ITI1100B

Chapter-3

GRAPHICAL SIMPLIFICATION (Using KARNAUGH MAPS)

Gate-level minimization refers to finding an optimal gate-level implementation of a Boolean function describing a digital circuit.

Algebraic minimization of a Boolean function may be hard to achieve
due to lack of specific rules
whereas
graphical minimization is simple and straightforward.

Karnaugh Map of a Boolean function defined over n variables
is a diagram made up of 2^n squares where each square represents one minterm.

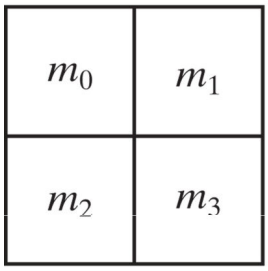
Minterms are placed in a map such that any two neighbours
(two adjacent squares) differ ONLY in one literal.
e.g., xyz and xyz' are two squares sharing an edge,
thus they are neighbours (they are adjacent)

* IN AN **n**-VARIABLE MAP, EACH SQUARE HAS **n** ADJACENT SQUARES.

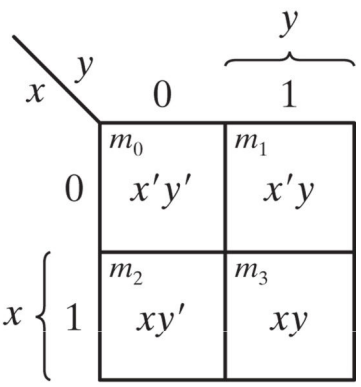
* ANY $\mathbf{2^k}$ ADJACENT SQUARES for k=0,1,2,...n IN AN **n**-VARIABLE MAP
PRESENTS AN AREA THAT GIVES A TERM OF **n-k** VARIABLES
IF n = k ⇒ ENTIRE AREA OF THE MAP ⇒ IDENTITY FUNCTION

Two Variable Karnaugh Map

		Minterms	
x	y	Terms	Designation
0	0	x′y′	m0
0	1	x′y	m1
1	0	xy′	m2
1	1	xy	m3



(a)



(b)

Note that
-**row 0** corresponds to **x′** and **row 1** corresponds to **x**
-**column 0** corresponds to **y′** and **column 1** corresponds to **y**

In the 2-VARIABLE MAP, each square has 2 adjacent squares.
i.e., m1 and m2 are adjacent to m0; m1 and m2 are adjacent to m3.
m0 and m3 are adjacent to m1; m0 and m3 are adjacent to m2.

Any 2^k adjacent squares for k=0,1,2 in a 2 -VARIABLE MAP
is an area that represents a term consisting of 2-k literals.
When k=0, 2^0=1.
e.g., m3 is an area that represents a term consisting of 2 literals , i.e., xy.
When k=1, 2^1=2.
e.g., m2 and m3 is an area that represents a term consisting of 1 literal , i.e., x.
e.g., m0 and m2 is an area that represents a term consisting of 1 literal , i.e., y′.
When k=2, 2^2=4. n=k. The entire map represents a term with no literals, i.e.,. 1 (IDENTITY)

Forming groups of squares in Karnaugh Maps

Groups of squares are formed in considering the following rules:

-Every square containing 1 must be considered at least once

-A square containing 1 can be included in as many groups as desired

-A group must be as large as possible (i.e., largest number of squares)

- The number of squares in a group must be equal to 2^k, i.e., 1, 2, 4, 8...

The simplified Boolean expression obtained as a SOP from a Karnaugh map is not always unique because groups can be formed in different ways.

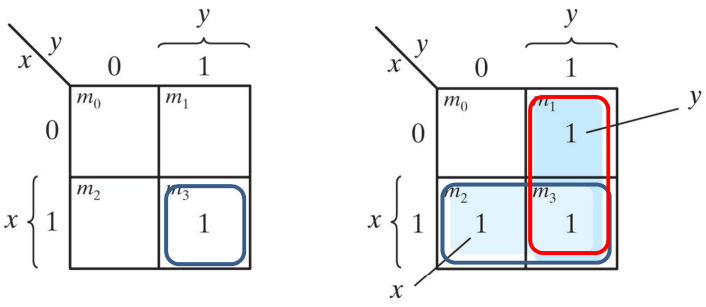
Examples of Two Variable Karnaugh Maps

	x	y	f1
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1

f1(x,y) = Σm(3)

	x	y	f2
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

f2(x,y) = Σm(1,2,3)



(a) xy (b) $x + y$

f1(x,y) = Σm(3)
= m_3
= xy

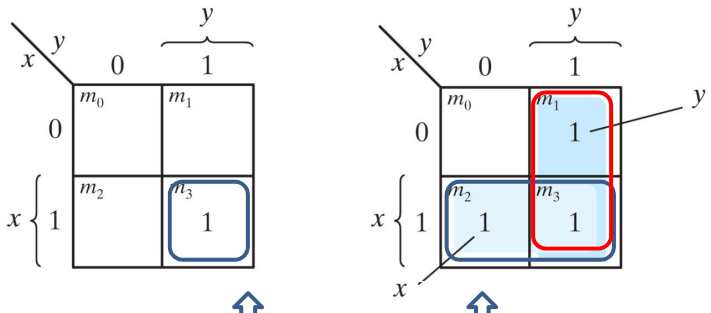
f2(x,y) = Σm(1,2,3)
= m_1+m_2+m_3
= x'y+xy' +xy = (x' + x)y + xy'
= y + xy'= (y + x)(y +y')= (y + x)
= x + y

GIVEN f = e in CSOP, OBTAIN MINIMAL SOP (CSOP to SOP Procedure)

1 - MARK **f** ON THE MAP
(i.e., PUT **1** INTO SQUARES CORRESPONDING TO MINTERMS FOR WHICH **f** TAKES VALUE **1**.

2 - OBTAIN MINIMAL e FOR **f** IN SOP BY COMBINING ADJACENT **1**-SQUARES INTO LARGEST POSSIBLE AREAS OF SIZE 2^k where k = 0,1, 2,... SUCH THAT ALL **1**-SQUARES ARE COVERED WITH MINIMAL NUMBER OF TERMS.

e.g., f1(x,y) = Σm(3) f2(x,y) = Σm(1,2,3)



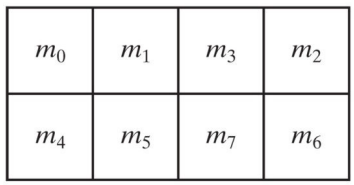
Intersection of area x and area y (a) xy (b) $x + y$ area x , area y

Note that if f is given in CPOS,
first convert CPOS to CSOP and then apply the above procedure.

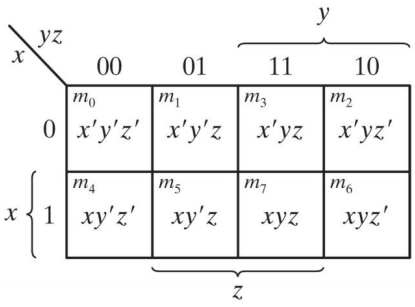
e.g., f1(x,y) = ΠM(0,1,2) = Σm(3)
f2(x,y) = ΠM(0) = Σm(1,2,3)

Three Variable Karnaugh Map

			Minterms	
x	y	z	Terms	Designation
0	0	0	x′y′z′	m0
0	0	1	x′y′z	m1
0	1	0	x′yz′	m2
0	1	1	x′yz	m3
1	0	0	x y′z′	m4
1	0	1	x y′z	m5
1	1	0	x y z′	m6
1	1	1	x y z	m7



(a)



(b)

Note that
-row 0 corresponds to x′ and row 1 corresponds to x
-columns 00 and 01 correspond to y′ and column 11 and 10 corresponds to y
-columns 00 and 10 correspond to z′ and column 01 and 11 corresponds to z
- column 00 corresponds to y′z′
- column 01 corresponds to y′z
- column 11 corresponds to yz
- column 10 corresponds to yz′

In the 3-VARIABLE MAP, each square has 3 adjacent squares.

e.g., m0, m3 and m5 are adjacent to m1; m3, m5 and m6 are adjacent to m7.
m1, m4 and m2 are adjacent to m0; m2, m7 and m4 are adjacent to m6.

