Instructor's Manual for Logic and Computer Design Fundamentals - 3rd Edition - Partial Preliminary Release

M. Morris Mano
Charles R. Kime

Pearson Prentice Hall
Pearson Education, Inc.
Upper Saddle River, New Jersey

Inclusion of the name of any product or vendor does not constitute an endorsement by either the authors or Pearson Prentice Hall. Altera and Flex 10K are trademarks of Altera Corporation. GAL and PAL are trademarks of Lattice Semiconductor Corporation. Mentor Graphics, Model Technology, and ModelSim are trademarks of Mentor Graphics Corporation. RAMBUS and RDRAM are registered trademarks of RAMBUS, Inc. Xilinx and Spartan are registered trademarks of Xilinx, Inc. Verilog is a registered trademark of Cadence Design Systems, Inc. PowerPoint is registered trademark of the Microsoft Corporation.

Pearson Prentice Hall
Pearson Education, Inc.
Upper Saddle River, NJ 07458

ISBN ...

...

PART 2 PROBLEM SOLUTIONS

NOTES ON SOLUTIONS:

1. **Legal Notice:** This publication is protected by United States copyright laws, and is designed exclusively to assist instructors in teaching their courses. It should not be made available to students, or to anyone except the authorized instructor to whom it was provided by the publisher, and should not be sold by anyone under any circumstances. Publication or widespread dissemination (i.e. dissemination of more than extremely limited extracts within the classroom setting) of any part of this material (such as by posting on the World Wide Web) is not authorized, and any such dissemination will violate the United States copyright laws. In consideration of the authors, your colleagues who do not want their students to have access to these materials, and the publisher, please respect these restrictions.

2. **Companion Website Problem Solutions:** The solutions to all problems marked with a * are available to students as well as instructors on the Companion Website.

3. **Problem Challenge:** The problems marked with a + are designated as more challenging than the typical problems.

4. **Text Errata Notations:** Text errata are noted at the beginning of a problem if those errata affect either the problem or its solution. These notes indicate only errors identified in the first printing of the 3rd Edition and are expected be removed after the first printing.

5. **Solutions Errata:** Errata for these solutions will be provided on the Companion Website in the Errata section.

CHAPTER 1

1-1.*

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

1-2.

$$
\begin{aligned}
48K &= 48 \times 2^{10} = 49,152 \text{ Bits} \\
384M &= 384 \times 2^{20} = 402,653,184 \text{ Bits} \\
8G &= 8 \times 2^{30} = 8,589,934,592 \text{ Bits}
\end{aligned}
$$

1-3.

$$
\begin{aligned}
&12 \text{ Bits } \Rightarrow 2^{12} - 1 = 4095 \\
&24 \text{ Bits} \Rightarrow 2^{24} - 1 = 16,777,215
\end{aligned}
$$

1-4.*

$$
\begin{aligned}
(1001101)_2 &= 2^6 + 2^3 + 2^2 + 2^0 = 77 \\
(1010011.101)_2 &= 2^6 + 2^4 + 2^1 + 2^0 + 2^{-1} + 2^{-3} = 83.625 \\
(10101110.1001)_2 &= 2^7 + 2^5 + 2^3 + 2^2 + 2^1 + 2^{-1} + 2^{-4} = 174.5625
\end{aligned}
$$

1-5.

$$
\begin{aligned}
125 &= 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = (1111101)_2 \\
610 &= 2^9 + 2^6 + 2^5 + 2^1 = (1001100010)_2 \\
2003 &= 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^4 + 2^1 + 2^0 = (11111010011)_2 \\
18944 &= 2^{14} + 2^{11} + 2^9 = (100101000000000)_2
\end{aligned}
$$

1-6.

$$
\begin{aligned}
(11100111)_2 &= 2^7 + 2^6 + 2^5 + 2^2 + 2^1 + 2^0 = 231 \\
(22120)_3 &= 2 \times 3^4 + 2 \times 3^3 + 1 \times 3^2 + 2 \times 3^1 = 231 \\
(3113)_4 &= 3 \times 4^3 + 1 \times 4^2 + 1 \times 4^1 + 3 \times 4^0 = 215 \\
(4110)_5 &= 4 \times 5^3 + 1 \times 5^2 + 1 \times 5^1 = 530 \\
(343)_8 &= 3 \times 8^2 + 4 \times 8^1 + 3 \times 8^0 = 227
\end{aligned}
$$

1-7. *

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

1-8.*

$$\begin{array}{lrcrcl}
\text{a)} & 7562/8 & = & 945 + 2/8 & \Rightarrow & 2 \\
 & 945/8 & = & 118 + 1/8 & \Rightarrow & 1 \\
 & 118/8 & = & 14 + 6/8 & \Rightarrow & 6 \\
 & 14/8 & = & 1 + 6/8 & \Rightarrow & 6 \\
 & 1/8 & = & 1/8 & \Rightarrow & 1 \\
 & & & & & \\
 & 0.45 \times 8 & = & 3.6 & \Rightarrow & 3 \\
 & 0.60 \times 8 & = & 4.8 & \Rightarrow & 4 \\
 & 0.80 \times 8 & = & 6.4 & \Rightarrow & 6 \\
 & 0.20\text{x}8 & = & 3.2 & \Rightarrow & 3 \\
 & (7562.45)_{10} & = & (16612.3463)_8 & & \\
 & & & & & \\
\text{b)} & (1938.257)_{10} & = & (792.41CA)_{16} & & \\
\text{c)} & (175.175)_{10} & = & (10101111.001011)_2 & &
\end{array}$$

1-9.*

$$\begin{array}{lrcl}
\text{a)} & (673.6)_8 & = & (110\ 111\ 011.110)_2 \\
 & & = & (1BB.C)_{16} \\
\text{b)} & (E7C.B)_{16} & = & (1110\ 0111\ 1100.1011)_2 \\
 & & = & (7174.54)_8 \\
\text{c)} & (310.2)_4 & = & (11\ 01\ 00.10)_2 \\
 & & = & (64.4)_8
\end{array}$$

1-10.

a)

$$\begin{array}{r}
1101 \\
\underline{\text{x}1001} \\
1101 \\
0000 \\
0000 \\
\underline{1101} \\
1110101
\end{array}$$

b)

$$\begin{array}{r}
0101 \\
\underline{\text{x}1011} \\
0101 \\
0101 \\
0000 \\
\underline{0101} \\
110111
\end{array}$$

c)

$$\begin{array}{r}
100101 \\
\underline{\text{x}011011} \\
100101 \\
100101 \\
000000 \\
100101 \\
100101 \\
\underline{000000} \\
1111100111
\end{array}$$

1-11.

1) $(101)_2 \times (10000)_2 = (1010000)_2$

$(1011110)_2 - (1010000)_2 = (0001110)_2$

Partial Quotient = $(10000)_2$

Partial Remainder = $(0001110)_2 \geq (101)_2 \geq (0)_2$

2) $(101)_2 \times (01000)_2 = (0101000)_2$

$(0001110)_2 - (0101000)_2 < (0)_2$

Partial Quotient = $(00000)_2$

Partial Remainder = $(0001110)_2 \geq (101)_2$

3) $(101)_2 \times (00100)_2 = (0010100)_2$

$(0001110)_2 - (0010100)_2 < (0)_2$

Partial Quotient = $(00000)_2$

Partial Remainder = $(0001110)_2 \geq (101)_2$

4) $(101)_2 \times (00010)_2 = (0001010)_2$

$(0001110)_2 - (0001010)_2 = (100)_2 \geq (0)_2$

Partial Quotient = $(00010)_2$

Partial Remainder = $(100)_2 < (101)_2$

Algorithm stops. (Partial Remainder < Divisor)

Quotient = $\sum$ Partial Quotients $= (10010)_2$

Remainder = Final Partial Remainder = $(100)_2$

1-12.

a)	0	1	2	3	4	5	6	7	8	9
	A	B	C	D	E	F	G	H	I	J

b)	2003/20	=	100 + 3/20	=>	3
	100/20	=	5 + 0/20	=>	0
	5/20	=	5/20	=>	5
	$(2003)_{10}$	=	$(503)_{20}$		

c) $(BCH.\mathrm{G}) = 11 \times 20^2 + 12 \times 20^1 + 17 \times 20^0 + 16 \times 20^{-1} = 4657.8$

1-13.*

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

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

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

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

1-14.

Noting the order of operations, first add $(35)_r$ and $(24)_r$

$$(35)_r = 3 \times r^1 + 5 \times r^0$$

$$(24)_r = 2 \times r^1 + 4 \times r^0$$

$$(35)_r + (24)_r = 5 \times r^1 + 9 \times r^0$$

Now, multiply the result by $(21)_r$

$$(2 \times r^1 + 1 \times r^0) \times (5 \times r^1 + 9 \times r^0) = 10 \times r^2 + 23 \times r^1 + 9$$

Next, set the result equal to $(1501)_r$ and reorganize.

$$10 \times r^2 + 23 \times r^1 + 9 = 1 \times r^3 + 5 \times r^2 + 1 \times r^0$$

$$1 \times r^3 - 5 \times r^2 - 23 \times r^1 - 8 \times r^0 = 0$$

Finally, find the roots of this cubic polynomial.

Solutions are: r = 8, – 2.618..., – 0.381...

ANSWER: The chicken has 4 toes on each foot (half of 8).

1-15.*

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

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

```
             1 ◄─┐
          0110   │      1001        0100
         +1000   │     +0011       +0101
          1111   │      1100        1001
         +0110   │     +0110       +0000
   0001   0101   └──── 1 0010        1001
```

1-16.*

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

$= (1001100000011)_2$

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

$= (101111010.11)_2$

1-17.

Binary Numbers from $(16)_{10}$ to $(31)_{10}$ with Odd and Even Parity

Decimal	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Odd	10000 0	10001 1	10010 1	10011 0	10100 1	10101 0	10110 0	10111 1	11000 1	11001 0	11010 0	11011 1	11100 0	11101 1	11110 1	11111 0
Even	10000 1	10001 0	10010 0	10011 1	10100 0	10101 1	10110 1	10111 0	11000 0	11001 1	11010 1	11011 0	11100 1	11101 0	11110 0	11111 1

1-18.

Gray Code for Hexadecimal Digits

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Gray	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000

1-19.

The percentage of power consumed by the Gray code counter compared to a binary code counter equals:

$$\frac{\text{Number of bit changes using Gray code}}{\text{Number of bit changes using binary code}}$$

As shown in table 1-4, and by definition, the number of bit changes per cycle of an n-bit Gray code counter is 1 per count = 2^n.

$$\text{Number of bit changes using Gray code} = 2^n$$

For a binary counter, notice that the least significant bit changes on every increment. The second least significant bit changes on every other increment. The third digit changes on every fourth increment of the counter, and so on. As shown in Table 1-4, the most significant digit changes twice per cycle of the binary counter.

$$\text{Number of bit changes using binary code} = 2^n + 2^{n-1} + \ldots + 2^1$$

$$= \sum_{i=1}^{n} 2^i = \left[\sum_{i=0}^{n} 2^i\right] - 1 = (2^{(n+1)} - 1) - 1 = 2^{n+1} - 2$$

$$\%\ \text{Power} = \frac{2^n}{2^{(n+1)} - 2}$$

1-20.

From Table 1-4, complementing the 6th bit will switch an uppercase letter to a lower case letter and vice versa.

1-21.

a) The name used is Brent M. Ledvina. An alternative answer: use both upper and lower case letters.

0100	0010	B	0101	0010	R	0100	0101	E
0100	1110	N	0101	0100	T	0010	0000	(SP)
0100	1101	M	0010	1110	.	0010	0000	(SP)
0100	1100	L	0100	0101	E	0100	0100	D
0101	0110	V	0100	1001	I	0100	1110	N
0100	0001	A						

b)	0100	0010	1101	0010	1100	0101
	0100	1110	1101	0100	1010	0000
	0100	1101	0010	1110	1010	0000
	1100	1100	1100	0101	0100	0100
	0101	0110	1100	1001	0100	1110
	0100	0001				

1-22.

ANSWER: John Doe

1-23.*

a) $(101101101)_2$

b) $(0011\ 0110\ 0101)_{BCD}$

c) 0011 0011 0011 0110 0011 0101_{ASCII}

1-24.

a) 2^{32} = 4,294,967,296 Integers

b) $32 \text{ Bits} \times (1 \text{ Digit})/(4\text{Bits}) = 8\text{Digits} = 10^8 = 100{,}000{,}000$ Integers

c) $32 \text{ Bits} \times (1 \text{ Digit})/(8 \text{ Bits}) = 4 \text{ Digits} = 10^4$ = 10,000 Integers

CHAPTER 2

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

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

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

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

b) $\overline{A}B + \overline{B}\,\overline{C} + AB + \overline{B}C \qquad = \qquad 1$

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

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

$\overline{= B + \overline{B}}$

$= 1$

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

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

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

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

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

$= X + Y + Z$

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

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

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

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

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

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

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

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

$= \overline{X}\,\overline{Y} + XZ + Y\overline{Z}$

2-3.+

a) $AB + B\overline{C}\,\overline{D} + \overline{A}BC + \overline{C}D \qquad = \qquad B + \overline{C}D$

$= (AB + ABC) + B\overline{C}\,\overline{D} + \overline{A}BC + (\overline{C}D + B\overline{C}D)$

$= AB + B\overline{C}(D + \overline{D}) + BC(A + \overline{A}) + \overline{C}D$

$= AB + B\overline{C} + BC + \overline{C}D$

$= B + AB + \overline{C}D$

$= B + \overline{C}D$

b) $WY + \overline{W}Y\overline{Z} + WXZ + \overline{W}X\overline{Y} \qquad = \qquad WY + \overline{W}X\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}Z$

$= (WY + W\overline{X}Y\overline{Z}) + (\overline{W}XY\overline{Z} + \overline{W}\,\overline{X}Y\overline{Z}) + (WXYZ + WX\overline{Y}Z) + (\overline{W}X\overline{Y}Z + \overline{W}X\overline{Y}\,\overline{Z})$

$= (WY + WXYZ) + (\overline{W}XY\overline{Z} + \overline{W}X\overline{Y}\,\overline{Z}) + (\overline{W}\,\overline{X}Y\overline{Z} + W\overline{X}Y\overline{Z}) + (WX\overline{Y}Z + \overline{W}X\overline{Y}Z)$

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

$= WY + \overline{W}X\overline{Z} + \overline{X}Y\overline{Z} + X\overline{Y}Z$

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

$= \overline{\overline{A\overline{C} + \overline{A}B + \overline{B}C + \overline{D}}}$

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

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

$= \overline{\overline{A}\,\overline{B}\,\overline{C}D + ABCD}$

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

2-4.+

Given: $A \cdot B = 0, A + B = 1$

Prove: $(A + C)(\overline{A} + B)(B + C) \quad = \quad BC$

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

$= AB + \overline{A}C + BC$

$= 0 + C(\overline{A} + B)$

$= C(\overline{A} + B)(0)$

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

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

$= BC$

2-5.+

Step 1: Define all elements of the algebra as four bit vectors such as *A, B* and *C*:

$A = (A_3, A_2, A_1, A_0)$

$B = (B_3, B_2, B_1, B_0)$

$C = (C_3, C_2, C_1, C_0)$

Step 2: Define OR_1, AND_1 and NOT_1 so that they conform to the definitions of AND, OR and NOT presented in Table 2-1.

a) $A + B = C$ is defined such that for all *i*, *i* = 0, ... ,3, C_i equals the OR_1 of A_i and B_i.

b) $A\ B = C$ is defined such that for all *i*, *i* = 0, ... ,3, C_i equals the AND_1 of A_i and B_i.

c) The element 0 is defined such that for *A* = "0", for all *i, i* = 0, ... ,3, A_i equals logical 0.

d) The element 1 is defined such that for *A* = "1", for all *i, i* = 0, ... ,3, A_i equals logical 1.

e) For any element *A*, $\overline{A}$ is defined such that for all *i*, *i* = 0, ... ,3, $\overline{A_i}$ equals the NOT_1 of A_i.

2-6.

a) $\overline{AC} + \overline{A}BC + \overline{B}C = (\overline{A} + \overline{C}) + \overline{A}BC + \overline{B}C = \overline{A} + (\overline{C} + \overline{B}\overline{C}) + \overline{B}C$

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

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

c) $ABC + \overline{A}C = (\overline{A}C + \overline{A}BC) + ABC = \overline{A}C + BC(A + \overline{A}) = \overline{A}C + BC = C(\overline{A} + B)$

d) $BC + B(AD + \overline{C}D) = BC + BAD + B\overline{C}D = (BC + BCD) + B\overline{C}D + BAD$

$= BC + BD(C + \overline{C}) + BAD = BC + BD + BAD = BC + BD = B(C + D)$

e) $(B + \overline{C} + B\overline{C})(BC + A\overline{B} + AC) = (B + \overline{C})(BC + A\overline{B} + AC) = BC + ABC + A\overline{B}\overline{C}$

$= BC + A\overline{B}\overline{C}$

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

$= ABCD + \overline{A} + \overline{C} = \overline{A} + \overline{C} + A(BCD) = \overline{A} + \overline{C} + C(BD) = \overline{A} + \overline{C} + BD$

2-8.

a) $F = \bar{A}BC + \bar{B}\bar{C} + A\bar{B}$

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

b) $\bar{\bar{F}} = \overline{\overline{\overline{\bar{A}BC + \bar{B}\bar{C} + A\bar{B}}}}$

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

$= \overline{\overline{(\bar{A}BC)}\,\overline{(\bar{B}\bar{C})}\,\overline{(A\bar{B})}}$

2-9.*

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

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

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

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

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
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: $\bar{X}YZ + X\bar{Y}Z + XY\bar{Z} + XYZ$

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

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

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

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

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

2-11.

a) $E = \Sigma m(0, 2, 5, 6) = \Pi M(1, 3, 4, 7)$, $F = \Sigma m(2, 4, 6, 7) = \Pi M(0, 1, 3, 5)$

b) $\bar{E} = \Sigma m(1, 3, 4, 7)$, $\bar{F} = \Sigma m(0, 1, 3, 5)$

c) $E + F = \Sigma m(0, 2, 4, 5, 6, 7)$, $E \cdot F = \Sigma m(2, 6)$

d) $E = \bar{X}\bar{Y}\bar{Z} + \bar{X}Y\bar{Z} + X\bar{Y}Z + XY\bar{Z}$, $F = \bar{X}Y\bar{Z} + X\bar{Y}\bar{Z} + XY\bar{Z} + XYZ$

e) $E = \bar{Z}(\bar{X} + Y) + X\bar{Y}Z$, $F = Y(\bar{Z} + X) + X\bar{Z}$

2-12.*

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

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

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

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

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

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

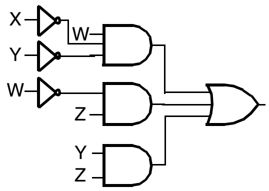
$= (A + B + C)(A + B + D)(A + \bar{C} + D)(\bar{B} + E)(\bar{B} + F)$ p.o.s.

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

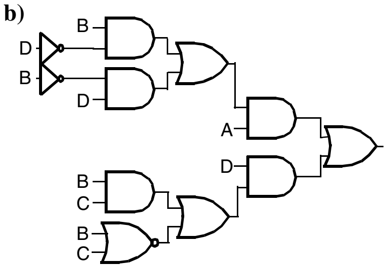
$= A\bar{B} + AEF + B\bar{C}EF + \bar{B}CD + CDEF$ s.o.p.

2-13.

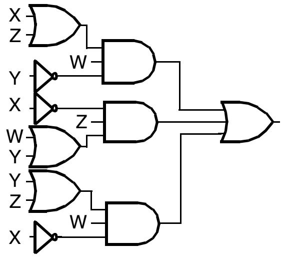
a)



b)

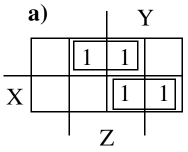


c)



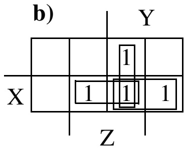
2-14.

a)



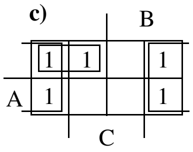
$\bar{X}Z + XY$

b)



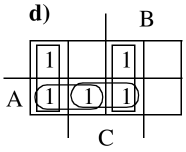
$YZ + XZ + XY$

c)



$\bar{C} + \bar{A}\bar{B}$

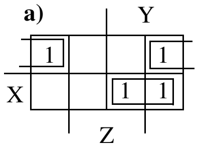
d)



$\bar{B}\bar{C} + BC + (AC \text{ or } A\bar{B})$

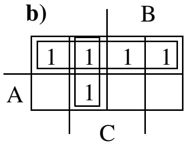
2-15. *

a)



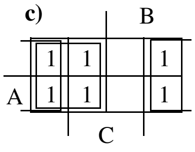
$\bar{X}\bar{Z} + XY$

b)



$\bar{A} + C\bar{B}$

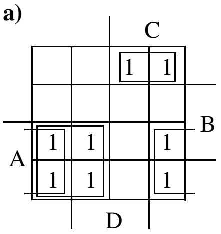
c)



$\bar{B} + \bar{C}$

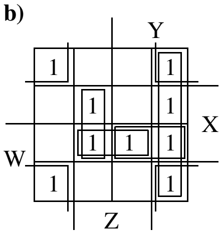
2-16.

a)



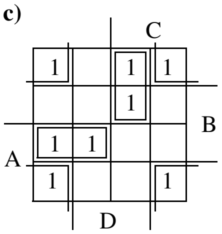
$A\overline{C} + A\overline{D} + \overline{A}\overline{B}C$

b)



$Y\overline{Z} + \overline{X}\overline{Z} + X\overline{Y}Z + (WXZ \text{ or } WXY)$

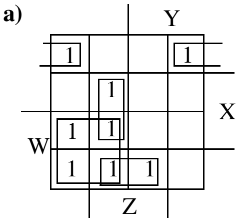
c)



$\overline{B}\overline{D} + AB\overline{C} + \overline{A}CD$

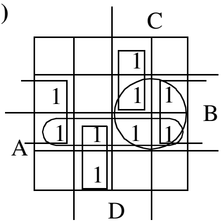
2-17.

a)



$F = W\overline{Y} + W\overline{X}Z + X\overline{Y}Z + \overline{W}\overline{X}\overline{Z}$

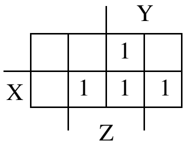
b)



$F = B\overline{D} + A\overline{C}D + \overline{A}CD + (AB \text{ or } BC)$

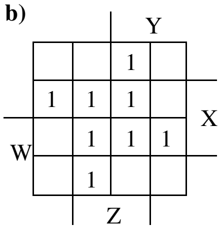
2-18. *

a)



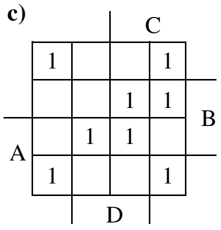
$\Sigma m(3, 5, 6, 7)$

b)



$\Sigma m(3, 4, 5, 7, 9, 13, 14, 15)$

c)



$\Sigma m(0, 2, 6, 7, 8, 10, 13, 15)$

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

2-20.

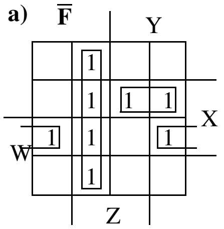
a) $Prime = \overline{X}\overline{Y}, \overline{W}\overline{Y}, \overline{Y}\overline{Z}, WX\overline{Z}, WXY, XYZ, \overline{W}XZ$
$Essential = \overline{X}\overline{Y}$
$F = \overline{X}\overline{Y} + (\overline{W}\overline{Y} + XYZ + WX\overline{Z} \text{ or } \overline{Y}\overline{Z} + WXY + \overline{W}XZ)$

b) $Prime = AB\overline{C}, ACD, \overline{A}BC, \overline{A}\overline{C}D, BD$
$Essential = AB\overline{C}, ACD, \overline{A}BC, \overline{A}\overline{C}D$
$F = AB\overline{C} + ACD + \overline{A}BC + \overline{A}\overline{C}D$

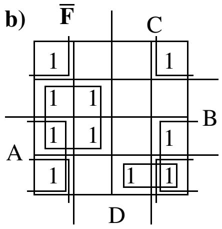
c) $Prime = \overline{X}\overline{Z}, X\overline{Y}, YZ, \overline{Y}\overline{Z}, XZ, \overline{X}Y$
$Essential = none$
$F = (\overline{X}\overline{Z} + X\overline{Y} + YZ) \text{ or } \overline{Y}\overline{Z} + XZ + \overline{X}Y$

2-21.

a)



b)



$\overline{F} = \Sigma m(1, 5, 6, 7, 9, 12, 13, 14)$
$F = \overline{\overline{Y}Z + WX\overline{Z} + \overline{W}XY}$
$F = (Y + \overline{Z})(\overline{W} + \overline{X} + Z)(W + \overline{X} + \overline{Y})$

$\overline{F} = \Sigma m(0, 2, 4, 5, 8, 10, 11, 12, 13, 14)$
$F = \overline{B\overline{C} + \overline{B}\overline{D} + A\overline{D} + A\overline{B}C}$
$F = (\overline{B} + C)(B + D)(\overline{A} + D)(\overline{A} + B + \overline{C})$

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

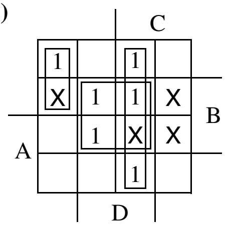
2-23.

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

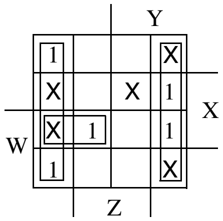
b) s.o.p. $XY + \overline{W}Z + \overline{Y}\overline{Z} + \overline{X}Z$ There are several others.
p.o.s. $(\overline{W} + \overline{X} + Y + \overline{Z})(\overline{W} + X + \overline{Y} + Z)(W + X + \overline{Y} + Z)$

2-24.

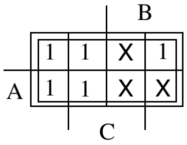
a)



b)



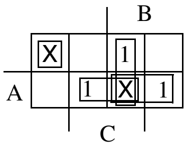
c)



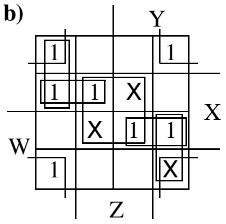
$F = \overline{A}\overline{C}\overline{D} + BD + CD$ $F = WX\overline{Y} + (\overline{Y}\overline{Z} + Y\overline{Z})$ or $(\overline{X}\overline{Z} + X\overline{Z})$ $F = 1$

2-25.*

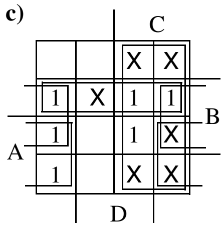
a)



b)



c)

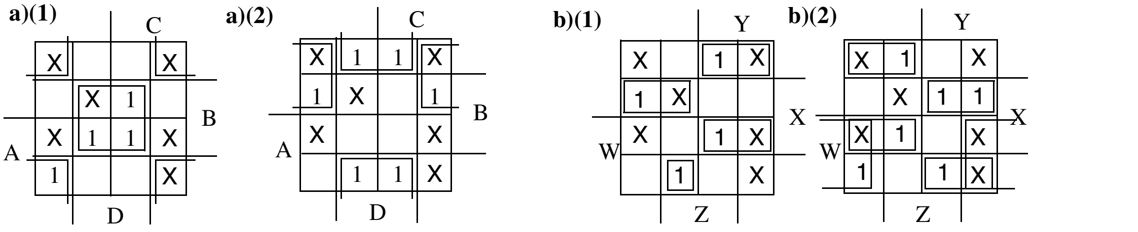


$Primes = AB, AC, BC, \overline{A}\overline{B}\overline{C}$
$Essential = AB, AC, BC$
$F = AB + AC + BC$

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

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

2-26.



$F = BD + (\bar{B}\bar{D} \text{ or } A\bar{D})$

$\bar{F} = \bar{B}D + (\bar{A}\bar{D} \text{ or } B\bar{D})$

$F = (B + \bar{D})((A + D) \text{ or } (\bar{B} + D)$

$F = \bar{W}\bar{X}Y + W\bar{X}\bar{Y}Z + WXY$
$+ (\bar{W}X\bar{Y} \text{ or } \bar{W}\bar{Y}\bar{Z} \text{ or } X\bar{Y}\bar{Z})$

$\bar{F} = \bar{W}XY + WX\bar{Y} + W\bar{Z}$
$+ W\bar{X}Y + (\bar{W}\bar{X}\bar{Y} \text{ or } \bar{W}\bar{Y}Z)$

$F = (W + \bar{X} + \bar{Y})(\bar{W} + \bar{X} + Y)(\bar{W} + Z)(\bar{W} + X + \bar{Y})$
$((W + X + Y) \text{ or } (W + Y + \bar{Z}))$

2-27.

a) $F = A\bar{B}C + \bar{A}BC + A\bar{B}D + \bar{A}BD$

$X_1 = A\bar{B}$

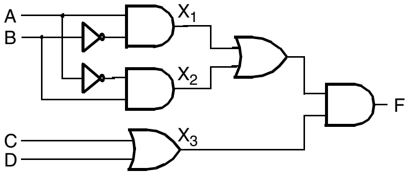
$X_2 = \bar{A}B$

$F = X_1C + X_1D + X_2C + X_2D$

$= (X_1 + X_2)(C + D)$

$X_3 = C + D$

$F = (X_1 + X_2)X_3$



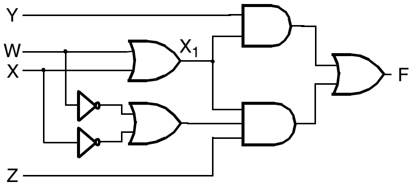
b) $F = WY + XY + \bar{W}XZ + W\bar{X}Z$

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

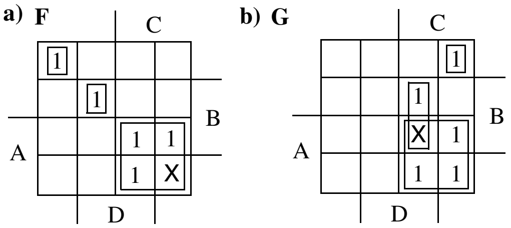
$= (W + X)Y + (W + X)(\bar{W} + \bar{X})Z$

$X_1 = W + X$

$F = X_1Y + X_1(\bar{W} + \bar{X})Z$



2-28.



$F = AC + \bar{A}B\bar{C}D + \bar{A}\bar{B}\bar{C}\bar{D}$

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

$G = AC + BCD + \bar{A}\bar{B}C\bar{D}$

$= AC + (ABCD + \bar{A}BCD) + \bar{A}\bar{B}C\bar{D}$

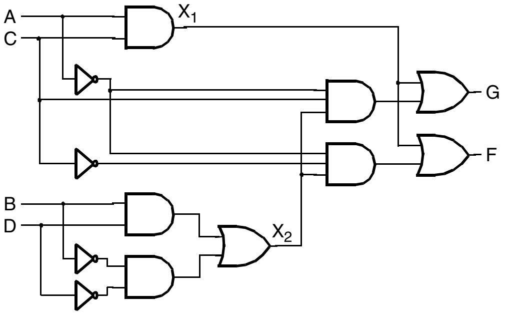
$= AC + \bar{A}C(BD + \bar{B}\bar{D})$

$X_1 = AC$

$X_2 = BD + \bar{B}\bar{D}$

$F = X_1 + \bar{A}\bar{C}X_2$

$G = X_1 + \bar{A}CX_2$



2-29.

