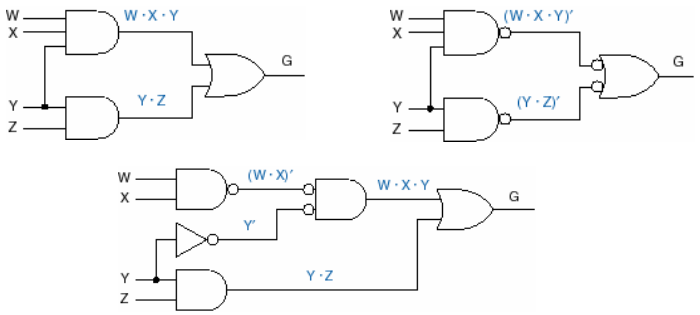
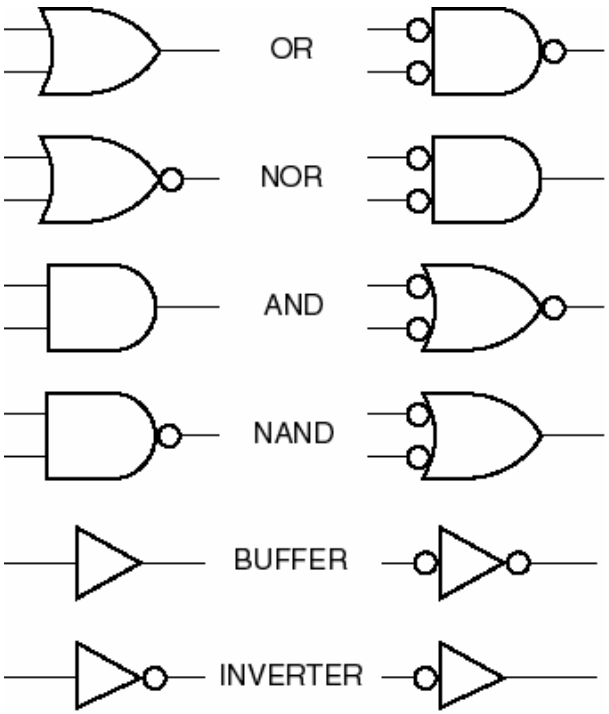


Combinatorial Logic Design Principles

ECGR2181

Chapter 4 Notes



Reading: Chapter 4

Boolean algebra

a.k.a. “switching algebra”

- deals with boolean values -- 0, 1

Positive-logic convention

- analog voltages LOW, HIGH --> 0, 1

Negative logic -- seldom used

Signal values denoted by variables (X, Y, FRED, etc.)

Boolean operators

Complement: X′ (opposite of X)

AND: $X \cdot Y$

OR: $X + Y$

binary operators, described functionally by truth table.

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

X	NOT X
0	1
1	0

More definitions

Literal: a variable or its complement

- X, X′, FRED′, CS_L

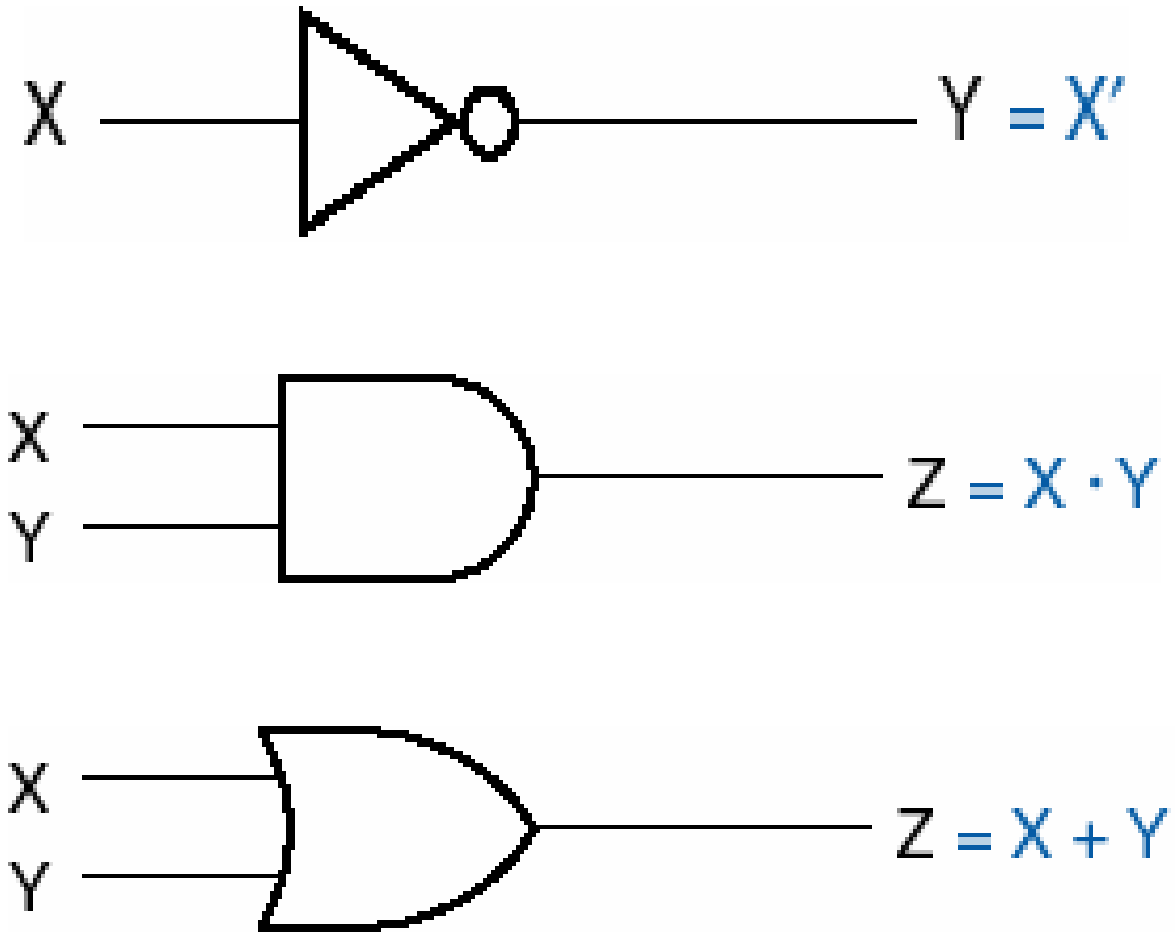
Expression: literals combined by AND, OR, parentheses, complementation

- X+Y
- P · Q · R
- A + B · C
- ((FRED · Z′) + CS_L · A · B′ · C + Q5) · RESET′

Equation: Variable = expression

- P = ((FRED · Z′) + CS_L · A · B′ · C + Q5) · RESET′

Logic symbols



Theorems

(T1)	$X + 0 = X$	(T1′)	$X \cdot 1 = X$	(Identities)
(T2)	$X + 1 = 1$	(T2′)	$X \cdot 0 = 0$	(Null elements)
(T3)	$X + X = X$	(T3′)	$X \cdot X = X$	(Idempotency)
(T4)	$(X')' = X$			(Involution)
(T5)	$X + X' = 1$	(T5′)	$X \cdot X' = 0$	(Complements)

Proofs by perfect induction

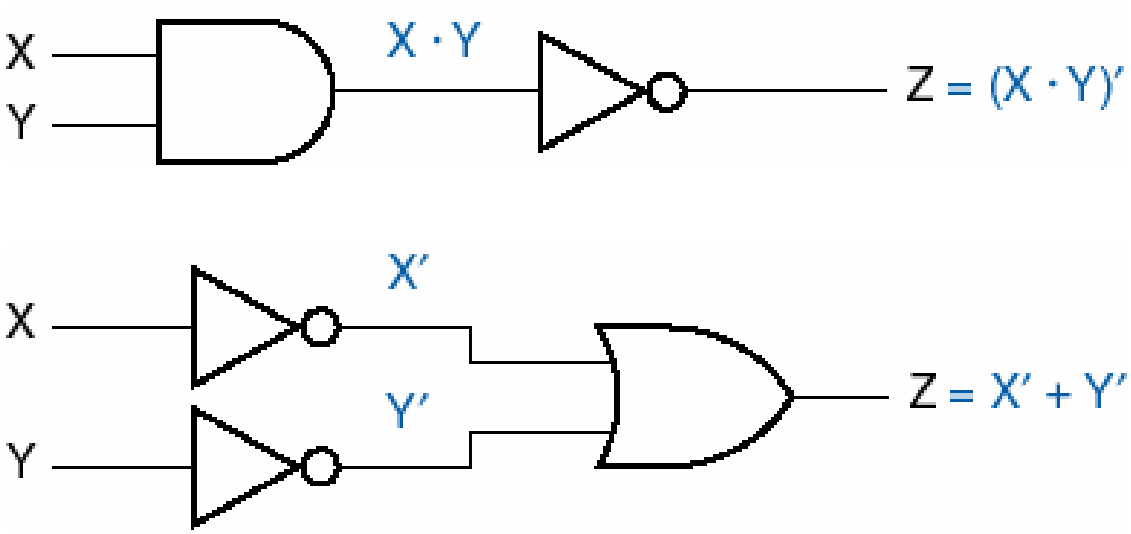
More Theorems

(T6)	$X + Y = Y + X$	(T6′)	$X \cdot Y = Y \cdot X$	(Commutativity)
(T7)	$(X + Y) + Z = X + (Y + Z)$	(T7′)	$(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$	(Associativity)
(T8)	$X \cdot Y + X \cdot Z = X \cdot (Y + Z)$	(T8′)	$(X + Y) \cdot (X + Z) = X + Y \cdot Z$	(Distributivity)
(T9)	$X + X \cdot Y = X$	(T9′)	$X \cdot (X + Y) = X$	(Covering)
(T10)	$X \cdot Y + X \cdot Y' = X$	(T10′)	$(X + Y) \cdot (X + Y') = X$	(Combining)
(T11)	$X \cdot Y + X' \cdot Z + Y \cdot Z = X \cdot Y + X' \cdot Z$			(Consensus)
(T11′)	$(X + Y) \cdot (X' + Z) \cdot (Y + Z) = (X + Y) \cdot (X' + Z)$			

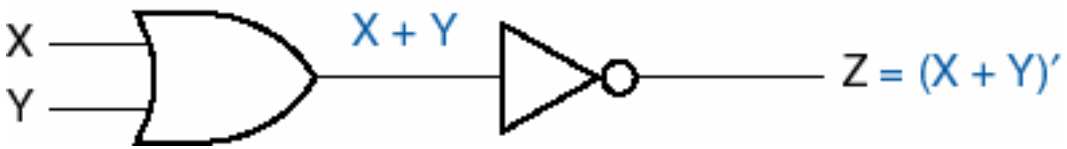
N-variable Theorems

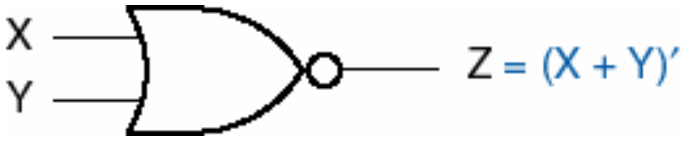
(T12)	$X + X + \cdots + X = X$	(Generalized idempotency)
(T12′)	$X \cdot X \cdot \ \cdots \ \cdot X = X$	
(T13)	$(X_1 \cdot X_2 \cdot \ \cdots \ \cdot X_n)' = X_1' + X_2' + \cdots + X_n'$	(DeMorgan's theorems)
(T13′)	$(X_1 + X_2 + \cdots + X_n)' = X_1' \cdot X_2' \cdot \ \cdots \ \cdot X_n'$	
(T14)	$[F(X_1, X_2, \ldots, X_n, +, \cdot\ )]' = F(X_1', X_2', \ldots, X_n', \cdot\ , +)$	(Generalized DeMorgan's theorem)
(T15)	$F(X_1, X_2, \ldots, X_n) = X_1 \cdot F(1, X_2, \ldots, X_n) + X_1' \cdot F(0, X_2, \ldots, X_n)$	(Shannon's expansion theorems)
(T15′)	$F(X_1, X_2, \ldots, X_n) = [X_1 + F(0, X_2, \ldots, X_n)] \cdot [X_1' + F(1, X_2, \ldots, X_n)]$	