Any 2^k adjacent squares for k=0,1,2,3 in a 3 -VARIABLE MAP
is an area that represents a term consisting of 3-k literals.
When k=0, 2^0=1. e.g., m3 is an area that represents an term consisting of 3 literals , i.e., x′yz.
When k=1, 2^1=2. e.g., m2 and m3 is an area that represents an term consisting of 2 literals , i.e., x′y.
When k=2, 2^2=4. e.g., m0, m1,m4 and m5 is an area that represents an term consisting of 1 literal , i.e., y′.
When k=3, 2^3=8. n=k. The entire map represents a term with no literals, i.e.,. 1 (IDENTITY)

FIGURE 3.4 Map for Example 3.1, f(x, y, z) = Σm(2, 3, 4, 5) = xy′ + x′y

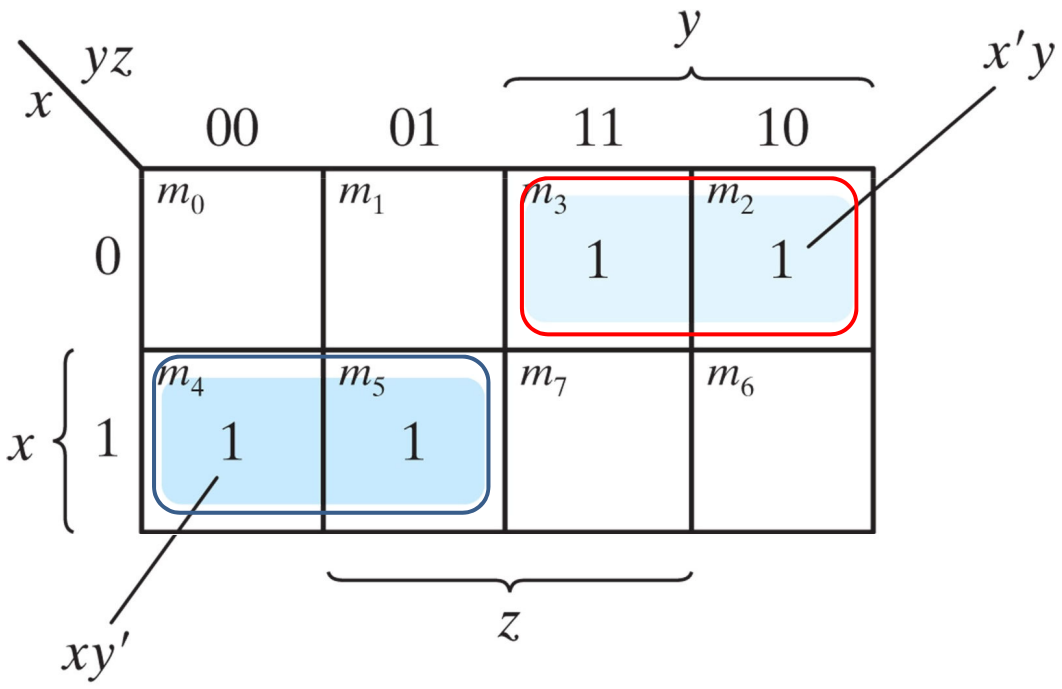
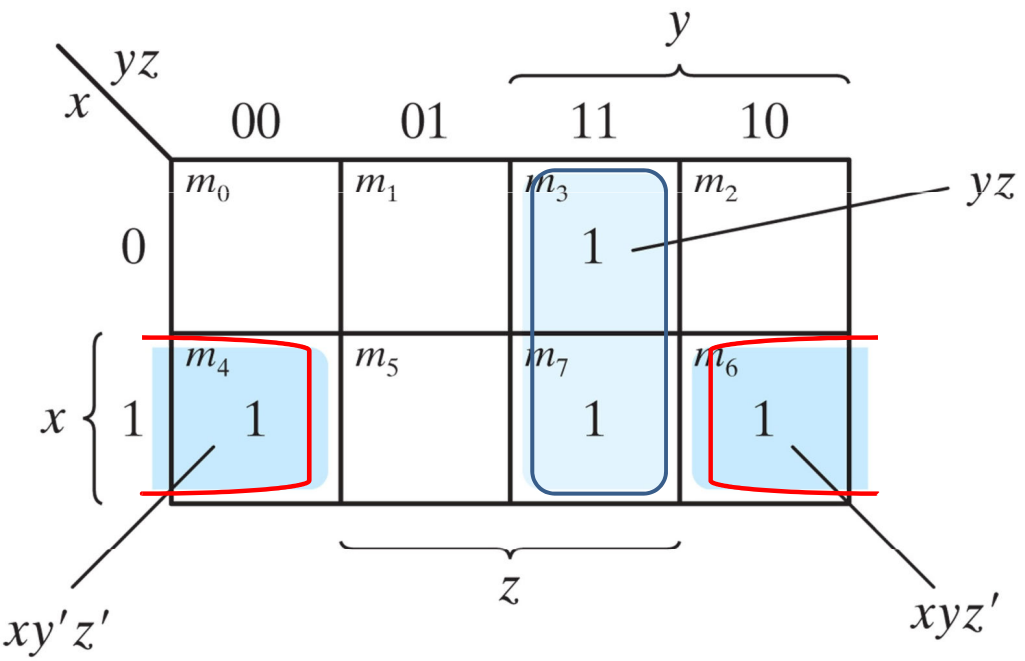
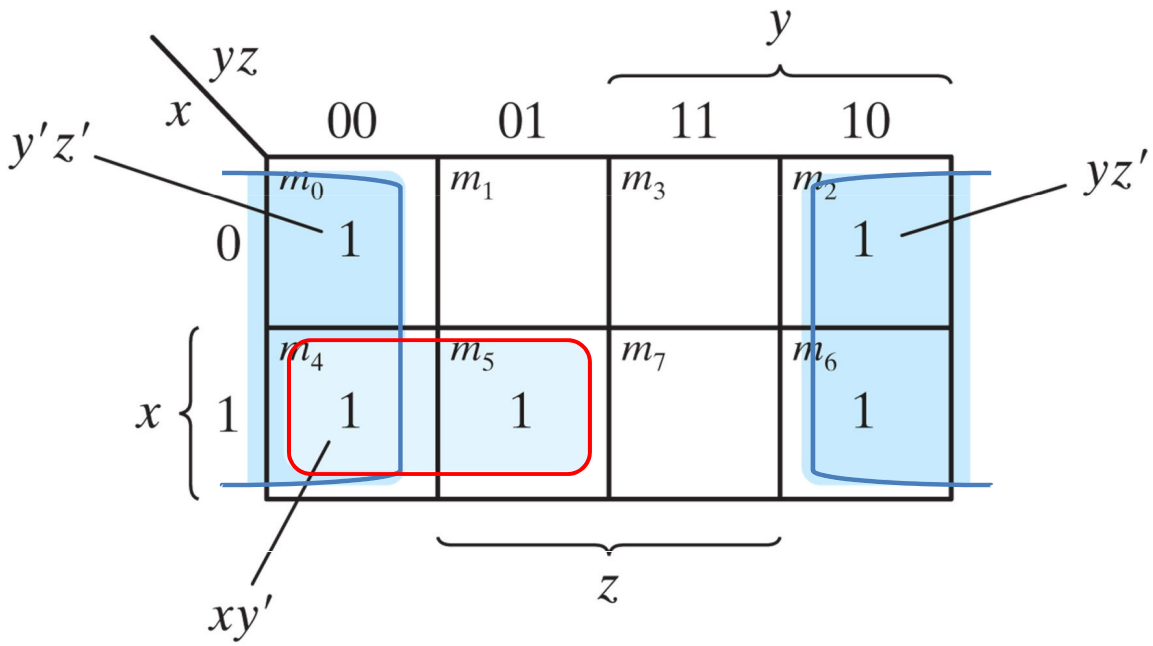


