Gate-Level Minimization

Jee-Hwan Ryu

School of Mechanical Engineering
Korea University of Technology and Education

Gate-Level Minimization-The Map Method

- Truth table is unique
- Many different algebraic expression
- Boolean expressions may be simplified by algebraic means
- But, awkward due to the lack of specific rules
- Karnaugh Map or K-map method
  - Pictorial form of truth table
  - A simple and straight forward procedure

Why Need to be Simple ?

- Produces a circuit diagram with a minimum number of gates and the minimum number of inputs to the gate
- Simplest expression is not unique

Two-Variable Map

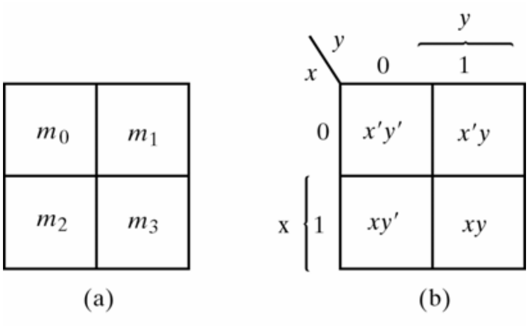


Fig. 3-1 Two-variable Map

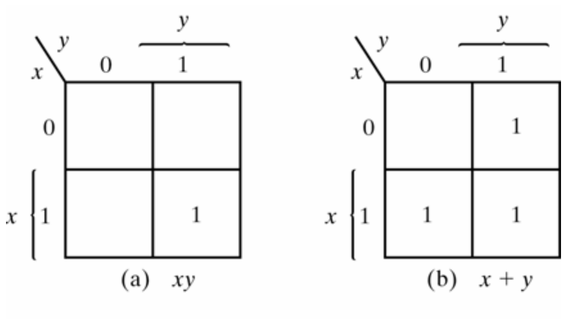


Fig. 3-2 Representation of Functions in the Map

$$m1+m2+m3 = x'y + xy' + xy = x + y$$

Three-Variable Map

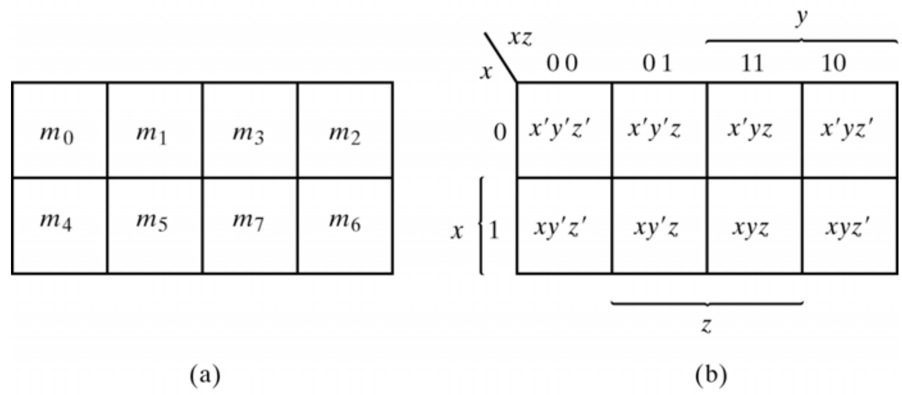


Fig. 3-3 Three-variable Map

Not in a binary sequence, but in a sequence similar to Gray code

$$m_5 + m_7 = xy'z + xyz = xz(y + y') = xz$$

$$m_0 + m_2 = x'y'z' + x'yz' = x'z'(y' + y) = x'z'$$

Examples

Ex 3-1) Simplify the Boolean function, $F(x, y, z) = \Sigma(2, 3, 4, 5)$

$F = x'y + xy'$

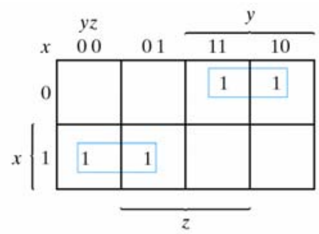
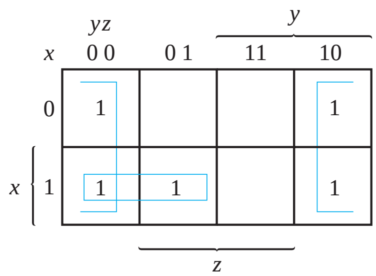


