CHAPTER 3

3.1

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

x \ yz	00	01	11	10
0	1	0	0	1
1	1	0	0	1

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

x \ yz	00	01	11	10
0	1	0	0	1
1	1	1	1	1

$F = x + z'$

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

x \ yz	00	01	11	10
0	1	1	1	1
1	1	0	0	1

$F = x' + z'$

(d) $F(x, y, z) = \Sigma(1, 4, 5, 7)$

x \ yz	00	01	11	10
0	0	1	0	0
1	1	1	1	0

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

3.2

(a) $F(x, y, z) = \Sigma(0, 1, 4, 7)$

x \ yz	00	01	11	10
0	1	1	0	0
1	1	0	1	0

$F = x'y' + y'z' + xyz$

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

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

(c) $F(x, y, z) = \Sigma(2, 3, 5, 6)$

x \ yz	00	01	11	10
0	0	0	1	1
1	0	1	0	1

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

(d) $F(x, y, z) = \Sigma(1, 2, 4, 7)$

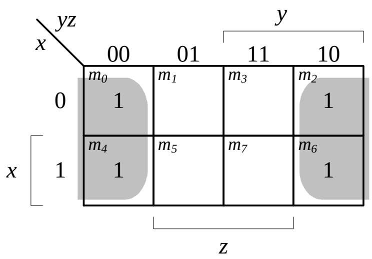
x \ yz	00	01	11	10
0	0	1	0	1
1	1	0	1	0

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

$= x'(y \oplus z) + x(y \oplus z)'$

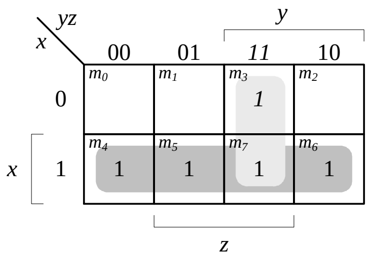
$= x \oplus y \oplus z$

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



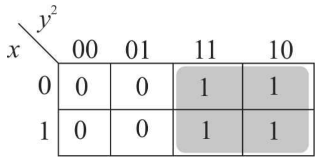
$F = z'$

(f) $F(x, y, z) = \Sigma(3, 4, 5, 6,7)$



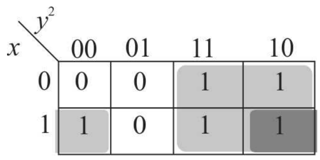
$F = x + y\,z$

3.3 **(a)** $F(x, y, z) = xyz + x'y + xyz'$
$= xyz + x'yz + x'yz' + xyz'$
$= \Sigma(2, 3, 6, 7)$



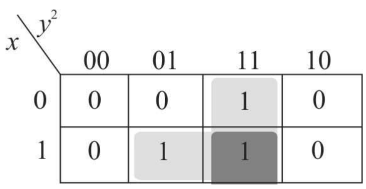
$F = y$

(b) $F(x, y, z) = x'yz + xyz' + xyz + x'yz' + xy'z'$
$= \Sigma(2, 3, 4, 6, 7)$



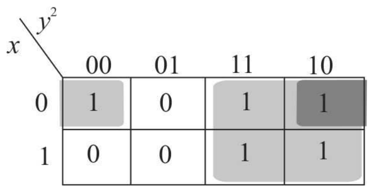
$F = y + xz'$

(c) $F(x, y, z) = x'yz + xz$
$= x'yz + xyz + xy'z$
$= \Sigma(3, 5, 7)$



$F = xz + yz$

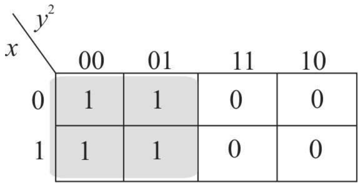
(d) $F(x, y, z) = xyz + x'y + xyz' + x'y'z'$
$= xyz + x'yz + x'yz' + xyz' + x'y'z'$
$= \Sigma(0, 2, 3, 6, 7)$



$F = y + x'z'$

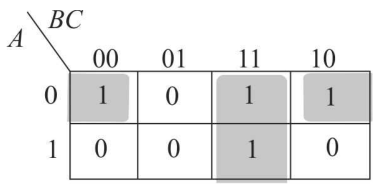
3.4

(a) $F(x, y, z) = \Sigma(0, 1, 4, 5)$



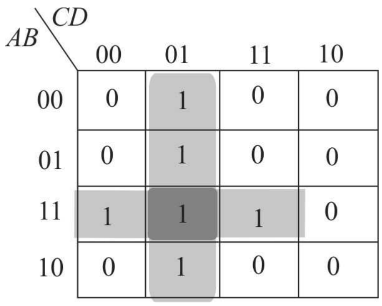
$F = y'$

(b) $F(A, B, C) = \Sigma(0, 2, 3, 7)$



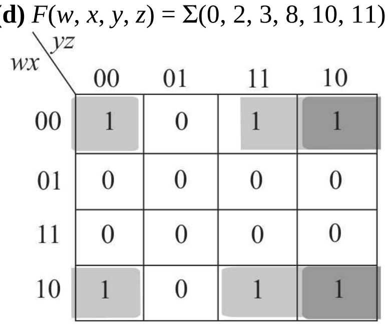
$F = BC + A'C'$

(c) $F(A, B, C, D) = \Sigma(1, 5, 9, 12, 13, 15)$

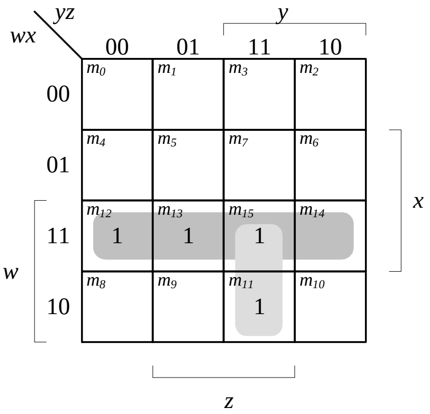


$F = C'D + ABC' + ABD$

(d) $F(w, x, y, z) = \Sigma(0, 2, 3, 8, 10, 11)$

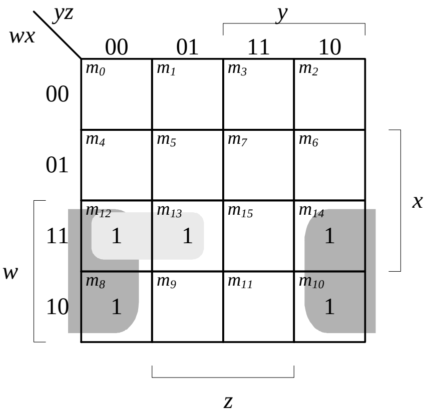


$F = x'y + x'z'$



$F = wx + wyz$

(e)

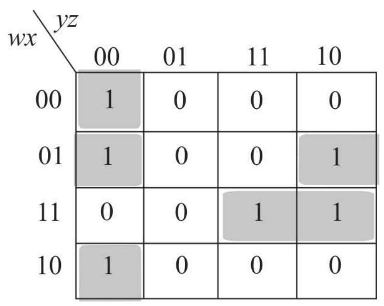


$F = wz' + xy'w$

(f)

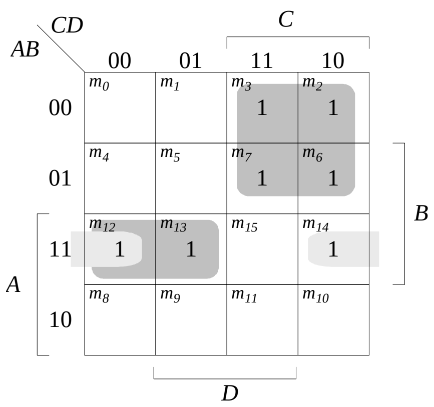
3.5

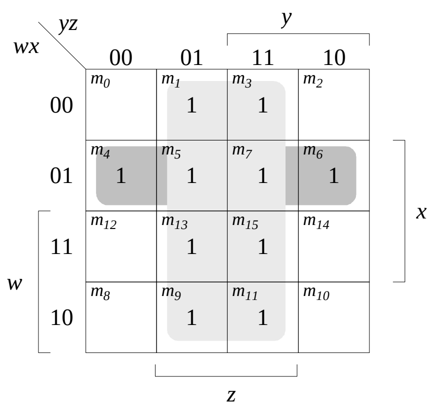
(a) $F(w, x, y, z) = \Sigma(0, 4, 6, 8, 14, 15)$



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

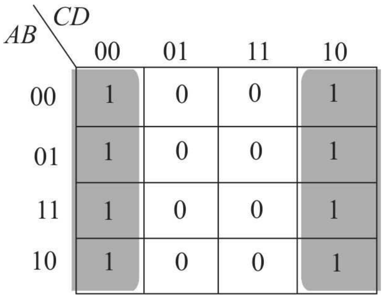
(b) $F = AC' + ABC' + ABD'$





(c) $F = z + xw'$

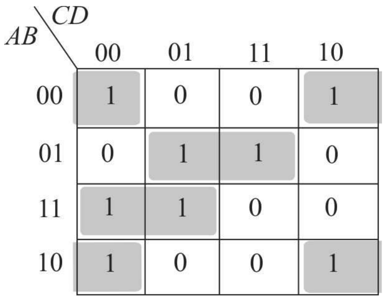
(d) $F(A, B, C, D) = \Sigma(0, 2, 4, 6, 8, 10, 12, 14)$



$F = D'$

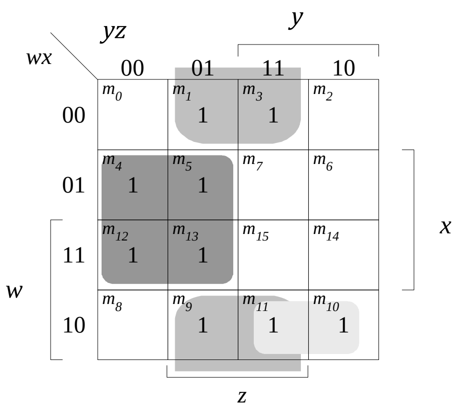
3.6

(a) $B'D'(A'C' + C) + AC'D' + BD(A'C + C')$
$= A'B'C'D' + B'CD' + AC'D' + A'BCD + BC'D$
$= \Sigma(0, 2, 5, 7, 8, 10, 12, 13)$

