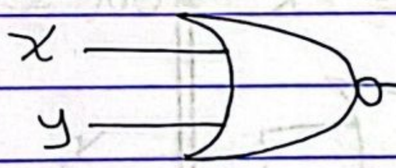
* يمكن رسمها بهذا الشكل أيضاً .



$x' + y'$

inverter

NOR Gate (NOT OR)

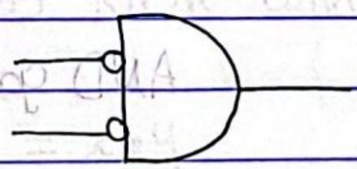


$(x+y)' = x' \cdot y'$

x	y	NOR
0	0	1
0	1	0
1	0	0
1	1	0

عكس نتائج ال OR

* يمكن رسمها هكذا أيضاً



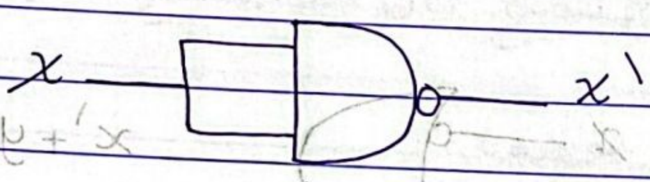
$x' \cdot y'$

NAND Gate is Universal

* This gate can implement any boolean function

* also can be used as inverter or to implement AND/OR

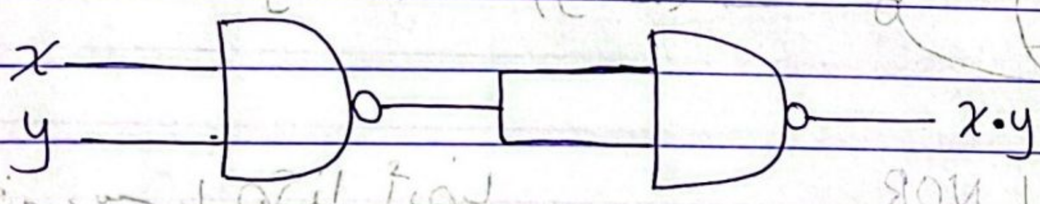
NAND $x = (x \cdot x)' = x'$



inverter

AND is equivalent to NAND with inverted output

$(x \text{ NAND } y)' = ((x \cdot y)')' = \underbrace{x \cdot y}_{\text{AND}}$