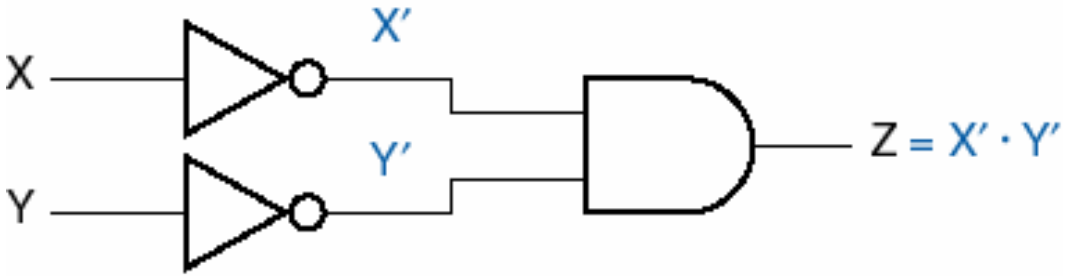
DeMorgan Symbol Equivalence

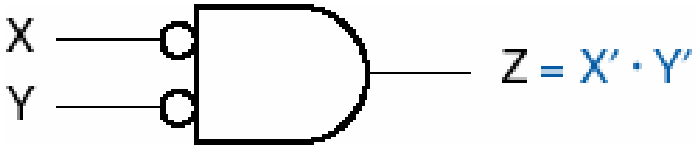


Likewise for OR

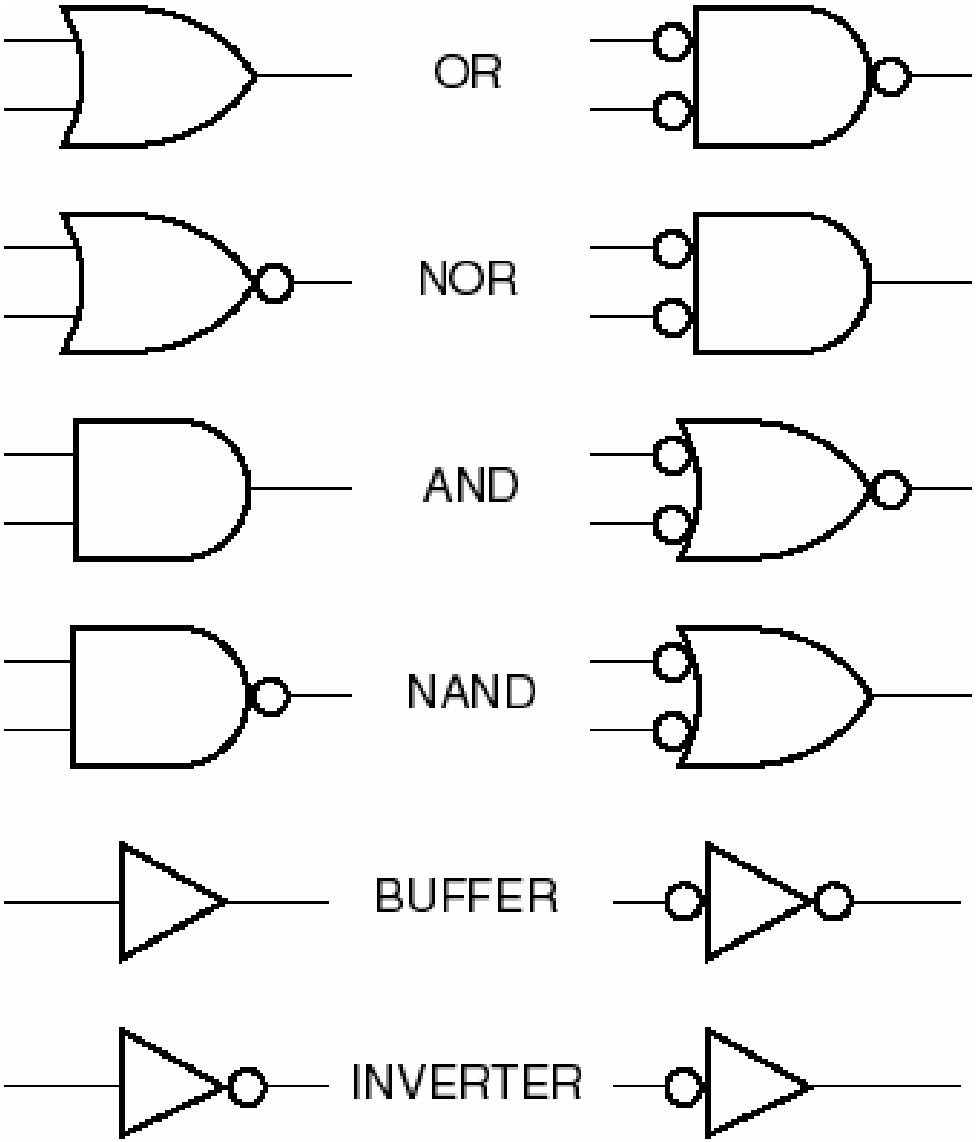








DeMorgan Symbols



Even more definitions (Sec. 4.1.6)

Product term

Sum-of-products expression

Sum term

Product-of-sums expression

Normal term

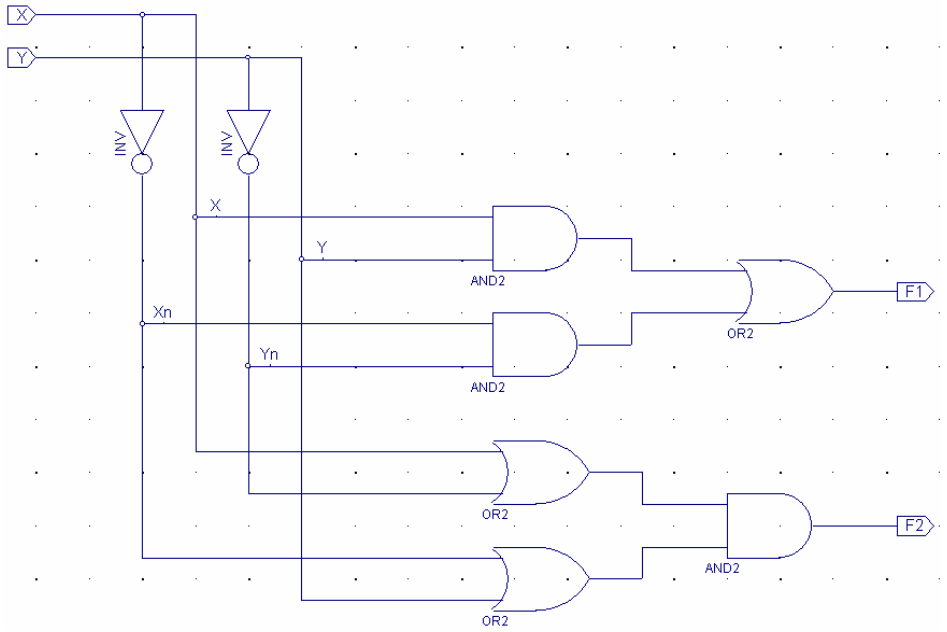
Minterm (n variables)

Maxterm (n variables)

Example Using 5 forms of a Logic Expression

1) **The Truth Table**:

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



2) **Sum-of-Products**: algebraic sum of minterms. $F = X' \cdot Y' + X \cdot Y$

3) **Minterm List**: using $\sum$ notation.
$F = \sum_{XY}(0,3)$

4) **Product-of-Sums**: algebraic product of maxterms. $F = (X + Y') \cdot (X' + Y)$

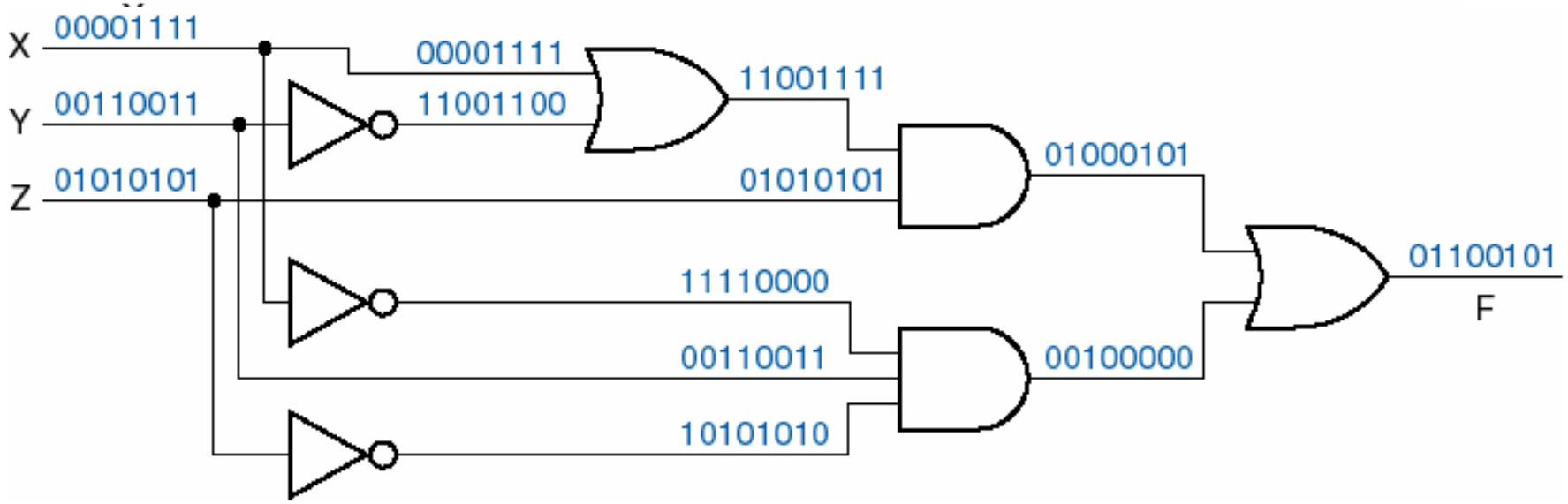
5) **Maxterm List**: using $\prod$ notation.
$F = \prod_{XY}(1,2)$

- Circuit is an Exclusive-NOR gate. A.K.A. an equivalence gate.
- F1 = F2

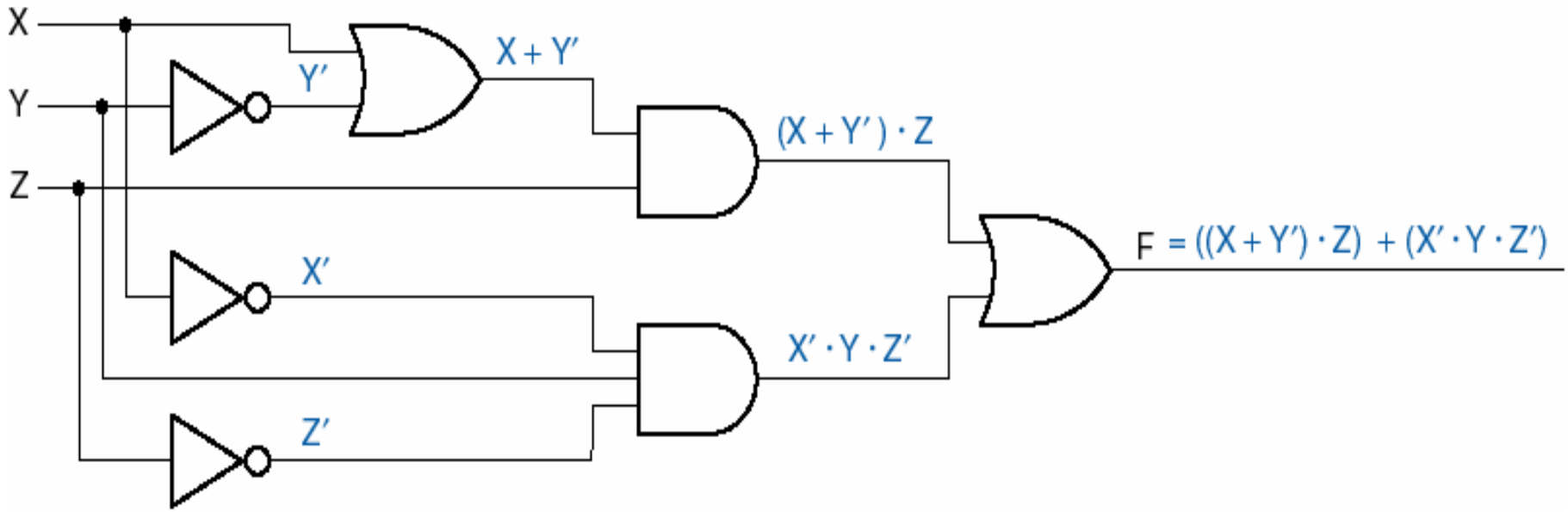
Truth table vs. minterms & maxterms

Row	X	Y	Z	F	Minterm	Maxterm
0	0	0	0	F(0,0,0)	$X' \cdot Y' \cdot Z'$	$X + Y + Z$
1	0	0	1	F(0,0,1)	$X' \cdot Y' \cdot Z$	$X + Y + Z'$
2	0	1	0	F(0,1,0)	$X' \cdot Y \cdot Z'$	$X + Y' + Z$
3	0	1	1	F(0,1,1)	$X' \cdot Y \cdot Z$	$X + Y' + Z'$
4	1	0	0	F(1,0,0)	$X \cdot Y' \cdot Z'$	$X' + Y + Z$
5	1	0	1	F(1,0,1)	$X \cdot Y' \cdot Z$	$X' + Y + Z'$
6	1	1	0	F(1,1,0)	$X \cdot Y \cdot Z'$	$X' + Y' + Z$
7	1	1	1	F(1,1,1)	$X \cdot Y \cdot Z$	$X' + Y' + Z'$

