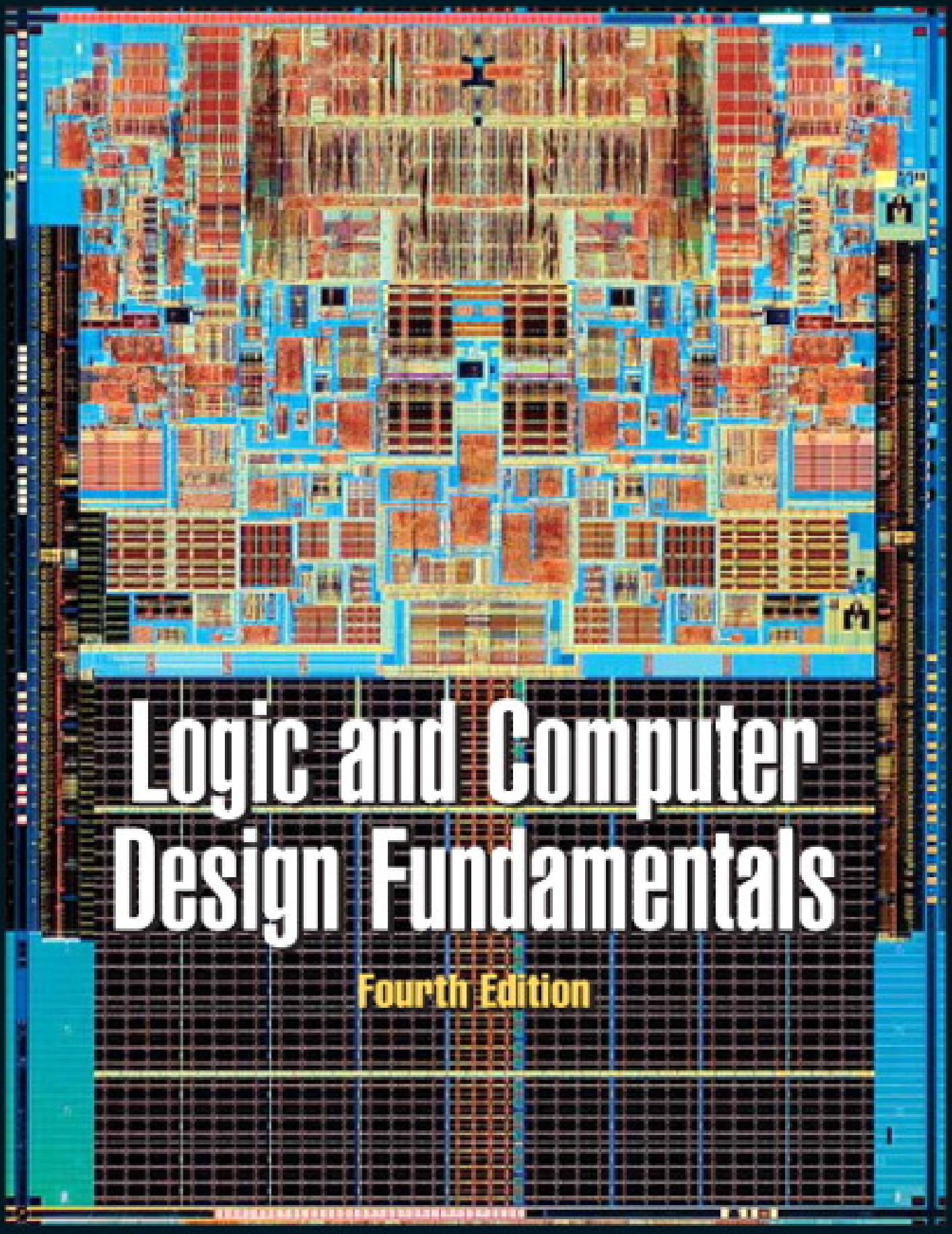


Logic and Computer Design Fundamentals

Fourth Edition

M. Morris Mano ▪ Charles R. Kime

Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 1

© 2008 Pearson Education, Inc.

1-3*

Decimal, Binary, Octal and Hexadecimal Numbers from $(16)_{10}$ to $(31)_{10}$

Dec	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Bin	1 0000	1 0001	1 0010	1 0011	1 0100	1 0101	1 0110	1 0111	1 1000	1 1001	1 1010	1 1011	1 1100	1 1101	1 1110	1 1111
Oct	20	21	22	23	24	25	26	27	30	31	32	33	34	35	36	37
Hex	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F

1-7*

$(1001101)_2 = 2^6 + 2^3 + 2^2 + 2^0 = 77$

$(1010011.101)_2 = 2^6 + 2^4 + 2^1 + 2^0 + 2^{-1} + 2^{-3} = 83.625$

$(10101110.1001)_2 = 2^7 + 2^5 + 2^3 + 2^2 + 2^1 + 2^{-1} + 2^{-4} = 174.5625$

1-9*

Decimal	Binary	Octal	Hexadecimal
369.3125	101110001.0101	561.24	171.5
189.625	10111101.101	275.5	BD.A
214.625	11010110.101	326.5	D6.A
62407.625	1111001111000111.101	171707.5	F3C7.A

1-10*

a)

8|7562 2 ↑ 16612
8|945 1
8|118 6
8|14 6
8|1 1
0

$0.45 \times 8 = 3.6 \Rightarrow 3$
$0.60 \times 8 = 4.8 \Rightarrow 4$
$0.80 \times 8 = 6.4 \Rightarrow 6$
$0.20 \times 8 = 3.2 \Rightarrow 3$ → 3463

$(7562.45)_{10} = (16612.3463)_8$

b) $(1938.257)_{10} = (792.41CB)_{16}$

c) $(175.175)_{10} = (10101111.001011)_2$

1-11*

a) $(673.6)_8 = (110\ 111\ 011.110)_2$
$= (1BB.C)_{16}$

b) $(E7C.B)_{16} = (1110\ 0111\ 1100.1011)_2$
$= (7174.54)_8$

c) $(310.2)_4 = (11\ 01\ 00.10)_2$
$= (64.4)_8$

1-16*

a) $(BEE)_r = (2699)_{10}$

$11 \times r^2 + 14 \times r^1 + 14 \times r^0 = 2699$

$11 \times r^2 + 14 \times r - 2685 = 0$

By the quadratic equation: $r = 15$ or ≈ -16.27

ANSWER: $r = 15$

b) $(365)_r = (194)_{10}$

$3 \times r^2 + 6 \times r^1 + 5 \times r^0 = 194$

$3 \times r^2 + 6 \times r - 189 = 0$

By the quadratic equation: $r = -9$ or 7

ANSWER: r = 7

1-18*

a) $(0100\ 1000\ 0110\ 0111)_{BCD} = (4867)_{10} = (1001100000011)_2$

b) $(0011\ 0111\ 1000.0111\ 0101)_{BCD} = (378.75)_{10} = (101111010.11)_2$

1-19*

$(694)_{10} = (0110\ 1001\ 0100)_{BCD}$

$(835)_{10} = (1000\ 0011\ 0101)_{BCD}$

	1 ←		
	0110	1001	0100
	+1000	+0011	+0101
	1111	1100	1001
	+0110	+0110	+0000
0001	0101	1 0010	1001

1-20*

(a)

	10^1	10^0		
	0111	1000		
Move R	011	1100	0	10^0 column > 0111
Subtract 3		-0011		
	011	1001	0	
Subtract 3		-0011		
	01	1001		
Move R	0	1100	110	10^0 column > 0111
Subtract 3		-0011		
	0	1001	110	
Move R		0100	1110	
Move R		010	01110	
Move R		01	001110	
Move R		0	1001110	Leftmost 1 in BCD number shifted out: Finished

(b)

	10^2	10^1	10^0		
	0011	1001	0111		
Move R	001	1100	1011	1	10^1 and 10^0 columns > 0111
Subtract 3		-0011	-0011		
	001	1001	1000	1	
Move R	00	1100	1100	01	10^1 and 10^0 columns > 0111
Subtract 3		-0011	-0011		
	00	1001	1001	01	
Move R	0	0100	1100	101	10^0 column > 0111
Subtract 3			-0011		
	0	0100	1001		
Move R		0010	0100	1101	
Move R		001	0010	01101	
Move R		00	1001	001101	100 column > 0111
Subtract 3			-0011		
		00	0110	001101	
Move R		0	0011	0001101	
Move R			0001	10001101	
Move R			000	110001101	Leftmost 1 in BCD number shifted out: Finished

1-25*

a) $(11111111)_2$

b) $(0010\ 0101\ 0101)_{BCD}$

c) $011\ 0010 \quad 011\ 0101 \quad 011\ 0101_{ASCII}$

d) $0011\ 0010 \quad 1011\ 0101 \quad 1011\ 0101_{ASCII\ with\ Odd\ Parity}$

Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 2

© 2008 Pearson Education, Inc.

2-1.*

a) $\overline{XYZ} = \bar{X} + \bar{Y} + \bar{Z}$

Verification of DeMorgan's Theorem

X	Y	Z	XYZ	$\overline{XYZ}$	$\bar{X}+\bar{Y}+\bar{Z}$
0	0	0	0	1	1
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	0	1	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	0	1	1
1	1	1	1	0	0

b) $X + YZ = (X + Y) \cdot (X + Z)$

The Second Distributive Law

X	Y	Z	YZ	$X+YZ$	$X+Y$	$X+Z$	$(X+Y)(X+Z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

c) $\bar{X}Y + \bar{Y}Z + X\bar{Z} = X\bar{Y} + Y\bar{Z} + \bar{X}Z$

X	Y	Z	$\bar{X}Y$	$\bar{Y}Z$	$X\bar{Z}$	$\bar{X}Y+\bar{Y}Z+X\bar{Z}$	$X\bar{Y}$	$Y\bar{Z}$	$\bar{X}Z$	$X\bar{Y}+Y\bar{Z}+\bar{X}Z$
0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0	0	1	1
0	1	0	1	0	0	1	0	1	0	1
0	1	1	1	0	0	1	0	0	1	1
1	0	0	0	0	1	1	1	0	0	1
1	0	1	0	1	0	1	1	0	0	1
1	1	0	0	0	1	1	0	1	0	1
1	1	1	0	0	0	0	0	0	0	0

2-2.*

a) $\bar{X}\bar{Y} + \bar{X}Y + XY = \bar{X} + Y$

$= (\bar{X}Y + \bar{X}\bar{Y}) + (\bar{X}Y + XY)$

$= \bar{X}(Y + \bar{Y}) + Y(X + \bar{X})$

$= \bar{X} + Y$

b) $\bar{A}B + \bar{B}\bar{C} + AB + \bar{B}C = 1$

$= (\bar{A}B + AB) + (\bar{B}\bar{C} + \bar{B}C)$

$= B(A + \bar{A}) + \bar{B}(C + \bar{C})$

$B + \overline{B} = 1$

c) $Y + \overline{X}Z + X\overline{Y} = X + Y + Z$

$= Y + X\overline{Y} + \overline{X}Z$
$= (Y + X)(Y + \overline{Y}) + \overline{X}Z$
$= Y + X + \overline{X}Z$
$= Y + (X + \overline{X})(X + Z)$
$= X + Y + Z$

d) $\overline{X}\overline{Y} + \overline{Y}Z + XZ + XY + Y\overline{Z} = \overline{X}\overline{Y} + XZ + Y\overline{Z}$

$= \overline{X}\overline{Y} + \overline{Y}Z(X + \overline{X}) + XZ + XY + Y\overline{Z}$
$= \overline{X}\overline{Y} + X\overline{Y}Z + \overline{X}\overline{Y}Z + XZ + XY + Y\overline{Z}$
$= \overline{X}\overline{Y}(1 + Z) + X\overline{Y}Z + XZ + XY + Y\overline{Z}$
$= \overline{X}\overline{Y} + XZ(1 + \overline{Y}) + XY + Y\overline{Z}$
$= \overline{X}\overline{Y} + XZ + XY(Z + \overline{Z}) + Y\overline{Z}$
$= \overline{X}\overline{Y} + XZ + XYZ + Y\overline{Z}(1 + X)$
$= \overline{X}\overline{Y} + XZ(1 + Y) + Y\overline{Z}$
$= \overline{X}\overline{Y} + XZ + Y\overline{Z}$