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

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

$= AB\overline{C}\overline{D} + ABCD + \overline{B}C\overline{D} + \overline{B}\overline{C}D + \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}\overline{D}$

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

$= WYZ + \overline{X}YZ + X\overline{Y}\overline{Z}$

2-30.*

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

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

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

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

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

$= \overline{X \oplus Y}$

2-31.

$AB\overline{C}D + A\overline{D} + \overline{A}D = AB\overline{C}D + (A \oplus D)$

Note that $X + Y = (X \oplus Y) + XY$

Letting $X = AB\overline{C}D$ and $Y = A \oplus D$,

We can observe from the map below or determine algebraically that XY is equal to 0.

			C	C	
		1	1		
		1	1		B
A	1			1	B
A	1	1		1	
		D	D		

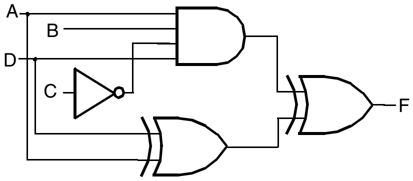
For this situation,

$X + Y = (X \oplus Y) + XY$

$= (X \oplus Y) + 0$

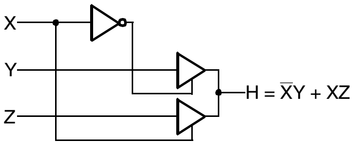
$= X \oplus Y$

So, we can write $F(A, B, C, D) = X \oplus Y = AB\overline{C}D \oplus (A \oplus D)$

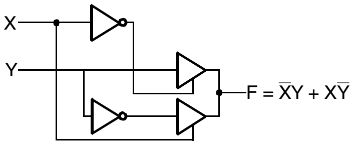


2-32.

a)

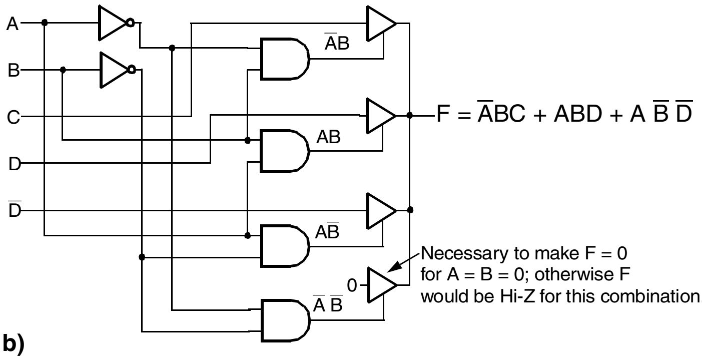


b)



2-33.

a)

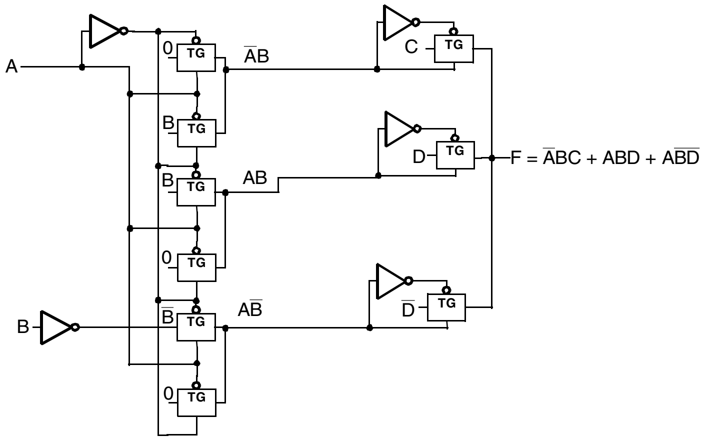

A
B
C
D
$\overline{D}$
$\overline{A}$B
AB
A$\overline{B}$
$\overline{A}$ $\overline{B}$
0
F = $\overline{A}$BC + ABD + A $\overline{B}$ $\overline{D}$

Necessary to make F = 0
for A = B = 0; otherwise F
would be Hi-Z for this combination.

b)

There are no three-state output conflicts.

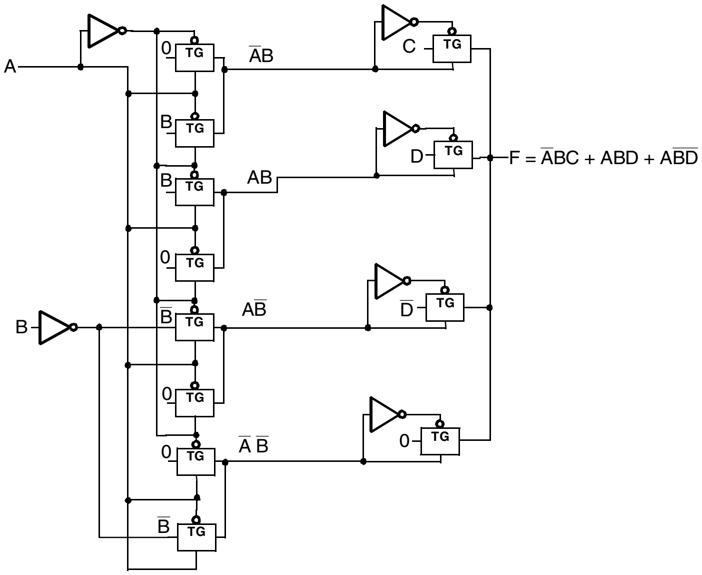
2-34.(Errata: “problem 2-32” should be “problem 2-33”)



2-35.(Errata: “problem 2-33” should be “problem 2-34”)

a) For the solution given in Problem 2-34, the output of F is in the Hi-Z state when A = 0 and B = 0.

b)



PART 2 PROBLEM SOLUTIONS

NOTES ON SOLUTIONS:

1. **Legal Notice:** This publication is protected by United States copyright laws, and is designed exclusively to assist instructors in teaching their courses. It should not be made available to students, or to anyone except the authorized instructor to whom it was provided by the publisher, and should not be sold by anyone under any circumstances. Publication or widespread dissemination (i.e. dissemination of more than extremely limited extracts within the classroom setting) of any part of this material (such as by posting on the World Wide Web) is not authorized, and any such dissemination will violate the United States copyright laws. In consideration of the authors, your colleagues who do not want their students to have access to these materials, and the publisher, please respect these restrictions.

2. **Companion Website Problem Solutions:** The solutions to all problems marked with a * are available to students as well as instructors on the Companion Website.

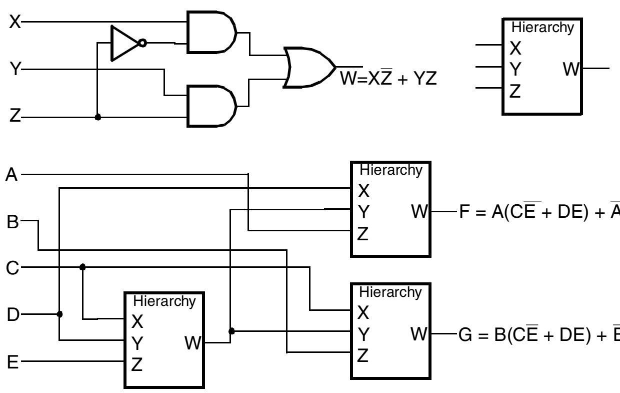
3. **Problem Challenge:** The problems marked with a + are designated as more challenging than the typical problems.

4. **Text Errata Notations:** Text errata are noted at the beginning of a problem if those errata affect either the problem or its solution. These notes indicate only errors identified in the first printing of the 3rd Edition and are expected be removed after the first printing.

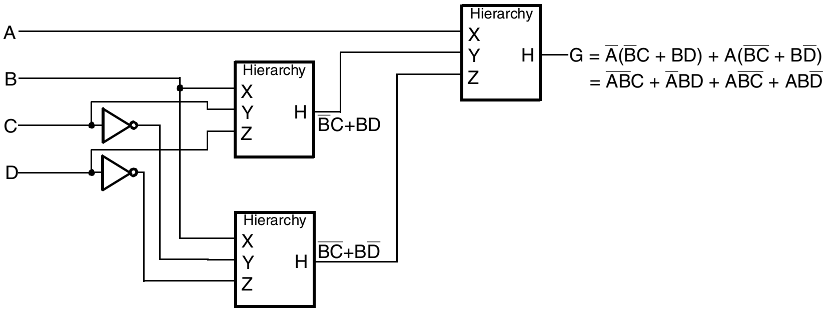
5. **Solutions Errata:** Errata for these solutions will be provided on the Companion Website in the Errata section.

CHAPTER 3

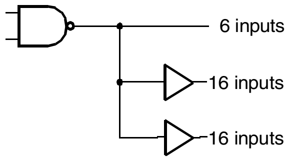
3-1.


X
Y
Z
$W = X\overline{Z} + YZ$
Hierarchy
X
Y
Z
W
A
B
C
D
E
Hierarchy
X
Y
Z
W
$F = A(C\overline{E} + DE) + \overline{A}D$
Hierarchy
X
Y
Z
W
$G = B(C\overline{E} + DE) + \overline{B}C$

3-2.


A
B
C
D
Hierarchy
X
Y
Z
H
$\overline{B}C+BD$
Hierarchy
X
Y
Z
H
$\overline{BC}+B\overline{D}$
Hierarchy
X
Y
Z
H
$G = \overline{A}(\overline{B}C + BD) + A(\overline{BC} + B\overline{D})$
$= \overline{AB}C + \overline{A}BD + A\overline{BC} + AB\overline{D}$

3-3.

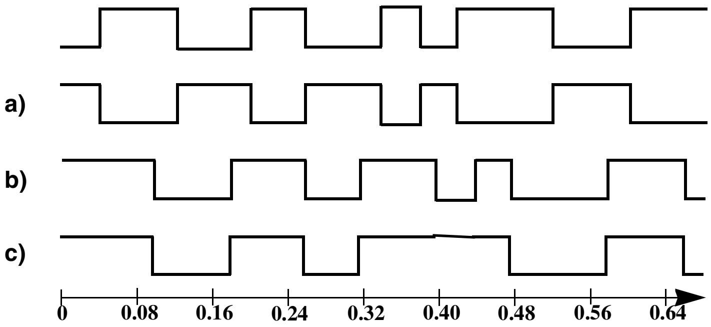


3-4.*

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

0.078 ns + 0.078 ns + 0.052 ns + 0.078 ns = 0.286 ns

3-5.



3-6.

	a)	b)
Input	Delay	Delay
C	1.6ns	1.6ns
D	1.6ns	1.6ns
$\overline{B}$	1.2ns	1.2ns
A	0.8ns	0.8ns
B	0.8ns	0.8ns
$\overline{C}$	0.8ns	0.8ns

c) The values are identical in all cases.

3-7.+

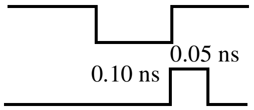
If the rejection time for inertial delays is greater than the propagation delay, then an output change can occur before it can be predicted whether or not it is to occur due to the rejection time.

For example, with a delay of 2 ns and a rejection time of 3 ns, For a 2.5 ns pulse, the initial edge will have already appeared at the output before the 3 ns has elapsed at which whether to reject or not is to be determined.

3-8.⁺

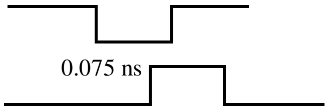
a) The propagation delay is $t_{pd} = (t_{PHL} + t_{PLH})/2 = (0.05 + 0.10)/2 = 0.075$ ns

For a positive output pulse, the following actually occurs:

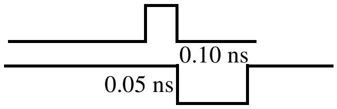


If the input pulse is narrower than 0.05 ns, no output pulse occurs so the rejection time is 0.05 ns.

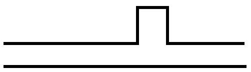
The model projects:



b) For a negative output pulse, the following actually occurs if the input pulse is narrower than the rejection time:



The model projects:

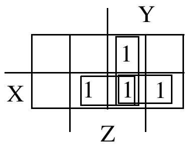


For the negative pulse, the rejection time should be 0.075ns (it can't be larger than the propagation delay) to represent that fact that the pulse is not narrower in the output than in the input.
The parameters provide a poor model in this situation.

3-9.*

				P-Logic				N-Logic			
X	Y	NAND	NOR	X	Y	NAND	NOR	X	Y	NAND	NOR
L	L	H	H	0	0	1	1	1	1	0	0
L	H	H	L	0	1	1	0	1	0	0	1
H	L	H	L	1	0	1	0	0	1	0	1
H	H	L	L	1	1	0	0	0	0	1	1

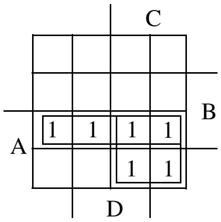
3-10.



$F = XZ + XY + YZ$

This is the same function as the carry for the full adder.

3-11.*



$F = AB + AC$

3-12.

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

$W = A\,B\,\overline{C}\,\overline{D} + A\,\overline{B}\,C\,D$

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

$X = \overline{A}\,B\,C\,D + A\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{D}$

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

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

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

$Z = \overline{A}\,B\,\overline{D} + B\,\overline{C}\,\overline{D} + A\,\overline{B}\,\overline{D}$

3-13.

a)

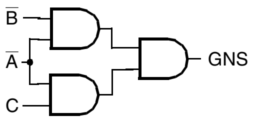
X1	X2	X3	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$Z = X1 \oplus X2 \oplus X3$$

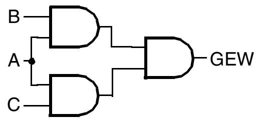
b) There are 2 different functions of Z: $Z = \overline{X1 \oplus X2 \oplus X3}$ and $Z = X1 \oplus X2 \oplus X3$

3-14.+

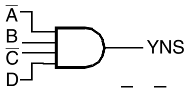
ABCD	GNS	YNS	RNS	GEW	YEW	REW
0000	1	0	0	0	0	1
0001	1	0	0	0	0	1
0011	1	0	0	0	0	1
0010	1	0	0	0	0	1
0110	1	0	0	0	0	1
0111	1	0	0	0	0	1
0101	0	1	0	0	0	1
0100	0	0	1	0	0	1
1100	0	0	1	1	0	0
1101	0	0	1	1	0	0
1111	0	0	1	1	0	0
1110	0	0	1	1	0	0
1010	0	0	1	1	0	0
1011	0	0	1	1	0	0
1001	0	0	1	0	1	0
1000	0	0	1	0	0	1



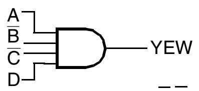
$GNS = \overline{A}C + \overline{A}\overline{B}$



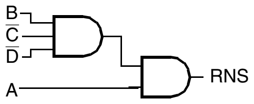
$GEW = AB + AC$



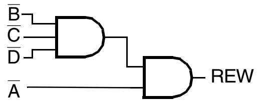
$YNS = \overline{A}B\overline{C}D$



$YEW = A\overline{B}\overline{C}D$



$RNS = A + B\overline{C}\overline{D}$



$REW = \overline{A} + \overline{B}\overline{C}\overline{D}$

3-15.

A	B	C	S5	S4	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1

$S0 = C$

$S1 = 0$

$S2 = \overline{A}B\overline{C} + AB\overline{C}$

$S3 = \overline{A}BC + A\overline{B}C$

$S4 = A\overline{B} + AC$

$S5 = AB$

3-16.+

A	B	C	D	S2	S1	S0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	1	1
1	0	0	1	0	1	1
1	0	1	0	0	1	1
1	0	1	1	0	1	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

$S0 = \overline{B}\,\overline{C}\,D + \overline{B}\,C\,\overline{D} + A\overline{B} + A\overline{C}\,\overline{D} + \overline{A}BCD$

$S1 = \overline{A}B + A\overline{B} + \overline{A}CD + B\overline{C}\,\overline{D}$

$S2 = ABC + ABD$

3-17.

A	B	C	D	W	X	Y	Z
0	0	0	0	0	1	1	0
0	0	0	1	0	1	1	1
0	0	1	0	1	0	0	0
0	0	1	1	1	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	1	1	0	0
0	1	1	1	1	1	0	1
1	0	0	0	1	1	1	0
1	0	0	1	1	1	1	1
1010 to 1111				XXXX			

$W = A + B + C$

$X = \overline{B}\,\overline{C} + BC$

$Y = \overline{C}$

$Z = D$

3-18.

a)

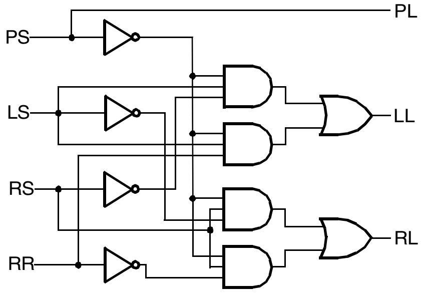
PS	LS	RS	RR	PL	LL	RL
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	0	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

$PL = PS$

$LL = \overline{PS}\ LS\ \overline{RS} + \overline{PS}\ LS\ RR$

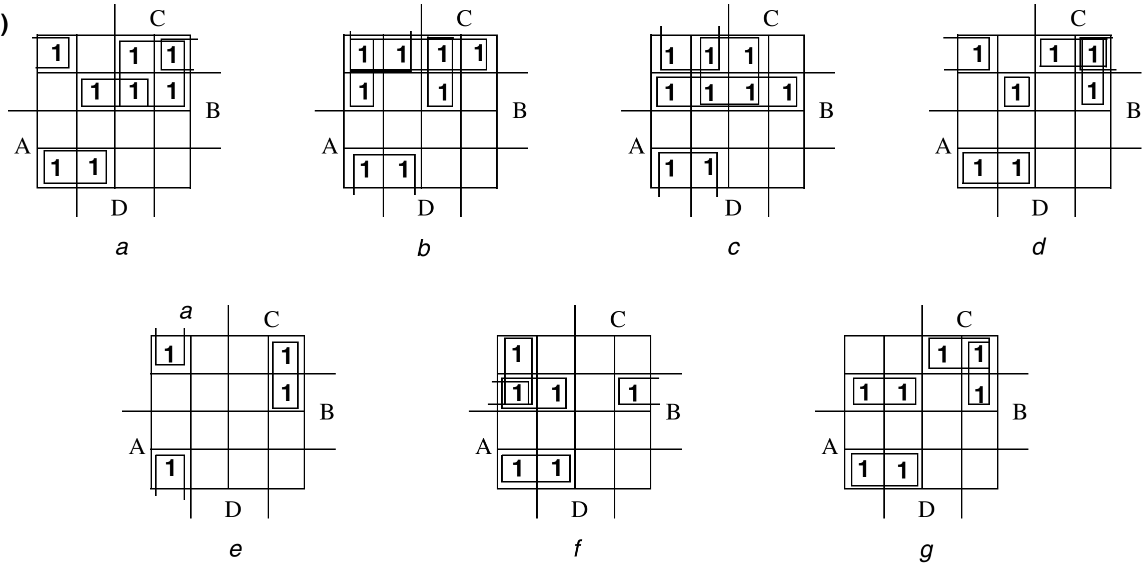
$RL = \overline{PS}\ \overline{LS}\ RS + \overline{PS}\ RS\ \overline{RR}$

b)



3-19.

a)



a *b* *c* *d*

e *f* *g*

b)

$a = \overline{A}C + \overline{A}\,\overline{B}\,\overline{D} + \overline{A}BD + A\overline{B}\,\overline{C}$

$b = \overline{A}\,\overline{B} + \overline{B}\,\overline{C} + \overline{A}\,\overline{C}\,\overline{D} + \overline{A}CD$

$c = \overline{A}B + \overline{B}\,\overline{C} + \overline{A}D$

$d = \overline{A}B\overline{C}D + A\overline{B}\,\overline{C} + \overline{A}\,\overline{B}\,\overline{D} + \overline{A}\,\overline{B}C + \overline{A}C\overline{D}$

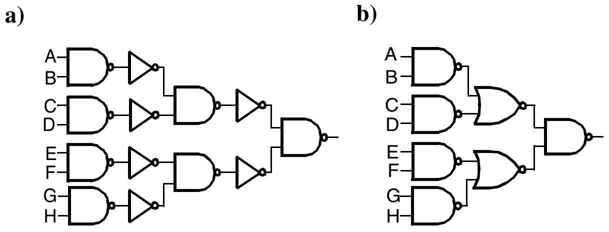
$e = \overline{B}\,\overline{C}\,\overline{D} \; + \overline{A}C\overline{D}$

$f = A\overline{B}\,\overline{C} + \overline{A}B\overline{D} + \overline{A}B\overline{C} + \overline{A}\,\overline{C}\,\overline{D}$

$g = A\overline{B}\,\overline{C} + \overline{A}B\overline{C} + \overline{A}\,\overline{B}C + \overline{A}C\overline{D}$

c) The following gate input counts include input inverters and share AND gates.
Total gate inputs for this solutions = 67. Total gate inputs for book solution is 70. This solution is better by 3 gate inputs.

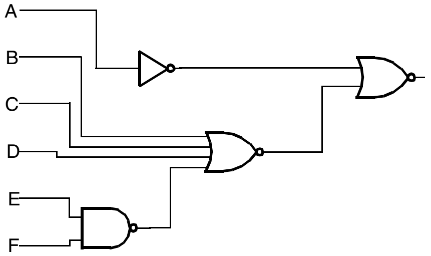
3-20.⁺



c)

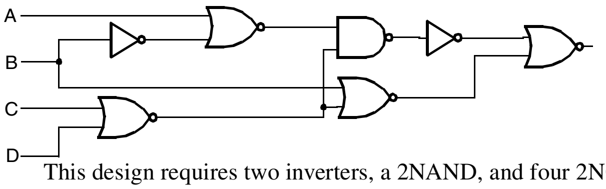
Part b requires 6 fewer gates.

3-21.



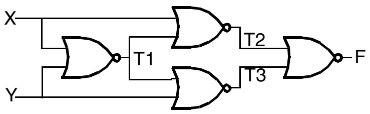
This design requires an inverter, 2NAND, 4NOR, and 2NOR for a total normalized area of
1.0 + 1.25 + 3.25 + 1.25 = 6.75

3-22.



This design requires two inverters, a 2NAND, and four 2NOR for a total normalized area of
2 * 1.0 + 1.25 + 4 * 1.25 = 8.25

3-23.



$T1 = \overline{X}\,\overline{Y}$
$T2 = \overline{X}\,Y$
$T3 = X\,\overline{Y}$
$F = XY + \overline{X}\,\overline{Y}$

3-24.*(Errata: Replace equations with F = $\overline{\text{W}}$ and G = $\overline{\text{W}}\ \overline{\text{Y}}$ + WZ. See Fig. 4-10 for decoder diagram/table.)

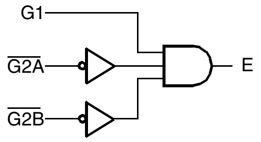
$$F = \overline{\overline{D_{0U}}\cdot\overline{D_{1U}}\cdot\overline{D_{2U}}\cdot\overline{D_{3U}}} = D_{0U}+D_{1U}+D_{2U}+D_{3U} = \overline{W}(\overline{X}\overline{Y}+\overline{X}Y+X\overline{Y}+XY) = \overline{W}$$

$$G = \overline{\overline{D_{0U}}\cdot\overline{D_{2U}}\cdot\overline{D_{1L}}\cdot\overline{D_{2L}}} = D_{0U}+D_{2U}+D_{1L}+D_{3L} = \overline{W}(\overline{X}\overline{Y}+X\overline{Y})+W(\overline{X}Z+XZ) = \overline{W}\overline{Y}+WZ$$

3-25. (Errata: See Fig. 4-10 for decoder diagram/table.)

					Upper				Lower				
W	X	Y	Z	D0	D1	D2	D3	D0	D1	D2	D3	F	G
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	1	1	1	1	1	1	1	1
0	0	1	0	1	0	1	1	1	1	1	1	1	0
0	0	1	1	1	0	1	1	1	1	1	1	1	0
0	1	0	0	1	1	0	1	1	1	1	1	1	1
0	1	0	1	1	1	0	1	1	1	1	1	1	1
0	1	1	0	1	1	1	0	1	1	1	1	1	0
0	1	1	1	1	1	1	0	1	1	1	1	1	0
1	0	0	0	1	1	1	1	0	1	1	1	0	0
1	0	0	1	1	1	1	1	1	0	1	1	0	1
1	0	1	0	1	1	1	1	0	1	1	1	0	0
1	0	1	1	1	1	1	1	1	0	1	1	0	1
1	1	0	0	1	1	1	1	1	1	0	1	0	0
1	1	0	1	1	1	1	1	1	1	1	0	0	1
1	1	1	0	1	1	1	1	1	1	0	1	0	0
1	1	1	1	1	1	1	1	1	1	1	0	0	1

3-26.

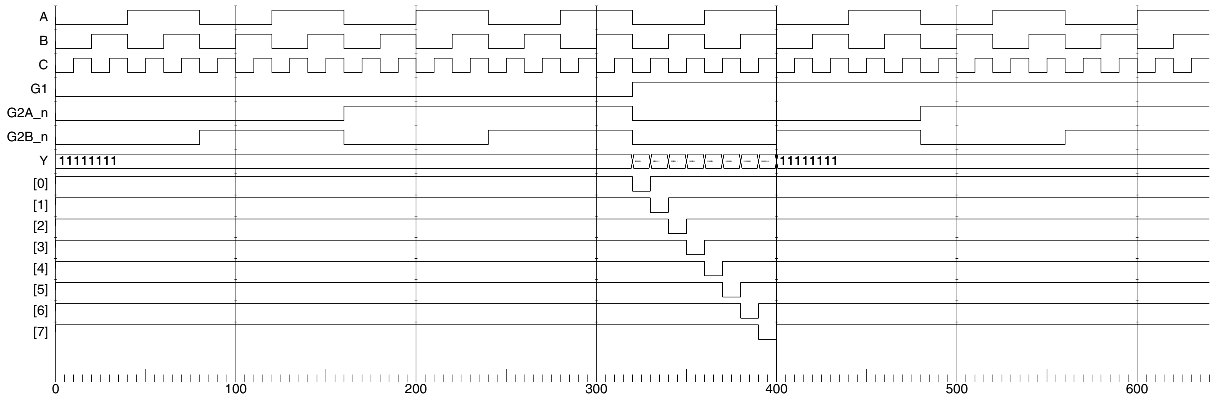


$$E = G1 \cdot \overline{G2A} \cdot \overline{G2B}$$

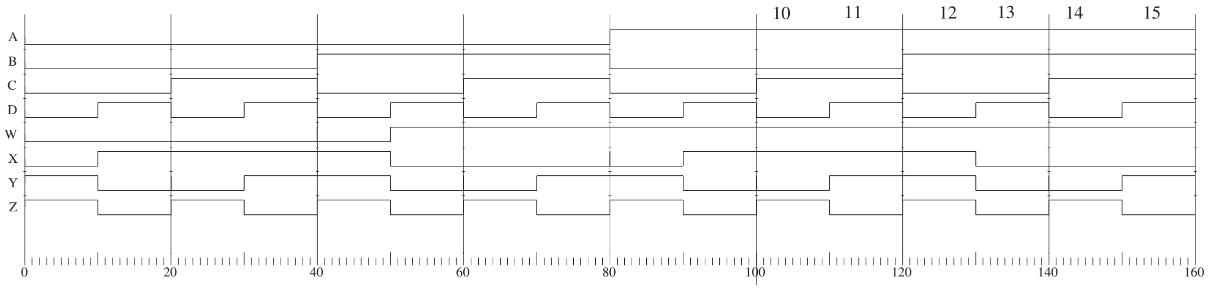
$$Y0 = \overline{\bar{A}\bar{B}\bar{C}E}$$
$$Y1 = \overline{A\bar{B}\bar{C}E}$$
$$Y2 = \overline{\bar{A}B\bar{C}E}$$
$$Y3 = \overline{AB\bar{C}E}$$
$$Y4 = \overline{\bar{A}\bar{B}CE}$$
$$Y5 = \overline{A\bar{B}CE}$$
$$Y6 = \overline{\bar{A}BCE}$$
$$Y7 = \overline{ABCE}$$

Except for G1 = 1 and G2A and G2B = 0, the outputs Y0 through Y7 are all 1's. Otherwise, one of Y0 through Y7 is equal to 0 with all others equal to 1. The output that is equal to 0 has index i = decimal value of the values of (A,B,C) in binary. E.g., if (A,B,C) = (1,1,0), then Y6 = 0.

3-27.

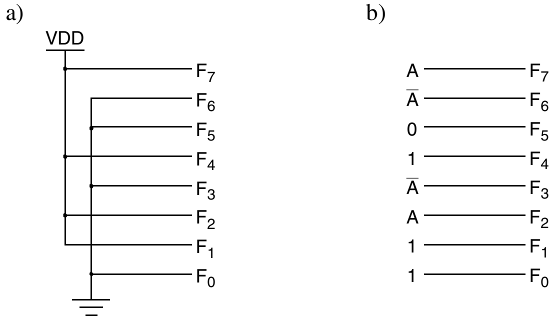


3-28.

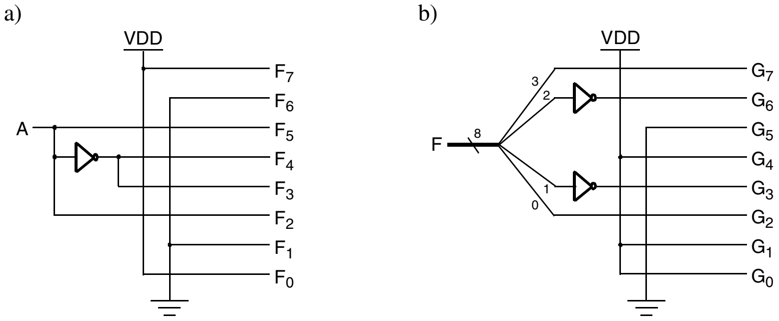


CHAPTER 4

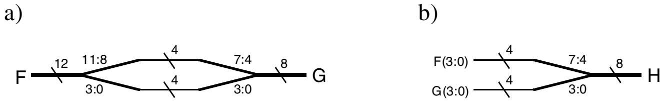
4-1.*



4-2.



4-3.