FIGURE 3.5 Map for Example 3.2, f(x, y, z) = Σm(3, 4, 6, 7) = yz + xz′



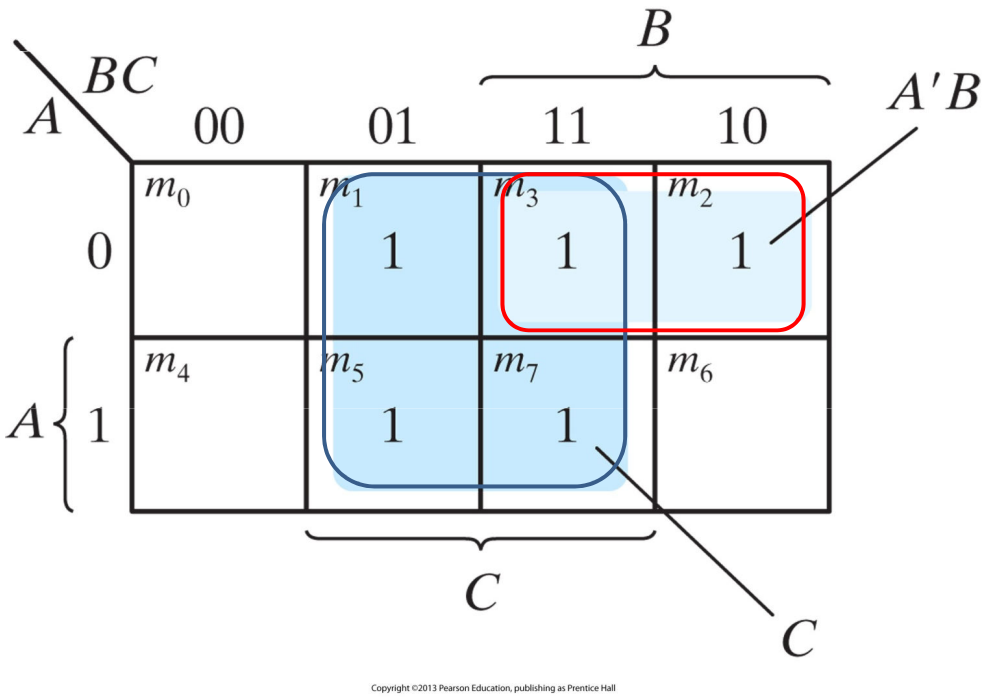
Note: $xy'z' + xyz' = xz'$

FIGURE 3.6 Map for Example 3.3, f(x, y, z) = Σm(0, 2, 4, 5, 6) = z′ + xy′



Note: $y'z' + yz' = z'$

FIGURE 3.7 Map of Example 3.4, f(x, y, z) = Σm(1, 2, 3, 5, 7) = C + A′B

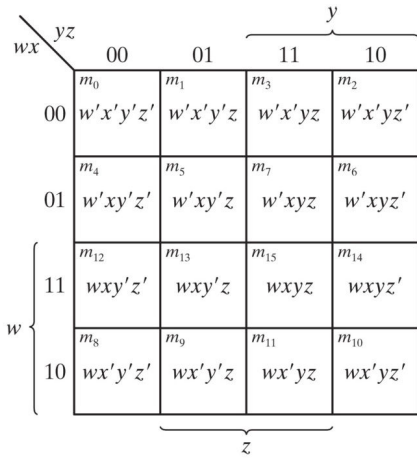


Four Variable Karnaugh Map

				Minterms	
w	x	y	z	Terms	Designation
0	0	0	0	w'x'y'z'	m0
0	0	0	1	w'x'y'z	m1
0	0	1	0	w'x'yz'	m2
0	0	1	1	w'x'yz	m3
0	1	0	0	w'x y'z'	m4
0	1	0	1	w'x y'z	m5
0	1	1	0	w'x yz'	m6
0	1	1	1	w'x yz	m7
1	0	0	0	wx'y'z'	m8
1	0	0	1	wx'y'z	m9
1	0	1	0	wx'yz'	m10
1	0	1	1	wx'yz	m11
1	1	0	0	wx y'z'	m12
1	1	0	1	wx y'z	m13
1	1	1	0	wx yz'	m14
1	1	1	1	wx yz	m15

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)

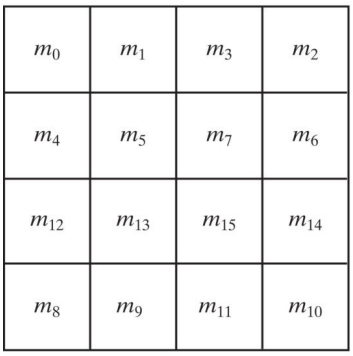


(b)

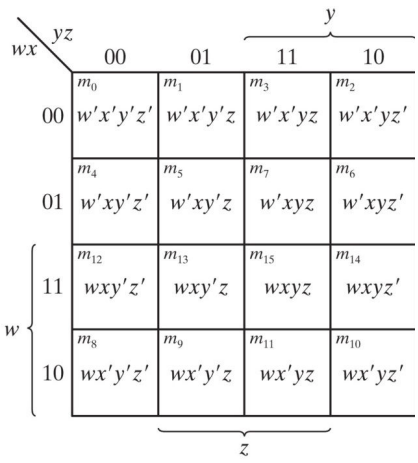
Note that
-rows 00 and 01 correspond to w′ and rows 11 and 10 correspond to w
-rows 00 and 10 correspond to x′ and rows 01 and 11 correspond to x
- -row 00 corresponds to w′x′
- -row 01 corresponds to w′x
- -row 11 corresponds to wx
- -row 10 corresponds to wx′

-columns 00 and 01 correspond to y′ and column 11 and 10 corresponds to y
-columns 00 and 10 correspond to z′ and column 01 and 11 corresponds to z
- - column 00 corresponds to y′z′
- - column 01 corresponds to y′z
- - column 11 corresponds to yz
- - column 10 corresponds to yz′

Four Variable Karnaugh Map



(a)



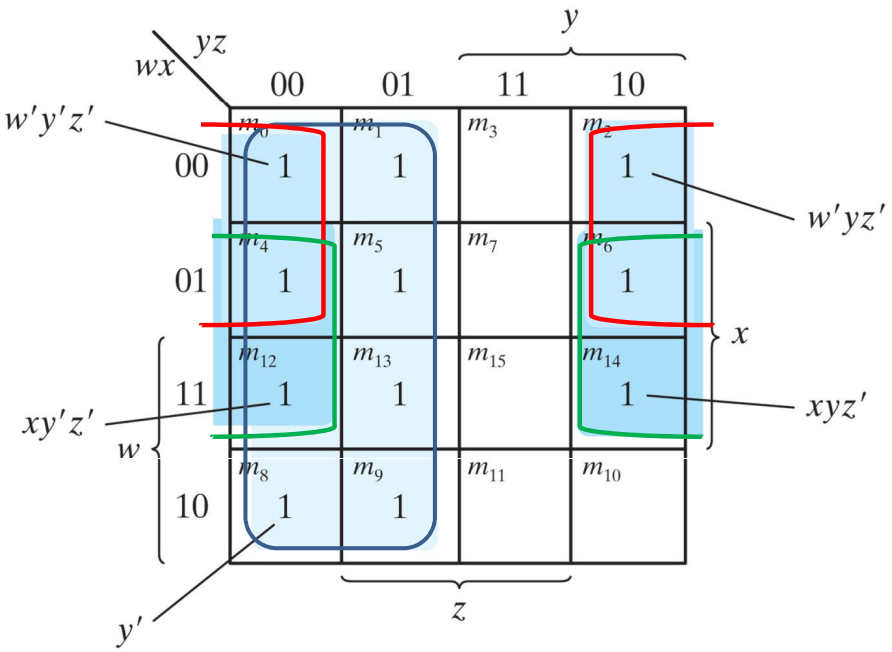
(b)

In the 4-VARIABLE MAP, each square has 4 adjacent squares.

e.g., m0, m3, m5 and m9 are adjacent to m1; m6, m10,m15 and m12 are adjacent to m14.
m1, m4, m2 and m8 are adjacent to m0; m11, m14, m2 and m8 are adjacent to m10.

Any 2^k adjacent squares for k=0,1,2,3,4 in a 4 -VARIABLE MAP
is an area that represents a term consisting of 4-k literals.
When k=0, 2^0=1. e.g., m3 is an area that represents an term consisting of 4 literals , i.e., w′x′yz.
When k=1, 2^1=2. e.g., m2 and m3 is an area that represents an term consisting of 3 literals , i.e., w′x′y.
When k=2, 2^2=4. e.g., m0, m1,m4 and m5 is an area that represents an term consisting of 2 literals, i.e., w′y′.
When k=3, 2^3=8. e.g., m0, m1, m2, m3, m4, m5, m6, and m7 is an area representing a term with 1 literal, i.e., w′.
When k=4, 2^4=16. n=k. The entire map represents a term with no literals, i.e.,. 1 (IDENTITY)

FIGURE 3.9 Map for Example 3.5, f(w, x, y, z) = Σm(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)
= y′ + w′z′ + xz′



Note: $w'y'z' + w'yz' = w'z'$

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

Q: Is the simplified expression in SOP
we obtain from a map by following the above procedure unique?

A: Not necessarily!
There may be many expressions in SOP that are equally simplified.

Q: How can we be sure that we found one of the most simplified expressions?

