Instructor's Manual for
Fundamentals of Logic Design, 5th Ed.

Charles H. Roth, Jr.
The University of Texas at Austin

Brooks/Cole Publishing

Printed in the United States of America
5 4 3 2 1
ISBN ----------

TABLE OF CONTENTS

I: INTRODUCTION

The text, *Fundamentals of Logic Design, 5th edition,* has been designed so that it can be used either for a standard lecture course or for a self-paced course. The text is divided into 20 study units in such a way that the average study time for each unit is about the same. The units have undergone extensive class testing in a self-paced environment and have been revised based on student feedback. Study guides and text material were expanded as required so that almost all students can achieve mastery of all of the objectives. For example, the material on Boolean algebra and algebraic simplification was expanded from 1½ units to 2½ units because students found this topic difficult. A separate unit was added on going from problem statements to state graphs because this topic was difficult for many students.

The textbook contains answers for all of the problems that are assigned in the study guides. This *Instructor's Manual* contains complete solutions to these problems. Solutions to the remaining homework problems as well as all design and simulation exercises are also included in this manual. In the solutions section of this manual, the abbreviation FLD stands for *Fundamentals of Logic Design* (5th ed.).

Information on the self-paced course we teach using the textbook is available at www.ece.utexas.edu/projects/ee316. This website also links to an updated errata list for the text. In addition to the textbook and study guides, teaching a self-paced course requires that a set of tests be prepared for each study unit. This manual contains a sample test for each unit.

1.1 Using the Text in a Lecture Course

Even though the text was developed in a self-paced environment, the text is well suited for use in a standard lecture course. Since the format of the text differs somewhat from a conventional text, a few suggestions for using the text in a lecture course may be appropriate. Except for the inclusion of objectives and study guides, the units in the text differ very little from chapters in a standard textbook. The study guides contain very basic questions, while the problems at the end of each unit are of a more comprehensive nature. For this reason, we suggest that specific study guide questions be assigned for students to work through on their own before working out homework problems selected from those at the end of the unit. The unit tests given in Part 4 of this manual provide a convenient source of additional homework assignments or a source of quiz problems. The text contains many examples that are completely worked out with detailed step-by-step explanations. Discussion of these detailed examples in lecture may not be necessary if the students study them on their own. The lecture time is probably better spent discussing general principles and applications as well as providing help with some of the more difficult topics. Since all of the units have study guides, it would be possible to assign some of the easier topics for self-study and devote the lectures to the more difficult topics.

Working with a class composed largely of Electrical Engineering and Computer Science sophomores and juniors, we cover the 18 units (all units except 6 and 19) of the text in one semester. Units 8, 10, 12, 16, 17, and 20 contain design problems that are suitable for simulation and lab exercises. The design problems help tie together and review the material from a number of preceding units. Units 10, 17, and 20 introduce the VHDL hardware description language. These units may be omitted if desired since no other units depend on them.

1.2 Some Remarks About the Text

In this text, students are taught how to use Boolean algebra effectively, in contrast with many texts that present Boolean algebra and a few examples of its application and then leave it to the student to try to figure out how to use it effectively. For example, use of the theorem $x + yz = (x + y)(x + z)$ in factoring and multiplying out expressions is taught explicitly, and detailed guidelines are given for algebraic simplification.

Sequential circuits are given proper emphasis, with over half of the text devoted to this subject. The pedagogical strategy the text uses in teaching sequential circuits has proven to be very effective. The concepts of state, next state, etc. are first introduced for individual flip-flops, next for counters, then for sequential circuits with inputs, and finally for more abstract sequential circuit models. The use of timing charts, a subject neglected by many texts, is taught both because it is a practical tool widely used by logic design engineers and because it aids in the understanding of sequential circuit behavior.

The most important and often most difficult part of sequential circuit design is formulating the state table or graph from the problem statement, but most texts devote only a few paragraphs to this subject because there is no algorithm. This text devotes a full unit to the subject, presents guidelines for deriving state tables and graphs, and provides programmed exercises that help the student learn this material. Most of the material in the text is treated in a fairly conventional manner with the following exceptions:

(1) The diagonal form of the 5-variable Karnaugh map is introduced in Unit 5. (We find that students make fewer mistakes when using the diagonal form of 5-variable map in comparison with the side-by-side form.) Unit 5 also presents a simple algorithm for finding all essential prime implicants from a Karnaugh map.
(2) Both the state graph approach (Unit 18) and the SM chart approach (Unit 19) for designing sequential control circuits are presented.
(3) The introduction to the VHDL hardware description language in Units 10, 17, and 20 emphasizes the relation between the VHDL code and the actual hardware.

1.3 Using the Text in a Self-Paced Course

This section introduces the personalized system of self-paced instruction (PSI) and offers suggestions for using the text in a self-paced course. PSI (Personalized System of Instruction) is one of the most popular and successful systems used for self-paced instruction. The essential features of the PSI method are

(a) Students are permitted to pace themselves through the course at a rate commensurate with their ability and available time.
(b) A student must demonstrate mastery of each study unit before going onto the next.
(c) The written word is stressed; lectures, if used, are only for motivation and not for transmission of critical information.
(d) Use of proctors permits repeated testing, immediate scoring, and significant personal interaction with the students.

These factors work together to motivate students toward a high level of achievement in a well-designed PSI course.

The PSI method of instruction and its implementation are described in detail in the following references:

Keller, Fred S. and J. Gilmour Sherman, *The Keller Plan Handbook, W. A.* Benjamin, Inc., 1974.
Sherman, J.G., ed., *Personalized System of Instruction: 41 Germinal Papers, W. A.* Benjamin, Inc., 1974.

Results of applying PSI to a first course in logic design of digital systems are described in

Roth, C.H., "Continuing Effectiveness of Personalized Self-Paced Instruction in Digital Systems Engineering", *Engineering Education, Vol. 63, No. 6,* March 1973.

The instructor in charge of a self-paced course will serve as course manager in addition to his role in the classroom. For a small class, he may spend a good part of his time acting as proctor in the classroom, but as class size increases he will have to devote more of his time to supervision of course activities and less time to individual interaction with students. In his managerial role, the instructor is responsible for organizing the course, selection and training of proctors, supervision of proctors, and monitoring of student progress. The proctors play an important role in the success of

a self-paced course, and therefore their selection, training, and supervision is very important. After an initial session to discuss proper ways of grading readiness tests and interacting with students, weekly proctor meetings to discuss course procedures and problems may be appropriate.

A progress chart showing the units completed by each student is very helpful in monitoring student progress through the course. The instructor may wish to have individual conferences with students who fall too far behind. The instructor needs to be available in the classroom to answer individual student questions and to assist with grading of readiness tests as needed. He should make a special point to speak with the weak or slow students and give them a word of encouragement. From time to time he may need to settle differences which arise between proctors and students.

Various strategies for organizing a PSI course are described in the *Keller Plan Handbook*. The procedures that we use for operating our self-paced digital logic course are described in "Unit 0", which is available on the web: www.ece.utexas.edu/projects/ee316. At the first class meeting, we hand out a copy of Unit 0. The students are asked to read through Unit 0 and take a short test on the course procedures. This test is immediately evaluated so that the student can complete Unit 0 before the end of the first class period. In this way, the student is exposed to the basic way the course operates and is ready to proceed immediately with Unit 1 in the textbook.

During a typical class period, some of the students will spend their time studying but most of the students will come prepared to take a unit test. At the beginning of the period, the instructor or a proctor will be available to answer student questions on an individual basis. Later in the period, most of the time will be spent evaluating unit tests. We have found that a standard 50 minute class period is not long enough for a PSI session. We usually schedule sessions of 1½ or 2 hours or longer depending on class size. This allow adequate time for a student to have his questions answered, take a unit test, and have his test graded. Interactive grading of the tests with the student present is an important part of the PSI system and adequate time must be allowed for this activity. If you have a large number of students and proctors, you may wish to prepare a manual for guidance of your proctors. The procedures that we use for evaluating unit tests are described in a Proctor's Manual, which can be obtained by writing to Professor Charles H. Roth.

1.4. Use of Computer Software

Three software packages are included on the CD that accompanies the textbook. The first is a logic simulator program called *SimUaid*, the second is a basic computer-aided logic design program called *LogicAid*, and the third is a VHDL Simulator called *DirectVHDL*. In addition, we use the Xilinx ISE software for synthesizing VHDL code and downloading to CPLD or FPGA circuit boards. The Xilinx ISE software is available at nominal cost through the Xilinx University Program (for information, go to www.xilinx.com/univ/overview.html). A "Webpack" version of the Xilinx software is also available for downloading from the Xilinx.com website.

SimUaid provides an easy way for students to test their logic designs by simulating them. We first introduce *SimUaid* in Unit 4, where we ask the students to design a simple logic circuit such as problem 4.13 or 4.14, and simulate it. *SimUaid* is easy to learn, and it is highly interactive so that students can flip a simulated switch and immediately observe the result. In Unit 8, students design a multiple-output combinational logic circuit using NAND and NOR gates and test its operation using *SimUaid*. Students can use the simulator to help them understand the operation of latches and flip-flops in Unit 11. In Unit 12, we ask them to design a counter and simulate it (one part of problem 12.10). In Unit 16, students use *SimUaid* to test their sequential circuit designs. They can also generate VHDL code from their *SimUaid* circuit, synthesize it, and download it to a circuit board for hardware testing. In Unit 18, students can use the advanced features of *SimUaid* to simulate a multiplier or divider controlled by a state machine.

LogicAid provides an easy way to introduce students to the use of the computer in the logic design process. It enables them to solve larger, more practical design problems than they could by hand. They can also use *LogicAid* to verify solutions that they have worked out by hand. Instructors can use the program for grading homework and quizzes. We first introduce *LogicAid* in Unit 5. The program has a Karnaugh Map Tutorial mode that is very useful in teaching students to solve Karnaugh map problems. This tutorial mode helps students learn to derive minimum solutions from a Karnaugh map by informing them at each step whether that step is correct or not. It also forces them to choose essential prime implicants first. When in the KMap tutor mode, *LogicAid* prints "KMT" at the top of each output page, so you can check to see if the problems were actually solved in the tutorial mode.

Students can use *LogicAid* to help them solve design problems in Units 8, 16, 18, 19 and other units. For designing sequential circuits, they can input a state graph, convert it to a state table, reduce the state table, make a state assignment, and derive minimized logic equations for outputs and flip-flop inputs.

The *LogicAid* State Table Checker is useful for Units 14 and 16, and for other units in which students construct state tables. It allows students to check their solutions without revealing the correct answers. If the solution is wrong, the program displays a short input sequence for which the student's table fails. The *LogicAid* folder on the CD contains encoded copies of solutions for most of the state graph problems in *Fundamentals of Logic Design, 5th Ed.* If you wish to create a password-protected solution file for other state table problems, enter the state table into *LogicAid*, syntax check it, and then hold down the Ctrl key while you select Save As on the file menu. The Partial Graph Checker serves as a state graph tutor that allows a student to check his work at each step while constructing a state graph. If the student makes a mistake, it provides feedback so that the student can correct his answer. The partial graph checker works with any state graph problem for which an encoded state table solution file is provided.

The DirectVHDL simulator helps students learn VHDL syntax because it provides immediate visual feedback when they make mistakes. Our students use it for simulating VHDL code in Units 10, 17, and 20. Students can simulate and debug their code at home and then bring the code into lab for synthesis and hardware testing.

1.5. Suggested Equipment for Laboratory Exercises

Many types of logic lab equipment are available that are adequate to perform the lab exercises. Since most logic design is done today using programmable logic instead of individual ICs, we now recommend use of CPLDs or FPGAs for hardware implementation of logic circuit designs. At present, we are using the Digilab XCR Plus boards, which use a Xilinx Coolrunner XCR 3064 CPLD. This CPLD has an adequate number of logic cells to implement the lab exercises in the text. The board has 8 switches, 4 pushbuttons, 8 single LEDs, and two 7-segment LEDs.. The board also has a breadboard area. Information about this board and other CPLD and FPGA boards made by Digilent can be found on their website, www.digilentinc.com. We use the board in conjunction with the Xiliinx ISE software mentioned earlier.

II. SOLUTIONS TO HOMEWORK PROBLEMS
Unit 1 Problem Solutions

1.1 (a) 757.25_{10}

16 | 757		0.25
16 | 47	r5	16
16 | 2	r15=F_{16}	(4).00
0	r2	

$\therefore 757.25_{10} = 2F5.40_{16}$
$= \underline{0010}\ \underline{1111}\ \underline{0101}.\underline{0100}\ \underline{0000}_2$
2 F 5 4 0

1.1 (b) 123.17_{10}

16 | 123		0.17
16 | 7	r11	16
0	r7	(2).72
		16
		(11).52
		16
		(8).32

$\therefore 123.17_{10} = 7B.2B_{16}$
$= \underline{0111}\ \underline{1011}.\underline{0010}\ \underline{1011}_2$
7 B 2 B

1.1 (c) 356.89_{10}

16 | 356		0.89
16 | 22	r4	16
16 | 1	r6	(14).24
0	r1	16
		(3).84
		16
		(13).44
		16
		(7).04

$\therefore 356.89_{10} = 164.E3_{16}$
$= \underline{0001}\ \underline{0110}\ \underline{0100}.\underline{1110}\ \underline{0011}_2$
1 6 4 E 3

1.1 (d) 1063.5_{10}

16 | 1063		0.5
16 | 66	r7	16
16 | 4	r2	(8).00
0	r4	

$\therefore 1063.5_{10} = 427.8_{16}$
$= \underline{0100}\ \underline{0010}\ \underline{0111}.\underline{1000}_2$
4 2 7 8

1.2 (a) $EB1.6_{16} = E \times 16^2 + B \times 16^1 + 1 \times 16^0 + 6 \times 16^{-1}$
$= 14 \times 256 + 11 \times 16 + 1 + 6/16 = 3761.375_{10}$
$\underline{1110}\ \underline{1011}\ \underline{0001}.\underline{011(0)}_2$
E B 1 6

$7261.3_8 = 7 \times 8^3 + 2 \times 8^2 + 6 \times 8^1 + 1 + 3 \times 8^{-1}$
$= 7 \times 512 + 2 \times 64 + 6 \times 8 + 1 + 3/8 = 3761.375_{10}$
$\underline{111}\ \underline{010}\ \underline{110}\ \underline{001}.\underline{011}_8$
7 2 6 1 3

1.2(b) $59D.C_{16} = 5 \times 16^2 + 9 \times 16^1 + D \times 16^0 + C \times 16^{-1}$
$= 5 \times 256 + 9 \times 16 + 13 + 12/16 = 1437.75_{10}$
$\underline{0101}\ \underline{1001}\ \underline{1101}.\underline{1100}_{16}$
5 9 D C

$2635.6_8 = 2 \times 8^3 + 6 \times 8^2 + 3 \times 8^1 + 5 \times 8^0 + 6 \times 8^{-1}$
$= 2 \times 512 + 6 \times 64 + 3 \times 8 + 5 + 6/8 = 1437.75_{10}$
$\underline{010}\ \underline{110}\ \underline{011}\ \underline{101}.\underline{110}_8$
2 6 3 5 6

1.3 $3BA.25_{14} = 3 \times 14^2 + 11 \times 14^1 + 10 \times 14^0 + 2 \times 14^{-1} + 5 \times 14^{-2}$
$= 588 + 154 + 10 + 0.1684 = 752.1684_{10}$

6 | 752		0.1684
6 | 125	r2	6
6 | 20	r5	(1).0104
6 | 3	r2	6
0	r3	(0).0624
		6
		(0).3744
		6
		(2).2464
		6
		(1).4784

$\therefore 3BA.25_{14} = 752.1684_{10} = 3252.1002_6$

1.4 (a) 1457.11_{10}

16 | 1457		0.11
16 | 91	r1	16
16 | 5	r11=B_{16}	(1).76
0	r5	16
		(12).16

$\therefore 1457.11_{10} = 5B1.1C_{16}$

1.4 (b) $5B1.1C_{16} = \overbrace{5}\ \overbrace{B}\ \overbrace{1}\ \overbrace{1}\ \overbrace{C}$
$\overbrace{0101}\ \overbrace{1011}\ \overbrace{0001}.\overbrace{0001}\ \overbrace{1100}_2 = 2661.070_8$
2 6 6 1 0 7 0

1.4 (c) $5B1.1C_{16} = \underline{11}\ \underline{23}\ \underline{01}.\underline{01}\ \underline{30}_4$
5 B 1 1 C

1.4 (d) $DEC.A_{16} = D \times 16^2 + E \times 16^1 + C \times 16^0 + A \times 16^{-1}$
$= 3328 + 224 + 12 + 0.625 = 3564.625_{10}$

1.5 (a)

Add	Sub	Multiply
(carries 111)		
1111	1111	1111
+1010	−1010	×1010
11001	0101	0000
		1111
		11110
		0000
		011110
		1111
		10010110

1.5 (b, c) *See FLD p. 625 for solution.*

1.6, 1.7, 1.8 *See FLD p. 625 for solution.*

1.10 (a) 1305.375_{10}

16 | 1305		0.375
16 | 81	r9	16
5	r1	(6).000

$\therefore 1305.375_{10} = 519.600_{16}$
$= 0101\ 0001\ 1001.0110\ 0000\ 0000_2$
(5 1 9 6 0 0)

1.10 (b) 111.33_{10}

16 | 111		0.33
6	$r15 = F_{16}$	16
		(5).28
		16
		(4).48

$\therefore 111.33_{10} = 6F.54_{16}$
$= 0110\ 1111.0101\ 0100_2$
(6 F 5 4)

1.10 (c) 301.12_{10}

16 | 301		0.12
16 | 18	r13	16
1	r2	(1).92
		16
		(14).72

$\therefore 301.12_{10} = 12D.1E_8$
$= 0001\ 0010\ 1101.0001\ 1110_2$
(1 2 D 1 E)

1.10 (d) 1644.875_{10}

16 | 1644		0.875
16 | 102	r12	16
6	r6	(14).000

$\therefore 1644.875_{10} = 66C.E00_{16}$
$= 0110\ 0110\ 1100.1110\ 0000\ 0000_2$
(6 6 C E 0 0)

1.11 (a) $101\ 111\ 010\ 100.101_2 = 5724.5_8$
$= 5 \times 8^3 + 7 \times 8^2 + 2 \times 8^1 + 4 \times 8^0 + 5 \times 8^{-1}$
$= 5 \times 512 + 7 \times 64 + 2 \times 8 + 4 + 5/8$
$= 3028.625_{10}$

1.11 (b) $100\ 001\ 101\ 111.010_2 = 4157.2_8$
$= 4 \times 8^3 + 1 \times 8^2\ 5 \times 8^1 + 7 \times 8^0 + 2 \times 8^{-1}$
$= 4 \times 512 + 1 \times 64 + 5 \times 8 + 7 + 2/8$
$= 2159.25_{10}$

$1011\ 1101\ 0100.1010_2 = BD4.A_{16}$
$B \times 16^2 + D \times 16^1 + 4 \times 16^0 + A \times 16^{-1}$
$11 \times 256 + 13 \times 16 + 4 + 10/16$
$= 3028.625_{10}$

