Online Instructor's Manual
to accompany

Digital Fundamentals
Tenth Edition

Thomas L. Floyd

PEARSON
Prentice Hall

Upper Saddle River, New Jersey
Columbus, Ohio

10 9 8 7 6 5 4 3 2 1

PEARSON
Prentice
Hall

ISBN-13: 978-0-13-712960-7
ISBN-10: 0-13-712960-2

CONTENTS

To access supplementary materials online, instructors need to request an instructor access code. Go to **<u>www.pearsonhighered.com/irc</u>**, where you can register for an access code. Within 48 hours after registering you will receive a confirming e-mail including an instructor access code. Once you have received your code, go the site and log on for full instructions on downloading the materials you wish to use.

NOTE: For access to hidden faults in Multisim circuits, the password is *book*.

PART 1

Problem Solutions

CHAPTER 1
INTRODUCTORY CONCEPTS

Section 1-1 Digital and Analog Quantities

1. Digital data can be transmitted and stored more efficiently and reliably than analog data. Also, digital circuits are simpler to implement and there is a greater immunity to noisy environments.

2. Pressure is an analog quantity.

3. A clock, a thermometer, and a speedometer can have either an analog or a digital output.

Section 1-2 Binary Digits, Logic Levels, and Digital Waveforms

4. In positive logic, a 1 is represented by a HIGH level and a 0 by a LOW level. In negative logic, a 1 is represented by a LOW level, and a 0 by a HIGH level.

5. HIGH = 1; LOW = 0. See Figure 1-1.

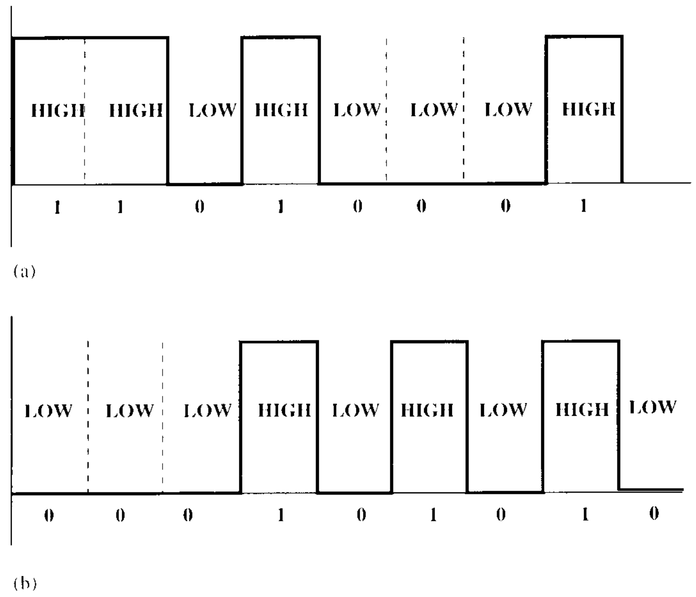


FIGURE 1-1

6. A 1 is a HIGH and a 0 is a LOW:
(a) HIGH, LOW, HIGH, HIGH, HIGH, LOW, HIGH
(b) HIGH, HIGH, HIGH, LOW, HIGH, LOW, LOW, HIGH

7. See Figure 1-2.

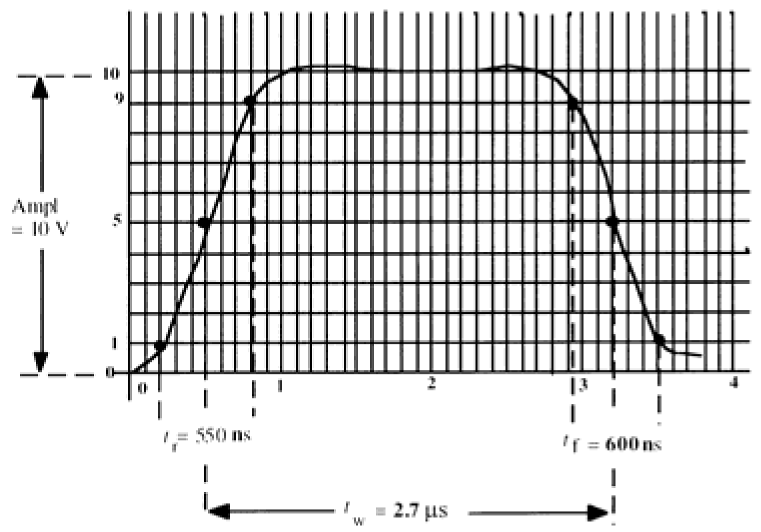


FIGURE 1-2

8. T = 4 ms. See Figure 1-3.

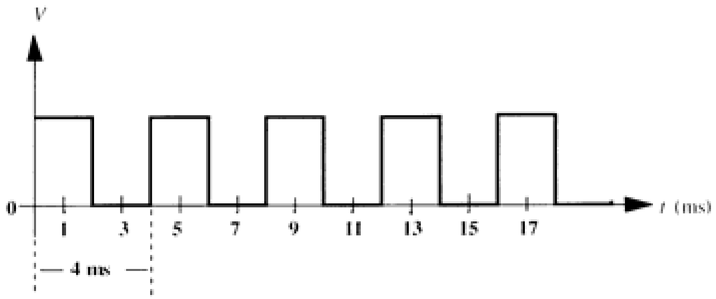


FIGURE 1-3

9. $f = \dfrac{1}{T} = \dfrac{1}{4\text{ ms}}$ = 0.25 kHz = 250 Hz

10. The waveform in Figure 1-61 is **periodic** because it repeats at a fixed interval.

11. t_W = 2 ms; T = 4 ms

% duty cycle = $\left(\dfrac{t_W}{T}\right)100 = \left(\dfrac{2\text{ ms}}{4\text{ ms}}\right)100$ = 50%

12. See Figure 1-4.

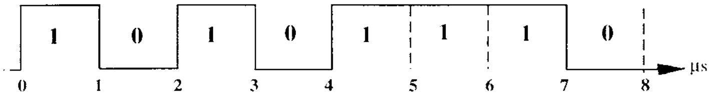


FIGURE 1-4

13. Each bit time = 1 μs
Serial transfer time = (8 bits)(1 μs/bit) = 8 μs

Parallel transfer time = 1 bit time = 1 μs

14. $T = \dfrac{1}{f} = \dfrac{1}{3.5\text{ GHz}} = \mathbf{0.286\text{ ns}}$

Section 1-3 Basic Logic Operations

15. L_{ON} = SW1 + SW2 + SW1 · SW2

16. An AND gate produces a HIGH output only when *all* of its inputs are HIGH.

17. AND gate. See Figure 1-5.

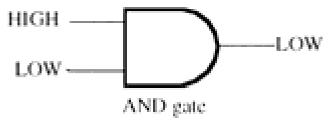
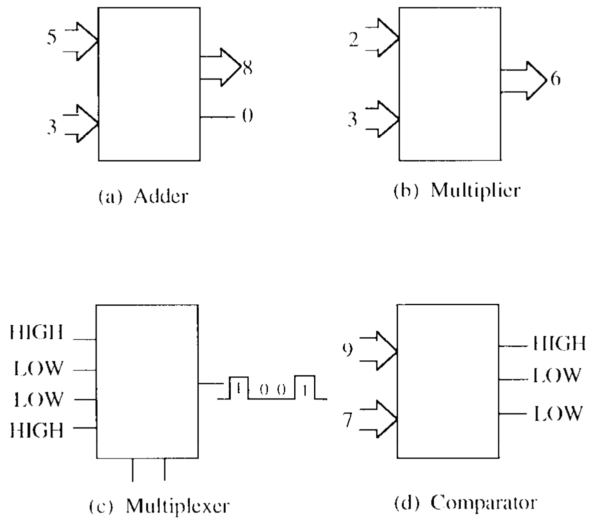


FIGURE 1-5

18. An OR gate produces a HIGH output when *either or both* inputs are HIGH. An exclusive-OR gate produces a HIGH if one input is HIGH and the other LOW.

Section 1-4 Introduction to the System Concept

19. See Figure 1-6.



(a) Adder

(b) Multiplier

(c) Multiplexer

(d) Comparator

FIGURE 1-6

20. $T = \dfrac{1}{10\,\text{kHz}} = 100\ \mu\text{s}$

Pulses counted $= \dfrac{100\,\text{ms}}{100\,\mu\text{s}} = 1000$

21. See Figure 1-7.

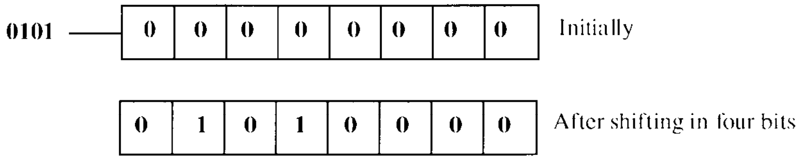


FIGURE 1-7

Section 1-5 Fixed-Function Integrated Circuits

22. Circuits with complexities of from 100 to 10,000 equivalent gates are classified as large scale integration (LSI).

23. The pins of an SMT are soldered to the pads on the surface of a pc board, whereas the pins of a DIP feed through and are soldered to the opposite side. Pin spacing on SMTs is less than on DIPs and therefore SMT packages are physically smaller and require less surface area on a pc board.

24. See Figure 1-8.

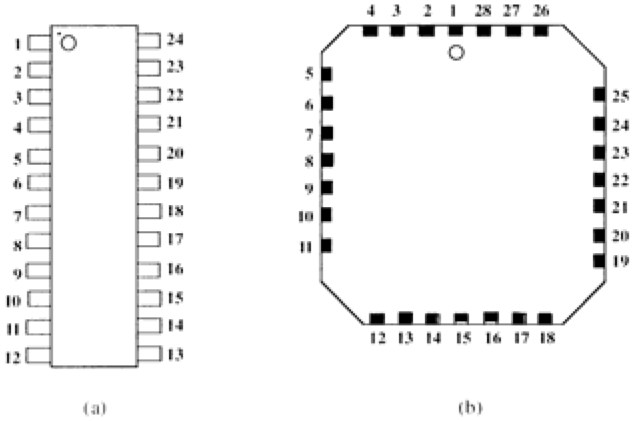


FIGURE 1-8

Section 1-6 Test and Measurement Instruments

25. Amplitude = top of pulse minus base line
V = 8 V – 1 V = 7 V

26. A flashing probe lamp indicates a continuous sequence of pulses (pulse train).

Section 1-7 Introduction to Programmable Logic

27. The following do not describe PLDs: VHDL, AHDL

28. SPLD: Simple Programmable Logic Device
CPLD: Complex Programmable Logic Device
HDL: Hardware Description Language
FPGA: Field-Programmable Gate Array
GAL: Generic Array Logic

29. (a) Design entry: The step in a programmable logic design flow where a description of the circuit is entered in either schematic (graphic) form or in text form using an HDL.

(b) Simulation: The step in a design flow where the entered design is simulated based on defined input waveforms.

(c) Compilation: A program process that controls the design flow process and translates a design source code to object code for testing and downloading.

(d) Download: The process in which the design is transferred from software to hardware.

30. Place and route or fitting is the process where the logic structures described by the netlist are mapped into the actual structure of the specific target device. This results in an output called a bitstream.

CHAPTER 2
NUMBER SYSTEMS, OPERATIONS, AND CODES

Section 2-1 Decimal Numbers

1. (a) $1386 = 1 \times 10^3 + 3 \times 10^2 + 8 \times 10^1 + 6 \times 10^0$
$= 1 \times 1000 + 3 \times 100 + 8 \times 10 + 6 \times 1$
The digit 6 has a weight of $10^0 = 1$

(b) $54{,}692 = 5 \times 10^4 + 4 \times 10^3 + 6 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$
$= 5 \times 10{,}000 + 4 \times 1000 + 6 \times 100 + 9 \times 10 + 2 \times 1$
The digit 6 has a weight of $10^2 = 100$

(c) $671{,}920 = 6 \times 10^5 + 7 \times 10^4 + 1 \times 10^3 + 9 \times 10^2 + 2 \times 10^1 + 0 \times 10^0$
$= 6 \times 100{,}000 + 7 \times 10{,}000 + 1 \times 1000 + 9 \times 100 + 2 \times 10 + 0 \times 1$
The digit 6 has a weight of $10^5 = 100{,}000$

2. (a) $10 = 10^1$ (b) $100 = 10^2$
(c) $10{,}000 = 10^4$ (d) $1{,}000{,}000 = 10^6$

3. (a) $471 = 4 \times 10^2 + 7 \times 10^1 + 1 \times 10^0$
$= 4 \times 100 + 7 \times 10 + 1 \times 1$
$= 400 + 70 + 1$

(b) $9{,}356 = 9 \times 10^3 + 3 \times 10^2 + 5 \times 10^1 + 6 \times 10^0$
$= 9 \times 1000 + 3 \times 100 + 5 \times 10 + 6 \times 1$
$= 9{,}000 + 300 + 50 + 6$

(c) $125{,}000 = 1 \times 10^5 + 2 \times 10^4 + 5 \times 10^3$
$= 1 \times 100{,}000 + 2 \times 10{,}000 + 5 \times 1000$
$= 100{,}000 + 20{,}000 + 5{,}000$

4. The highest four-digit decimal number is 9999.

Section 2-2 Binary Numbers

5. (a) $11 = 1 \times 2^1 + 1 \times 2^0 = 2 + 1 = 3$
(b) $100 = 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 4$
(c) $111 = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 4 + 2 + 1 = 7$
(d) $1000 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 8$
(e) $1001 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 8 + 1 = 9$
(f) $1100 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 = 8 + 4 = 12$
(g) $1011 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 2 + 1 = 11$
(h) $1111 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 4 + 2 + 1 = 15$

Chapter 2

6. (a) $1110 = 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 = 8 + 4 + 2 = 14$

(b) $1010 = 1 \times 2^3 + 1 \times 2^1 = 8 + 2 = 10$

(c) $11100 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 = 16 + 8 + 4 = 28$

(d) $10000 = 1 \times 2^4 = 16$

(e) $10101 = 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^0 = 16 + 4 + 1 = 21$

(f) $11101 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 = 16 + 8 + 4 + 1 = 29$

(g) $10111 = 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 4 + 2 + 1 = 23$

(h) $11111 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 16 + 8 + 4 + 2 + 1 = 31$

7. (a) $110011.11 = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2}$
$= 32 + 16 + 2 + 1 + 0.5 + 0.25 = 51.75$

(b) $101010.01 = 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^{-2} = 32 + 8 + 2 + 0.25$
$= 42.25$

(c) $1000001.111 = 1 \times 2^6 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}$
$= 64 + 1 + 0.5 + 0.25 + 0.125 = 65.875$

(d) $1111000.101 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^{-1} + 1 \times 2^{-3}$
$= 64 + 32 + 16 + 8 + 0.5 + 0.125 = 120.625$

(e) $1011100.10101 = 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^{-1} + 1 \times 2^{-3} + 1 \times 2^{-5}$
$= 64 + 16 + 8 + 4 + 0.5 + 0.125 + 0.03125$
$= 92.65625$

(f) $1110001.0001 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^0 + 1 \times 2^{-4}$
$= 64 + 32 + 16 + 1 + 0.0625 = 113.0625$

(g) $1011010.1010 = 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-3}$
$= 64 + 16 + 8 + 2 + 0.5 + 0.125 = 90.625$

(h) $1111111.11111 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1$
$+ 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4} + 1 \times 2^{-5}$
$= 64 + 32 + 16 + 8 + 4 + 2 + 1 + 0.5 + 0.25 + 0.125 + 0.0625 + 0.03125$
$= 127.96875$

8. (a) $2^2 - 1 = 3$ (b) $2^3 - 1 = 7$

(c) $2^4 - 1 = 15$ (d) $2^5 - 1 = 31$

(e) $2^6 - 1 = 63$ (f) $2^7 - 1 = 127$

(g) $2^8 - 1 = 255$ (h) $2^9 - 1 = 511$

(i) $2^{10} - 1 = 1023$ (j) $2^{11} - 1 = 2047$

9. (a) $(2^4 - 1) < 17 < (2^5 - 1)$; 5 bits

(b) $(2^5 - 1) < 35 < (2^6 - 1)$; 6 bits

(c) $(2^5 - 1) < 49 < (2^6 - 1)$; 6 bits

(d) $(2^6 - 1) < 68 < (2^7 - 1)$; 7 bits

(e) $(2^6 - 1) < 81 < (2^7 - 1)$; 7 bits

(f) $(2^6 - 1) < 114 < (2^7 - 1)$; 7 bits

(g) $(2^7 - 1) < 132 < (2^8 - 1)$; 8 bits

(h) $(2^7 - 1) < 205 < (2^8 - 1)$; 8 bits

10. (a) 0 through 7:
000, 001, 010, 011, 100, 101, 110, 111

(b) 8 through 15:
1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

(c) 16 through 31:
10000, 10001, 10010, 10011, 10100, 10101, 10110, 10111, 11000, 11001, 11010, 11011, 11100, 11101, 11110, 11111

(d) 32 through 63:
100000, 100001, 100010, 100011, 100100, 100101, 100110, 100111, 10100, 101001, 101010, 101011, 101100, 101101, 101110, 101111, 110000, 110001, 110010, 110011, 110100, 110101, 110110, 110111, 111000, 111001, 111010, 111011, 111100, 111101, 111110, 111111

(e) 64 through 75:
1000000, 1000001, 1000010, 1000011, 1000100, 1000101, 1000110, 1000111, 1001000, 1001001, 1001010, 1001011

Section 2-3 Decimal-to-Binary Conversion

11. (a) $10 = 8 + 2 = 2^3 + 2^1 = 1010$
(b) $17 = 16 + 1 = 2^4 + 2^0 = 10001$
(c) $24 = 16 + 8 = 2^4 + 2^3 = 11000$
(d) $48 = 32 + 16 = 2^5 + 2^4 = 110000$
(e) $61 = 32 + 16 + 8 + 4 + 1 = 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = 111101$
(f) $93 = 64 + 16 + 8 + 4 + 1 = 2^6 + 2^4 + 2^3 + 2^2 + 2^0 = 1011101$
(g) $125 = 64 + 32 + 16 + 8 + 4 + 1 = 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^0 = 1111101$
(h) $186 = 128 + 32 + 16 + 8 + 2 = 2^7 + 2^5 + 2^4 + 2^3 + 2^1 = 10111010$

12. (a) $0.32 \cong 0.00 + 0.25 + 0.0625 + 0.0 + 0.0 + 0.0078125 = 0.0101001$
(b) $0.246 \cong 0.0 + 0.0 + 0.125 + 0.0625 + 0.03125 + 0.015625 = 0.001111$
(c) $0.0981 \cong 0.0 + 0.0 + 0.0 + 0.0625 + 0.03125 + 0.0 + 0.0 + 0.00390625 = 0.0001101$

Chapter 2

13. (a)

$\frac{15}{2} = 7,\ R = 1\text{(LSB)}$

$\frac{7}{2} = 3,\ R = 1$

$\frac{3}{2} = 1,\ R = 1$

$\frac{1}{2} = 0,\ R = 1\text{ (MSB)}$

(b)

$\frac{21}{2} = 10,\ R = 1\text{ (LSB)}$

$\frac{10}{2} = 5,\quad R = 0$

$\frac{5}{2} = 2,\quad R = 1$

$\frac{2}{2} = 1,\quad R = 0$

$\frac{1}{2} = 0,\quad R = 1\text{ (MSB)}$

(c)

$\frac{28}{2} = 14,\quad R = 0\text{ (LSB)}$

$\frac{14}{2} = 7,\quad R = 0$

$\frac{7}{2} = 3,\quad R = 1$

$\frac{3}{2} = 1,\quad R = 1$

$\frac{1}{2} = 0,\quad R = 1\text{ (MSB)}$

(d)

$\frac{34}{2} = 17,\ R = 0\text{ (LSB)}$

$\frac{17}{2} = 8,\quad R = 1$

$\frac{8}{2} = 4,\quad R = 0$

$\frac{4}{2} = 2,\quad R = 0$

$\frac{2}{2} = 1,\quad R = 0$

$\frac{1}{2} = 0,\quad R = 1\text{ (MSB)}$

(e)

$\frac{40}{2} = 20,\ R = 0\text{ (LSB)}$

$\frac{20}{2} = 10,\ R = 0$

$\frac{10}{2} = 5,\quad R = 0$

$\frac{5}{2} = 2,\quad R = 1$

$\frac{2}{2} = 1,\quad R = 0$

$\frac{1}{2} = 0,\quad R = 1\text{ (MSB)}$

(f)

$\frac{59}{2} = 29,\ R = 1\text{ (LSB)}$

$\frac{29}{2} = 14,\ R = 1$

$\frac{14}{2} = 7,\quad R = 0$

$\frac{7}{2} = 3,\quad R = 1$

$\frac{3}{2} = 1,\quad R = 1$

$\frac{1}{2} = 0,\quad R = 1\text{ (MSB)}$

(g)

$\frac{65}{2} = 32,\ R = 1\text{ (LSB)}$

$\frac{32}{2} = 16,\ R = 0$

$\frac{16}{2} = 8,\quad R = 0$

$\frac{8}{2} = 4,\quad R = 0$

$\frac{4}{2} = 2,\quad R = 0$

$\frac{2}{2} = 1,\quad R = 0$

$\frac{1}{2} = 0,\quad R = 1\text{(MSB)}$

(h)

$\frac{73}{2} = 36,\ R = 1\text{ (LSB)}$

$\frac{36}{2} = 18,\ R = 0$

$\frac{18}{2} = 9,\quad R = 0$

$\frac{9}{2} = 4,\quad R = 1$

$\frac{4}{2} = 2,\quad R = 0$

$\frac{2}{2} = 1,\quad R = 0$

$\frac{1}{2} = 0,\quad R = 1\text{ (MSB)}$

14. (a) 0.98 × 2 = 1.96 1 (MSB)
0.96 × 2 = 1.92 1
0.92 × 2 = 1.84 1
0.84 × 2 = 1.68 1
0.68 × 2 = 1.36 1
0.36 × 2 = 0.72 0
continue if more accuracy is desired
0.111110

(b) 0.347 × 2 = 0.694 0 (MSB)
0.694 × 2 = 1.388 1
0.388 × 2 = 0.776 0
0.776 × 2 = 1.552 1
0.552 × 2 = 1.104 1
0.104 × 2 = 0.208 0
0.208 × 2 = 0.416 0
continue if more accuracy is desired
0.0101100

(c) 0.9028 × 2 = 1.8056 1 (MSB)
0.8056 × 2 = 1.6112 1
0.6112 × 2 = 1.2224 1
0.2224 × 2 = 0.4448 0
0.4448 × 2 = 0.8896 0
0.8896 × 2 = 1.7792 1
0.7792 × 2 = 1.5584 1
continue if more accuracy is desired
0.1110011

Section 2-4 Binary Arithmetic

15. (a) $$\begin{array}{r} 11 \\ +\,01 \\ \hline 100 \end{array}$$ (b) $$\begin{array}{r} 10 \\ +\,10 \\ \hline 100 \end{array}$$ (c) $$\begin{array}{r} 101 \\ +\,011 \\ \hline 1000 \end{array}$$

(d) $$\begin{array}{r} 111 \\ +\,110 \\ \hline 1101 \end{array}$$ (e) $$\begin{array}{r} 1001 \\ +\,0101 \\ \hline 1110 \end{array}$$ (f) $$\begin{array}{r} 1101 \\ +\,1011 \\ \hline 11000 \end{array}$$

16. (a) $$\begin{array}{r} 11 \\ -\,01 \\ \hline 10 \end{array}$$ (b) $$\begin{array}{r} 101 \\ -\,100 \\ \hline 001 \end{array}$$ (c) $$\begin{array}{r} 110 \\ -\,101 \\ \hline 001 \end{array}$$

(d) $$\begin{array}{r} 1110 \\ -\,0011 \\ \hline 1011 \end{array}$$ (e) $$\begin{array}{r} 1100 \\ -\,1001 \\ \hline 0011 \end{array}$$ (f) $$\begin{array}{r} 11010 \\ -\,10111 \\ \hline 00011 \end{array}$$

17. (a) $$\begin{array}{r} 11 \\ \times 11 \\ \hline 11 \\ 11 \\ \hline 1001 \end{array}$$ (b) $$\begin{array}{r} 100 \\ \times 10 \\ \hline 000 \\ 100 \\ \hline 1000 \end{array}$$ (c) $$\begin{array}{r} 111 \\ \times 101 \\ \hline 111 \\ 000 \\ 111 \\ \hline 100011 \end{array}$$ (d) $$\begin{array}{r} 1001 \\ \times 110 \\ \hline 0000 \\ 1001 \\ 1001 \\ \hline 110110 \end{array}$$

(e) $$\begin{array}{r} 1101 \\ \times 1101 \\ \hline 1101 \\ 0000 \\ 1101 \\ 1101 \\ \hline 10101001 \end{array}$$ (f) $$\begin{array}{r} 1110 \\ \times 1101 \\ \hline 1110 \\ 0000 \\ 1110 \\ 1110 \\ \hline 10110110 \end{array}$$

18. (a) $\dfrac{100}{10} = 010$ (b) $\dfrac{1001}{0011} = 0011$ (c) $\dfrac{1100}{0100} = 0011$

Section 2-5 1's and 2's Complements of Binary Numbers

19. Zero is represented in 1's complement as all 0's (for +0) or all 1's (for –0).

20. Zero is represented by all 0's only in 2's complement.

21. (a) The 1's complement of 101 is 010.
(b) The 1's complement of 110 is 001.
(c) The 1's complement of 1010 is 0101.
(d) The 1's complement of 11010111 is 00101000.
(e) The 1's complement of 1110101 is 0001010.
(f) The 1's complement of 00001 is 11110.

22. Take the 1's complement and add 1:

(a) 01 + 1 = 10 (b) 000 + 1 = 001
(c) 0110 + 1 = 0111 (d) 0010 + 1 = 0011
(e) 00011 + 1 = 00100 (f) 01100 + 1 = 01101
(g) 01001111 + 1 = 01010000 (h) 11000010 + 1 = 11000011

Section 2-6 Signed Numbers

23. (a) Magnitude of 29 = 0011101
+ 29 = 00011101

(b) Magnitude of 85 = 1010101
–85 = 11010101

(c) Magnitude of 100_{10} = 1100100
+100 = 01100100

(d) Magnitude of 123 = 1111011
–123 = 11111011

24. (a) Magnitude of 34 = 0100010
–34 = 11011101

(b) Magnitude of 57 = 0111001
+57 = 00111001

(c) Magnitude of 99 = 1100011
–99 = 10011100

(d) Magnitude of 115 = 1110011
+115 = 01110011

25. (a) Magnitude of 12 = 1100
+12 = 00001100

(b) Magnitude of 68 = 1000100
–68 = 10111100

(c) Magnitude of 101_{10} = 1100101
$+101_{10}$ = 01100101

(d) Magnitude of 125 = 1111101
–125 = 10000011

26. (a) 10011001 = –25 (b) 01110100 = +116 (c) 10111111 = –63

27. (a) 10011001 = –(01100110) = –102
(b) 01110100 = +(1110100) = +116
(c) 10111111 = –(1000000) = –64

28. (a) 10011001 = –(1100111) = –103
(b) 01110100 = +(1110100) = +116
(c) 10111111 = –(1000001) = –65

29. (a) 0111110000101011 → sign = 0
$1.11110000101011 \times 2^{14}$ → exponent = 127 + 14 + 141 = 10001101
Mantissa = 11110000101011000000000
01000110111110000101011000000000

(b) 100110000011000 → sign = 1
$1.10000011000 \times 2^{11}$ → exponent = 127 + 11 = 138 = 10001010
Mantissa = 11000001100000000000000
11000101011000001100000000000000

30. (a) 11000000101001001110001000000000
Sign = 1
Exponent = 10000001 = 129 – 127 = 2
Mantissa = $1.01001001110001 \times 2^2$ = 101.001001110001
–101.001001110001 = **–5.15258789**

(b) 01100110010000111110100100000000
Sign = 0
Exponent = 11001100 = 204 – 127 = 77
Mantissa = 1.100001111101001
$1.100001111101001 \times 2^{77}$

Section 2-7 Arithmetic Operations with Signed Numbers

31. (a)
$$\begin{array}{r} 33 = 00100001 \\ 15 = 00001111 \end{array} \qquad \begin{array}{r} 00100001 \\ \underline{+\ 00001111} \\ 00110000 \end{array}$$

(b)
$$\begin{array}{r} 56 = 00111000 \\ 27 = 00011011 \\ -27 = 11100101 \end{array} \qquad \begin{array}{r} 00111000 \\ \underline{+\ 11100101} \\ 00011101 \end{array}$$

(c)
$$\begin{array}{r} 46 = 00101110 \\ -46 = 11010010 \\ 25 = 00011001 \end{array} \qquad \begin{array}{r} 11010010 \\ \underline{+\ 00011001} \\ 11101011 \end{array}$$

(d)
$$\begin{array}{r} 110_{10} = 01101110 \\ -110_{10} = 10010010 \\ 84 = 01010100 \\ -84 = 10101100 \end{array} \qquad \begin{array}{r} 10010010 \\ \underline{+\ 10101100} \\ 100111110 \end{array}$$

32. (a)
$$\begin{array}{r} 00010110 \\ \underline{+\ 00110011} \\ 01001001 \end{array}$$

(b)
$$\begin{array}{r} 01110000 \\ \underline{+\ 10101111} \\ 100011111 \end{array}$$

33. (a)
$$\begin{array}{r} 10001100 \\ \underline{+\ 00111001} \\ 11000101 \end{array}$$

(b)
$$\begin{array}{r} 11011001 \\ \underline{+\ 11100111} \\ 11000000 \end{array}$$

34. (a)
$$\begin{array}{r} 00110011 \\ \underline{-\ 00010000} \end{array} \qquad \begin{array}{r} 00110011 \\ \underline{+\ 11110000} \\ \cancel{1}\ 00100011 \end{array}$$

(b)
$$\begin{array}{r} 01100101 \\ \underline{-\ 11101000} \end{array} \qquad \begin{array}{r} 01100101 \\ \underline{+\ 00011000} \\ 01111101 \end{array}$$

35.
$$\begin{array}{r} 01101010 \\ \underline{\times\ 11110001} \end{array} \qquad \begin{array}{r} 01101010 \\ \underline{\times\ 00001111} \\ 01101010 \\ \underline{01101010} \\ 100111110 \\ \underline{01101010} \\ 1011100110 \\ \underline{01101010} \\ 11000110110 \end{array}$$

Changing to 2’s complement with sign: 100111001010

36.
$$\frac{01000100}{00011001} = 00000010$$
$$\frac{68}{25} = 2, \text{ remainder of } 18$$

Section 2-8 Hexadecimal Numbers

37. (a) 38_{16} = 0011 1000
(b) 59_{16} = 0101 1001
(c) $A14_{16}$ = 1010 0001 0100
(d) $5C8_{16}$ = 0101 1100 1000
(e) 4100_{16} = 0100 0001 0000 0000
(f) $FB17_{16}$ = 1111 1011 0001 0111
(g) $8A9D_{16}$ = 1000 1010 1001 1101

38. (a) 1110 = E_{16}
(b) 10 = 2_{16}
(c) 0001 0111 = 17_{16}
(d) 1010 0110 = $A6_{16}$
(e) 0011 1111 0000 = $3F0_{16}$
(f) 1001 1000 0010 = 982_{16}

39. (a) $23_{16} = 2 \times 16^1 + 3 \times 16^0 = 32 + 3 = 35$
(b) $92_{16} = 9 \times 16^1 + 2 \times 16^0 = 144 + 2 = 146$
(c) $1A_{16} = 1 \times 16^1 + 10 \times 16^0 = 16 + 10 = 26$
(d) $8D_{16} = 8 \times 16^1 + 13 \times 16^0 = 128 + 13 = 141$
(e) $F3_{16} = 15 \times 16^1 + 3 \times 16^0 = 240 + 3 = 243$
(f) $EB_{16} = 14 \times 16^1 + 11 \times 16^0 = 224 + 11 = 235$
(g) $5C2_{16} = 5 \times 16^2 + 12 \times 16^1 + 2 \times 16^0 = 1280 + 192 + 2 = 1474$
(h) $700_{16} = 7 \times 16^2 = 1792$

40. (a) $\frac{8}{16}$ = 0, remainder = 8
hexadecimal number = 8_{16}

(b) $\frac{14}{16}$ = 0, remainder = 14 = E_{16}
hexadecimal number = E_{16}

(c) $\frac{33}{16}$ = 2, remainder = 1 (LSD)
$\frac{2}{16}$ = 0, remainder = 2
hexadecimal number = 21_{16}

(d) $\frac{52}{16}$ = 3, remainder = 4 (LSD)
$\frac{3}{16}$ = 0, remainder = 3
hexadecimal number = 34_{16}

(e) $\frac{284}{16}$ = 17, remainder = 12 = C_{16} (LSD)
$\frac{17}{16}$ = 1, remainder = 1
$\frac{1}{16}$ = 0, remainder = 1
hexadecimal number = $11C_{16}$

(f) $\frac{2890}{16}$ = 180, remainder = 10 = A_{16} (LSD)
$\frac{180}{16}$ = 11, remainder = 4
$\frac{11}{16}$ = 0 , remainder = 11 = B_{16}
hexadecimal number = $B4A_{16}$

(g) $\frac{4019}{16}$ = 251, remainder = 3 (LSD)
$\frac{251}{16}$ = 15, remainder = 11 = B_{16}
$\frac{15}{16}$ = 0, remainder = 15 = F_{16}
hexadecimal number = $FB3_{16}$

(h) $\frac{6500}{16}$ = 406, remainder = 4 (LSD)
$\frac{406}{16}$ = 25, remainder = 6
$\frac{25}{16}$ = 1, remainder = 9
$\frac{1}{16}$ = 0, remainder = 1
hexadecimal number = 1964_{16}

41. (a) $37_{16} + 29_{16} = 60_{16}$
(b) $A0_{16} + 6B_{16} = 10B_{16}$
(c) $FF_{16} + BB_{16} = 1BA_{16}$

42. (a) $51_{16} – 40_{16} = 11_{16}$
(b) $C8_{16} – 3A_{16} = 8E_{16}$
(c) $FD_{16} – 88_{16} = 75_{16}$

Section 2-9 Octal Numbers

43. (a) $12_8 = 1 \times 8^1 + 2 \times 8^0 = 8 + 2 = 10$
(b) $27_8 = 2 \times 8^1 + 7 \times 8^0 = 16 + 7 = 23$
(c) $56_8 = 5 \times 8^1 + 6 \times 8^0 = 40 + 6 = 46$
(d) $64_8 = 6 \times 8^1 + 4 \times 8^0 = 48 + 4 = 52$
(e) $103_8 = 1 \times 8^2 + 3 \times 8^0 = 64 + 3 = 67$
(f) $557_8 = 5 \times 8^2 + 5 \times 8^1 + 7 \times 8^0 = 320 + 40 + 7 = 367$
(g) $163_8 = 1 \times 8^2 + 6 \times 8^1 + 3 \times 8^0 = 64 + 48 + 3 = 115$
(h) $1024_8 = 1 \times 8^3 + 2 \times 8^1 + 4 \times 8^0 = 512 + 16 + 4 = 532$
(i) $7765_8 = 7 \times 8^3 + 7 \times 8^2 + 6 \times 8^1 + 5 \times 8^0 = 3584 + 448 + 48 + 5 = 4085$

44. (a) $\frac{15}{8}$ = 1, remainder = 7 (LSD)
$\frac{1}{8}$ = 0, remainder =1
octal number = 17_8

(b) $\frac{27}{8}$ = 3, remainder = 3 (LSD)
$\frac{3}{8}$ = 0, remainder = 3
octal number = 33_8

(c) $\frac{46}{8}$ = 5, remainder = 6 (LSD)
$\frac{5}{8}$ = 0, remainder = 5
octal number = 56_8

(d) $\frac{70}{8}$ = 8, remainder = 6 (LSD)
$\frac{8}{8}$ = 1, remainder = 0
$\frac{1}{8}$ = 0, remainder = 1
octal number = 106_8

(e) $\frac{100}{8}$ = 12, remainder = 4 (LSD)
$\frac{12}{8}$ = 1, remainder = 4
$\frac{1}{8}$ = 0, remainder = 1
octal number = 144_8

(f) $\frac{142}{8}$ = 17, remainder = 6 (LSD)
$\frac{17}{8}$ = 2, remainder = 1
$\frac{2}{8}$ = 0, remainder = 2
octal number = 216_8

(g) $\frac{219}{8}$ = 27, remainder = 3 (LSD)
$\frac{27}{8}$ = 3, remainder = 3
$\frac{3}{8}$ = 0, remainder = 3
octal number = 333_8

(h) $\frac{435}{8}$ = 54, remainder = 3 (LSD)
$\frac{54}{8}$ = 6, remainder = 6
$\frac{6}{8}$ = 0, remainder = 6
octal number = 663_8

45. (a) 13_8 = 001 011
(b) 57_8 = 101 111
(c) 101_8 = 001 000 001
(d) 321_8 = 011 010 001
(e) 540_8 = 101 100 000
(f) 4653_8 = 100 110 101 011
(g) 13271_8 = 001 011 010 111 001
(h) 45600_8 = 100 101 110 000 000
(i) 100213_8 = 001 000 000 010 001 011

46. (a) 111 = 7_8
(b) 010 = 2_8
(c) 110 111 = 67_8
(d) 101 010 = 52_8
(e) 001 100 = 14_8
(f) 001 011 110 = 136_8
(g) 101 100 011 001 = 5431_8
(h) 010 110 000 011 = 2603_8
(i) 111 111 101 111 000 = 77570_8

Section 2-10 Binary Coded Decimal (BCD)

47. (a) 10 = 0001 0000
(b) 13 = 0001 0011
(c) 18 = 0001 1000
(d) 21 = 0010 0001
(e) 25 = 0010 0101
(f) 36 = 0011 0110
(g) 44 = 0100 0100
(h) 57 = 0101 0111
(i) 69 = 0110 1001
(j) 98 = 1001 1000
(k) 125 = 0001 0010 0101
(l) 156 = 0001 0101 0110

48. (a) 10 = 1010_2 4 bits binary, 8 bits BCD
(b) 13 = 1101_2 4 bits binary, 8 bits BCD
(c) 18 = 10010_2 5 bits binary, 8 bits BCD
(d) 21 = 10101_2 5 bits binary, 8 bits BCD
(e) 25 = 11001_2 5 bits binary, 8 bits BCD
(f) 36 = 100100_2 6 bits binary, 8 bits BCD
(g) 44 = 101100_2 6 bits binary, 8 bits BCD
(h) 57 = 111001_2 6 bits binary, 8 bits BCD
(i) 69 = 1000101_2 7 bits binary, 8 bits BCD
(j) 98 = 1100010_2 7 bits binary, 8 bits BCD
(k) 125 = 1111101_2 7 bits binary, 12 ibts BCD
(l) 156 = 10011100_2 8 bits binary, 12 bits BCD

49. (a) 104 = 0001 0000 0100
(b) 128 = 0001 0010 1000
(c) 132 = 0001 0011 0010
(d) 150 = 0001 0101 0000
(e) 186 = 0001 1000 0110
(f) 210 = 0010 0001 0000
(g) 359 = 0011 0101 1001
(h) 547 = 0101 0100 0111
(i) 1051 = 0001 0000 0101 0001

50. (a) 0001 = 1 (b) 0110 = 6
(c) 1001 = 9 (d) 0001 1000 = 18
(e) 0001 1001 = 19 (f) 0011 0010 = 32
(g) 0100 0101 = 45 (h) 1001 1000 = 98
(i) 1000 0111 0000 = 870

51. (a) 1000 0000 = 80
(b) 0010 0011 0111 = 237
(c) 0011 0100 0110 = 346
(d) 0100 0010 0001 = 421
(e) 0111 0101 0100 = 754
(f) 1000 0000 0000 = 800
(g) 1001 0111 1000 = 978
(h) 0001 0110 1000 0011 = 1683
(i) 1001 0000 0001 1000 = 9018
(j) 0110 0110 0110 0111 = 6667

52. (a)
$$\begin{array}{r} 0010 \\ +\ 0001 \\ \hline 0011 \end{array}$$
(b)
$$\begin{array}{r} 0101 \\ +\ 0011 \\ \hline 1000 \end{array}$$
(c)
$$\begin{array}{r} 0111 \\ +\ 0010 \\ \hline 1001 \end{array}$$

(d)
$$\begin{array}{r} 1000 \\ +\ 0001 \\ \hline 1001 \end{array}$$
(e)
$$\begin{array}{r} 00011000 \\ +\ 00010001 \\ \hline 00101001 \end{array}$$
(f)
$$\begin{array}{r} 01100100 \\ +\ 00110011 \\ \hline 10010111 \end{array}$$

(g)
$$\begin{array}{r} 01000000 \\ +\ 01000111 \\ \hline 10000111 \end{array}$$
(h)
$$\begin{array}{r} 10000101 \\ +\ 00010011 \\ \hline 10000111 \end{array}$$

53. (a)

$$\begin{array}{rl} 1000 & \\ +\,0110 & \\ \hline 1110 & \textit{invalid} \\ +\,0110 & \\ \hline 00010100 & \end{array}$$

(b)

$$\begin{array}{rl} 0111 & \\ +\,0101 & \\ \hline 1100 & \textit{invalid} \\ +\,0110 & \\ \hline 00010010 & \end{array}$$

(c)

$$\begin{array}{rl} 1001 & \\ +\,1000 & \\ \hline 10001 & \textit{invalid} \\ +\,0110 & \\ \hline 00010111 & \end{array}$$

(d)

$$\begin{array}{rl} 1001 & \\ +\,0111 & \\ \hline 10000 & \textit{invalid} \\ +\,0110 & \\ \hline 00010110 & \end{array}$$

(e)

$$\begin{array}{rl} 00100101 & \\ +\,00100111 & \\ \hline 01001100 & \textit{invalid} \\ +\,0110 & \\ \hline 01010010 & \end{array}$$

(f)

$$\begin{array}{rl} 01010001 & \\ +\,01011000 & \\ \hline 10101001 & \textit{invalid} \\ +\,0110 & \\ \hline 000100001001 & \end{array}$$

(g)

$$\begin{array}{rl} 10011000 & \\ +\,10010111 & \\ \hline 100101111 & \textit{invalid} \\ +\,01100110 & \\ \hline 000110010101 & \end{array}$$

(h)

$$\begin{array}{rl} 010101100001 & \\ +\,011100001000 & \\ \hline 110001101001 & \textit{invalid} \\ +\,0110 & \\ \hline 0001001001101001 & \end{array}$$

54. (a) 4 + 3

$$\begin{array}{r} 0100 \\ +\,0011 \\ \hline 0111 \end{array}$$

(b) 5 + 2

$$\begin{array}{r} 0101 \\ +\,0010 \\ \hline 0111 \end{array}$$

(c) 6 + 4

$$\begin{array}{r} 0110 \\ +\,0100 \\ \hline 1010 \\ +\,0110 \\ \hline 00010000 \end{array}$$

(d) 17 + 12

$$\begin{array}{r} 00010111 \\ +\,00100010 \\ \hline 00101001 \end{array}$$

(e) 28 + 23

$$\begin{array}{r} 00101000 \\ +\,00100011 \\ \hline 01001011 \\ +\,0110 \\ \hline 01010001 \end{array}$$

(f) 65 + 58

$$\begin{array}{r} 01100101 \\ +\,01011000 \\ \hline 10111101 \\ +\,01100110 \\ \hline 000100100011 \end{array}$$

(g) 113 + 101

$$\begin{array}{r} 000100010011 \\ +\,000100000001 \\ \hline 001000010100 \end{array}$$

(h) 295 + 157

$$\begin{array}{r} 001010010101 \\ +\,000101010111 \\ \hline 001111101100 \\ +\,01100110 \\ \hline 010001010010 \end{array}$$

Section 2-11 Digital Codes

55. The Gray code makes only one bit change at a time when going from one number in the sequence to the next number.
Gray for 1111_2 = 1000
Gray for 0000_2 = 0000

56. (a) 1 + 1 + 0 + 1 + 1 *Binary*
1 0 1 1 0 *Gray*

(b) 1 + 0 + 0 + 1 + 0 + 1 + 0 *Binary*
1 1 0 1 1 1 1 *Gray*

(c) 1 + 1 + 1 + 1 + 0 + 1 + 1 + 1 + 0 + 1 + 1 + 1 + 0 *Binary*
1 0 0 0 1 1 0 0 1 1 0 0 1 *Gray*

57. (a) 1 0 1 0 *Gray*
1 1 0 0 *Binary*

(b) 0 0 0 1 0 *Gray*
0 0 0 1 1 *Binary*

(c) 1 1 0 0 0 0 1 0 0 0 1 *Gray*
1 0 0 0 0 0 1 1 1 1 0 *Binary*

58. (a) 1 → 00110001
(b) 3 → 00110011
(c) 6 → 00110110
(d) 10 → 0011000100110000
(e) 18 → 0011000100111000
(f) 29 → 0011001000111001
(g) 56 → 0011010100110110
(h) 75 → 0011011100110101
(i) 107 → 001100010011000000110111

59. (a) 0011000 → CAN (b) 1001010 → **J**
(c) 0111101 → **=** (d) 0100011 → **#**
(e) 0111110 → > (f) 1000010 → **B**

60.

1001000	1100101	1101100	1101100	1101111	0101110	0100000
H	**e**	**l**	**l**	**o**	**.**	**#**
1001000	1101111	1110111	0100000	1100001	1110010	1100101
H	**o**	**w**	**#**	**a**	**r**	**e**
0100000	1111001	1101111	1110101	0111111		
#	**y**	**o**	**u**	**?**		

61.

1001000	1100101	1101100	1101100	1101111	0101110	0100000
48	**65**	**6C**	**6C**	**6F**	**2E**	**20**
1001000	1101111	1110111	0100000	1100001	1110010	1100101
48	**6F**	**77**	**20**	**61**	**72**	**65**
0100000	1111001	1101111	1110101	0111111		
20	**79**	**6F**	**75**	**3F**		

62. 30 INPUT A, B

3	0110011	33_{16}
0	0110000	30_{16}
SP	0100000	20_{16}
I	1001001	49_{16}
N	1001110	$4E_{16}$
P	1010000	50_{16}
U	1010101	55_{16}
T	1010100	54_{16}
SP	0100000	20_{16}
A	1000001	41_{16}
,	0101100	$2C_{16}$
B	1000010	42_{16}

Section 2-12 Error Detection Codes

63. Code (b) 011101010 has five 1s, so it is in error.

64. Codes (a) 11110110 and (c) 01010101010101010 are in error because they have an even number of 1s.

65. (a) 1 10100100 (b) 0 00001001 (c) 1 11111110

66. (a) $$\begin{array}{r} 1100 \\ +1011 \\ \hline 0111 \end{array}$$ (b) $$\begin{array}{r} 1111 \\ +0100 \\ \hline 1011 \end{array}$$ (c) $$\begin{array}{r} 100011100 \\ +\ \ 10011001 \\ \hline 110000101 \end{array}$$

67. (a) $$\begin{array}{r} 1100 \\ +0111 \\ \hline 1011 \end{array}$$ (b) $$\begin{array}{r} 1111 \\ +1011 \\ \hline 0100 \end{array}$$ (c) $$\begin{array}{r} 100011100 \\ +110000101 \\ \hline 010011001 \end{array}$$

In each case, you get the other number.

68.

```
101100100000
1010
   1001
   1010
     1100
     1010
      1100
      1010
       1100
       1010
        1100
        1010
Remainder = 0110
```

```
101100100110
1010
   1001
   1010
     1100
     1010
      1101
      1010
       1111
       1010
        1010
        1010
        0000
```

Append remainder to data. CRC is 101100100110.

69. Error in MSB of transmitted CRC:

```
001100100110
1010
1001
1010
  1100
  1010
   1101
   1010
    1110
    1010
     1000
     1010
       1011
       1010
          10
```

Remainder is 10, indicating an error.

CHAPTER 3
LOGIC GATES

Section 3-1 The Inverter

1. See Figure 3-1.

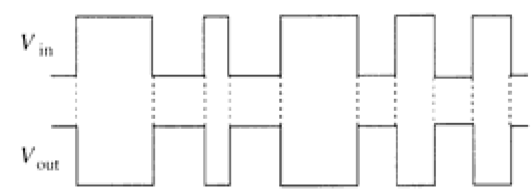


FIGURE 3-1

2. *B*: LOW, *C*: HIGH, *D*: LOW, *E*: HIGH, *F*: LOW

3. See Figure 3-2.

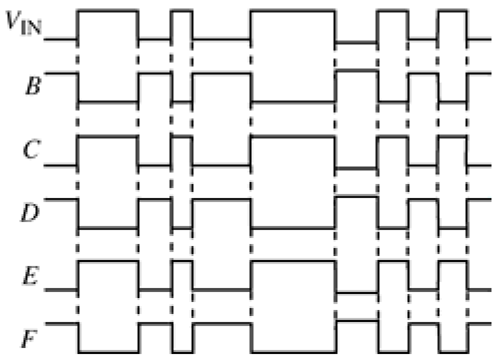


FIGURE 3-2

Section 3-2 The AND Gate

4. See Figure 3-3.

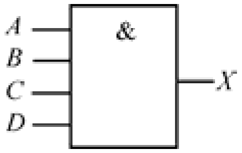


FIGURE 3-3

5. See Figure 3-4.

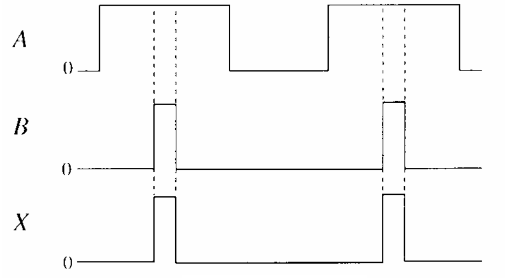


FIGURE 3-4

6. See Figure 3-5.

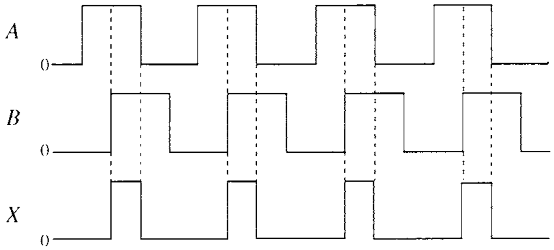


FIGURE 3-5

7. See Figure 3-6.

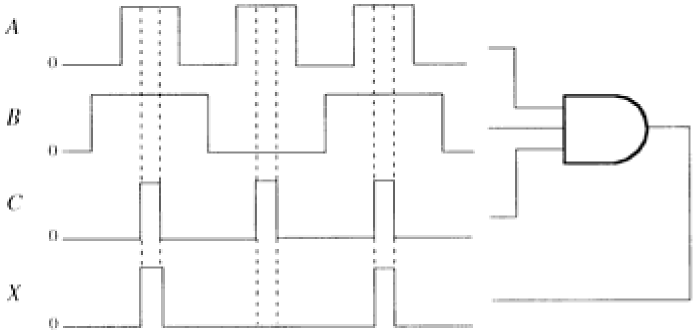


FIGURE 3-6

8. See Figure 3-7.

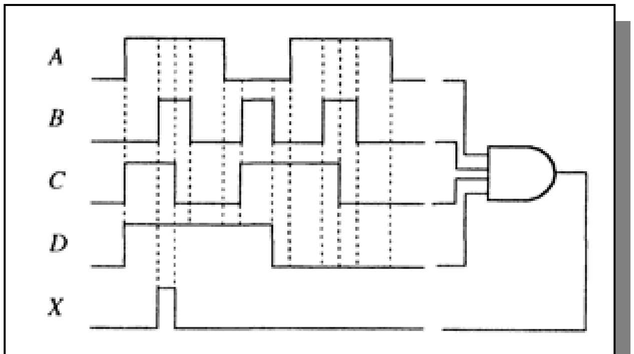


FIGURE 3-7

Section 3-3 The OR Gate

9. See Figure 3-8.

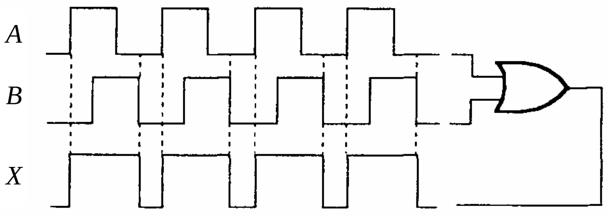


FIGURE 3-8

10. See Figure 3-9.

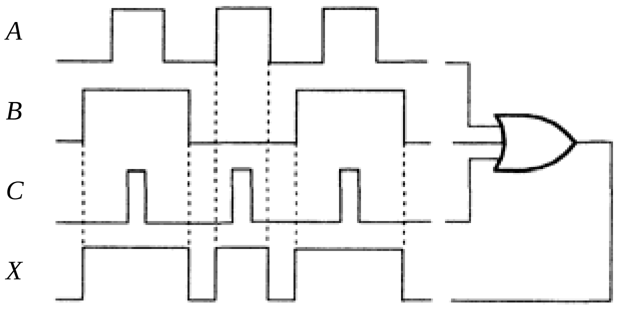


FIGURE 3-9

11. See Figure 3-10.

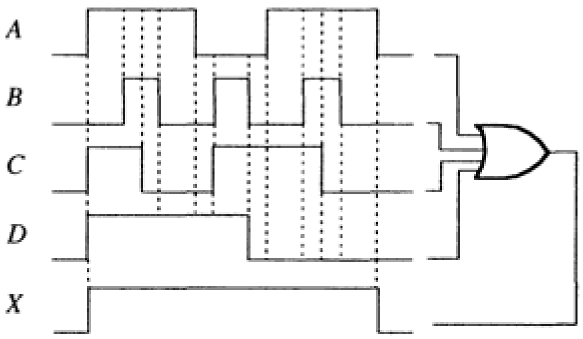


FIGURE 3-10

12. See Figure 3-11.

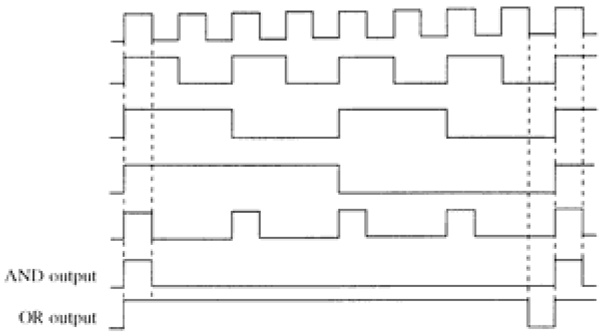


FIGURE 3-11

13. See Figure 3-12.

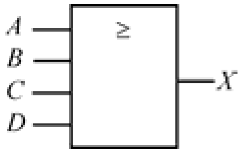


FIGURE 3-12

14.

A	*B*	*C*	*X*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Section 3-4 The NAND Gate

15. See Figure 3-13.

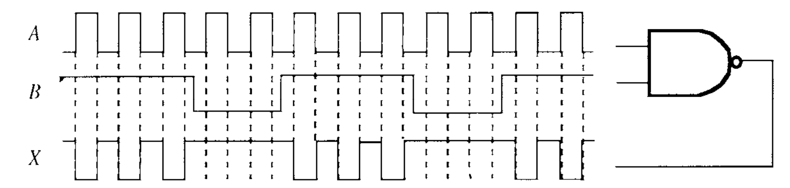


FIGURE 3-13

16. See Figure 3-14.

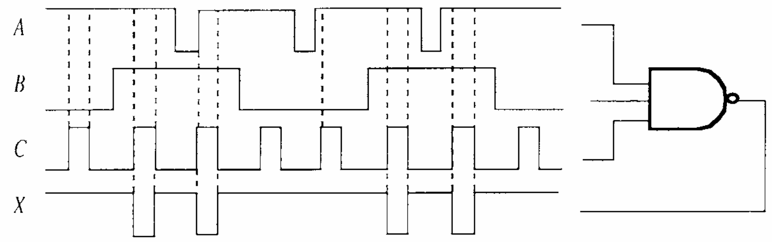


FIGURE 3-14

17. See Figure 3-15.

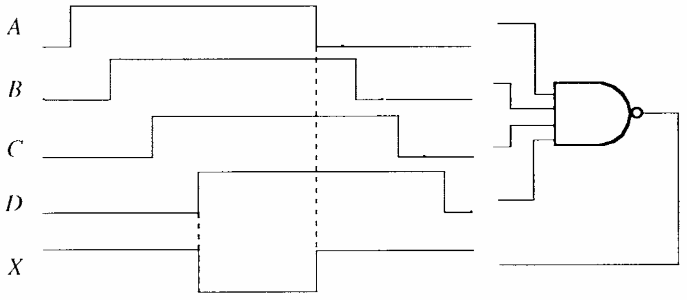
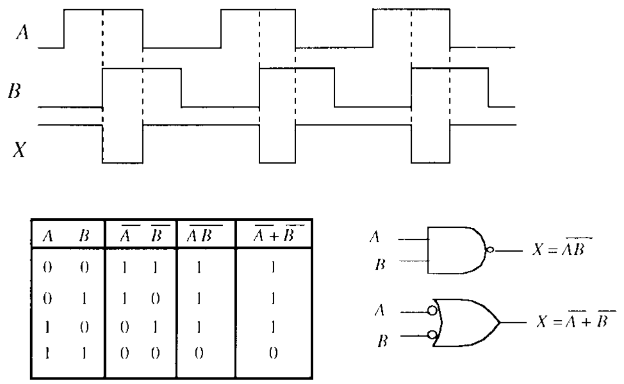


FIGURE 3-15

18. See Figure 3-16.



A	B	$\overline{A}$	$\overline{B}$	$\overline{AB}$	$\overline{A}+\overline{B}$
0	0	1	1	1	1
0	1	1	0	1	1
1	0	0	1	1	1
1	1	0	0	0	0

FIGURE 3-16

Section 3-5 The NOR Gate

19. See Figure 3-17.

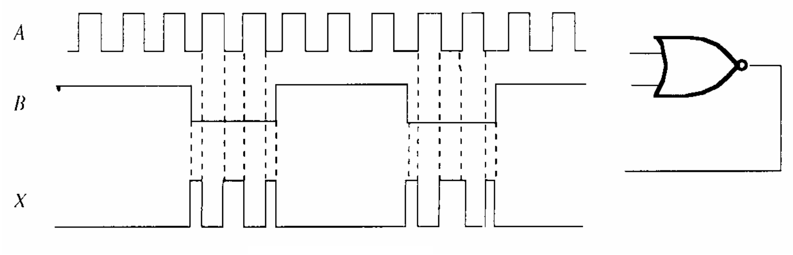


FIGURE 3-17

20. See Figure 3-18.

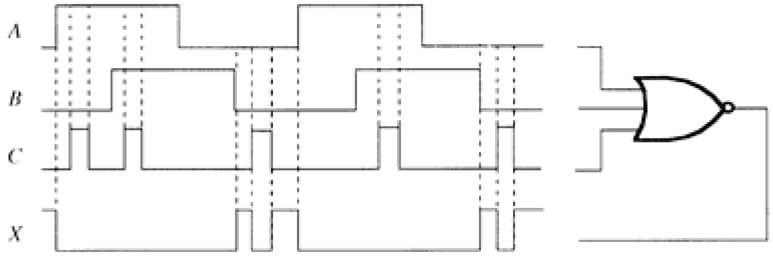


FIGURE 3-18

21. See Figure 3-19.

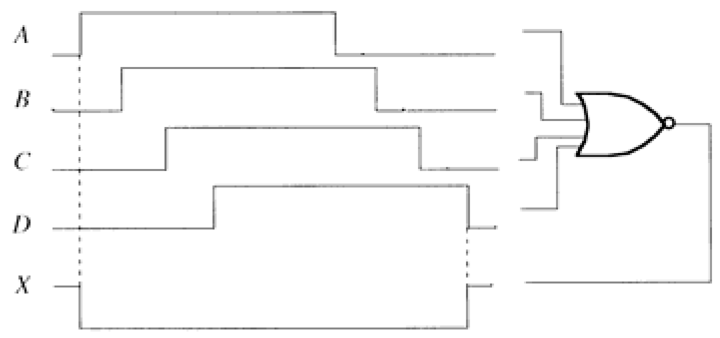
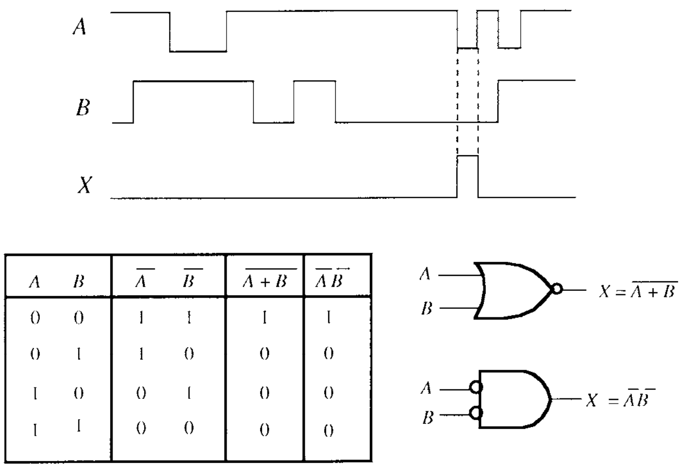


FIGURE 3-19

22. See Figure 3-20.



A	B	$\overline{A}$	$\overline{B}$	$\overline{A+B}$	$\overline{A}\,\overline{B}$
0	0	1	1	1	1
0	1	1	0	0	0
1	0	0	1	0	0
1	1	0	0	0	0

FIGURE 3-20

Section 3-6 The Exclusive-OR and Exclusive-NOR Gates

23. The output of the XOR gate is HIGH only when one input is HIGH. The output of the OR gate is HIGH any time one or more inputs are HIGH.

XOR = $A\overline{B} + \overline{A}B$
OR = $A + B$

24. See Figure 3-21.

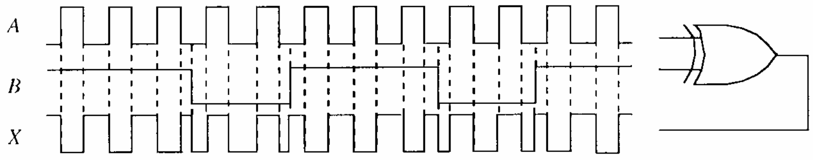


FIGURE 3-21

25. See Figure 3-22.

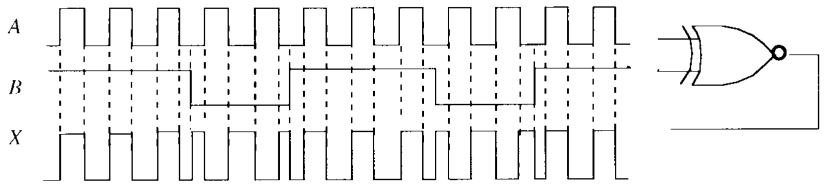


FIGURE 3-22

26. See Figure 3-23.

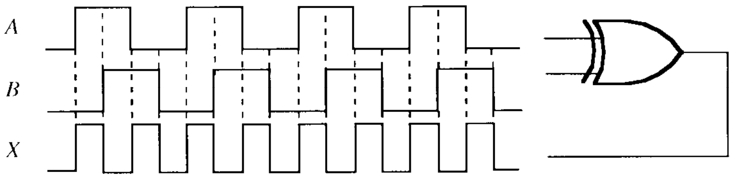


FIGURE 3-23

Section 3-7 Fixed-Function Logic

27. The power dissipation of **CMOS** increases with frequency.

28. (a) $P = \left(\frac{I_{\text{CCH}} + I_{\text{CCL}}}{2}\right)V_{\text{CC}} = \left(\frac{1.6\text{ mA} + 4.4\text{ mA}}{2}\right)5.5\text{ V} = 16.5\text{ mW}$

(b) $V_{\text{OH(min)}} = 2.7$ V

(c) $t_{PLH} = T_{PHL} = 15$ ns

(d) $V_{\text{OL}} = 0.4$ V (max)

(e) @ $V_{\text{CC}} = 2$ V, $t_{PHL} = t_{PLH} = 75$ ns; @ $V_{\text{CC}} = 6$ V, $t_{PHL} = t_{PLH} = 13$ ns

29. See Figure 3-24.

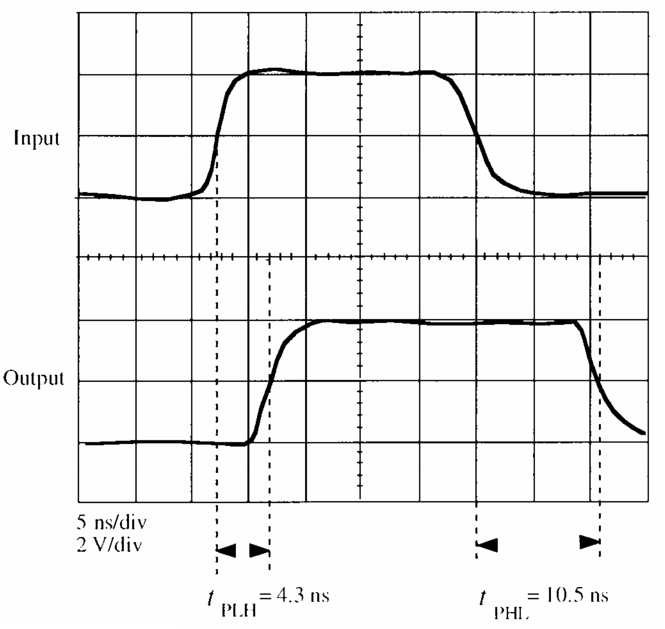


FIGURE 3-24

30. Gate A can be operated at the highest frequency because it has shorter propagation delay times than Gate B.

31. $P_D = V_{CC}I_C$ = (5 V)(4 mA) = **20 mW**

32. I_{CCH} = 4 mA; P_D = (5 V)(4 mA) = **20 mW**

Section 3-8 Troubleshooting

33. (a) NAND gate OK
(b) AND gate faulty
(c) NAND gate faulty
(d) NOR gate OK
(e) XOR gate faulty
(f) XOR gate OK

34. (a) NAND gate faulty. Input A open.
(b) NOR gate faulty. Input B shorted to ground.
(c) NAND gate OK
(d) XOR gate faulty. Input A open.

35. (a) The gate does not respond to pulses on either input when the other input is HIGH. It is unlikely that both inputs are open. The most probable fault is that the output is stuck in the LOW state (shorted to ground, perhaps) although it could be open.

(b) Pin 4 input or pin 6 output internally open.

36. The timer input to the AND gate is open. Check for 30-second HIGH level on this input when ignition is turned on.

37. An open seat-belt input to the AND gate will act like a constant HIGH just as if the seat belt were unbuckled.

38. Two possibilities: An input stuck LOW or the output stuck HIGH.

Section 3-9 Programmable Logic

39. $X_1 = \overline{A}B$
$X_2 = \overline{A}\,\overline{B}$
$X_3 = A\overline{B}$

40. $X_1 = \overline{A}BC$
Row 1: blow A, B, $\overline{B}$, C, and $\overline{C}$ column fuses
Row 2: blow A, $\overline{A}$, $\overline{B}$, C, and $\overline{C}$ column fuses
Row 3: blow A, $\overline{A}$, B, $\overline{B}$, and $\overline{C}$ column fuses
$X_2 = AB\overline{C}$
Row 4: blow $\overline{A}$, B, $\overline{B}$, C, and $\overline{C}$ column fuses
Row 5: blow A, $\overline{A}$, $\overline{B}$, C, and $\overline{C}$ column fuses
Row 6: blow A, $\overline{A}$, B, $\overline{B}$, and C column fuses
$X_3 = \overline{A}B\overline{C}$
Row 7: blow A, B, $\overline{B}$, C, and $\overline{C}$ column fuses
Row 8: blow A, $\overline{A}$, $\overline{B}$, C, and $\overline{C}$ column fuses
Row 9: blow A, $\overline{A}$, B, $\overline{B}$, and C column fuses

Special Design Problems

41. See Figure 3-25.

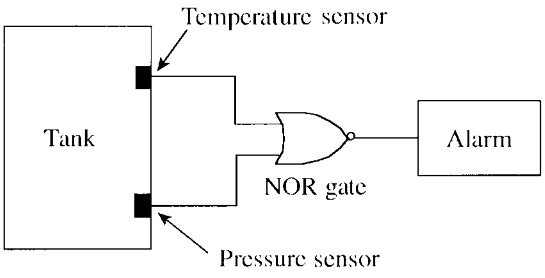


FIGURE 3-25

42. See Figure 3-26.

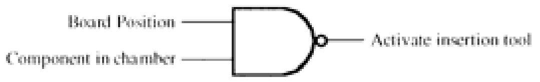


FIGURE 3-26

43. Add an inverter to the Enable input line of the AND gate as shown in Figure 3-27.

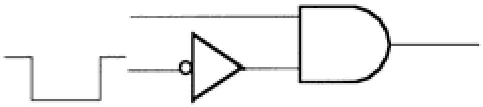


FIGURE 3-27

44. See Figure 3-28.

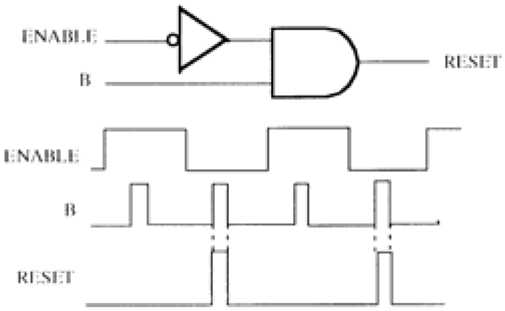


FIGURE 3-28

45. See Figure 3-29.

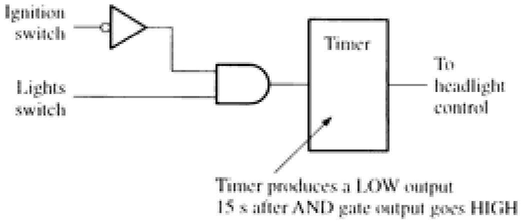

Ignition switch
Lights switch
Timer
To headlight control
Timer produces a LOW output 15 s after AND gate output goes HIGH

FIGURE 3-29

46. See Figure 3-30.

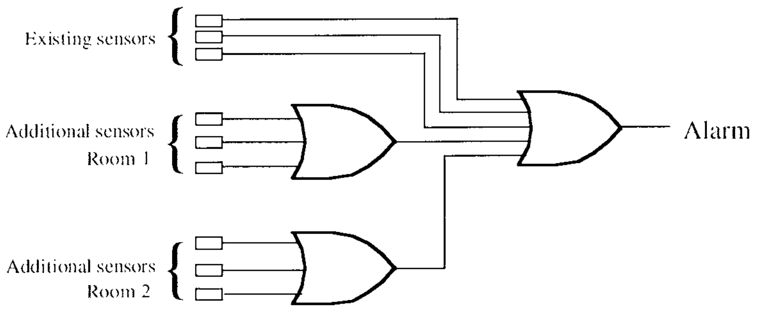


FIGURE 3-30

47. See Figure 3-31.

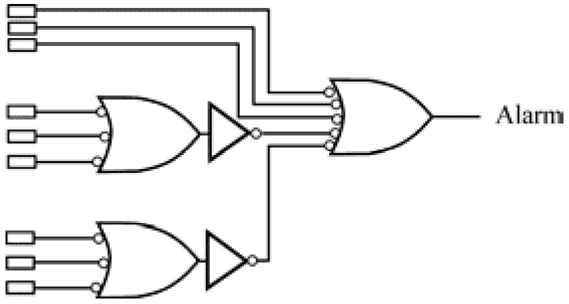


FIGURE 3-31

Multisim Troubleshooting Practice

48. Input A shorted to output.

49. Inputs shorted together.

50. No fault.

51. Output open.

CHAPTER 4
BOOLEAN ALGEBRA AND LOGIC SIMPLIFICATION

Section 4-1 Boolean Operations and Expressions

1. $X = A + B + C + D$
This is an OR configuration.

2. $Y = ABCDE$

3. $X = \overline{A} + \overline{B} + \overline{C}$

4.

(a)	$0 + 0 + 1 = 1$	(b)	$1 + 1 + 1 = 1$
(c)	$1 \cdot 0 \cdot 0 = 1$	(d)	$1 \cdot 1 \cdot 1 = 1$
(e)	$1 \cdot 0 \cdot 1 = 0$	(f)	$1 \cdot 1 + 0 \cdot 1 \cdot 1 = 1 + 0 = 1$

5.
(a) $AB = 1$ when $A = 1, B = 1$
(b) $A\overline{B}C = 1$ when $A = 1, B = 0, C = 1$
(c) $A + B = 0$ when $A = 0, B = 0$
(d) $\overline{A} + B + \overline{C} = 0$ when $A = 1, B = 0, C = 1$
(e) $\overline{A} + \overline{B} + C = 0$ when $A = 1, B = 1, C = 0$
(f) $\overline{A} + B = 0$ when $A = 1, B = 0$
(g) $A\overline{B}\,\overline{C} = 1$ when $A = 1, B = 0, C = 0$

6. (a) $X = (A + B)C + B$

A	B	C	$A + B$	$(A + B)C$	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	1	1

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

A	B	C	$\overline{A + B}$	X
0	0	0	1	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	0	0

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

A	B	C	$A\overline{B}C$	AB	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	0	1	1

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

A	B	$A + B$	$\overline{A} + B$	X
0	0	0	1	0
0	1	1	1	1
1	0	1	0	0
1	1	1	1	1

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

A	B	C	$A + BC$	$\overline{B} + \overline{C}$	X
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	0	0

Section 4-2 Laws and Rules of Boolean Algebra

7. (a) Commutative law of addition
(b) Commutative law of multiplication
(c) Distributive law

8. Refer to Table 4-1 in the textbook.

(a) Rule 9: $\overline{\overline{A}} = A$
(b) Rule 8: $A\overline{A} = 0$ (applied to 1st and 3rd terms)
(c) Rule 5: $A + A = A$
(d) Rule 6: $A + \overline{A} = 1$
(e) Rule 10: $A + AB = A$
(f) Rule 11: $A + \overline{A}B = A + B$ (applied to 1st and 3rd terms)

Section 4-3 DeMorgan's Theorems

9. (a) $\overline{A+\overline{B}} = \overline{A}\,\overline{\overline{B}} = \overline{\boldsymbol{A}}\boldsymbol{B}$

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

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

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

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

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

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

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

10. (a) $\overline{A\overline{B}(C+\overline{D})} = \overline{A\overline{B}}+\overline{(C+\overline{D})} = \overline{\boldsymbol{A}}+\boldsymbol{B}+\overline{\boldsymbol{C}}\boldsymbol{D}$

(b) $\overline{AB(CD+EF)} = \overline{AB}+\overline{(CD+EF)} = \overline{A}+\overline{B}+\overline{(CD)}\,\overline{(EF)}$

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

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

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

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

(e) $\overline{\overline{AB}(CD+\overline{E}F)(\overline{AB}+\overline{CD})} = \overline{\overline{AB}}+\overline{(CD+\overline{E}F)}+\overline{(\overline{AB}+\overline{CD})}$

$= AB+(\overline{CD})(\overline{\overline{E}F})+(\overline{\overline{AB}})(\overline{\overline{CD}})$

$= \boldsymbol{AB}+(\overline{\boldsymbol{C}}+\overline{\boldsymbol{D}})(\boldsymbol{E}+\overline{\boldsymbol{F}})+\boldsymbol{ABCD}$

11. (a) $\overline{\overline{(\overline{ABC})(\overline{EFG})}+\overline{(\overline{HIJ})(\overline{KLM})}} = \overline{\overline{\overline{ABC}}+\overline{\overline{EFG}}+\overline{\overline{HIJ}}+\overline{\overline{KLM}}}$

$= \overline{ABC+EFG+HIJ+KLM} = (\overline{ABC})(\overline{EFG})(\overline{HIJ})(\overline{KLM})$

$= (\overline{\boldsymbol{A}}+\overline{\boldsymbol{B}}+\overline{\boldsymbol{C}})(\overline{\boldsymbol{E}}+\overline{\boldsymbol{F}}+\overline{\boldsymbol{G}})(\overline{\boldsymbol{H}}+\overline{\boldsymbol{I}}+\overline{\boldsymbol{J}})(\overline{\boldsymbol{K}}+\overline{\boldsymbol{L}}+\overline{\boldsymbol{M}})$

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

$= \overline{A}B\overline{C}(\overline{C}+\overline{D})+BC = \overline{A}B\overline{C}+\overline{A}B\overline{C}\,\overline{D}+BC = \overline{A}B\overline{C}(1+\overline{D})+BC$

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

(c) $\overline{\overline{\overline{(A+B)}\,\overline{(C+D)}\,\overline{(E+F)}\,\overline{(G+H)}}}$

$= \overline{(A+B)}\,\overline{(C+D)}\,\overline{(E+F)}\,\overline{(G+H)} = \overline{\boldsymbol{A}}\,\overline{\boldsymbol{B}}\,\overline{\boldsymbol{C}}\,\overline{\boldsymbol{D}}\,\overline{\boldsymbol{E}}\,\overline{\boldsymbol{F}}\,\overline{\boldsymbol{G}}\,\overline{\boldsymbol{H}}$

Section 4-4 Boolean Analysis of Logic Circuits

12. (a) $AB = X$

(b) $\overline{A} = X$

(c) $A + B = X$

(d) $A + B + C = X$

13. See Figure 4-1.

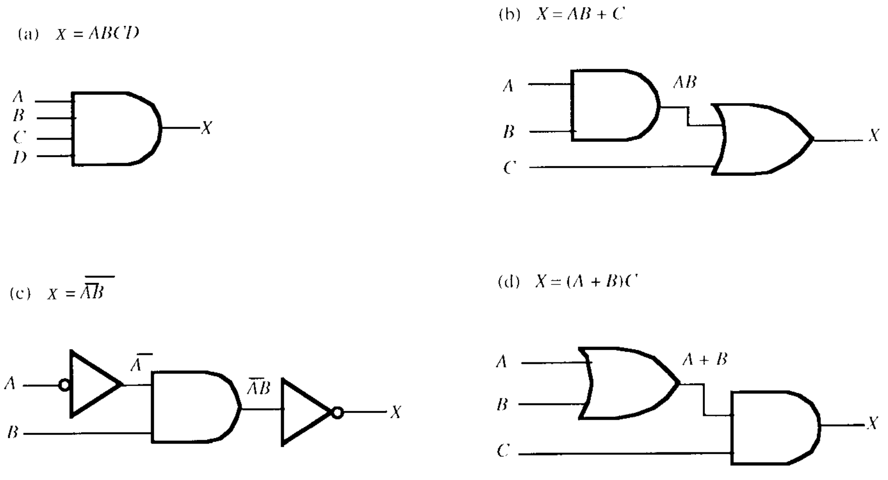


FIGURE 4-1

14. See Figure 4-2.

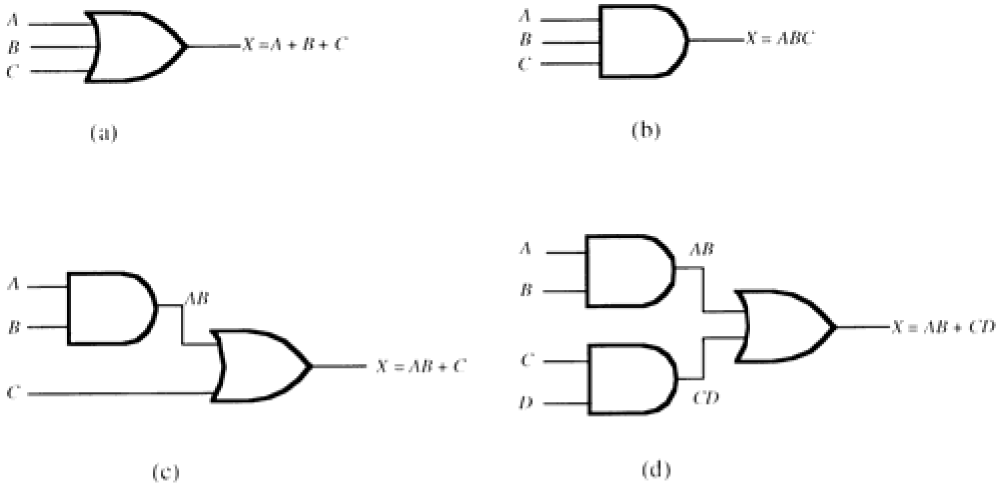
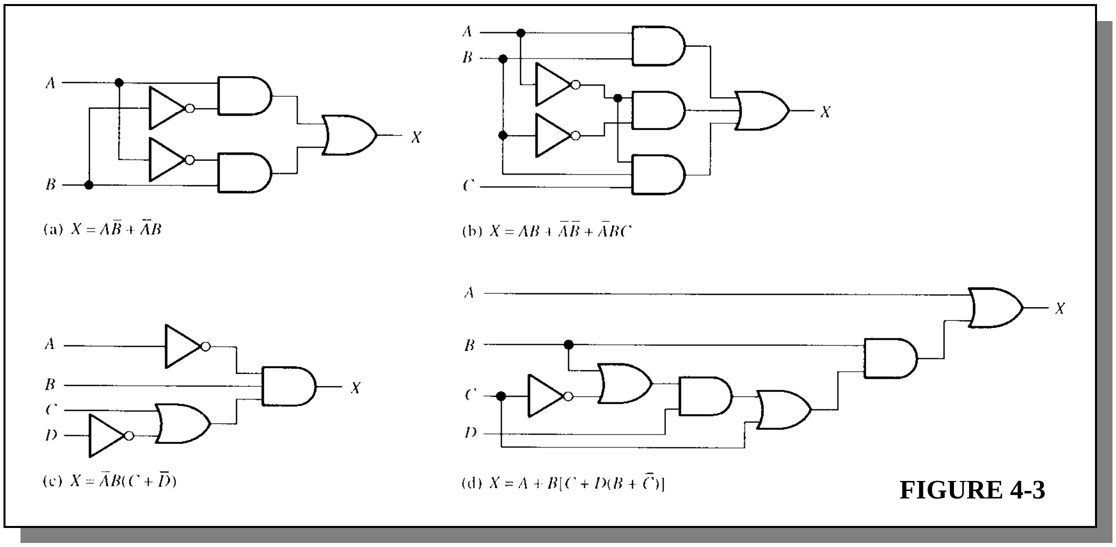


FIGURE 4-2

15. See Figure 4-3.



A
B
X
(a) $X = A\overline{B} + \overline{A}B$
A
B
C
X
(b) $X = AB + \overline{A}\overline{B} + \overline{A}BC$
A
B
C
D
X
(c) $X = \overline{A}B(C + \overline{D})$
A
B
C
D
X
(d) $X = A + B[C + D(B + \overline{C})]$

FIGURE 4-3

16. (a) See Figure 4-4(a).
(b) See figure 4-4(b).

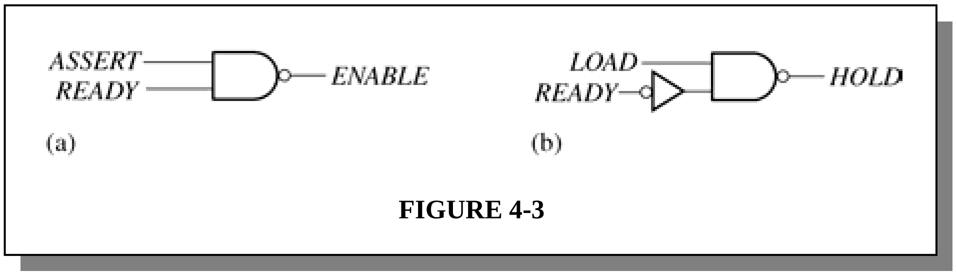


FIGURE 4-3

17. See Tables 4-1 and 4-2.

Table 4-1

INPUTS			OUTPUT
$\overline{VCR}$	$\overline{CAMI}$	$\overline{RDY}$	$\overline{RECORD}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 4-2

INPUTS			OUTPUT
RTS	*ENABLE*	$\overline{BUSY}$	$\overline{SEND}$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

18. (a) $X = A + B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(b) $X = AB$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(c) $X = AB + BC$

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

(d) $X = (A + B)C$

A	B	C	X
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	0
1	1	1	1

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

A	B	C	$A + B$	$\overline{B} + C$	X
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	0	0
0	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

Section 4-5 Simplification Using Boolean Algebra

19. (a) $A(A+ B) = AA + BB = A + AB = A(1 + B) = \boldsymbol{A}$

(b) $A(\overline{A} + AB) = A\overline{A} + AAB = 0 + AB = \boldsymbol{AB}$

(c) $BC + \overline{B}C = C(B + \overline{B}) = C(1) = \boldsymbol{C}$

(d) $A(A + \overline{A}B) = AA + A\overline{A}B = A + (0)B = A + 0 = \boldsymbol{A}$

(e) $A\overline{B}C + \overline{A}BC + \overline{A}\overline{B}C = A\overline{B}C + \overline{A}C(B + \overline{B}) = A\overline{B}C + \overline{A}C(1)$

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

20. (a) $(A+\bar{B})(A+C) = AA + AC + A\bar{B} + \bar{B}C = A + AC + A\bar{B} + \bar{B}C$

$= A(1+C+\bar{B}) + \bar{B}C = A(1) + \bar{B}C = \boldsymbol{A + \bar{B}C}$

(b) $\bar{A}B + \bar{A}B\bar{C} + \bar{A}BCD + \bar{A}B\bar{C}\bar{D}E = \bar{A}B(1+\bar{C}+CD+\bar{C}\bar{D}E) = \bar{A}B(1)$

$= \boldsymbol{\bar{A}B}$

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

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

$= \boldsymbol{A + C}$

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

$= AB + AB\bar{C} + 0 + 0 = AB(1+\bar{C}) = \boldsymbol{AB}$

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

$= AB + \overline{AB}C = \boldsymbol{AB} + \boldsymbol{C}$

21. (a) $BD + B(D+E) + \bar{D}(D+F) = BD + BD + BE + \bar{D}D + \bar{D}F$

$= BD + BE + 0 + \bar{D}F = \boldsymbol{BD} + \boldsymbol{BE} + \boldsymbol{\bar{D}F}$

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

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

(c) $(B+BC)(B+\bar{B}C)(B+D) = B(1+C)(B+C)(B+D)$

$= B(B+C)(B+D) = (BB+BC)(B+D) = (B+BC)(B+D)$

$= B(1 + C)(B + D) = B(B + D) = BB + BD = B + BD = B(1 + D) = \boldsymbol{B}$

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

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

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

$= CD(1+\bar{A}) + AB\bar{C} + AB\bar{D} = CD + AB\bar{C} + AB\bar{D} = CD + AB(\overline{CD}) = \boldsymbol{CD} + \boldsymbol{AB}$

(e) $ABC[AB + \bar{C}(BC+AC)] = ABABC + ABC\bar{C}(BC+AC)$

$= ABC + 0(BC + AC) = \boldsymbol{ABC}$

22. First develop the Boolean expression for the output of each gate network and simplify.

(a) See Figure 4-5.

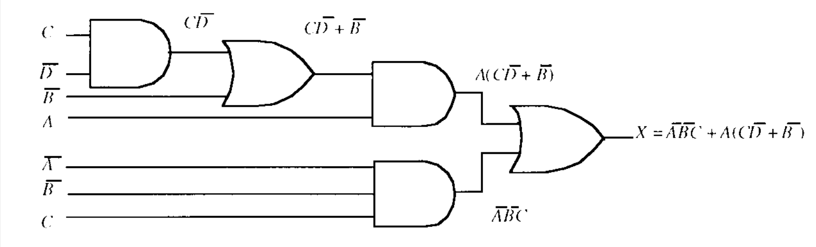


FIGURE 4-5

$$X = \overline{A}\overline{B}C + A(C\overline{D}+\overline{B}) = \overline{A}\overline{B}C + AC\overline{D} + A\overline{B} = \overline{B}(A+\overline{A}C) + AC\overline{D}$$
$$= \overline{B}(A+C) + ACD = \boldsymbol{A\overline{B}} + \boldsymbol{\overline{B}C} + \boldsymbol{AC\overline{D}}$$

(b) See Figure 4-6.

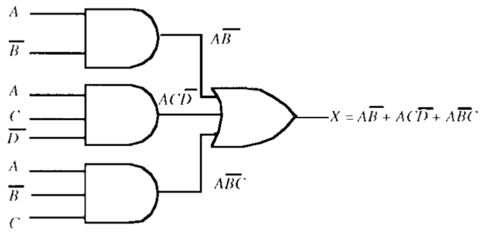


FIGURE 4-6

$$X = A\overline{B} + AC\overline{D} + A\overline{B}C = A\overline{B}(1+C) + AC\overline{D} = \boldsymbol{A\overline{B}} + \boldsymbol{AC\overline{D}}$$

(c) See Figure 4-7.

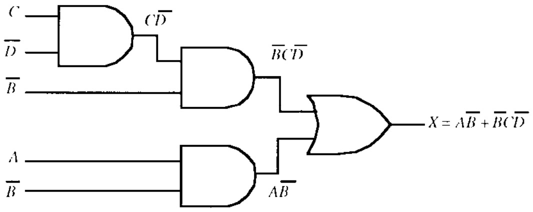


FIGURE 4-7

$X = \boldsymbol{A\overline{B}} + \boldsymbol{\overline{B}C\overline{D}}$ No further simplification is possible.

(d) See Figure 4-8.

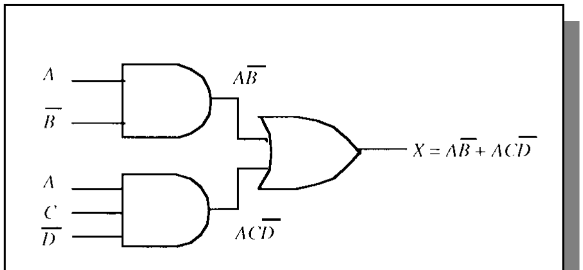


FIGURE 4-8

$X = \boldsymbol{A\overline{B} + AC\overline{D}}$ No further simplification is possible.