Fig. 3-4 Map for Example 3-1; $F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$

Ex 3-4) Simplify the Boolean Function, $F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$

$F = z' + xy'$



Example

Ex 3-4) Given Boolean function, $F = A'C + A'B + AB'C + BC$

a) express it in sum of minterms

$F(x, y, z) = \Sigma(1, 2, 3, 5, 7)$

b) find the minimal sum of products

$F = C + A'B$

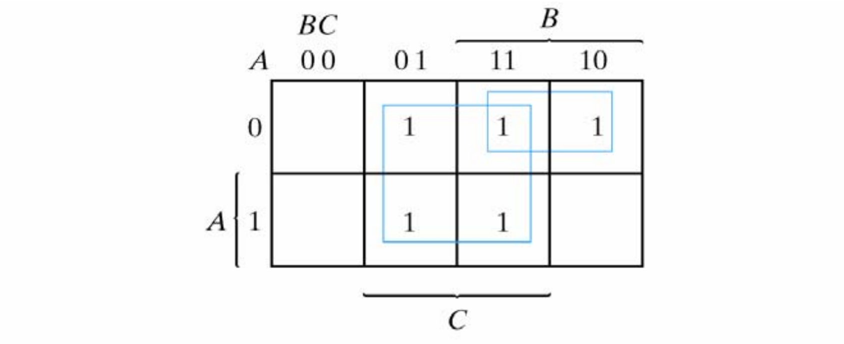


Fig. 3-7 Map for Example 3-4; $A'C + A'B + AB'C + BC = C + A'B$

Four-Variable Map

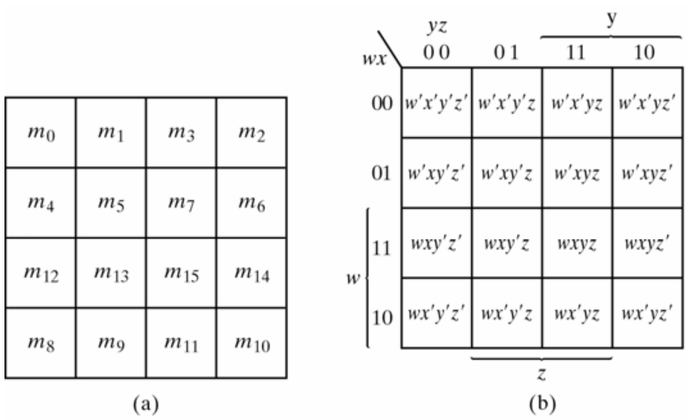


Fig. 3-8 Four-variable Map

Ex 3-5) Simplify the Boolean function,

$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$

$F = y' + w'z' + xz'$

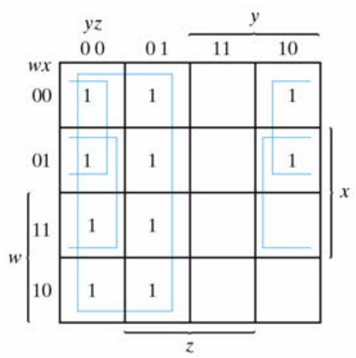


Fig. 3-9 Map for Example 3-5; $F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$

Examples

1. Simplify the Boolean function

$$F(x, y, z) = \Sigma(3, 4, 6, 7)$$

$$yz + xz'$$

2. Simplify the Boolean function

$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$$

$$z' + xy'$$

Prime Implicants

Prime Implicant is a product term obtained by combining the maximum possible number of adjacent squares in the map.