Combinational analysis

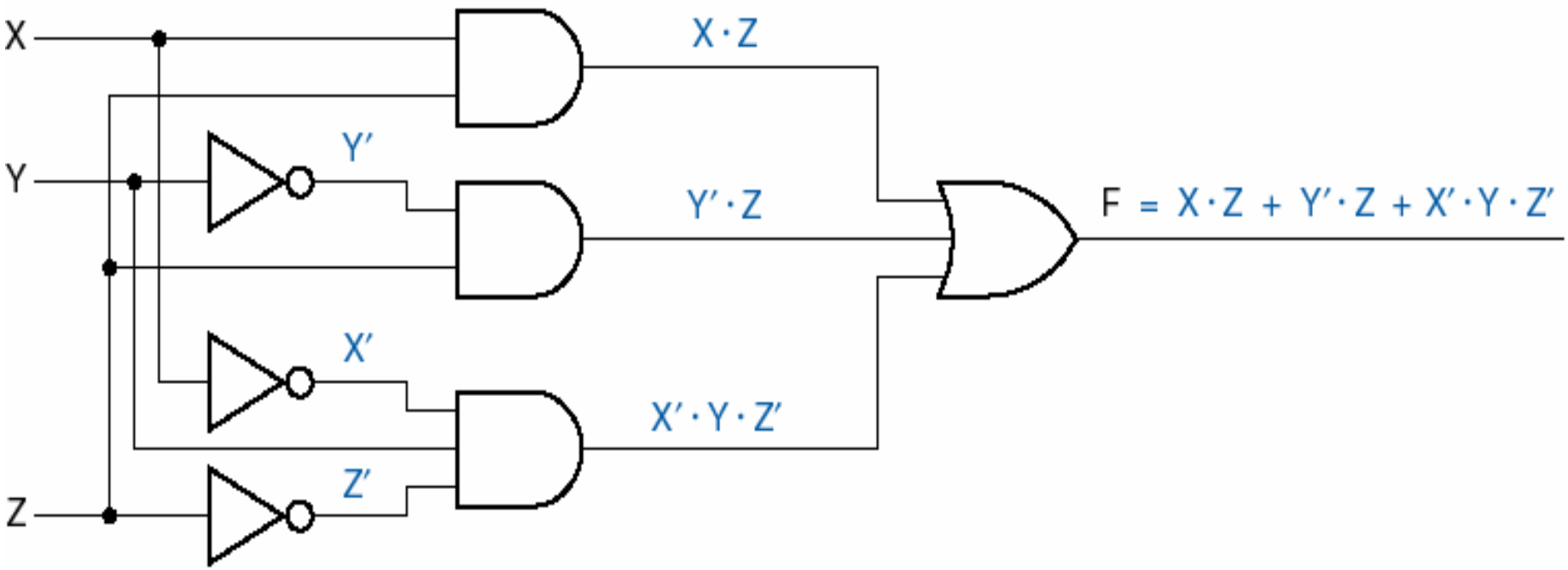


Signal expressions



$$F = ((X + Y') \cdot Z) + (X' \cdot Y \cdot Z')$$
$$= (X \cdot Z) + (Y' \cdot Z) + (X' \cdot Y \cdot Z')$$

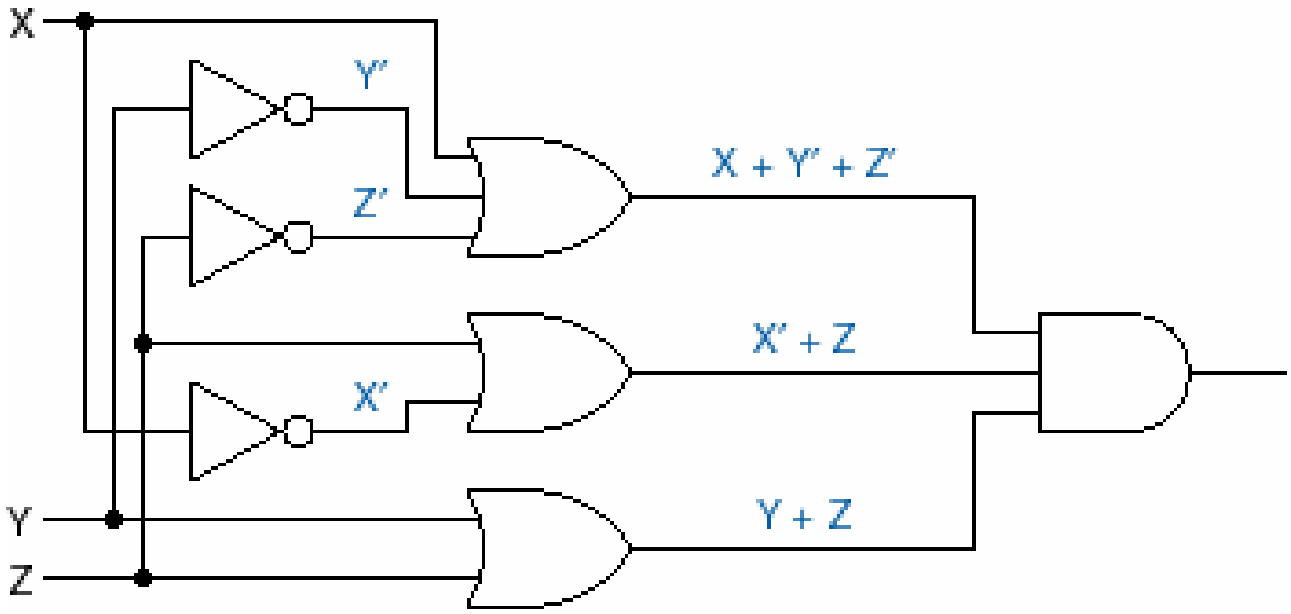
New circuit, same function



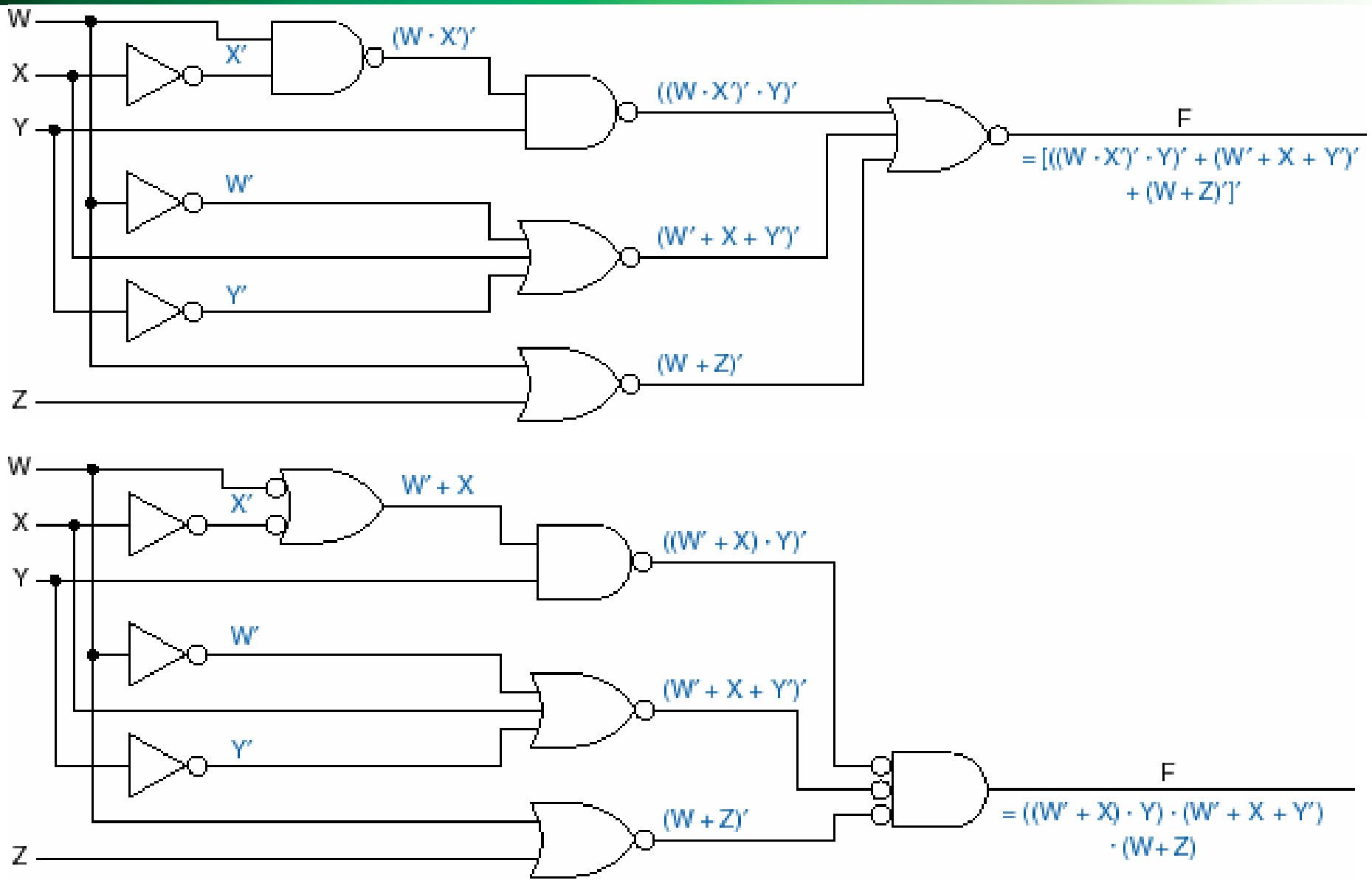
"Add out" logic function

$$
\begin{aligned}
F &= ((X + Y') \cdot Z) + (X' \cdot Y \cdot Z') \\
&= (X + Y' + X') \cdot (X + Y' + Y) \cdot (X + Y' + Z') \cdot (Z + X') \cdot (Z + Y) \cdot (Z + Z') \\
&= 1 \cdot 1 \cdot (X + Y' + Z') \cdot (X' + Z) \cdot (Y + Z) \cdot 1 \\
&= (X + Y' + Z') \cdot (X' + Z) \cdot (Y + Z)
\end{aligned}
$$

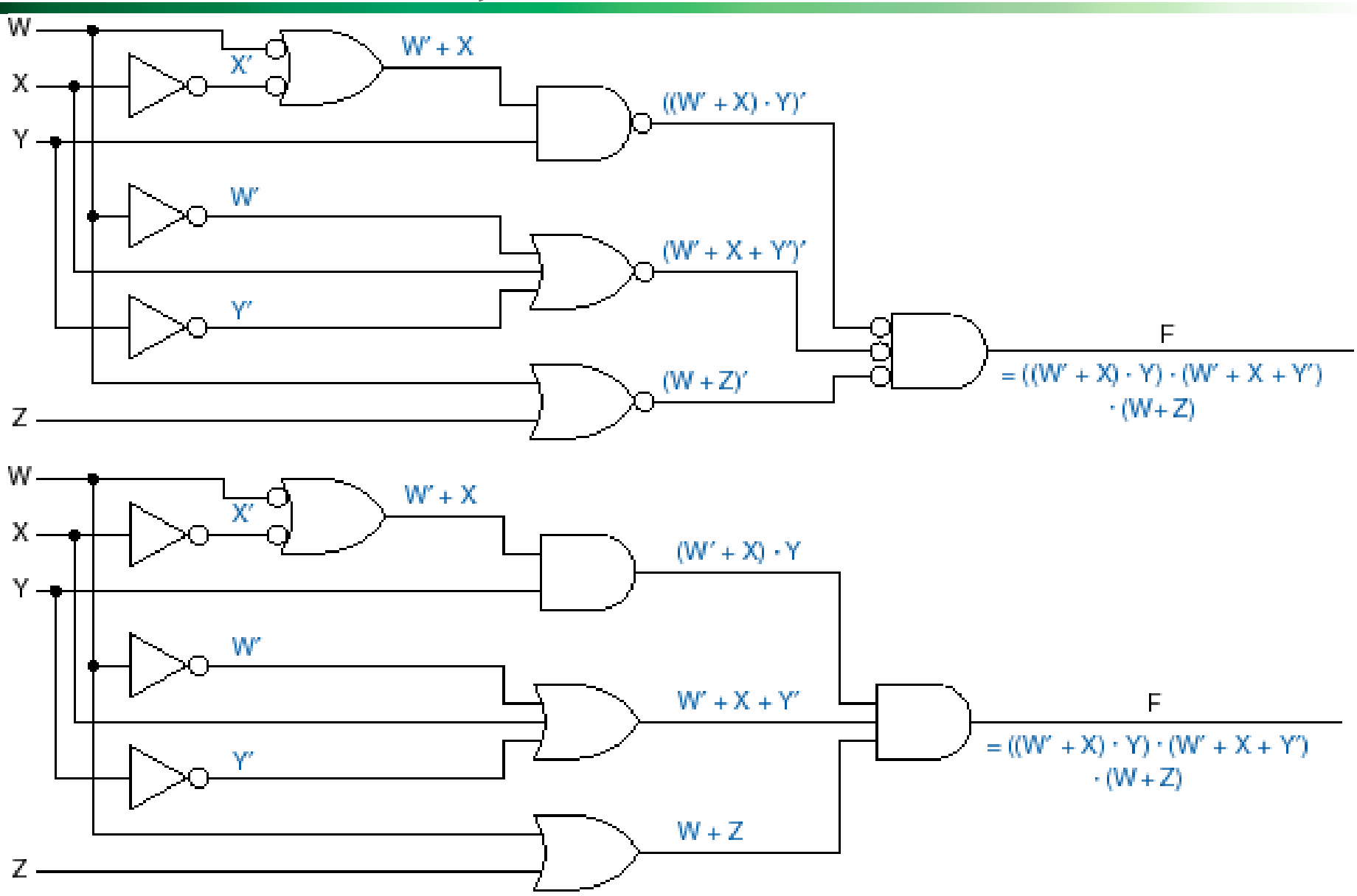
Circuit:



Shortcut: Symbol substitution

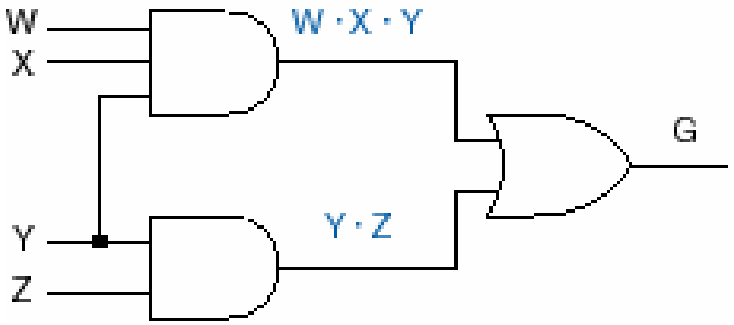


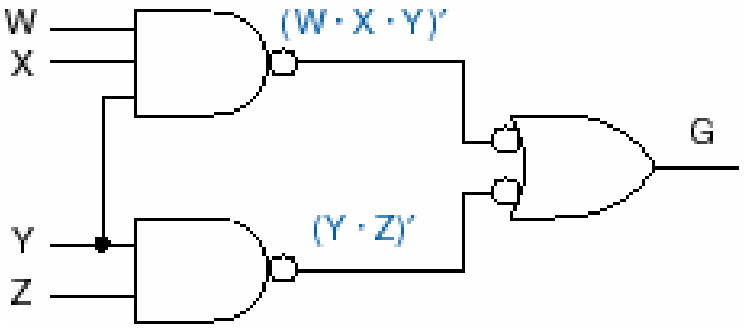
Different circuit, same function

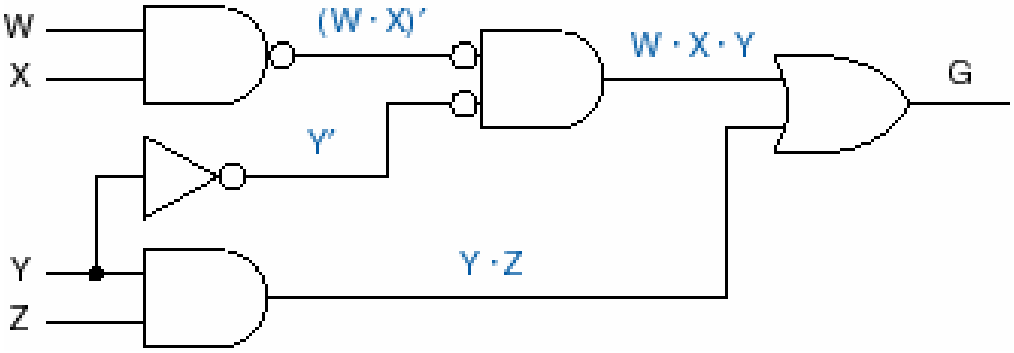


Another example

$$G(W, X, Y, Z) = W \cdot X \cdot Y + Y \cdot Z$$







Combinational-Circuit Analysis

Combinational circuits -- outputs depend only on current inputs (not on history).

Kinds of combinational analysis:

- exhaustive (truth table)
- algebraic (expressions)
- simulation / test bench
  - Write functional description in HDL
  - Define test conditions / test vecors
  - Compare circuit output with functional description (or known-good realization)

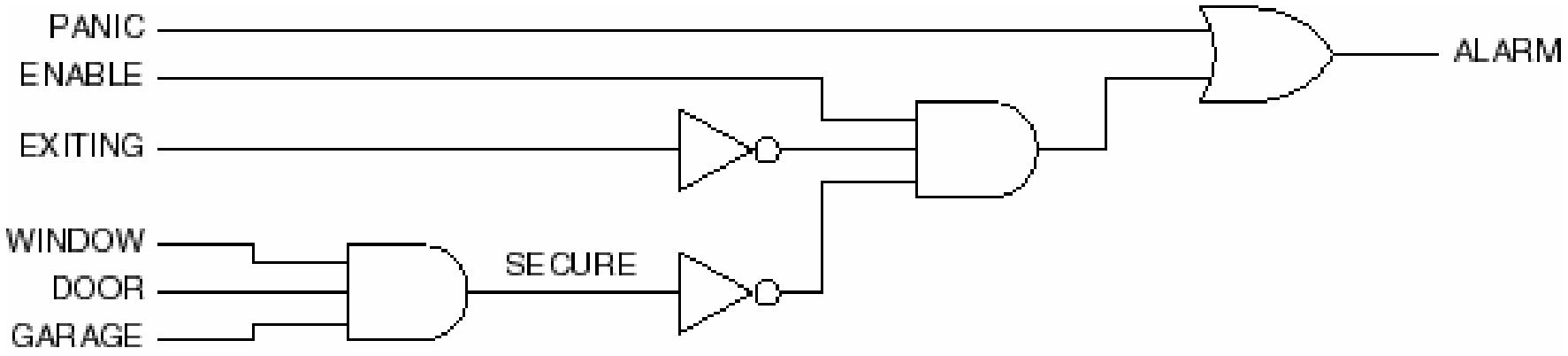
Combinational-Circuit Design

Sometimes you can write an equation or equations directly .

Example (alarm circuit):

$$\begin{aligned} ALARM &= PANIC + ENABLE \cdot EXITING' \cdot SECURE' \\ SECURE &= WINDOW \cdot DOOR \cdot GARAGE \\ ALARM &= PANIC + ENABLE \cdot EXITING' \cdot (WINDOW \cdot DOOR \cdot GARAGE)' \end{aligned}$$

Corresponding circuit:

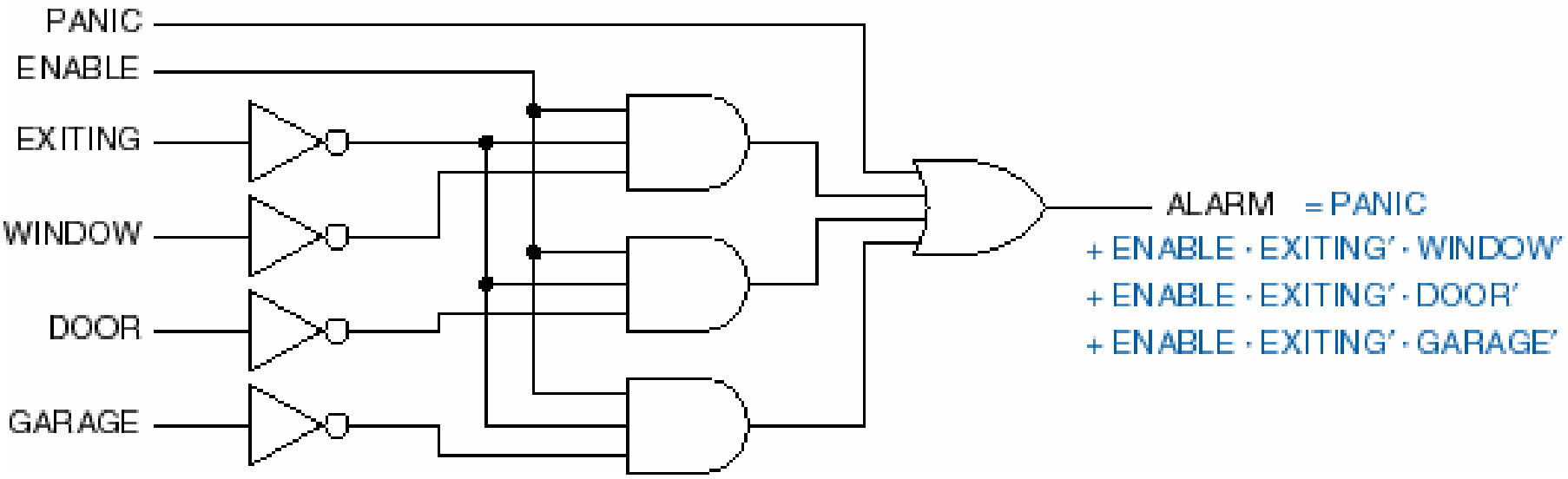


Alarm-circuit transformation

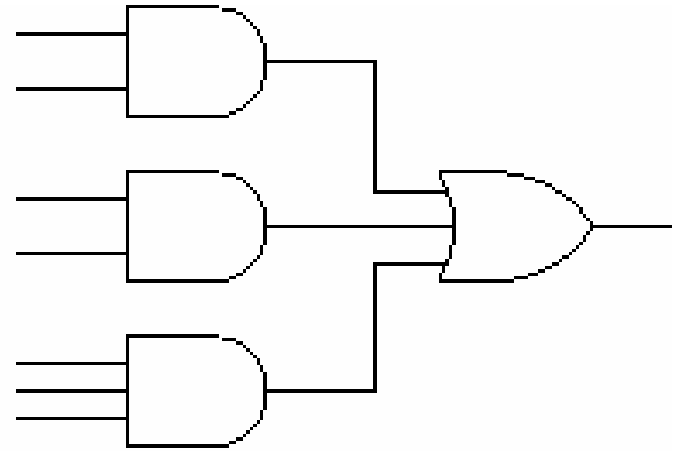
Sum-of-products form

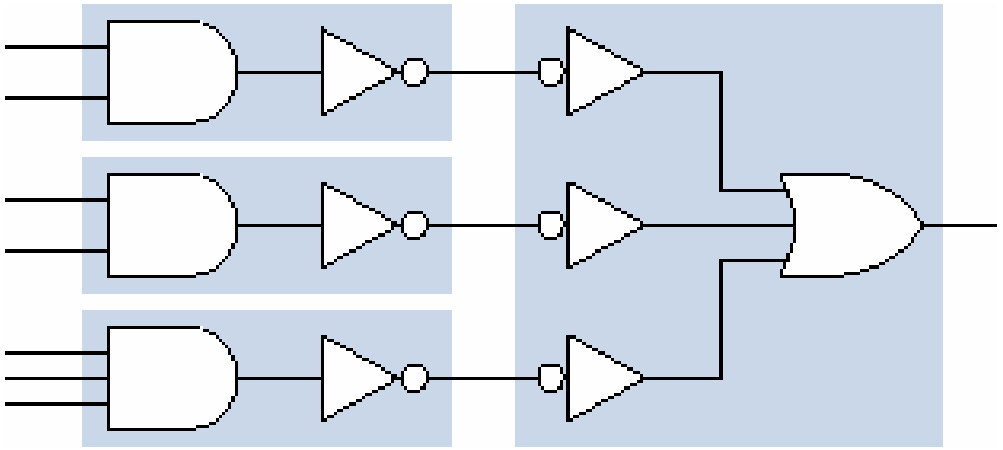
– Useful for programmable logic devices

"Multiply out":



Sum-of-products form

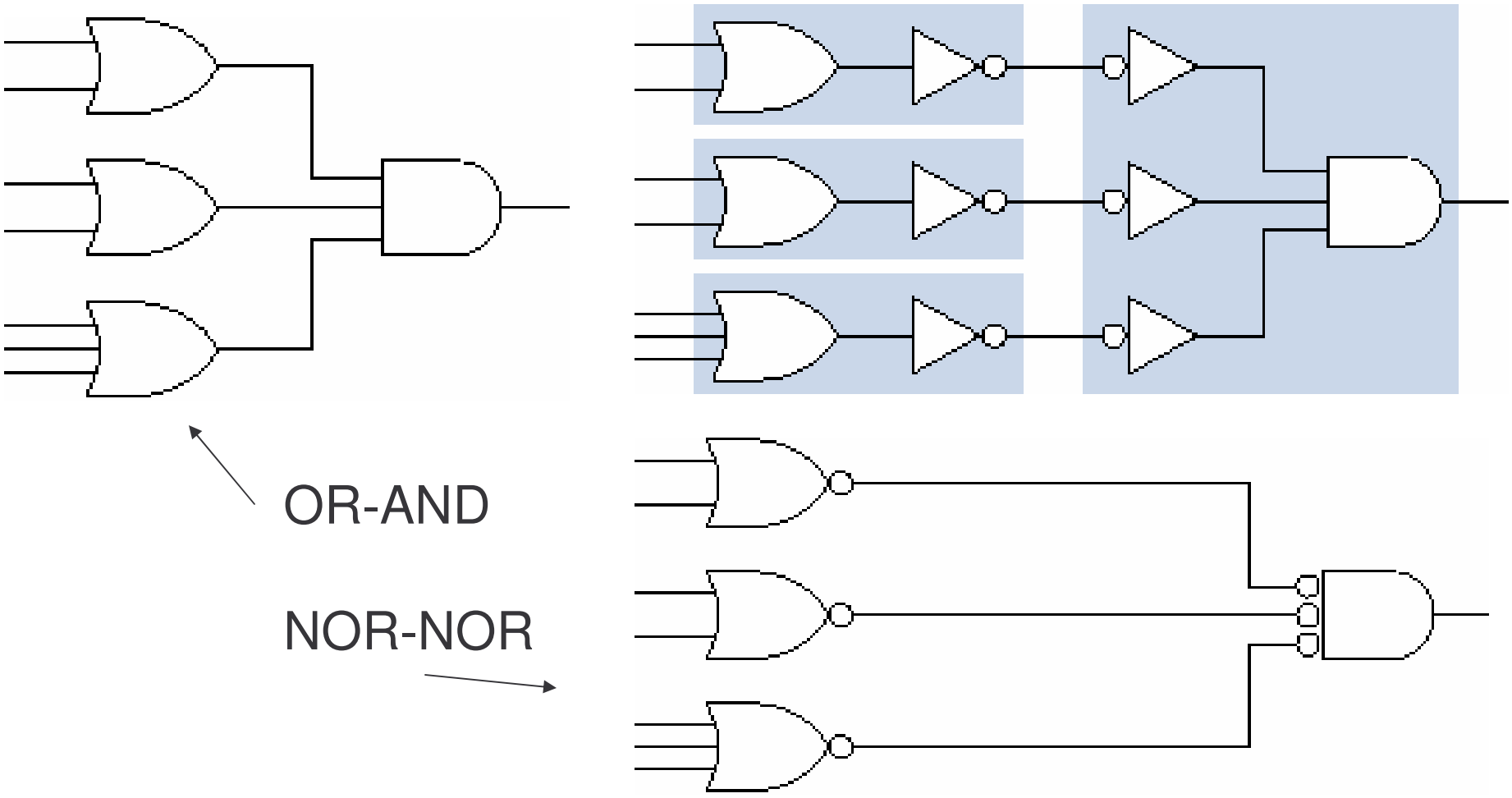




AND-OR

NAND-NAND

Product-of-sums form



Brute-force design

Truth table -->
- canonical sum
  (sum of minterms)