$1000\ 0110\ 1111.0100_2 = 86F.4_{16}$
$= 8 \times 16^2 + 6 \times 16^1 + F \times 16^0 \times 4 \times 16^{-1}$
$= 8 \times 256 + 6 \times 16 + 15 + 4/16$
$= 2159.25_{10}$

1.12 (a) $375.54_8 = 3 \times 64 + 7 \times 8 + 5 + 5/8 + 4/64 = 253.6875_{10}$

3 | 253		0.69
3 | 84	r1	3
3 | 28	r0	(2).07
3 | 9	r1	3
3 | 3	r0	(0).21
3 | 1	r0	3
0	r1	(0).63
		3
		(1).89

$\therefore 375.54_8 = 100101.2001_3$

1.12 (b) 384.74_{10}

4 | 384		0.74
4 | 96	r0	4
4 | 24	r0	(2).96
4 | 6	r0	4
4 | 1	r2	(3).84
0	r1	4
		(3).36

$\therefore 384.74_{10} = 12000.233113_4\ldots$

1.12 (c) $A52.A4_{11} = 10 \times 121 + 5 \times 11 + 2 + 10/11 + 4/121$
$= 1267.94_{10}$

9 | 1267		0.94
9 | 140	r7	9
9 | 15	r5	(8).46
9 | 1	r6	9
0	r1	(4).14

$\therefore A52.A4_{11} = 1267.94_{10} = 1657.8427_9...$

1.13 1110212.20211_3
01 11 02 12.20 21 10 = 1425.673_9

Base 3	Base 9
00	0
01	1
02	2
10	3
11	4
12	5
20	6
21	7
22	8

1.14 (a) $2983\ 63/64_{10}$ =

8 | 2983		0.984
8 | 372	r7	8
8 | 46	r4	(7).872
9 | 5	r6	8
0	r5	(6).976

$\therefore 2983\ 63/64_{10} = 5647.76_8$ (or 5647.77_8)
$= 101\ 110\ 100\ 111.111\ 110_2$
(or $101\ 110\ 100\ 111.111\ 111_{2)}$

1.14 (b) 93.70_{10}

8 | 93		0.70
8 | 11	r5	8
8 | 1	r3	(5).60
0	r1	8
		(4).80

$\therefore 93.70_{10} = 135.54_8 = 001\ 011\ 101.101\ 100_2$

1.14 (c) $1900\ 31/32_{10}$

8 | 1900		0.969
8 | 273	r4	8
8 | 29	r5	(7).752
9 | 3	r5	8
0	r3	(6).016

$\therefore 1900\ 31/32_{10} = 3554.76_8 = 011\ 101\ 101\ 100.111\ 110_2$

1.14 (d) 109.30_{10}

8 | 109		0.30
8 | 13	r5	8
8 | 1	r5	(2).40
0	r1	8
		(3).20

$\therefore 109.30_{10} = 155.23_8$
$= 001\ 101\ 101.010\ 011_2$

1.15(a)

```
 1 1 1          1 1 1
 1111 (Add)     1111 (Subtract)
 1001           1001
11000           0110
```

```
    1111 (Multiply)
    1001
    1111
   0000
   01111
  0000
 1 1 1
 001111
  1111
 10000111
```

1.15(b)

```
   1              1 1  1 1
 1101001 (Add)    1101001 (Sub)
  110110           110110
10011111          110011
```

```
    1101001  (Mult)
     110110
    0000000
   1101001
   11010010
  1101001
 1001110110
 0000000
 1 1 1
 1001110110
  1101001
 100100000110
 1101001
 1011000100110
```

1.15(c)

$$\begin{array}{r} {\scriptstyle 1}\\ 110010\\ 11101\\ \hline 1001111 \end{array}\ \text{(Add)} \qquad \begin{array}{r} {\scriptstyle 111\ 1}\\ 110010\\ 11101\\ \hline 10101 \end{array}\ \text{(Sub)} \qquad \begin{array}{r} 110010\\ 11101\\ \hline 110010\\ 000000\\ 0110010\\ 110010\\ \hline 1111010\\ 110010\\ \hline 1010001010\\ 110010\\ \hline 10110101010 \end{array}\ \text{(Mult)}$$

1.16(a)

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1}\\ 10100100\\ 01110011\\ \hline 0110001 \end{array}$$

(b)

$$\begin{array}{r} {\scriptstyle 1\ \ 1}\\ 10010011\\ 01011001\\ \hline 00111010 \end{array}$$

(c)

$$\begin{array}{r} {\scriptstyle 11}\\ 11110011\\ 10011110\\ \hline 01010101 \end{array}$$

1.17(a)

```
      101110  Quotient
101 )11101001
     101
      1001
      101
       1000
       101
        110
        101
         11  Remainder
```

1.17(b)

```
      11011  Quotient
1110 )110000001
      1110
      10100
      1110
       11000
       1110
        10101
        1110
          111 Remainder
```

1.17(c)

```
      1100  Quotient
1001 )1110010
      1001
       1010
       1001
        110 Remainder
```

1.18 (a)

```
      10111  Quotient
110 )10001101
     110
      1011
      110
       1010
       110
        1001
        110
          11 Remainder
```

1.18(b)

```
      100011  Quotient
1011 )110000011
      1011
       10001
       1011
        1101
        1011
          10 Remainder
```

1.18(c)

```
      1011  Quotient
1010 )1110100
     1010
      10010
      1010
       10000
       1010
        110 Remainder
```

1.19

	4 3 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 1 0 0
4	1 0 0 0
5	1 0 0 1
6	1 0 1 0
7	1 1 0 0
8	1 1 0 1
9	1 1 1 0

9154 = 1110 0001 1001 1000

1.20 5-3-1-1 is possible, but 6-4-1-1 is not, because there is no way to represent 3 or 9.

Alternate Solutions:

	5 3 1 1	
0	0 0 0 0	
1	0 0 0 1	(0010)
2	0 0 1 1	
3	0 1 0 0	
4	0 1 0 1	(0110)
5	1 0 0 0	
6	1 0 0 1	(1010)
7	1 0 1 1	
8	1 1 0 0	
9	1 1 0 1	(1110)

1.21 5-4-1-1 is not possible, because there is no way to represent 3 or 8. 6-3-2-1 is possible:

	6 3 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 1 0 0
4	0 1 0 1
5	0 1 1 0
6	1 0 0 0
7	1 0 0 1
8	1 0 1 0
9	1 1 0 0

1.22 Alternate Solutions:

	6 2 2 1	
0	0 0 0 0	
1	0 0 0 1	
2	0 0 1 0	(0100)
3	0 0 1 1	(0101)
4	0 1 1 0	
5	0 1 1 1	
6	1 0 0 0	
7	1 0 0 1	
8	1 0 1 0	(1100)
9	1 0 1 1	(1101)

1100 0011 = 83

1.23 Alternate Solutions:

	5 2 2 1	
0	0 0 0 0	
1	0 0 0 1	
2	0 0 1 0	(0100)
3	0 0 1 1	(0101)
4	0 1 1 0	
5	1 0 0 0	
6	1 0 0 1	
7	1 0 1 0	(1100)
8	1 0 1 1	(1101)
9	1 1 1 0	

1110 0110 = 94

1.24 Alternate Solutions:

	7 3 2 1	
0	0 0 0 0	
1	0 0 0 1	
2	0 0 1 0	
3	0 1 0 0	(0011)
4	0 1 0 1	
5	0 1 1 0	
6	0 1 1 1	
7	1 0 0 0	
8	1 0 0 1	
9	1 0 1 0	
A	1 1 0 0	(1011)
B	1 1 0 1	

B4A9 = 1101 0101 1100 1010
Alt.: = " " 1011 "

1.25 (a) 222.22_{10}

16 | 222			0.22
16 | 13	r14		16
0	r13		(3).52
			16
			(8).32

$\therefore 222.22_{10} = DE.38_{16}$
$= \underline{1000100}\ \underline{1000101}\ \underline{0101110}\ \underline{0110011}\ \underline{0111000}$
D E . 3 8

1.25 (b) 183.81_{10}

16 | 183			0.81
16 | 11	r7		16
0	r11		(12).96
			16
			(15).36

$\therefore 183.81_{10} = B7.CF_{16}$
$= \underline{1000010}\ \underline{0110111}\ \underline{0101110}\ \underline{1000011}\ \underline{1000110}$
B 7 . C F

1.26 (a)

In 2's complement

(–10) + (–11)
110110
110101
(1)101011 (–21)

In 1's complement

(–10) + (–11)
110101
110100
1101001
1
101010 (–21)

1.26 (b)

In 2's complement

(–10) + (–6)
110110
11010
(1)110000 (–16)

In 1's complement

(–10) + (–6)
110101
111001
1101110
1
101111 (–16)

1.26 (c)

(–8) + (–11)
111000
110101
(1)101101 (–19)

(–8) + (–11)
110111
110100
1101011
1
101100 (–19)

1.26 (d)

11 + 9
001011
001001
010100 (20)

11 + 9
001011
001001
010100 (20)

1.26 (e)

(–11) + (–4)
110101
111100
(1)110001 (–15)

(–11) + (–4)
110100
111011
1101111
1
110000 (–15)

1.27 (a)

01001–11010

In 2's complement

01001
+ 00110
01111

In 1's complement

01001
+ 00101
01110

1.27 (b)

In 2's complement

11010
+ 00111
(1)00001

In 1's complement

11010
+ 00110
100000
1
00001

1.27 (c)

10110
+ 10011
(1)01001
overflow

10110
+ 10010
101000
1
01001
overflow

1.27 (d)

11011
+11001
(1)10100

11011
+11000
110011
1
10100

1.27 (e)

In 2's complement

11100
+ 01011
(1)00111

In 1's complement

11100
+ 01010
100110
1
00111

1.28 (a)

In 2's complement

11010
+ 01100
(1)00110

In 1's complement

11010
+ 01011
100101
1
00110

1.28 (b)

01011
+ 01000
10011

01011
+ 00111
10010

1.28 (c)

10001
+ 10110
(1)00111
overflow

10001
+ 10101
100110
1
00111
overflow

1.28 (d)

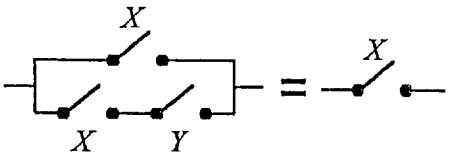
10101
+ 00110
11011

10101
+ 00101
11010

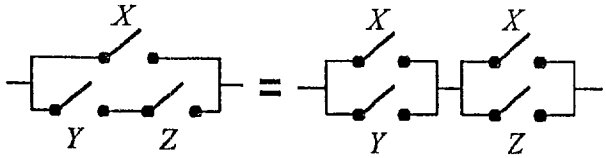
Unit 2 Problem Solutions

2.1 *See FLD p. 626 for solution.*

2.2 (a) In both cases, if X = 0, the transmission is 0, and if X = 1, the transmission is 1.



2.2 (b) In both cases, if X = 0, the transmission is YZ, and if X = 1, the transmission is 1.



2.3 *For the answer to 2.3, refer to FLD p. 626*

2.4 (a) $F = [(A \cdot 1) + (A \cdot 1)] + E + BCD = A + E + BCD$

2.4 (b) $Y = (AB' + (AB + B))\,B + A = (AB' + B)\,B + A$
$= (A + B)\,B + A = AB + B + A = A + B$

2.5 (a) $(A + B)\,(C + B)\,(D' + B)\,(ACD' + E)$
$= (AC + B)\,(D' + B)\,(ACD' + E)$ By Th. 8D
$= (ACD' + B)\,(ACD' + E)$ By Th. 8D
$= ACD' + BE$ By Th. 8D

2.5 (b) $(A' + B + C')\,(A' + C' + D)\,(B' + D')$
$= (A' + C' + BD)\,(B' + D')$
{By Th. 8D with $X = A' + C'$}
$= A'B' + B'C' + B'BD + A'D' + C'D' + BDD'$
$= A'B' + A'D' + C'B' + C'D'$

2.6 (a) $AB + C'D' = (AB + C')\,(AB + D')$
$= (A + C')\,(B + C')\,(A + D')\,(B + D')$

2.6 (b) $WX + WY'X + ZYX = X(W + WY' + ZY)$
$= X(W + ZY)$ {By Th. 10}
$= X(W + Z)\,(W + Y)$

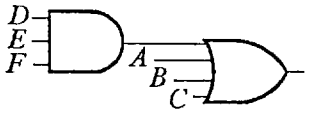
2.6 (c) $A'BC + EF + DEF' = A'BC + E(F + DF')$
$= A'BC + E(F + D) = (A'BC + E)\,(A'BC + F + D)$
$= (A' + E)\,(B + E)\,(C + E)\,(A' + F + D)$
$(B + F + D)\,(C + F + D)$

2.6 (d) $XYZ + W'Z + XQ'Z = Z(XY + W' + XQ')$
$= Z[W' + X(Y + Q')]$
$= Z(W' + X)\,(W' + Y + Q')$ By Th. 8D

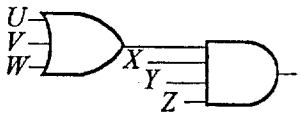
2.6 (e) $ACD' + C'D' + A'C = D'\,(AC + C') + A'C$
$= D'\,(A + C') + A'C$ By Th. 11D
$= (D' + A'C)\,(A + C' + A'C)$
$= (D' + A')\,(D' + C')\,(A + C' + A')$ By Th. 11D
$= (A' + D')\,(C + D')$

2.6 (f) $A + BC + DE = (A + BD + D)\,(A + BC + E)$
$= (A + B + D)\,(A + C + D)\,(A + B + E)$
$(A + C + E)$

2.7 (a) $\underline{(A + B + C} + D)\,\underline{(A + B + C} + E)\,\underline{(A + B + C} + F)$
$= \underline{A + B + C} + DEF$
Apply second distributive law (Th. 8D) twice



2.7 (b) $W\underline{XYZ} + V\underline{XYZ} + U\underline{XYZ} = \underline{XYZ}\,(W + V + U)$
By first distributive law (Th. 8)



2.8 (a) $[(AB)' + C'D]' = AB(C'D)' = AB(C + D')$
$= ABC + ABD'$

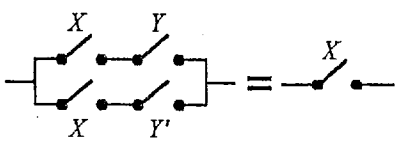
2.8 (b) $[A + B\,(C' + D)]' = A'(B(C' + D))'$
$= A'(B' + (C' + D)') = A'(B' + CD') = A'B' + A'CD'$

2.8 (c) $((A + B')\,C)'\,(A + B)\,(C + A)'$
$= (A'B + C')\,(A + B)C'A' = (A'B + C')A'BC'$
$= A'BC'$

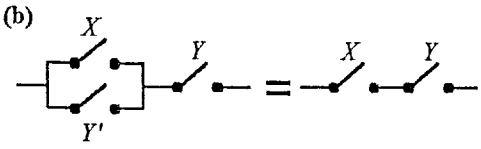
2.9 (a) $F = [(A + B)' + (A + (A + B)')'] (A + (A + B)')'$
$= (A + (A + B)')'$ *By Th. 10D with* $X=(A+(A+B)')'$
$= A'(A + B) = A'B$

2.9 (b) $G = \{[(R + S + T)' PT(R + S)']' T\}'$
$= (R + S + T)' PT(R + S)' + T'$
$= T' + (R'S'T') P(R'S')T = T' + PR'S'T'T = T'$

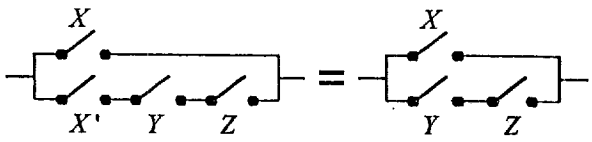
2.10 (a)



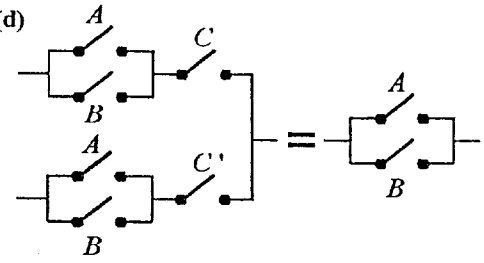
2.10 (b)



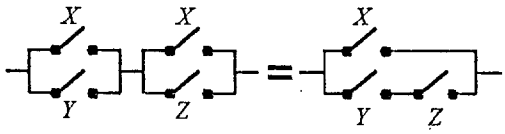
2.10 (c)



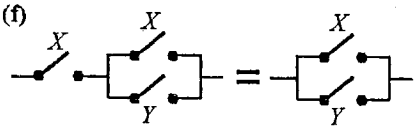
2.10 (d)



2.10 (e)



2.10 (f)



2.11 (a) $A'B'C + (A'B'C)' = 1$ By Th. 5

2.11 (b) $A(B + C'D) + B + C'D = B + C'D$ By Th. 10

2.11 (c) $A + B + C'D(A + B)' = A + B + C'D$
By Th. 11D

2.11 (d) $(A'B + CD')(A'B + CE) = A'B + CD'E$ By Th. 8D

2.11 (e) $[AB' + (CD)' + E'F]CD = AB'CD + E'FCD$
By Th. 8

2.11 (f) $(A' + BC)(D'E + F)' + (D'E + F)$
$= D'E + F + A' + BC$ By Th. 11D

2.12 (a) $(X + Y'Z)(X + Y'Z)' = 0$ By Th. 5

2.12 (b) $(W + X' + YZ)(W' + X' + YZ) = X' + YZ$ By Th. 9D

2.12 (c) $(V'W + X)' (X + Y + Z + V'W) = (V'W + X)' (Y + Z)$
By Th. 11

2.12 (d) $(V' + W'X)(V' + W'X + Y'Z) = V' + W'X$
By Th. 10D

2.12 (e) $(W' + X)YZ' + (W' + X)'YZ' = YZ'$ By Th. 9

2.12 (f) $(V' + U + W)(WX + Y + UZ') + (WX + UZ' + Y)$
$= WX + UZ' + Y$ By Th. 10

2.13 (a) $F1 = A'A + B + (B + B) = 0 + B + B = B$

2.13 (b) $F2 = A'A' + AB' = A' + AB' = A' + B'$

2.13 (c) $F3 = [(AB + C)'D][(AB + C) + D]$
$= (AB + C)'D (AB + C) + (AB + C)'D$
$= (AB + C)'D$ By Th. 5D & Th. 2D

2.13 (d) $Z = [(A + B)C]' + (A + B)CD = [(A + B)C]' + D$
By Th. 11D with $Y = [(A + B) C]'$
$= A'B' + C' + D'$

2.14 (a) $ACF(B + E + D)$

2.14 (b) $W + Y + Z + VUX$

2.15 (a) $H'I' + JK = (H'I' + J)(H'I' + K) = (H' + J)(I' + J)(H' + K)(I' + K)$

2.15 (b) $ABC + A'B'C + CD' = C(AB + A'B' + D') = C[(A' + B)(A + B') + D'] = C(A' + B + D')(A + B' + D')$

2.15 (c) $AB' + ACD + ADE' = A(B' + CD + DE') = A[B' + D(C + E')] = A(B' + D)(B' + C + E')$

2.15 (d) $AB'C + B'CD' + EF' = AB'C + B'CD' + EF' = B'C(A + D') + EF' = (B'C + EF')(A + D' + EF')$
$= (B' + E)(B' + F')(C + E)(C + F')(A + D' + E)(A + D' + F')$