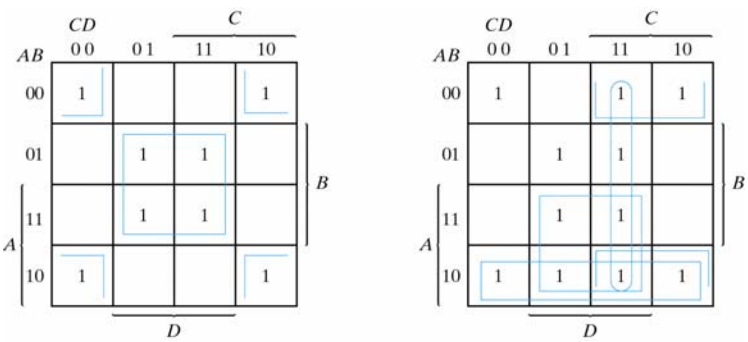
$$F(A,B,C,D) = \Sigma(0,2,3,5,7,8,9,10,11,13,15)$$

$F = BD + B'D' + CD + AD$

$\quad = BD + B'D' + CD + AB'$

$\quad = BD + B'D' + B'C + AD$

$\quad = BD + B'D' + B'C + AB'$



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

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

Fig. 3-11 Simplification Using Prime Implicants

Five-Variable Map

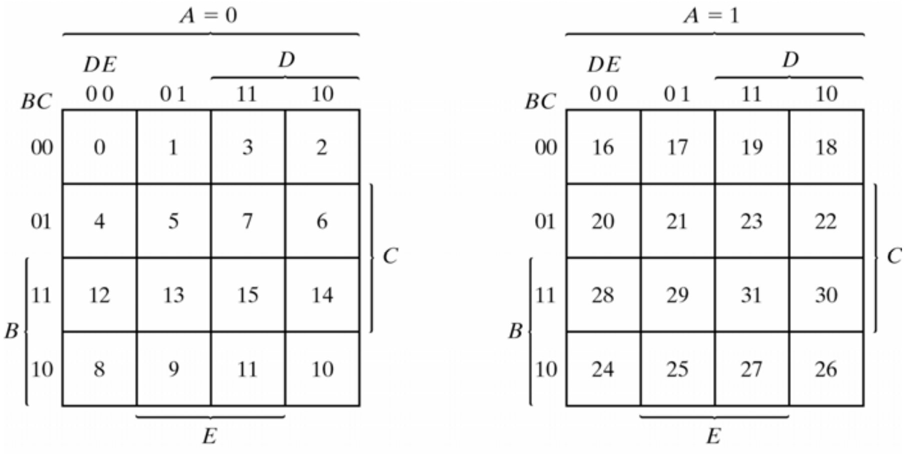


Fig. 3-12 Five-variable Map

Example

Ex 3-7) Simplify the Boolean function,

$F(A,B,C,D,E)=\Sigma(0,2,4,6,9,13,21,23,25,29,31)$

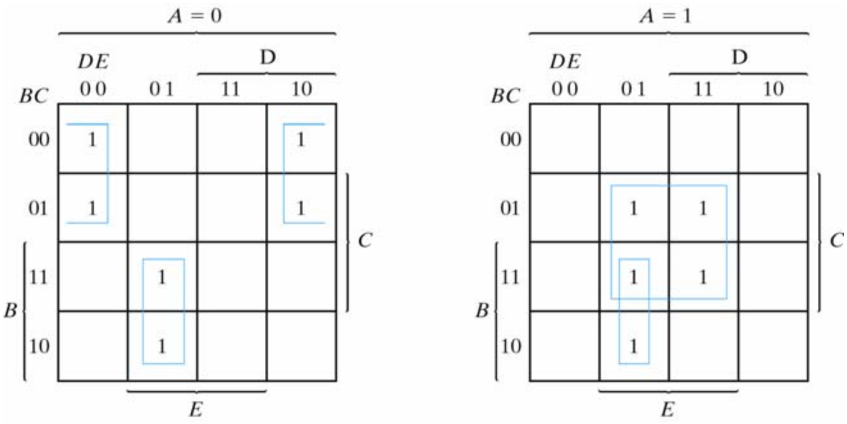


Fig. 3-13 Map for Example 3-7; $F = A'B'E' + BD'E + ACE$

$F = A'B'E' + BD'E + ACE$

Examples

1. Simplify the following Boolean functions by first finding the essential prime implicants:

$$F(A,B,C,D)=\Sigma(0,2,3,5,7,8,10,11,14,15)$$

i) find the essential prime implicants $CD+B'D'$
ii) find the non essential prime implicants $AC+A'BD$
iii) simplify function F $CD+B'D'+AC+A'BD$

2. Simplify the following Boolean functions, using five-variable maps:

$$F(A,B,C,D,E)=\Sigma(0,1,4,5,16,17,21,25,29)$$

Ans) $A'B'D'+B'C'D'+AD'E$

Product of Sums Simplification

Ex 3-8) Simplify the Boolean function,