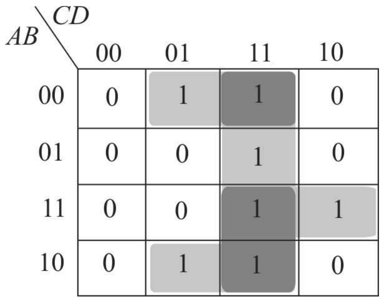


$F = B'D' + ABC' + A'BD$

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

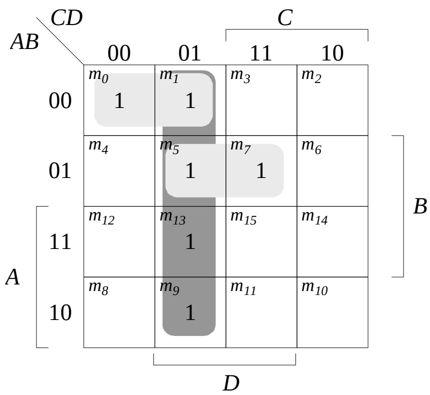


(c) $F = A'BCD + ABC + CD + B'D$
$= \Sigma(1, 3, 7, 9, 11, 14, 15)$

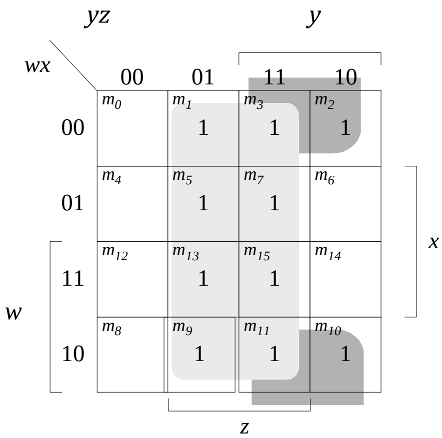


$F = B'D + CD + ABC$

(d) $F = C'D + A'BD + A'B'C'$

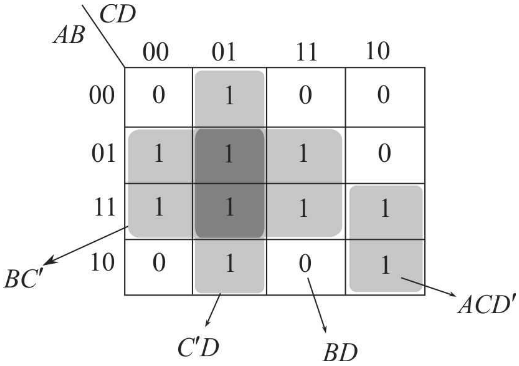


3.7



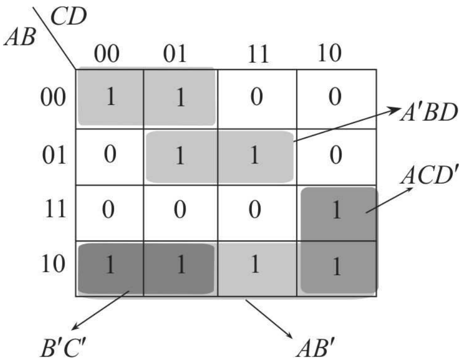
(a) $F = z + x'y$

(b) $ACD' + B'C'D + BCD + BC'$
$= \Sigma(1, 4, 5, 7, 9, 10, 12, 13, 14, 15)$



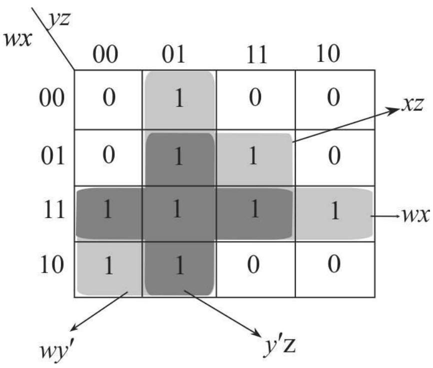
$F = BC' + BD + C'D + ACD'$

(c) $AB'C + B'C' + A'BCD + ACD' + AB'C' + A'C'D$
$= \Sigma(0, 1, 5, 7, 8, 9, 10, 11, 14)$



$F = B'C' + AB' + A'BD + ACD'$

(d) $wxy + xz + w'xz + y'z + wy'$
$= \Sigma(1, 5, 7, 8, 9, 12, 13, 14, 15)$



$F = wy' + y'z + wx + xz$

3.8

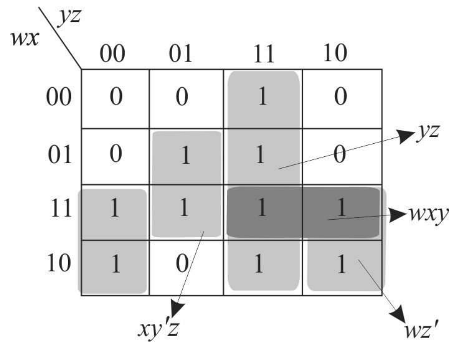
(a) $wxy + yz + xy'z + wz'$

$wxy \rightarrow 111\_ \rightarrow 1110(14), 1111(15)$

$yz \rightarrow \_\ \_11 \rightarrow 0011(3), 0111(7), 1011(11), 1111(15)$

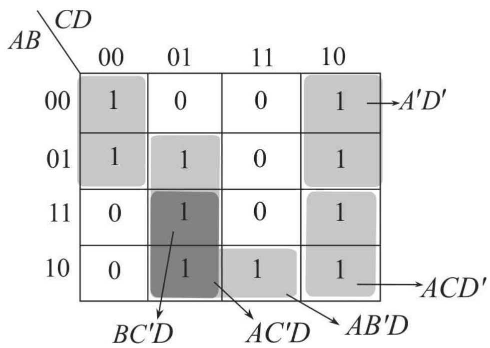
$xy'z \rightarrow \_101 \rightarrow 0101(5), 1101(13)$

$wz' \rightarrow 1\_\ \_0 \rightarrow 1000(8), 1010(10), 1100(12), 1110(14)$



$F = \Sigma(3, 5, 7, 8, 10, 11, 12, 13, 14, 15)$

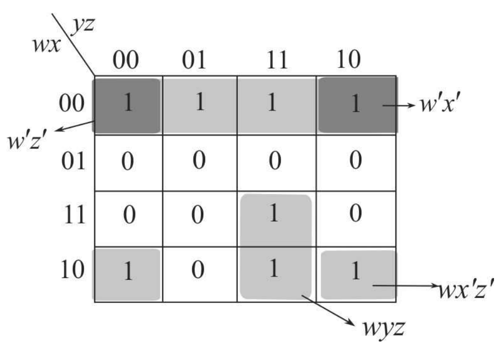
(b) $AC'D + BC'D + ACD' + A'B'D + A'D'$



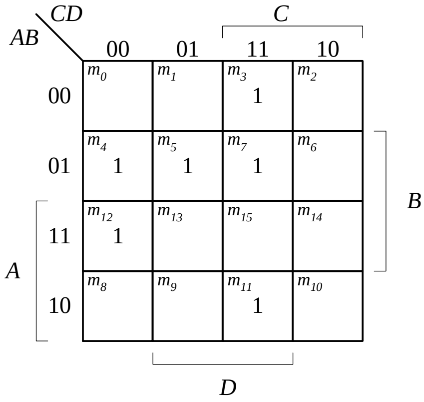
$F = \Sigma(0, 2, 4, 5, 6, 9, 10, 11, 13, 14)$

(c) $wyz + w'x' + wx'z' + x'z'$

$= \Sigma(0, 1, 2, 3, 8, 10, 11, 15)$

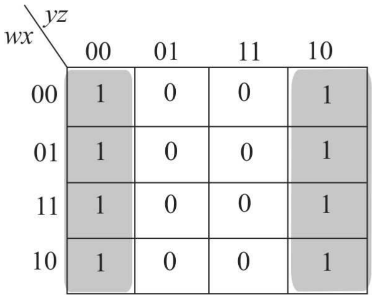


(d) $F = \Sigma(3, 4, 5, 7, 11, 12)$



3.9

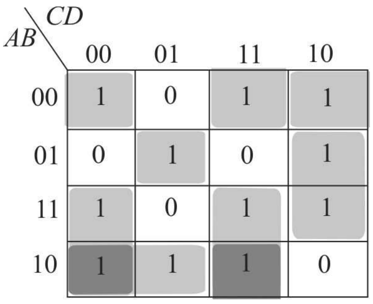
(a) $F(w, x, y, z) = \Sigma(0, 2, 4, 6, 8, 10, 12, 14)$



Essential Prime Implicant: z'

$F = z'$

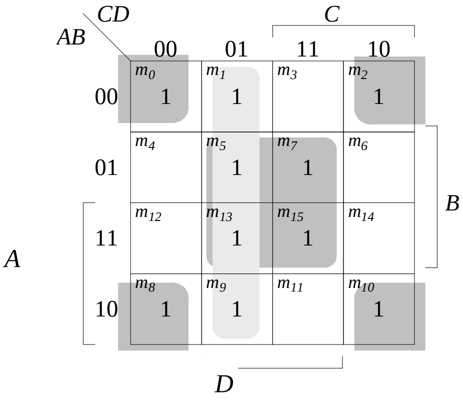
(b) $F(A, B, C, D) = \Sigma(0, 2, 3, 5, 6, 8, 9, 11, 12, 14, 15)$



Prime Implicants: $A'BC'D$, $A'B'D'$, $B'C'D'$, $A'B'C$,
$A'CD'$, $AB'C'$, $AC'D'$, $B'CD$, $AB'D$,
ABD', ACD, ABC, BCD'