Section 4-6 Standard Forms of Boolean Expressions

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

(b) $(A+\overline{B}C)C = AC + \overline{B}CC = \boldsymbol{AC + \overline{B}C}$

(c) $(A + C)(AB + AC) = AAB + AAC + ABC + ACC = AB + AC + ABC + ACC$
$= (AB + AC)(1 + C) = \boldsymbol{AB + AC}$

24. (a) $AB + CD(A\overline{B} + CD) = AB + A\overline{B}CD + CDCD = AB + A\overline{B}CD + CD$

$= AB(A\overline{B} + 1)CD = \boldsymbol{AB + CD}$

(b) $AB(\overline{B}\overline{C} + BD) = AB\overline{B}\overline{C} + ABBD = 0 + ABD = \boldsymbol{ABD}$

(c) $A + B[AC + (B + \overline{C})D] = A + ABC + (B + \overline{C})BD$

$= A + ABC + BD + B\overline{C}D = A(1 + BC) + BD + B\overline{C}D = A + BD(1 + \overline{C})$

$= \boldsymbol{A + BD}$

25. (a) The domain is *A*, *B*, *C*
The standard SOP is: $A\overline{B}C + A\overline{B}\overline{C} + ABC + \overline{A}BC$

(b) The domain is *A*, *B*, *C*
The standard SOP is: $ABC + A\overline{B}C + \overline{A}\overline{B}C$

(c) The domain is *A*, *B*, *C*
The standard SOP is: $ABC + AB\overline{C} + A\overline{B}C$

26. (a) $AB + CD = ABCD + ABC\overline{D} + AB\overline{C}D + AB\overline{C}\overline{D} + \overline{A}\overline{B}CD + \overline{A}BCD + A\overline{B}CD$

(b) $ABD = ABCD + AB\overline{C}D$

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

27. (a) $A\overline{B}C + A\overline{B}\,\overline{C} + ABC + \overline{A}BC$: 101 + 100 + 111 + 011

(b) $ABC + A\overline{B}C + \overline{A}\,\overline{B}C$: 111 + 101 + 001

(c) $ABC + AB\overline{C} + A\overline{B}C$: 111 + 110 + 101

28. (a) $ABCD + ABC\overline{D} + AB\overline{C}D + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}CD + \overline{A}BCD + A\overline{B}CD$:

1111 + 1110 + 1101 + 1100 + 0011 + 0111 + 1011

(b) $ABCD + AB\overline{C}D$: 1111 + 1101

(c) $A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D + A\overline{B}C\overline{D} + A\overline{B}CD + AB\overline{C}\,\overline{D} + AB\overline{C}D$

$+\ ABC\overline{D} + ABCD + \overline{A}B\overline{C}D + \overline{A}BCD$:

1000 + 1001 + 1010 + 1011 + 1100 + 1101 + 1110 + 1111 + 0101 + 0111

29. (a) $(A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(\overline{A}+\overline{B}+C)$

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

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

30. (a) $(A+B+C+D)(A+B+C+\overline{D})(A+B+\overline{C}+D)(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})$

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

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

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

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

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

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

Section 4-7 Boolean Expressions and Truth Tables

31. (a) Table 4-3

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

(b) Table 4-4

X	Y	Z	Q
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

32. (a) Table 4-5

A	*B*	*C*	*D*	*X*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

(b) Table 4-6

W	*X*	*Y*	*Z*	*Q*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

33. (a) $\overline{A}B + AB\overline{C} + \overline{A}\,\overline{C} + A\overline{B}C = \overline{A}BC + \overline{A}B\overline{C} + AB\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}C$

(b) $\overline{X} + Y\overline{Z} + WZ + X\overline{Y}Z = \overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z} + \overline{W}\,\overline{X}\,\overline{Y}Z + \overline{W}\,\overline{X}Y\overline{Z} + \overline{W}\,\overline{X}YZ$
$+ \overline{W}X\overline{Y}Z + \overline{W}XY\overline{Z} + W\overline{X}\,\overline{Y}\,\overline{Z} + W\overline{X}\,\overline{Y}Z$
$+ W\overline{X}Y\overline{Z} + W\overline{X}YZ + WX\overline{Y}Z + WXY\overline{Z} + WXYZ$

Table 4-7

A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Table 4-8

W	*X*	*Y*	*Z*	*Q*
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

34. (a) Table 4-9

A	*B*	*C*	*X*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

(b) Table 4-10

A	*B*	*C*	*D*	*X*
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

35. (a) Table 4-11

A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(b) Table 4-12

A	*B*	*C*	*D*	*X*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

36. (a) $X = \overline{A}\,\overline{B}C + A\overline{B}\,\overline{C} + A\overline{B}C + ABC$

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

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

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

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

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

(d) $X = \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}CD + ABC\bar{D} + ABCD$

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

Section 4-8 The Karnaugh Map

37. See Figure 4-9.

38. See Figure 4-10.

39. See Figure 4-11.

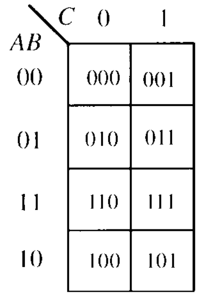


FIGURE 4-9

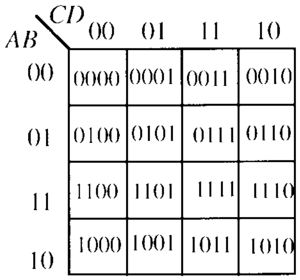


FIGURE 4-10

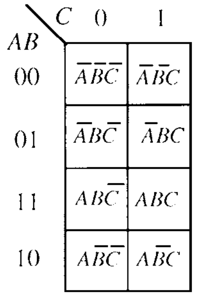


FIGURE 4-11

Section 4-9 Karnaugh Map SOP Minimization

40. See Figure 4-12.

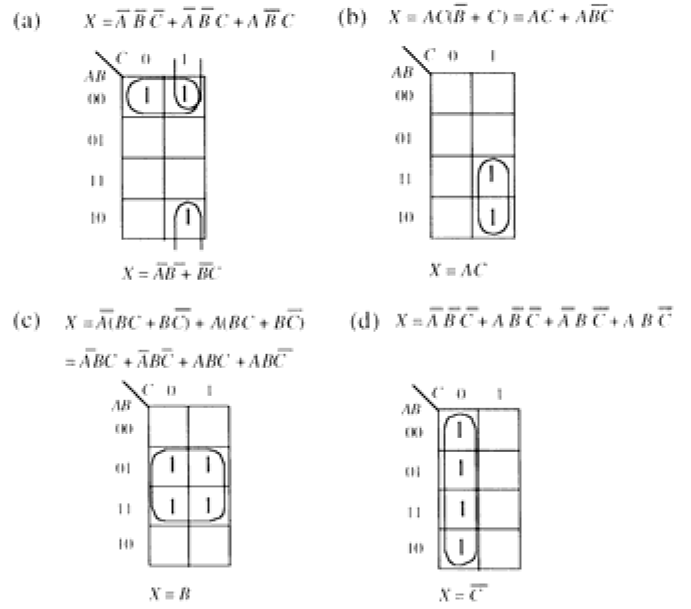

(a)
$X = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}C$
C
AB
0
1
00
01
11
10
1
1
1
$X = \bar{A}\bar{B} + \bar{B}C$
(b)
$X = AC(\bar{B} + C) = AC + A\bar{B}C$
C
AB
0
1
00
01
11
10
1
1
$X = AC$
(c)
$X = \bar{A}(BC + B\bar{C}) + A(BC + B\bar{C})$
$= \bar{A}BC + \bar{A}B\bar{C} + ABC + AB\bar{C}$
C
AB
0
1
00
01
11
10
1
1
1
1
$X = B$
(d)
$X = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + \bar{A}B\bar{C} + AB\bar{C}$
C
AB
0
1
00
01
11
10
1
1
1
1
$X = \bar{C}$

FIGURE 4-12

41. See Figure 4-13.

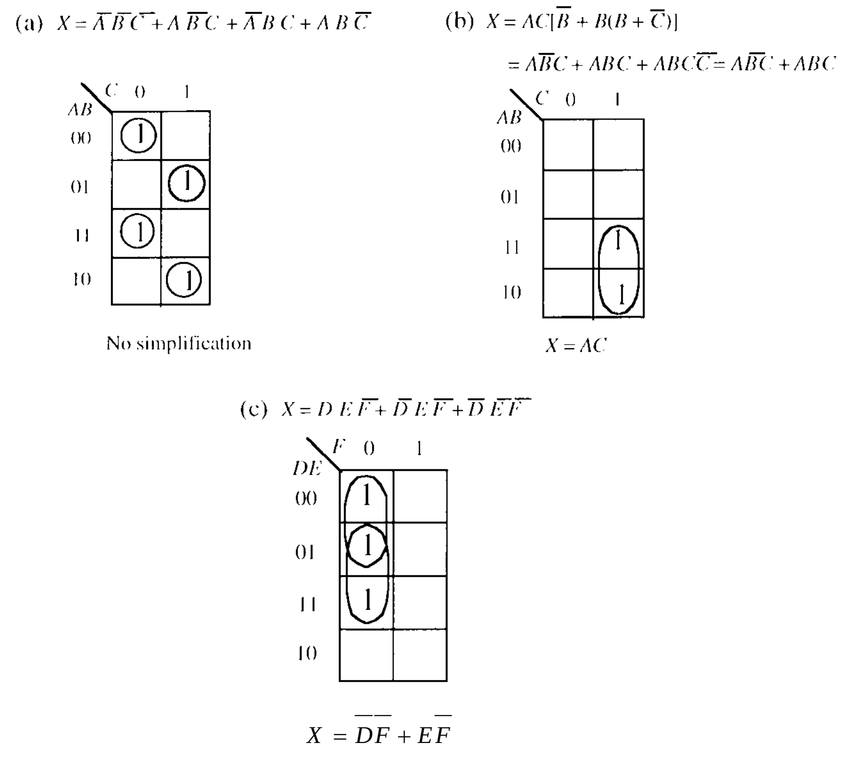

(a) X = ĀB̄C̄ + AB̄C + ĀBC + ABC̄
AB
C
0
1
00
01
11
10
1
1
1
1
No simplification
(b) X = AC[B̄ + B(B + C̄)]
= AB̄C + ABC + ABCC̄ = AB̄C + ABC
AB
C
0
1
00
01
11
10
1
1
X = AC
(c) X = DĒF̄ + D̄ĒF̄ + D̄EF̄
DE
F
0
1
00
01
11
10
1
1
1
X = D̄F̄ + ĒF̄

FIGURE 4-13

42. (a) $AB + A\overline{B}C + ABC = AB(C + \overline{C}) + A\overline{B}C + ABC$

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

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

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

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

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

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

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

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

(d) $A\overline{B} + A\overline{B}\,\overline{C}D + CD + B\overline{C}D + ABCD$

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

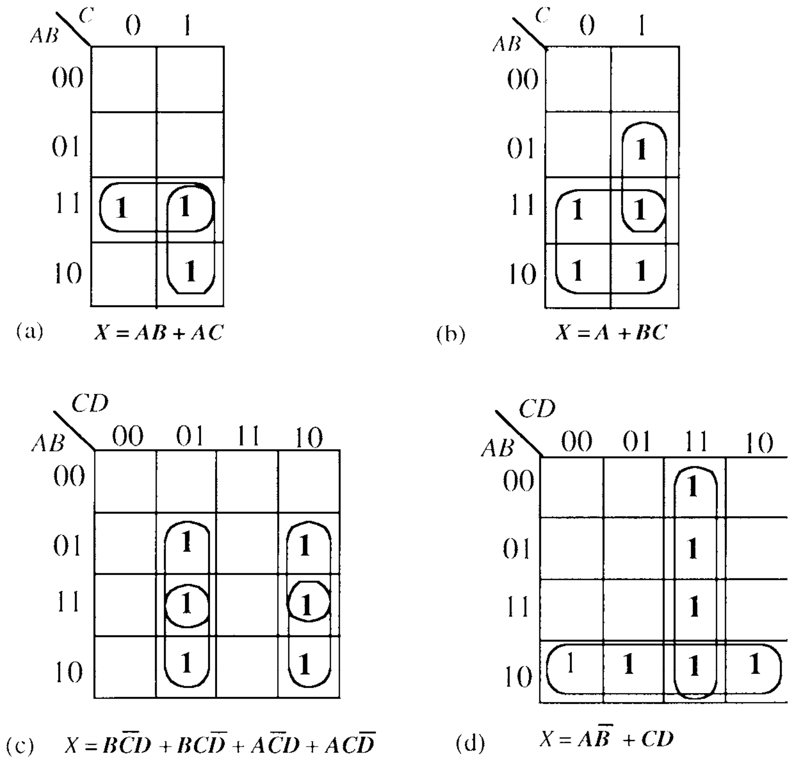
$= A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D + ABC\overline{D} + ABCD + A\overline{B}\,\overline{C}D + ABCD + A\overline{B}CD + \overline{A}BCD$

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

$= A\overline{B}\,\overline{C}\,\overline{D} + A\overline{B}\,\overline{C}D + ABC\overline{D} + ABCD + A\overline{B}CD + \overline{A}BCD + \overline{A}\,\overline{B}CD + AB\overline{C}D + \overline{A}B\overline{C}D$

$= \overline{A}\,\overline{B}CD + \overline{A}B\overline{C}D + \overline{A}BCD + A\overline{B}C\overline{D} + A\overline{B}\,\overline{C}D + A\overline{B}CD + AB\overline{C}D + ABCD + ABC\overline{D}$

43. See Figure 4-14.

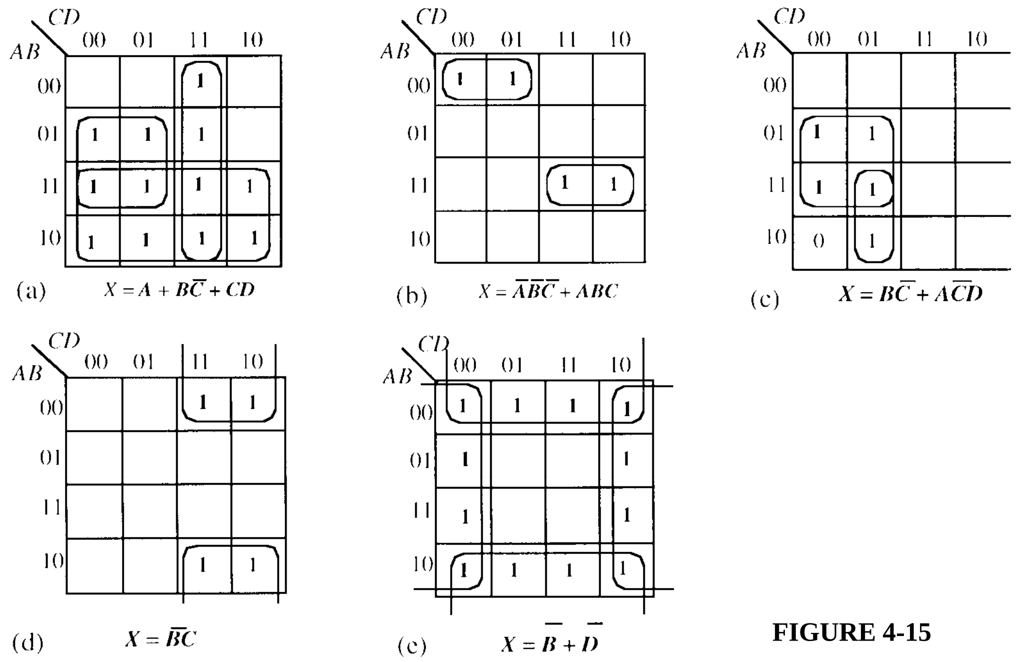


(a) $X = AB + AC$ (b) $X = A + BC$

(c) $X = B\overline{C}D + BC\overline{D} + A\overline{C}D + AC\overline{D}$ (d) $X = A\overline{B} + CD$

FIGURE 4-14

44. See Figure 4-15.

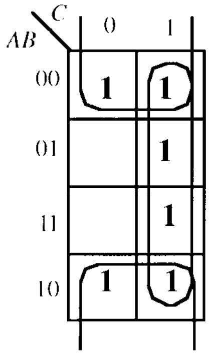


(a) $X = A + B\overline{C} + CD$ (b) $X = \overline{A}\overline{B}\overline{C} + ABC$ (c) $X = B\overline{C} + A\overline{C}D$

(d) $X = \overline{B}C$ (e) $X = \overline{B} + \overline{D}$

FIGURE 4-15

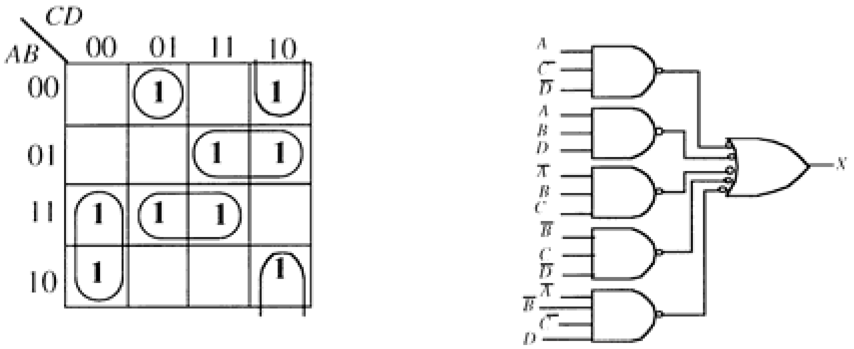
45. Plot the 1's from Table 4-11 in the text on the map as shown in Figure 4-16 and simplify.



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

FIGURE 4-16

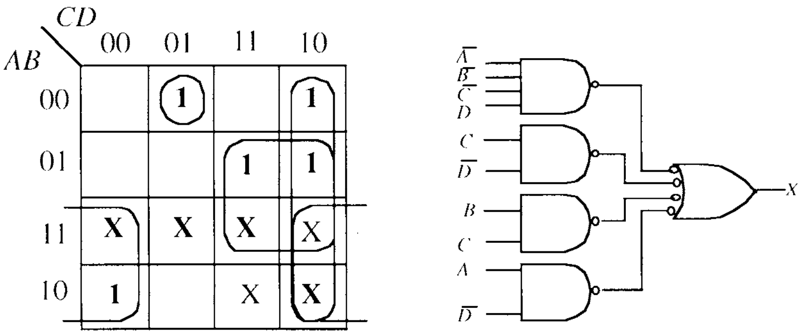
46. Plot the 1's from Table 4-12 in the text on the map as shown in Figure 4-17 and simplify.



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

FIGURE 4-17

47. See Figure 4-18.



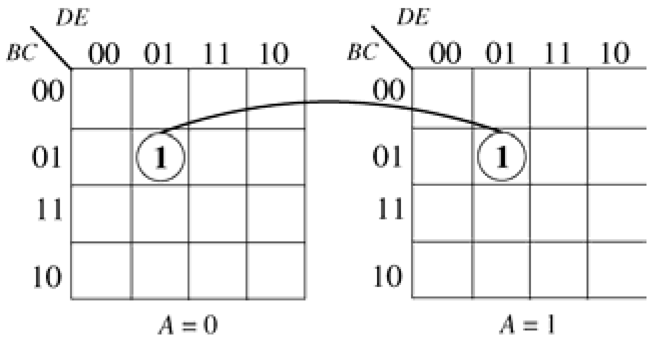
$$X = \overline{A}\overline{B}\overline{C}D + C\overline{D} + BC + A\overline{D}$$

FIGURE 4-18

Section 4-10 Five-Variable Karnaugh Maps

48. $X = A\bar{B}C\bar{D}E + \bar{A}\,\bar{B}\,C\bar{D}E$.

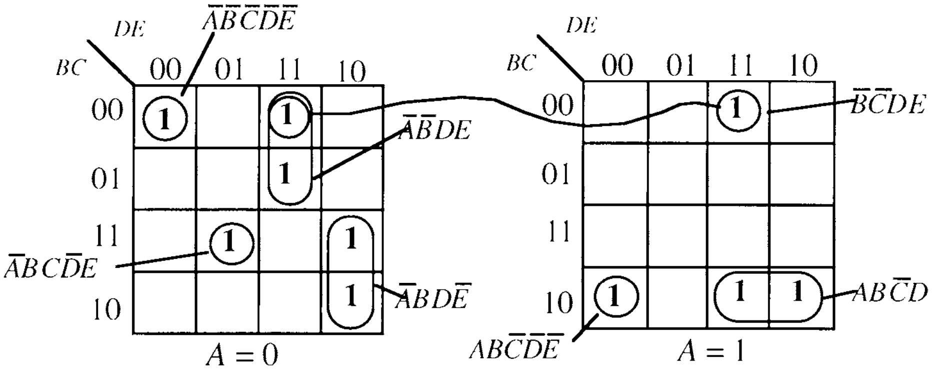
See Figure 4-19.



$X = \bar{B}C\bar{D}E$

FIGURE 4-19

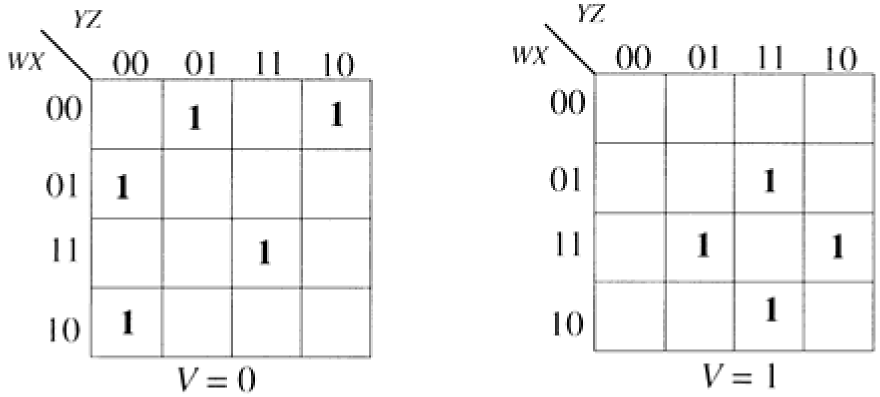
49. See Figure 4-20.



$$X = \bar{A}\bar{B}\bar{C}\bar{D}\bar{E} + \bar{A}BC\bar{D}E + AB\bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}DE + \bar{A}BD\bar{E} + \bar{B}\bar{C}DE + AB\bar{C}D$$

FIGURE 4-20

50. See Figure 4-21.



WX \ YZ	00	01	11	10
00		1		1
01	1			
11			1	
10	1			

V = 0

WX \ YZ	00	01	11	10
00				
01			1	
11		1		1
10			1	

V = 1

No simplification is possible

FIGURE 4-21

Section 4-11 Describing Logic with an HDL

51.

```
entity AND_OR is
        port (A, B, C, D, E, F, G, H, I: in bit; X: out bit);
end entity AND_OR;
architecture Logic of AND_OR is
begin
        X <= (A and B and C) or (D and E and F) or (G and H and I);
end architecture Logic;
```

52. The VHDL program:

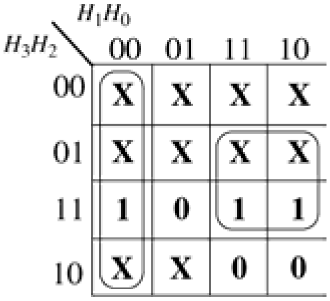
```
entity SOP is
        port (A, B, C: in bit; X: out bit);
end entity SOP;
architecture Logic of SOP is
begin
        Y <= (A and not B and C) or (not A and not B and C) or
                (A and not B and not C) or (not A and B and C);
end architecture Logic;
```

System Application Activity

53. An LED display is more suitable for low-light conditions because LEDs emit light and LCDs do not.

54. The purpose of the invalid code detector is to detect the codes 1010, 1011, 1100, 1101, 1110, and 1111 to activate the display for letters.

55. The standard SOP expression for segment *c* is:
$c = H_3\overline{H}_2H_1\overline{H}_0 + H_3\overline{H}_2H_1H_0 + H_3H_2\overline{H}_1H_0$
This expression is minimized in Figure 4-22.



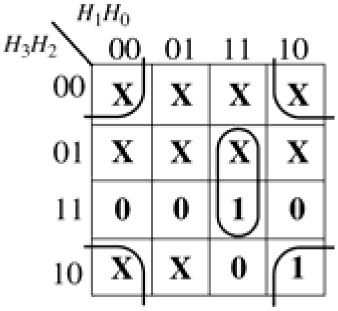
H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	1	0	1	1
10	X	X	0	0

$c = \overline{H}_1\overline{H}_0 + H_2H_1$

The standard expression requires three 4-input AND gates, one 3-input OR gate, and 3 inverters. The minimum expression requires two 2-input AND gates, one 2 input OR gate, and 2 inverters.

FIGURE 4-22

56. The standard SOP expression for segment *d* is:
$d = H_3\overline{H}_2H_1H_0 + H_3H_2\overline{H}_1\overline{H}_0 + H_3H_2\overline{H}_1H_0 + H_3H_2H_1\overline{H}_0$
This expression is minimized in Figure 4-23.



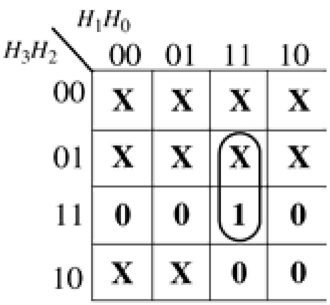
H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	1	0
10	X	X	0	1

$d = H_2H_1H_0 + \overline{H}_2\overline{H}_0$

The standard expression requires four 4-input AND gates, one 4-input OR gate, and 3 inverters. The minimum expression requires one 2-input AND gates, one 3-input AND gate, one 2-input OR gate, and 2 inverters.

FIGURE 4-23

The standard SOP expression for segment *e* is:
$e = H_3\overline{H}_2H_1\overline{H}_0 + H_3\overline{H}_2H_1H_0 + H_3H_2\overline{H}_1\overline{H}_0 + H_3H_2\overline{H}_1H_0 + H_3H_2H_1\overline{H}_0$
This expression is minimized in Figure 4-24.



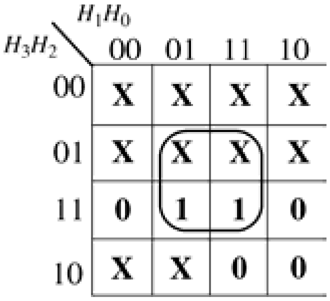
H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	1	0
10	X	X	0	0

$e = H_2H_1H_0$

The standard expression requires five 4-input AND gates, one 5-input OR gate, and 3 inverters.
The minimum expression requires one 3-input AND gate.

FIGURE 4-24

The standard SOP expression for segment *f* is:
$f = H_3\overline{H}_2H_1\overline{H}_0 + H_3\overline{H}_2H_1H_0 + H_3H_2\overline{H}_1\overline{H}_0 + H_3H_2H_1\overline{H}_0$
This expression is minimized in Figure 4-25.



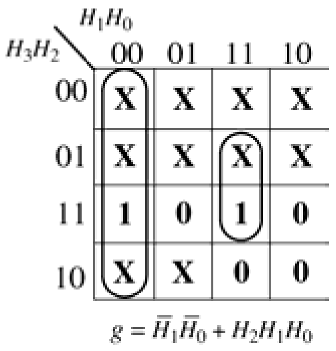
H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	1	1	0
10	X	X	0	0

$f = H_2H_0$

The standard expression requires four 4-input AND gates, one 4-input OR gate, and 3 inverters.
The minimum expression requires one 2-input AND gate.

FIGURE 4-25

The standard SOP expression for segment *g* is:
$g = H_3\overline{H}_2H_1\overline{H}_0 + H_3\overline{H}_2H_1H_0 + H_3H_2\overline{H}_1H_0 + H_3H_2H_1\overline{H}_0$
This expression is minimized in Figure 4-26.



H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	1	0	1	0
10	X	X	0	0

$g = \overline{H}_1\overline{H}_0 + H_2H_1H_0$

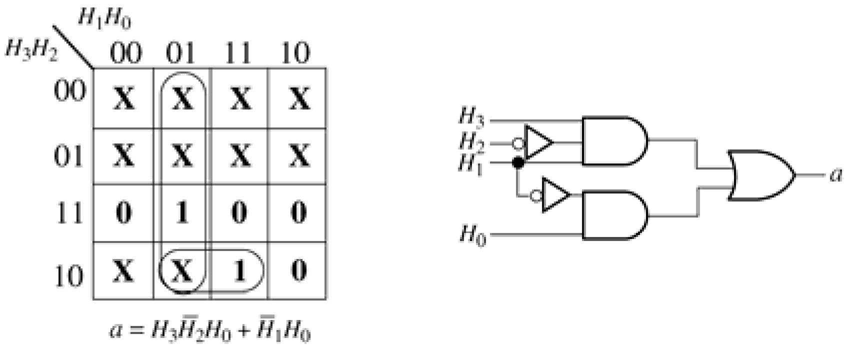
The standard expression requires four 4-input AND gates, one 4-input OR gate, and 3 inverters. The minimum expression requires one 2-input AND gates, one 3-input AND gate, one 2 input OR gate, and 2 inverters.

FIGURE 4-26

Special Design Problems

57. Connect the OR gate output for each segment to an inverter and then use the inverter output to drive the segment with a HIGH.

58. See Figure 4-27. *F* = 1111
The expression for segment *a* to include the letter *F* is:
$a = H_3\overline{H}_2H_1\overline{H}_0 + H_3H_2\overline{H}_1\overline{H}_0 + H_3H_2H_1\overline{H}_0 + H_3H_2H_1H_0$
The expression is minimized in Figure 4-27.

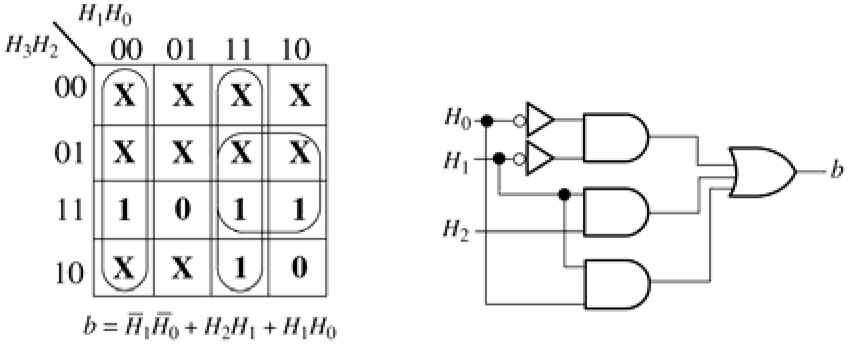


H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	1	0	0
10	X	X	1	0

$a = H_3\overline{H}_2H_0 + \overline{H}_1H_0$

FIGURE 4-27

59. See Figure 4-28. Segment *b* is used for letters *A* and *d*.
$b = H_3\overline{H}_2H_1\overline{H}_0 + H_3H_2\overline{H}_1H_0$

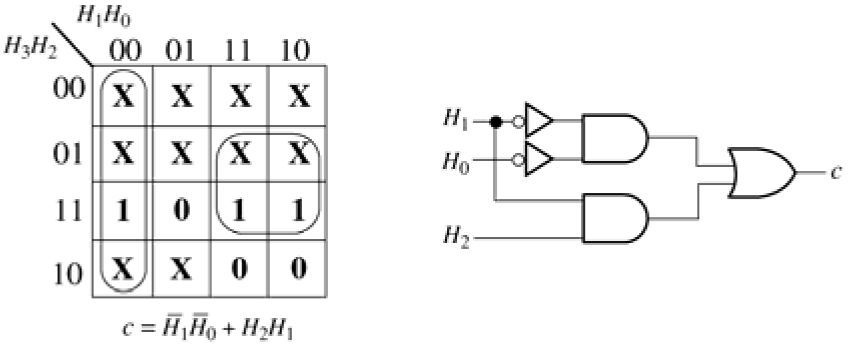


$H_3H_2 \backslash H_1H_0$	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	1	0	1	1
10	X	X	1	0

$b = \overline{H}_1\overline{H}_0 + H_2H_1 + H_1H_0$

FIGURE 4-28

See Figure 4-29. Segment *c* is used for letters *A*, *b*, and *d*.
$c = H_3\overline{H}_2H_1\overline{H}_0 + H_3\overline{H}_2H_1H_0 + H_3H_2\overline{H}_1H_0$

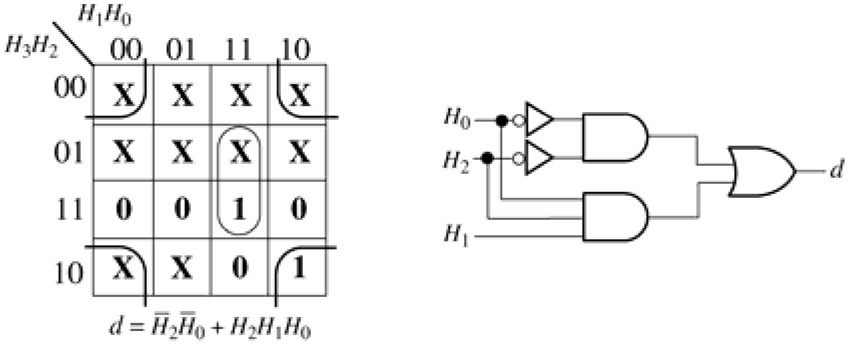


$H_3H_2 \backslash H_1H_0$	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	1	0	1	1
10	X	X	0	0

$c = \overline{H}_1\overline{H}_0 + H_2H_1$

FIGURE 4-29

See Figure 4-30. Segment *d* is used for *b*, *C*, *d*, and *E*.
$d = H_3\overline{H}_2H_1H_0 + H_3H_2\overline{H}_1\overline{H}_0 + H_3H_2\overline{H}_1H_0 + H_3H_2H_1\overline{H}_0$

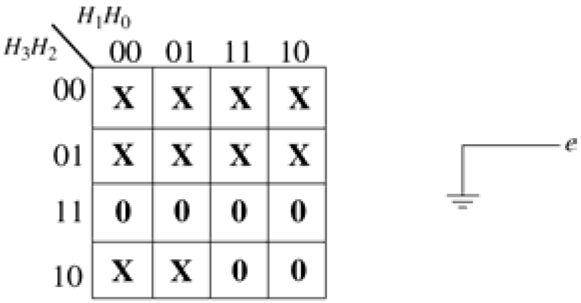


H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	1	0
10	X	X	0	1

$d = \overline{H}_2\overline{H}_0 + H_2H_1H_0$

FIGURE 4-30

See Figure 4-31. Segment *e* is used for *A*, *b*, *C*, *d*, *E*, and *F*.
$e = H_3\overline{H}_2H_1\overline{H}_0 + H_3\overline{H}_2H_1H_0 + H_3H_2\overline{H}_1\overline{H}_0 + H_3H_2\overline{H}_1H_0 + H_3H_2H_1\overline{H}_0 + H_3H_2H_1H_0$

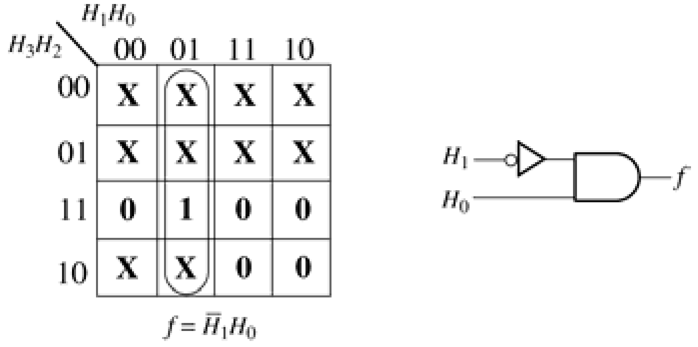


H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	0	0
10	X	X	0	0

Since segment *e* is active-LOW for all letters, *e* = 0.

FIGURE 4-31

See Figure 4-32. Segment *f* is used for *A*, *b*, *C*, *E*, and *F*.
$f = H_3\bar{H}_2H_1\bar{H}_0 + H_3\bar{H}_2H_1H_0 + H_3H_2\bar{H}_1\bar{H}_0 + H_3H_2H_1\bar{H}_0 + H_3H_2H_1H_0$

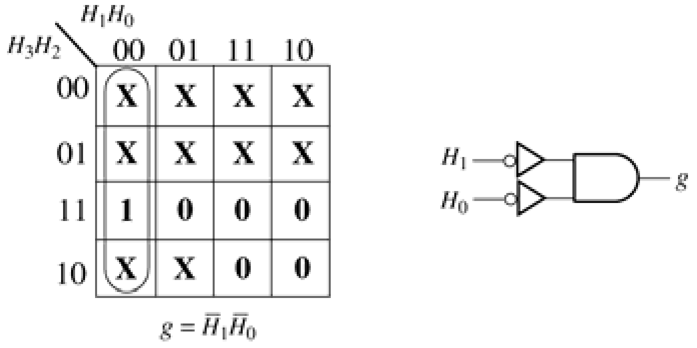


H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	1	0	0
10	X	X	0	0

$f = \bar{H}_1H_0$

FIGURE 4-32

See Figure 4-33. Segment *g* is used in *A*, *b*, *d*, *E*, and *F*.
$g = H_3\bar{H}_2H_1\bar{H}_0 + H_3\bar{H}_2H_1H_0 + H_3H_2\bar{H}_1H_0 + H_3H_2H_1\bar{H}_0 + H_3H_2H_1H_0$

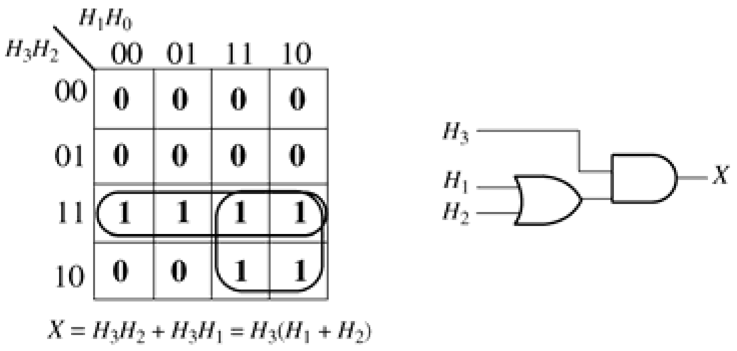


H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	1	0	0	0
10	X	X	0	0

$g = \bar{H}_1\bar{H}_0$

FIGURE 4-33

60. The invalid code detector must disable the display when any numerical input (0-9) occurs. A HIGH enables the display and a LOW disables it. A circuit that detects the numeric codes and produces a LOW is shown in Figure 4-34.



H_3H_2 \ H_1H_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$X = H_3H_2 + H_3H_1 = H_3(H_1 + H_2)$

FIGURE 4-34

Multisim Troubleshooting Practice

61. Input *A* inverter output open.

62. Input *A* of segment *e* OR gate open.

63. Segment *b* OR gate output open.

CHAPTER 5
COMBINATIONAL LOGIC ANALYSIS

Section 5-1 Basic Combinational Logic Circuits

1. See Figure 5-1.

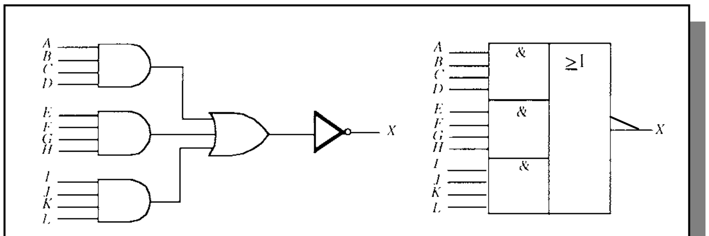


FIGURE 5-1

2. (a) $X = \overline{A}B + \overline{A} + AC$

(b) $X = \overline{\overline{A}B + \overline{A}CD + DB\overline{D}}$

3. (a) $X = ABB$

(b) $X = AB + B$

(c) $X = \overline{A} + B$

(d) $X = (A + B) + AB$

(e) $X = \overline{\overline{A}BC}$

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

4. See Figure 5-2 for the circuit corresponding to each expression.

(a) $X = (A + B)(C + D) = AC + AD + BC + BD$

(b) $X = \overline{\overline{\overline{A}B}C + \overline{CD}} = (\overline{AB}C)(CD) = (\overline{A} + \overline{B})CCD = \overline{A}CD + \overline{B}CD$

(c) $X = (AB + C)D + E = ABD + CD + E$

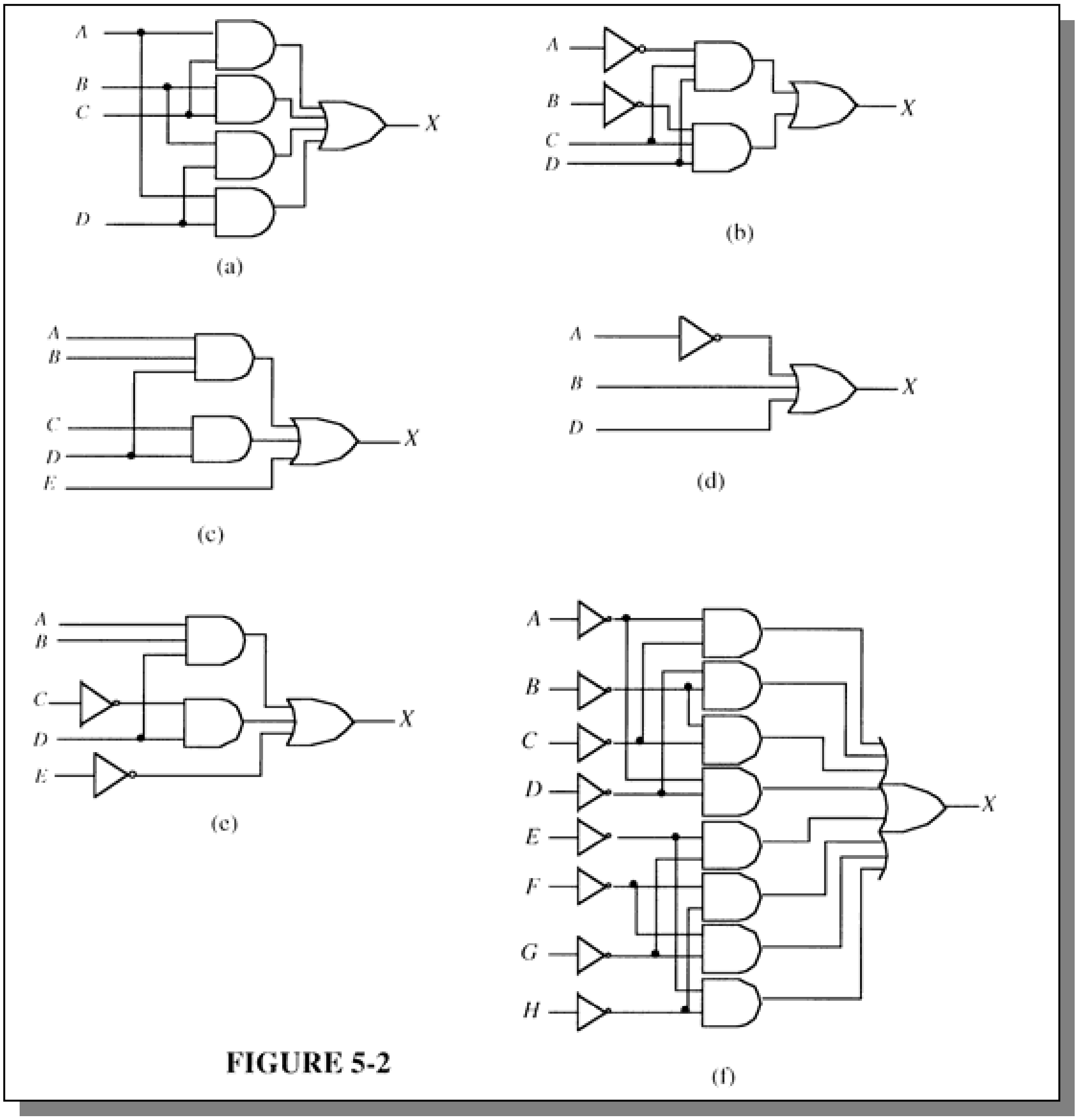
(d) $X = \overline{\overline{\overline{(\overline{\overline{A} + B})(\overline{BC})}} + D} = (\overline{\overline{\overline{A} + B})(\overline{BC})} + D = \overline{A} + B + BC + D = \overline{A} + B + D$

(e) $X = \overline{\overline{(\overline{\overline{AB}} + \overline{C})D + \overline{E}}} = (AB + \overline{C})D + \overline{E} = ABD + \overline{C}D + \overline{E}$

(f) $X = \overline{(\overline{\overline{AB}} + \overline{\overline{CD}})(\overline{\overline{EF}} + \overline{\overline{GH}})} = \overline{(AB + CD)(EF + GH)} = \overline{(AB + CD)} + \overline{(EF + GH)}$

$= (\overline{AB})(\overline{CD}) + (\overline{EF})(\overline{GH})$

$= (\overline{A} + \overline{B})(\overline{C} + \overline{D}) + (\overline{E} + \overline{F})(\overline{G} + \overline{H}) = \overline{A}\,\overline{C} + \overline{B}\,\overline{C} + \overline{A}\,\overline{D} + \overline{B}\,\overline{D} + \overline{E}\,\overline{G} + \overline{F}\,\overline{G} + \overline{E}\,\overline{H} + \overline{F}\,\overline{H}$



(a) (b) (c) (d) (e) (f)

FIGURE 5-2

5. (a) $X = ABB$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(b) $X = AB + B$

A	B	X
0	0	0
0	1	1
1	0	0
1	1	1

(c) $X = \overline{A} + B$

A	B	X
0	0	1
0	1	1
1	0	0
1	1	1

(d) $X = (A + B) + AB$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e) $X = \overline{\overline{A}BC}$

A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

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

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Chapter 5

6. (a) $X = (A + B)(C + D)$

A	*B*	*C*	*D*	*X*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(b) $X = \overline{\overline{\overline{AB}C} + \overline{CD}}$

A	*B*	*C*	*D*	*X*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

(c) $X = (AB + C)D + E$

A	*B*	*C*	*D*	*E*	*X*
0	0	0	0	0	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	1	1
0	0	1	1	0	1
0	0	1	1	1	1
0	1	0	0	0	0
0	1	0	0	1	1
0	1	0	1	0	0
0	1	0	1	1	1
0	1	1	0	0	0
0	1	1	0	1	1
0	1	1	1	0	1
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	0	0	1	1
1	1	0	1	0	1
1	1	0	1	1	1
1	1	1	0	0	0
1	1	1	0	1	1
1	1	1	1	0	1
1	1	1	1	1	1

(d) $X = \overline{\overline{\overline{\overline{(\overline{A} + B)}(\overline{BC})} + D}}$

A	*B*	*C*	*D*	*X*
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

(e) $X = \overline{\overline{(\overline{\overline{AB}}+\overline{C})D}}+\overline{E}$

A	B	C	D	E	X	A	B	C	D	E	X
0	0	0	0	0	1	1	0	0	0	0	1
0	0	0	0	1	0	1	0	0	0	1	0
0	0	0	1	0	1	1	0	0	1	0	1
0	0	0	1	1	1	1	0	0	1	1	1
0	0	1	0	0	1	1	0	1	0	0	1
0	0	1	0	1	0	1	0	1	0	1	0
0	0	1	1	0	1	1	0	1	1	0	1
0	0	1	1	1	0	1	0	1	1	1	0
0	1	0	0	0	1	1	1	0	0	0	1
0	1	0	0	1	0	1	1	0	0	1	0
0	1	0	1	0	1	1	1	0	1	0	1
0	1	0	1	1	1	1	1	0	1	1	1
0	1	1	0	0	1	1	1	1	0	0	1
0	1	1	0	1	0	1	1	1	0	1	0
0	1	1	1	0	1	1	1	1	1	0	1
0	1	1	1	1	0	1	1	1	1	1	1

(f) $X = \overline{(\overline{\overline{AB}}+\overline{\overline{CD}})(\overline{\overline{EF}}+\overline{\overline{GH}})}$

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

For all other entries X = 0.
X = *don't care*
An abbreviated table is shown because there are 256 combinations.

7. $X = \overline{A\overline{B}+\overline{A}B} = (\overline{A\overline{B}})(\overline{\overline{A}B}) = (\overline{A}+B)(A+\overline{B})$

Section 5-2 Implementing Combinational Logic

8. Let G = guard, S = switch, M = motor temp, and P = power. See Figure 5-3.
$P = \overline{\overline{G}S + MS}$

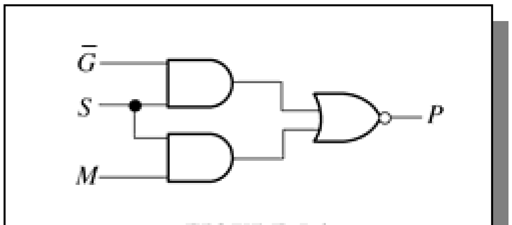


FIGURE 5-3

9. $\boldsymbol{X} =\ \overline{ABCD + EFGH}$

10. See Figure 5-4.

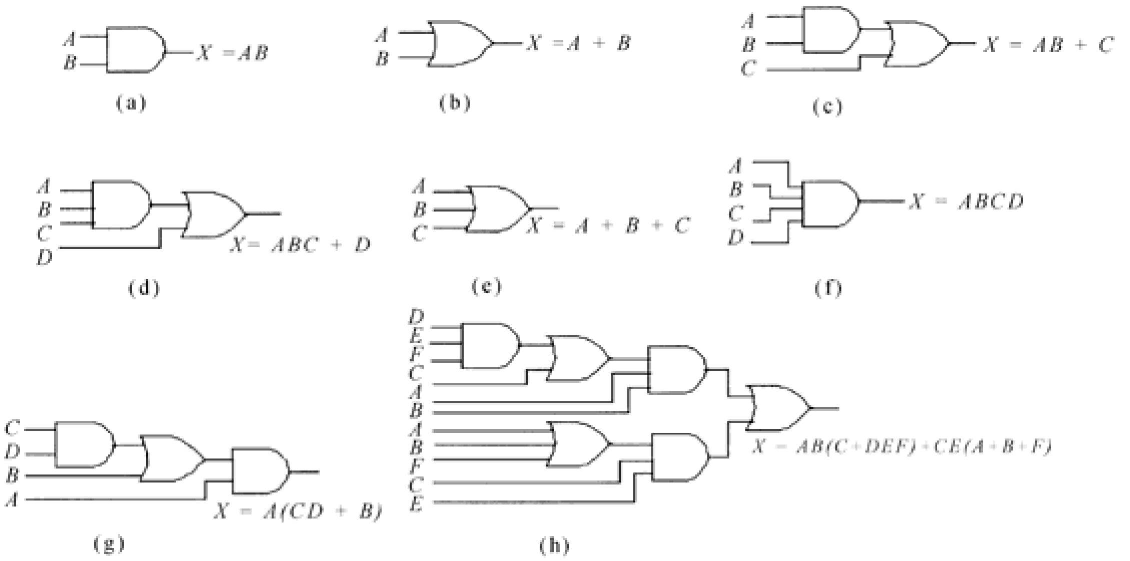


FIGURE 5-4

11. See Figure 5-5.

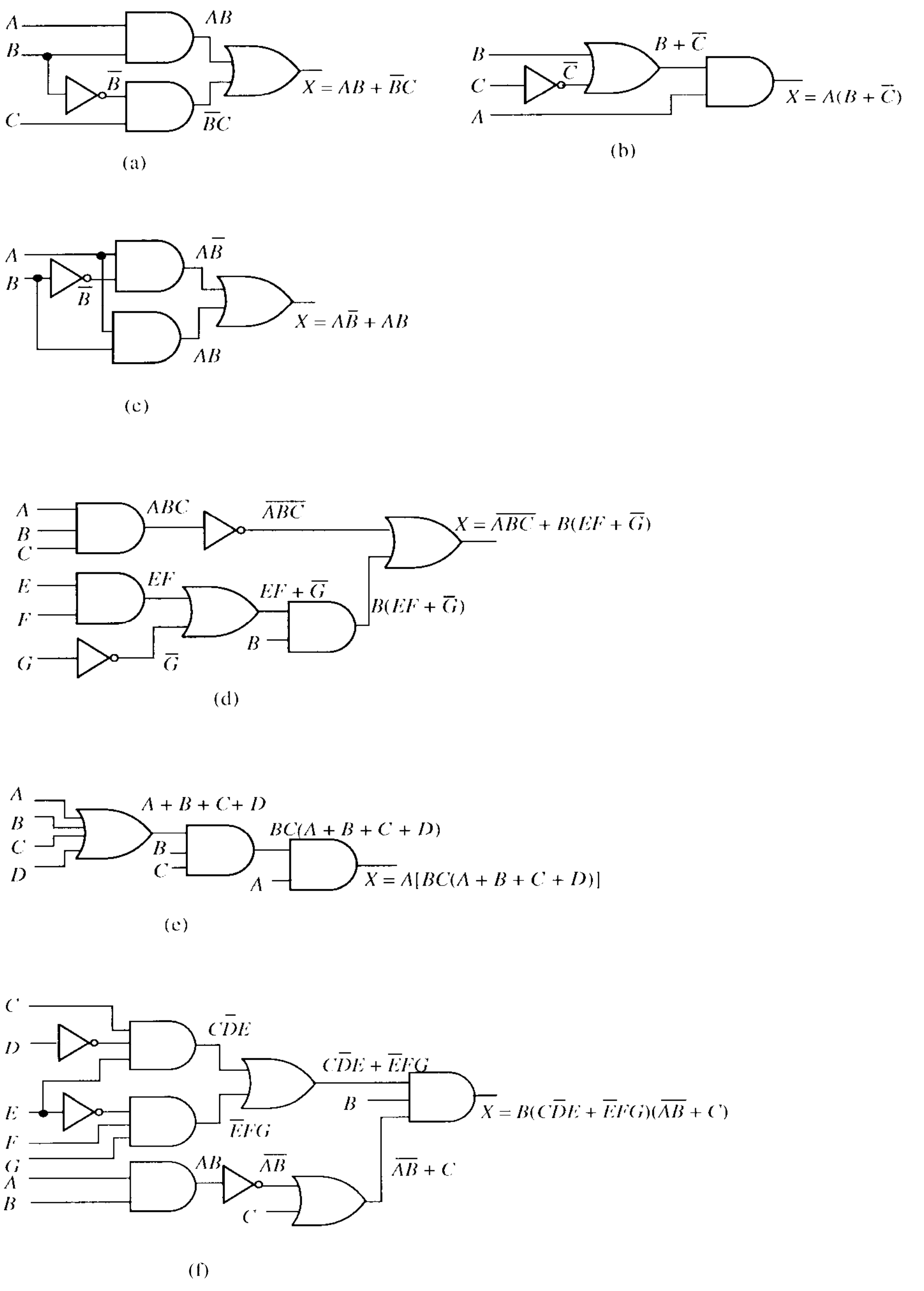
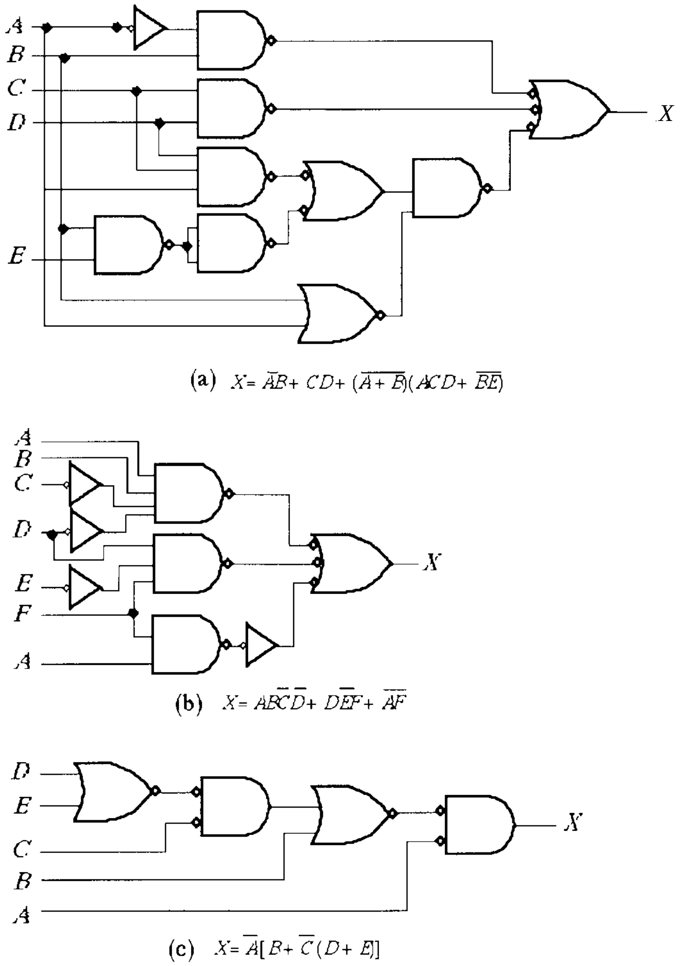


FIGURE 5-5

12. See Figure 5-6.



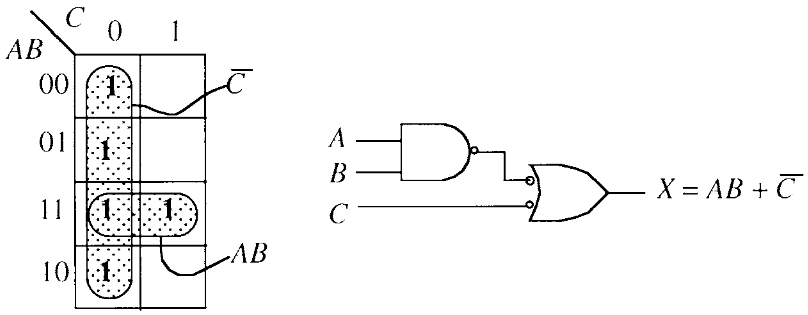
(a) $X = \bar{A}B + CD + (\overline{A + B})(ACD + \overline{BE})$

(b) $X = AB\bar{C}\bar{D} + D\bar{E}F + \overline{AF}$

(c) $X = \bar{A}[B + \bar{C}(D + E)]$

FIGURE 5-6

13. $X = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{C} + A\overline{B}\overline{C} + AB\overline{C} + ABC$
See Figure 5-7.



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

FIGURE 5-7

14. $X = \overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}CD + \overline{A}B\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D + A\overline{B}C\overline{D} + A\overline{B}CD + ABCD$
See Figure 5-8.

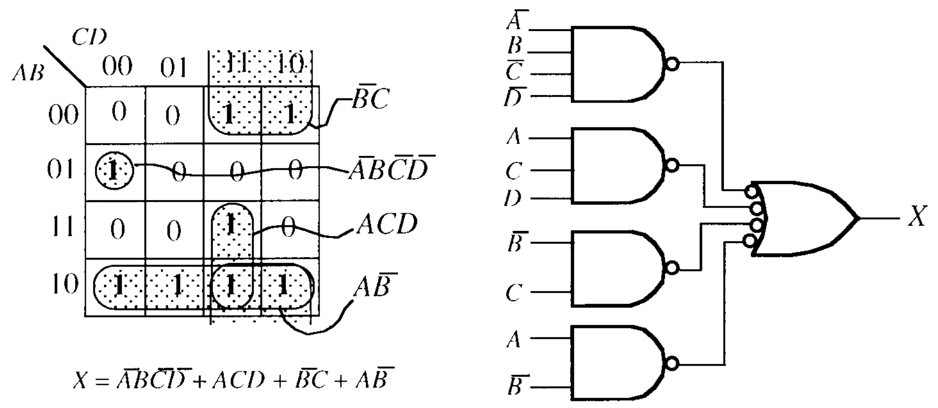


FIGURE 5-8

15. $X = AB + ABC = AB(1 + C) = \boldsymbol{AB}$

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$X = 1$ when $AB = 1$, no matter what C is.

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Since C is a don't care variable, the output depends only on A and B as shown by the two-variable truth table above which is implemented with the AND gate in Figure 5-9.

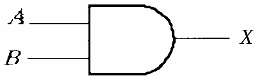


FIGURE 5-9

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

$= (\overline{A}\,\overline{B}\,\overline{C} + \overline{B}\,\overline{C})\overline{C} = \overline{A}\,\overline{B}\,\overline{C} + \overline{B}\,\overline{C} = \overline{B}\,\overline{C}(A+1) = \boldsymbol{\overline{B}\,\overline{C}}$

See Figure 5-10.

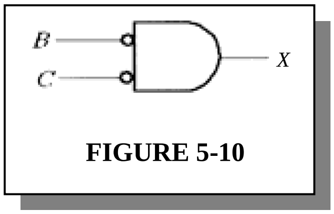


FIGURE 5-10

The output is dependent only on *B* and *C*. The value of *A* does not matter. The NOR gate behaves as a negative-AND.

A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

17. (a) $X = AB + \overline{B}C$

No simplification. See Figure 5-11.

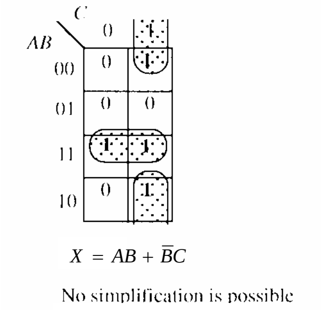


FIGURE 5-11

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

No simplification. Equation can be expressed in another form, as indicated in Figure 5-12.

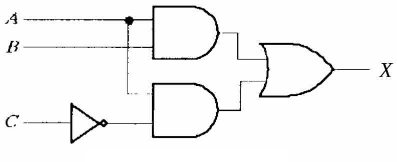


FIGURE 5-12

(c) $X = AB + A\overline{B} = A(B+\overline{B}) = \boldsymbol{A}$

A direct connection from input to output. No gates required.

(d) $X = \overline{ABC} + B(EF + \overline{G}) = \overline{A} + \overline{B} + \overline{C} + BEF + B\overline{G}$

$= \overline{A} + \overline{C} + BEF + \overline{B} + \overline{G} = \boldsymbol{\overline{A} + \overline{C} + \overline{B} + EF + \overline{G}}$

See Figure 5-13.

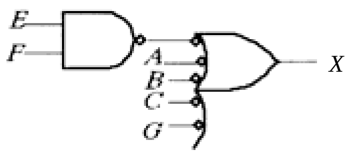


FIGURE 5-13

(e) $X = A(BC(A + B + C + D)) = ABCA + ABCB + ABCC + ABCD$

$= ABC + ABC + ABC + ABCD = ABC + ABC(1 + D)$

$= ABC + ABC = \boldsymbol{ABC}$

See Figure 5-14.

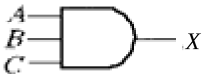


FIGURE 5-14

(f) $X = B(C\overline{D}E + \overline{E}FG)(\overline{AB} + C) = (BC\overline{D}E + B\overline{E}FG)(\overline{A} + \overline{B} + C)$

$= \overline{A}BC\overline{D}E + \overline{A}B\overline{E}FG + BC\overline{D}E + BC\overline{E}FG$

$= BC\overline{D}E(\overline{A} + 1) + \overline{A}B\overline{E}FG + BC\overline{E}FG$

$= \boldsymbol{BC\overline{D}E + \overline{A}B\overline{E}FG + BC\overline{E}FG}$

See Figure 5-15.

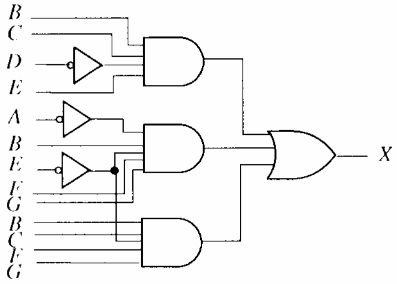


FIGURE 5-15

18. (a) $X = \overline{A}B + CD + (\overline{A + B})(ACD + \overline{BE}) = \overline{A}B + CD + \overline{A}\,\overline{B}(ACD + \overline{B} + \overline{E})$

$= \overline{A}B + CD + \overline{A}\,\overline{B} + \overline{A}\,\overline{B}\,\overline{E} = \overline{A}(B + \overline{B}) + CD + \overline{A}\,\overline{B}\,\overline{E}$

$= \overline{A} + \overline{A}\,\overline{B}\,\overline{E} + CD = \overline{A}(1 + \overline{B}\,\overline{E}) + CD = \boldsymbol{\overline{A} + CD}$

See Figure 5-16.

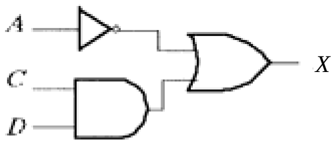


FIGURE 5-16

(b) $X = AB\overline{C}\,\overline{D} + D\overline{E}F + \overline{AF} = AB\overline{C}\,\overline{D} + \overline{D}EF + \overline{A} + \overline{F}$

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

See Figure 5-17.

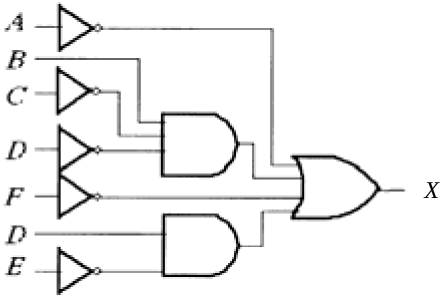


FIGURE 5-17

(c) $X = \overline{A}(B + \overline{C}(D + E)) = \overline{A}(B + \overline{C}D + \overline{C}E) = \boldsymbol{\overline{A}B + \overline{A}\,\overline{C}D + \overline{A}\,\overline{C}E}$

See Figure 5-18.

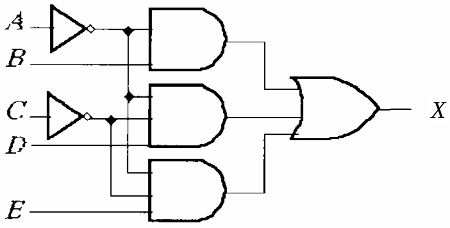


FIGURE 5-18

19. The SOP expressions are developed as follows and the resulting circuits are shown in Figure 5-19.

(a) $X = (A + B)(C + D) = \mathbf{AC + AD + BC + BD}$

(b) $X = \overline{\overline{\overline{AB}C} + \overline{CD}} = (\overline{AB}C)(CD) = (\overline{A} + \overline{B})CCD = \mathbf{\overline{A}CD} + \mathbf{\overline{B}CD}$

(c) $X = (AB + C)D + E = \mathbf{ABD + CD + E}$

(d) $X = \overline{\overline{\overline{(\overline{\overline{A} + B})(\overline{BC})} + D}} = \overline{(\overline{\overline{A} + B})(\overline{BC})} + D = \overline{A} + B + BC + D$

$= \overline{A} + B(1 + C) + D = \mathbf{\overline{A}} + \mathbf{B} + \mathbf{D}$

(e) $X = \overline{\overline{(\overline{\overline{AB}} + \overline{C})D}} + \overline{E} = (AB + \overline{C})D + \overline{E} = \mathbf{ABD} + \mathbf{\overline{C}D} + \mathbf{\overline{E}}$

(f) $X = (\overline{\overline{AB}} + \overline{\overline{CD}})(\overline{\overline{EF}} + \overline{\overline{GH}}) = \overline{(AB + CD)(EF + GH)} = \overline{(AB + CD)} + \overline{(EF + GHG)}$

$= (\overline{AB})(\overline{CD}) + (\overline{EF})(\overline{GH}) = (\overline{A} + \overline{B})(\overline{C} + \overline{D}) + (\overline{E} + \overline{F})(\overline{G} + \overline{H})$

$= \mathbf{\overline{A}\,\overline{C}} + \mathbf{\overline{B}\,\overline{C}} + \mathbf{\overline{A}\,\overline{D}} + \mathbf{\overline{B}\,\overline{D}} + \mathbf{\overline{E}\,\overline{G}} + \mathbf{\overline{F}\,\overline{G}} + \mathbf{\overline{E}\,\overline{H}} + \mathbf{\overline{F}\,\overline{H}}$

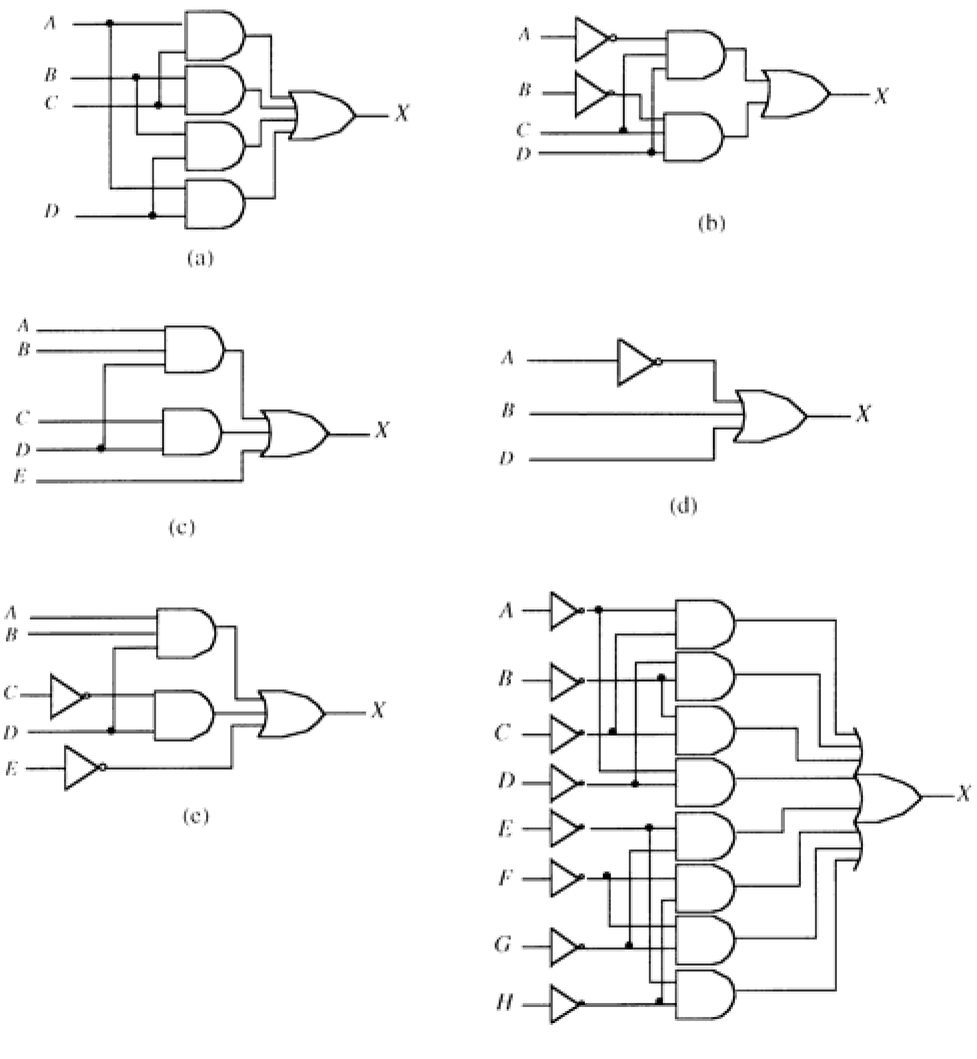


FIGURE 5-19

Chapter 5

Section 5-3 The Universal Property of NAND and NOR Gates

20. See Figure 5-20.

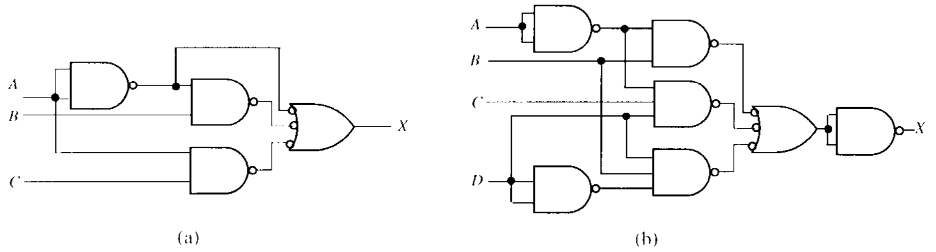


FIGURE 5-20

21. $X = \overline{\overline{\overline{(\overline{AB})(\overline{B+C})}}+C}$

See Figure 5-21.

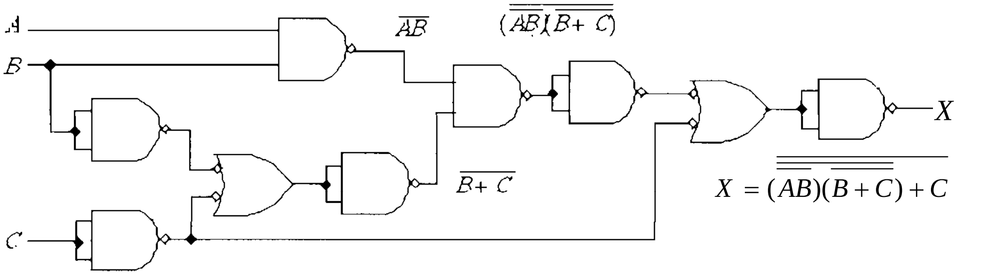

A
B
C
$\overline{AB}$
$(\overline{\overline{AB}})(\overline{\overline{B+C}})$
$\overline{B+C}$
X
$X = \overline{\overline{\overline{(\overline{AB})(\overline{B+C})}}+C}$

FIGURE 5-21

22. See Figure 5-22.

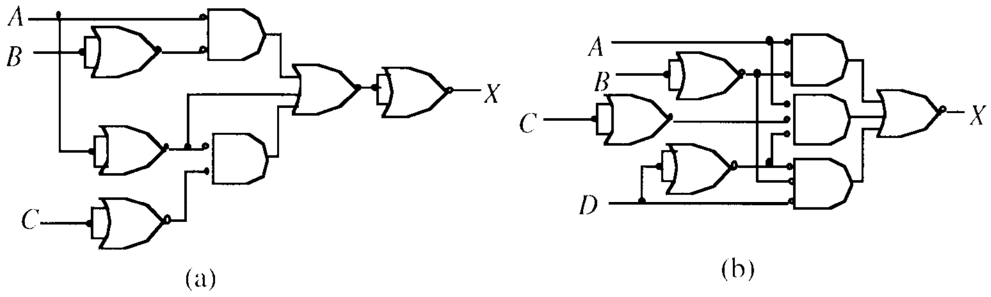


FIGURE 5-22

23. See Figure 5-23.

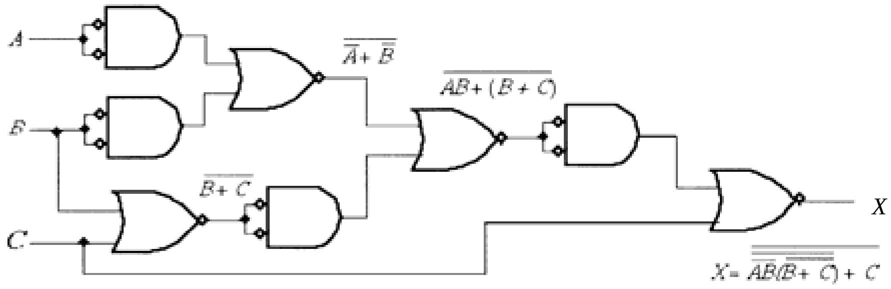

A
B
C
$\overline{\overline{A}+\overline{B}}$
$\overline{AB+(B+C)}$
$\overline{B+C}$
X
$X=\overline{\overline{\overline{AB}\,\overline{(B+C)}}+C}$

FIGURE 5-23

Section 5-4 Combinational Logic Using NAND and NOR Gates

24. (a) $X = ABC$

See Figure 5-24.

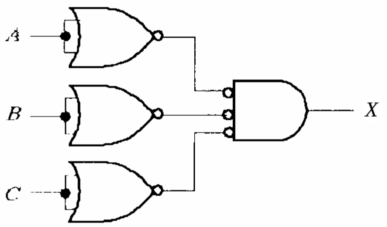


FIGURE 5-24

(b) $X = \overline{ABC}$

See Figure 5-25.

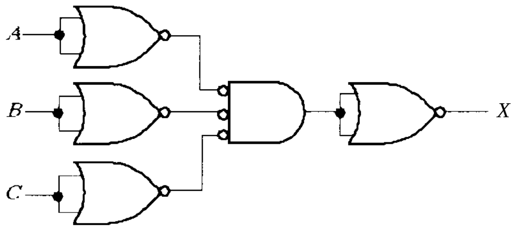


FIGURE 5-25

(c) $X = A + B$

See Figure 5-26.

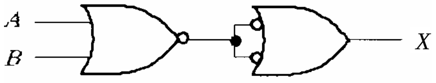


FIGURE 5-26

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

See Figure 5-27.

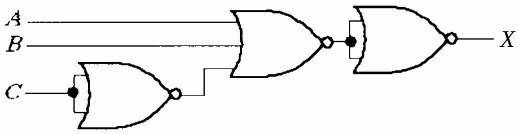


FIGURE 5-27

(e) $X = \overline{AB} + \overline{CD}$

See Figure 5-28.

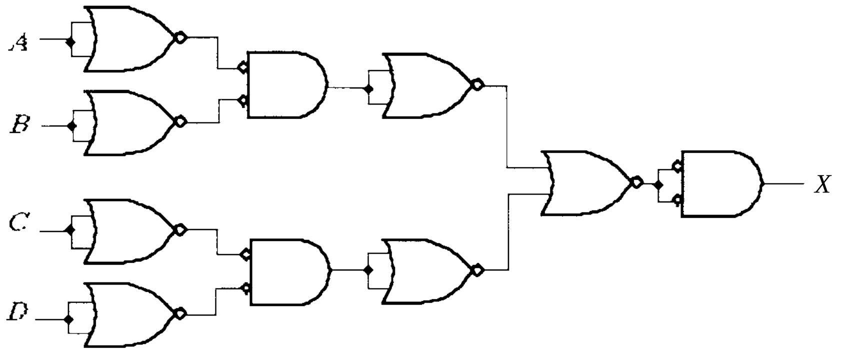


FIGURE 5-28

(f) $X = (A + B)(C + D)$

See Figure 5-29.

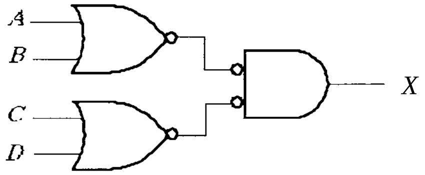


FIGURE 5-29

(g) $X = AB[C(\overline{DE} + \overline{AB}) + \overline{BCE}]$

See Figure 5-30.

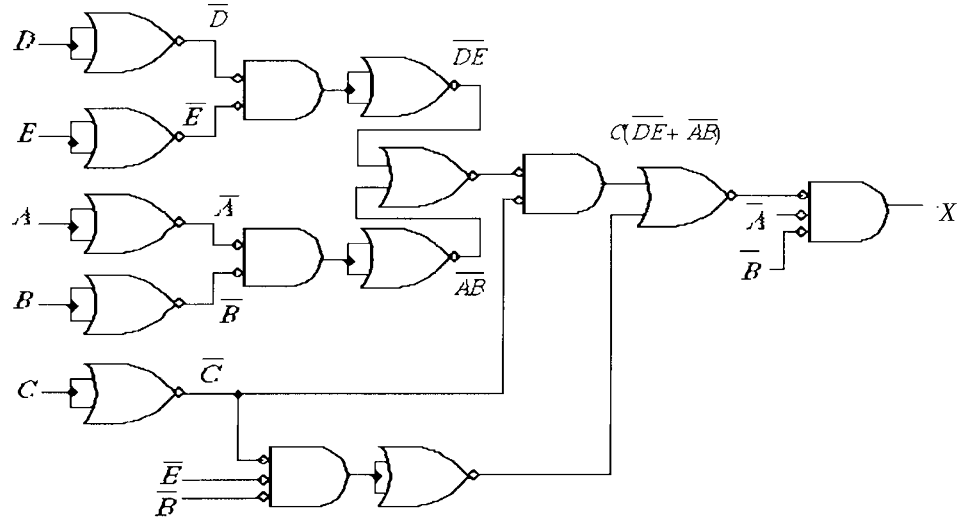


FIGURE 5-30

25. (a) $X = ABC$

See Figure 5-31.

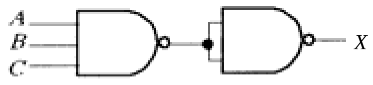


FIGURE 5-31

(b) $X = \overline{ABC}$

See Figure 5-32.

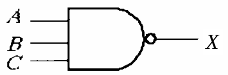


FIGURE 5-32

(c) $X + A + B$

See Figure 5-33.

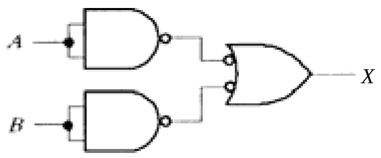


FIGURE 5-33

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

See Figure 5-34.

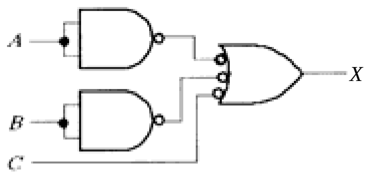


FIGURE 5-34

(e) $X = \overline{AB} + \overline{CD}$

See Figure 5-35.

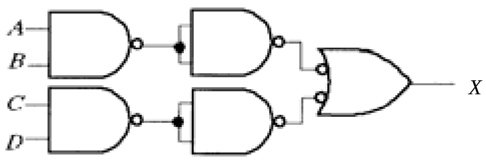


FIGURE 5-35

(f) $X = (A + B)(C + D)$

See Figure 5-36.

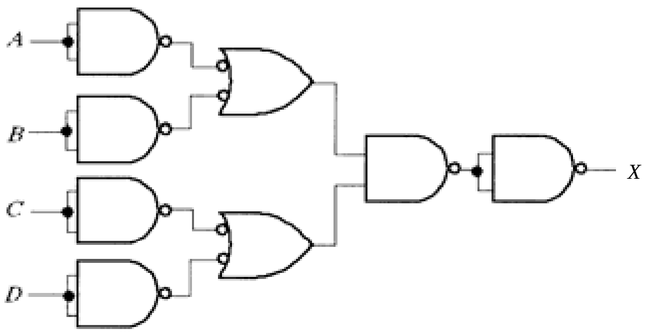


FIGURE 5-36

(g) $X = AB[C(\overline{DE} + \overline{AB}) + \overline{BCE}]$

See Figure 5-37.

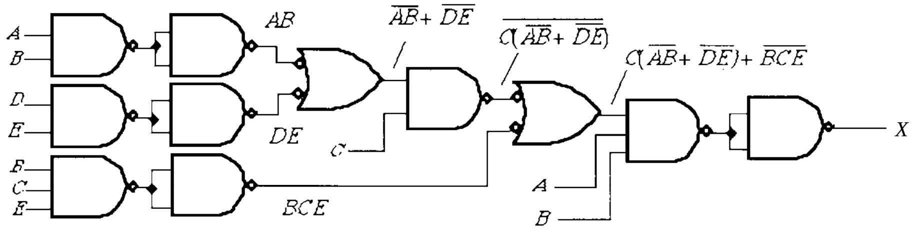

A
B
D
E
B
C
E
AB
DE
BCE
$\overline{AB} + \overline{DE}$
C
$\overline{C(\overline{AB} + \overline{DE})}$
$C(\overline{AB} + \overline{DE}) + \overline{BCE}$
A
B
X

FIGURE 5-37

26. (a) $X = AB$

See Figure 5-38.

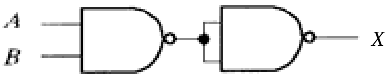


FIGURE 5-38

(b) $X = A + B$

See Figure 5-39.

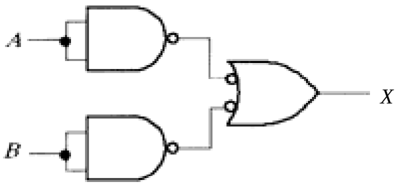


FIGURE 5-39

(c) $X = AB + C$

See Figure 5-40.

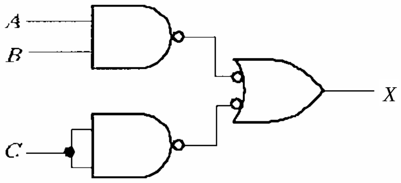


FIGURE 5-40

(d) $X = ABC + D$

See Figure 5-41.

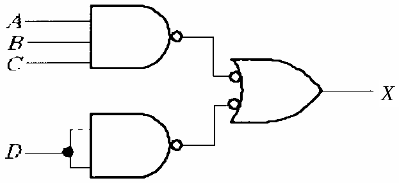


FIGURE 5-41

(e) $X = A + B + C$
See Figure 5-42.

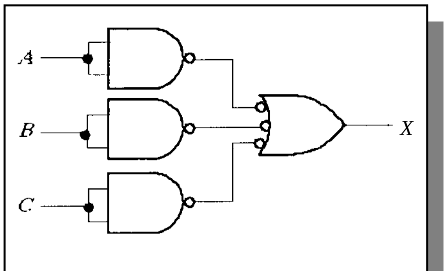


FIGURE 5-42

(f) $X = ABCD$
See Figure 5-43.

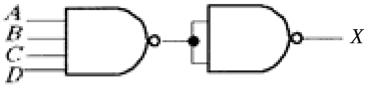


FIGURE 5-43

(g) $X = A(CD + B) = \boldsymbol{ACD + AB}$

See Figure 5-44.

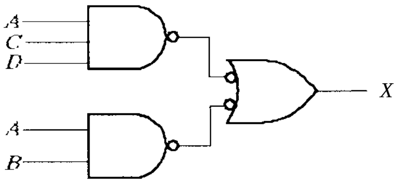


FIGURE 5-44

(h) $X = AB(C + DEF) + CE(A + B + F)$
$= \boldsymbol{ABC + ABDEF + CEA + CEB + CEF}$

See Figure 5-45.

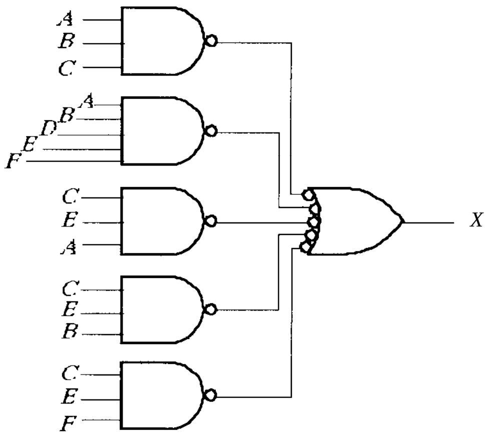


FIGURE 5-45

27. (a) $X = AB + \overline{B}C$

See Figure 5-46.

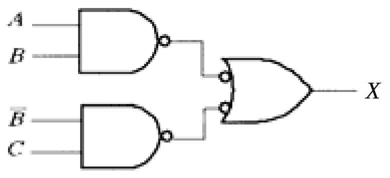


FIGURE 5-46

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

See Figure 5-47.

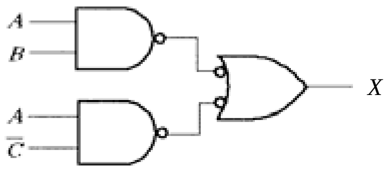


FIGURE 5-47

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

See Figure 5-48.

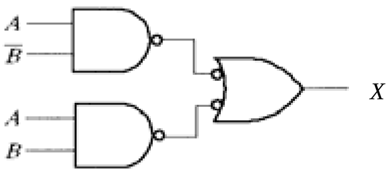


FIGURE 5-48

(d) $X = \overline{ABC} + B(EF + \overline{G}) = \boldsymbol{\overline{A}} + \boldsymbol{\overline{B}} + \boldsymbol{\overline{C}} + \boldsymbol{BEF} + \boldsymbol{B\overline{G}}$

See Figure 5-49.

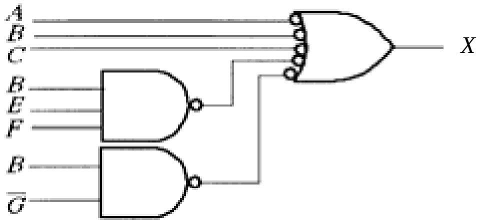


FIGURE 5-49

(e) $X = A[BC(A + B + C + D)] = ABCA + ABCB + ABCC + ABCD$

$= ABC + ABC + ABC + ABCD + ABC(1 + D) = \mathbf{ABC}$

See Figure 5-50.

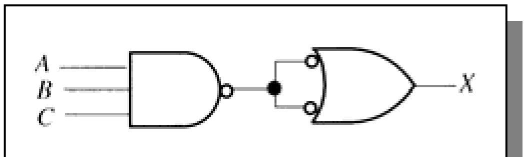


FIGURE 5-50

(f) $X = B(C\overline{D}E + \overline{E}FG)(\overline{AB} + C) = B(C\overline{D}E + \overline{E}FG)(\overline{A} + \overline{B} + C)$

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

$= \overline{A}B\overline{E}FG + B\overline{B}\overline{E}FG + BC\overline{D}E + BC\overline{E}FG$

$= \boldsymbol{\overline{A}B\overline{E}FG + BC\overline{D}E + BC\overline{E}FG}$

See Figure 5-51.

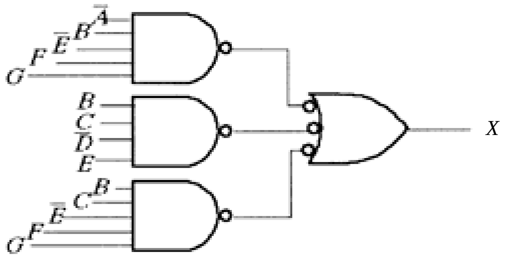


FIGURE 5-51

Section 5-5 Logic Circuit Operation with Pulse Waveform Inputs

28. $\boldsymbol{X} =$ $\overline{\overline{A} + \overline{B} + B} = AB\overline{B} = 0$
The output X is always LOW.

29. $X = \overline{(\overline{AB})B} = A + \overline{B} + \overline{B} = \boldsymbol{A + \overline{B}}$

See Figure 5-52.

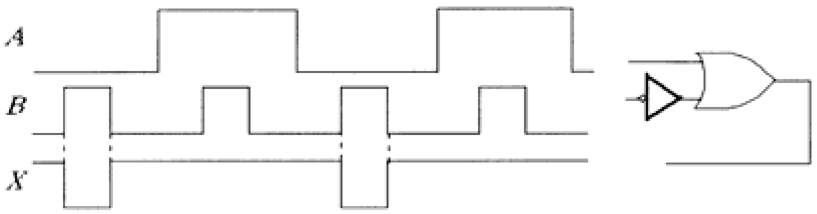


FIGURE 5-52

30. *X* is HIGH when *ABC* are all HIGH or when *A* is HIGH and *B* is LOW and *C* is LOW or when *A* is HIGH and *B* is LOW and *C* is HIGH.

$X = ABC + A\overline{B}\overline{C} + A\overline{B}C$

See Figure 5-53.

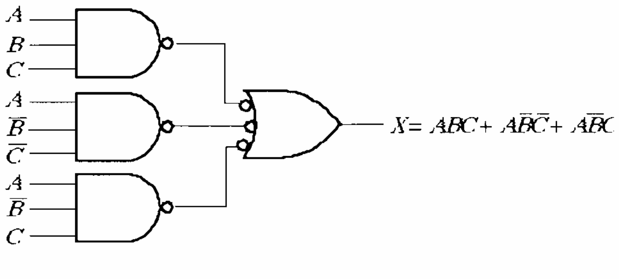


FIGURE 5-53

31. *X* is HIGH when *A* is HIGH, *B* is LOW, and *C* is LOW. We do not know if *X* is HIGH when all inputs are HIGH.

$X = A\overline{B}\overline{C}$

See Figure 5-54.

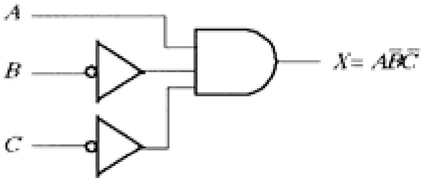


FIGURE 5-54

32. See Figure 5-55.

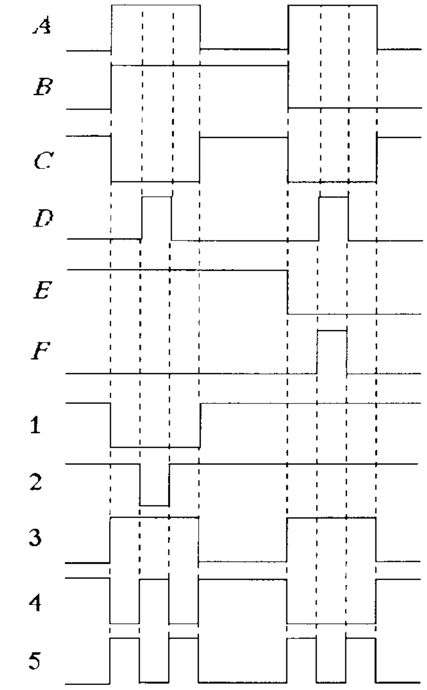


FIGURE 5-55

33. The output pulse is sufficiently wide. It is greater than 25 ns. A maximum is not specified. See Figure 5-56.

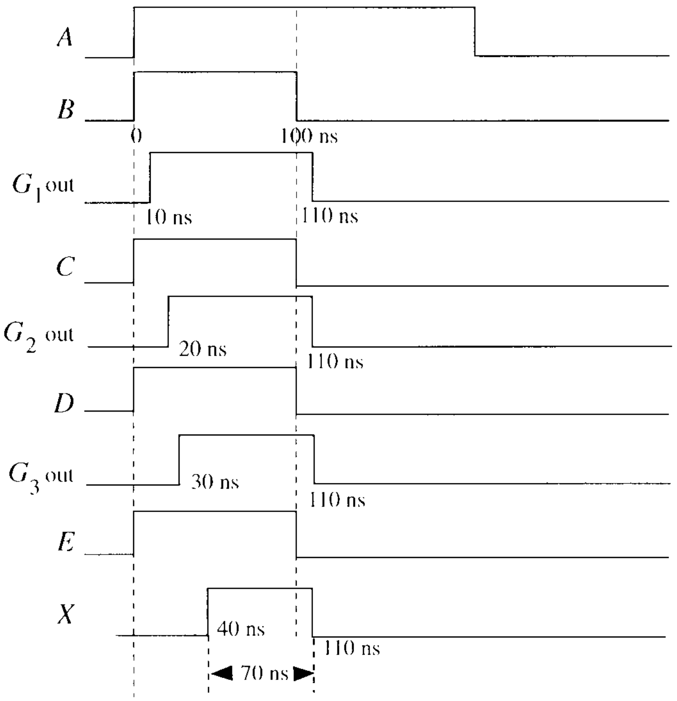


FIGURE 5-56

Section 5-6 Troubleshooting

34. $X = \overline{\overline{AB} + \overline{CD}} = ABCD$

X is HIGH only when $ABCD$ are all HIGH. This does not occur in the waveforms, so X should remain LOW. **The output is incorrect.**

35. $X = ABC + D\overline{E}$

Since X is the same as the G_3 output, either G_1 or G_2 has failed with its output *stuck LOW*.

36. $X = AB + CD + EF$

X does not go HIGH when C and D are HIGH. G_2 has failed with the output *open* or *stuck HIGH* or the corresponding input to G_4 is *open*.

37. See Figure 5-57.

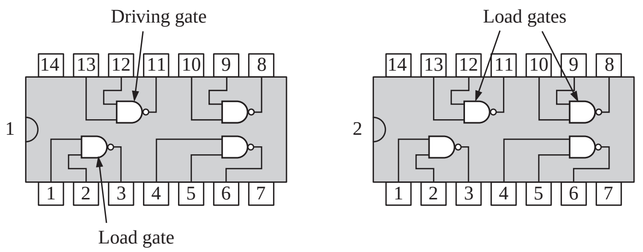


FIGURE 5-57

38. $X = \overline{\overline{A}\,\overline{B} + \overline{C}\,\overline{D} + \overline{E}\,\overline{F}} = (\overline{\overline{A}\,\overline{B}})(\overline{\overline{C}\,\overline{D}})(\overline{\overline{E}\,\overline{F}}) = (A + B)(C + D)(E + F)$

Since X does not go HIGH when C or D is HIGH, the output of gate G_2 must be *stuck LOW*.

39. (a) $X = (\overline{A} + \overline{B} + C)E + (C + \overline{D})E = \overline{A}E + \overline{B}E + CE + \cancel{CE} + \overline{D}E$

$= \overline{A}E + \overline{B}E + CE + \overline{D}E$

See Figure 5-58.

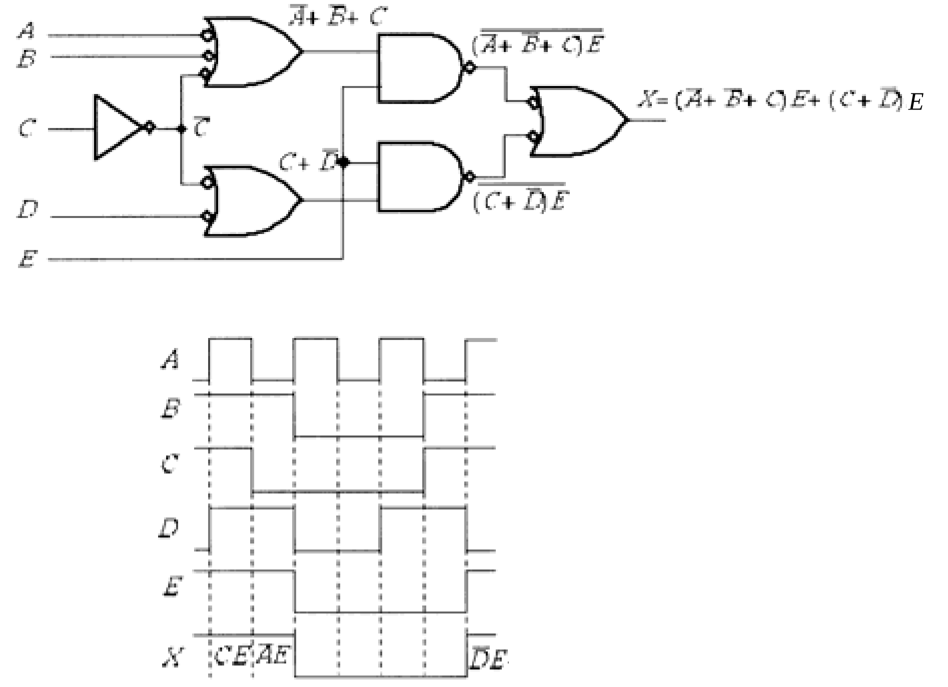

A
B
C
D
E
$\overline{A} + \overline{B} + C$
$\overline{C}$
$C + \overline{D}$
$\overline{(\overline{A} + \overline{B} + C)E}$
$\overline{(C + \overline{D})E}$
$X = (\overline{A} + \overline{B} + C)E + (C + \overline{D})E$
A
B
C
D
E
X
CE
$\overline{A}E$
$\overline{D}E$

FIGURE 5-58

(b) $X = E + E(\overline{D} + C) = E(1 + \overline{D} + C) = \boldsymbol{E}$

Waveform *X* is the same as waveform *E*, in Figure 5-58. Since this is the correct waveform, the open output of gate G_3 does not show up for this *particular* set of input waveforms.