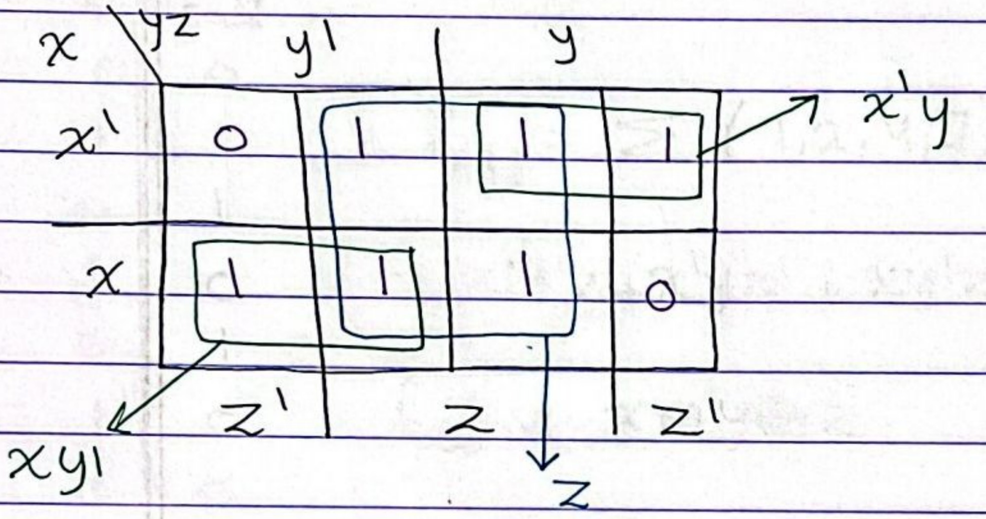
$\Downarrow$

AND gate تكافئ

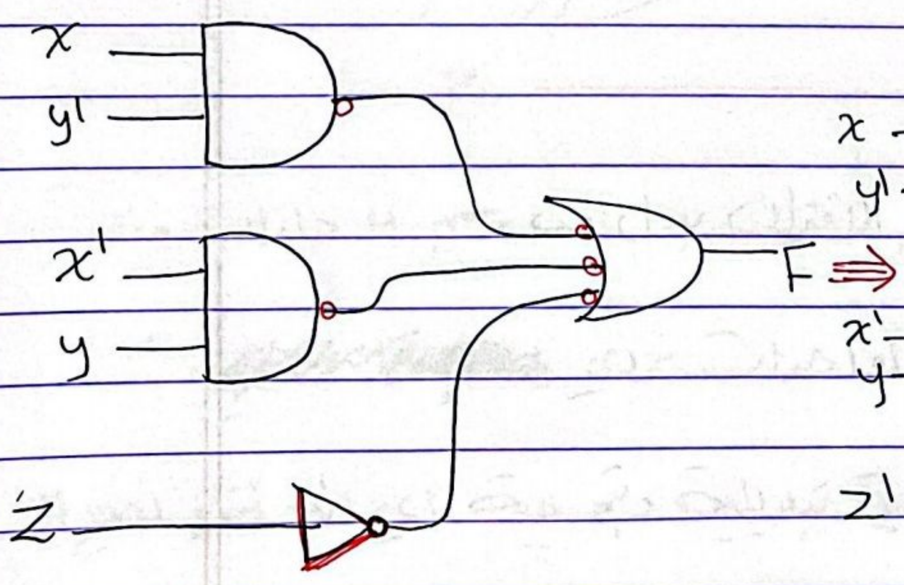
Boolean function with NAND Gates

* Implement the boolean function : $f_{(x,y,z)} = \Sigma(1,2,3,4,5,7)$

Using only NAND gates

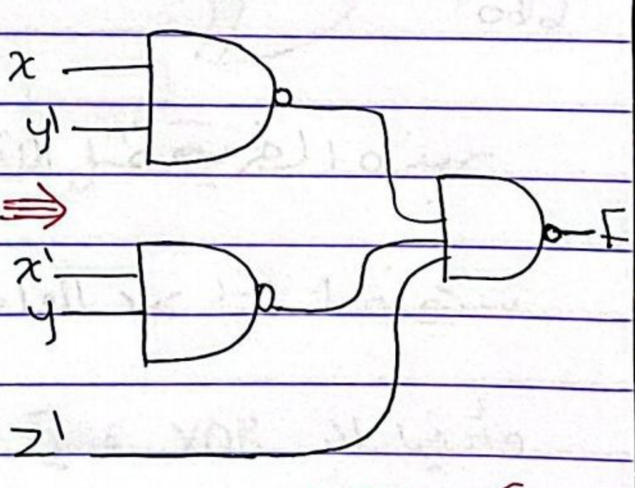


$\therefore F = xy' + x'y + z$



(b)

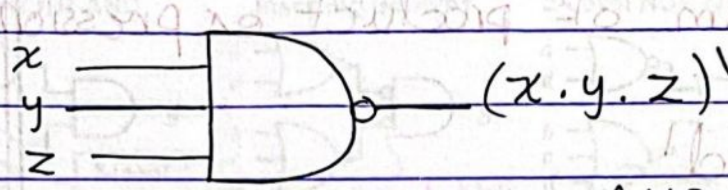
تصبح هكذا



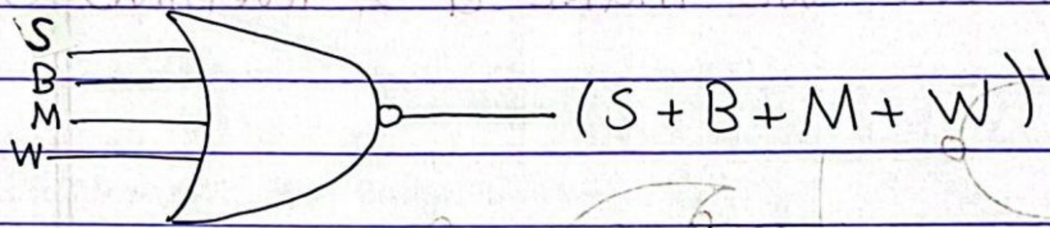
(c)

NAND/NOR can have 2, 3, 4 ... inputs

Ex



3-input NAND gate



4-inputs NOR gate

وممكن نعمل أي عدد من ال inputs سواء NAND/NOR gates

Note that all multiple-input NAND/NOR gates are single gates not a combination of 2-input gates.