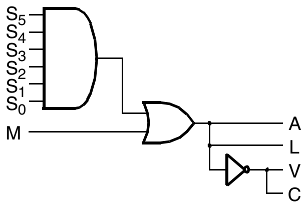
4-4.

$$A = (S_0 \cdot S_1 \cdot S_2 \cdot S_3 \cdot S_4 \cdot S_5) + M$$

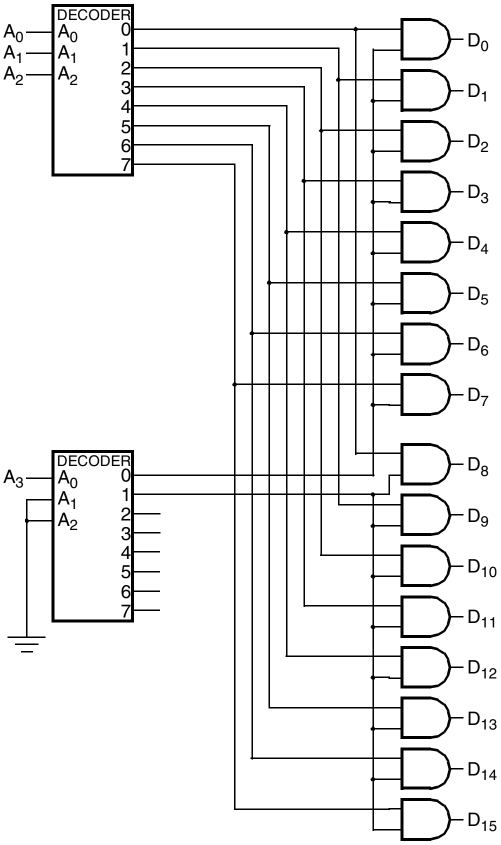
$$L = A$$

$$V = \bar{A} = \overline{(S_0 \cdot S_1 \cdot S_2 \cdot S_3 \cdot S_4 \cdot S_5) + M}$$

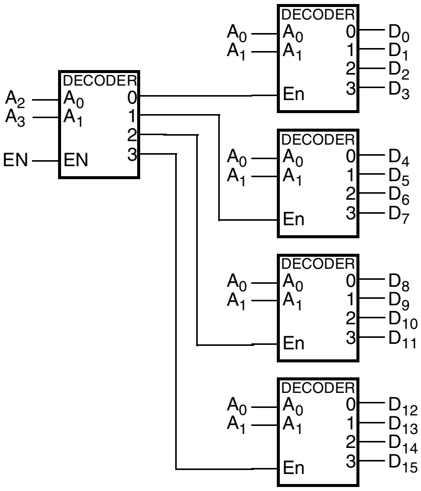
$$C = V$$



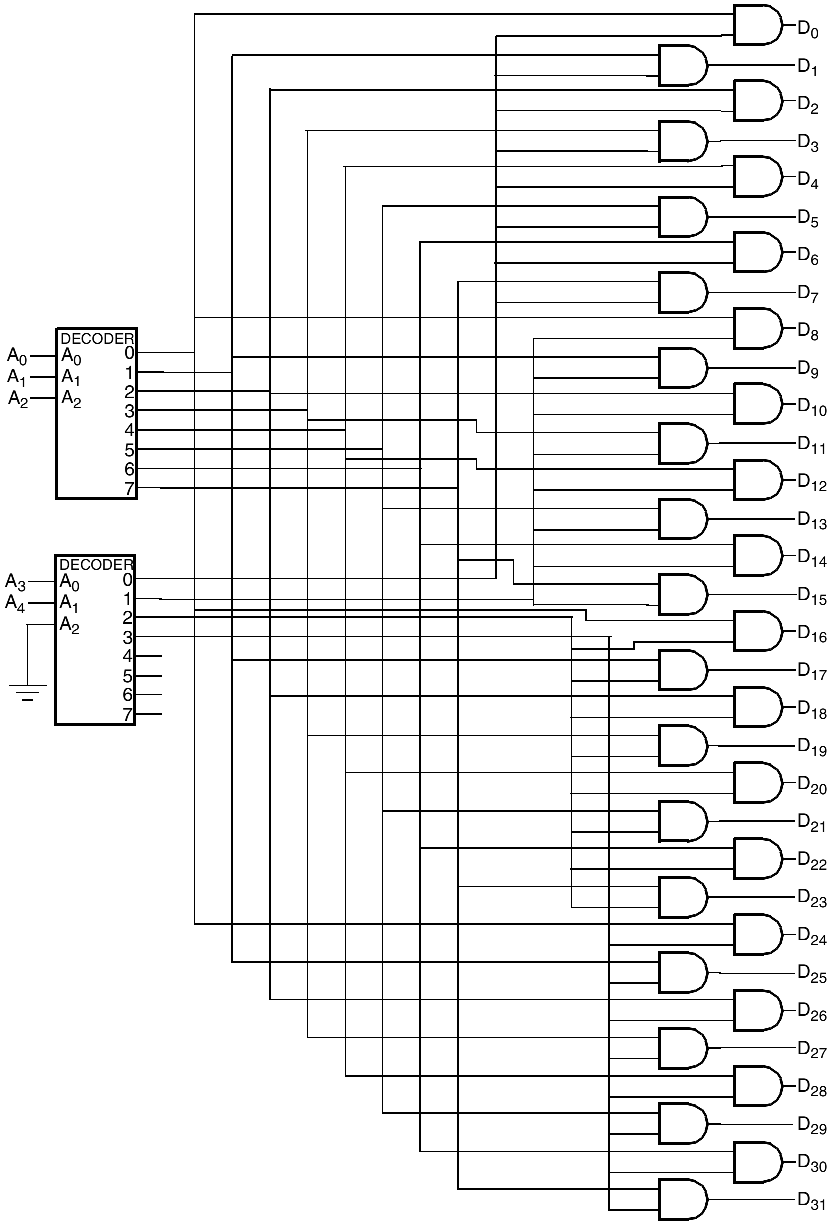
4-5.


DECODER
A0
A1
A2
0
1
2
3
4
5
6
7
D0
D1
D2
D3
D4
D5
D6
D7
DECODER
A3
A0
A1
A2
0
1
2
3
4
5
6
7
D8
D9
D10
D11
D12
D13
D14
D15

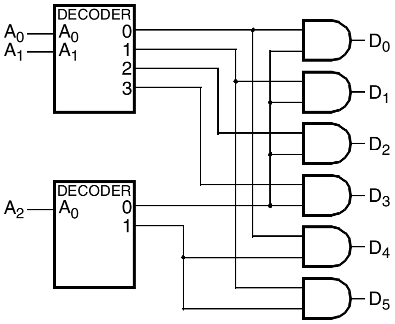
4-6.


DECODER
A2
A3
EN
A0
A1
EN
0
1
2
3
DECODER
A0
A1
En
0
1
2
3
D0
D1
D2
D3
DECODER
D4
D5
D6
D7
DECODER
D8
D9
D10
D11
DECODER
D12
D13
D14
D15

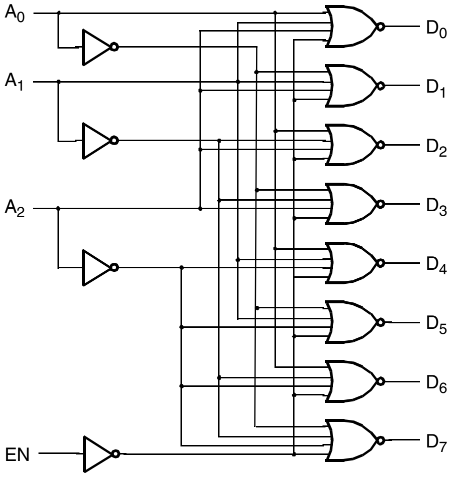
4-7.*(Errata: "four" should be "two" and "48" should be "32")



4-8.



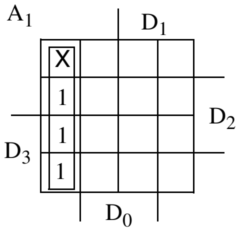
4-9.

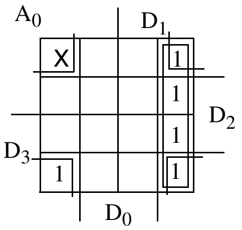


4-10.*

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

$$V = D_0 + D_1 + D_2 + D_3$$
$$A_0 = \overline{D_0}(D_1 + \overline{D_2})$$
$$A_1 = \overline{D_0}\,\overline{D_1}$$





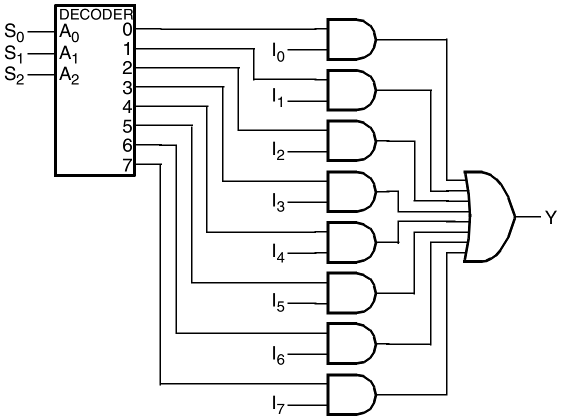
4-11.("BCD" should be "decimal")

9 is the highest priority.

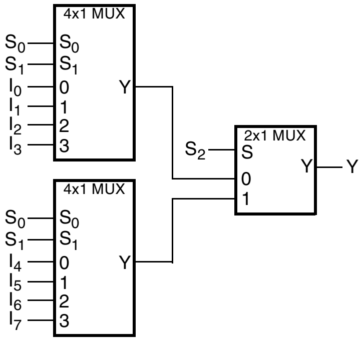
Decimal Inputs										Binary Outputs				
9	8	7	6	5	4	3	2	1	0	A_3	A_2	A_1	A_0	V
0	0	0	0	0	0	0	0	0	0	X	X	X	X	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	0	1	X	0	0	0	1	1
0	0	0	0	0	0	0	1	X	X	0	0	1	0	1
0	0	0	0	0	0	1	X	X	X	0	0	1	1	1
0	0	0	0	0	1	X	X	X	X	0	1	0	0	1
0	0	0	0	1	X	X	X	X	X	0	1	0	1	1
0	0	0	1	X	X	X	X	X	X	0	1	1	0	1
0	0	1	X	X	X	X	X	X	X	0	1	1	1	1
0	1	X	X	X	X	X	X	X	X	1	0	0	0	1
1	X	X	X	X	X	X	X	X	X	1	0	0	1	1

4-12.

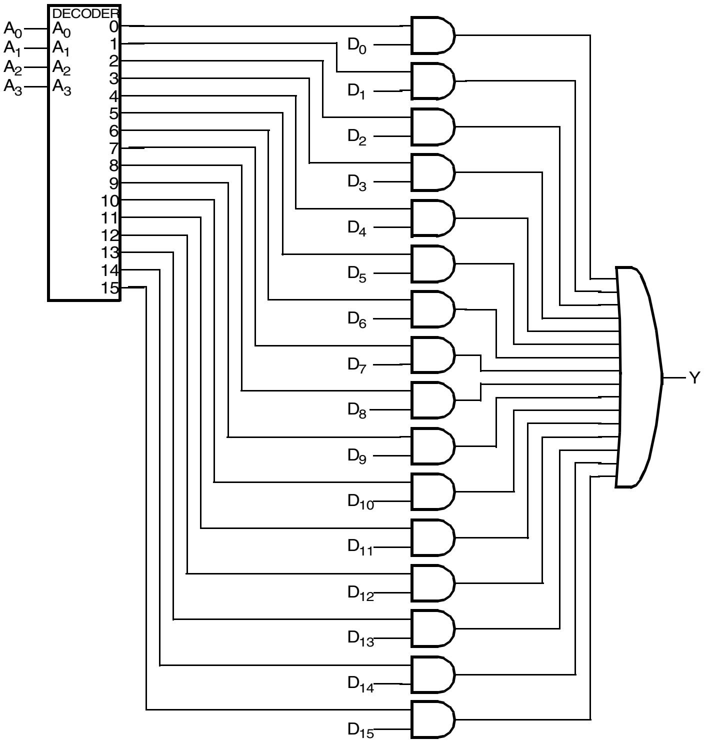
a)


DECODER
S0
S1
S2
A0
A1
A2
0
1
2
3
4
5
6
7
I0
I1
I2
I3
I4
I5
I6
I7
Y

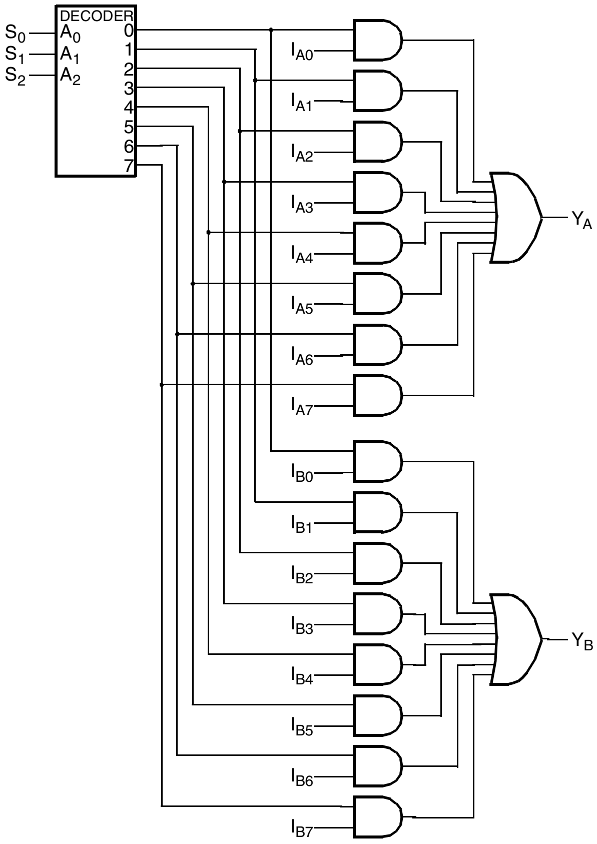
b)


4x1 MUX
S0
S1
I0
I1
I2
I3
0
1
2
3
Y
4x1 MUX
S0
S1
I4
I5
I6
I7
0
1
2
3
Y
S2
2x1 MUX
S
0
1
Y
Y

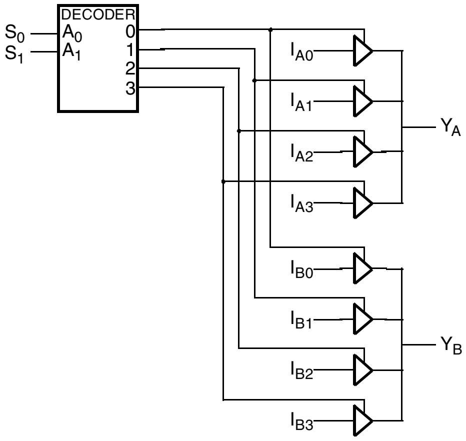
4-13.


DECODER
A0
A1
A2
A3
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
Y

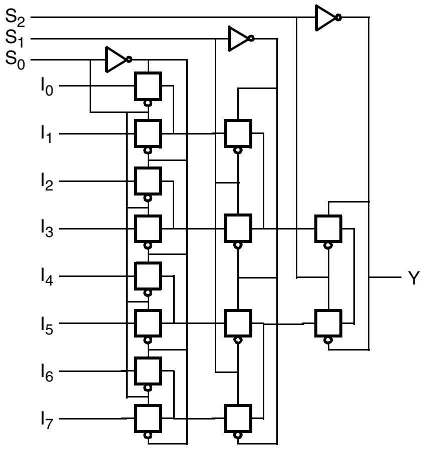
4-14.


DECODER
S0
S1
S2
A0
A1
A2
0
1
2
3
4
5
6
7
IA0
IA1
IA2
IA3
IA4
IA5
IA6
IA7
IB0
IB1
IB2
IB3
IB4
IB5
IB6
IB7
YA
YB

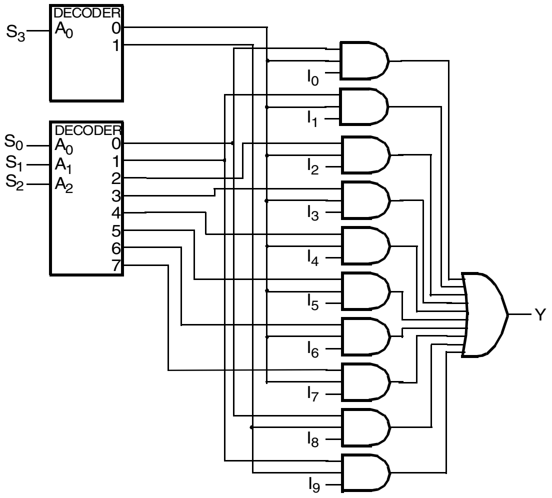
4-15.


DECODER
S0
S1
A0
A1
0
1
2
3
IA0
IA1
IA2
IA3
IB0
IB1
IB2
IB3
YA
YB

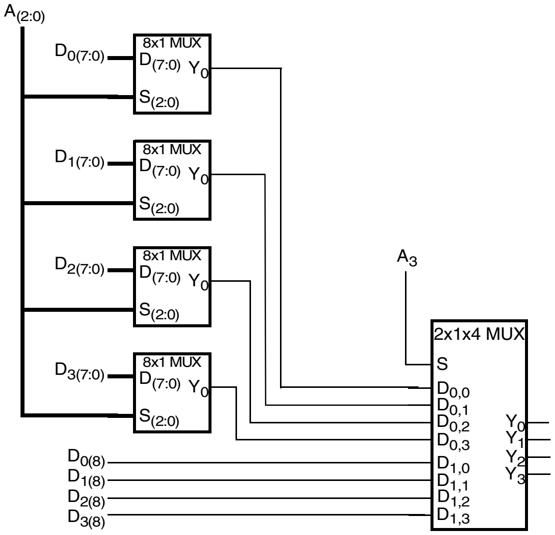
4-16.


S_2
S_1
S_0
I_0
I_1
I_2
I_3
I_4
I_5
I_6
I_7
Y

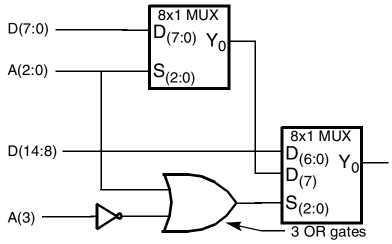
4-17.


DECODER
S_3
A_0
0
1
DECODER
S_0
S_1
S_2
A_0
A_1
A_2
0
1
2
3
4
5
6
7
I_0
I_1
I_2
I_3
I_4
I_5
I_6
I_7
I_8
I_9
Y

4-18.


$A_{(2:0)}$
$D_{0(7:0)}$
8x1 MUX
$D_{(7:0)}$
Y_0
$S_{(2:0)}$
$D_{1(7:0)}$
8x1 MUX
$D_{(7:0)}$
Y_0
$S_{(2:0)}$
$D_{2(7:0)}$
8x1 MUX
$D_{(7:0)}$
Y_0
$S_{(2:0)}$
$D_{3(7:0)}$
8x1 MUX
$D_{(7:0)}$
Y_0
$S_{(2:0)}$
A_3
2x1x4 MUX
S
$D_{0,0}$
$D_{0,1}$
$D_{0,2}$
$D_{0,3}$
$D_{1,0}$
$D_{1,1}$
$D_{1,2}$
$D_{1,3}$
Y_0
Y_1
Y_2
Y_3
$D_{0(8)}$
$D_{1(8)}$
$D_{2(8)}$
$D_{3(8)}$

4-19.*

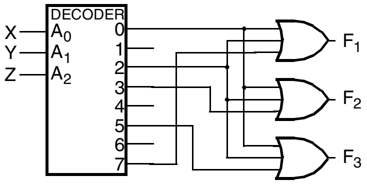


4-20.

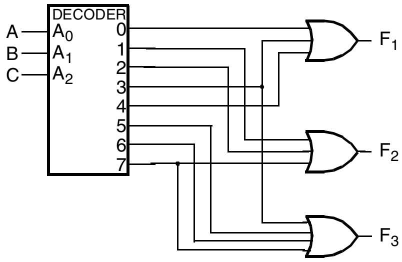
A_1	A_0	E	D_0	D_1	D_2	D_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

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.

4-21.

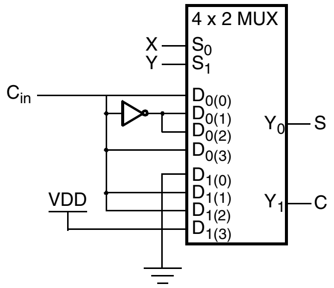


4-22.



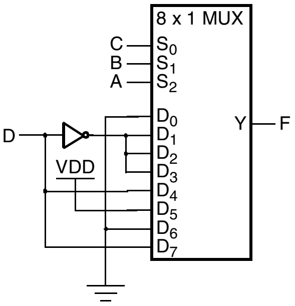
4-23.

X	Y	C_{in}	S	C	
0	0	0	0	0	$S = C_{in}$
0	0	1	1	0	$C = 0$
0	1	0	1	0	$S = \overline{C}_{in}$
0	1	1	0	1	$C = C_{in}$
1	0	0	1	0	$S = \overline{C}_{in}$
1	0	1	0	1	$C = C_{in}$
1	1	0	0	1	$S = C_{in}$
1	1	1	1	1	$C = 1$



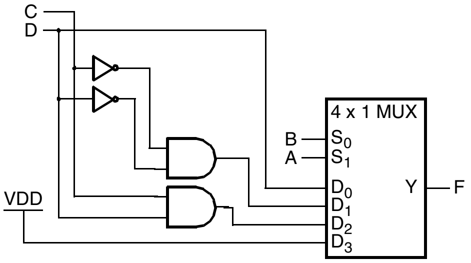
4-24.

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

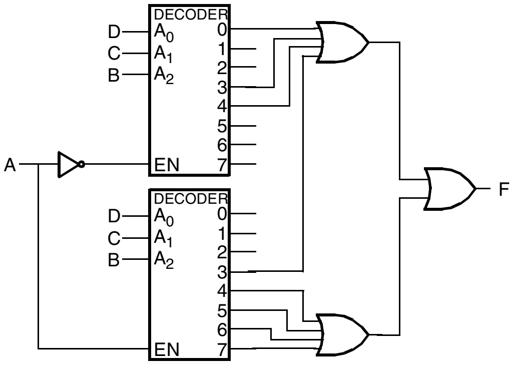


4-25.*

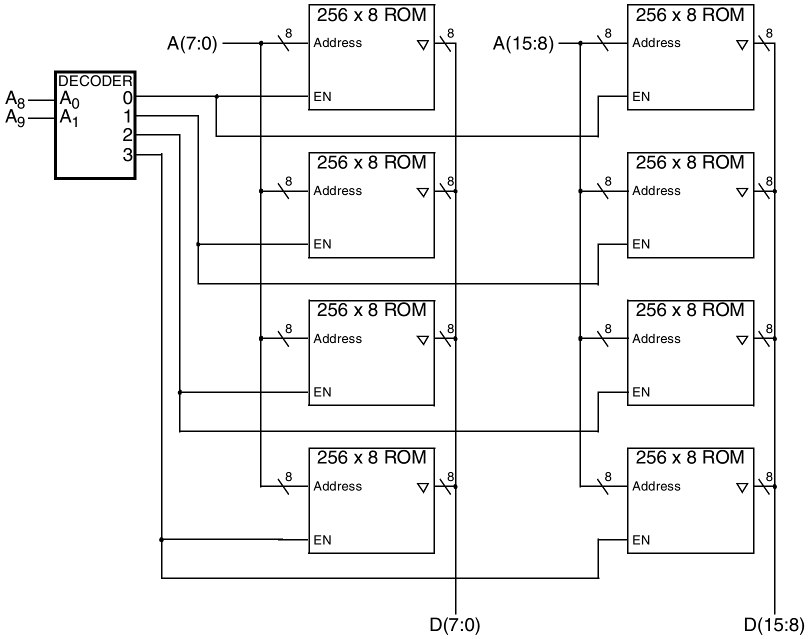
A	B	C	D	F	
0	0	0	0	0	F=D
0	0	0	1	1	
0	0	1	0	0	
0	0	1	1	1	
0	1	0	0	1	$F=\overline{C}\,\overline{D}$
0	1	0	1	0	
0	1	1	0	0	
0	1	1	1	0	
1	0	0	0	0	F=C D
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	1	F=1
1	1	0	1	1	
1	1	1	0	1	
1	1	1	1	1	



4-26.



4-27.



4-28.*

IN	OUT	IN	OUT	IN	OUT	IN	OUT
00000	00 0000	01000	00 1000	10000	01 0110	11000	10 0100
00001	00 0001	01001	00 1001	10001	01 0111	11001	10 0101
00010	00 0010	01010	01 0000	10010	01 1000	11010	10 0110
00011	00 0011	01011	01 0001	10011	01 1001	11011	10 0111
00100	00 0100	01100	01 0010	10100	10 0000	11100	10 1000
00101	00 0101	01101	01 0011	10101	10 0001	11101	10 1001
00110	00 0110	01110	01 0100	10110	10 0010	11110	11 0000
00111	00 0111	01111	01 0101	10111	10 0011	11111	11 0001

4-29.

a) 8 + 8 + 1 + 1 = 18 address bits and 8 + 1 = 9 output bits, 256K × 9

b) 64K × 16 c) To represent maximum input 9999, 14 output bits are needed, 64K × 14

4-30.

Input			Output			
X	Y	Z	A	B	C	D
0	0	0	1	0	0	0
0	0	1	1	0	0	1
0	1	0	1	1	1	1
0	1	1	0	1	0	1
1	0	0	0	1	0	0
1	0	1	0	1	0	1
1	1	0	1	1	1	1
1	1	1	1	0	0	1

4-31.

PTERMS		INPUTS			OUTPUTS			
		X	Y	Z	$\overline{A}$	$\overline{B}$	$\overline{C}$	$\overline{D}$
$X\,\overline{Y}$	1	1	0	–	1	–	1	–
$\overline{X}\,\overline{Y}$	2	0	0	–	–	1	1	–
$\overline{X}\,Y\,Z$	3	0	1	1	1	–	1	–
$X\,Y\,Z$	4	1	1	1	–	1	1	–
$Y\,\overline{Z}$	5	–	0	0	–	–	–	1

4-32.*

PTERMS		INPUTS			OUTPUTS					
		X	Y	Z	A	$\overline{B}$	$\overline{C}$	D	E	$\overline{F}$
XY	1	1	1	–	1	–	1	–	–	–
$\overline{X}$	2	0	–	–	–	1	–	–	–	–
$Y\,\overline{Z}$	3	–	1	0	–	1	–	1	–	–
$\overline{X}\,\overline{Y}$	4	0	0	–	–		1	–	–	–
$\overline{Z}$	5	–	–	0	–	–	1	–	–	1

4-33.

PTERM		INPUTS				OUTPUTS			
		A	B	C	D	$\overline{W}$	X	Y	Z
$\overline{A}\,\overline{B}\,D$	1	0	0	–	1	1	1	–	–
$\overline{A}\,\overline{B}\,C$	2	0	0	1	–	1	1	–	–
$\overline{A}\,\overline{C}\,\overline{D}$	3	0	-	0	0	1	–	–	–
$B\,\overline{C}\,\overline{D}$	4	–	0	1	-	–	1	–	–
AD	5	–	0	-	1	–	1	–	–
C D	7	–	–	1	1	–	–	1	–
$\overline{C}\,\overline{D}$	8	–	–	0	0	–	–	1	–
$\overline{D}$									1

4-34.*

Assume 3-input OR gates.

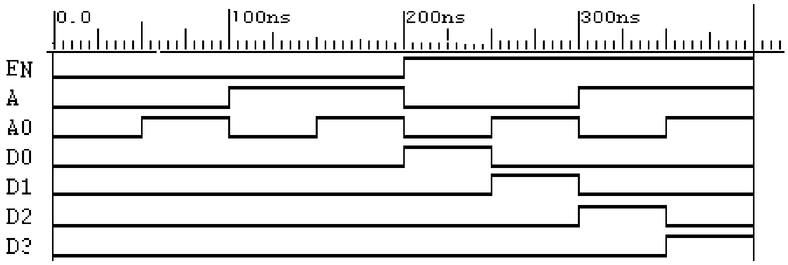
PTERM		INPUTS			
		A	B	C	D
A	1	1	–	-	–
BC	2	–	1	1	–
BD	3	–	1-	-	1
$\overline{B}C$	4	–	0	1	–
$\overline{B}D$	5	–	0	-	1
$B\overline{C}\overline{D}$	6	–	1	0	0
CD	7	–	–	1	1
$\overline{C}\,\overline{D}$	8	–	–	0	0
-	-	–	–	–	–
$\overline{D}$	9	–	–	–	0
-	-	–	–	–	–
-	-	–	–	–	–

4-35.

Assume 3-input OR gates.

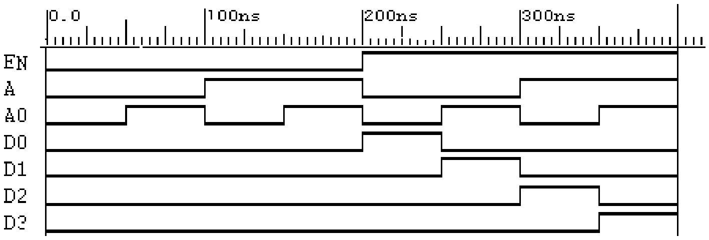
PTERM		INPUTS			
		X	Y	Z	A
$Y\overline{Z}$	1	–	1	0	–
$\overline{X}\,\overline{Y}Z$	2	0	0	1	–
$X\overline{Z}$	3	1	-	0	–
$\overline{X}\,\overline{Y}$	4	0	0	–	–
YZ	5	–	1	1	–
XY	6	1	1	–	–
A	7	–	–	–	1
XY	8	1	1	–	–
–	9	–	–	–	–
Z	10	–	–	1	–
$\overline{X}Y$	11	0	1	–	–
–	12	–	–	–	–

4-36.

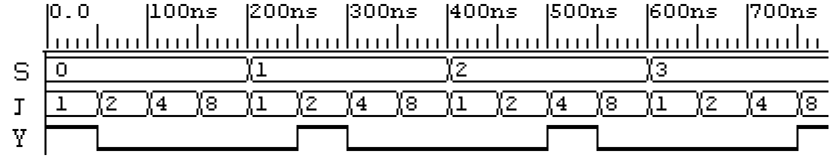


4-37.

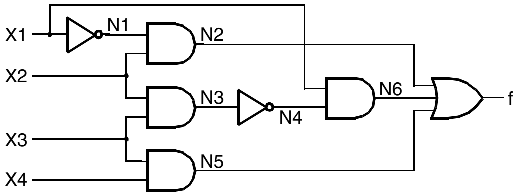
```
entity decoder_2_to_4 is
   port(EN: in std_logic;
         A: in std_logic_vector(0 to 1);
         D: out std_logic_vector(0 to 3));
end decoder_2_to_4;
...
signal not_A: std_logic_vector(0 to 1);
begin
   g0: NOT1 port map (A(0), not_A(0));
   g1: NOT1 port map (A(1), not_A(1));
   g2: NAND3 port map (not_A(0), not_A(1), EN, D(0));
   g3: NAND3 port map (A(0), not_A(1), EN, D(1));
   g4: NAND3 port map (not_A(0), A(1), EN, D(2));
   g5: NAND3 port map (A(0), A(1), E, D(3));
```



4-38.