$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$

a) sum of products
F = B'D' + B'C' +A'C'D

b) product of sum
F' = AB + CD + BD'
F = (A' +B')(C' +D')(B' +D)

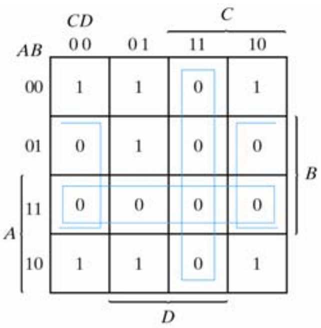
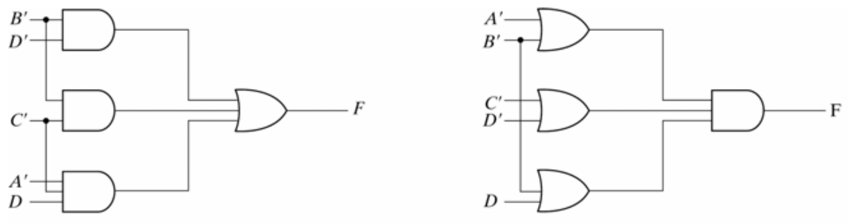


Fig. 3-14 Map for Example 3-8; $F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$
$= B'D' + B'C' + A'C'D = (A'+B')(C'+D')(B'+D)$



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

Fig. 3-15 Gate Implementation of the Function of Example 3-8

Product of Sums Simplification

Table 3-2
Truth Table of Function F

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

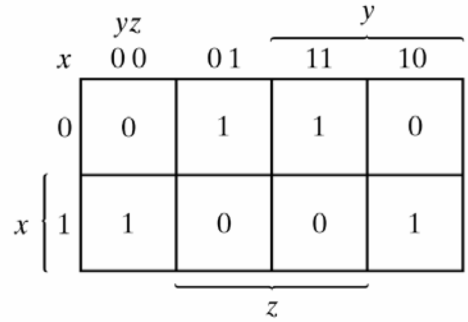


Fig. 3-16 Map for the Function of Table 3-2

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

$F = x'z + xz'$

$F' = xz + x'z'$

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

Examples

Simplify the following Boolean functions in product of sums:

1. $F(w, x, y, z) = \Sigma(0, 2, 5, 6, 7, 8, 10)$

   Ans) $(w'+x')(x+z')(x'+y+z)$

2. $F(A,B,C,D) = \Pi(1,3,5,7,13,15)$

   Ans) $(B'+D')(A+D')$

Don't-Care Conditions

Ex 3-9) Simplify the Boolean function, $F(w, x, y, z) = \Sigma(1,3,7,11,15)$
Don't-care conditions, $d(w, x, y, z) = \Sigma(0, 2, 5)$