(c) $X = E + E(\overline{A} + \overline{B} + C) = E(1 + \overline{A} + \overline{B} + C) = \boldsymbol{E}$

Again waveform *X* is the same as waveform *E*. As strange as it may seem, the shorted input to G_5 does not affect the output for this *particular* set of input waveforms.

Conclusion: the two faults are not indicated in the output waveform for these particular inputs.

40. TP = $\overline{\overline{A}\,\overline{B}+\overline{C}\,\overline{D}}$

The output of the $\overline{C}\,\overline{D}$ gate is *stuck LOW*. See Figure 5-59.

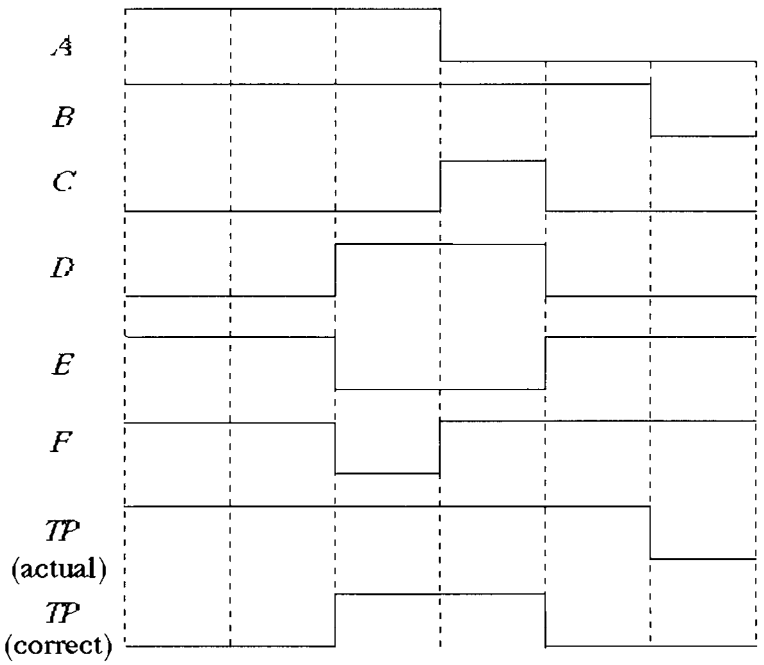


FIGURE 5-59

Section 5-7 Combinational Logic with VHDL

41. X <= A **and** B **and** C

42.

```
entity Circuit5_54b is
        port (A, B, C, D: in bit; X: out bit);
end entity Circuit5_54b;
architecture LogicFunction of Circuit5_54b is
begin
        X <= not(not A and B) or (not A and C and D) or (D and B and not D);
end architecture LogicFunction;
```

43.(e)

```
entity Circuit5_55e is
        port (A, B, C: in bit; X: out bit);
end entity Circuit5_55e;
architecture LogicFunction of Circuit5_55e is
begin
        X <= (not A and B) or B or (B and not C) or (not A and not C) or (B and not C) or not C;
        end architecture LogicFunction;
```

(f)
```
entity Circuit5_55f is
        port (A, B, C: in bit;  X: out bit);
end entity Circuit5_55f;
architecture LogicFunction of Circuit5_55f is
begin
        X <= (A or B) and (not B or C);
end architecture Logic Function;
```

44. See Figure 5-60 for input/output, gate, and signal labeling.

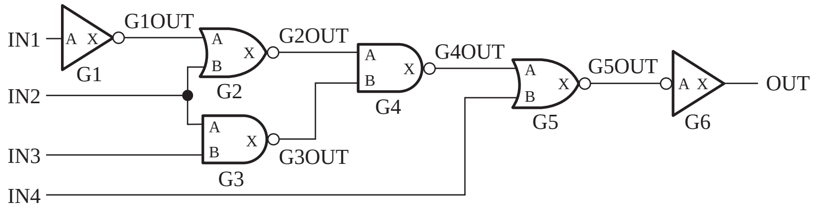


FIGURE 5-60

```
--Program for the logic circuit in Figure 5-60 (textbook Figure 5-56(d))
entity (Circuit5_56d is
        port (IN1, IN2, IN3, IN4: in bit; OUT: out bit);
end entity Circuit5_56d;
architecture LogicOperation of Circuit5_56d is
--Component declaration for inverter
component Inverter is
        port (A: in bit; X: out bit);
end component Inverter;
--Component declaration for NOR gate
component NORgate is
        port (A, B: in bit; X: out bit);
end component NOR gate;
--Component declaration for NAND gate
component NANDgate is
        port (A, B: in bit; X: out bit);
end component NANDgate;
signal G1OUT, G2OUT, G3OUT, G4OUT, G5OUT: bit;
begin
        G1:  Inverter port map (A => IN1, X => G1OUT);
        G2:  NORgate port map (A => G1OUT, B => IN2, X => G2OUT);
        G3:  NAND gate port map (A => IN2, B => IN3, X => G3OUT);
        G4:  NANDgate port map (A => G2OUT, B => G3OUT, X => G4OUT);
        G5:  NORgate port map (A => G4OUT, B => IN4, X => G5OUT);
        G6:  Inverter port map  (A => G5OUT, X => OUT);
end architecture  LogicOperation;
```

45. See Figure 5-61 for input/output, gate, and signal labeling.

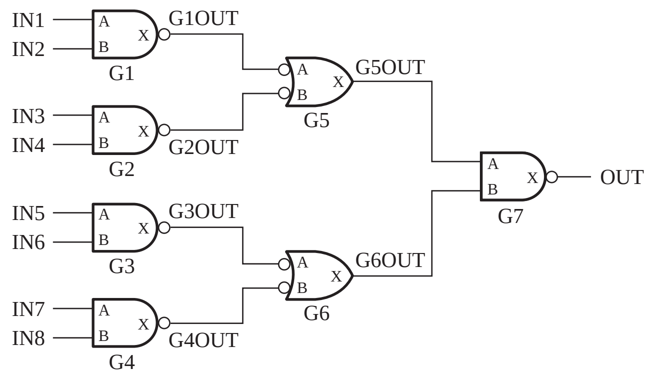


FIGURE 5-61

```
--Program for the logic circuit in Figure 5-61 (textbook Figure 5-56(f))
entity  Circuit5_56f is
        port (IN1, IN2, IN3, IN4, IN5, IN6, IN7, IN8:  in bit; OUT:  out bit);
end entity Circuit5_56f;
architecture LogicFunction of Circuit5_56f is
--Component declaration for NAND gate
component NANDgate is
        port (A, B: in bit; X: out bit);
end component NANDgate;
signal G1OUT, G2OUT, G3OUT, G4OUT, G5OUT, G6OUT:  bit;
begin
        G1:  NANDgate port map (A => IN1, B => IN2, X => G1OUT);
        G2:  NANDgate port map (A => IN3, B => IN4, X => G2OUT);
        G3:  NANDgate port map (A => IN5, B => IN6, X => G3OUT);
        G4:  NANDgate port map (A => IN7, B => IN8, X => G4OUT);
        G5:  NANDgate port map (A => G1OUT, B => G2OUT, X => G5OUT);
        G6:  NANDgate port map (A => G3OUT, B => G4OUT, X => G6OUT);
        G7:  NANDgate port map (A => G5OUT, B => G6OUT, X => OUT);
end architecture LogicFunction;
```

46. $X = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}B\overline{C} + A\overline{B}\,\overline{C} + AB\overline{C} + ABC$

This is the SOP expression for the function in Table 5-8 of the textbook. The following program applies the data flow approach for this logic function.

```
--Program for Table5_8 SOP logic
entity Table5_8 is
        port (A, B, C:  in bit; X: out bit);
end entity Table5_8;
architecture LogicOperation of Table5_8 is
begin
        X <= (not A and not B and not C) or (not A and B and not C)
                or (A and not B and not C) or (A and B and not C) or (A and B and C);
end architecture LogicOperation;
```

47.

```
--Program for textbook Figure 5-66 data flow approach
entity Fig5_66 is
        port (A, B, C, D, E:  in bit; X: out bit);
end entity Fig5_66;
architecture DataFlow of Fig5_66 is
begin
        X <= (A and B and C) or (D and not E)
end architecture DataFlow;
```

See Figure 5-62 for the circuit in textbook Figure 5-66 modified for the structural approach.

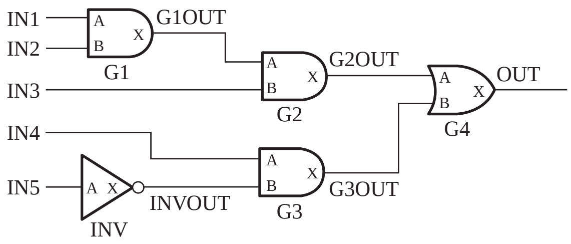


FIGURE 5-62

```
--Program for textbook Figure 5-66 structural approach
entity Fig5_66 is
        port (IN1, IN2, IN3, IN4, IN5:  in bit; OUT: out bit);
end entity Fig5_66;
architecture Structure of Fig5_66 is
--Component declaration for AND gate
component AND_gate is
        port (A, B: in bit; X: out bit);
end component AND_gate;
--Component declaration for OR gate
component OR_gate is
        port (A, B: in bit; X: out bit);
end component OR_gate;
--Component declaration for Inverter
component Inverter is
        port (A: in bit; X: out bit);
end component Inverter;
signal G1OUT, G2OUT, G3OUT, INVOUT: bit;
begin
        G1:  AND_gate port map (A => IN1, B => IN2, X => G1OUT);
        G2:  AND_gate port map (A => G1OUT, B => IN3, X => G2OUT);
        INV:  Inverter port map (A => IN5, X => INVOUT);
        G3:  AND_gate port map (A => IN4, B => INVOUT, X => G3OUT);
        G4:  OR_gate port map (A => G2OUT, B => G3OUT, X => OUT);
end architecture Structure;
```

48.

```
--Program for textbook Figure 5-70 data flow approach
entity Fig5_70 is
        port (A, B, C, D, E:  in bit; X: out bit);
end entity Fig5_70;
architecture DataFlow of Fig5_70 is
begin
        X <= (not A or not B or C) and E or (C or not D) and E;
end architecture DataFlow;
```

See Figure 5-63 for the circuit in textbook Figure 5-70 labeled for the structural approach.

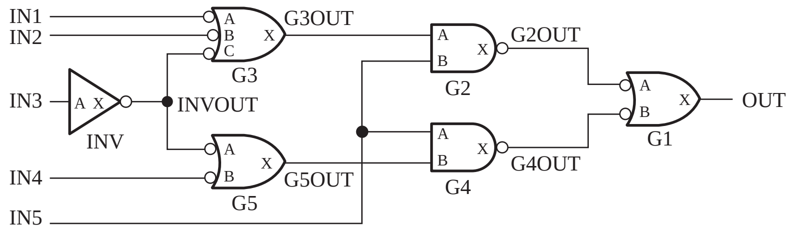


FIGURE 5-63

```
--Program for textbook Figure 5-70 structural approach
entity Fig5_70 is
        port (IN1, IN2, IN3, IN4, IN5:  in bit; OUT: out bit);
end entity Fig5_70;
architecture Structure of Fig5_70 is
--Component declaration for 3-input NAND gate
component NAND_gate3 is
        port (A, B, C: in bit; X: out bit);
end component NAND_gate3;
--Component declaration for 2-input NAND gate
component NAND_gate2 is
        port (A, B: in bit; X: out bit);
end component NAND_gate2;
--Component declaration for Inverter
component Inverter is
        port (A: in bit; X: out bit);
end component Inverter;
signal G2OUT, G3OUT, G4OUT, G5OUT, INVOUT: bit;
begin
        G1:  NAND_gate2 port map (A => G2OUT, B => G4OUT, X => OUT);
        G2:  NAND_gate2 port map (A => G3OUT, B => IN5, X => G2OUT);
        INV:  Inverter port map (A => IN3, X => INVOUT);
        G3:  NAND_gate3 port map (A => IN1, B => IN2, C => INVOUT, X => G3OUT);
        G4:  NAND_gate2 port map (A => IN5, B => G5OUT, X => G4OUT);
        G5:  NAND_gate2 port map (A => INVOUT, B => IN4, X => G5OUT);
end architecture Structure;
```

49. From the VHDL program, the logic expression is stated as a Boolean expression as follows:

$$\begin{aligned} X &= (\overline{\overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{A}\,\overline{D} + \overline{B}\,\overline{C} + \overline{B}\,\overline{D} + \overline{D}\,\overline{C}}) \\ &= ((A+B)(A+C)(A+D)(B+C)(B+D)(D+C)) \\ &= (A+B)(A+C)(A+D)(B+C)(B+D)(D+C) \end{aligned}$$

The truth table is:

A	*B*	*C*	*D*	*X*
0	0	0	0	0
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0
0	0	1	0	0
1	0	1	0	0
0	1	1	0	0
1	1	1	0	0
0	0	0	1	0
1	0	0	1	0
0	1	0	1	0
1	1	0	1	1
0	0	1	1	0
1	0	1	1	1
0	1	1	1	1
1	1	1	1	1

50.

```
--Program for textbook Figure 5-72 data flow approach
entity Fig5_72 is
        port (A1, A2, B1, B2:  in bit; X: out bit);
end entity Fig5_72;
architecture LogicCircuit of Fig5_72 is
begin
        X <= (A1 and A2) or (A2 and not B1) or (not B1 and not B2) or (not B2 and A1);
end architecture LogicCircuit;
```

51. The AND gates are numbered top to bottom G1, G2, G3, G4. The OR gate is G5 and the inverters are, top to bottom. G6 and G7. Change A_1, A_2, B_1, B_2 to IN1, IN2, IN3, IN4 respectively. Change X to OUT.

```
entity Circuit5_72 is
        port (IN1, IN2, IN3, (IN4: in bit; OUT: out bit);
end entity Circuit 5_72;
architecture Logic of Circuit 5_72 is
component AND_gate is
        port (A, B: in bit; X: out bit);
end component AND_gate;
component OR_gate is
        port (A, B, C, D: in bit; X: out bit);
end component OR_gate;
component Inverter is
        port (A: in bit; X: out bit);
end component Inverter;
        signal G1OUT, G2OUT, G3OUT, G4OUT, G5OUT, G6OUT, G7OUT: bit;
begin
        G1:  AND_gate port map (A => IN1, B => IN2, X => G1OUT);
        G2:  AND_gate port map (A => IN2, B => G6OUT, X => G2OUT);
        G3:  AND_gate port map (A => G6OUT, B => G7OUT, X => G3OUT);
        G4:  AND_gate port map (A => G7OUT, B => IN1, X => G4OUT);
        G5:  OR_gate port map (A => G1OUT, B => G2OUT, X => G3OUT,
               D => G4OUT, X => OUT);
        G6:  Inverter port map (A => IN3, X => G6OUT);
        G7:  Inverter port map (A => IN4, X => G7OUT);
end architecture Logic;
```

System Application Activity

52. $V_{\text{inlet}} = \overline{L}_{\text{min}} + \overline{L}_{\text{max}} F_{\text{inlet}}$

See Figure 5-64.

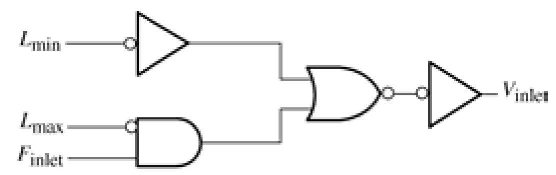


FIGURE 5-64

53. $V_{\text{outlet}} = L_{\text{min}}\bar{F}_{\text{inlet}}T$
See Figure 5-65.

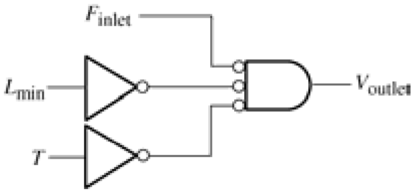


FIGURE 5-65

54. See Figure 5-66.

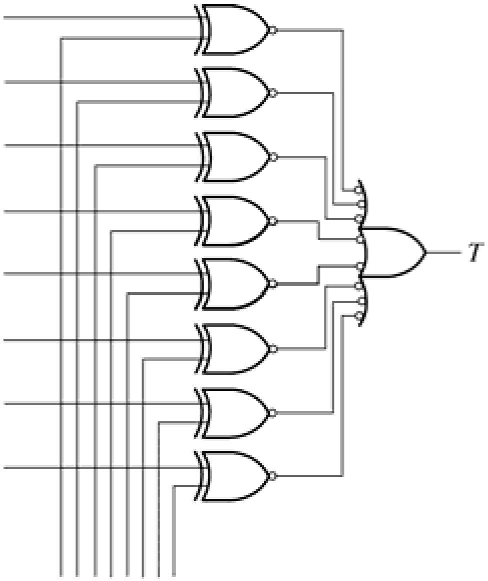


FIGURE 5-66

55. $V_{\text{additive}} = TL_{\text{min}}$
See Figure 5-67.

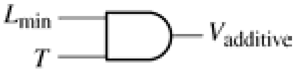


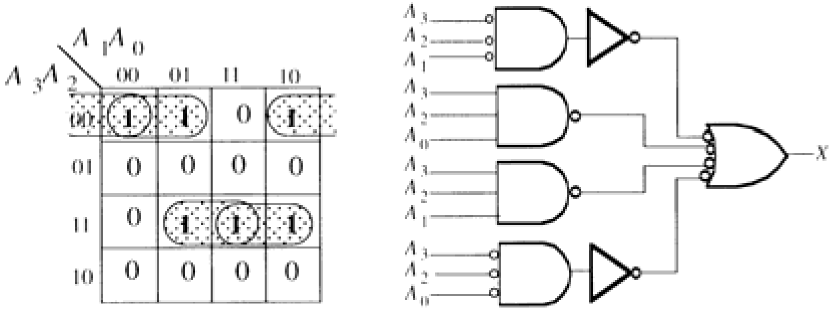
FIGURE 5-67

Special Design Problems

56.

A_3	A_2	A_1	A_0	X
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

See Figure 5-68.



A_3A_2 \ A_1A_0	00	01	11	10
00	1	1	0	1
01	0	0	0	0
11	0	1	1	1
10	0	0	0	0

$$X = \overline{A}_3\overline{A}_2\overline{A}_1 + A_3A_2A_0 + A_3A_2A_1 + \overline{A}_3\overline{A}_2\overline{A}_0$$

FIGURE 5-68

57. Let

X = Lamp on
A = Front door switch on
$\overline{A}$ = Front door switch off
B = Back door switch on
$\overline{B}$ = Back door switch off
$X = A\overline{B} + \overline{A}B$. This is an XOR operation.

See Figure 5-69.

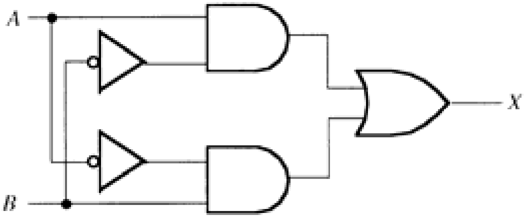


FIGURE 5-69

58. See Figure 5-70.

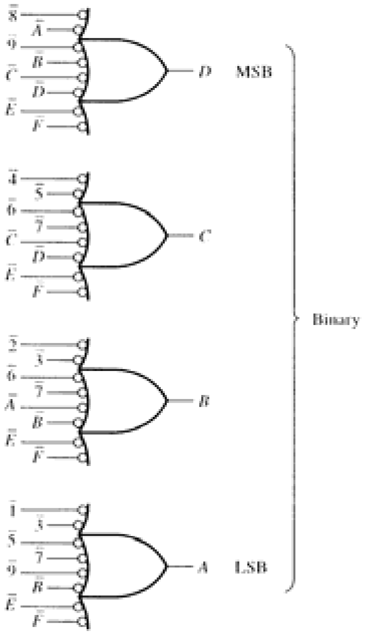


FIGURE 5-70

Chapter 5

Multisim Troubleshooting Practice

59. Pin B of G1 open.

60. Pin C of OR gate open.

61. Inverter input open.

62. No fault.

CHAPTER 6
FUNCTIONS OF COMBINATIONAL LOGIC

Section 6-1 Basic Adders

1. (a) XOR (upper) output = 0, Sum output = 1, AND (upper) output = 0, AND (lower) output = 1, Carry output = 1

(b) XOR (upper) output = 1, Sum output = 0, AND (upper) output = 1, AND (lower) output = 0, Carry output = 1

(c) XOR (upper) output = 1, Sum output = 1, AND (upper) output = 0, AND (lower) output = 0, Carry output = 0

2. (a) $A = 0$, $B = 0$, $C_{\text{in}} = 0$

(b) $A = 1$, $B = 0$, $C_{\text{in}} = 0$ or $A = 0$, $B = 1$, $C_{\text{in}} = 0$ or $A = 0$, $B = 0$, $C_{\text{in}} = 1$

(c) $A = 1$, $B = 1$, $C_{\text{in}} = 1$

(d) $A = 1$, $B = 1$, $C_{\text{in}} = 0$ or $A = 0$, $B = 1$, $C_{\text{in}} = 1$ or $A = 1$, $B = 0$, $C_{\text{in}} = 1$

3. (a) $\Sigma = 1$, $C_{\text{out}} = 0$ (b) $\Sigma = 1$, $C_{\text{out}} = 0$

(c) $\Sigma = 0$, $C_{\text{out}} = 1$ (d) $\Sigma = 1$, $C_{\text{out}} = 1$

Section 6-2 Parallel Binary Adders

4.

$$\begin{array}{r} 111\\ \underline{\ 101}\\ 1100 \end{array}$$

See Figure 6-1.

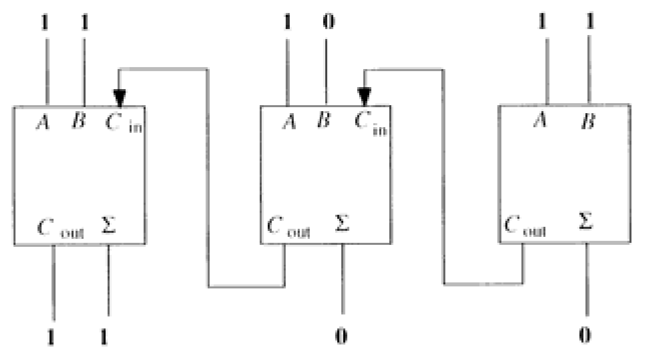


FIGURE 6-1

5. 10101 See Figure 6-2.
00111
11100

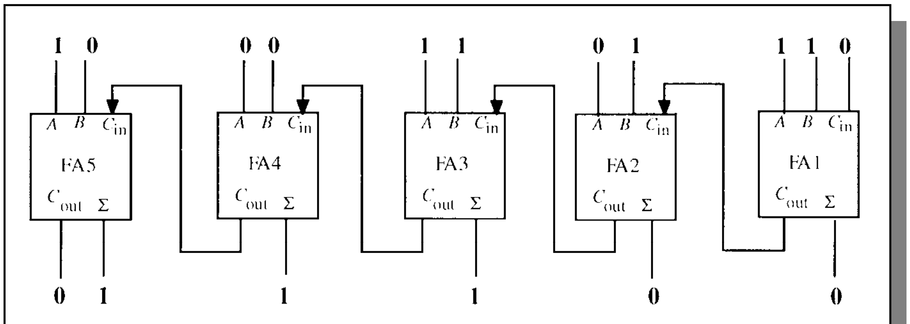


FIGURE 6-2

6. (a) When the $\overline{Add}/Subt$ is HIGH, the two numbers are subtracted.

(b) When the input is LOW, the numbers are added.

7. A = 1001 = –7, B = 1100 = –4

1001
0011 ← Complement of B
1 ← LSB Carry input
1101 = –3 in 2's comp

8. See Figure 6-3.

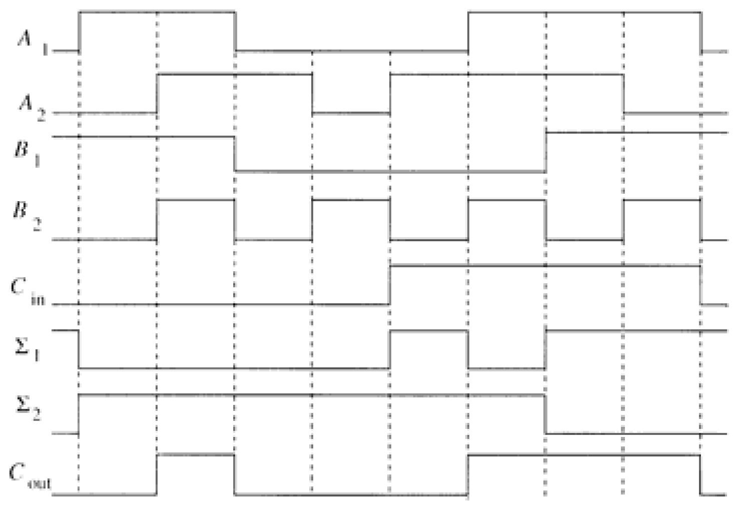


FIGURE 6-3

9.

A_4	A_3	A_2	A_1	B_4	B_3	B_2	B_1	Σ_5	Σ_4	Σ_3	Σ_2	Σ_1
1	0	0	1	0	0	0	1	0	1	0	1	0
1	0	1	0	1	1	0	1	0	0	1	1	1
0	0	1	0	0	0	1	1	0	0	1	0	1
1	0	1	1	0	1	1	1	1	0	0	1	0

Σ_1 = 0110
Σ_2 = 1011
Σ_3 = 0110
Σ_4 = 0001
Σ_5 = 1000

10.

$$\begin{array}{r} 0100 \\ \underline{1110} \\ 10010 \end{array}$$

Σ outputs should be $C_{out}\Sigma_4\Sigma_3\Sigma_2\Sigma_1$ = 10010.
The Σ_3 output (pin 2) is HIGH and should be LOW.

See Figure 6-4.

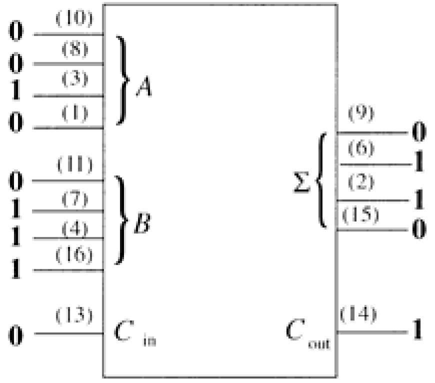


FIGURE 6-4

Chapter 6

Section 6-3 Ripple Carry Versus Look-Ahead Carry Adders

11. $t_{p(\text{tot})}$ = 40 ns + 6(25 ns) + 35 ns = **225 ns**

12. Full-adder 5:
$C_{\text{in}5} = C_{\text{out}}4$
$C_{\text{out}5} = C_{g5} + C_{p5}C_{g4} + C_{p5}C_{p4}C_{g3} + C_{p5}C_{p4}C_{p3}C_{g2} + C_{p5}C_{p4}C_{p3}C_{g2}C_{g1} + C_{p5}C_{p4}C_{p3}C_{p2}C_{p1}C_{\text{in}1}$

The logic to be added to text Figure 6-18 is shown in Figure 6-5.

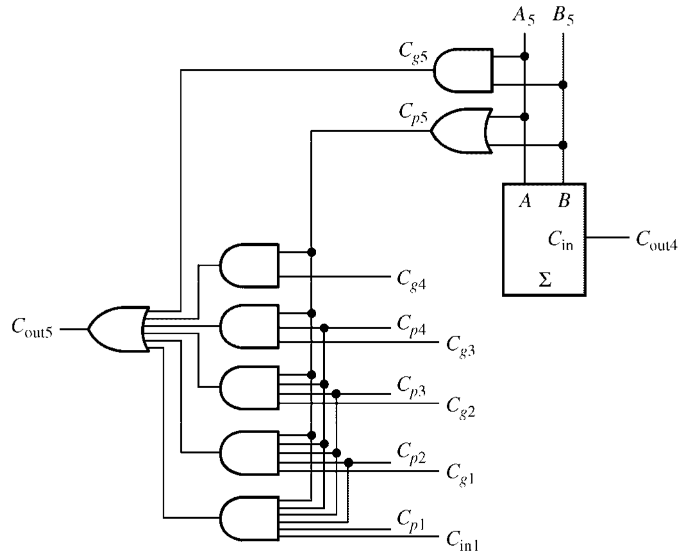


FIGURE 6-5

Section 6-4 Comparators

13. The $A = B$ output is HIGH when $A_0 = B_0$ and $A_1 = B_1$.

See Figure 6-6.

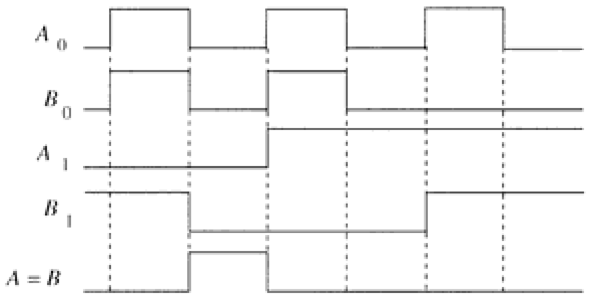


FIGURE 6-6

14. See Figure 6-7.

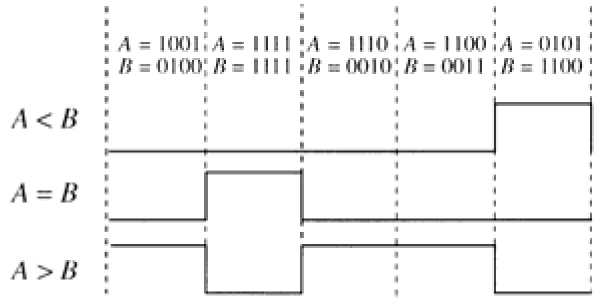


FIGURE 6-7

15. (a) $A > B$: 1, $A = B$: 0, $A < B$: 0
(b) $A > B$: 0, $A = B$: 0, $A < B$: 1
(c) $A > B$: 0, $A = B$: 1, $A < B$: 0

Section 6-5 Decoders

16. (a) $A_3A_2A_1A_0$ = **1110** (b) $A_3A_2A_1A_0$ = **1100**
(c) $A_3A_2A_1A_0$ = **1111** (d) $A_3A_2A_1A_0$ = **1000**

17. See Figure 6-8.

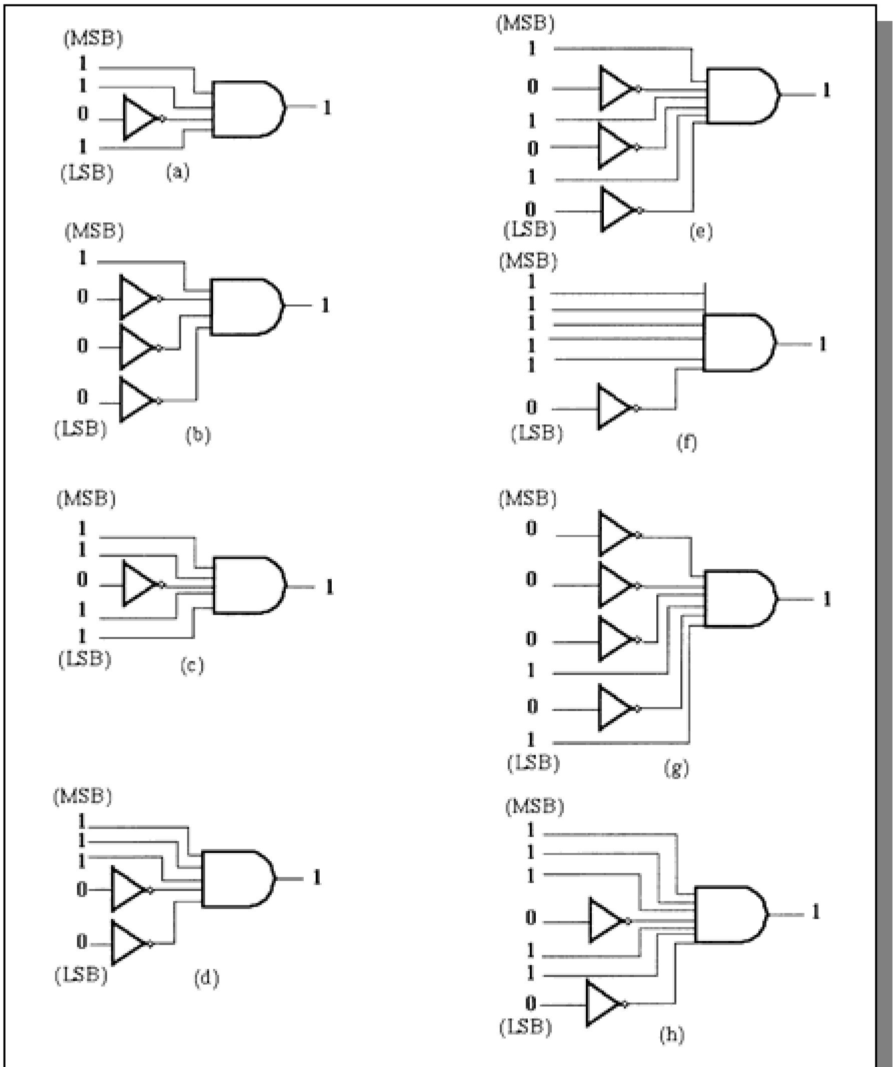
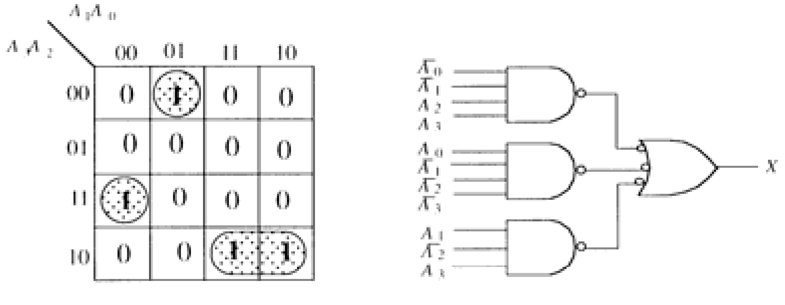


FIGURE 6-8

18. Change the AND gates to NAND gates in Figure 6-8.

19. $X = \overline{A}_3\overline{A}_2\overline{A}_1A_0 + A_3\overline{A}_2A_1\overline{A}_0 + A_3A_2\overline{A}_1\overline{A}_0 + A_3\overline{A}_2A_1A_0$

See Figure 6-9.



A_3A_2 \ A_1A_0	00	01	11	10
00	0	1	0	0
01	0	0	0	0
11	1	0	0	0
10	0	0	1	1

$X = \overline{A}_3\overline{A}_2\overline{A}_1A_0 + A_3A_2\overline{A}_1\overline{A}_0 + A_3\overline{A}_2A_1$

FIGURE 6-9

20. $Y = A_2A_1\overline{A_0} + A_2\overline{A_1}A_0 + \overline{A_2}A_1\overline{A_0}$

See Figure 6-10.

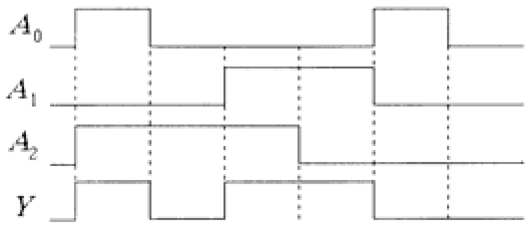


FIGURE 6-10

21. See Figure 6-11.

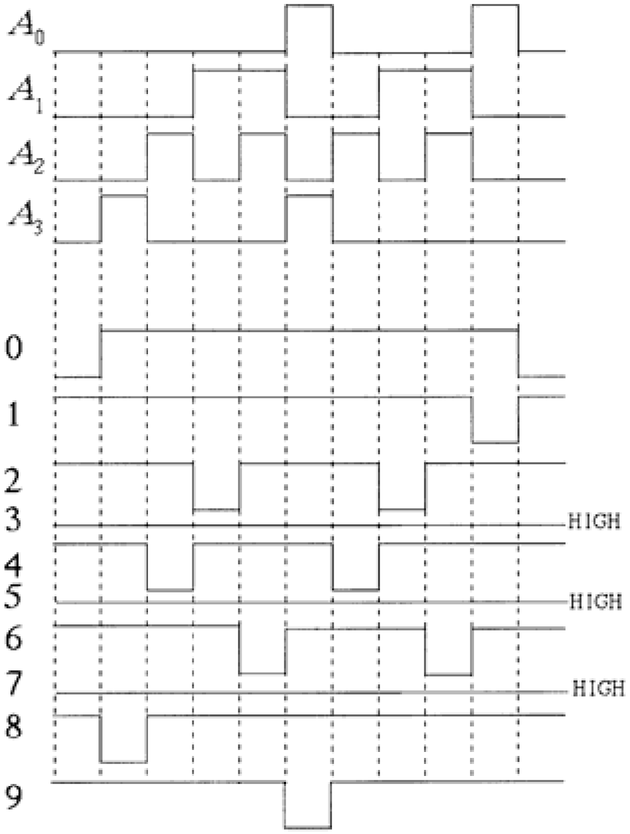


FIGURE 6-11

22. **0 1 6 9 4 4 4 8 0**

Section 6-6 Encoders

23. A_0, A_1, and A_3 are HIGH. $A_3A_2A_1A_0$ = 1011, which is an invalid BCD code.

24. Pin 2 is for decimal 5, pin 5 is for decimal 8, and pin 12 is for decimal 2. The highest priority input is pin 5.

The completed outputs are: $\overline{A_3}\,\overline{A_2}\,\overline{A_1}\,\overline{A_0}$ = 0111, which is binary 8 (1000).

Section 6-7 Code Converters

25. (a) 2_{10} = $\mathbf{0010}_{BCD}$ = $\mathbf{0010}_2$

(b) 8_{10} = $\mathbf{1000}_{BCD}$ = $\mathbf{1000}_2$

(c) 13_{10} = $\mathbf{00010011}_{BCD}$ = $\mathbf{1101}_2$

(d) 26_{10} = $\mathbf{00100110}_{BCD}$ = $\mathbf{11010}_2$

(e) 33_{10} = $\mathbf{00110011}_{BCD}$ = $\mathbf{100001}_2$

26.	(a)	1010101010	binary	(b)	1111100000	binary
		1111111111	gray		1000010000	gray
	(c)	0000001110	binary	(d)	1111111111	binary
		0000001001	gray		1000000000	gray

See Figure 6-12.

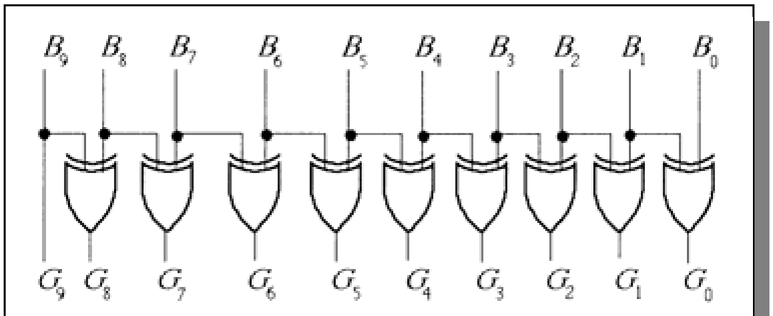


FIGURE 6-12

27.	(a)	1010000000	gray	(b)	0011001100	gray
		1100000000	binary		0010001000	binary
	(c)	1111000111	gray	(d)	0000000001	gray
		1010000101	binary		0000000001	binary

See Figure 6-13.

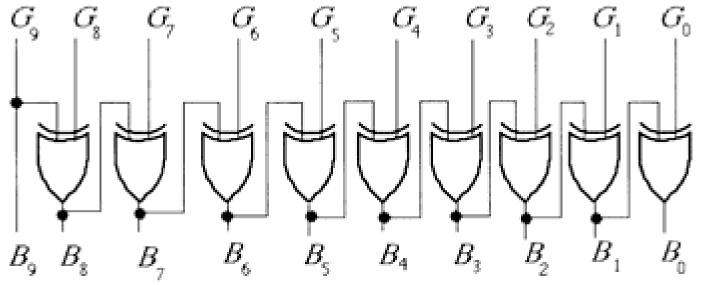


FIGURE 6-13

Section 6-8 Multiplexers (Data Selectors)

28. S_1S_0 = 01 selects, D_1, therefore Y = 1.

29. See Figure 6-14.

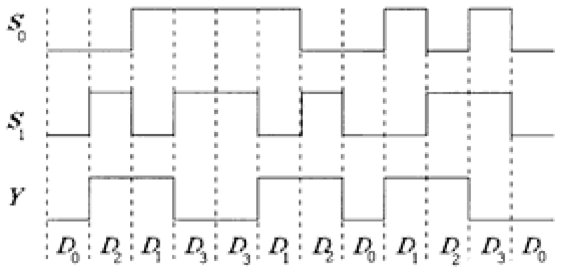


FIGURE 6-14

30. See Figure 6-15.

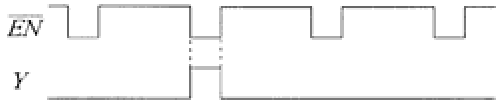


FIGURE 6-15

Section 6-9 Demultiplexers

31. See Figure 6-16.

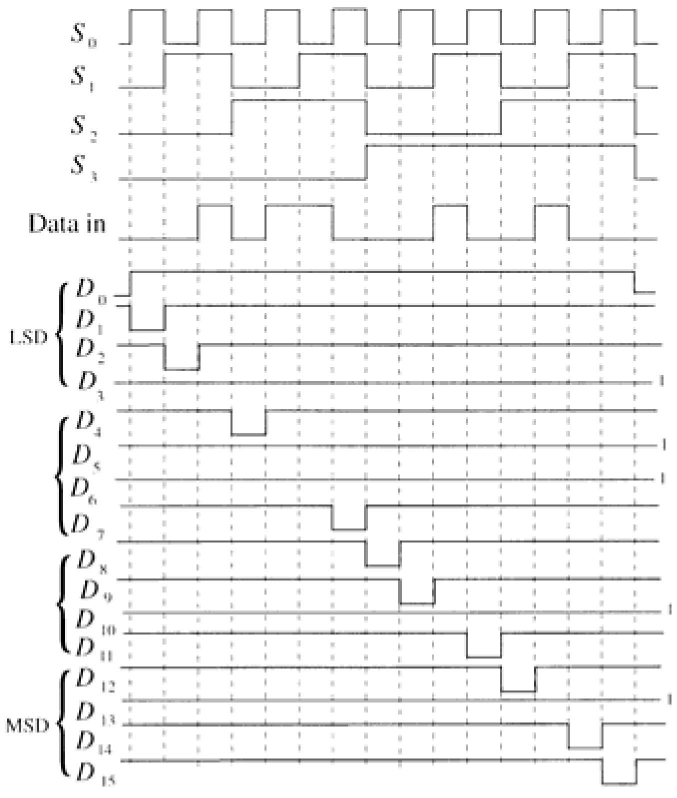
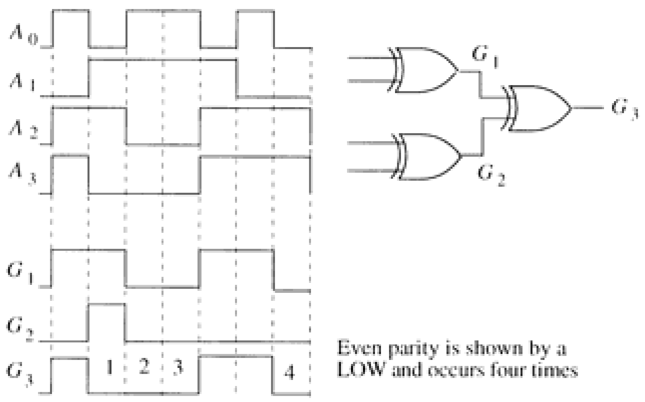


FIGURE 6-16

Section 6-10 Parity Generators/Checkers

32. See Figure 6-17.



Even parity is shown by a LOW and occurs four times

FIGURE 6-17

33. See Figure 6-18.

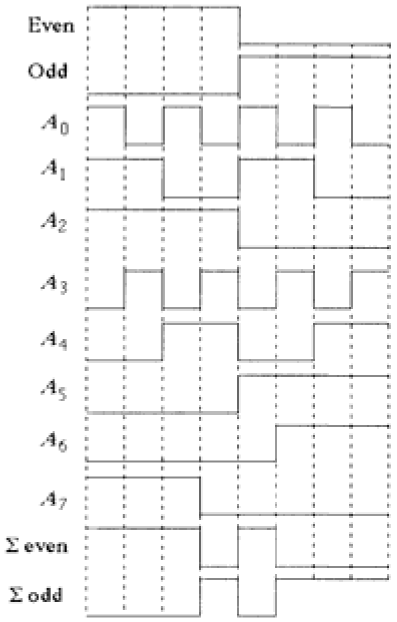


FIGURE 6-18

Section 6-11 Troubleshooting

34. The outputs given in the problem are incorrect. By observation of these incorrect waveforms, we can conclude that the outputs of the device are not open or shorted because both waveforms are changing.

Observe that at the beginning of the timing diagram all inputs are 0 but the sum is 1. This indicates that an input is stuck HIGH. Start by assuming that C_{in} is stuck HIGH. This results in Σ and C_{out} output waveforms that match the waveforms given in the problem, indicating that C_{in} is indeed stuck HIGH, perhaps shorted to V_{CC}.

See Figure 6-19 for the correct output waveforms.

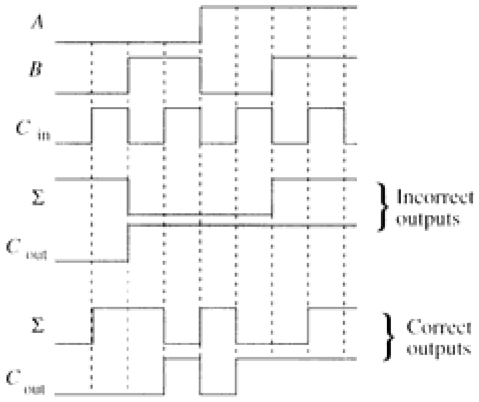


FIGURE 6-19

35. (a) OK (b) Segment *g* burned out; output G open (c) Segment *b* output stuck LOW

36. *Step 1:* Verify that the supply voltage is applied.
Step 2: Go through the key sequence and verify the output code in Table 1.

Key	A_3	A_2	A_1	A_0
None	1	1	1	1
0	1	1	1	1
1	1	1	1	0
2	1	1	0	1
3	1	1	0	0
4	1	0	1	1
5	1	0	1	0
6	1	0	0	1
7	1	0	0	0
8	0	1	1	1
9	0	1	1	0

TABLE 1

Step 3: Check for proper priority operation by repeating the key sequence in Table 1 except that for each key closure, hold that key down and depress each lower-valued key as specified in Table 2.

Hold down keys	Depress keys one at a time	A_3	A_2	A_1	A_0
1	0	1	1	1	0
2	1, 0	1	1	0	1
3	2, 1, 0	1	1	0	0
4	3, 2, 1, 0	1	0	1	1
5	4, 3, 2, 1, 0	1	0	1	0
6	5, 4, 3, 2, 1, 0	1	0	0	1
7	6, 5, 4, 3, 2, 1, 0	1	0	0	0
8	7, 6, 5, 4, 3, 2, 1, 0	0	1	1	1
9	8, 7, 6, 5, 4, 3, 2, 1, 0	0	1	1	0

TABLE 2

37. (a) Open A_1 input acts as a HIGH. All binary values corresponding to a BCD number having a 1's value of 0, 1, 4, 5, 8, or 9 will be off by 2. This will first be seen for a BCD value of 00000000.
(b) Open C_{out} of top adder. All values not normally involving a carry out will be off by 32. This will first be seen for a BCD value of 00000000.
(c) The Σ_4 output of top adder is shorted to ground. Same binary values above 15 will be short by 16. The first BCD value to indicate this will be 00011000.
(d) Σ_3 of bottom adder is shorted to ground. Every other set of 16 value starting with 16 will be short 16. The first BCD value to indicate this will be 00010110.

38. (a) The 1Y1 output of the 74LS139 is *stuck HIGH* or *open*; B cathode open.
(b) No power; EN input to the 74LS139 is *open*.
(c) The *f* output of the 74LS47 is *stuck HIGH*.
(d) The frequency of the data select input is too *low*.

39. 1. Place a LOW on pin 7 (Enable).
2. Apply a HIGH to D_0 and a LOW to D_1 through D_7.
3. Go through the binary sequence on the select inputs and check Y and $\overline{Y}$ according to Table 3.

S_2	S_1	S_0	Y	$\overline{Y}$
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

TABLE 3

4. Repeat the binary sequence of select inputs for each set of data inputs listed in Table 4. A HIGH on the *Y* output should occur only for the corresponding combinations of select inputs shown.

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y	$\overline{Y}$	S_2	S_1	S_0
L	H	L	L	L	L	L	L	1	0	0	0	1
L	L	H	L	L	L	L	L	1	0	0	1	0
L	L	L	H	L	L	L	L	1	0	0	1	1
L	L	L	L	H	L	L	L	1	0	1	0	0
L	L	L	L	L	H	L	L	1	0	1	0	1
L	L	L	L	L	L	H	L	1	0	1	1	0
L	L	L	L	L	L	L	H	1	0	1	1	1

TABLE 4

40. The Σ EVEN output of the 74LS280 should be HIGH and the output of the error gate should be HIGH because of the error condition. Possible faults are:

1. Σ EVEN output of the 74LS280 *stuck LOW.*
2. Error gate faulty.
3. The ODD input to the 74LS280 is *open* thus acting as a HIGH.
4. The inverter going to the ODD input of the 74LS280 has an *open* output or the output is *stuck HIGH.*

41. Apply a HIGH in turn to each Data input, D_0 through D_7 with LOWs on all the other inputs. For each HIGH applied to a data input, sequence through all eight binary combinations of select inputs ($S_2S_1S_0$) and check for a HIGH on the corresponding data output and LOWs on all the other data outputs.

One possible approach to implementation is to decode the $S_2S_1S_0$ inputs and generate an inhibit pulse during any given bit time as determined by the settings of seven switches. The inhibit pulse effectively changes a LOW on the *Y* serial data line to a HIGH during the selected bit time(s), thus producing a bit error. A basic diagram of this approach is shown in Figure 6-20.

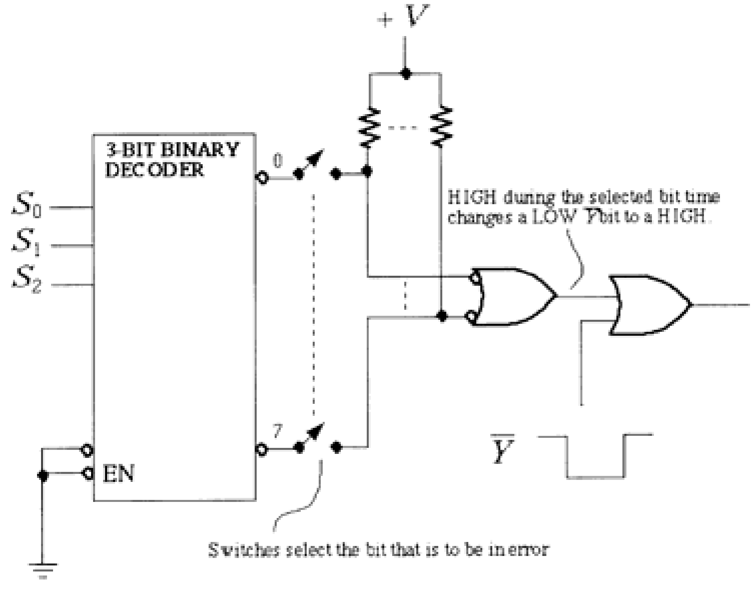

+ V
3-BIT BINARY DECODER
S_0
S_1
S_2
0
7
EN
HIGH during the selected bit time changes a LOW $\overline{Y}$ bit to a HIGH.
$\overline{Y}$
Switches select the bit that is to be in error

FIGURE 6-20

System Application Activity

42. See Figure 6-21.

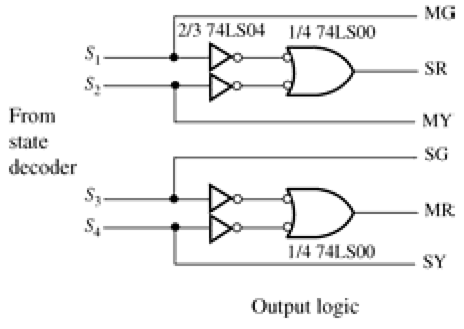


FIGURE 6-21

43. See Figure 6-22.

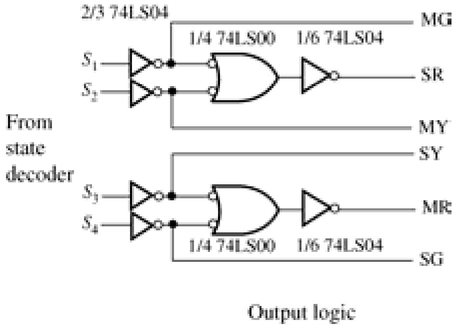


FIGURE 6-22

Special Design Problems

44. See Figure 6-23.

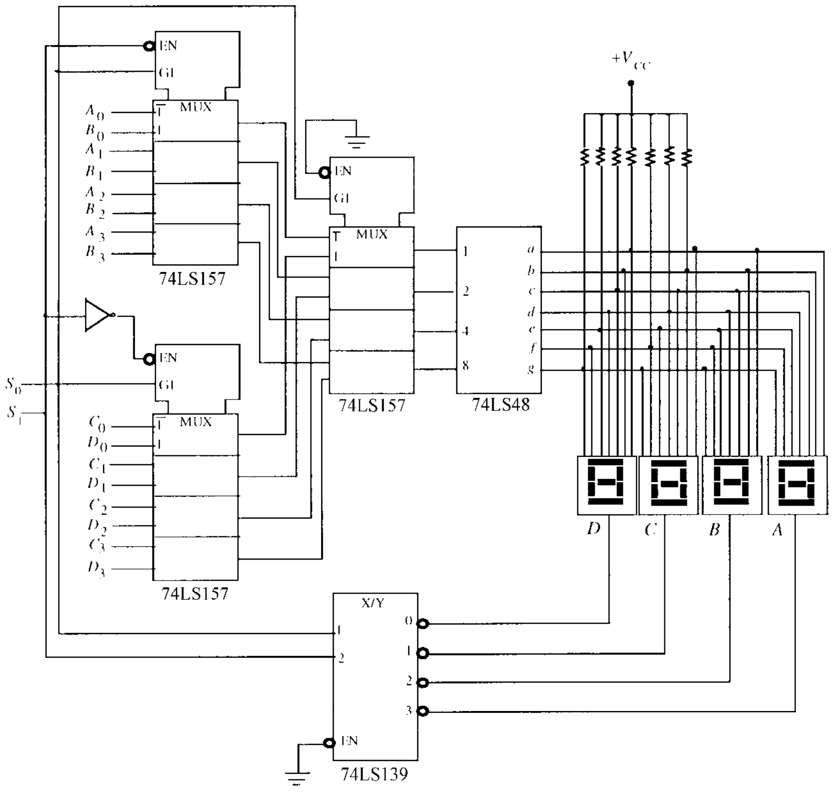
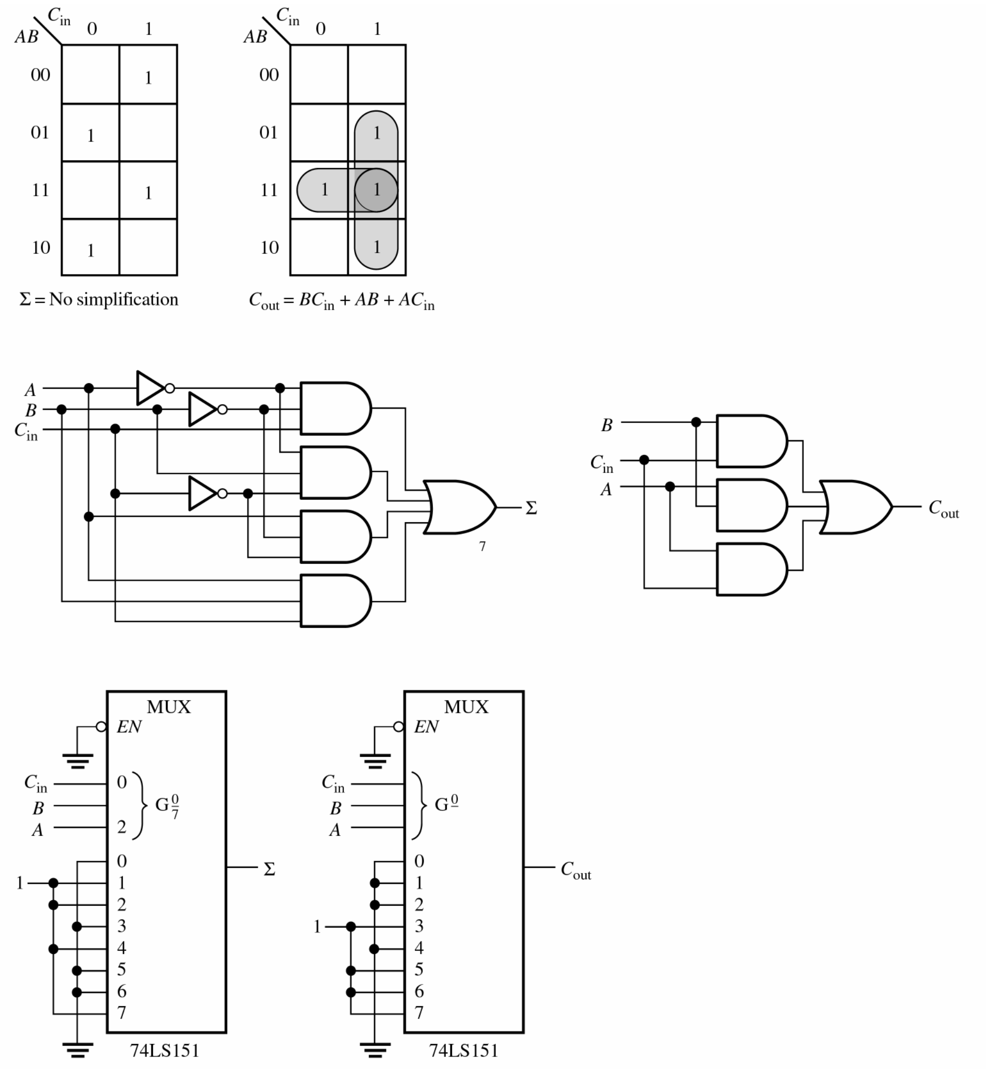


FIGURE 6-23

45. $\Sigma = \overline{A}\overline{B}C_{\text{in}} + \overline{A}B\overline{C}_{\text{in}} + A\overline{B}\overline{C}_{\text{in}} + ABC_{\text{in}}$

$C_{\text{out}} = ABC_{\text{in}} + \overline{A}BC_{\text{in}} + A\overline{B}C_{\text{in}} + AB\overline{C}_{\text{in}}$

See Figure 6-24.



Σ = No simplification

$C_{\text{out}} = BC_{\text{in}} + AB + AC_{\text{in}}$

FIGURE 6-24

46. $$Y = \overline{A_3}\,\overline{A_2}A_1\overline{A_0} + \overline{A_3}\,\overline{A_2}A_1A_0 + \overline{A_3}A_2A_1\overline{A_0} + \overline{A_3}A_2A_1A_0 + A_3\overline{A_2}\,\overline{A_1}\,\overline{A_0}$$
$$+\ A_3\overline{A_2}A_1\overline{A_0} + A_3\overline{A_2}A_1A_0 + A_3A_2\overline{A_1}A_0 + A_3A_2A_1A_0$$

See Figure 6-25.

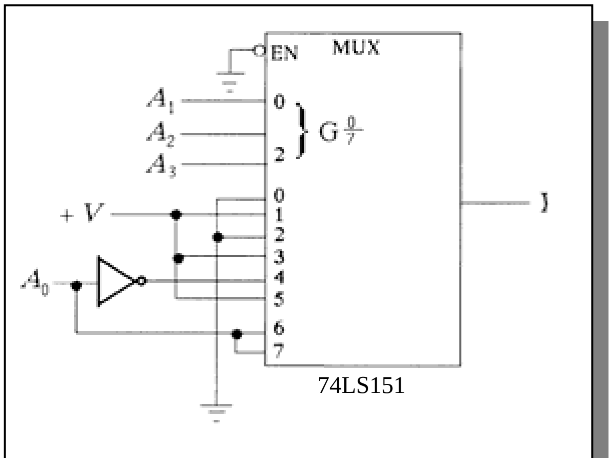


FIGURE 6-25

47. See Figure 6-26.

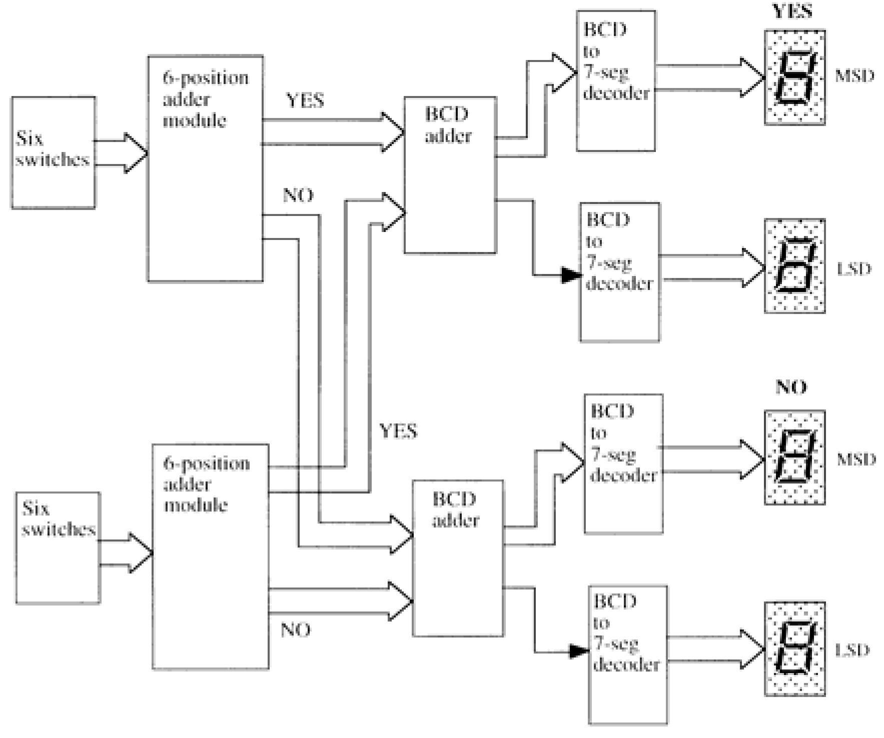


FIGURE 6-26

48. See Figure 6-27.

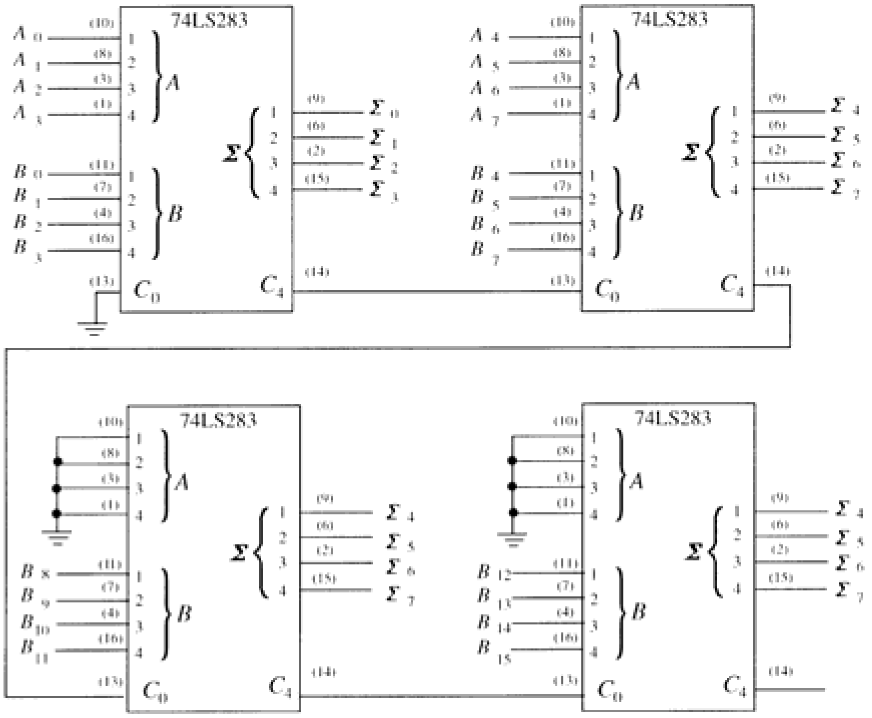


FIGURE 6-27

49. See Figure 6-28.

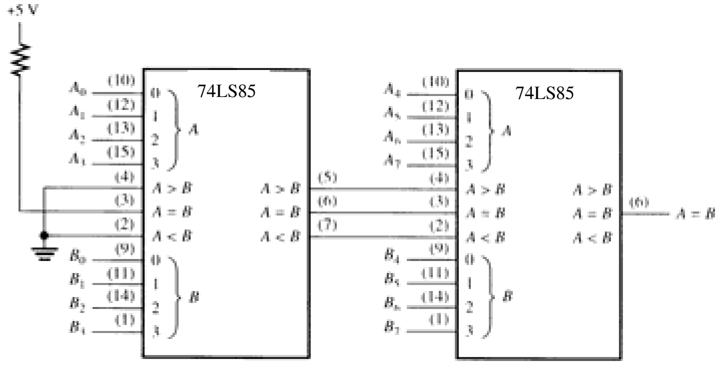
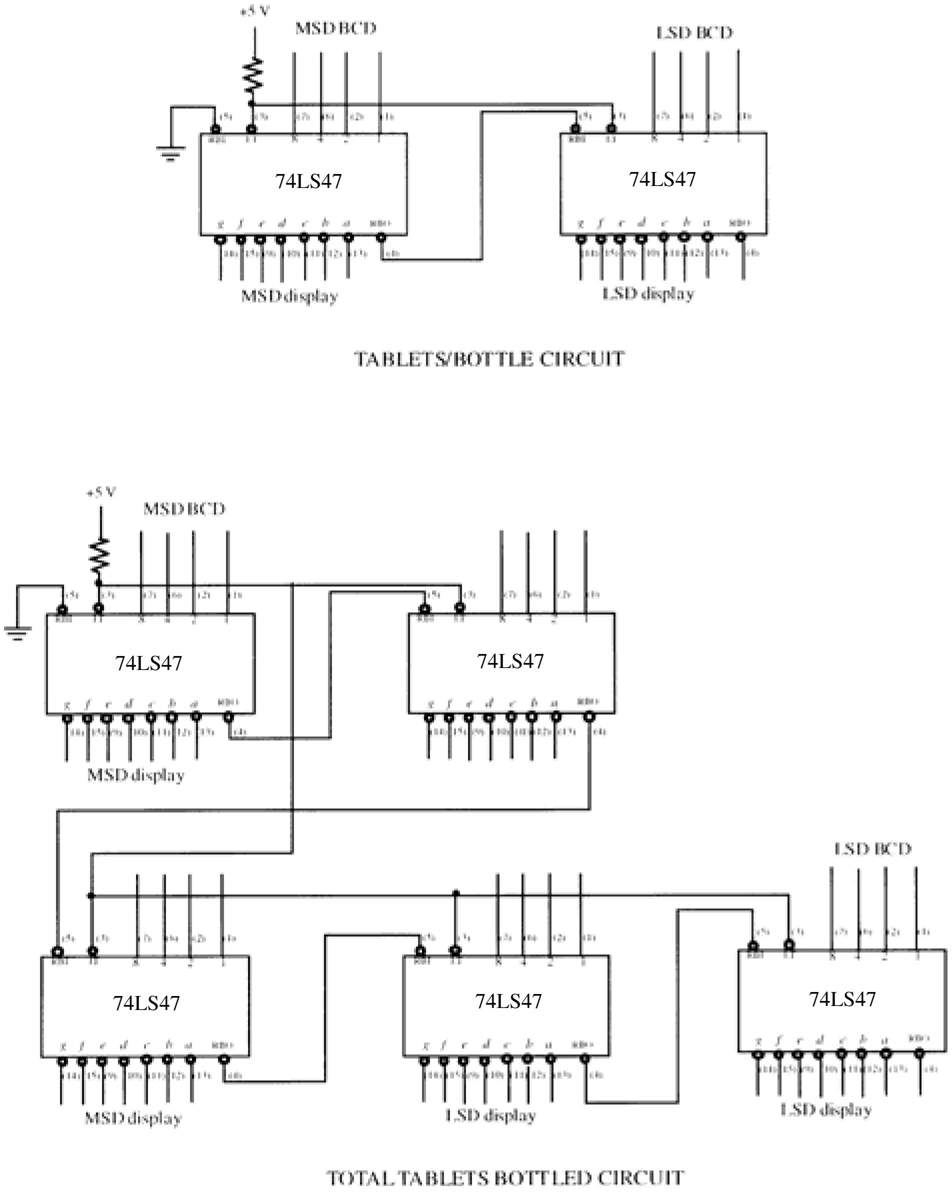


FIGURE 6-28

50. See Figure 6-29.

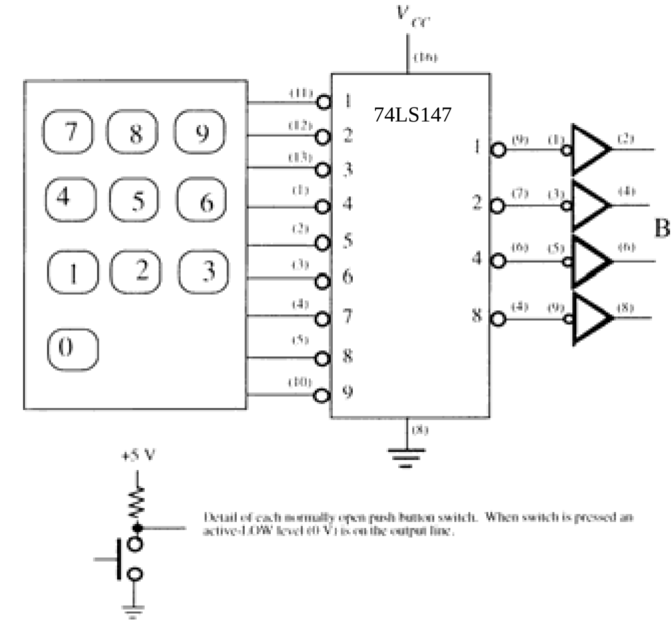


TABLETS/BOTTLE CIRCUIT

TOTAL TABLETS BOTTLED CIRCUIT

FIGURE 6-29

51. See Figure 6-30.



Detail of each normally open push button switch. When switch is pressed an active-LOW level (0 V) is on the output line.

FIGURE 6-30

52. See Figure 6-31.

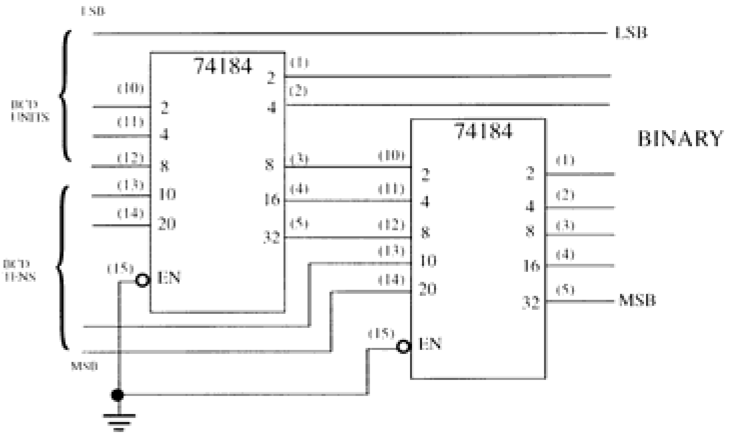


FIGURE 6-31

Multisim Troubleshooting Practice

53. LSB adder carry output open.

54. Pins 4 and 5 shorted together.

55. Pin 12 of upper 74148 open.

56. Pin 3 of upper 74151 open.

CHAPTER 7
LATCHES, FLIP-FLOPS, and TIMERS

Section 7-1 Latches

1. See Figure 7-1.

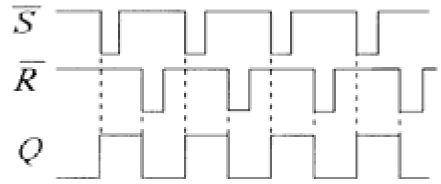


FIGURE 7-1

2. See Figure 7-2.

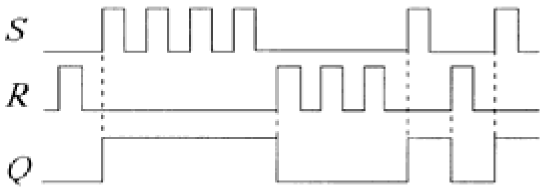


FIGURE 7-2

3. See Figure 7-3.

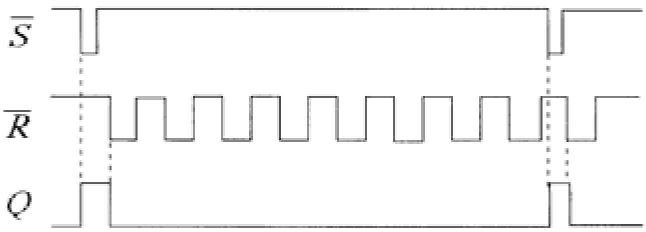


FIGURE 7-3

4. See Figure 7-4.

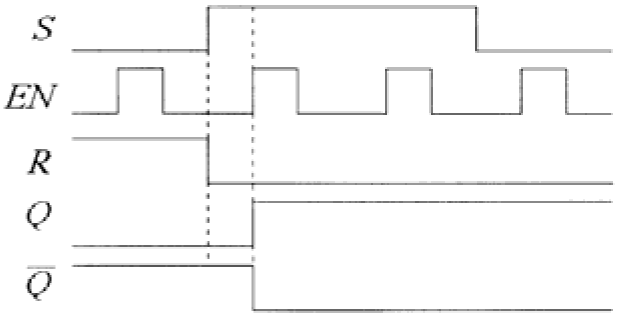


FIGURE 7-4

5. See Figure 7-5.

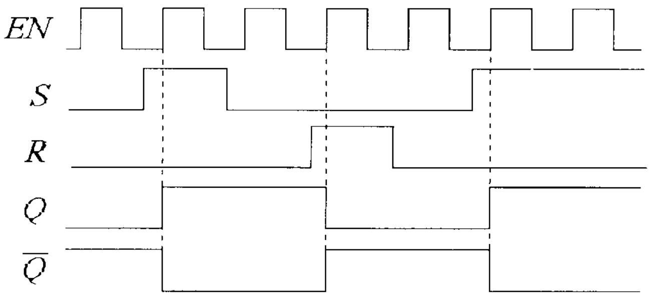


FIGURE 7-5

6. See Figure 7-6.

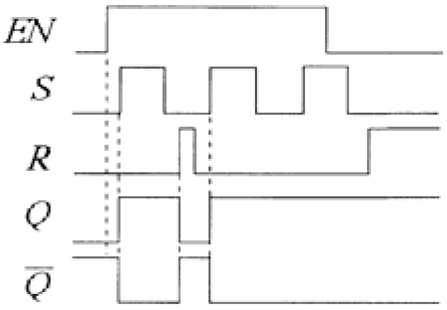


FIGURE 7-6

7. See Figure 7-7.

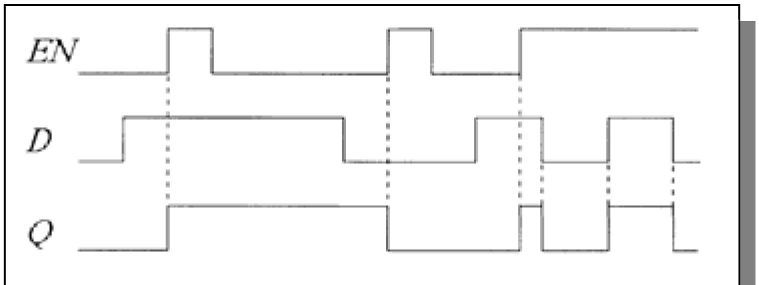
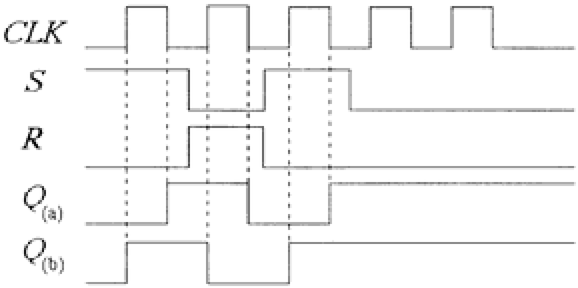


FIGURE 7-7

Section 7-2 Edge-Triggered Flip-Flops

8. See Figure 7-8.



(a) The flip-flop triggers on the negative edge of the clock pulse.

(b) The flip-flop triggers on the positive edge of the clock pulse

FIGURE 7-8

9. See Figure 7-9.

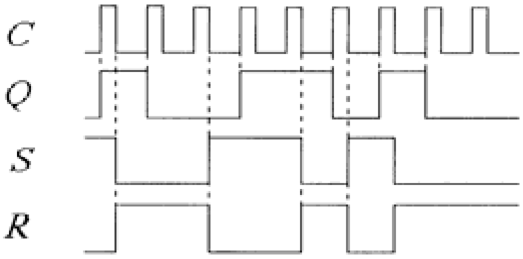


FIGURE 7-9

10. See Figure 7-10.

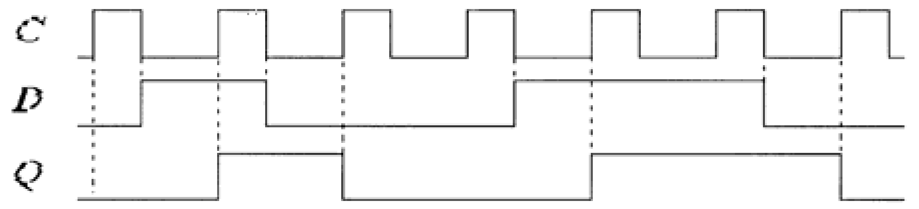


FIGURE 7-10

11. See Figure 7-11.

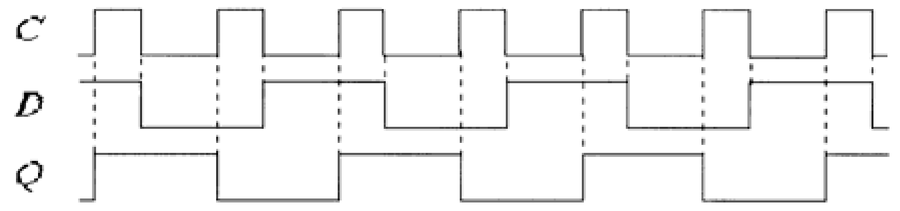


FIGURE 7-11

12. See Figure 7-12.

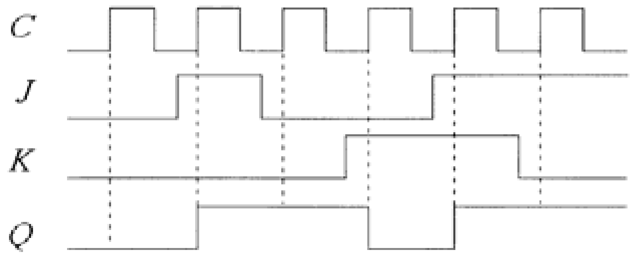


FIGURE 7-12

13. See Figure 7-13.

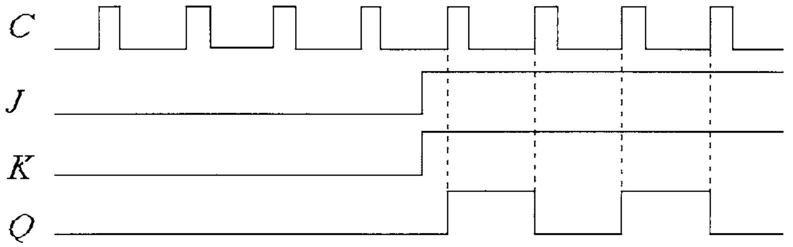


FIGURE 7-13

14. See Figure 7-14.

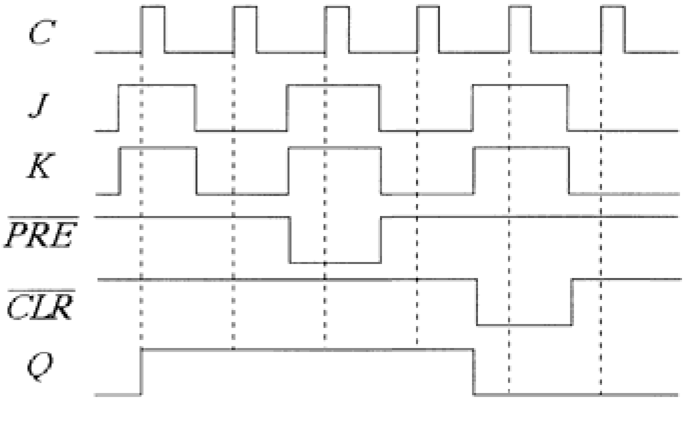


FIGURE 7-14

15. See Figure 7-15.

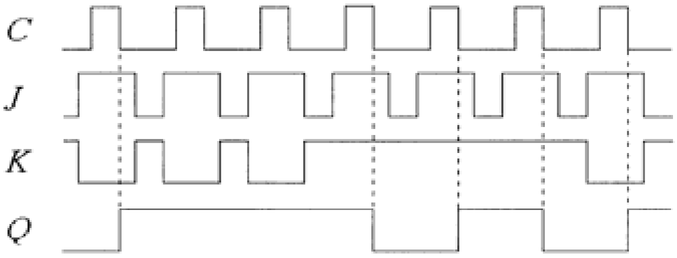


FIGURE 7-15

16. *J*: 0010000
K: 0000100
Q: 0011000

17. See Figure 7-16.

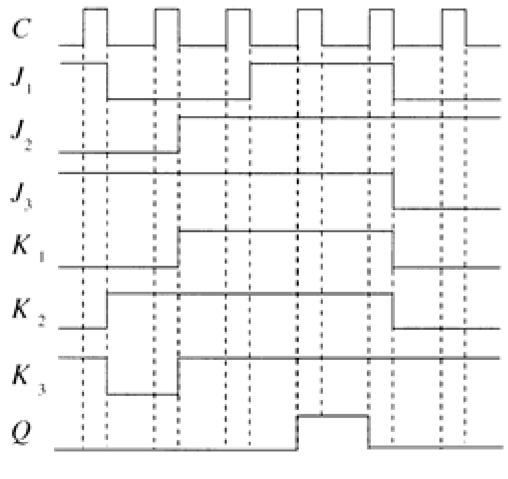


FIGURE 7-16

18. See Figure 7-17.

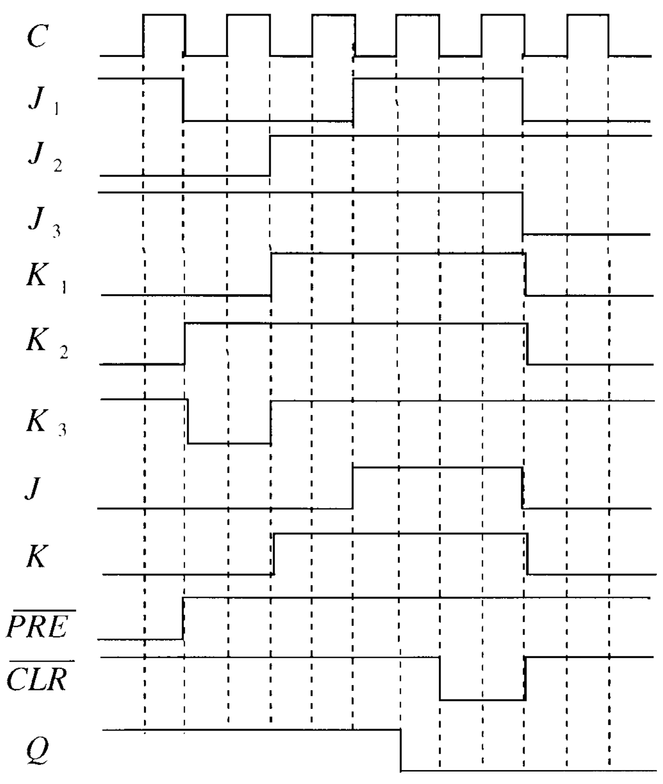


FIGURE 7-17

Section 7-3 Flip-Flop Operating Characteristics

19. The direct current and dc supply voltage

20. t_{PLH} (Clock to Q):
Time from triggering edge of clock to the LOW-to-HIGH transition of the Q output.
t_{PHL} (Clock to Q):
Time from triggering edge of clock to the HIGH-to-LOW transition of the Q output.
t_{PLH} ($\overline{\text{PRE}}$ to Q):
Time from assertion of the Preset input to the LOW-to-HIGH transition of the Q output.
t_{PHL} ($\overline{\text{CLR}}$ to Q):
Time from assertion of the clear input to the HIGH-to-LOW transition of the Q output.

21. T_{min} = 30 ns + 37 ns = 67 ns

$$f_{max} = \frac{1}{T_{min}} = \mathbf{14.9\ MHz}$$

22. See Figure 7-18.

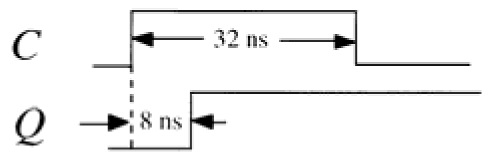
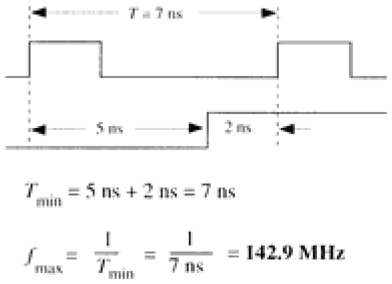


FIGURE 7-18



$T_{min} = 5\text{ ns} + 2\text{ ns} = 7\text{ ns}$

$f_{max} = \frac{1}{T_{min}} = \frac{1}{7\text{ ns}} = \mathbf{142.9\ MHz}$

FIGURE 7-19

23. $I_T = 15(10\text{ mA}) = \mathbf{150\ mA}$
$P_T = (5\text{ V})(150\text{ mA}) = \mathbf{750\ mW}$

24. See Figure 7-19.

Section 7-4 Flip-Flop Applications

25. See Figure 7-20.

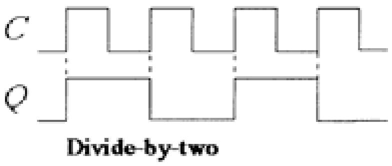


FIGURE 7-20

26. See Figure 7-21.

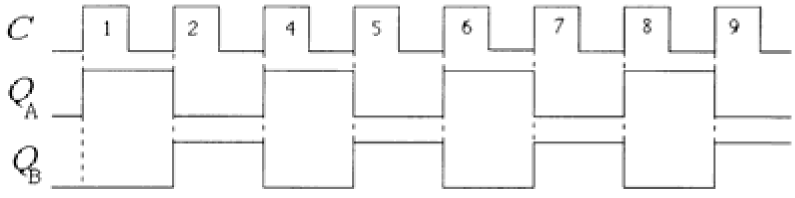


FIGURE 7-21

Section 7-5 One-Shots

27. $t_W = 0.7RC_{EXT} = 0.7(3.3\text{ k}\Omega)(2000\text{ pF}) = \mathbf{4.62\ \mu s}$

28. $R_X = \dfrac{t_W}{RC_{\text{EXT}}} - 0.7 = \dfrac{5000\text{ ns}}{0.32 \times 10{,}000\text{ pF}} - 0.7 = \mathbf{1.56\ k\Omega}$

29. See Figure 7-22.

$t_W = 0.25\text{ s} = 1.1R_1C_1$

Choose $C_1 = 1\ \mu F$

$R_1 = \dfrac{t_W}{1.1C_1} = \dfrac{0.25\text{ s}}{(1.1)(1\ \mu\text{F})} = 227\text{ k}\Omega$ (use standard 220 kΩ)

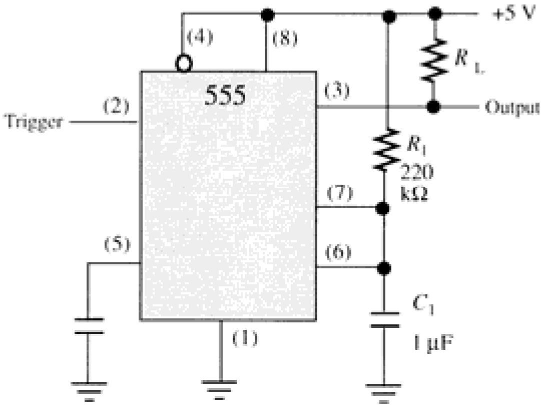


FIGURE 7-22

Section 7-6 Astable Multivibrator

30. $f = \dfrac{1}{0.7(R_1 + 2R_2)C_2} = \dfrac{1}{0.7(1000\,\Omega + 2200\,\Omega)(0.01\,\mu\text{F})} = \mathbf{44.6\ kHz}$

31. $T = \dfrac{1}{f} = \dfrac{1}{20\text{ kHz}} = 50\ \mu\text{s}$

For a duty cycle of 75%:

$t_H = 37.5\ \mu\text{s}$ and $t_L = 12.5\ \mu\text{s}$

$R_1 + R_2 = \dfrac{t_H}{0.7C} = \dfrac{37.5\ \mu\text{s}}{0.7(0.002\ \mu\text{F})} = 26{,}786\ \Omega$

$R_2 = \dfrac{t_L}{0.7C} = \dfrac{12.5\ \mu\text{s}}{0.7(0.002\ \mu\text{F})} = \mathbf{8{,}929\ \Omega}$ (use 9.1 kΩ)

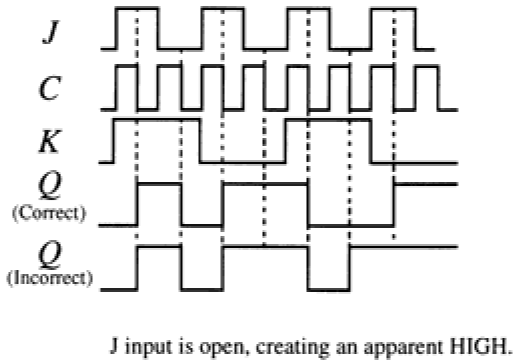
$R_1 = 26{,}786\ \Omega - R_2 = 26{,}786\ \Omega - 8{,}929\ \Omega = \mathbf{17{,}857\ \Omega}$ (use 18 kΩ)

Section 7-7 Troubleshooting

32. The flip-flop in Figure 7-94 of the text has an internally open *J* input.

33. The wire from pin 6 to pin 10 and the ground wire are reversed. Pin 7 should be at ground and pin 6 connected to pin 10.

34. See Figure 7-23.



J input is open, creating an apparent HIGH.

FIGURE 7-23

35. Since none of the flip-flops change, the problem must be a fault that affects all of them. The two functions common to all the flip-flops are the clock (CLK) and clear $(\overline{\text{CLR}})$ inputs. One of these lines must be shorted to ground because a LOW on either one will prevent the flip-flops from changing state. Most likely, the $\overline{\text{CLR}}$ line is shorted to ground because if the clock line were shorted chances are that all of the flip-flops would not have ended up reset when the power was turned on unless an initial LOW was applied to the $\overline{\text{CLR}}$ at power on.

36. Small differences in the switching times of flip-flop A and flip-flop B due to propagation delay cause the glitches as shown in the expanded timing diagram in Figure 7-24. The delays are exaggerated greatly for purposes of illustration. Glitches are eliminated by strobing the output with the clock pulse.

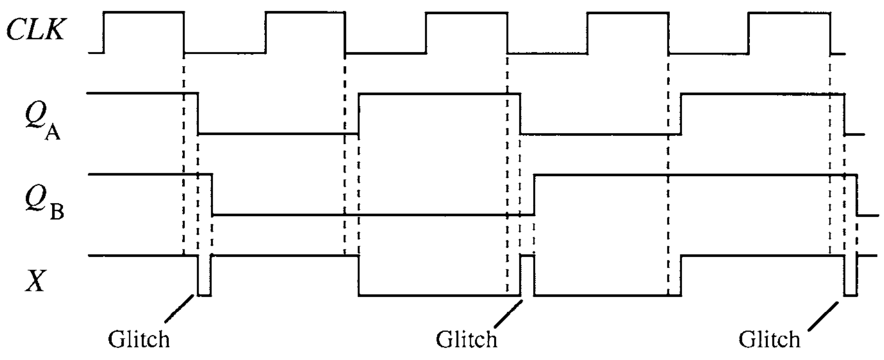


FIGURE 7-24

37. (a) See Figure 7-25.

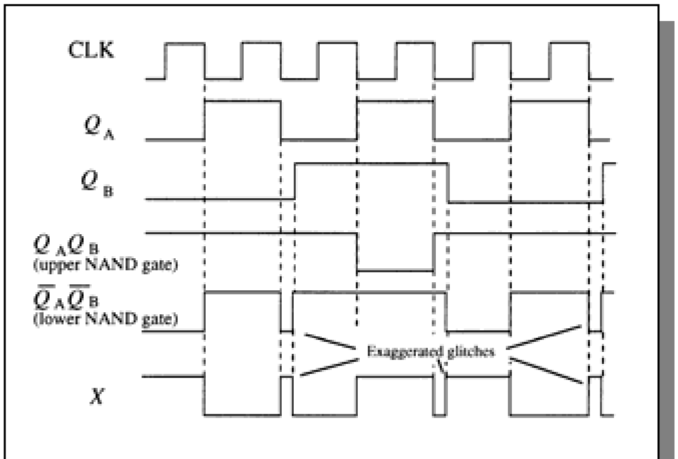


FIGURE 7-25

(b) K_B open acts as a HIGH and the operation is normal. The timing diagram is the same as Figure 7-25.