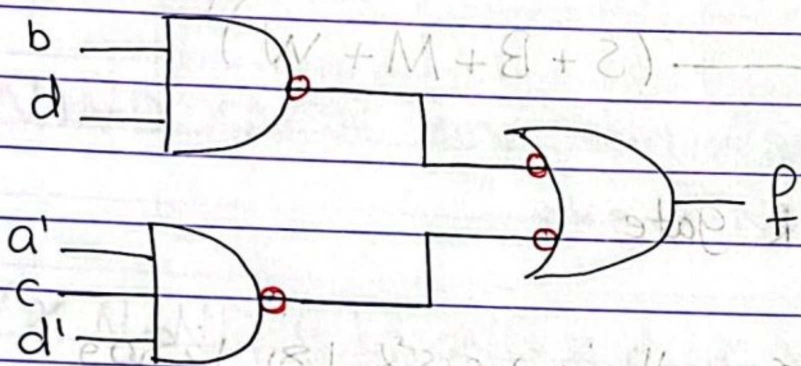
NAND-NAND Implementation

يعني هنا 2 level في كل وحده NAND

if we have sum of product expression like

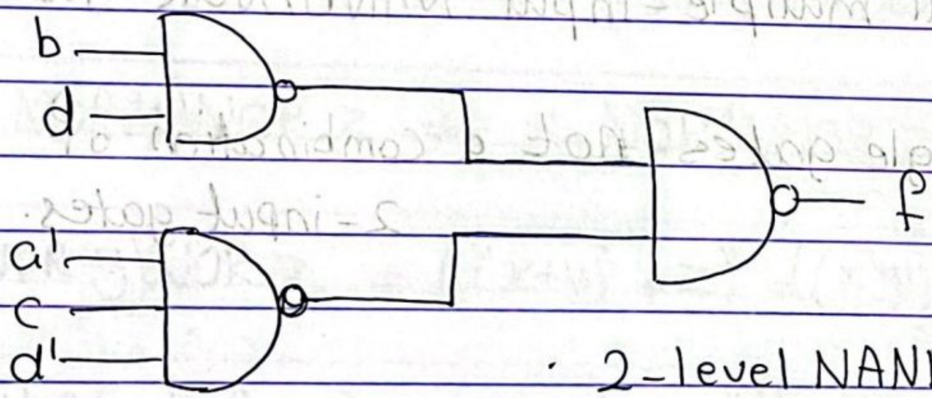
$f = bd + a'cd'$

* First of all we make a 2-level AND-OR circuit



2-level AND-OR

* then we add bubble on the same line (they cancel each other)



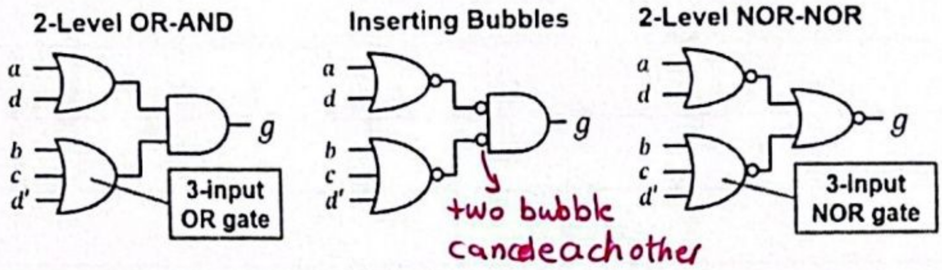
2-level NAND-NAND

NOR-NOR Implementation

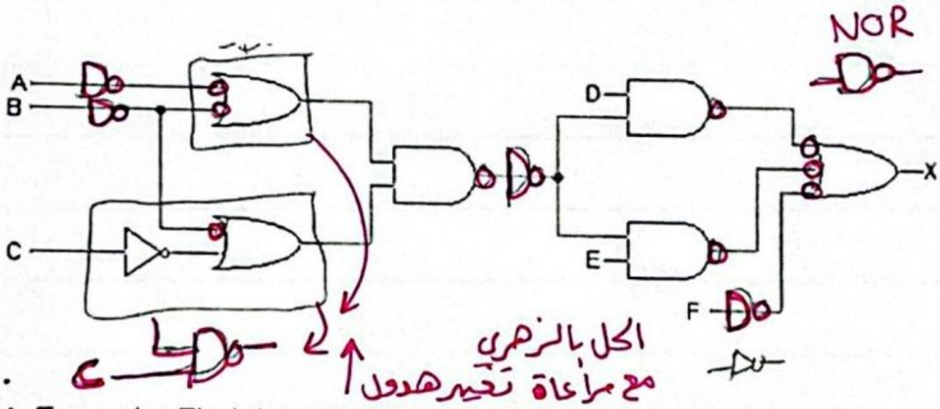
❖ Consider the following product-of-sums expression:

$$g = (a + d)(b + c + d')$$

❖ A 2-level **OR-AND** circuit can be converted easily to a 2-level **NOR-NOR implementation**