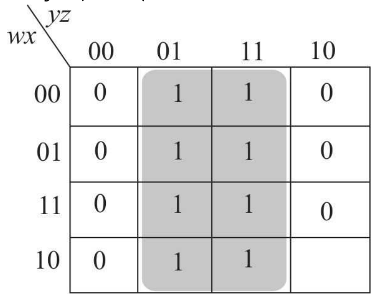
One set of Essential Prime Implicants: $A'BC'D$, $B'CD$, $AB'C'$, $A'B'D'$
BCD', ABC, $AB'C'$

(c)



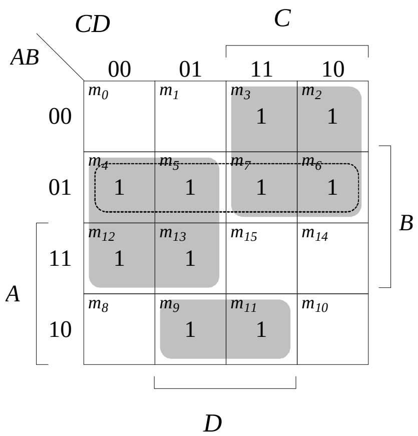
Essential: *BC'*, *AC*, *A'B'D*
Non-Essential: *A'B*
F = *BC'* + *AC* + *A'B'D*

(d) *F*(*w*, *x*, *y*, *z*) = Σ(1, 3, 5, 7, 9, 11, 13, 15)



Essential Prime Implicant = *z*

(e)



Essential: *BD*, *B'C'*, *C'D*
F = *BD* + *B'C'* + *C'D*

(f) $F(w, x, y, z) = \Sigma(1, 3, 4, 6, 7, 9, 10, 12, 13, 15)$

wx \ yz	00	01	11	10
00	0	1	1	0
01	1	0	1	1
11	1	1	1	0
10	0	1	0	1

Prime Implicants: $wx'yz'$, $w'x'z$, $x'y'z$, $w'xy'$,
$xy'z'$, $w'yz$, $w'xy$, $wy'z$,
wxy', xyz, wxz.

One set of Essential Prime Implicants: $wx'yz'$, $w'x'z$, $wy'z$,
$x'y'z$, $w'yz$, $w'xz'$, xyz, wxy'

3.10 **(a)** $F(w, x, y, z) = \Sigma(0, 2, 5, 7, 8, 10, 13, 14, 15)$

Using K-map:

wx \ yz	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	1	1	1
10	1	0	0	1

Using Quine-McCluskey method:

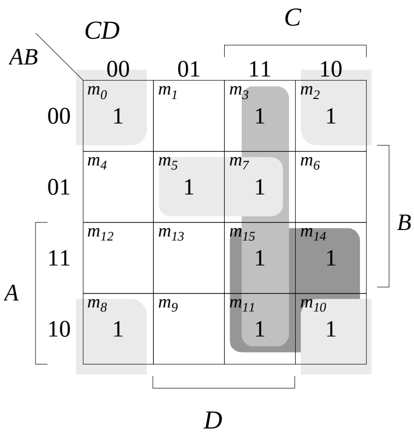
0 (0000) ✓	0.2 (00_0) ✓	0, 2, 8, 10 (_0_0)
2 (0010) ✓	0.8 (_000) ✓	
8 (1000) ✓	2,10(_010) ✓	
5 (0101) ✓	8,10(10_0) ✓	5, 7, 13, 15 (_1_1)
10(1010) ✓	5, 7 (01_1) ✓	
7 (0111) ✓	5,13(_101) ✓	
13(1101) ✓	10,14(1_10)	
14(1110) ✓	7,15(_111) ✓	
15(1111) ✓	13,15(11_1) ✓	
	14,15(111_)	

Prime Implicant Table:

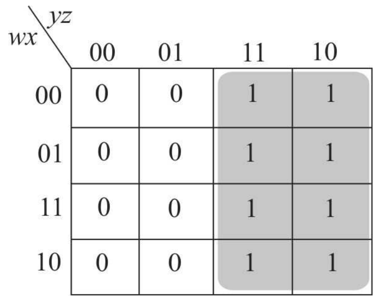
	0	2	5	7	8	10	13	14	15
0, 2, 8, 10	x	x			x	x			
5, 7, 13, 15			x	x			x		x
10, 14						x		x	
14, 15								x	x

$F = (-0-0) + (-1-1) + (1-10) = x'z' + xz + wyz'$

(b) $F = \Sigma(0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$
Essential: AC, $B'D'$, CD, $A'BD$
$F = AC + B'D' + CD + A'BD$



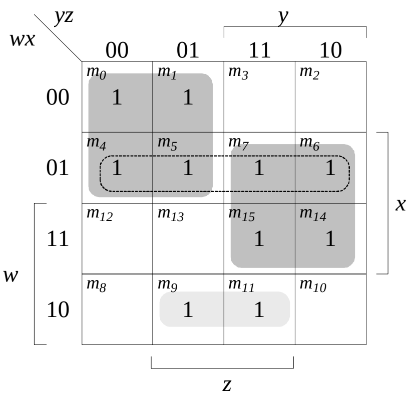
(c) $F(w, x, y, z) = \Sigma(2, 3, 6, 7, 10, 11, 14, 15)$



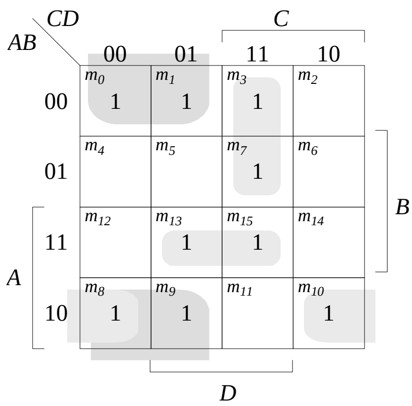
2 (0010) ✓	2, 3 (001_) ✓	2, 3, 6, 7(0_1_) ✓	(2, 3, 6, 7, 10, 11, 14, 15)
3 (0011) ✓	2, 6 (0_10) ✓	2, 3, 10, 11(_01_) ✓	
6 (0110) ✓	2, 10(_010) ✓	2, 6, 10, 14(_ _10) ✓	(_ _1_)
10 (1010) ✓	3, 7(0_11) ✓	3, 7, 11, 15(_ _11) ✓	
7 (0111) ✓	3, 11(_011) ✓	6, 7, 14, 15(_11_) ✓	
11 (1011) ✓	6, 7(011_) ✓	10, 11, 14, 15(1_1_) ✓	
14 (1110) ✓	6, 14(_110) ✓		
15 (1111) ✓	10, 11(101_) ✓		
	10, 14(1_10) ✓		
	7, 15 (_111) ✓		
	11, 15(1_11) ✓		
	14, 15(111_) ✓		

$F = C$

(d) $F = \Sigma(0, 1, 4, 5, 6, 7, 9, 11, 14, 15)$
Essential: *w'y'*, *xy*, *wx'z*
Non-essential: *wx*, *x'y'z, w'wz, w'x'z*
F = *w'y' + xy + wx'z*

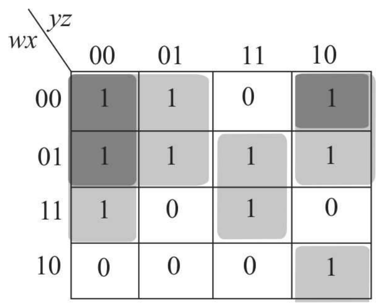


(e) F(A, B, C, D) = S(0, 1, 3, 7 8, 9, 10,13,15)



Essential: *B'C'*, *AB'D'*
Non-essential: *ABD*, *A'CD*, *BCD*
F = B'C' + AB'D' +A'CD +ABD

(f) $F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 7, 10, 12, 15)$



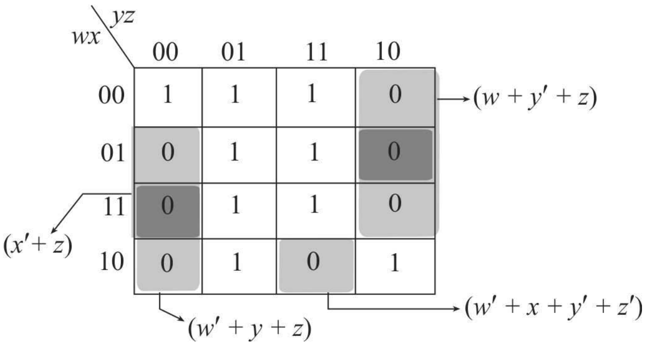
0 (0000) ✓	0, 1(000_) ✓	0, 1, 4, 5 (0_0_)
1 (0001) ✓	0, 2(00_0) ✓	0, 2, 4, 6 (0_ _0)

2 (0010) ✓	0, 4(0_00) ✓	4, 5, 6, 7 (01_ _)
4 (0100) ✓	1, 5(0_01) ✓	
5 (0101) ✓	2, 6(0_10) ✓	
6 (0110) ✓	2, 10(−010) ✓	
10 (1010) ✓	4, 5(010_) ✓	
12 (1100) ✓	4, 6(01_0) ✓	
7 (0111) ✓	4, 12(_100) ✓	
15 (1111) ✓	5, 7(01_1) ✓	
	6, 7(011_) ✓	
	7, 15(_111) ✓	

Essential Prime Implicants: (0, 1, 4, 5)(2, 10)(4, 12)(7, 15)(0, 2, 4, 6)

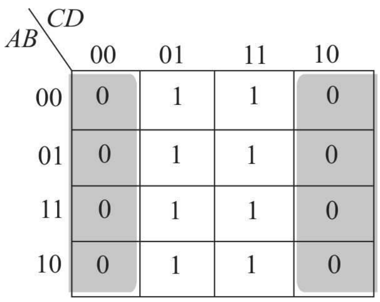
$$F = w'y' + x'yz' + xy'z' + xyz + w'z'$$

3.11 $F(w, x, y, z) = \Sigma(0, 1, 3, 5, 7, 9, 10, 13, 15)$



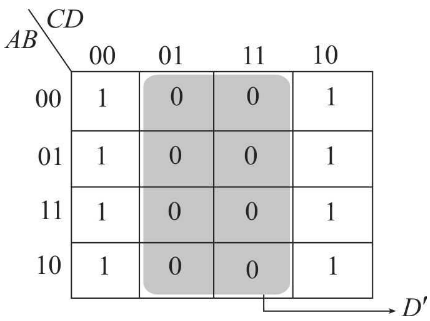
$F = (x' + z)(w' + y + z)\,(w + y' + z)\,(w' + x + y' + z')$