2-7.*

a) $\overline{X}\overline{Y} + XYZ + \overline{X}Y = \overline{X} + XYZ = (\overline{X} + XY)(\overline{X} + Z) = (\overline{X} + X)(\overline{X} + Y)(\overline{X} + Z)$
$= (\overline{X} + Y)(\overline{X} + Z) = \overline{X} + YZ$

b) $X + Y(Z + \overline{X + Z}) = X + Y(Z + \overline{X}\overline{Z}) = X + Y(Z + \overline{X})(Z + \overline{Z}) = X + YZ + \overline{X}Y$
$= (X + \overline{X})(X + Y) + YZ = X + Y + YZ = X + Y$

c) $\overline{W}X(\overline{Z} + \overline{Y}Z) + X(W + \overline{W}YZ) = \overline{W}X\overline{Z} + \overline{W}X\overline{Y}Z + WX + \overline{W}XYZ$
$= \overline{W}X\overline{Z} + \overline{W}XZ + WX = \overline{W}X + WX = X$

d) $(AB + \overline{A}\overline{B})(\overline{C}\overline{D} + CD) + \overline{AC} = AB\overline{C}\overline{D} + ABCD + \overline{A}\overline{B}CD + \overline{A}\overline{B}\overline{C}\overline{D} + \overline{A} + \overline{C}$
$= ABCD + \overline{A} + \overline{C} = \overline{A} + \overline{C} + A(BCD) = \overline{A} + \overline{C} + C(BD) = \overline{A} + \overline{C} + BD$

2-9.*

a) $\overline{F} = (\overline{A} + B)(A + \overline{B})$

b) $\overline{F} = ((V + \overline{W})\overline{X} + \overline{Y})Z$

c) $\overline{F} = [\overline{W} + \overline{X} + (Y + \overline{Z})(\overline{Y} + Z)][W + X + Y\overline{Z} + \overline{Y}Z]$

d) $\overline{F} = \overline{A}B\overline{C} + (A + B)\overline{C} + \overline{A}(B + C)$

2-10.*

Truth Tables a, b, c

X	Y	Z	a
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

A	B	C	b
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

W	X	Y	Z	c
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0

Truth Tables a, b, c

1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

a) Sum of Minterms: $\overline{X}YZ + X\overline{Y}Z + XY\overline{Z} + XYZ$

Product of Maxterms: $(X + Y + Z)(X + Y + \overline{Z})(X + \overline{Y} + Z)(\overline{X} + Y + Z)$

b) Sum of Minterms: $\overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,\overline{B}C + \overline{A}BC + ABC$

Product of Maxterms: $(A + \overline{B} + C)(\overline{A} + B + C)(\overline{A} + B + \overline{C})(\overline{A} + \overline{B} + C)$

c) Sum of Minterms: $\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + \overline{W}X\overline{Y}\,\overline{Z} + W\overline{X}\,\overline{Y}\,\overline{Z} + WX\overline{Y}\,\overline{Z} + WX\overline{Y}Z + WXY\overline{Z} + WXYZ$

Product of Maxterms: $(W + X + Y + Z)(W + X + Y + \overline{Z})(W + X + \overline{Y} + \overline{Z})(W + \overline{X} + Y + Z)(W + \overline{X} + Y + \overline{Z})(W + \overline{X} + \overline{Y} + \overline{Z})(\overline{W} + X + Y + Z)(\overline{W} + X + Y + \overline{Z})(\overline{W} + X + \overline{Y} + \overline{Z})$

2-12.*

a) $(AB + C)(B + \overline{C}D) = AB + AB\overline{C}D + BC = AB + BC$ s.o.p.

$= B(A + C)$ p.o.s.

b) $\overline{X} + X(X + \overline{Y})(Y + \overline{Z}) = (\overline{X} + X)(\overline{X} + (X + \overline{Y})(Y + \overline{Z}))$

$= (\overline{X} + X + \overline{Y})(\overline{X} + Y + \overline{Z})$ p.o.s.

$= (1 + \overline{Y})(\overline{X} + Y + \overline{Z}) = \overline{X} + Y + \overline{Z}$ s.o.p.

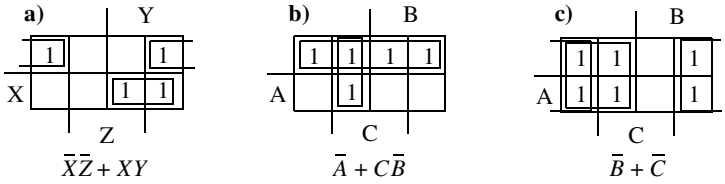
c) $(A + B\overline{C} + CD)(\overline{B} + EF) = (A + B + C)(A + B + D)(A + \overline{C} + D)(\overline{B} + EF)$

$= (A + B + C)(A + B + D)(A + \overline{C} + D)(\overline{B} + E)(\overline{B} + F)$ p.o.s.

$(A + B\overline{C} + CD)(\overline{B} + EF) = A(\overline{B} + EF) + B\overline{C}(\overline{B} + EF) + CD(\overline{B} + EF)$

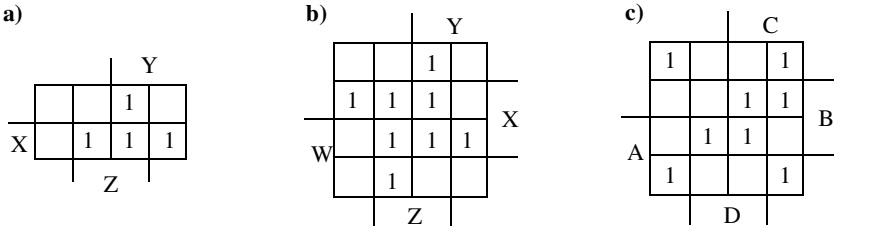
$= A\overline{B} + AEF + B\overline{C}EF + \overline{B}CD + CDEF$ s.o.p.

2-15.*



$\overline{X}\overline{Z} + XY$ $\quad$ $\overline{A} + C\overline{B}$ $\quad$ $\overline{B} + \overline{C}$

2-18.*



$\Sigma m(3, 5, 6, 7)$ $\quad$ $\Sigma m(3, 4, 5, 7, 9, 13, 14, 15)$ $\quad$ $\Sigma m(0, 2, 6, 7, 8, 10, 13, 15)$

2-19.*

a) $Prime = XZ, WX, \overline{X}\overline{Z}, W\overline{Z}$
$Essential = XZ, \overline{X}\overline{Z}$

b) $Prime = CD, AC, \overline{B}\overline{D}, \overline{A}BD, \overline{B}C$
$Essential = AC, \overline{B}\overline{D}, \overline{A}BD$

c) $Prime = AB, AC, AD, B\overline{C}, \overline{B}D, \overline{C}D$
$Essential = AC, B\overline{C}, \overline{B}D$

2-22.*

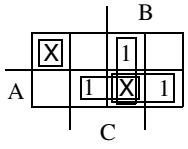
a) s.o.p. $CD + A\overline{C} + \overline{B}D$
p.o.s. $(\overline{C} + D)(A + D)(A + \overline{B} + C)$

b) s.o.p. $\overline{A}\overline{C} + \overline{B}\overline{D} + A\overline{D}$
p.o.s. $(\overline{C} + \overline{D})(\overline{A} + \overline{D})(A + \overline{B} + \overline{C})$

c) s.o.p. $\overline{B}\overline{D} + \overline{A}BD + (\overline{A}BC \text{ or } \overline{A}C\overline{D})$
p.o.s. $(\overline{A} + \overline{B})(B + \overline{D})(\overline{B} + C + D)$

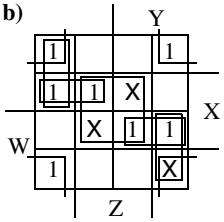
2-25.*

a)



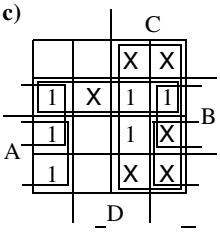
$Primes = AB, AC, BC, \overline{A}\overline{B}\overline{C}$
$Essential = AB, AC, BC$
$F = AB + AC + BC$

b)



$Primes = \overline{X}\overline{Z}, XZ, \overline{W}X\overline{Y}, WXY, \overline{W}\overline{Y}\overline{Z}, WY\overline{Z}$
$Essential = \overline{X}\overline{Z}$
$F = \overline{X}\overline{Z} + \overline{W}X\overline{Y} + WXY$

c)



$Primes = \overline{A}B, C, A\overline{D}, B\overline{D}$
$Essential = C, A\overline{D}$
$F = C + A\overline{D} + (B\overline{D} \text{ or } \overline{A}B)$

2-32.*

$$X \oplus Y = X\overline{Y} + \overline{X}Y$$

$$\text{Dual } (X \oplus Y) = \text{Dual } (X\overline{Y} + \overline{X}Y)$$

$$= (X + \overline{Y})(\overline{X} + Y)$$

$$= \overline{\overline{X}Y + X\overline{Y}}$$

$$= \overline{X\overline{Y} + \overline{X}Y}$$

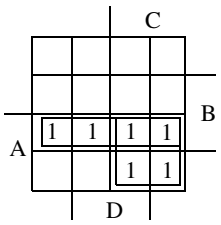
$$= \overline{X \oplus Y}$$

Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 3

© 2008 Pearson Education, Inc.

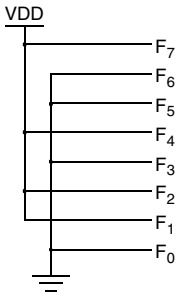
3-2.*



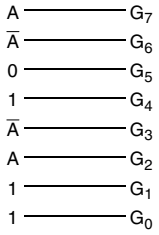
$F = AB + AC$

3-24.*

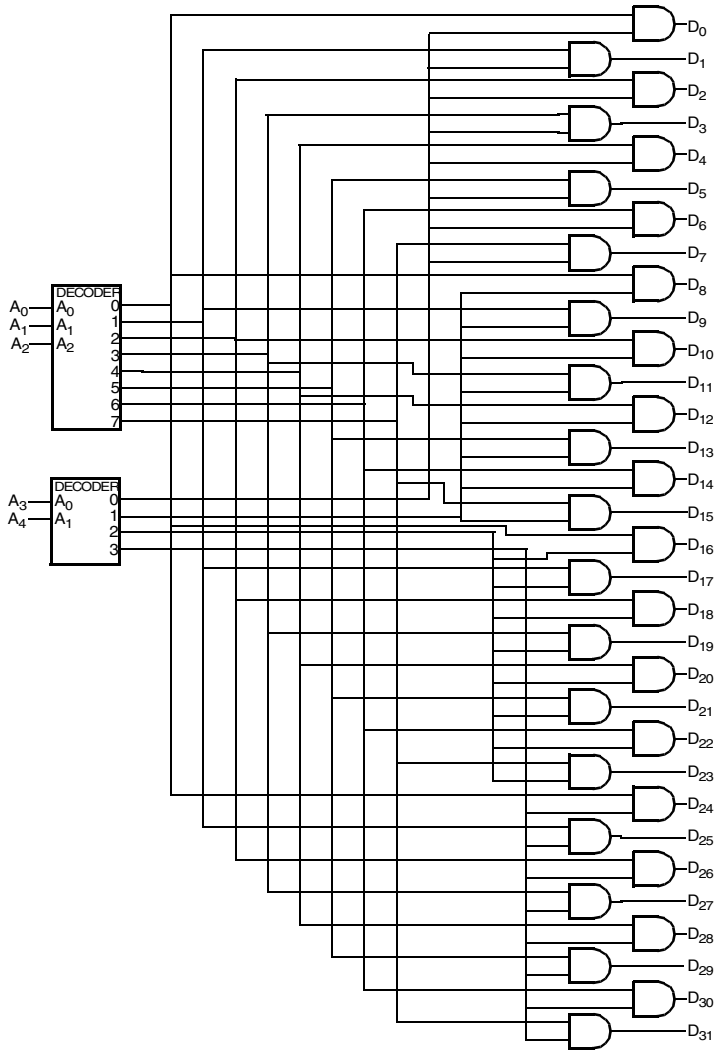
a)



b)