(c) See Figure 7-26.

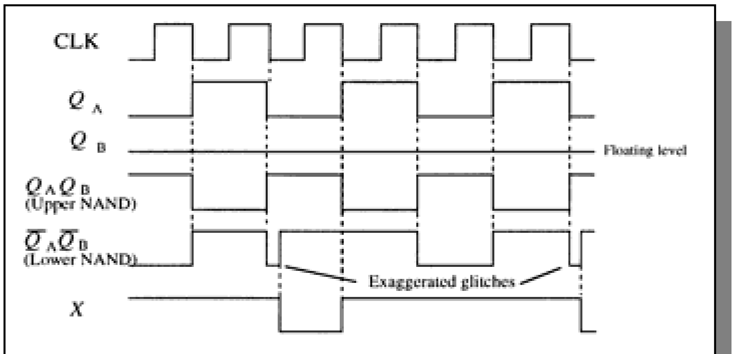


FIGURE 7-26

(d) X remains LOW if $Q_B = 1$ $(\overline{Q_B} = 0)$. X follows $\overline{Q_A}$ if $Q_B = 0$ $(\overline{Q_B} = 1)$.

(e) See Figure 7-27.

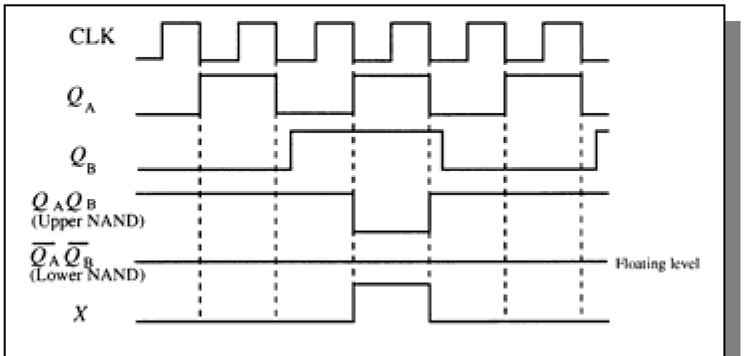


FIGURE 7-27

38. $t_W = 0.7RC_{\text{EXT}}$

One-shot A: $t_W = 0.7(0.22\ \mu\text{F})(100\ \text{k}\Omega) = 15.4\ \text{ms}$

One-shot B: $t_W = 0.7(0.1\ \mu\text{F})(100\ \text{k}\Omega) = 7\ \text{ms}$

The pulse width of one shot A is apparently not controlled by the external components and the one-shot is producing its minimum pulse width of about 40 ns. An *open pin 11* would cause this problem. See Figure 7-28.

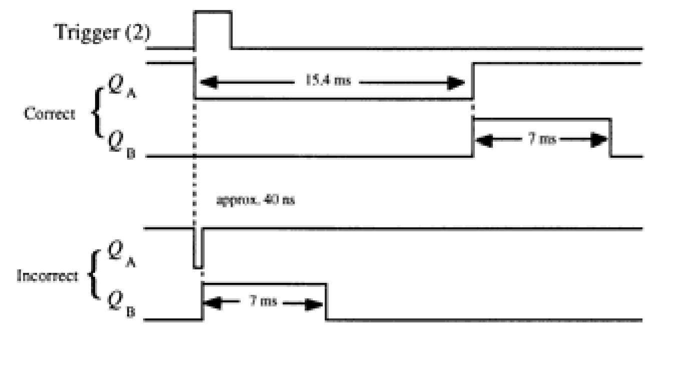


FIGURE 7-28

System Application Activity

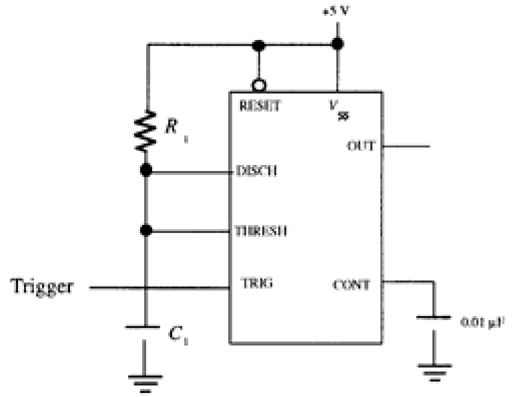
39. For the 6 s timer let $C_1 = 1\ \mu\text{F}$

$$R_1 = \frac{6\ \text{s}}{(1.1)(1\ \mu\text{F})} = 5.5\ \text{M}\Omega\ \text{(use 5.6 M}\Omega\text{)}$$

For the 40 s timer let $C_1 = 2.2\ \mu\text{F}$

$$R_1 = \frac{40\ \text{s}}{(1.1)(2.2\ \mu\text{F})} = 16.5\ \text{M}\Omega\ \ \text{(use 15 M}\Omega\text{)}$$

See Figure 7-29.



6 s timer and 40 s timer are the same except for the component values calculated above.

FIGURE 7-29

40. t_W = 6 s. Let C_{EXT} = 1 µF.
$t_{\text{W}} = 0.7RC_{\text{EXT}}$

$$R = \frac{t_W}{0.7C_{\text{EXT}}} = \frac{6\text{ s}}{0.7(1\ \mu\text{F})} = 8.6\ \text{M}\Omega$$

t_W = 40 s. Let C_{EXT} = 10 µF.

$$R = \frac{t_W}{0.7C_{\text{EXT}}} = \frac{40\text{ s}}{0.7(10\ \mu\text{F})} = \ = 5.7\ \text{M}\Omega$$

See Figure 7-30.

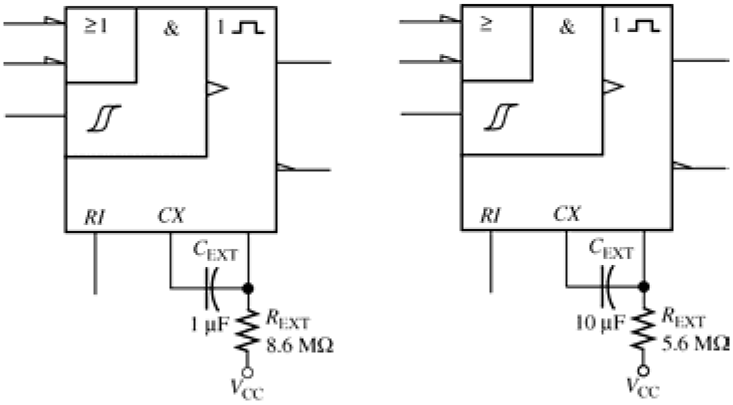


FIGURE 7-30

41. t_W = 6 s. Let C_{EXT} = 1 µF.

$$t_W = 0.32R_{\text{EXT}}C_{\text{EXT}}\left(1+\frac{0.7}{R_{\text{EXT}}}\right) = 0.32R_{\text{EXT}}C_{\text{EXT}} + (0.7)(0.32)C_{\text{EXT}}$$

$$R_{\text{EXT}} = \frac{t_W - (0.7)(0.32)C_{\text{EXT}}}{0.32C_{\text{EXT}}} = \frac{6\text{ s} - (0.224)(1\ \mu\text{F})}{0.32\ (1\ \mu\text{F})} = 18.8\ \text{M}\Omega$$

t_W = 40 s. Let C_{EXT} = 10 µF.

$$R_{\text{EXT}} = \frac{40\text{ s} - (0.224)10\ \mu\text{F}}{0.32(10\ \mu\text{F})} = \ = 12.5\ \text{M}\Omega$$

See Figure 7-31.

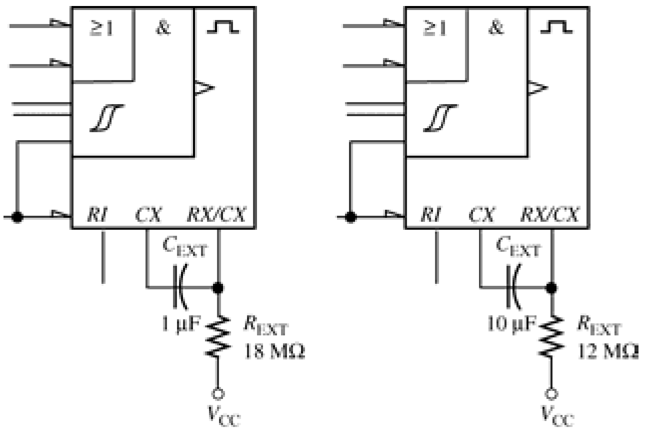


FIGURE 7-31

Special Design Problems

42. See Figure 7-32.

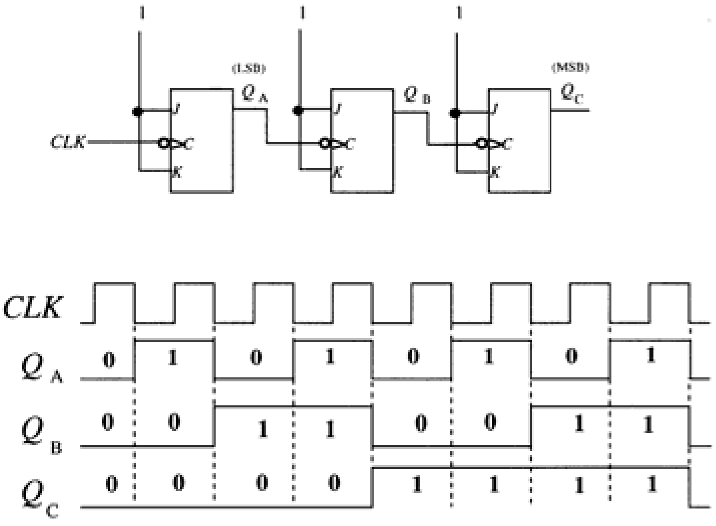


FIGURE 7-32

43. See Figure 7-33 for one possibility.

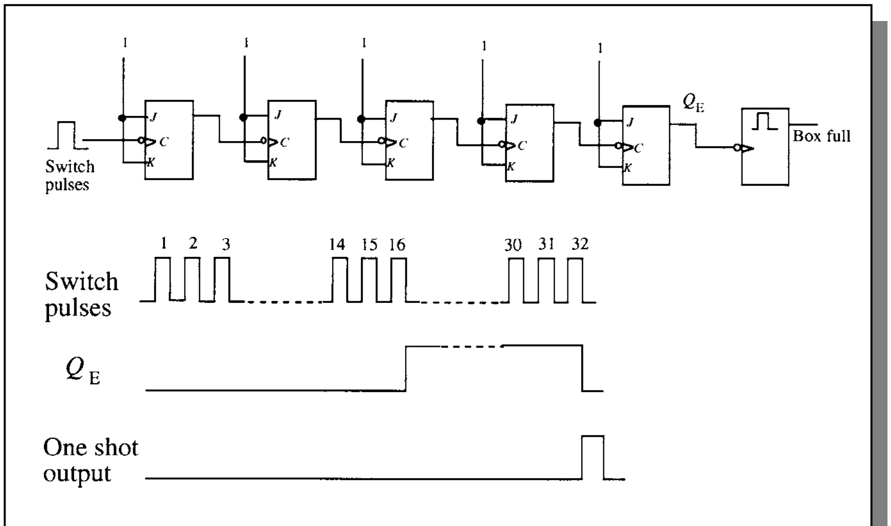


FIGURE 7-33

44. Changes required for the system to incorporate a 15 s left turn signal on main:

1. Change the 2-bit gray code sequence to a 3-bit sequence.
2. Add decoding logic to the State Decoder to decode the turn signal state.
3. Change the Output Logic to incorporate the turn signal output.
4. Change the Trigger Logic to incorporate a trigger output for the turn signal timer.
5. Add a 15 second timer.

See Figure 7-34.

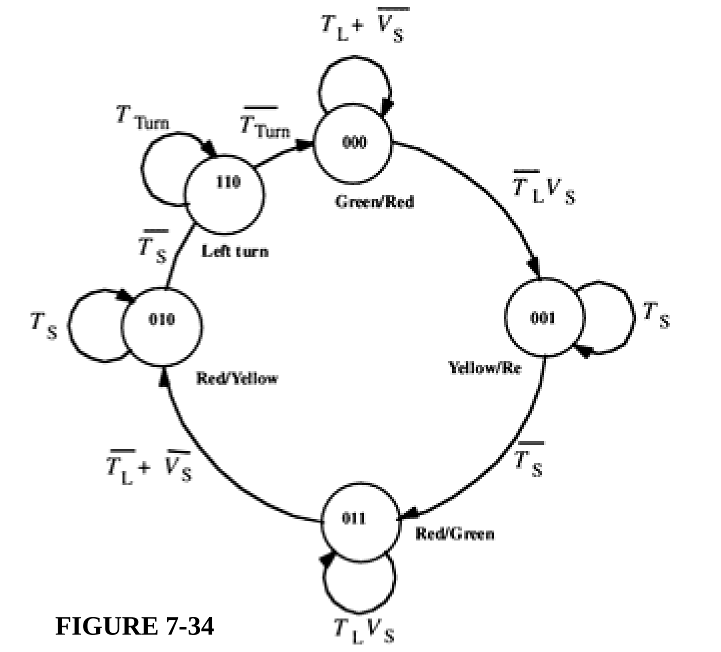


FIGURE 7-34

Multisim Troubleshooting Practice

45. $\overline{Q}$ output of U1 open.

46. *K* input of U2 open.

47. $\overline{SET}$ input of U1 open.

48. No fault.

49. *K* input of U2 open.

CHAPTER 8
COUNTERS

Section 8-1 Asynchronous Counters

1. See Figure 8-1.

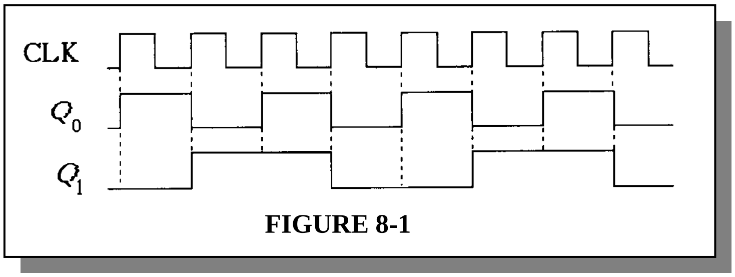


FIGURE 8-1

2. See Figure 8-2.

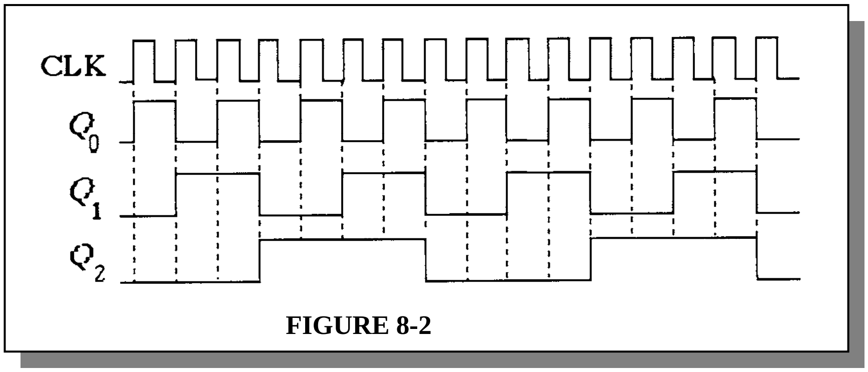


FIGURE 8-2

3. $t_{p(max)} = 3(8\text{ ns}) =$ **24 ns**

Worst-case delay occurs when all flip-flops change state from 011 to 100 or from 111 to 000.

4. See Figure 8-3.

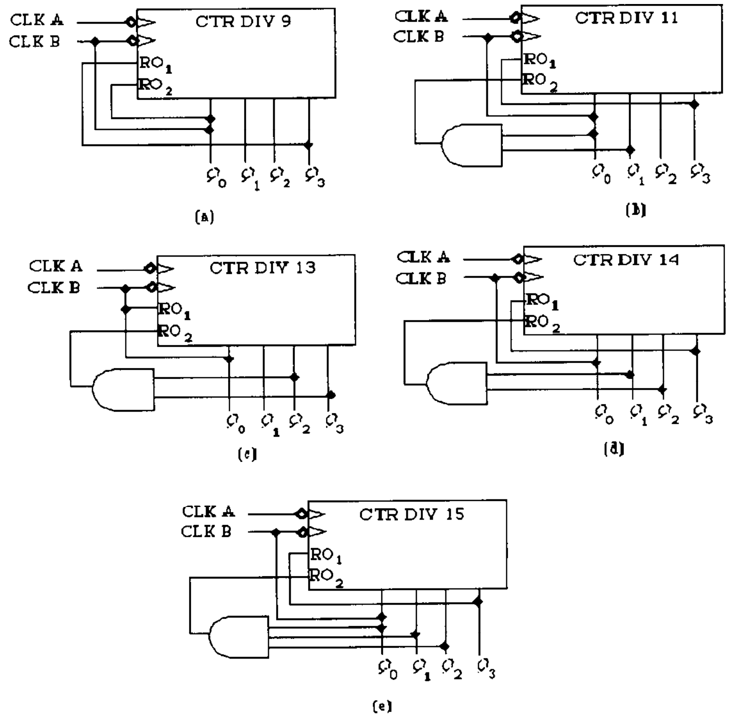


FIGURE 8-3

Section 8-2 Synchronous Counters

5. **8 ns**, the time it takes one flip-flop to change state.

6. See Figure 8-4.

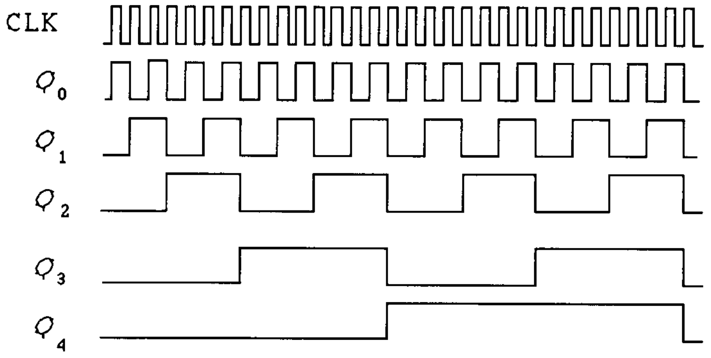


FIGURE 8-4

7. Each flip-flop is initially reset.

CLK	J_0K_0	J_1K_1	J_2K_2	J_3K_3	Q_0	Q_1	Q_2	Q_3
1	1	0	0	0	1	0	0	0
2	1	1	0	0	0	1	0	0
3	1	0	0	0	1	1	0	0
4	1	1	1	0	0	0	1	0
5	1	0	0	0	1	0	1	0
6	1	1	0	0	0	1	1	0
7	1	0	0	0	1	1	1	0
8	1	1	1	1	0	0	0	1
9	1	0	0	0	1	0	0	1
10	1	0	0	1	0	0	0	0

8. See Figure 8-5.

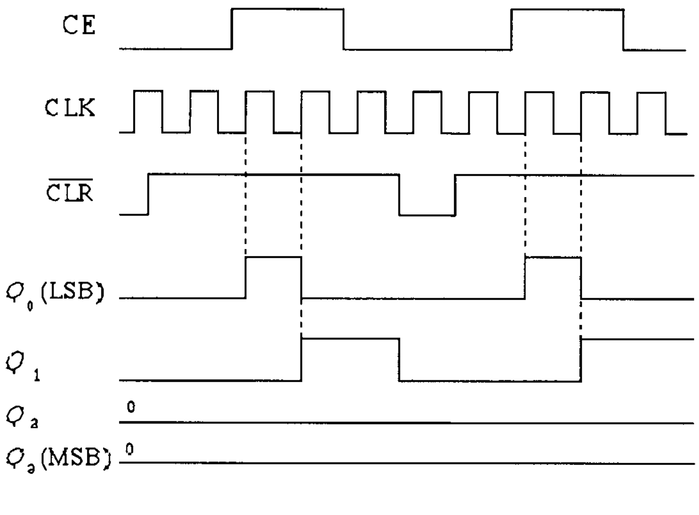


FIGURE 8-5

9. See Figure 8-6.

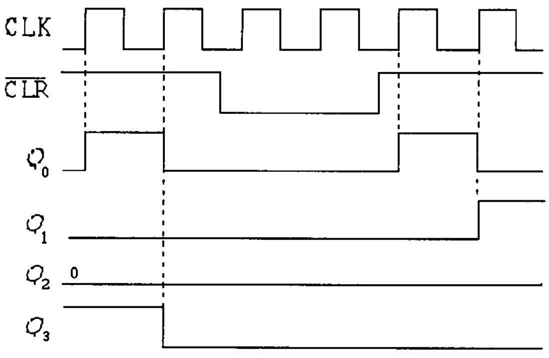


FIGURE 8-6

10. See Figure 8-7.

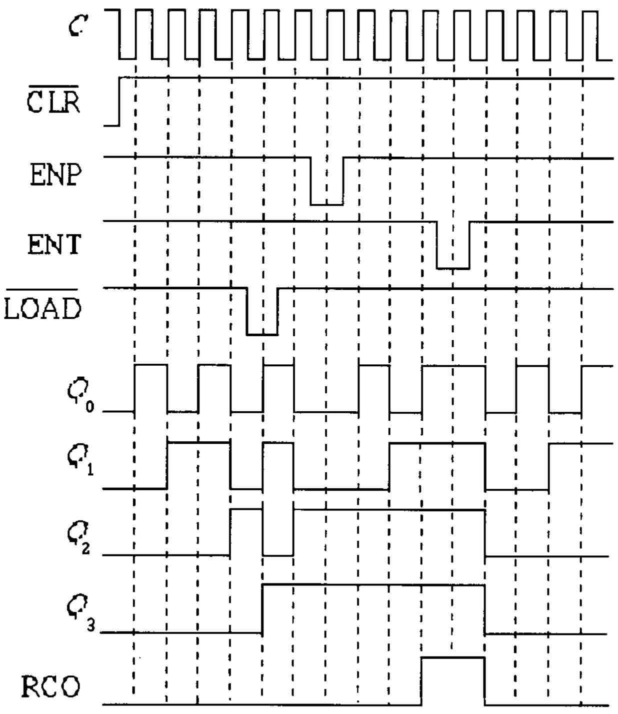


FIGURE 8-7

11. See Figure 8-8.

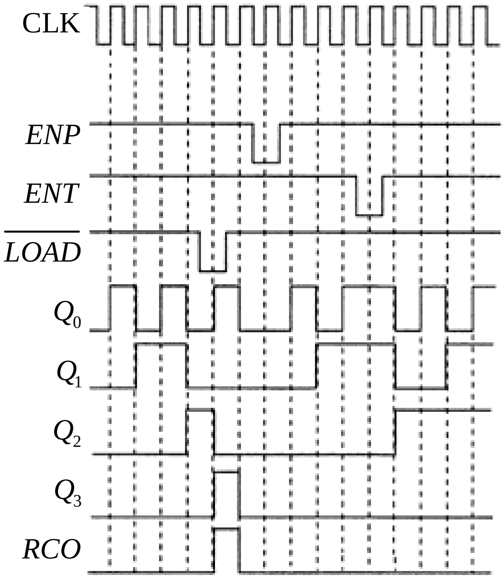


FIGURE 8-8

Section 8-3 Up/Down Synchronous Counters

12. See Figure 8-9.

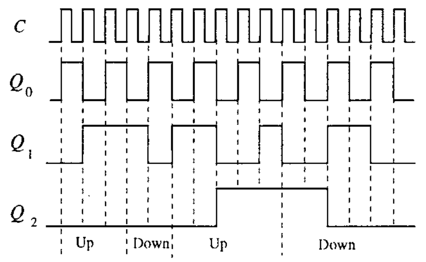


FIGURE 8-9

13. See Figure 8-10.

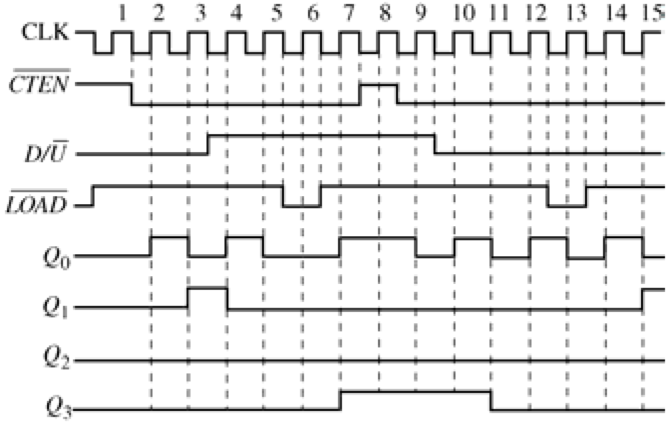


FIGURE 8-10

14. See Figure 8-11.

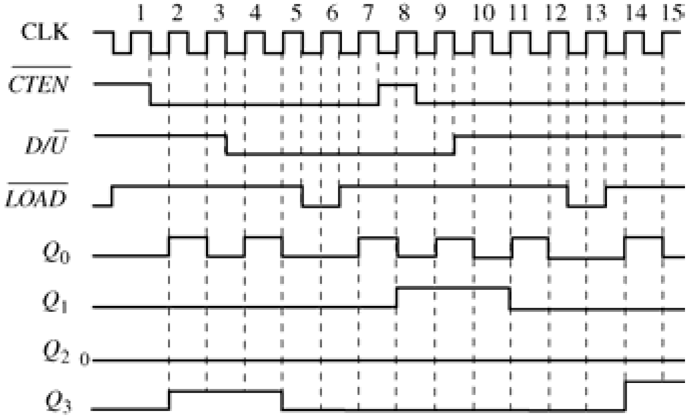


FIGURE 8-11

15. See Figure 8-12.
(*NOTE:* The text answer, Figure P-64, is incorrect in the first printing. It will be corrected to match Figure 8-12 in the 2^{nd} printing.)

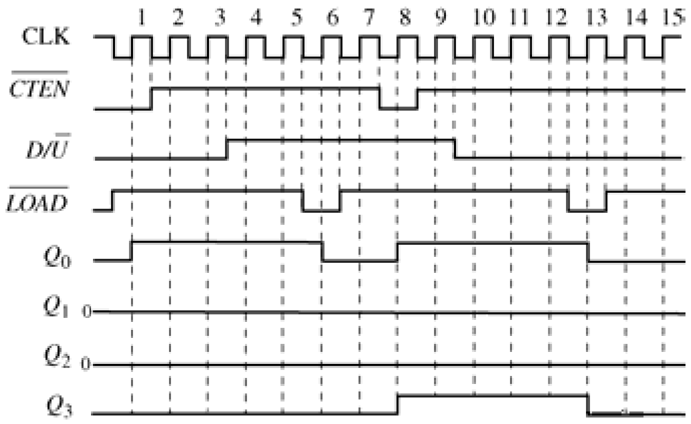


FIGURE 8-12

Section 8-4 Design of Synchronous Counters

16.

	Q_2	Q_1	Q_0	D_2	D_1	D_0
Initially	0	0	0	0	0	1
At CLK 1	0	0	1	0	1	1
At CLK 2	0	1	1	1	1	1
At CLK 3	1	1	1	1	1	0
At CLK 4	1	1	0	1	0	0
At CLK 5	1	0	0	0	0	1
At CLK 6	0	0	1	0	1	1

The sequence is 000 to 001 to 011 to 111 to 110 to 100 and back to 001, etc.

17.

	FF3	FF2	FF1	FF0	Q_3	Q_2	Q_1	Q_0
Initially	Tog	Tog	Tog	Tog	0	0	0	0
After CLK 1	NC	NC	NC	Tog	1	1	1	1
After CLK 2	NC	NC	Tog	Tog	1	1	1	0
After CLK 3	NC	Tog	Tog	Tog	1	1	0	1
After CLK 4	Tog	Tog	Tog	Tog	1	0	1	0
After CLK 5	Tog	Tog	Tog	Tog	0	1	0	1

Tog = toggle, NC = no change

The counter locks up in the 1010 and 0101 states, alternating between them.

18. NEXT-STATE TABLE

Present State		**Next State**	
Q_1	Q_0	Q_1	Q_0
0	0	1	0
1	0	0	1
0	1	1	1
1	1	0	0

TRANSITION TABLE

Output State Transitions (Present state to next state)		**Flip-Flop Inputs**			
Q_1	Q_0	J_1	K_1	J_0	K_0
0 to 1	0 to 0	1	X	0	X
1 to 0	0 to 1	X	1	1	X
0 to 1	1 to 1	1	X	X	0
1 to 0	1 to 0	X	1	X	1

See Figure 8-13.

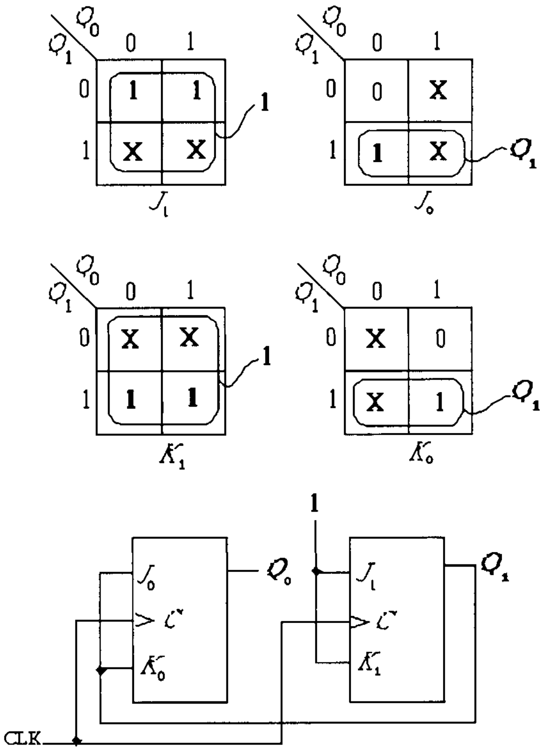


FIGURE 8-13

19. NEXT-STATE TABLE

Present State			**Next State**		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	1	1	0	0
1	0	0	0	1	1
0	1	1	1	0	1
1	0	1	1	1	1
1	1	1	1	1	0
1	1	0	0	1	0
0	1	0	0	0	1

TRANSITION TABLE

Output State Transitions (Present state to next state)			**Flip-flop Inputs**					
Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
0 to 1	0 to 0	1 to 0	1	X	0	X	X	1
1 to 0	0 to 1	0 to 1	X	1	1	X	1	X
0 to 1	1 to 0	1 to 1	1	X	X	1	X	0
1 to 1	0 to 1	1 to 1	X	0	1	X	X	0
1 to 1	1 to 1	1 to 0	X	0	X	0	X	1
0 to 0	1 to 0	0 to 1	0	X	X	1	1	X
1 to 0	1 to 1	0 to 0	X	1	X	0	0	X

See Figure 8-14.

$Q_2Q_1 \backslash Q_0$	0	1
00	X	1
01	0	1
11	X	X
10	X	X

$J_2 = Q_0$

$Q_2Q_1 \backslash Q_0$	0	1
00	X	0
01	X	X
11	X	X
10	1	1

$J_1 = Q_2$

$Q_2Q_1 \backslash Q_0$	0	1
00	X	X
01	1	X
11	0	X
10	1	X

$J_0 = \overline{Q}_1 + \overline{Q}_2$

$Q_2Q_1 \backslash Q_0$	0	1
00	X	X
01	X	X
11	1	0
10	1	0

$K_2 = \overline{Q}_0$

$Q_2Q_1 \backslash Q_0$	0	1
00	X	X
01	1	1
11	0	0
10	X	X

$K_1 = \overline{Q}_2$

$Q_2Q_1 \backslash Q_0$	0	1
00	X	1
01	X	0
11	X	1
10	X	0

$K_0 = \overline{Q}_1\overline{Q}_2 + Q_1Q_2$

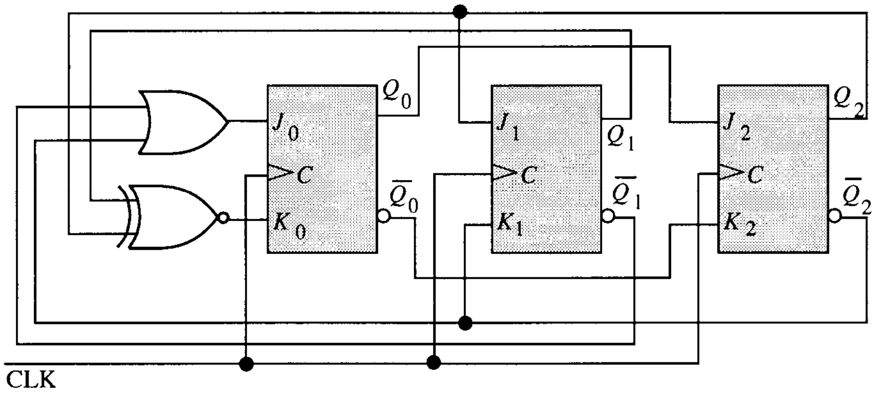


FIGURE 8-14

20. NEXT-STATE TABLE

Present State				Next State			
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	1	0	0	1
1	0	0	1	0	0	0	1
0	0	0	1	1	0	0	0
1	0	0	0	0	0	1	0
0	0	1	0	0	1	1	1
0	1	1	1	0	0	1	1
0	0	1	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	0	0	0

TRANSITION TABLE

Output State Transition (Present State to next state)				Flip-flop Inputs							
Q_3	Q_2	Q_1	Q_0	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0 to 1	0 to 0	0 to 0	0 to 1	1	X	0	X	0	X	1	X
1 to 0	0 to 0	0 to 0	0 to 1	X	1	0	X	0	X	X	0
0 to 1	0 to 0	0 to 0	1 to 0	1	X	0	X	0	X	X	1
1 to 0	0 to 0	0 to 1	0 to 0	X	1	0	X	1	X	0	X
0 to 0	0 to 1	1 to 1	0 to 1	0	X	1	X	X	0	1	X
0 to 0	1 to 0	1 to 1	1 to 1	0	X	X	1	X	0	X	0
0 to 0	0 to 1	1 to 1	1 to 0	0	X	1	X	X	0	X	1
0 to 0	1 to 1	1 to 0	0 to 0	0	X	X	0	X	1	0	X
0 to 0	1 to 1	0 to 0	0 to 1	0	X	X	0	0	X	1	X
0 to 0	1 to 0	0 to 0	1 to 0	0	X	X	1	0	X	X	1

Binary states for 10, 11, 12, 13, 14, and 15 are unallowed and can be represented by don't cares.

See Figure 8-15. Counter implementation is straightforward from input expressions.

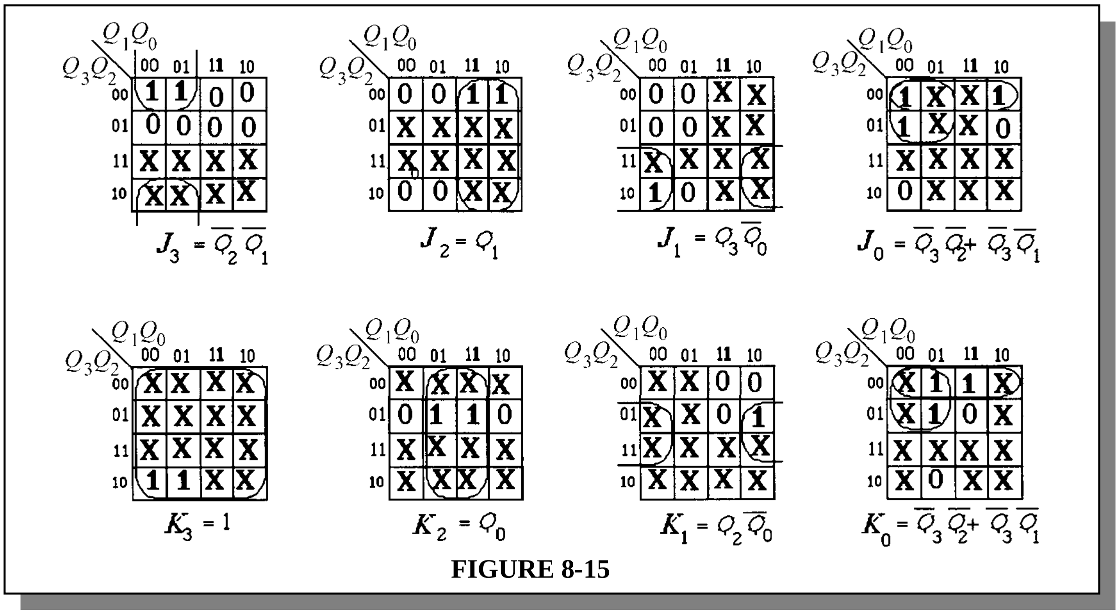

Q_1Q_0
Q_3Q_2
00 01 11 10
$J_3 = \overline{Q}_2\overline{Q}_1$
$J_2 = Q_1$
$J_1 = Q_3\overline{Q}_0$
$J_0 = \overline{Q}_3\overline{Q}_2 + \overline{Q}_3\overline{Q}_1$
$K_3 = 1$
$K_2 = Q_0$
$K_1 = Q_2\overline{Q}_0$
$K_0 = \overline{Q}_3\overline{Q}_2 + \overline{Q}_3\overline{Q}_1$

FIGURE 8-15

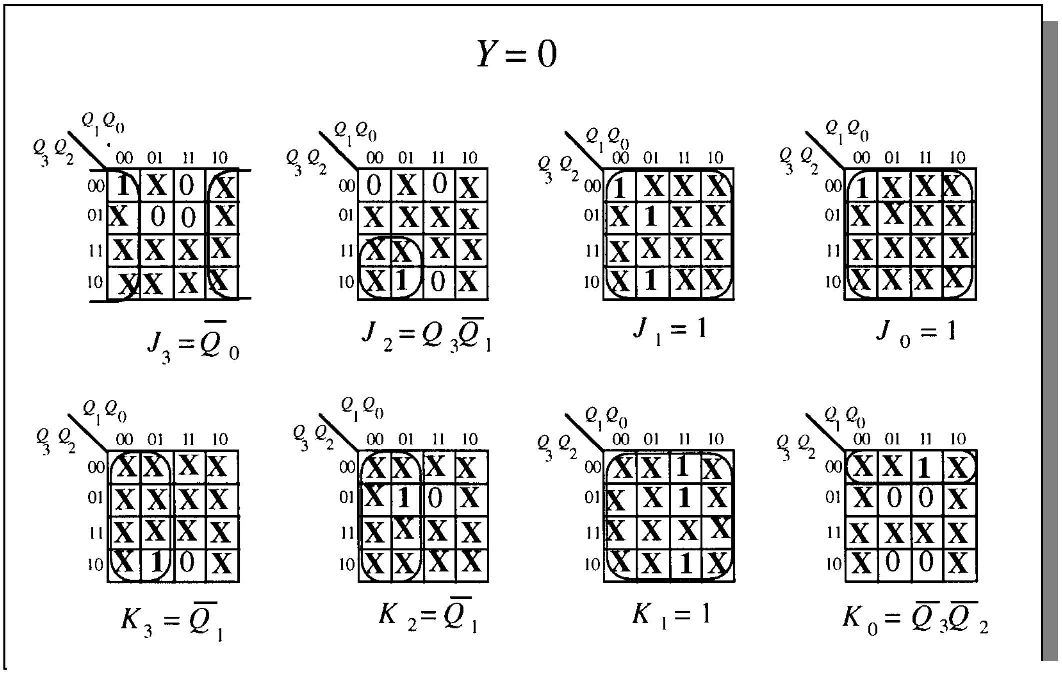
21. NEXT-STATE TABLE

Present State				**Next State**							
				Y = 1 (Up)				Y = 0 (Down)			
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	1	1	1	0	1	1
0	0	1	1	0	1	0	1	0	0	0	0
0	1	0	1	0	1	1	1	0	0	1	1
0	1	1	1	1	0	0	1	0	1	0	1
1	0	0	1	1	0	1	1	0	1	1	1
1	0	1	1	0	0	0	0	1	0	0	1

TRANSITION TABLE

Output State Transitions				***Y***	**Flip-flop Inputs**			
(Present State to next state)								
Q_3	Q_2	Q_1	Q_0		J_3K_3	J_2K_2	J_1K_1	J_0K_0
0 to 1	0 to 0	0 to 1	0 to 1	0	1X	0X	1X	1X
0 to 0	0 to 0	0 to 1	0 to 1	1	0X	0X	1X	1X
0 to 0	0 to 0	1 to 0	1 to 0	0	0X	0X	X1	X1
0 to 0	0 to 1	1 to 0	1 to 1	1	0X	1X	X1	X0
0 to 0	1 to 0	0 to 1	1 to 1	0	0X	X1	1X	X0
0 to 0	1 to 1	0 to 1	1 to 1	1	0X	X0	1X	X0
0 to 0	1 to 1	1 to 0	1 to 1	0	0X	X0	X1	X0
0 to 1	1 to 0	1 to 0	1 to 1	1	1X	X1	X1	X0
1 to 0	0 to 1	0 to 1	1 to 1	0	X1	1X	1X	X0
1 to 1	0 to 0	0 to 1	1 to 1	1	X0	0X	1X	X0
1 to 1	0 to 0	1 to 0	1 to 1	0	X0	0X	X1	X0
1 to 0	0 to 0	1 to 0	1 to 0	1	X1	0X	X1	X1

See Figure 8-16.



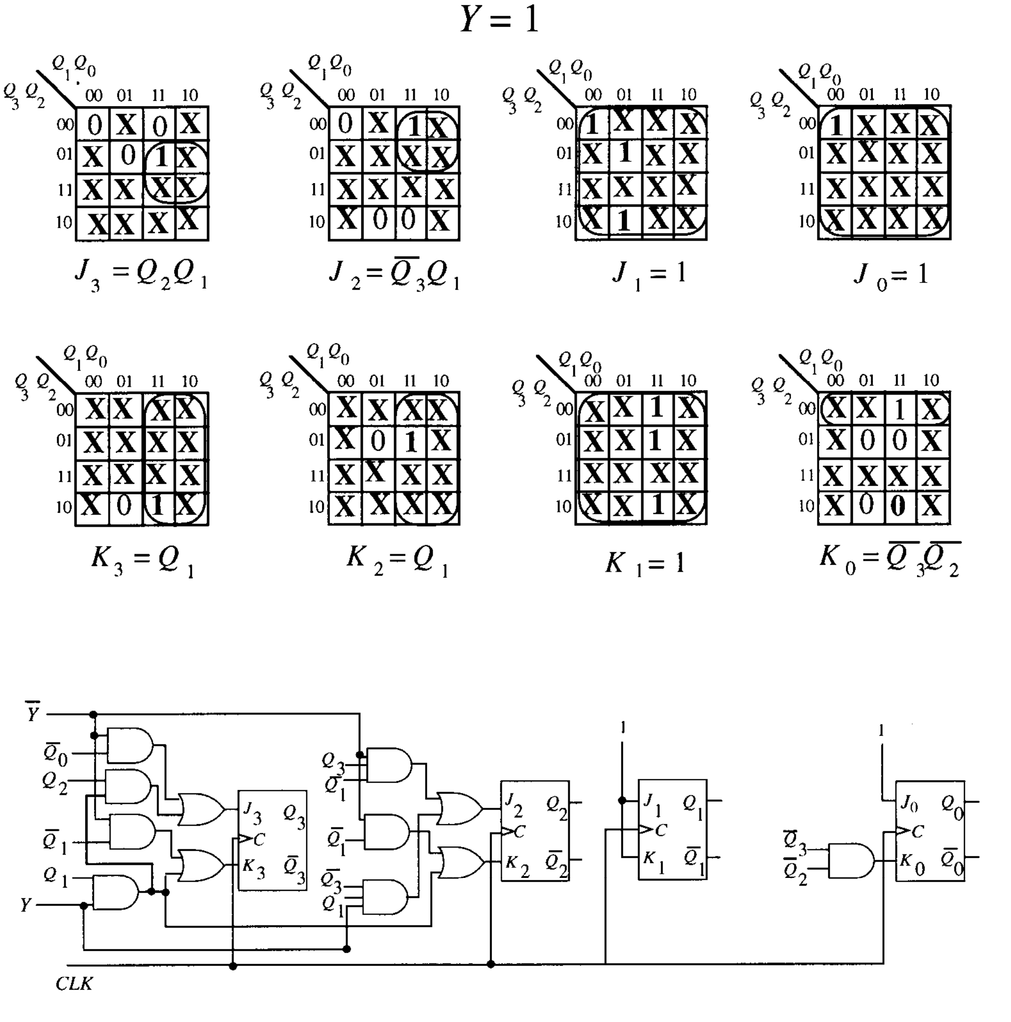


FIGURE 8-16

Section 8-5 Cascaded Counters

22. (a) Modulus = 4 × 8 × 2 = **64**

$$f_1 = \frac{1\,\text{kHz}}{4} = \mathbf{250\ Hz}$$

$$f_2 = \frac{250\,\text{Hz}}{8} = \mathbf{31.25\ Hz}$$

$$f_3 = \frac{31.25\,\text{Hz}}{2} = \mathbf{15.625\ Hz}$$

(b) Modulus = 10 × 10 × 10 × 2 = **2000**

$$f_1 = \frac{100\text{ kHz}}{10} = \mathbf{10\text{ kHz}}$$

$$f_2 = \frac{10\text{ kHz}}{10} = \mathbf{1\text{ kHz}}$$

$$f_3 = \frac{1\text{ kHz}}{10} = \mathbf{100\text{ Hz}}$$

$$f_4 = \frac{100\text{ Hz}}{2} = \mathbf{50\text{ Hz}}$$

(c) Modulus = 3 × 6 × 8 × 10 × 10 = **14400**

$$f_1 = \frac{21\text{ MHz}}{3} = \mathbf{7\text{ MHz}}$$

$$f_2 = \frac{7\text{ MHz}}{6} = \mathbf{1.167\text{ MHz}}$$

$$f_3 = \frac{1.167\text{ MHz}}{8} = \mathbf{145.875\text{ kHz}}$$

$$f_4 = \frac{145.875\text{ kHz}}{10} = \mathbf{14.588\text{ kHz}}$$

$$f_5 = \frac{14.588\text{ kHz}}{10} = \mathbf{1.459\text{ kHz}}$$

(d) Modulus = 2 × 4 × 6 × 8 × 16 = **6144**

$$f_1 = \frac{39.4\text{ kHz}}{2} = \mathbf{19.7\text{ kHz}}$$

$$f_2 = \frac{19.7\text{ kHz}}{4} = \mathbf{4.925\text{ kHz}}$$

$$f_3 = \frac{4.925\text{ kHz}}{6} = \mathbf{820.83\text{ Hz}}$$

$$f_4 = \frac{820.683}{8} = \mathbf{102.6\text{ Hz}}$$

$$f_5 = \frac{102.6\text{ Hz}}{16} = \mathbf{6.41\text{ Hz}}$$

23. See Figure 8-17.

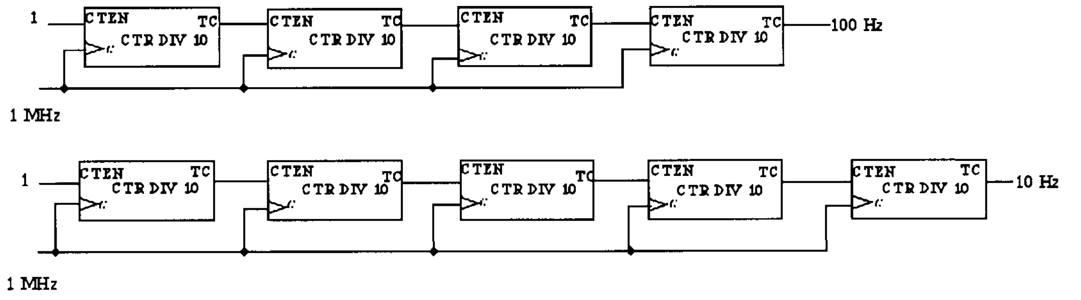


FIGURE 8-17

24. See Figure 8-18.

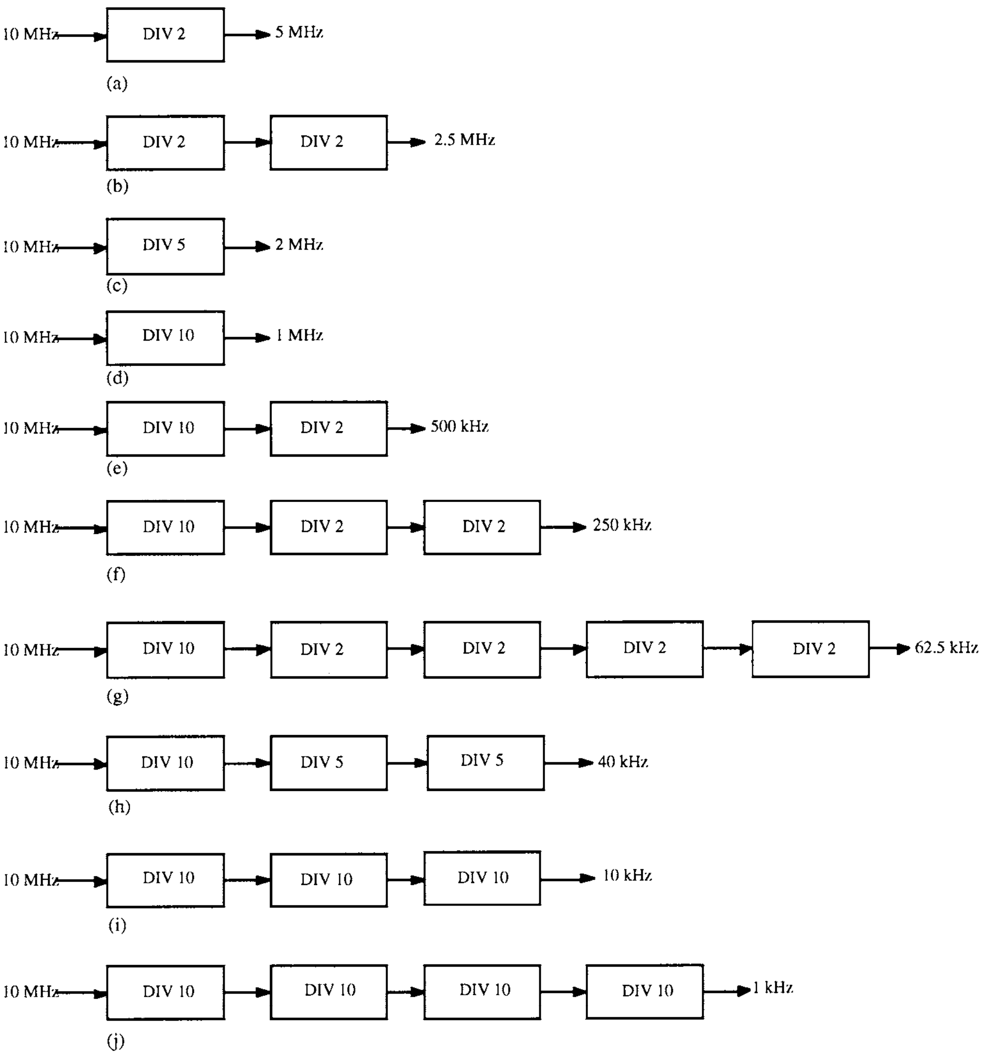


FIGURE 8-18

Section 8-6 Counter Decoding

25. See Figure 8-19.

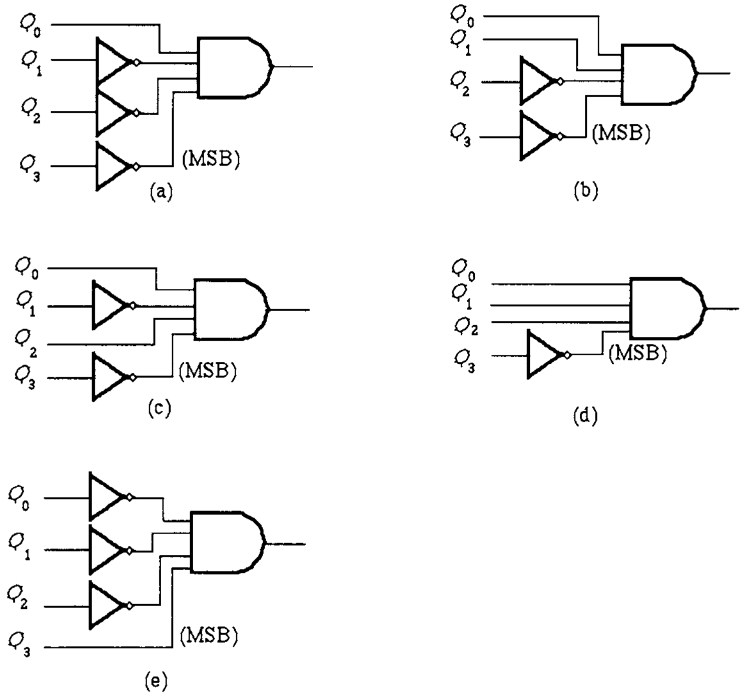


FIGURE 8-19

26. See Figure 8-20.

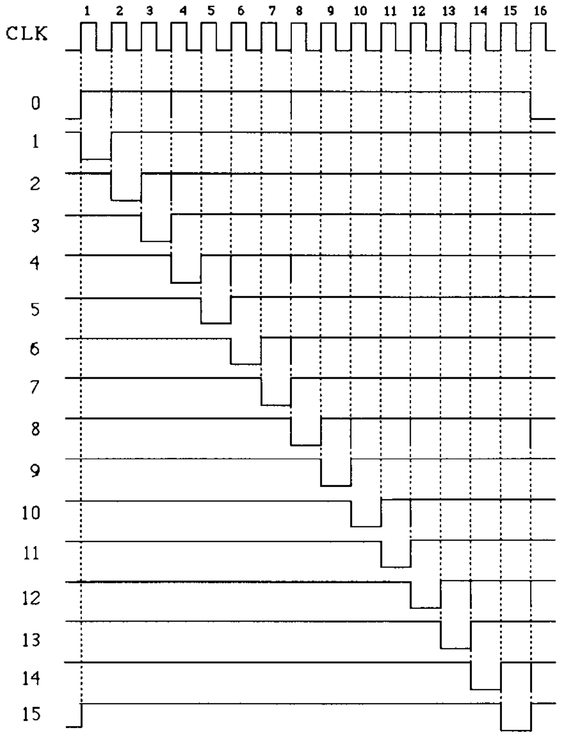


FIGURE 8-20

27. The states with an asterisk are the transition states that produce glitches on the decoder outputs. The glitches are indicated on the waveforms in Figure 8-20 (Problem 8-26) by short vertical lines.

Initial	0000
CLK 1	0001
CLK 2	0000 *
	0010
CLK 3	0011
CLK 4	0010 *
	0000 *
	0100
CLK 5	0100
CLK 6	0100 *
	0110
CLK 7	0111
CLK 8	0110 *
	0100 *
	0000 *
	1000
CLK 9	1001
CLK 10	1000*
	1010
CLK 11	1011
CLK 12	1010 *
	1000 *
	1100
CLK 13	1101
CLK 14	1100 *
	1110
CLK 15	1111
CLK 16	1110 *
	1100 *
	1000 *
	0000

28. See Figure 8-21.

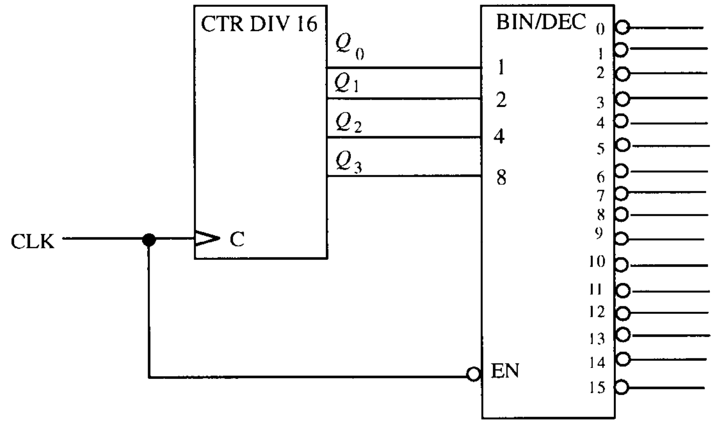


FIGURE 8-21

29. See Figure 8-22.

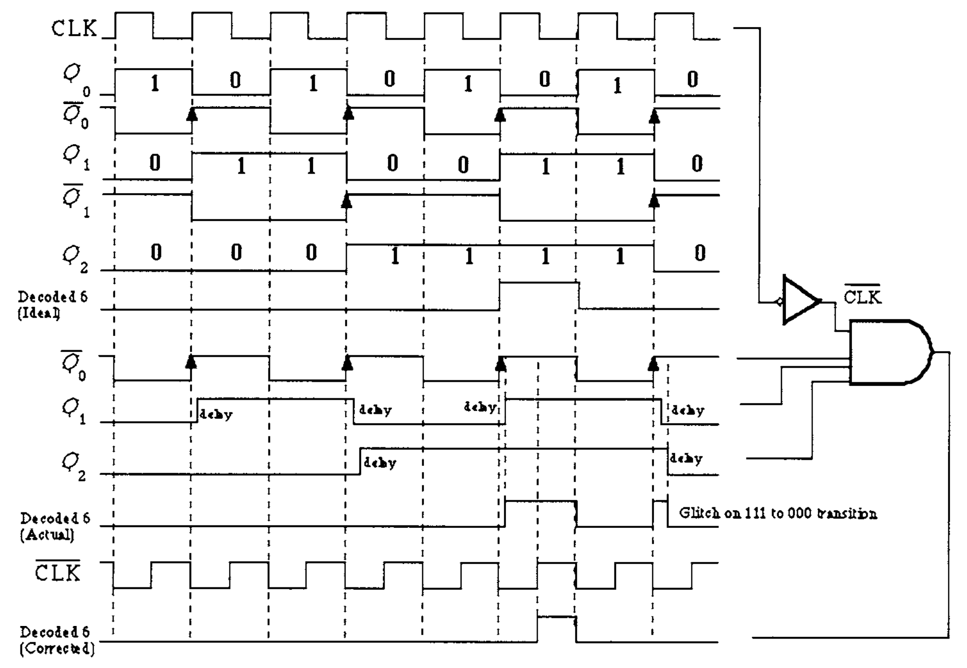


FIGURE 8-22

30. ① There is a possibility of a glitch on decode 2 at the positive-going edge of CLK 4 if the propagation delay of FF0 is less than FF1 or FF2.

② There is a possibility of a glitch on decode 7 at the positive-going edge of CLK 4 if the propagation delay of FF2 is less than FF0 and FF1.

③ There is a possibility of a glitch on decode 7 at the positive-going edge of CLK 6 if the propagation delay of FF1 is less than FF0.

See the timing diagram in Figure 8-23 which is expanded to show the delays.

Any glitches can be prevented by using CLK as an input to both decode gates.

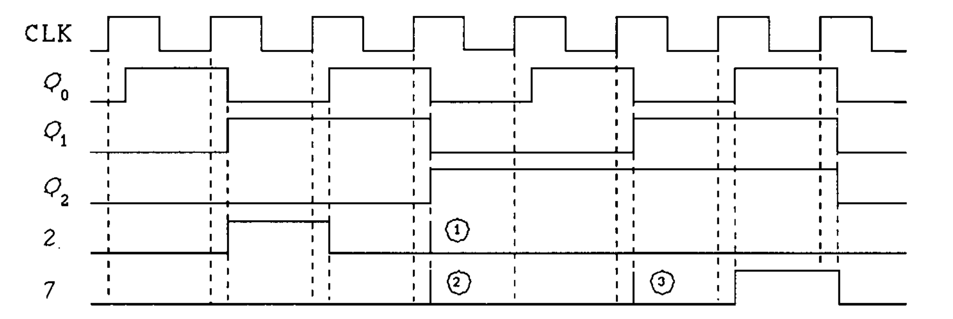


FIGURE 8-23

Section 8-7 Counter Applications

31. For the digital clock in Figure 8-49 of the text reset to 12:00:00, the binary state of each counter after sixty-two 60-Hz pulses are:

Hours, tens: **0001**
Hours, units: **0010**
Minutes, tens: **0000**
Minutes, units: **0001**
Seconds, tens: **0000**
Seconds, units: **0010**

32. For the digital clock, the counter output frequencies are:
Divide-by-60 input counter:

$$\frac{60\text{ Hz}}{60} = 1\text{ Hz}$$

Seconds counter:

$$\frac{1\text{ Hz}}{60} = 16.7\text{ mHz}$$

Minutes counter:

$$\frac{16.7\text{ mHz}}{60} = 278\ \mu\text{Hz}$$

Hours counter:

$$\frac{278\ \mu\text{Hz}}{12} = 23.1\ \mu\text{Hz}$$

33. 53 + 37 – 22 = **68**

34. See Figure 8-24.

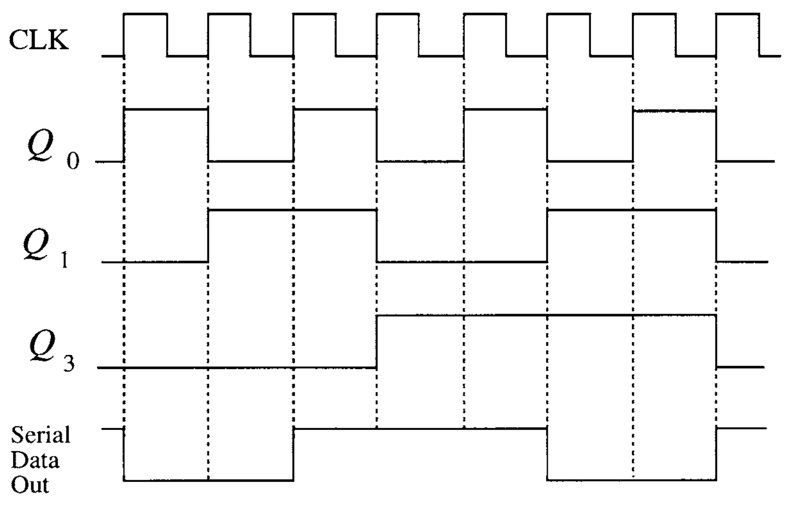


FIGURE 8-24

Section 8-9 Troubleshooting

35. (a) Q_0 and Q_1 will not change due to the clock shorted to ground at FF0.

(b) Q_0 being open does not affect normal operation. See Figure 8-25.

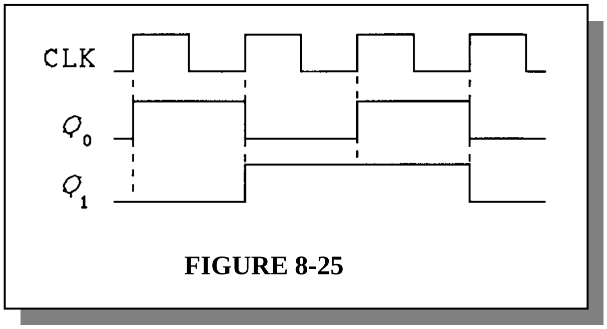


FIGURE 8-25

(c) See Figure 8-26.

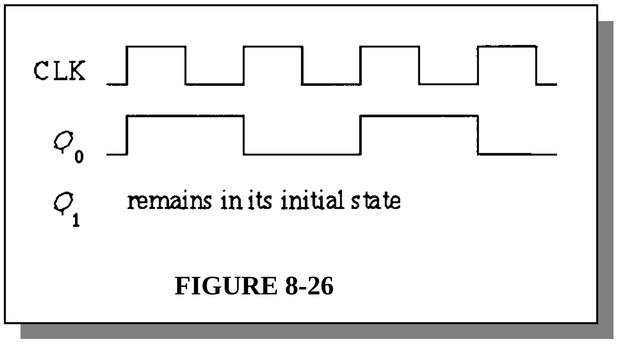


FIGURE 8-26

(d) Normal operation because an open *J* input acts as a HIGH.

(e) A shorted *K* input will pull all *J* and *K* inputs LOW and the counter will not change from its initial state.

36. (a) Q_0 and Q_1 will not change from initial states.

(b) See Figure 8-27.

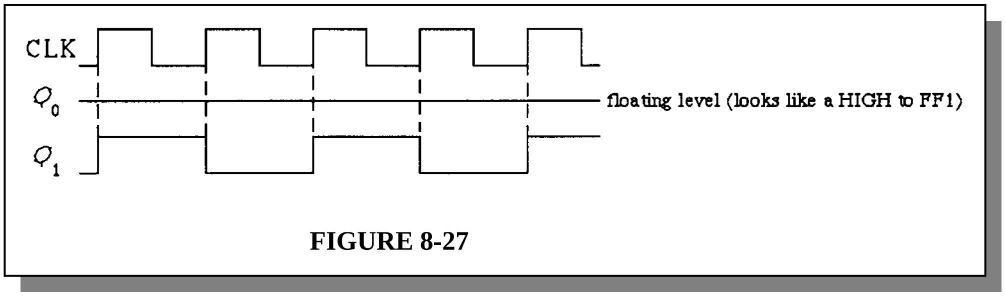


FIGURE 8-27

(c) See Figure 8-28.

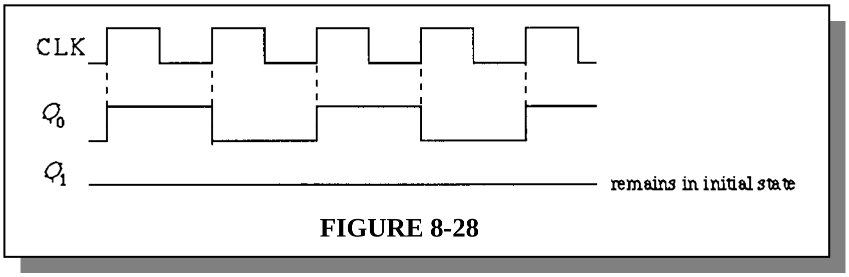


FIGURE 8-28

(d) Normal operation. See Figure 8-29.

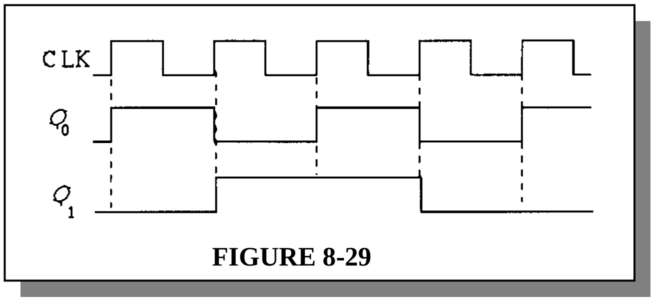


FIGURE 8-29

(e) Both *J* and *K* of FF1 are pulled LOW if *K* is grounded, producing a no-change condition. Q_0 also grounded. See Figure 8-30.

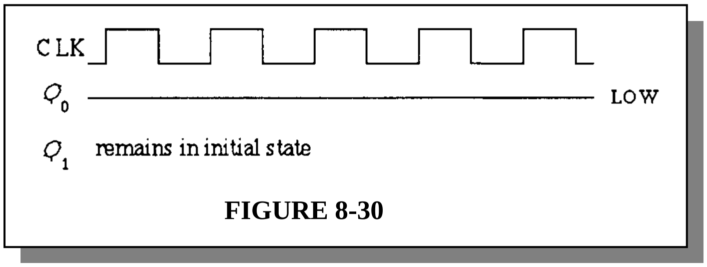


FIGURE 8-30

37. First, determine the correct waveforms and observe that Q_0 is correct but Q_1 and Q_2 are incorrect in Figure 8-83 in the text. See Figure 8-31.

Since Q_1 goes HIGH and stays HIGH, FF1 must be in the SET state ($J = 1$, $K = 0$). There must be a wiring error at the *J* and *K* inputs to FF1; *K* must be connected to ground rather than to the *J* input.

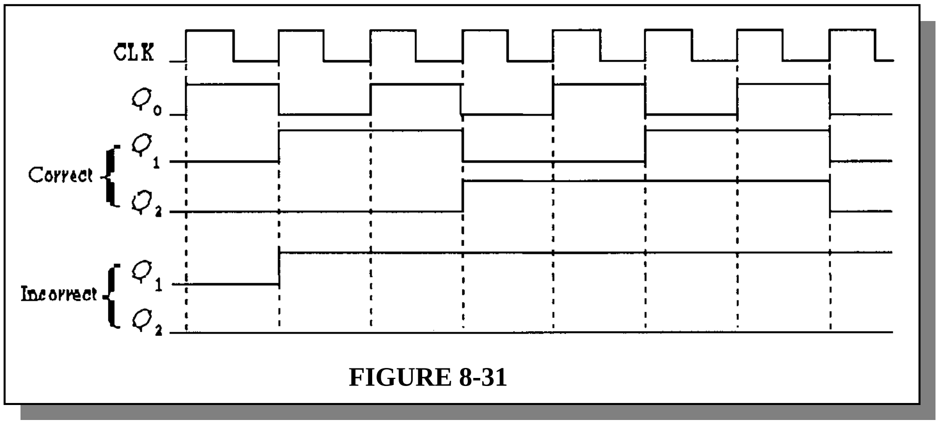


FIGURE 8-31

38. Since Q_2 toggles on each clock pulse, its J and K inputs must be constantly HIGH. The most probable fault is that the AND gate's output is *open*.

39. If the Q_0 input to the AND gate is *open*, the JK inputs to FF2 are as shown in Figure 8-32.

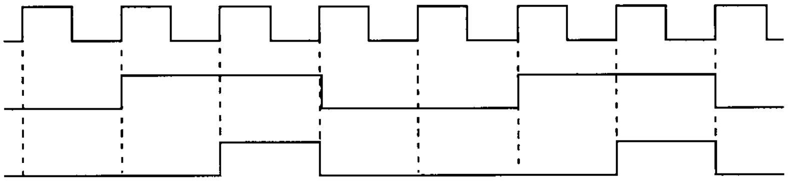


FIGURE 8-32

40. Number of states = 40,000

$$f_{out} = \frac{5\text{ MHz}}{40{,}000} = 125\text{ Hz}$$

76.2939 Hz is not correct. The faulty division factor is

$$\frac{5\text{ MHz}}{76.2939\text{ Hz}} = 65{,}536$$

Obviously, the counter is going through all of its states. This means that the $63C0_{16}$ on its parallel inputs is not being loaded. Possible faults are:

- Inverter output is stuck HIGH or open.
- RCO output of last counter is stuck LOW.

41.

Stage	Open	Loaded Count	f_{out}
1	0	63C1	250.006 Hz
1	1	63C2	250.012 Hz
1	2	63C4	250.025 Hz
1	3	63C8	250.050 Hz
2	0	63D0	250.100 Hz
2	1	63E0	250.200 Hz
2	2	63C0	250 Hz
2	3	63C0	250 Hz
3	0	63C0	250 Hz
3	1	63C0	250 Hz
3	2	67C0	256.568 Hz
3	3	6BC0	263.491 Hz
4	0	73C0	278.520 Hz
4	1	63C0	250 Hz
4	2	63C0	250 Hz
4	3	E3C0	1.383 kHz

42. ▪ The flip-flop output is stuck HIGH or open.

▪ The least significant BCD/7-segment input is open.

See Figure 8-33.

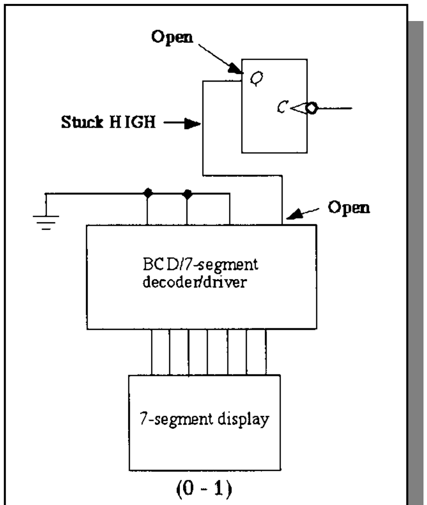


FIGURE 8-33

43. Th DIV 6 is the tens of minutes counter. Q_1 open causes a continuous apparent HIGH output to the decode 6 gate and to the BCD/7-segment decoder/driver.

The apparent counter sequence is shown in the table.

Actual State of Ctr.	**Apparent state**			
	Q_3	Q_2	Q_1	Q_0
0	0	0	1	0
1	0	0	1	1
2	0	0	1	0
3	0	0	1	1
4	0	1	1	0

The decode 6 gate interprets count 4 as a 6 (0110) and clears the counter back to 0 (actually 0010). Thus, the apparent (not actual) sequence is as shown in the table.

44. There are several possible causes of the malfunction. First check power to all units. Other possible faults are listed below.

- Sensor Latch
  Action: Disconnect entrance sensor and pulse sensor input.
  Observation: Latch should SET.
  Conclusion: If latch does not SET, replace it.

- NOR gate
  Action: Pulse sensor input.
  Observation: Pulse on gate output.
  Conclusion: If there is no pulse, replace gate.

- Counter

  Action: Pulse sensor input.
  Observation: Counter should advance.
  Conclusion: If counter does not advance, replace it.

- Output Interface

  Action: Pulse sensor input until terminal count is reached.
  Observation: FULL indication and gate lowered
  Conclusion: No FULL indication or if gate does not lower, replace interface.

- Sensor/Cable

  Action: Try to activate sensor.
  Observation: If all previous checks are OK, sensor or cable is faulty.
  Conclusion: Replace sensor or cable.

System Application Activity

45. The expressions for the D_0 and the D_1 flip-flop inputs in the sequential logic portion of the system were developed in the System Application Activity. Figure 8-34 shows the NAND implementation.

$$D_0 = \overline{Q}_1 Q_0 + \overline{Q}_1 \overline{T}_L V_S + Q_0 T_L V_S$$

$$D_1 = Q_0 \overline{T}_L + Q_1 T_S$$

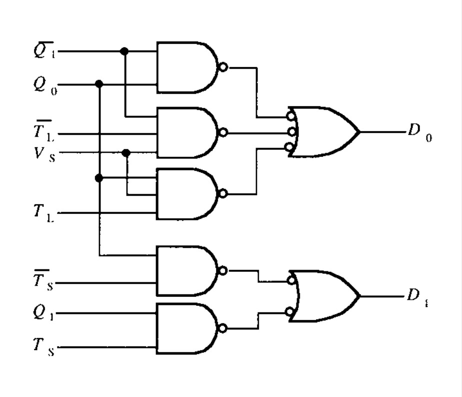


FIGURE 8-34

46. See Figure 8-35.

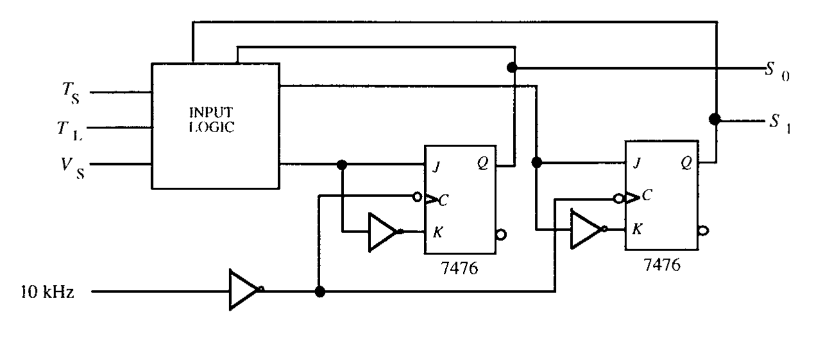


FIGURE 8-35

47. The time interval for the green light can be increased from 25 s to 60 s by increasing the value of either the resistor or the capacitor value by

$$\frac{60\,\text{s}}{25\,\text{s}} = 2.4 \text{ times}$$

Special Design Problems

48. See Figure 8-36.

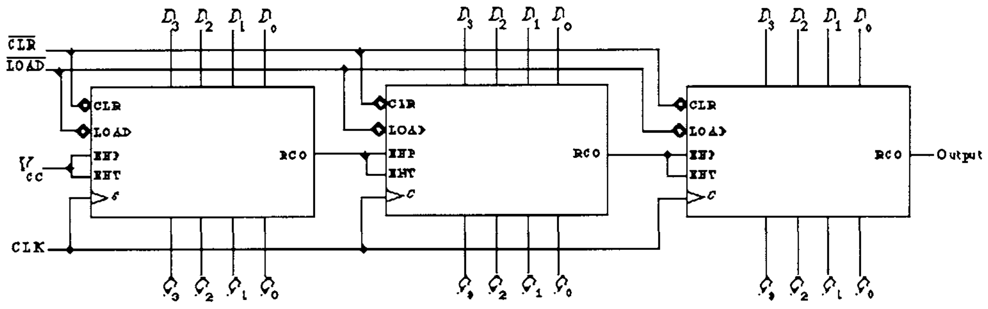


FIGURE 8-36

49. 65,536 – 30,000 = 35,536

Preset the counter to 35,536 so that it counts from 35,536 up to 65,536 on each full cycle, thus producing a sequence of 30,000 states (modulus 30,000).

35,536 = 1000101011010000_2 = **8AD0$_{16}$**

See Figure 8-37.

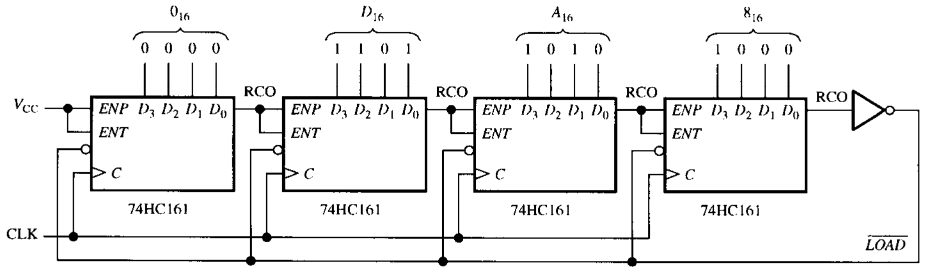


FIGURE 8-37

50. 65,536 – 50,000 = 15,536
Preset the counter to 15,536 so that it counts from 15,536 up to 65,536 on each full cycle, thus producing a sequence of 50,000 states (modulus 50,000).

15,536 = 11110010110000_2 = **3CB0**$_{16}$

See Figure 8-38.

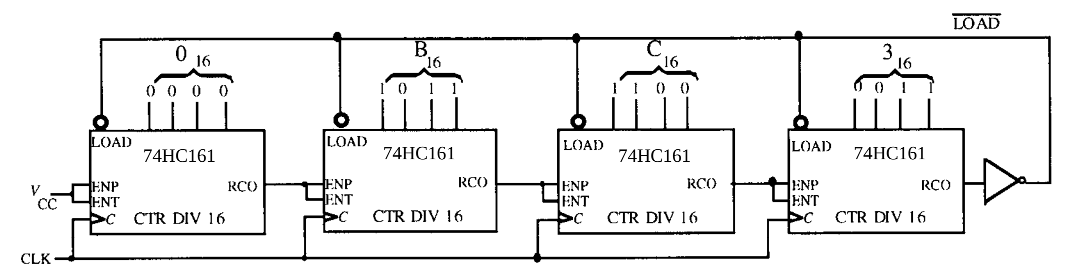


FIGURE 8-38

51. The approach is to preset the hours and minutes counters independently, each with a fast or slow preset mode. The seconds counter is not preset. One possible implementation is shown in Figure 8-39.

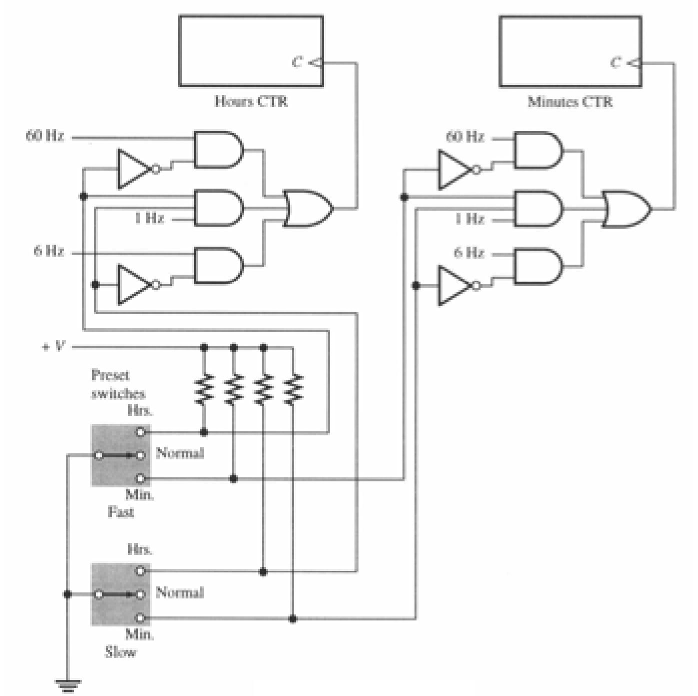


FIGURE 8-39

52. See Figure 8-40.

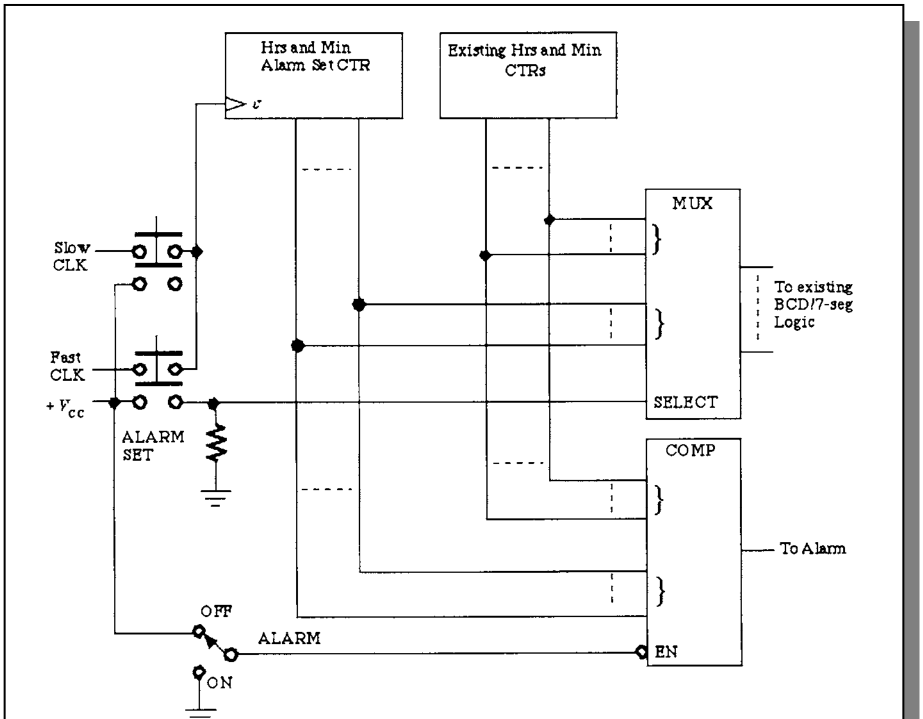
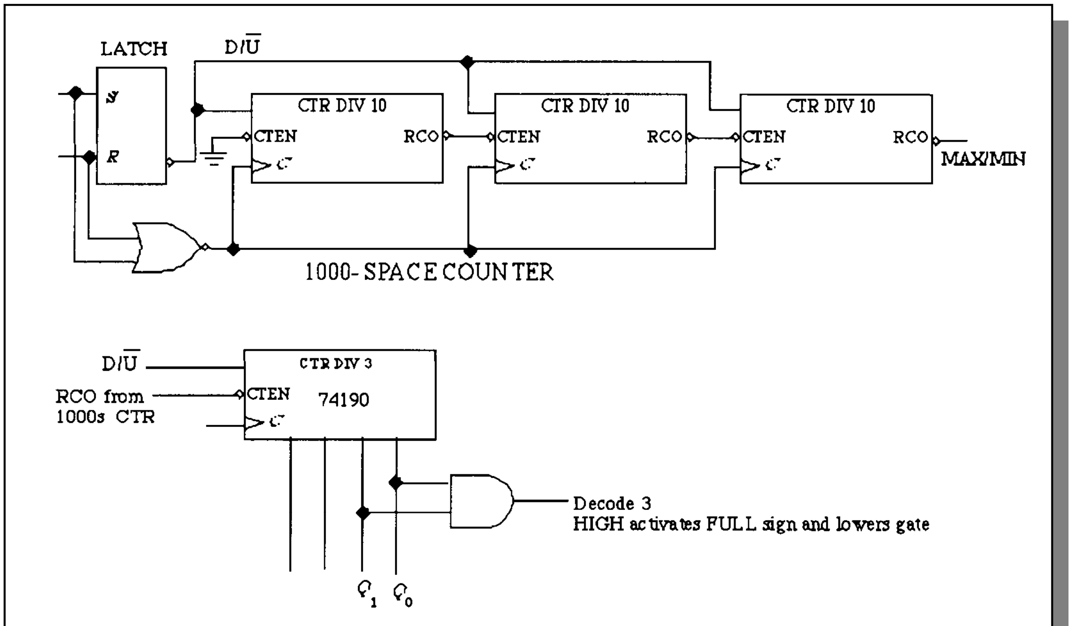


FIGURE 8-40

53. See Figure 8-41.



Add to 1000-space counter for expansion to 3000 spaces

FIGURE 8-41

Chapter 8

54. See Figure 8-42.

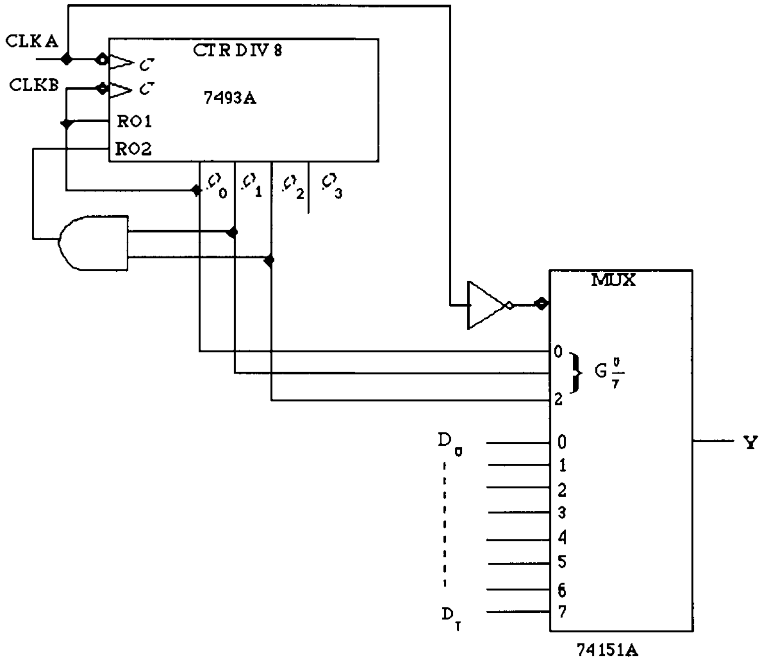


FIGURE 8-42

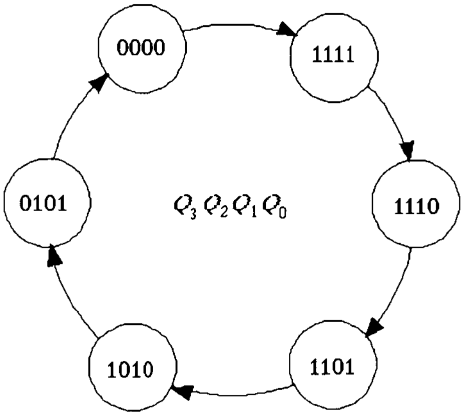
55. NEXT-STATE TABLE

Present State				**Next State**			
Q_3	Q_2	Q_1	Q_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	1	1	1	1
1	1	1	1	1	1	1	0
1	1	1	0	1	1	0	1
1	1	0	1	1	0	1	0
1	0	1	0	0	1	0	1
0	1	0	1	0	0	0	0

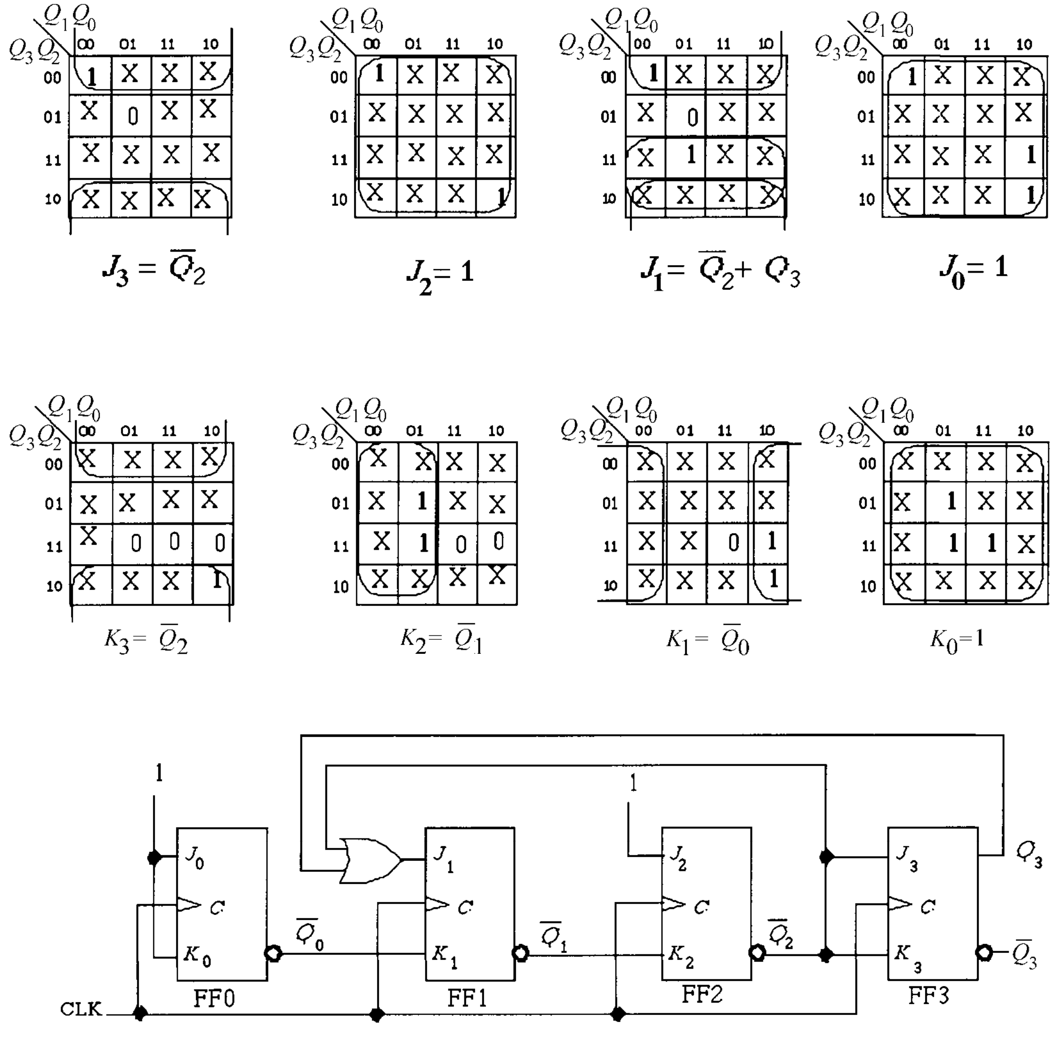
TRANSITION TABLE

Output State Transitions				**Flip-flop Inputs**			
Q_3	Q_2	Q_1	Q_0	J_3K_3	J_2K_2	J_1K_1	J_0K_0
0 to 1	0 to 1	0 to 1	0 to 1	1X	1X	1X	1X
1 to 1	1 to 1	1 to 1	1 to 0	X0	X0	X0	X1
1 to 1	1 to 1	1 to 0	0 to 1	X0	X0	X1	1X
1 to 1	1 to 0	0 to 1	1 to 0	X0	X1	1X	X1
1 to 0	0 to 1	1 to 0	0 to 1	X1	1X	X1	1X
0 to 0	1 to 0	0 to 0	1 to 0	0X	X1	0X	X1

See Figure 8-43.