Implement the given circuit using only NOR gates

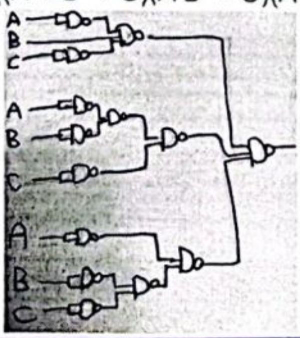
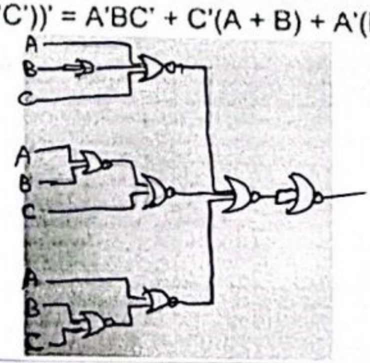


❖ Example: Find the complement of the following expression and implement it using (1) NAND gates, and (2) NOR gates:

$$G(A, B, C) = (A + B' + C)(A'B' + C)(A + B'C')$$

❖ Solution:

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

Boolean Function with NOR Gates

Implement the boolean function

$f_{(x,y,z)} = \Sigma(1,2,3,5,7)$ Using only NOR gates

$x \backslash yz$	$y'z'$	$y'z$	yz	yz'
x'	0	1	1	1
x	0	1	1	0

$y'z'$

xz'

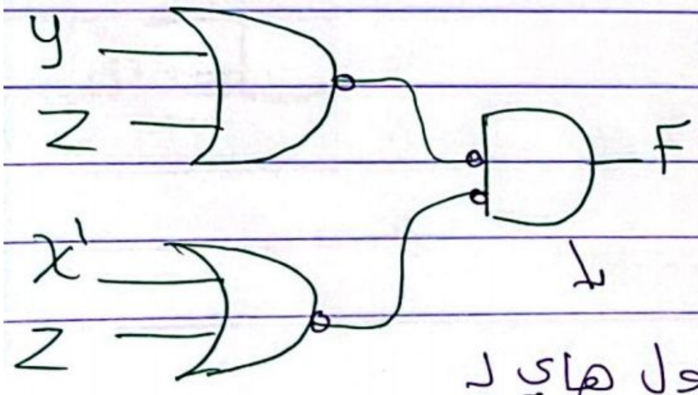
* بعد ما رسمنا ال k-map للاقتران هنا بدنا ان نجد

minimal product of sums to to implement the function using NOR gate

نجد صيغة الاقتران باعتبار الأصفار واحدات ونجد الصيغة ثم نأخذ ال prime لها.

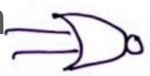
$\therefore F' = y'z' + xz'$

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



* نرسمها ثم نزيح bubbles

أو نحول هاي ل



Exclusive OR / Eclusive NOR

x	y	xOR
0	0	0
0	1	1
1	0	1
1	1	0

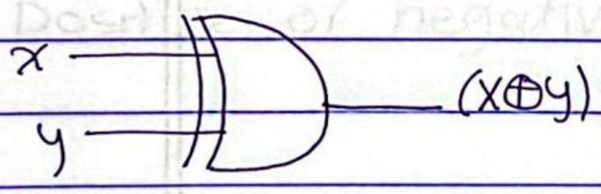
القيم مختلفات

	x	y	$xNOR$
نفس بعض ←	0	0	1
	0	1	0
	1	0	0
نفس بعض ←	1	1	1

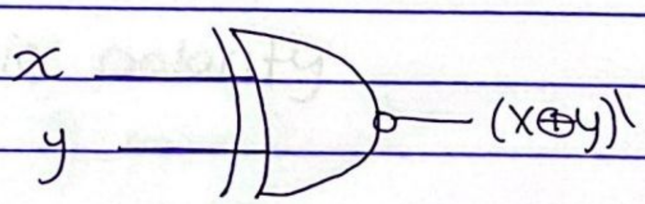
xOR function is:
$x'y + xy'$

$xNOR$ function is:
$xy + x'y'$

(xOR) and $xNOR$ don't exist for more than two inputs because they are complex for exampe: for 3 inputs we use two gates of them not one