3-30.*



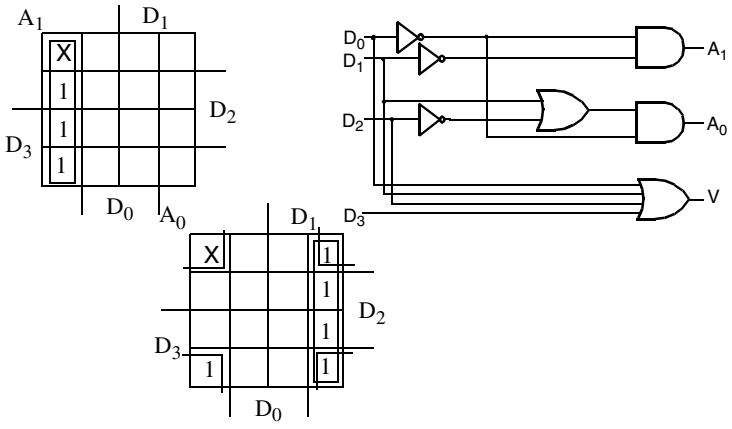
3-35.*

D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	X	X	0
X	X	X	1	0	0	1
X	X	1	0	0	1	1
X	1	0	0	1	0	1
1	0	0	0	1	1	1

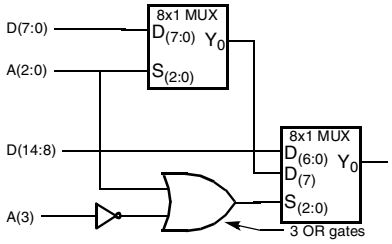
$V = D_0 + D_1 + D_2 + D_3$

$A_0 = \overline{D_0}(D_1 + \overline{D}_2)$

$A_1 = \overline{D_0}\,\overline{D_1}$



3-42.*



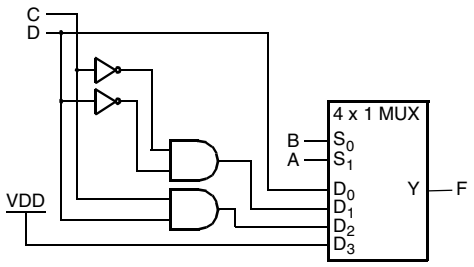
3-43.*

A_1	A_0	E	D_0	D_1	D_2	D_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

Consider E as the data input and A0, A1 as the select lines. For a given combination on (A1, A0), the value of E is distributed to the corresponding D output. For example for (A1, A0) = (10), the value of E appears on D2, while all other outputs have value 0.

3-47.*

A	B	C	D	F	
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	0	F=D
0	0	1	1	1	
0	1	0	0	1	
0	1	0	1	0	
0	1	1	0	0	$F=\overline{C}\,\overline{D}$
0	1	1	1	0	
1	0	0	0	0	
1	0	0	1	0	
1	0	1	0	0	F=C D
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	1	
1	1	1	0	1	F=1
1	1	1	1	1	



Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 4

© 2008 Pearson Education, Inc.

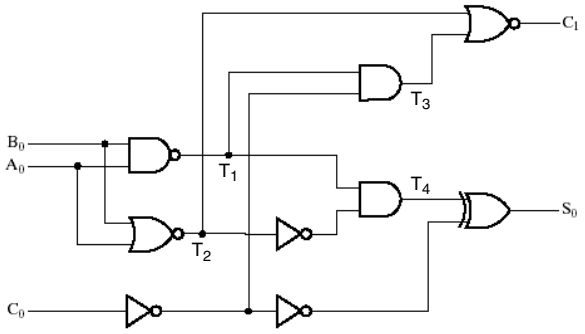
4-2.*

$$C_1 = \overline{T_3 + T_2} = \overline{T_1\bar{C}_0 + T_2} = \overline{\overline{A_0B_0}\,\overline{C_0} + \overline{A_0 + B_0}} = \overline{(\bar{A}_0 + \bar{B}_0)\bar{C}_0 + \bar{A}_0\bar{B}_0} = (A_0B_0 + C_0)(A_0 + B_0)$$

$$C_1 = A_0B_0 + A_0C_0 + B_0C_0$$

$$S_0 = C_0 \oplus T_4 = C_0 \oplus T_1\bar{T}_2 = C_0 \oplus \overline{A_0B_0}(A_0 + B_0) = C_0 \oplus (\bar{A}_0 + \bar{B}_0)(A_0 + B_0) = C_0 \oplus A_0\bar{B}_0 + \bar{A}_0B_0$$

$$S_0 = A_0 \oplus B_0 \oplus C_0$$



4-3.*

Unsigned	1001 1100	1001 1101	1010 1000	0000 0000	1000 0000
1's Complement	0110 0011	0110 0010	0101 0111	1111 1111	0111 1111
2's Complement	0110 0100	0110 0011	0101 1000	0000 0000	1000 0000

4-6.*

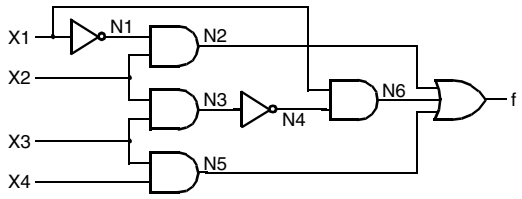
+36 = 0100100
- 24 = 1101000
- 35 = 1011101

36		0100100
+(–24)	+	1101000
		10001100
= 12	=	0001100
–35		1011101
- (–24)	+	0011000
= –11	=	1110101

4-16.*

	S	A	B	C_4	S_3	S_2	S_1	S_0
a)	0	0111	0111	0	1	1	1	0
b)	1	0100	0111	0	1	1	0	1
c)	1	1101	1010	1	0	0	1	1
d)	0	0111	1010	1	0	0	0	1
e)	1	0001	1000	0	1	0	0	1

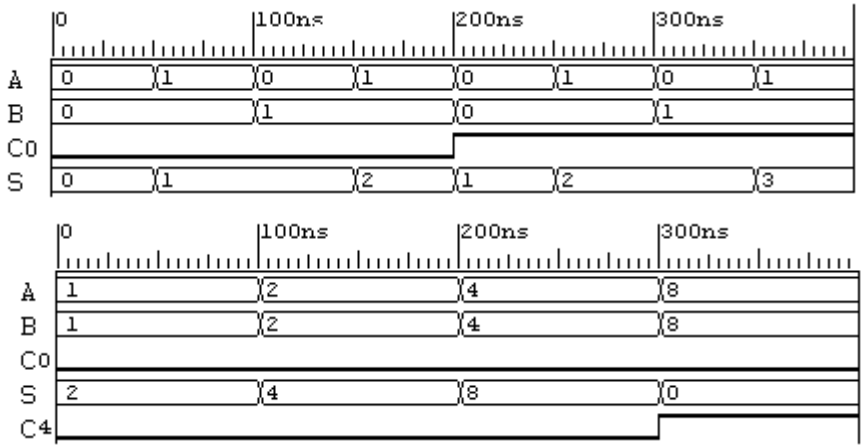
4-20.*



4-24.*

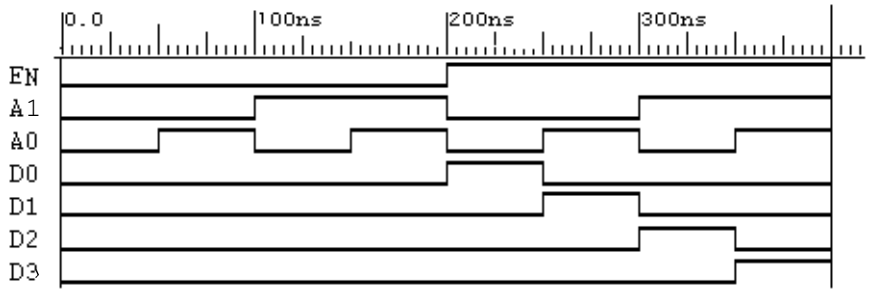
```
begin
    F <= (X and Z) or ((not Y) and Z);
end;
```

4-29.*

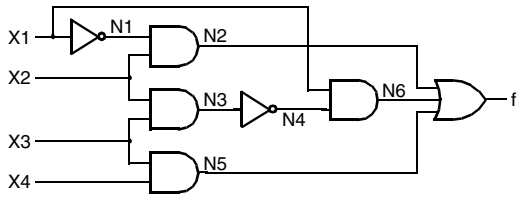


The solution given is very thorough since it checks each of the carry connections between adjacent cells transferring 0 and 1. In contrast a test applying C0 = 1 and A = 15 with B = 0 would allow a whole variety of incorrect connections between cells that would not be detected.

4-31.*(Errata: Replace “E” with “EN”.)



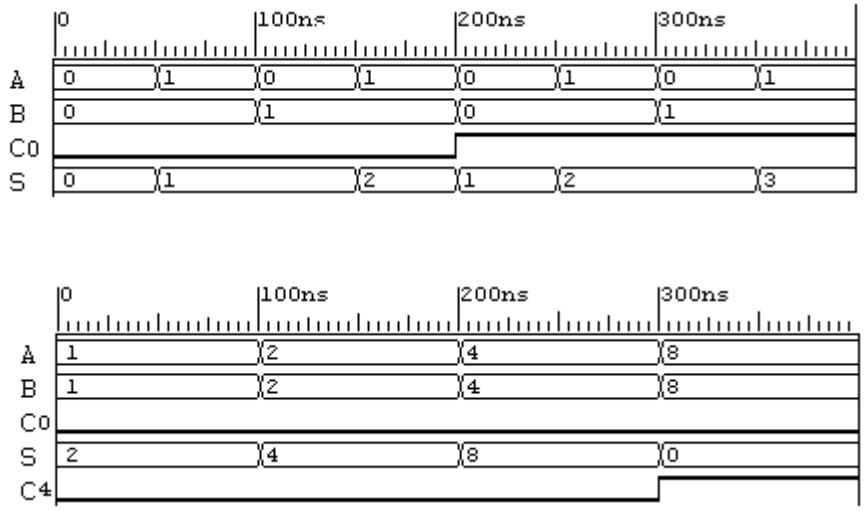
4-34.*



4-38.*

```
module circuit_4_53(X, Y, Z, F);
  input X, Y, Z;
  output F;
  assign F = (X & Z) | (Z & ~Y);
endmodule
```

4-43.*



The solution given is very thorough since it checks each of the carry connections between adjacent cells transferring 0 and 1. In contrast a test applying C0 = 1 and A = 15 with B = 0 would allow a whole variety of incorrect connections between cells that would not be detected.

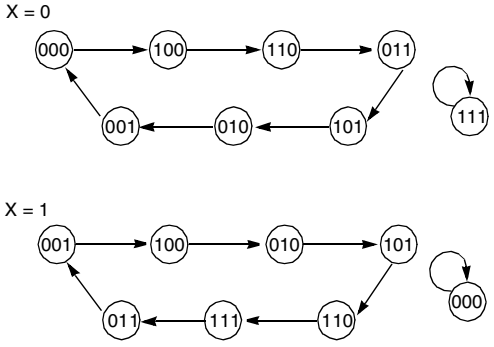
Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 5

© 2008 Pearson Education, Inc.