2.15 (e) $WX'Y + W'X' + W'Y' = X'(WY + W') + W'Y' = X'(W' + Y) + W'Y' = (X' + W')(X' + Y')(W' + Y + W')(W' + Y + Y')$
$= (X' + W')(X' + Y')(W' + Y)$

2.15 (f) $AB' + (CD' + E) = AB' + (C + E)(D' + E) = (AB' + C + E)(AB' + D' + E)$
$= (A + C + E)(B' + C + E)(A + D' + E)(B' + D' + E)$

2.16 (a) $W + X'YZ = (W + X')(W + Y)(W + Z)$

2.16 (b) $VW + XY' + Z$
$= (V+X+Z)(V+Y'+Z)(W+X+Z)(W+Y'+Z)$

2.16 (c) $A'B'C + B'CD' + B'E' = B'(A'C + CD' + E')$
$= B'[E' + C(A' + D')]$
$= B'(E' + C)(E' + A' + D')$

2.16 (d) $ABC + ADE' + ABF' = A(BC + DE' + BF')$
$= A[DE' + B(C + F')]$
$= A(DE' + B)(DE' + C + F')$
$= A(B + D)(B + E')(C + F' + D)(C + F' + E')$

2.17 (a) $[(XY)' + (X' + Y')'Z] = X' + Y' + (X' + Y')'Z$
$= X' + Y' + Z$ By Th. 11D with $Y = (X' + Y')$

2.17 (b) $(X + (Y(Z + W)')')' = X'Y(Z + W)' = X'YZ'W'$

2.17 (c) $[(A' + B')' + (A'B'C)' + C'D']'$
$= (A' + B')A'B'C(C + D) = A'B'C$

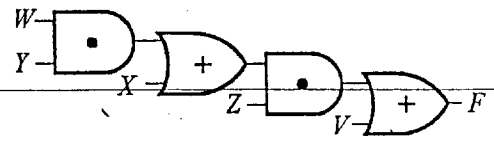
2.17 (d) $(A + B)CD' + (A + B)' = CD' + (A + B)'$
{By Th. 11D with $Y = (A + B)'$}
$= CD' + A'B'$

2.18 (a) $F = [(A' + B)'B]'C + B = [A' + B + B']C + B$
$= C + B$

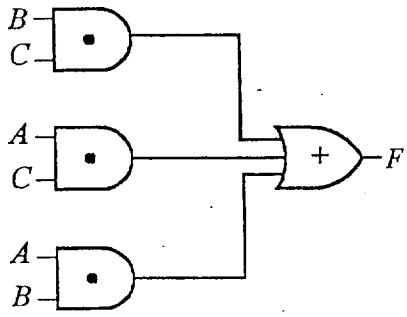
2.18 (b) $G = [(AB)'(B + C)]'C = (AB + B'C')C = ABC$

2.18 (c) $H = [W'X'(Y' + Z')]' = W + X + YZ$

2.19 $F = (\underline{V + X} + W)(\underline{V + X} + Y)(V + Z)$
$= (V + X + WY)(V + Z) = V + Z(X + WY)$
By Th. 8D with $X = V$

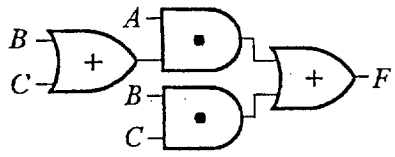


2.20 (a) $F = ABC + A'BC + AB'C + ABC'$
$= BC + AB'C + ABC'$ (By Th. 9)
$= C(B + AB') + ABC' = C(A + B) + ABC'$
(By Th. 11D)
$= AC + BC + ABC' = AC + B(C + AC')$
$= AC + B(A + C) = AC + AB + BC$



2.20 (b) Beginning with the answer to (a):

$F = A(B + C) + BC$



Alternate solutions:

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

$F = AC + B(A + C)$

2.21 (a)

W X Y Z	*W'XY*	*WZ*	*W'XY+WZ*	*W'+Z*	*W+XY*	*(W'+Z)(W+XY)*
0 0 0 0	0	0	0	1	0	0
0 0 0 1	0	0	0	1	0	0
0 0 1 0	0	0	0	1	0	0
0 0 1 1	0	0	0	1	0	0
0 1 0 0	0	0	0	1	0	0
0 1 0 1	0	0	0	1	0	0
0 1 1 0	1	0	1	1	1	1
0 1 1 1	1	0	1	1	1	1
1 0 0 0	0	0	0	0	1	0
1 0 0 1	0	1	1	1	1	1
1 0 1 0	0	0	0	0	1	0
1 0 1 1	0	1	1	1	1	1
1 1 0 0	0	0	0	0	1	0
1 1 0 1	0	1	1	1	1	1
1 1 1 0	0	0	0	0	1	0
1 1 1 1	0	1	1	1	1	1

2.21 (b)

A B C	*A+C*	*AB+C'*	*(A+C)(AB+C')*	*AB*	*AC'*	*AB+AC'*
0 0 0	0	1	0	0	0	0
0 0 1	1	0	0	0	0	0
0 1 0	0	1	0	0	0	0
0 1 1	1	0	0	0	0	0
1 0 0	1	1	1	0	1	1
1 0 1	1	0	0	0	0	0
1 1 0	1	1	1	1	1	1
1 1 1	1	1	1	1	0	1

2.21 (c)

X Y Z	*X+Y*	*X'+Z*	*(X+Y)(X'+Z)*	*XZ*	*X'Y*	*XZ+X'Y*
0 0 0	0	1	0	0	0	0
0 0 1	0	1	0	0	0	0
0 1 0	1	1	1	0	1	1
0 1 1	1	1	1	0	1	1
1 0 0	1	0	0	0	0	0
1 0 1	1	1	1	1	0	1
1 1 0	1	0	0	0	0	0
1 1 1	1	1	1	1	0	1

2-21 (d)

X Y Z	*XY*	*YZ*	*X'Z*	*XY+YZ+X'Z*	*XY+X'Z*
0 0 0	0	0	0	0	0
0 0 1	0	0	1	1	1
0 1 0	0	0	0	0	0
0 1 1	0	1	1	1	1
1 0 0	0	0	0	0	0
1 0 1	0	0	0	0	0
1 1 0	1	0	0	1	1
1 1 1	1	1	0	1	1

2.21 (e)

XYZ	$X+Y$	$Y+Z$	$X'+Z$	$(X+Y)(Y+Z)(X'+Z)$	$(X+Y)(X'+Z)$
000	0	0	1	0	0
001	0	1	1	0	0
010	1	1	1	1	1
011	1	1	1	1	1
100	1	0	0	0	0
101	1	1	1	1	1
110	1	1	0	0	0
111	1	1	1	1	1

2.22 $(X+0)^D = X \cdot 1,\ X^D = X$

$[(X+Y')Y]^D = XY' + Y,\ (XY)^D = X + Y$

Unit 3 Problem Solutions

3.6 (a) $(W + X' + Z')\ (W' + Y')\ (W' + X + Z')\ (W + X')\ (W + Y + Z)$

$= (W + X')\ (W' + Y')\ (W' + X + Z')\ (W + Y + Z)$

$= (W + X')\ [W' + Y'\ (X + Z')]\ (W + Y + Z)$

$= [W + X'\ (Y + Z)]\ [W' + Y'(X + Z')] = WY'\ (X + Z') + W'X'\ (Y + Z)$ {Using $(X + Y)\ (X' + Z) = X'Y + XZ$ with $X = W$}

$= WY'X + WY'Z' + W'X'Y + W'X'Z$

3.6 (b) $(A + B + C + D)\ (A' + B' + C + D')\ (A' + C)\ (A + D)\ (B + C + D)$

$= (B + C + D)\ (A' + C)\ (A + D) = (B + C + D)\ (A'D + AC)$ {Using $(X + Y)\ (X' + Z) = X'Y + XZ$ with $X = A$}

$= A'DB + A'DC + A'D + ABC + AC + ACD = A'D + AC$

3.7 (a) $BCD + C'D' + B'C'D + CD$

$= CD + C'(D' + B'D) = (C' + D)\ [C + (D' + B'D)]$ {Using $(X + Y)\ (X' + Z) = X'Y + XZ$ with $X = C$}

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

3.7 (b) $A'C'D' + ABD' + A'CD + B'D$

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

$= D'\ [(A' + B)\ (A + C')] + D\ [(B' + A')\ (B' + C)]$ {Using $XY + X'Z = (X' + Y)\ (X + Z)$ twice inside the brackets}

$= [D + (A' + B)\ (A + C')]\ [D' + (B' + A')\ (B' + C)]$ {Using $XY + X'Z = (X' + Y)\ (X + Z)$ with $X = D$}

$= (D + A' + B)\ (D + A + C')\ (D' + B' + A')\ (D' + B' + C)$ {Using the Distributive Law}

3.8

$F = AB \oplus [(A \equiv D) + D] = AB \oplus (AB + A'D' + D) = AB \oplus (A'D' + D) = AB \oplus (A' + D)$

$= (AB)'\ (A' + D) + AB(A' + D)' = (A' + B')\ (A' + D) + AB(AD')$

$= A' + B'D + ABD'$ {Using $(X + Y)\ (X + Z) = X + YZ$} $= A' + BD' + B'D$ {Using $X + X'Y = X + Y$}

3.9 $A \oplus BC = (A \oplus B)\ (A \oplus C)$ is not a valid distributive law. PROOF: Let $A = 1, B = 1, C = 0$.

LHS: $A \oplus BC = 1 \oplus 1 \cdot 0 = 1 \oplus 0 = 1$. RHS: $(A \oplus B)\ (A \oplus C) = (1 \oplus 1)\ (1 \oplus 0) = 0 \cdot 1 = 0$.

3.10 (a) $(X + W)(Y \oplus Z) + XW'$
$= (X + W)(YZ' + Y'Z) + XW'$

$= XYZ' + XY'Z + WYZ' + WY'Z + XW'$

Using Consensus Theorem
$WYZ' + WY'Z + XW'$

3.10 (b) $(A \oplus BC) + BD + ACD = A'BC + A(BC)' + BD + ACD$
$= A'BC + A(B' + C') + BD + ACD$
$= A'BC + AB' + AC' + BD + ACD$

$= A'BC + AB' + AC' + AD + BD + ACD$
(Add consensus term AD, eliminate ACD)
$= A'BC + AB' + AC' + BD$
(Remove consensus term AD)

3.10 (c) $(A' + C' + D')(A' + B + C')(A + B + D)(A + C + D)$

$= (A' + C' + D')(B + C' + D')(A' + B + C')(A + B + D)(A + C + D)$ Add consensus term

$= (A' + B + C')(A + B + D)$
$= (A' + C' + D')(B + C' + D')(A + C + D)$ Removing consensus terms

3.11 $(A + B' + C + E')(A + B' + D' + E)(B' + C' + D' + E') = [A + B' + (C + E')(D' + E)](B' + C' + D' + E')$
$= (A + B' + D'E' + CE)(B' + C' + D' + E') = B' + (A + D'E' + CE)(C' + D' + E')$

CD' {Add consensus term}

$= B' + AC' + AD' + AE' + CD'E' + D'E' + D'E' + CD'E = B' + AC' + AD' + AE' + CD' + CD'E + D'E'$

$= B' + AC' + AE' + CD' + D'E'$

3.12 $A'CD'E + A'B'D' + ABCE + ABD = A'B'D' + ABD + BCD'E$

Proof: *LHS:* $A'CD'E + BCD'E + A'B'D' + ABCE + ABD$ Add consensus term to left-hand side and use it to eliminate two consensus terms

$= BCD'E + A'B'D' + ABD$ This yields the right-hand side.

$\therefore$ LHS = RHS

3.13 (a) $(A' + C' + D)(A' + C)(B + C' + D')(A' + B + C)(C + D)$

$= (C' + DB + A'D')(C + A'D) = C(BD + A'D') + (C'A'D)$ {Using $XY + X'Z = (X + Z)(X' + Y)$ with $X = C$}
$= CBD + CA'D' + C'A'D$

3.13 (b) $(A' + B' + C')(A + C + D')(A + B)(A' + D)(A' + C + D)$

$= [A' + D(B' + C')][A + B(C + D')] = AD(B' + C') + A'B(C + D') = ADB' + ADC' + A'BC + A'BD'$

3.13 (c) $(A' + B' + C)(A' + D')(A' + B + D')(A + B)(A + C + D')$

$= [A' + (B' + C)(B + D')](A + BD') = (A' + BC + B'D')(A + BD')$ {By Th. 14 with $X = B$}
$= A(BC + B'D') + A'BD'$ {By Th. 14 with $X = A$}
$= ABC + AB'D' + A'BD'$

3.13 (d) $(A + B + C)(A' + B' + D')(A' + B' + C')(A + B + D) = (A + B + CD)(A' + B' + C'D')$
$= A(B' + C'D') + A'(B + CD)$ {By Th. 14 with $X = A$} $= AB' + AC'D' + A'B + A'CD$

3.13 (e) $(A + B + C)(A + C + D)(A' + B' + C')(A' + C' + D') = (A + C + BD)(A' + C' + B'D')$
$= A(C' + B'D') + A'(C + BD) = AC' + AB'D' + A'C + A'BD$
Alt. soln's: $AC' + A'C + B'CD' + BC'D$ (or) $AC' + A'C + A'BD + B'CD'$ (or) $AC' + A'C + AB'D' + BC'D$

3.14 (a) $ABCD' + A'B'CD + CD' = A'B'CD + CD' = C(A'B'D + D') = C(D' + A'B')$ {By Th. 11D with $Y = D'$} $= CD' + A'B'C$

3.14 (b) $AB'C' + C\underline{D'} + BC'\underline{D'} = AB'C' + D'(\underline{C} + B\underline{C'}) = AB'C' + D'(C + B) = AB'C' + CD' + BD'$

3.14 (c) $(A + \underline{B'})\ (A' + \underline{B'} + D)\ (\underline{B'} + C + D') = B' + \underline{A}\ (\underline{A'} + D)\ (C + D') = B' + A\underline{D}\ (C + \underline{D'}) = B' + ACD$

3.14 (d) $(\underline{A'} + B + \underline{C'} + D)\ (\underline{A' + C'} + D + E)\ (\underline{A' + C'} + D + E')\ AC$
$= [A' + C' + (B + D)\ (\underline{D + E})\ (\underline{D + E'})]\ AC$ {By Th. 8D twice with $X = A' + C'$} $= [A' + C' + (B + D)D]\ AC$
$= [\underline{A'} + \underline{C'} + D]\ \underline{AC} = ACD$

3.15 (a) $A'B'\underline{C} + A\underline{C'}D + AB\underline{C} + B\underline{C'}D' = C'\ (AD + BD') + C\ (A'B' + AB)$
$= C'\ [(A + D')\ (B + D)] + C\ [(A' + B)\ (A + B')]$ {By Th. 14 twice with $X = D$ and $X = B$}
$= [C + (A + D')\ (B + D)\ ]\ [C' + (A' + B)\ (A + B')]$ {By Th. 14 with $X = C$}
$= (C + A + D')\ (C + B + D)\ (C' + A' + B)\ (C' + A + B')$ {By Distributive Law}

3.15 (b) $A\underline{B} + A'\underline{B'} + \underline{B}C'D' + \underline{B}CD' = B'\ (A' + C'D') + B\ (A + CD')$
$= (B + A' + C'D')\ (B' + A + CD')$ {By Th. 14 with $X = B$}
$= (B + A' + C')\ (B + A' + D')\ (B' + A + C)\ (B' + A + D')$

3.15 (c) $A\underline{B} + A'\underline{B'}C + \underline{B}C'D + \underline{B}C'D' = B'\ [A'\underline{C} + \underline{C'}D] + B\ [\underline{A + C'D'}] = \underline{B'}\ [(C + D)\ (C' + A')] + \underline{B}\ [(A + C')\ (A + D')]$
$= [B + (C + D)\ (C' + A')][B' + (A + C')\ (A + D')] = (B + C + D)\ (B + C' + A')\ (B' + A + C')\ (B' + A + D')$

3.15 (d) $A'C'\underline{D} + AB'\underline{D'} + A'C\underline{D'} + B\underline{D} = D\ (\underline{A'C' + B}) + D'\ (\underline{A}B' + \underline{A'}C) = \underline{D}\ (B + A')\ (B + C') + \underline{D'}\ (B' + A')\ (A + C)$
$= [D' + (B + A')\ (B + C')]\ [D + (B' + A')\ (A + C)] = (D' + B + A')\ (D' + B + C')\ (D + B' + A')\ (D + A + C)$

3.15 (e) $\underline{WXY} + \underline{WX'Y} + \underline{WYZ} + XYZ' = WY\ (\underline{X} + \underline{X'} + Z) + XYZ' = W\underline{Y} + X\underline{YZ'} = Y\ (W + XZ') = Y\ (W + X)\ (W + Z')$

3.16 (a) $(\underline{AB \oplus C}) + C'D' = \underline{(AB)'}C + ABC' + C'D' = (A' + B')\ C + AB\underline{C'} + \underline{C'}D' = \underline{C}\ (A' + B') + \underline{C'}\ (AB + D')$
$= (C' + A' + B')\ (\ C + D' + AB) = (C' + A' + B')\ (C + D' + A)\ (C + D' + B)$

3.16 (b) $C'\ (\underline{A \oplus D'}) + CD + A'D = C'\ [A'D' + AD] + CD + A'D = A'C'D' + AC'\underline{D} + C\underline{D} + A'\underline{D} = A'C'D' + D\ (C + \underline{A}C' + \underline{A'})$
$= A'C'D' + D\ (\underline{C} + A' + \underline{C'}) = A'C'\underline{D'} + \underline{D} = D + A'C' = (A' + D)\ (C' + D)$

3.17 (a) $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$ Proof:
LHS: Let $X \oplus Y = A$.
$A \oplus Z = AZ' + A'Z = (X \oplus Y)\ Z' + (X \oplus Y)'\ Z = (X \oplus Y\ )\ Z' + (X \equiv Y)\ Z$ {By (3-18) on FLD p. 61)
$= (X'Y + XY')\ Z' + (XY + X'Y')\ Z$
$= X'YZ' + XY'Z' + XYZ + X'Y'Z$
RHS: Let $Y \oplus Z = B$.
$X \oplus B = XB' + X'B = X\ (Y \oplus Z)' + X'\ (Y \oplus Z) = X\ (Y \equiv Z) + X'\ (Y \oplus Z) = X\ [YZ + Y'Z'] + X'\ [YZ' + Y'Z]$
$= XYZ + XY'Z' + X'YZ' + X'Y'Z$ $\therefore$ LHS = RHS