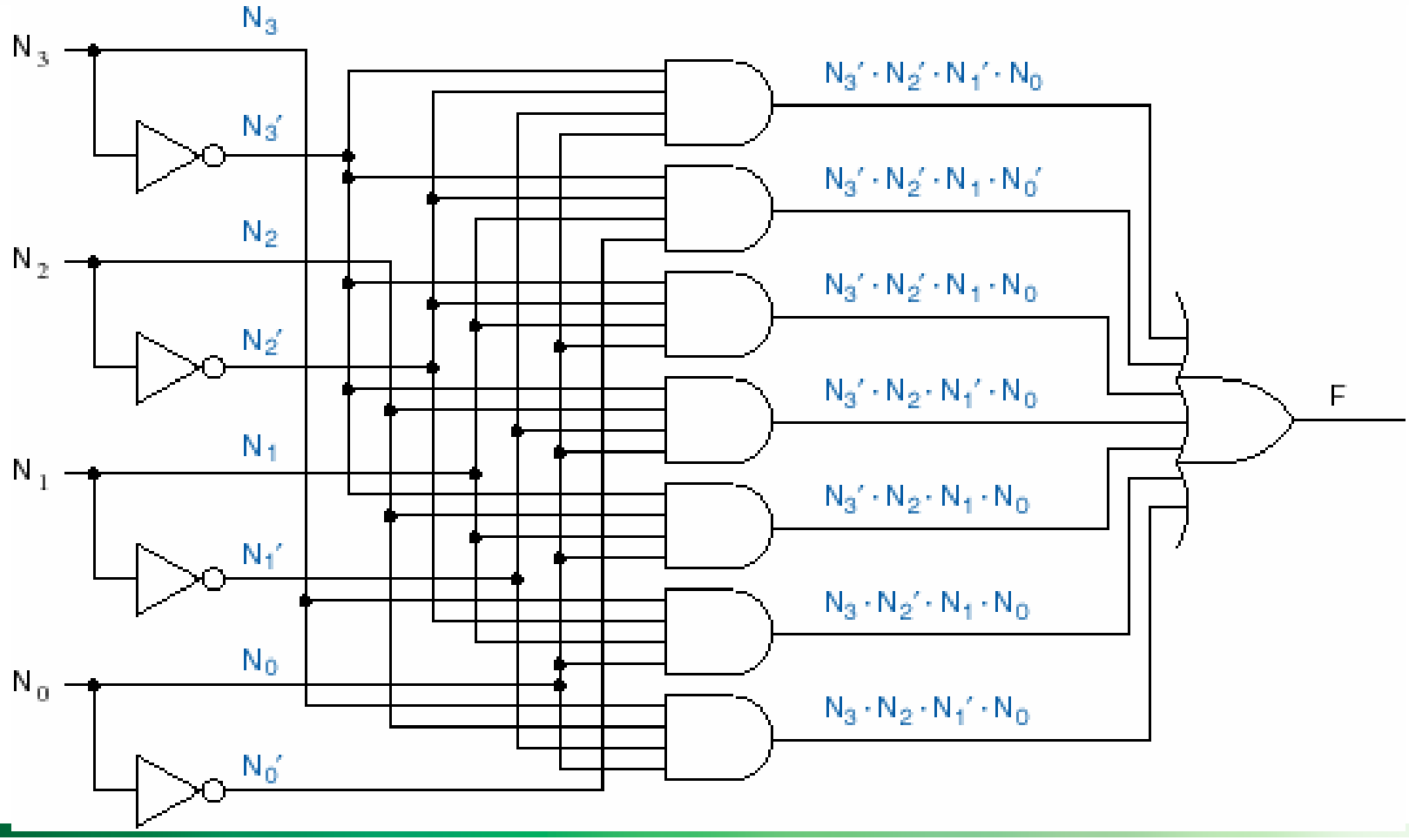
Example:
- prime-number detector
  - 4-bit input, $N_3N_2N_1N_0$

$F = \Sigma_{N3N2N1N0}(1,2,3,5,7,11,13)$

row	N_3	N_2	N_1	N_0	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	0	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	0
15	1	1	1	1	0

Minterm list --> canonical sum

$$F = \Sigma_{N_3,N_2,N_1,N_0}(1, 2, 3, 5, 7, 11, 13)$$
$$= N_3' \cdot N_2' \cdot N_1' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0' + N_3' \cdot N_2' \cdot N_1 \cdot N_0 + N_3' \cdot N_2 \cdot N_1' \cdot N_0$$
$$+ N_3' \cdot N_2 \cdot N_1 \cdot N_0 + N_3 \cdot N_2' \cdot N_1 \cdot N_0 + N_3 \cdot N_2 \cdot N_1' \cdot N_0$$



Algebraic simplification

Theorem T8,

$$X \cdot Y + X \cdot Y' = X$$

$$F = \Sigma_{N_3,N_2,N_1,N_0}(1, 3, 5, 7, 2, 11, 13)$$

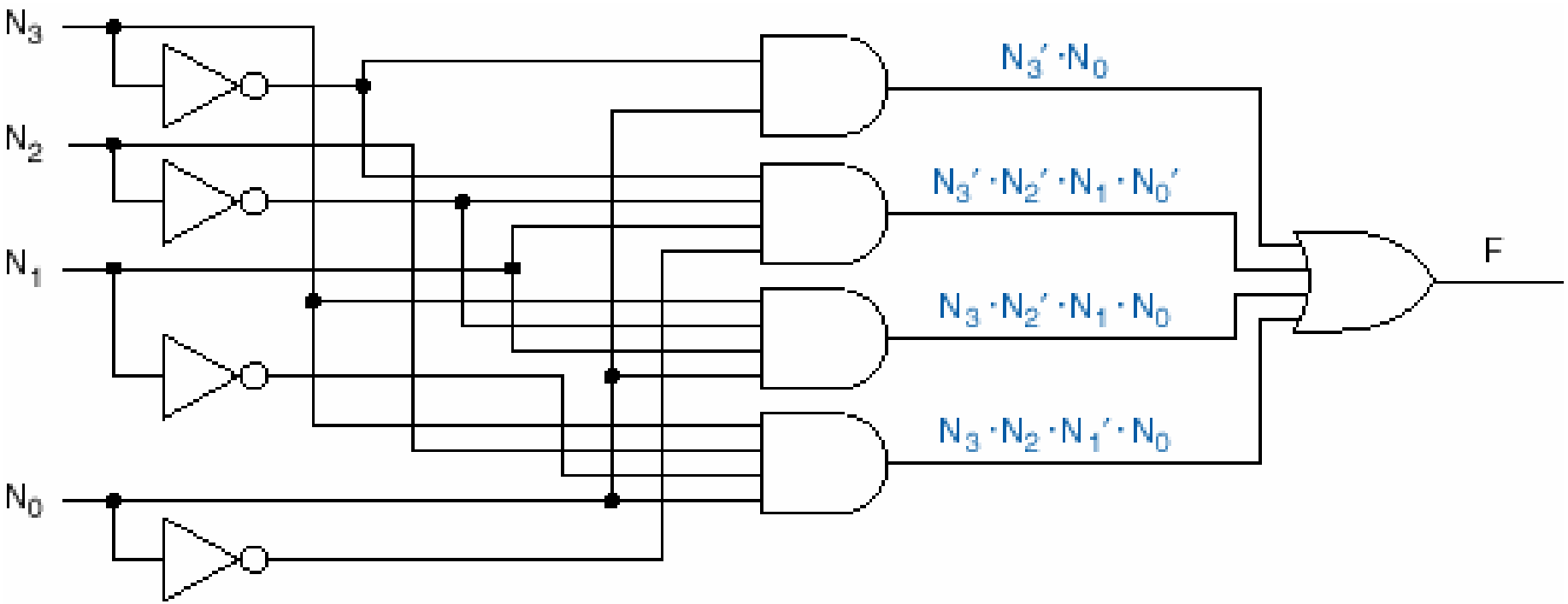
$$= N_3' \cdot N_2' N_1' N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0 + N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0 + \ldots$$

$$= (N_3' \cdot N_2' \cdot N_1' \cdot N_0 + N_3' \cdot N_2' \cdot N_1 \cdot N_0) + (\cdot N_3' \cdot N_2 \cdot N_1' \cdot N_0 + N_3' \cdot N_2 \cdot N_1 \cdot N_0) + \ldots$$

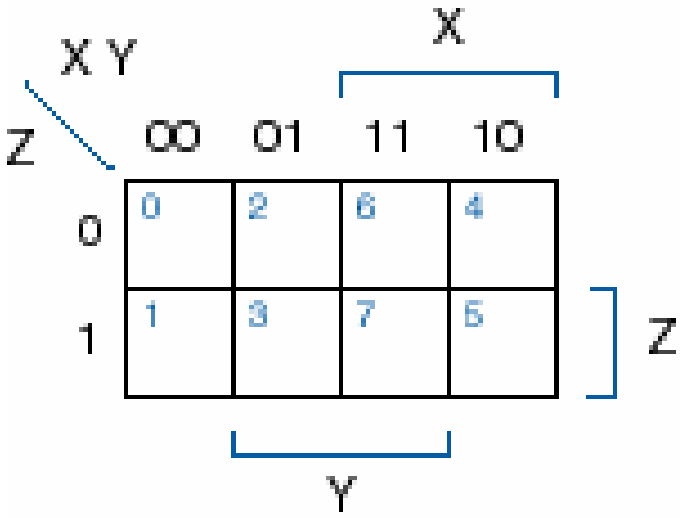
$$= N_3' N_2' \cdot N_0 + N_3' \cdot N_2 \cdot N_0 + \ldots$$