5-7.*

Present state			Input	Next state		
A	*B*	*C*	*X*	*A*	*B*	*C*
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	1	1	0
1	1	0	0	0	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	0	1	1



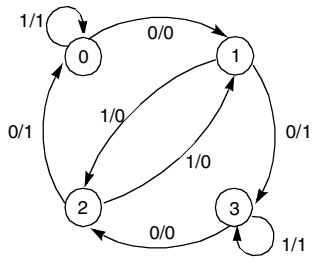
State diagram is the combination of the above two diagrams.

5-11.*

$S_A = B \qquad S_B = \overline{X \oplus A}$

$R_A = \overline{B} \qquad R_B = X \oplus A$

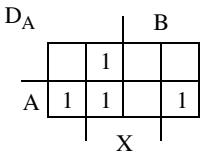
Present state		Input	Next state		Output
A	*B*	*X*	*A*	*B*	*Y*
0	0	0	0	1	0
0	0	1	0	0	1
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	1	1	1



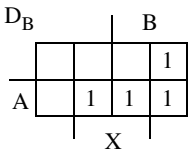
Format: X/Y

5-13.*

Present state		Input	Next state	
A	*B*	*X*	*A*	*B*
0	0	0	0	0
0	0	1	1	0
0	1	0	0	1
0	1	1	0	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	1



$D_A = A\overline{X} + \overline{B}X$

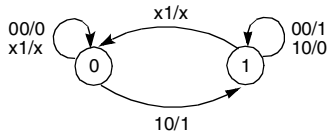


$D_B = AX + B\overline{X}$

Logic diagram not given.

5-18.*

Format: XY/Z (x = unspecified)



Present state	Inputs		Next state	Output
Q(t)	*X*	*Y*	*Q(t+1)*	*Z*
0	0	0	0	0
0	0	1	0	X
0	1	0	1	1
0	1	1	0	X
1	0	0	1	1
1	0	1	0	X
1	1	0	1	0
1	1	1	0	X

5-26.*

To use a one-hot assignment, the two flip-flops A and B need to be replaced with four flip-flops Y4, Y3, Y2. Y1.

No Reset State Specified.

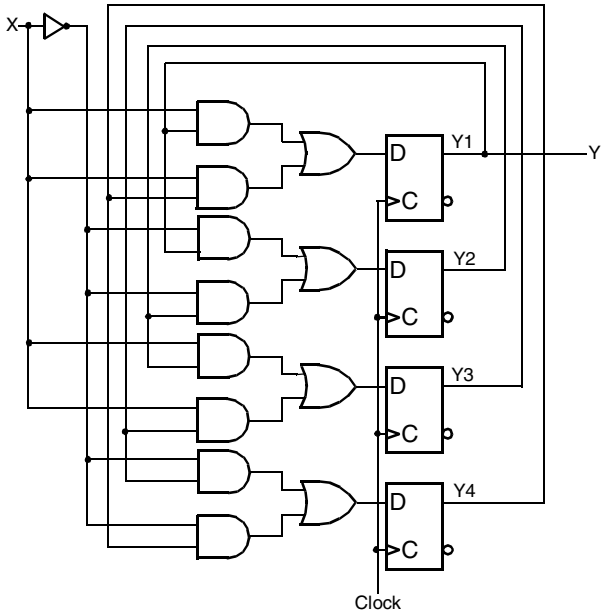
Present State		Input	Next State		Output
A B	*Y4 Y3 Y2 Y1*	*X*	*A' B"*	*Y4'Y3'Y2'Y1*	*Z*
0 0	0 0 0 1	0	0 1	0 0 1 0	1
0 0	0 0 0 1	1	0 0	0 0 0 1	1
0 1	0 0 1 0	0	0 1	0 0 1 0	0
0 1	0 0 1 0	1	1 0	0 1 0 0	0
1 0	0 1 0 0	0	1 1	1 0 0 0	0
1 0	0 1 0 0	1	1 0	0 1 0 0	0
1 1	1 0 0 0	0	1 1	1 0 0 0	0
1 1	1 0 0 0	1	0 0	0 0 0 1	0

$D1 = Y1' = X \cdot Y1 + X \cdot Y4$

$D2 = Y2' = \overline{X} \cdot Y1 + \overline{X} \cdot Y2$

$D3 = Y3' = X \cdot Y2 + X \cdot Y3$

$D4 = Y4' = \overline{X} \cdot Y3 + \overline{X} \cdot Y4$

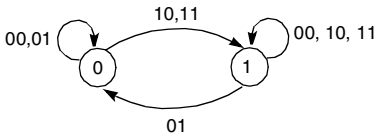


5-27.*

a)

S	R	Q	
0	0	Q	No Change
0	1	0	Reset
1	0	1	Set
1	1	1	Set

b)

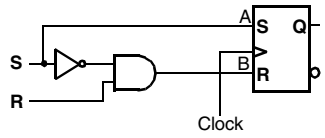


c)

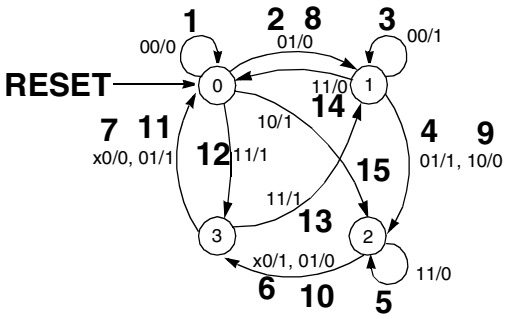
Present state	Input		Next state		
Q	*S*	*R*	*Q(t+1)*	*A*	*B*
0	0	0	0	0	x
0	0	1	0	0	x
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	1	x	0
1	0	1	0	0	1
1	1	0	1	x	0
1	1	1	1	x	0

A = S

B = $\overline{S}$R



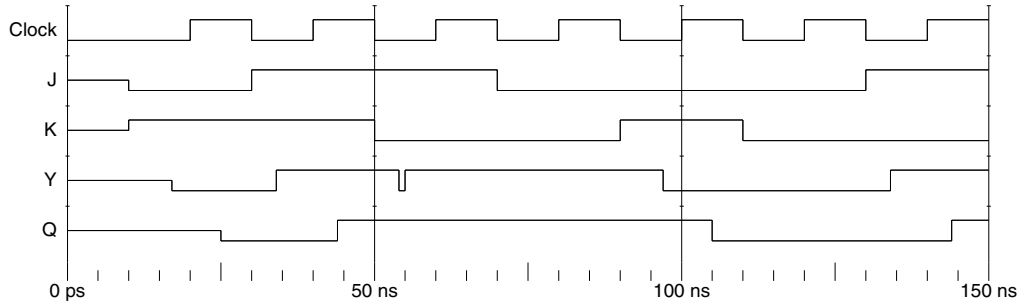
*5-31.



Format: XY/Z

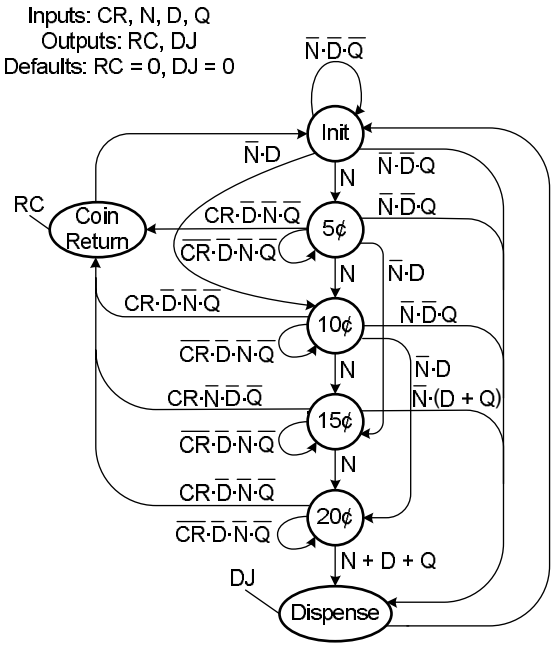
Reset, 00, 01, 00, 01, 11, x0, x0, 01, 10, 01, 01, 11, 11, 11, 10.

5-33.*



This simulation was performed without initializing the state of the latches of the flip-flop beforehand. Each gate in the flip-flop implementation has a delay of 1 ns. The interaction of these delays with the input change times produced a narrow pulse in Y at about 55 ns. In this case, the pulse is not harmful since it dies out well before the positive clock edge occurs. Nevertheless, a thorough examination of such a pulse to be sure that it does not represent a design error or important timing problem is critical.

5-37.*



5-40.*

```
library IEEE;
use IEEE.std_logic_1164.all;

entity mux_4to1 is
     port (
          S: in STD_LOGIC_VECTOR (1 downto 0);
          D: in STD_LOGIC_VECTOR (3 downto 0);
          Y: out STD_LOGIC
     );
end mux_4to1;

-- (continued in the next column)
```

```
architecture mux_4to1_arch of mux_4to1 is
begin

process (S, D)
    begin
    case S is
         when "00" => Y <= D(0);
         when "01" => Y <= D(1);
         when "10" => Y <= D(2);
         when "11" => Y <= D(3);
         when others => null;
    end case;

end process;
end mux_4to1_arch;
```

5-45.*

```
library IEEE;
use IEEE.std_logic_1164.all;
entity jkff is
   port (
      J,K,CLK: in STD_LOGIC;
      Q: out STD_LOGIC
   );
end jkff;

architecture jkff_arch of jkff is
signal q_out: std_logic;
begin

state_register: process (CLK)
begin
  if CLK'event and CLK='0' then  --CLK falling edge

-- (continued in the next column)
```

```
    case J is
        when '0' =>
            if K = '1' then
                q_out <= '0';
            end if;
        when '1' =>
            if K = '0' then
                q_out <= '1';
            else
                q_out <= not q_out;
            end if;
        when others => null;
    end case;
  end if;
end process;

Q <= q_out;
end jkff_arch;
```

5-49.*

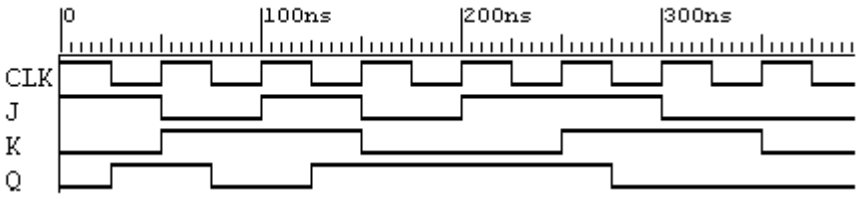
```
module problem_6_39 (S, D, Y) ;

input [1:0] S ;
input [3:0] D ;
output Y;
reg Y ;

// (continued in the next column)
```

```
always @(S or D)
     begin
          if (S == 2'b00) Y <= D[0];
          else if (S == 2'b01) Y <= D[1];
          else if (S == 2'b10) Y <= D[2];
          else Y <= D[3];

     end
endmodule
```

5-53.*

```
module JK_FF (J, K, CLK, Q) ;

input J, K, CLK ;
output Q;
reg Q;
```

```
always @(negedge CLK)
    case (J)
        0'b0: Q <= K ? 0: Q;
        1'b1: Q <= K ? ~Q: 1;
    endcase
endmodule
```



Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 6

© 2008 Pearson Education, Inc.

6-1.*

a) $F = (\overline{A} + B)\ C\ D$

b) $G = (A + \overline{B})\ (\overline{C} + D)$

6-4*

The longest path is from input C or $\overline{D}$.