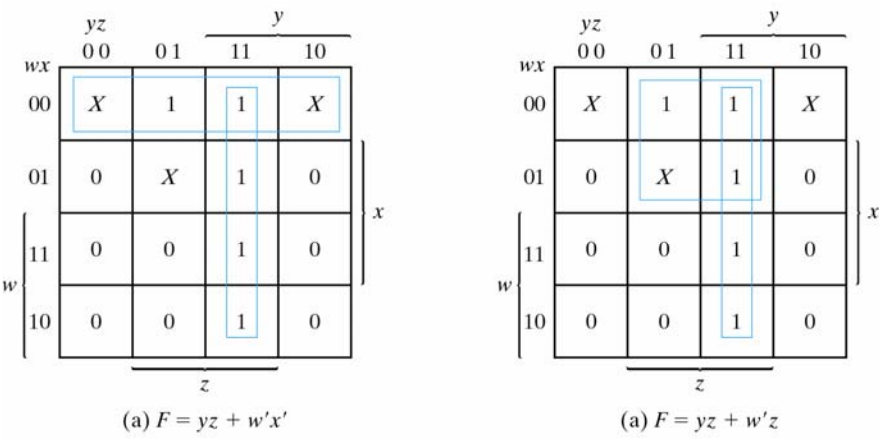


Fig. 3-17 Example with don't-care Conditions

$F(w, x, y, z) = yz + w'x' = \Sigma(0, 1, 2, 3, 7, 11, 15)$
$F(w, x, y, z) = yz + w'z = \Sigma(1, 3, 5, 7, 11, 15)$

NAND and NOR Implementation

- Digital circuits are frequently constructed with NAND or NOR gates rather than AND and OR gates
- NAND and NOR gates are easier to fabricate with electronic components
- Basic gates used in all IC digital logic families

NAND Circuits

- NAND Circuit

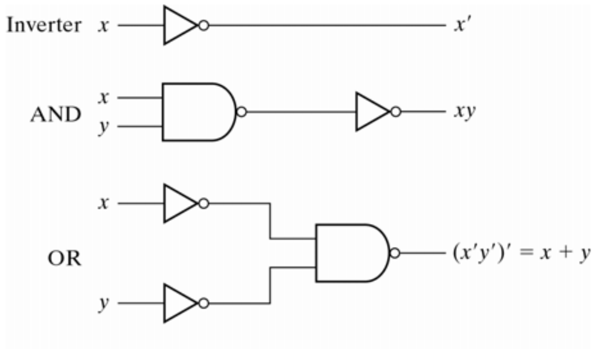


Fig. 3-18 Logic Operations with NAND Gates

Single input NAND gate
= Complement

NAND gate is a universal gate

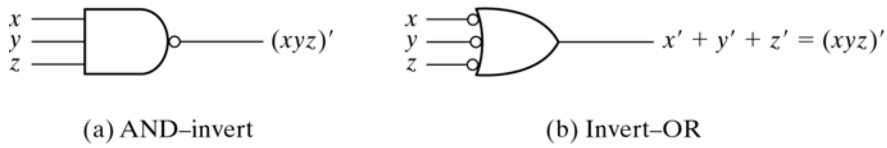


Fig. 3-19 Two Graphic Symbols for NAND Gate

Two-Level Implementation

F = ((AB)'(CD)')' = AB + CD

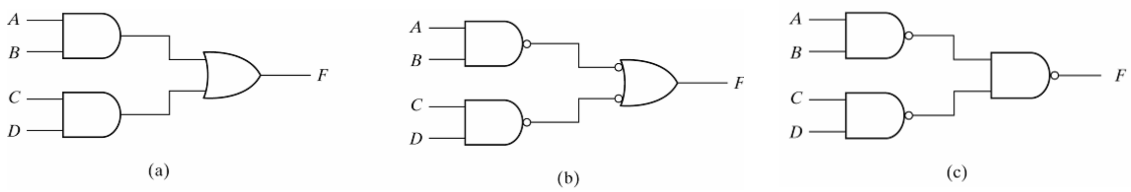


Fig. 3-20 Three Ways to Implement $F = AB + CD$

Ex 3-10) Implement the following Boolean function with NAND gates:

$F(x, y, z) = \Sigma(1, 2, 3, 4, 5, 7) = xy' + x'y + z$

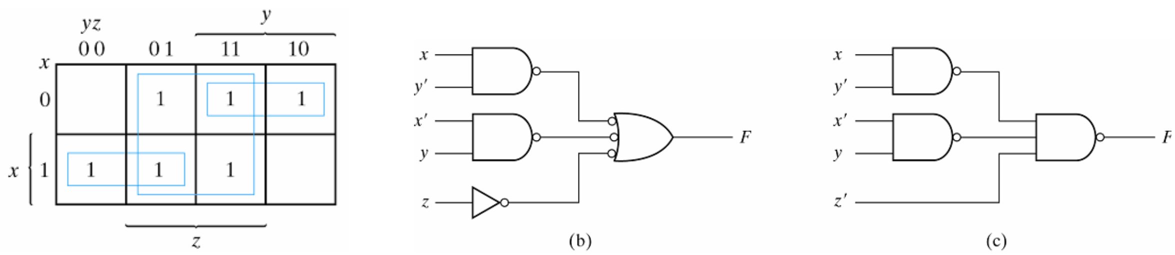


Fig. 3-21 Solution to Example 3-10

Multilevel NAND Circuits

- Convert all AND to NAND with NAND-inverter
- Convert all OR to NAND with inverter-NAND
- Check all the inverter in the diagram. For every inverter that is not compensated by another circle along the same line, insert an inverter (one-input NAND gate) or complement the input literal

Example-Multilevel NAND Circuits

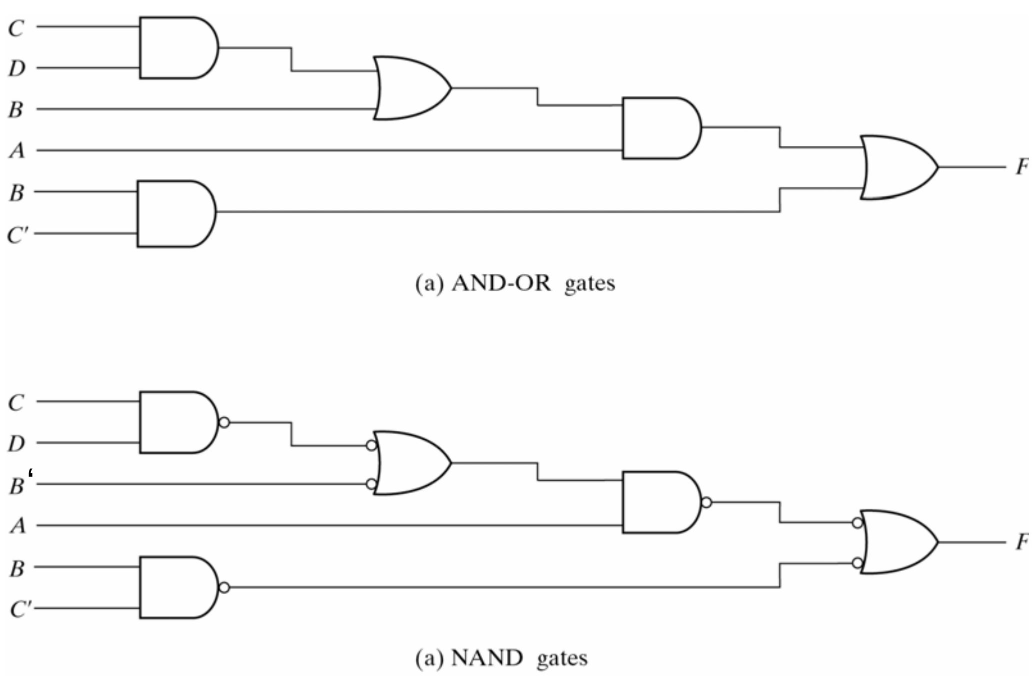
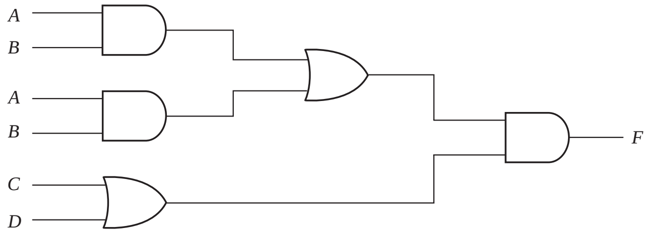