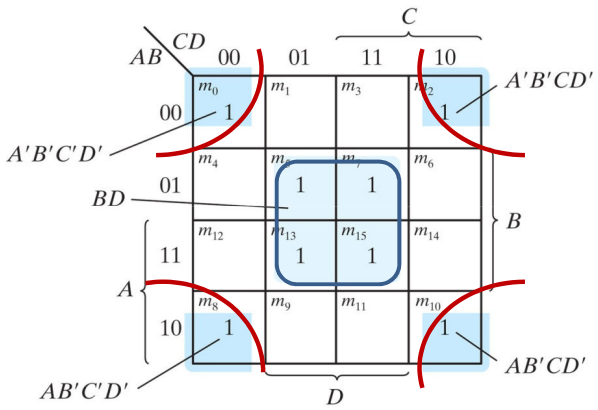
A: Do the following:

1) Identify all prime implicants
(a prime implicant is a product term obtained by combining the maximum possible number of adjacent 1-squares)

2) Identify which prime implicants are essential prime implicants
(a prime implicant is essential if it is the only prime implicant that covers some1-square)

3) A most simplified SOP is obtained by taking
the logical OR of
all the essential prime implicants
and
prime implicants that may be needed to
cover any remaining 1-squares not covered by essential prime implicants.

e.g., f(A,B,C,D) = Σm(0,2,5,7,8,10,13,15)

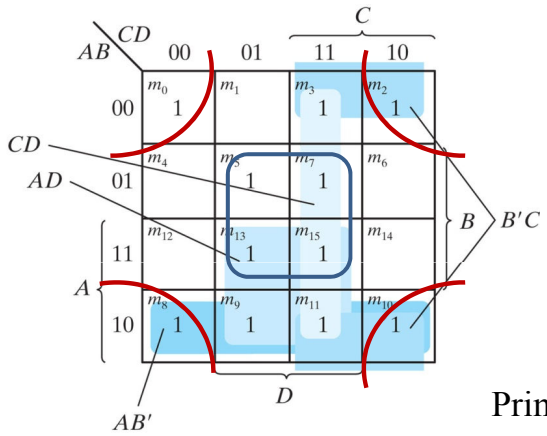


Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$

(a) Essential prime implicants BD and $B'D'$

f(A,B,C,D) = BD + B′D′

e.g., f(A,B,C,D) = Σm(0,2,3,5,7,8,9,10,11,13,15)



(b) Prime implicants CD, $B'C$, AD, and AB'

Prime implicants : BD, B′D′, CD, B′C, AD, AB′

Essential prime implicants : BD, B′D′
(because only BD covers m5 and only B′D′ covers m0)

(1-squares not covered by BD and B′D′ are m3, m9, m11.
We must cover m3, m9, m11 by minimal number of other prime implicants.
There are four pairs of prime implicants to cover them:
These pairs are, CD + AD, CD + AB′, B′C + AD, B′C + AB′)
Thus, there are four equally (most) simplified SOP for f(A,B,C,D) :

f(A,B,C,D) = BD + B′D′ + CD + AD
= BD + B′D′ + CD + AB′
= BD + B′D′ + B′C + AD
= BD + B′D′ + B′C + AB′

GIVEN f = e IN CSOP, OBTAIN MINIMAL POS (CSOP to POS Procedure)

1- MARK **f** ON THE MAP
(i.e., PUT **1** INTO SQUARES CORRESPONDING TO MINTERMS FOR WHICH **f** TAKES VALUE **1**.)

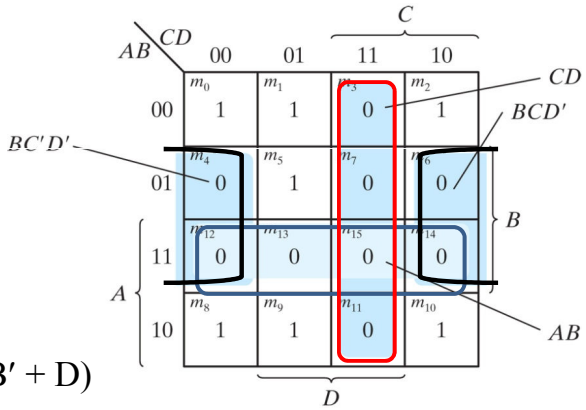
2- **f′** IS DEFINED BY THOSE SQUARES NOT MARKED AS **1**
(**f′** IS DEFINED BY THOSE SQUARES MARKED AS **0**)

3- OBTAIN A SIMPLIFIED EXPRESSION FOR **f′** IN SOP BY COMBINING ADJACENT **0**-SQUARES INTO LARGEST POSSIBLE AREAS OF SIZE 2^k where k = 1, 2,... SUCH THAT ALL **0**-SQUARES ARE COVERED WITH MINIMAL NUMBER OF TERMS

4- TAKE COMPLEMENT OF **f′** IN SOP WHICH GIVES **f** IN POS

e.g., f (A,B,C,D) = Σm(0,1,2,5,8,9,10)

FIGURE 3.12 Map for Example 3.7



f′(A,B,C,D) = AB + CD + BD′
f (A,B,C,D) = (f′)′
= (AB + CD + BD′)′
= (AB)′ (CD)′ (BD′)′
= (A′ + B′)(C′ + D′)(B′ + D)

NOTE: f in SOP is B′ D′ + B′ C′ + A′ C′ D

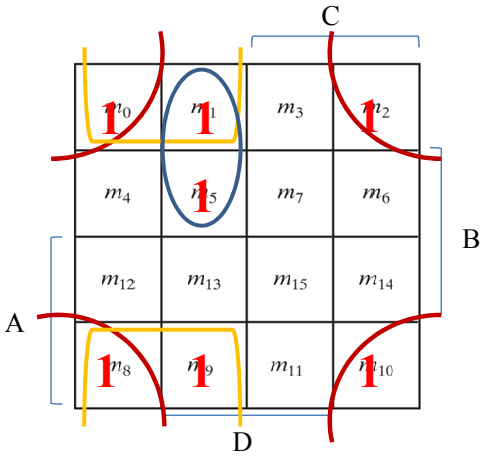
Let's summarize with an example:

Simplify f (A,B,C,D) = Σm(0, 1, 2, 5, 8, 9, 10) into

(a) sum-of-products form

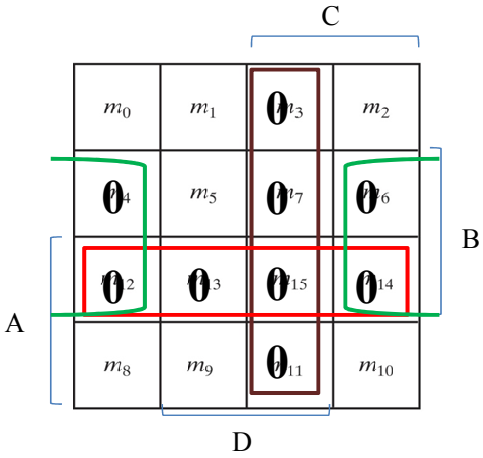
(b) product-of-sums form

(a) sum-of-products form



f (A,B,C,D) = B′D′ + B′C′ + A′C′D

(b) product-of-sums form



f ′ (A,B,C,D) = CD + BD′ + AB

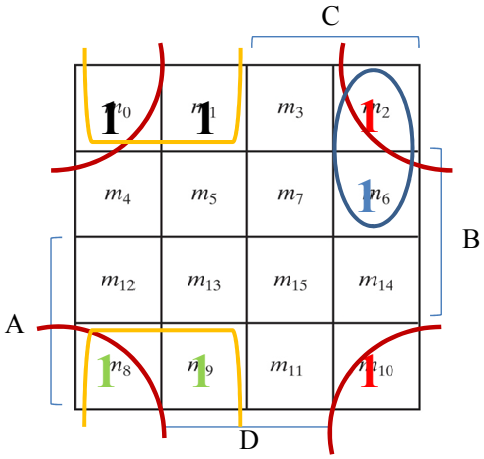
f (A,B,C,D) = (f ′)′

=(C′+D′)(B′+D)(A′+B

Q: How can we verify that what we were given is one of the most simplified expressions of a Boolean function f?

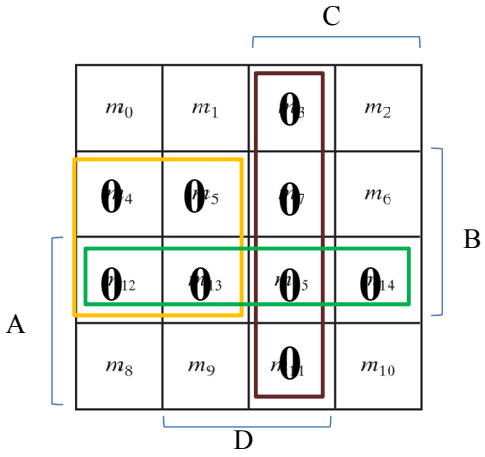
A: Use Karnaugh map method. It will help you to determine whether f is in its most simplified form and if it is not then you can simplify it using the procedures we have seen.