4-39.*

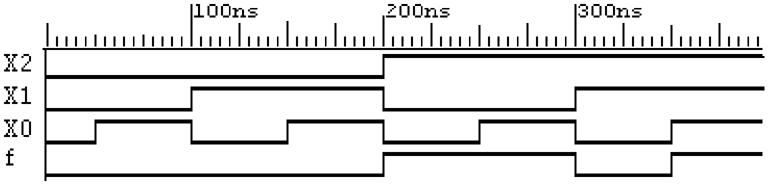


4-40.

```
-- Figure 4-40: Structural VHDL Description
library ieee;
use ieee.std_logic_1164.all;
entity nand2 is
   port(in1, in2: in std_logic;
       out1 : out std_logic);
end nand2;

architecture concurrent of nand2 is
begin
   out1 <= not (in1 and in2);
end architecture;

library ieee;
use ieee.std_logic_1164.all;
entity nand3 is
   port(in1, in2, in3 : in std_logic;
       out1 : out std_logic);
end nand3;

architecture concurrent of nand3 is
begin
   out1 <= not (in1 and in2 and in3);
end concurrent;

library ieee;
use ieee.std_logic_1164.all;
entity nand4 is
   port(in1, in2, in3, in4: in std_logic;
       out1 : out std_logic);
end nand4;
-- The code above this point could be eliminated by using the library, func_prims.

library ieee;
use ieee.std_logic_1164.all;
entity fig440 is
   port(X: in std_logic_vector(0 to 2);
       f: out std_logic);
end fig440;
architecture structural_2 of fig440 is

component NAND2
   port(in1, in2: in std_logic;
       out1: out std_logic);
end component;
```

```
component NAND3
   port(in1, in2, in3: in std_logic;
       out1: out std_logic);
end component;

signal T: std_logic_vector(0 to 4);
begin
      g0: NAND2 port map (X(0),X(1),T(0));
      g1: NAND2 port map (X(0),T(0),T(1));
      g2: NAND2 port map (X(1),T(0),T(2));
      g3: NAND3 port map (X(2),T(1),T(2),T(3));
      g4: NAND2 port map (X(2),T(2),T(4));
      g5: NAND2 port map (T(3),T(4),f);
end structural_2;
```

$$F = X_0X_2 + \overline{X}_1X_2$$

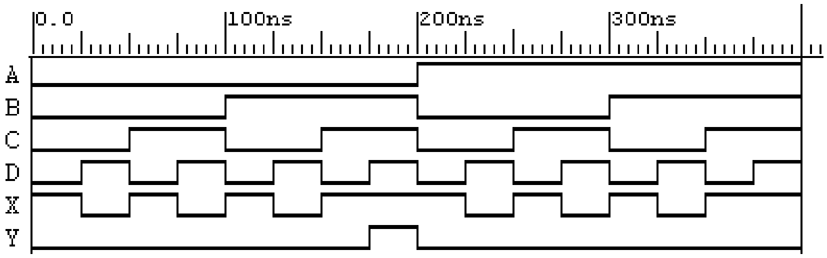


4-41.

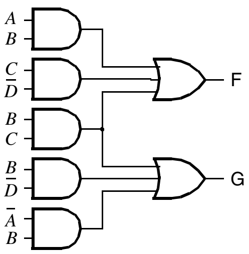
```
begin
   g0: NOT_1 port map (D, x1);
   g1: AND_2 port map (B, C, x2);
   g2: NOR_2 port map (A, x1, x3);
   g3: NAND_2 port map (x1, x3, x4);
   g4: OR_2 port map (x1, x2, x5);
   g5: AND_2 port map (x4, x5, X);
   g6: AND_2 port map (x3, x5, Y);
end structural_1;
```

$X = \overline{D} + BC$

$Y = \overline{A}\ BCD$



4-42.

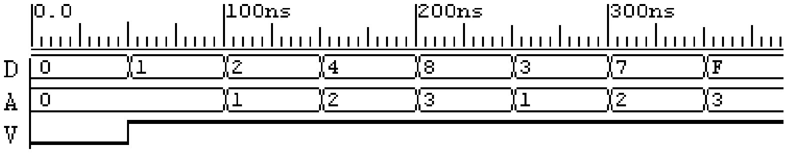


4-43.*

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

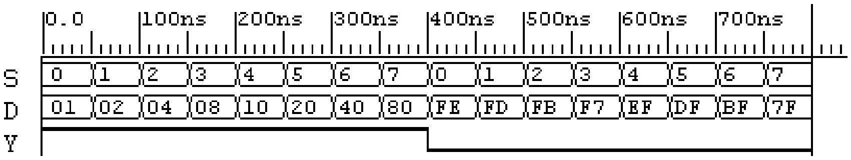
4-44.+

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

entity priority_4 is
    port (
        D: in STD_LOGIC_VECTOR (3 downto 0);
        A: out STD_ULOGIC_VECTOR (1 downto 0);
        V: out STD_LOGIC
        );
end priority_4;

architecture priority_4_arch of priority_4 is
begin
    V <= '0' when D ="0000" else '1';
    A <=  "11" when D(3) = '1' else
        "10" when D(2) = '1' else
        "01" when D(1) = '1' else
        "00" when D(0) = '1' else "00";

end priority_4_arch;
```

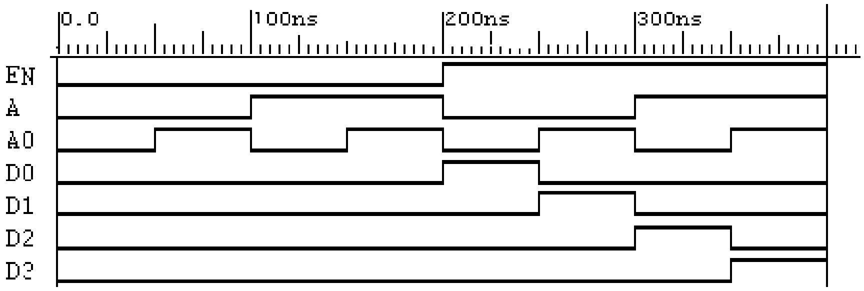


4-45.

```
tity multiplexer_8_to_1 is
   port(S: in std_logic_vector(2 downto 0);
        D: in std_logic_vector(7 downto 0);
        Y: out std_logic);
d multiplexer_8_to_1;

chitecture function_table of multiplexer_8_to_1 is
gnal not_S: std_logic_vector(0 to 1);
gnal N: std_logic_vector(0 to 3);
gin
  with S select
   Y <=    D(0) when "000"
        D(1) when "001"
        D(2) when "010"
        D(3) when "011"
        D(4) when "100"
        D(5) when "101"
        D(6) when "110"
        D(7) when "111"
        'X';
d function_table;
```

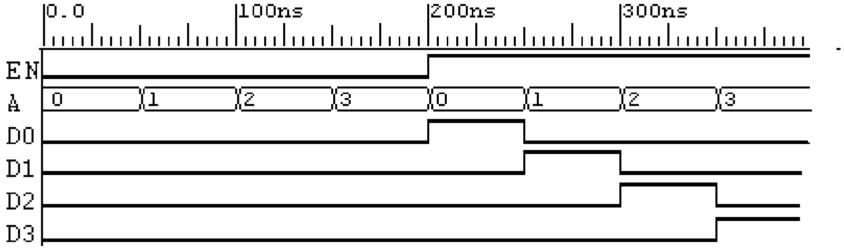


4-46.*

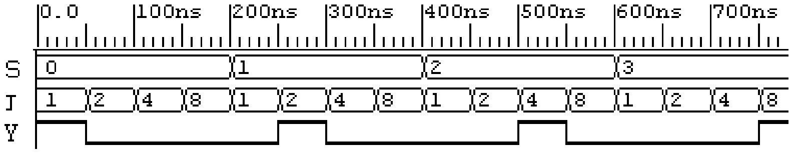


4-47.

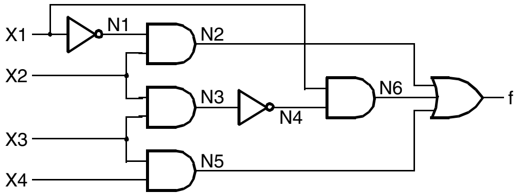
```
module decoder_2_to_4_st(A, EN, D);
  input [1:0] A;
  input EN;
  output [3:0] D;

  wire [1:0] not_A;

  not
    go(not_A[0], A[0]),
    g1(not_A[1], A[1]);

  nand
    g2(D[0], not_A[0], not_A[1], EN),
    g3(D[1],A[0], not_A[1], EN),
    g4(D[2], not_A[0], A[1], EN),
    g5(D[3], A[0],A[1], EN);
endmodule

end structural_1;
```



4-48.

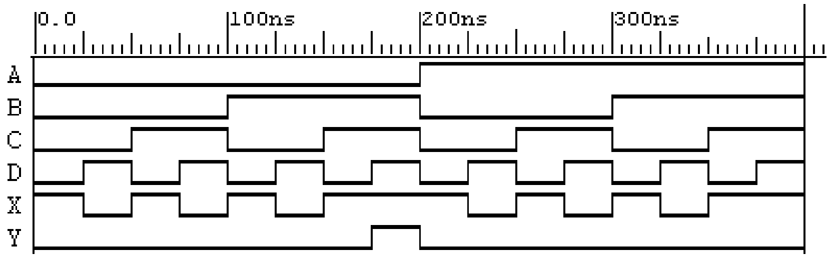


4-49.*



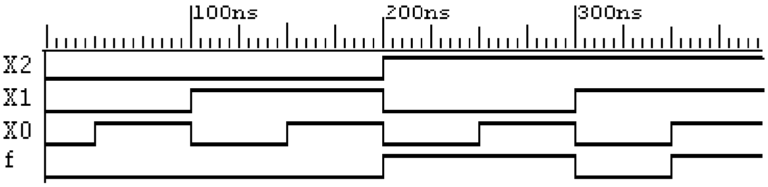
4-50.

```
module circuit_4_50(A, B, C, D, X, Y);
      input A, B, C, D;
      output X, Y;

      wire n1, n2, n3, n4, n5;

      not
            go(n1, D);

      nand
            g1(n4, n1, n3);

      and
            g2(n2, B, C),
            g3(X, n4, n5),
            g4(Y, n3, n5);

      or
            g5(n5, n1, n2);

      nor
            g6(n3, n1, A);

endmodule
```

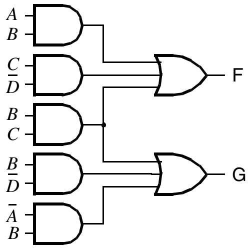


4-51.

```
module circuit_4_51(X, F);
      input [2:0] X;
      output F;

      wire [0:4] T;

      nand
            g0(T[0],X[0],X[1]),
            g1(T[1],X[0],T[0]),
            g2(T[2],X[1],T[0]),
            g3(T[3],X[2],T[1],T[2]),
            g4(T[4],X[2],T[2]),
            g5(F,T[3],T[4]);
endmodule
```



4-52.

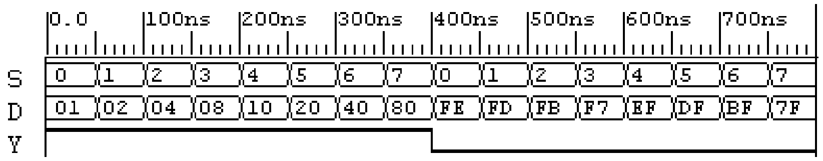


4-53.*

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

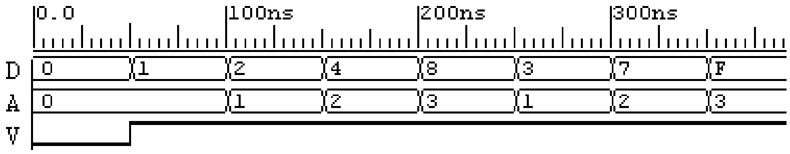
4-54.

```
module multiplexer_8_to_1_cf_v(S, D, Y);
  input [2:0] S;
  input [7:0] D;
  output Y;

  assign Y = (S == 3'b000) ? D[0] :
         (S == 3'b001) ? D[1] :
         (S == 3'b010) ? D[2] :
         (S == 3'b011) ? D[3] :
         (S == 3'b100) ? D[4] :
         (S == 3'b101) ? D[5] :
         (S == 3'b110) ? D[6] :
         (S == 3'b111) ? D[7] : 1'bx ;
endmodule
```



4-55.+

```
module prioencoder_4_to_1_(D, A, V);
  input [3:0] D;
  output [1:0] A;
  output V;

  assign V = D[0] | D[1] | D[2] | D[3];

  assign A[0] = D[3] ? 1'b1:(D[2] ? 1'b0:(D[1] ? 1'b1:(D[0] ? 1'b0:1'bx)));
  assign A[1] = D[3] ? 1'b1:(D[2] ? 1'b1:(D[1] ? 1'b0:(D[0] ? 1'b0:1'bx)));
endmodule
```



PART 2 PROBLEM SOLUTIONS

NOTES ON SOLUTIONS:

1. **Legal Notice:** This publication is protected by United States copyright laws, and is designed exclusively to assist instructors in teaching their courses. It should not be made available to students, or to anyone except the authorized instructor to whom it was provided by the publisher, and should not be sold by anyone under any circumstances. Publication or widespread dissemination (i.e. dissemination of more than extremely limited extracts within the classroom setting) of any part of this material (such as by posting on the World Wide Web) is not authorized, and any such dissemination will violate the United States copyright laws. In consideration of the authors, your colleagues who do not want their students to have access to these materials, and the publisher, please respect these restrictions.

2. **Companion Website Problem Solutions:** The solutions to all problems marked with a * are available to students as well as instructors on the Companion Website.

3. **Problem Challenge:** The problems marked with a + are designated as more challenging than the typical problems.

4. **Text Errata Notations:** Text errata are noted at the beginning of a problem if those errata affect either the problem or its solution. These notes indicate only errors identified in the first printing of the 3rd Edition and are expected be removed after the first printing.

5. **Solutions Errata:** Errata for these solutions will be provided on the Companion Website in the Errata section.

CHAPTER 5

5-1.

$$
\begin{aligned}
{}^*S_0 &= C_0\overline{A_0}\,\overline{B_0} + \overline{C_0}A_0\overline{B_0} + \overline{C_0}\,\overline{A_0}B_0 + C_0A_0B_0 \\
S_1 &= C_1\overline{A_1}\,\overline{B_1} + \overline{C_1}A_1\overline{B_1} + \overline{C_1}\,\overline{A_1}B_1 + C_1A_1B_1 \\
{}^*S_1 &= C_0A_0\overline{A_1}\,\overline{B_1} + A_0B_0\overline{A_1}\,\overline{B_1} + C_0B_0\overline{A_1}\,\overline{B_1} \\
&\quad + \overline{C_0}\,\overline{A_0}A_1\overline{B_1} + \overline{A_0}\,\overline{B_0}A_1\overline{B_1} + \overline{C_0}\,\overline{B_0}A_1\overline{B_1} \\
&\quad + \overline{C_0}\,\overline{A_0}\,\overline{A_1}B_1 + \overline{A_0}\,\overline{B_0}\,\overline{A_1}B_1 + \overline{C_0}\,\overline{B_0}\,\overline{A_1}B_1 \\
&\quad + C_0A_0A_1B_1 + A_0B_0A_1B_1 + C_0B_0A_1B_1
\end{aligned}
$$

$$
\begin{aligned}
C_1 &= C_0A_0 + A_0B_0 + C_0B_0 \\
C_2 &= C_1A_1 + A_1B_1 + C_1B_1 \\
{}^*C_2 &= C_0A_0A_1 + A_0B_0A_1 + C_0B_0A_1 \\
&\quad + C_0A_0B_1 + A_0B_0B_1 + C_0B_0B_1 + A_1B_1
\end{aligned}
$$

* These are the three equations for the outputs. The logic diagram consists of a sum-of-products implementation of these equations.

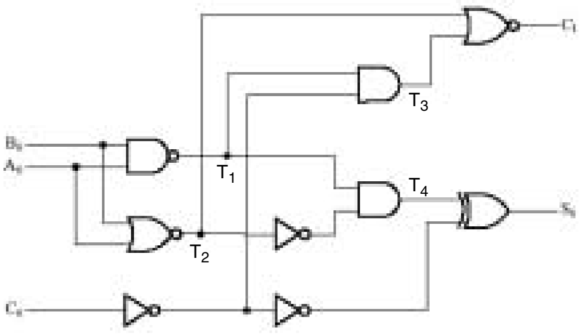
5-2.*

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

$$C_1 = A_0B_0 + A_0C_0 + B_0C_0$$

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

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



5-3.*

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

5-4.

a) $\begin{array}{r} 11111 \\ +\ \underline{10000} \\ 01111 \end{array}$ b) $\begin{array}{r} 10110 \\ +\ \underline{10001} \\ 00111 \end{array}$ c) $\begin{array}{r} 1011110 \\ +\ \underline{0100010} \\ 0000000 \end{array}$ d) $\begin{array}{r} 000101 \\ +\ \underline{011000} \\ 011101 \\ = \ -100011 \end{array}$

5-5.

a)	11111	b)	10110	c)	1011110	d)	111101
+	10000	+	00001	+	0100010	+	011000
	01111		10111		0000000		010101

Overflow on Complement
Overflow on Subtract

5-6.*

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

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

5-7.

	a)			**b)**			**c)**			**d)**	
	100111	-25		001011	11		110001	-15		101110	-18
+	111001	-7	+	100110	-26	+	101110	-18	+	001001	9
	100000	-32		110001	-15		011111	-33		110111	-9
							Overflow				

5-8.+

a) $H = D$

$G = C \oplus D$

$F = \overline{B}C + \overline{B}D + B\overline{C}\,\overline{D}$

$E = \overline{A}B + A\overline{B}\,\overline{C}\,\overline{D} + \overline{A}C + \overline{A}D$

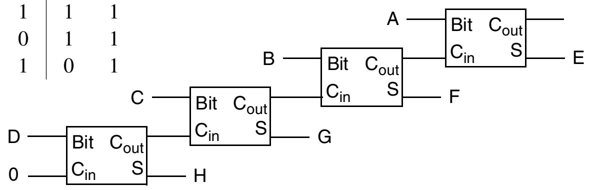
A	B	C	D	E	F	G	H
0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	1	1	0	1
0	1	0	0	1	1	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	0	1	0
0	1	1	1	1	0	0	1
1	0	0	0	1	0	0	0
1	0	0	1	0	1	1	1
1	0	1	0	0	1	1	0
1	0	1	1	0	1	0	1
1	1	0	0	0	1	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	0	1	0
1	1	1	1	0	0	0	1

b)

Bit	Cin	S	Cout
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

$S = Bit \oplus Cin$

$Cout = Bit + Cin$



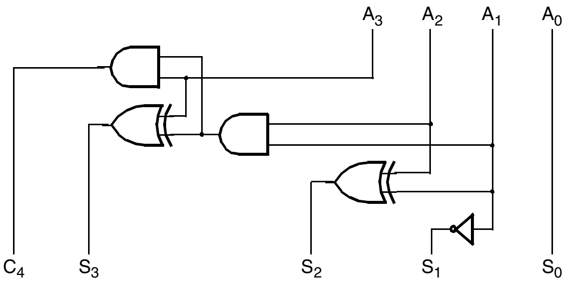
c) $H = D$

$G = C \oplus D$

$F = B \oplus (C + D)$

$E = A \oplus (B + C + D)$

5-9.



5-10.

$B_0 = 0$ $\qquad$ $C_0 = 0$

$S_0 = A_0 \oplus S \oplus S = A_0$

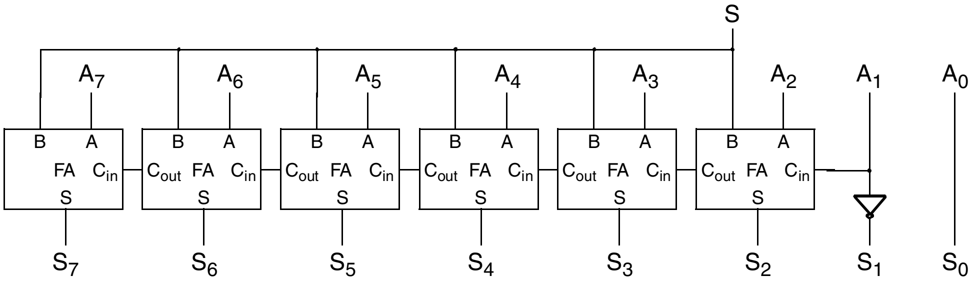
$C_1 = A_0(B_0 \oplus S) + (B_0 \oplus S)C_0 + A_0C_0 = A_0S + S \cdot S + S \cdot S = S$

$B_1 = 1$

$S_1 = A_1 \oplus \bar{S} \oplus S = \overline{A_1}$ $\qquad$ $C_2 = A_1\bar{S} + A_1S + S \cdot \bar{S} = A_1$

$B_{2-7} = S$

Bits 2-6 use regular full adder/subtractor logic. For bit 7, the carry logic is omitted.



5-11.+

a) $B_0 = B_2 = B_3 = 0$

$S_0 = A_0 \oplus B_0 \oplus C_0 = A_0 \oplus C_0$ $\qquad$ $P_0 = A_0 \oplus B_0 = A_0$ $\qquad$ $G_0 = A_0 \cdot B_0 = 0$

$C_1 = G_0 + P_0C_0 = A_0C_0$

$B_1 = 1$

$S_1 = A_1 \oplus 1 \oplus C_1 = \overline{A_1} \oplus C_1$ $\qquad$ $P_1 = A_1 \oplus 1 = \overline{A_1}$ $\qquad$ $G_1 = A_1 \cdot 1 = A_1$

$C_2 = G_1 + P_1(G_0 + P_0C_0) = A_1 + \overline{A_1}A_0C_0 = A_1 + A_0C_0$

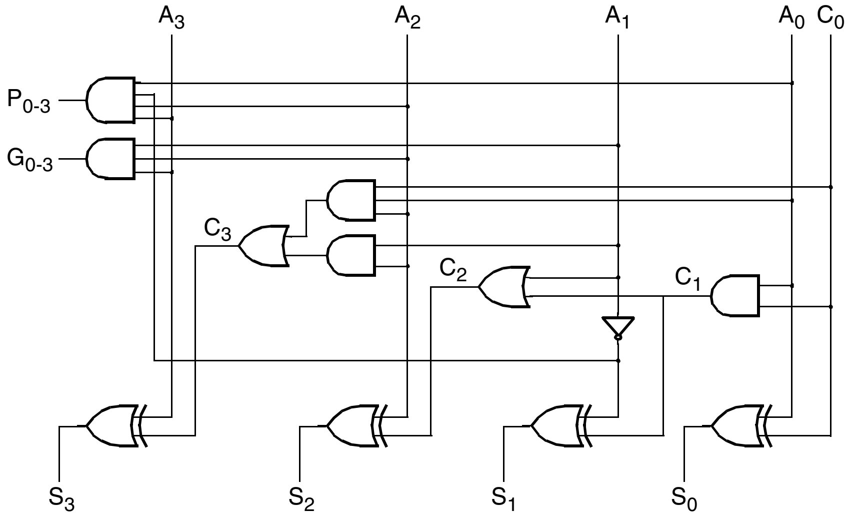
$S_2 = A_2 \oplus C_2$ $\qquad$ $P_2 = A_2$ $\qquad$ $G_2 = 0$

$C_3 = G_2 + P_2(G_1 + P_1(G_0 + P_0C_0)) = A_2A_1 + A_2\overline{A_1}A_0C_0 = A_2A_1 + A_2A_0C_0$

a)

$S_3 = A_3 \oplus C_3 \qquad\qquad P_3 = A_3 \qquad\qquad G_3 = 0$

$$P_{0-3} = P_3P_2P_1P_0 = A_3A_2\overline{A}_1A_0$$
$$G_{0-3} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 = A_3A_2A_1$$



b) $B_0 = B_1 = B_2 = B_3 = 0$

$S_0 = A_0 \oplus B_0 \oplus C_0 = A_0 \oplus C_0 \qquad P_0 = A_0 \oplus B_0 = A_0 \qquad G_0 = A_0 \cdot B_0 = 0$

$C_1 = G_0 + P_0C_0 = A_0C_0$

$S_1 = A_1 \oplus C_1 \qquad\qquad P_1 = A_1 \qquad\qquad G_1 = 0$

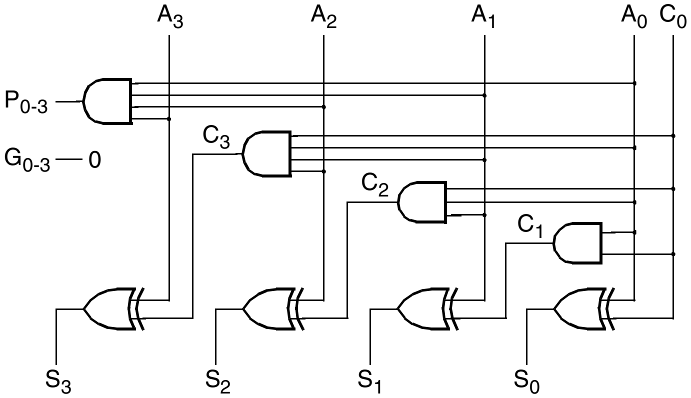
$C_2 = G_1 + P_1(G_0 + P_0C_0) = A_1A_0C_0$

$S_2 = A_2 \oplus C_2 \qquad\qquad P_2 = A_2 \qquad\qquad G_2 = 0$

$C_3 = G_2 + P_2(G_1 + P_1(G_0 + P_0C_0)) = A_2A_1A_0C_0$

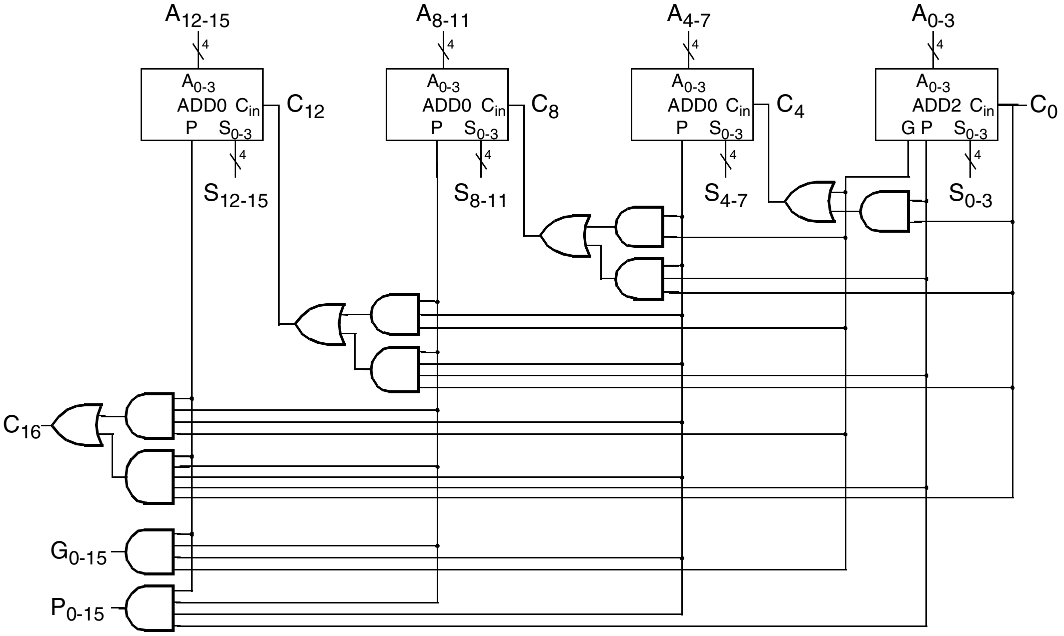
$S_3 = A_3 \oplus C_3 \qquad\qquad P_3 = A_3 \qquad\qquad G_3 = 0$

$$P_{0-3} = P_3P_2P_1P_0 = A_3A_2A_1A_0$$
$$G_{0-3} = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 = 0$$



c) $$\begin{aligned} C_4 &= G_{0-3} + P_{0-3}C_0 \\ C_8 &= G_{4-7} + P_{4-7}C_4 = P_{4-7}G_{0-3} + P_{4-7}P_{0-3}C_0 \\ C_{12} &= G_{8-11} + P_{8-11}C_8 = P_{8-11}P_{4-7}G_{0-3} + P_{8-11}P_{4-7}P_{0-3}C_0 \\ C_{16} &= G_{12-15} + P_{12-15}C_{12} = P_{12-15}P_{8-11}P_{4-7}G_{0-3} + P_{12-15}P_{8-11}P_{4-7}P_{0-3}C_0 \end{aligned}$$

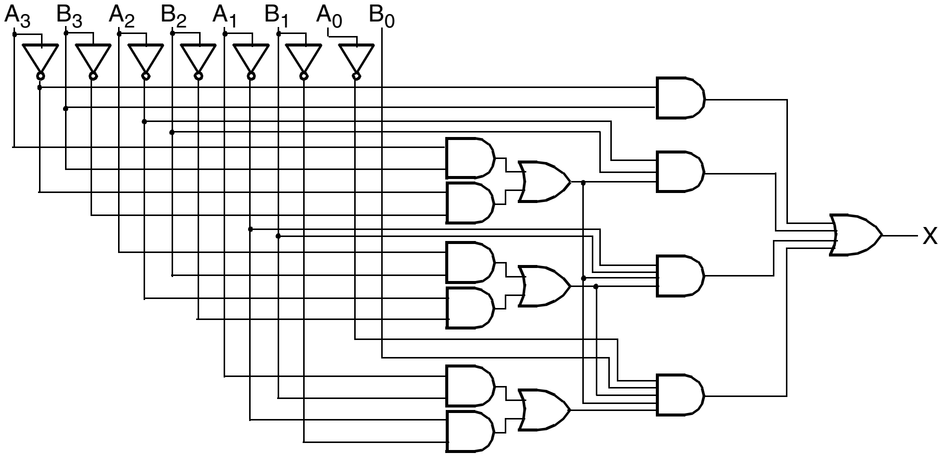
$$\begin{aligned} P_{0-15} &= P_{0-3}P_{4-7}P_{8-11}P_{12-15} \\ G_{0-15} &= G_{12-15} + P_{12-15}G_{8-11} + P_{12-15}P_{8-11}G_{4-7} + P_{12-15}P_{8-11}P_{4-7}G_{0-3} \\ &= P_{12-15}P_{8-11}P_{4-7}G_{0-3} \end{aligned}$$



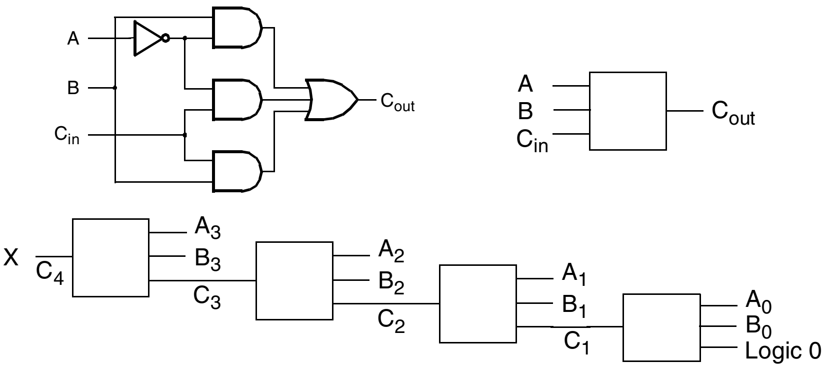
5-12.

Proceeding from MSB to LSB: $A < B$ if $A_i < B_i(\bar{A}_i B_i = 1)$ and for all j > i, $A_j = B_j(A_jB_j + \bar{A}_j\bar{B}_j = 1)$
Based on the above,

$$\begin{aligned} X = {} & \bar{A}_3B_3 + (A_3B_3 + \bar{A}_3\bar{B}_3)\bar{A}_2B_2 + (A_3B_3 + \bar{A}_3\bar{B}_3)(A_2B_2 + \bar{A}_2\bar{B}_2)\bar{A}_1B_1 \\ & + (A_3B_3 + \bar{A}_3\bar{B}_3)(A_2B_2 + \bar{A}_2\bar{B}_2)(A_1B_1 + \bar{A}_1\bar{B}_1)\bar{A}_0B_0 \end{aligned}$$



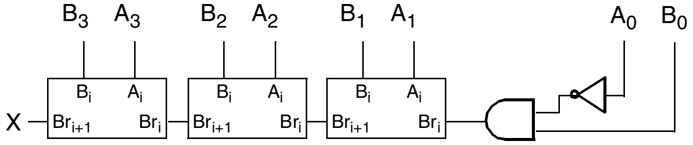
5-13.



5-14.

In a subtractor, the sum is replaced by the difference and the carry is replaced by the borrow. The borrow at any given point is a 1 only if in the LSB direction from that point, $A < B$.

Only borrow logic is needed to produce X, so the difference logic is discarded in contraction.The remaining equation for borrow into the i + 1 position is: $Br_{i+1} = \overline{A}_i\, B_i + \overline{A}_i\, Br_i + B_i\, Br_i$ for i = 0,1,2,3. $Br_0 = 0$ giving $Br_1 = \overline{A}_0 B_0$ When the borrow $Br_4 = 1$, then $A < B$. Thus, $X = Br_4$. The resulting circuit using the borrow logic is:



5-15.+

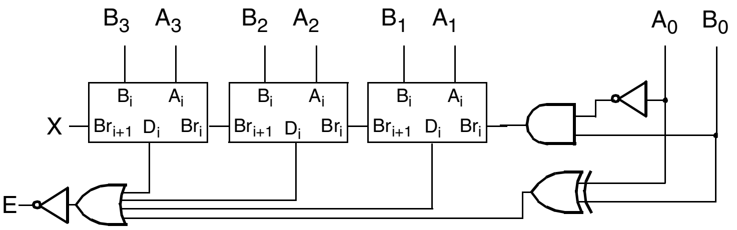
X=1 for $A < B$, X = 0 otherwise. E=1 for $A = B$, E = 0 otherwise. Use the carry logic from problem 5-14 to produce X, with the addition of difference logic to find E:

$$D_i = A_i \oplus B_i \oplus Br_i$$

$$D_0 = A_0 \oplus B_0 \oplus 0 = A_0 \oplus B_0$$

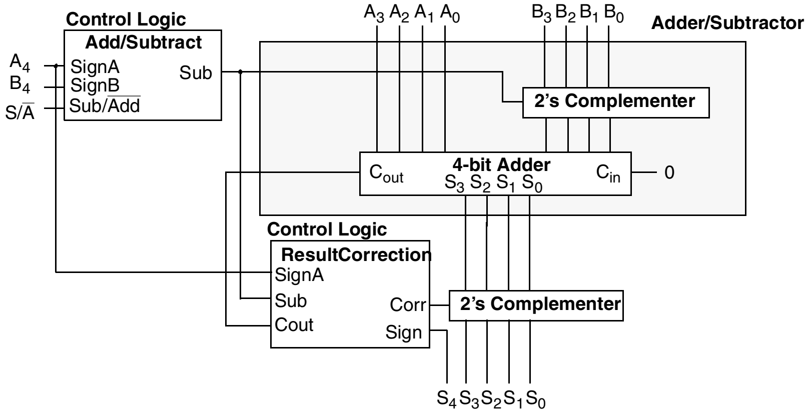
Bits 1-3 use regular subtractor logic.

$$E = \overline{D_3}\,\overline{D_2}\,\overline{D_1}\,\overline{D_0} = \overline{D_3 + D_2 + D_1 + D_0}$$



5-16.+

Circuit Diagram:



Control Logic Truth Tables

Inputs				Inputs					
S/$\overline{A}$	SignA	SignB	Sub	Sub	SignA	Cout	Corr	Sign	Over-flow
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0	1
0	1	0	1	0	1	0	0	1	0
0	1	1	0	0	1	1	0	1	1
1	0	0	1	1	0	0	1	1	0
1	0	1	0	1	0	1	0	0	0
1	1	0	0	1	1	0	1	0	0
1	1	1	1	1	1	1	0	1	0

Sub = S/$\overline{A}$ ⊕ SignA ⊕ SignB

Corr = Sub $\overline{\text{Cout}}$

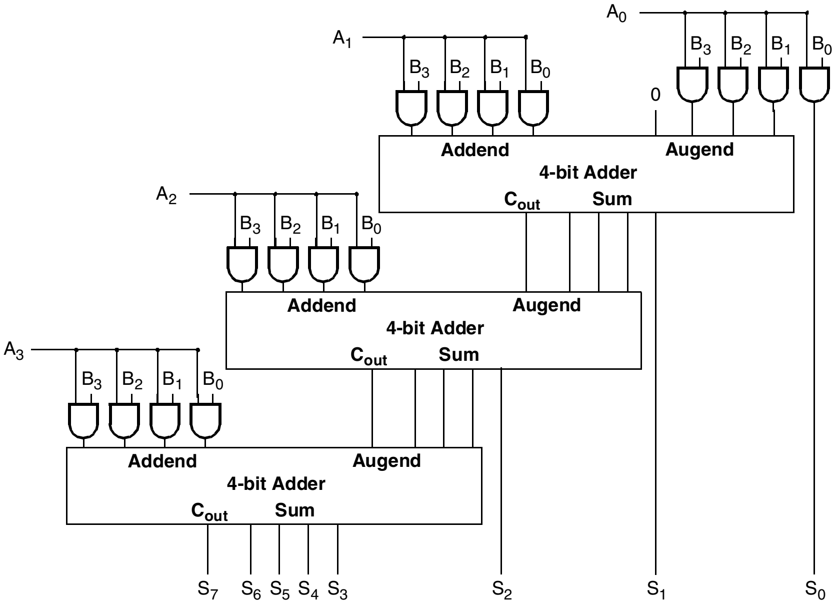
Sign = SignA ⊕ Corr

Overflow = $\overline{\text{Sub}}$ Cout

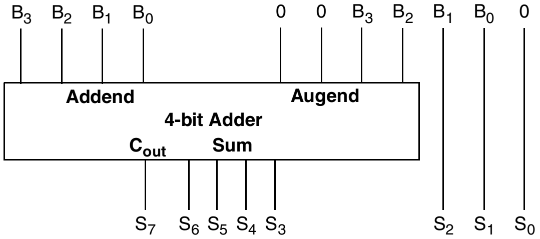
5-17.*

	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

5-18.*

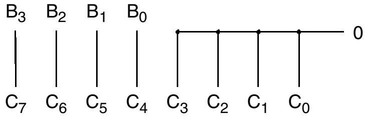


5-19.

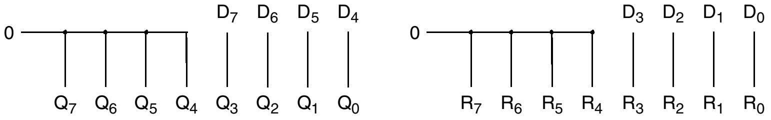


5-20.+

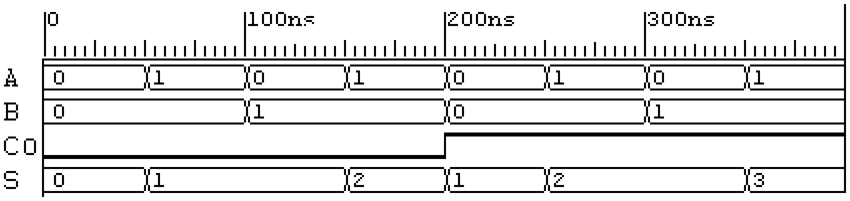
a)

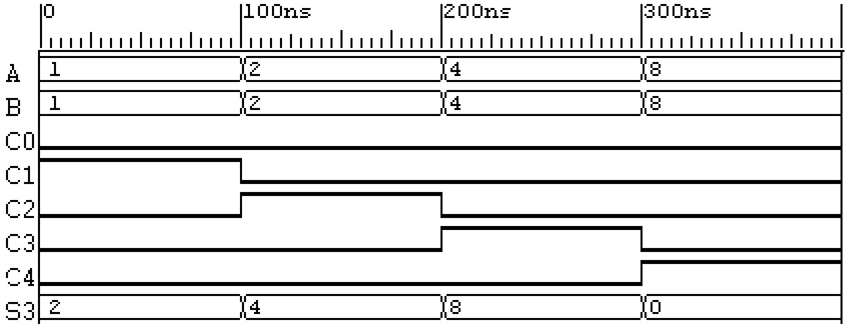


b)

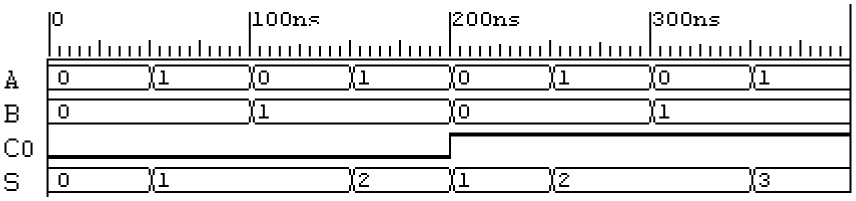


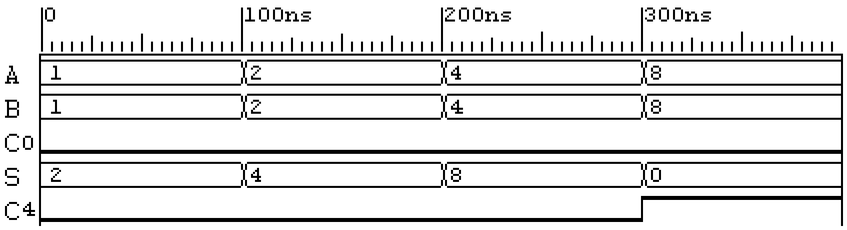
5-21.





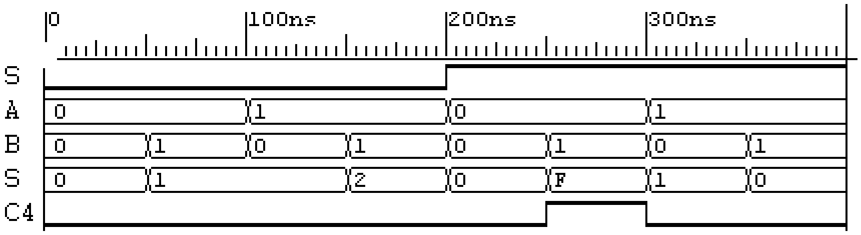
5-22.*

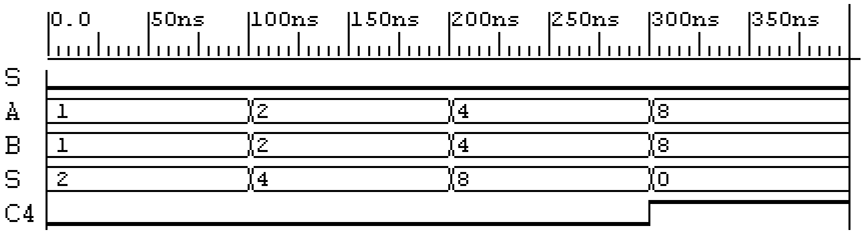




5-23.+

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

entity ad_sub_4_bit is
    port (
        A, B: in STD_LOGIC_VECTOR (3 downto 0);
                 Sel: in STD_LOGIC;
        S: out STD_LOGIC_VECTOR (3 downto 0);
        C4: out STD_LOGIC
    );
end ad_sub_4_bit;

architecture ad_sub_4_bit_arch of ad_sub_4_bit is
signal temp: std_logic_vector(4 downto 0);
begin

 temp <=  ('0' & A) + ('0' & B) when Sel = '0' else
                        ('0' & A) + not('0' & B) + "00001" when Sel = '1' else
                        "00000";

S <= temp(3 downto 0);
C4 <= temp(4);
end ad_sub_4_bit_arch;
```