XOR gate



XNOR gate

its also known as equivalence

Odd Functions

* إذا كان عدد ال 1 فردي في ال input فإن ال Output = 1

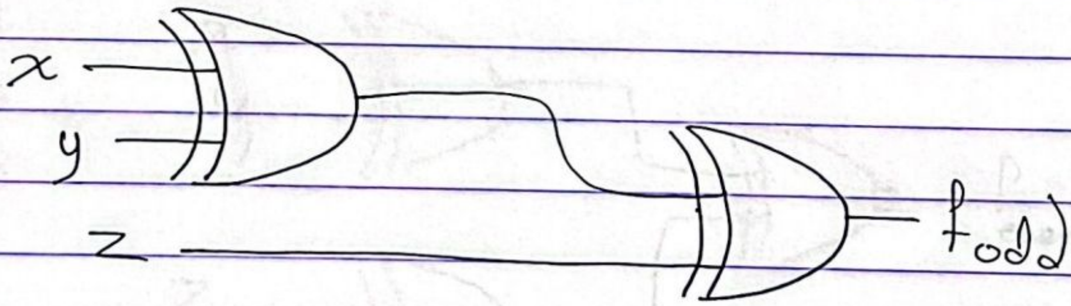
وهو نفسه ال XOR على ال inputs الاثنين عدد ال 1 فردي

x	y	z	f_{odd}
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$f_{odd} = \sum(1,2,4,7)$$

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

$$f_{odd} = x \oplus y \oplus z$$



بهذه الطريقة نمثل الاقتران بدل من رسم 4 دارات والناتج

سيكون 1 اذا عدد الواحدات فردي ~~[illegible]~~

وأيضا ال XOR عملية تبديلية وتجميعية اي انه نقدر نحط ال

X and z at the same gate and y alone

Even Function

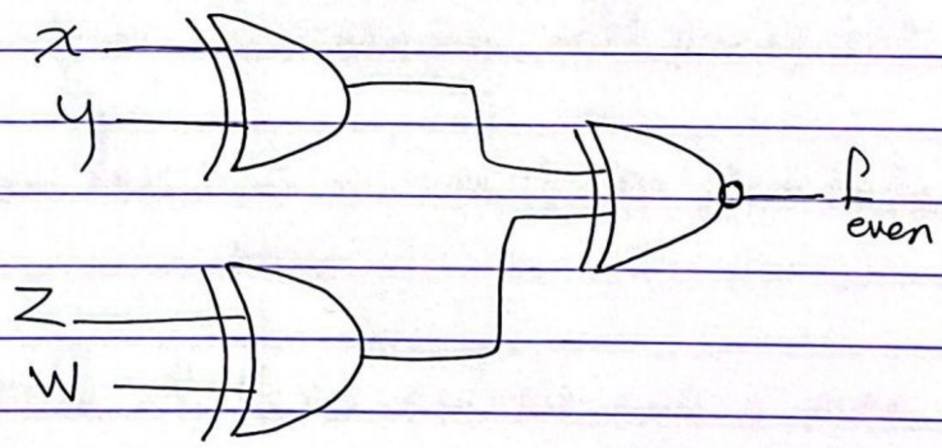
* هنا عدد الواحدات سيكون زوجي وال output لهن واحد.
(the complement of odd function)

- Output is XNOR operation on all inputs

x	y	z	W	f_{even}
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

$$f_{even} = \sum(0,3,5,6,9,10,12,15)$$

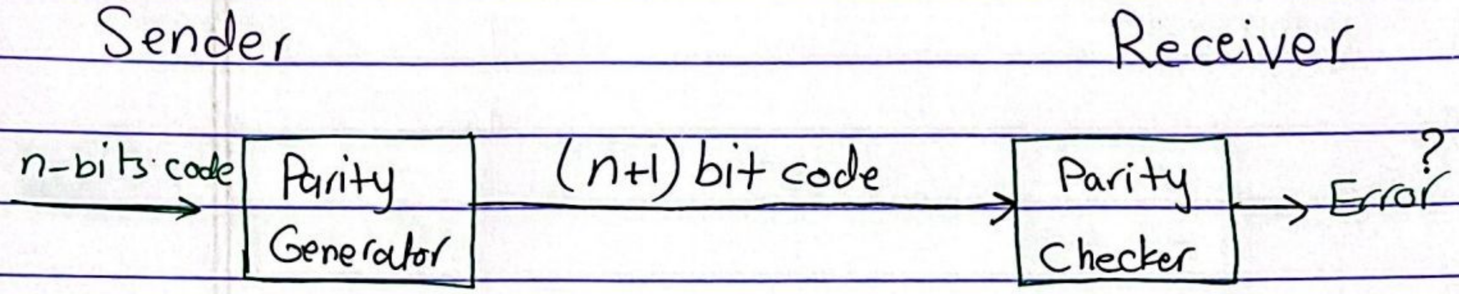
$$f_{even} = (X \oplus Y \oplus Z \oplus W)'$$