0.073 ns + 0.073 ns + 0.048 ns + 0.073 ns = 0.267 ns

6-10.*

a) The longest direct path delay is from input X through the two XOR gates to the output Y.

$$t_{delay} = t_{pdXOR} + t_{pdXOR} = 0.20 + 0.20 = 0.40 \text{ ns}$$

b) The longest path from an external input to a positive clock edge is from input X through the XOR gate and the inverter to the B Flip-flop.

$$t_{delay} = t_{pdXOR} + t_{pd\ INV} + t_{sFF} = 0.20 + 0.05 + 0.1 = 0.35 \text{ ns}$$

c) The longest path delay from the positive clock edge is from Flip-flop A through the two XOR gates to the output Y.

$t_{delay} = t_{pdFF} + 2\ t_{pdXOR} = 0.40 + 2(0.20) = 0.80$ ns

d) The longest path delay from positive clock edge to positive clock edge is from clock on Flip-flop A through the XOR gate and inverter to clock on Flip-flop B.

$$t_{delay\text{-clock edge to clock edge}} = t_{pdFF} + t_{pdXOR} + t_{pdINV} + t_{sFF} = 0.40 + 0.20 + 0.05 + 0.10 = 0.75 \text{ ns}$$

e) The maximum frequency is $1/t_{delay\text{- clock edge to clock edge}}$. For this circuit, $t_{delay\text{-clock edge to clock edge}}$ is 0.75 ns, so the maximum frequency is 1/0.75 ns = 1.33 GHz.

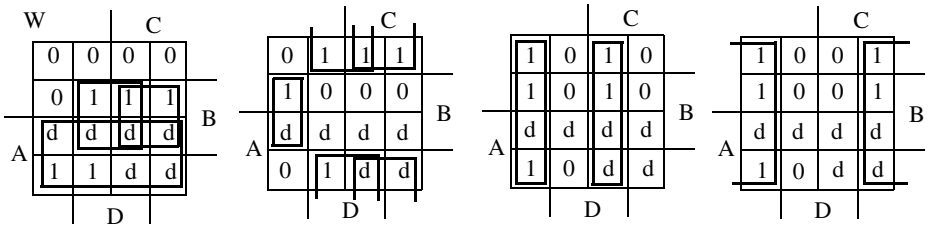
Comment: The clock frequency may need to be lower due to other delay paths that pass outside of the circuit into its environment. Calculation of this frequency cannot be performed in this case since data for paths through the environment is not provided.

6-13.* (Errata: Change "32 X 8" to "64 X 8" ROM)

IN	OUT	IN	OUT	IN	OUT	IN	OUT
000000	0000 0000	010000	0001 0110	100000	0011 0010	110000	0100 1000
000001	0000 0001	010001	0001 0111	100001	0011 0011	110001	0100 1001
000010	0000 0010	010010	0001 1000	100010	0011 0100	110010	0101 0000
000011	0000 0011	010011	0001 1001	100011	0011 0101	110011	0101 0001
000100	0000 0100	010100	0010 0000	100100	0011 0110	110100	0101 0010
000101	0000 0101	010101	0010 0001	100101	0011 0111	110101	0101 0011
000110	0000 0110	010110	0010 0010	100110	0011 1000	110110	0101 0100
000111	0000 0111	010111	0010 0011	100111	0011 1001	110111	0101 0101
001000	0000 1000	011000	0010 0100	101000	0100 0000	111000	0101 0110
001001	0000 1001	011001	0010 0101	101001	0100 0001	111001	0101 0111
001010	0001 0000	011010	0010 0110	101010	0100 0010	111010	0101 1000
001011	0001 0001	011011	0010 0111	101011	0100 0011	111011	0101 1001
001100	0001 0010	011100	0010 1000	101100	0100 0100	111100	0110 0000
001101	0001 0011	011101	0010 1001	101101	0100 0101	111101	0110 0001
001110	0001 0100	011110	0011 0000	101110	0100 0110	111110	0110 0010
001111	0001 0101	011111	0011 0001	101111	0100 0111	111111	0110 0011

6-19.*

Assume 3-input OR gates.



$W = A + BC + BD$ $X = B\overline{C}\,\overline{D} + \overline{B}C + \overline{B}D$ $Y = CD + \overline{C}\,\overline{D}$ $Z = \overline{D}$

Each of the equations above is implemented using one 3-input OR gate. Four gates are used.

Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 7

© 2008 Pearson Education, Inc.

7-2. *

1001 1001	
1100 0011	
1000 0001	AND
1101 1011	OR
0101 1010	XOR

7-4.*

sl 1001 0100 sr 0110 0101

7-5.*

Q_i remains connected to MUX data input 0. Connect D_i to MUX data input 1 instead of Mux data input 3. Connect Q_{i-1} to MUX data input 2 instead of MUX data input 1. Finally, 0 is connected to MUX data input 3.

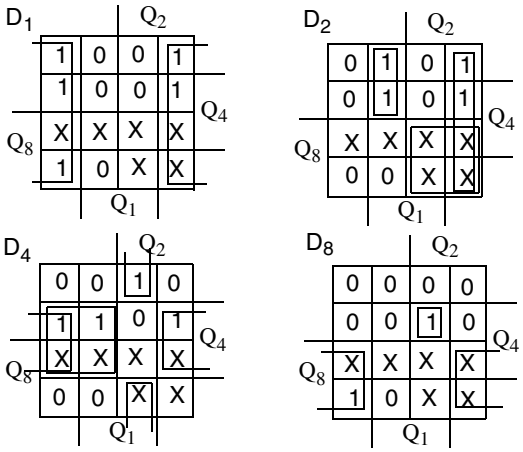
7-6.*

a) 1000, 0100, 0010, 0001, 1000. ...

b) # States = *n*

7-13.*

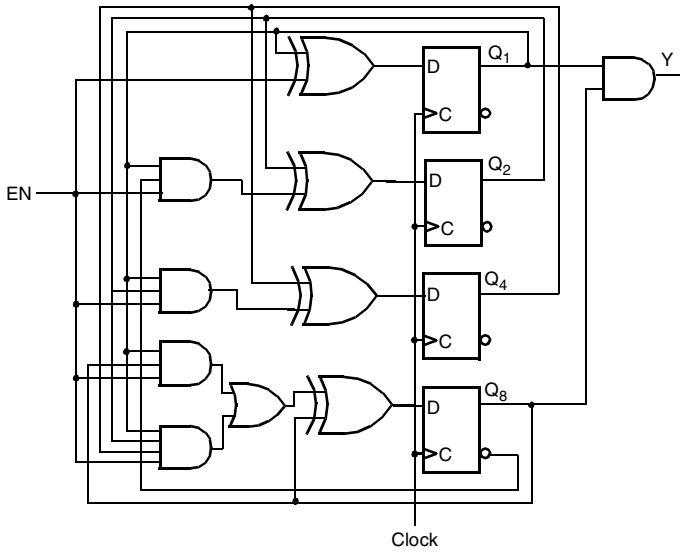
The equations given on page 364-5 can be manipulated into SOP form as follows: $D_1 = \overline{Q}_1$, $D_2 = Q_2 \oplus Q_1\overline{Q}_8 = Q_1\overline{Q}_2\overline{Q}_8 + \overline{Q}_1Q_2 + Q_2Q_8$, $D_4 = Q_4 \oplus Q_1Q_2 = Q_1Q_2\overline{Q}_4 + \overline{Q}_1Q_4 + \overline{Q}_2Q_4$, $D_8 = Q_8 \oplus (Q_1Q_8 + Q_1Q_2Q_4) = \overline{Q}_8(Q_1Q_8+Q_1Q_2Q_4) + Q_8(\overline{Q}_1 + \overline{Q}_8)(\overline{Q}_1 + \overline{Q}_2 + \overline{Q}_4) = Q_1Q_2Q_4\overline{Q}_8 + \overline{Q}_1 Q_8$. These equations are mapped onto the K-maps for Table 7-9 below and meet the specifications given by the maps and the table.



To add the enable, change D1 to:

$D_1 = Q_1 \oplus EN$.

For the other three functions, AND EN with the expression XORed with the state variable. The circuit below results.



7-14.*

Present state			Next state		
A	*B*	*C*	*A*	*B*	*C*
	0	0		0	1
	0	1		1	0
	1	0		0	0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0

a) $D_B = C$

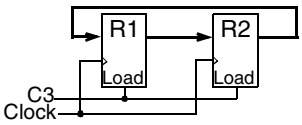
$D_C = \overline{B}\,\overline{C}$

b) $D_A = BC + A\overline{C}$

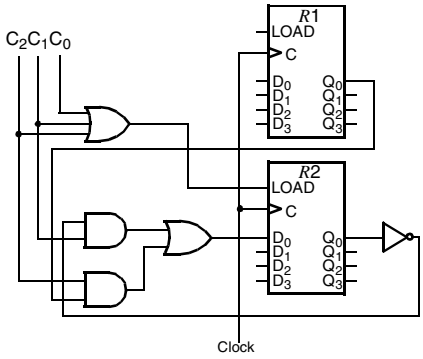
$D_B = \overline{A}\,\overline{B}C + B\overline{C}$

$D_C = \overline{C}$

7-17.*

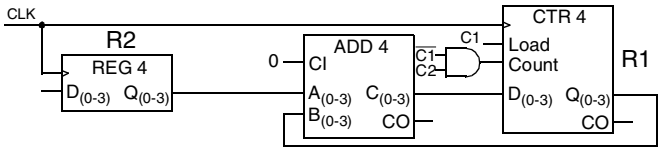


7-19.*

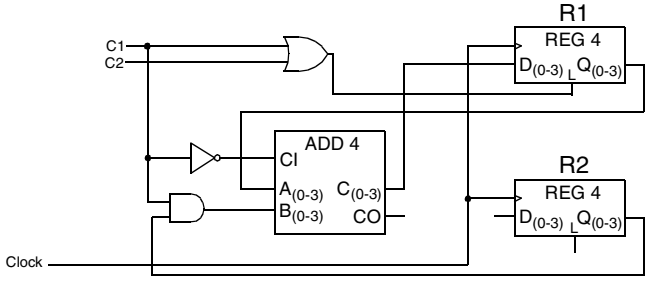


7-24.*

a)



b)



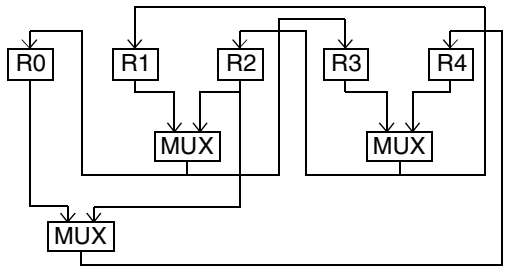
7-27.*

a) Destination <- Source Registers
R0 <- R1, R2
R1 <- R4
R2 <- R3, R4
R3 <- R1
R4 <- R0, R2

b) Source Registers -> Destination
R0 -> R4
R1 -> R0, R3
R2 -> R0, R4
R3 -> R2
R4 -> R1, R2

c) The minimum number of buses needed for operation of the transfers is three since transfer Cb requires three different sources.

d)



7-30.*

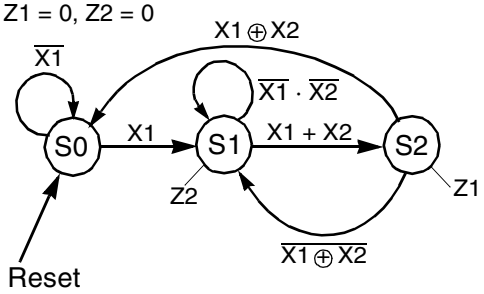
0101, 1010, 0101, 1010, 1101, 0110, 0011, 0001, 1000

7-31.*

Shifts:	0	1	2	3	4
A	0111	0011	0001	1000	1100
B	0101	0010	0001	0000	0000
C	0	1	1	1	0