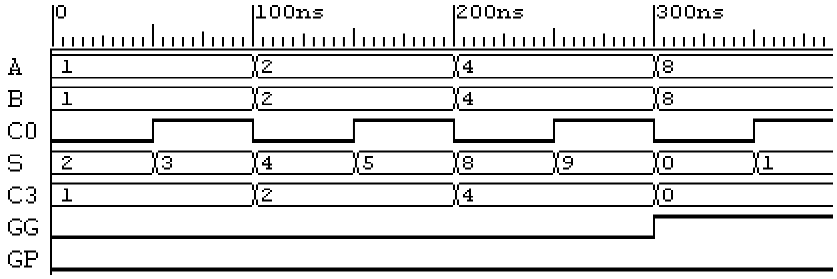




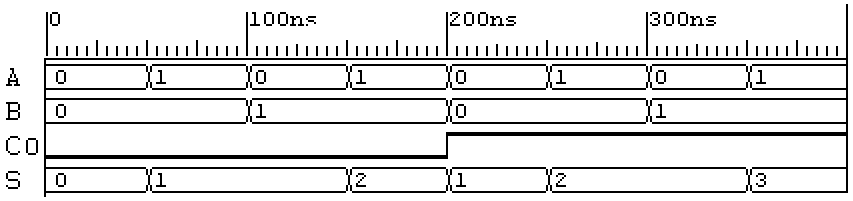
5-24.+

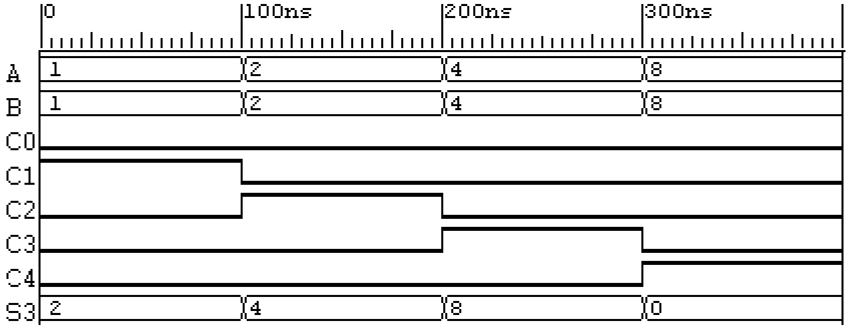
```
library IEEE;
use IEEE.std_logic_1164.all;
entity PFA is
     port (
               A, B, C: in STD_LOGIC;
               P, G, S: out STD_LOGIC);
end PFA;

architecture PFA_arch of PFA is
begin
     P <= A xor B;
     G <= A and B;
     S <= A xor B xor C;
end PFA_arch;

library IEEE;
use IEEE.std_logic_1164.all;
entity cla_logic is
     port (
               P, G: in STD_LOGIC_VECTOR(3 downto 0 );
               C0: in STD_LOGIC;
               GG, GP, C1, C2, C3: out STD_LOGIC);
end cla_logic;

architecture cla_logic_arch of cla_logic is
begin
C1 <= G(0) or (P(0) and C0);
C2 <= G(1) or (P(1) and G(0)) or (P(1) and P(0) and C0);
C3 <= G(2) or (P(2) and G(1)) or (P(2) and P(1) and G(0)) or (P(2) and P(1) and P(0) and C0);
GP <= P(3) and P(2) and P(1) and P(0);
GG <= G(3) or (P(3) and G(2)) or (P(3) and P(2) and G(1)) or (P(3) and P(2) and P(1) and G(0));
end cla_logic_arch;


library IEEE;
use IEEE.std_logic_1164.all;
entity cla_4_bit is
     port (
               A, B: in STD_LOGIC_VECTOR(3 downto 0 );
               C0: in STD_LOGIC;
               S: out STD_LOGIC_VECTOR(3 downto 0 );
               C4, GG, GP: out STD_LOGIC);

end cla_4_bit;

architecture cla_4_bit_arch of cla_4_bit is

component PFA
     port (
               A, B, C: in STD_LOGIC;
               P, G, S: out STD_LOGIC);
end component;

component cla_logic
     port (
               P, G: in STD_LOGIC_VECTOR(3 downto 0 );
               C0: in STD_LOGIC;
               GG, GP, C1, C2, C3: out STD_LOGIC);
end component;
```

```
signal P, G: STD_LOGIC_VECTOR(3 downto 0);
signal C: STD_LOGIC_VECTOR(3 downto 1);

begin
     bit0: PFA
          port map(A(0),B(0),C0,P(0),G(0),S(0));
     bit1: PFA
          port map(A(1),B(1),C(1),P(1),G(1),S(1));
     bit2: PFA
          port map(A(2),B(2),C(2),P(2),G(2),S(2));
     bit3: PFA
          port map(A(3),B(3),C(3),P(3),G(3),S(3));

     carry_logic: cla_logic
          port map(P, G, C0, GG, GP,C(1), C(2), C(3));

end cla_4_bit_arch;
```

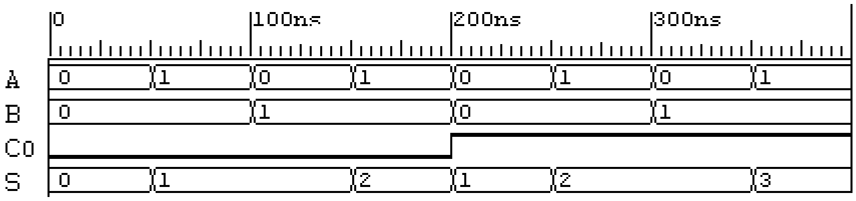


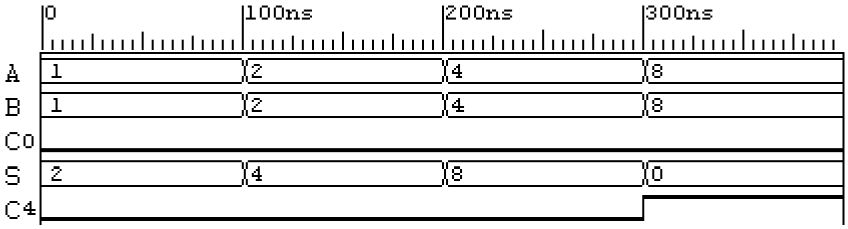
5-25.





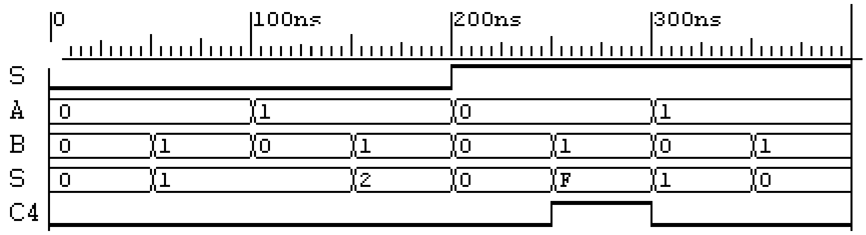
5-26.*

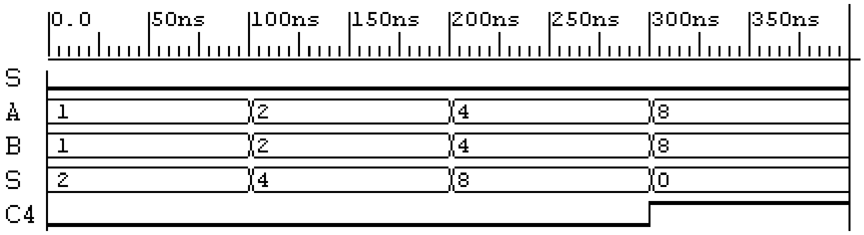




5-27.

```
/ 4-bit Adder: Behavioral Verilog Description
module adder_4_b_v(A, B, Sel, S, C4);
      input[3:0] A, B;
      input Sel;
      output[3:0] S;
      output C4;

      assign {C4, S} = Sel ? (A - B):(A + B);
endmodule
```

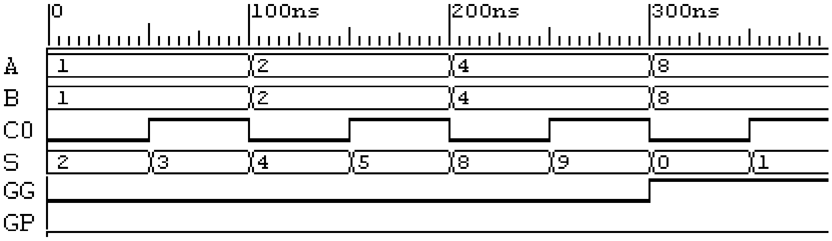




5-28.+

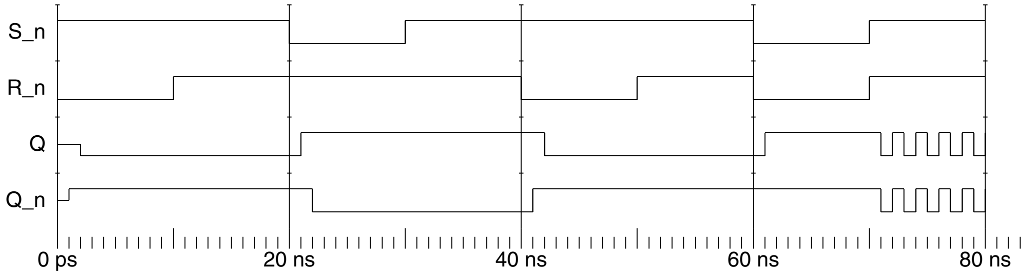
```
module PFA(A, B, C, P, G, S);
      input A, B, C;
      output P, G, S;

      assign P= A ^ B;
      assign G= A & B;
      assign S= (A ^ B) ^ C;
endmodule

module cla_logic (P, G, GG, GP, C0, C1, C2, C3) ;

      input [3:0] P, G ;
      output GG, GP ;
      input C0 ;
      output C1, C2, C3;

      assign C1= G[0] | (P[0] & C0);
      assign C2= G[1] | P[1] & G[0] | P[1] & P[0] & C0;
      assign C3= G[2] | P[2] & G[1] | P[2] & P[1] & G[0] | P[2] & P[1] & P[0] & C0;

      assign GP= P[3] & P[2] & P[1] & P[0];
      assign GG= G[3] | P[3] & G[2] | P[3] & P[2] & G[1] | P[3] & P[2] & P[1] & G[0];

endmodule

endmodule
```

```
module cla_4_bit (A, B, C0, S, C4, GG ,GP);

    input [3:0] A, B ;
    input C0 ;
    output [3:0] S;
    output GG, GP;

    wire[3:0] P, G;
    wire[3:1] C;

    PFA bit0(A[0],B[0],C0,P[0],G[0],S[0]),
        bit1(A[1],B[1],C[1],P[1],G[1],S[1]),
        bit2(A[2],B[2],C[2],P[2],G[2],S[2]),
        bit3(A[3],B[3],C[3],P[3],G[3],S[3]);

    cla_logic  logic(P, G, GG, GP, C0, C[1], C[2], C[3]);

endmodule
```

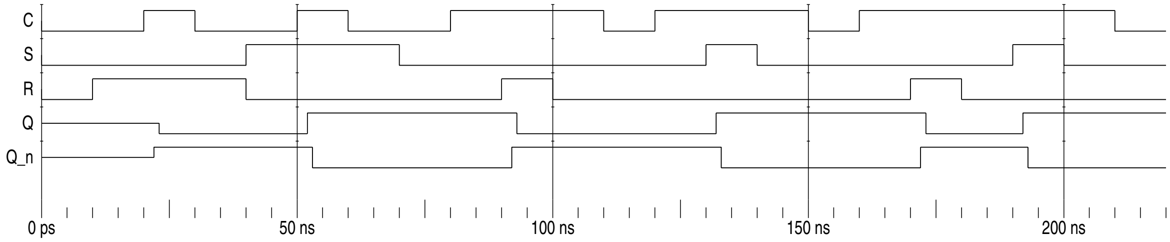


CHAPTER 6

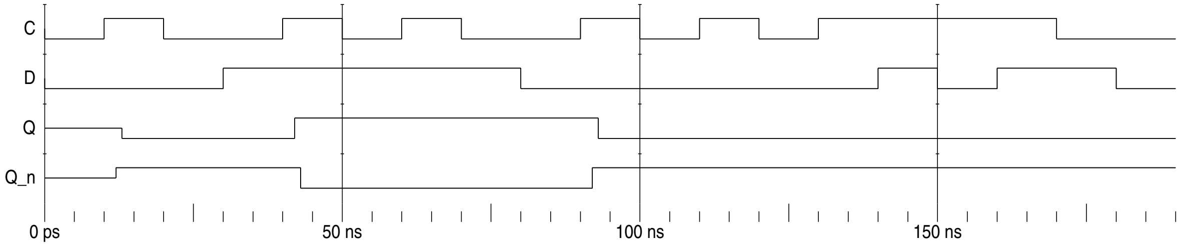
6-1.



6-2.



6-3.



6-4. (Errata: Flip-flop 1 is master-slave.)

a) There are no setup time violations. There is a hold time violation at 28 ns. There is an input combination violation just before 24 ns.

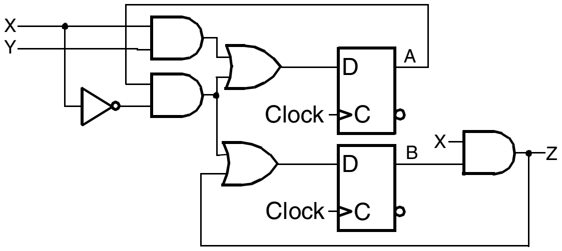
b) There are no setup time violations. There is a hold time violation just before 24 ns. There is an input combination violation just before 24 ns.

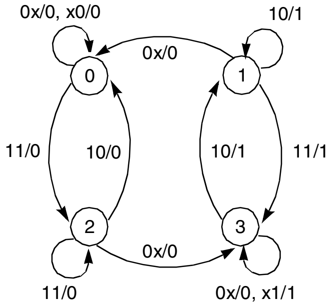
c) There is a setup time violation at 28ns.

d) There is a hold time violation at 16ns and a setup time violation at 24ns.

6-5.

Present state		Inputs		Next state		Output
A	***B***	***X***	***Y***	***A***	***B***	***Z***
0	0	0	0	0	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	0
0	1	0	1	0	0	0
0	1	1	0	0	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	0
1	0	0	1	1	1	0
1	0	1	0	0	0	0
1	0	1	1	1	0	0
1	1	0	0	1	1	0
1	1	0	1	1	1	0
1	1	1	0	0	1	1
1	1	1	1	1	1	1

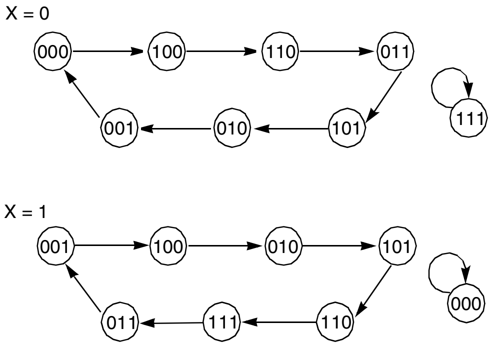




Format: XY/Z (x = unspecified)

6-6.*

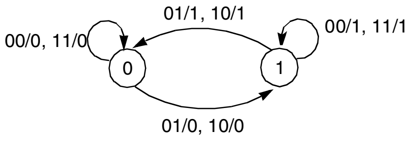
Present state			Input	Next state		
A	***B***	***C***	***X***	***A***	***B***	***C***
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	0	0	1
1	0	0	0	1	1	0
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	1	1	0
1	1	0	0	0	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1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



State diagram is the combination of the above two diagrams.

6-7.

Present state	Inputs		Next state	Output
Q	***X***	***Y***	***Q***	***S***
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

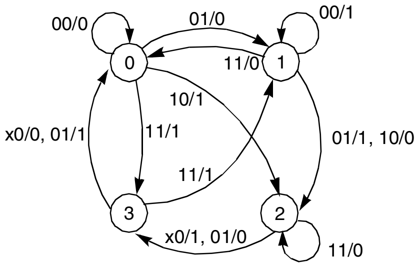


Format: XY/S

6-8.

Present State	00	01	00	00	01	11	00	01	11	10	10
Input	1	0	0	1	1	0	1	1	1	1	0
Output	0	1	0	0	0	1	0	0	0	0	1
Next State	01	00	00	01	11	00	01	11	10	10	00

6-9.*



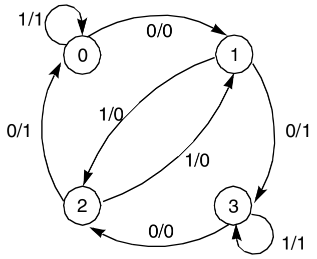
Format: XY/Z

6-10.*

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

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

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

6-11.

a)

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

$t_{delay} = t_{pdXOR} + t_{pdXOR}$

$= 2.0ns + 2.0ns$

$= 4.0ns$

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 FlipFlop.

$t_{delay} = t_{pdXOR} + t_{pd\ INV} + t_{sFF}$

$= 2.0ns + 0.5ns + 1.0ns$

$= 3.5ns$

c)

The longest path delay from the positive clock edge is from FlipFlop A through the two XOR gates to the output Y.

$t_{delay} = t_{pdFF} + t_{pdXOR} + t_{pdXOR}$

$= 2.0ns + 2.0ns + 2.0ns$

$= 6.0ns$

d)

The longest path delay from positive clock edge to positive clock edge is from FlipFlop A through the XOR gate and inverter to FlipFlop B.

$t_{delay} = t_{pdFF} + t_{pdXOR} + t_{pdINV} + t_{sFF}$
$= 2.0ns + 2.0ns + 0.5ns + 1.0ns$
$= 5.5ns$

e)
The maximum frequency is $1/t_{delaymax}$. For this circuit, the longest delay is 6.0 ns, so the maximum frequency is 1/5.5 ns = 181.82 MHz.

6-12.

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

$t_{delay} = t_{pdXOR} + t_{pdXOR} + t_{pdXOR} + t_{pdXOR}$
$= 2.0ns + 2.0ns + 2.0ns + 2.0ns$
$= 8.0ns$

b)
The longest path from an external input to a positive clock edge is from input X through three XOR gates and the inverter to the second B FlipFlop.

$t_{delay} = t_{pdXOR} + t_{pdXOR} + t_{pdXOR} + t_{pd\ INV} + t_{sFF}$
$= 2.0ns + 2.0ns +2.0ns + 0.5ns + 1.0ns$
$= 7.5ns$

c)
The longest path delay from the positive clock edge is from the first FlipFlop A through the four XOR gates to the output Y.

$t_{delay} = t_{pdFF} + t_{pdXOR} + t_{pdXOR} + t_{pdXOR} + t_{pdXOR}$
$= 2.0ns + 2.0ns + 2.0ns + 2.0ns + 2.0ns$
$= 10.0ns$

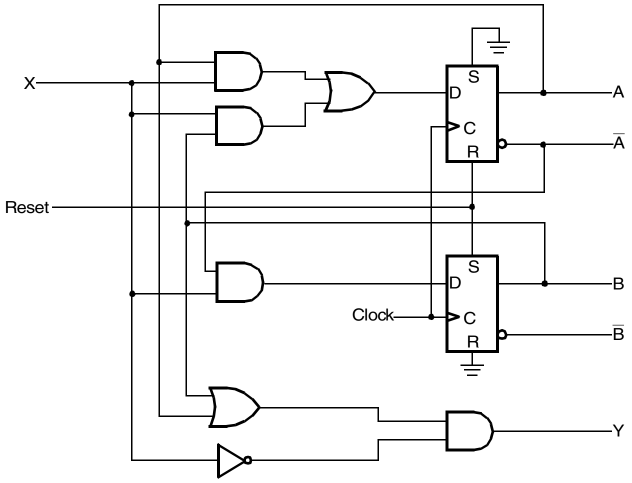
d)
The longest path delay from positive clock edge to positive clock edge is from the first FlipFlop A through three XOR gates and one inverter to the second FlipFlop B.