Reduce number of gates and gate inputs

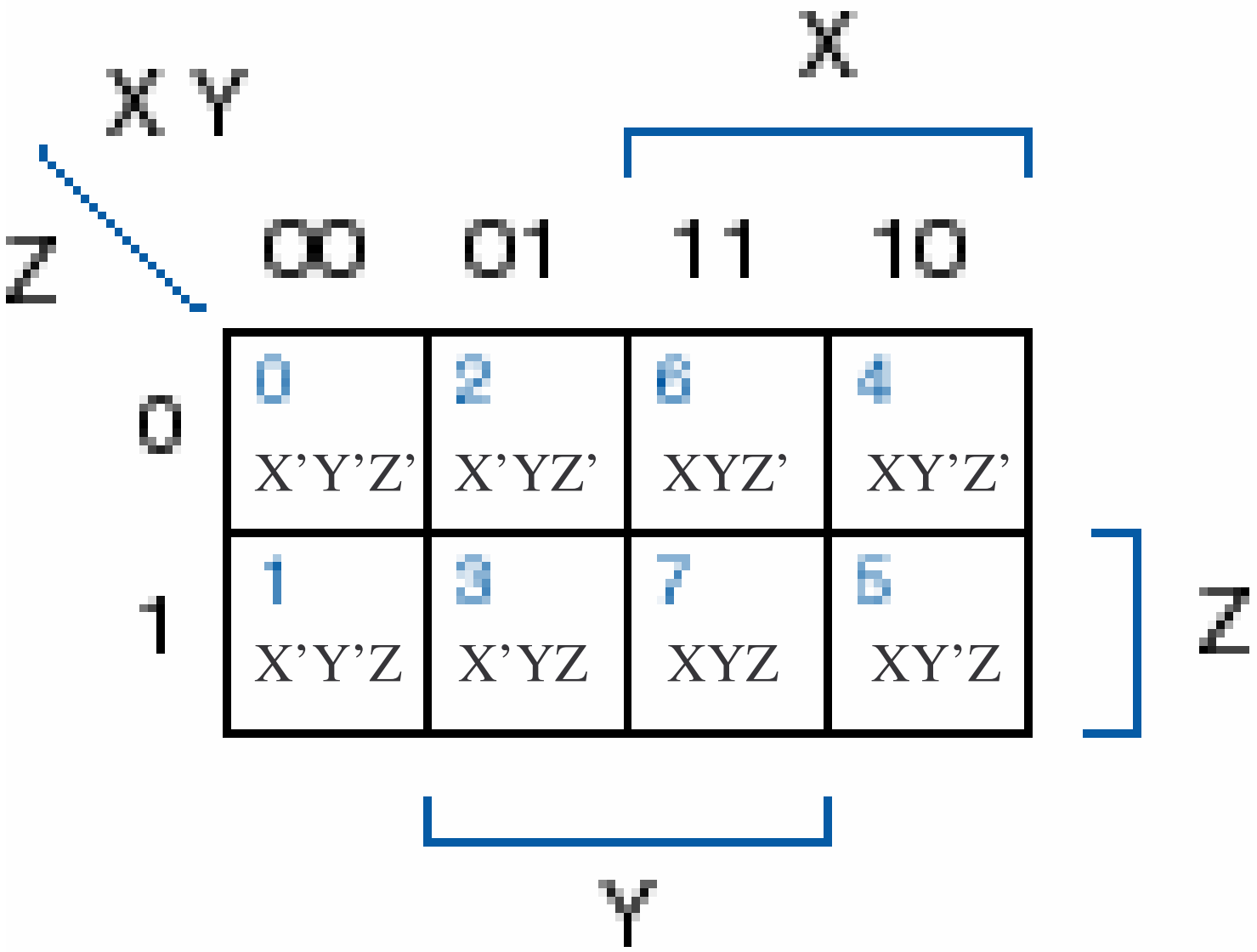
Resulting circuit



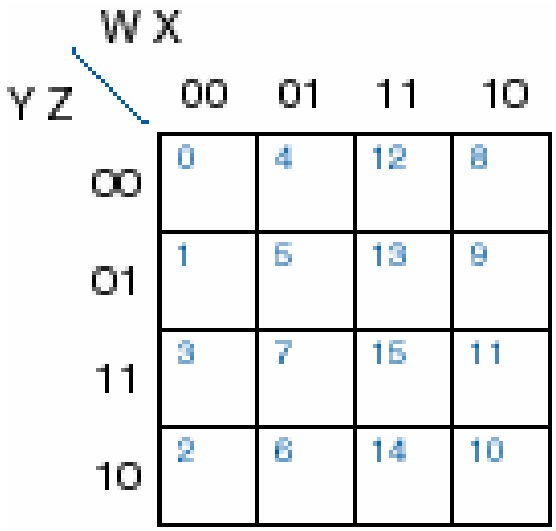
3-variable Karnaugh map

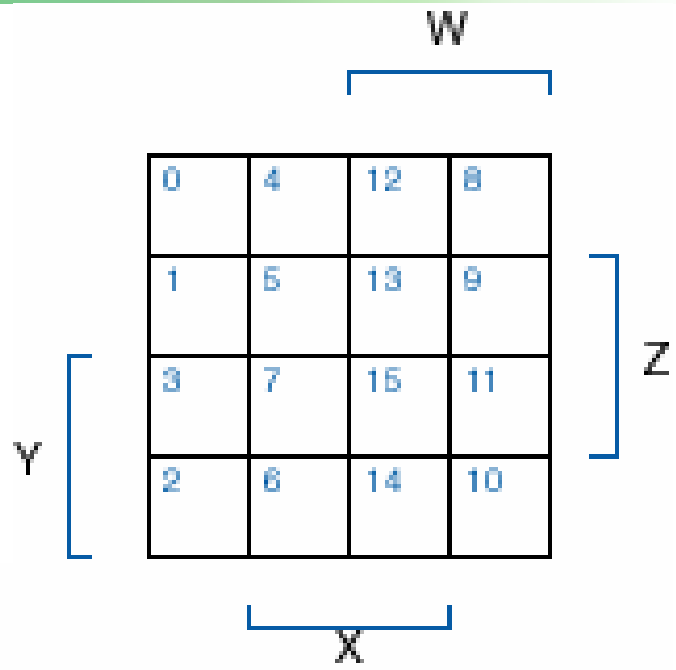


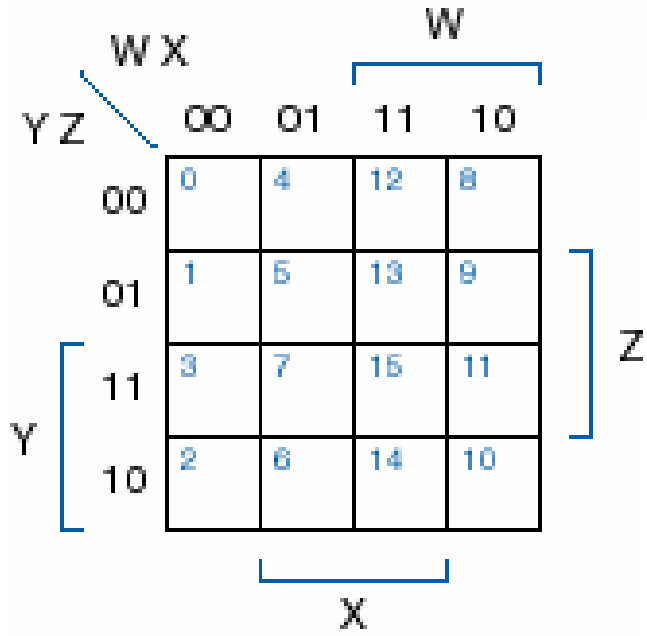
3-variable Karnaugh map



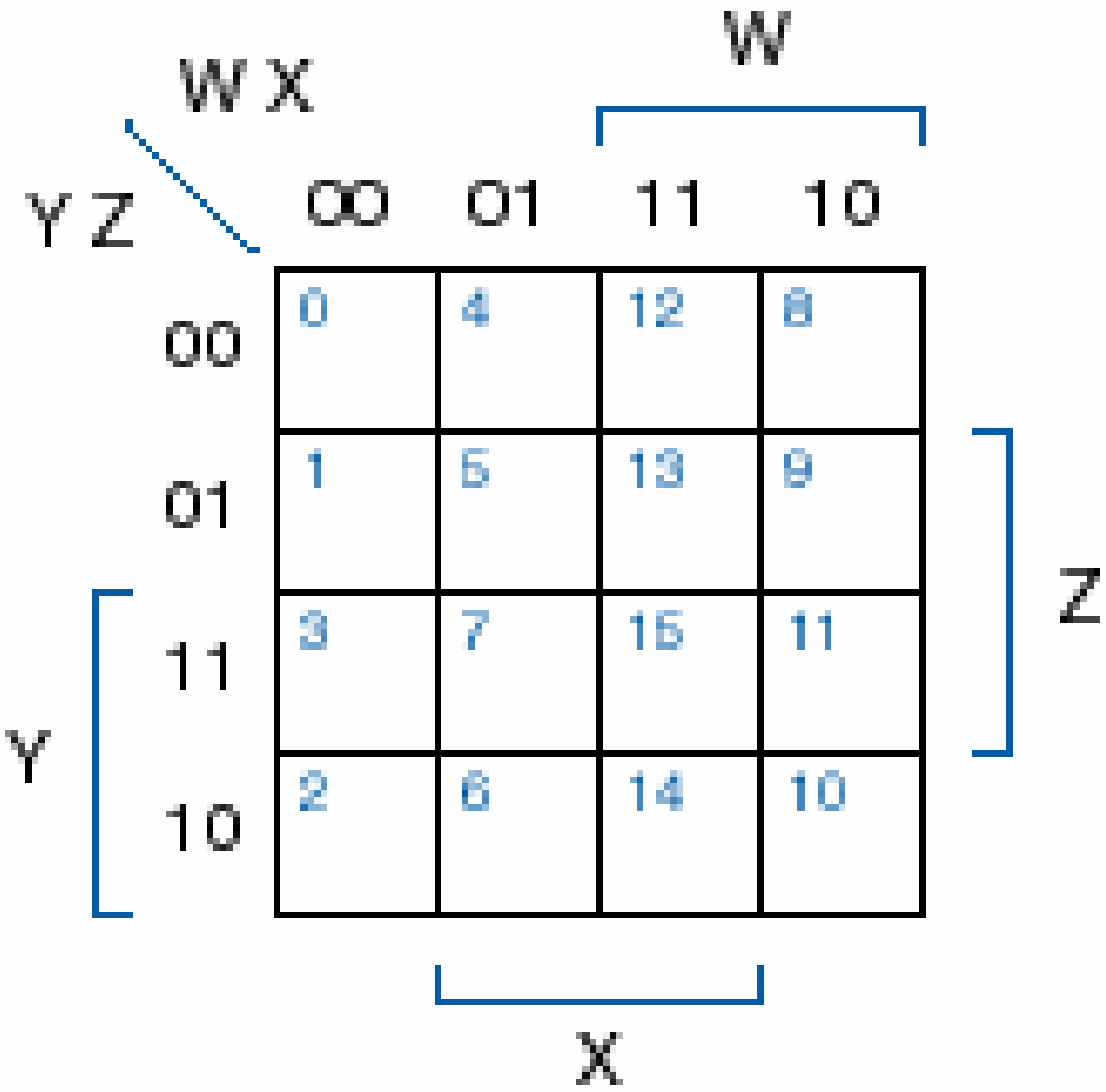
Visualizing T10 -- Karnaugh maps





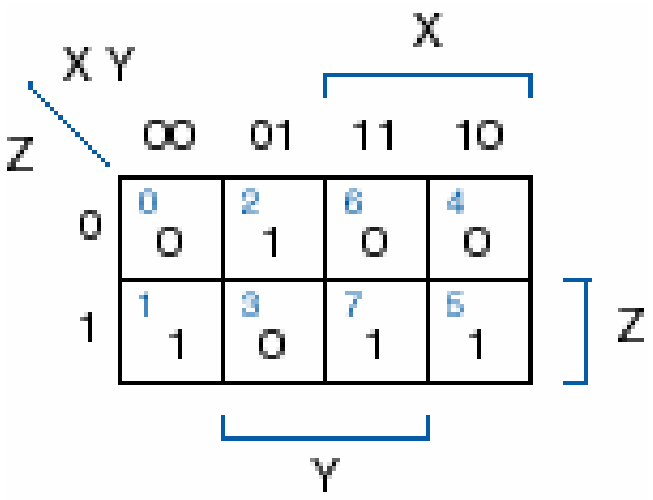


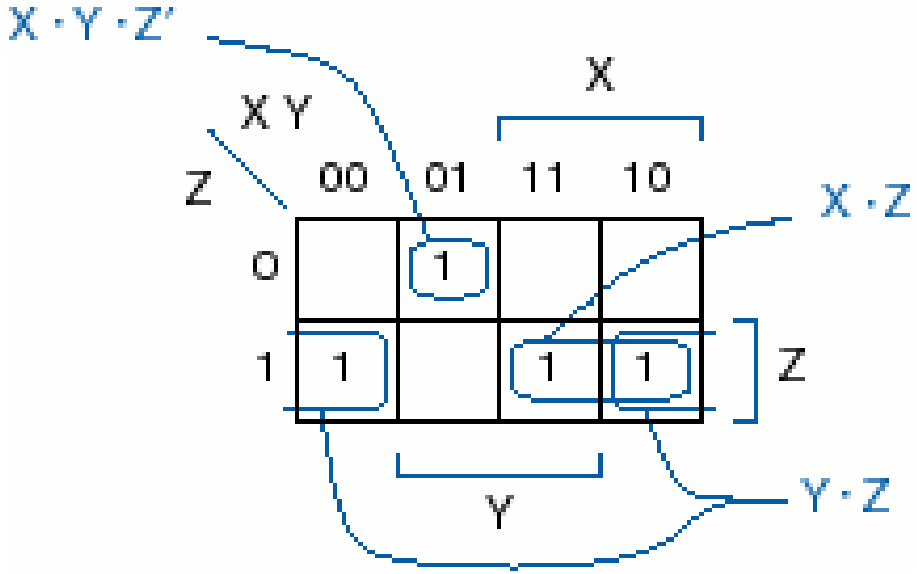
Visualizing T10 -- Karnaugh maps



Example: F = Σ(1,2,5,7)

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





Karnaugh-map usage

Plot 1s corresponding to minterms of function.

Circle largest possible rectangular sets of 1s.

- # of 1s in set must be power of 2
- OK to cross edges

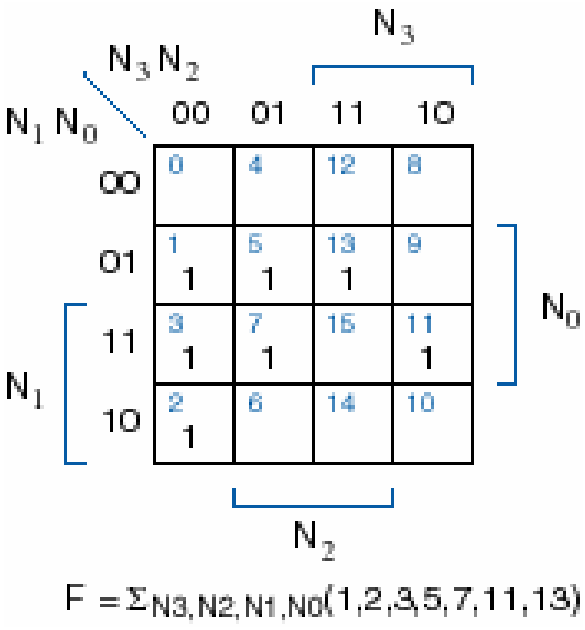
Read off product terms, one per circled set.