7-32.*

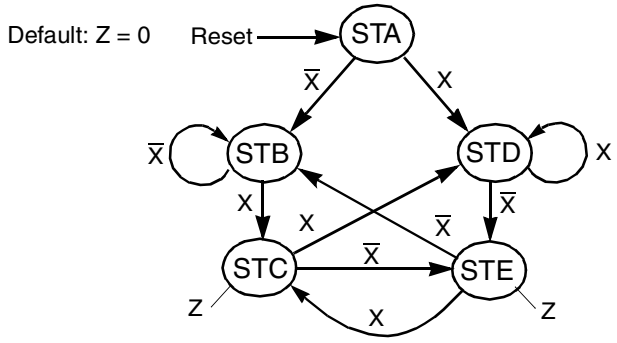
Default: Z1 = 0, Z2 = 0



7-33.*

State: STA, STA, STB, STC, STA, STB, STC, STA, STB
Z: 0, 0, 1, 1, 0, 0, 1, 0, -

7-36.*



7-39.*

Present state				Input	Next state			Output
	A	*B*	*C*		*A*	*B*	*C*	
STA	1	0	0	$\overline{W}$	1	0	0	
	1	0	0	W	0	1	0	
STB	0	1	0	$\overline{X}Y$	1	0	0	
	0	1	0	X	0	0	1	
	0	1	0	$\overline{X}\,\overline{Y}$	0	0	1	Z
STC	0	0	1		1	0	0	Z

$$D_A = A\overline{W} + B\overline{X}Y + C$$

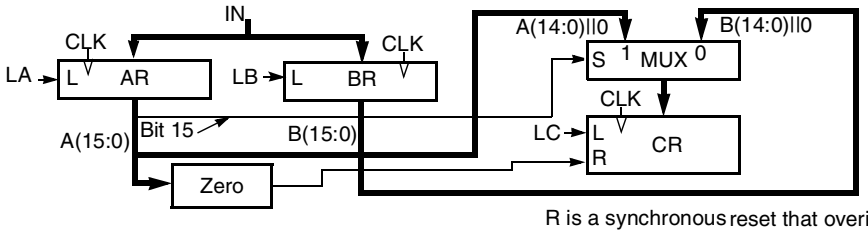
$$D_B = AW$$

$$D_C = B\,(X + \overline{Y})$$

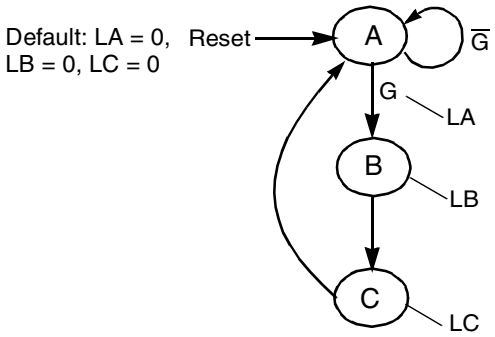
$$Z = B\,\overline{X}\,\overline{Y} + C$$

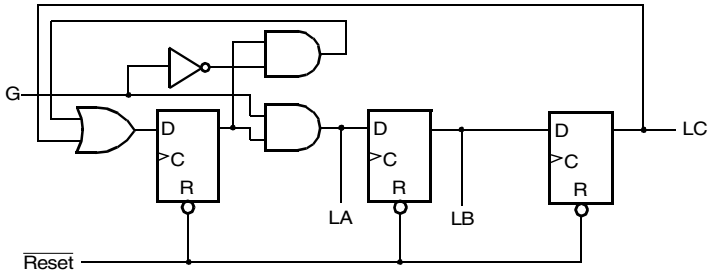
The implementation consists of the logic represented by the above equations and three D flip-flops with Reset connected to S on the first flip-flop and to R on the other two flip-flops.

7-46.*



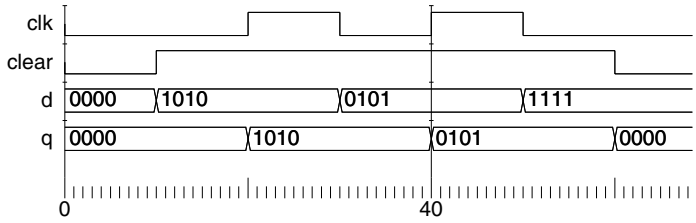
R is a synchronous reset that overides any simultaneous synchronous transfer.





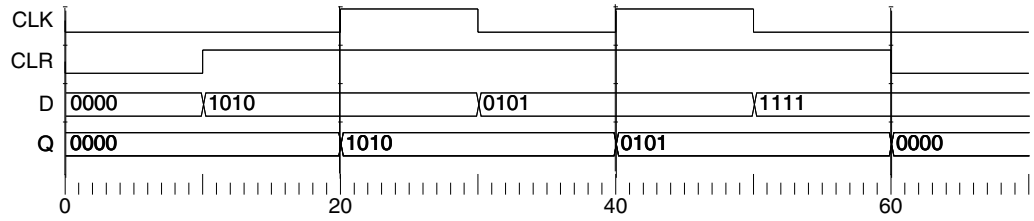
7-48.*

```
library IEEE;
use IEEE.std_logic_1164.all;

entity reg_4_bit is
   port (
      CLEAR, CLK: in STD_LOGIC;
      D: in STD_LOGIC_VECTOR (3 downto 0);
      Q: out STD_LOGIC_VECTOR (3 downto 0)
   );
end reg_4_bit;

architecture reg_4_bit_arch of reg_4_bit is
begin

process (CLK, CLEAR)
begin
  if CLEAR ='0' then                        --asynchronous RESET active Low
    Q <= "0000";
  elsif (CLK'event and CLK='1') then        --CLK rising edge
    Q <= D;
  end if;
end process;

end reg_4_bit_arch;
```



7-51.*

```
module register_4_bit (D, CLK, CLR, Q) ;

input [3:0] D ;
input CLK, CLR ;
output [3:0] Q ;
reg [3:0] Q ;

always @(posedge CLK or negedge CLR)
begin
      if (~CLR)                  //asynchronous RESET active low
            Q = 4'b0000;
      else                       //use CLK rising edge
      Q = D;
end
endmodule
```



7-53.*

```
library IEEE;
use IEEE.std_logic_1164.all;
entity prob_7_53  is
   port (clk, RESET, W, X, Y : in STD_LOGIC;
         Z  : out STD_LOGIC);
end prob_7_53;

architecture process_3 of prob_7_53 is
type state_type is (STA, STB, STC);
signal state, next_state: state_type;
begin

-- Process 1 - state register
state_register: process (clk, RESET)
begin
     if (RESET = '1') then
          state <= STA;
     else if (CLK'event and CLK='1') then
          state <= next_state;
          end if;
     end if;
end process;

-- Process 2 - next state function
next_state_func: process (W, X, Y, state)
begin
     case state is
          when STA =>
-- Continued in next column
               if W = '1' then
                    next_state <= STB;
               else
                    next_state <= STA;
               end if;
          when STB =>
               if X = '0' and Y = '1' then
                    next_state <= STA;
               else
                    next_state <= STC;
               end if;
          when STC =>
                    next_state <= STA;
     end case;
end process;

-- Process 3 - output function
output_func: process (X, Y, state)
begin
     case state is
       when STA =>
         Z <= '0';
         when STB =>
           if X = '0' and Y = '0' then
                Z <= '1';
           else
             Z <= '0';
           end if;
         when STC =>
             Z <= '1';
     end case;
 end process;
end process_3;
```

7-54.*

```
// State Diagram in Figure 5-40 using Verilog
module prob_7_54 (clk, RESET, W, X, Y, Z);
input clk, RESET, W, X, Y;
output Z;

reg[1:0] state, next_state;
parameter STA = 2'b00, STB = 2'b01, STC = 2'b10;
reg Z;

// State Register
always@(posedge clk or posedge RESET)
begin
if (RESET == 1)
     state <= STA;
else
     state <= next_state;
end

// Next StateFunction
always@(W or X or Y or state)
begin
     case (state)
     STA: if (W == 1)
               next_state <= STB;
          else
// (continued in the next column)
               next_state <= STA;
     STB: if (X == 0 & Y == 1)
               next_state <= STA;
          else
               next_state <= STC;
     STC:
               next_state <= STA;
     endcase
end

// Output Function
always@(X or Y or state)
begin
     Z <= 0;
     case (state)
          STB:  if (X == 0 & Y == 0)
               Z <= 1;
           else
               Z <= 0;
     STC:
               Z <= 1;
     endcase
end
endmodule
```

Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 8

© 2008 Pearson Education, Inc.

8-1.*

a) A = 16, D = 8 b) A = 19, D = 32 c) A = 26, D = 64 d) A = 31, D = 1

8-3.*

Number of bits in array = 2^{16} x 2^4 = 2^{20} = $2^{10} * 2^{10}$

Row Decoder size = 2^{10}

a) Row Decoder = 10 to 1024, AND gates = 2^{10} = 1024 (assumes 1 level of gates with 10 inputs/gate)
Column Decoder = 6 to 64, AND gates = 2^6 = 64 (assumes 1 level of gates with 6 inputs/gate)
Total AND gates required = 1024 + 64 =1088

b) $(32000)_{10} = (0111110100\ \ 000000)_2$, Row = 500, Column = 0

8-8.*

a) 2 MB/128 K x 16 = 2MB/ 256 KB = 8 b) With 2 byte/word, 2MB/2B = 2^{20}, Add Bits = 20
128K addresses per chip implies 17 address bits. c) 3 address lines to decoder, decoder is 3-to-8-line

Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 9

© 2008 Pearson Education, Inc.

9-2.*

$C = C_8$

$V = C_8 \oplus C_7$

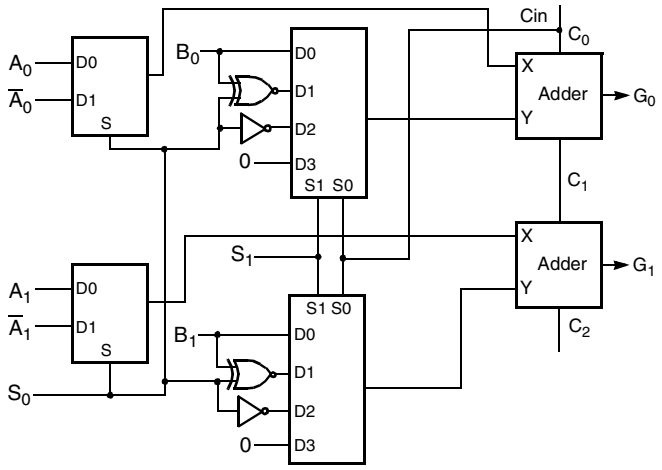
$Z = \overline{F_7 + F_6 + F_5 + F_4 + F_3 + F_2 + F_1 + F_0}$

$N = F_7$

9-3.*

$X = S_0\overline{A} + \overline{S}_0 A$

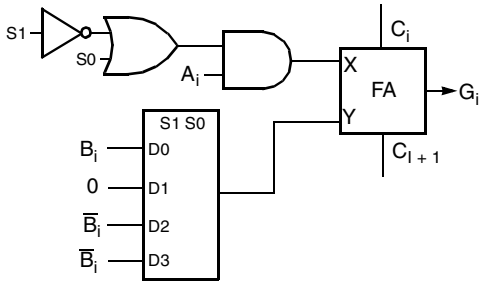
$Y = \overline{S}_1 \overline{C}_{in} B + \overline{S}_1 S_0 B + \overline{S}_1 \overline{S}_0 \overline{B} + S_1 \overline{S}_0 \overline{C}_{in}$



9-4.* (Errata: Delete "1" after problem number)

$X = A\,\overline{S1} + A\,S0$

$Y = B\,\overline{S1}\,\overline{S0} + \overline{B}\,S_1$



9-6.*

a) XOR = 00, NAND = 01, NOR = 10 XNOR = 11

Out = $S_1\,\overline{A}\,\overline{B} + \overline{S}_1 \overline{A} B + \overline{S}_1 A \overline{B} + S_1 S_0 AB$ + (one of $S_0\,\overline{A}\,\overline{B} + \overline{S}_1\,S_0\,\overline{A}$)

b) The above is a simplest result.