$t_{delay} = t_{pdFF} + t_{pdXOR} + t_{pdXOR} + t_{pdXOR} + t_{pdINV} + t_{sFF}$
$= 2.0ns + 2.0ns + 2.0ns + 2.0ns + 0.5ns + 1.0ns$
$= 9.5ns$

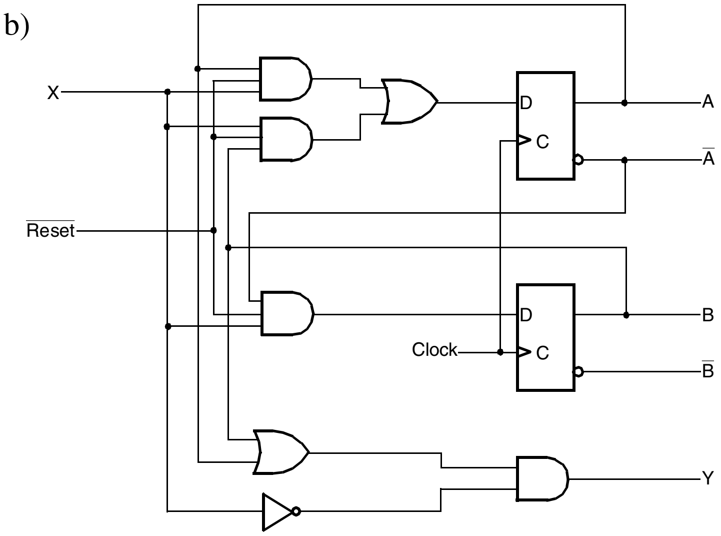
e)
The maximum frequency is $1/t_{delay\text{-}clockedge\ to\ clockedge}$. For this circuit, the longest delay is 10.0 ns, so the maximum frequency is 1/9.5 ns = 105.26 MHz.

6-13.

a)



b)



6-14.*

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

			B	B
		1		
A	1	1		1
		X	X	

D_B

			B	B
				1
A		1	1	1
		X	X	

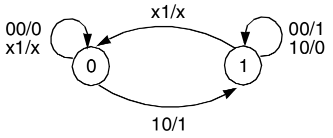
$D_A = A\overline{B} + A\overline{X} + \overline{B}X$ $D_B = AX + B\overline{X}$

Logic diagram not given.

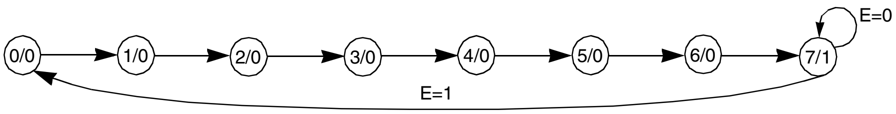
6-15.*

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

Format: XY/Z (x = unspecified)

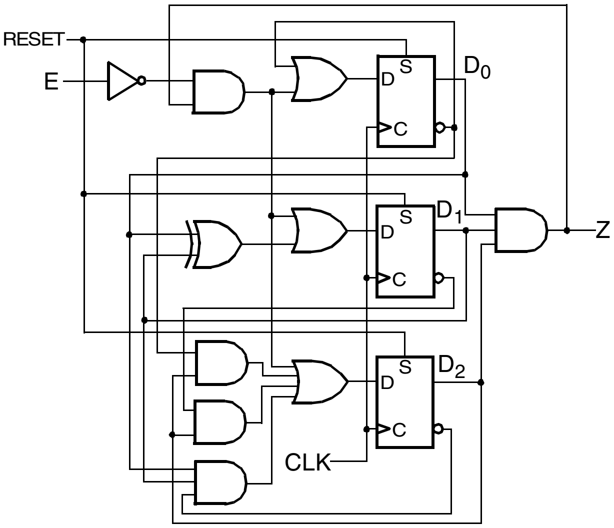


6-16.

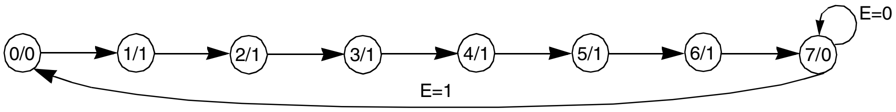


Present state	Next State For Input		Output
$D_2D_1D_0$	*E=0*	*E=1*	*Z*
000	001	001	0
001	010	010	0
010	011	011	0
011	100	100	0
100	101	101	0
101	110	110	0
110	111	111	0
111	111	000	1

The state assignment could be different. E. g., state 7 could be 000 with state 0 001. This would permit use of R inputs on the D flip-flops for RESET.

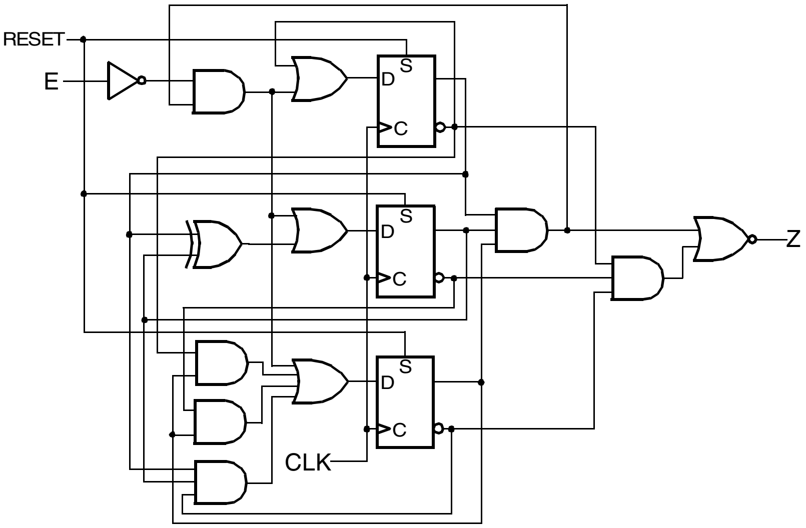


6-17.

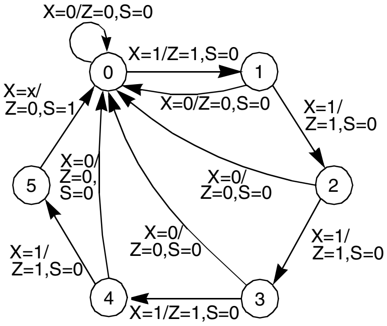


Assumes for E = 0, the output remains at 0.

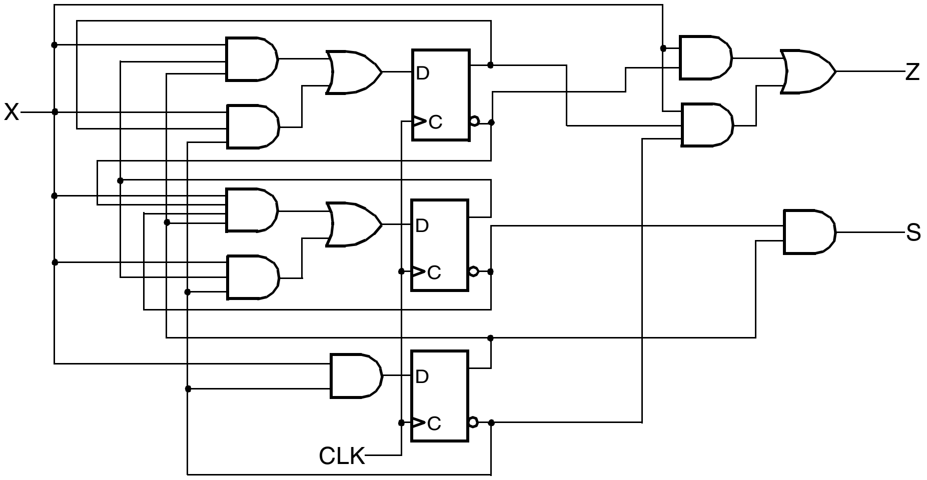
Present state	Next State For Input		Output
$D_2D_1D_0$	*E=0*	*E=1*	*Z*
000	001	001	0
001	010	010	1
010	011	011	1
011	100	100	1
100	101	101	1
101	110	110	1
110	111	111	1
111	111	000	0



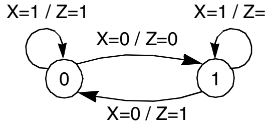
6-18.$^+$



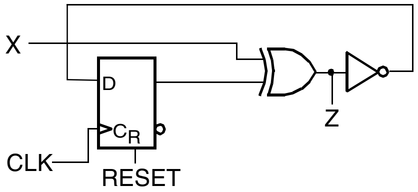
Present state			Input	Next state			Output	
A	***B***	***C***	***X***	***A***	***B***	***C***	***Z***	***S***
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1	0
0	0	1	0	0	0	0	0	0
0	0	1	1	0	1	0	1	0
0	1	0	0	0	0	0	0	0
0	1	0	1	0	1	1	1	0
0	1	1	0	0	0	0	0	0
0	1	1	1	1	0	0	1	0
1	0	0	0	0	0	0	0	0
1	0	0	1	1	0	1	1	0
1	0	1	0	0	0	0	0	1
1	0	1	1	0	0	0	0	1



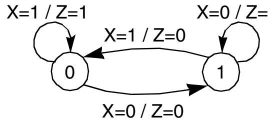
6-19. (Errata: Last two bits of NRSI Message: change "01" to "10")



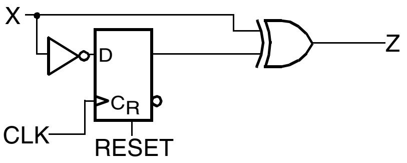
Present state	Input	Next state	Output
A	*X*	*A*	*Z*
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0



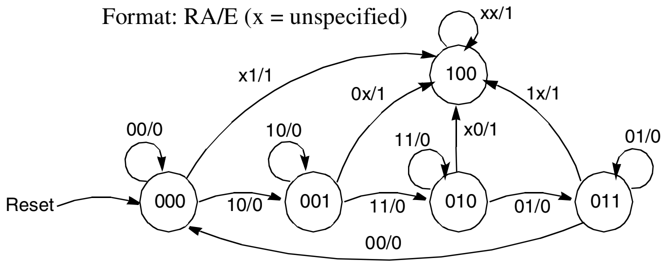
6-20.$^+$



Present state	Input	Next state	Output
A	*X*	*A*	*Z*
0	0	1	0
0	1	0	1
1	0	1	1
1	1	0	0



6-21.

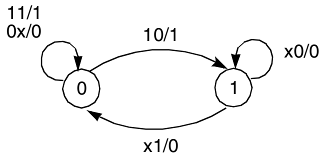


Present state			Inputs		Next state			Output
B	*C*	*D*	*R*	*A*	*B*	*C*	*D*	*E*
0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	1
0	0	0	1	0	0	0	1	0
0	0	0	1	1	1	0	0	1
0	0	1	0	0	1	0	0	1
0	0	1	0	1	1	0	0	1
0	0	1	1	0	0	0	1	0
0	0	1	1	1	0	1	0	0
0	1	0	0	0	1	0	0	1
0	1	0	0	1	0	1	1	0
0	1	0	1	0	1	0	0	1
0	1	0	1	1	0	1	0	0
0	1	1	0	0	0	0	0	0
0	1	1	0	1	0	1	1	0
0	1	1	1	0	1	0	0	1
0	1	1	1	1	1	0	0	1
1	0	0	0	0	1	0	0	1
1	0	0	0	1	1	0	0	1
1	0	0	1	0	1	0	0	1
1	0	0	1	1	1	0	0	1

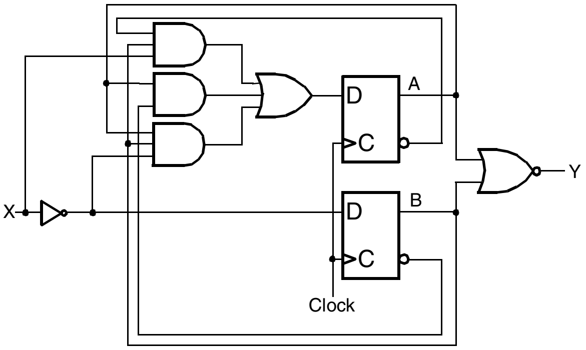
6-22.

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

Format: XY/Z (x = unspecified)



6-23.*



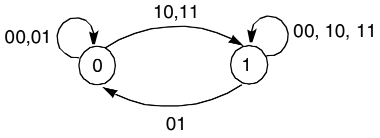
6-24.*

a)

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

b)

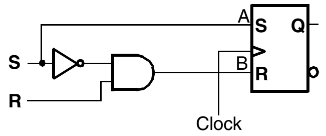
Format: SR



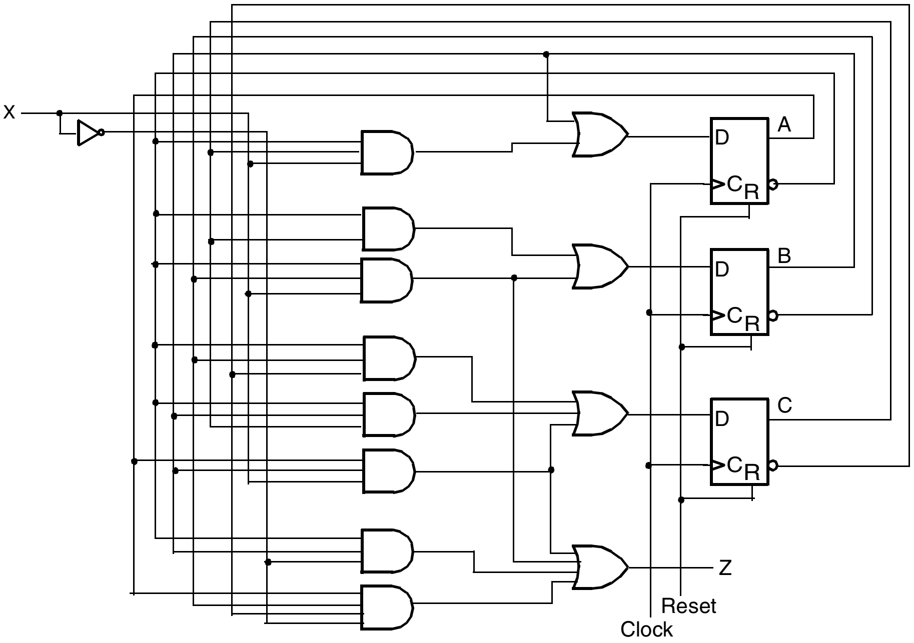
c)

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 = $\overline{\text{S}}$R



6-25.(Errata: Change “state table in Table 6-5” to “state diagram in Figure 6-25(d).”)



6-26.$^+$

Present State	Next State
ABC	ABC
000	100
001	000
010	XXX
011	001
100	110
101	XXX
110	111
111	011

a) $D_A = \overline{C}$

$D_B = A$

$D_C = B$

b) Clear A = $\overline{\text{Reset}}$

Clear B = $\overline{\text{Reset}}$

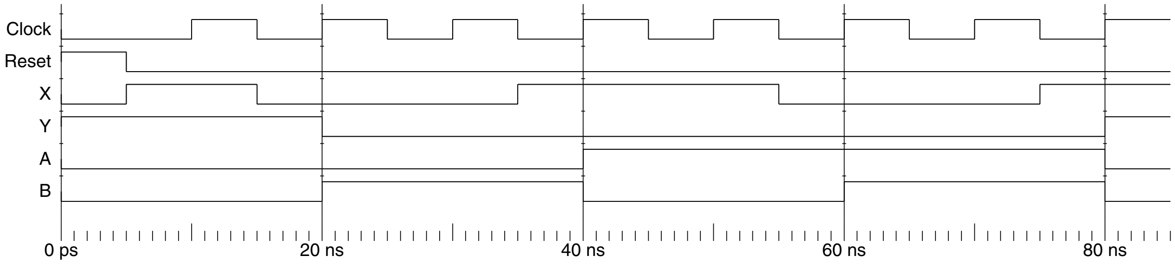
Clear C = $\overline{\text{Reset}}$

c, d, e, f) The circuit is suitable for child’s toy, but not for life critical applications. In the case of the child’s toy, it is the cheapest implementation. If an error occurs the child just needs to reset it. In life critical applications, the immediate detection of errors is critical. The circuit above enters invalid states for some errors. For a life critical application, additional circuitry is needed for immediate detection of the error (Error = $\overline{A}B\overline{C} + A\overline{B}C$). This circuit using the design in a), does return from the invalid states to a valid state automatically after one or two clock periods.

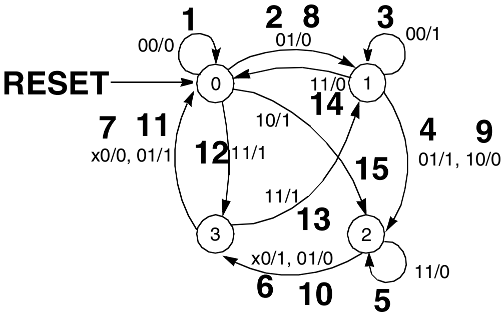
6-27.

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

6-28.(Errata: Change “6-25” to “6-23.” Change “Table 6-6” to “Figure 6-40.” Change “Z” to “Y.”)



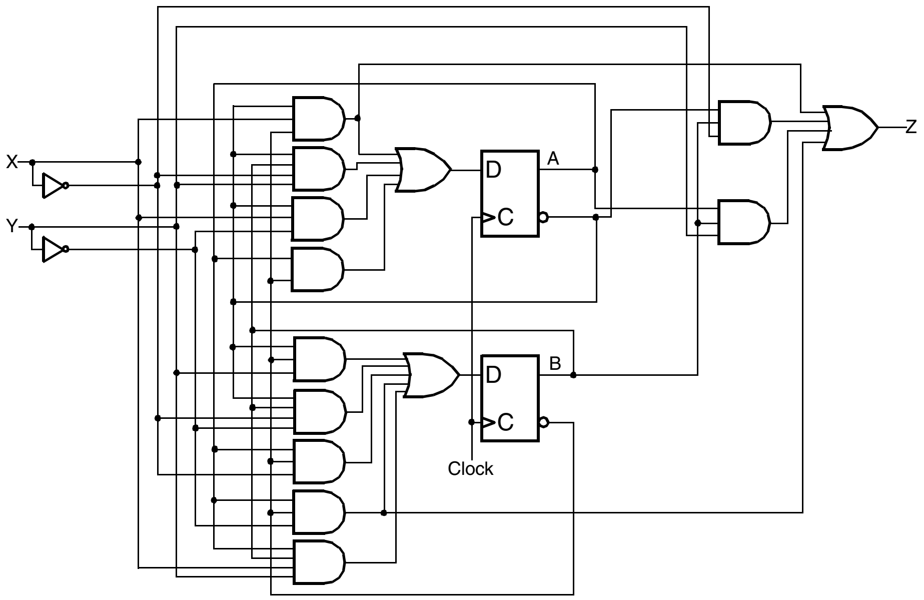
6-29.*

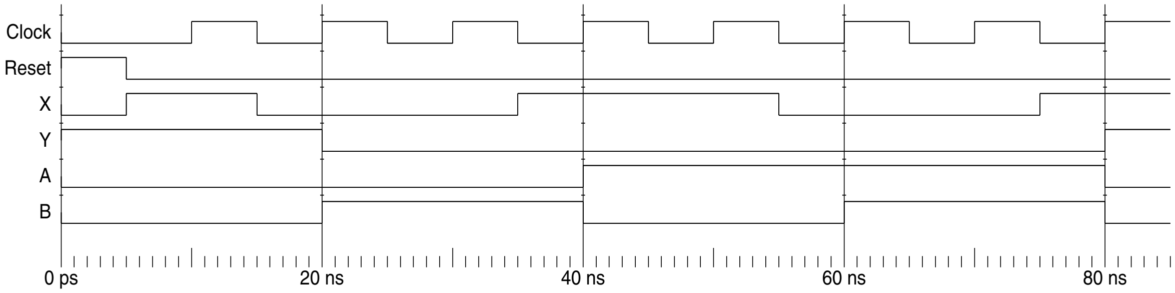


Format: XY/Z

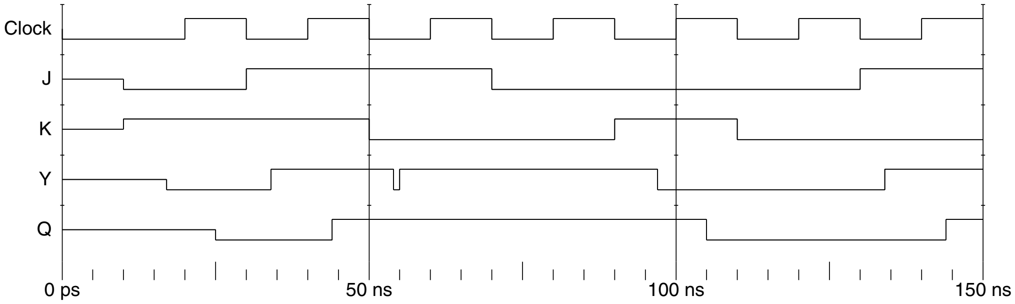
Reset, 00, 01, 00, 01, 11, x0, x0, 01, 10, 01, 01, 11, 11, 11, 10.

6-30.





6-31.*



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.

6-32.*

```
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;
```

6-33.

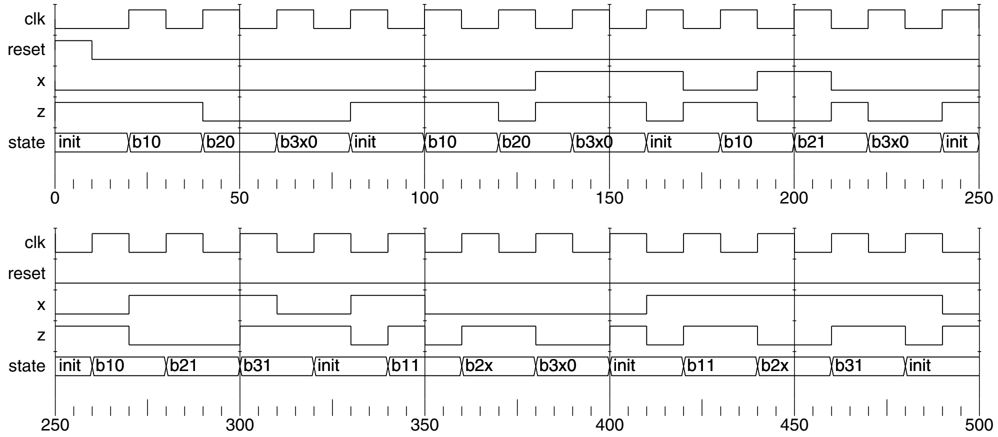
```
library IEEE;
use IEEE.std_logic_1164.all;

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

    if S = "00" then Y <= D(0);
    elsif S = "01" then Y <= D(1);
    elsif S = "10" then Y <= D(2);
    elsif S = "11" then Y <= D(3);
    else null;
    end if;

end process;
end mux_4to1_arch;
```

6-34.+

```
library IEEE;
use IEEE.std_logic_1164.all;
entity serial_BCD_Ex3  is
   port (clk, reset, X : in STD_LOGIC;
          Z  : out STD_LOGIC);
end serial_BCD_Ex3;

architecture process_3 of serial_BCD_Ex3  is
type  state_type  is  (Init,  B10,  B11,  B20,  B21,  B2X,
B3X0, B31);
signal state, next_state: state_type;
begin

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

-- (continued in the next column)

-- Process 2 - next state function
next_state_func: process (X, state)
begin
    case state is
        when Init =>
    if (X = '0') then
        next_state <= B10;
    else
        next_state <= B11;
    end if;
        when B10 =>
    if (X = '0') then
        next_state <= B20;
    else
        next_state <= B21;
    end if;
        when B11 =>
        next_state <= B2X;
        when B20 =>
        next_state <= B3X0;
        when B21 =>
    if (X = '0') then
        next_state <= B3X0;
    else
        next_state <= B31;
    end if;
```

```
        when B2X =>
    if (X = '0') then
        next_state <= B3X0;
    else
        next_state <= B31;
    end if;
        when B3X0 =>
        next_state <= Init;
        when B31 =>
        next_state <= Init;
end case;
end process;

-- Process 3 -output function
output_func: process (X, state)
begin
    case state is
        when Init =>
    if (X = '0') then
        Z <= '1';
    else
        Z <= '0';
    end if;
        when B10 =>
    if (X = '0') then
        Z <= '1';

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

```
    else
        Z<= '0';
    end if;
        when B11 =>
        Z <= X;
        when B20 =>
        Z <= X;
        when B21 =>
    if (X = '0') then
        Z <= '1';
    else
        Z <= '0';
    end if;
        when B2X =>
    if (X = '0') then
        Z <= '1';
    else
        Z <= '0';
    end if;
        when B3X0 =>
        Z <= X;
        when B31 =>
        Z <= '1';
end case;
end process;

end process_3;
```



6-35.

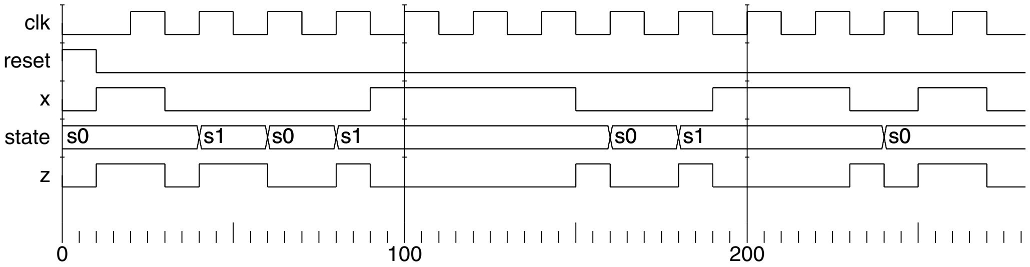
```
library IEEE;
use IEEE.std_logic_1164.all;
entity Zseq_circuit is
   port (
      X, Y, CLK, RESET: in STD_LOGIC;
      Z: out STD_LOGIC
   );
end seq_circuit;

architecture seq_circuit_arch of seq_circuit is
type state_type is (A,B);
signal state, next_state: state_type;
begin
 state_register: process (CLK, RESET)
begin
  if RESET='1' then--asynchronous RESET active High
    state <= A;
  elsif (CLK'event and CLK='1') then  --CLK rising edge
    state <= next_state;
  end if;
end process;

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

```
next_state_process: process (X, Y, state)
begin
    case state is
         when A =>
              if (X = '1' and Y = '0') then next_state <= B;
              else next_state <= A;
              end if;
         when B =>
              if Y = '1' then next_state <= A;
              else next_state <= B;
              end if;
    end case;
end process;
output_func: process (X, state)
begin
    case state is
         when A =>
              Z <= X;
         when B =>
              Z <= not X;
    end case;
end process;
end seq_circuit_arch;
```

6-36.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity NRZI is
   port (
      X, CLK, RESET: in STD_LOGIC;
      Z: out STD_LOGIC
   );
end seq_circuit;
architecture process_3 of NRZI is
type state_type is (S0,S1);
signal state, next_state: state_type;
begin
 state_register: process (CLK, RESET)
begin
  if RESET='1' then--asynchronous RESET active High
    state <= S0;
  elsif (CLK'event and CLK='1') then  --CLK rising edge
    state <= next_state;
  end if;
end process;
-- (continued in the next column)
```

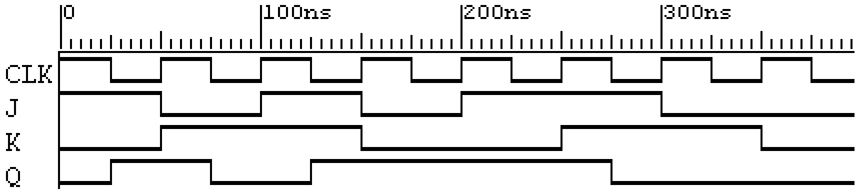
```
next_state_process: process (X,  state)
begin
    case state is
         when S0 =>
              if (X = '1') then next_state <= S0;
              else next_state <= S1;
              end if;
         when S1 =>
              if (X = '1') then next_state <= S0;
              else next_state <= S1;
              end if;
    end case;
end process;
    output_func: process (X, state)
begin
    case state is
         when S0 => Z <= X;
         when S1 => Z <= not X;
    end case;
end process;
end process_3;
```



6-37.*

```
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;
```



6-38.