The desired sequence



$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	1	X	X	X
01	X	0	X	X
11	X	X	X	X
10	X	X	X	X

$J_3 = \overline{Q}_2$

$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	1	X	X	X
01	X	X	X	X
11	X	X	X	X
10	X	X	X	1

$J_2 = 1$

$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	1	X	X	X
01	X	0	X	X
11	X	1	X	X
10	X	X	X	X

$J_1 = \overline{Q}_2 + Q_3$

$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	1	X	X	X
01	X	X	X	X
11	X	X	X	1
10	X	X	X	1

$J_0 = 1$

$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	0	0	0
10	X	X	X	1

$K_3 = \overline{Q}_2$

$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	X	X	X	X
01	X	1	X	X
11	X	1	0	0
10	X	X	X	X

$K_2 = \overline{Q}_1$

$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	0	1
10	X	X	X	1

$K_1 = \overline{Q}_0$

$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	X	X	X	X
01	X	1	X	X
11	X	1	1	X
10	X	X	X	X

$K_0 = 1$

FIGURE 8-43

Chapter 8

56. See Figure 8-44.

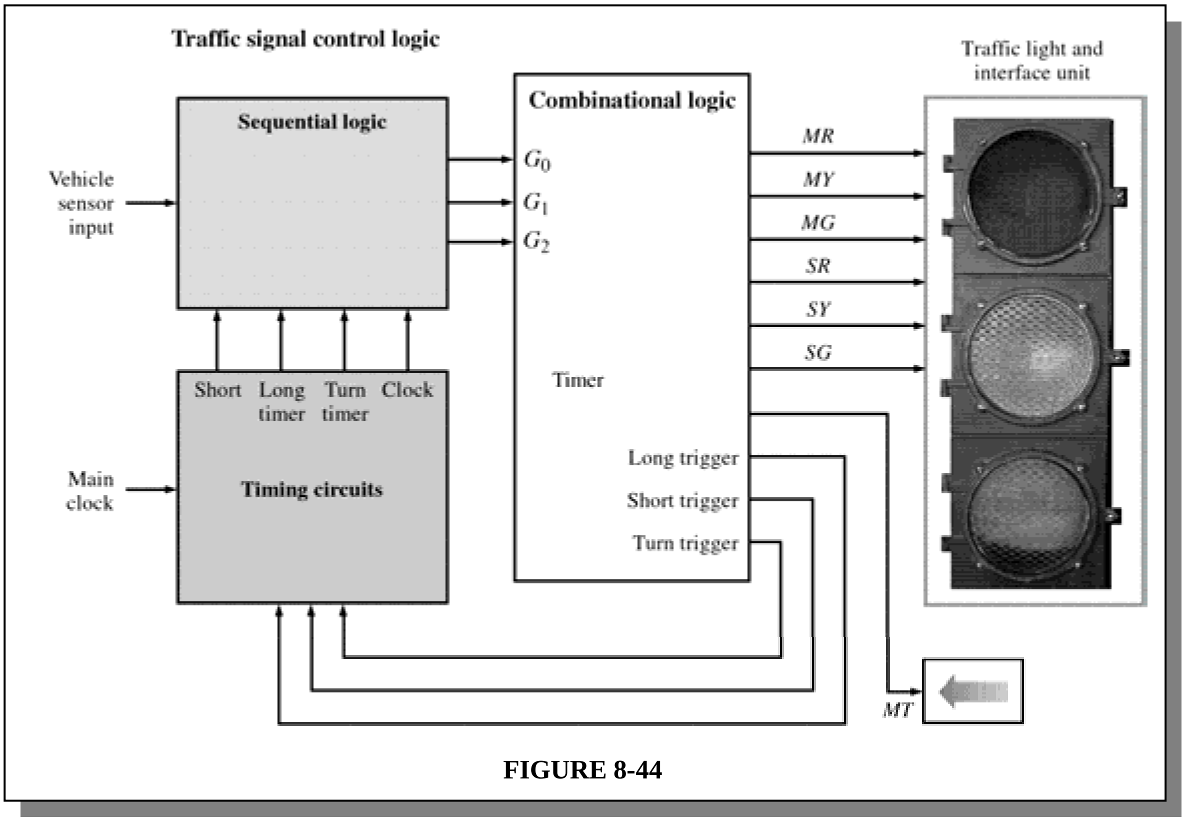


FIGURE 8-44

Multisim Troubleshooting Practice

57. *Q* output of U3 open.

58. $\overline{SET}$ input of U1 open.

59. Pin A of G3 open.

60. No fault.

61. Pin 9 open.

CHAPTER 9
SHIFT REGISTERS

Section 9-1 Basic Shift Register Operations

1. Shift registers store binary data in a series of flip-flops or other storage elements.

2. 1 byte = **8 bits**; 2 bytes = **16 bits**

3. Shift data and store data

Section 9-2 Serial In/Serial Out Shift Registers

4. Initially: 0000
1st CLK: 1000
2nd CLK: 1100
3rd CLK: 0110

5. See Figure 9-1.

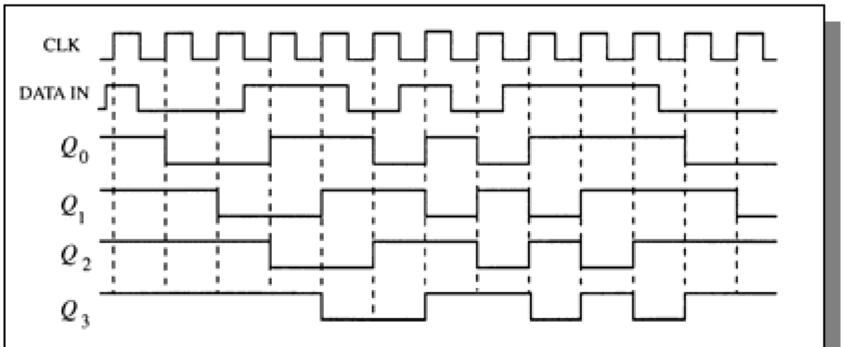


FIGURE 9-1

6. See Figure 9-2.

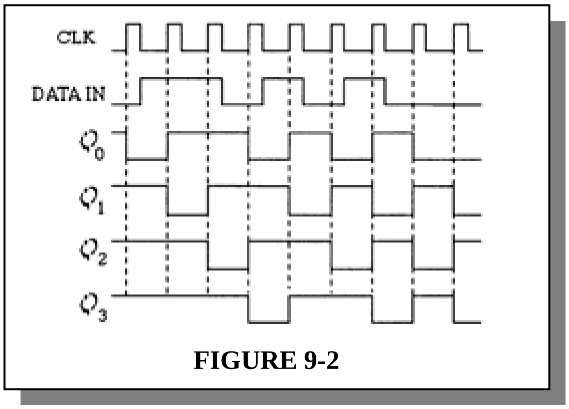


FIGURE 9-2

7.

Initially	101001111000
CLK 1	010100111100
CLK 2	001010011110
CLK 3	000101001111
CLK 4	000010100111
CLK 5	100001010011
CLK 6	110000101001
CLK 7	111000010100
CLK 8	011100001010
CLK 9	001110000101
CLK 10	000111000010
CLK 11	100011100001
CLK 12	110001110000

8. See Figure 9-3.

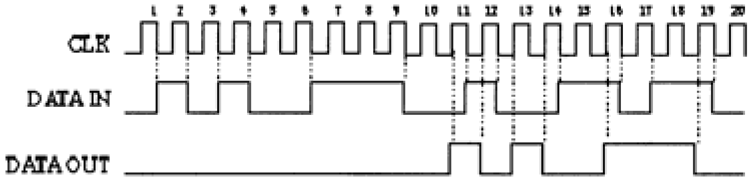


FIGURE 9-3

9. See Figure 9-4.

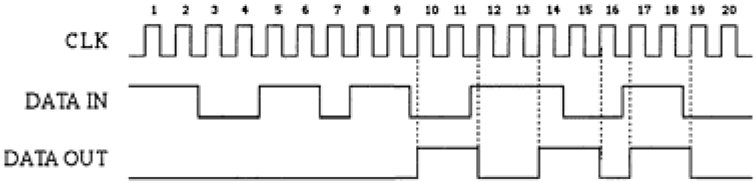
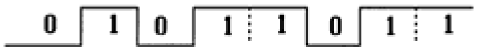


FIGURE 9-4

10. See Figure 9-5.



The number binary number 11011010 is stored in the register after eight clock pulses.

FIGURE 9-5

Section 9-3 Serial In/Parallel Out Shift Registers

11. See Figure 9-6.

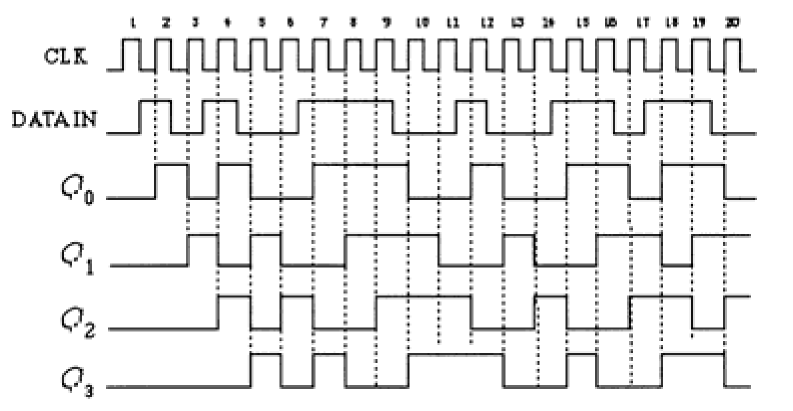


FIGURE 9-6

12. See Figure 9-7.

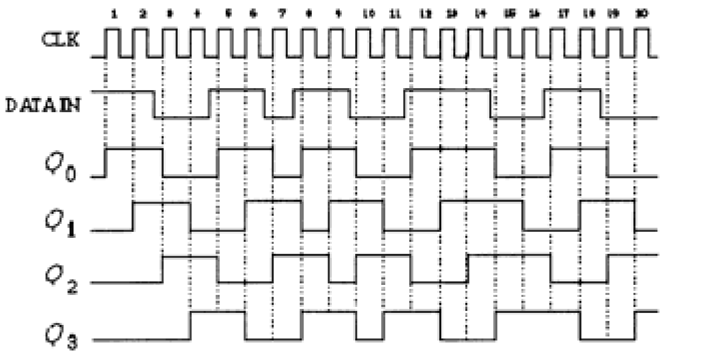


FIGURE 9-7

13. See Figure 9-8.

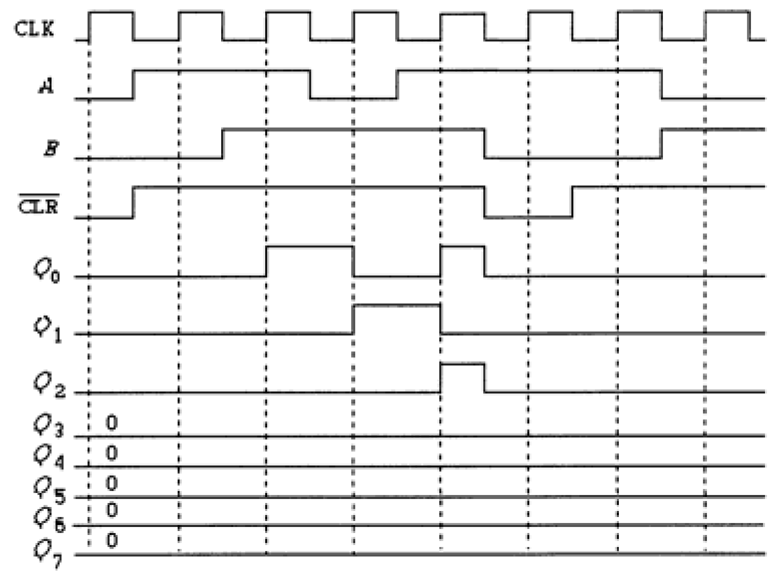


FIGURE 9-8

Section 9-4 Parallel In/Serial Out Shift Registers

14. See Figure 9-9.

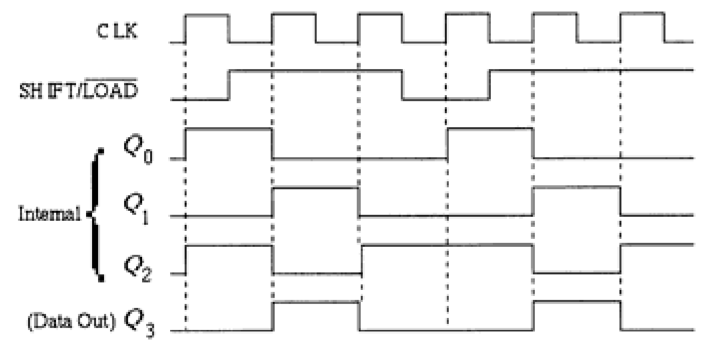


FIGURE 9-9

15. See Figure 9-10.

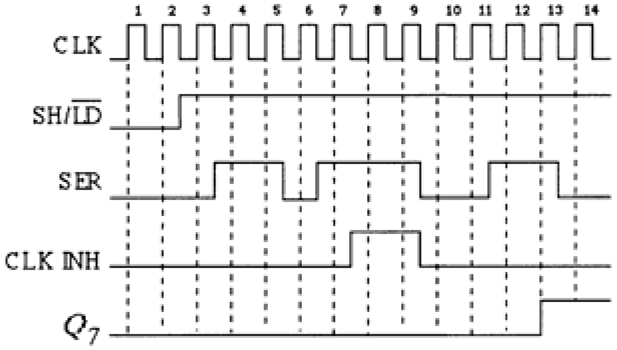


FIGURE 9-10

16. See Figure 9-11.

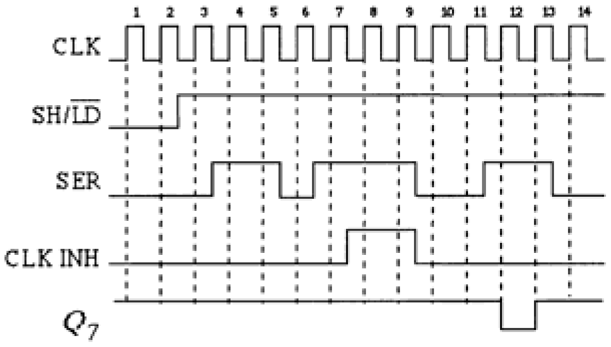


FIGURE 9-11

17. See Figure 9-12.

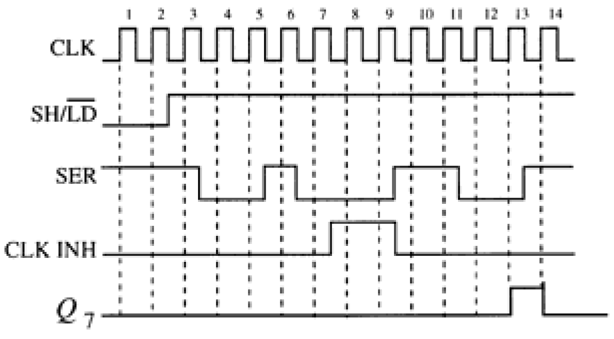


FIGURE 9-12

Section 9-5 Parallel In/Parallel Out Shift Registers

18. See Figure 9-13.

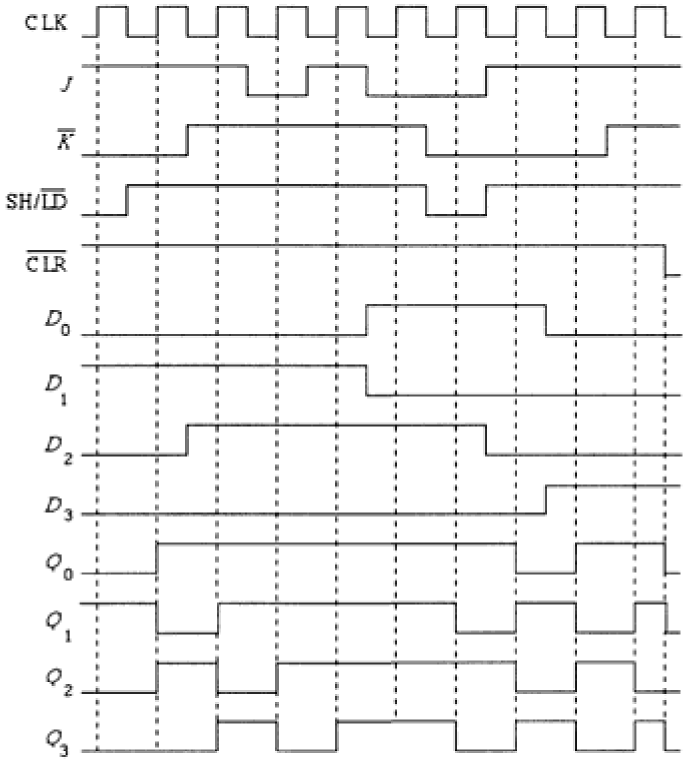


FIGURE 9-13

19. See Figure 9-14.

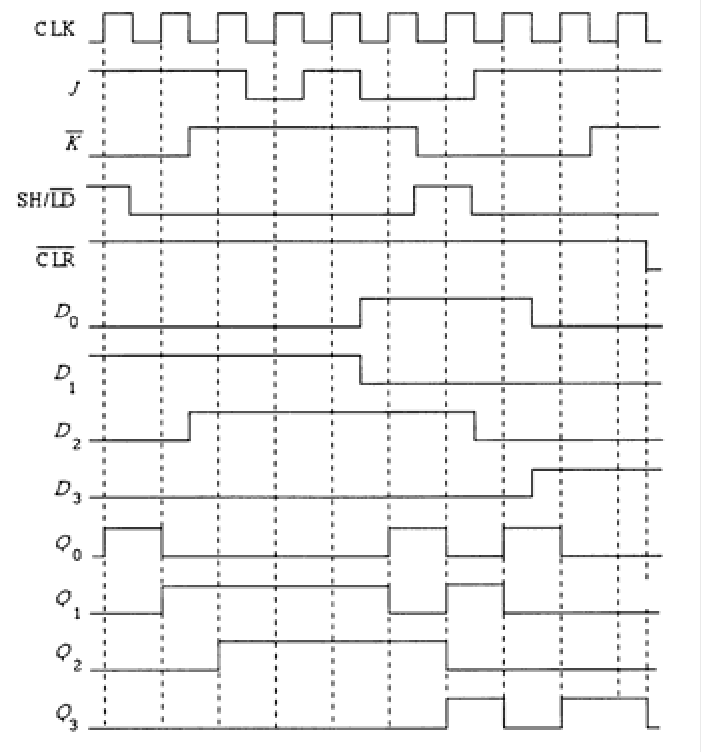


FIGURE 9-14

20. See Figure 9-15.

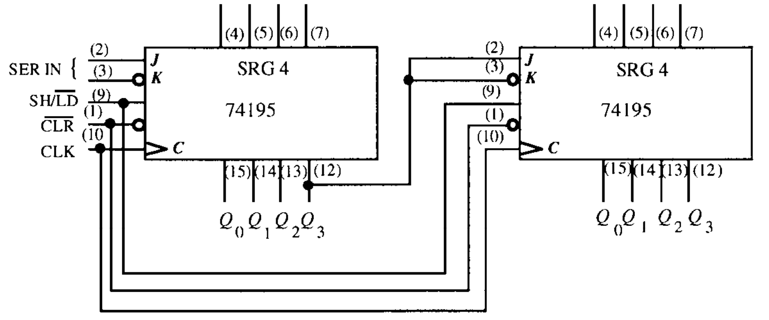


FIGURE 9-15

Section 9-6 Bidirectional Shift Registers

21.

Initially (76)	01001100	
CLK 1	10011000	Shift left
CLK 2	01001100	Shift right
CLK 3	00100110	Shift right
CLK 4	00010011	Shift right
CLK 5	00100110	Shift left
CLK 6	01001100	Shift left
CLK 7	00100110	Shift right
CLK 8	01001100	Shift left
CLK 9	00100110	Shift right
CLK 10	01001100	Shift left
CLK 11	10011000	Shift left

22.

Initially (76)	01001100	
CLK 1	00100110	Shift right
CLK 2	00010011	Shift right
CLK 3	00001001	Shift right
CLK 4	00010010	Shift left
CLK 5	00100100	Shift left
CLK 6	01001000	Shift left
CLK 7	00100100	Shift right
CLK 8	01001000	Shift left
CLK 9	10010000	Shift left
CLK 10	00100000	Shift left
CLK 11	00010000	Shift right
CLK 12	00001000	Shift right

23. See Figure 9-16.

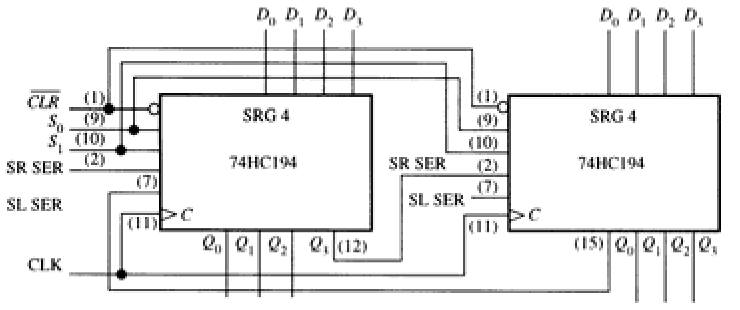


FIGURE 9-16

Chapter 9

24. See Figure 9-17.

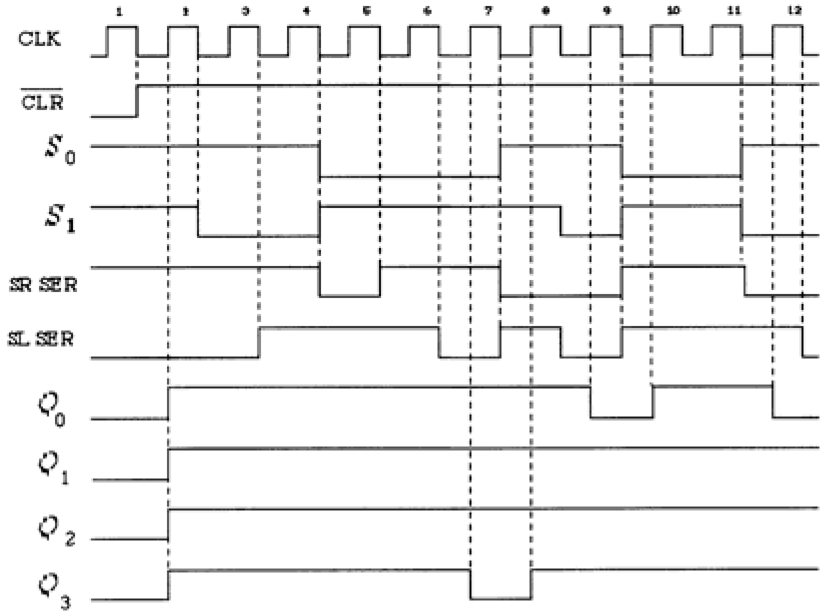


FIGURE 9-17

Section 9-7 Shift Register Counters

25. (a) $2n = 6$
$n =$ **3**

(b) $2n = 10$
$n =$ **5**

(c) $2n = 14$
$n =$ **7**

(d) $2n = 16$
$n =$ **8**

26. $2n = 18$; $n =$ **9** flip-flops

Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0
1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	0	0
1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1
0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1
0	0	0	0	0	0	1	1	1
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

See Figure 9-18.

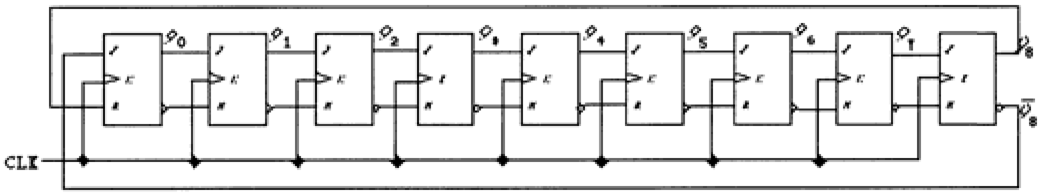


FIGURE 9-18

27. See Figure 9-19.

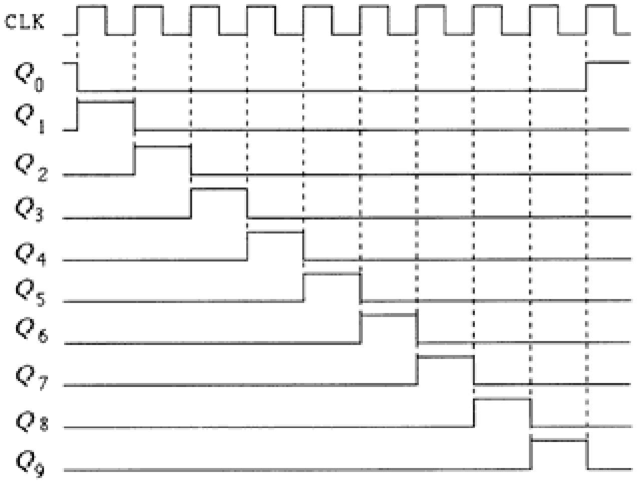


FIGURE 9-19

28. A 15-bit ring counter with stages **3**, **7**, and **12** SET and the remaining stages RESET. See Figure 9-20.

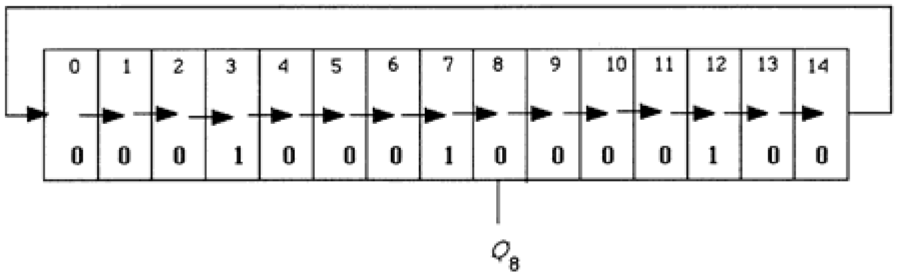


FIGURE 9-20

Chapter 9

Section 9-8 Shift Register Applications

29. See Figure 9-21.

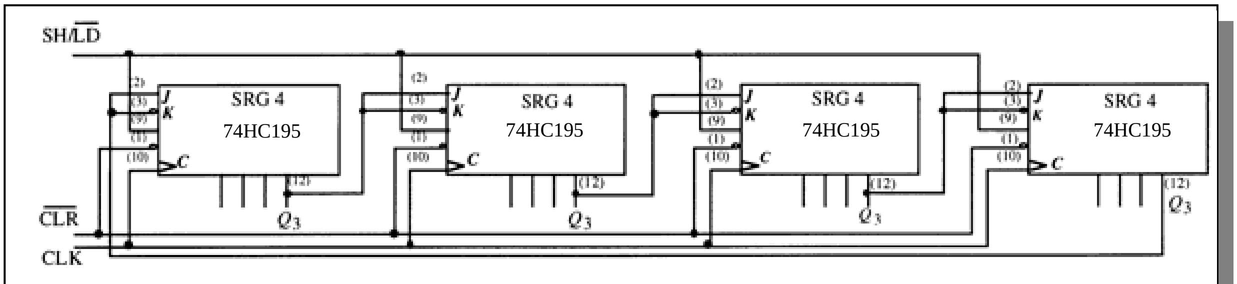


FIGURE 9-21

30. The power-on $\overline{\text{LOAD}}$ input provides a momentary LOW to parallel load the ring counter when power is turned on.

31. An incorrect code may be produced.

Section 9-10 Troubleshooting

32. Q_2 goes HIGH on the first clock pulse indicating that the *D* input is open. See Figure 9-22.

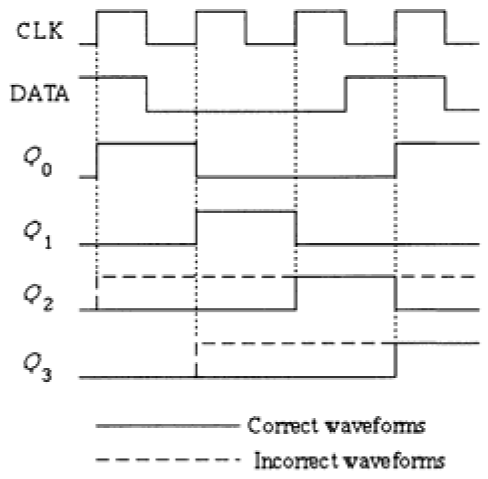


FIGURE 9-22

33. Since the LSB flip-flop works during serial shift, the problem is most likely in gate G3. An open D_3 input at G3 will cause the observed waveform. See Figure 9-23.

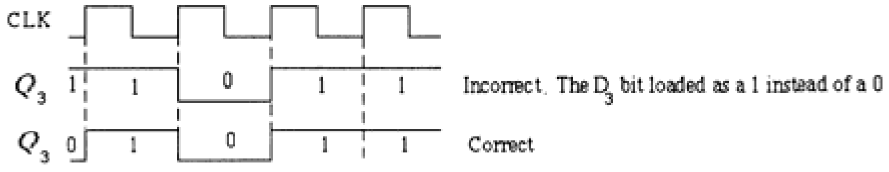


FIGURE 9-23

34. It takes a LOW on the RIGHT/LEFT input to shift data left. An open inverter input will keep the inverter output LOW thus disabling all of the shift-left control gates G5, G6, G7, and G8.

35. (a) No clock at switch closure due to faulty NAND gate or one-shot; open clock input to key code register; open $SH/\overline{LD}$ input to key code register.

(b) The diode in the third row is open; Q_2 output of ring counter is open.

(c) The NAND (negative-OR) gate input connected to the first column is shorted to ground or open, preventing a switch closure transition.

(d) The “2” input to the column encoder is open.

36. *1.* Number the switches in the matrix according to the following format:

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

2. Depress switches one at a time and observe the key code output according to the following Table 1.

Switch number	Key Code Register					
	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
1	0	1	1	0	1	1
2	0	1	1	1	0	1
3	0	1	1	0	0	1
4	0	1	1	1	1	0
5	0	1	1	0	1	0
6	0	1	1	1	0	0
7	0	1	1	0	0	0
8	0	1	1	1	1	1
9	1	0	1	0	1	1
10	1	0	1	1	0	1
11	1	0	1	0	0	1
12	1	0	1	1	1	0
13	1	0	1	0	1	0
14	1	0	1	1	0	0
15	1	0	1	0	0	0
16	1	0	1	1	1	1
17	0	0	1	0	1	1
18	0	0	1	1	0	1
19	0	0	1	0	0	1
20	0	0	1	1	1	0
21	0	0	1	0	1	0
22	0	0	1	1	0	0
23	0	0	1	0	0	0
24	0	0	1	1	1	1
25	1	1	0	0	1	1
26	1	1	0	1	0	1
27	1	1	0	0	0	1
28	1	1	0	1	1	0
29	1	1	0	0	1	0
30	1	1	0	1	0	0
31	1	1	0	0	0	0
32	1	1	0	1	1	1
33	0	1	0	0	1	1
34	0	1	0	1	0	1
35	0	1	0	0	0	1
36	0	1	0	1	1	0
37	0	1	0	0	1	0
38	0	1	0	1	0	0
39	0	1	0	0	0	0
40	0	1	0	1	1	1
41	1	0	0	0	1	1
42	1	0	0	1	0	1
43	1	0	0	0	0	1
44	1	0	0	1	1	0
45	1	0	0	0	1	0
46	1	0	0	1	0	0

47	1	0	0	0	0	0
48	1	0	0	1	1	1
49	0	0	0	0	1	1
50	0	0	0	1	0	1
51	0	0	0	0	0	1
52	0	0	0	1	1	0
53	0	0	0	0	1	0
54	0	0	0	1	0	0
55	0	0	0	0	0	0
56	0	0	0	1	1	1
57	1	1	1	0	1	1
58	1	1	1	1	0	1
59	1	1	1	0	0	1
60	1	1	1	1	1	0
61	1	1	1	0	1	0
62	1	1	1	1	0	0
63	1	1	1	0	0	0
64	1	1	1	1	1	1

TABLE 1

37. (a) Contents of Data Output Register remain constant.
(b) Contents of both registers do not change.
(c) Third stage output of Data Output Register remains HIGH.
(d) Clock generator is disabled after each pulse by the flip-flop being continuously SET and then RESET.

System Application Activity

38. The purpose of the Security Code logic is to accept a 4-digit code, compare it with a stored code, and if the codes match, to disarm the system for entry.

39. The states of shift registers A and C after two correct key closures are:

Shift Register A: 1001
Shift Register C: 00000100

40. The states of shift registers A and B after each key closure when entering 7645 are:

After key 7 is pressed:
Shift register A contains 0111
Shift register B contains 11000
After key 6 is pressed:
Shift register A contains 0110
Shift register B contains 11100
After key 4 is pressed:
Shift register A contains 0100
Shift register B contains 11110
After key 5 (an incorrect entry) is pressed:
Shift register A contains 0000
Shift register B contains 10000

Chapter 9

Special Design Problems

41. See Figure 9-24.

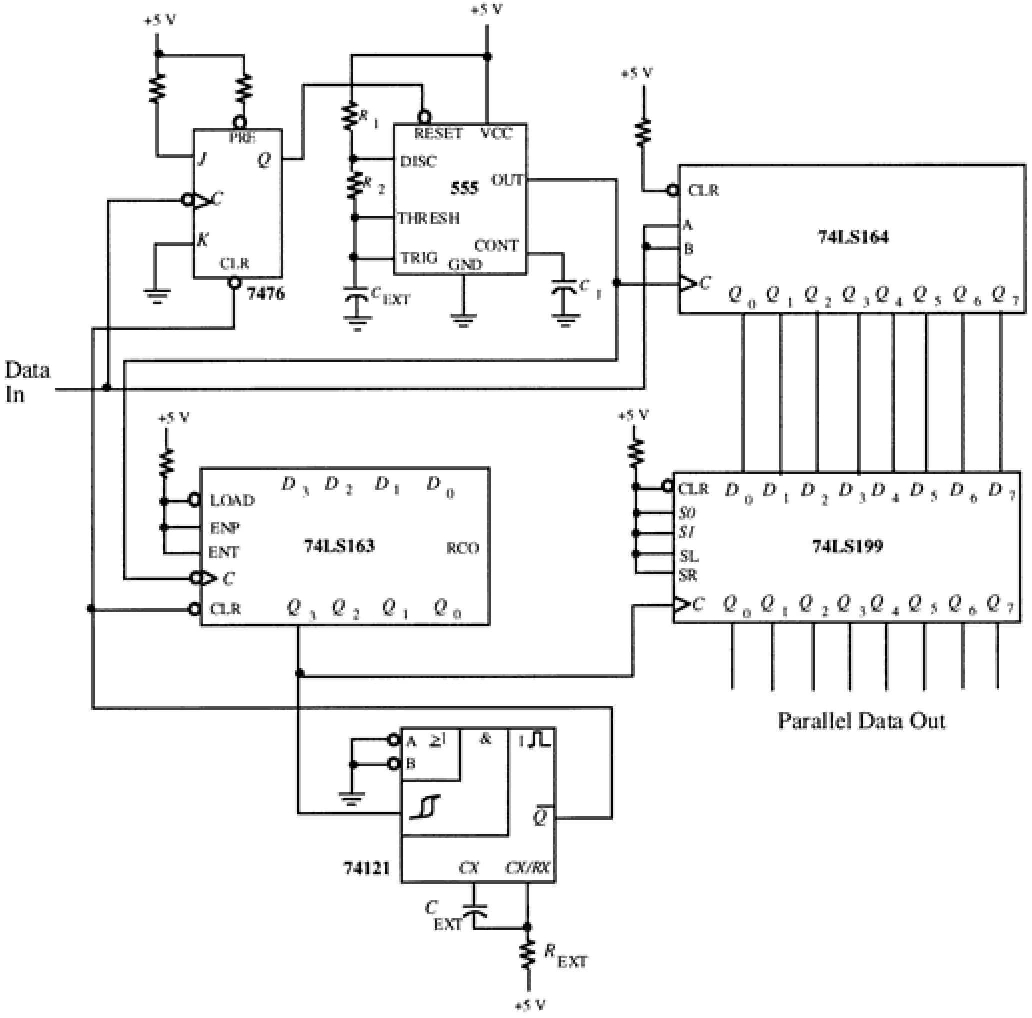


FIGURE 9-24

42. Figure 9-25 shows only the 74LS164, 74LS199, and 74LS163 portions of the circuit that require modification for 16-bit conversion.

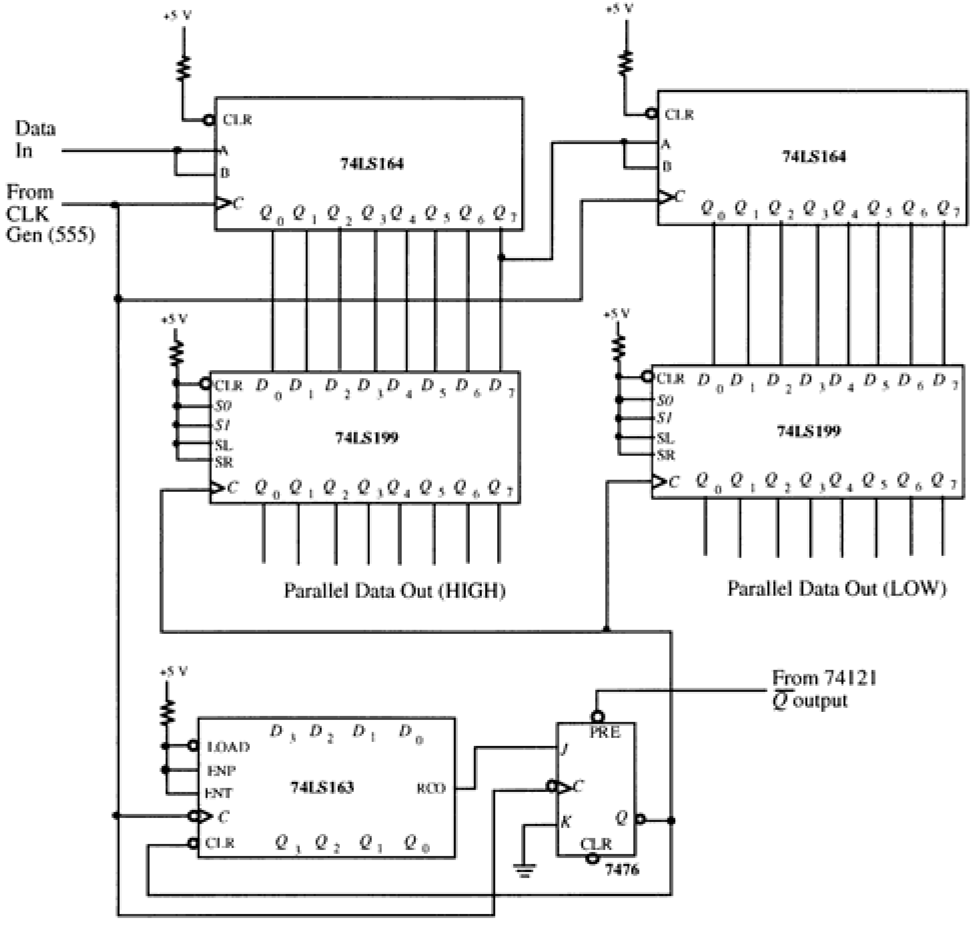

+5 V
CLR
Data In
A
B
From CLK Gen (555)
C
74LS164
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7
+5 V
CLR
A
B
C
74LS164
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7
+5 V
CLR
S0
S1
SL
SR
C
D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7
74LS199
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7
Parallel Data Out (HIGH)
+5 V
CLR
S0
S1
SL
SR
C
D_0 D_1 D_2 D_3 D_4 D_5 D_6 D_7
74LS199
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7
Parallel Data Out (LOW)
+5 V
LOAD
ENP
ENT
C
CLR
D_3 D_2 D_1 D_0
74LS163
RCO
Q_3 Q_2 Q_1 Q_0
From 74121 $\overline{Q}$ output
PRE
J
C
K
Q
CLR
7476

FIGURE 9-25

43. See Figure 9-26 for one possible implementation.

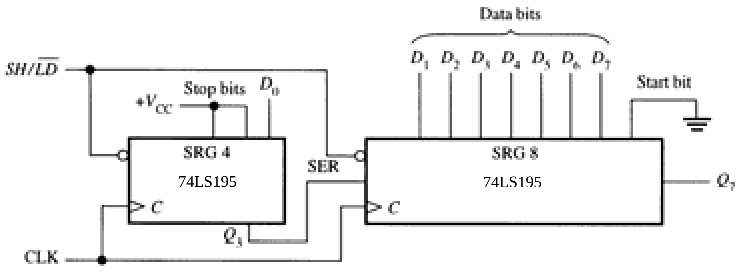
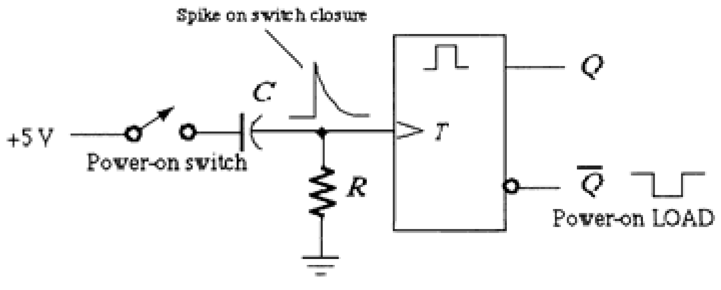


FIGURE 9-26

44. One possible approach is shown in Figure 9-27.



This is one way to implement a power-on LOAD circuit.

FIGURE 9-27

45. See Figure 9-28.

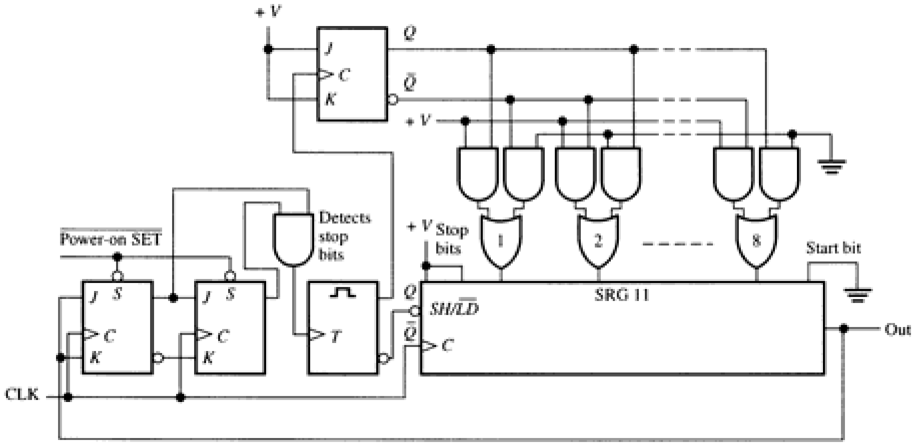


FIGURE 9-28

46. Register A requires 8 bits and can be implemented with one 74199. Register B requires 16 bits and can be implemented with two 74199s.

Multisim Troubleshooting Practice

47. CLK input of U3 open.

48. No fault.

49. Pin 14 open.

50. No fault.

51. CLK input of U6 open.

CHAPTER 10
MEMORY AND STORAGE

Section 10-1 Memory Basics

1. (a) ROM: no read/write control
(b) RAM

2. They are random access memories because any address can be accessed at any time. You do not have to go through all the preceding addresses to get to a specific address.

3. ***Address bus*** provides for transfer of address code to memory for accessing any memory location in any order for a read or a write operation.
Data bus provides for transfer of data between the microprocessor and memory or input/output devices.

4. (a) $0A_{16} = 00001010_2 = \mathbf{10_{10}}$
(b) $3F_{16} = 00111111_2 = \mathbf{63_{10}}$
(c) $CD_{16} = 11001101_2 = \mathbf{205_{10}}$

Section 10-2 The Random-Access Memory (RAM)

5.

	BIT 0	BIT 1	BIT 2	BIT 3
ROW 0	1	0	0	0
ROW 1	0	0	0	0
ROW 2	0	0	1	0
ROW 3	0	0	0	0

6. See Figure 10-1.

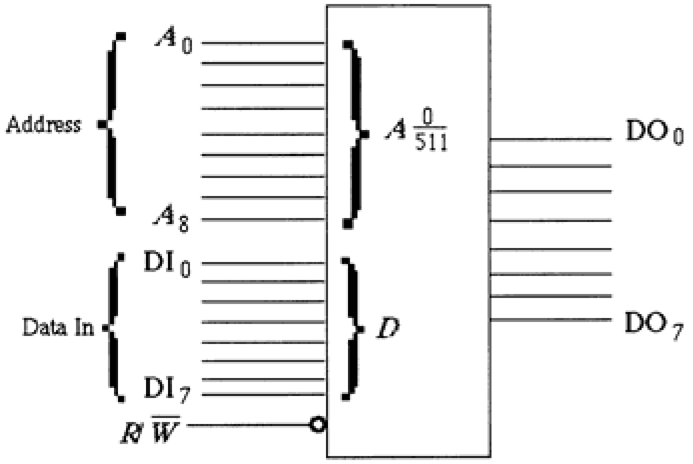


FIGURE 10-1

7. 64k × 8 = 512 × 128 × 8 = **512 rows × 128 8-bit columns**

8. See Figure 10-2.

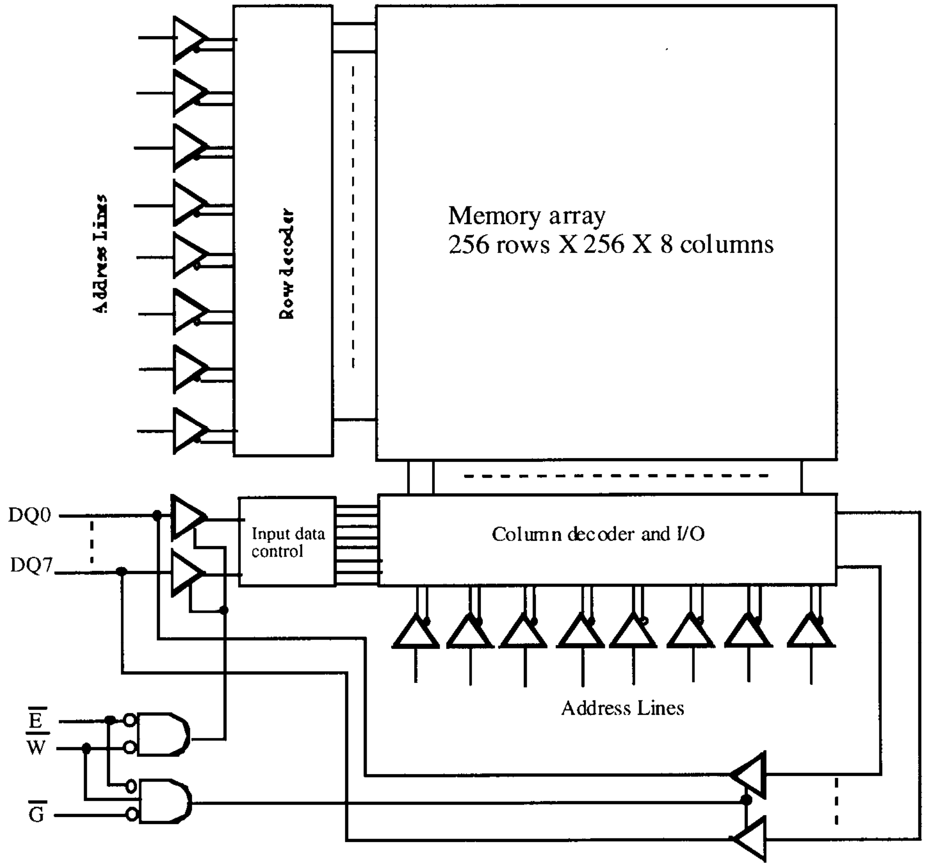


FIGURE 10-2

9. The difference between SRAM and DRAM is that data in a SRAM are stored in latches or flip-flops indefinitely as long as power is applied while data in a DRAM are stored in capacitors which require periodic refreshing to retain the stored data.

10. The bit capacity of a DRAM with 12 address lines is

$$2^{2 \times 12} = 2^{24} = 16{,}777{,}216 \text{ bits} = 16 \text{ Mbits}$$

Section 10-3 The Read-Only Memory (ROM)

11.

Inputs		**Outputs**			
A_1	A_0	O_3	O_2	O_1	O_0
0	0	0	1	0	1
0	1	1	0	0	1
1	0	1	1	1	0
1	1	0	0	1	0

12.

Inputs			**Outputs**			
A_2	A_1	A_0	O_3	O_2	O_1	O_0
0	0	0	0	1	0	0
0	0	1	1	1	1	1
0	1	0	1	0	1	1
0	1	1	1	0	0	1
1	0	0	1	1	1	0
1	0	1	1	0	0	0
1	1	0	0	0	1	1
1	1	1	0	1	0	1

13.

BCD				**Excess-3**			
D_3	D_2	D_1	D_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

See Figure 10-3.

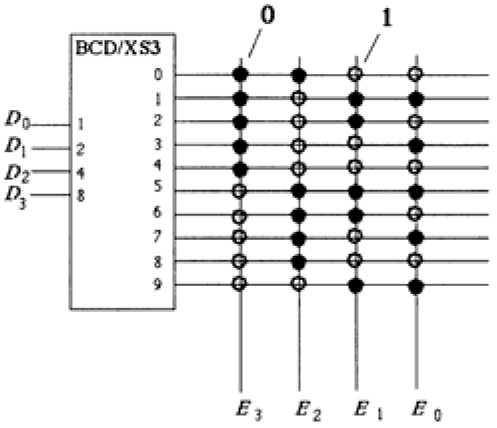


FIGURE 10-3

14. $2^{14} = 16{,}384$ addresses
$16{,}384 \times 8$ bits = **131,072 bits**

Chapter 10

Section 10-4 Programmable ROMs

15. **Blown links: 1 – 17, 19 – 23, 25 – 31, 34, 37, 38, 40 – 47, 53, 55, 58, 61, 62, 63, 65, 67, 69.**

	X Input				**X Output**								
	X_2	X_1	X_0	X^3	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	1
2	0	1	0	8	0	0	0	0	0	1	0	0	0
3	0	1	1	27	0	0	0	0	1	1	0	1	1
4	1	0	0	64	0	0	1	0	0	0	0	0	0
5	1	0	1	125	0	0	1	1	1	1	1	0	1
6	1	1	0	216	0	1	1	0	1	1	0	0	0
7	1	1	1	343	1	0	1	0	1	0	1	1	1

16.

Address	**Contents**
A_{13}--------------A_0	Q_7-------Q_0
01001100010011	10101100
11011101011010	00100101
01011010011001	10110011
11010010001110	00101000
01010010100101	10001011
01010000110100	11010101
01001001100001	11001001
11011011100100	01001001
01101110001111	01010010
10111110011010	01001000
10101110011010	11001000

Section 10-6 Memory Expansion

17. 16k × 4 DRAMS can be connected to make a 64k × 8 DRAM as shown in Figure 10-4.

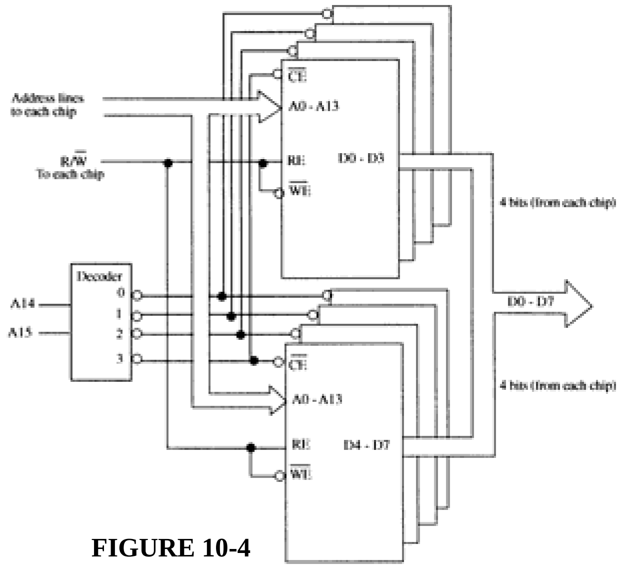


FIGURE 10-4

18. See Figure 10-5.

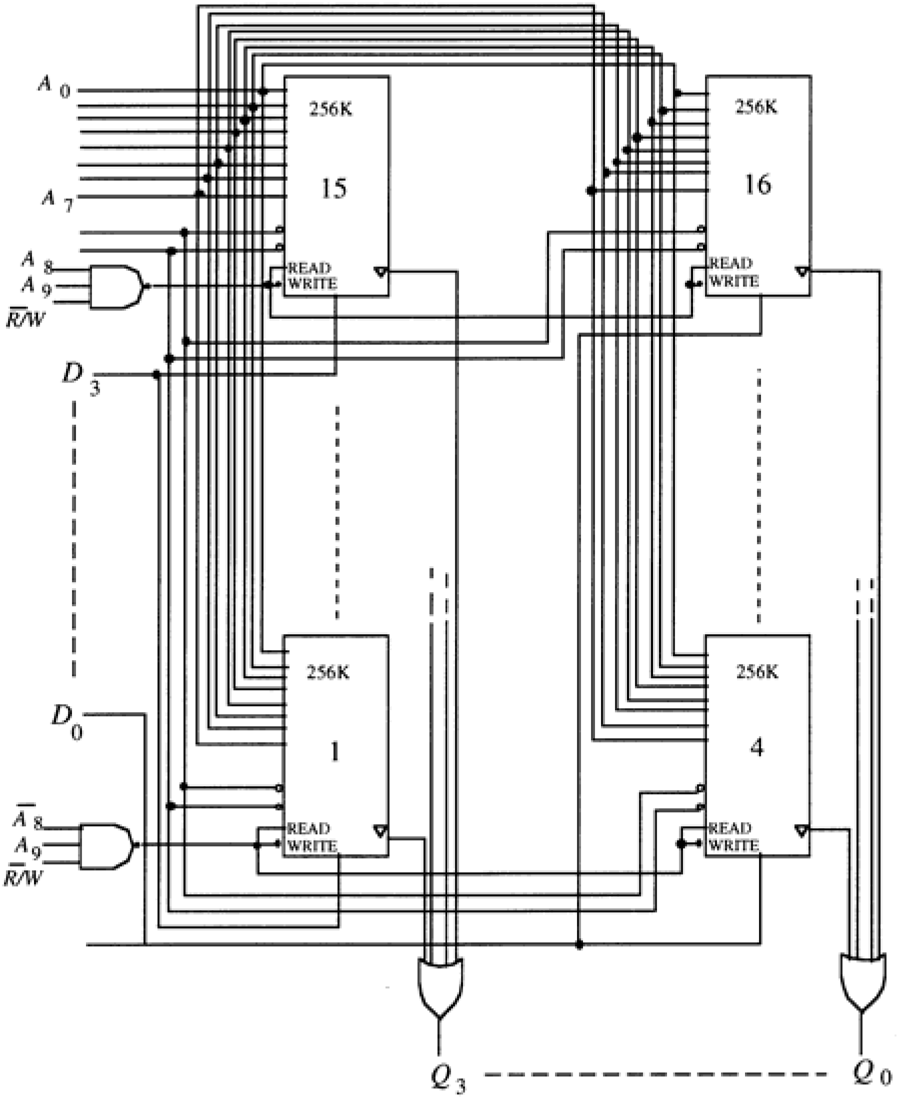


FIGURE 10-5

19. Word length = 8 bits, word capacity = **64k words**
Word length = 4 bits, word capacity = **256k words**

Section 10-7 Special Types of Memories

20. See Figure 10-6.

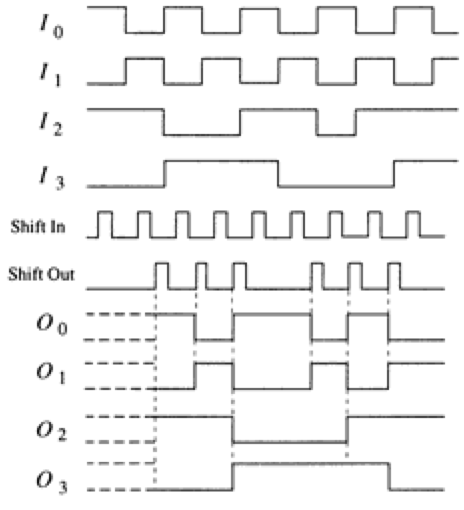


FIGURE 10-6

21. See Figure 10-7.

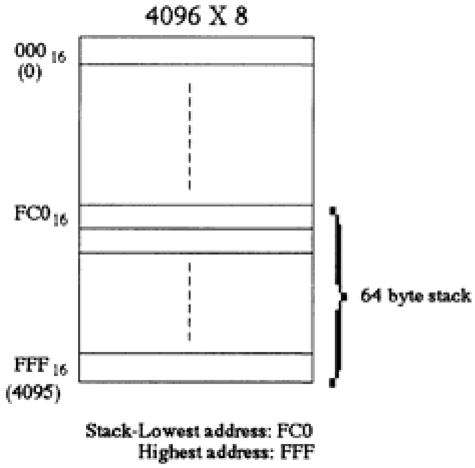


FIGURE 10-7

22. The first byte goes into **FFF**$_{16}$.
The last byte (16th) goes into a lower address: $16_{10} = 10_{16}$
$FFF_{16} - 10_{16} =$ **FEF**$_{16}$

See Figure 10-8.

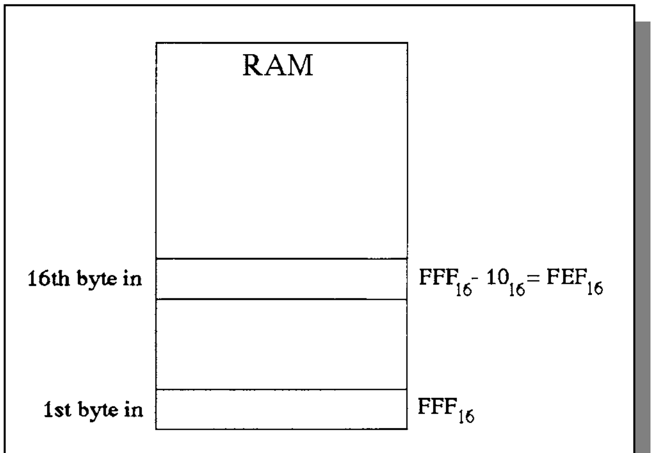


FIGURE 10-8

Section 10-8 Magnetic and Optical Storage

23. A hard disk is formatted into tracks and sectors. Each track is divided into a number of sectors with each sector of a track having a physical address. Hard disks typically have from a few hundred to a few thousand tracks.

24. Seek time is the average time required to position the drive head over the track containing the desired data. The latency period is the average time required for the data to move under the drive head.

25. Magnetic tape has a longer access time than disk because data must be accessed sequentially rather than randomly.

26. A magneto-optic disk is a read/write medium using lasers and magnetic fields.
A CD-ROM (compact-disk ROM) is a read-only optical (laser) medium.
A WORM (write-once-read-many) is an optical medium in which data can be written once and read many times.

Section 10-9 Troubleshooting

27. The correct checksum is **00100**.
The actual checksum is 01100. The second bit from the left is in error.

28. (a)

ROM0:	Low address - 00_{16}	High address - $1F_{16}$
ROM1:	Low address - 20_{16}	High address - $3F_{16}$
ROM2:	Low address - 40_{16}	High address - $5F_{16}$
ROM3:	Low address - 60_{16}	High address - $7F_{16}$

(b) Same as flow chart in Figure 10-68 in text except that the last data address is specified as **7E**$_{16}$ (7F$_{16}$ – 1).

(c) See Figure 10-9.

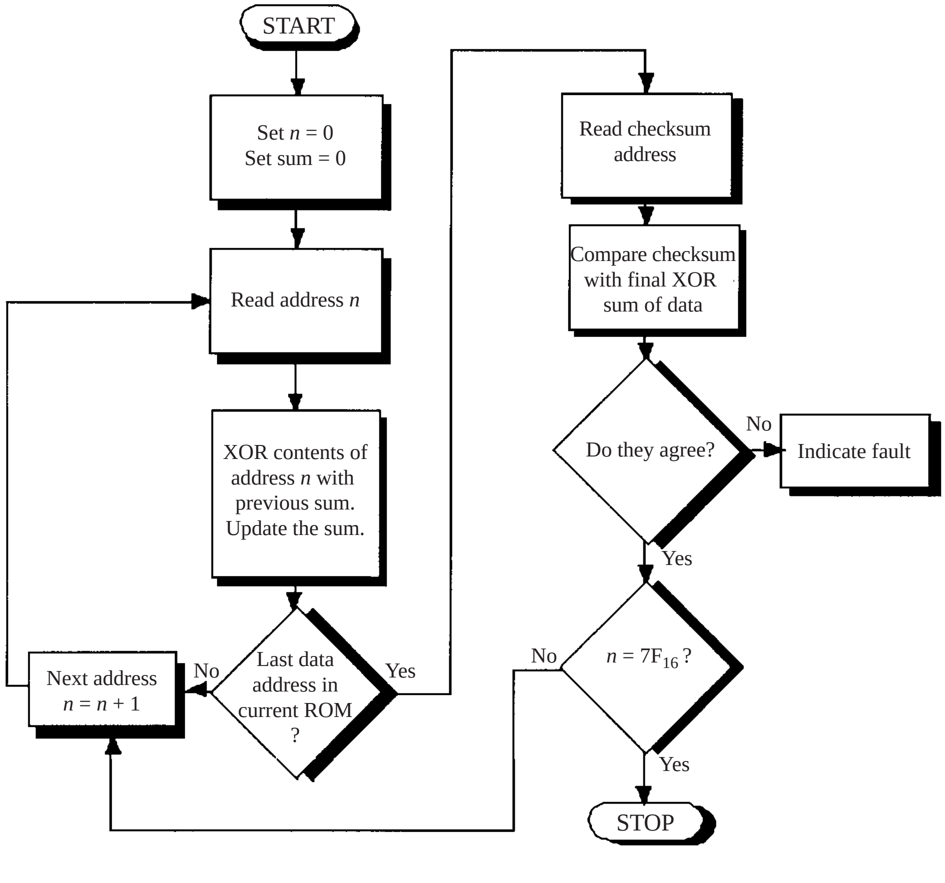


FIGURE 10-9

(d) A single checksum will not isolate the faulty chip. It will only indicate that there is an error in one of the chips.

29. (a) $40_{16} – 5F_{16}$ is 64 – 95 decimal; ROM 2
(b) $20_{16} – 3F_{16}$ is 32 – 63 decimal; ROM 1
(c) $00_{16} – 7F_{16}$ is 0 – 127 decimal; All ROMs

System Application Activity

30. See Figure 10-10.

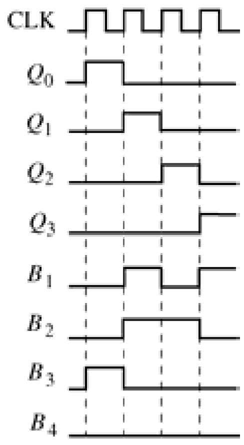


FIGURE 10-10

31. On first digit entry, the register state is 0001. On second digit entry the register state is 0010.

32. The purpose of the switch memory is to store a 4-digit security code and permit easy code change.

33. A new code can be entered by simply moving the DIP switch settings. No power is required.

Special Design Problems

34. NAND gates U1A-U4D: four 74HC00
NAND gates U5A-U6B: two 74HC20
Shift register U7: 74HC195

35. See Figure 10-11.

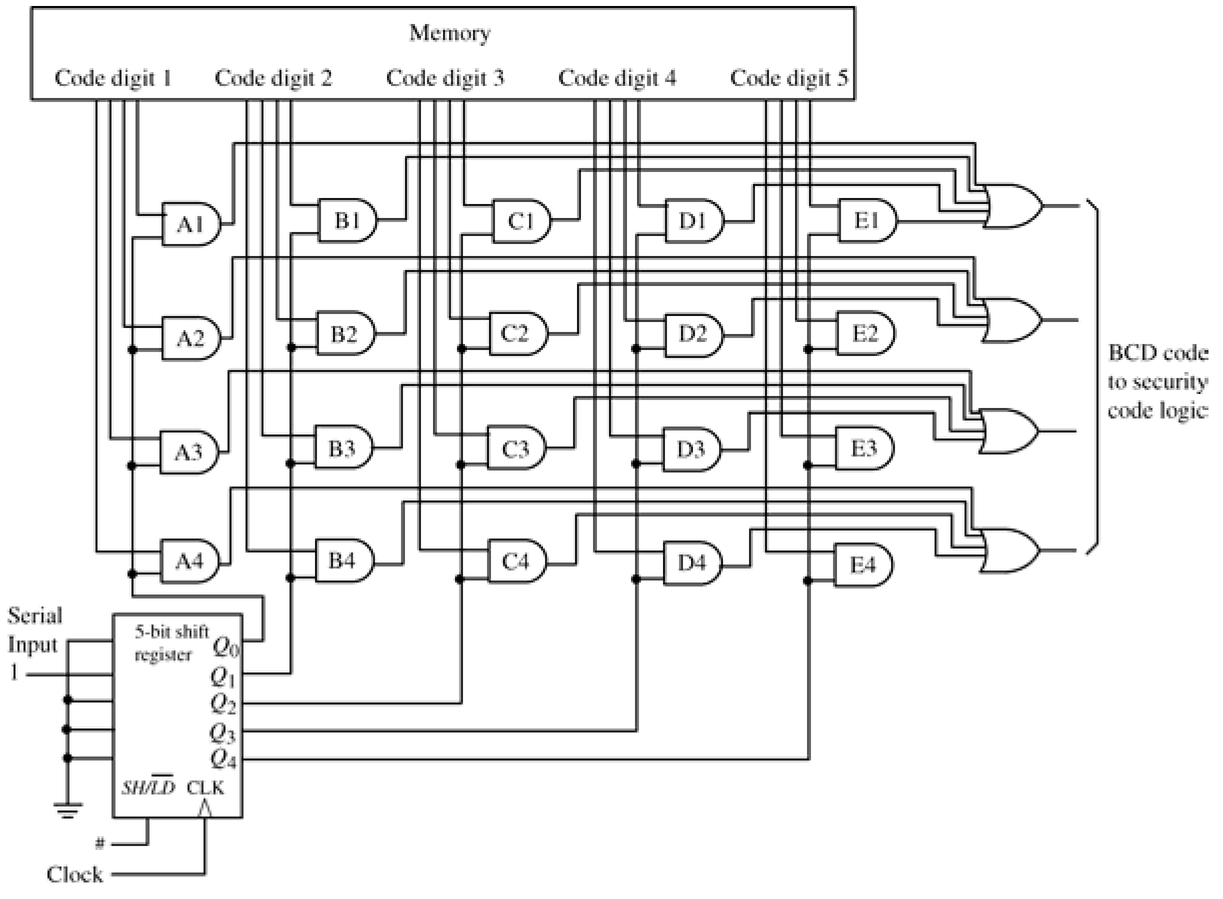
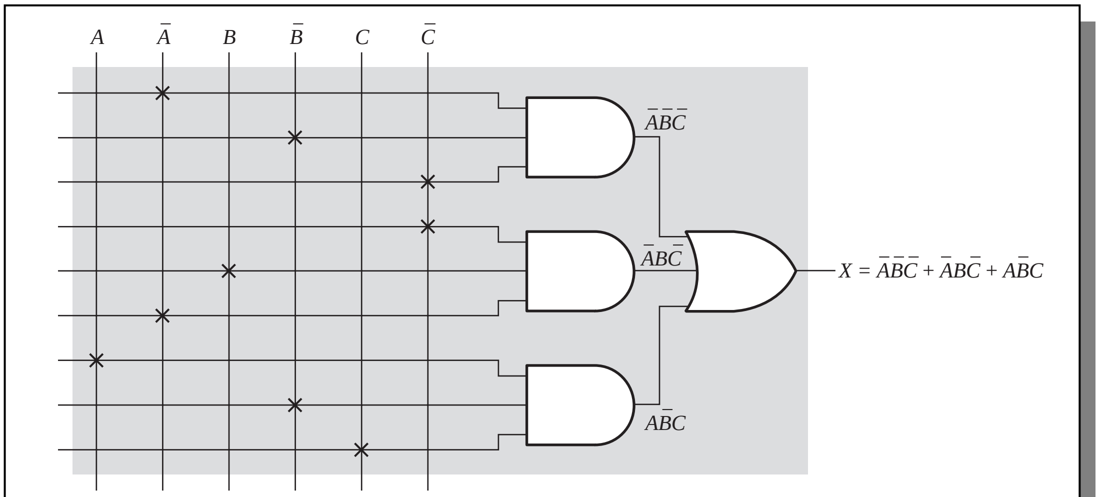


FIGURE 10-11

CHAPTER 11
PROGRAMMABLE LOGIC AND SOFTWARE

Section 11-1 Programmable Logic: SPLDs and CPLDs

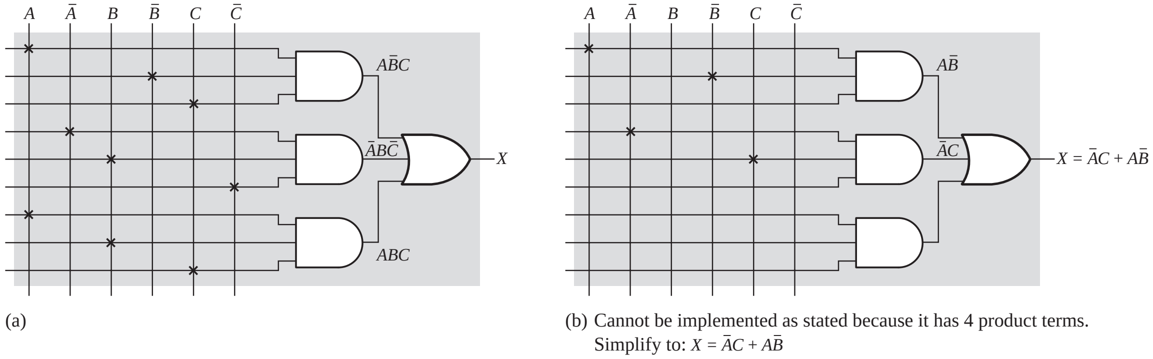
1. $X = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}B\overline{C} + A\overline{B}C$. See Figure 11-1.



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

FIGURE 11-1

2. See Figure 11-2.



(b) Cannot be implemented as stated because it has 4 product terms. Simplify to: $X = \overline{A}C + A\overline{B}$

FIGURE 11-2

3. (a) PAL16L2 is a programmable array logic device with 16 inputs and two active-LOW outputs.

(b) PAL12H6 is a programmable array logic device with 12 inputs and 6 active-HIGH outputs.

Chapter 11

4. Typically, an exclusive-OR gate is used to determine the polarity of the output. When a 1 is applied to one input of the XOR gate, the output of the XOR is the complement of the signal on the other input. When a 0 is applied to one input of the XOR, the signal on the output of the XOR is the same as the signal on the other input.

5. A CPLD basically consists of multiple SPLDs that can be connected with a programmable interconnect array.

Section 11-2 Altera CPLDs

6. (a) Inputs from PIA to LAB: **36** (b) Outputs from LAB to PIA: **16**
(c) Inputs from I/O to PIA: **8 to 16** (d) Outputs from LAB to I/O: **8 to 16**

7. (a) $\overline{A}BC\overline{D}$ (b) $ABC(\overline{DE}) = ABC(\overline{D}+\overline{E}) = ABC\overline{D} + ABC\overline{E}$

8. $A\overline{B}C\overline{D} + EFGH + AB\overline{C}D + \overline{A}BCD$

Section 11-3 Xilinx CPLDs

9. $A\overline{B} + \overline{A}B$

10. (a) Inputs from AIM to FB: **40** (b) Outputs from FB to AIM: **16**
(c) Inputs from I/O to AIM: **16** (d) Outputs from FB to I/O: **16**

11. $X_1 = A\overline{B}C\overline{D} + \overline{A}BCD + ABC\overline{D}$; $X_2 = ABCD + AB\overline{C}D + \overline{A}BC\overline{D} + A\overline{B}CD$

Section 11-4 Macrocells

12. (a) A 0 on the select line selects D_0. The output is **1**.

(b) A 1 on the select line selects D_1. The output is **0**.

13. (a) Since the D_0 (upper) input of MUX 5 is selected, the macrocell is configured for **combinational** logic**.** The output of the XOR goes through MUX 5 to the "To I/O" output making it a **1**.

(b) Since the D_1 (lower) input of MUX 5 is selected, the macrocell is configured for **registered** logic**.** The output of the flip-flop goes through MUX 5 to the "To I/O" output making it a **0**.

14. (a) The macrocell is configured for **registered** logic because the D_1 input of MUX 8 is selected, allowing the flip-flop output to pass through.

(b) The **GCK1** clock is applied to the flip-flop because the D_1 input of MUX 3 and the D_1 input of MUX 5 are selected.

(c) The OR gate output is applied to the XOR which is set for noninversion by MUX 1. The output of the XOR is selected by MUX2 and a **1** is applied to the D/T input of the flip-flop.

(d) The output of MUX 8 is a **1** because MUX 8 selects the Q output of the flip-flop (assuming that the S and R inputs are 0).

15. (a) The macrocell is configured for **registered** logic because the D_1 input of MUX 8 is selected, allowing the flip-flop output to pass through.

(b) The **GCK1** clock is applied to the flip-flop because the D_1 input of MUX 3 and the D_1 input of MUX 5 are selected.

(c) The OR gate output is applied to the XOR which is set for inversion by MUX 1. The output of the XOR is selected by MUX 2 and a **0** is applied to the D/T input of the flip-flop.

(d) The output of MUX 8 is a **0** because MUX 8 selects the Q output of the flip-flop (assuming that the S and R inputs are 0).

Section 11-5 Programmable Logic: FPGAs

16. An FPGA typically consists of configurable logic blocks (CLBs). Each CLB is made up of a number of logic modules with a local interconnect. Each logic module typically consists of a look-up table (LUT) and associated logic. Global column and row interconnects are used to connect the CLBs to I/Os as well as each other.

17. SOP output = $\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + AB\bar{C}$

18. See Figure 11-3.

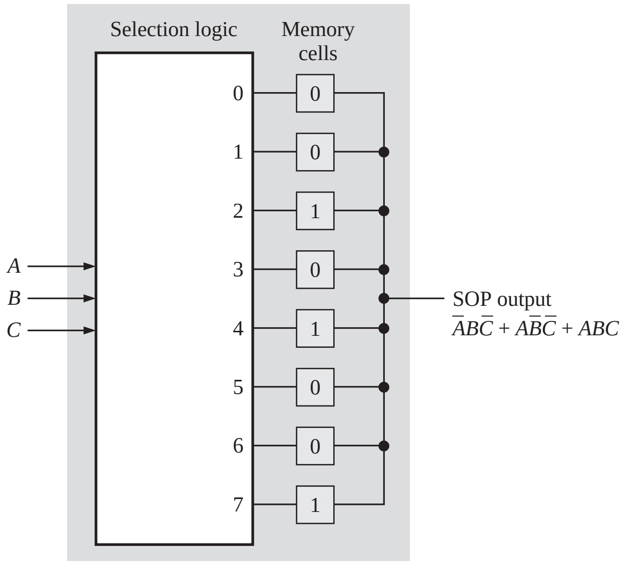


FIGURE 11-3

Chapter 11

Section 11-6 Altera FPGAs

19. An ALM consists of an LUT for combinational logic, adder logic, and register logic.

20. The modes of operation of an ALM are: Normal, Extended LUT, Arithmetic, and Shared arithmetic.

21. See Figure 11-4.

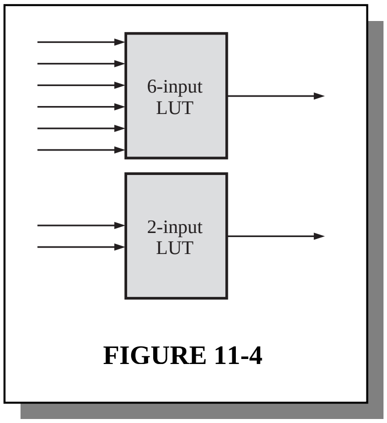


FIGURE 11-4

22. $(A_4A_3\overline{A}_2A_1 + \overline{A}_4\overline{A}_3\overline{A}_2A_1)A_0 + (\overline{A}_5A_3A_2A_1 + A_5\overline{A}_3A_2\overline{A}_1 + A_5A_3A_2\overline{A}_1)A_0$

$= A_4A_3\overline{A}_2A_1A_0 + \overline{A}_4\overline{A}_3\overline{A}_2A_1A_0 + \overline{A}_5A_3A_2A_1\overline{A}_0 + A_5\overline{A}_3A_2\overline{A}_1\overline{A}_0 + A_5A_3A_2\overline{A}_1\overline{A}_0$

Section 11-7 Xilinx FPGAs

23. See Figure 11-5.

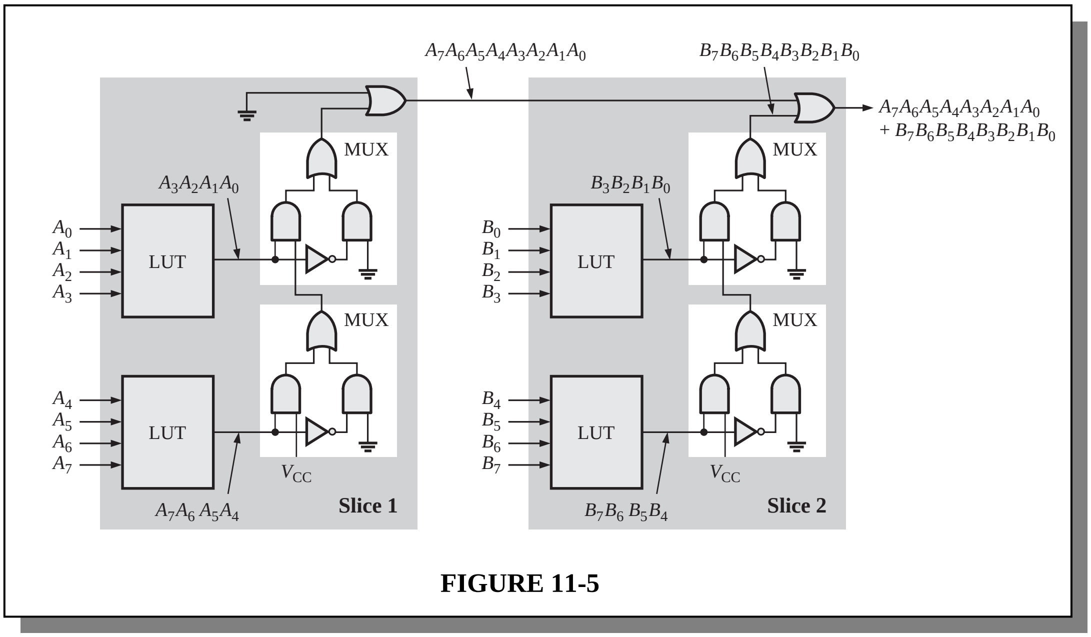


FIGURE 11-5

24. See Figure 11-6.

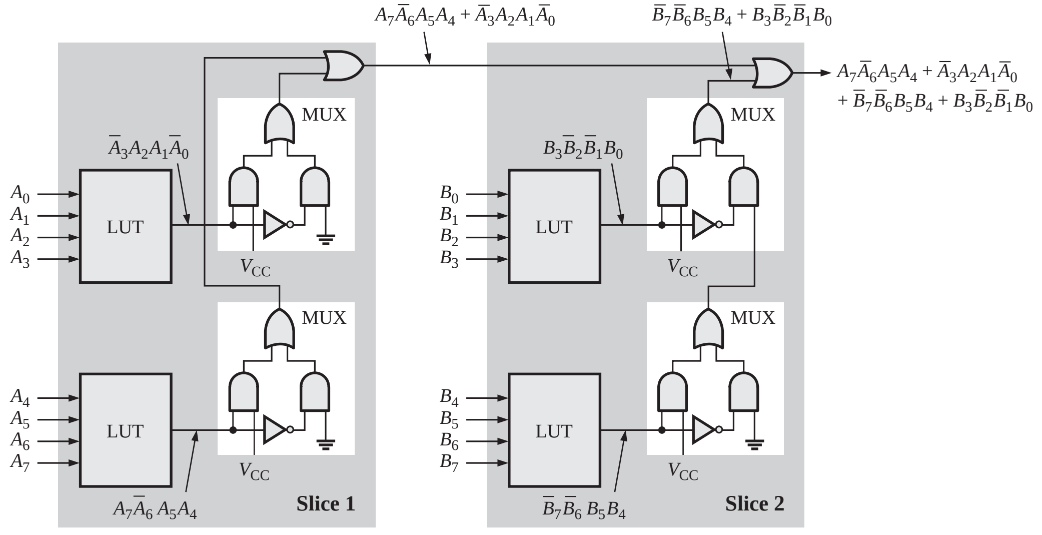

$A_7\bar{A}_6A_5A_4 + \bar{A}_3A_2A_1\bar{A}_0$
$\bar{B}_7\bar{B}_6B_5B_4 + B_3\bar{B}_2\bar{B}_1B_0$
$A_7\bar{A}_6A_5A_4 + \bar{A}_3A_2A_1\bar{A}_0$
$+ \bar{B}_7\bar{B}_6B_5B_4 + B_3\bar{B}_2\bar{B}_1B_0$
MUX
$\bar{A}_3A_2A_1\bar{A}_0$
A_0
A_1
A_2
A_3
LUT
V_{CC}
MUX
A_4
A_5
A_6
A_7
LUT
V_{CC}
$A_7\bar{A}_6\ A_5A_4$
Slice 1
MUX
$B_3\bar{B}_2\bar{B}_1B_0$
B_0
B_1
B_2
B_3
LUT
V_{CC}
MUX
B_4
B_5
B_6
B_7
LUT
V_{CC}
$\bar{B}_7\bar{B}_6\ B_5B_4$
Slice 2

FIGURE 11-6

25. One slice. See Figure 11-7.

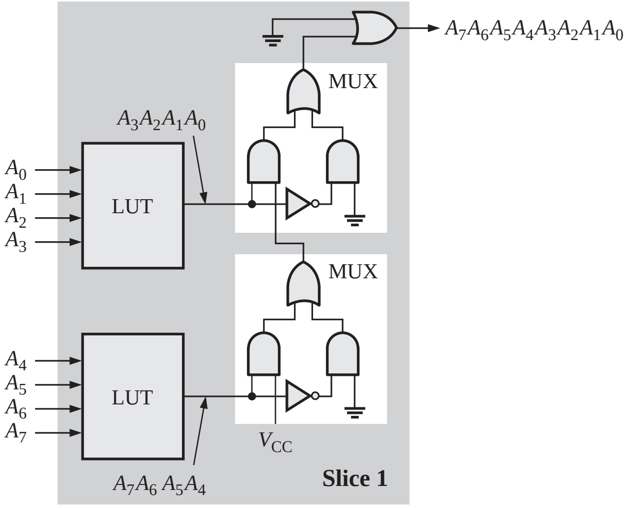


FIGURE 11-7

Chapter 11

26. Three slices are required. See Figure 11-8.

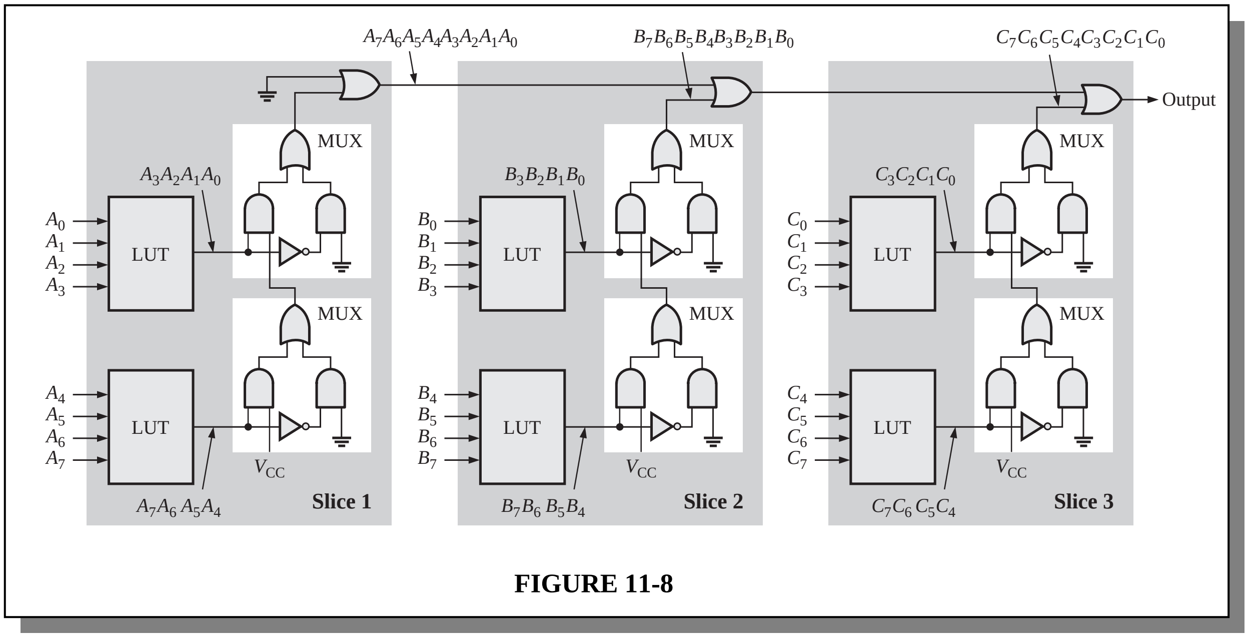


FIGURE 11-8

Section 11-8 Programmable Logic Software

27. See Figure 11-9.

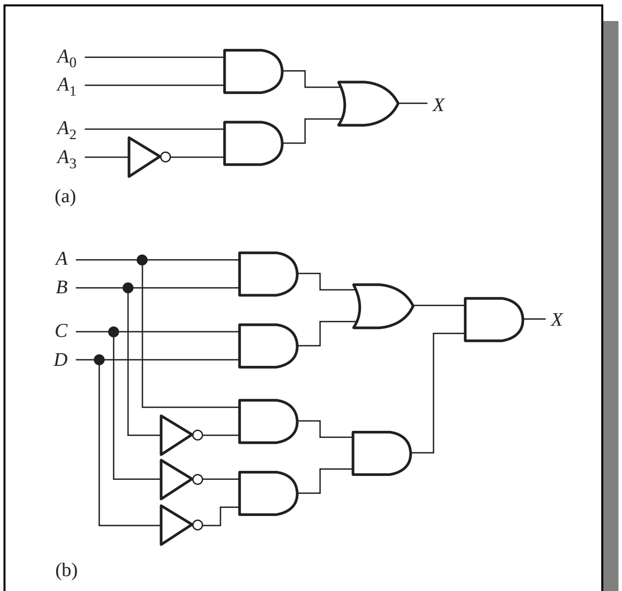


FIGURE 11-9

28. $X = \overline{A}BCD + A\overline{B}CD + AB\overline{C}D + ABC\overline{D} + ABCD + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$

$= ABD + ACD + ABC + BCD + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$

See Figure 11-10.

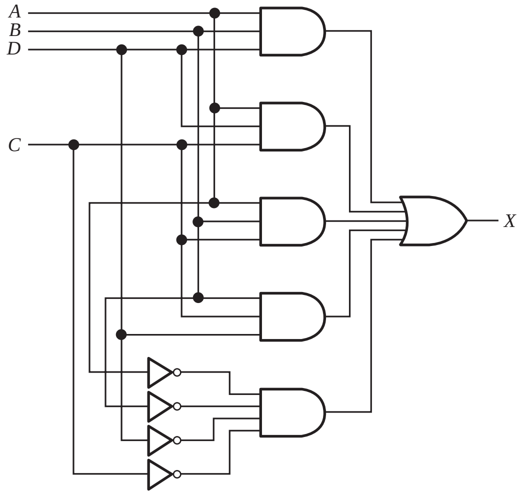


FIGURE 11-10

29. See Figure 11-11.

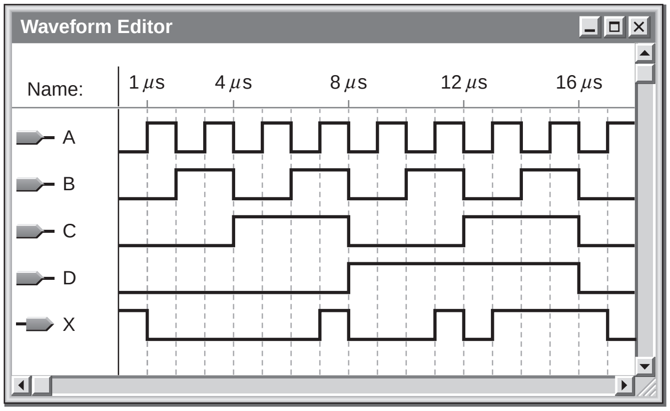


FIGURE 11-11

30. $X = \overline{A}BC\overline{D} + A\overline{B}\,\overline{C}D + ABCD + A\overline{B}C\overline{D} + \overline{A}B\overline{C}D$. See Figure 11-12.

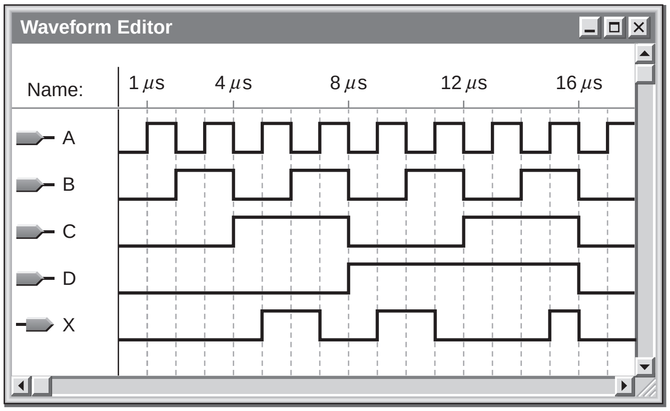


FIGURE 11-12

Section 11-9 Boundary Scan Logic

31. The Shift input = 1, data are applied to SDI, go through the MUX, and are clocked into Capture register A on the leading edge of the clock pulse. From the output of Capture register A, the data go through the upper MUX and are clock into Capture register B on the trailing edge of the clock pulse.

32. PDI/O = 0 and OE = 1. The data from the internal programmable logic pass through the selected MUX and through the output buffer to the pin.

33. PDI/O = 0 and OE = 0. The data are applied to the input pin and go through the selected MUX to the internal programmable logic.

34. SHIFT = 1, PDI/O = 1, and OE = 0. Data are applied to SDI, go through the MUX, and are clocked into Capture register A on the leading edge of the clock pulse. From the output of Capture register A, the data go through the upper MUX and are clocked into Capture register B on the trailing edge of the clock pulse. A pulse on the UPDATE input clocks the data into Update register B. The data on the output of Capture Register B go through the MUX to the internal programmable logic. The data also appear on the SDO.

Section 11-10 Troubleshooting

35. 000011001010001111011 shifted from TDI to TDO, left-most bit first. The bold-faced code will appear on the logic inputs in the sequence shown.

0	**0000**11001010001111011
1	0**0001**1001010001111011
3	00**0011**001010001111011
6	000**0110**01010001111011
12	0000**1100**1010001111011
9	00001**1001**010001111011
2	000011**0010**10001111011
5	0000110**0101**0001111011
10	00001100**1010**001111011
4	000011001**0100**01111011
8	0000110010**1000**1111011
1	00001100101**0001**111011
3	000011001010**0011**11011
7	0000110010100**0111**1011
15	00001100101000**1111**011
14	000011001010001**1110**11
13	0000110010100011**1101**1
11	00001100101000111**1011**

System Application Activity

36. 11 inverters can be eliminated. Only four are needed to produce the complements of *A*, *B*, *C*, and *D*.

There are three AND gates that produce the product term $\overline{A}\,\overline{C}$. Two can be eliminated.

There are three AND gates that produce the product term $\overline{A}B$. Two can be eliminated.

There are two AND gates that produce the product term $B\overline{C}$. One can be eliminated.

There are two AND gates that produce the product term $\overline{B}C$. One can be eliminated.

There are two AND gates that produce the product term $\overline{A}\,\overline{B}$. One can be eliminated.