9-8.*

(a) 1010 (b) 1110 (c) 0101 (d) 1101

9-10.*

(a)	$R5 \leftarrow R4 \wedge \overline{R5}$	R5 = 0000 0100	(d)	$R5 \leftarrow R0$	R5 = 0000 0000
(b)	$R6 \leftarrow R2 + \overline{R4} + 1$	R6 = 1111 1110	(e)	$R4 \leftarrow sr$Constant	R4 = 0000 0011
(c)	$R5 \leftarrow R0$	R5 = 0000 0000	(f)	$R3 \leftarrow$ Data in	R3 = 0001 1011

9-13.*

a) Opcode = 8 bits b) 18 bits c) 262,144 d) +131,071 and –131,072

Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 10

© 2008 Pearson Education, Inc.

10-2.*

a)
```
LD    R1, E
LD    R2, F
MUL   R1, R1, R2
LD    R2, D
SUB   R1, R2, R1
LD    R2, C
DIV   R1, R2, R1
LD    R2, A
LD    R3, B
ADD   R2, R2, R3
MUL   R1, R1, R2
ST    Y, R1
```

b)
```
MOV   T1, A
ADD   T1, B
MUL   T1, C
MOV   T2, E
MUL   T2, F
MOV   T3, D
SUB   T3, T2
DIV   T1, T3
MOV   Y, T1
```

c)
```
LD    E
MUL   F
ST    T1
LD    D
SUB   T1
ST    T1
LD    A
ADD   B
MUL   C
DIV   T1
ST    Y
```

10-3.*

a) (A - B) x (A + C) x (B - D) = A B - A C +x B D -x

b, c)

PUSH A	PUSH B	SUB	PUSH A	PUSH C	ADD
A	B	A-B	A	C	A+C
	A		A-B	A	A-B
				A-B	

MUL	PUSH B	PUSH D	SUB	MUL	POP X
(A-B)x(A+C)	B	D	B-D	(A-B)x(A+C)x(B-D)	
	(A-B)x(A+C)	B	(A-B)x(A+C)		
		(A-B)x(A+C)			

10-6.*

a) X = 195 – 208 – 1 = –14 b) X = 1111 1111 1111 0010

The number is negative because the branch is backwards. The – 1 assumes that the PC has been incremented to point to the address after that of the address word of the instruction.

10-10.*

a) 3 Register Fields x 4 bits/Field = 12 bits. 32 bits - 12 bits = 20 bits. 2^{20} = 1048576

b) 64 < 100 < 128 => 7 bits. 2 Register Fields x 4 bits/Field => 8 bits. 32 bits - 7 bits - 8 bits => 17 Address Bits

10-14.*

a)
```
ADD   R0, R4
ADC   R1, R5
ADC   R2, R6
ADC   R3, R7
```

b) $R0 \leftarrow 7B + 4B$, R0 = C6, C = 0

$R1 \leftarrow 24 + ED + 0$, R1 = 11, C = 1

$R2 \leftarrow C6 + 57 + 1$, R2 = 1E, C = 1

$R3 \leftarrow 1F + 00 + 1$, R3 = 20, C = 0

10-17.*

OP	Result Register	C
	0110 1001	1
SHR	0011 0100	1
SHL	0110 1000	0
SHRA	0011 0100	0
SHLA	0110 1000	0
ROR	0011 0100	0
ROL	0110 1000	0
RORC	0011 0100	0
ROLC	01101000	0

10-19.*

$$\text{Smallest Number} = 0.5 \times 2^{-255}$$

$$\text{Largest Number} = (1 - 2^{-26}) \times 2^{+255}$$

10-20.*

E	e	$(e)_2$
+8	15	1111
+7	14	1110
+6	13	1101
+5	12	1100
+4	11	1011
+3	10	1010
+2	9	1001
+1	8	1000
0	7	0111
–1	6	0110
–2	5	0101
–3	4	0100
–4	3	0011
–5	2	0010
–6	1	0001
–7	0	0000

10-23.*

```
TEST  (0001)16, R     (AND Immediate 1 with Register R)
BNZ   ADRS            (Branch to ADRS if Z = 0)
```

10-25.*

a) A = 0101 1101 93
B = 0101 1100 - 92
A – B = 0000 0001 1

b) C (borrow) = 0, Z = 0

c) BA, BAE, BNE

10-27.*

	PC	SP	TOS
a) Initially	2000	4000	5000
b) After Call	0502	4001	2002
c) After Return	2002	4000	5000

10-30.*

External Interrupts:
1) Hard Drive
2) Mouse
3) Keyboard
4) Modem
5) Printer

Internal Interrupts:
1) Overflow
2) Divide by zero
3) Invalid opcode
4) Memory stack overflow
5) Protection violation

A software interrupt provides a way to call the interrupt routines normally associated with external or internal interrupts by inserting an instruction into the code. Privileged system calls for example must be executed through interrupts in order to switch from user to system mode. Procedure calls do not allow this change.

Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 11

© 2008 Pearson Education, Inc.

11-2.*

a) The latency time = 0.5 ns x 8 = 4.0 ns.

b) The maximum throughput is 1 instruction per cycle or 2 billion instructions per second.

c) The time required to execute is 10 instruction + 8 pipe stages -1 = 17 cycles *0.5ns = 8.5ns

11-6.*

Cycle 1: PC = 10F

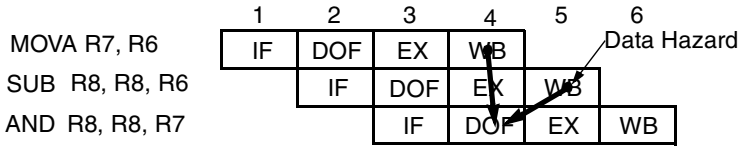
Cycle 2: PC_{-1} = 110, IR = 4418 $2F01_{16}$

Cycle 3: PC_{-2} = 110, RW = 1, DA = 01, MD = 0, BS = 0, PS = X, MW = 0, FS = 2, SH = 01, MA = 0, MB = 1
BUS A = 0000 001F, BUS B = 0000 2F01

Cycle 4: RW = 1, DA = 01, MD = 0, D0 = 0000 2F20, D1 = XXXX XXXX, D2 = 0000 00000

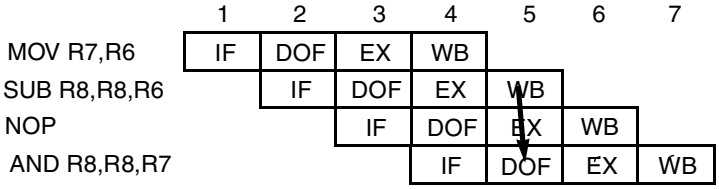
Cycle 5: R1 = 0000 2F20

11-10.*

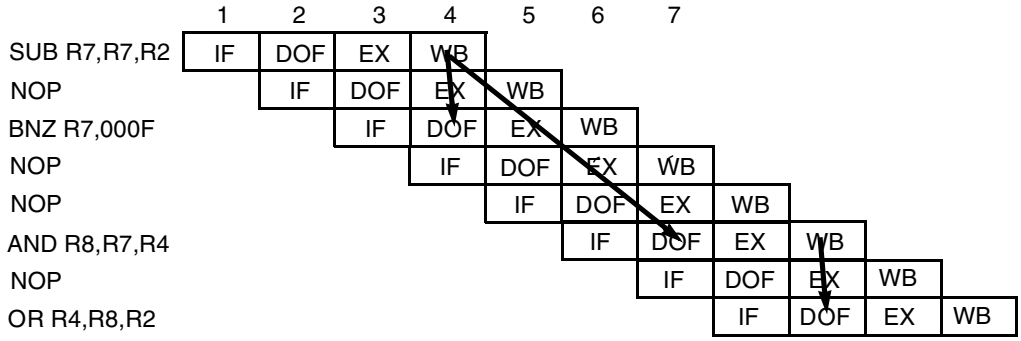


11-12.*

a)



b)



11-15.*

Time Cycle 1

IF PC: 0000 0001
DOF PC_{-1}:XXXXXXXX IR: XXXXXXXX
EX PC_{-2}:XXXXXXXX A: XXXXXXXX B: XXXXXXXX RW:X DA:XX MD:X BS:X PS:X MW:X FS:X MB:X MA:X CS:X D':X
WB D0: XXXXXXXX D1:XXXXXXXX D2:XXXXXXXX RW:X DA:XX MD:X

Time Cycle 2

IF PC: 0000 0002
DOF PC_{-1}:0000 0002 IR: 0A73 8800
EX PC_{-2}:XXXXXXXX A: XXXXXXXX B: XXXXXXXX RW:X DA:XX MD:X BS:X PS:X MW:X FS:X MB:X MA:X CS:X D':X
WB D0: XXXXXXXX D1:XXXXXXXX D2:XXXXXXXX RW:X DA:XX MD:X

Time Cycle 3

IF PC: 0000 0003
DOF PC_{-1}:0000 0003 IR: 9003 800F
EX PC_{-2}:0000 0002 A: 0000 0030 B: 0000 0010 RW:1 DA:07 MD:0 BS:0 PS:X MW:0 FS:5 MB:0 MA:0 CS:X D':X
WB D0: XXXXXXXX D1:XXXXXXXX D2:XXXXXXXX RW:X DA:XX MD:X

Time Cycle 4

IF PC: 0000 0004
DOF PC_{-1}:0000 0004 IR:1083 9000
EX PC_{-2}:0000 0003 A: 0000 0020 B: XXXXXXXX RW:0 DA:XX MD:X BS:1 PS:1 MW:0 FS:0 MB:1 MA:2 CS:1 D':1
WB D0: 0000 0020 D1:XXXX XXXX D2:0000 0000 RW:1 DA:07 MD: 0 PC: 0000 0012

Time Cycle 5

IF PC: 0000 0013 R7: 0000 0020
DOF PC_{-1}: 0000 00013 IR: 1244 0800
EX PC_{-2}: 0000 0004 A: 0000 0020 B: 0000 0020 RW: 1 DA: 08 MD: 0 BS: 0 PS: X MW: 0 FS: 8 MB:0 MA:0 CS:X D':X
WB D0: 0000 0020 D1: XXXXXXXX D2: 0000 0000 RW: 0 DA: 00 MD: 0

Time Cycle 6

IF PC: 0000 0014
DOF PC_{-1}: 0000 0014 IR: XXXX XXXX
EX PC_{-2}: 0000 0013 A: 0000 0020 B: 0000 0010 RW: 1 DA: 04 MD: 0 BS: 0 PS: X MW: 0 FS: 9 MB:0 MA:0 CS:X D':X
WB D0: 0000 0010 D1: XXXX XXXX D2: 0000 0000 RW: 1 DA: 08 MD: 0

Time Cycle 7

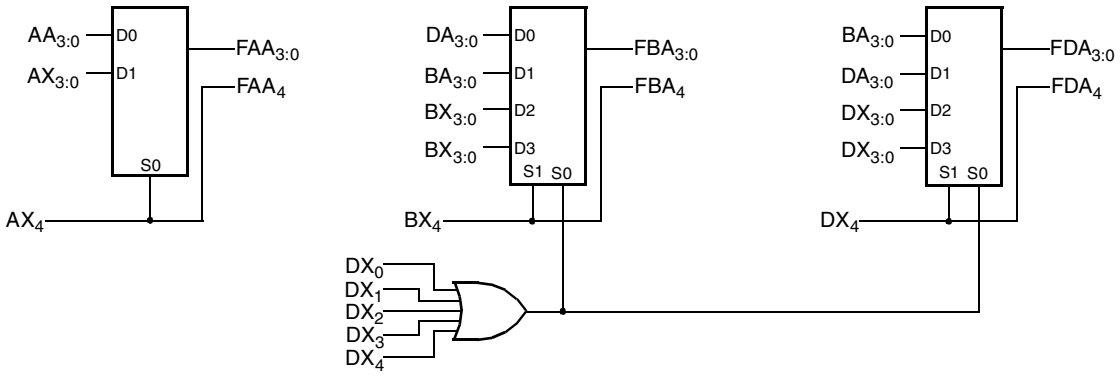
IF PC: 0000 0014 R7: 0000 0020
DOF PC_{-1}: 0000 0014 IR: XXXX XXXX
EX PC_{-2}: 0000 0013 A:XXXX XXXX B: XXXX XXXX RW: X DA: XX MD:X BS: X PS:X MW:X FS:X MB:X CS:X D':X
WB D0: 0000 0010 D1: XXXX XXXX D2: 0000 0000 RW: 1 DA: 04 MD:0

Time Cycle 8

R4: 0000 0010

Fields not specified above have fixed values throughout or are unused: SH. Based on the register contents, the branch is taken. The data hazards are avoided, but due to the control hazard, the last two instructions are erroneously executed.