3.17 (b) $(X \equiv Y) \equiv Z = X \equiv (Y \equiv Z)$ Proof:
LHS: Let $X \equiv Y = A$.
$(A \equiv Z) = AZ + A'Z' = (X \equiv Y)\ Z + (X \equiv Y)'\ Z' = (X \equiv Y\ )\ Z + (X \oplus Y)\ Z' = (XY + X'Y')\ Z + (XY' + X'Y)\ Z'$
$= XYZ + X'Y'Z + XY'Z' + X'YZ'$
RHS: Let $Y \equiv Z = B$.
$(X \equiv B) = XB + X'B' = X\ (Y \equiv Z) + X'\ (Y \equiv Z)' = X\ (Y \equiv Z) + X'\ (Y \oplus Z) = X\ [YZ + Y'Z'] + X'\ [Y'Z + YZ']$
$= XYZ + XY'Z' + X'Y'Z + X'YZ'$ $\therefore$ LHS = RHS

3.18 (a) $BC'D' + \underline{ABC'} + \underline{AC'D} + \underline{AB'D} + A'BD' = BC'D' + \underline{ABC'} + AB'D + \underline{A'BD'} = ABC' + AB'D + A'BD'$

3.18 (b) $W'Y' + \underline{WYZ} + \underline{XY'Z} + WX'Y + \underline{WXZ} = \underline{W'Y'} + WYZ + \underline{XY'Z} + WX'Y + \underline{WXZ} = W'Y' + \underline{WYZ} + \underline{WX'Y} + \underline{WXZ}$
$= W'Y' + WX'Y + WXZ$

3.18 (c) $(B + C + D)\ (A + B + C)\ (A' + C + D)\ (B' + C' + D') = (A + B + C)\ (A' + C + D)\ (B' + C' + D')$

3.18 (d) $W'XY + WXZ + WY'Z + W'Z' = W'XY + WXZ + WY'Z + W'Z' + XYZ = WY'Z + W'Z' + XYZ$

XYZ (add consensus term)

3.18 (e) $A'BC' + BC'D' + A'CD + B'CD + A'BD = BC'D' + B'CD + A'BD$

3.18 (f) $(A + B + C)\ (B + C' + D)\ (A + B + D)\ (A' + B' + D') = (A + B + C)\ (B + C' + D)\ (A' + B' + D')$

3.19 $Z = ABC + DE + ACF + AD' + AB'E' = A\ (BC + CF + D' + B'E') + DE$

$= (A + DE)\ (DE + BC + CF + D' + B'E')$ {By Th. 8D with $X = DE$}

$= (A + D)\ (A + E)\ (BC + CF + D' + E + B'E')$

$= (A + D)\ (A + E)\ (D' + E + B' + BC + CF)$ {Since $E + B'E' = E + B'$}

$= (A + D)\ (A + E)\ (D' + E + B' + C + CF)$ {Since $B' + BC = B' + C$}

$= (A + D)\ (A + E)\ (D' + E + B' + C)$ {Since $C + CF = C$}

$= (A + DE)\ (D' + E + B' + C)$

$= AD' + AE + AB' + AC + DE + DEB' + DEC$ {eliminate consensus term AE; use $X + XY = X$ where $X = DE$}

$= AD' + AB' + AC + DE$

3.20 $F = A'B + AC + BC'D' + BEF + BDF = (A + B)\ (A' + C) + B\ (C'D' + EF + DF)$

$= [(A + B)\ (A' + C) + B]\ [(A + B)\ (A' + C) + C'D' + EF + DF]$

$= (A + B)\ (A' + C + B)\ (A + B + C'D' + EF + DF)\ (A' + C + C'D' + EF + DF)$

($B + C$; $C + D'$)

$= (A + B)\ (A' + C + B)\ (C + B)\ (A + B + C'D' + EF + DF)\ (A' + C + D' + EF + DF)$

$= (A + B)\ (B + C)\ (A' + C + D' + FE + DF) = (A + B)\ (B + C)\ (A' + C + D' + F + FE)$

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

$= (B + AC)\ (A' + C + D' + F)$

$= (A'B + BC + BD' + BF + AC + ACD' + ACF = A'B + BD' + BF + AC$

use consensus, $X + XY = X$ where $X = AC$

3.21 $X'Y'Z' + XYZ = (X + Y'Z')\ (X' + YZ) = (X + Y')\ (X + Z')\ (X' + Y)\ (X' + Z)\ (Y + Z')$

$= (X + Y')\ (X + Z')\ (X' + Y)\ (X' + Z)\ (Y + Z') = (X + Y')\ (X + Z')\ (X' + Z)\ (Y + Z')$

$= (X + Y')\ (X' + Z)\ (Y + Z')$

Alt.: $(X' + Y)\ (Y' + Z)\ (X + Z')$ by adding $(Y' + Z)$ as consensus in 3rd step

3.22 (a) $xy + x'yz' + yz = y\ (x + x'z') + yz = xy + yz' + yz$

$= xy + y = y$

Alternate Solution: $xy + x'yz' + yz = y\ (x + x'z' + z)$

$= y\ (x + z' + z) = y\ (x + 1) = y$

3.22 (b) $(xy' + z)\ (x + y')\ z = (xy' + xz + y'z)\ z$

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

Alternate Solution: $(xy' + z)\ (x + y')\ z = z\ (x + y')$

$= zx + zy'$

3.22 (c) $xy' + z + (x' + y)\ z'$

$= x'y + (x' + y)$ {By Th. 11D with $Y = z$}

$= xy' + x' + y = x + x' + y = 1 + y = 1$

Alt.: $xy' + z + (x' + y)\ z' = (xy' + z) + (xy' + z)' = 1$

3.22 (d) $a'd\ (b' + c) + a'd'\ (b + c') + (b' + c)\ (b + c')$

$= a'b'd + a'cd + a'bd' + a'cd' + b'c' + bc$

$= a'b'd + a'bd' + b'c' + bc$

Other Solutions: $b'c' + bc + a'c'd' + a'b'd$

$b'c' + bc + a'c'd' + a'cd$

$b'c' + bc + a'bd' + a'cd$

3.22 (e) $w'x' + x'y' + yz + w'z' + x'z$ Add redundant term

$= w'x' + x'y' + yz + w'z' + x'z$

$= x'y' + yz + w'z' + x'z$ Remove redundant term

$= x'y' + yz + w'z'$

3.22 (f) $A'BCD + A'BC'D + B'EF + CDE'G + A'DEF + A'B'EF$

$= A'BD + B'EF + CDE'G + A'DEF$ (consensus)

$= A'BD + B'EF + CDE'G$

3.22 (g) $[(a' + d' + b'c)(b + d + ac')]' + b'c'd' + a'c'd$

$= ad(b + c') + b'd'(a' + c) + b'c'd' + a'c'd$

$= abd + ac'd + a'b'd' + b'cd' + b'c'd' + a'c'd$

$c'd \qquad b'd'$

$= abd + a'b'd' + b'd' + c'd = abd + b'd' + c'd$

3.23 (a) $A'C'D' + AC' + BCD + A'CD' + A'BC + AB'C'$

$= A'D' + AC' + BCD + A'BC$ consensus

$= A'D' + AC' + BCD$

3.23 (b) $A'B'C' + ABD + A'C + A'CD' + AC'D + AB'C'$

$= B'C' + ABD + A'C + AC'D$

$= B'C' + ABD + A'C$

3.24 $WXY' + (W'Y' \equiv X) + (Y \oplus WZ)$

$= WXY' + W'Y'X + (W'Y')'X' + Y(WZ)' + Y'WZ$

$= WXY' + W'XY' + (W + Y)X' + Y(W' + Z') + Y'WZ$

$= XY' + WX' + X'Y + W'Y + YZ' + WY'Z + WY'$

$= XY' + WX' + YY' + W'Y + YZ' + WY'Z + WY'$

$= XY' + WX' + W'Y + YZ' + WY'$

$= X + WX' + W'Y + YZ'$

Alternate Solutions: $F = W'Y + WX' + WZ' + XY'$

$F = YZ' + W'X + XY' + WY'$

$F = W'X + X'Y + XZ' + WY'$

$F = W'X + XY' + WZ' + WY'$

3.25 (a) VALID: $a'b + b'c + c'a$

$= a'b(c + c') + (a + a')b'c + (b + b')ac'$

$= a'bc + a'bc' + ab'c + a'b'c + abc' + ab'c'$

$= a'c + \quad bc' + \quad ab'$

Alternate Solution: $a'b + b'c + c'a$

Add all consensus terms: ab', bc', ca'

$\therefore$ We get $= a'b + b'c + c'a + ab' + bc' + ca'$

$= ab' + bc' + ca'$

3.25 (b) NOT VALID. Counterexample: a = 0, b = 1, c = 0. LHS = 0, RHS = 1. $\therefore$ This equation is *not* always valid.

In fact, the two sides of the equation are complements: $[(a + b)(b + c)(c + a)]'$

$= [(b + ac)(a + c)]' = [ab + ac + bc]'$

$= (a' + b')(a' + c')(b' + c')$

3.25 (c) VALID. Starting with the right side, add consensus terms

$RHS = abc + ab'c' + b'cd + bc'd + acd + ac'd$

$= abc + ab'c' + b'cd + bc'd + acd + ac'd$

$= abc + ab'c' + b'cd + bc'd + ad = LHS$

3.25 (d) VALID: LHS $= xy' + x'z + yz'$

consensus terms: $y'z, xz', x'y$

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

$= y'z + xz' + x'y =$ RHS

3.25 (e) NOT VALID. Counterexample: x = 0, y = 1, z = 0, then LHS = 0, RHS = 1. $\therefore$ This equation is *not* always valid. In fact, the two sides of the equations are complements.

LHS $= (x + y)(y + z)(x + z)$

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

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

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

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

$\neq (x' + y')(y' + z')(x' + z')$

3.25 (f) VALID: LHS = $abc' + ab'c + b'c'd + bcd$

consensus terms: $ab'd$, abd

$= abc' + ab'c + b'c'd + bcd + ab'd + abd$

ad

$abc' + ab'c + ad + bcd + b'c'd$ = RHS

3.26 (a) VALID:

LHS = $(X' + Y')(X \equiv Z) + (X + Y)(X \oplus Z)$

$= (X' + Y')(X'Z' + XZ) + (X + Y)(X'Z + XZ')$

$= X'Z' + X'YZ' + XY'Z + X'YZ + XZ' + XYZ$

$= X'Z' + (XY' + X'Y)Z + XZ'$

$= Z' + Z(X \oplus Y) = Z' + (X \oplus Y)$ = RHS

3.26 (b) LHS = $(W' + X + Y')(W + X' + Y)(W + Y' + Z) = (W' + X + Y')(W + (X' + Y)(Y' + Z))$

$= (W' + X + Y')(W + (X'Y' + YZ)) = (W'(X'Y' + YZ) + W(X + Y')) = W'X'Y' + W'YZ + WX + WY'$

consensus terms: $X'Y'$, XYZ

$= W'X'Y' + W'YZ + WX + WY' + XYZ + X'Y' = W'X'Y' + W'X'Z + W'YZ + XYZ + WX + WY' + X'Y'$

$= W'X'Z + W'YZ + XYZ + WX + X'Y' = W'YZ + XYZ + WX + X'Y'$

3.26 (c) LHS = $ABC + A'C'D' + A'BD' + ACD = AC(B + D) + A'D'(B + C') = (A + D'(B + C'))(A' + C(B + D))$

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

consensus: $B + C' + D$

$= (A + D')(A + B + C')(A' + C)(B + C' + D) = (A + D')(A' + C)(B + C' + D)$ = RHS

3.27 (a) VALID. $[A + B = C] \Rightarrow [D'(A + B) = D'(C)]$

$[A + B = C] \Rightarrow [AD' + BD' = CD']$

3.27 (b) NOT VALID. Counterexample: $A = 1, B = C = 0$ and $D = 1$ then LHS $= 0 \cdot 0 + 0 \cdot 0 = 0$

RHS $= 0 \cdot 1 = 0 =$ LHS

but $B + C = 0 + 0 = 0; D = 1 \neq B + C$

$\therefore$ The statement is false.

3.27 (c) VALID. $[A + B = C] \Rightarrow [(A + B) + D = (C) + D]$

$[A + B = C] \Rightarrow [A + B + D = C + D]$

3.27 (d) NOT VALID. Counterexample: $C = 1, A = B = 0$ and $D = 1$ then LHS $= 0 + 0 + 1 = 1$

RHS $= 1 + 1 = 1 =$ LHS

but $A + B = 0 + 0 = 0 \neq D$

$\therefore$ The statement is false.

3.28 (a) $A'C' + BC + AB' + A'BD + B'C'D' + ACD'$

Consensus terms: (1) $B'C'$ using $A'C' + AB'$

(2) $A'B$ using $A'C' + BC$ (3) AC using $AB' + BC$

(4) $AB'D'$ using $B'C'D' + ACD'$

Using 1, 2, 3: $A'C' + BC + AB' + A'BD + B'C'D' + ACD' + B'C' + A'B + AC = A'C' + BC + AB'$

(Using the consensus theorem to remove the added terms since the terms that generated them are still present.)

3.28 (b) $A'C'D' + BC'D + AB'C' + A'BC$

Consensus terms: (1) $A'BC'$ using $A'C'D' + BC'D$

(2) $AC'D$ using $AB'C' + BC'D$ (3) $B'C'D'$ using $A'C'D' + AB'C'$ (4) $A'BD'$ using $A'C'D' + A'BC$ (5) $A'BD$ using $BC'D + A'BC$

Using 1: $A'C'D' + BC'D + AB'C' + A'BC + A'B$, which is the minimum solution.

Unit 4 Problem Solutions

4.1 *See FLD p. 628 for solution.*

4.2

A B C D E		*y*	*z*
0 0 0 0 0	(less than 10 gpm)	+	
1 0 0 0 0	(at least 10 gpm)	+	
1 1 0 0 0	(at least 20 gpm)	+	+
1 1 1 0 0	(at least 30 gpm)		+
1 1 1 1 0	(at least 40 gpm)		+
1 1 1 1 1	(at least 50 gpm)		

4.2 (a) $Y = A'B'C'D'E' + AB'C'D'E' + ABC'D'E'$

4.2 (b) $Z = ABC'D'E' + ABCD'E' + ABCDE'$

4.3 $F_1 = \sum m(0, 4, 5, 6); F_2 = \sum m(0, 3, 4, 6, 7); F_1 + F_2 = \sum m(0, 3, 4, 5, 6, 7)$
General rule: $F_1 + F_2$ is the sum of all minterms that are present in either F_1 or F_2.

Proof: Let $F_1 = \sum_{i=0}^{2^n-1} a_i m_i; F_2 = \sum_{j=0}^{2^n-1} b_j m_j;\ F_1 + F_2 = \sum_{i=0}^{2^n-1} a_i m_i + \sum_{j=0}^{2^n-1} b_j m_j = a_0 m_0 + a_1 m_1 + a_2 m_2 + \ldots$

$+ b_0 m_0 + b_1 m_1 + b_2 m_2 + \ldots = (a_0 + b_0)\, m_0 + (a_1 + b_1)\, m_1 + (a_2 + b_2)\, m_2 + \ldots = \sum_{i=0}^{2^n-1} (a_i + b_i)\, m_i$

4.4 (a) $2^{2^n} = 2^{2^2} = 2^4 = 16$

4.4 (b)

x y	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	z_9	z_{10}	z_{11}	z_{12}	z_{13}	z_{14}	z_{15}
0 0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0 1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1 0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1 1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	0	$x'y'$	$x'y$	x'	xy'	y'	$x'y+xy'$	$x'+y'$	xy	$x'y'+xy$	y	$x'+y$	x	$x+y'$	$x+y$	1

4.5

A B C	*D E F*	*Z*
0 0 0	1 1 X[3]	1
0 0 1	X[2] X[2] 1	1
0 1 0	X[1] X[1] X[1]	X
0 1 1	X[2] X[2] 1	1
1 0 0	X[4] 0 0	0
1 0 1	X[2] X[2] 1	1
1 1 0	X[1] X[1] X[1]	X
1 1 1	X[4] 0 0	0

Alternate Solutions

A B C	*D E F*	*Z*
0 1 1	1 1 X[3]	1
1 1 1	0 X[4] 0	0

[1] These truth table entries were made don't cares because $ABC = 110$ and $ABC = 010$ can never occur

[2] These truth table entries were made don't cares because when F is 1, the output Z of the OR gate will be 1 regardless of its other input. So changing D and E cannot affect Z.

[3] These truth table entries were made don't cares because when D and E are both 1, the output Z of the OR gate will be 1 regardless of the value of F.

[4] These truth table entries were made don't cares because when one input of the AND gate is 0, the output will be 0 regardless of the value of its other input.

4.6 (a) Of the four possible combinations of d_1 & d_5, $d_1 = 1$ and $d_5 = 0$ gives the best solution:
$F = A'B'C' + A'B'C + ABC' + ABC = A'B' + AB$

4.6 (b) By inspection, $G = C$ when both don't cares are set to 0.