e.g., f (A,B,C,D) = A′B′C′ + B′CD′ + A′BCD′ + AB′C′ in SOP



f (A,B,C,D) = B′D′ + B′C′ + A′CD′

in SOP by combining 1-squares



f ′(A,B,C,D) = CD + BC′ + AB

in SOP by combining 0-squares

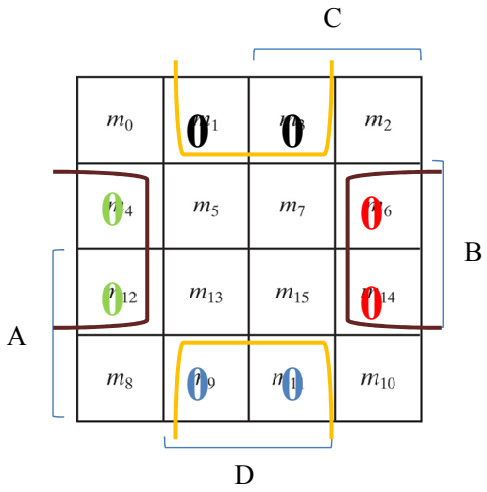
f (A,B,C,D) = (CD + BC′ + AB)′

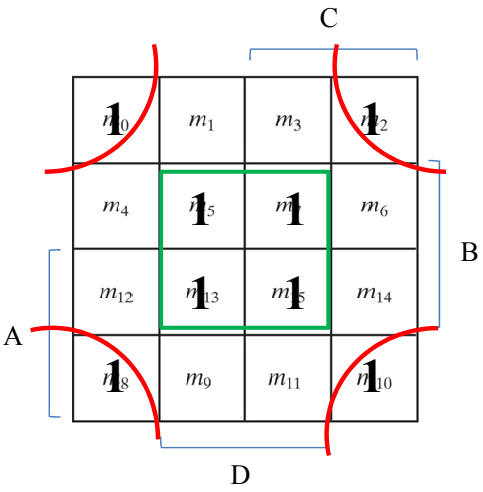
= (C′ + D′)(B′ + C)(A′ + B′)

in POS by taking complement of f ′

e.g., $f(A,B,C,D) = (A+B+D')(B'+C'+D)(A'+B+D')(B'+C+D)$ in POS

$f'(A,B,C,D) = A'B'D + BCD' + AB'D + BC'D'$ in SOP by taking complement of f





$f'(A,B,C,D) = BD' + B'D$

in SOP by combining 0-squares

$f(A,B,C,D) = BD + B'D'$

in SOP by combining 1-squares

$f(A,B,C,D) = (BD' + B'D)'$
$= (B' + D)(B + D')$

in POS by taking complement of f'

SUMMARY

Given f = e in CSOP, obtain minimal SOP
Apply CSOP to SOP Procedure

Given f = e in CPOS, obtain minimal SOP
Obtain CSOP from the given CPOS
Apply CSOP to SOP Procedure

Given f = e in CSOP, obtain minimal POS
Apply CSOP to POS Procedure

Given f = e in CPOS, obtain minimal POS
Obtain CSOP from the given CPOS
Apply CSOP to POS Procedure

SIMPLIFICATION WITH DON'T CARE CONDITIONS

When the value of a function corresponding to a combination of values input variables is not specified, the corresponding minterm or maxterm is represented by DON'T CARE. For example, for a digital circuit whose input is a decimal digit coded in BCD code, we have four input variables: w,x,y, and z, taking 10 combinations of values from 0000 to 1001. The remaining 5 combination of values, 1010, 1011, 1100, 1101, 1111 are not expected to occur at the input lines of the circuit and thus when we define the functionality of the circuit by a Boolean function, the function can take a DON'T CARE value for each of these five combination of values.

A DON'T CARE
-is represented by X
- can take a value 0 or 1 depending on whether we can form a larger area of squares when we combine squares during the application of the Karnaugh Map method.

e.g., $f(w,x,y,z) = \Sigma m(1,3,7,11,15) + \Sigma d(0,2,5)$

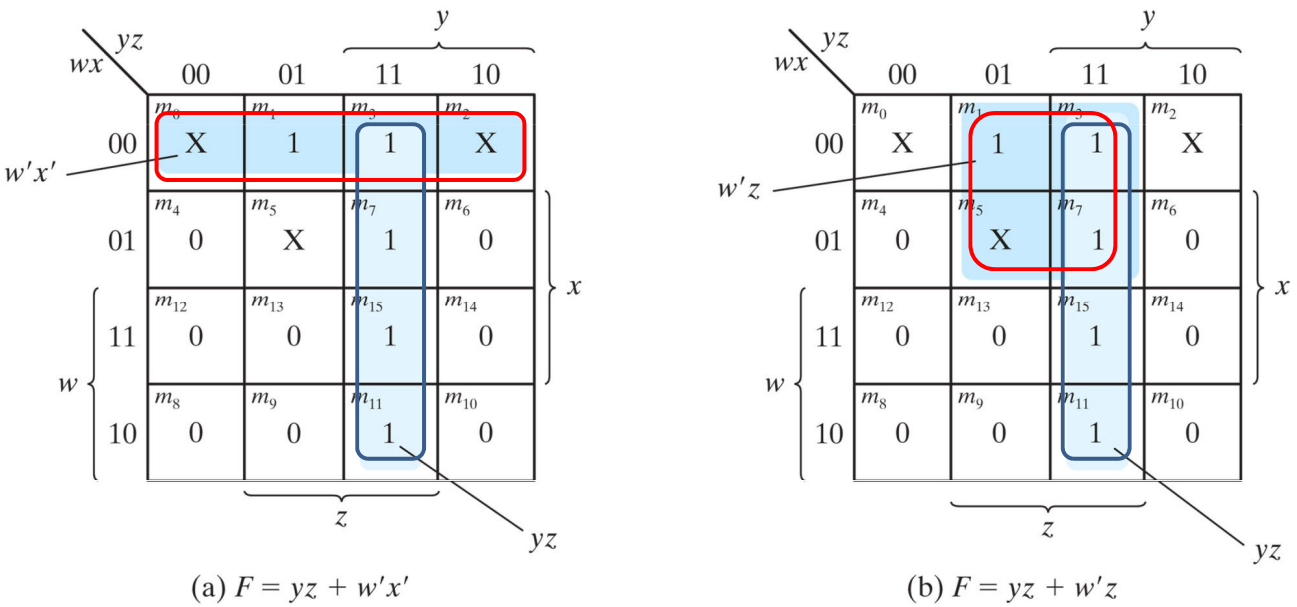


FIGURE 3.15 Example with don't-care conditions

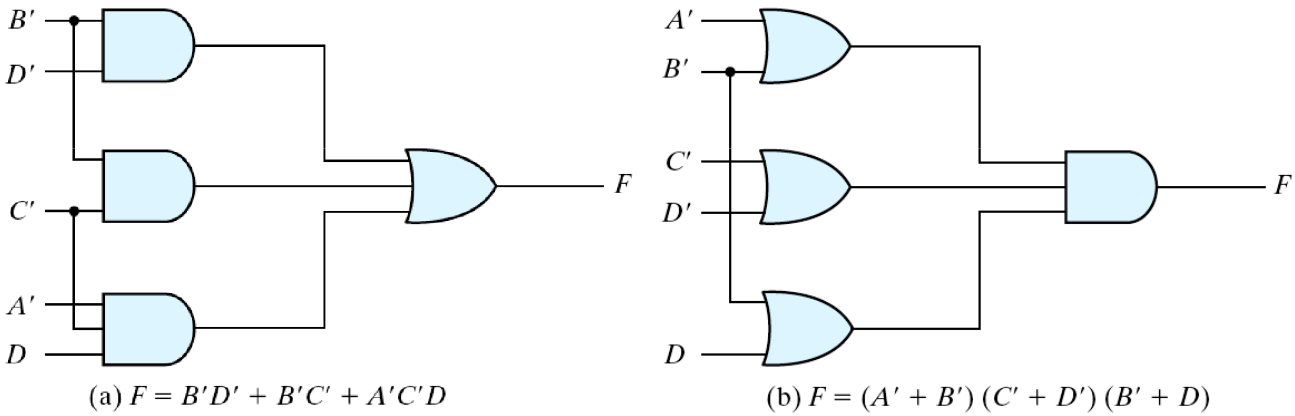
Two-Level Implementations of Boolean Expressions using non-degenerate forms

There are eight nondegenerate forms:

AND-OR	OR-AND
NAND-NAND	NOR-NOR
NOR-OR	NAND-AND
OR-NAND	AND-NOR

Given f in minimal SOP, implement it in AND-OR form

Given f in minimal POS, implement it in OR-AND form

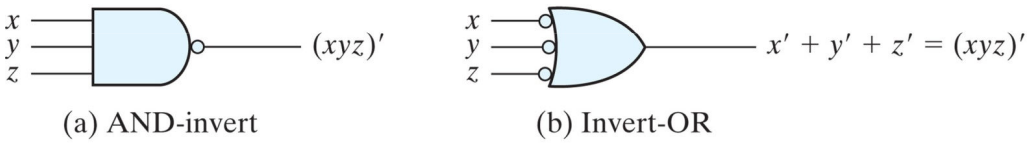


(a) $F = B'D' + B'C' + A'C'D$ (b) $F = (A' + B')(C' + D')(B' + D)$

Remaining six nondegenerate forms for two-level implementations of Boolean expressions involve NAND or NOR gates.