3.12 **(a)** $F(A, B, C, D) = \pi(0, 2, 4, 6, 8, 10, 12, 14)$



$F = D$

(b) $F(A, B, C, D) = \pi(1, 3, 5, 7, 9, 11, 13, 15)$

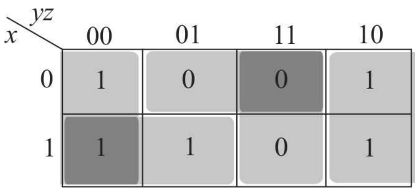


$F = D'$

It shows a logical product of all even maxterms is equal to the complement of logical product of all odd maxterms. For *n* variable,

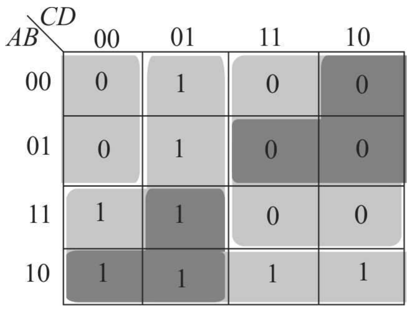
$\pi(M_2i) = (\pi(M_2i + 1))'$ where, $i = 0, 1, 2, \ldots, (2^n/2 - 1)$

3.13 **(a)** $F = xz' + y'z' + yz' + xy' = \Sigma(0, 2, 4, 5, 6)$



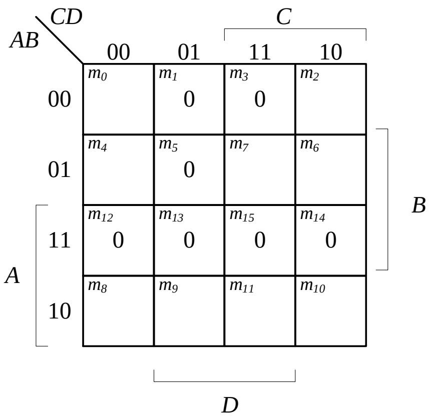
$F = z' + xy'$ (Sum of Product)
$F = (x + z')(y' + z')$ (Product of Sum)

(b) $F = AC'D' + C'D + AB' + AB'CD$
$= \Sigma(1, 5, 8, 9, 10, 11\ 12, 13)$
$= \pi(0, 2, 3, 4, 6, 7, 14, 15)$



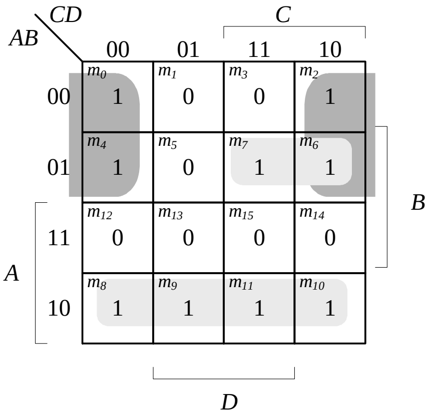
$F = AC' + C'D + AB'$ (Sum of Product)
$F = (A + D)(A + C')(B' + C')$ (Product of Sum)

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

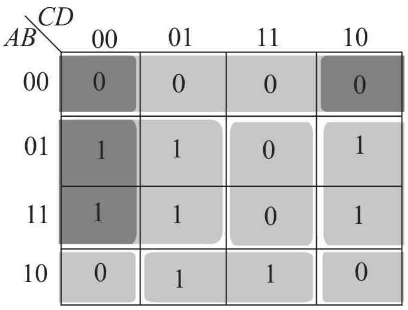


$F' = AB + BC'D$

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

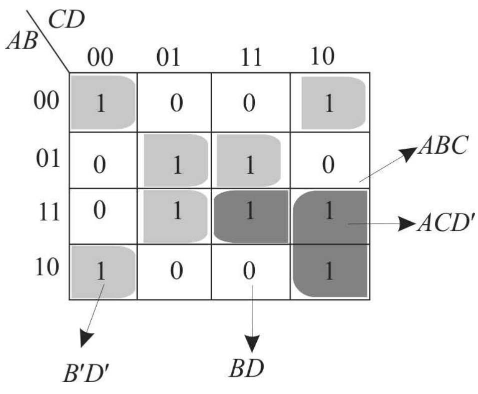


(d) $F = BD' + AC'D + BC' + AB'CD$
$= \Sigma(4, 5, 6, 9, 11, 12, 13, 14)$
$= \pi(0, 1, 2, 3, 7, 8, 10, 15)$



$F = BC' + BD' + AB'D \rightarrow$ SOP
$F = (A + B)(B + D)(B' + C' + D') \rightarrow$ POS

3.14 $F(A, B, C, D) = A'B'D' + AB'D' + BD + ABCD'$



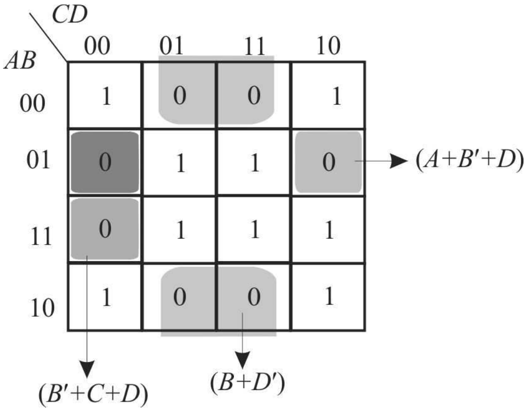
Two SOP form of F is,

No. of literals

$F(A, B, C, D) = B'D' + BD + ACD'$ $\qquad 2 + 2 + 3 = 7$

and

$F(A, B, C, D) = B'D' + BD + ABC$ $\qquad 2 + 2 + 3 = 7$

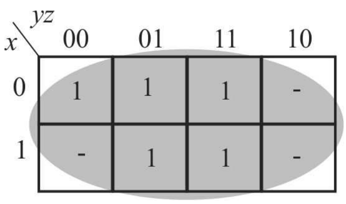


POS of form of F is,

$F(A, B, C, D) = (B + D')\,(B' + C + D)\,(A + B' + D)$ $\qquad$ No. of literals = 2 + 3 + 3 = 7

3.15 **(a)** $F(x, y, z) = \Sigma(0, 1, 3, 5, 7)$

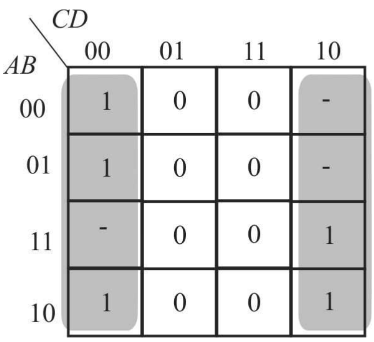
$d \;= \Sigma(2, 4, 6)$



$F = 1$

(b) $F = \Sigma(0, 4, 8, 10, 14)$

$d = \Sigma(2, 6, 12)$



$F = D'$

(c) $F = \Sigma(5, 6, 7, 11, 14, 15)$
$d = \Sigma(3, 9, 13)$

AB \ CD	00	01	11	10
00	0	0	-	0
01	0	1	1	1
11	0	-	1	1
10	0	-	1	0

$F = BD + BC + AD$

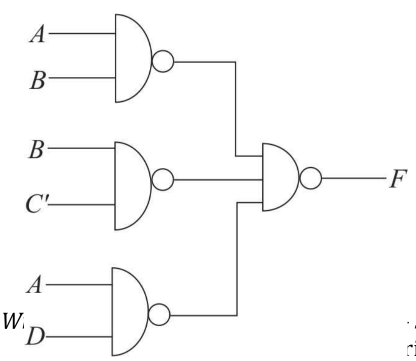
(d)

AB \ CD	00	01	11	10
00	-	0	-	1
01	1	0	1	-
11	1	-	0	1
10	-	0	0	1

$F = D' + A'C$

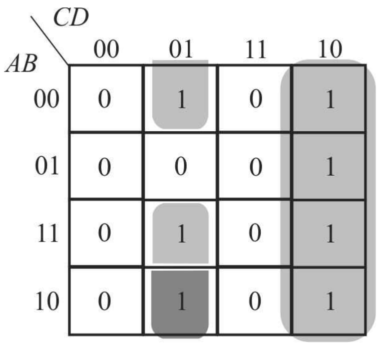
3.16 **(a)** $F(A, B, C, D) = AD + BC'D' + ABC + A'BC'D$

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	1	1
10	0	1	1	0

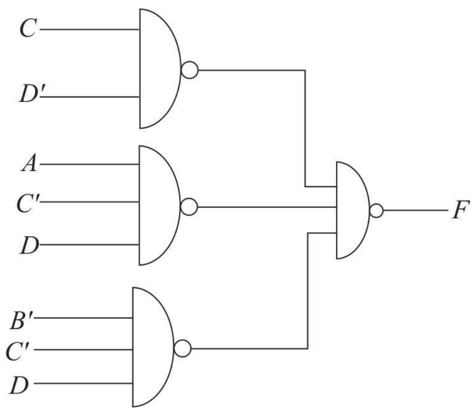


$F = AB + BC' + AD$
$= [(AB)'(BC')'(AD)']'$

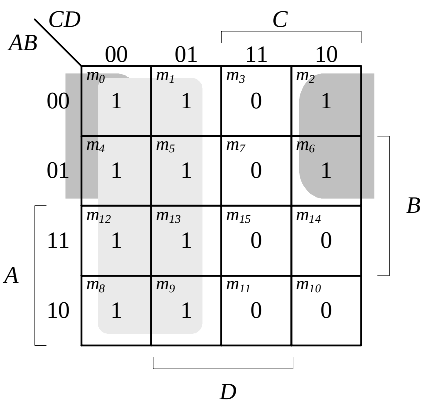
(b) $F(A, B, C, D) = A'B'C'D + CD' + AC'D$