7 AND gates can be eliminated.

37. The D input to the logic is faulty or not connected. See Figure 11-13.

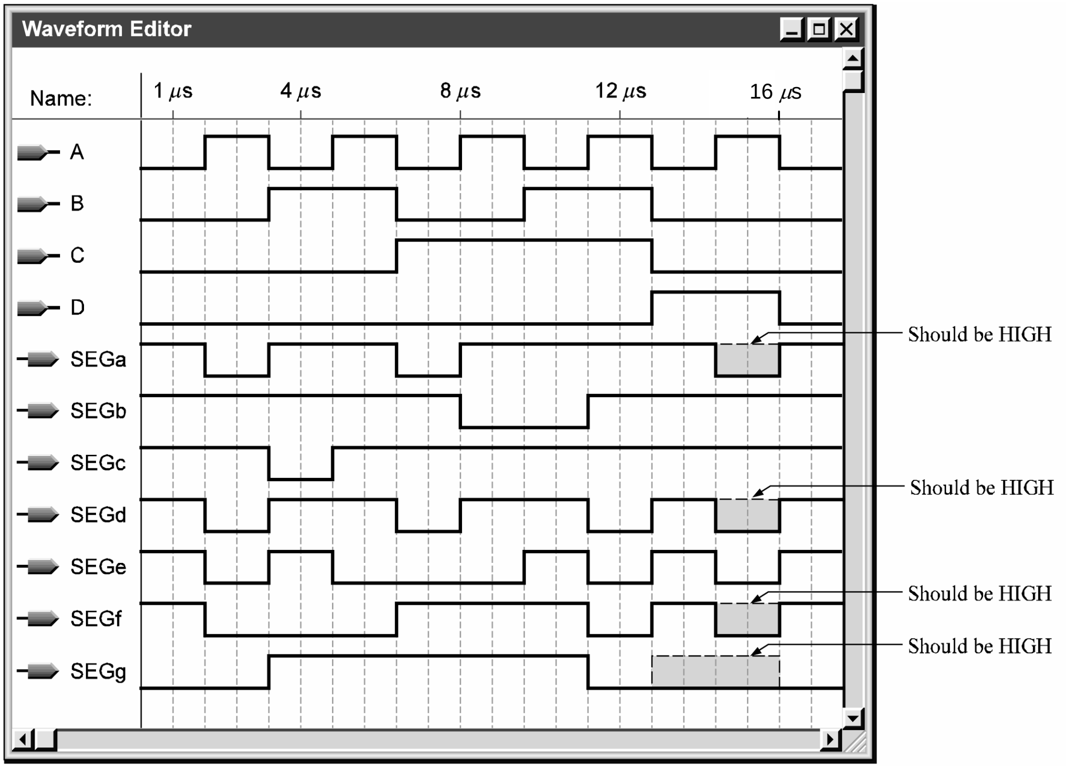


FIGURE 11-13

CHAPTER 12
SIGNAL INTERFACING AND PROCESSING

Section 12-1 Converting Analog Signals to Digital

1. See Figure 12-1.

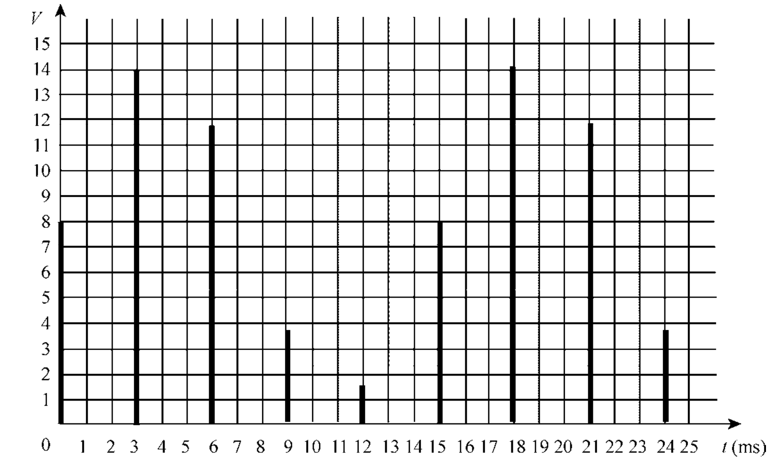


FIGURE 12-1

2. See Figure 12-2.

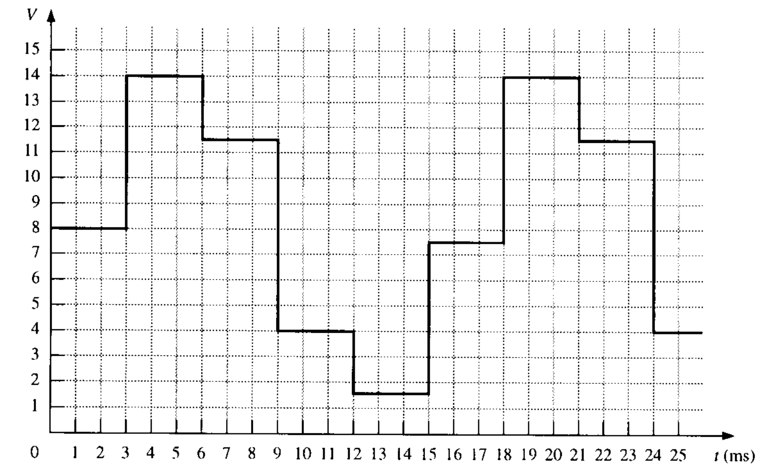
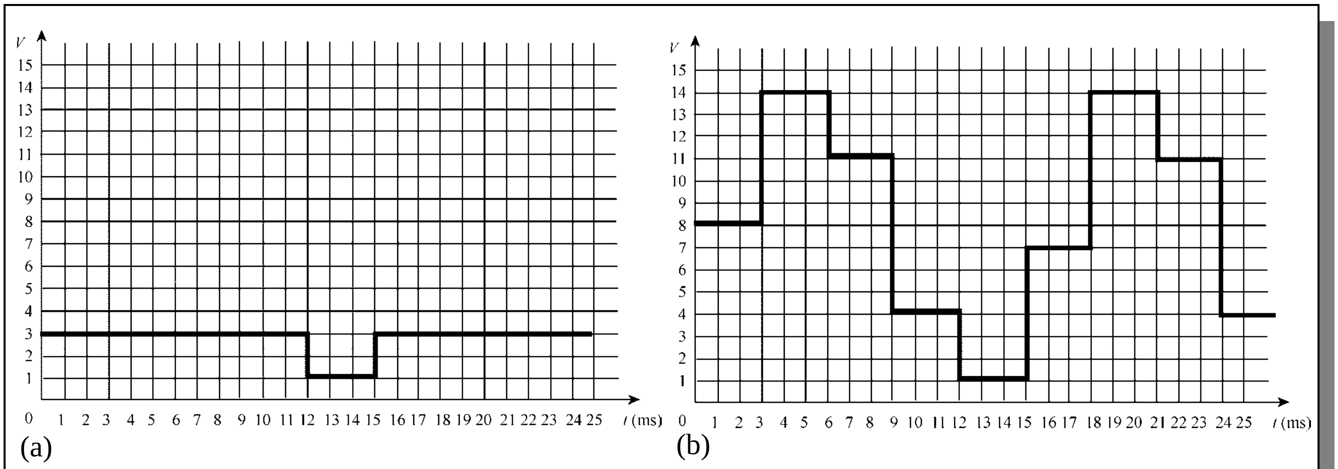


FIGURE 12-2

3. 11, 11, 11, 11, 01, 11, 11, 11, 11

4. 1000, 1110, 1011, 0100, 0001, 0111, 1110, 1011, 0100

5. See Figure 12-3.



(a) (b)

FIGURE 12-3

6. See Figure 12-4.

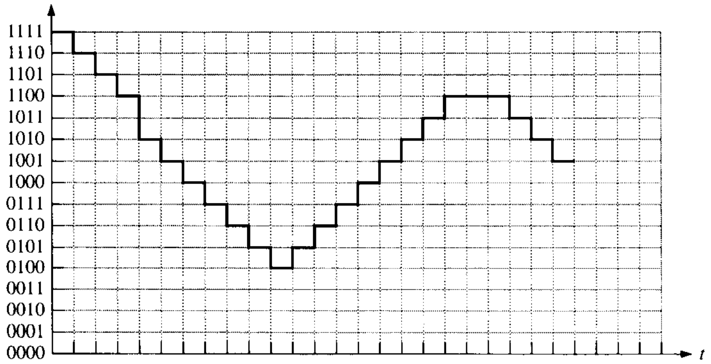


FIGURE 12-4

Section 12-2 Analog-to-Digital Conversion Methods

7 $\frac{V_{\text{out}}}{V_{\text{in}}} = \frac{2\text{ V}}{10\text{ mV}} = 200$

8. $$\frac{V_{\text{OUT}}}{V_{\text{IN}}} = \frac{R_{\text{F}}}{R_{\text{IN}}}$$

$$R_{\text{F}} = R_{\text{in}}\left(\frac{V_{\text{OUT}}}{V_{\text{IN}}}\right) = 1\text{ k}\Omega(330) = \mathbf{330\text{ k}\Omega}$$

9. $A_v = -\frac{R_f}{R_i} = -\frac{47\text{ k}\Omega}{2.2\text{ k}\Omega} = \mathbf{-21.4}$

10. Number of comparators = $2^n - 1 = 2^8 - 1$ = **255**

11. 001, 010, 011, 101, 110, 111, 111, 111, 111, 110, 101, 101, 110, 110, 110, 101, 100, 011, 010, 001.

See Figure 12-5.

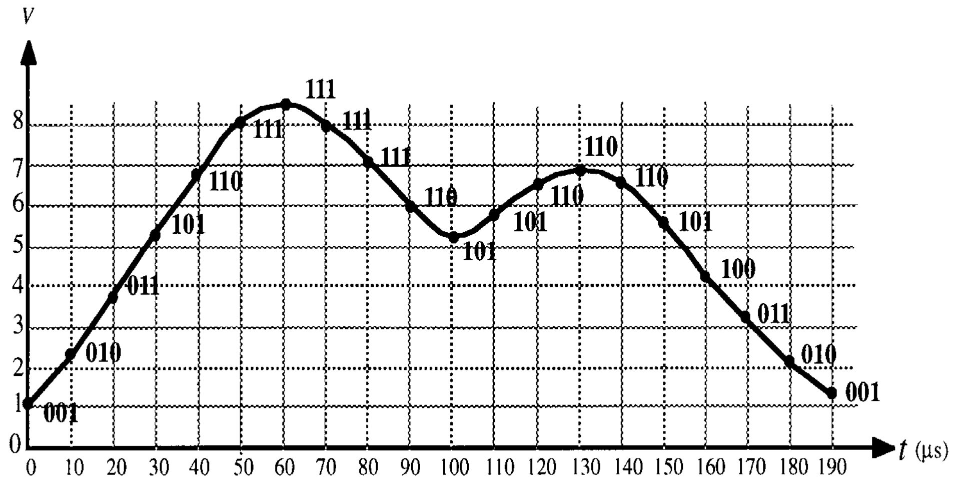


FIGURE 12-5

12. Output of 3-bit converter: 000, 001, 100, 110, 101, 100, 011, 010, 001, 001, 011, 110, 111, 111, 111, 111, 111, 111, 111, 100.

See Figure 12-6.

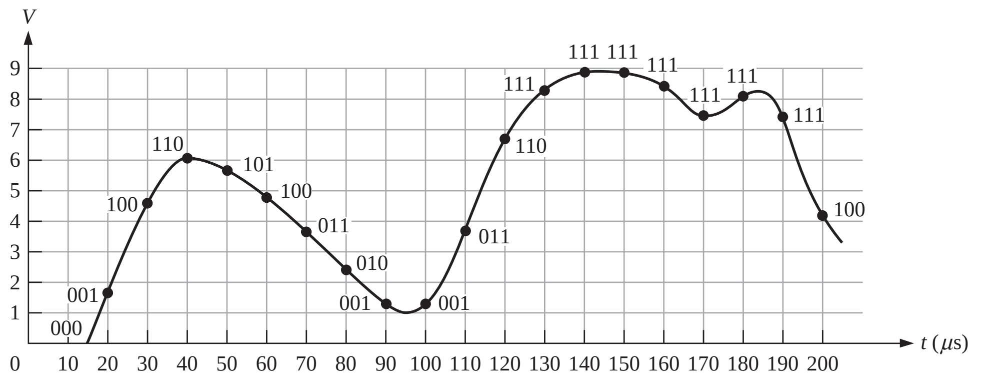


FIGURE 12-6

13.

SAR	Comment
11	Less than V_{in}. Keep the 1.
11	Less than V_{in}. Keep the 1.
11	Less than V_{in}. Keep the 1.

Conversion never terminates since 2 bits cannot represent the input.

14.

SAR	Comment
1000	Greater than V_{in}. Reset MSB.
0100	Less than V_{in}. Keep the 1.
0110	Equal to V_{in}. Keep the 1 (final state)

15. See Figure 12-7.

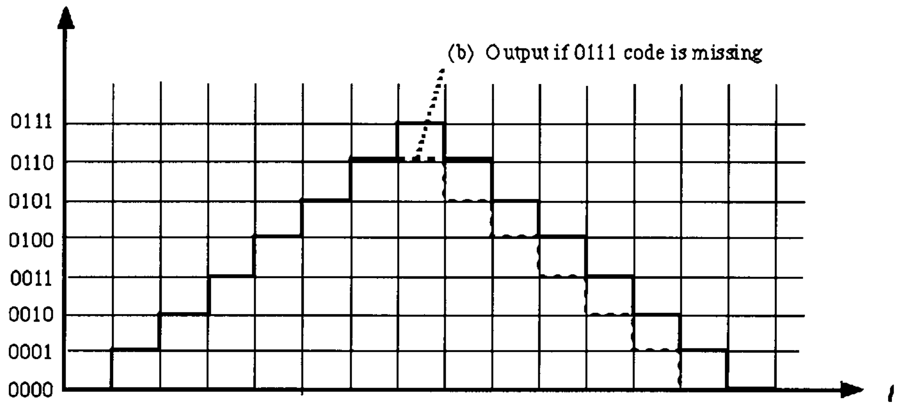


FIGURE 12-7

Section 12-3 Digital-to-Analog Conversion Methods

16. $R_0 = 10\ \text{k}\Omega$

$$R_1 = \frac{R_0}{2} = \frac{10\ \text{k}\Omega}{2} = 5\ \text{k}\Omega$$

$$R_2 = \frac{R_0}{4} = \frac{10\ \text{k}\Omega}{4} = 2.5\ \text{k}\Omega$$

$$R_3 = \frac{R_0}{8} = \frac{10\ \text{k}\Omega}{8} = 1.25\ \text{k}\Omega$$

17. See Figure 12-8.

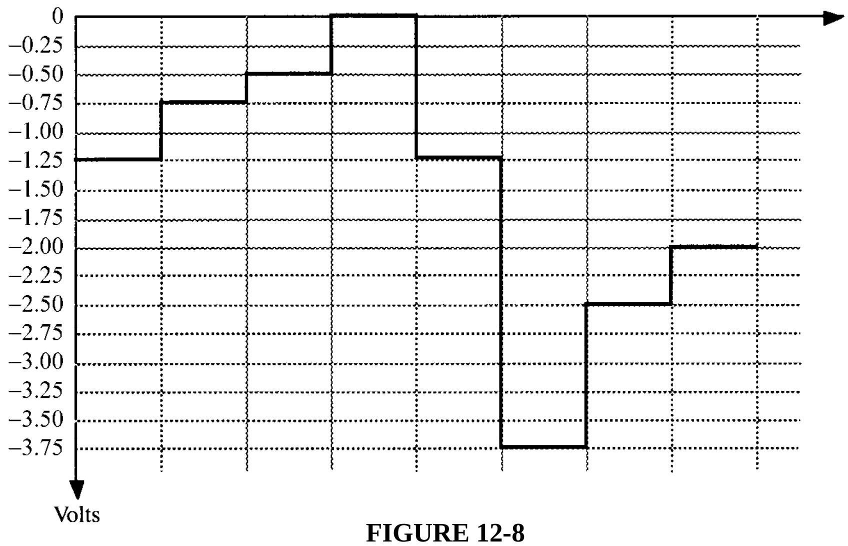


FIGURE 12-8

18. See Figure 12-9.

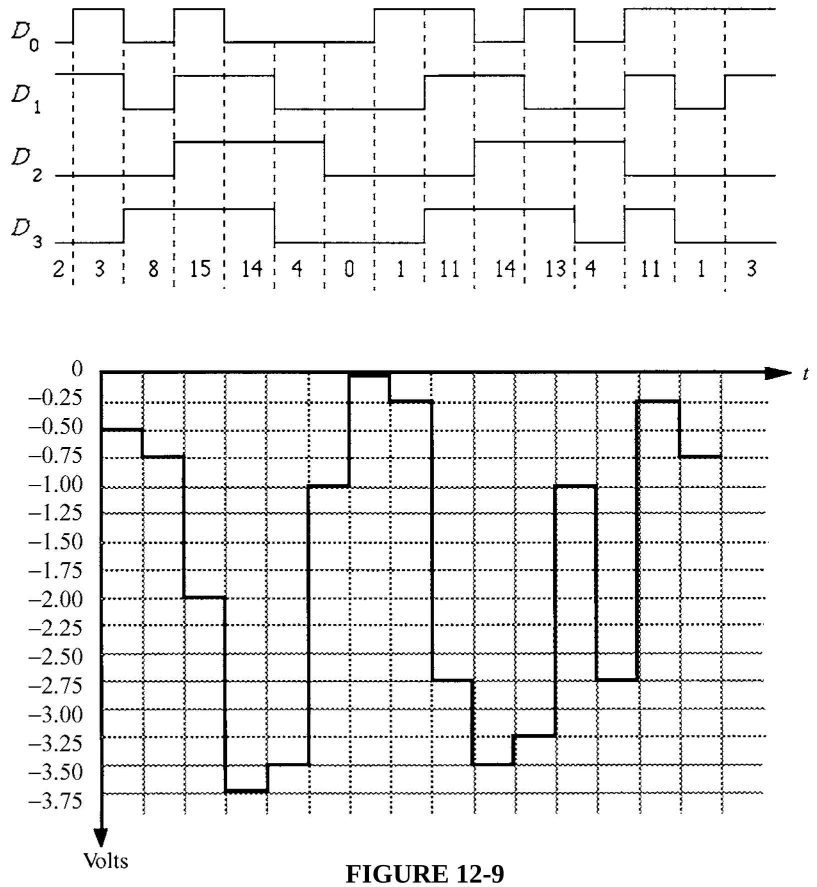


FIGURE 12-9

19. (a) $\left(\frac{1}{(2^3-1)}\right)100 \ = 14.3\%$

(b) $\left(\frac{1}{2^{10}-1}\right)100 \ = 0.098\%$

(c) $\left(\frac{1}{2^{18}-1}\right)100 \ = 0.00038\%$

20. See Figure 12-10.

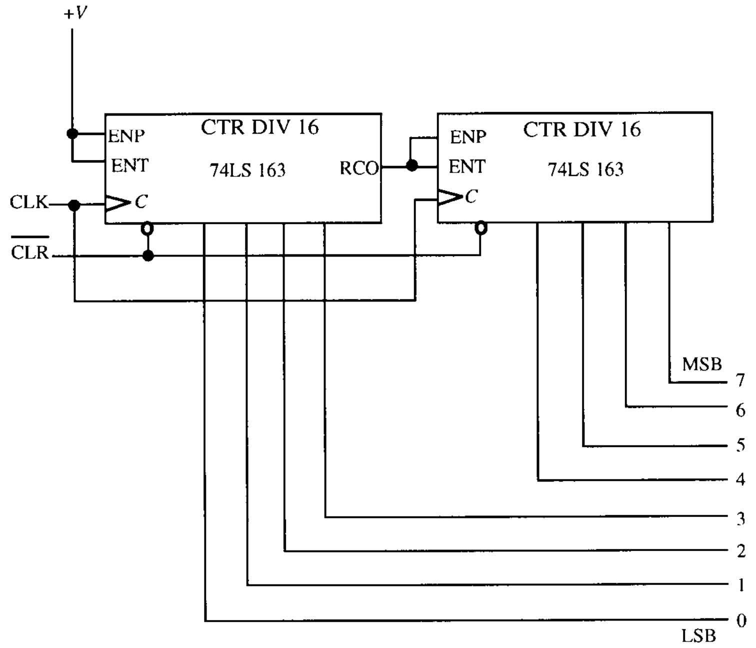


FIGURE 12-10

21. See Figure 12-11.

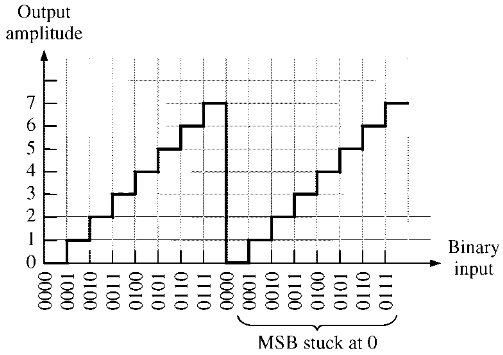
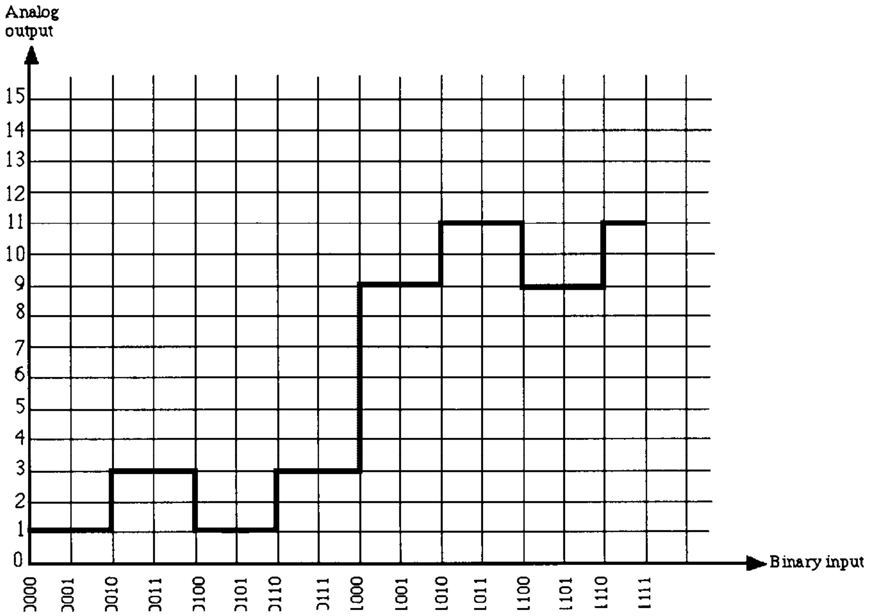


FIGURE 12-11

22. See Figure 12-12.



The LSB D_0 is stuck HIGH and D_3 is stuck LOW.

FIGURE 12-12

Section 12-4 Digital Signal Processing Basics

23. The purpose of analog-to-digital conversion is to change an analog signal into a sequence of digital codes that represent the amplitude of the analog signal with respect to time.

24. See Figure 12-13.

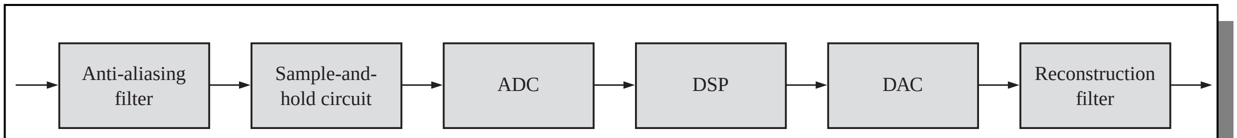


FIGURE 12-13

25. The purpose of digital-to-analog conversion is to change a sequence of digital codes into an analog signal represented by the digital codes.

Section 12-5 The Digital Signal Processor (DSP)

26. $2000\text{ MIPS} \times \dfrac{32\text{ bit/instruction}}{8\text{ bits/byte}}$

= 2000 MIPS × 4 bytes/instruction
= 8000 Mbytes/s

27. $\dfrac{400\text{ Mbits/s}}{32\text{ bits/instruction}}$ = 12.5 million instructions/s

28. 1000 MFLOPS = 1,000,000,000 floating-point operations/s

29.
1. Program address generate (PG). The program address is generated by the CPU.
2. Program address send (PS). The program address is sent to the memory.
3. Program access ready wait (PW). A memory read operation occurs.
4. Program fetch packet receive (PR). The CPU receives the packet of instructions.

30.
1. Instruction dispatch (DP): Instruction packets are split into execute packets and assigned to functional units;
2. Instruction decode (DC): Instructions are decoded.

CHAPTER 13
COMPUTER CONCEPTS

Section 13-1 The Basic Computer

1. The basic elements of a computer are central processing unit (CPU), memory unit, and input/output ports.

2. Two types of software are system and application.

3. A bus is a set of physical connections over which data and other information is transferred in a computer according to a standard set of specifications.

4. A port is a physical interface on a computer through which data is passed to and from peripherals.

Section 13-2 The Microprocessor

5. The basic elements of a microprocessor are arithmetic logic unit (ALU), instruction decoder, control unit, and register array.

6. A microprocessor performs arithmetic operations, logic operations, data movements, and decision functions.

7. The three microprocessor buses are address, data, and control.

8. Groups of Pentium instructions are: data transfer, arithmetic and logic, bit manipulation, loops and jumps, strings, subroutines and interrupts, and control.

9. Pipelining is the process by which a microprocessor begins executing the next instruction before the previous instruction has been completed.

10. Multitasking is the process of executing more than one program at a time. Multithreading is the process of executing different parts (threads) of a single program simultaneously.

Section 13-3 Basic Microprocessor Operation

11. A microprocessor repeatedly cycles through *fetch, decode, execute.*

12 Pipelining is the process of fetching and executing at the same time so that more than one instruction can be processed simultaneously.

13. The six segment registers of the 80386 and above are:
CS, DS, SS, ES, FS, GS

14. The code segment (CS) register contains 0F05 and the instruction pointer contains 0100. The physical address is

0F050 + 0100 = 0F150

15. AH and AL are 8-bit registers and represent the high and low part of the 16-bit AX register. The EAX is a 32-bit register which includes the AX register as the lower 16 bits.

16. (a) A *flag* is a bit stored in the flag register that is set or cleared by the processor.
(b) A flag indicates a status or a control condition. A *status* flag is an indicator of a condition after an arithmetic or logic operation. A *control* flag alters processor operations under certain conditions.

17. Instruction pairing allows two instructions to execute at the same time.

Section 13-4 Computer Programming

18. An assembler is a program that translates mnemonics and operands into machine code.

19. The flowchart in Figure 13-1 shows the process for adding numbers from one to ten and saving the results in a memory location named TOTAL.

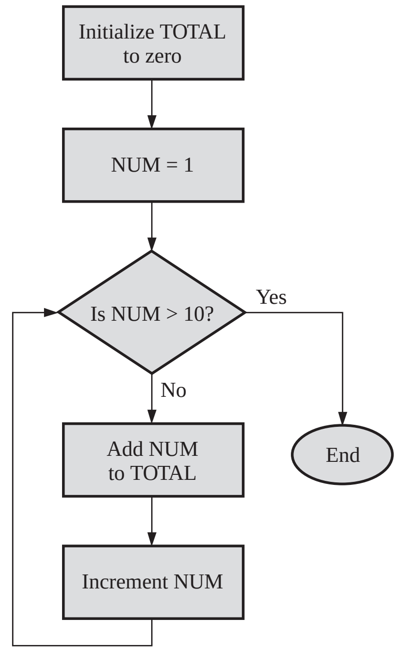


FIGURE 13-1

20. The flowchart in Figure 13-2 shows how you can count the number of bytes in a string and place the count in a memory location called COUNT. The string starts at a location named START and uses 20H (space) to indicate the end.

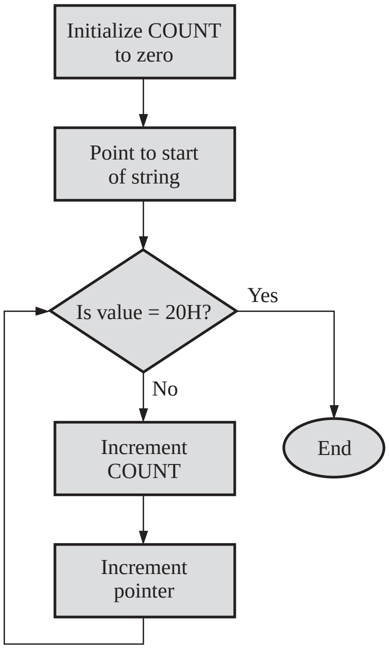


FIGURE 13-2

21. When the instruction **mov ax, [bx]** is executed, the word in memory pointed to by the bx register is copied to the ax register.

22. A compiler is a program that compiles or translates a program written in high-level language and converts it to machine code.

Section 13-5 Interrupts

23. In a polled I/O, the CPU polls each device in turn to see if it needs service; in an interrupt-driven system, the peripheral device signals the CPU when it requires service.

24. Vectoring is when the PIC provides a pointer to a service routine.

25. A software interrupt is a program instruction that invokes an interrupt service routine.

Section 13-6 Direct Memory Access (DMA)

26. In a DMA operation, the DMA controller is given control by the CPU and allows data to flow between memory and a peripheral directly, bypassing the CPU.

27. The CPU is bypassed in DMA.

Section 13-7 Internal Interfacing

28. See Figure 13-3.

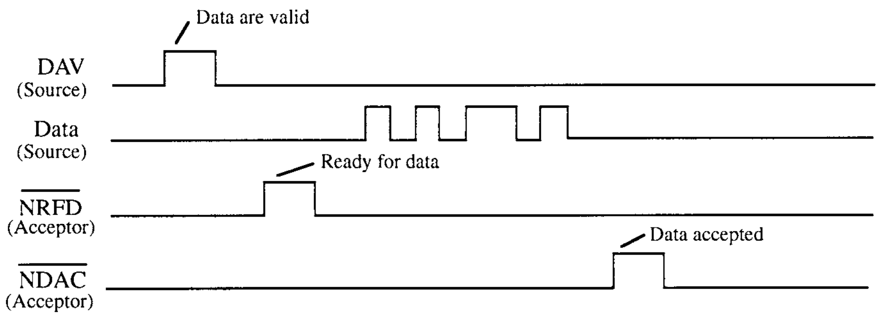


FIGURE 13-3

29. See Figure 13-4.

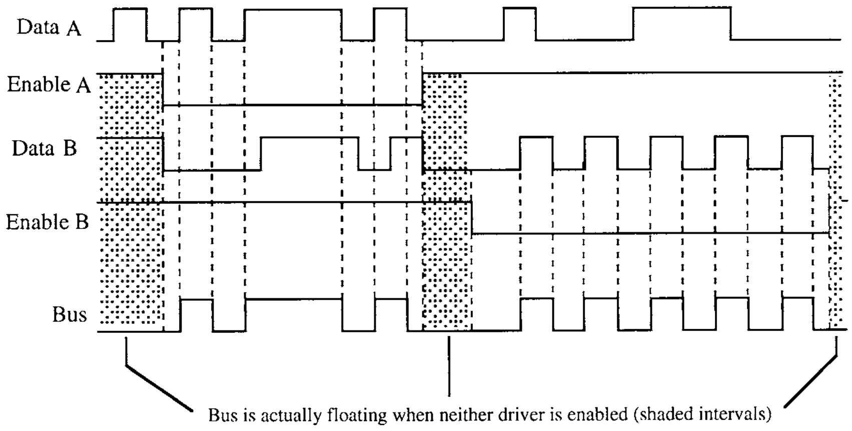

Data A
Enable A
Data B
Enable B
Bus
Bus is actually floating when neither driver is enabled (shaded intervals)

FIGURE 13-4

30. See Figure 13-5.

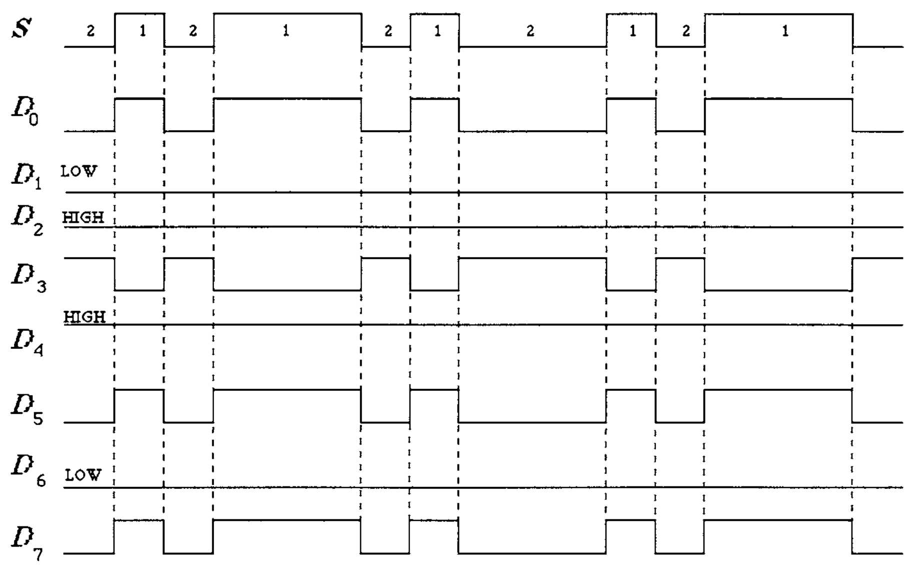


FIGURE 13-5

Section 13-8 Bus Standards

31. The local bus is the collection of buses interfacing directly with the processor. The PCI bus is used for expansion devices and is connected to the local bus through a bus controller.

32. Plug-and-Play refers to self-configuring hardware that can be installed into and used in a computer system without the need for manual installation of jumpers or setting of switches.

33. The PCI bus is a 33 or 66 MHz, 32- or 64-bit, plug-and-play compatible expansion bus. ISA is an 8- or 16-bit 8.33 MHz expansion bus. PCI supports 3.3 V supplies while ISA supports 5 V and 12 V supplies.

34. A shorter RS-232C cable can support faster communication rates.

35. DCE stands for data communications equipment, such as a modem. DTE stands for data terminal equipment, such as a computer. Both acronyms are associated with the RS-232/EIA-232 standard.

36. A USB cable consists of a power line, ground line, and two differential data lines.

37. Since there are eight instruments already on the bus and the limit is fourteen, six more instruments can be connected.

38. Three data bytes are transferred because the NDAC line goes HIGH three times, each time indicating that a data byte is accepted.

39. A controller is sending data to two listeners. The first two bytes of data (3F and 41) go to the listener with address 001A. The second two bytes go to the listener with address 001B. The handshake signals (DAV, NRFD, and NDAC) indicate that the data transfer is successful. See Figure 13-6.

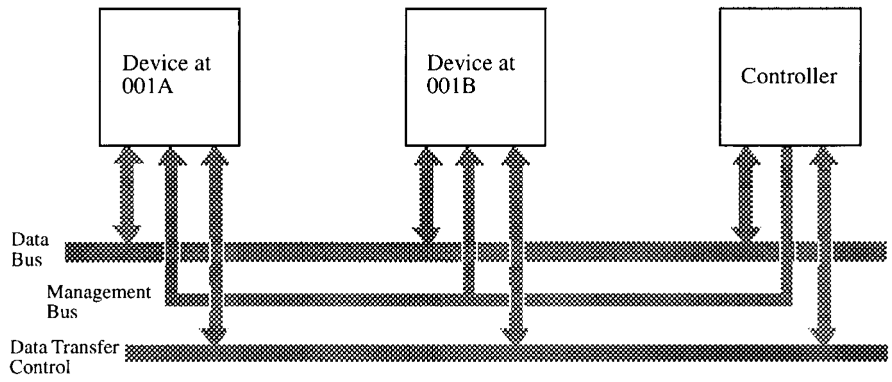


FIGURE 13-6

40. If a talker sends a data byte to a listener on a GPIB system and a DTE sends a data byte to a DCE on an RS-232C system, the RS-232C system will receive the data first. This is because GPIB requires significantly more setup and handshaking than RS-232C.

CHAPTER 14
INTEGRATED CIRCUIT TECHNOLOGIES

Section 14-1 Basic Operational Characteristics and Parameters

1. No, because the $V_{\text{OH(min)}}$ is less than the $V_{\text{IH(min)}}$. The gate may interpret 2.2 V as a LOW.

2. Yes, they are compatible because the $V_{\text{OL(max)}}$ is less than the $V_{\text{IL(max)}}$.

3. $V_{\text{NH}} = V_{\text{OH(min)}} - V_{\text{IH(min)}} = 2.4\text{ V} - 2.25\text{ V} = 0.15\text{ V}$
$V_{\text{NL}} = V_{\text{IL(max)}} - V_{\text{OL(max)}} = 0.65\text{ V} - 0.4\text{ V} = 0.25\text{ V}$

4. The maximum amplitudes equal the noise margins of 0.15 V and 0.25 V.

5. Gate A: $V_{\text{NH}} = 2.4\text{ V} - 2\text{ V} = 0.4\text{ V}$
$V_{\text{NL}} = 0.8\text{ V} - 0.4\text{ V} = 0.4\text{ V}$
Gate B: $V_{\text{NH}} = 3.5\text{ V} - 2.5\text{ V} = 1\text{ V}$
$V_{\text{NL}} = 0.6\text{ V} - 0.2\text{ V} = 0.4\text{ V}$
Gate C: $V_{\text{NH}} = 4.2\text{ V} - 3.2\text{ V} = 1\text{ V}$
$V_{\text{NL}} = 0.8\text{ V} - 0.2\text{ V} = 0.6\text{ V}$

Gate C has the highest noise margins.

6. $P_{\text{D(LOW)}} = (5\text{ V})(2\text{ mA}) = 10\text{ mW}$
$P_{\text{D(HIGH)}} = (5\text{ V})(3.5\text{ mA}) = 17.5\text{ mW}$

$$P_{\text{D(avg)}} = \frac{P_{\text{D(LOW)}} + P_{\text{D(HIGH)}}}{2} = \frac{27.5\text{ mW}}{2} = 13.75\text{ mW}$$

7. The pulse goes through three gates in the shortest path.
3 × 4 ns = 12 ns

8. $$t_{\text{p(avg)}} = \frac{t_{\text{PLH}} + t_{\text{PHL}}}{2} = \frac{2\text{ ns} + 3\text{ ns}}{2} = 2.5\text{ ns}$$

9. Gate A average propagation delay:

$$\frac{t_{\text{PLH}} + t_{\text{PHL}}}{2} = \frac{1\text{ ns} + 1.2\text{ ns}}{2} = 1.1\text{ ns}$$

Speed/Power product = (1.1 ns)(15 mW) = 16.5 pJ

Gate B average propagation delay:

$$\frac{5\text{ ns} + 4\text{ ns}}{2} = 4.5\text{ ns}$$

Speed/Power product = (4.5 ns)(8 mW) = 36 pJ

Gate C average propagation delay: $\frac{10\text{ ns}+10\text{ ns}}{2}$ = 10 ns

Speed/Power product = (10 ns)(0.5 mW) = 5 pJ

Gate C has the best speed/power product.

10. Gate A can be operated at the highest frequency because it has the shortest propagation delay.

11. G2 is overloaded because it has 12 unit loads.

12. The network in (a) can operate at the highest frequency because the driving gate has fewer loads.

Section 14-2 CMOS Circuits

13. (a) ON (b) OFF
(c) OFF (d) ON

14. Unused inputs should be connected as follows:
Negative-OR gate (NAND) to V_{CC}
NAND gate to + V_{CC}
NOR gate to ground

15. See Figure 14-1 for another possible approach in addition to circuit given in text answers.

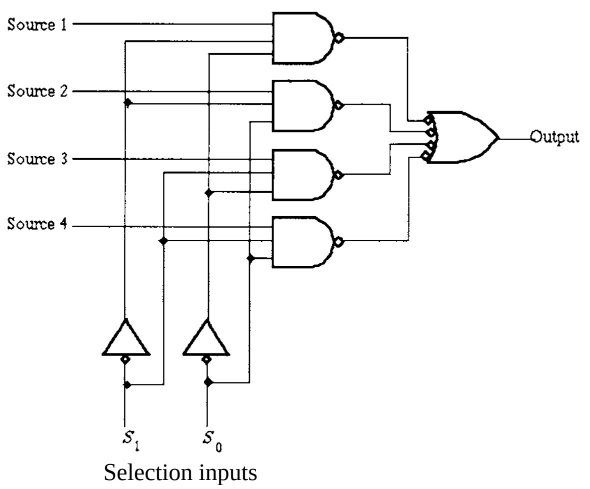


FIGURE 14-1

Section 14-3 TTL (Bipolar) Circuits

16. (a) ON: high voltage on base forward-biases the base-emitter junction.
(b) OFF: insufficient voltage on base to forward-bias the base-emitter junction.
(c) OFF: emitter is more positive than the base which reverse-biases the base-emitter junction.
(d) OFF: base and emitter at same voltage. No forward bias.

17. See Figure 14-2.

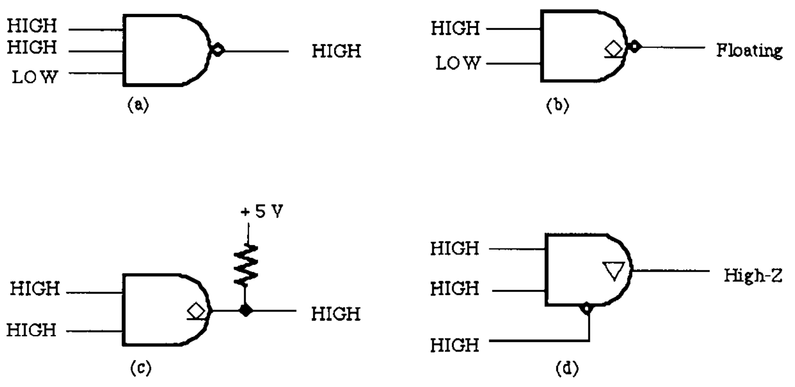


FIGURE 14-2

18. Connect a 1 kΩ pull-up resistor to the unused inputs of the two NAND gates. Connect the unused input of the NOR gate to ground. Connect a pull-up resistor to the open collector of the NOR gate (value depends on load).

Section 14-4 Practical Considerations in the Use of TTL

19. See Figure 14-3.

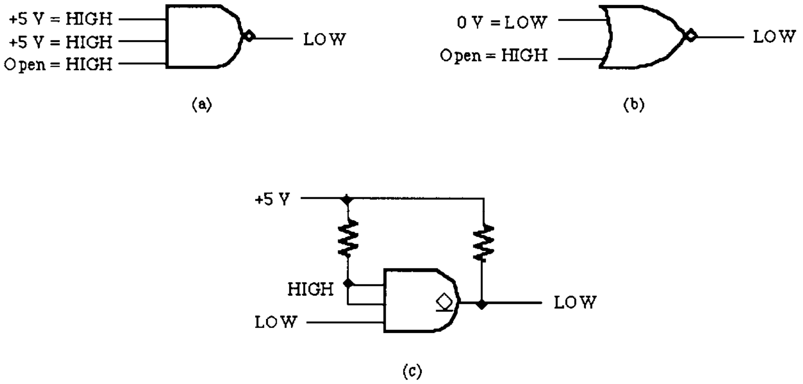


FIGURE 14-3

20. (a) The driving gate output is HIGH, it is sourcing 3 unit loads.
$I_T = 3(40\ \mu A) = \mathbf{120\ \mu A}$

(b) The driving gate output is LOW, it is sinking current from 2 unit loads.
$I_T = 2(-1.6\ mA) = \mathbf{-3.2\ mA}$

(c) G1 output is HIGH, it is sourcing 6 unit loads.
$I_T = 6(40\ \mu A) = \mathbf{240\ \mu A}$

G2 output is LOW, it is sinking current from 2 unit loads.
$I_T = 2(-1.6\ mA) = \mathbf{-3.2\ mA}$

G3 output is HIGH, it is sourcing 2 unit loads.
$I_T = 2(40\ \mu A) = \mathbf{80\ \mu A}$

21. See Figure 14-4. Pull-up resistors of second-level inverters are not shown.

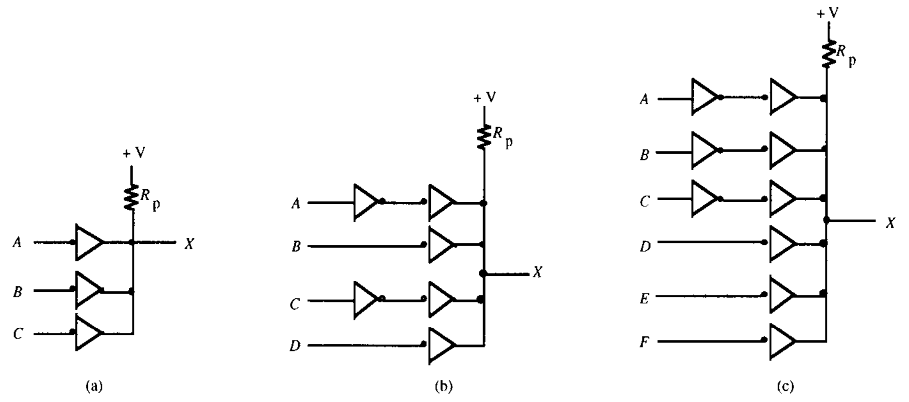


FIGURE 14-4

22. (a) $X = AB\overline{C}\,\overline{D}$

(b) $X = (\overline{ABC})(\overline{DE})(\overline{FG})$

(c) $X = (\overline{A+B})(\overline{C+D})(\overline{E+F})(\overline{G+H}) = \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}\,\overline{E}\,\overline{F}\,\overline{G}\,\overline{H}$

23. Worst case for determining minimum R_p is when only one gate is sinking all of the current (40 mA maximum).

For 10 UL: $I_L = 10(1.6\ mA) = 16\ mA$
For each gate: $I_{Rp(max)} = I_{OL(max)} - 16\ mA = 40\ mA - 16\ mA = 24\ mA$

$$V_{Rp} = 5\ V - 0.25\ V = 4.75\ V$$
$$R_{p(min)} = \frac{V_{Rp}}{I_{Rp(max)}} = \frac{4.75\ V}{24\ mA} = 198\ \Omega$$

$R_{p(min)}$ for (a), (b), and (c) is the same value.

24. See Figure 14-5.

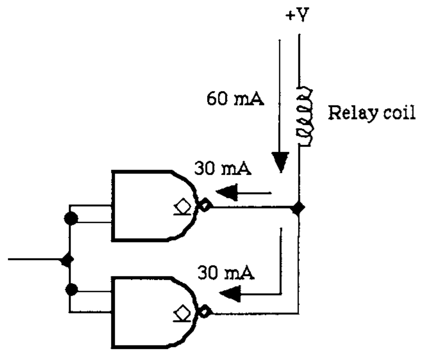


FIGURE 14-5

Section 14-5 Comparison of CMOS and TTL Performance

25. **F series**: SPP = 3.3 ns × 6 mW = 19.8 pJ
LS series: SPP = 10 ns × 2.2 mW = 22 pJ
ALS series: SPP = 7 ns × 1.4 mW = 9.8 pJ
ABT series: SPP = 3.2 ns × 17 µW = 0.0544 pJ
HC series: SPP = 7 ns × 2.75 µW = 0.01925 pJ
AC series: SPP = 5 ns × 0.55 µW = 0.00275 pJ
AHC series: SPP = 3.7 ns × 2.75 µW = 0.010175 pJ
LV series: SPP = 9 ns × 1.6 µW = 0.0144 pJ
LVC series: SPP = 4.3 ns 0.8 µW = 0.00344 pJ
ALVC series: SPP = 3 ns × 0.8 µW = 0.0024 pJ

ALVC has the best (lowest value) speed-power product. It is, however, misleading to compare CMOS and TTL in terms of SPP because the power of CMOS goes up with frequency.

26. (a) ALVC
(b) AHC
(c) AC
(d) ALVC

27. (a) A and B to X: 3(3.3 ns) = 9.9 ns
C and D to X: 2(3.3 ns) = 6.6 ns

(b) A to X1, X2, X3: 2(7 ns) = 14 ns
B to X1: 7 ns
C to X2: 7 ns
D to X3: 7 ns

(c) A, B to X: 3(3.7 ns) = 11.1 ns
C, D, to X: 2(3.7 ns) = 7.4 ns

28. (a) HC has an $f_{\max}$ = 50 MHz

$$f_{\text{clock}} = \frac{1}{50\,\text{ns}} = 20\text{ MHz}$$

(b) LS has an $f_{\max}$ = 33 MHz

$$f_{\text{clock}} = \frac{1}{60\,\text{ns}} = 16.7\text{ MHz}$$

(c) AHC has an $f_{\max}$ = 170 MHz

$$f_{\text{clock}} = \frac{1}{4\,\text{ns}} = 250\text{ MHz}$$

Since $f_{\text{clock}} > f_{\max}$ for the AHC flip-flop, the output will be erratic.

Section 14-6 Emitter-Coupled Logic (ECL) Circuits

29. ECL operates with nonsaturated BJTs whereas TTL transistors saturate when turned on.

30. (a) Lowest propagation delay - ECL
(b) Lowest power – CMOS HC series
(c) Lowest speed/power product – CMOS HC series

PART 2

System Application Activity Solutions

CHAPTER 4
SYSTEM ACTIVITY APPLICATION

1. The digit 2 is formed by segments *a*, *b*, *d*, *e*, *g*.

2. The digit 5 is formed by segments *a*, *c*, *d*, *f*, *g*.

3. The letter A is formed by segments *a*, *b*, *c*, *e*, *f*, *g*.

4. The letter E is formed by segments *a*, *d*, *e*, *f*, *g*.

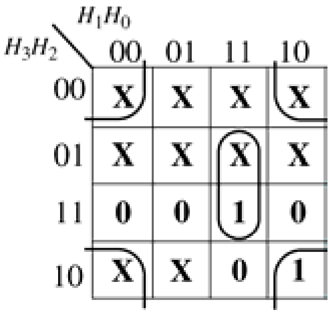
5. There is no segment common to all digits.

6. There is no segment common to all the letters shown in text Figure 4-46.

7. Segment *d* is used in letters b, C, d, and E. The hexadecimal code for each letter is as follows: b—1011; C—1100; d—1101; E—1100.

   The expression for segment *d* is

$$d = H_3\overline{H}_2H_1H_0 + H_3H_2\overline{H}_1\overline{H}_0 + H_3H_2\overline{H}_1H_0 + H_3H_2H_1\overline{H}_0$$



H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	1	0
10	X	X	0	1

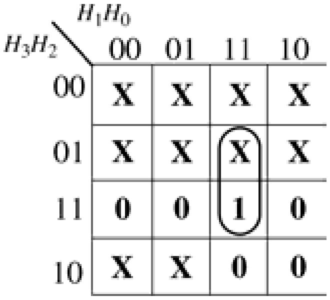
   The minimized expression for segment *d* is

$$d = \overline{H}_2\overline{H}_0 + H_2H_1H_0$$

8. Segment *e* is used in letters A, b, C, d, and E. The hexadecimal code for each letter is as follows: A—1010; b—1011; C—1100; d—1101; E—1110.

The expression for segment *e* is

$$e = H_3\bar{H}_2H_1\bar{H}_0 + H_3\bar{H}_2H_1H_0 + H_3H_2\bar{H}_1\bar{H}_0 + H_3H_2\bar{H}_1H_0 + H_3H_2H_1\bar{H}_0$$



H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	1	0
10	X	X	0	0

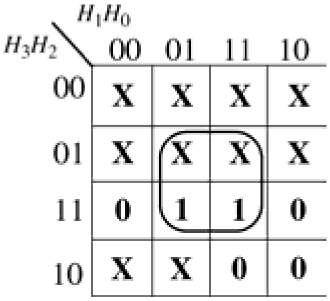
The minimized expression for segment *e* is

$$e = H_2H_1H_0$$

9. Segment *f* is used in letters A, b, C, and E. The hexadecimal code for each letter is as follows: A—1010; b—1011; C—1100; E—1110.

The expression for segment *f* is

$$f = H_3\bar{H}_2H_1\bar{H}_0 + H_3\bar{H}_2H_1H_0 + H_3H_2\bar{H}_1\bar{H}_0 + H_3H_2H_1\bar{H}_0$$



H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	1	1	0
10	X	X	0	0

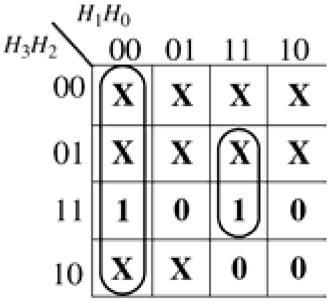
The minimized expression for segment *f* is

$$f = H_2H_0$$

10. Segment *g* is used in letters A, b, d, and E. The hexadecimal code for each letter is as follows: A—1010; b—1011; d—1101; E—1110.

The expression for segment *g* is

$$g = H_3\bar{H}_2H_1\bar{H}_0 + H_3\bar{H}_2H_1H_0 + H_3H_2\bar{H}_1H_0 + H_3H_2H_1\bar{H}_0$$

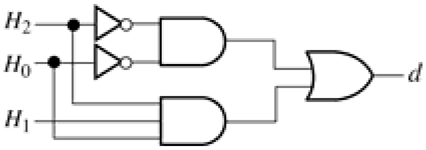


H_3H_2 \ H_1H_0	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	1	0	1	0
10	X	X	0	0

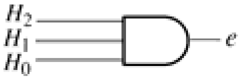
The minimized expression for segment *g* is

$$g = H_2H_1H_0 + \bar{H}_1\bar{H}_0$$

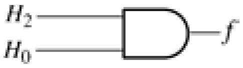
11. The location for segment *d* is



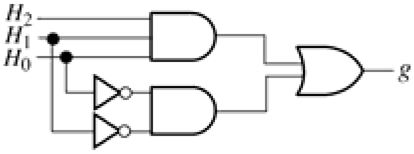
12. The logic for segment *e* is



13. The logic for segment *f* is



14. The logic for segment *g* is



Putting Your Knowledge to Work
To modify the decoder for a common-cathode display, the segment outputs must be active-HIGH. An inverter in each segment output will provide an active-HIGH.

CHAPTER 5
SYSTEM APPLICATION ACTIVITY

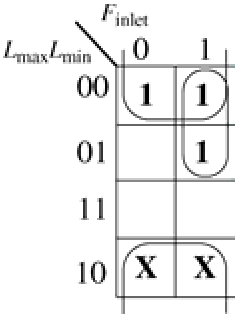
1. The two “cannot occur” conditions in text Table 5-6 are not possible because if the maximum level, L_{MAX}, has been reached as indicated by a 1, the minimum level L_{MIN} also must have been reached, but the 0 indicates that it has not.

2. The inlet valve is open under *three* conditions as indicated by $V_{\text{INLET}} = 1$.

3. The inlet valve turns off when the maximum level is reached.

4. The standard SOP expression for V_{inlet} is

$$V_{\text{inlet}} = \overline{L}_{\text{max}}\overline{L}_{\text{min}}\overline{F}_{\text{inlet}} + \overline{L}_{\text{max}}\overline{L}_{\text{min}}F_{\text{inlet}} + \overline{L}_{\text{max}}L_{\text{min}}F_{\text{inlet}}$$

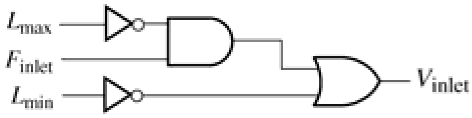


$L_{\text{max}}L_{\text{min}}$ \ F_{inlet}	0	1
00	1	1
01		1
11		
10	X	X

The simplified V_{inlet} expression is correct.

$$V_{\text{inlet}} = \overline{L}_{\text{min}} + \overline{L}_{\text{max}}F_{\text{inlet}}$$

5. The simplified logic for the inlet valve control:



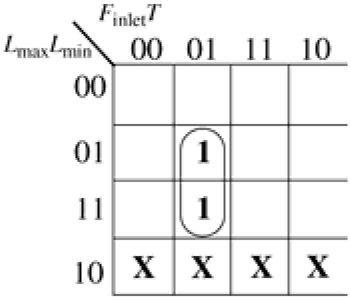
6. The additional input to the outlet valve control is T for temperature. The corn syrup must be at a specified temperature for proper viscosity before the syrup can be released into the mixing vat.

7. The outlet valve is open under two conditions as specified by the 1s in text Table 5-7.

8. Once the tank starts draining, the outlet value remains open (on) until the minimum level is reached.

9. The standard SOP expression for V_{outlet} is

$$V_{\text{outlet}} = \overline{L}_{\text{max}} L_{\text{min}} \overline{F}_{\text{inlet}} T + L_{\text{max}} L_{\text{min}} \overline{F}_{\text{inlet}} T$$

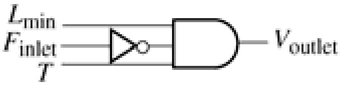


$L_{\text{max}}L_{\text{min}}$ \ $F_{\text{inlet}}T$	00	01	11	10
00				
01		1		
11		1		
10	X	X	X	X

The simplified V_{outlet} expression is correct.

$$V_{\text{outlet}} = L_{\text{min}} \overline{F}_{\text{inlet}} T$$

10. The simplified logic for the outlet valve control:



Putting Your Knowledge to Work

This assumes that the corn syrup is acceptable as long as its temperature is no more than 9° below the specified temperature. The temperature control circuit could be modified by eliminating the LSD in the BCD code for the measured temperature.

CHAPTER 6
SYSTEM APPLICATION ACTIVITY

1. The system remains in the first state (00) for 25 s if there is a vehicle on the side street or as long as there is no vehicle on the side street.

2. The system remains in the fourth state (10) for 4 s.

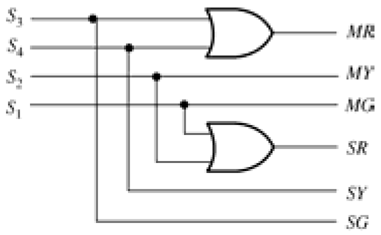
3. The expression for the condition producing a transition from the first state to the second state is

$$\overline{T}_{\mathrm{L}}V_{\mathrm{s}}$$

4. The expression for the condition that keeps the system in the second state is

$$T_{\mathrm{S}}$$

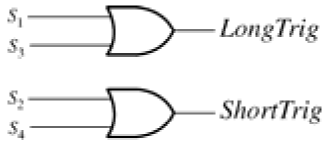
5. The diagram for the light output logic:



6. The truth table for the light output logic:

Inputs				Outputs					
S_4	S_3	S_2	S_1	*MR*	*MY*	*MG*	*SR*	*SY*	*SG*
0	0	0	1	0	0	1	1	0	0
0	0	1	0	0	1	0	1	0	0
0	1	0	0	1	0	0	0	0	1
1	0	0	0	1	0	0	0	1	0

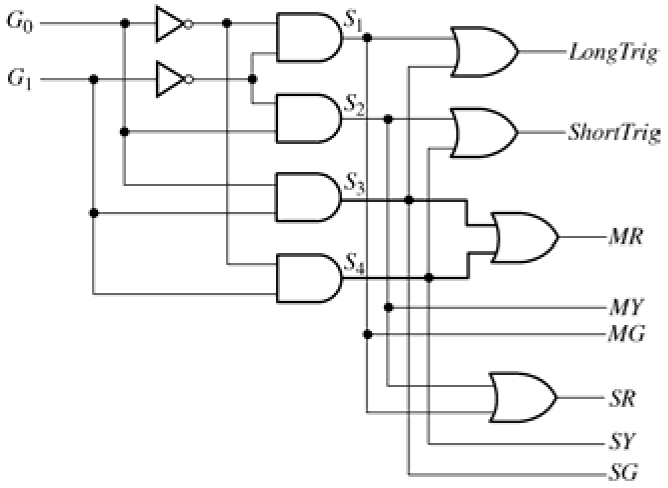
7. The diagram for the trigger logic:



8. The truth table for the trigger logic:

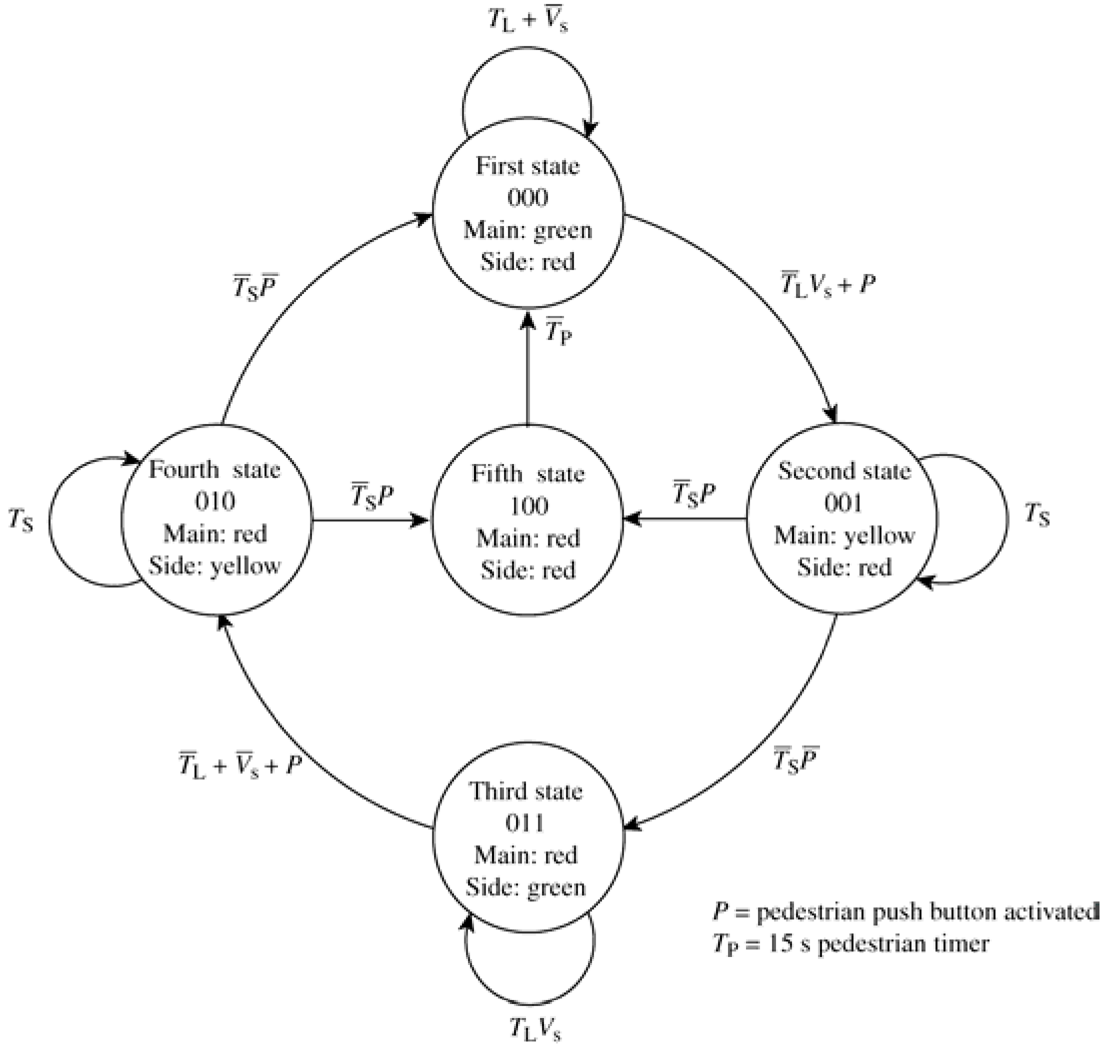
Inputs				**Outputs**	
S_4	S_3	S_2	S_1	*LongTrig*	*ShortTrig*
0	0	0	1	1	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	0	1

9. The diagram of the complete combinational logic:



Putting Your Knowledge to Work

(a) One approach is shown by the modified state diagram.



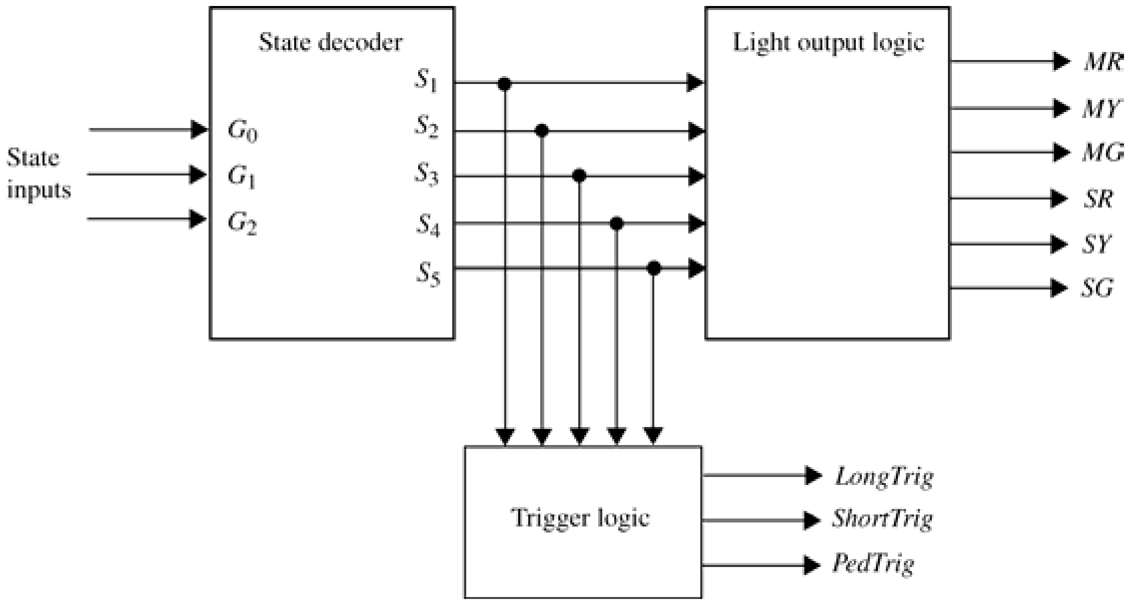
When a pedestrain pushes the button, one of the following sequences occurs:

□ If system is in first state (000), it goes immediately to second state (001) for 4 s and then to fifth state (100) for 15 s and then back to first state (000).

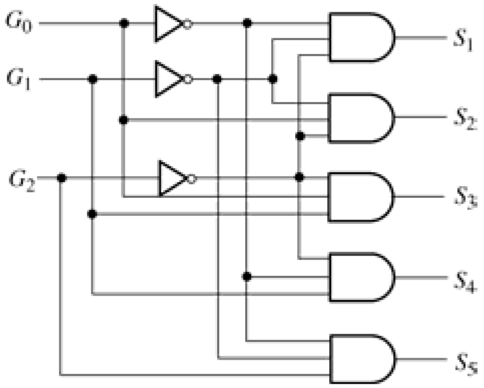
□ If system is in third state (011), it goes immediately to fourth state (010) for 4 s and then to fifth state (100) for 15 s and then back to first state (000).

□ If system is in either second or fourth states, the short timer is retriggerd and the system remains in that state for another 4 s before going to the fifth state.

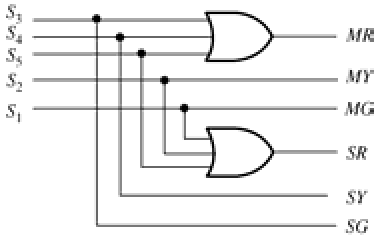
(b) The modified combinational logic block diagram:



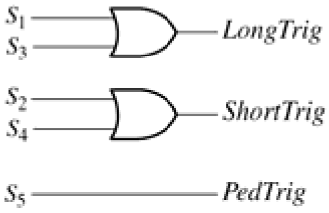
The modified state decoder logic:



The modified light output logic:



The modified trigger logic:



CHAPTER 7
SYSTEM APPLICATION ACTIVITY

1. The resistor and capacitor values for the 25 s timer are determined as follows.

   Let C_1 = 100 μF
   $t_W = 1.1R_1C_1$
   $R_1 = \dfrac{t_W}{1.1C_1} = \dfrac{25\text{ s}}{1.1(100\ \mu\text{F})} \cong 227\text{ k}\Omega$
   Other combinations are possible.

2. The resistor and capacitor values for the 4 s timer are determined as follows.

   Let C_1 = 100 μF
   $t_W = 1.1R_1C_1$
   $R_1 = \dfrac{4\text{ s}}{1.1(100\ \mu\text{F})} = 36.4\text{ k}\Omega$
   Other combinations are possible.

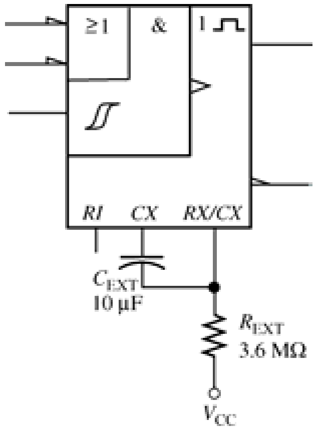
3. The resistor and capacitor values for the 10 kHz oscillator are determined as follows.

   Let C_1 = 0.01 μF
   $f = \dfrac{1.44}{(R_1 + 2R_2)C_1}$
   $R_1 + 2R_2 = \dfrac{1.44}{fC_1} = \dfrac{1.44}{(10\text{ kHz})(0.01\ \mu\text{F})} = 14.4\text{ k}\Omega$
   Let R_1 = 1.5 kΩ
   $R_2 = \dfrac{14.4\text{ k}\Omega - 1.5\text{ k}\Omega}{2} = 6.45\text{ k}\Omega$

4. The 25 s timer and the 4 s timer using 74121 one-shots:

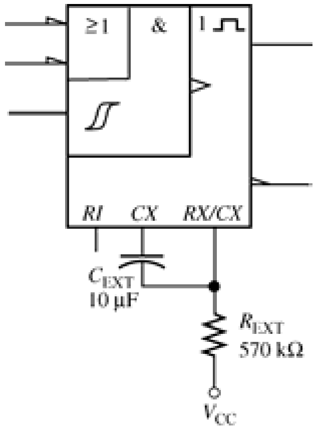
t_W = 25 s; let C_{EXT} = 10 μF
$t_W = 0.7R_{\text{EXT}}C_{\text{EXT}}$

$$R_{\text{EXT}} = \frac{t_W}{0.7C_{\text{EXT}}} = \frac{25\text{ s}}{0.7(10\ \mu\text{F})} = 3.6\text{ M}\Omega$$



t_W = 4 s; let C_{EXT} = 10 μF
$t_W = 0.7R_{\text{EXT}}C_{\text{EXT}}$

$$R_{\text{EXT}} = \frac{4\text{ s}}{0.7(10\ \mu\text{F})} = 571\text{ k}\Omega$$



Putting Your Knowledge to Work

Real time is determined by connecting a Multisim probe to the one-shot output and measuring its on-time. Simulation time is determined by connecting a virtual oscilloscope to the one-shot output and measuring the pulse-width.

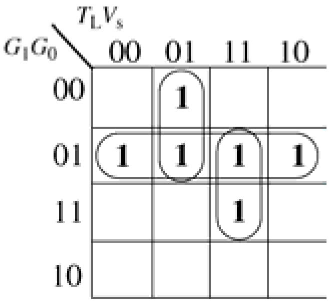
CHAPTER 8
SYSTEM APPLICATION ACTIVITY

1. The Boolean rule used is $A + \overline{A} = 1$.

   In this case, the terms G_1G_0 were factored out leaving $T_S + \overline{T}_S$:
   $\overline{G}_1G_0(T_S + \overline{T}_S) = \overline{G}_1G_0(1) = \overline{G}_1G_0$

2. D_0 is reduced by first expanding to standard SOP form:

$$D_0 = \overline{G}_1\overline{G}_0\overline{T}_LV_s + \overline{G}_1G_0 + G_1G_0T_LV_s$$
$$= \overline{G}_1\overline{G}_0\overline{T}_LV_s + \overline{G}_1G_0\overline{T}_L\overline{V}_s + \overline{G}_1G_0\overline{T}_LV_s + \overline{G}_1G_0T_L\overline{V}_s + \overline{G}_1G_0T_LV_s + G_1G_0T_LV_s$$



G_1G_0 \ T_LV_s	00	01	11	10
00		1		
01	1	1	1	1
11			1	
10				

   The reduced expression is

$$D_0 = \overline{G}_1G_0 + \overline{G}_1\overline{T}_LV_s + G_0T_LV_s$$

3. D_1 has five variables and is reduced by applying Boolean rules:

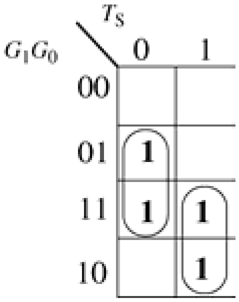
$$D_1 = \overline{G}_1G_0\overline{T}_S + G_1G_0T_LV_s + G_1G_0\overline{T}_L + G_1G_0\overline{V}_s + G_1\overline{G}_0T_S$$
$$= \overline{G}_1G_0\overline{T}_S + G_1G_0\left(T_LV_s + \overline{T}_L\right) + G_1G_0\overline{V}_s + G_1\overline{G}_0T_S$$

   Applying the rule 11: $A + AB = A + B$

$$D_1 = \overline{G}_1G_0\overline{T}_S + G_1G_0\left(V_s + \overline{T}_L\right) + G_1G_0\overline{V}_s + G_1\overline{G}_0T_S$$
$$D_1 = \overline{G}_1G_0\overline{T}_S + G_1G_0V_s + G_1G_0\overline{T}_L + G_1G_0\overline{V}_s + G_1\overline{G}_0T_S$$
$$= \overline{G}_1G_0\overline{T}_S + G_1G_0\left(V_s + \overline{T}_L + \overline{V}_s\right) + G_1\overline{G}_0T_S$$

Applying the rules 2 and 6: $A + 1 = 1$ and $A + A = 1$

$$D_1 = \overline{G}_1 G_0 \overline{T}_S + G_1 G_0 (1) + G_1 \overline{G}_0 T_S = \overline{G}_1 G_0 \overline{T}_S + G_1 G_0 T_S + G_1 G_0 \overline{T}_S + G_1 \overline{G}_0 T_S$$



The reduced expression is

$$D_1 = G_0 \overline{T}_S + G_1 T_S$$

4. The expressions agree with text Figure 8-64.

Putting Your Knowledge to Work

The combinational logic portion of the system was modified in Chapter 6 to accommodate a pedestrian input to turn both lights red for 15 s. To accomplish this exercise, the student must recognize that the timing circuits and the sequential logic must also be modified.

Timing circuit modification: An additional 555 timer configured as a one-shot with a 15 s output pulse is addded:

Let C_1 = 100 μF.
$t_W = 1.1R_1C_1$
$R_1 = \dfrac{15\text{ s}}{1.1(100\ \mu\text{F})} = 136\text{ k}\Omega$

Sequential logic modification: A third flip-flop is added. First, the next-state table is modified to accommodate the pedestrian input (P).

PRESENT STATE			NEXT STATE			INPUT CONDITIONS	FF INPUTS		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0		D_2	D_1	D_0
0	0	0	0	0	0	$T_L + \overline{V}_s$	0	0	0
0	0	0	0	0	1	$\overline{T}_L V_s + P$	0	0	1
0	0	1	0	0	1	T_S	0	0	1
0	0	1	0	1	1	$\overline{T}_S \overline{P}$	0	1	1
0	0	1	1	0	0	$\overline{T}_S P$	1	0	0
0	1	1	0	1	1	$T_L V_s$	0	1	1
0	1	1	0	1	0	$\overline{T}_L + \overline{V}_s + P$	0	1	0
0	1	0	0	1	0	T_S	0	1	0
0	1	0	0	0	0	$\overline{T}_S \overline{P}$	0	0	0
0	1	0	1	0	0	$\overline{T}_S P$	1	0	0
1	0	0	0	0	0	$\overline{T}_P$	0	0	0

$$\begin{aligned} D_0 &= \overline{G}_2\overline{G}_1\overline{G}_0\left(\overline{T}_L V_s + P\right) + \overline{G}_2\overline{G}_1 G_0 T_S + \overline{G}_2\overline{G}_1 G_0 \overline{T}_S \overline{P} + \overline{G}_2 G_1 G_0 T_L V_s \\ &= \overline{G}_2\overline{G}_1\overline{G}_0\overline{T}_L V_s + \overline{G}_2\overline{G}_1\overline{G}_0 P + \overline{G}_2\overline{G}_1 G_0 T_S + \overline{G}_2\overline{G}_1 G_0 \overline{T}_S \overline{P} + \overline{G}_2 G_1 G_0 T_L V_s \end{aligned}$$

$$\begin{aligned} D_1 &= \overline{G}_2\overline{G}_1 G_0 \overline{T}_S \overline{P} + \overline{G}_2 G_1 G_0 T_L V_s + \overline{G}_2 G_1 G_0 \left(\overline{T}_L + \overline{V}_s + P\right) + \overline{G}_2 G_1 \overline{G}_0 T_S \\ &= \overline{G}_2\overline{G}_1 G_0 \overline{T}_S \overline{P} + \overline{G}_2 G_1 G_0 T_L V_s + \overline{G}_2 G_1 G_0 \overline{T}_L + \overline{G}_2 G_1 G_0 \overline{V}_s + \overline{G}_2 G_1 G_0 P + \overline{G}_2 G_1 \overline{G}_0 T_S \end{aligned}$$

$$D_2 = \overline{G}_2\overline{G}_1 G_0 \overline{T}_S P + \overline{G}_2 G_1 \overline{G}_0 \overline{T}_S P$$

Notice that the expressions contain up to seven variables. Karnaugh map simplification would not be feasible. If you wish to minimize the expressions, the Quine-McCluskey method can be used but will be left as an exercise that can be assigned to students.

D_0 will take three 5-input AND (or NAND) gates, two 4-input AND (or NAND) gates, and one 5-input OR (or NAND) gate.

D_1 will take two 5-input AND (or NAND) gates, four 4-input AND (or NAND) gates, and one 6-input OR (or NAND).

D_2 will take two 5-input AND (or NAND) gates and one 2-input OR (or NAND) gate.

CHAPTER 9
SYSTEM APPLICATION ACTIVITY

1. The BCD code sequence for 4739 is 0100, 0111, 0011, 1001.

2. The state of shift register C after two correct code digits are entered is 00000100.

3. The purpose of the OR gate is to produce a trigger when a key is pressed in order to generate the clock pulses for the system.

4. When the digit 4 is entered, 0100 appears on the outputs of register A.

Putting Your Knowledge to Work

The security code logic can be modified for a 5-digit code by moving the HIGH parallel input of shift register C back one position. It would then take five clock pulses to shift the 1 to the output.

CHAPTER 10
SYSTEM APPLICATION ACTIVITY

1. The shift register initial outputs are $Q_0 = 1$, $Q_1 = 0$, $Q_2 = 0$, $Q_3 = 0$.

2. The state of the shift register after three clock pulses is

$$Q_0 = 0,\ Q_1 = 0,\ Q_2 = 0,\ Q_3 = 1$$

3. The purpose of the OR gates is to produce the BCD code from the selected AND gates.

4. In text Figure 10-73, NAND gates are used to implement the AND-OR logic of Figure 10-72.

5. The resistor packs provide pull-up resistors for the DIP switch outputs.

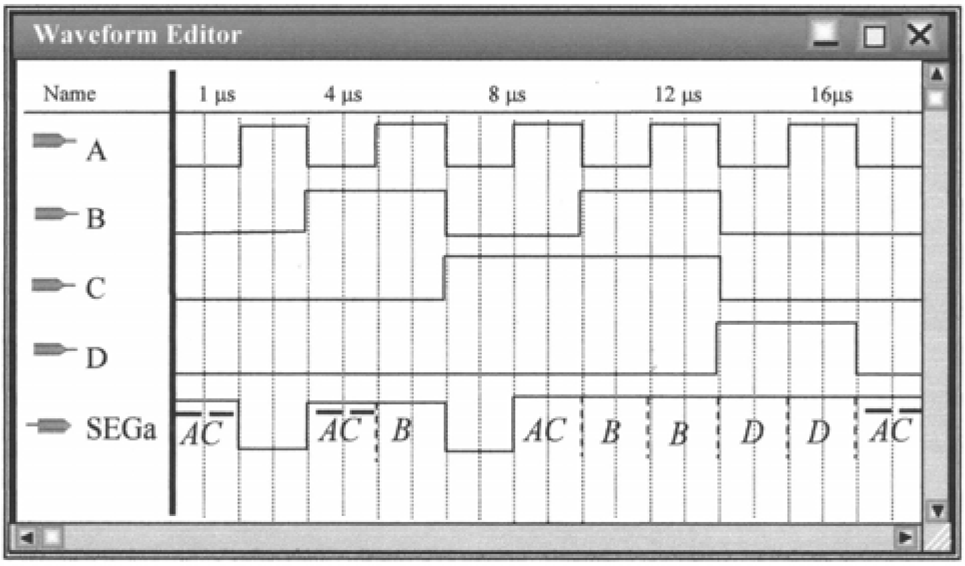
Putting Your Knowledge to Work

The memory and code selection logic can be modified for a 5-bit code by adding a DIP switch with resistor pull-ups, four more AND gates, changing the OR gates to five inputs, and changing the shift register from four bits to five bits.

CHAPTER 11
SYSTEM APPLICATION ACTIVITY

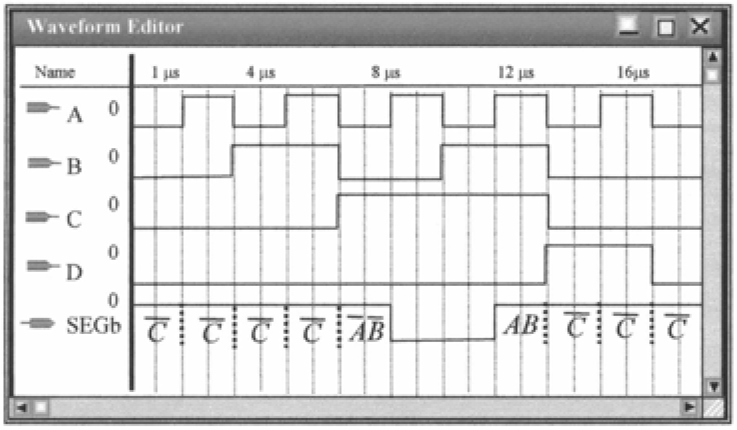
1. The logic expression for SEGa is SEGa = $AC + \overline{A}\overline{C} + B + D$.

   The conditions for SEGa to be HIGH are indicated on the waveform in the following figure, showing that the waveform is correct.


Waveform Editor
Name
1 µs
4 µs
8 µs
12 µs
16µs
A
B
C
D
SEGa
$\overline{A}\overline{C}$
$\overline{A}\overline{C}$
B
AC
B
B
D
D
$\overline{A}\overline{C}$

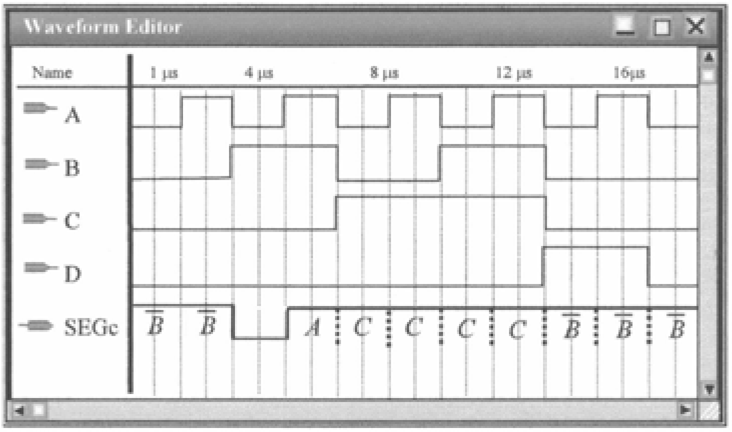
2. The logic expression for SEGb is SEGb = $AB + \overline{A}\overline{B} + \overline{C}$.

   The conditions for SEGb to be HIGH are indicated on the waveform in the following figure, showing that the waveform is correct.



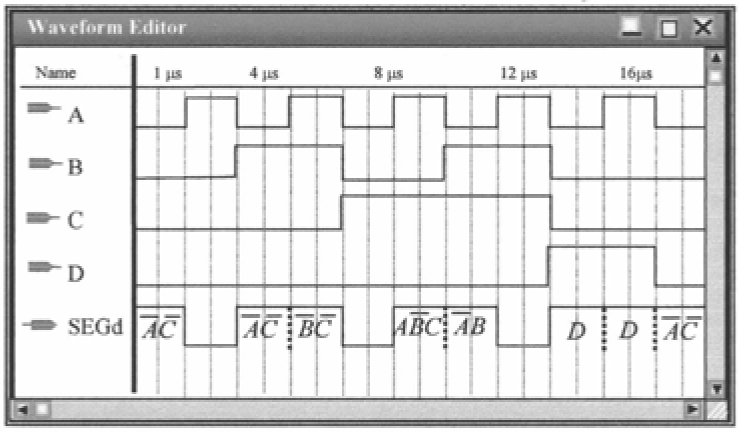
The logic expression for SEGc is SEGc = $A + \overline{B} + C$.

The conditions for SEGc to be HIGH are indicated on the waveform in the following figure showing that the waveform is correct.



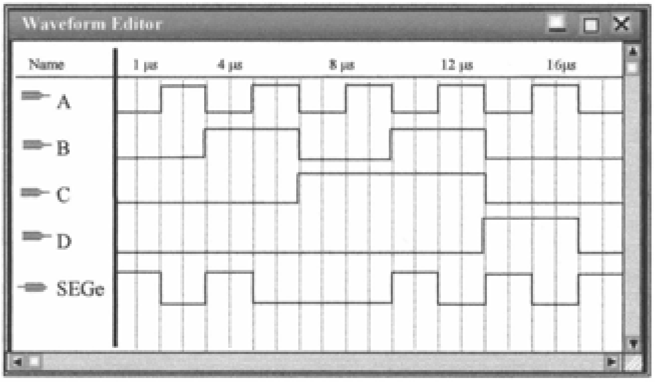
The logic expression for SEGd is SEGd = $\overline{A}B + \overline{A}\,\overline{C} + B\overline{C} + A\overline{B}C + D$.

The conditions for SEGd to be HIGH are indicated on the waveform in the following figure, showing that the waveform is correct.


Waveform Editor
Name
1 µs
4 µs
8 µs
12 µs
16µs
A
B
C
D
SEGd
$\overline{A}\,\overline{C}$
$\overline{A}\,\overline{C}$
$\overline{B}\,\overline{C}$
$A\overline{B}C$
$\overline{A}B$
D
D
$\overline{A}\,\overline{C}$

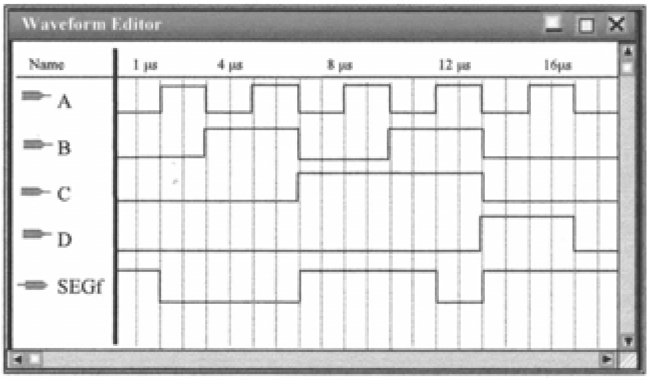
3. The logic expression for SEGe is SEGe = $\overline{A}B + \overline{A}\,\overline{C}$.

The waveform for SEGe is shown in the following figure.



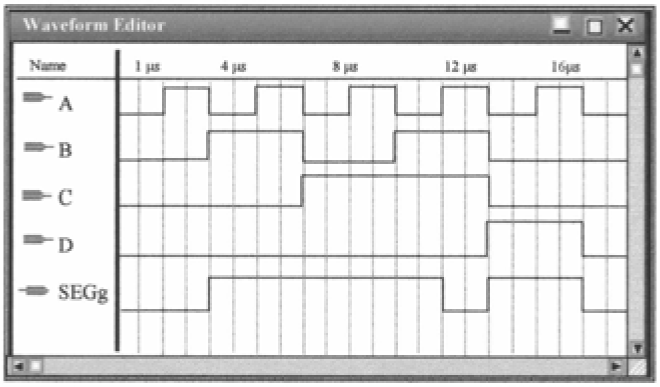
The logic expression for SEGf is SEGf = $\overline{A}\,\overline{B} + \overline{A}C + \overline{B}C + D$.

The waveform for SEGf is shown in the following figure.



The logic expression for SEGg is SEGg = $\overline{A}B + B\overline{C} + \overline{B}C + D$.

The waveform for SEGg is shown in the following figure.



PART 3

Overview of IEEE Standard 91-1984

Explanation of Logic Symbols

Courtesy of Texas Instruments

1.0 INTRODUCTION

The International Electrotechnical Commission (IEC) has been developing a very powerful symbolic language that can show the relationship of each input of a digital logic circuit to each output without showing explicitly the internal logic. At the heart of the system is dependency notation, which will be explained in Section 4.

The system was introduced in the USA in a rudimentary form in IEEE/ANSI Standard Y32.14-1973. Lacking at that time a complete development of dependency notation, it offered little more than a substitution of rectangular shapes for the familiar distinctive shapes for representing the basic functions of AND, OR, negation, etc. This is no longer the case.

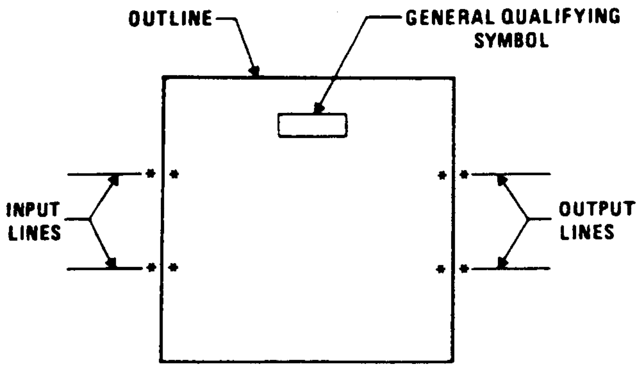
Internationally, Working Group 2 of IEC Technical Committee TC 3 has prepared a new document (Publication 617-12) that consolidates the original work started in the mid 1960's and published in 1972 (Publication 117-15) and the amendments and supplements that have followed. Similarly for the USA, IEEE Committee SCC 11.9 has revised the publication IEEE Std 91/ANSI Y32.14. Now numbered simply IEEE Std 91-1984, the IEEE standard contains all of the IEC work that has been approved, and also a small amount of material still under international consideration. Texas Instruments is participating in the work of both organizations and this document introduces new logic symbols in accordance with the new standards. When changes are made as the standards develop, future editions will take those changes into account.

The following explanation of the new symbolic language is necessarily brief and greatly condensed from what the standards publications will contain. This is not intended to be sufficient for those people who will be developing symbols for new devices. It is primarily intended to make possible the understanding of the symbols used in various data books and the comparison of the symbols with logic diagrams, functional block diagrams, and/or function tables will further help that understanding.

2.0 SYMBOL COMPOSITION

A symbol comprises an outline or a combination of outlines together with one or more qualifying symbols. The shape of the symbols is not significant. As shown in Figure 1, general qualifying symbols are used to tell exactly what logical operation is performed by the elements. Table I shows general qualifying symbols defined in the new standards. Input lines are placed on the left and output lines are placed on the right. When an exception is made to that convention, the direction of signal flow is indicated by an arrow as shown in Table II.

All outputs of a single, unsubdivided element always have identical internal logic states determined by the function of the element except when otherwise indicated by an associated qualifying symbol or label inside the element.



*Possible positions for qualifying symbols relating to inputs and outputs

Figure 1. Symbol Composition

The outlines of elements may be abutted or embedded in which case the following conventions apply. There is no logic connection between the elements when the line common to their outlines is in the direction of signal flow. There is at least one logic connection between the elements when the line common to their outlines is perpendicular to the direction of signal flow. The number of logic connections between elements will be clarified by the use of qualifying symbols and this is discussed further under that topic. If no indications are shown on either side of the common line, it is assumed there is only one connection.

When a circuit has one or more inputs that are common to more than one element of the circuit, the common-control block may be used. This is the only distinctively shaped outline used in the IEC system. Figure 2 shows that unless otherwise qualified by dependency notation, an input to the common-control block is an input to each of the elements below the common-control block.

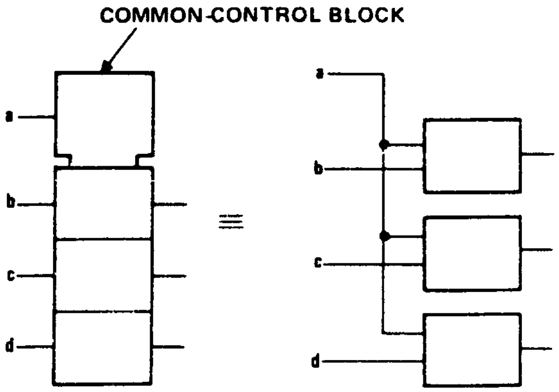


Figure 2. Common-Control Block

A common output depending on all elements of the array can be shown as the output of a common-output element. Its distinctive visual feature is the double line at its top. In addition the common-output element may have other inputs as shown in Figure 3. The function of th common-output element must be shown by use of a general qualifying symbol.

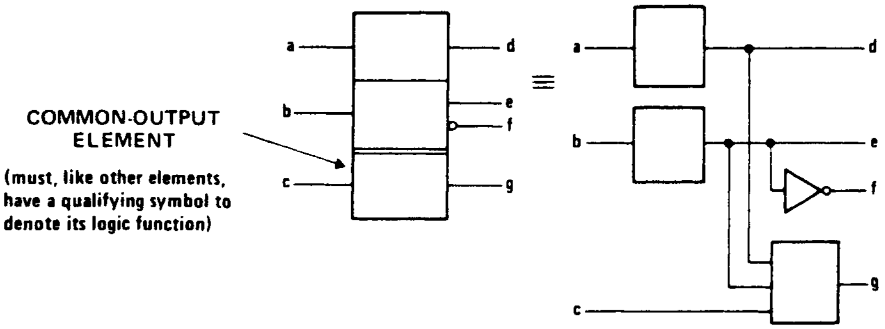

COMMON-OUTPUT ELEMENT
(must, like other elements, have a qualifying symbol to denote its logic function)
a
b
c
d
e
f
g
≡
a
b
c
d
e
f
g

Figure 3. Common-Output Element

3.0 QUALIFYING SYMBOLS

3.1 General Qualifying Symbols

Table I shows general qualifying symbols defined by IEEE Standard 91. These characters are placed near the top center or the geometric center of a symbol or symbol element to define the basic function of the device represented by the symbol or of the element.

3.2 Qualifying Symbols for Inputs and Outputs

Qualifying symbols for inputs and outputs are shown in Table II and will be familiar to most users with the possible exception of the logic polarity and analog signal indicators. The older logic negation indicator means that the external 0 state produces the internal 1 state. The internal 1 state means the active state. Logic negation may be used in pure logic diagrams; in order to tie the external 1 and 0 logic states to the levels H (high) and L (low), a statement of whether positive logic (1 = H, 0 = L) or negative logic (1 = L, 0 = H) is being used is required or must be assumed. Logic polarity indicators eliminate the need for calling out the logic convention and are used in various data books in the symbology for actual devices. The presence of the triangular polarity indicator indicates that the L logic level will produce the internal 1 state (the active state) or that, in the case of an output, the internal 1 state will produce the external L level. Note how the active direction of transition for a dynamic input is indicated in positive logic, negative logic, and with polarity indication.

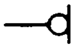
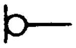
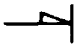
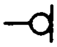
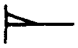
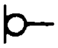
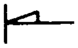
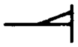
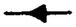
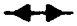
The internal connections between logic elements abutted together in a symbol may be indicated by the symbols shown in Table II. Each logic connection may be shown by the presence of qualifying symbols at one or both sides of the common line and if confusion can arise about the numbers of connections, use can be made of one of the internal connection symbols.

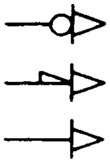
Table I. General Qualifying Symbols

SYMBOL	DESCRIPTION	CMOS EXAMPLE	TTL EXAMPLE
&	AND gate or function.	'HC00	SN7400
≥1	OR gate or function. The symbol was chosen to indicate that at least one active input is needed to activate the output.	'HC02	SN7402
=1	Exclusive OR. One and only one input must be active to activate the output.	'HC86	SN7486
=	Logic identity. All inputs must stand at the same state.	'HC86	SN74180
2k	An even number of inputs must be active.	'HC280	SN74180
2k+1	An odd number of inputs must be active.	'HC86	SN74ALS86
1	The one input must be active.	'HC04	SN7404
▷ or ◁	A buffer or element with more than usual output capability (symbol is oriented in the direction of signal flow).	'HC240	SN74S436
⎎	Schmitt trigger; element with hysteresis.	'HC132	SN74LS18
X/Y	Coder, code converter (DEC/BCD, BIN/OUT, BIN/7-SEG, etc.).	'HC42	SN74LS347
MUX	Multiplexer/data selector.	'HC151	SN74150
DMUX or DX	Demultiplexer.	'HC138	SN74138
Σ	Adder.	'HC283	SN74LS385
P−Q	Subtracter.	•	SN74LS385
CPG	Look-ahead carry generator	'HC182	SN74182
π	Multiplier.	•	SN74LS384
COMP	Magnitude comparator.	'HC85	SN74LS682
ALU	Arithmetic logic unit.	'HC181	SN74LS381
⎍	Retriggerable monostable.	'HC123	SN74LS422
1⎍	Nonretriggerable monostable (one-shot)	'HC221	SN74121
G ⎍⎍	Astable element. Showing waveform is optional.	•	SN74LS320
!G ⎍⎍	Synchronously starting astable.	•	SN74LS624
G! ⎍⎍	Astable element that stops with a completed pulse.	•	•
SRGm	Shift register. m = number of bits.	'HC164	SN74LS595
CTRm	Counter. m = number of bits; cycle length = 2^{m}.	'HC590	SN54LS590
CTR DIVm	Counter with cycle length = m.	'HC160	SN74LS668
RCTRm	Asynchronous (ripple-carry) counter; cycle length = 2^{m}.	'HC4020	•
ROM	Read-only memory.	•	SN74187
RAM	Random-access read/write memory.	'HC189	SN74170
FIFO	First-in, first-out memory.	•	SN74LS222
I=0	Element powers up cleared to 0 state.	•	SN74AS877
I=1	Element powers up set to 1 state.	'HC7022	SN74AS877
Φ	Highly complex function; "gray box" symbol with limited detail shown under special rules.	•	SN74LS608

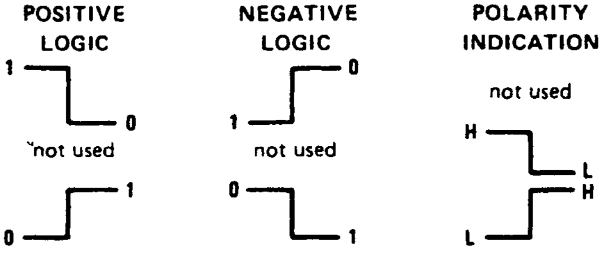
•Not all of the general qualifying symbols have been used in TI's CMOS and TTL data books, but they are included here for the sake of completeness.