11-18.*



11-22.*

(a) Add with carry

Action	Address	MZ	CA	RW	DX	MD	BS	PS	MW	FS	LC	MA	MB	AX	BX	CS
$R_{31} \leftarrow CC \wedge 00010$	AWC0	01	02	1	1F	0	00	0	0	8	0	10	1	00	00	11
$R_{16} \leftarrow R[SA] + R[SB]$	AWC1	01	00	1	10	0	00	0	0	2	0	00	0	00	00	00
if (R_{31}=0) $MC \leftarrow AWC5$ else $MC \leftarrow MC + 1$	AWC2	11	AWC5	0	00	0	00	0	0	0	0	00	0	1F	00	00
$MC \leftarrow MC + 1$ (NOP)	AWC3	01	00	0	00	0	00	0	0	0	0	00	0	00	00	00
$R[DR] \leftarrow R_{16} + 1$	AWC4	01	01	1	01	0	00	0	0	2	0	00	1	10	00	11
$MC \leftarrow IDLE$	AWC5	00	IDLE	0	00	0	00	0	0	0	0	00	0	00	00	00

(a) Subtract with borrow

Action	Address	MZ	CA	RW	DX	MD	BS	PS	MW	FS	LC	MA	MB	AX	BX	CS
$R_{31} \leftarrow CC \wedge 00010$	SWB0	01	02	1	1F	0	00	0	0	8	0	10	1	00	00	11
$R_{16} \leftarrow R[SA]–R[SB]$	SWB1	01	00	1	10	0	00	0	0	5	0	00	0	00	00	00
if (R_{31}≠0) $MC \leftarrow SWB5$ else $MC \leftarrow MC + 1$	SWB2	11	SWB5	0	00	0	00	1	0	0	0	00	0	1F	00	00
$MC \leftarrow MC + 1$ (NOP)	SWB3	01	00	0	00	0	00	0	0	0	0	00	0	00	00	00
$R[DR] \leftarrow R_{16} - 1$	SWB4	01	01	1	01	0	00	0	0	5	0	00	1	10	00	11
$MC \leftarrow IDLE$	SWB5	00	IDLE	0	00	0	00	0	0	0	0	00	0	00	00	00

11-24.*

Memory Scalar Add (Assume R[SB] > 0 to simplify coding)

Action	Address	MZ	CA	RW	DX	MD	BS	PS	MW	FS	LC	MA	MB	AX	BX	CS
$R_{16} \leftarrow R[SB]$	MSA0	01	00	1	10	0	00	0	0	0	0	00	0	00	00	00
$R_{18} \leftarrow R_0$	MSA1	01	00	1	12	0	00	0	0	0	0	00	0	00	00	00
$R_{16} \leftarrow R_{16} - 1$	MSA2	01	01	1	10	0	00	0	0	5	0	00	1	10	00	11
$MC \leftarrow MC + 1$ (NOP)	MSA3	01	00	0	00	0	00	0	0	0	0	00	0	00	00	00
$R_{17} \leftarrow R[SA] + R_{16}$	MSA4	01	00	1	11	0	00	0	0	2	0	00	0	00	10	00
$MC \leftarrow MC + 1$ (NOP)	MSA5	01	00	0	00	0	00	0	0	0	0	00	0	00	00	00
if ($R_{16} \neq 0$) $MC \leftarrow MSA2$ else $MC \leftarrow MC + 1$	MSA6	11	MSA2	0	00	0	00	0	0	0	0	00	0	10	00	00
$R_{18} \leftarrow M[R_{17}] + R_{18}$	MSA7	01	00	1	12	1	00	0	0	0	0	00	0	11	12	00
$R[DR] \leftarrow R_{17}$	MSA8	01	00	1	01	0	00	0	0	0	0	00	0	11	00	00
$MC \leftarrow IDLE$	MSA9	00	IDLE	0	00	0	00	0	0	0	0	00	0	00	00	00

Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 12

© 2008 Pearson Education, Inc.

12-1.*

Heads x (cylinders/Head) x (sectors/cylinder) x (1 cylinder/track) x (bytes/sector)

a) 1 x 1023 x 63 x 512 = 32,224.5 Kbytes (K = 1024)

b) 4 x 8191 x 63 x 512 = 1,032,066 Kbytes

c) 16 x 16383 x 63 x 512 = 8,257,032 Kbytes

12-5.*

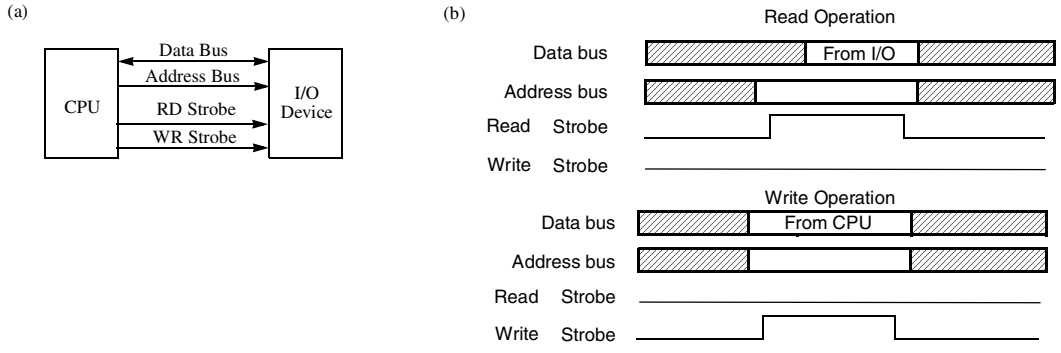
a) If each address line is used for a different CS input, there will be no way to address the four registers so 2 bits are needed to address the registers. Only 14 lines can be used for CS inputs permitting at most 14 I/O Interface Units to be supported.

b) Since two bits must be used to address the four registers, there are 14 bits remaining and 2^{14} or 16,384 distinct I/O Interface Units can be supported.

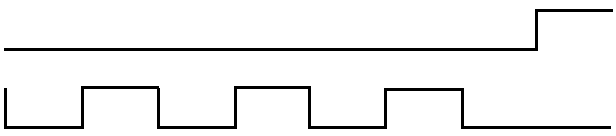
12-7.*

A given address can be shared by two registers if one is write only and one is read only. If a register is both written to and read from the bus, then it needs its own address. An 8-bit address provides 256 addresses. Suppose that the 50 % of registers requiring 1 address is X. Then the remaining 50 % of the registers, also X can share addresses requiring only 0.5 addresses. So 1.5 X = 256 and X = 170.67 registers for a total of 341.33 registers. To meet the original constraints exactly, the total number of registers must be divisible by 4, so 340 registers can be used, 170 of which are read/write, 85 of which are read only and 85 of which are write only. There is one more address available for either one read/write register or up to a pair with a read only register and a write only register.

12-9.*



12-11.*



There are 7 edges in the NRZI waveform for the SYNC pattern that can be used for synchronization.

12-13.*

SYNC 8 bits	Type 4 bits 1001	Check 4 bits 0110	Device Address 0100111	Endpoint Address 0010	CRC	EOP

(a) Output packet

SYNC 8 bits	Type 4 bits 1100	Check 4 bits 0011	Data 01000010100111101010011 0	CRC	EOP

(b) Data packet (Data0 type) (bits LSB first)

SYNC 8 bits	Type 4 bits 0111	Check 4 bits 1000	EOP

(c) Handshake packet (Stall type)

12-16.*

Description	Device 0				Device 1				Device 2			
	PI	PO	RF	VAD	PI	PO	RF	VAD	PI	PO	RF	VAD
Initially	0	0	0	-	0	0	0	-	0	0	1	-
Before CPU acknowledges Device 2	0	0	1	-	0	0	0	-	0	0	1	-
After CPU sends acknowledge	1	0	1	0	0	0	0	-	0	0	1	-

12-18.*

Replace the six leading 0's with 000110.

12-20.*

This is Figure 13-17 with the Interrupt and Mask Registers increased to 6 bits each, and the 4x2 Priority Encoder replaced by a 8x3 Priority Encoder. Additionally, VAD must accept a 3rd bit from the Priority Encoder.

12-22.*

When the CPU communicates with the DMA, the read and write lines are used as DMA inputs. When the DMA communicates with the Memory, these lines are used as outputs from the DMA.

Solutions to Problems Marked with a * in
Logic and Computer Design Fundamentals, 4th Edition

Chapter 13

© 2008 Pearson Education, Inc.

13-3.*

Since the lines are 32 bytes, 5 bits are used to address bytes in the lines.

Since there are 1K bytes, there are $1024/32 = 2^5$ cache lines.

a) Index = 5 Bits,

b) Tag = 32 – 5 – 5 = 22 Bits

c) $32 \times (32 \times 8 + 22 + 1) = 8928$ bits

13-5.*

a) See Instruction and Data Caches section on page 635 of the text.

b) See Write Methods section on page 631 of the text.

13-7. *

000000 00 00 (i0)	000001 00 00 (i4)	000001 10 00 (i6)	000010 10 00 (i10)
000000 01 00 (i1)	000011 00 00 (d)	00001 11 00 (i7)	000011 10 00 (d)
000000 10 00 (i2)	000001 01 00 (i5)	000010 00 00 (i8)	000010 11 00 (i11)
000000 11 00 (i3)	000011 01 00 (d)	000010 01 00 (i9)	000011 11 00 (d)

Addresses of instructions (i) and Data (d) in sequence down and then to the right with the instructions in a loop with instruction i0 following i11. For the split cache, the hit - miss pattern for instructions is (assuming the cache initially empty and LRU replacement) M, M, M, M, M, M, M, M, M, M, M, M, M, ... since there are only eight locations available for instructions. For the unified cache, the hit-miss pattern for instructions with the same assumptions is M, M, M, M, M, M, M, M, M, M, M, M, H, H, H, ... since there are 12 locations indexed appropriately for instructions and four indexed appropriately for data.

13-10.*

a) Effective Access Time = 0.91 * 4ns + 0.09 * 40 ns = 7.24 ns

b) Effective Access Time = 0.82 * 4ns + 0.18* 40 ns = 10.48 ns

c) Effective Access Time = 0.96 * 4ns + 0.04 * 40 ns = 5.44 ns

13-13.*

a) Each page table handles 512 pages assuming 64-bit words. There are 4263 pages which requires 4263/512 8.33 page tables. So 9 page tables are needed.

b) 9 directory entries are needed, requiring 1 directory page.

c) 4263 - 8*512 = 167 entries in the last page table.

13-17.*

In section 14-3, it is mentioned that write-through in caches can slow down processing, but this can be avoided by using write buffering. When virtual memory does a write to the secondary device, the amount of data being written is typically very large and the device very slow. These two factors generally make it impossible to do write-through with virtual memory. Either the slow down is prohibitively large, or the buffering cost is just too high.