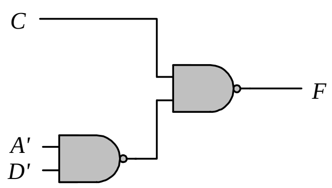
$F(A, B, C, D) = CD' + AC'D + B'C'D$
$= [(CD)'(AC'D)'(B'C'D)']'$



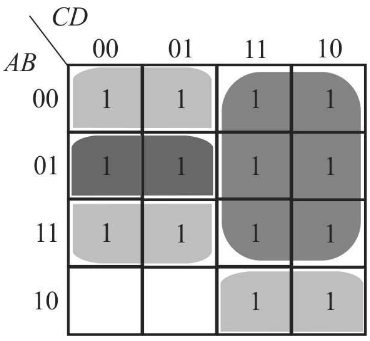
(c) $F = (A' + C' + D')(A' + C')(C' + D')$
$F' = (A' + C' + D')' + (A' + C')' + (C' + D')'$
$F' = ACD + AC + CD$



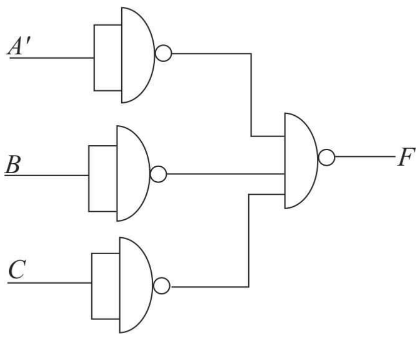
$F = C' + A'D'$
$F = (C(A + D))'$
$F = (C(A'D')')'$



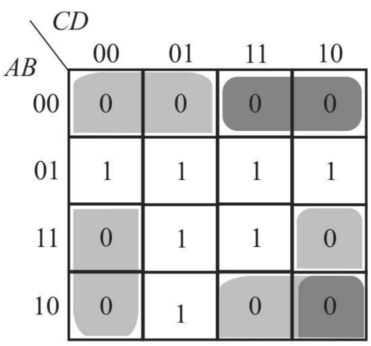
(d) $F(A, B, C, D) = A' + AB + B'C + ACD$



$F = A' + B + C$
$\quad = [(A'A')' (BB)' (CC)']'$

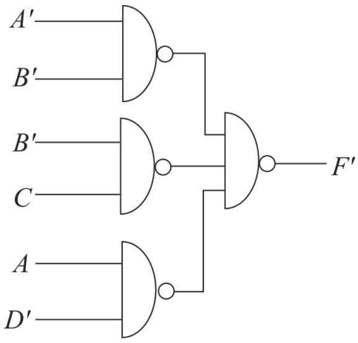


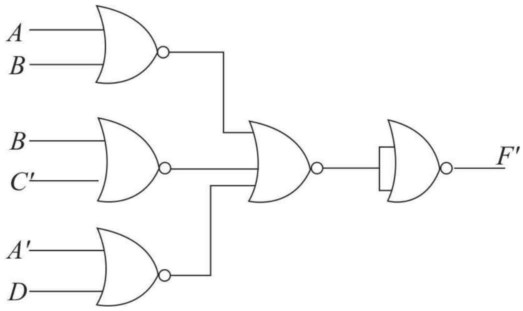
3.17 $F(A, B, C, D) = \Sigma(4, 5, 6, 7, 9, 13, 15)$



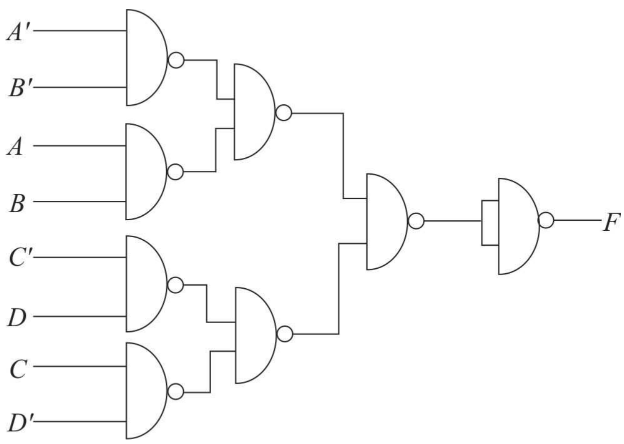
$F' = B'A' + B'C + AD'$

By NAND:
$\quad = [(B'A')'(B'C)'(AD')']'$
By NOR:
$\quad = [(A + B)' + (B + C')' + (A' + D)']'$





3.18 $F(A, B, C, D) = (A \oplus B)' (C \oplus D)$
$= (A'B + AB')' (C'D + CD')$
$= (A + B') (A' + B)(C'D + CD')$
$= (A'B' + AB)(C'D + CD')$
$= [(A'B')'(AB)']' + [(C'D)'(CD')']'$

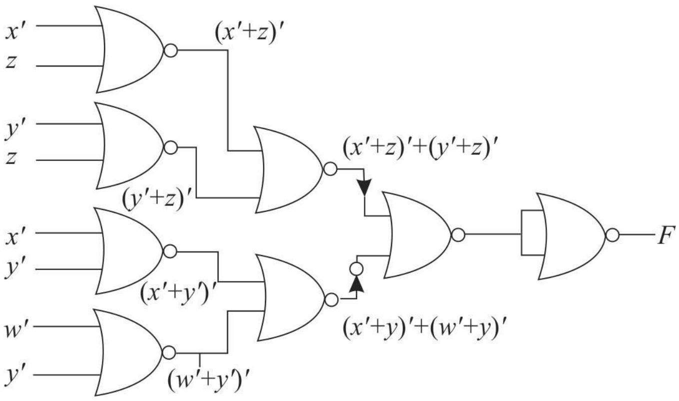


3.19 **(a)** $F(w, x, y, z) = wx'y + xy'z' + w'yz' + xy$

wx \ yz	00	01	11	10
00	0	0	0	1
01	1	0	1	1
11	1	0	1	1
10	0	0	1	1

Simplified Boolean equation:

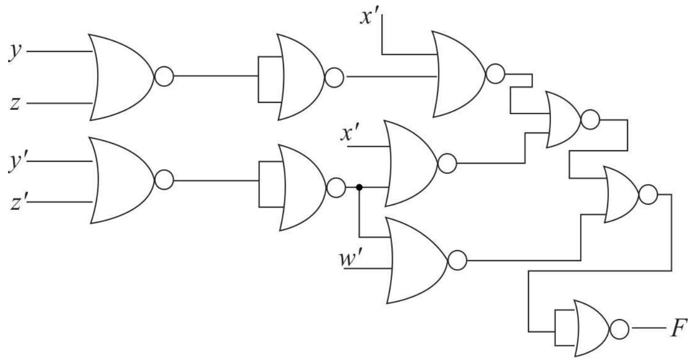
$$F(w, x, y, z) = xz' + yz' + xy + wy$$
$$= (x' + z)' + (y' + z)' + (x' + y')' + (w'+y')'$$



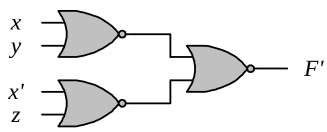
(b) $F(w, x, y, z) = \Sigma(4, 7, 11, 12, 15)$

wx \ yz	00	01	11	10
00	0	0	0	0
01	1	0	1	0
11	1	0	1	0
10	0	0	1	0

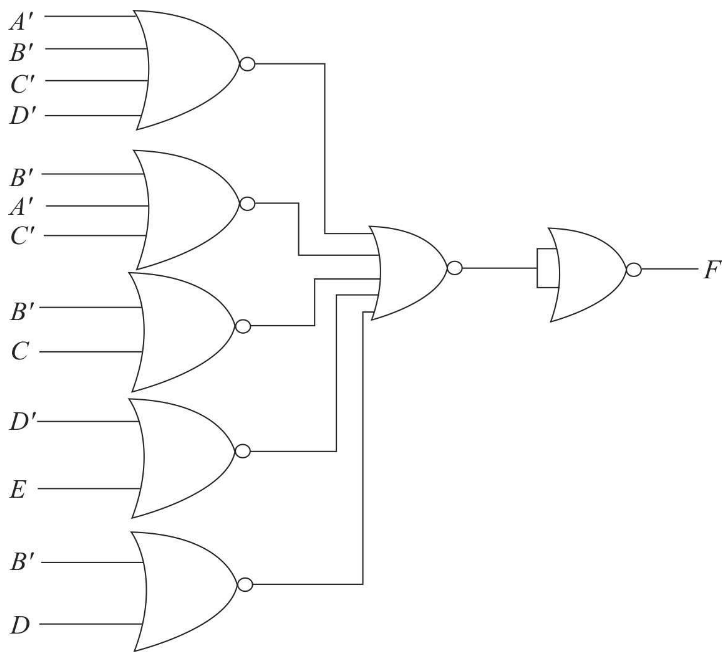
$$F = xy'z' + xyz + wyz$$
$$= [(x' + (y + z))' + (x' + (y' + z'))' + (w' + (y' + z'))']$$



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



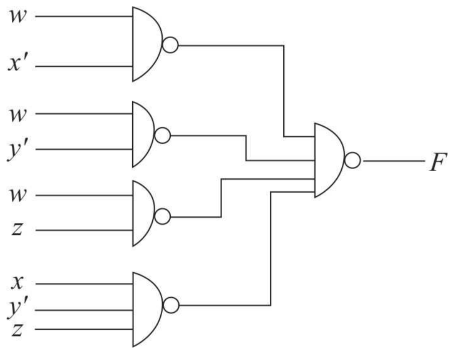
3.20 $F = BC(D + C)A + (BC' + DE') + BD'$
$= ABCD + ABC + BC' + DE' + BD'$
$= [[(A' + B' + C' + D')' + (A' + B' + C')' + (B' + C)' + (D' + E)' + (B' + D)']]'$