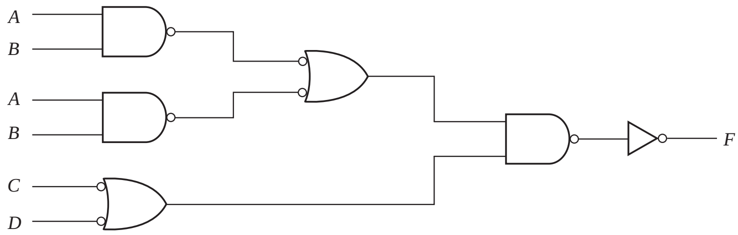
Fig. 3-22 Implementing $F = A(CD + B) + BC'$

Example–Multilevel NAND Circuits

$$F = (AB' + A'B)(C + D')$$



(a) AND-OR gates



(b) NAND gates

NOR Implementation

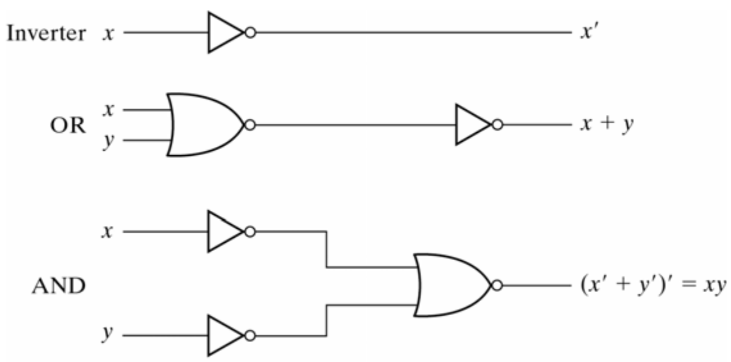
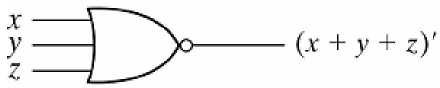
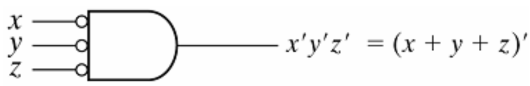


Fig. 3-24 Logic Operations with NOR Gates



(a) OR–invert



(a) Invert–AND

Fig. 3-25 Two Graphic Symbols for NOR Gate

NOR Operation is the Dual of the NAND

- OR gates to NOR gates with NOR-invert
- AND gates to NOR gates with invert-NOR
- Any Inverter that is not compensated by another inverter along the same line needs an inverter or the complementation of the input literal