4.7 (a) Exactly one variable not complemented: $F = A'B'C + A'BC' + AB'C' = \sum m(1, 2, 4)$

4.7 (b) Remaining terms are maxterms:
$F = \prod M(0, 3, 5, 6, 7) = (A + B + C)\ (A + B' + C')\ (A' + B + C')\ (A' + B' + C)\ (A' + B' + C')$

4.8

$ABCD$		F
0 0 0 0	$0 \times 0 = 0 \leq 2$	1
0 0 0 1	$0 \times 1 = 0 \leq 2$	1
0 0 1 0	$0 \times 2 = 0 \leq 2$	1
0 0 1 1	$0 \times 3 = 0 \leq 2$	1
0 1 0 0	$1 \times 0 = 0 \leq 2$	1
0 1 0 1	$1 \times 1 = 1 \leq 2$	1
0 1 1 0	$1 \times 2 = 2 \leq 2$	1
0 1 1 1	$1 \times 3 = 3 > 2$	0
1 0 0 0	$2 \times 0 = 0 \leq 2$	1
1 0 0 1	$2 \times 1 = 2 \leq 2$	1
1 0 1 0	$2 \times 2 = 4 > 2$	0
1 0 1 1	$2 \times 3 = 6 > 2$	0
1 1 0 0	$3 \times 0 = 0 \leq 2$	1
1 1 0 1	$3 \times 1 = 3 > 2$	0
1 1 1 0	$3 \times 2 = 6 > 2$	0
1 1 1 1	$3 \times 3 = 9 > 2$	0

4.8 (a) $F(A, B, C, D) = \sum m(0, 1, 2, 3, 4, 5, 6, 8, 9, 12)$
Refer to FLD for full term expansion

4.8 (b) $F(A, B, C, D) = \prod M(7, 10, 11, 13, 14, 15)$
Refer to FLD for full term expansion

4.9 (a) $F = abc' + b'(a + a')(c + c') = abc' + ab'c + ab'c' + a'b'c + a'b'c'$; $F = \sum m(0, 1, 4, 5, 6)$

4.9 (b) Remaining terms are maxterms: $F = \prod M(2, 3, 7)$

4.9 (c) Maxterms of F are minterms of F':
$F' = \sum m(2, 3, 7)$

4.9 (d) Minterms of F are maxterms of F':
$F' = \prod M(0, 1, 4, 5, 6)$

4.10

$$F(a, b, c, d) = \underline{(a + b + d)}\ \underline{(a' + c)}\ \underline{(a' + b' + c')}\ (a + b + c' + d')$$

$$= (a + b + c + d)\ (a + b + c' + d)\ (a' + c + bb' + dd')\ (a' + b' + c' + d)\ (a' + b' + c' + d')\ (a + b + c' + d')$$
$$= (a + b + c + d)\ (a + b + c' + d)\ (a' + b + c + d)\ (a' + b + c + d')\ (a' + b' + c + d)\ (a' + b' + c + d')$$
$$(a' + b' + c' + d)\ (a' + b' + c' + d')\ (a + b + c' + d')$$

4.10 (a) $F = \sum m(1, 4, 5, 6, 7, 10, 11)$

4.10 (b) $F = \prod M(0, 2, 3, 8, 9, 12, 13, 14, 15)$

4.10 (c) $F' = \sum m(0, 2, 3, 8, 9, 12, 13, 14, 15)$

4.10 (d) $F' = \prod M(1, 4, 5, 6, 7, 10, 11)$

4.11 (a) difference, $d_i = x_i \oplus y_i \oplus b_i$; $b_{i+1} = b_i x_i' + x_i' y_i + b_i y_i$

$x_i y_i b_i$	b_{i+1} d_i
0 0 0	0 0
0 0 1	1 1
0 1 0	1 1
0 1 1	1 0
1 0 0	0 1
1 0 1	0 0
1 1 0	0 0
1 1 1	1 1

4.11 (b) $d_i = s_i$; b_{i+1} is the same as c_{i+1} with x_i replaced by x_i'

4.12 *See FLD p. 629 for solution.*

4.13

A B C D	Z
0 0 0 0	1
0 0 0 1	1
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	1
1 0 0 1	0
1 0 1 0	0
1 0 1 1	0
1 1 0 0	0
1 1 0 1	0
1 1 1 0	1
1 1 1 1	1

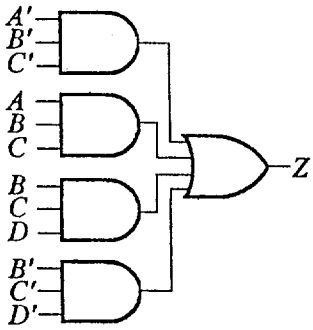
$Z = A'B'C'D' + A'B'C'D + AB'C'D' + ABCD' + ABCD + A'BCD$

$= A'B'C' + ABC + AB'C'D' + A'BCD$

$= A'B'C' + ABC + AB'C'D' + A'BCD + \underline{BCD} + \underline{B'C'D'}$

(Added consensus terms)

$\therefore Z = A'B'C' + ABC + BCD + B'C'D'$



4.14

A B C D	Z
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	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	0
1 1 1 0	0
1 1 1 1	0

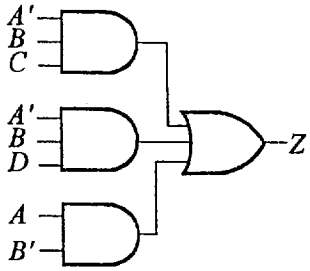
$Z = A'BC'D + A'BCD' + A'BCD + AB'C'D' + AB'C'D + AB'CD' + AB'CD$

$= A'BD + AB'C' + AB'C + A'BCD'$

$= AB' + A'BD + A'BCD' + \underline{A'BC}$

(Added consensus terms)

$\therefore Z = AB' + A'BD + A'BC$



4.15 (a) The $\underbrace{\text{buzzer will sound}}_{B}$ if $\underbrace{\text{the key is in the ignition switch}}_{K}$ and $\underbrace{\text{the car door is open}}_{D}$, or $\underbrace{\text{the seat belts are not fastened}}_{S'}$.

$\therefore$ The two possible interpretations are: $B = KD + S'$ and $B = K(D + S')$

4.15 (b) You $\underbrace{\text{will gain weight}}_{W}$ if you $\underbrace{\text{eat too much}}_{F}$, or you do $\underbrace{\text{not exercise enough}}_{E'}$ and $\underbrace{\text{your metabolism rate is too low}}_{M}$.

$\therefore$ The two possible interpretations are: $W = (F + E')M$ and $W = F + E'M$

4.15 (c) The $\underbrace{\text{speaker will be damaged}}_{D}$ if the $\underbrace{\text{volume is set too high}}_{V}$ and $\underbrace{\text{loud music is played}}_{M}$ or the $\underbrace{\text{stereo is too powerful}}_{S}$.

$\therefore$ The two possible interpretations are: $D = VM + S$ and $D = V(M + S)$

4.15 (d) The $\underbrace{\text{roads will be very slippery}}_{V}$ if $\underbrace{\text{it snows}}_{S}$ or $\underbrace{\text{it rains}}_{R}$ and $\underbrace{\text{there is oil on the road}}_{O}$.

$\therefore$ The two possible interpretations are: $V = (S + R)O$ and $V = S + RO$

4.16 $Z = AB + AC + BC$

4.17 $Z = (ABCDE + A'B'C'D'E')'$; $Y = A'B'CD'E$

4.18 (a) $13_{10} = D_{16} = 0001101$; $\therefore X = A'B'C'DEF'G$

4.18 (b) $10_{10} = 0001010$; $\therefore Y = A'B'C'DE'FG'$

4.18 (c) $0_{10} = 0000000_2$; $64_{10} = 1000000_2$; $31_{10} = 0011111_2$; $127_{10} = 1111111_2$; $32_{10} = 0100000_2$; $\therefore Z = (A'B')' = A + B$

4.19 $F_1F_2 = \prod M(0, 3, 4, 5, 6, 7)$. General rule: F_1F_2 is the product of all maxterms that are present in either F_1 or F_2. Proof:

$$\text{Let } F_1 = \prod_{i=0}^{2^n-1}(a_i + M_i);\ F_2 = \prod_{j=0}^{2^n-1}(b_j + M_j);\ F_1F_2 = \prod_{i=0}^{2^n-1}(a_i + M_i)\prod_{j=0}^{2^n-1}(b_j + M_j)$$

$$= (a_0 + M_0)(b_0 + M_0)(a_1 + M_1)(b_1 + M_1)(a_2 + M_2)(b_2 + M_2)\ldots = (a_0b_0 + M_0)(a_1b_1 + M_1)(a_2b_2 + M_2)\ldots$$

$$= \prod_{i=0}^{3^n-1}(a_ib_i + M_i)$$

Maxterm M_i is present in F_1F_2 iff $a_ib_i = 0$. Maxterm M_i is present in F_1 iff $a_i = 0$. Maxterm M_i is present in F_2 iff $a_j = 0$. Therefore, maxterm M_i is present in F_1F_2 iff it is present in F_1 or F_2.

4.20

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

(a) $F(A, B, C, D) = \sum m(5,6,7,10,11,13,14,15) = \prod M(0, 1, 2, 3, 4, 8, 9, 12)$

(b) G (A, B, C, D) $= \sum m(0, 2, 4, 6) = \prod M(1,3,5,7,8,9,10,11, 12, 13, 14, 15)$

(c) $H(A, B, C, D) = \sum m(7, 11, 13, 14, 15) = \prod M(0,1,2,3,4,5,6,8,9, 10, 12)$

(d) $J(A, B, C, D) = \sum m(4, 8, 12, 13, 14) = \prod M(0,1,2,3,5,6,7,9, 10, 11, 15)$

4.22

a b c d	f
0 0 0 0	0
0 0 0 1	1
0 0 1 0	1
0 0 1 1	0
0 1 0 0	1
0 1 0 1	1
0 1 1 0	1
0 1 1 1	0
1 0 0 0	0
1 0 0 1	0
1 0 1 0	1
1 0 1 1	1
1 1 0 0	1
1 1 0 1	0
1 1 1 0	1
1 1 1 1	1

(a) $f = \sum m(1, 2, 4, 5, 6, 10, 11, 12, 14, 15)$

(b) $f = \prod M(0, 3, 7, 8, 9, 13)$

(c) $f' = \sum m(0, 3, 7, 8, 9, 13)$

(d) $f' = \prod M(1, 2, 4, 5, 6, 10, 11, 12, 14, 15)$

You can also work this problem algebraically, as in problem 4.21.

4.21 You can also work this problem using a truth table, as in problem 4.22.

$f(a, b, c) = a'(b + c') = a'b + a'c' = a'b(c + c') + a'(b + b')c' = \underset{m_3}{a'bc} + \underset{m_2}{a'bc'} + \underset{m_2}{a'bc'} + \underset{m_0}{a'b'c'}$

4.21 (a) $f = \sum m(0, 2, 3)$ **4.21 (b)** $f = \prod M(1, 4, 5, 6, 7)$

4.21 (c) $f' = \sum m(1, 4, 5, 6, 7)$ **4.21 (d)** $f' = \prod M(0, 2, 3)$

4.23 (a) $F(A, B, C, D) = \sum m(3, 4, 5, 8, 9, 10, 11, 12, 14)$

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

4.23 (b) F (A, B, C, D) $= \prod M(0, 1, 2, 6, 7, 13, 15)$

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

4.24 (a) $F(A, B, C, D) = \sum m(0, 3, 4, 7, 8, 9, 11, 12, 13, 14) = \underset{m_0}{A'B'C'D'} + \underset{m_3}{A'B'CD} + \underset{m_4}{A'BC'D'} + \underset{m_7}{A'BCD} + \underset{m_8}{AB'C'D'} + \underset{m_9}{AB'C'D} + \underset{m_{11}}{AB'CD} + \underset{m_{12}}{ABC'D'} + \underset{m_{13}}{ABC'D} + \underset{m_{14}}{ABCD'}$

4.24 (b) $F(A, B, C, D) = \prod M(1, 2, 5, 6, 10, 15) = \underset{M_1}{(A + B + C + D')}\ \underset{M_2}{(A + B + C' + D)}\ \underset{M_5}{(A + B' + C + D')}\ \underset{M_6}{(A + B' + C' + D)}\ \underset{M_{10}}{(A' + B + C' + D)}\ \underset{M_{15}}{(A' + B' + C' + D')}$

4.25 (a) If don't cares are changed to (1, 1), respectively,

$F_1 = A'B'C' + ABC + A'B'C + AB'C$

$= A'B' + AC,$

4.25 (b) If don't cares are changed to (1, 0), respectively

$F_2 = A'B'C' + A'BC' + AB'C' + ABC' = C'$

4.25 (c) If don't cares are changed to (1, 1), respectively

$F_3 = (A + B + C)(A + B + C') = A + B$

4.25 (d) If don't cares are changed to (0, 1), respectively

$F_4 = A'B'C' + A'BC + AB'C' + ABC$

$= B'C' + BC$

4.26

A B C	D	E	F	Z
0 0 0	1	1	X[2]	0
0 0 1	0	1	X[2]	1
0 1 0	0	X[2]	1	1
0 1 1	X[1]	X[1]	X[1]	X
1 0 0	0	1	X[2]	1
1 0 1	0	X[2]	1	1
1 1 0	X[1]	X[1]	X[1]	X
1 1 1	1	X[2]	1	0

[1] These truth table entries were made don't cares because $ABC = 110$ and $ABC = 011$ can never occur.

[2] These truth table entries were made don't cares because when one input of the OR gate is 1, the output will be 1 regardless of the value of its other input.

4.27 (a) $G_1(A, B, C) = \sum m(0, 7) = \prod M(1, 2, 3, 4, 5, 6)$

4.27 (b) $G_2(A, B, C) = \sum m(0, 1, 6, 7) = \prod M(2, 3, 4, 5)$

4.28

A B C D	1's	X Y Z
0 0 0 0	0	0 0 0
0 0 0 1	1	0 0 1
0 0 1 0	1	0 0 1
0 0 1 1	2	0 1 0
0 1 0 0	1	0 0 1
0 1 0 1	2	0 1 0
0 1 1 0	2	0 1 0
0 1 1 1	3	0 1 1
1 0 0 0	1	0 0 1
1 0 0 1	2	0 1 0
1 0 1 0	2	0 1 0
1 0 1 1	3	0 1 1
1 1 0 0	2	0 1 0
1 1 0 1	3	0 1 1
1 1 1 0	3	0 1 1
1 1 1 1	4	1 0 0

(a) $X = ABCD$

$Y = A'B'CD + A'BC'D + A'BCD' + A'BCD + AB'C'D + AB'CD' + AB'CD + ABC'D' + ABC'D + ABCD'$

$Z = A'B'C'D + A'B'CD' + A'BC'D' + A'BCD + AB'C'D' + AB'CD + ABC'D + ABCD'$

4.28 (b) $Y = (A + B + C + D)(A + B + C + D')(A + B + C' + D)(A + B' + C + D)(A' + B + C + D)(A' + B' + C' + D')$

$Z = (A + B + C + D)(A + B' + C + D')(A + B' + C' + D)(A' + B + C + D')(A' + B + C' + D)(A' + B' + C + D)(A' + B' + C' + D')$

4.29

A B C D	W X Y Z
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 0
0 0 1 1	0 1 0 1
0 1 0 0	0 1 0 0
0 1 0 1	0 1 0 1
0 1 1 0	0 1 0 1
0 1 1 1	0 1 1 0
1 0 0 0	0 1 0 0
1 0 0 1	0 1 0 1
1 0 1 0	0 1 0 1
1 0 1 1	0 1 1 0
1 1 0 0	0 1 0 1
1 1 0 1	0 1 1 0
1 1 1 0	0 1 1 0
1 1 1 1	0 1 1 1

(a) $X = A'B'C'D + A'B'CD' + A'B'CD + A'BC'D' + A'BC'D + A'BCD' + A'BCD + AB'C'D' + AB'C'D + AB'CD' + AB'CD + ABC'D' + ABC'D + ABCD' + ABCD$

$Y = A'B'C'D' + A'BCD + ABC'D + ABCD' + ABCD$

$Z = A'B'C'D' + A'B'CD + A'BC'D + A'BCD' + AB'C'D + AB'CD' + AB'CD + ABC'D' + ABCD$

4.29 (b) $Y = (A + B + C + D')(A + B + C' + D)(A + B + C' + D')(A + B' + C + D)(A + B' + C + D')(A + B' + C' + D)(A' + B + C + D)(A' + B + C + D')(A' + B + C' + D)(A' + B + C' + D')(A' + B' + C + D)$

$Z = (A + B + C + D')(A + B + C' + D)(A + B' + C + D)(A + B' + C' + D)(A' + B + C + D)(A' + B + C' + D)(A' + B' + C + D')(A' + B' + C' + D)$

4.30

A B C D		*S T U V*	*W X Y Z*
0 0 0 0	0 × 5 = 00	0 0 0 0	0 0 0 0
0 0 0 1	1 × 5 = 05	0 0 0 0	0 1 0 1
0 0 1 0	2 × 5 = 10	0 0 0 1	0 0 0 0
0 0 1 1	3 × 5 = 15	0 0 0 1	0 1 0 1
0 1 0 0	4 × 5 = 20	0 0 1 0	0 0 0 0
0 1 0 1	5 × 5 = 25	0 0 1 0	0 1 0 1
0 1 1 0	6 × 5 = 30	0 0 1 1	0 0 0 0
0 1 1 1	7 × 5 = 35	0 0 1 1	0 1 0 1
1 0 0 0	8 × 5 = 40	0 1 0 0	0 0 0 0
1 0 0 1	9 × 5 =45	0 1 0 0	0 1 0 1

Note: Rows 1010 through 1111 have don't care outputs.

$S = 0, T = A, U = B, V = C, W = 0, X = D, Y = 0, Z = D$

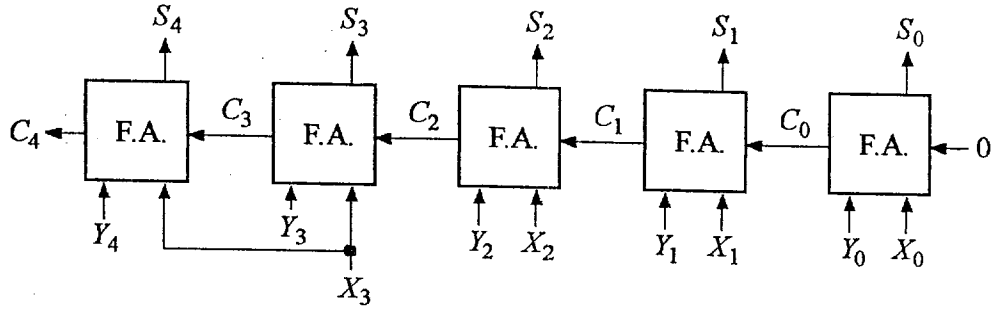
4.31

A B C D		*S T U V*	*W X Y Z*
0 0 0 0	0 × 4 + 1 = 01	0 0 0 0	0 0 0 1
0 0 0 1	1 × 4 + 1 = 05	0 0 0 0	0 1 0 1
0 0 1 0	2 × 4 + 1 = 09	0 0 0 0	1 0 0 1
0 0 1 1	3 × 4 + 1 = 13	0 0 0 1	0 0 1 1
0 1 0 0	4 × 4 + 1 = 17	0 0 0 1	0 1 1 1
0 1 0 1	5 × 4 + 1 = 21	0 0 1 0	0 0 0 1
0 1 1 0	6 × 4 + 1 = 25	0 0 1 0	0 1 0 1
0 1 1 1	7 × 4 + 1 = 29	0 0 1 0	1 0 0 1
1 0 0 0	8 × 4 + 1 = 33	0 0 1 1	0 0 1 1
1 0 0 1	9 × 4 + 1 =37	0 0 1 1	0 1 1 1

Note: Rows 1010 through 1111 have don't care outputs.

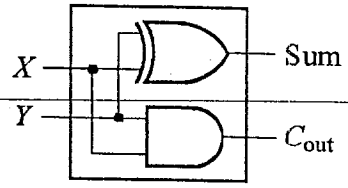
$S = 0, T = 0, U = BD + BC + A,$
$V = B'CD + BC'D' + A, W = B'CD' + BCD,$
$X = B'C'D + BD', Y = B'CD + BC'D' + A, Z = 1$

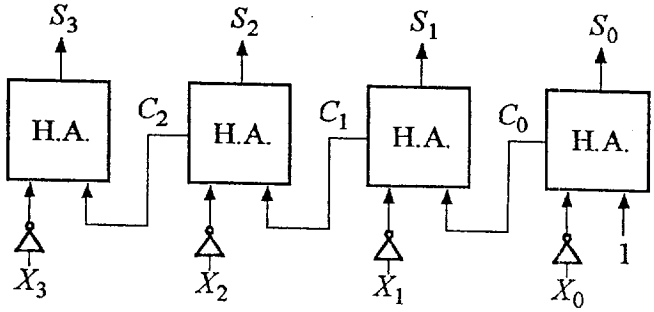
4.32 Notice that the sign bit X_3 of the 4-bit number is extended to the leftmost full adder as well.