```
module problem_6_38 (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
   case (S)
      2'b00 : Y <= D[0] ;
      2'b01 : Y <= D[1] ;
      2'b10 : Y <= D[2] ;
      2'b11 : Y <= D[3] ;
   endcase;
   end
endmodule
```

6-39.*

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

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

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

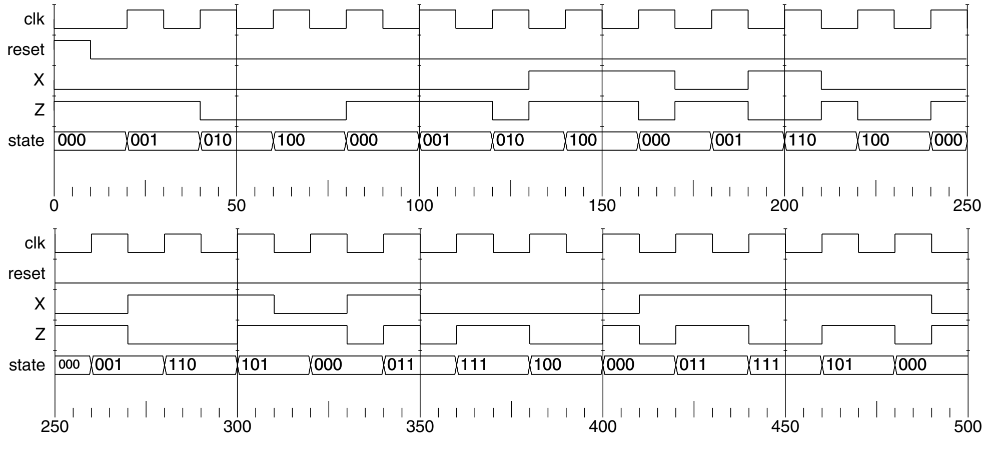
```
always @(S or D)
   begin
      if (S == 2'b00) Y <= D[0];
      else if (S == 2'b01) Y <= D[1];
      else if (S == 2'b10) Y <= D[2];
      else Y <= D[3];

   end
endmodule
```

6-40.+

```
//Serial BCD to Excess 3 Converter
module serial_BCD_Ex3(clk, reset, X, Z);
input clk, reset, X;
output Z;

reg[2:0] state, next_state;
parameter Init = 3'b000, B10 = 3'b001,
B11=3'b011, B20= 3'b010, B21 = 3'b110,
B2X = 3'b111, B3X0 = 3'b100, B31 = 3'b101;
reg Z;

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

// Next StateFunction
always@(X or state)
begin
    case (state)
    Init: if (X == 0)
            next_state <= B10;
        else
            next_state <= B11;
    B10: if (X == 0)
            next_state <= B20;
        else
            next_state <= B21;
    B11: next_state <= B2X;
    B20: next_state <= B3X0;
    B21: if (X == 0)
            next_state <= B3X0;
        else
            next_state <= B31;
// (continued in the next column)
    B2X: if (X ==0)
            next_state <= B3X0;
        else
            next_state <= B31;

    B3X0: next_state <= Init;
    B31: next_state <= Init;
endcase
end

// Output Function
always@(X or state)
begin
    case (state)
    Init: if (X == 0)
            Z <= 1;
        else
            Z <= 0;
    B10: if (X == 0)
            Z <= 1 ;
        else
            Z <= 0;
    B11: Z <= X;
    B20: Z <= X;
    B21: if (X == 0)
            Z <= 1;
        else
            Z <= 0;
    B2X: if (X == 0)
        Z <= 1;
    else
        Z <= 0;
    B3X0: Z <= X;
    B31: Z <= 1;
endcase
end
endmodule
```

State Assignment

State Code	**State Name**
000	Init
001	B10
010	B20
011	B11
100	B3X0
101	B31
110	B21
111	B2X



6-41.

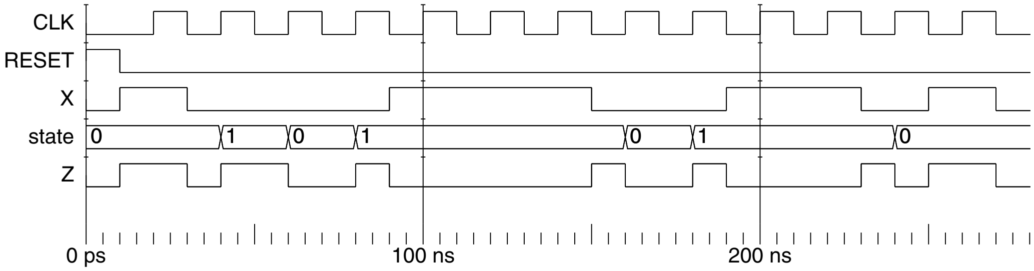
```
module seq_circuit (X, Y, CLK, RESET, Z) ;

input X, Y, CLK, RESET ;
output Z ;
reg [2:0] state, next_state, Z;
// (continued in the next column)
parameter S0 = 3'b000, S1 = 3'b001;

//State register process
always @(posedge CLK or posedge RESET)
begin
    if (RESET)
```

```
            state <= S0;
        else
            state <= next_state;
end
//Next state function
always @(X or Y or state)
begin
    case (state)
        S0: next_state <= X ? (Y ? S0 : S1) : S0;
        S1: next_state <= Y ? S0 : S1;
    endcase
end
// (continued in the next column)
```

```
//Output function
always @(X or Y or state)
begin
    case (state)
        S0: Z <= X ? (Y ? 1'b0 : 1'b1) : 1'b0;
        S1: Z <= X ? 1'b0 : (Y ? 1'b0 : 1'b1);
    endcase
end
endmodule
```

6-42.

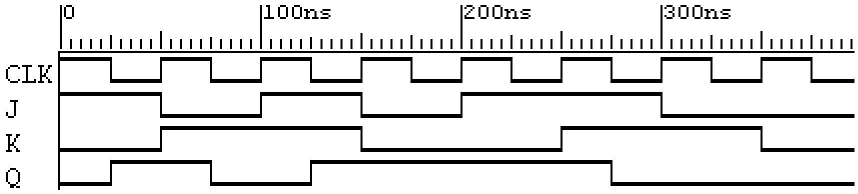
```
//NRZI Code Generator
module NRZI (X, CLK, RESET, Z);
input X, CLK, RESET;
output Z;
reg state, next_state, Z;
parameter S0 = 0, S1 = 1;

//State Register
always@(posedge CLK or posedge RESET)
begin
  if (RESET == 1) // asynchronous RESET active High
    state <= S0;
  else
        state <= next_state;
 end

//Next State Function
always@(X or state)
begin
    case (state)
        S0:
            if (X == 1)
                next_state <= S0;
            else next_state <= S1;
        S1:
            if (X == 1)
                next_state <= S1;
            else next_state <= S0;
    endcase
end

always@(X, state)
begin
    case (state)
        S0:
            Z <= X;
        S1:
            Z <= ~X;
    endcase
end
endmodule
```



6-43.*

```
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
```



PART 2 PROBLEM SOLUTIONS

NOTES ON SOLUTIONS:

1. **Legal Notice:** This publication is protected by United States copyright laws, and is designed exclusively to assist instructors in teaching their courses. It should not be made available to students, or to anyone except the authorized instructor to whom it was provided by the publisher, and should not be sold by anyone under any circumstances. Publication or widespread dissemination (i.e. dissemination of more than extremely limited extracts within the classroom setting) of any part of this material (such as by posting on the World Wide Web) is not authorized, and any such dissemination will violate the United States copyright laws. In consideration of the authors, your colleagues who do not want their students to have access to these materials, and the publisher, please respect these restrictions.

2. **Companion Website Problem Solutions:** The solutions to all problems marked with a * are available to students as well as instructors on the Companion Website.

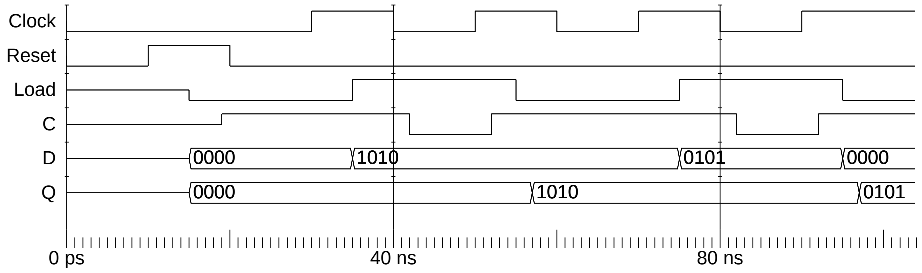
3. **Problem Challenge:** The problems marked with a + are designated as more challenging than the typical problems.

4. **Text Errata Notations:** Text errata are noted at the beginning of a problem if those errata affect either the problem or its solution. These notes indicate only errors identified in the first printing of the 3rd Edition and are expected be removed after the first printing.

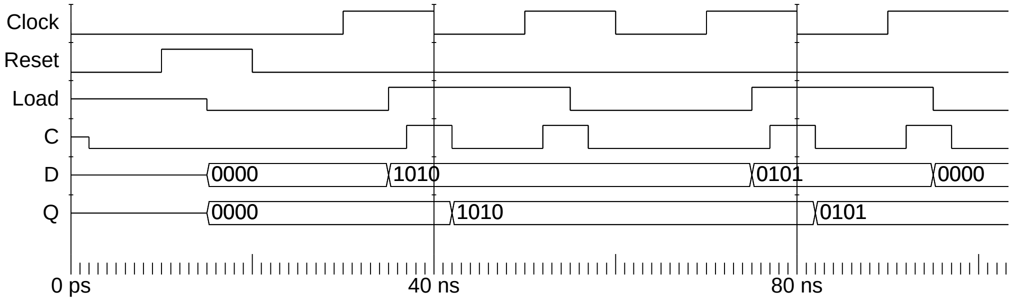
5. **Solutions Errata:** Errata for these solutions will be provided on the Companion Website in the Errata section.

CHAPTER 7

7-1.



7-2.



7-3.

(a) R1 + 2’s complement of R2 = 2^n + R1 - R2. If R1 ≥ R2, the result is ≥ 2^n. The 2^n gives C = 1.

R1 + 2’s complement of R2 = 2^n + R1 - R2, if R1 < R2, the result is < 2^n giving C = 0.

(b) If C = 1 then R1 ≥ R2 and there is no borrow.

If C = 0 then R1 < R2 and there is a borrow. Thus, the borrow is the complement of the C status bit.

7-4.*

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

7-5.

(a) AND, 1010 1010 1010 1010 (b) OR, 1111 0000 0000 0000

(c) XOR, 0000 1111 1111 0000

7-6.*

sl 1010 0110 sr 0010 1001

7-7.*

Connections to MUX data input 0 and data input 3 remain the same. Q_{i-1} is connected to MUX data input 2 instead of MUX Data input 1. Finally, 0 is connected to MUX data input 1.

7-8.*

a) 1000, 0100, 0010, 0001, 1000

b) # States = n

7-9.

a) 0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001, 0000

b) # States = 2*n*

7-10.

a) 5 b) 8

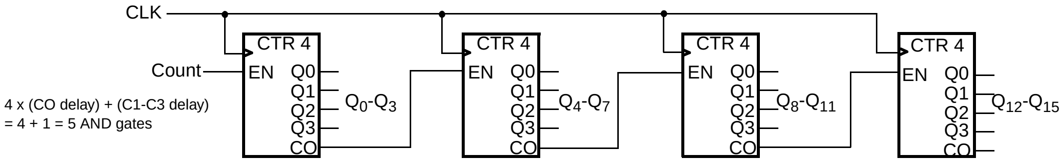
7-11.$^{+}$

Examine an n-bit ripple counter and an n-bit synchronous counter. If either of these counters cycles through all of its states, there are $2(2^n) = 2^{n+1}$ transitions for the clock, and there are $2^{n+1} – 2$ total transitions for all flip-flop outputs. For the ripple counter, the clock transitions occur on the input of only one stage, the 0th stage. For the synchronous counter, the clock transistions occur on the inputs to all of the n stages. Combining the transition counts above, the ration of the input + output transitions for the synchronous counter compared to the ripple counter is:

$[n\ 2^{n+1} + 2^{n+1} – 2]/[2^{n+1} + 2^{n+1} – 2] \approx (n + 1)\ 2^{n+1}/2\ (2^{n+1}) = (n+1)/2$

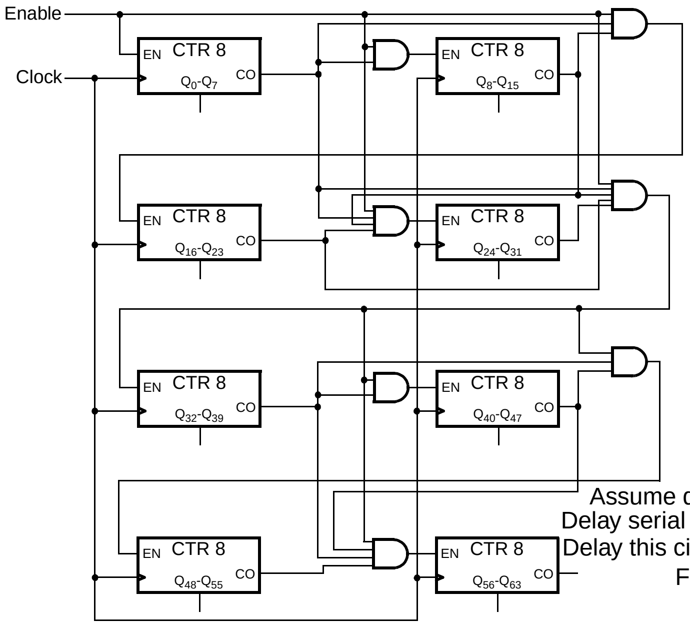
Thus, the power dissipated by the synchronous counter is at least as large as that dissipated by the ripple counter in all cases and grows more rapidly with the number of stages.

7-12.



4 x (CO delay) + (C1-C3 delay)
= 4 + 1 = 5 AND gates

7-13.+



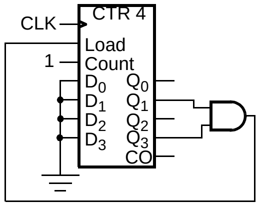
Assume delay of XOR = delay of AND.

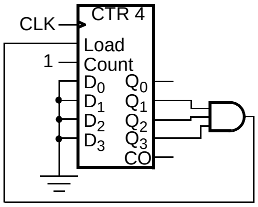
Delay serial = 2 + 1 + 7(1) + 1 + 1 = 12

Delay this circuit = 2 + 1+ 2(1) + 1 + 1 = 7

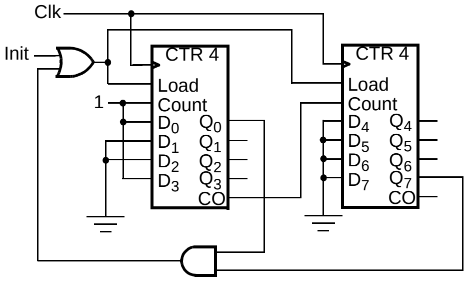
Frequency ratio = 12/7

7-14.



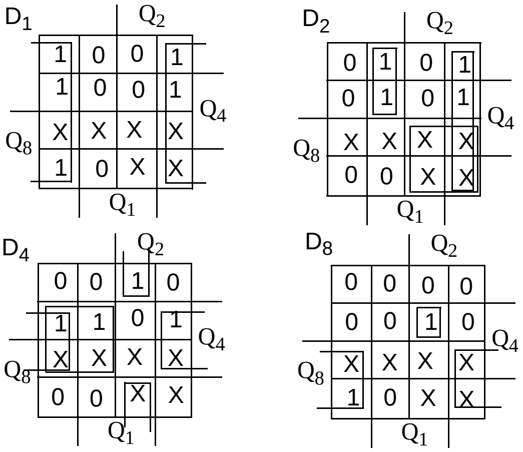


7-15.



7-16.*

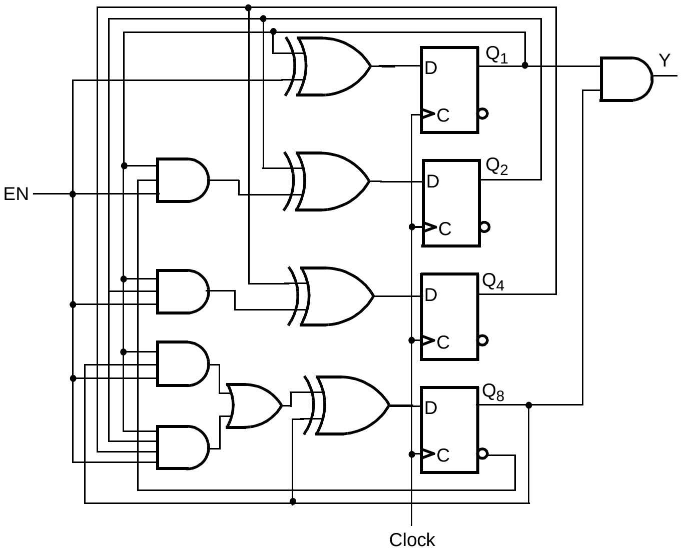
The equations given on page 337 can be manipulated into SOP form as follows: $D_1 = \overline{Q_1}$, $D_2 = Q_2 \oplus Q_1\overline{Q_8} = Q_1\overline{Q_2}\overline{Q_8} + \overline{Q_1}Q_2 + Q_2Q_8$, $D_4 = Q_4 \oplus Q_1Q_2 = Q_1Q_2\overline{Q_4} + \overline{Q_1}Q_4 + \overline{Q_2}Q_4$, $D_8 = Q_8 \oplus (Q_1Q_8 + Q_1Q_2Q_4) = \overline{Q_8}(Q_1Q_8+Q_1Q_2Q_4) + Q_8(\overline{Q_1} + \overline{Q_8})(\overline{Q_1} + \overline{Q_2} + \overline{Q_4}) = Q_1Q_2Q_4\overline{Q_8} + \overline{Q_1}\, Q_8$. 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.



7-17.*

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

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

$D_C = \overline{C}$

7-18.

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

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

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

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

7-19.

Present state					Next state					FF Inputs				
A	*B*	*C*	*D*	*E*	*A*	*B*	*C*	*D*	*E*	D_A	D_B	D_C	D_D	D_E
0	0	0	0	1	0	0	0	1	0	0	0	0	1	0
0	0	0	1	0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	1	0	0	0	0	1	0	0	0
0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0	0	1	0	0	0	0	1

$D_A = B$

$D_B = C$

$D_C = D$

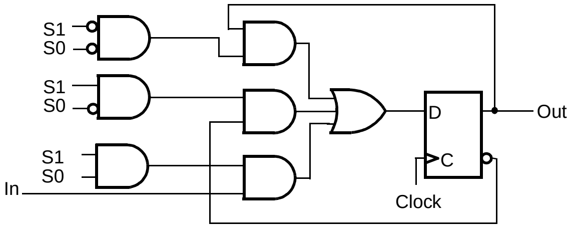
$D_D = E$

$D_E = A$

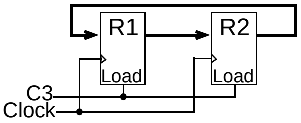
Use ABCD as counter

7-20.

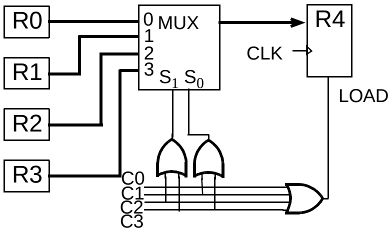
The basic cell of the register is as follows:



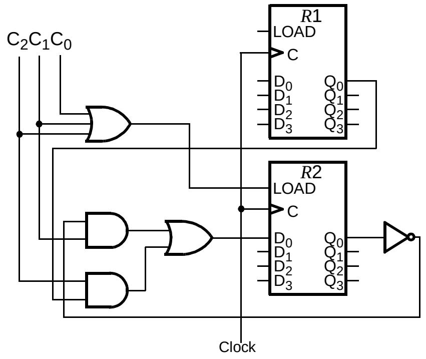
7-21.*



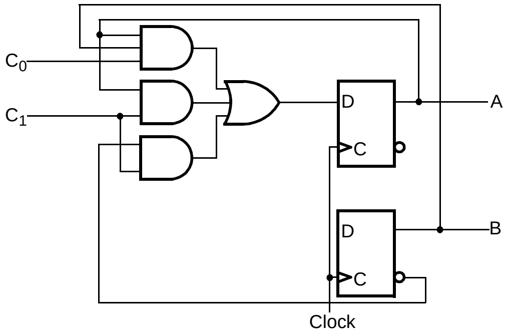
7-22.



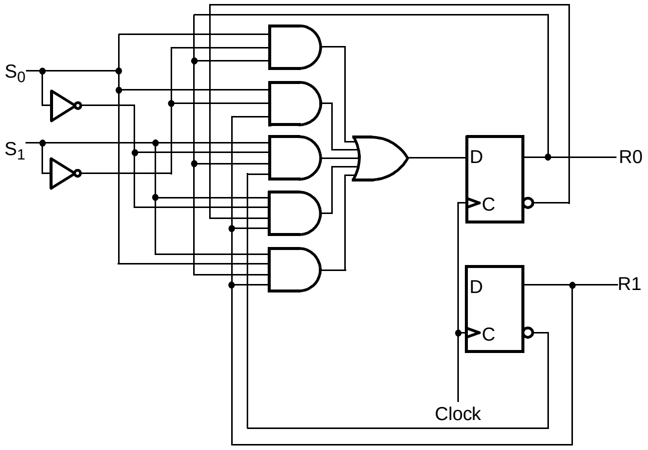
7-23.*



7-24.

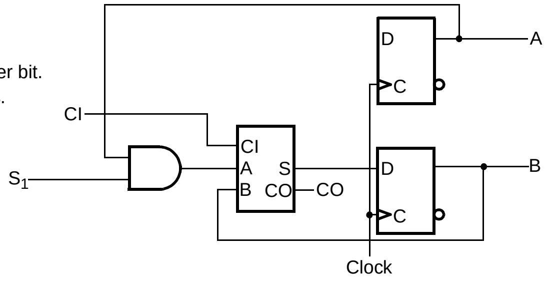


7-25.

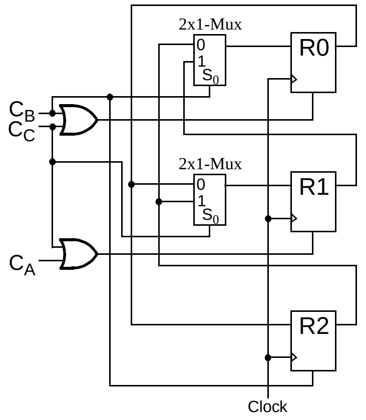


7-26.

CI= S_0 for lowest order bit.
CI = CO for other bits.

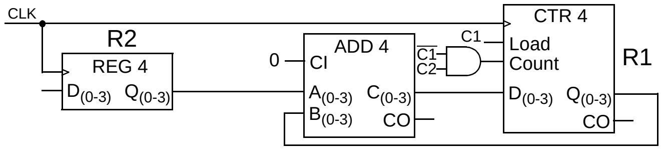


7-27.

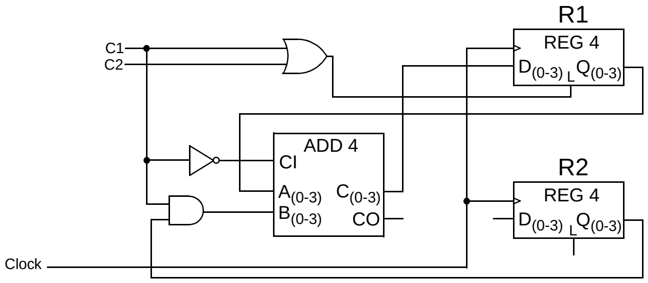


7-28.*

a)



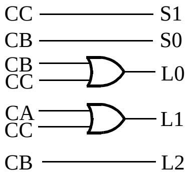
b)

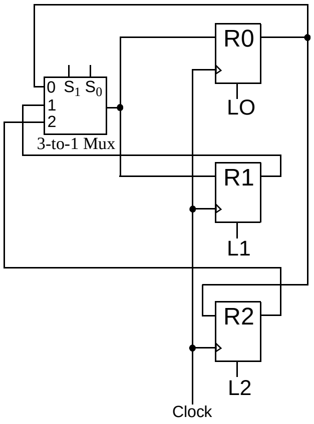


7-29.

The register transfer logic is as follows:

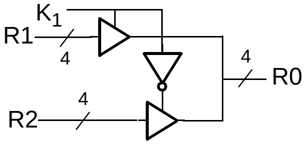
Operation	Select		Load		
	S1	S0	L0	L1	L2
CA: R1 <- R0	0	0	0	1	0
CB: R0 <- R1, R2 <- R0	0	1	1	0	1
CC: R1 <- R2, R0 <- R2	1	0	1	1	0





7-30.

Replace multiplexer with:

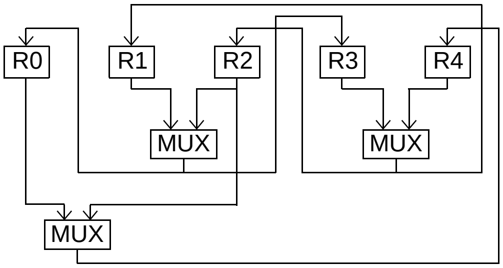


7-31.*

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

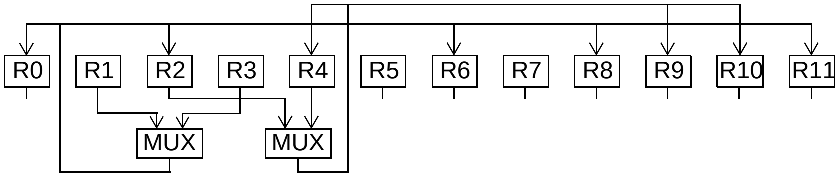
d)



7-32.

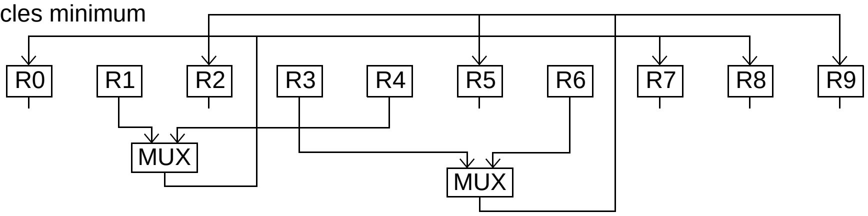
a) Using two clock cycles, the minimum # of buses is 2 .

b)



7-33.

Two clock cycles minimum



7-34.*

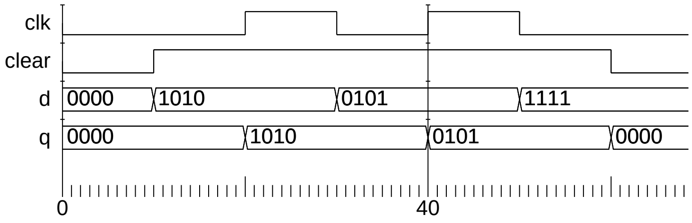
1000, 0100, 1010, 1101 0110, 0011, 0001, 1000

7-35.*

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

7-36.*

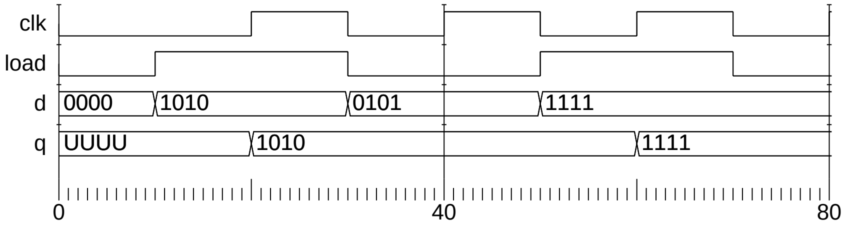
```
library IEEE;
use IEEE.std_logic_1164.all;

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;
```



7-37.

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

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

-- (continued in next column)
```

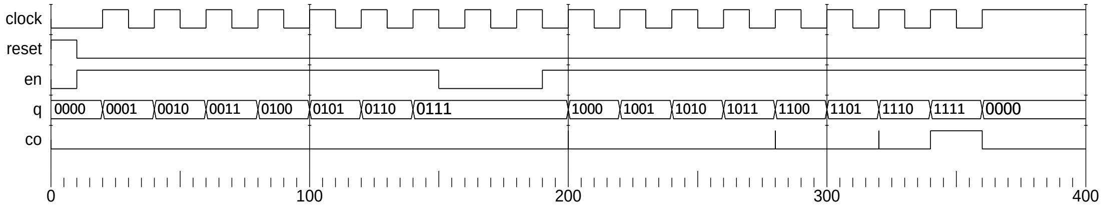
```
architecture reg_4_bit_load_arch of reg_4_bit is
begin

process (CLK)
begin
     if (CLK'event and CLK='1') then  --CLK rising edge
          if LOAD = '1' then
               Q <= D;
          end if;
     end if;
end process;

endreg_4_bit_load_arch;
```



7-38.

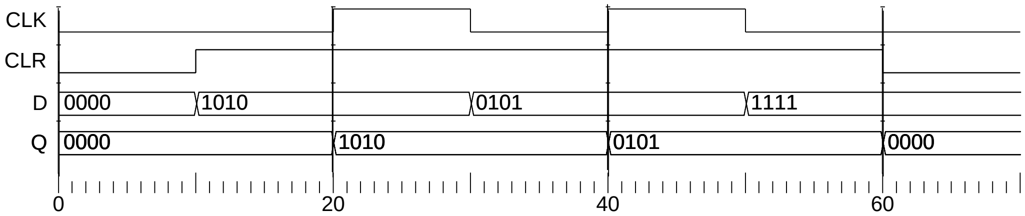
```
library ieee;
use ieee.std_logic_1164.all;
entity dff is
   port(CLK, RESET, D: in std_logic;
      Q : out std_logic);
end dff;

architecture pet_pr of dff is
-- Implements positive edge-triggered bit state storage
-- with asynchronous reset.
   signal state: std_logic;
begin
   Q <= state;
   process (CLK, RESET)
   begin
     if (RESET = '1') then
       state <= '0';
     else
        if (CLK'event and ClK = '1') then
          state <= D;
        end if;
     end if;
   end process;
end;

library IEEE;
use IEEE.std_logic_1164.all;
entity counter_4_bit is
    port (
        Clock, Reset, EN: in STD_LOGIC;
        Q: out STD_LOGIC_VECTOR (3 downto 0);
        CO: out STD_LOGIC
        );
end counter_4_bit;
```

```
architecture counter_4_bit_arch of counter_4_bit is
component dff
     port(CLK, RESET, D: in std_logic;
        Q: out std_logic
    );
end component ;
signal D_in, C,  Q_out: std_logic_vector(3 downto 0);

begin
        C(0) <= EN;
        C(1) <= C(0) and Q_out(0);
        C(2) <= C(1) and Q_out(1);
        C(3) <= C(2) and Q_out(2);
        CO <= C(3) and Q_out(3);

        D_in(0) <= C(0) xor Q_out(0);
        D_in(1) <= C(1) xor Q_out(1);
        D_in(2) <= C(2) xor Q_out(2);
        D_in(3) <= C(3) xor Q_out(3);

        bit0: dff
               port map (Clock, Reset, D_in(0), Q_out(0));
        bit1: dff
               port map (Clock, Reset, D_in(1), Q_out(1));
        bit2: dff
               port map (Clock, Reset, D_in(2), Q_out(2));
        bit3: dff
               port map (Clock, Reset, D_in(3), Q_out(3));

        Q <= Q_out;

end counter_4_bit_arch;
```



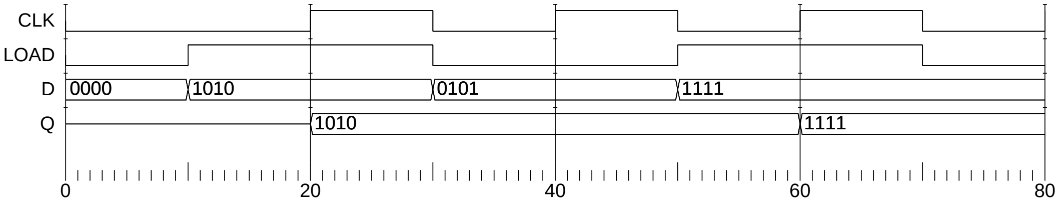
7-39.*

```
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
```



7-40.

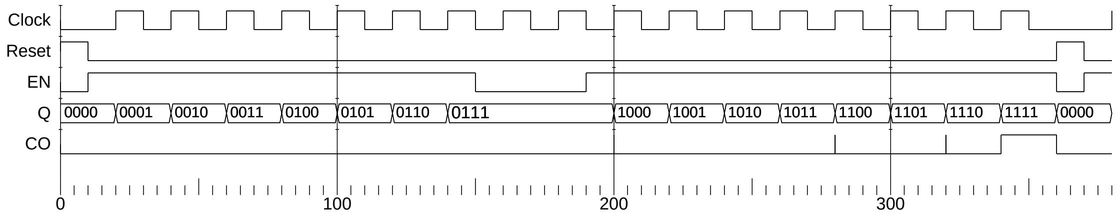
```
module register_4_bit_load (D, CLK, LOAD, Q) ;

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

always @(posedge CLK)
begin
      if (LOAD)
            Q = D;
end
endmodule
```



7-41.

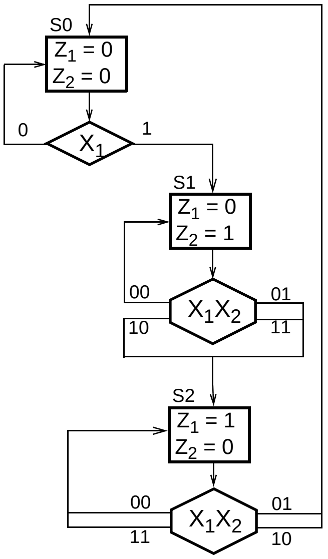
```
// 4-bit Binary Counter

// Positive Edge-Triggered D Flip-Flop with Reset:

module dff_v(CLK, RESET, D, Q);
  input CLK, RESET, D;
  output Q;
  reg state;

  assign Q = state;

always @(posedge CLK or posedge RESET)
begin
  if (RESET)
     state <= 0;
  else
     state <= D;
end
endmodule

module Counter_4bit (Clock, Reset, EN, Q, CO) ;

input Clock, Reset, EN ;
output [3:0] Q ;
output CO ;
wire[3:0] Q ;

wire [3:0] C, D_in;

// (continued in next column)
```

```
        C[0] = EN,
        C[1] = C[0] & Q[0],
        C[2] = C[1] & Q[1],
        C[3] = C[2] & Q[2],
        CO = C[3] & Q[3];

assign
        D_in[0] = C[0] ^ Q[0],
        D_in[1] = C[1] ^ Q[1],
        D_in[2] = C[2] ^ Q[2],
        D_in[3] = C[3] ^ Q[3];


dff_v
        g1(Clock, Reset, D_in[0], Q[0]),
        g2(Clock, Reset, D_in[1], Q[1]),
        g3(Clock, Reset, D_in[2], Q[2]),
        g4(Clock, Reset, D_in[3], Q[3]);

endmodule
```



CHAPTER 8

8-1.*

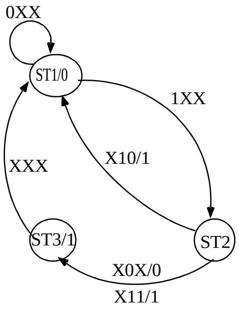


Inputs: X_1X_2
Outputs: Z_1Z_2

8-2.*

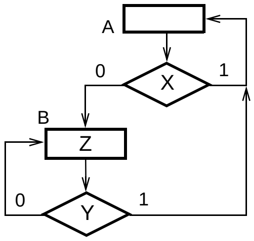
A:	0	1	1	0	1	1	0	1	
B:	1	1	0	1	0	1	0	1	
C:	0	1	0	1	0	1	0	1	
State:	ST1	ST1	ST2	ST3	ST1	ST2	ST3	ST1	ST2
Z:	0	0	0	1	0	1	1	0	

8-3.