Example

- $F = (AB' + A'B)(C + D')$

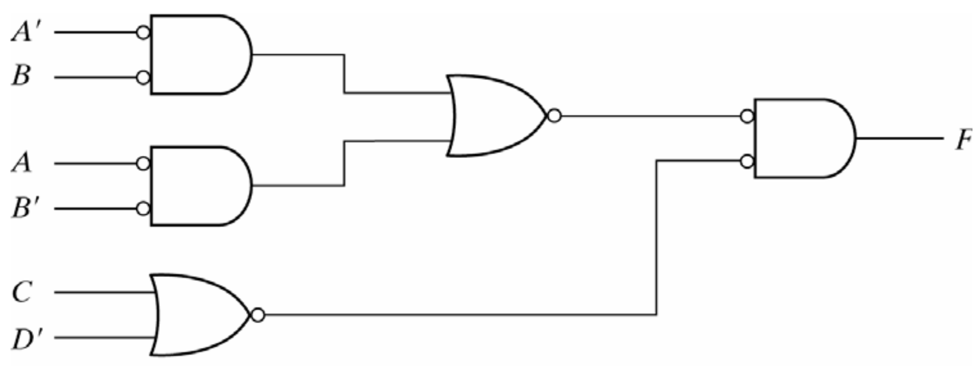
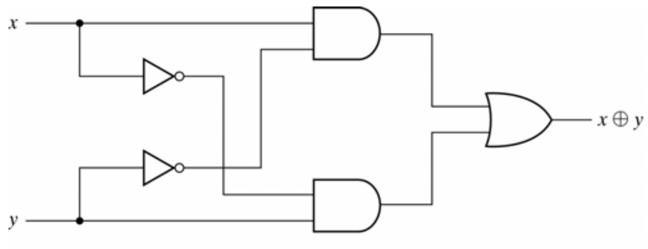


Fig. 3-27 Implementing $F = (AB' + A'B)(C + D')$ with NOR Gates

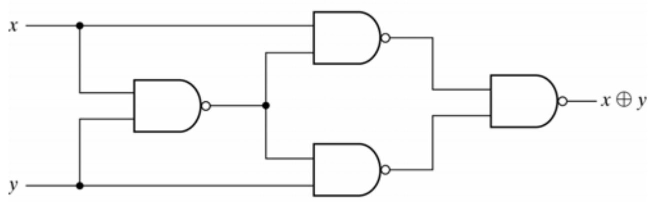
Exclusive-OR Function

$$XOR: x \oplus y = xy' + x'y$$
$$XNOR: (x \oplus y)' = xy + x'y'$$

$$x \oplus 0 = x$$
$$x \oplus 1 = x'$$
$$x \oplus x = 0$$
$$x \oplus x' = 1$$
$$x \oplus y' = x' \oplus y = (x \oplus y)'$$



(a) With AND-OR-NOT gates



(b) With NAND gates

Fig. 3-32 Exclusive-OR Implementations

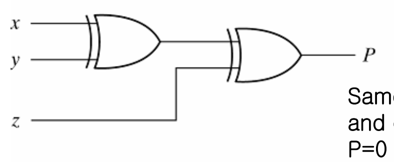
Parity Generation and Checking

Table 3-4
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

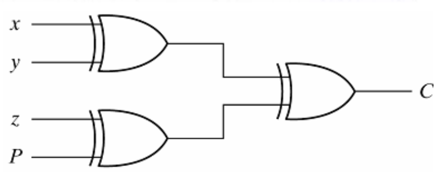
Table 3-5
Even-Parity-Checker Truth Table

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



(a) 3-bit even parity generator

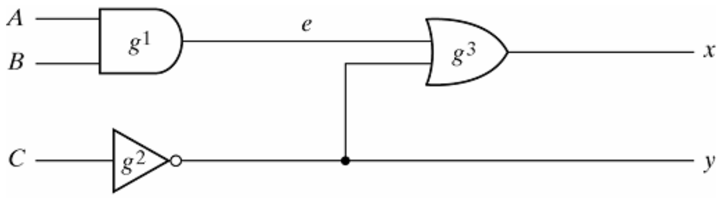
Same circuit for checking and generation
P=0



(a) 4-bit even parity checker

Fig. 3-36 Logic Diagram of a Parity Generator and Checker

HDL(Hardware Description Language)




```
//HDL Example 3-1
//Description of the simple circuit of Fig. 3-37
module smpl_circuit(A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and g1(e,A,B);
   not g2(y, C);
   or  g3(x,e,y);
endmodule
```