4.33

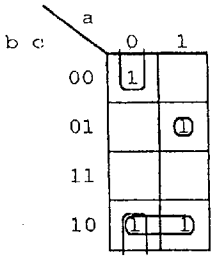
XY	Sum	Cout
0 0	0	0
0 1	1	0
1 0	1	0
1 1	0	1





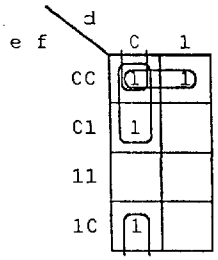
Unit 5 Problem Solutions

5.3 (a)



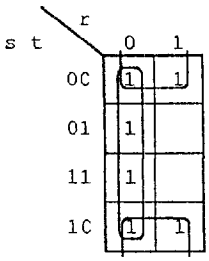
f1 = a'c' + a b'c + b c'

5.3 (b)



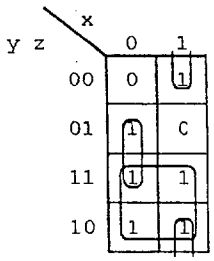
f2 = d'e' + d'f' + e'f'

5.3 (c)



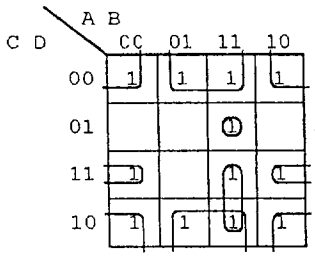
f3 = r' + t'

5.3 (d)



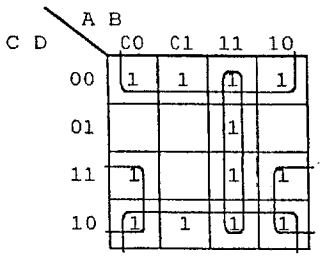
f4 = x'z + y + xz'

5.4 (a)



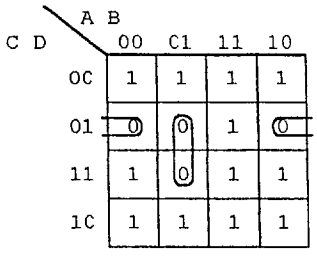
F = B D' + B'C D + A B C + A B C'D + B'D'

5.4 (b)



F = D' + B'C + A B

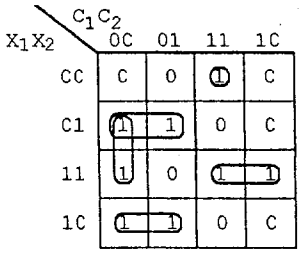
5.4 (c)



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

5.5 (a) *See FLD p. 630 for solution.*

5.5 (b)

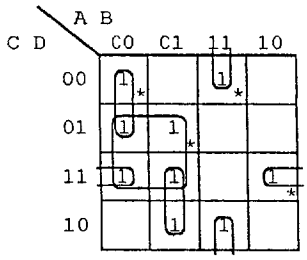


$Z = C_1'X_1'X_2 + C_1'X_1X_2' + C_1C_2X_1'X_2' + C_1X_1X_2 + C_1'C_2'X_2$

Alt: $Z = C_1'X_1'X_2 + C_1'X_1X_2' + C_1C_2X_1'X_2' + C_1X_1X_2 + C_1'C_2'X_1$

Alt: $Z = C_1'X_1'X_2 + C_1'X_1X_2' + C_1C_2X_1'X_2' + C_1X_1X_2 + C_2'X_1X_2$

5.6 (a)



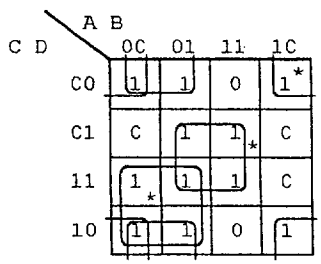
F = A'B'C' + A'D + B'C D + A B D' + B C D'

Alt: F = A'B'C' + A'D + B'C D + A B D' + A'B C

A (*) indicates a minterm that makes the corresponding prime implicant essential.

A'D $\rightarrow m_5$; A'B'C' $\rightarrow m_0$; B'CD $\rightarrow m_{11}$; ABD' $\rightarrow m_{12}$

5.6 (b)



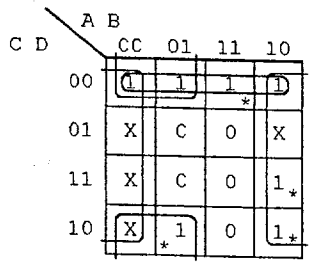
F = A'C + B'D' + B D + A'D'

Alt: F = A'C + B'D' + B D + A'B

(*) Indicates a minterm that makes the corresponding prime implicant essential.

BD $\rightarrow m_{13}$ or m_{15}; A'C $\rightarrow m_3$; B'D' $\rightarrow m_8$ or m_{10}

5.6 (c)

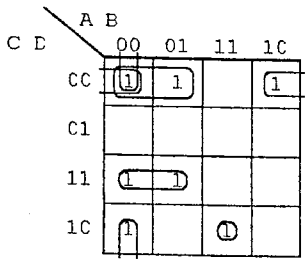


F = A'D' + B' + C'D'

(*) Indicates a minterm that makes the corresponding prime implicant essential.

C'D'→m_{12}; A'D'→m_6; B'→m_{10} or m_{11}

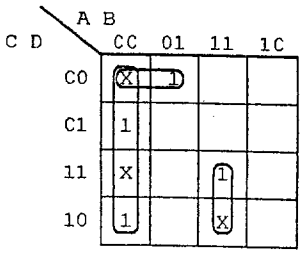
5.7 (a)



F = A'C'D' + A'C D + B'C'D' + A B C D' + A'B'D'

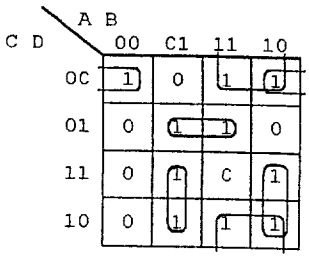
Alt: F = A'C'D' + A'C D + B'C'D' + A B C D' + A'B'C

5.7 (b)



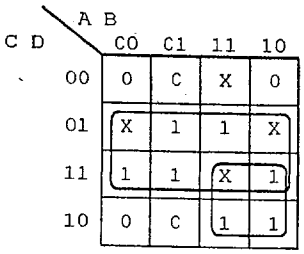
F = A'B' + A'C'D' + A B C

5.7 (c)



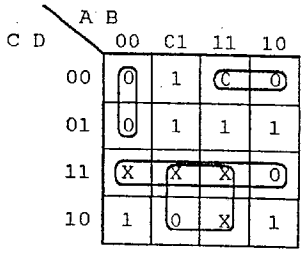
F = B'C'D' + AB'C + A'BC + BC'D + AD'

5.7 (d)

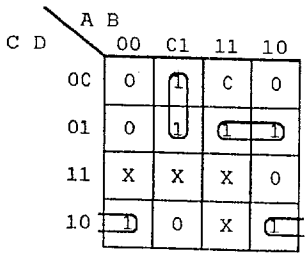


F = D + A C

5.8 (a)

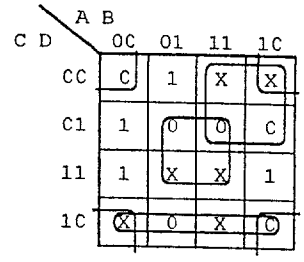


F = (C'+ D') (B'+ C') (A + B + C) (A'+ C + D)



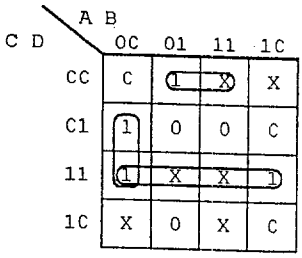
F = A'B C' + A C'D + B'C D'

5.8 (b)



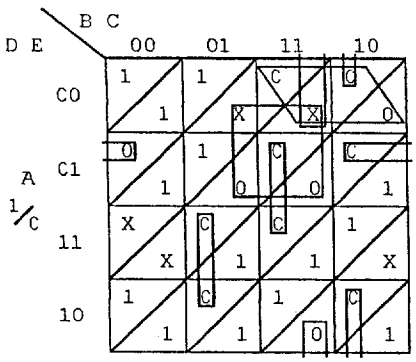
F = (A'+ C) (B'+ D') (B + D) (C'+ D)

Alt: F = (A'+ C) (B'+ D') (B + D) (B'+ C')

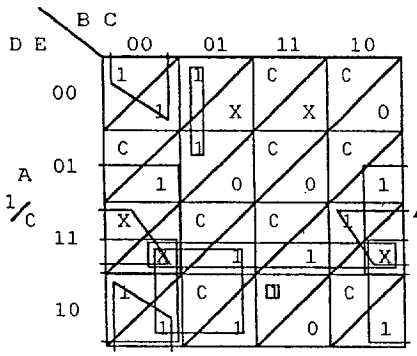


F = A'B'D + B C'D' + C D

5.9 (a)



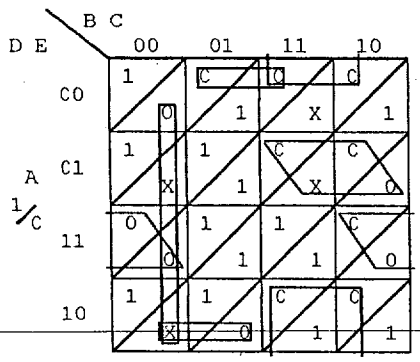
F = (A' + B' + C + E) (A' + B + C' + D') (A + B' + C' + E) (B' + D + E) (A + C' + D) (A' + C + D + E') (A' + B' + C' + E')



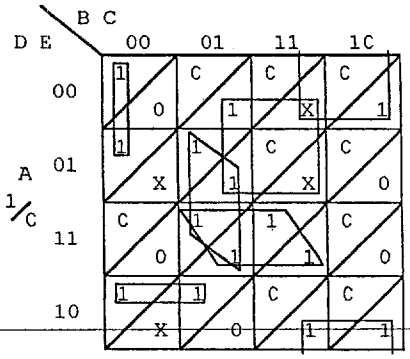
F = A'C'E + A'C'D + A'D E + A B'C D' + C'D E + A B C D E' + B'C'E' + A'B'D

Alt: F = A'C'E + A'C'D + A'D E + A B'C D' + C'D E + A B C D E' + B'C'E' + A'B'E'

5.9 (b)



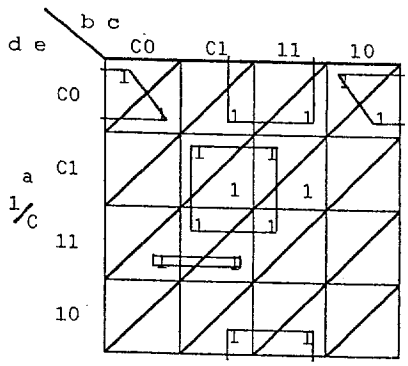
F = (A' + B' + E) (A' + C' + D + E) (C + D' + E') (A + B + D' + E) (A + B + C) (B' + D + E')



F = A'C D' + A'B E' + C D E + A B'C'D' + A B'D E' + B'C E

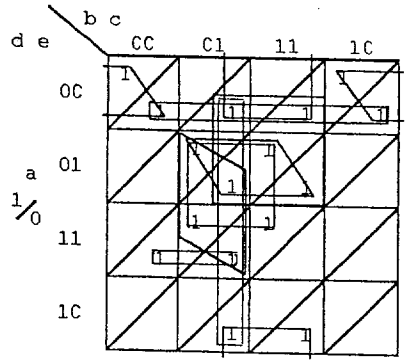
Alt:
F = A'C D' + A'B E' + C D E + A B'C'E' + A B'C D + B'D'E
F = A'C D' + A'B E' + C D E + A B'C'D' + A B'D E' + B'D'E
F = A'C D' + A'B E' + C D E + A B'C'E' + A B'D E' + B'D'E

5.10 (a)



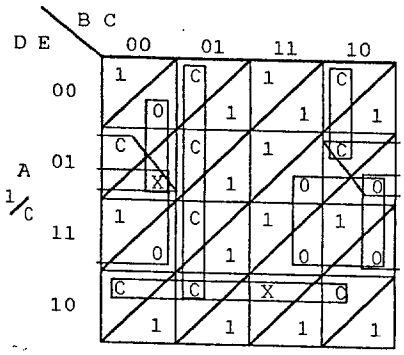
Essential prime implicants: $C'D'E'$ (m_{16}, m_{24}), $A'CE'$ (m_{14}), ACE (m_{31}), $A'B'DE$ (m_3)

5.10 (b)



Prime implicants: $A'B'DE$, $A'D'E'$, $CD'E$, $A'CE'$, ACE, $A'B'C$, $B'CE$, $C'D'E'$, $A'CD'$

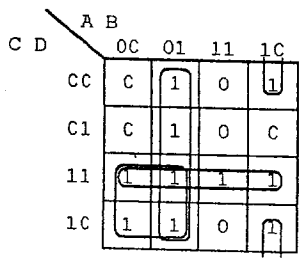
5.11



$$F = \frac{(A'+B+C')(A'+D'+E)(A+B'+E')(A+C+E')}{(A+B+C+D)(A'+B'+C+D)(C+D+E')}$$

Alt: $$F = \frac{(A'+B+C')(A'+D'+E)(A+B'+E')(A+C+E')}{(A+B+C+D)(A'+B'+C+E)(C+D+E')}$$

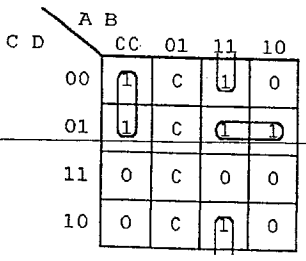
5.12 (a)



$F = A\,B'D' + A'B + A'C + C\,D$

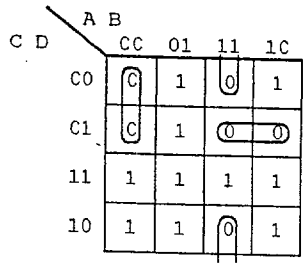
$F = \prod M(0, 1, 9, 12, 13, 14) = (A + B + C + D)$
$(A + B + C + D')(A' + B' + C + D)$
$(A' + B' + C + D')(A' + B' + C' + D)$
$(A' + B + C + D')$

5.12 (b)



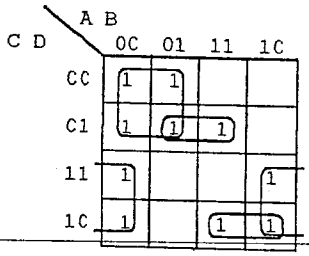
$F' = A'B'C' + A\,B\,D' + A\,C'D$

5.12 (c)



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

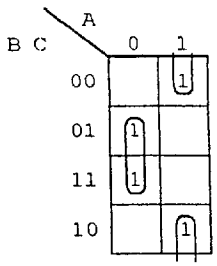
5.13



$F = A\,C\,D' + B\,C'D + B'C + A'C'$

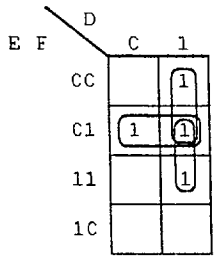
Minterms m_0, m_1, m_2, m_3, m_4, m_{10}, and m_{11} can be made don't cares, individually, without changing the given expression. However, if m_{13} or m_{14} is made a don't care, the term BC'D or the term ACD' (respectively) is not needed in the expression.

5.14 (a)



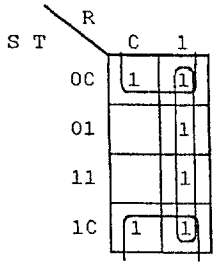
F = A'C + AC'

5.14 (b)



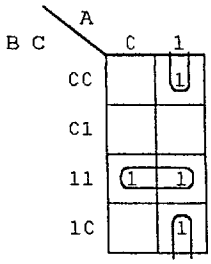
F2 = E'F + DE' + DF

5.14 (c)



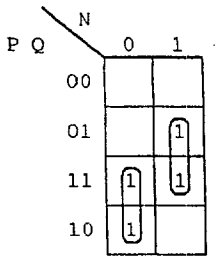
F = T + R

5.14 (d)



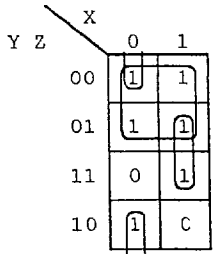
F = BC + AC'

5.14 (e)



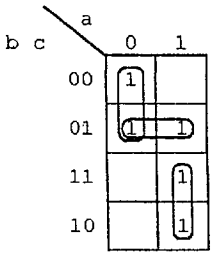
F = N'P + NQ

5.14 (f)



F = Y' + X'Z' + XZ

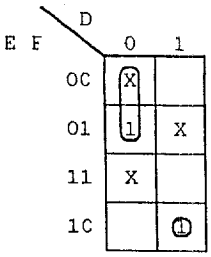
5.15 (a)



f = a'b' + ab + b'c

f = a'b' + ab + ac

5.15 (b)

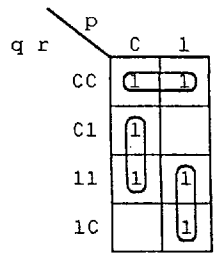


G = DEF' + D'E'

G = DEF' + D'F

G = DEF' + E'F

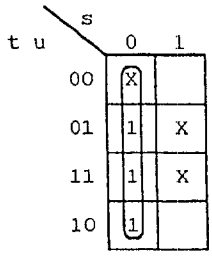
5.15 (c)



F = p'r + q'r' + pq

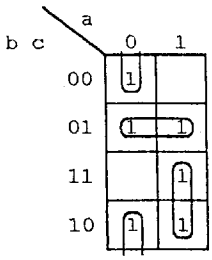
F = p'q' + pr' + qr

5.15 (d)



F = s'

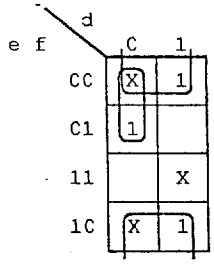
5.15 (e)



F = a'c' + b'c + ab

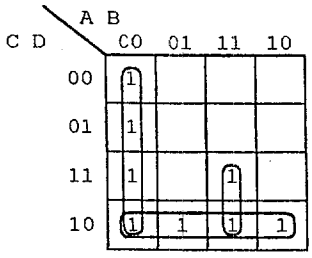
F = a'b' + b c' + ac

5.15 (f)



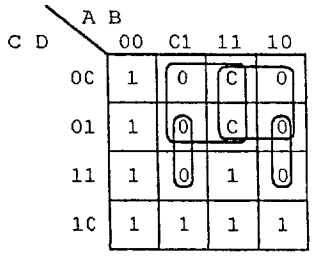
g = d'e' + f'

5.16 (a)



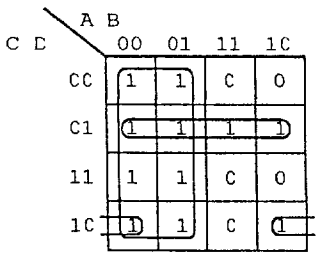
5.16 (b) F = A'B' + CD' + ABC

5.16 (c)



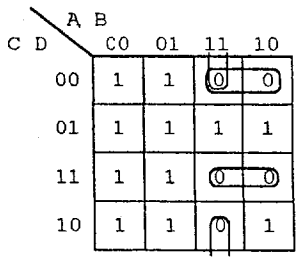
F = (B' + C)(A + B' + D')(A' + C)(A' + B + D')

5.17 (a), (b)



F = A' + C'D + B'CD'