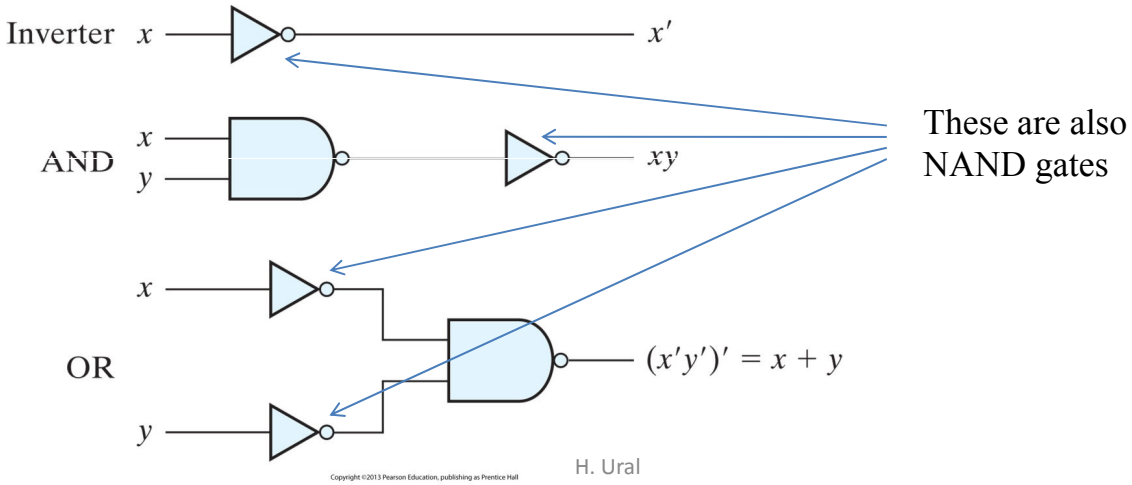
Let's recall the NAND and NOR gates:

A) NAND gates = AND-INVERT / INVERT-OR



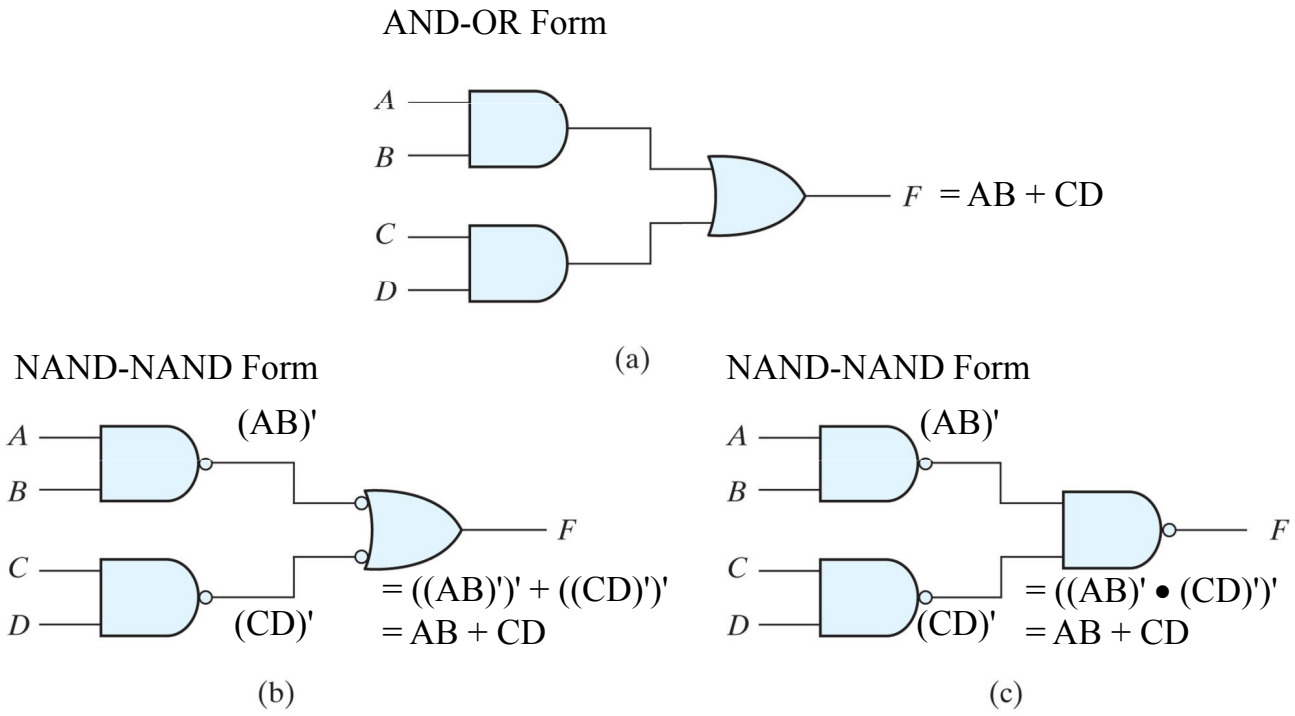
(a) AND-invert (b) Invert-OR

Any Boolean expression can be implemented using only NAND gates:

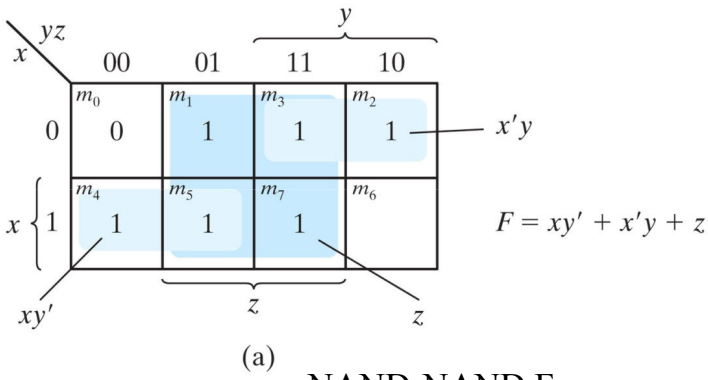


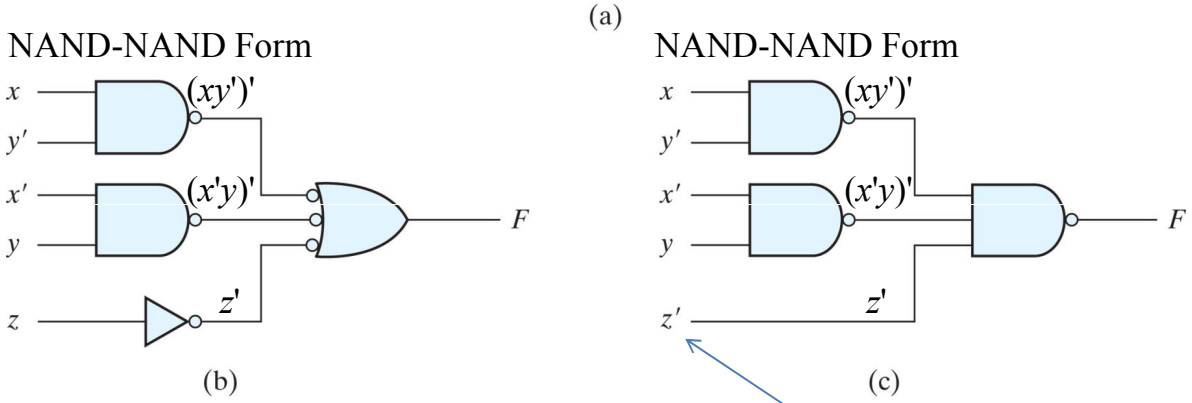
Implementing Minimal SOP in NAND-NAND form

e.g., $F = AB + CD$



e.g., $F = xy' + x'y + z$

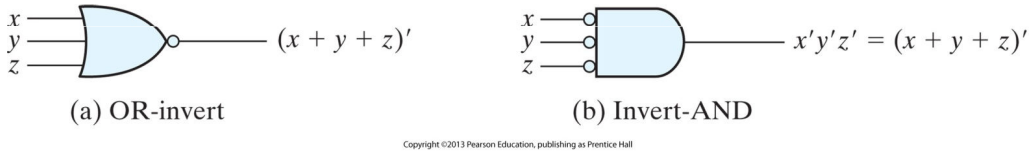




Note that z is primed

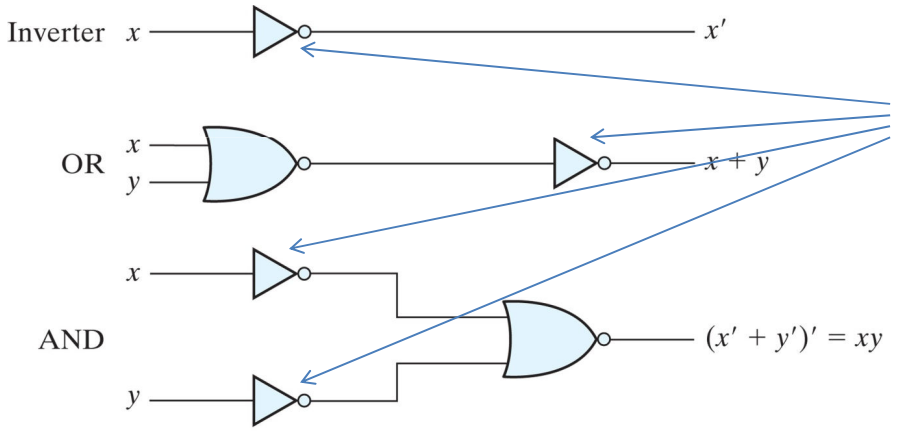
Let's recall the NAND and NOR gates:

B) NOR gates = OR-INVERT / INVERT-AND



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

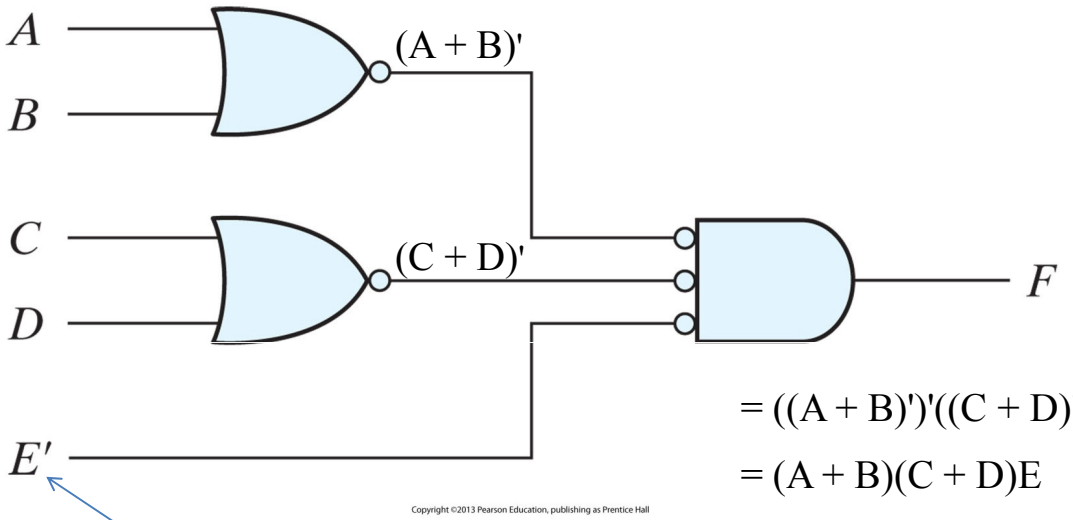
Any Boolean expression can be implemented using only NOR gates:



These are also NOR gates

Implementing Minimal POS in NOR-NOR form

e.g.,: $F = (A + B)(C + D)E$



= ((A + B)')'((C + D)')'(E')'

= (A + B)(C + D)E

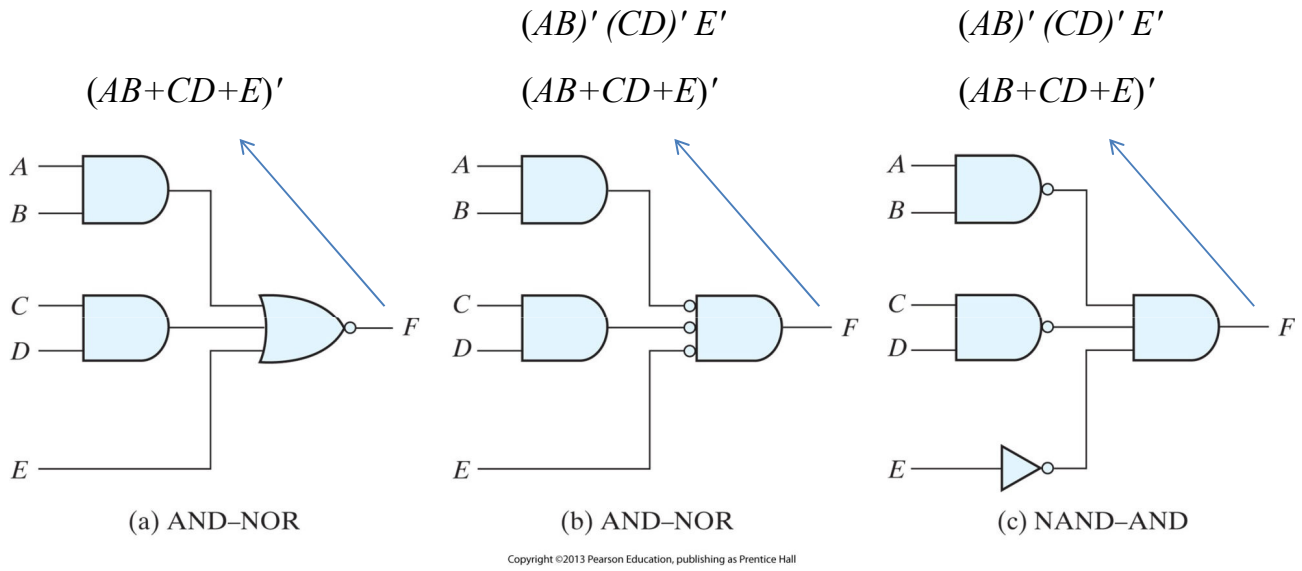
Note that E is primed

AND-OR-INVERT (AOI) Implementations

NAND-AND = AND-NOR = AOI

e.g., $F = (AB+CD+E)'$

What is implemented is $(F')'$
Where $F' = AB+CD+E$ (in SOP form)



OR-AND-INVERT (OAI) Implementation

OR-NAND = NOR-OR = OAI

e.g., $F = ((A+B)(C+D)E)'$

What is implemented is $(F')'$
Where $F' = (A+B)(C+D)E$ (in POS form)

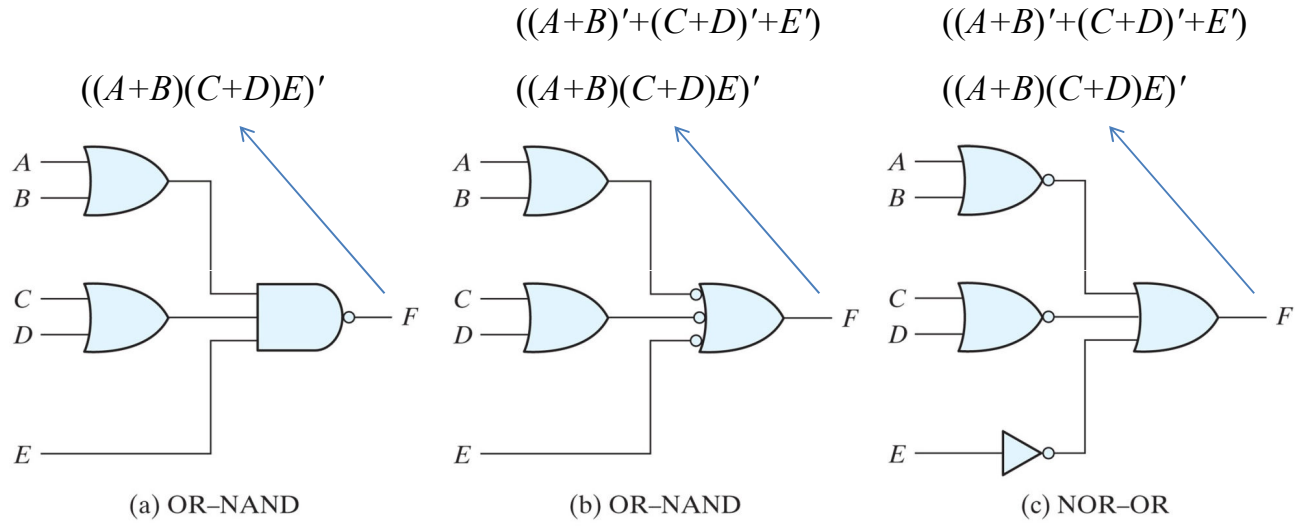
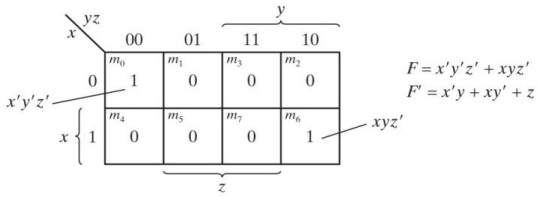


Table 3.2
Implementation with Other Two-Level Forms

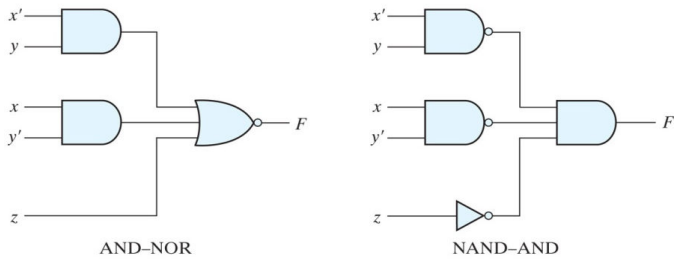
Equivalent Nondegenerate Form (a)	Equivalent Nondegenerate Form (b)*	Implements the Function	Simplify F' into	To Get an Output of
AND–NOR	NAND–AND	AND–OR–INVERT	Sum-of-products form by combining 0's in the map.	F
OR–NAND	NOR–OR	OR–AND–INVERT	Product-of-sums form by combining 1's in the map and then complementing.	F

*Form (b) requires an inverter for a single literal term.