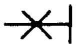
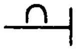
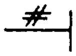
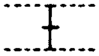
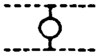
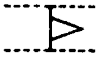
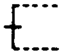
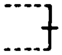
Table II. Qualifying Symbols for Inputs and Outputs

	Logic negation at input. External 0 produces internal 1.
	Logic negation at output. Internal 1 produces external 0.
	Active-low input. Equivalent to  in positive logic.
	Active-low output. Equivalent to  in positive logic.
	Active-low input in the case of right-to-left signal flow.
	Active-low output in the case of right-to-left signal flow.
	Signal flow from right to left. If not otherwise indicated, signal flow is from left to right.
	Bidirectional signal flow.



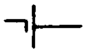
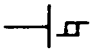
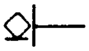
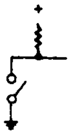
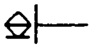
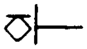
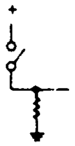
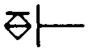
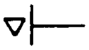
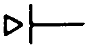
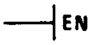
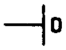
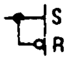
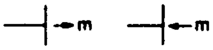
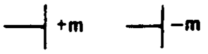
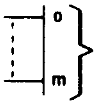
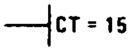
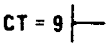
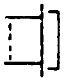
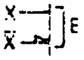
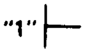
Dynamic inputs active on indicated transition



	Nonlogic connection. A label inside the symbol will usually define the nature of this pin.
	Input for analog signals (on a digital symbol) (see Figure 14).
	Input for digital signals (on an analog symbol) (see Figure 14).
	Internal connection. 1 state on left produces 1 state on right.
	Negated internal connection. 1 state on left produces 0 state on right.
	Dynamic internal connection. Transition from 0 to 1 on left produces transitory 1 state on right.
	Internal input (virtual input). It always stands at its internal 1 state unless affected by an overriding dependency relationship.
	Internal output (virtual output). Its effect on an internal input to which it is connected is indicated by dependency notation.

The internal (virtual) input is an input originating somewhere else in the circuit and is not connected directly to a terminal. The internal (virtual) output is likewise not connected directly to a terminal. The application of internal inputs and outputs requires an understanding of dependency notation, which is explained in Section 4.

Table III. Symbols Inside the Outline

	Postponed output (of a pulse-triggered flip-flop). The output changes when input initiating change (e.g., a C input) returns to its initial external state or level. See § 5.	
	Bi-threshold input (input with hysteresis).	
	N-P-N open-collector or similar output that can supply a relatively low-impedance L level when not turned off. Requires external pull-up. Capable of positive-logic wired-AND connection.	
	Passive-pull-up output is similar to N-P-N open-collector output but is supplemented with a built-in passive pull-up.	
	N-P-N open-emitter or similar output that can supply a relatively low-impedance H level when not turned off. Requires external pull-down. Capable of positive-logic wired-OR connection.	
	Passive-pull-down output is similar to N-P-N open-emitter output but is supplemented with a built-in passive pull-down.	
	3-state output.	
	Output with more than usual output capability (symbol is oriented in the direction of signal flow).	
	Enable input When at its internal 1-state, all outputs are enabled. When at its internal 0-state, open-collector and open-emitter outputs are off, three-state outputs are at normally defined internal logic states and at external high-impedance state, and all other outputs (e.g., totem-poles) are at the internal 0-state.	
J, K, R, S, T	Usual meanings associated with flip-flops (e.g., R = reset, T = toggle)	
	Data input to a storage element equivalent to: 	
	Shift right (left) inputs, m = 1, 2, 3, etc. If m = 1, it is usually not shown.	
	Counting up (down) inputs, m = 1, 2, 3, etc. If m = 1, it is usually not shown.	
	Binary grouping. m is highest power of 2.	
	The contents-setting input, when active, causes the content of a register to take on the indicated value.	
	The content output is active if the content of the register is as indicated.	
	Input line grouping . . . indicates two or more terminals used to implement a single logic input. e.g., The paired expander inputs of SN7450. 	
	Fixed-state output always stands at its internal 1 state. For example, see SN74185.	

In an array of elements, if the same general qualifying symbol and the same qualifying symbols associated with inputs and outputs would appear inside each of the elements of the array, these qualifying symbols are usually shown only in the first element. This is done to reduce clutter and to save time in recognition. Similarly, large identical elements that are subdivided into smaller elements may each be represented by an unsubdivided outline. The SN54HC242 or SN54LS440 symbol illustrates this principle.

3.3 Symbols Inside the Outline

Table III shows some symbols used inside the outline. Note particularly that open-collector (open-drain), open-emitter (open-source), and three-state outputs have distinctive symbols. Also note that an EN input affects all of the outputs of the circuit and has no effect on inputs. When an enable input affects only certain outputs and/or affects one or more inputs, a form of dependency notation will indicate this (see 4.9). The effects of the EN input on the various types of outputs are shown.

It is particularly important to note that a D input is always the data input of a storage element. At its internal 1 state, the D input sets the storage element to its 1 state, and at its internal 0 state it resets the storage element to its 0 state.

The binary grouping symbol will be explained more fully in Section 8. Binary-weighted inputs are arranged in order and the binary weights of the least-significant and the most-significant lines are indicated by numbers. In this document weights of input and output lines will be represented by powers of two usually only when the binary grouping symbol is used, otherwise decimal numbers will be used. The grouped inputs generate an internal number on which a mathematical function can be performed or that can be an identifying number for dependency notation (Figure 28). A frequent use is in addresses for memories.

Reversed in direction, the binary grouping symbol can be used with outputs. The concept is analogous to that for the inputs and the weighted outputs will indicate the internal number assumed to be developed within the circuit.

Other symbols are used inside the outlines in accordance with the IEC/IEEE standards but are not shown here. Generally these are associated with arithmetic operations and are self-explanatory.

When nonstandardized information is shown inside an outline, it is usually enclosed in square brackets [like these].

4.0 DEPENDENCY NOTATION

4.1 General Explanation

Dependency notation is the powerful tool that sets the IEC symbols apart from previous systems and makes compact, meaningful, symbols possible. It provides the means of denoting the relationship between inputs, outputs, or inputs and outputs without actually showing all the elements and interconnections involved. The information provided by dependency notation supplements that provided by the qualifying symbols for an element's function.

In the convention for the dependency notation, use will be made of the terms ''affecting'' and ''affected.'' In cases where it is not evident which inputs must be considered as being the affecting or the affected ones (e.g., if they stand in an AND relationship), the choice may be made in any convenient way.

So far, eleven types of dependency have been defined and all of these are used in various TI data books. X dependency is used mainly with CMOS circuits. They are listed below in the order in which they are presented and are summarized in Table IV following 4.12.

Section	Dependency Type or Other Subject
4.2	G, AND
4.3	General Rules for Dependency Notation
4.4	V, OR
4.5	N, Negate (Exclusive-OR)
4.6	Z, Interconnection
4.7	X, Transmission
4.8	C, Control
4.9	S, Set and R, Reset
4.10	EN, Enable
4.11	M, Mode
4.12	A, Address

4.2 G (AND) Dependency

A common relationship between two signals is to have them ANDed together. This has traditionally been shown by explicitly drawing an AND gate with the signals connected to the inputs of the gate. The 1972 IEC publication and the 1973 IEEE/ANSI standard showed several ways to show this AND relationship using dependency notation. While ten other forms of dependency have since been defined, the ways to invoke AND dependency are now reduced to one.

In Figure 4 input **b** is ANDed with input **a** and the complement of **b** is ANDed with **c**. The letter G has been chosen to indicate AND relationships and is placed at input **b**, inside the symbol. A number considered appropriate by the symbol designer (1 has been used here) is placed after the letter G and also at each affected input. Note the bar over the 1 at input **c**.

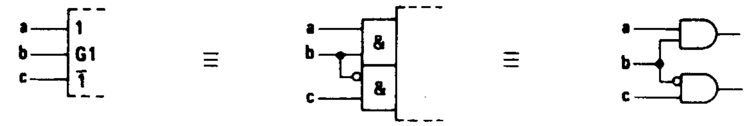


Figure 4. G Dependency Between Inputs

In Figure 5, output **b** affects input **a** with an AND relationship. The lower example shows that it is the internal logic state of **b**, unaffected by the negation sign, that is ANDed. Figure 6 shows input **a** to be ANDed with a dynamic input **b**.

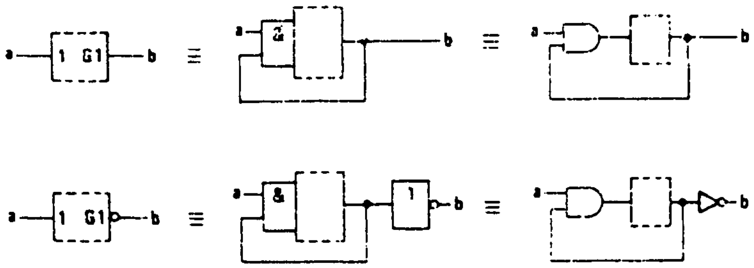


Figure 5. G Dependency Between Outputs and INputs

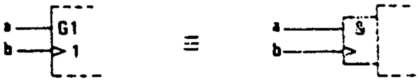


Figure 6. G Dependency with a Dynamic Input

The rules for G dependency can be summarized thus:

> When a G*m* input or output (*m* is a number) stands at its internal 1 state, all inputs and outputs affected by G*m* stand at their normally defined internal logic states. When the G*m* input or output stands at its 0 state, all inputs and outputs affected by G*m* stand at their internal 0 states.

4.3 Conventions for the Application of Dependency Notation in General

The rules for applying dependency relationships in general follow the same pattern as was illustrated for G dependency.

Application of dependency notation is accomplished by:

1) labeling the input (or output) *affecting* other inputs or outputs with the letter symbol indicating the relationship involved (e.g., G for AND) followed by an identifying number, appropriately chosen, and
2) labeling each input or output *affected* by that affecting input (or output) with that same number.

If it is the complement of the internal logic state of the affecting input or output that does the affecting, then a bar is placed over the identifying numbers at the affected inputs or outputs (Figure 4).

If two affecting inputs or outputs have the same letter and same identifying number, they stand in an OR relationship to each other (Figure 7).

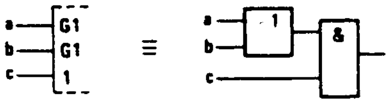


Figure 7. ORed Affecting Inputs

If the affected input or output requires a label to denote its function (e.g., "D"), this label will be *prefixed* by the identifying number of the affecting input (Figure 15).

If an input or output is affected by more than one affecting input, the identifying numbers of each of the affecting inputs will appear in the label of the affected one, separated by commas. The normal reading order of these numbers is the same as the sequence of the affecting relationships (Figure 15).

If the labels denoting the functions of affected inputs or outputs must be numbers (e.g., outputs of a coder), the identifying numbers to be associated with both affecting inputs and affected inputs or outputs will be replaced by another character selected to avoid ambiguity, e.g., Greek letters (Figure 8).

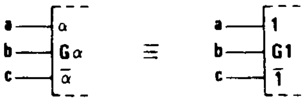


Figure 8. Substitution for Numbers

4.4 V (OR) Dependency

The symbol denoting OR dependency is the letter V (Figure 9).

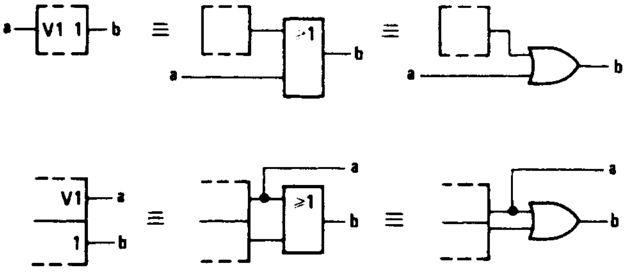
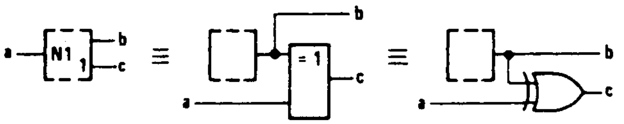


Figure 9. V (OR) Dependency

When a V*m* input or output stands at its internal 1 state, all inputs and outputs affected by V*m* stand at their internal 1 states. When the V*m* input or output stands at its internal 0 state, all inputs and outputs affected by V*m* stand at their normally defined internal logic states.

4.5 N (Negate) (Exclusive-OR) Dependency

The symbol denoting negate dependency is the letter N (Figure 10). Each input or output affected by an N*m* input or output stands in an Exclusive-OR relationship with the N*m* input or output.



If a = 0, then c = b
If a = 1, then c = b

Figure 10. N (Negate) (Exclusive-OR) Dependency

When an N*m* input or output stands at its internal 1 state, the internal logic state of each input and each output affected by N*m* is the complement of what it would otherwise be. When an N*m* input or output stands at its internal 0 state, all inputs and outputs affected by N*m* stand at their normally defined internal logic states.

4.6 Z (Interconnection) Dependency

The symbol denoting interconnection dependency is the letter Z.

Interconnection dependency is used to indicate the existence of internal logic connections between inputs, outputs, internal inputs, and/or internal outputs.

The internal logic state of an input or output affected by a Z*m* input or output will be the same as the internal logic state of the Z*m* input or output, unless modified by additional dependency notation (Figure 11).

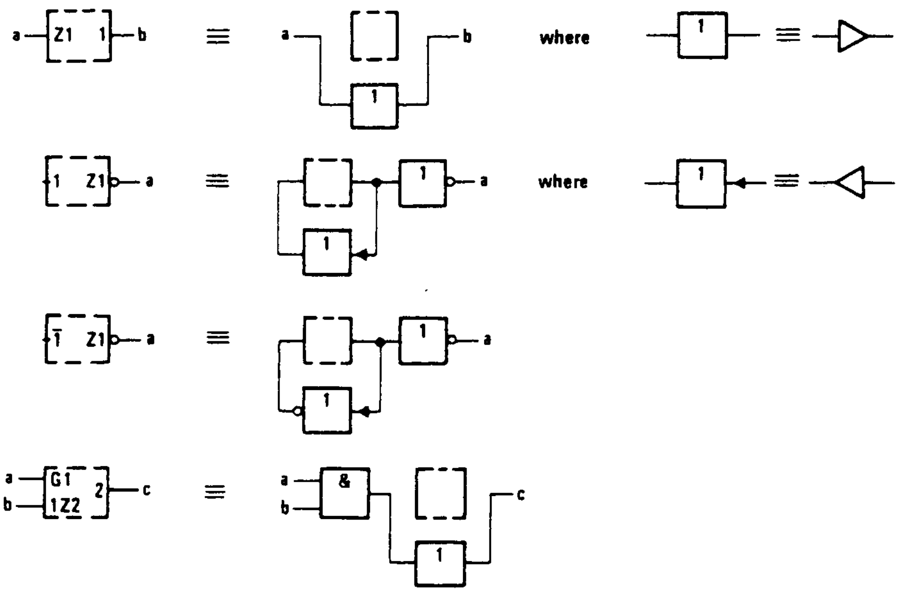
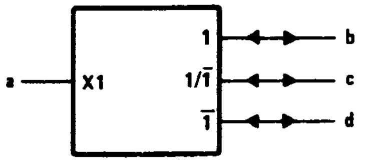


Figure 11. Z (Interconnection) Dependency

4.7 X (Transmission) Dependency

The symbol denoting transmission dependency is the letter X.

Transmission dependency is used to indicate controlled bidirectional connections between affected input/output ports (Figure 12).



If a = 1, there is a bidirectional connection between b and c.

If a = 0, there is a bidirectional connection between c and d.

Figure 12. X (Transmission) Dependency

When an X*m* input or output stands at its internal 1 state, all input-output ports affected by this X*m* input or output are bidirectionally connected together and stand at the same internal logic state or analog signal level. When an X*m* input or output stands at its internal 0 state, the connection associated with this set of dependency notation does not exist.

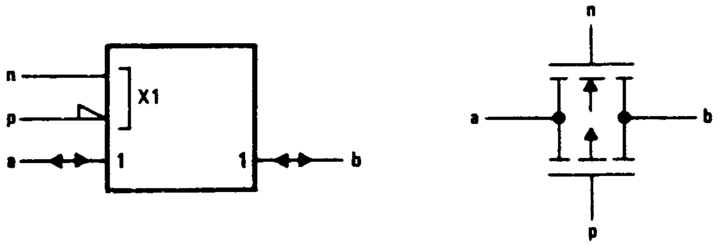


Figure 13. CMOS Transmission Gate Symbol and Schematic

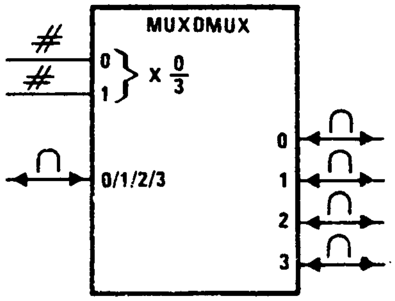


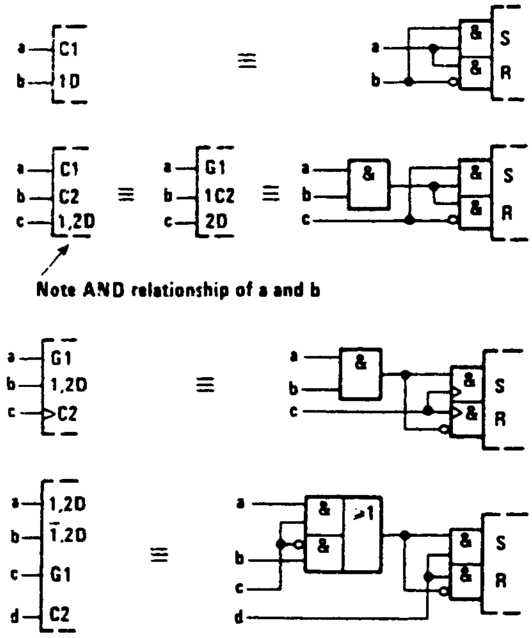
Figure 14. Analog Data Selector (Multiplexer/Demultiplexer)

Although the transmission paths represented by X dependency are inherently bidirectinal, use is not always made of this property. This is analogous to a piece of wire, which may be constrained to carry current in only one direction. If this is the case in a particular application, then the directional arrows shown in Figures 12, 13, and 14 would be omitted.

4.8 C (Control) Dependency

The symbol denoting control dependency is the letter C.

Control inputs are usually used to enable or disable the data (D, J, K, R, or S) inputs of storage elements. They may take on their internal 1 states (be active) either statically or dynamically. In the latter case the dynamic input symbol is used as shown in the third example of Figure 15.



Note AND relationship of a and b

Input c selects which of a or b is stored when d goes low.

Figure 15. C (Control) Dependency

When a C*m* input or output stands at its internal 1 state, the inputs affected by C*m* have their normally defined effect on the function of the element, i.e., these inputs are enabled. When a C*m* input or output stands at its internal 0 state, the inputs affected by C*m* are disabled and have no effect on the function of the element.

4.9 S (Set) and R (Reset) Dependencies

The symbol denoting set dependency is the letter S. The symbol denoting reset dependency is the letter R.

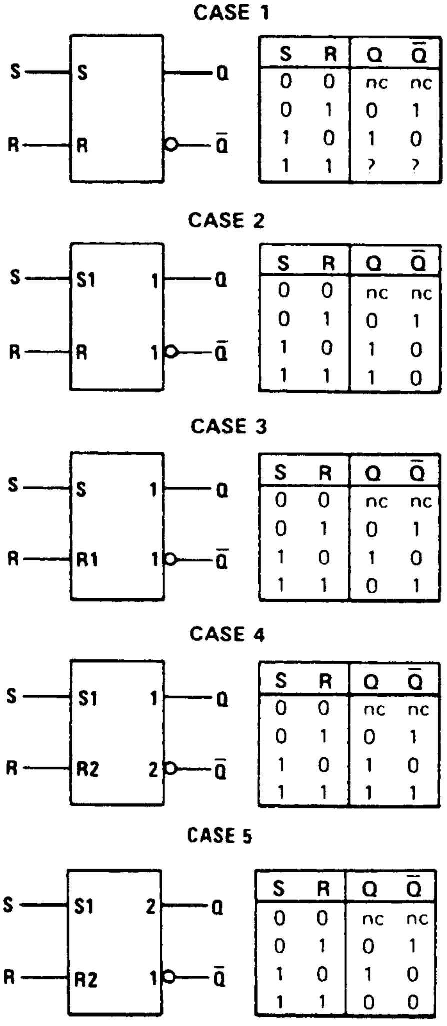
Set and reset dependencies are used if it is necessary to specify the effect of the combination R = S = 1 on a bistable element. Case 1 in Figure 16 does not use S or R dependency.

When an S*m* input is at its internal 1 state, outputs affected by the S*m* input will react, regardless of the state of an R input, as they normally would react to the combination S = 1, R = 0. See cases 2, 4, and 5 in Figure 16.

When an R*m* input is at its internal 1 state, outputs affected by the R*m* input will react, regardless of the state of an S input, as they normally would react to the combination S = 0, R = 1. See cases 3, 4, and 5 in Figure 16.

When an S*m* or R*m* input is at its internal 0 state, it has no effect.

Note that the noncomplementary output patterns in cases 4 and 5 are only pseudo stable. The simultaneous return of the inputs to S = R = 0 produces an unforeseeable stable and complementary output pattern.



CASE 1

S	R	Q	$\bar{Q}$
0	0	nc	nc
0	1	0	1
1	0	1	0
1	1	?	?

CASE 2

S	R	Q	$\bar{Q}$
0	0	nc	nc
0	1	0	1
1	0	1	0
1	1	1	0

CASE 3

S	R	Q	$\bar{Q}$
0	0	nc	nc
0	1	0	1
1	0	1	0
1	1	0	1

CASE 4

S	R	Q	$\bar{Q}$
0	0	nc	nc
0	1	0	1
1	0	1	0
1	1	1	1

CASE 5

S	R	Q	$\bar{Q}$
0	0	nc	nc
0	1	0	1
1	0	1	0
1	1	0	0

0 = external 0 state 1 = external 1 state

nc = no change ? = unspecified

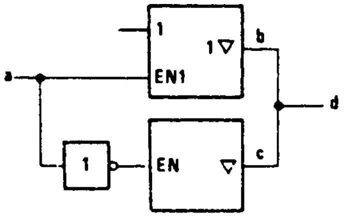
Figure 16. S (Set) and R (Reset) Dependencies

4.10 EN (Enable) Dependency

The symbol denoting enable dependency is the combination of letters EN.

An EN*m* input has the same effect on outputs as an EN input, see 3.1, but it affects only those outputs labeled with the identifying number *m*. It also affects those inputs labeled with the identifying number *m*. By contrast, an EN input affects all outputs and no inputs. The effect of an EN*m* input on an affected input is identical to that of a C*m* input (Figure 17).

When an EN*m* input stands at its internal 1 state, the inputs affected by EN*m* have their normally defined effect on the function of the element and the outputs affected by this input stand at their normally defined internal logic states, i.e., these inputs and outputs are enabled.



If a = 0, b is disabled and d = c
If a = 1, c is disabled and d = b

Figure 17. EN (Enable) Dependency

When an EN*m* input stands at its internal 0 state, the inputs affected by EN*m* are disabled and have no effect on the function of the element, and the outputs affected by EN*m* are also disabled. Open-collector outputs are turned off, three-state outputs stand at their normally defined internal logic states but externally exhibit high impedance, and all other outputs (e.g., totem-pole outputs) stand at their internal 0 states.

4.11 M (MODE) Dependency

The symbol denoting mode dependency is the letter M.

Mode dependency is used to indicate that the effects of particular inputs and outputs of an element depend on the mode in which the element is operating.

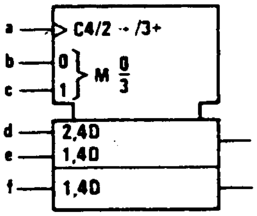
If an input or output has the same effect in different modes of operation, the identifying numbers of the relevant affecting M*m* inputs will appear in the label of that affected input or output between parentheses and separated by solidi (Figure 22).

4.11.1 M Dependency Affecting Inputs

M dependency affects inputs the same as C dependency. When an M*m* input or M*m* output stands at its internal 1 state, the inputs affected by this M*m* input or M*m* output have their normally defined effect on the function of the element, i.e., the inputs are enabled.

When an M*m* input or M*m* output stands at its internal 0 state, the inputs affected by this M*m* input or M*m* output have no effect on the function of the element. When an affected input has several sets of labels separated by solidi (e.g., C4/2—/3+), any set in which the identifying number of the M*m* input or M*m* output appears has no effect and is to be ignored. This represents disabling of some of the functions of a multifunction input.

The circuit in Figure 18 has two inputs, b and c, that control which one of four modes (0, 1, 2, or 3) will exist at any time. Inputs d, e, and f are D inputs subject to dynamic control (clocking) by the a input. The numbers 1 and 2 are in the series chosen to indicate the modes so inpu e and f are only enabled in mode 1 (for parallel loading) and input d is only enabled in mode 2 (for serial loading). Note that input a has three functions. It is the clock for entering data. In mode 2, it causes right shifting of data, which means a shift away from the control block. In mode 3, it causes the contents of the register to be incremented by one count.



Note that all operations are synchronous.

In MODE 0 (b = 0, c = 0), the outputs remain at their existing states as none of the inputs has an effect.

In MODE 1 (b = 1, c = 0), parallel loading takes place thru inputs e and f.

In MODE 2 (b = 0, c = 1), shifting down and serial loading thru input d take place.

In MODE 3 (b = c = 1), counting up by increment of 1 per clock pulse takes place.

Figure 18. M (Mode) Dependency Affecting Inputs

4.11.2 M Dependency Affecting Outputs

When an M*m* input or M*m* output stands at its internal 1 state, the affected outputs stand at their normally defined internal logic states, i.e., the outputs are enabled.

When an M*m* input or M*m* output stands at its internal 0 state, at each affected output any set of labels containing the identifying number of that M*m* input or M*m* output has no effe and is to be ignored. When an output has several different sets of labels separated by solidi (e.g., 2,4/3,5), only those sets in which the identifying number of this M*m* input or M*m* output appears are to be ignored.

Figure 19 shows a symbol for a device whose output can behave like either a 3-state output or an open-collector output depending on the signal applied to input a. Mode 1 exists when input a stands at its internal 1 state and, in that case, the three-state symbol applies and the open-element symbol has no effect. When a = 0, mode 1 does not exist so the three-state symbol has no effect and the open-element symbol applies.

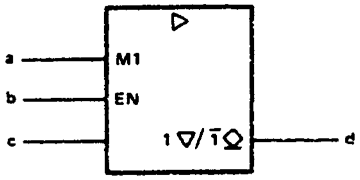


Figure 19. Type of Output Determined by Mode

In Figure 20, if input **a** stands at its internal 1 state establishing mode 1, output **b** will stand at its internal 1 state only when the content of the register equals 9. Since output **b** is located in the common-control block with no defined function outside of mode 1, the state of this output outside of mode 1 is not defined by the symbol.

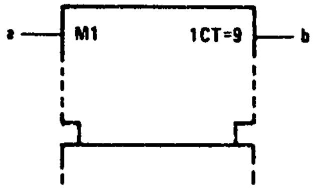


Figure 20. An Output of the Common-Control Block

In Figure 21, if input **a** stands at its internal 1 state establishing mode 1, output **b** will stand at its internal 1 state only when the content of the register equals 15. If input **a** stands at its internal 0 state, output **b** will stand at its internal 1 state only when the content of the register equals 0.

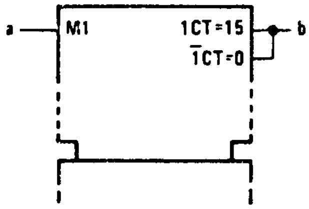


Figure 21. Determining and Output's Function

In Figure 22 inputs **a** and **b** are binary weighted to generate the numbers 0, 1, 2, or 3. This determines which one of the four modes exists.

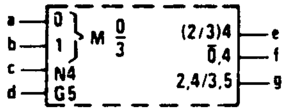


Figure 22. Dependent Relationships Affected by Mode

At output **e** the label set causing negation (if **c** = 1) is effective only in modes 2 and 3. In modes 0 and 1 this output stands at its normally defined state as if it had no labels. At output **f** the label set has effect when the mode is not 0 so output **e** is negated (if **c** = 1) in modes 1, 2, and 3. In mode 0 the label set has no effect so the output stands at its normally defined state. In this example $\overline{0}$,4 is equivalent to (1/2/3)4. At output **g** there are two label sets. The first set, causing negation (if **c** = 1), is effective only in mode 2. The second set, subjecting **g** to AND dependency on **d**, has effect only in mode 3.

Note that in mode 0 none of the dependency relationships has any effect on the outputs, so **e**, **f**, and **g** will all stand at the same state.

4.12 A (Address) Dependency

The symbol denoting address dependency is the letter A.

Address dependency provides a clear representation of those elements, particularly memories, that use address control inputs to select specified sections of a multildimensional arrays. Such a section of a memory array is usually called a word. The purpose of address dependency is to allow a symbolic presentation of the entire array. An input of the array shown at a particular element of this general section is common to the corresponding elements of all selected sections of the array. An output of the array shown at a particular element of this general section is the result of the OR function of the outputs of the corresponding elements of selected sections.

Inputs that are not affected by any affecting address input have their normally defined effect on all sections of the array, whereas inputs affected by an address input have their normally defined effect only on the section selected by that address input.

An affecting address input is labeled with the letter A followed by an identifying number that corresponds with the address of the particular section of the array selected by this input. Within the general section presented by the symbol, inputs and outputs affected by an A*m* input are labeled with the letter A, which stands for the identifying numbers, i.e., the addresses, of the particular sections.

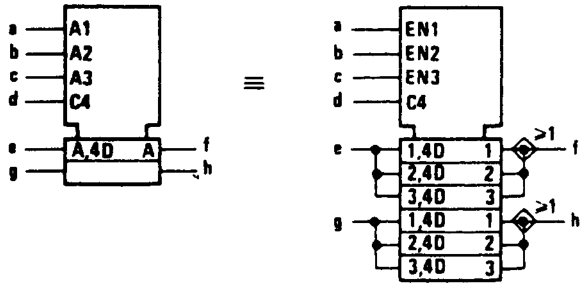


Figure 23. A (Address) Dependency

Figure 23 shows a 3-word by 2-bit memory having a separate address line for each word and uses EN dependency to explain the operation. To select word 1, input a is taken to its 1 state, which establishes mode 1. Data can now be clocked into the inputs marked "1,4D." Unless words 2 and 3 are also selected, data cannot be clocked in at the inputs marked "2,4D" and "3,4D." The outputs will be the OR functions of the selected outputs, i.e., only those enabled by the active EN functions.

The identifying numbers of affecting address inputs correspond with the addresses of the sections selected by these inputs. They need not necessarily differ from those of other affecting dependency-inputs (e.g., G, V, N, . . .), because in the general section presented by the symbol they are replaced by the letter A.

If there are several sets of affecting A*m* inputs for the purpose of independent and possibly simultaneous access to sections of the array, then the letter A is modified to 1A, 2A, ... Because they have access to the same sections of the array, these sets of A inputs may have the same identifying numbers. The symbols for 'HC170 or SN74LS170 make use of this.

Figure 24 is another illustration of the concept.

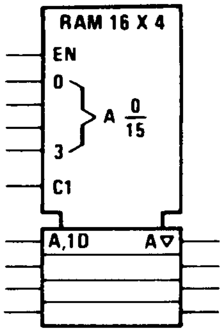


Figure 24. Array of 16 Sections of Four Transparent Latches with 3-State Outputs Comprising a 16-Word X 4-Bit Random-Access Memory

Table IV. Summary of Dependency Notation

TYPE OF DEPENDENCY	LETTER SYMBOL*	AFFECTING INPUT AT ITS 1-STATE	AFFECTING INPUT AT ITS 0-STATE
Address	A	Permits action (address selected)	Prevents action (address not selected)
Control	C	Permits action	Prevents action
Enable	EN	Permits action	Prevents action of inputs ◇outputs off ▽outputs at external high impedance, no change in internal logic state Other outputs at internal 0 state
AND	G	Permits action	Imposes 0 state
Mode	M	Permits action (mode selected)	Prevents action (mode not selected)
Negate (Ex-OR)	N	Complements state	No effect
Reset	R	Affected output reacts as it would to S = 0, R = 1	No effect
Set	S	Affected output reacts as it would to S = 1, R = 0	No effect
OR	V	Imposes 1 state	Permits action
Transmission	X	Bidirectional connection exists	Bidirectional connection does not exist
Interconnection	Z	Imposes 1 state	Imposes 0 state

*These letter symbols appear at the AFFECTING input (or output) and are followed by a number. Each input (or output) AFFECTED by that input is labeled with that same number. When the labels EN, R, and S appear at inputs without the following numbers, the descriptions above do not apply. The action of these inputs is described under "Symbols Inside the Outline," see 3.3.

5.0 BISTABLE ELEMENTS

The dynamic input symbol, the postponed output symbol, and dependency notation provide the tools to differentiate four main types of bistable elements and make synchronous and asynchronous inputs easily recognizable (Figure 25). The first column shows the essential distinguishing features; the other columns show examples.

Transparent latches have a level-operated control input. The D input is active as long as the C input is at its internal 1 state. The outputs respond immediately. Edge-triggered elements accept data from D, J, K, R, or S inputs on the active transition of C. Pulse-triggered elements require the setup of data before the start of the control pulse; the C input is considered static since the data must be maintained as long as C is at its 1 state. The output is postponec til C returns to its 0 state. The data-lock-out element is similar to the pulse-triggered version exuept that the C input is considered dynamic in that shortly after C goes through its active transition, the data inputs are disabled and data does not have to be held. However, the output is still postponed until the C input returns to its initial external level.

Notice that synchronous inputs can be readily recognized by their dependency labels (1D, 1J, 1K, 1S, 1R) compared to the asynchronous inputs (S, R), which are not dependent on the C ioputs.

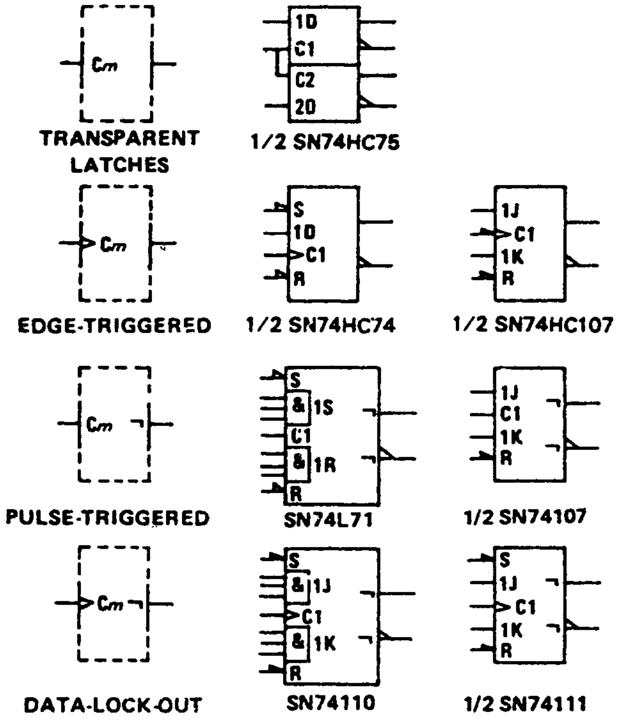


Figure 25. Four Types of Bistable Circuits

6.0 CODERS

The general symbol for a coder or code converter is shown in Figure 26. X and Y may be replaced by appropriate indications of the code used to represent the information at the inputs and at the outputs, respectively.

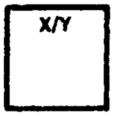


Figure 26. Coder General Symbol

Indication of code conversion is based on the following rule:

> Depending on the input code, the internal logic states of the inputs determine an internal value. This value is reproduced by the internal logic states of the outputs, depending on the output code.

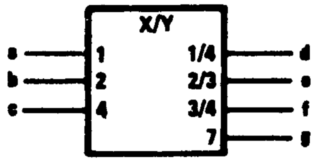
The indication of the relationships between the internal logic states of the inputs and the internal value is accomplished by:

1) labeling the inputs with numbers. In this case the internal value equals the sum of the weights associated with those inputs that stand at their internal 1-state, or by
2) replacing X by an appropriate indication of the input code and labeling the inputs with characters that refer to this code.

The relationships between the internal value and the internal logic states of the outputs are indicated by:

1) labeling each output with a list of numbers representing those internal values that lead to the internal 1-state of that output. These numbers shall be separated by solidi as in Figure 27. This labeling may also be applied when Y is replaced by a letter denoting a type of dependency (see Section 7). If a continuous range of internal values produces the internal 1 state of an output, this can be indicated by two numbers that are inclusively the beginning and the end of the range, with these two numbers separated by three dots (e.g., 4 . . . 9 = 4/5/6/7/8/9) or by
2) replacing Y by an appropriate indiction of the output code and labeling the outputs with characters that refer to this code as in Figure 28.

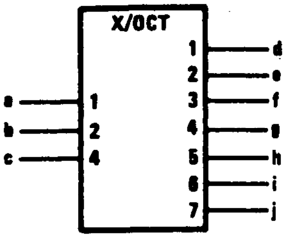
Alternatively, the general symbol may be used together with an appropriate reference to a table in which the relationship between the inputs and outputs is indicated. This is a recommended way to symbolize a PROM after it has been programmed.



FUNCTION TABLE

INPUTS			OUTPUTS			
c	b	a	g	f	e	d
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	1	0
0	1	1	0	1	1	0
1	0	0	0	1	0	1
1	0	1	0	0	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	0

Figure 27. An X/Y Code Converter



FUNCTION TABLE

INPUTS			OUTPUTS						
c	b	a	j	i	h	g	f	e	d
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	1	0
0	1	1	0	0	0	0	1	0	0
1	0	0	0	0	0	1	0	0	0
1	0	1	0	0	1	0	0	0	0
1	1	0	0	1	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0

Figure 28. An X/Octal Code Converter

7.0 USE OF A CODER TO PRODUCE AFFECTING INPUTS

It often occurs that a set of affecting inputs for dependency notation is produced by decoding the signals on certain inputs to an element. In such a case use can be made of the symbol for a coder as an embedded symbol (Figure 29).

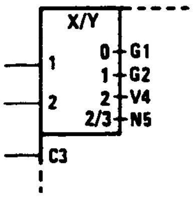


Figure 29. Producing Various Types of Dependencies

If all affecting inputs produced by a coder are of the same type and their identifying numbers shown at the outputs of the coder, Y (in the qualifying symbol X/Y) may be replaced by the letter denoting the type of dependency. The indications of the affecting inputs should then be omitted (Figure 30).

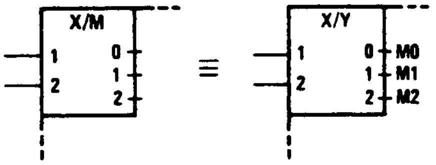


Figure 30. Producing One Type of Dependency

8.0 USE OF BINARY GROUPING TO PRODUCE AFFECTING INPUTS

If all affecting inputs produced by a coder are of the same type and have consecutive identifying numbers not necessarily corresponding with the numbers that would have been shown at the outputs of the coder, use can be made of the binary grouping symbol. *k* external lines effectively generate 2^k internal inputs. The bracket is followed by the letter denoting the type of dependency followed by m1/m2. The m1 is to be replaced by the smallest identifying number and the m2 by the largest one, as shown in Figure 31.

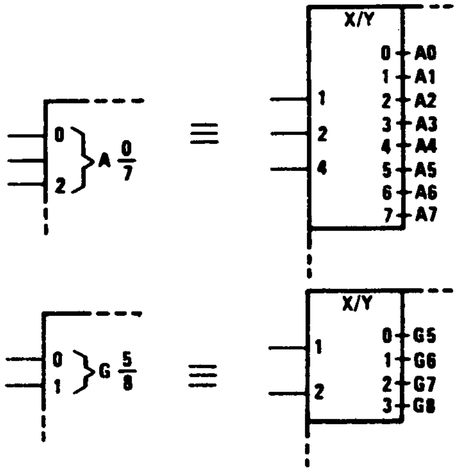


Figure 31. Use of the Binary Grouping Symbol

9.0 SEQUENCE OF INPUT LABELS

If an input having a single functional effect is affected by other inputs, the qualifying symbol (if there is any) for that functional effect is preceded by the labels corresponding to the affecting inputs. The left-to-right order of these preceding labels is the order in which the effects or modifications must be applied. The affected input has no functional effect on the element if the logic state of any one of the affecting inputs, considered separately, would cause the affected input to have no effect, regardless of the logic states of other affecting inputs.

If an input has several different functional effects or has several different sets of affecting inputs, depending on the mode of action, the input may be shown as often as required. However, there are cases in which this method of presentation is not advantageous. In those cases the input may be shown once with the different sets of labels separated by solidi (Figure 32). No meaning is attached to the order of these sets of labels. If one of the functional effects of an input is that of an unlabeled input to the element, a solidus will precede the first set of labels shown.

If all inputs of a combinational element are disabled (caused to have no effect on the function of the element), the internal logic states of the outputs of the element are not specified by the symbol. If all inputs of a sequential element are disabled, the content of this element is not changed and the outputs remain at their existing internal logic states.

Labels may be factored using algebraic techniques (Figure 33).

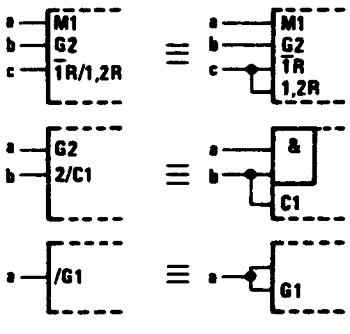


Figure 32. Input Labels

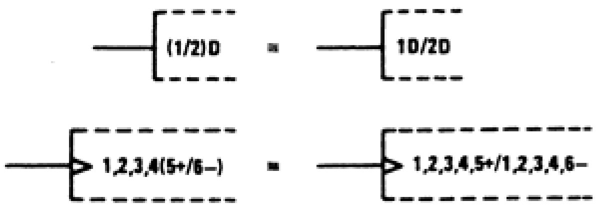


Figure 33. Factoring Input Labels

10.0 SEQUENCE OF OUTPUT LABELS

If an output has a number of different labels, regardless of whether they are identifying numbers of affecting inputs or outputs or not, these labels are shown in the following order:

1) If the postponed output symbol has to be shown, this comes first, if necessary preceded by the indications of the inputs to which it must be applied
2) Followed by the labels indicating modifications of the internal logic state of the output, such that the left-to-right order of these labels corresponds with the order in which their effects must be applied
3) Followed by the label indicating the effect of the output on inputs and other outputs of the element.

Symbols for open-circuit or three-state outputs, where applicable, are placed just inside the outside boundary of the symbol adjacent to the output line (Figure 34).

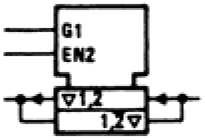


Figure 34. Placement of 3-State Symbols

If an output needs several different sets of labels that represent alternative functions (e.g., depending on the mode of action), these sets may be shown on different output lines that must be connected outside the outline. However, there are cases in which this method of presentation is not advantageous. In those cases the output may be shown once with the different sets of labels separated by solidi (Figure 35).

Two adjacent identifying numbers of affecting inputs in a set of labels that are not already separated by a nonnumeric character should be separated by a comma.

If a set of labels of an output not containing a solidus contains the identifying number of an affecting M*m* input standing at its internal 0 state, this set of labels has no effect on that output.

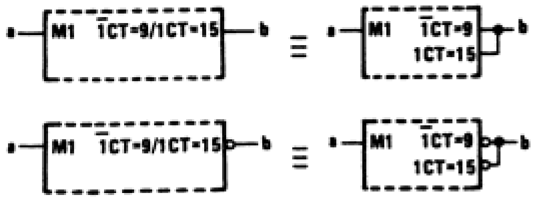


Figure 35. Output Labels

PART 4

Laboratory Solutions for *Experiments in Digital Fundamentals*

Tenth Edition

David Buchla

Table of Contents

Instructor Multisim Solutions

There are Multisim files for Multisim 9 and Multisim 10 files for seven experiments (5, 8, 10, 12, 18, 19, and 23) on the companion website. To access supplementary materials online, instructors need to request an instructor access code. Go to **www.pearsonhighered.com/irc**, where you can register for an instructor access code. Within 48 hours after registering, you will receive a confirming e-mail, including an instructor access code. Once you have received your code, go to the site and log on for full instructions on downloading the materials you wish to use.

The Multisim suffix is .ms9 (version 9) or .ms10 (version 10). The naming convention for all files is Exp-xxnf for files with no fault ("xx" refers to the experiment number). The "nf" in the file name is replaced with f1 and f2 for files with hidden faults. The circuit restrictions password that will allow faults to be revealed is **df10lm** for both Multisim versions. The circuit and fault descriptions are as follows:

File name	**Description**
Exp-05nf	The circuit is from Figure 5-2 and performs either a 1's or 2's complement depending on the position of the "Complement" switch.
Exp-05f1	The 330 Ω resistor on the output of D_1 is open.
Exp-05f2	The Complement switch is shorted.
***********	***********
Exp-08nf	The circuit is the combinational logic circuit given in Figure 8-4, the BCD invalid code detector.
Exp-08f1	The input from the D switch to NAND2 is open at the NAND gate.
Exp-08f2	Switch C is shorted.
***********	***********
Exp-10nf	The circuit is a simulation of the outlet valve for the model-1 Molasses tank given in Figure 10-2. To simulate the operation, open the L_L switch (tank is filling), then open the L_H switch (tank full). Close the L_H switch and note that the valve remains open until L_L is closed.
Exp-10f1	Resistor R_3 is 3.3 kΩ (value not shown).
Exp-10f2	NAND gate U2B has an open output.
***********	***********
Exp-12nf	The circuit is an overflow detection circuit, given as a design problem using gates in Experiment 11. A 74153 MUX is used in this problem.
Exp-12f1	The inverter has an open input.
Exp-12f2	The inverter has an input to output short.
***********	***********
Exp-18nf	The circuit is counter shown in Figure 18-6 with an oscilloscope. Glitches are easily seen in Multisim, but they are more difficult to see in lab.
Exp-18f1	The CLK input to the FFB is open.
Exp-18f2	The Q output of FFA is open.

***********	***********
Exp-19nf	The circuit is the J-K counter in Figure 19-3 with no faults. The logic analyzer is used in place of LEDs on the output.
Exp-19f1	The preset switch is shorted to ground.
Exp-19f2	The third NAND gate from the top has an open output.
***********	
Exp-23nf	The circuit is the J-K counter in Figure 19-3 with no faults. The logic analyzer is used in place of LEDs on the output.
Exp-23f1	The start bit flip-flop has an open on the CLR input.
Exp-23f2	The inverter has an open output.

Experiment 1: Laboratory Instrument Familiarization

Data and Observations

Data and the number of significant figures will vary according to signal generator and measurement equipment. Sample (measured) data is shown.

TABLE 1-1

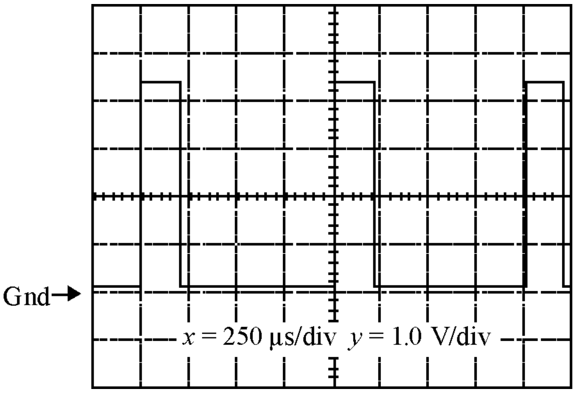
Voltage Setting = 5.0 V	Voltage Reading
Power Supply meter	*5.0 V*
DMM	*5.02 V*
Oscilloscope	*5.04 V*

TABLE 1-2

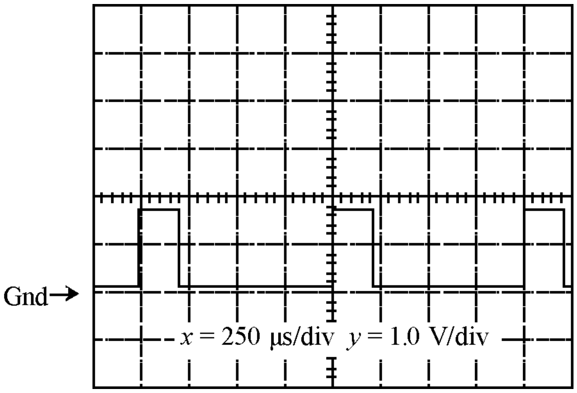
Function Generator Parameters (at 1.0 kHz)	Measured Values
Pulse Width	*197 µs*
Period	*1.00 ms*
Amplitude	*4.3 V*

TABLE 1-3

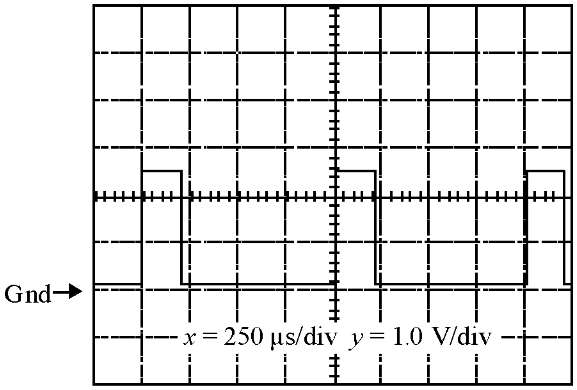
Step	Digital Oscillator Parameters	Measured Values
12	Period	*306 µs*
	Duty cycle	*78%*
	Amplitude	*3.75 V*
	Frequency	*3.27 kHz*
13	Period	*352 ms*
	Frequency	*2.84 Hz*



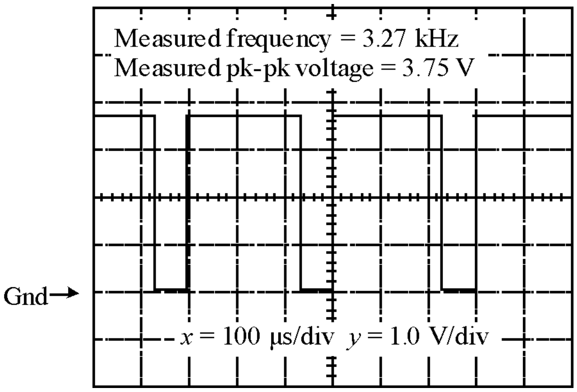
Plot 1 Generator waveform



Plot 2 Voltage across LED



Plot 3 Voltage across R_1



Plot 4 Digital Oscillator Output (pin 3)

Further Investigation Results:
With the partial failure consisting of a 100 Ω resistor in parallel with the LED, most of the current goes through the resistor (Figure 1-8b). This current can be traced through R_1 and R_2 and back to the supply using a logic pulser and current tracer as shown in Figure 1-9. With the short circuit (Figure 1-10), the current can be traced along the protoboard's bus, through the short and back to the supply with no current in the resistors or the LED.

Evaluation and Review Questions

1. The circuit can be damaged if the wrong voltage is connected to it.
2. Vertical section: sets the amplitude of the input signal and develops voltages for display section; it sends signals to the trigger section.
   Trigger section: causes the start of the acquisition of the waveform. A stable display requires the trigger to occur at the same point on the waveform for each acquisition.
   Horizontal section: controls the time base.
   Display section: changes the intensity or other display parameters.
3. A probe from each channel is connected across the ungrounded component. All grounds are connected to the circuit ground. The scope is configured to measure the difference between the two channels. (This will vary between scope types; it sometimes requires the channels to be added with one channel inverted).
4. (a) The oscillator works but the LED is off. The voltage at pin 3 and the LED are identical.
   (b) The oscillator works but the frequency is too low (depending on the capacitor).
   (c) Reversing the power and ground leads will result in destructively high current.
   (d) The oscillator works but the LED is off. No voltage appears at the LED (but pin 3 is okay).
5. A digital oscilloscope is typically as accurate as a DMM and allows detection of noise or ripple with a power supply. A DMM is typically more portable. An analog scope is not as accurate as a DMM but does not show potential problems such as noise or ripple. A digital scope may be as accurate as a DMM because of automated measurement capability and gives more information than a DMM about potential problems such as the presence of noise.
6. Pulses from the logic pulser are applied to the circuit board trace that normally carries power but with power removed. The current tracer can be moved along this line, following the path of current until the short is found. (Note that this technique can be used for any board, analog or digital).

Experiment 2: Constructing a Logic Probe

Data and Observations

Step 3: Logic thresholds HIGH *1.98* V LOW *0.76* V

TABLE 2-1

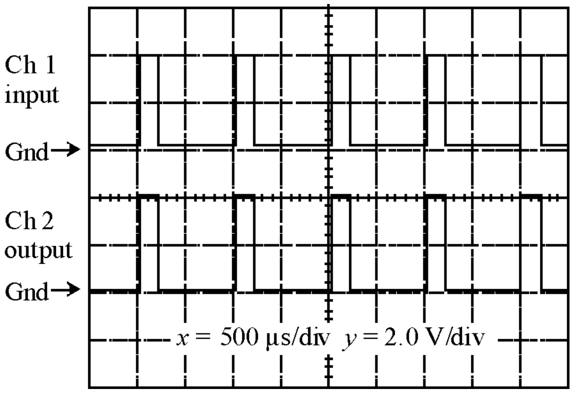
Step		Output Logic Level Input is LOW	 Input is OPEN	 Input is HIGH
4	one inverter	*HIGH*	*LOW*	*LOW*
5	two series inverters	*LOW*	*HIGH*	*HIGH*

TABLE 2-2

Step		Input Logic Level (pin 3)	Output Logic Level (pin 3)	Logic Level (pin 5)	Logic Level (pin 6)
7	V_{in} momentarily on ground.		*HIGH*		
8	V_{in} momentarily on +5.0 V.		*LOW*		
9	Fault condition: open at pin 5	*LOW*	*HIGH*	*INVALID*	*LOW*
10	Voltages with fault: DMM	*0.06 V*	*4.14 V*	*1.64 V*	*0.06 V*
11	Voltages with fault: (scope)	*0.06 V*	*4.18 V*	*1.65 V*	*0.06 V*

Further Investigation Results:

The signals are nearly identical as shown. In the experimental circuit, a slight difference in amplitude was noted and the output is delayed by 14 ns on the leading edge and 16 ns on the trailing edge. The time difference is due to transition time through the gates.



Plot 1

Evaluation and Review Questions:

1. Increase the value of the 330 Ω resistor in the input voltage divider.

2. Two inverters in series form a buffer that can allow additional drive current from the same logic output.

3. (a) The circuits are identical.
   (b) If the input to the first inverter is open, an invalid level of approximately 1.6 V will be observed.
   (c) If the input to the first inverter is open, then the output will be LOW.

4. A logic probe can quickly determine if a valid logic HIGH or LOW is present at some point in a circuit whereas a DMM can assign a numeric quantity to the level.

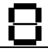
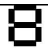
5. (a) V_{out} will alternate between a HIGH and LOW at a rapid rate.
   (b) Approximately 50 ns.
   (c) Since the period will be approximately 100 ns, the oscillation frequency is 10 MHz.

6. A steady state invalid level is an indication of an open input.

Experiment 3: Number Systems

Data and Observations

TABLE 3-1

Inputs		Output
Binary Number	BCD Number	Seven-Segment Display
0 0 0 0	***0000***	
0 0 0 1	***0001***	
0 0 1 0	***0010***	
0 0 1 1	***0011***	
0 1 0 0	***0100***	
0 1 0 1	***0101***	
0 1 1 0	***0110***	
0 1 1 1	***0111***	
1 0 0 0	***1000***	
1 0 0 1	***1001***	
1 0 1 0	INVALID	
1 0 1 1	INVALID	
1 1 0 0	INVALID	
1 1 0 1	INVALID	
1 1 1 0	INVALID	
1 1 1 1	INVALID	

Step 6. Method to cause leading zero suppression: The 7447A's ripple-blanking input for the most significant digit is connected to a LOW. If inputs *A*, *B*, *C*, *D* are also LOW, the display will be off and the ripple blanking output is LOW. Successive digits are blanked by connecting the ripple blanking output from the most significant digit to the next decoder's ripple-blanking input.

TABLE 3-2

Trouble Number	Trouble	Observations
1	LED for the *C* input is open	*The "C" switch has no effect on the output. The "C" input on the 7447A is always LOW.*
2	*A* input to 7447A is open	*Only odd numbers can be observed on the display because the open A input is interpreted as a HIGH.*
3	$\overline{\text{LAMP TEST}}$ is shorted to ground	*All segments are on; switches have no effect.*
4	Resistor connected to pin 15 of the 7447A is open	*Upper left (f) segment is always off. Numbers 0, 4, 5, 6, 8, 9 are shown incorrectly.*

Further Investigation Results:

The circuit needed is the same as Figure 3-2 except the MSB is disconnected and grounded at the BCD input to the 7447A. The LED used to indicate the D input can be connected to an LED that represents the octal MSB or it could be shown on the A input of a second 7447A decoder. The idea of this investigation is to reinforce converting a binary number to an octal number by grouping the binary bits by groups of three.

Evaluation and Review Questions:

1. Segment *g* open on seven segment display, open wire or resistor to *g* segment, bad 7447A.

2. Leave switches with binary 1000 and test logic at segment *g*. If the *g* segment is LOW, the segment is bad; if it is HIGH, move to pin 14 of 7447A and test the logic. If it is LOW, the resistive path is open; if it is HIGH, test inputs to 7447A.

3. All segments will be off.

4. Binary is a base two weighted number system. Column values increase by powers of two. BCD is a code that represents each decimal digit with 4 binary bits.

5.

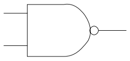
Binary	*Octal*	*Hexadecimal*	*Decimal*	*BCD*
01001100	*114*	*4C*	*76*	*0111 0110*
11000100	**304**	*C4*	*196*	*0001 1001 0110*
11100110	*346*	**E6**	*230*	*0010 0011 0000*
111001	*71*	*39*	**57**	*0101 0111*
110001	*61*	*31*	*49*	**0100 1001**

6. a. The base can be found algebraically by solving $1x^2 + 2x^1 + 5 = 85$ (x represents the base). The positive root (base) is 8.
   b. 16

Experiment 4: Logic Gates

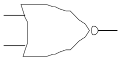
Data and Observations

TABLE 4-2
NAND Gate.



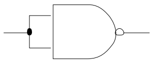
Inputs		Output	Measured Output Voltage
A	*B*	*X*	
0	0	1	3.95 V
0	1	1	3.95 V
1	0	1	3.95 V
1	1	0	0.06 V

TABLE 4-3
NOR Gate.



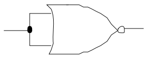
Inputs		Output	Measured Output Voltage
A	*B*	*X*	
0	0	1	4.01 V
0	1	0	0.06 V
1	0	0	0.06 V
1	1	0	0.06 V

TABLE 4-4
Truth table for Figure 4-4.



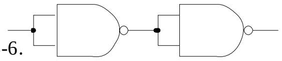
Input	Output	Measured Output Voltage
A	*X*	
0	1	3.95 V
1	0	0.06 V

TABLE 4-5
Truth table for Figure 4-5.



Input	Output	Measured Output Voltage
A	*X*	
0	1	4.01 V
1	0	0.06 V

TABLE 4-6
Truth table for Figure 4-6.



Input	Output	Measured Output Voltage
A	*X*	
0	0	0.05 V
1	1	3.97 V

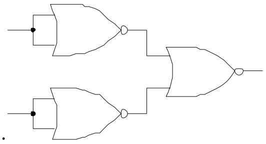


TABLE 4-7
Truth table for Figure 4-7.

Inputs		Output	Measured Output Voltage
A	*B*	*X*	
0	0	0	0.06 V
0	1	0	0.06 V
1	0	0	0.06 V
1	1	0	3.96 V

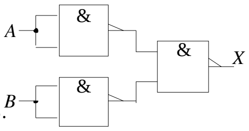
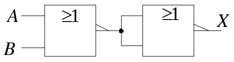


TABLE 4-8
Truth table for Figure 4-8.

Inputs		Output	Measured Output Voltage
A	*B*	*X*	
0	0	0	0.05 V
0	1	1	3.98 V
1	0	1	3.98 V
1	1	1	3.98 V

TABLE 4-9
Truth table for Figure 4-9.



Inputs		Output	Measured Output Voltage
A	*B*	*X*	
0	0	0	0.06 V
0	1	1	3.96 V
1	0	1	3.96 V
1	1	1	3.95 V

Further Investigation Results:

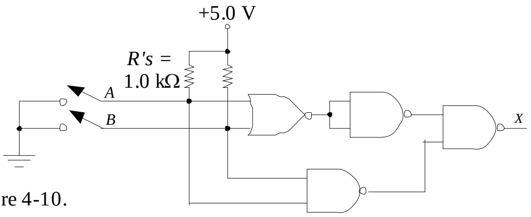


TABLE 4-10
Truth table for Figure 4-10.

Inputs		Output	Measured Output Voltage
A	*B*	*X*	
0	0	***1***	***3.97 V***
0	1	***0***	***0.06 V***
1	0	***0***	***0.06 V***
1	1	***1***	***3.97 V***

The equivalent gate is an XOR gate.

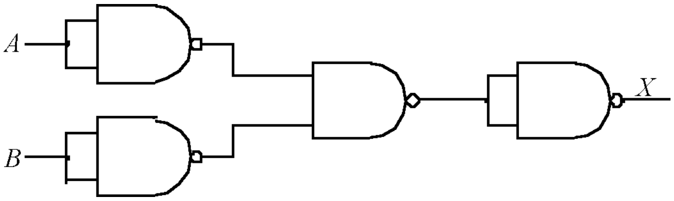
Evaluation and Review Questions:

1. a. Figures 4-4 and 4-5.
   b. Figure 4-7.
   c. Figures 4-8 and 4-9

2. Betty. A LOW on any input produces a LOW on the output. This describes an AND gate.

3. A HIGH on any input causes a HIGH output. This describes an OR gate.

4.



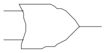
5. When the signal is HIGH, data is transmitted; when it is LOW, data is received.

6. The fact that an output is at a constant level does not in itself indicate a bad gate. If the circuit is not working properly, check the inputs first. If all inputs are HIGH, and the output is HIGH, the gate may be bad, but connections to the output should be checked to see if another component is holding the output HIGH.

Experiment 5: More Logic Gates

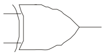
Data and Observations:

TABLE 5-2
OR Gate



Inputs		Output	Measured Output Voltage
A	*B*	*X*	
0	0	*0*	*0.07 V*
0	1	*1*	*3.93 V*
1	0	*1*	*3.93 V*
1	1	*1*	*3.93 V*

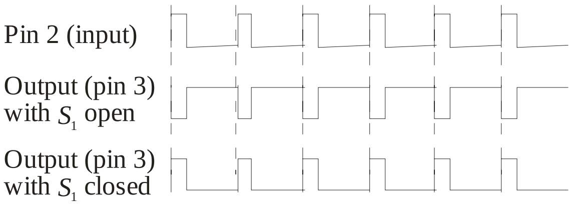
TABLE 5-3
XOR Gate



Inputs		Output	Measured Output Voltage
A	*B*	*X*	
0	0	*0*	*0.06 V*
0	1	*1*	*3.94 V*
1	0	*1*	*3.94 V*
1	1	*0*	*0.06 V*

TABLE 5-4

D_3	D_2	D_1	D_0	Q_3	Q_2	Q_1	Q_0
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*



PLOT 1

TABLE 5-5

Symptom Number	Symptom	Possible Cause
1	None of the LEDs operate; the switches have no effect.	*Power supply not on or open lead from the power or ground line to the board.*
2	LEDs on the output side do not work; those on the input side do work.	*Power or ground connection to the 7486 open.*
3	The LED representing Q_3 is sometimes on when it should be off.	*An input to the top XOR gate is likely to be open.*
4	The Complement switch has no effect on the outputs.	*Open or shorted connection from switch or a bad switch.*

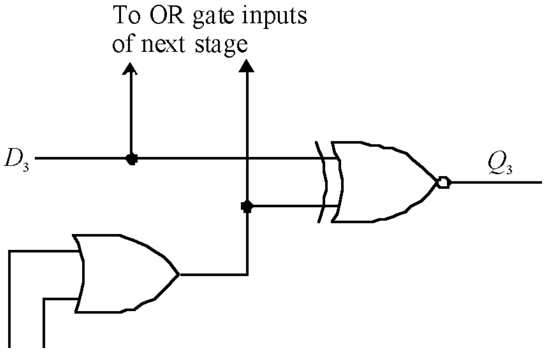
Further Investigation Results:
Each time a switch is thrown, no matter which one, the LED will change states (either on or off).

Evaluation and Review Questions:

1. One input is a control input and the other is the data. The data is unchanged if the control input is LOW but is inverted if the control input is HIGH. This is easily seen on a truth table to the right.

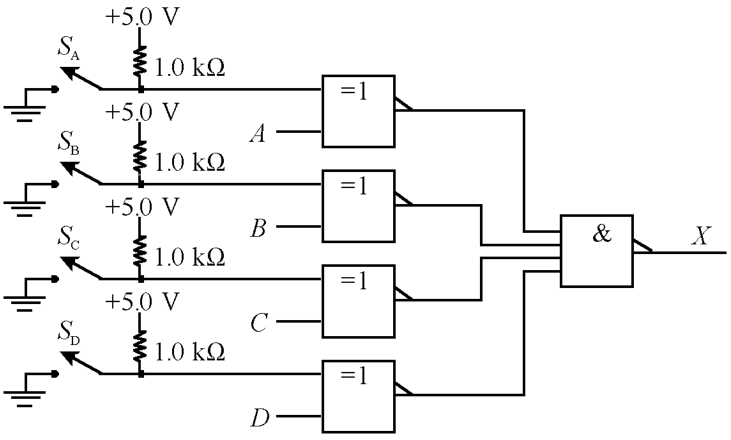
Control	Data	Output
0	0	0
0	1	1
1	0	1
1	1	0

2. The circuit can be expanded by "chaining" additional OR gates to D3 and the output of the upper OR gate as shown in the partial schematic to the right. The output of the OR gate is connected to the input of the next XOR gate. The other XOR input is connected to the data line as before.

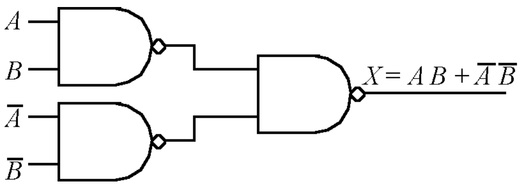


3. The output of an XNOR is HIGH when the inputs agree. If the *A B C D* inputs all agree with their respective switch settings, then all XOR gates have a HIGH output and the NAND gate output (labeled *X*) will be LOW; otherwise it will be HIGH.

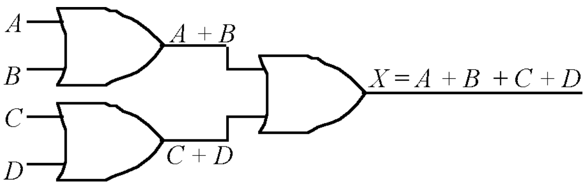
4. The comparator schematic, drawn with ANSI/IEEE Std 91-1984 symbols, is shown below.



5. *XOR* function with *NAND* gates:



6. Four input *OR* function with 2-input gates:



Experiment 6: Interpreting Manufacturer's Data Sheets

Data and Observations:

TABLE 6-1

TTL 7404

Recommended Operating Conditions									
		DM5405			DM7404			Units	Measured Value
Symbol	Parameter	Min	Nom	Max	Min	Nom	Max		
V_{CC}	Supply Voltage	4.5	5	5.5	4.75	5	5.25	V	
V_{IH}	High-Level Input Voltage	2			2			V	a. ***4.98*** **V**
V_{IL}	Low-Level Input Voltage			0.8			0.8	V	b. ***0.289*** **V**
I_{OH}	High-Level Output Current			-0.4			-0.4	mA	c. ***–0.245*** **mA**
I_{OL}	Low-Level Output Current			16			16	mA	d. ***14.5*** **mA**
T_A	Free Air Operating Temperature	-55		125	0		70	°C	

Electrical Characteristics Over Recommended Operating Free Air Temperature (unless otherwise noted)							
Symbol	Parameter	Conditions	Min	Typ	Max	Units	
V_I	Input clamp voltage	V_{CC}=Min, I_I=-12mA			-1.5	V	
V_{OH}	High-level output voltage	V_{CC}=Min, I_{OH}=Max V_{IL} =Max	2.4	3.4		V	e. ***3.68*** **V**
V_{OL}	Low Level Output Voltage	V_{CC}=Min, I_{OL}=Max V_{IH} =Min		0.2	0.4	V	f. ***0.222*** **V**
I_I	Input Current @ max Input Voltage	V_{CC}=Max, V_I =5.5 V			1	mA	

Electrical Characteristics Over Recommended Operating Free Air Temperature (unless otherwise noted)								
Symbol	Parameter	Conditions		Min	Type	Max	Units	Measured Value
I_{IH}	High-Level Input Current	V_{CC}=Max, V_I =2.4 V				40	μA	g. ***7.4*** **μA**
I_{IL}	Low-Level Input Current	V_{CC}=Max, V_I=0.4 V				-1.6	μA	h. ***–0.88*** **mA**
I_{OS}	Short Circuit Output Current	V_{CC}=Max	DM54	-20		-55	mA	
			DM74	-18		-55		
I_{CCH}	Supply Current With Outputs High	V_{CC}=Max			8	16	mA	
I_{CCL}	Supply Current With Outputs Low	V_{CC} = Max			14	27	mA	

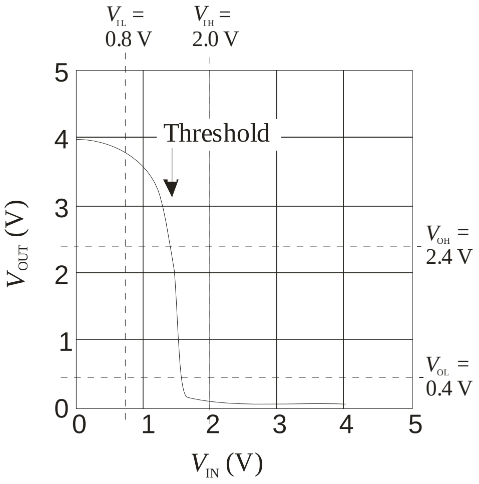
TABLE 6-2 CMOS 4081

	Quantity	Manufacturer’s Specified Value	Measured Value
(a)	$V_{OL(max)}$, low-level output voltage	***0.05*** **V**	***0.1*** **mV**
(b)	$V_{OH(min)}$, high-level output voltage	***4.95*** **V**	***4.99*** **V**
(c)	$V_{IL(max)}$, low-level input voltage	***1.5*** **V**	***1.5*** **V**
(d)	$V_{IH(min)}$, high-level output voltage	***3.5*** **V**	***3.5*** **V**
(e)	$I_{OL(min)}$, high-level output current	***0.51*** **mA**	***1.10*** **mA**
(f)	$I_{OH(min)}$, high-level output current	***–0.51*** **mA**	***–1.05*** **mA**
(g)	$I_{IN(typ)}$, input current	**±** $\boldsymbol{10^{-5}}$ **μA**	***0.04*** **μA***

* limited by instruments

Further Investigation Results:

V_{in} (V)	V_{out} (V)
0.4	***3.98***
0.8	***3.64***
1.2	***2.88***
1.3	***2.52***
1.4	***1.39***
1.5	***0.03***
1.6	***0.03***
2.0	***0.03***
2.4	***0.03***
2.8	***0.03***
3.2	***0.03***
3.6	***0.03***
4.0	***0.03***



PLOT 1

Evaluation and Review Questions:

1. A load causes the output LOW to be higher. (The measured LOW with no load was 30 mV; with the minimum load, the measured LOW was 222 mV).

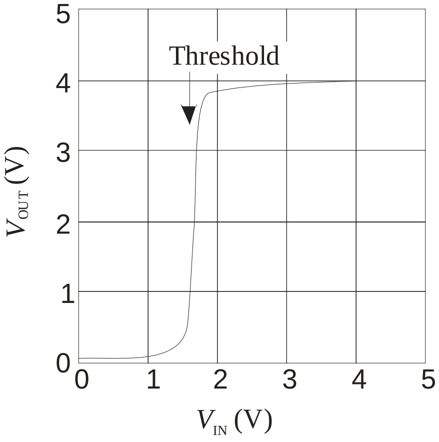
2. $R = 2.4\text{ V} / 0.4\text{ mA} = 6\text{ k}\Omega$.

3. $V_{NL(LOW)} = 0.3\text{ V}$
   $V_{NL(HIGH)} = 0.5\text{ V}$

4. a. Circuit (a) is better.
   b. The I_{OH} specification is exceeded in circuit (b). In (a), the LED is turned on with I_{OL} and is within the specified limit.

5. Logic levels are specified with reference to ground and can produce incorrect logic if the ground level is wrong. The scope should be dc coupled to assure the troubleshooter knows the ground level of the signal. (Note that in some digital scopes, the ground level is shown with a small arrow to the side of the display.)

6. The transfer curve for an AND gate (with inputs tied together) is shown below. The threshold is expected to occur at +1.6 V.



Plot for Question 6

Experiment 7: Boolean Laws and DeMorgan's Theorems

Data and Observations:

TABLE 7-2

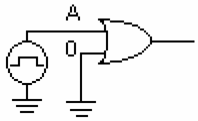
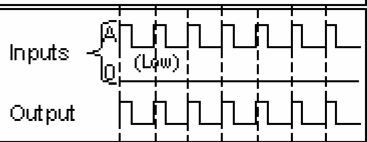
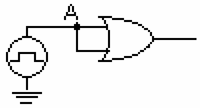
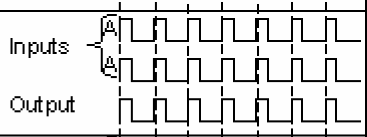
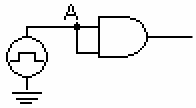
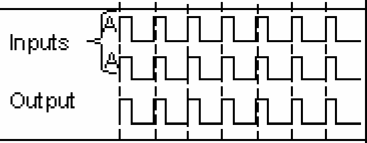
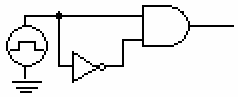
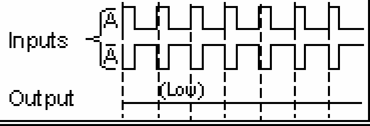
Schematic	Timing Diagram	Boolean Rule
		A + 0 = A
		A + A = A
		A • A = A
		A • $\overline{A}$ = 0

TABLE 7-3

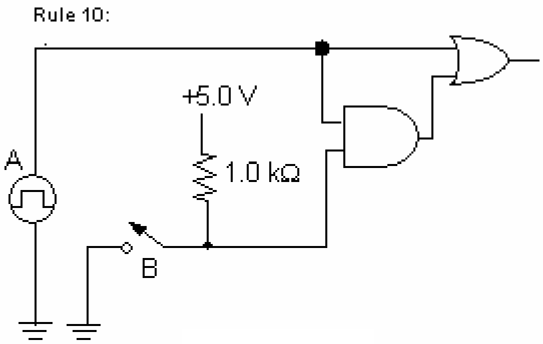
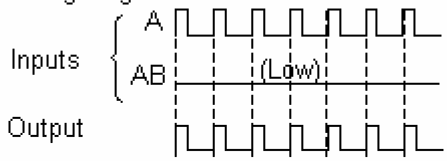
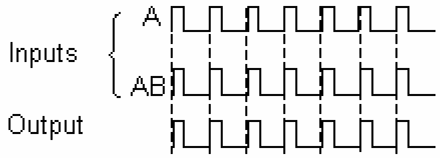
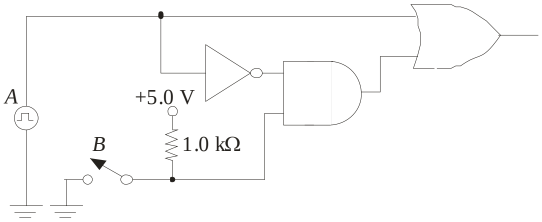
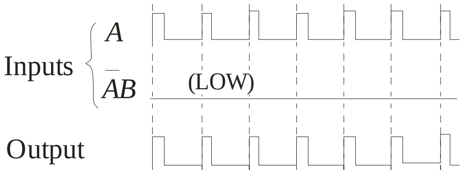
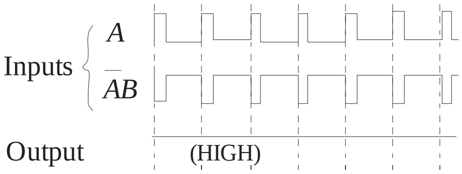
Schematic	Timing Diagram
 *A + AB = A*	Timing diagram for *B* = 0:  Timing diagram for *B* = 1: 

TABLE 7-4

Schematic	Timing Diagram
Rule 11:  $A + \overline{A}B = A + B$	Timing diagram for $B = 0$:  Timing diagram for $B = 1$: 

Further Investigation Results:

TABLE 7-5

Truth table for Figure 7-5

Inputs			Output
A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

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

TABLE 7-6

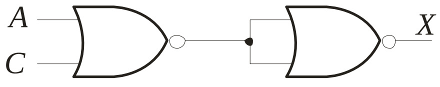
Truth table for Figure 7-6

Inputs			Output
A	*B*	*C*	*X*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

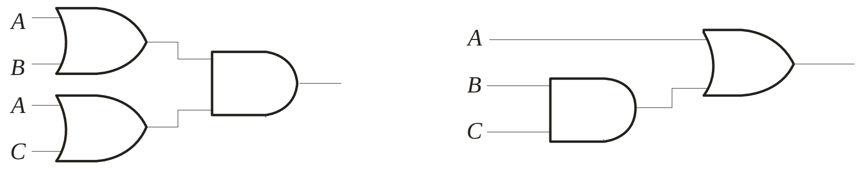
$X = \overline{\overline{A}B + C}$

Evaluation and Review Questions:

1. $A(A + B) + C = AA + AB + C$ (distribution)
   $= A + AB + C$ (rule 7)
   $= A + C$ (rule 10)

2. 

3. Rule 12: $(A + B)(A + C) = A + BC$



4. Because the two circuits have identical truth tables, they perform the same logic.

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

5. $(\overline{\overline{A} \cdot 1}) = A + 0 = A$

6. Answers vary but should follow a logical sequence. An example is:

   1. Verify that each IC has power and ground connected and that it is the correct voltage.
   2. Select an input that should turn on the LED (such as all switches closed to ground.) With this input, verify that the *A*, *B*, and *C* inputs to their respective gates are all LOW.
   3. With all inputs LOW, verify that the output the 4069 gates are both HIGH and that the output of the 4071 is HIGH. Verify that the output of the 4081 is HIGH.
   4. Check that the LED is inserted in the correct direction and that there is a path from the output of the 4081 through a 1.0 kΩ resistor through the LED to ground.

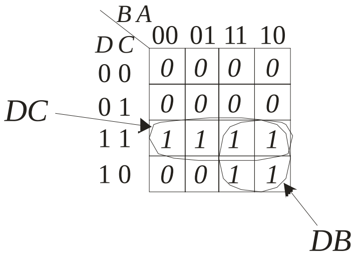
Experiment 8: Logic Circuit Simplification

Data and Observations

Note: Truth tables 8-2 and 8-3 are identical. Only Table 8-2 is shown.

TABLE 8-2
Truth table for BCD invalid code detector

Inputs				Output
D	*C*	*B*	*A*	*X*
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



B A *D C*	00	01	11	10
0 0	0	0	0	0
0 1	0	0	0	0
1 1	1	1	1	1
1 0	0	0	1	1

FIGURE 8-3
Karnaugh Map of truth table for the BCD invalid code detector

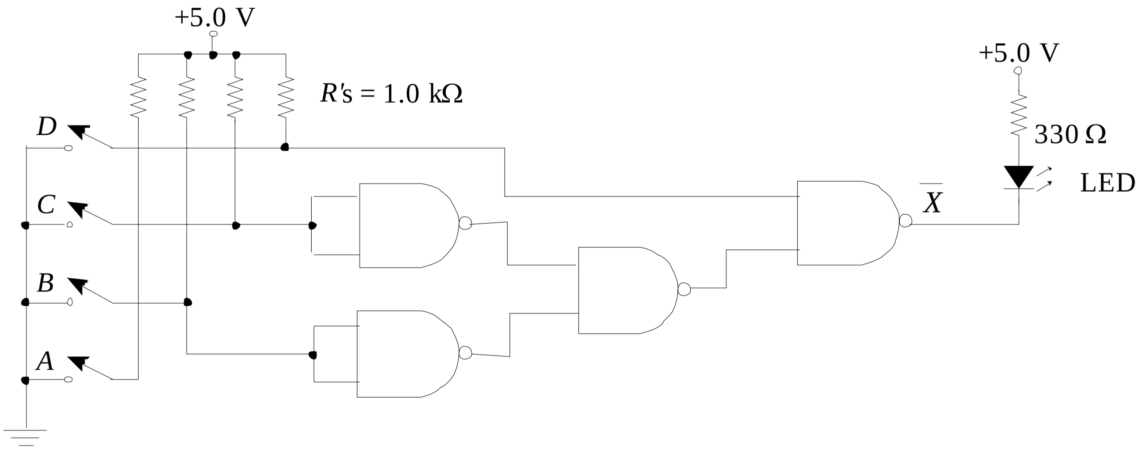
Minimum sum-of-products read from map:

X = DB + DC

Factoring *D* from both product terms gives:

X = D(B + C)

Step 5: Circuit for BCD invalid code detector (replacing OR gate with equivalent NAND gates):



Note that the A switch is not used in the invalid code detector.

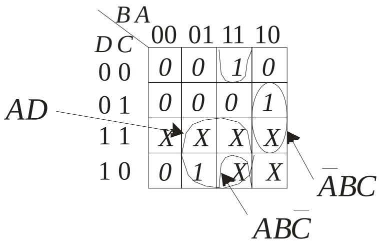
TABLE 8-4

Problem Number	Problem	Effect
1	The pull-up resistor for the *D* switch is open.	*The D input will be an invalid HIGH when the D switch is open. The circuit will perform normally except is susceptible to noise.*
2	The ground to the NAND gate in Figure 8-4 is open.	*The LED will not turn on under any conditions since there is no return path for LED current.*
3	A 3.3 kΩ resistor was accidentally used in place of the 330 Ω resistor.	*Circuit operates normally but LED is slightly dimmer (may depend on the LED).*
4	The LED was inserted backward.	*The LED will not turn on under any conditions.*
5	Switch A is shorted to ground.	*The A input is not used so it will have no effect.*

Further Investigation Results:

TABLE 8-5
Truth table for BCD numbers divisible by three

Inputs				Output
D	*C*	*B*	*A*	*X*
0	0	0	0	*0*
0	0	0	1	*0*
0	0	1	0	*0*
0	0	1	1	*1*
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	*1*
1	0	1	0	*X*
1	0	1	1	*X*
1	1	0	0	*X*
1	1	0	1	*X*
1	1	1	0	*X*
1	1	1	1	*X*



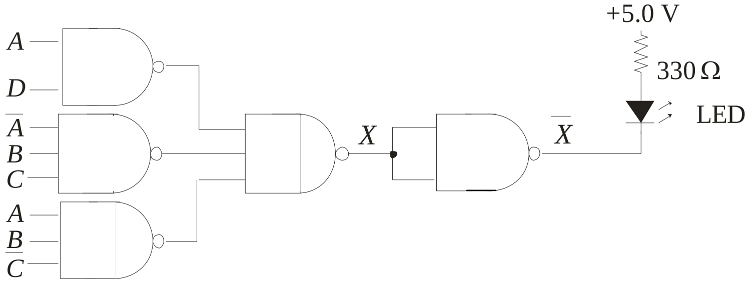
DC \ *BA*	00	01	11	10
0 0	*0*	*0*	*1*	*0*
0 1	*0*	*0*	*0*	*1*
1 1	*X*	*X*	*X*	*X*
1 0	*0*	*1*	*X*	*X*

FIGURE 8-5
Karnaugh Map of truth table for BCD numbers divisible by three.

Minimum sum-of-products read from map:

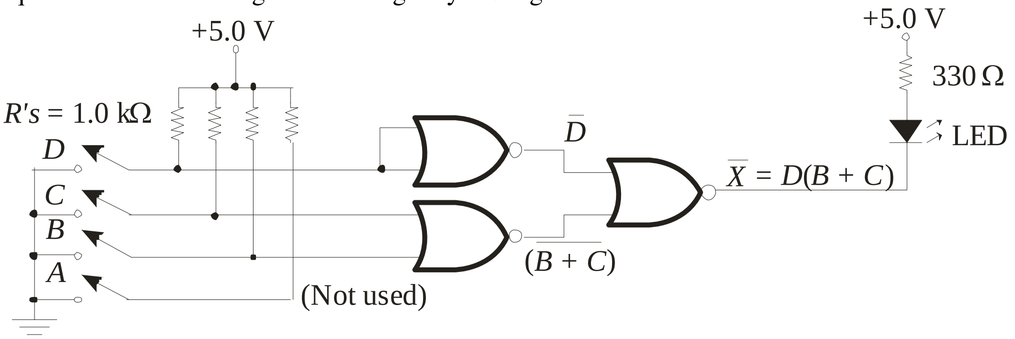
$$X = AD + AB\overline{C} + \overline{A}BC$$

Circuit:



Evaluation and Review Questions:

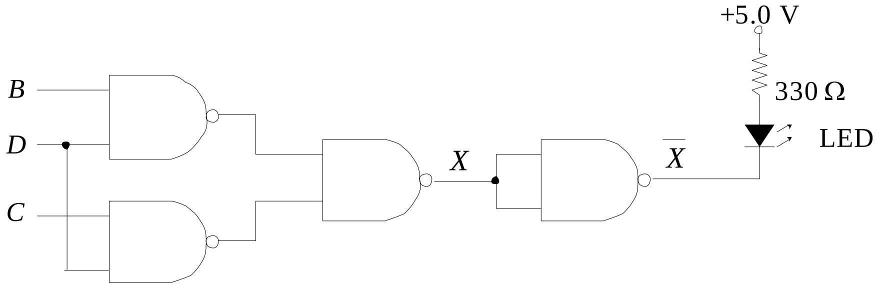
1. The path from the OR gate to the NAND gate may be open or the inputs to the NAND gate may both be shorted to *D*. To troubleshoot the problem, put the circuit in state 1000 or 1001 and test the logic at the OR gate output and at the input of the NAND gate.
2. Equivalent circuit for Figure 8-4 using only NOR gates:



3. The *A* input was not connected in the circuit in Figure 8-4 because the *A* variable was eliminated by Boolean algebra from the output expression.
4. $\overline{X} = \overline{D}\,(\overline{BC})$

   $X = \overline{\overline{D}\,(\overline{BC})}$

   $X = D + BC$
5. Sum of products (SOP) form $X = BD + CD$. Product of sums (POS form) $X = D(B+C)$. This can be done by factoring or by reading the 0's directly from the map (see Appendix *B* of text).
6. Implementation from the expression in step 2 ($X = BD + CD$) is shown in the schematic below:

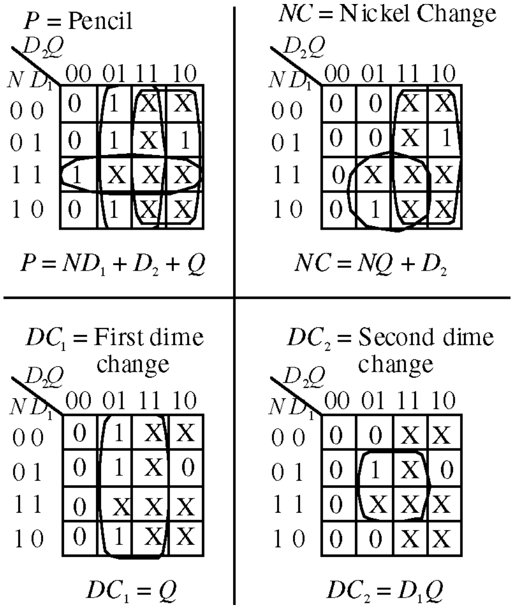


Experiment 9: The Perfect Pencil Machine

The model-2 of the perfect pencil machine is a combinational logic circuit that delivers a pencil or change for certain combinations of inputs representing coin switches. Coin switches are provided for one nickel, (N_1), two dimes (D_1 and D_2), and a quarter (Q). (Only one nickel switch is used to simplify the problem). The two dime switches are stacked on top of each other such that D_1 is always activated before D_2. The truth table has a number of don't care (X) entries, considered to be impossible inputs as described in the experiment. For example, D_2 cannot be inserted in the machine until D_1 has activated its switch and more than two coins cannot be entered in the machine. These "don't cares" allow the logic to be simplified considerably.

TABLE 9-2
Truth table for model-2 perfect pencil machine

Inputs				Outputs			
N_1	D_1	D_2	Q	P	NC	DC_1	DC_2
0	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0
0	0	1	0	X	X	X	X
0	0	1	1	X	X	X	X
0	1	0	0	0	0	0	0
0	1	0	1	1	0	1	1
0	1	1	0	1	1	0	0
0	1	1	1	X	X	X	X
1	0	0	0	0	0	0	0
1	0	0	1	1	1	1	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	1	0	0	0
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



P = Pencil

ND_1 \ D_2Q	00	01	11	10
0 0	0	1	X	X
0 1	0	1	X	1
1 1	1	X	X	X
1 0	0	1	X	X

$P = ND_1 + D_2 + Q$

NC = Nickel Change

ND_1 \ D_2Q	00	01	11	10
0 0	0	0	X	X
0 1	0	0	X	1
1 1	0	X	X	X
1 0	0	1	X	X

$NC = NQ + D_2$

DC_1 = First dime change

ND_1 \ D_2Q	00	01	11	10
0 0	0	1	X	X
0 1	0	1	X	0
1 1	0	X	X	X
1 0	0	1	X	X

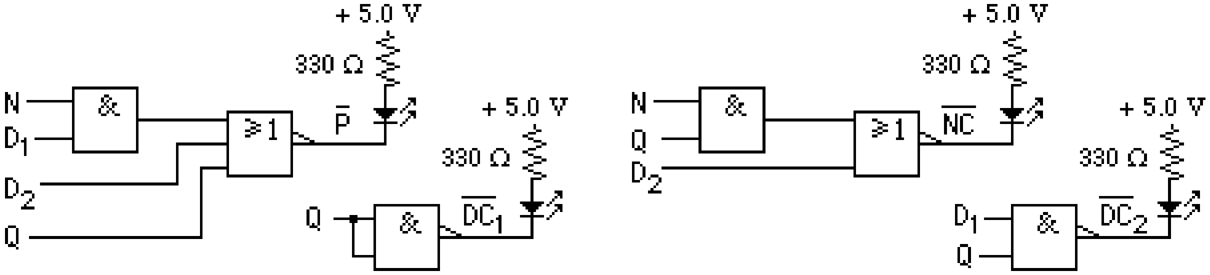
$DC_1 = Q$

DC_2 = Second dime change

ND_1 \ D_2Q	00	01	11	10
0 0	0	0	X	X
0 1	0	1	X	0
1 1	0	X	X	X
1 0	0	0	X	X

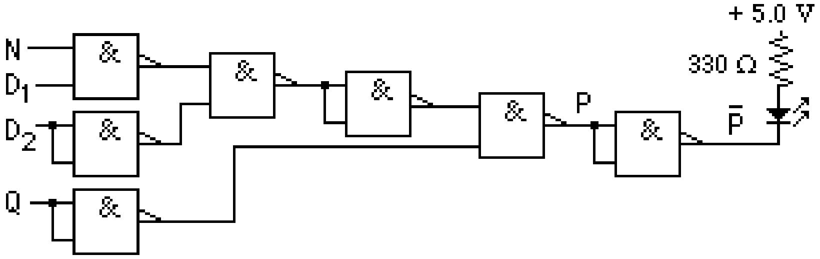
$DC_2 = D_1Q$

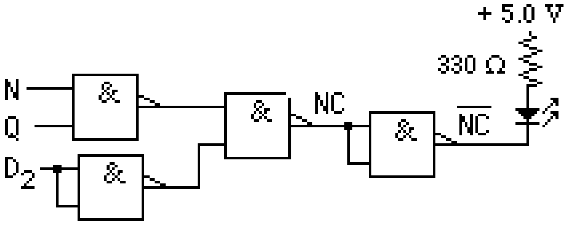
A simplified implementation is shown (switches not shown). The output logic from the maps has been inverted to light LEDs with a LOW rather than a HIGH in keeping with TTL specified I_O values.