5.17 (c)



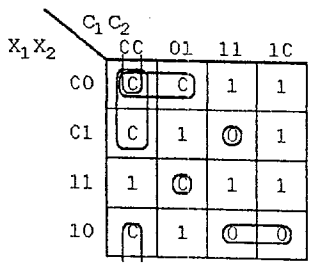
F = (A' + C + D)(A' + C' + D')(A' + B' + D)

Alt: F = (A' + C + D)(A' + C' + D')(A' + B' + C')

5.18 (a)

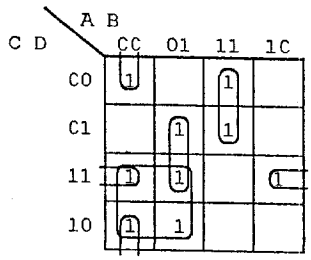
$C_1 C_2 X_1 X_2$	Z
0 0 0 0	0
0 0 0 1	0
0 0 1 0	0
0 0 1 1	1
0 1 0 0	0
0 1 0 1	1
0 1 1 0	1
0 1 1 1	0
1 0 0 0	1
1 0 0 1	1
1 0 1 0	0
1 0 1 1	1
1 1 0 0	1
1 1 0 1	0
1 1 1 0	0
1 1 1 1	1

5.18 (b)



F = (C1 + C2 + X1) (C1 + X1 + X2) (C1 + C2' + X1' + X2')
(C1' + C2' + X1 + X2') (C1' + X1' + X2) (C1 + C2 + X2)

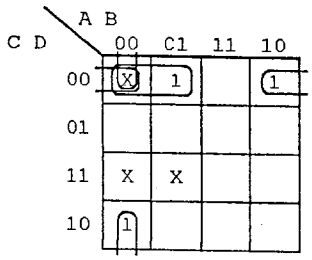
5.19 (a)



F = A B C' + B'C D + A'C + A'B'D' + A'B D

Alt: F = A B C' + B'C D + A'C + A'B'D' + B C'D

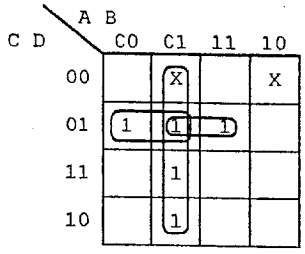
5.19 (b)



F = A'C'D' + B'C'D' + A'B'D'

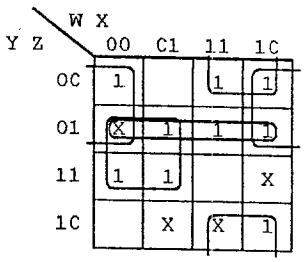
Alt: F = A'C'D' + B'C'D' + A'B'C

5.19 (c)



F = A'C'D + A'B + B C'D

5.19 (d)

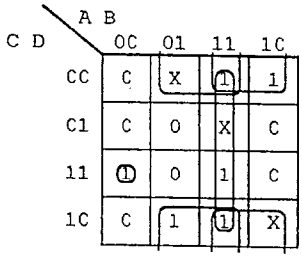


F = X'Y' + W'Z + Y'Z + W Z'

Alt: F = X'Y' + W Y' + W'Z + W Z'

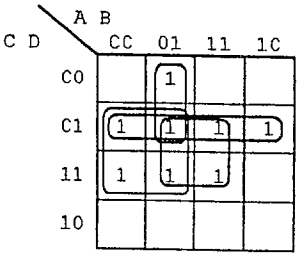
F = X'Y' + W Y' + W'Z + W X'

5.19 (e)



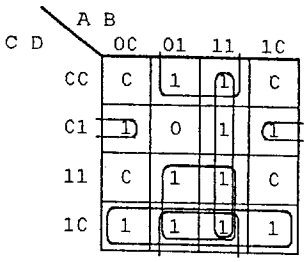
F = A'B'C D + B D' + A D' + A B

5.20 (a)



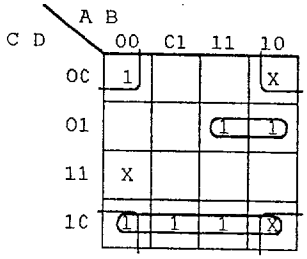
F = A'D + A'B C' + C'D + B D

5.20 (b)



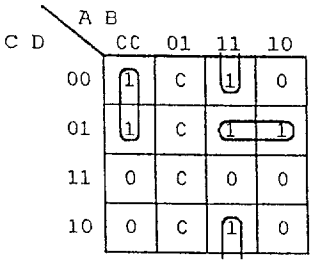
F = B'C'D + C D' + B D' + B C + A B

5.20 (c)



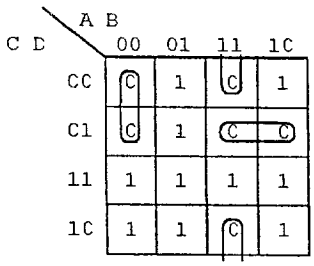
F = B'D' + C D' + A C'D

5.24 (b)



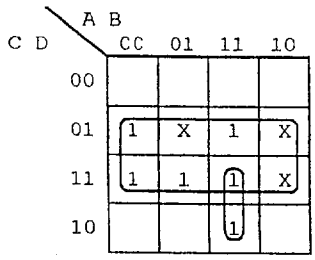
F' = A B D' + A'B'C' + A C'D

5.24 (c)



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

5.25



F = D + A B C

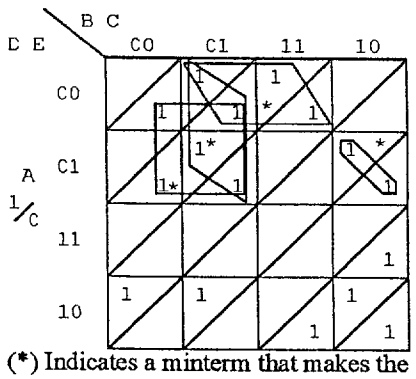
5.26 Prime implicants for f': abc'e, ac'd', ab'e, a'ce, b'c'de', c'd'e, a'd'e

Prime implicants for f: a'd'e', ace, a'ce', bde', abc, bce', b'c'de, a'c'de, a'bc'd, ab'de

5.27 For F: b'c'de', a'ce, ab'e', ac'd', abc'e, c'd'e, a'd'e

For G: ab'ce, a'bcd, a'bde', cde, b'de, a'b'c'd, a'c'e'

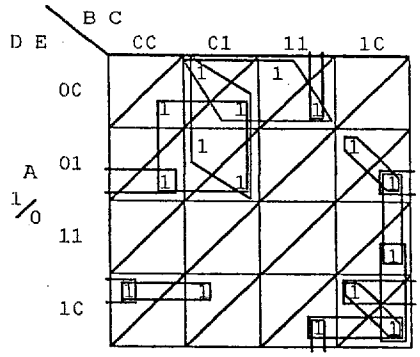
5.28 (a)



(*) Indicates a minterm that makes the corresponding prime implicant essential.

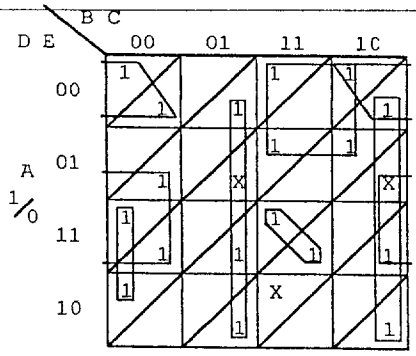
a'b'd'→m_1; cd'e'→m_{28}; bc'd'e→m_{25}; b'cd'→m_{21}

5.28 (b)



a'b'd', cd'e', bc'd'e, b'cd', ac'de', ab'ce', ab'de', a'c'd'e, a'bc'e, a'bc'd, bc'de', a'bde', a'bce'

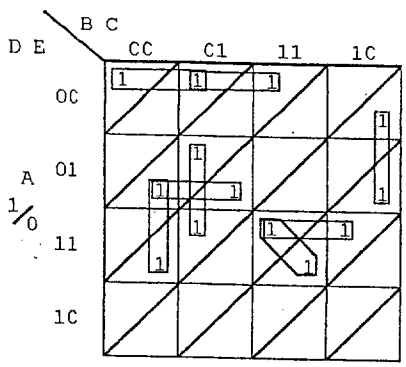
5.29 (a)



F = A'B'C + A'B C' + A B'C'D + C'D'E' + A B D' + B C D E + A'C'E

Alt: F = A'B'C + A'B C' + A B'C'D + C'D'E' + A B D' + B C D E + A'B'E

5.29 (b)

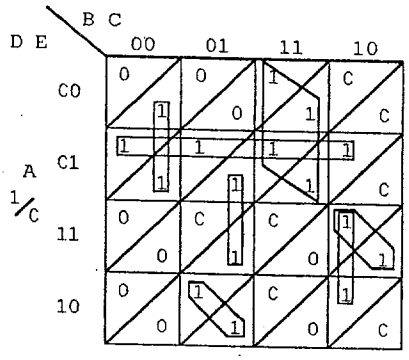


F = A'B'C'E + A'B C'D' + B C D E + A B'D'E' + A B D E + A C D'E' + A'B'D'E + AB'C E

Alt: F = A'B'C'E + A'B C'D' + B C D E + A B'D'E' + A B D E + A C D'E' + B'C D'E + AB'C E

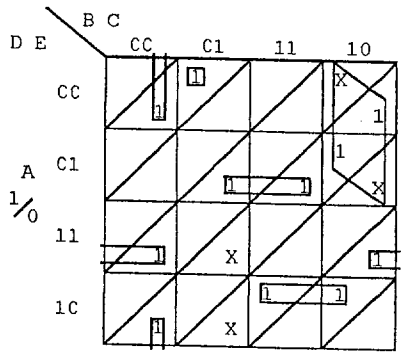
F = A'B'C'E + A'B C'D' + B C D E + A B'D'E' + A B D E + A C D'E' + B'C D'E + AC D E

5.30



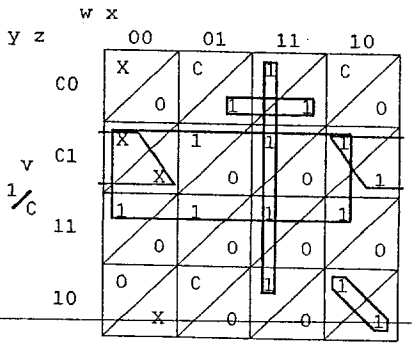
F = A'B'C'D' + BC'DE + BCD' + B'CDE' + ABCD + A'B'CE + AD'E

5.31



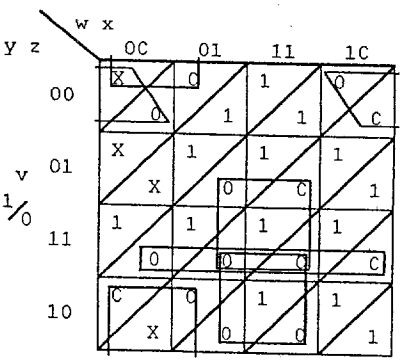
F = AB'CD'E' + BCD' + ABDE' + A'B'C'E' + A'C'DE + A'CD'E

5.32 (a)



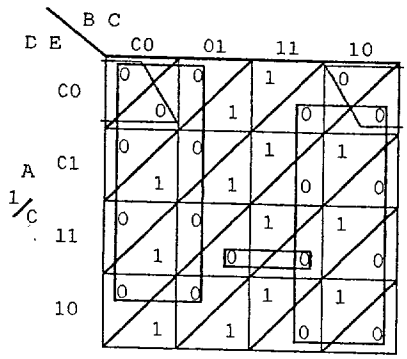
F = v'x y'z' + x'y'z + v z + w x'y z' + v w x

5.32 (b)



F = (x + y + z) (v + y' + z') (v + x' + z') (v + x' + y') (v' + w + z)

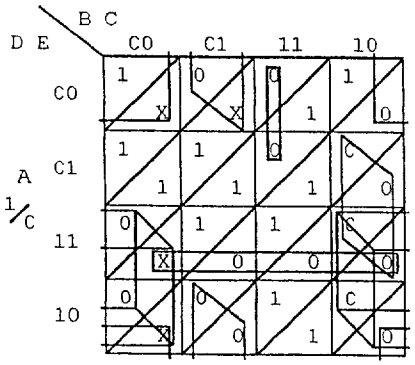
5.33 (a)



F = (C + D + E) (A'+ B) (A + B') (A + C'+ D'+ E')

Alt: F = (C + D + E) (A'+ B) (A + B') (B + C'+ D'+ E')

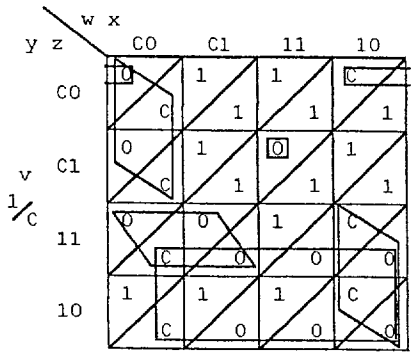
5.33 (b)



F = (C + D') (A + D'+ E') (B'+ C + E') (A'+ B'+ C'+ D)
(A + C + E) (B + C'+ E)

Alt: F = (C + D') (A + D'+ E') (A + B'+ C) (B'+ C + E')
(A'+ B'+ C'+ D) (B + C'+ E)

5.34 (a)

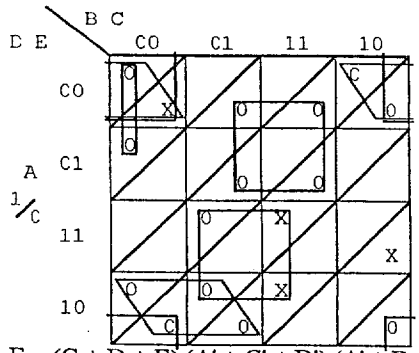


F = (v'+ w'+ x'+ y + z') (w + y'+ z') (v + y') (w + x + y)
(v'+ x + y + z) (w'+ x + y')

Alt: F = (v'+ w'+ x'+ y + z') (w + y'+ z') (v + y') (w + x + y)
(v'+ w'+ x + z) (w'+ x + y')

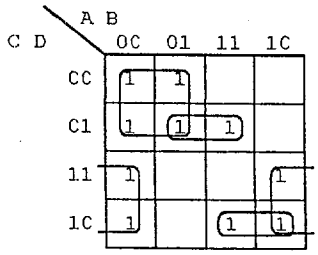
F = (v'+ w'+ x'+ y + z') (w + y'+ z') (v + y') (w + x + y)
(v'+ w'+ x + z) (x + y'+ z')

5.34 (b)



F = (C + D + E) (A' + C' + D') (A' + B + C + D)
(A + C' + D) (B + D' + E) (A + C + E)

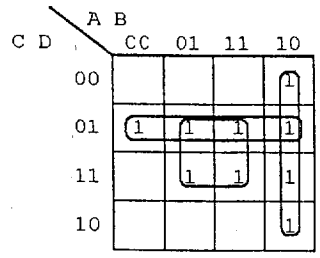
5.35 (a)



F = A C D' + B C'D + B'C + A'C'

m_4, m_{13}, or m_{14} change the minimum sum of products, removing A'C', BC'D, or ACD', respectively.

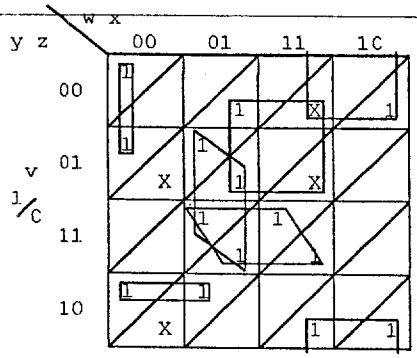
5.35 (b)



F = C'D + B D + A B'

Changing m_1 to a don't care removes *C'D* from the solution.

5.36 (a)



F = $\underset{m_4}{v'x\,y'} + \underset{m_8}{v'w\,z'} + \underset{m_{31}}{x\,y\,z}$ + v w'x'y' + v w'y z' + w'x z

F = $\underline{v'x\,y'} + \underline{v'w\,z'} + \underline{x\,y\,z}$ + v w'x'z' + v w'x y + w'y'z

F = $\underline{v'x\,y'} + \underline{v'w\,z'} + \underline{x\,y\,z}$ + v w'x'y' + v w'y z' + w'y'z

F = $\underline{v'x\,y'} + \underline{v'w\,z'} + \underline{x\,y\,z}$ + v w'x'z' + v w'y z' + w'y'z

5.36 (b) $v'wz' \rightarrow m_8$; $xyz \rightarrow m_{31}$; $v'xy' \rightarrow m_4$

Unit 6 Problem Solutions

6.2 (a)

1	0001✓	1, 5	0-01	a'c'd
5	0101✓	1, 9	-001	b'c'd
9	1001✓	5, 7	01-1	a'bd
12	1100✓	9, 11	10-1	ab'd
7	0111✓	12, 14	11-0	abd'
11	1011✓	7, 15	-111	bcd
14	1110✓	11, 15	1-11	acd
15	1111✓	14, 15	111-	abc

Prime implicants: *a'c'd, b'c'd, a'bd, ab'd, abd', bcd, acd, abc*

6.2 (b)

0	0000✓	0, 1	000- a'b'c'	1, 3, 5, 7	0--1 a'd
1	0001✓	0, 8	-000 b'c'd'	~~1, 5, 3, 7~~	~~0--1~~
8	1000✓	1, 3	00-1✓	6, 7, 14, 15	-11- bc
3	0011✓	1, 5	0-01✓	~~6, 14, 7, 15~~	~~-11-~~
5	0101✓	8, 10	10-0 ab'd'		
6	0110✓	3, 7	0-11✓		
10	1010✓	5, 7	01-1✓		
7	0111✓	6, 7	011-✓		
14	1110✓	6, 14	-110✓		
15	1111✓	10, 14	1-10 acd'		
		7, 15	-111✓		
		14, 15	111-✓		

Prime implicants: *a'b'c', b'c'd', ab'd', acd', a'd, bc*

6.3 (a)

		1	5	7	9	11	12	14	15
1, 5	a'c'd	×	×						
1, 9	b'c'd	×			×				
5, 7	a'bd		×	×					
9, 11	ab'd				×	×			
12, 14	abd'						⊗	⊗	
7, 15	bcd			×					×
11, 15	acd					×			×
14, 15	abc							⊗	×

f = abd' + a'c'd + ab'd + bcd
f = abd' + b'c'd + a'bd + acd

6.3 (b)

		0	1	3	5	6	7	8	10	14	15
1, 3, 5, 7	a'd			⊗	⊗		×				
6, 7, 14, 15	bc					⊗	×			⊗	
0, 1	a'b'c'	×	×								
0, 8	b'c'd'	×						×			
8, 10	ab'd'							×	×		
10, 14	acd'								×		×

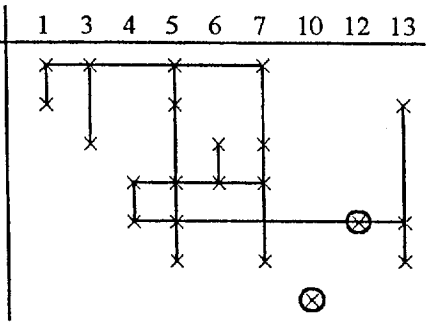
f = a'd + bc + a'b'c' + ab'd'
f = a'd + bc + b'c'd' + ab'd'
f = a'd + bc + b'c'd' + acd'