3.21 $F = w(x' + y' + z) + xy'z$

$= wx' + wy' + wz + xy'z$

$= [(wx')' (wy')' (wz)' (xy'z)']'$



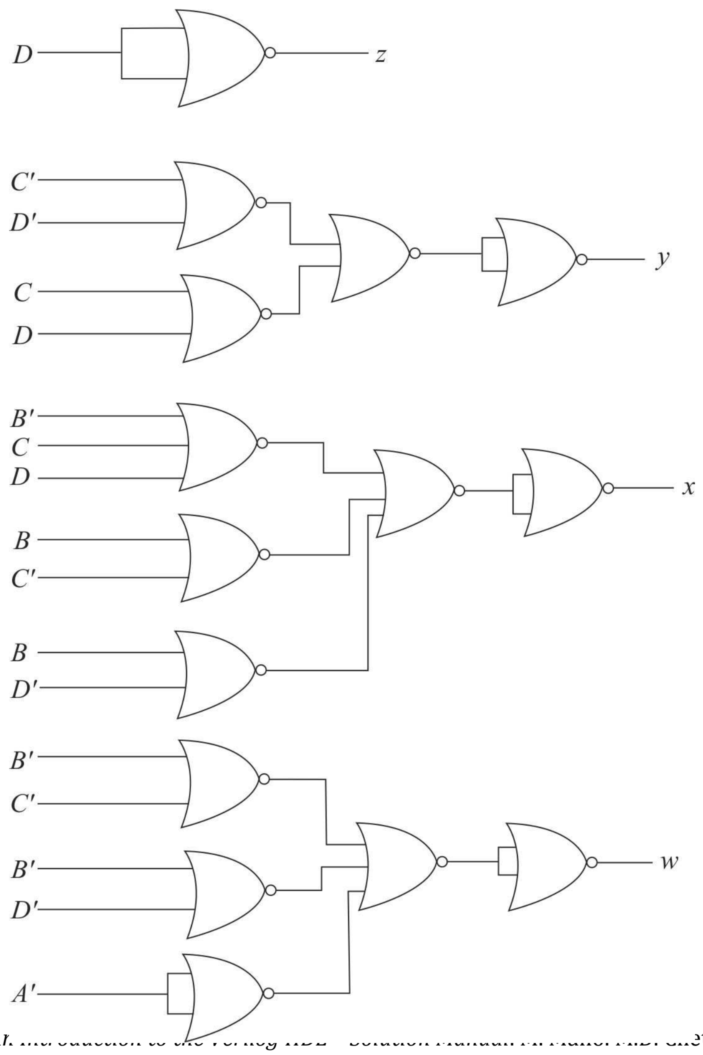
3.22 $z = D'$

$y = CD + (C + D)' = [[(C' + D')' + (C + D)']']'$

$x = B(C + D)' + B'(C + D)$

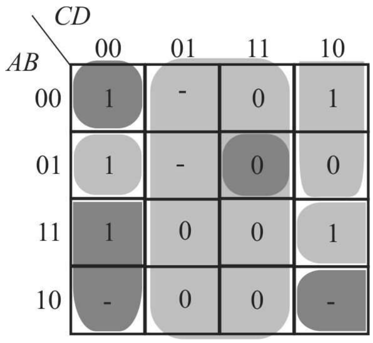
$= BC'D' + B'C + B'D$

$= [[(B' + C + D)' + (B + C')' + (B + D')']'$

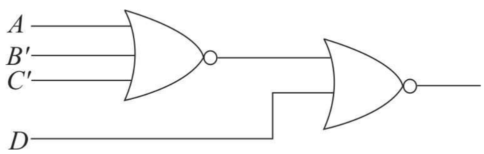


$w = B(C + D) + A$
$= BC + BD + A$
$= [[(B' + C')' + (B' + D')' + (A'A')']]'$

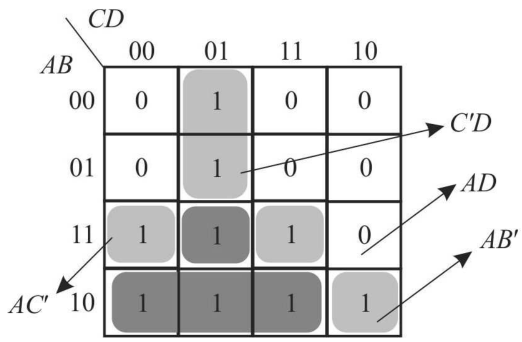
3.23 $F(A, B, C, D) = \Sigma(0, 2, 4, 12, 14) + d(1, 5, 8, 10)$



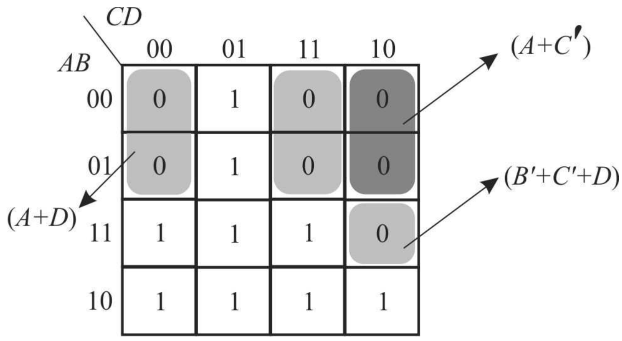
$F = C'D' + AD' + B'D'$
OR
$F' = D + A'BC$
$F = [D + A'BC]'$
$= [D + (A + B' + C')']'$



3.24 $F(A, B, C, D) = \Sigma(1, 5, 8, 9, 10, 11, 12, 13, 15)$

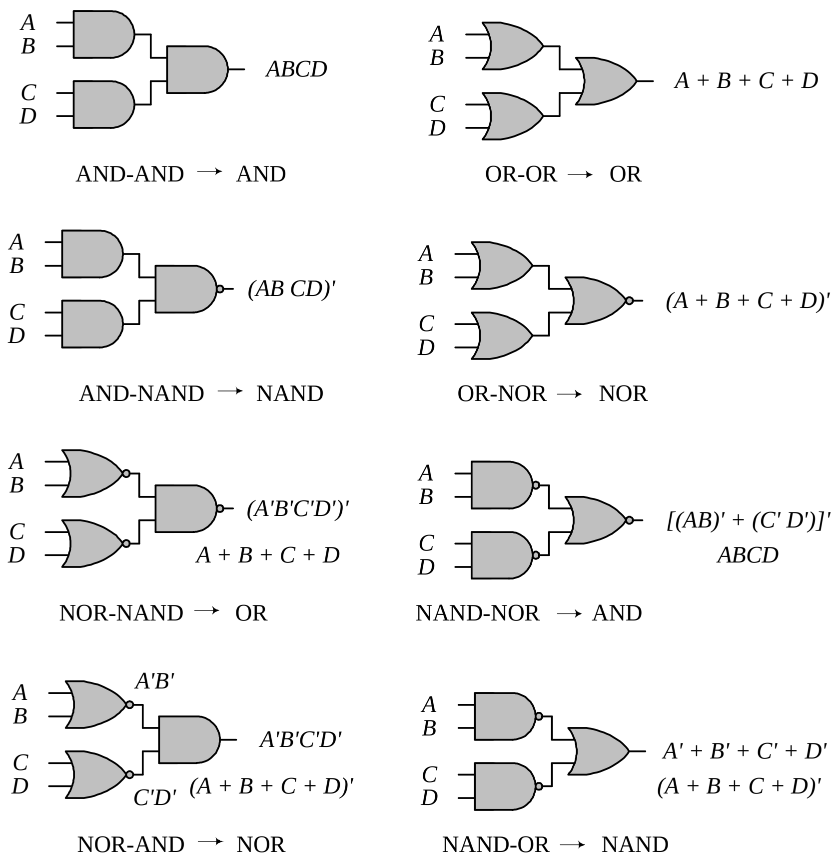


$F(A, B, C, D) = C'D + AB' + AC' + AD$ → **(a)** AND-OR
$= ((C'D)'(AB')'(AC')'(AD)')'$
$= ((C+D')(A'+B)(A'+C)(A'+D'))'$ →**(b)** OR-NAND
$= ((C + D')' + (A' + B)' + (A' + C)' + (A' + D')'$ →**(c)** NOR-OR
$= ((C'D)'(AB')'(AC')'(AD)')'$ →**(d)** NAND-NAND



$F(A, B, C, D) = (A + D)(A + C')(B' + C' + D)$ →**(e)** OR-AND
$= ((A + D)' + (A + C')' + (B' + C' + D')')'$ →**(f)** NOR-NOR
$= (A'D')' (A'C)' (BCD')'$ →**(g)** NAND-AND

3.25



The degenerate forms use 2-input gates to implement the functionality of 4-input gates.

3.26 $f = abc' + b'd' + a'd' + b'cd'$

$ab \backslash cd$	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	1	0	
10	1	0	0	1

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

$ab \backslash cd$	00	01	11	10
00	1	1	0	1
01	1	1	1	1
11	0	0	0	0
10	1	0	0	1

Group the overlapping is

$F = fg = \Sigma(0, 2, 4, 5, 8, 10)$

$ab \backslash cd$	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	0	0
10	1	0	0	1

$F = a'd' + b'd'$

3.27 $(x \text{ XNOR } y) = x'y' + xy$

Dual of $(x \text{ XNOR } y) = (x + y)\,(x' + y')$

$= xy' + x'y$

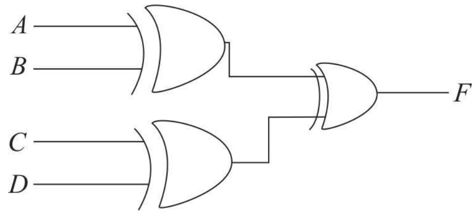
$= ((x' + y)(x + y'))'$

$= (x'y' + xy)'$

$= (x \text{ XNOR } y)'$

3.28 4 bit even parity generator

$F = \Sigma(1, 2, 4, 7, 8, 11, 13, 14)$

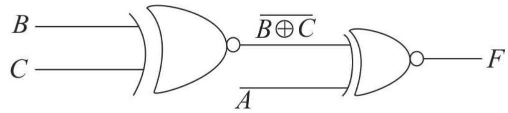


$= A \oplus B \oplus C \oplus D$

3-bit parity checker using even parity

$F(A, B, C) = \Sigma(0, 3, 6, 5)$

$= A \oplus B \oplus C$

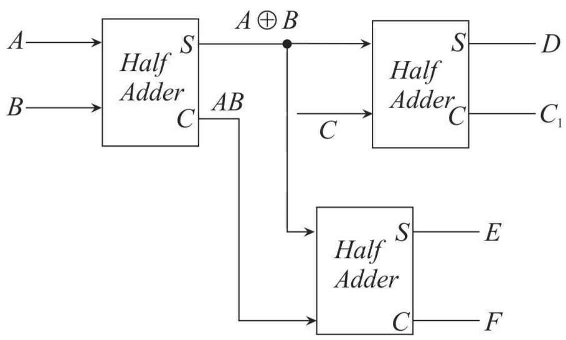


3.29 $D = (\overline{A \oplus B})\, C + (A \oplus B)\, C'$

$E = (A \oplus B) \oplus AB$

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

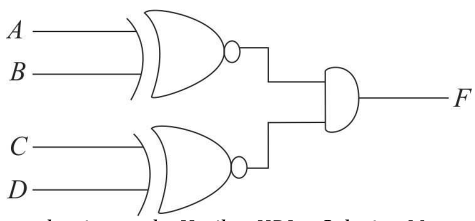
$C_1 = (A \oplus B)\, C$



3.30 $F = A'B'C'D' + ABC'D' + A'B'CD + ABCD$

$= C'D'\,(\overline{A \oplus B}) + CD(A \oplus B)$

$= (\overline{A \oplus B})\,(\overline{C \oplus D})$



3.31 Note: It is assumed that a complemented input is generated by another circuit that is not part of the circuit that is to be described.