Further Investigation Results:

This investigation requires the student to modify his or her experiment to use only 2-input NAND gates. The two circuits for dime change logic already meet that requirement. Modifications for the pencil and nickel change logic are shown below:

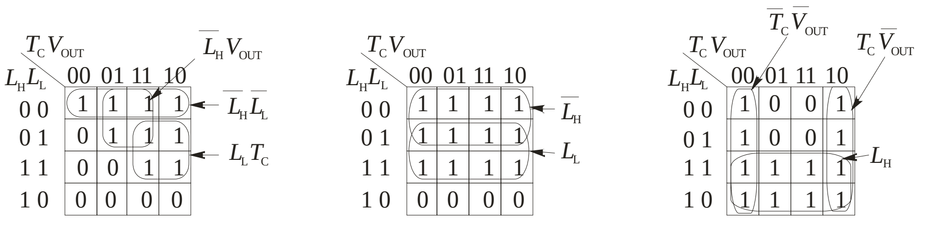




Experiment 10: The Molasses Tank

The model-2 Molasses Tank controller uses standard combinational logic but one of the inputs to the truth table is the output valve logic, V_{OUT}, to produce latching action using feedback. The truth table for the output valve is given in the experiment and the truth table for the alarm, A, and the input valve, V_{IN}, are given here. (The input valve is implemented in the Further Investigation with an extra Karnaugh map provided in the manual for this). Maps for these three outputs are also shown. (The heater output was not required in the experiment, but could be implemented as $H = T_C \cdot L_L$.) A circuit implementation using inverters and NAND gates is shown (note that A is implemented after applying DeMorgan's theorem). Active LOW outputs are used for the LEDs as given in the truth tables. Other implementations are possible.

Multisim files for the Model-1 controller are available on the website that was described previously. The file Exp-10nf illustrates how the feedback avoids the oscillation that could occur if the high level detector was the only control for the outlet valve by not closing the valve until the lower level is sensed.



$L_H L_L$ \ $T_C V_{OUT}$	00	01	11	10
0 0	1	1	1	1
0 1	0	1	1	1
1 1	0	0	1	1
1 0	0	0	0	0

$$V_{OUT} = \overline{L_H}\,\overline{L_L} + \overline{L_H} V_{OUT} + L_L T_C$$

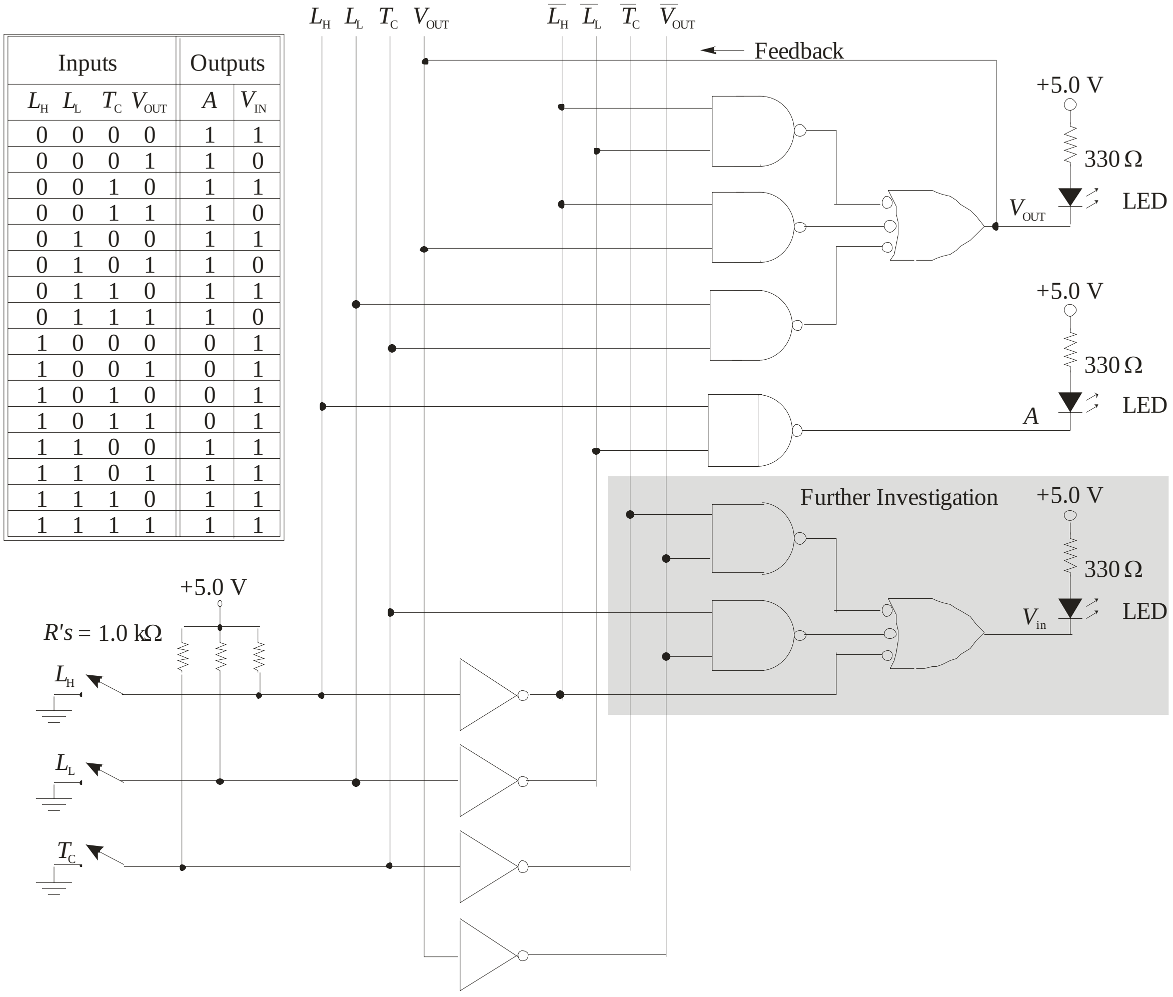
$L_H L_L$ \ $T_C V_{OUT}$	00	01	11	10
0 0	1	1	1	1
0 1	1	1	1	1
1 1	1	1	1	1
1 0	0	0	0	0

$$A = \overline{L_H} + L_L = \overline{L_H \cdot \overline{L_L}}$$

$L_H L_L$ \ $T_C V_{OUT}$	00	01	11	10
0 0	1	0	0	1
0 1	1	0	0	1
1 1	1	1	1	1
1 0	1	1	1	1

$$V_{IN} = \overline{T_C}\,\overline{V_{OUT}} + T_C \overline{V_{OUT}} + L_H$$

Inputs				Outputs	
L_H	L_L	T_C	V_{OUT}	A	V_{IN}
0	0	0	0	1	1
0	0	0	1	1	0
0	0	1	0	1	1
0	0	1	1	1	0
0	1	0	0	1	1
0	1	0	1	1	0
0	1	1	0	1	1
0	1	1	1	1	0
1	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	1	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1



Experiment 11: Adder and Magnitude Comparator

Data and Observations

The connections to the adder needed to complete the circuit in Figure 11-2 for a binary to excess-three conversion circuit are:

input:	connect to:
B_4	A>B
B_3	Ground
B_2	$\overline{A>B}$
B_1	+5.0 V
C_0	Ground

Results for the binary to excess-3 conversion are given in Table 11-4. The inverters are considered part of the display, therefore, logic shown on Table 11-4 is before the 7404 inverters (a logic 1 turns ON an LED).

Further Investigation Results:

TABLE 11-4
Truth table for Figure 11-2

Inputs (Binary)				Outputs (Excess-3)				
D	*C*	*B*	*A*	*A'*	*D*	*C*	*B*	*A*
0	0	0	0	0	0	0	1	1
0	0	0	1	0	0	1	0	0
0	0	1	0	0	0	1	0	1
0	0	1	1	0	0	1	1	0
0	1	0	0	0	0	1	1	1
0	1	0	1	0	1	0	0	0
0	1	1	0	0	1	0	0	1
0	1	1	1	0	1	0	1	0
1	0	0	0	0	1	0	1	1
1	0	0	1	0	1	1	0	0
1	0	1	0	1	0	0	1	1
1	0	1	1	1	0	1	0	0
1	1	0	0	1	0	1	0	1
1	1	0	1	1	0	1	1	0
1	1	1	0	1	0	1	1	1
1	1	1	1	1	1	0	0	0

TABLE 11-5
Truth table for overflow error.

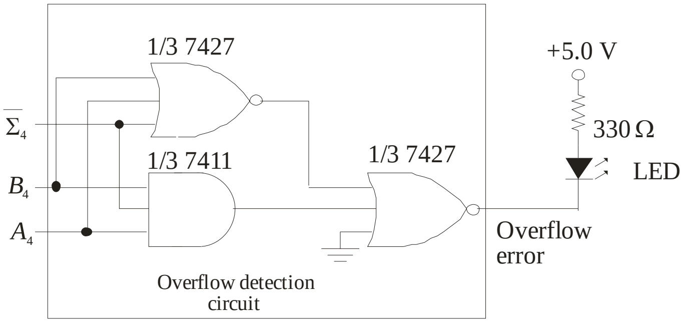
Sign Bits			Error
A_4	B_4	Σ_4	*X*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

A_4B_4 \ Σ_4	0	1
0 0	0	1
0 1	0	0
1 1	1	0
1 0	0	0

FIGURE 11-5
Karnaugh map for overflow error.

$$X = A_4B_4\overline{\Sigma_4} + \overline{A_4}B_4\Sigma_4$$

$$X = A_4B_4\overline{\Sigma_4} + \overline{A_4B_4\overline{\Sigma_4}}$$



Evaluation and Review Questions:

1. a. The input to the 7404 will see an illegal logic HIGH, causing the *A*' LED to be on.
   b. Approx 1.6 V.
2. The carry-out (C_4) of the lower order adder is connected to the carry-in (C_0) of the higher order adder. The schematic is shown in text as Figure 6-12(a).
3. It is the carry-in.
4. Connect the *B* inputs of the upper 7485 to 1000 and the lower comparator *B* inputs to 1100. Connect cascading inputs on both so that *A*=*B* inputs are HIGH; all other cascading inputs are LOW.
5. To form the two's complement, invert the bits of the subtrahend and add one by connecting the carry-in to a HIGH.
6. An additional voting switch can be connected to the carry-in logic of the parallel adders.

Experiment 12: Combinational Logic Using Multiplexers

Data and Observations

TABLE 12-1

Truth table for 2-bit comparator $A \geq B$.

Inputs				Output	Connect Data to:
A_2	A_1	B_2	B_1	X	
0	0	0	0	1	$\overline{B_1}$
0	0	0	1	0	
0	0	1	0	0	0
0	0	1	1	0	
0	1	0	0	1	1
0	1	0	1	1	
0	1	1	0	0	0
0	1	1	1	0	
1	0	0	0	1	1
1	0	0	1	1	
1	0	1	0	1	$\overline{B_1}$
1	0	1	1	0	
1	1	0	0	1	1
1	1	0	1	1	
1	1	1	0	1	1
1	1	1	1	1	

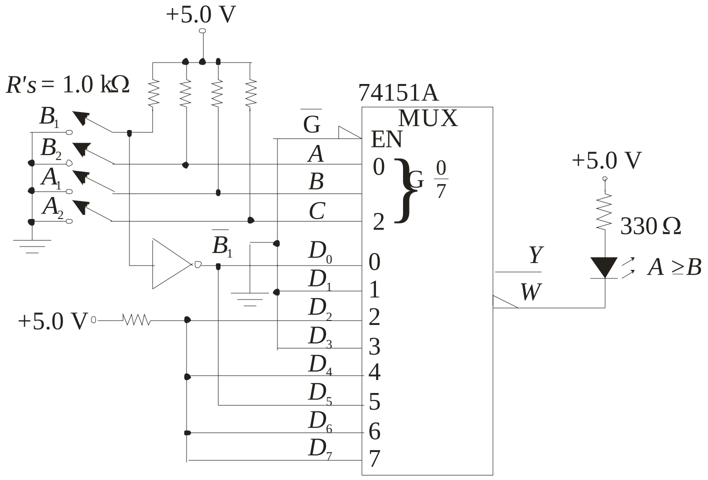


Figure 12-4

Further Investigation Results:

TABLE 12-2

Truth table for even parity generator.

Inputs				Output	Connect Data to:
A_3	A_2	A_1	A_0	X	
0	0	0	0	0	A_0
0	0	0	1	1	
0	0	1	0	1	$\overline{A_0}$
0	0	1	1	0	
0	1	0	0	1	$\overline{A_0}$
0	1	0	1	0	
0	1	1	0	0	A_0
0	1	1	1	1	
1	0	0	0	1	$\overline{A_0}$
1	0	0	1	0	
1	0	1	0	0	A_0
1	0	1	1	1	
1	1	0	0	0	A_0
1	1	0	1	1	
1	1	1	0	1	$\overline{A_0}$
1	1	1	1	0	

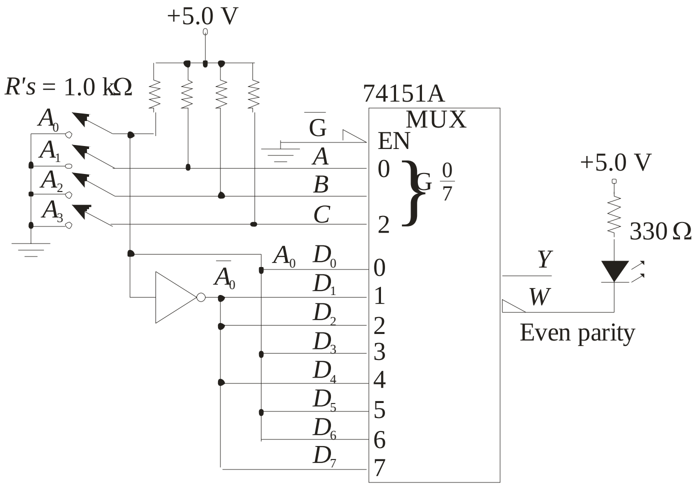
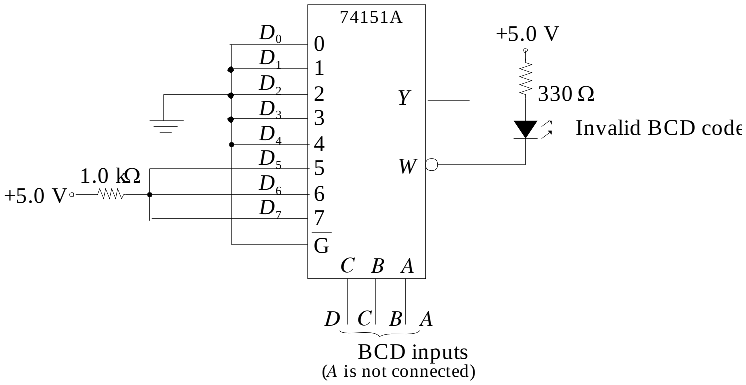


Figure 12-5

Evaluation and Review Questions:

1. The BCD invalid code detector can be designed as shown:



2. $\overline{A_0}\,\overline{A_1}A_2 + A_0\overline{A_1}\,\overline{A_2} + A_0\overline{A_1}A_2 + \overline{A_0}A_1A_2$

3. The input which affects the second half of the truth table is the A_2 input. Place the circuit in a fault condition and check the inputs to the multiplexer beginning with the A_2 input.

4. The inverter output is used with minterms 0, 1, 10, and 11. However it will be incorrect only when B_1 should be HIGH - which is a fault condition for minterm 0 and 10. Place the circuit in one of these fault conditions and check the appropriate inputs to the multiplexer. Since the multiplexer input is incorrect, the troubleshooter is led to test the inverter and discover the fault.

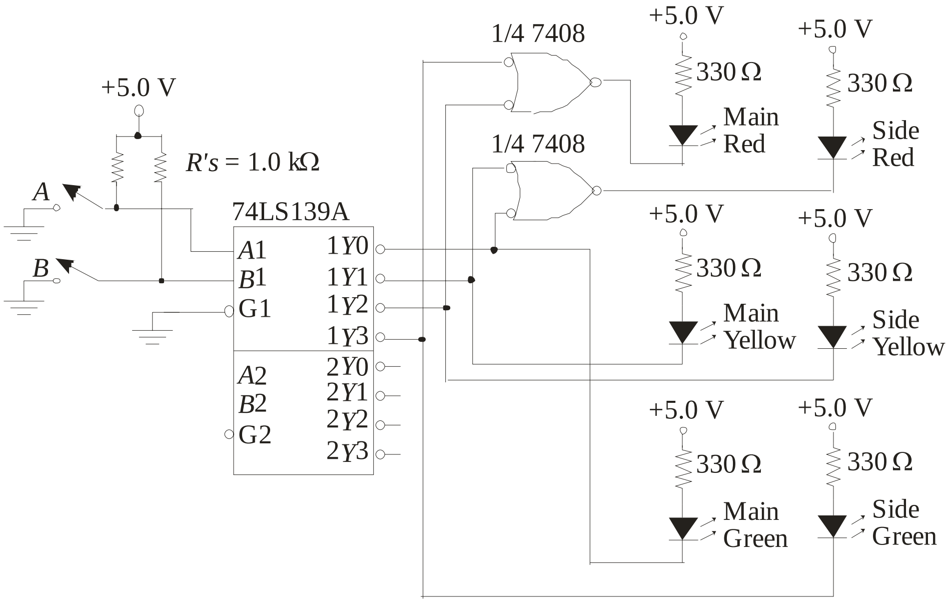
5. The lines on the truth table that would be incorrect are $A_2A_1B_2B_1$ = 0 0 0 1 and $A_2A_1B_2B_1$ = 1 0 1 1. All other lines will read correctly.

6. Odd parity can be obtained from the *Y* output of the 74151A.

Experiment 13: Combinational Logic Using Demultiplexers

Traffic Light Output logic (Figure 13-6):

Connect $1Y_1$ to <u>Main Yellow</u> and to <u>Side Red</u> through the 7408 AND gate. Connect $1Y_3$ to <u>Main Red</u> through a 7408 AND gate and to <u>Side Green</u>. Connect $1Y_2$ to <u>Main Red</u> through a 7408 AND gate and to <u>Side Yellow</u>. The enable (G1) is wired LOW with switches to the A_1 and B_1 select inputs. The circuit is shown below:



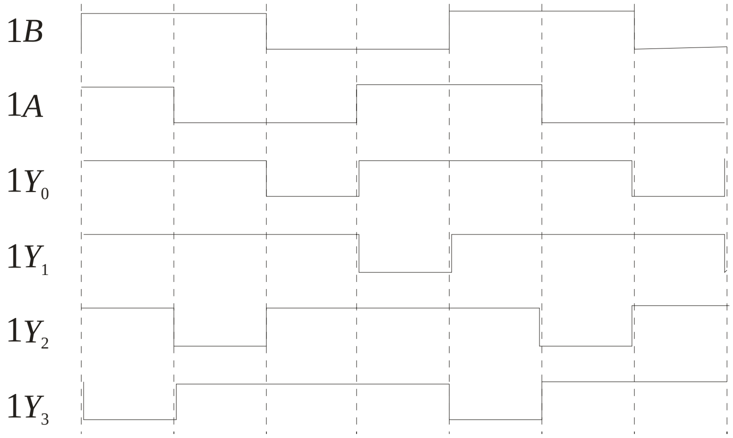


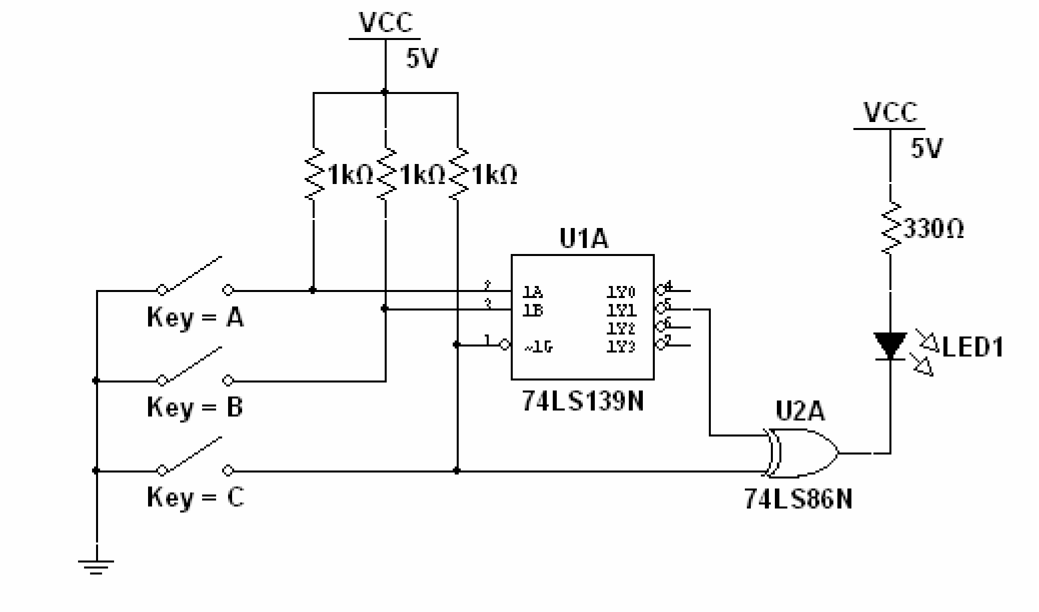
Figure 13-8
Timing diagram for step 5

Further Investigation Results:

The NAND gate serves as a decoder for a particular output state from the counter. The decoded output then enables the decoder and lights a corresponding LED. As long as the CLK input to the counter is above approximately 100 Hz, the LED appears to be on steady. Different LEDs can be selected by changing the decoder's inputs.

Evaluation and Review Questions:

1. Use one-half of the 74LS139A for the least significant four bits and the other half for the most significant four bits. Use the enable input on each decoder for the third select input, inverting it for the most significant four bits.
2. To emphasize the active-LOW inputs and outputs.
3. Gray code is useful to prevent glitches in the decoded outputs.
4. a. The input is an illegal HIGH, preventing the $1Y_0$ and the $1Y_1$ outputs from being selected.
   b. The input is LOW, preventing the $1Y_2$ and the $1Y_3$ outputs from being selected.
   c. The chip is not selected; therefore all outputs will be HIGH.
5. The sequence of the lights would be incorrect. This can be corrected by reversing the output logic for states 2 and 3. The disadvantage is possible glitches in the decoded outputs.
6. The LED will be on whenever *C* is open (HIGH) or if *C* and *B* are closed (LOW) and *A* is open (HIGH). (Multisim file is used for illustration.)



Experiment 14: The D Latch and D Flip-Flop

Step 3 Observations: The output of the latch changes only when the switch makes initial contact with the *A* or *B* terminals. The latch does not change when the switch is "bounced".

Step 5 Observations: As long as the enable is HIGH, the output (*Q*) follows the pulse generator input. When enable is LOW, changes on the D input are not seen on the output, however; the outputs can be latched by grounding either one momentarily, even when enable is not asserted.

Step 6 Observations: The burglar alarm basically illustrates the functions of the D latch. When enable is asserted HIGH, the opening of a door or window switch causes the alarm LED to latch on. Closing the door or window switch has no effect on the output. The alarm LED is turned off by closing all switches (or placing it in standby) and pressing the reset pushbutton.

Step 7 and 8 Observations: The setup time specification for a 7474 is 20 ns. The four inverters provide sufficient setup time to read the pulse as a HIGH, causing the output to remain a constant HIGH. When the delay is removed, the output is a constant LOW.

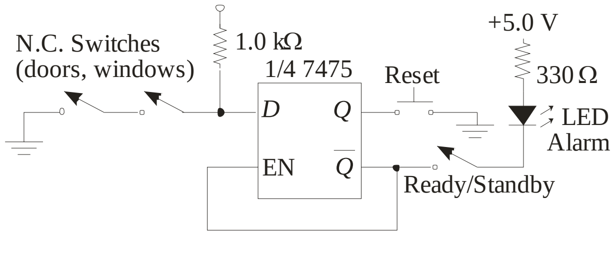
Step 10 Observations: The D flip-flop needs setup time as shown with the clock delay circuit. When sufficient setup time is given, the output will go HIGH; otherwise, it is LOW. The preset and clear inputs are asserted with a LOW input and are observed to be asynchronous inputs. With the clock delay in place and connecting the *Q* to D, the scope signal can be made to appear as if timing is changing when in fact it is not. (Trigger from the clock to show the timing problem).

Further Investigation Results:
The serial parity test circuit changes the parity every time a logic 1 is received but does not change parity when a logic 0 is received. The circuit has an advantage over the parallel parity test circuit in Floyd's text only when the data is already in serial form and needs to be tested. In this case, the serial tester circuit is simpler and just as fast.

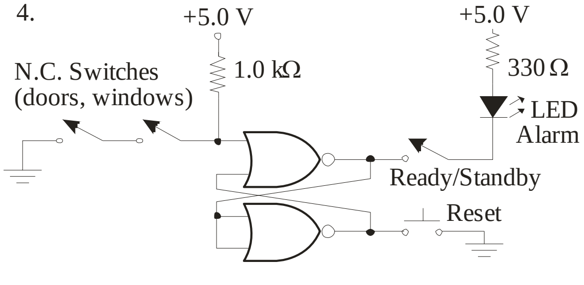
Evaluation and Review Questions:

1. The switch debounce circuit requires two contacts (throws) because the output will not change if a single contact were used.
2. No. With NOR gates, the output would change as the switch pole is first moved from the contact but the other NOR gate output would be unaffected – causing bouncing to be seen on the output. A similar problem would occur on the other contact.

3.



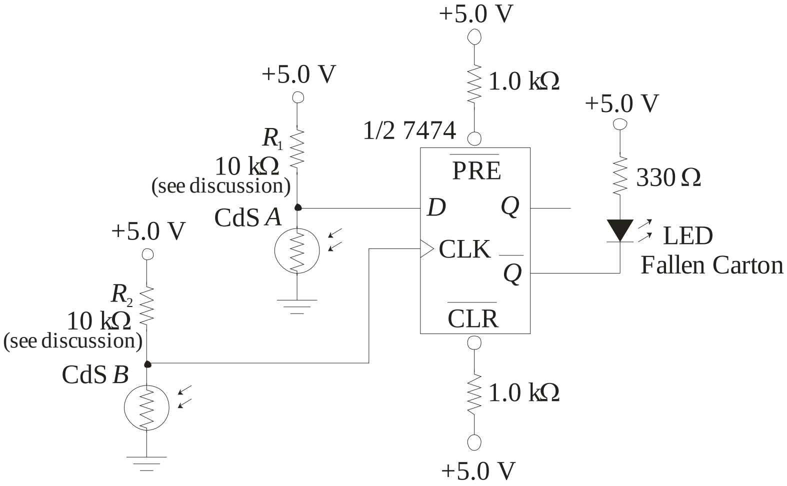
4.



5. There are a several possibilities. Among them are:
   1. Reset or enable switch stuck LOW.
   2. Open path to the alarm LED including open resistor or open diode.
   3. Faulty NAND gate (open output on lower right NAND gate, shorted output on top right gate, etc.) The most likely fault is an open connection rather than a faulty component.
6. A latch would oscillate very rapidly whenever the data was HIGH.

Experiment 15: The Fallen Carton Detector

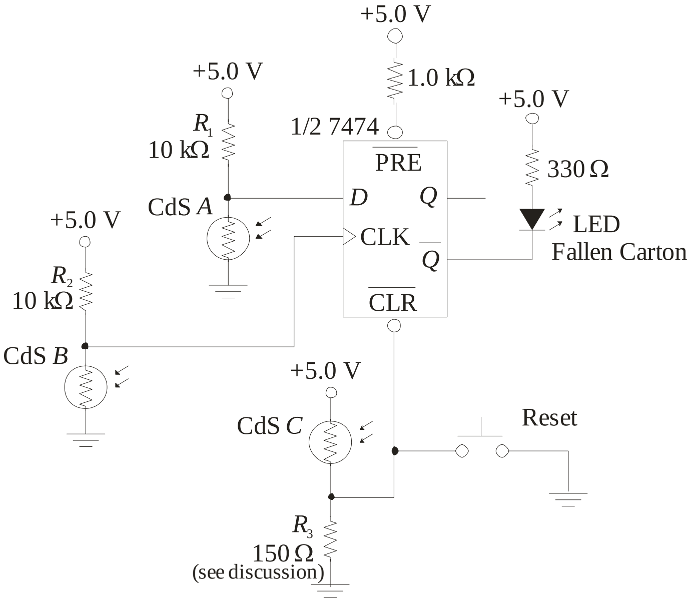
The fallen carton detector is a sequential logic circuit which is designed to detect if a carton, moving on a conveyor belt, is upright or has fallen over. If the carton has fallen, the circuit is to trip a solenoid (indicated by an LED). The circuit can be implemented with a D flip-flop as shown. Student circuits may vary.



Students will need to experiment to determine optimum series resistors (shown as R_1 and R_2) to use with the photocells and with the particular room lighting. With a CdS photocell, a value of about 10 kΩ will work in normal room light when connected to a 7474 D flip-flop. The voltage will be LOW when uncovered (< 0.2 V) and will be HIGH (> 3.0 V) when covered.

Further Investigation Results:

As the carton is sent into the reject hopper, a photocell can sense its presence. The photocell output can be connected to the $\overline{\text{CLR}}$ input of the D flip-flop to produce a LOW input when it is covered by reversing the photocell and the series resistor. Because TTL logic requires more current to pull it down, the value of R_3 must be much smaller than in the first circuit. A value of 150 Ω worked well with the photocell and the particular room light tested. A reset button is also connected to the $\overline{\text{CLR}}$ input of the D flip-flop. The modified circuit is shown below.



Experiment 16: The J-K Flip-Flop

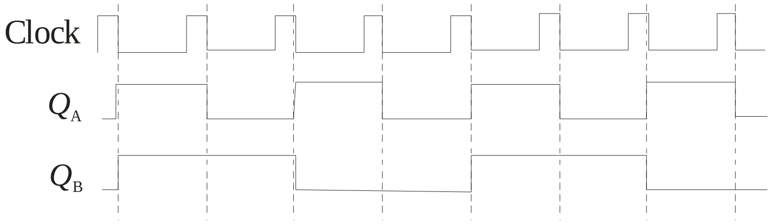
Data and Observations

Step 1 Observations: The preset and clear inputs are asserted with a LOW input and are asynchronous. When both are LOW at the same time, both Q and $\overline{Q}$ are HIGH.

Step 2 Observations: The truth table is verified – Q follows J when the J and K inputs are different. The output does not change when both inputs are LOW and the output toggles when the inputs are both HIGH. The output is observed to change on the trailing edge of the clock. The output duty cycle is seen to be 50%.

Step 3 Observations: The circuit toggles for each clock pulse.

Step 4 Observations: The ripple counter divides the input frequency in two for each stage as shown in the timing diagram. Q_A is ½ the Clock frequency; Q_B is ¼ the clock frequency.



Plot 1

Further Investigation Results:

The 7476 measured and specified values are:

Measured t_{PLH} = 20 ns. Manufacturer's specified t_{PLH} = 16 ns (typ), 25 ns (max).
Measured t_{PHL} = 25 ns. Manufacturer's specified t_{PLH} = 25 ns (typ), 40 ns (max).

The 74LS76A is faster with specified values as follows:

Manufacturer's specified t_{PLH} = 15 ns (typ), 20 ns (max). (Data is from ON Semiconductor.)
Manufacturer's specified t_{PLH} = 15 ns (typ), 20 ns (max). (Data is from ON Semiconductor.)

Evaluation and Review Questions:

1. An asynchronous input is independent of the clock and can affect the output without the presence of a clock signal. A synchronous input can only affect the output when a clock signal is present.

2. a. Assert the preset input ($\overline{\text{PRE}}$) which is asynchronous.
   b. Clock the flip-flop when J is HIGH and K is LOW.

3. The output is latched.

4. If Q is connected to J and $\overline{Q}$ is connected to K, the clock will not change the output logic.

5. The CLK input is open; the Q output is shorted LOW; the $\overline{\text{CLR}}$ input is shorted LOW.

6. Green LED could be open, 390 Ω resistor open, +5.0 V supply not connected to 390 Ω resistor, bad 74LS76A.

Experiment 17: One-Shots and Astable Multivibrators

TABLE 17-1
Data for 74121 monostable multivibrator

Quantity	Computed Value	Measured Value
Timing Resistor, R_T	***7.14 kΩ***	***6.89 kΩ***
External Capacitor, C_{EXT}	0.01 μF	***0.01 μF***
Pulse width, t_W	***48.2 μs***	***46.0 μs***

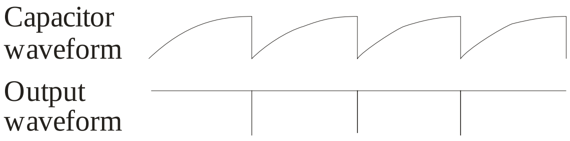
TABLE 17-2
Data for 555 timer as an astable multivibrator

Quantity	Computed Value	Measured Value
Resistor, R_1	7.5 kΩ	***7.46 kΩ***
Resistor, R_2	10 kΩ	***10.1 kΩ***
Capacitor, C_1	0.01 μF	***0.01 μF***
Frequency	***5.24 kHz***	***5.50 kHz***
Duty Cycle	***0.64***	***0.66***

Step 3. Input logic levels and generator connection: To trigger the 74121 with a leading edge clock, either A_1 or A_2 must be held LOW and the signal generator is connected to the B input.

Step 5. Observations as the frequency is raised to 50 kHz: The pulse width remains the same; it does not stay HIGH when the frequency is increased.

Step 8.



Plot 1

Step 9. Observations with a short across R_2: The output stays HIGH except for a short "spike" that appears each time a new cycle is initiated.

Step 10. Various solutions are possible. One solution is to modify the circuit shown in Figure 17-2 by changing R_2 to 3.3 kΩ. (The calculated value is 3.45 kΩ.)

Further Investigation Results:
Both timers need trailing edge triggers that can be obtained on the 74121 by connecting A_1 and A_2 together and connecting B HIGH (see Figure 17-1, 3 lines from the end of the table). The manufacturer specifies that the external resistor cannot be larger than 40 kΩ. An approximate 4 s pulse can be obtained by selecting a 150 μF capacitor and a 39 kΩ resistor as shown in Figure 17-1 (computed = 4.1 s). A 25 s pulse can be obtained by selecting a 1000 μF capacitor and a 36 kΩ resistor (computed = 25.2 s).

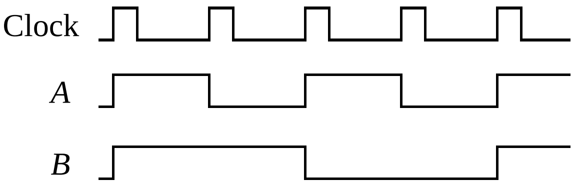
Evaluation and Review Questions:

1. A non-retriggerable monostable multivibrator ignores any triggers that occur during the timing cycle.
2. a. 0.036 μF.
   b. Select an external variable resistor. For the computed capacitance, the external resistance needs to vary from 2 kΩ to 10 kΩ.
3. Largest recommended resistor is 40 kΩ. The largest capacitor is 1000 μF. With this combination the pulse width is 28 seconds.
4. Duty cycle = 50.1%; frequency = 400 Hz.
5. The required frequency is 0.083 Hz. The sum of $R_1 + 2R_2 = 1.72$ MΩ. From the duty cycle equation, the resistors are found to be: $R_1 = 0.36$ MΩ and $R_2 = 0.68$ MΩ.
6. The voltage ranges from +5.0 V to +10.0 V.

Experiment 18: Asynchronous Counters

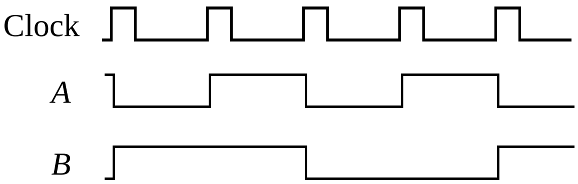
Data and Observations

Waveforms from step 1:



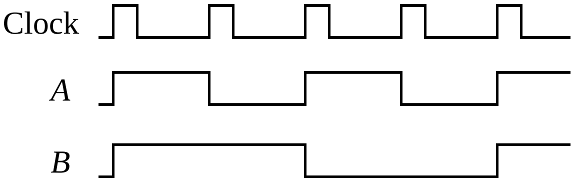
Counter is a down counter.

Waveforms from step 2:



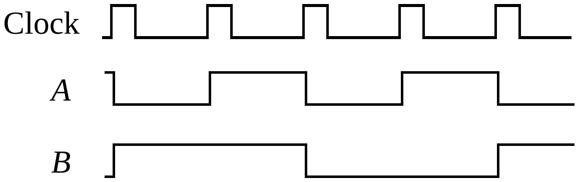
Counter is an up counter.

Waveforms from step 3: (same as step 1)



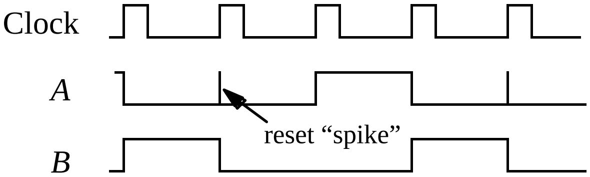
Counter is a down counter.

Waveforms from step 4: (same as step 2)



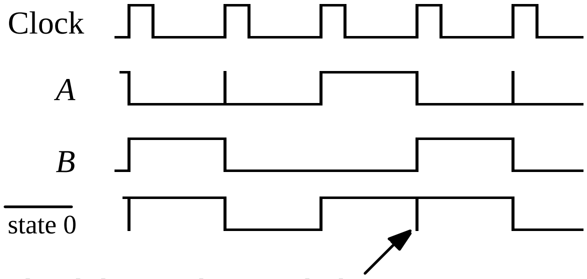
Counter is an up counter.

Waveforms from step 5:



Counter sequence is 0 - 1 - 2 - reset. The short "spike" is caused by the reset in state 3.

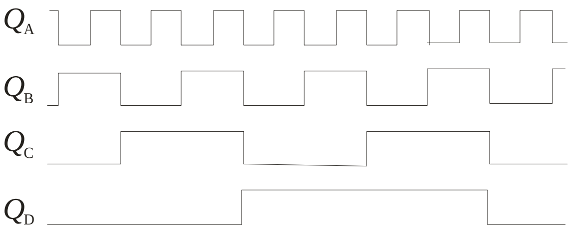
Waveforms from step 6:



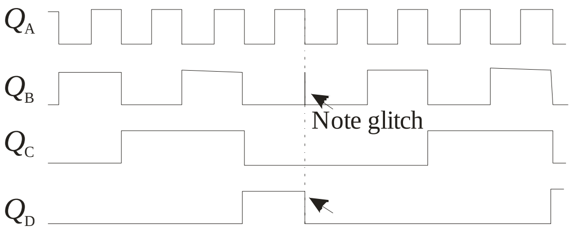
The decoded output shows a "glitch" when both A and B change together.

Notice that the "glitch" in step 5 is a consequence of detecting state 3 and using it to reset the counter. The "glitch" in the state zero decoded output is different. It is caused by the fact that the counter is momentarily in state zero due to propagation delay between the two flip-flops.

Waveforms from step 7:

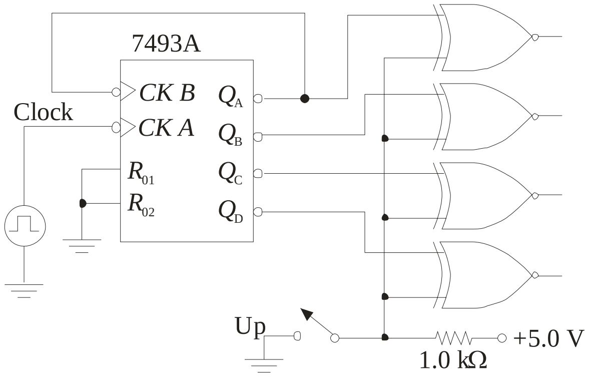


Waveforms from step 8:



The sequence is 0-1-2-3-4-5-6-7-8-9 - reset (momentarily in state 10)

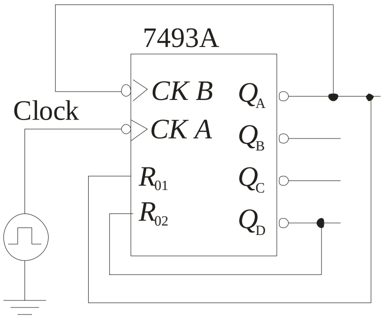
Further Investigation Results:

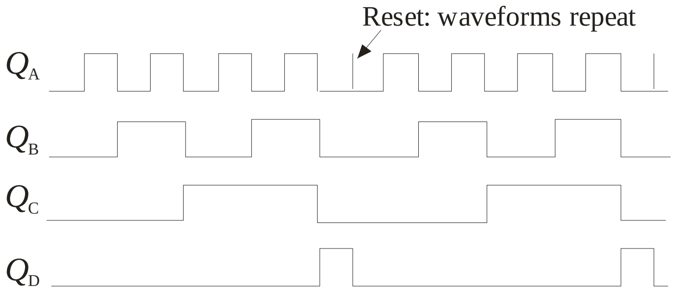


The XOR gates act as inverters when the Up switch is closed and act as buffers when it is open. By inverting or not inverting the waveforms, the counter can appear as an up or a down counter. The disadvantage of the method is that the reversal of the bits when the switch is open or closed will cause the count to change at the switching time.

Evaluation and Review Questions:

1. a. The A signal is 4X faster than the clock. Since the measurement shows that it is 4.00 kHz, the clock frequency is 1.00 kHz.
   b. If you want to check the time relationship of any other signal with respect to the displayed signals, it is easier to make the comparison with the slower waveform. Since it is a digital scope, the two channels are digitized together and so this reason is for convenience only. If this were an analog scope, a timing error can be made by triggering on a faster signal to observe (and compare) a slower signal.
2. a. The count sequence is truncated by decoding one number more than the ending number and using the decoded output to reset the counter.
   b. The procedure produces a glitch because the counter is in the decoded state for a very short interval.
3. The clear input may be shorted LOW, producing a HIGH output on both flip-flops; the line from $\overline{Q}$ to *D* on the *A* flip flop may be open causing an (invalid) HIGH to be clocked into the *A* flip-flop at each clock pulse.
4. The time relationship determined from a two channel oscilloscope could be interpreted incorrectly because the scope can be triggered by *any* arbitrary clock pulse. Relative timing is unambiguous when the various waveforms are compared to the slowest waveform.
5. a. Circuit: b. Waveforms:





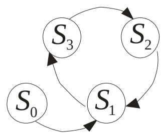
6. The sequence is 0-1-2-3-4-5-8-9-reset (momentarily in state 10)

Experiment 19: Analysis of Synchronous Counters with Decoding

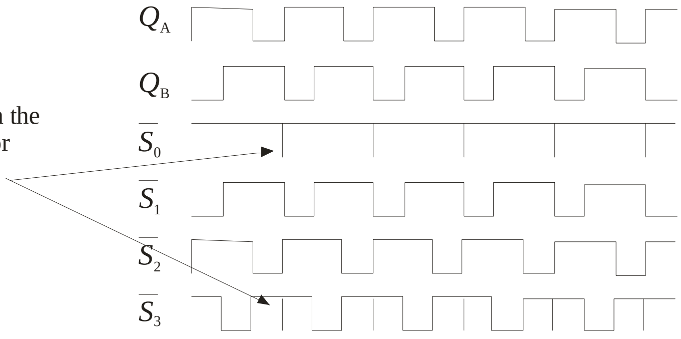
TABLE 19-2
Analysis of synchronous counter
shown in Figure 19-3

Outputs	Inputs			
Q_B Q_A	$J_B = \mathbf{Q_A}$	$K_B = \overline{\mathbf{Q_A}}$	$J_A = 1$	$K_A = \mathbf{Q_B}$
0 0	0	1	1	0
0 1	1	0	1	0
1 1	1	0	1	1
1 0	0	1	1	1
0 1	(Repeats...)			

State diagram:



Note the glitches in the decoded outputs for state 0 and state 3.



Plot 1

Step 4: State diagram:

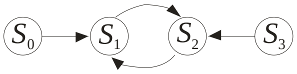
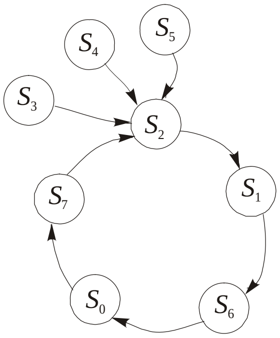


TABLE 19-3
Analysis of counter shown in Figure 19-4

Outputs	Inputs					
Q_C Q_B Q_A	$J_C = \overline{\mathbf{Q_A}}$	$K_C = 1$	$J_B = 1$	$K_B = \overline{\mathbf{Q_A}}$	$J_A = \overline{\mathbf{Q_C}}$	$K_A = 1$
0 0 0	1	1	1	1	1	1
1 1 1	0	1	1	0	0	1
0 1 0	0	1	1	1	1	1
0 0 1	1	1	1	0	1	1
1 1 0	0	1	1	1	0	1
0 0 0	Repeats..test unused states:					
0 1 1	0	1	1	0	1	1
0 1 0	Returns to a tested state (main seq)					
1 0 0	1	1	1	1	0	1
0 1 0	Returns to a tested state (main seq)					
1 0 1	1	1	1	0	0	1
0 1 0	Returns to a tested state (main seq)					

Step 6: State diagram:

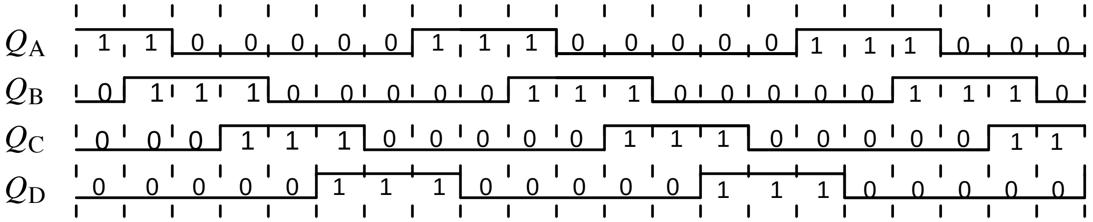


Further Investigation Results:

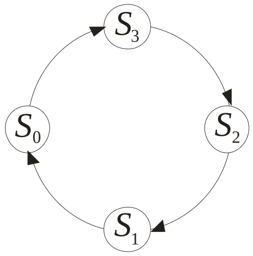
The circuit shows the letters for the word C-L-U-E in the seven segment display.

Evaluation and Review Questions:

1.


Q_A 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0
Q_B 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0
Q_C 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1
Q_D 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0

2. The sequence is that of a down counter:



3. Full decoding means all possible states must be decoded. Because there are three outputs, full decoding can be accomplished with a 3-to-8 decoder such as the 74LS138.

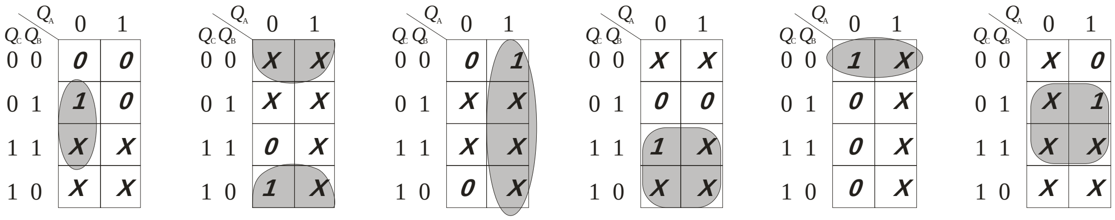
4. Connect the pushbutton so that it clears the *A* and *C* counters and presets the *B* counter using the asynchronous clear and preset inputs.

5. The counter can be locked in state 3 if there are no clock pulses or if the common preset line is shorted LOW.

6. State 9 (*DCBA* = 1001) should go to state 1 (*DCBA* = 0001) but isn't changing. By observation, three of the F/F's do not change states for this transition (FF*A*, FF*B*, and FF*C*). FF*D* should change and isn't changing, so it is the likely culprit.

Experiment 20: Design of Synchronous Counters

Present State			Next State		
Q_C	Q_B	Q_A	Q_C	Q_B	Q_A
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	0	0
1	0	0	0	0	0



Q_CQ_B \ Q_A	0	1
0 0	0	0
0 1	1	0
1 1	X	X
1 0	X	X

$J_C = \overline{Q}_A Q_B$

Q_CQ_B \ Q_A	0	1
0 0	X	X
0 1	X	X
1 1	0	X
1 0	1	X

$K_C = \overline{Q}_B$

Q_CQ_B \ Q_A	0	1
0 0	0	1
0 1	X	X
1 1	X	X
1 0	0	X

$J_B = Q_A$

Q_CQ_B \ Q_A	0	1
0 0	X	X
0 1	0	0
1 1	1	X
1 0	X	X

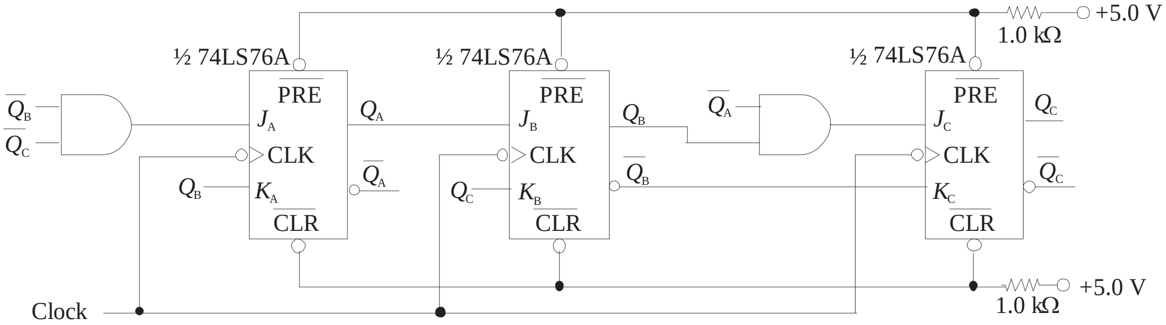
$K_B = Q_C$

Q_CQ_B \ Q_A	0	1
0 0	1	X
0 1	0	X
1 1	0	X
1 0	0	X

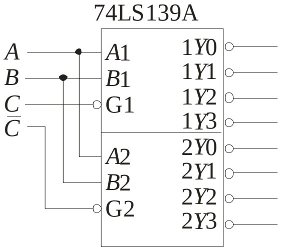
$J_A = \overline{Q}_B\overline{Q}_C$

Q_CQ_B \ Q_A	0	1
0 0	X	0
0 1	X	1
1 1	X	X
1 0	X	X

$K_A = Q_B$



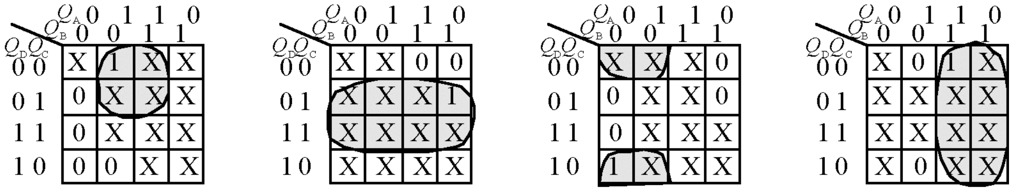
Further Investigation Results:
The 74LS139A can be connected to form a 3-to-8 decoder as shown below:



Evaluation and Review Questions:

Note: A variation for drawing Karnaugh maps was presented in the lab book in Figure 20-3 and repeated here for reference. The Q_B and Q_A variables are not written side-by-side; rather they are written nearly vertically. Some students find it easier to read the map to relate the number to the variable when they are drawn like this.

1.



Q_DQ_C \ Q_AQ_B	00	10	11	01
00	X	1	X	X
01	0	X	X	X
11	0	X	X	X
10	0	0	X	X

$J_B = Q_A\overline{Q_D}$

Q_DQ_C \ Q_AQ_B	00	10	11	01
00	X	X	0	0
01	X	X	X	1
11	X	X	X	X
10	X	X	X	X

$K_B = Q_C$

Q_DQ_C \ Q_AQ_B	00	10	11	01
00	X	X	X	0
01	0	X	X	0
11	0	X	X	X
10	1	X	X	X

$J_A = \overline{Q_B}\,\overline{Q_C}$

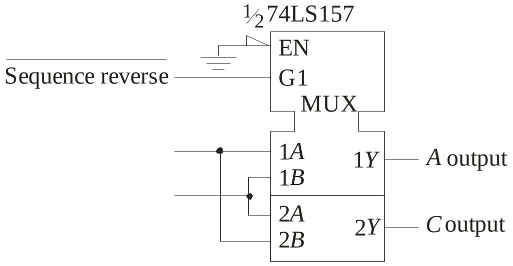
Q_DQ_C \ Q_AQ_B	00	10	11	01
00	X	0	1	X
01	X	X	X	X
11	X	X	X	X
10	X	0	X	X

$K_A = Q_B$

2. Since the largest binary number in the circuit for the experiment is less than ten, a BCD to seven-segment decoder (such as the 74LS47A) can be used to convert the binary number into a value that can be shown in a seven-segment display.

3. The reset button is connected between ground and the preset input of FF*C* and FF*B*; it is also connected between ground and the clear input of FF*A*.

4. The count sequence can be reversed by reversing the *A* and *C* bits from the counter. This can be accomplished by using a 2-input MUX, such as the 74LS157. (Note that only a part of the 74LS157 is shown.)



5. When the counter goes from state 3 to state 2, the *A* output is seen to do a 1 to 0 transition. Likewise, when the counter goes from state 6 to state 4, the *B* output does a 1 to 0 transition. These transitions are unique, therefore trigger by connecting the *A* and *B* outputs to the *A*1 and *A*2 trigger inputs of the 74121 (*B* trigger is HIGH). (See the function table for the 74121, given in Figure 17-1).

6. a. The *D* transition table is shown. The Q_{N+1} output follows *D* at the clock edge.

Q_N	Q_{N+1}	D
0 →	0	0
0 →	1	1
1 →	0	0
1 →	1	1

b. The *J-K* flip-flop is more versatile because it has the latch and toggle states. For this reason, there are *X*'s on the transition table which aid in designing a simpler circuit for irregular count sequences.

Experiment 21: The Traffic Signal Controller

Note - Table 21-1 lists present states in the gray-code sequence selected for the controller (0-1-3-2) rather than binary sequence.

Table 21-1

Present State		Next State		Input Conditions	Input Product Term for Data Selector-1	Input Product Term for Data Selector-0
Q_1	Q_0	Q_1	Q_0			
0	0	0	0	$T_L + \overline{V_S}$	0	$\overline{T_L} V_S$
0	0	0	1	$\overline{T_L} V_S$		
0	1	0	1	T_S	$\overline{T_S}$	1
0	1	1	1	$\overline{T_S}$		
1	1	1	1	$T_L V_S$	1	$T_L V_S$
1	1	1	0	$\overline{T_L} + \overline{V_S}$		
1	0	1	0	T_S	T_S	0
1	0	0	0	$\overline{T_S}$		

Schematic:

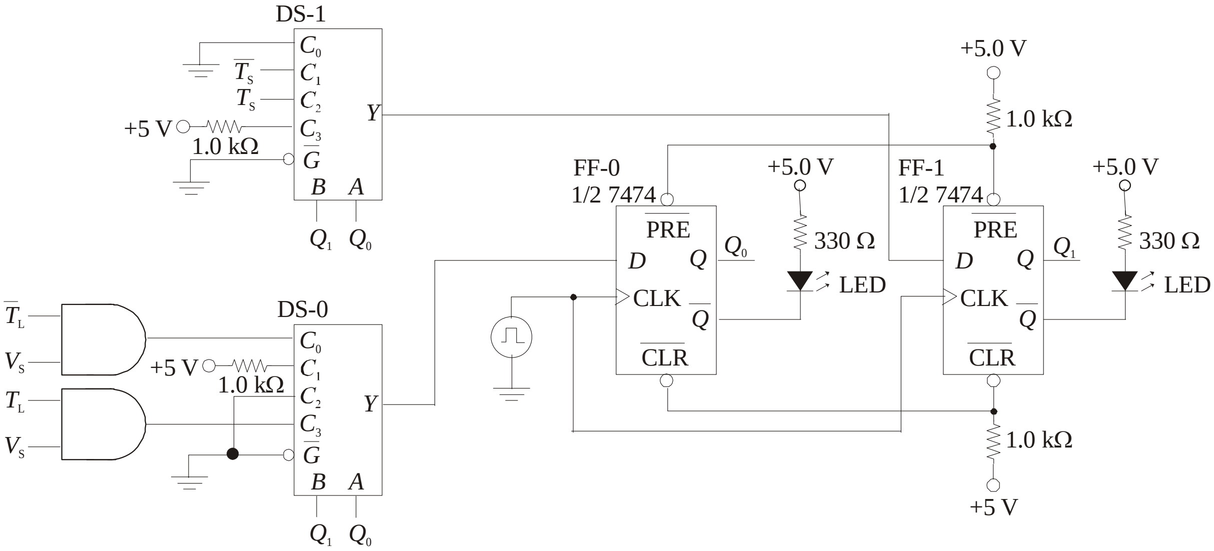


Figure 21-4

Step 6: To return to the first state, the vehicle sensor must be LOW and remain LOW (no vehicle on side street).

Further Investigation Results (TTL)

The first idea is sound; the short timer can be eliminated if states change in no less than 4 s. The equations would need to be changed to implement this. The second idea will not allow the state machine to function as designed.

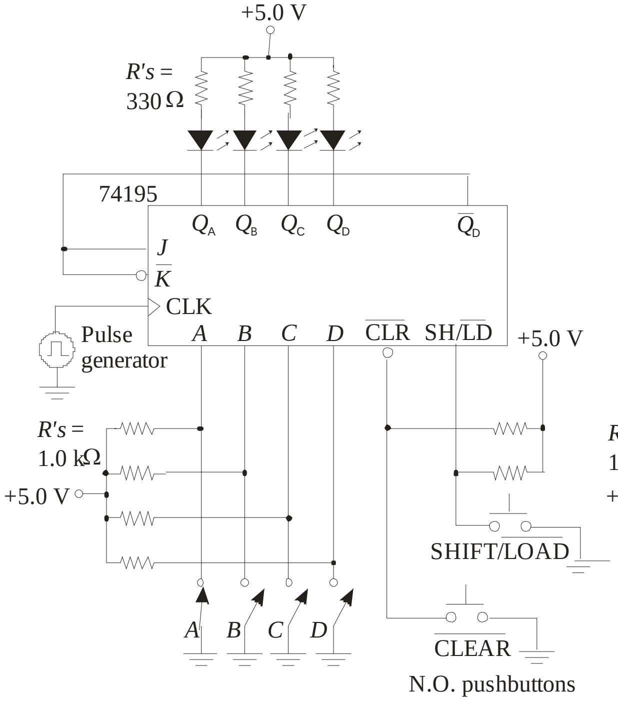
Evaluation and Review Questions:

1. Gray code is selected to avoid false states due to decoding glitches.
2. The long timer (T_L) and the vehicle sensor (V_S) must both be HIGH.
3. A third flip-flop is needed for the counter and the MUXs would be changed to 3-to-8 lines.

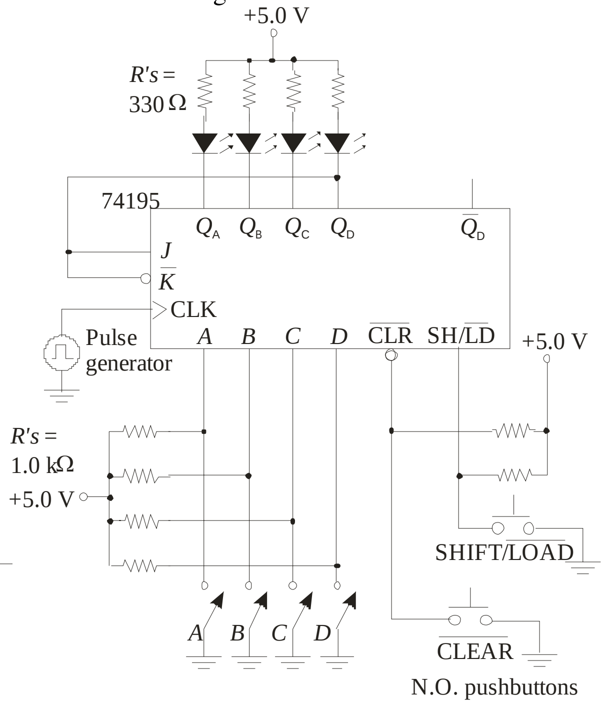
4. Connect the *Y* output from the MUXs to *J*; this line is inverted and connected to *K*.
5. Refer to the 7th line down on the table in Figure 17-1. In order to have both *A* inputs trigger on a trailing edge, *B* must be HIGH.
6. The next state will sequence only if the long timer (T_L) is LOW and the vehicle sensor (V_S) is HIGH. Because the long timer is okay, test V_S. If it is okay, check the counter for clock pulses and correct inputs on *D*. If okay, check asynchronous inputs, power, and grounds.

Experiment 22: Shift Register Counters

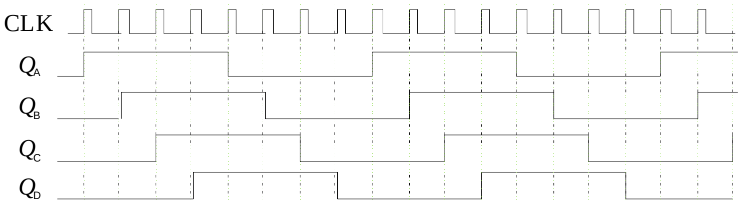
Schematic for Johnson Counter:



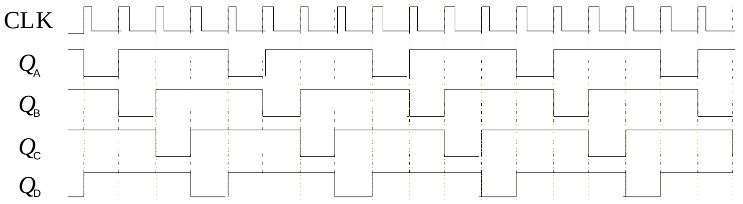
Schematic for Ring Counter:



Timing diagram for Johnson counter:

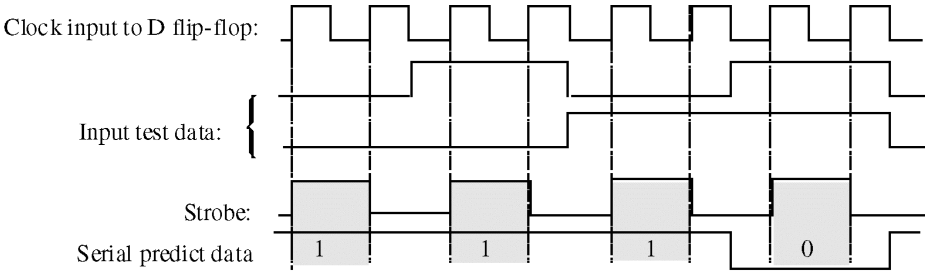


Timing diagram for ring counter loaded with 1110:



Further Investigation Results

The **Serial predict data** is compared with the data from the device when the **Strobe** line goes HIGH as shown in the following timing diagram:



Data shown is for a 2-input NAND gate (1-1-1-0)

Evaluation and Review Questions:

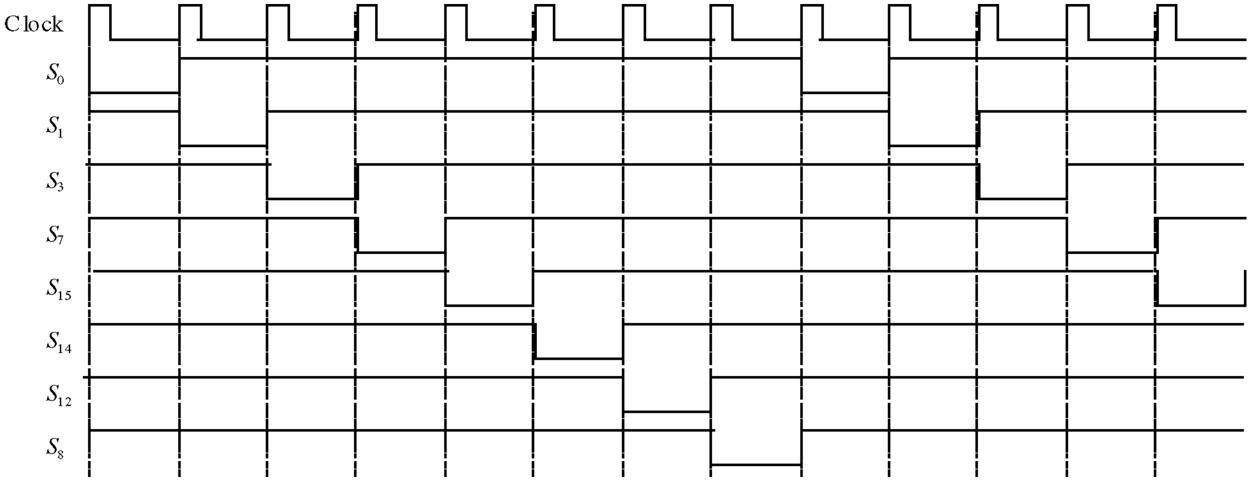
1. The Q_D output of the first 74195 is connected to the J-$\overline{K}$ inputs of the second. The $\overline{Q}_D$ output from the second 74195 is connected to the J-$\overline{K}$ inputs of the first.

2. To prevent the data from shifting more than one position for each clock pulse.

3. a. 101-110-011. b. 101-010-101. (An interesting but generally not useful result!).

4. After the stored data has been clocked out, the same input data will appear at all outputs.

5. The 8 active states are shown in the diagram. (Repeats after 8 states).



6. Politely tell the boss that a normal ring counter is self-decoding and that a decoder is not necessary.

Experiment 23: Application of Shift Register Circuits

Data and Observations:

Step 2: After loading the data, the start bit flip-flop is observed to have a LOW (LED on). Data from the shift register can be seen to move to the right at each clock pulse. The data at the Q_A output moves to the *Serial data out* position after 5 clock pulses. The register is then filled with HIGHs (LEDs all off) until reloaded.

Step 3: Sample calculations of the resistors for the 555 configured as an astable multivibrator are given. (See Figure 17-2 for the circuit for the astable configuration.)

$$f = \frac{1}{0.7\left(R_1 + 2R_2\right)C} = \frac{1}{1\text{ ms}} \qquad t_H = 0.8\text{ ms} \qquad t_L = 0.2\text{ ms}$$

Let $C = 0.1\ \mu\text{F}$.

Then:

$$R_1 + R_2 = \frac{t_H}{0.7\ C} = \frac{0.8\text{ ms}}{0.7\ (0.1\ \mu\text{F})} = 11.4\text{ k}\Omega$$

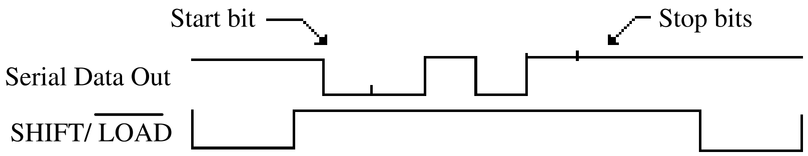
and

$$R_2 = \frac{t_L}{0.7\ C} = \frac{0.2\text{ ms}}{0.7\ (0.1\ \mu\text{F})} = 2.86\text{ k}\Omega\ \ (\text{choose } 2.7\text{ k}\Omega)$$

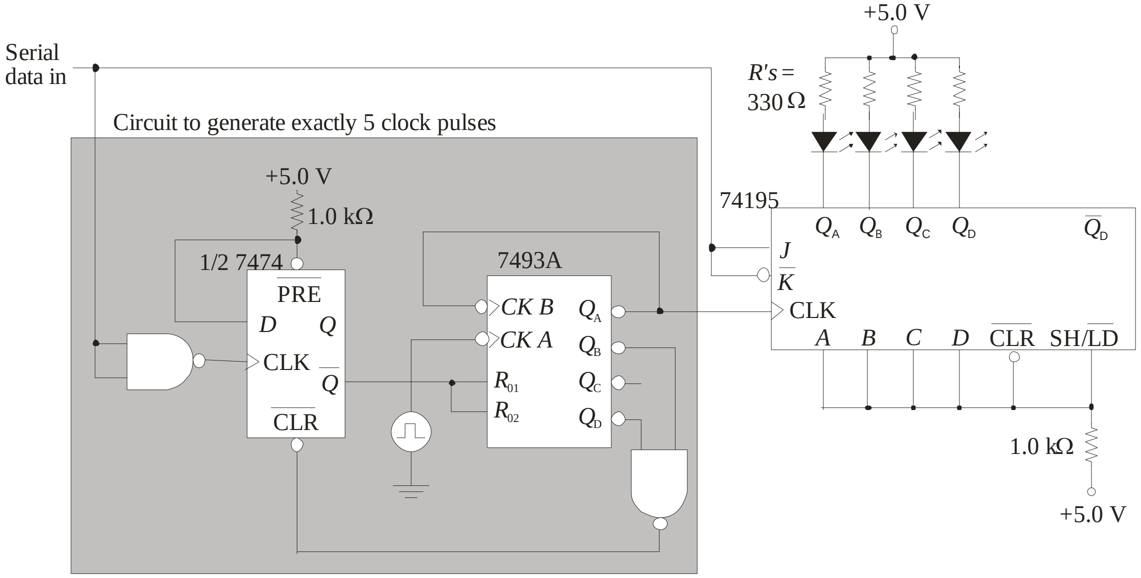
$$R_1 = 11.4\text{ k}\Omega - 2.7\text{ k}\Omega = 8.7\text{ k}\Omega\ \ (\text{choose } 9.1\text{ k}\Omega)$$

With the values selected above, the duty cycle is 81% and the frequency is 985 Hz.

Step 4: Because of the asynchronous relation between the data clock and the $\text{SHIFT}/\overline{\text{LOAD}}$ pulse, the time between the serial data out and the $\text{SHIFT}/\overline{\text{LOAD}}$ and the start bit is not fixed.



Step 5: The circuit to send exactly five clock pulses is shown in the shaded box. When the start bit is detected, the data is shifted in the 74195. The frequency of the pulse generator is set to twice that of the transmitter in order to generate one complete pulse with every two clocks. (The 7493A counts to ten during this process.)



Further Investigation Results:

The receiver pulse generator is set twice as fast as the transmiitter's pulse generator because the circuit that generates 5 clock pulses divides the receiver frequency by two. The generator frequencies should be close to the 2-1 ratio, but it is not necessary to be exact. Five data pulses are used to send the start bit right through the receiver (into the "bit bucket"), moving only the data that was set at the transmitter into the receiver. The stop bits are represented by HIGHs, and since the line is held HIGH (no transition), the receiver will not be clocked until a new start bit is received.

Evaluation and Review Questions:

1. The *J-K* inputs are connected HIGH. This HIGH is shifted into the 74195A at each clock pulse.

2. The first bit is a start bit which is not part of the data. The extra clock pulse causes it to be clocked through the receiver's shift register.

3. The start bit transition is a falling edge (HIGH to LOW). The NAND gate, which acts as an inverter, changes the active edge so that it can trigger the 7474 with a leading edge.

4. a. The numbers to be added are entered into the input shift registers. The full adder produces the sum and the carry bit sequentially as each bit is shifted through the input registers. As the sum is produced, it is moved sequentially through the output shift register. The carry-out flip-flop stores the carry bit to be added to the next most significant bit.

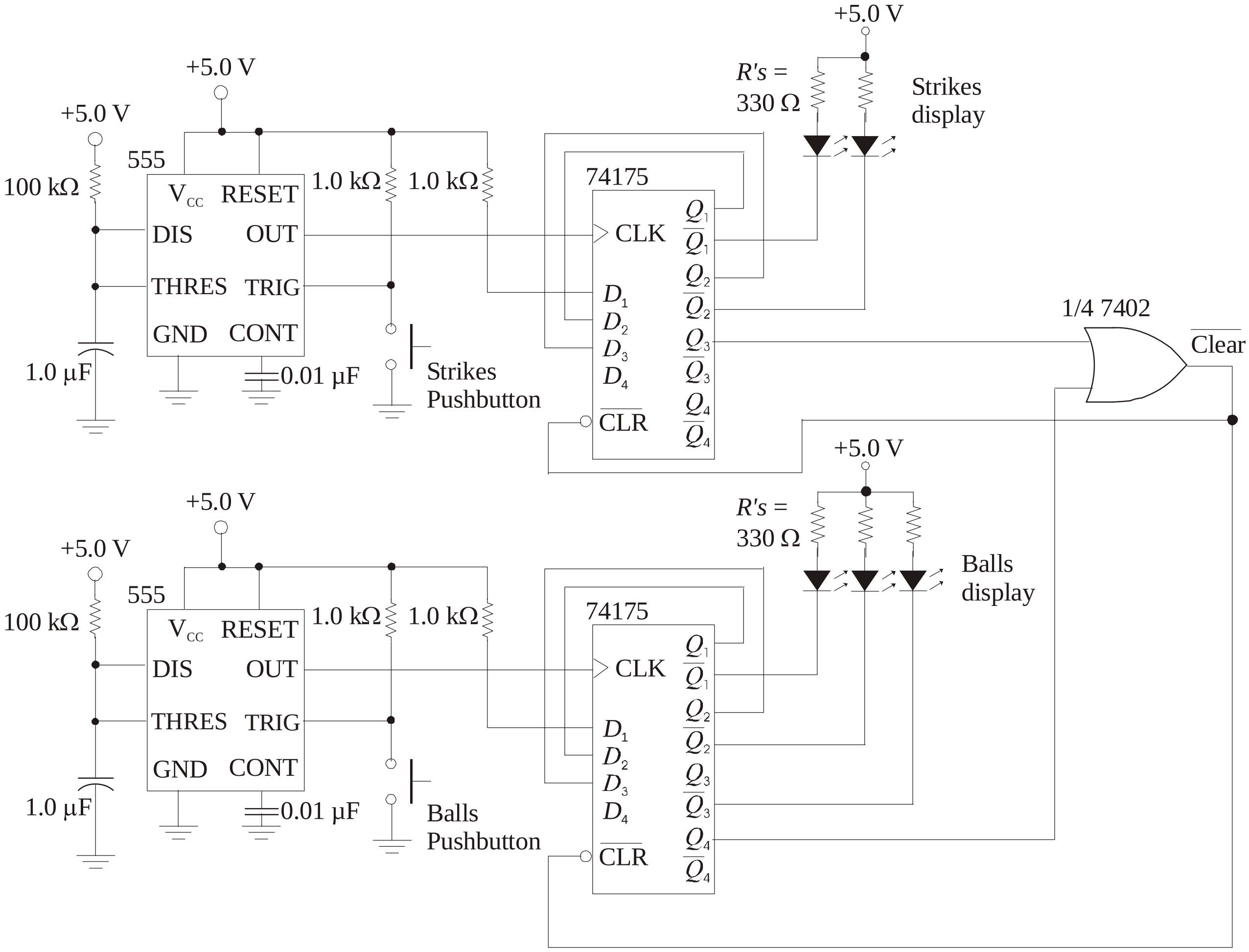
   b. The output shift register will contain 0011; the carry-out flip-flop will contain 1.

5. A 74195A shift register can be configured as a ring counter. The register is loaded with the pattern 1000 which is recirculated. The outputs are taken from either Q_A and Q_C or from Q_B and Q_D.

6. If the start bit flip-flop is loaded with a HIGH instead of a LOW, the start bit will not occur. This could occur if the clear input to the start bit flip-flop were open.

Experiment 24: The Baseball Scoreboard

The baseball scoreboard is a sequential logic circuit that can be implemented with shift registers or with a counter. One possible solution, using 74175 quad D flip-flops connected as shift registers is shown. Switch debounce is provided by the 555 timers.

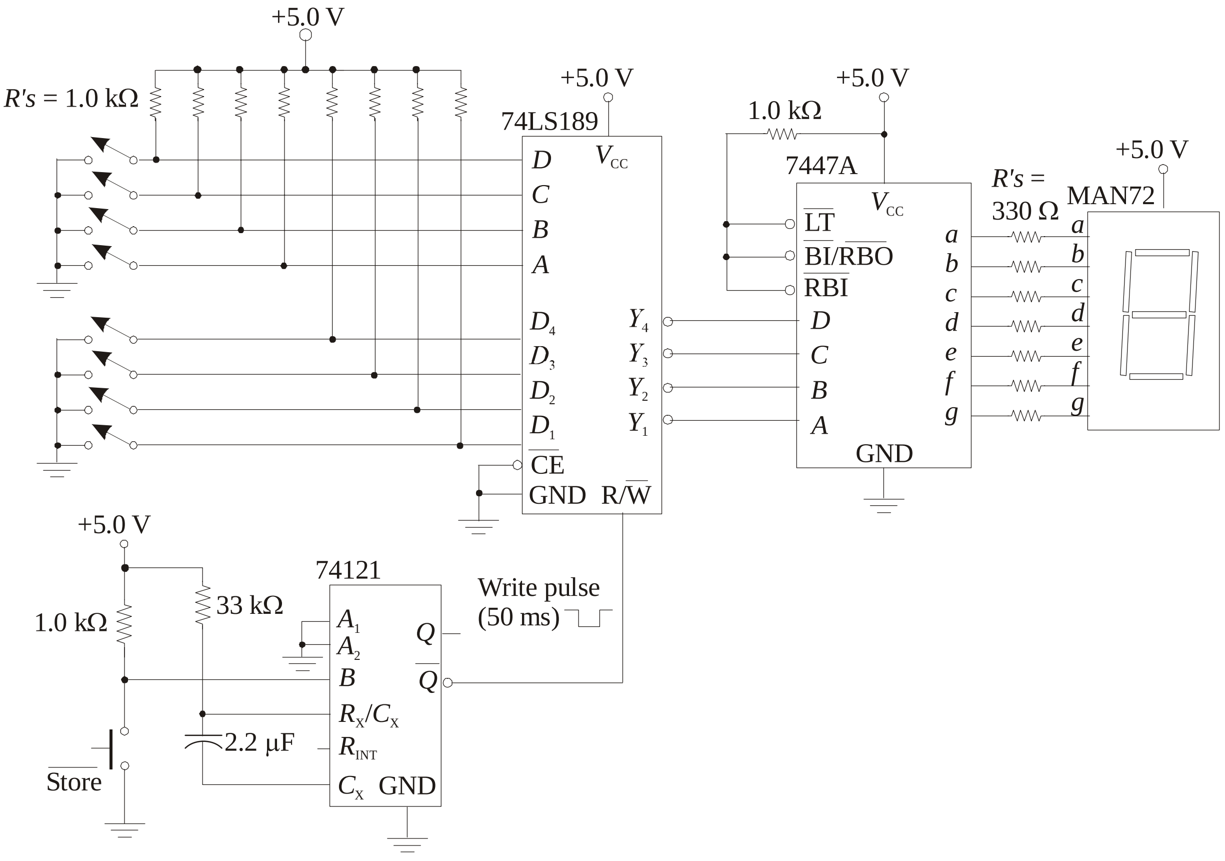


Further Investigation Results (TTL)

Student answers will vary. The inning display can be set up as a 16-position shift register with a single LED on at any given time. Another approach is to use a 16-bit DMUX driven with a manually clocked 4-bit counter controlling the select inputs of the DMUX. Outs can be connected as a 2-bit shift register made from D flip-flops controlled by a debounced pushbutton, similar to the circuits shown above.

Experiment 25: Semiconductor Memories

The schematic for the addition to the security entry system is shown. The combination is entered as the *complements* of the BCD numbers (95614) stored in locations 0000 through 0100. Location 0101 should have 0000 stored to blank the 7-segment display after the last number.



Further Investigation Results:

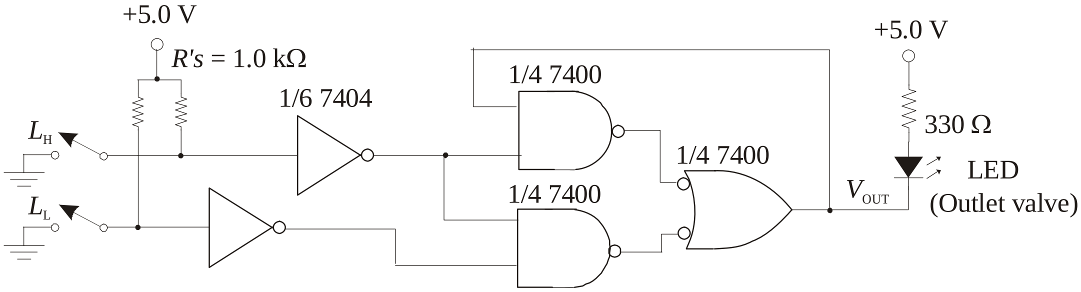
The 7493A is set up to count from 0 through 5 by connecting the Q_B and Q_C outputs to the reset inputs and Q_A output to the CLK *B* input.

Evaluation and Review Questions:

1. For this experiment, address and data switches are set manually, and do not change. Therefore, it is not necessary to debounce the start push-button. It is included in the system because this is a simulation of a portion of a larger system.
2. The number to be stored is 0000. The output is the complement of the stored number.
3. a. Locations are addressed by binary numbers. On chip decoding means that the binary addresses are converted internally to the specific row information needed.
   b. Outputs are in a high-Z state.
4. 256 locations
5. a. The word length needs to be expanded to 7 bits by expanding memory.
   b. Locations would store the encoded combination for the seven segment display.
6. When R/$\overline{W}$ is LOW, the outputs are the complements of the data inputs.

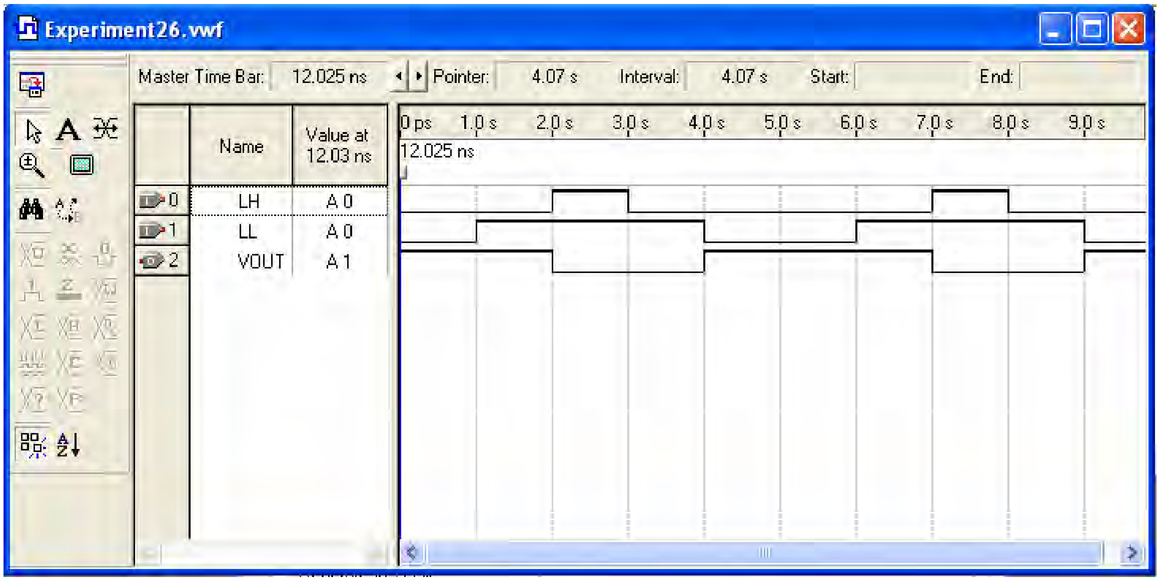
Experiment 26: Introduction to Quartus II Software

Note: This experiment is a tutorial exercise that introduces the Quartus II design tool from Altera. Students work at a computer that has the latest version of Quartus II software loaded. The software is available at www.altera.com. Click on *Download*, *Quartus II software*, and follow the instructions to download the free web edition. A license file is necessary which is specific to that computer. The steps in this tutorial exercise illustrate schematic entry, using the model-1 Molasses Tank outlet valve logic as a basic circuit. It was selected for this exercise because it has a variation on strictly combinational logic (adding feedback) and the software simulation can show this effect nicely. The TTL version of the circuit is shown for reference.



If you want to go beyond this basic introduction to the software, you will need a PLD trainer board, such as the PLDT-2 from RSR Electronics or the Altera board, available from Altera. The trainers allow students to download the experiment to hardware and test the results. For detailed instructions on using both Altera and Xilinx PLD programming software and including downloading to various trainers, see the lab manual *Experiments in Digital Fundamentals with PLD Programming* by David Buchla and Doug Joksch. Five trainers are discussed in detail in the *PLD* manual, with diagrams and tables for the boards, pin assignments, and more.

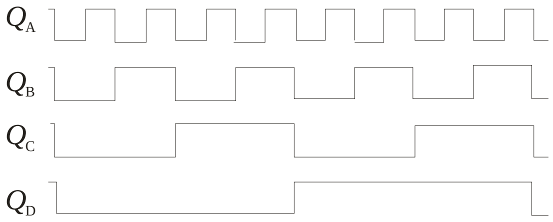
The final output signal that the student should draw is shown on the display. The VOUT signal goes LOW when the valve is open. Notice that it opens only after the upper level sensor is covered but closes only after the lower level sensor is *uncovered* (hence the hysteresis).


Experiment26.vwf

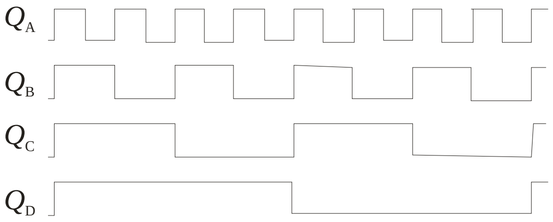
Master Time Bar: 12.025 ns Pointer: 4.07 s Interval: 4.07 s Start: End:

	Name	Value at 12.03 ns
0	LH	A 0
1	LL	A 0
2	VOUT	A 1

Experiment 27: D/A and A/D Conversion



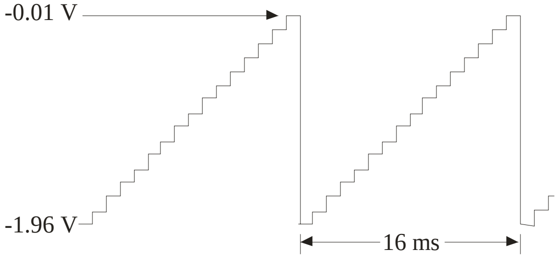
Plot 1



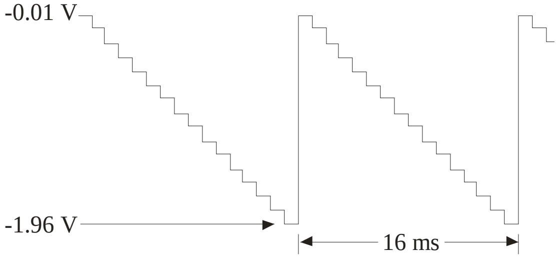
Plot 2

Table 27-1

Quantity	Measured Value
DC full-scale output voltage	***-1.96*** **V**
Volts/step	***0.13*** **V/step**

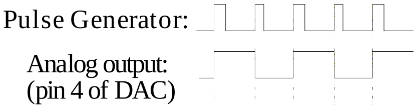


Plot 3



Plot 4

Steps 7 and 8: The digital light meter has an oscillating behavior because of the tracking A/D converter as shown. The D/A output waveform increases its dc level from approximately –2 V to approximately –0.2 V as the light level increases.



Further Investigation Results

Measured data for the circuit in Figure 27-1 is shown on the table. The voltages listed are at the thresholds.

Voltage	Display	Voltage	Display	Voltage	Display	Voltage	Display
0.009 V	0	1.35 V	4	2.59 V	8	3.83 V	⊔
0.437 V	1	1.63 V	5	2.91 V	9	4.17 V	⊑
0.675 V	2	1.93 V	6	3.23 V	⊏	4.49 V	⊢
1.009 V	3	2.26 V	7	3.53 V	⊐	4.85 V	blank

Evaluation and Review Questions:

1. a. Binary 1010 is represented by –1.30 V. b. The output frequency is 62.5 Hz.

2. (–1.95 V)/255 steps = –7.65 mV/step

3 a. The output voltage is larger because the reference current is larger.
b. The resolution measured in volts/step increases. (Note that if measured simply by number of bits, the resolution is unchanged.)

4. The oscillation is caused by the quantizing error between the analog voltage from the photocell and the digitized representation from the converter. If the least significant bit is not used, the oscillation can be eliminated.

5. a. Decrease the reference current by increasing the 5.1 kΩ resistor or decrease the 2.0 kΩ load resistor.
b. No. The change only effects the digital display, not the basic sensitivity of the photocell.

6. A higher clock frequency causes the meter to respond faster to changes but has the disadvantage of causing the display to change rapidly, making it more difficult to read.

Experiment 28: Introduction to the Intel Processors

Note: This experiment is an introduction to microprocessor concepts and uses a tutorial format to acquaint students with how computers store data, the Intel register structure, a few assembly language instructions, and an introduction to programming. *Debug* is the program used for the tutorial and is available on any DOS based computer. Debug is limited to the 16-bit registers. No hardware is needed and questions are answered in the Procedure section. Although some steps do not require a student answer, the sequence of the steps is shown here to aid in following the experiment.

Procedure:

Step 1: Student enters Debug and observes some Debug commands.

Step 2: Answers will vary depending on the computer. The equipment status word 27 C2 given as an example is interpreted as follows:
Number of parallel printer ports attached = __3__
Number of serial ports attached = __1___
Number of diskette devices = __1___
Initial video mode = ___80 x 25______________
Math coprocessor is present? ___yes_______
Diskette drive is present? ___yes_______

Step 3: Answers will vary depending on the computer.

Step 4: The right side will be filled with zeros since 30H represents ASCII 0.

Step 5: Location DS:20 is changed to 31H (ASCII 1).

Step 6: The display will show DS:0020 31 30 DS:0050. (DS will have the current data segment value). This indicates that location DS:20 does not correspond to location DS:50. The two corresponding data points in these locations are 31H and 30H.

Step 7: The 16-bit register set is displayed.

Step 8: The command sequence is **R BX <cr>** followed by :**200 <cr>** and **R CX <cr>** followed by :**F003 <cr>** .

Step 9: The Debug assembler is used to enter a short code.

Step 10: The code in step 9 is traced. The *caps lock* is activated (and shown with an LED).

Step 11: Each loop toggles the *caps lock* function; the light will turn on and off twice before exiting the loop.

Step 12: "Byte-size" data is entered into 17 locations (the 17th location is DS:60 and will have a zero entered as an end of record signal.

Step 13: The modified "BIG" code is assembled. After it is executed with the "go" command, AX will have the largest number in the list, BX will contain 0060H (the last location), and CX will contain 0000H (the final value of the loop counter).

Step 14: The largest number is observed to have been copied into location DS:60.

Further Investigation Results:

The smallest byte value in the list (from DS:50 to DS:5F will be moved into the AL register and into location DS:60. The AX register will show FFH. BX contains 0060H (as before) and CX contains 0000H (as before).

Evaluation and Review Questions:

1 The data in step 2 is from the BIOS RAM area, the data from step 3 is in ROM.

2. Locations from DS:1F0 to DS:1FF are filled with the value FAH.

3. **D 0:0 9**

4. The BIOS work area starts at location 40:0. An example of data that is in this area is information about the peripherals connected to the computer.

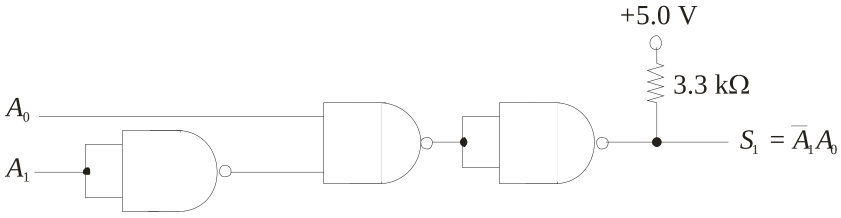
5. When any number is XORed with itself, the result is always zero because 0 XOR 0 = 0 and 1 XOR 1 = 0.

6. The LOOP instruction decrements the CX register and jumps to the start of the loop if CX is not zero; otherwise it executes the next instruction.

Experiment 29: Applications of Bus Systems

Step 3: With all switches on the 14532B open, the chip outputs are all LOW, including the GS output as shown on the second line of the 14532B truth table. The transistors act as inverters, causing the corresponding inputs on the 14051B to be HIGH. If the inhibit line is tied LOW on the 14051B, the internal FET switch is closed that connects X7 to the common out/in line. This completes the path from the 555 timer to the LED representing the least significant bit, causing it to blink.

Decoder schematic: The decoder schematic depends on the particular address assigned to each student. As an example, a 7400 NAND gate, shown decoding address 01, is drawn below:



Further Investigation Results:

Answers may vary. A 4-bit counter such as the 7493A may be selected or the student may design a two-bit counter with *J-K* or *D* flip-flops (such as the ripple counter drawn in Figure 18-2). The counter output is connected to a 7447A BCD to seven segment decoder similar to Figure 3-2.

Evaluation and Review Questions:

1 a. The transistors are used to provide an open-collector connection to the data bus.
b. The display is an input device. The bus can have multiple input devices connected to it without bus contention problems.

2. Each priority encoder is located at a unique address as determined by the individual decoders.
3. The 14051B can be connected as a MUX by connecting inputs to the *X* lines and taking the output from the common *out/in* line. The select inputs behave the same for both configurations.
4. R_{16} is a current limiting resistor for the LEDs. If it opens, no LED will be on.
5. a. The constant HIGH is connected to the selected output, causing the selected LED to stay on.
b. All LEDs will be off.
6. a. The line will be LOW.
b. A data line can be pulled LOW by any transistor connected to the line that is conducting because it provides a direct path to ground.
c. Check the address bus to determine which indicator is being addressed.