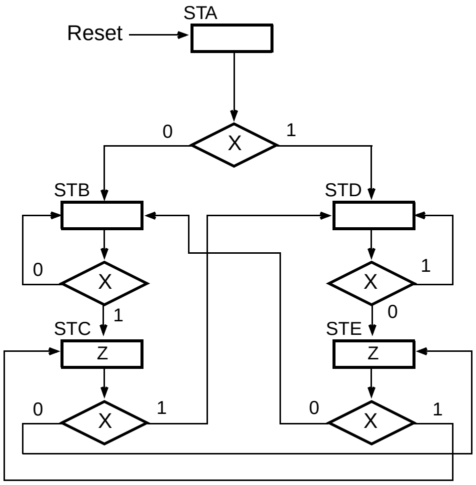


PS	IN			NS	Z	PS	IN			NS	Z	PS	IN			NS	Z
	A	B	C				A	B	C				A	B	C		
ST1	0	0	0	ST1	0	ST2	0	0	0	ST3	0	ST3	0	0	0	ST1	1
ST1	0	0	1	ST1	0	ST2	0	0	1	ST3	0	ST3	0	0	1	ST1	1
ST1	0	1	0	ST1	0	ST2	0	1	0	ST1	1	ST3	0	1	0	ST1	1
ST1	0	1	1	ST1	0	ST2	0	1	1	ST3	1	ST3	0	1	1	ST1	1
ST1	1	0	0	ST2	0	ST2	1	0	0	ST3	0	ST3	1	0	0	ST1	1
ST1	1	0	1	ST2	0	ST2	1	0	1	ST3	0	ST3	1	0	1	ST1	1
ST1	1	1	0	ST2	0	ST2	1	1	0	ST1	1	ST3	1	1	0	ST1	1
ST1	1	1	1	ST2	0	ST2	1	1	1	ST3	1	ST3	1	1	1	ST1	1

8-4.

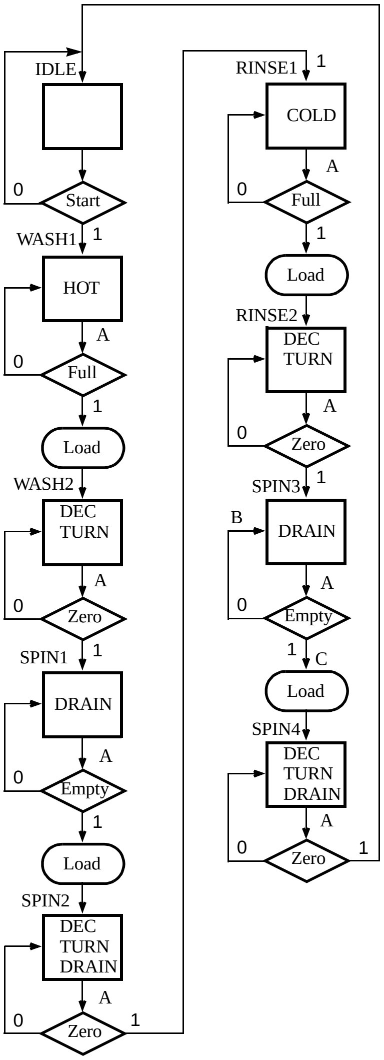


8-5.*

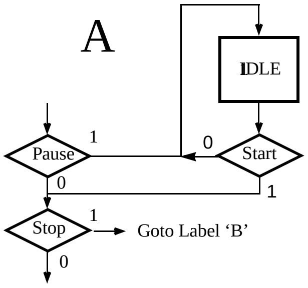


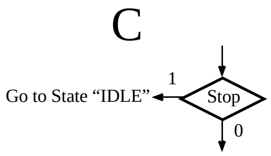
8-6.+

a) Regions marked A, B, and C are defined in part b.

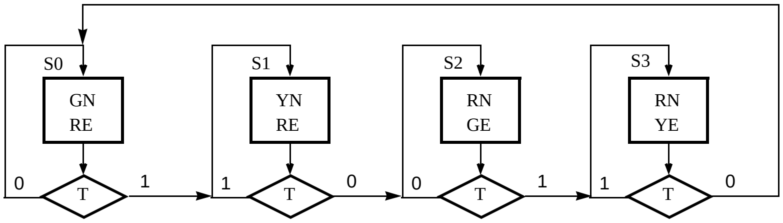


b) Insert the following in the ASM chart in the A and C regions, respectively:





8-7.



8-8.*

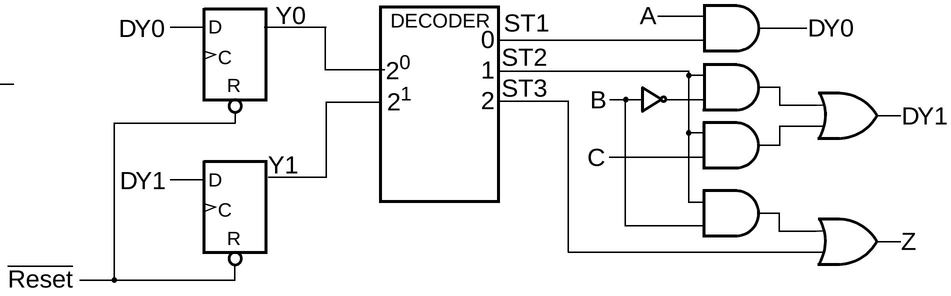
$ST1(t + 1) = ST1\cdot\overline{A} + ST2\cdot B\cdot\overline{C} + ST3$, $ST2(t + 1) = ST1\cdot A$, $ST3(t + 1)= ST2\cdot(\overline{B} + C)$, $Z = ST2\cdot B + ST3$

For the D flip-flops, $D_{STi} = STi(t + 1)$ and $STi = Q_{STi}$. Reset initializes the flip-flops: ST1 = 1, ST2 = ST3 = 0.

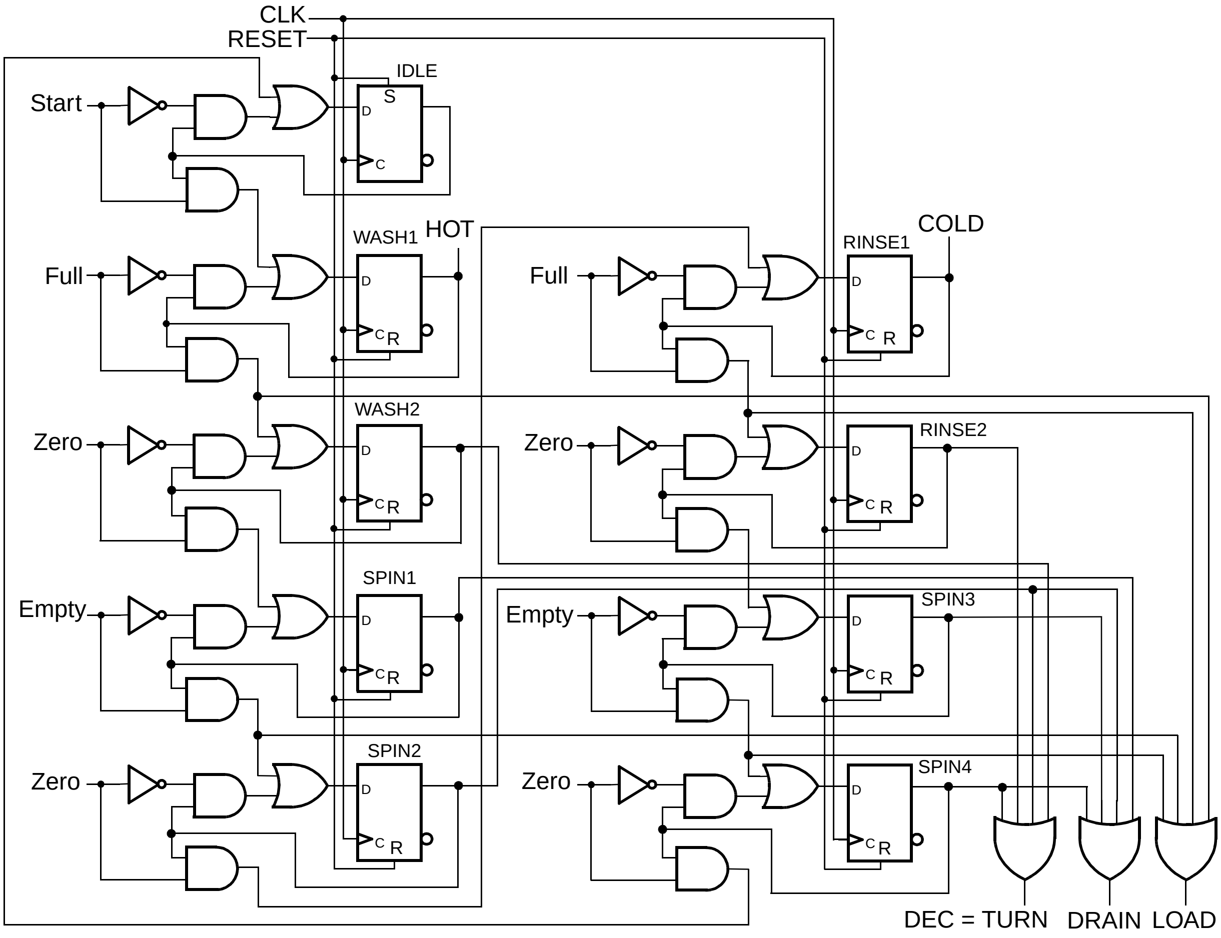
8-9.*

State Assignment

	Y1 Y0
ST1	0 0
ST2	0 1
ST3	1 0



8-10.+



8-11.*

```
     100110     (38)
   × 110101    (× 53)
     100110
    000000
   100110
  000000
 100110
100110
11111011110   (2014)
```

100110	
$\underline{110101}$	
000000	Init PP
$\underline{100110}$	Add
100110	After Add
0100110	After Shift
00100110	After Shift
$\underline{100110}$	Add
10111110	After Add
010111110	After Shift
0010111110	After Shift
$\underline{100110}$	Add
1100011110	After Add
01100011110	After Shift
$\underline{100110}$	Add
11111011110	After Add
$\underline{011111011110}$	After Shift

8-12.

STATE	P	B	C	A	Q
IDLE	X	1010	X	X	1011
MUL0	011	1010	0	1010	1011
MUL1	011	1010	0	0101	0101
MUL0	010	1010	0	1111	0101
MUL1	010	1010	0	0111	1010
MUL0	001	1010	0	0111	1010
MUL1	001	1010	0	0011	1101
MUL0	000	1010	0	1101	1101
MUL1	000	1010	0	0110	1110
IDLE	111	1010	0	0110	1110

8-13.

Time = 2(n+1)f

8-14.

Max Value 2n bits = $2^{2n} - 1$

Max Multiplicand = Max Multiplier = $2^n - 1$

$(2^n - 1) \times (2^n - 1) = 2^{2n} - 2^{(n+1)} + 1 \leq 2^{2n} - 1$? Yes, since $2^{(n+1)} \geq 2$ for $n > 0$

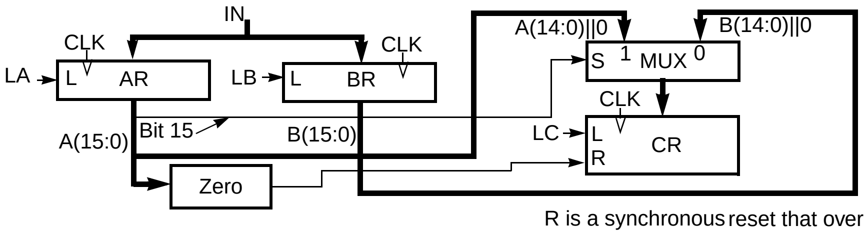
8-15.

(a) The maximum product size is 32 bits. The product is available in registers A and Q.

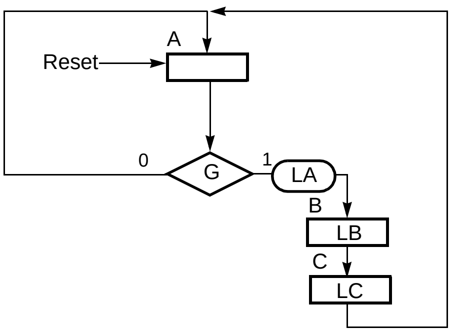
(b) The counter P is 4 bits wide. The initial value loaded is $(1111)_2$.

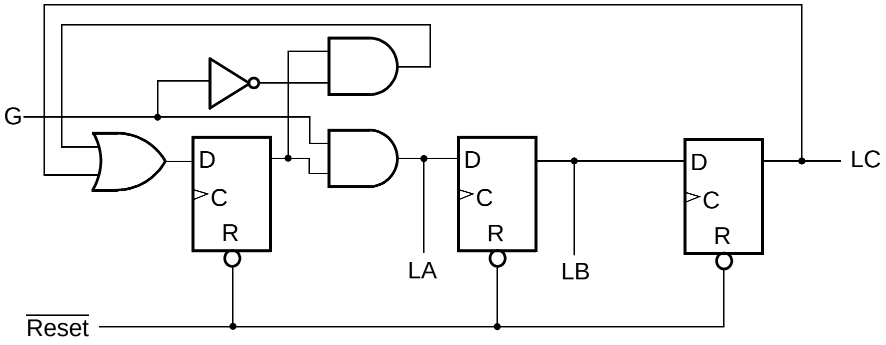
(c) Z is generated by a 4-bit NOR gate driven by the P counter.

8-16.*

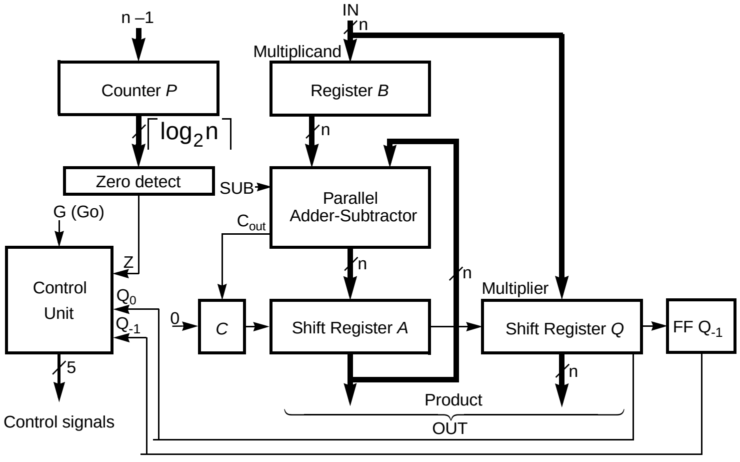


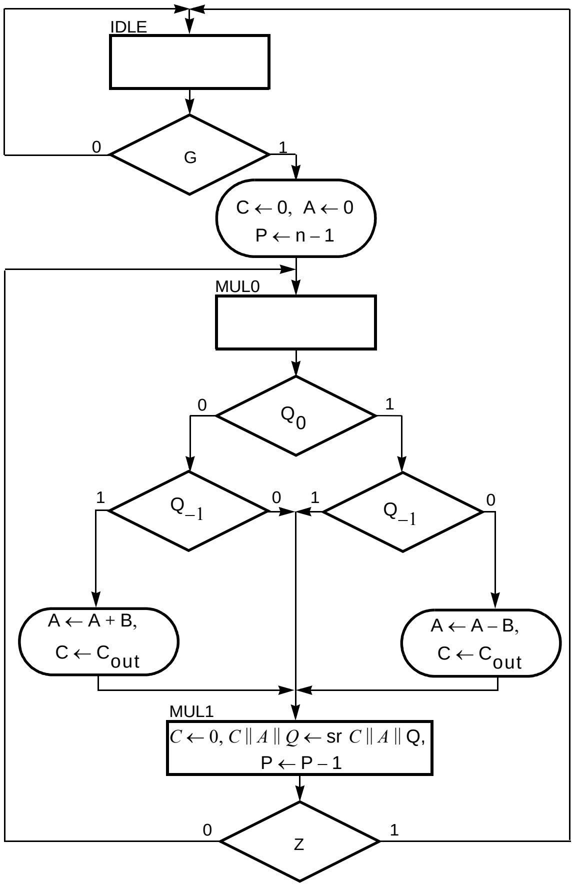
R is a synchronous reset that overides any simultaneous synchronous transfer.



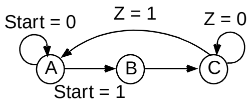


8-17.+

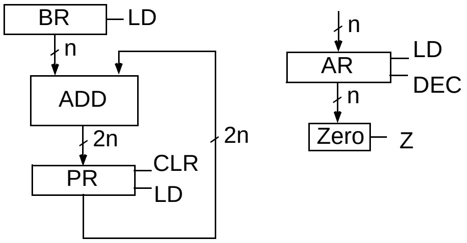


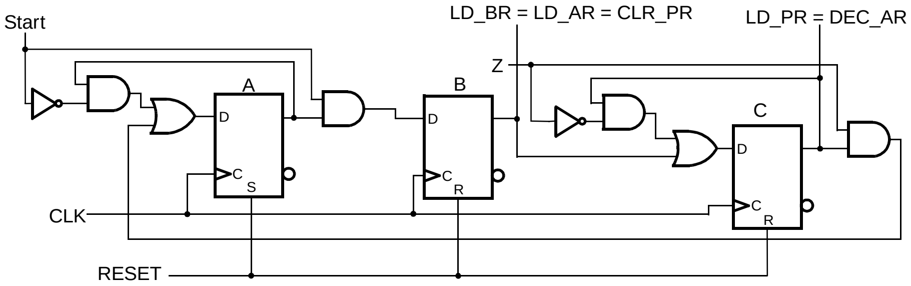


8-18.+



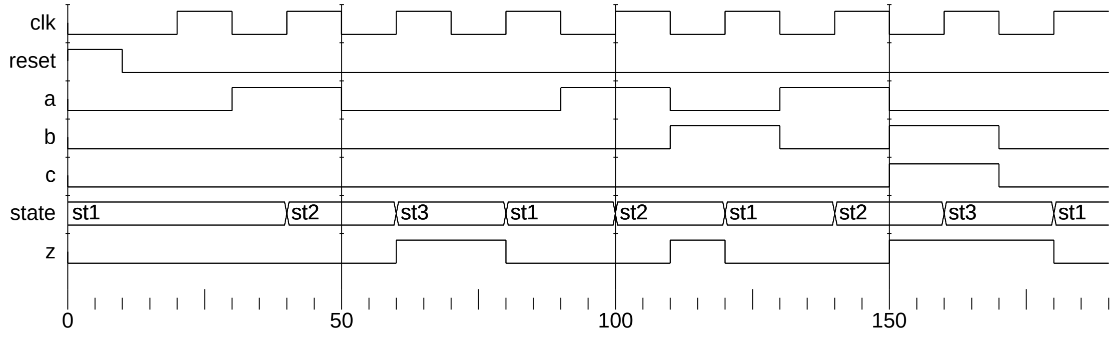
A: Initial State
B: BR <- Input A, AR <- Input B, PR <- 0
C: PR <- BR + PR, AR <- AR – 1





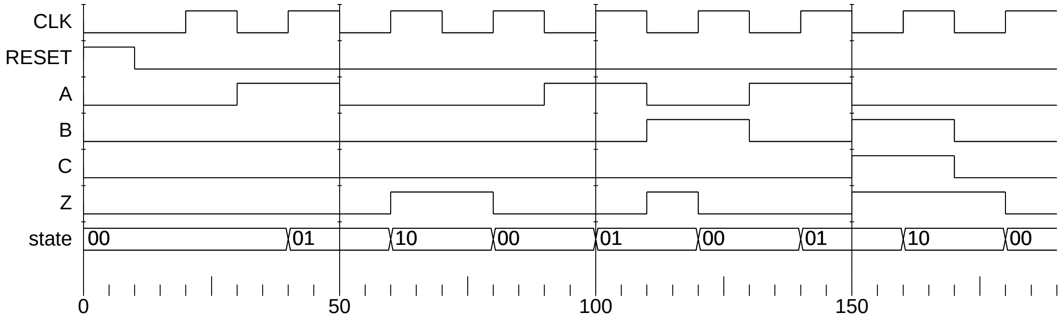
8-19.*

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

entity asm_819 is
    port (
        A, B, C, CLK, RESET: in STD_LOGIC;
        Z: out STD_LOGIC
    );
end asm_819;

architecture asm_819_arch of asm_819 is
    type state_type is (ST1, ST2, ST3);
    signal state, next_state : state_type;
begin

state_register: process (CLK, RESET)
begin
    if RESET='1' then--asynchronous RESET active High
        state <= ST1;
    elsif (CLK'event and CLK='1') then  --CLK rising edge
        state <= next_state;
    end if;
end process;

next_state_func: process (A, B, C, state)
begin
    case (state) is
        when ST1 =>
            if A = '0' then
                next_state <= ST1;
            else
                next_state <= ST2;
            end if;
        when ST2 =>
            if ((B = '1') and (C = '0')) then
                next_state <= ST1;
            else
                next_state <= ST3;
            end if;
        when ST3 =>
            next_state <= ST1;
    end case;
end process;

output_func: process (B, state)
begin
    case (state) is
        when ST1 =>
            Z <= '0';
        when ST2 =>
            if (B = '1') then
                Z <= '1';
            else
                Z <= '0';
            end if;
        when ST3 =>
            Z <= '1';
    end case;
end process;

end asm_819_arch;
```



8-20.*

```
module asm_820 (CLK, RESET, A, B, C, Z) ;

input CLK, RESET, A, B, C ;
output Z ;
reg [1:0] state, next_state;
parameter ST1=2'b00, ST2=2'b01, ST3=2'b10;
reg Z;

//State register

always @(posedge CLK or posedge RESET)
begin
   if (RESET)  //asynchronous RESET active High
      state <= ST1;
   else            //use CLK rising edge
      state <= next_state;
end

//Next state function
always @(A or B or C or state)
begin
     case (state)
          ST1: next_state <= A ? ST2: ST1;
          ST2: next_state <= (B && ! C) ? ST1: ST3;
          ST3: next_state <= ST1;
          default: next_state <= ST1;
     endcase
end

//Output function
always @(B or state)
begin
     case (state)
          ST1: Z <= 1'b0;
          ST2: Z <= B ? 1'b1: 1'b0;
          ST3: Z <= 1'b1;
          default: Z <= 0'b0;
     endcase
end

endmodule
```

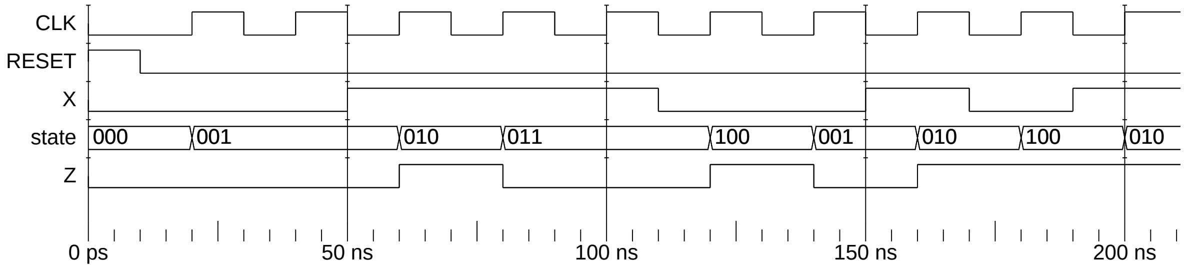


8-21.

```
module asm_821 (X, CLK, RESET, Z) ;
input X ;
input CLK ;
input RESET ;
output Z ;

reg [2:0] state, next_state;
parameter S0= 3'b000, S1 = 3'b001, S2 = 3'b010,
            S3 = 3'b011, S4 = 3'b100;
reg Z;
//State register
always@(posedge CLK or posedge RESET)
begin
   if (RESET == 1)
     state <= S0;
   else
      state <= next_state;
end
```

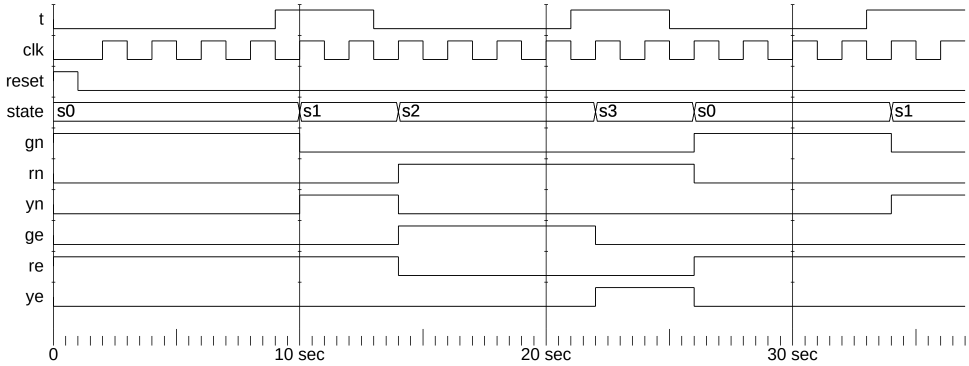
```
 //Next state function
always@(X or state)
begin
   case (state)
     S0: if (X) next_state <= S3;
           else next_state <= S1;
     S1: if (X) next_state <= S2;
            else next_state <= S1;
     S2: if (X) next_state <= S3;
            else next_state <= S4;
     S3: if (X) next_state <= S3;
            else next_state <= S4;
     S4: if (X) next_state <= S2;
            else next_state <= S1;
     default: next_state <= S0;
   endcase
end
```

```
//Output function
always@(state)
begin
  case (state)
    S2: Z <= 1'b1;
    S4: Z <= 1'b1;
    default: Z <= 1'b0;
  endcase
end
endmodule
```



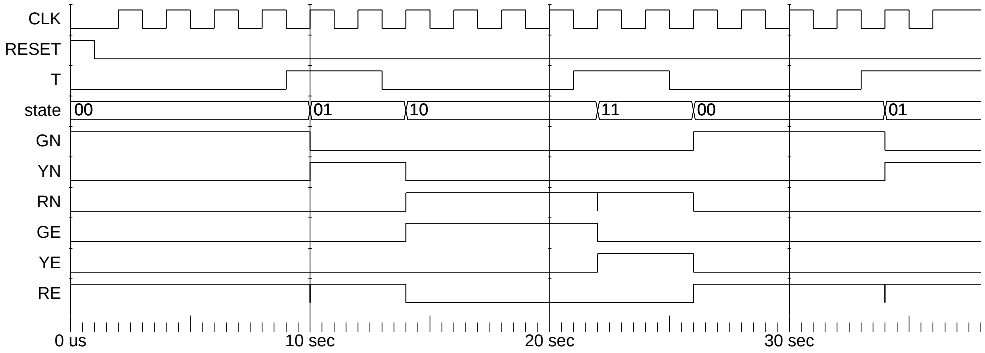
8-22.+

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

entity asm_822 is
   port (T: in STD_LOGIC;
      CLK: in STD_LOGIC;
      RESET: in STD_LOGIC;
      GN: out STD_LOGIC;
      RN: out STD_LOGIC;
      YN: out STD_LOGIC;
      GE: out STD_LOGIC;
      RE: out STD_LOGIC;
      YE: out STD_LOGIC);
end asm_822;

architecture asm_822_arch of asm_822  is
  type state_type is (S0, S1, S2, S3);
  signal state, next_state: state_type;
begin
state_register: process (CLK, RESET)
begin
  if (RESET = '1') then
    state <= S0;
  elsif (CLK'event and CLK = '1') then
    state <= next_state;
  end if;
end process;

next_state_func: process (T, state)
begin
  case state is
  when S0 =>
    if T = '1' then
      next_state <= S1;
    else
      next_state <= S0;
    end if;
  when S1 =>
    if T = '1' then
      next_state <= S1;
    else
      next_state <= S2;

    end if;
  when S2 =>
    if T = '1' then
      next_state <= S3;
    else
      next_state <= S2;
    end if;
  when S3 =>
    if T = '1' then
      next_state <= S3;
    else
      next_state <= S0;
    end if;
  when others => null;
  end case;
end process;

output_func: process (state)
begin
  GN <= '0'; RN <= '0'; YN <= '0';
  GE <= '0'; RE <= '0'; YE <= '0';
  case state is
    when S0 =>
      GN <= '1'; RE <= '1';
    when S1 =>
      YN <= '1'; RE <= '1';
    when S2 =>
      RN <= '1'; GE <= '1';
    when S3 =>
      RN <= '1'; YE <= '1';
    when others => null;
  end case;
end process;

end asm_822_arch;
```



8-23.+

```
module asm_823 (CLK, T, RESET, GN, YN, RN, GE, YE, RE) ;

input CLK, T, RESET;
output GN,YN, RN, GE, YE, RE;

reg [1:0] state, next_state;
parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
reg GN, YN, RN, GE, YE, RE;

//State register
always@(posedge CLK or posedge RESET)
begin
   if (RESET == 1'b1)
      state <= S0;
   else
      state <= next_state;
end

//Next state function
always@(T or state)
begin
   case (state)
      S0: if (T)
             next_state <= S1;
          else
              next_state <= S0;
      S1: if (T)
              next_state <= S1;
          else
              next_state <= S2;
      S2: if (T)
              next_state <= S3;
          else
              next_state <= S2;
      S3: if (T)
             next_state <= S3;
          else
             next_state <= S0;
      default: next_state <= S0;
   endcase
end

//Output function
always@(state)
begin
   GN <= 1'b0; RN <= 1'b0; YN <= 1'b0;
   GE <= 1'b0; RE <= 1'b0; YE <= 1'b0;
   case (state)
      S0: begin
         GN <= 1'b1; RE <= 1'b1;
         end
      S1: begin
         YN <= 1'b1; RE <= 1'b1;
         end
      S2: begin
         RN <= 1'b1; GE <= 1'b1;
         end
      S3: begin
         RN <= 1'b1; YE <= 1'b1;
          end
      default  begin
      GN <= 1'b0; RN <= 1'b1; YN <= 1'b0;
      GE <= 1'b0; RE <= 1'b0; YE <= 1'b1;
      end
   endcase
end
endmodule
```


CLK
RESET
T
state
GN
YN
RN
GE
YE
RE
0 us
10 sec
20 sec
30 sec