(a)
```
module Fig_3_20a_gates (F, A, B, C, C_bar, D);
  output  F;
  input       A, B, C, C_bar, D;
  wire    w1, w2, w3, w4;
  and         (w1, C, D);
  or      (w2, w1, B);
  and         (w3, w2, A);
  and         (w4, B, C_bar);
  or          (F, w3, w4);
endmodule
```

(b)
```
module Fig_3_20b_gates (F, A, B, B_Bar, C, C_bar, D);
  output  F;
  input       A, B, B_bar, C, C_bar, D;
  wire    w1, w2, w3, w4;
  not     (w1_bar, w1);
  not     (w3_bar, w3);
  not     (w4_bar, w4);
  nand        (w1, C, D);
  or          (w2, w1_bar, B);
  nand        (w3, w2, A);
  nand        (w4, B, C_bar);
  or          (F, w3_bar, w4_bar);
endmodule
```

(c)
```
module Fig_3_21a_gates (F, A, A_bar, B, B_bar, C, D_bar);
  output F;
  input   A, A_bar, B, B_bar, C, D_bar;
  wire        w1, w2, w3, w4;
  and     (w1, A, B_bar);
  and     (w2, A_bar, B);
  or      (w3, w1, w2);
  or      (w4, C, D_bar);
  and     (F, w3, w4);
endmodule
```

(d)
```
module Fig_3_21b_gates (F, A, A_bar, B, B_bar, C_bar, D);
  output  F;
  input       A, A_bar, B, B_bar, C_bar, D;
  wire    w1, w2, w3, w4, F_bar;
  nand        (w1, A, B_bar);
  nand        (w2, A_bar, B);
  not     (w1_bar, w1);
  not     (w2_bar, w2);
  or      (w3, w1_bar, w2_bar);
  or          (w4, w5, w6);
  not     (w5, C_bar);
  not     (w6, D);
  nand        (F_bar, w3, w4);
  not     (F, F_bar);
endmodule
```

(e)
```
module Fig_3_24_gates (F, A, A_bar, B, B_bar, C, D_bar);
  output  F;
  input       A, A_bar, B, B_bar, C, D_bar
```

```
    wire     w1, w2, w3, w4, w5, w6, w7, w8, w7_bar, w8_bar;
    not      (w1_bar, w1);
    not      (w2_bar, w2);
    not      (w3, E_bar);
    nor      (w1, A, B);
    nor      (W2, C, D);
    and      (F, w1_bar, w2_bar, w3);
  endmodule
```

(f)

```
  module Fig_3_25_gates (F, A, A_bar, B, B_bar, C, D_bar);
    output F;
    input  A, A_bar, B, B_bar, C, D_bar;
    wire       w1, w1_bar, w2, w2_bar;
    wire       w3, w4, w5, w6, w7, w8;
    not        (w1, A_bar);
    not        (w2, B);
    not        (w3, A);
    not    (w4, B_bar);
    and    (w5, w1_bar, w2_bar));
    and    (w6, w3, w4);
    nor    (w7, w5, w6);
    nor    (w8, c, d_bar);
    and    (F, w7, w8);
  endmodule
```

3.32 Note: It is assumed that a complemented input is generated by another circuit that is not part of the circuit that is to be described.

Note: Because the signals here are all scalar–valued, the logical operators (!, &&, and ||) are equivalent to the bitwise operators (~, &, |). If the operands are vectors the bitwise operators produce a vector result; the logical operators would produce a sclara result (true or false).

(a)

```
  module Fig_3_20a_CA (F, A, B, C, C_bar, D);
    output  F;
    input       A, B, C, C_bar, D;
    wire     w1, w2, w3, w4;
    assign  w1 = C && D;
    assign  w2 = w1 || B;
    assign  w3 = !(w2 && A);
    assign  w4 = !w3;
    assign  w5 = !(B && C_bar);
    assign  w5_bar = !w5;
    assign  F = w4 || w5_bar);
  endmodule
```

(b)

```
  module Fig_3_20b_CA (F, A, B, C, C_bar, D);
    output  F;
    input       A, B, B_bar, C, C_bar, D;
    wire     w2 = !w1;
    wire     w3 = !B_bar;
    wire     w4, w5, w5_bar, w6, w6_bar;
    assign  w1 = !(C && D);
    assign  w4 = w2 || w3;
    assign  w5 = !(w4 && A);
    assign  w5_bar = !w5;
    assign  w6 = !(B && C_bar);
    assign  w6_bar = !w6;
```

```
    assign  F = w5_bar || w6_bar;
  endmodule
```

(c)

```
  module Fig_3_21a_CA (F, A, A_bar, B, B_bar, C, D_bar);
    output  F;
    input        A, A_bar, B, B_bar, C, D_bar;
    wire    w1, w2, w3, w4;
    assign  w1 = A && B_bar;
    assign  w2 = A_bar && B;
    assign  w3 = w1 || w2);
    assign  w4 = C || D_bar;
    assign  F = w3 || w4;
  endmodule
```

(d)

```
  module Fig_3_21b_CA (F, A, A_bar, B, B_bar, C_bar, D);
    output  F;
    input        A, A_bar, B, B_bar, C_bar, D;
    wire    w1, w2, w1_bar, w2_bar, w3, w4, w5, w6, F_bar;
    assign  w1 = !(A && B_bar);
    assign  w2 = !(A_bar && B);
    assign  w1_bar = !w1;
    assign  w2_bar = !w2;
    assign  w3 = w1_bar || w2_bar;
    assign  w4 = !C_bar;
    assign  w5 = !D;
    assign  w6 = w4 || w5;
    assign  F_bar = !(w3 && w6);
    assign  F = !F_bar;
  endmodule
```

(e)

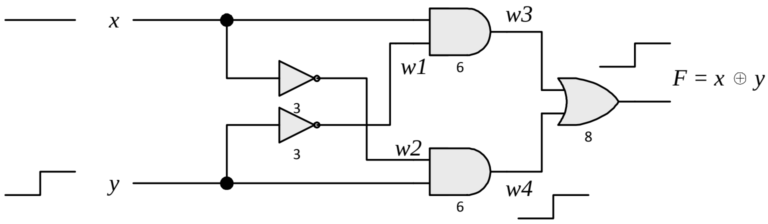
```
  module Fig_3_24_CA (F, A, B, C, D, E_bar);
    output  F;
    input        A, B, C, D, E_bar;
    wire    w1, w2, w1_bar, w2_bar, w3_bar;
    assign  w1 = !(A || B);
    assign  w1_bar = !w1;
    assign  w2 = !(C || D);
    assign  w2_bar = !w2;
    assign  w3 = !E_bar;
    assign  F = w1_bar && w2_bar && w3;
  endmodule
```

(f)

```
  module Fig_3_25_CA (F, A, A_bar, B, B_bar, C, D_bar);
    output  F;
    input        A, A_bar, B, B_bar, C, D_bar
    wire    w1, w2, w3, w4, w5, w6, w7, w8, w9, w10;
    assign  w1 = !A _bar;
    assign  w2 = !B;
    assign  w3 = w1 && w2;
    assign  w4 = !A;
    assign  w5 = !B_bar;
    assign  w6 = w4 && w5;
    assign  w7 = !(C || D_bar);
    assign  w8 = !(w3 || w6);
    assign  w9 = !w8;
    assign  w10 =    !w7;
    assign  F = w9 && w10;
  endmodule
```

3.33 **(a)**

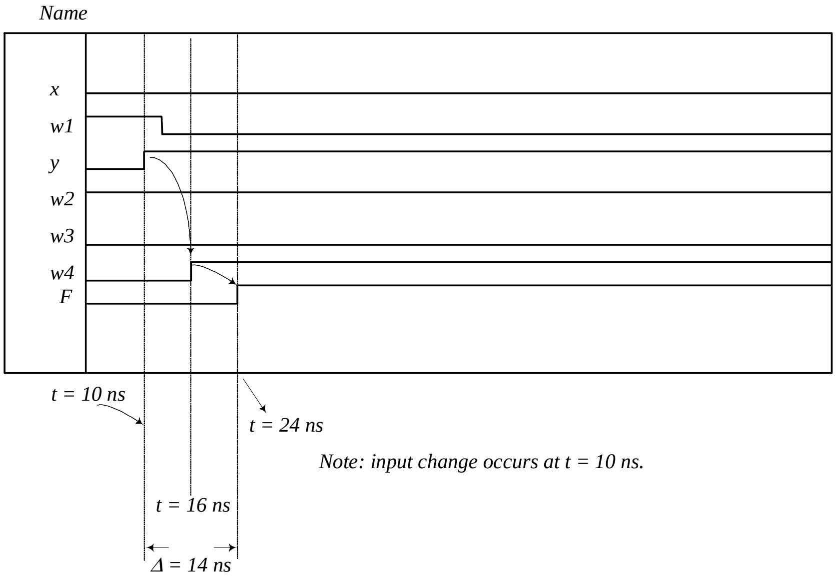
Initially, with xy = 00, $w1$ = w2 = 1, $w3$ = $w4$ = 0 and F = 0. $w1$ should change to 0 3ns after xy changes to 01. $w4$ should change to 1 6ns after xy changes to 01. F should change from 0 to 1 8ns after $w4$ changes from 0 to 1, i.e., 14 ns after xy changes from 00 to 01.