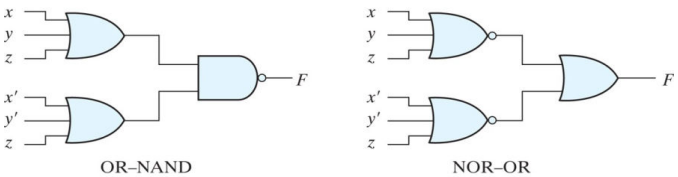
e.g., Two level implementations with AND-NOR, NAND-AND, OR-NAND, NOR-OR



(a) Map simplification in sum of products

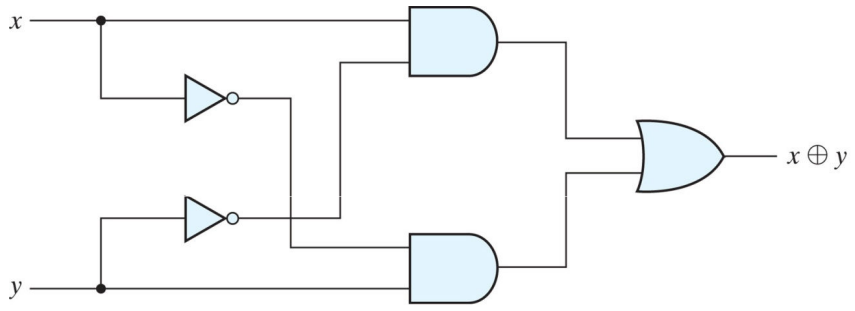


(b) $F = (x'y + xy' + z)'$

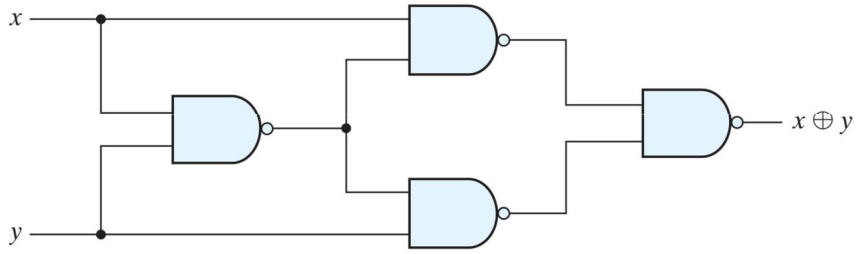


(c) $F = [(x + y + z)(x' + y' + z)]'$

Exclusive-OR implementations



(a) Exclusive-OR with AND–OR–NOT gates



(b) Exclusive-OR with NAND gates

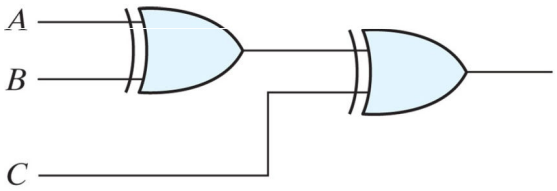
Three-variable exclusive-OR functions

A \ BC	00	01	11	10
0	m_0	m_1 1	m_3	m_2 1
1	m_4 1	m_5	m_7 1	m_6

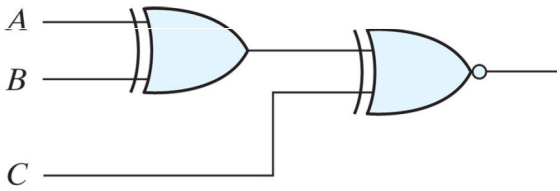
(a) Odd function $F = A \oplus B \oplus C$

A \ BC	00	01	11	10
0	m_0 1	m_1	m_3 1	m_2
1	m_4	m_5 1	m_7	m_6 1

(b) Even function $F = (A \oplus B \oplus C)'$

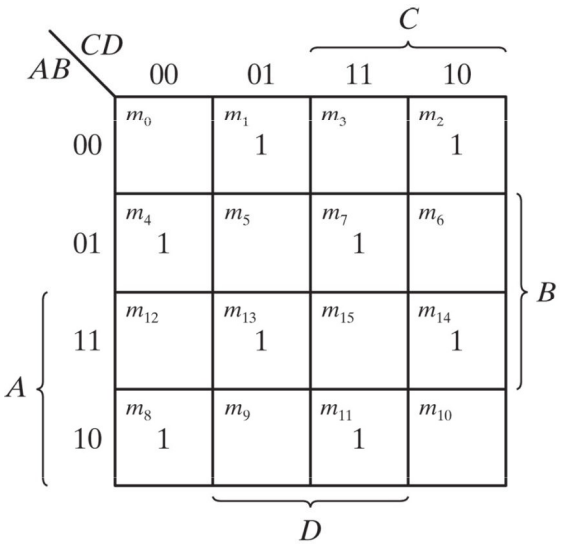


(a) 3-input odd function

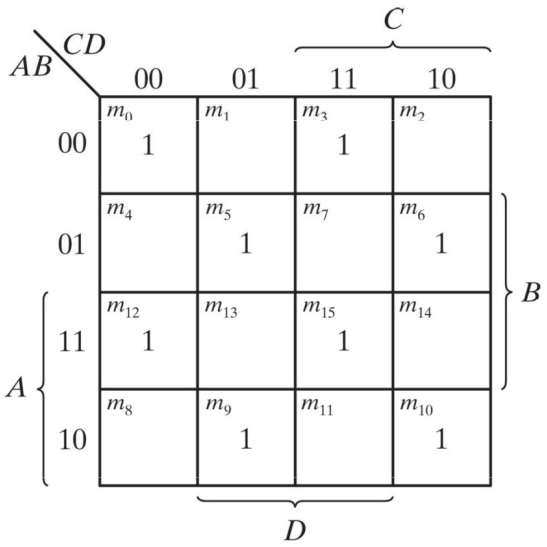


(b) 3-input even function

Four-variable exclusive-OR functions



(a) Odd function $F = A \oplus B \oplus C \oplus D$

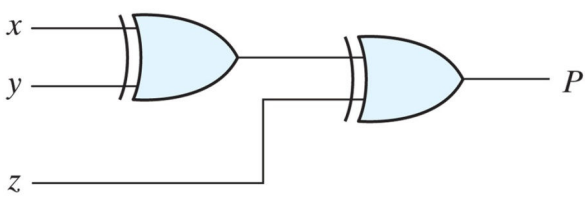


(b) Even function $F = (A \oplus B \oplus C \oplus D)'$

e.g., Use of even parity in error detecting codes

Table 3.3
Even-Parity-Generator Truth Table

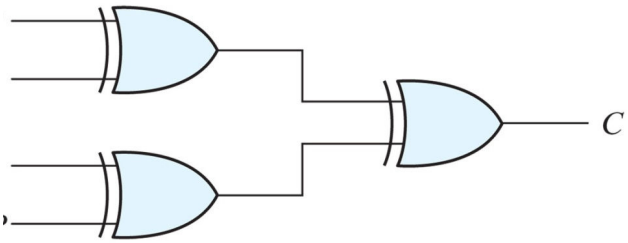
Three-Bit Message			Parity Bit
x	**y**	**z**	**P**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



3-bit even parity generator

Table 3.4
Even-Parity-Checker Truth Table

Four Bits Received				Parity Error Check
x	*y*	*z*	*P*	*C*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0



4-bit even parity checker

Summary

Implementation of f (x,y,z)= x'y'z' + xyz' in SOP in
AND-OR-INVERT form
NAND-NAND form
NOR-NOR form

Implementation of f (x,y,z)= (x+y')(x'+y)z' in POS in
AND-OR-INVERT form
NAND-NAND form
NOR-NOR form