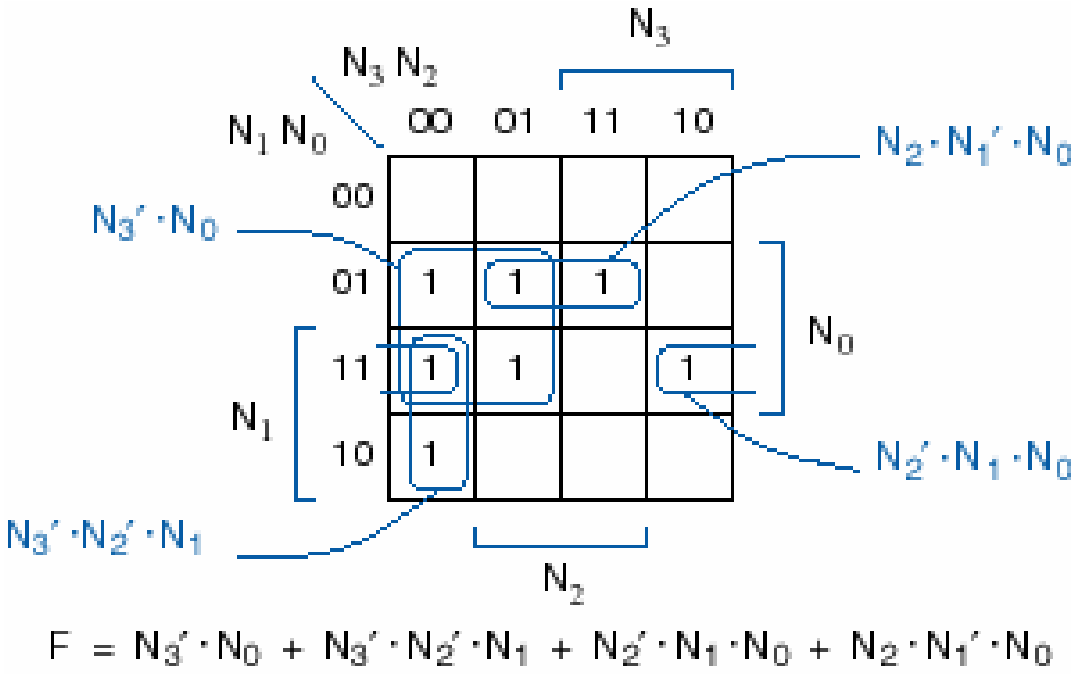
- Variable is 1 ==> include variable
- Variable is 0 ==> include complement of variable
- Variable is both 0 and 1 ==> variable not included

Circled sets and corresponding product terms are called "prime implicants"

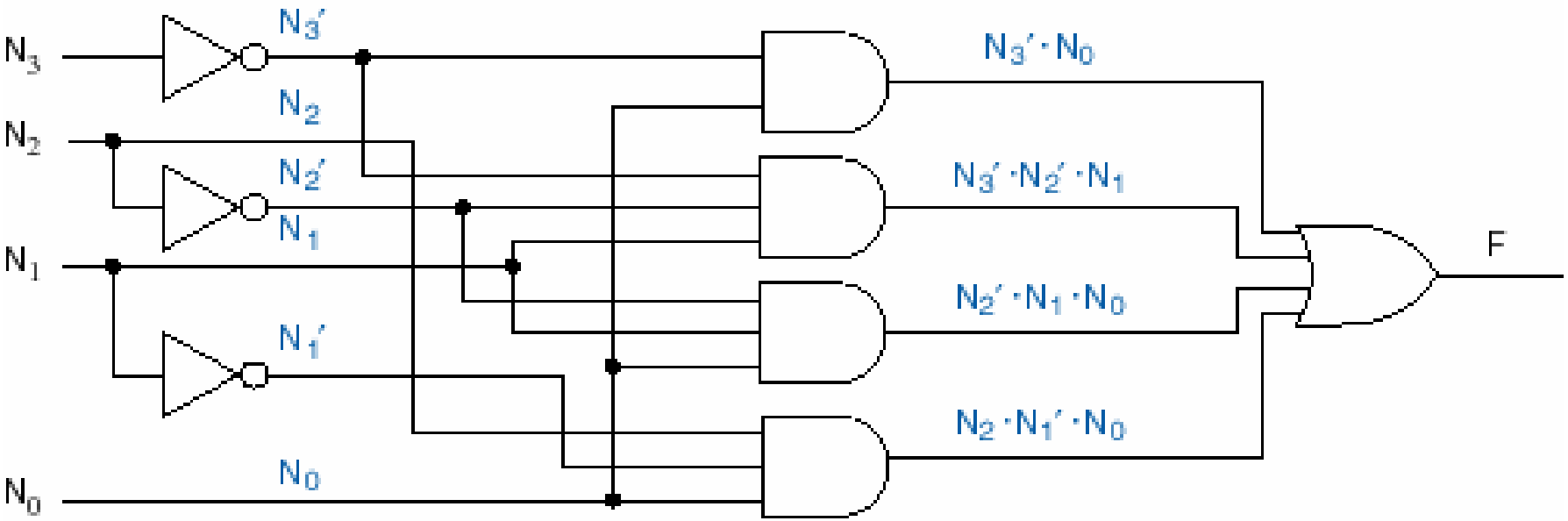
Minimum number of gates and gate inputs

Prime-number detector

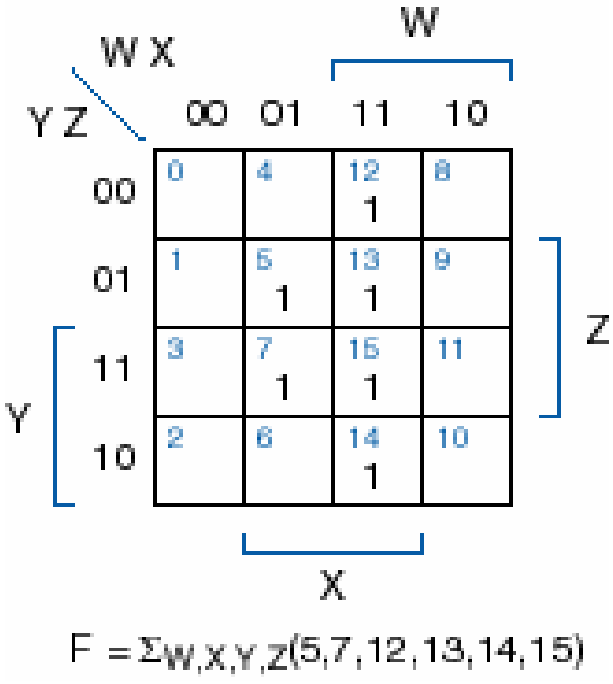


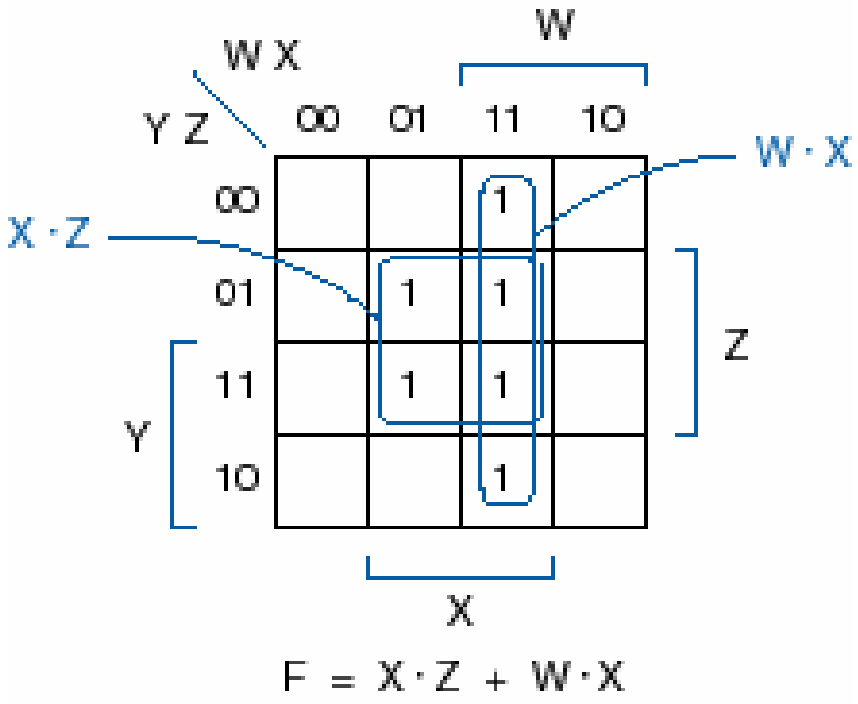


Resulting Circuit.

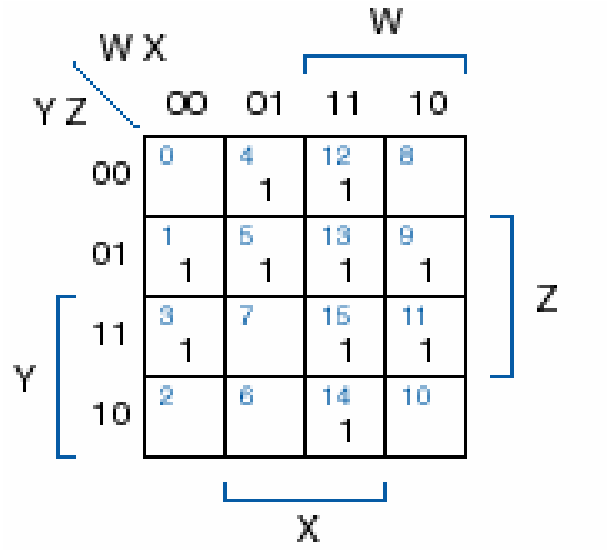


Another example

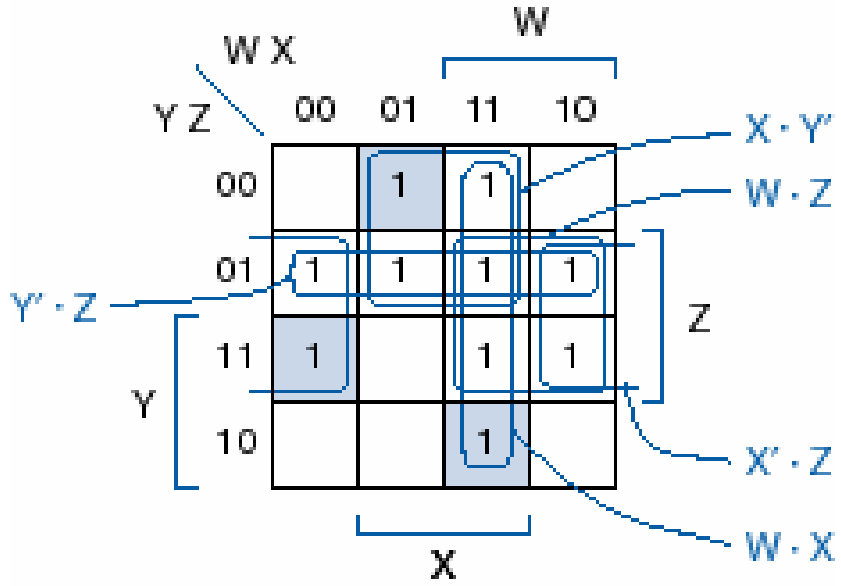




Yet another example



$F = \Sigma_{W,X,Y,Z}(1,3,4,5,9,11,12,13,14,15)$



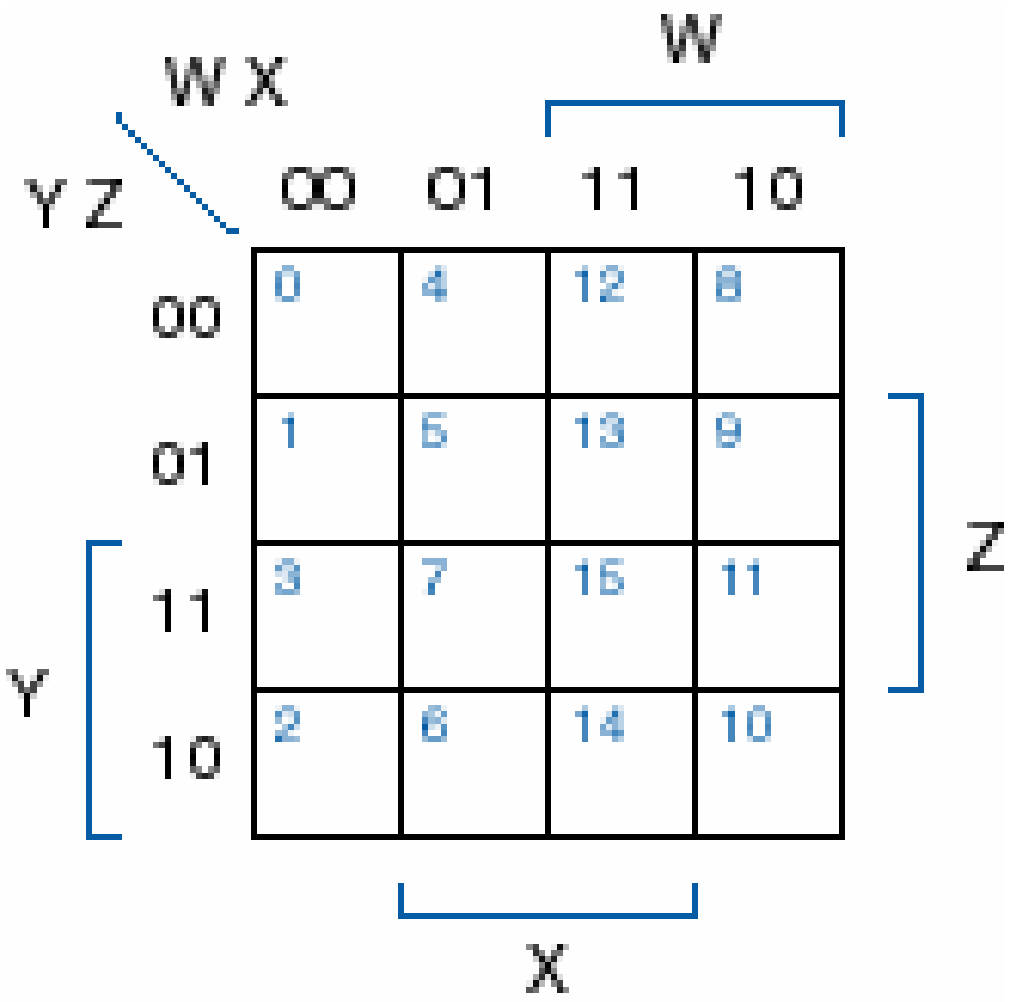
$F = X \cdot Y' + X' \cdot Z + W \cdot X$