(b)

```
`timescale 1ns/1ps

module Prob_3_33 (F, x, y);
wire w1, w2, w3, w4;

and #6 (w3, x, w1);
not #3 (w1, x);
and #6 (w4, y, w1);
not #3 (w2, y);
or   #8 (F, w3, w4);

endmodule

module t_Prob_3_33 ();
  reg x, y;
  wire F;

  Prob_3_33 M0 (F, x, y);

  initial #200 $finish;
  initial fork
    x = 0;
    y = 0;
    #20 y = 1;
  join
endmodule
```

(c) To simulate the circuit, it is assumed that the inputs xy = 00 have been applied sufficiently long for the circuit to be stable before xy = 01 is applied. The testbench sets xy = 00 at t = 0 ns, and xy = 1 at t = 10 ns. The simulator assumes that xy = 00 has been applied long enough for the circuit to be in a stable state at t = 0 ns, and shows F = 0 as the value of the output at t = 0. For illustration, the waveforms show the response to xy = 01 applied at t = 10 ns.



3.34

```
module Prob_3_34 (Out_1, Out_2, Out_3, A, B, C, D);
  output  Out_1, Out_2, Out_3;
  input       A, B, C, D;
  wire     A_bar, B_bar, C_bar, D_bar;
  assign  A_bar = !A;
  assign  B_Bar = !B;
  assign  C_bar = !C;
  assign  D_bar = !D;
  assign  Out_1 = (A + B_bar) && C_bar && ( C || D);
  assign  Out_2 = ( (C_bar && D) || (B && C && D) || (C && D_bar) ) && (A_bar || B);
  assign  Out_3 = (((A && B) || C) && D) || (B_bar && C);
endmodule
```

3.35

```
module Exmpl-3(A, B, C, D, F)        // Line 1
  inputs A, B, C, Output D, F,       // Line 2
  output B                           // Line 3
  and g1(A, B, B);                   // Line 4
  not (D, B, A),                     // Line 5
  OR (F, B; C);                      // Line 6
endofmodule;                         // Line 7
```

Line 1: Dash not allowed character in identifier; use underscore: Exmpl_3. Terminate line with semicolon (;).

Line 2: **inputs** should be **input** (no s at the end). Change last comma (,) to semicolon (;). *Output* is declared but does not appear in the port list, and should be followed by a comma if it is intended to be in the list of inputs. If *Output* is a mispelling of **output** and is to declare output ports, *C* should be followed by a semicolon (;) and *F* should be followed by a semicolon (;).

Line 3: *B* cannot be declared both as an input (Line 2) and output (Line 3). Terminate the line with a semicolon (;).

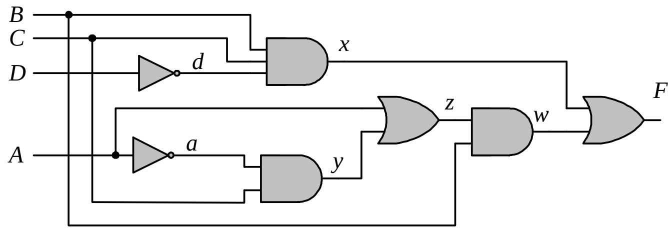
Line 4: *A* cannot be an output of the primitive if it is an input to the module

Line 5: Too many entries for the not gate (may have only a single input, and a single output). Termiante the line with a semicolon, not a comma.

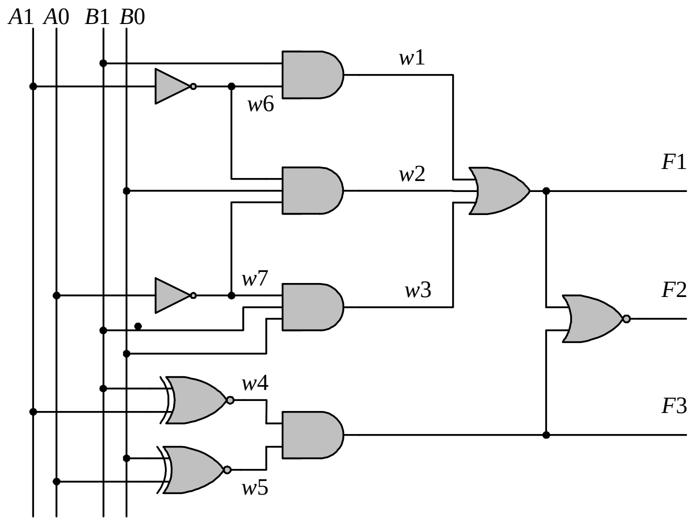
Line 6: OR must be in lowercase: change to "or". Replace semicolon by a comma (B,) in the list of ports.

Line 7: Remove semicolon (no semicolon after endmodule).

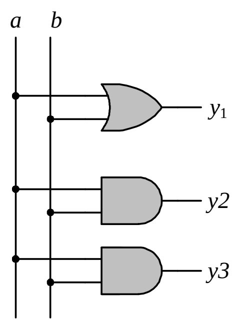
3.36 **(a)**



(b)



(c)



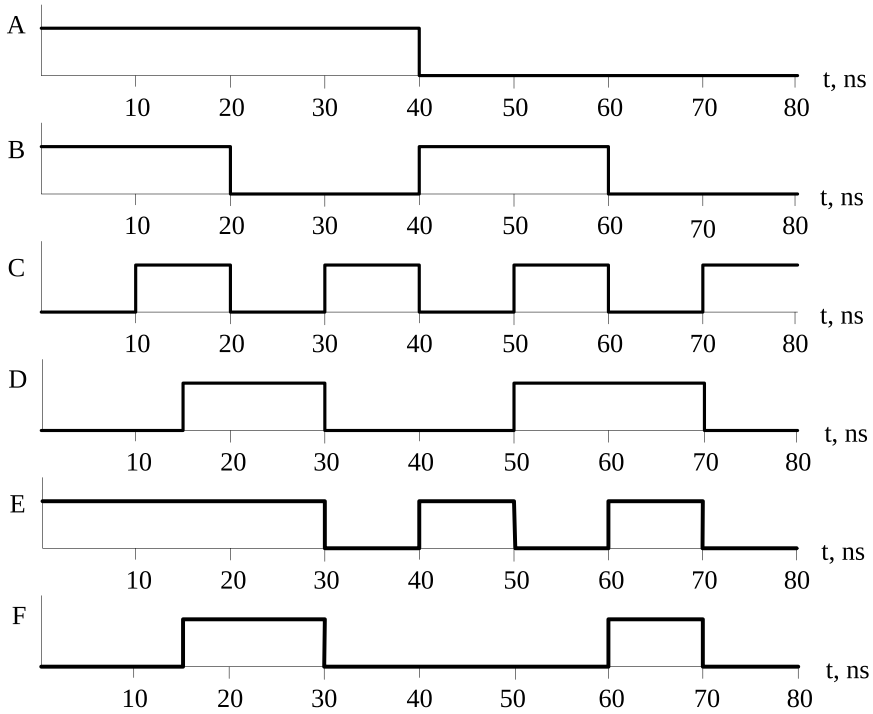
3.37

```
 UDP_Majority_4 (y, a, b, c, d);
output    y;
input     a, b, c, d;
table
// a   b   c   d   :   y
    0   0   0   0   :   0;
    0   0   0   1   :   0;
    0   0   1   0   :   0;
    0   0   1   1   :   0;
    0   1   0   0   :   0;
    0   1   0   1   :   0;
    0   1   1   0   :   0;
    0   1   1   1   :   1;

    1   0   0   0   :   0;
    1   0   0   1   :   0;
    1   0   1   0   :   0;
    1   0   1   1   :   0;
    1   1   0   0   :   0;
    1   1   0   1   :   0;
    1   1   1   0   :   1;
    1   1   1   1   :   1;
  endtable
endprimitive
```

3.38

```
module t_Circuit_with_UDP_02467;
  wire    E, F;
  reg  A, B, C, D;
  Circuit_with_UDP_02467 m0 (E, F, A, B, C, D);

  initial #100 $finish;
  initial fork
    A = 0; B = 0; C = 0; D = 0;
    #40 A = 1;
    #20 B = 1;
    #40 B = 0;
    #60 B = 1;
    #10 C = 1; #20 C = 0; #30 C = 1; #40 C = 0; #50 C = 1; #60 C = 0; #70 C = 1;
    #20 D = 1;
  join
endmodule
// Verilog model: User-defined Primitive
primitive UDP_02467 (D, A, B, C);
  output D;
  input   A, B, C;
// Truth table for D = f (A, B, C) =  . (0, 2, 4, 6, 7);
  table
//   A  B  C  :   D   // Column header comment
     0  0  0  :   1;
     0  0  1  :   0;
     0  1  0  :   1;
     0  1  1  :   0;
     1  0  0  :   1;
     1  0  1  :   0;
     1  1  0  :   1;
     1  1  1  :   1;
```

```
   endtable
endprimitive
// Verilog model: Circuit instantiation of Circuit_UDP_02467
module Circuit_with_UDP_02467 (e, f, a, b, c, d);
   output  e, f;
   input         a, b, c, d;

  UDP_02467       M0 (e, a, b, c);
   and            (f, e, d);         //Option gate instance name omitted
endmodule
```



3.39

a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

s = a'b + ab' = a ^ b
c = ab = a && b

```
module Prob_3_39 (s, c, a, b);
  input a, b;
  output s, c;

  xor (s, a, b);
  and (c, a, b);
endmodule
```