استخدمنا دارتين XOR ودارة XNOR

وممكن ننقل ال bubble لأي دارة في ال inputs ونزيلها من عند ال output

Parity Generators and Checkers



* A parity bit is added to the n-bit code

← ال parity checker فقط بفحص اذا في error أم لا
اذا الاوتبوت 1 يعني في error واذا كان 0 يعني لا يوجد

* Odd parity : num of 1's in (n+1) bit code is odd

- Use even function to generate the odd parity bit

- also we use even function to check (n+1) bit code
لـ نتحقق يعني انه ما في error

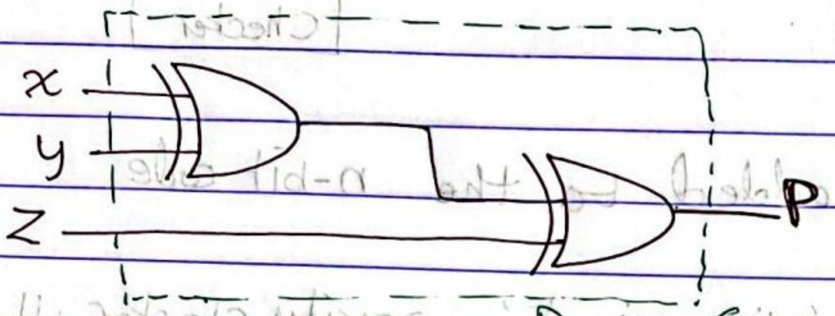
* Even parity : num of 1's in (n+1) bit code is even

- we use odd function to generate the even parity bit

- also we use odd function to check the (n+1) bit code

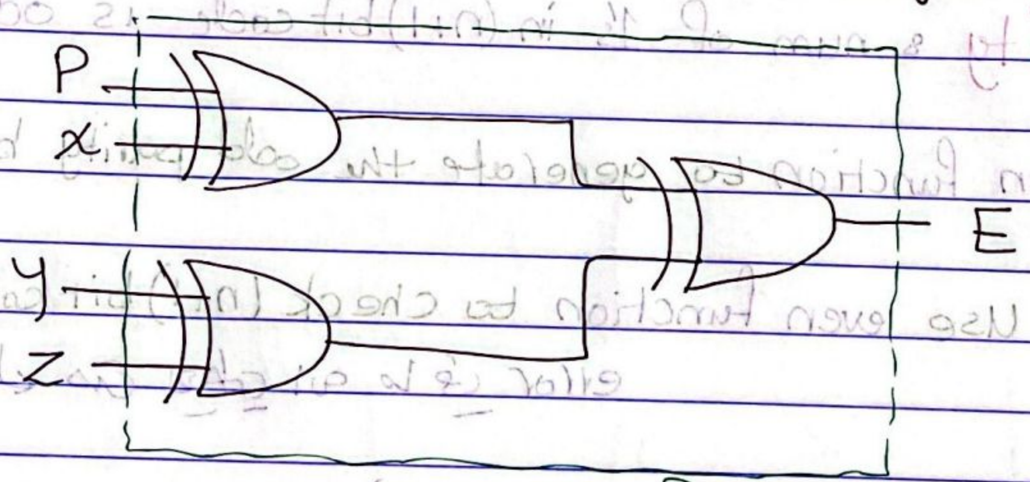
Ex : Design even parity generator & Checker for 3-bit Codes

1- We use 3-bit odd function to generate even parity bit (P)



Parity Generator

2- Use 4-bit odd function to check if there is an error (E) in even parity.



Parity checker

* Given that XYZ = 001 then P = 1
∴ the sender transmits (تنقل) 'PXYZ = 1001'
* في حال حدث تغير على قيمة مثلاً "y و اصبحت = 1
قبل الوصول الى ال checker وهنا لما تصل سيدرك
ايرور لان القيمة ستصل 1011 وليس 1001
then E = 1