Distinguished 1 cells

Essential prime implicants

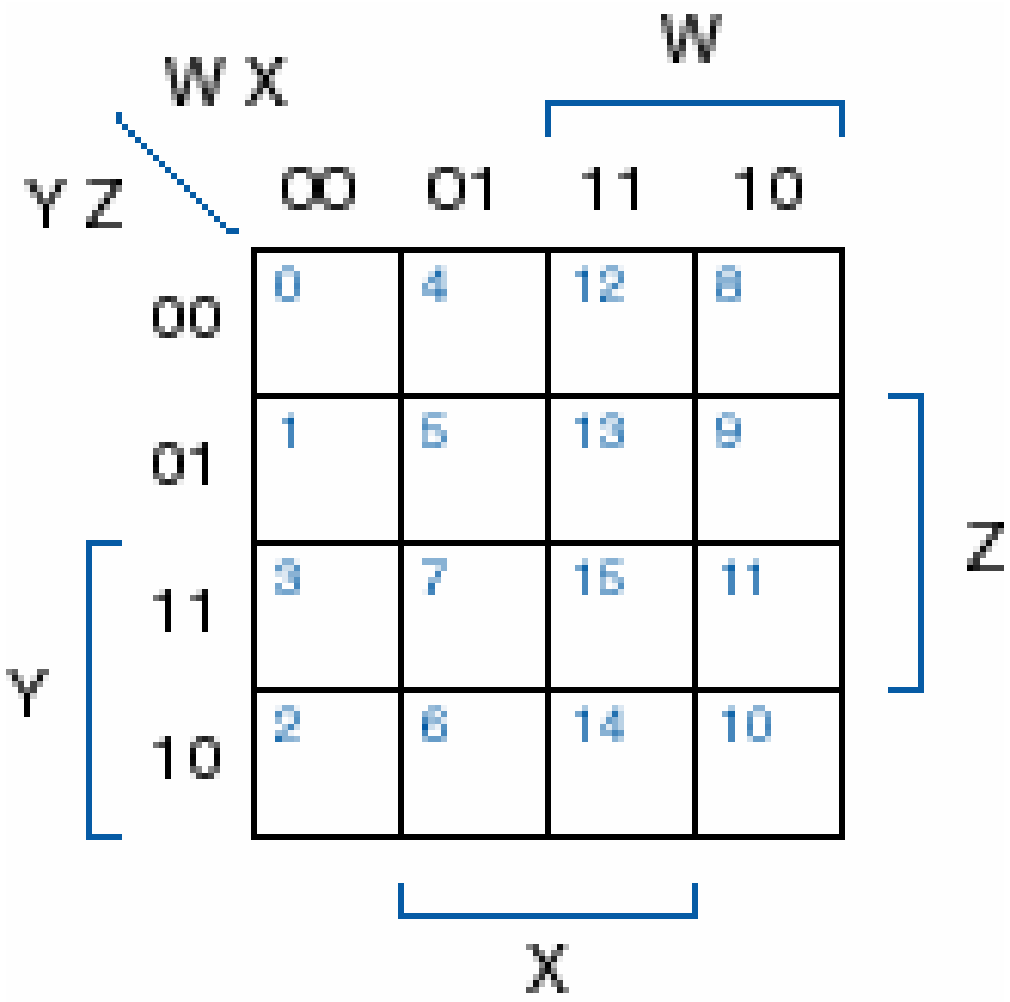
Another Example

$F(W,X,Y,Z) = \Sigma m(0,1,2,4,5,6,8,9,12,13,14)$

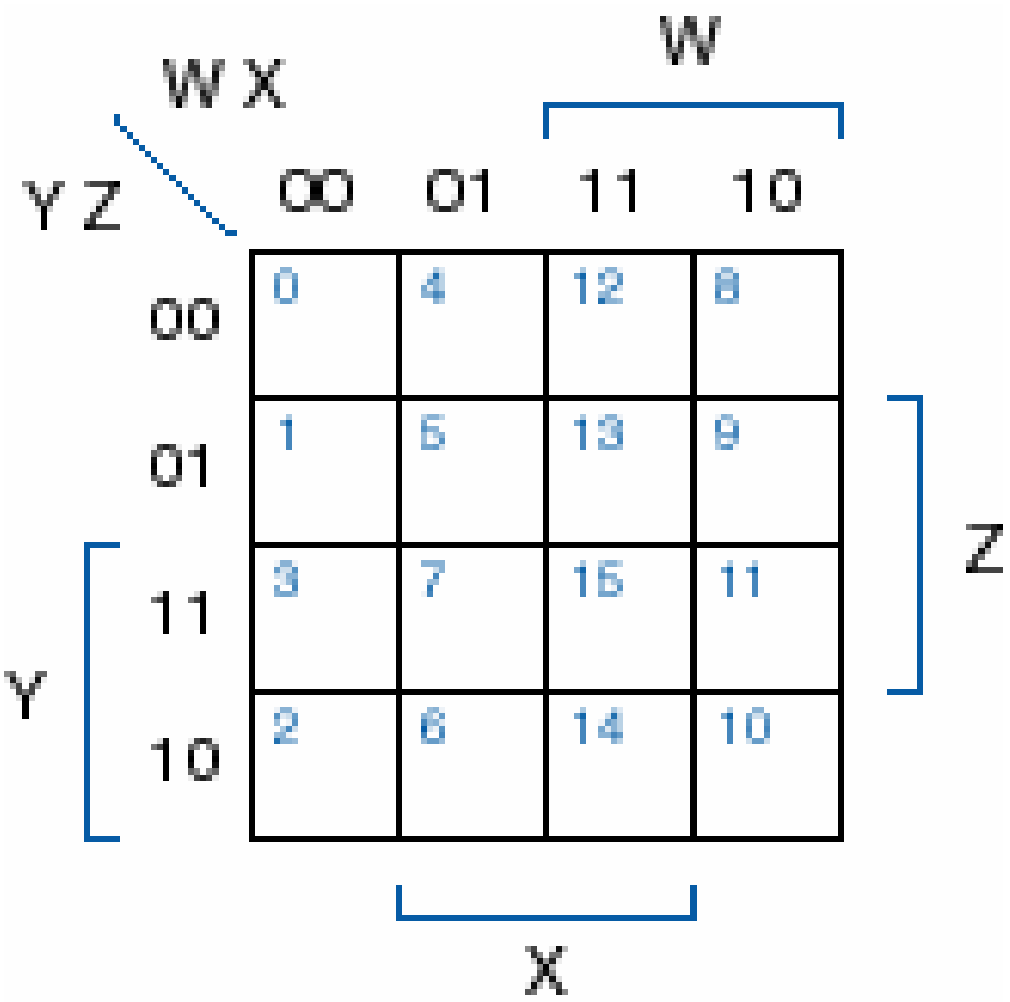


Another Example

$F(W,X,Y,Z) = \Sigma m(0,1,2,3,6,8,9,10,11,14)$



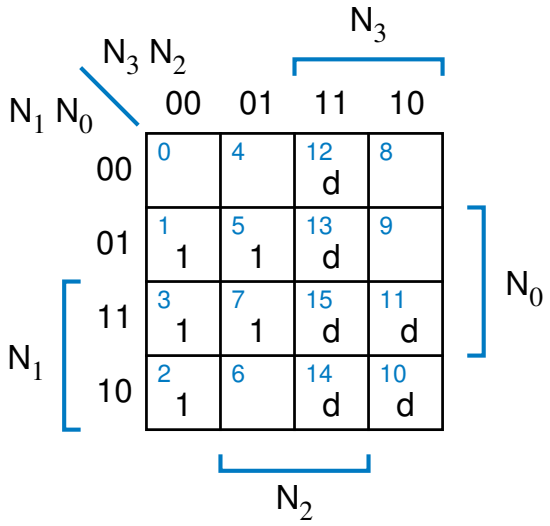
Another Example



Don't Cares

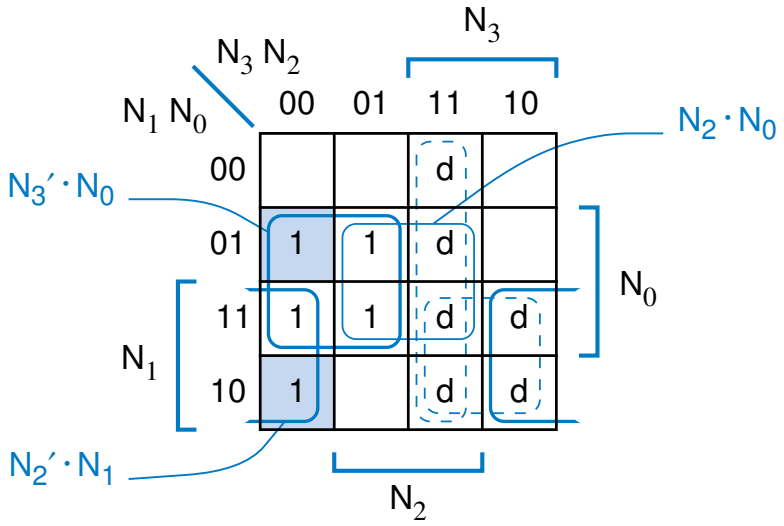
Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e

(a)



$F = \Sigma_{N3,N2,N1,N0}(1,2,3,5,7) + d(10,11,12,13,14,15)$

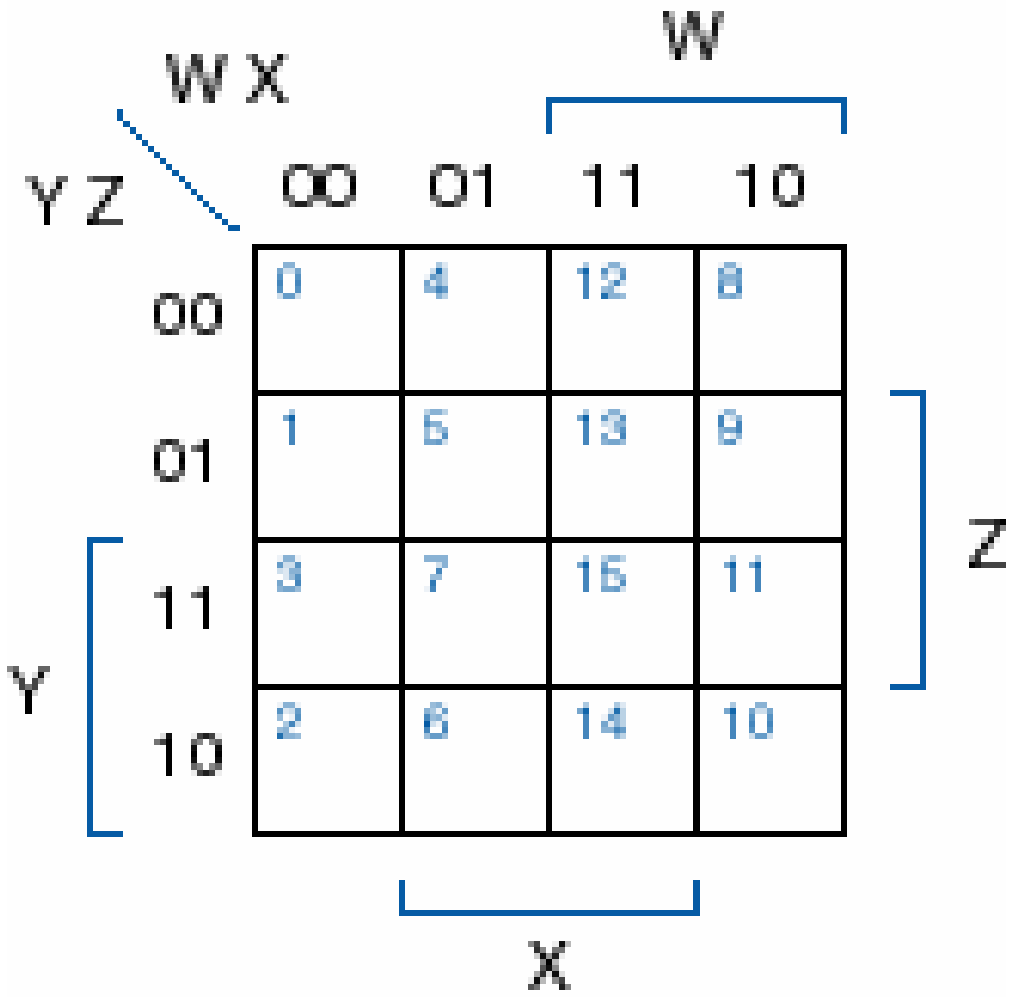
(b)



$F = N_3' \cdot N_0 + N_2' \cdot N_1$

Another Example

$F(W,X,Y,Z) = \Sigma m(0,1,2,3,6,8,9,10,11,14) + d(7,15)$



Current Logic Design

Lots more than 6 inputs -- can't use Karnaugh maps

Use software to synthesize logic expressions and minimize logic

Hardware Description Languages -- VHDL and Verilog