6.4

1	0001✓	1, 3	00-1✓	1, 3, 5, 7	0--1	a'd
2	0010✓	1, 5	0-01✓	~~1, 5, 3, 7~~	~~0--1~~	
4	0100✓	1, 9	-001✓	1, 5, 9, 13	--01	c'd
3	0011✓	2, 3	001-✓	~~1, 9, 5, 13~~	~~--01~~	
5	0101✓	2, 6	0-10✓	2, 3, 6, 7	0-1-	a'c
6	0110✓	2, 10	-010 b'cd'	~~2, 6, 3, 7~~	~~0-1-~~	
9	1001✓	4, 5	010-✓	4, 5, 6, 7	01--	a'b
10	1010✓	4, 6	01-0✓	4, 5, 12, 13	-10-	bc'
12	1100✓	4, 12	-100✓	~~4, 6, 5, 7~~	~~01--~~	
7	0111✓	3, 7	0-11✓	~~4, 12, 5, 13~~	~~-10-~~	
13	1101✓	5, 7	01-1✓	5, 7, 13, 15	-1-1	bd
15	1111✓	5, 13	-101✓	~~5, 13, 7, 15~~	~~-1-1~~	
		6, 7	011-✓			
		9, 13	1-01✓			
		12, 13	110-✓			
		7, 15	-111✓			
		13, 15	11-1✓			
		13, 15	11-1✓			

Prime implicants: *b'cd', a'd, c'd, a'c, a'b, bc', bd*

1, 3, 5, 7	a'd
1, 5, 9, 13	c'd
2, 3, 6, 7	a'c
4, 5, 6, 7	a'b
4, 5, 12, 13	bc'
5, 7, 13, 15	bd
2, 10	b'cd'



$f = \underline{bc'} + \underline{b'cd'} + a'd + a'b$
$f = \underline{bc'} + \underline{b'cd'} + c'd + a'c$
$f = \underline{bc'} + \underline{b'cd'} + a'c + a'd$

6.5

1	0001✓	1, 5	0-01✓	1, 5, 9, 13	--01	c'd
4	0100✓	1, 9	-001✓	~~1, 9, 5, 13~~	~~--01~~	
8	1000✓	4, 5	010-✓	4, 5, 12, 13	-10-	bc'
5	0101✓	4, 12	-100✓	~~4, 12, 5, 13~~	~~-10-~~	
9	1001✓	8, 9	100-✓	5, 7, 13, 15	-1-1	bd
12	1100✓	8, 12	1-00✓	~~5, 13, 7, 15~~	~~-1-1~~	
7	0111✓	5, 7	01-1✓	8, 9, 12, 13	1-0-	ac'
11	1011✓	5, 13	-101✓	~~8, 12, 9, 13~~	~~1-0-~~	
13	1101✓	9, 11	10-1✓	~~9, 11, 13, 15~~	~~1--1~~	ad
14	1110✓	9, 13	1-01✓	~~9, 13, 11, 15~~	~~1--1~~	
15	1111✓	12, 13	110-✓	12, 13, 14, 15	11--	ab
		12, 14	11-0✓	~~12, 14, 13, 15~~	~~11--~~	
		7, 15	-111✓			
		11, 15	1-11✓			
		13, 15	11-1✓			
		14, 15	111-✓			

Prime implicants: *c'd, bc', bd, ac', ad, ab*

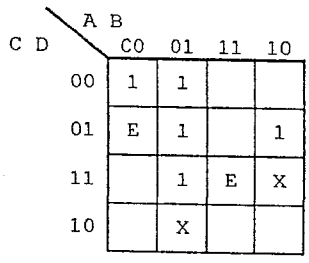
6.5 (contd)

			9	12	13	15
P1	(1, 5, 9, 13)	c'd	×		×	
P2	(4, 5, 12, 13)	bc'		×	×	
P3	(5, 7, 13, 15)	bd			×	×
P4	(8, 9, 12, 13)	ac'	×	×	×	
P5	(9, 11, 13, 15)	ad	×		×	×
P6	(12, 13, 14, 15)	ab		×	×	×

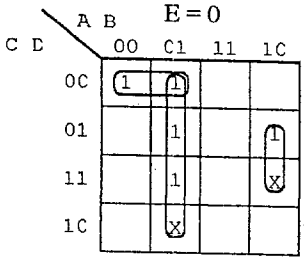
(P1 + P4 + P5) (P2 + P4 + P6) ~~*(P1 + P2 + P3 + P4 + P5 + P6)*~~ *(P3 + P5 + P6)*
= (P4 + P1P2 + P1P6 + P2P5 + P5P6) (P3 + P5 + P6)
= P3P4 + P4P5 + P4P6 + P1P2P3 + ~~*P1P2P5*~~ *+* ~~*P1P2P6*~~ *+* ~~*P1P3P6*~~
+ ~~*P1P5P6*~~ *+ P1P6 +* ~~*P2P3P5*~~ *+ P2P5 +* ~~*P2P5P6*~~ *+* ~~*P3P5P6*~~ *+ P5P6 = 1*

F = (AC' + BD) **or** *(AD + BC')* **or** *(AD + AC')* **or** *(AB + AD)* **or** *(AB + AC')* **or** *(AB + C'D)*
P4 P3 P5 P2 P5 P4 P6 P5 P6 P4 P6 P1

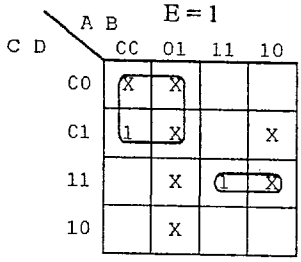
6.6 (a)



$F = MS_0 + EMS_1 = A'B + A'C'D' + AB'D + E(A'C' + ACD)$
or $E(A'C' + BCD)$

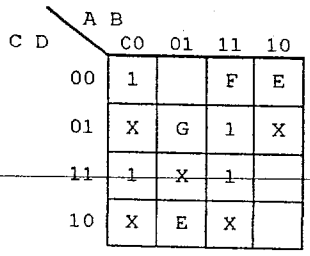


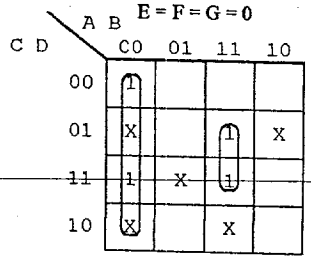
MS_0 = A'C'D' + A'B + A B'D



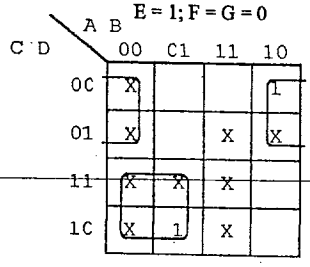
MS_1 = A'C' + ACD
MS_1 = A'C' + BCD

6.6 (b)

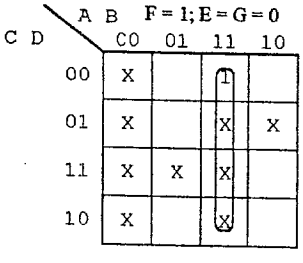




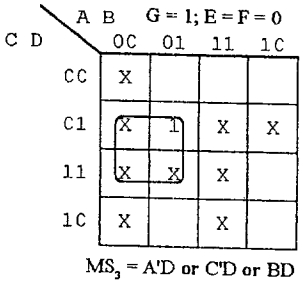
MS_0 = A'B' + A B D



MS_1 = B'C' + A'C
MS_1 = B'C' + B C



MS_2 = A B



MS_3 = A'D or C'D or BD

$Z = A'B' + ABD + E(B'C' + A'C) + F(AB) + G(A'D)$

6.7 (a)

0	0000✓	0, 4	0-00	a'c'd'
4	0100✓	4, 5	010-	a'bc'
3	0011✓	3, 7	0-11	a'cd
5	0101✓	3, 11	-011	b'cd
9	1001✓	5, 7	01-1	a'bd
7	0111✓	5, 13	-101	bc'd
11	1011✓	9, 11	10-1	ab'd
13	1101✓	9, 13	1-01	ac'd

Prime implicants: *a'c'd', a'bc', a'cd, b'cd, a'bd, bc'd, ab'd, ac'd*

6.7 (b)

2	0010✓	2, 6	0-10	a'cd'	4, 5, 12, 13	-10- bc'
4	0100✓	2, 10	-010	b'cd'	~~4, 12, 5, 13~~	~~-10-~~
5	0101✓	4, 5	010-✓		9, 11, 13, 15	1--1 ad
6	0110✓	4, 6	01-0	a'bd'	~~9, 13, 11, 15~~	~~1--1~~
9	1001✓	4, 12	-100✓			
10	1010✓	5, 13	-101✓			
12	1100✓	9, 11	10-1✓			
11	1011✓	9, 13	1-01✓			
13	1101✓	10, 11	101-	ab'c		
15	1111✓	12, 13	110-✓			
		11, 15	1-11✓			
		13, 15	11-1✓			

Prime implicants: *ad, bc', a'cd', b'cd', a'bd', ab'c*

6.8 (a)

		0	3	4	5	7	9	11	13
0, 4	a'c'd'	⊗		×					
4, 5	a'bc'			×	×				
3, 7	a'cd		×			×			
3, 11	b'cd		×					×	
5, 7	a'bd				×	×			
5, 13	bc'd				×			×	
9, 11	ab'd						×	×	
9, 13	ac'd						×		×

f = *a'c'd'* + *a'cd* + *ab'd* + *bc'd*
f = *a'c'd'* + *ac'd* + *a'bd* + *b'cd*

6.8 (b)

		2	4	5	6	9	10	11	12	13	15
2, 6	a'cd'	×			×						
2, 10	b'cd'	×					×				
4, 6	a'bd'		×		×						
10, 11	ab'c						×	×			
4, 5, 12, 13	bc'		×	×					⊗	⊗	
9, 11, 13, 15	ad							×	⊗	×	⊗

f = *bc'* + *ad* + *a'cd'* + *b'cd'*
f = *bc'* + *ad* + *a'cd'* + *ab'c*
f = *bc'* + *ad* + *a'bd'* + *b'cd'*

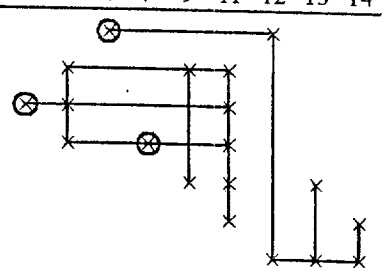
6.9 (a)

1	0001✓	1, 3	00-1✓	1, 3, 9, 11	-0-1	b'd
2	0010✓	1, 9	-001✓	~~1, 9, 3, 11~~	~~-0-1~~	
4	0100✓	2, 3	001-✓	2, 3, 10, 11	-01-	b'c
3	0011✓	2, 10	-010✓	~~2, 10, 3, 11~~	~~-01-~~	
9	1001✓	4, 12	-100	bc'd'	3, 7, 11, 15	--11 cd
10	1010✓	3, 7	0-11✓		~~3, 11, 7, 15~~	~~--11~~
12	1100✓	3, 11	-011✓		9, 11, 13, 15	1--1 ad
7	0111✓	9, 11	10-1✓		~~9, 13, 11, 15~~	~~1--1~~
11	1011✓	9, 13	1-01✓		10, 11, 14, 15	1-1- ac
13	1101✓	10, 11	101-✓		~~10, 14, 11, 15~~	~~1-1-~~
14	1110✓	10, 14	1-10✓		12, 13, 14, 15	11-- ab
15	1111✓	12, 13	110-✓		~~12, 14, 13, 15~~	~~11--~~
		12, 14	11-0✓			
		7, 15	-111✓			
		11, 15	1-11✓			
		13, 15	11-1✓			
		14, 15	111-✓			

Prime implicants: *bc'd', b'd, b'c, cd, ad, ac, ab*

6.9 (a) (contd)

		2	3	4	7	9	11	12	13	14
4, 12	bc'd'			⊗						
1, 3, 9, 11	b'd					×	×			
2, 3, 10, 11	b'c									
3, 7, 11, 15	cd									
9, 11, 13, 15	ad									
10, 11, 14, 15	ac									
12, 13, 14, 15	ab									



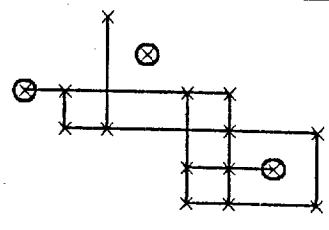
$f = \underline{b'c} + \underline{bc'd'} + \underline{cd} + b'd + ab$
$f = \underline{b'c} + \underline{bc'd'} + \underline{cd} + ad + ab$
$f = \underline{b'c} + \underline{bc'd'} + \underline{cd} + ad + ac$

6.9 (b)

0	0000✓	0, 1	000-✓	0, 1, 8, 9	-00- b'c'
1	0001✓	0, 8	-000✓	~~0, 8, 1, 9~~	~~-00-~~
8	1000✓	1, 5	0-01✓	1, 5, 9, 13	--01 c'd
5	0101✓	1, 9	-001✓	~~1, 9, 5, 13~~	~~--01~~
6	0110✓	8, 9	100-✓	8, 9, 10, 11	10-- ab'
9	1001✓	8, 10	10-0✓	~~8, 10, 9, 11~~	~~10--~~
10	1010✓	8, 12	1-00✓	8, 9, 12, 13	1-0- ac'
12	1100✓	5, 7	01-1 a'bd	~~8, 12, 9, 13~~	~~1-0-~~
7	0111✓	5, 13	-101✓		
11	1011✓	6, 7	011- a'bc		
13	1101✓	9, 11	10-1✓		
		9, 13	1-01✓		
		10, 11	101-✓		
		12, 13	110-✓		

Prime implicants: *a'bd, a'bc, b'c', c'd, ab', ac'*

		0	1	5	6	8	9	11	13
5, 7	a'bd								
6, 7	a'bc								
0, 1, 8, 9	b'c'								
1, 5, 9, 13	c'd								
8, 9, 10, 11	ab'								
8, 9, 12, 13	ac'								



$f = a'bc + b'c' + ab' + c'd$

6.9 (c) $f = a'b + bc + ab'c' + bd + cd$
~~$f = a'b + bc + ab'c' + ad + cd$~~
$f = a'b + bc + ab'c' + ad + a'c$

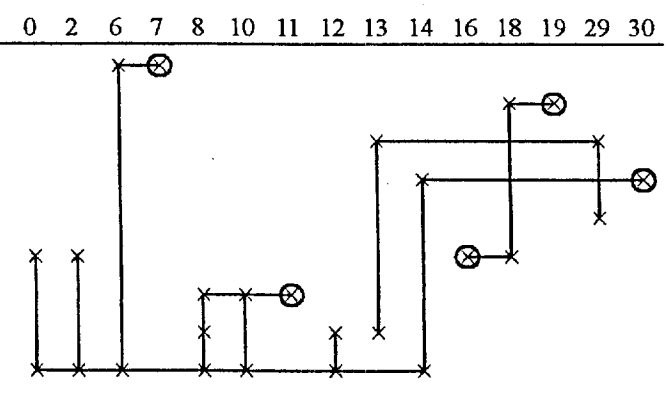
6.10 Prime implicants: *abc', bc'd, a'bd, b'cd, a'c, a'b'd'*

$f = abc' + b'cd + a'c + a'b'd' + a'bd$
$f = abc' + b'cd + a'c + a'b'd' + bc'd$

6.11

0	00000✓	0, 2	000-0✓	0, 2, 4, 6	00--0✓	0, 2, 4, 6, 8, 10, 12, 14	0---0 A'E'
2	00010✓	0, 4	00-00✓	0, 2, 8, 10	0-0-0✓	~~0, 2, 8, 10, 4, 6, 12, 14~~	~~0---0~~
4	00100✓	0, 8	0-000✓	0, 2, 16, 18	-00-0 B'C'E'	~~0, 4, 8, 12, 2, 6, 10, 14~~	~~0---0~~
8	01000✓	0, 16	-0000✓	~~0, 4, 2, 6~~	~~00--0~~		
16	10000✓	2, 6	00-10✓	0, 4, 8, 12	0--00✓		
6	00110✓	2, 10	0-010✓	~~0, 8, 2, 10~~	~~0-0-0~~		
9	01001✓	2, 18	-0010✓	~~0, 8, 4, 12~~	~~0--00~~		
10	01010✓	4, 6	001-0✓	~~0, 16, 2, 18~~	~~-00-0~~		
12	01100✓	4, 12	0-100✓	2, 6, 10, 14	0--10✓		
18	10010✓	8, 9	0100-✓	~~2, 10, 6, 14~~	~~0--10~~		
7	00111✓	8, 10	010-0✓	4, 6, 12, 14	0-1-0✓		
11	01011✓	8, 12	01-00✓	~~4, 12, 6, 14~~	~~0-1-0~~		
13	01101✓	16, 18	100-0✓	8, 9, 10, 11	010-- A'BC'		
14	01110✓	6, 7	0011- A'B'CD	8, 9, 12, 13	01-0- A'BD'		
19	10011✓	6, 14	0-110✓	~~8, 10, 9, 11~~	~~010--~~		
21	10101✓	9, 11	010-1✓	8, 10, 12, 14	01--0✓		
29	11101✓	9, 13	01-01✓	~~8, 12, 9, 13~~	~~01-0-~~		
30	11110✓	10, 11	0101-✓	~~8, 12, 10, 14~~	~~01--0~~		
		10, 14	01-10✓				
		12, 13	0110-✓				
		12, 14	011-0✓				
		18, 19	1001- AB'C'D				
		13, 29	-1101 BCD'E				
		14, 30	-1110 BCDE'				
		21, 29	1-101 ACD'E				

6, 7	A'B'CD
18, 19	AB'C'D
13, 29	BCD'E
14, 30	BCDE'
21, 29	ACD'E
0, 2, 16, 18	B'C'E'
8, 9, 10, 11	A'BC'
8, 9, 12, 13	A'BD'
0, 2, 4, 6, 8, 10, 12, 14	A'E'



$F = BCDE' + AB'C'D + B'C'E' + A'BC' + A'B'CD + BCD'E + A'E'$

6.12 (a)

0	00000✓	0, 1	0000-✓	0, 1, 2, 3	000--✓	0, 1, 2, 3, 8, 9, 10, 11	0-0-- A'C'
1	00001✓	0, 2	000-0✓	0, 1, 8, 9	0-00-✓	~~0, 1, 8, 9, 2, 3, 10, 11~~	~~0-0--~~
2	00010✓	0, 8	0-000✓	~~0, 2, 1, 3~~	~~000--~~	~~0, 2, 8, 10, 1, 3, 9, 11~~	~~0-0--~~
8	01000✓	1, 3	000-1✓	0, 2, 8, 10	0-0-0✓		
3	00011✓	1, 9	0-001✓	~~0, 8, 1, 9~~	~~0-00-~~		
6	00110✓	1, 17	-0001✓	~~0, 8, 2, 10~~	~~0-0-0~~		
9	01001✓	2, 3	0001-✓	1, 3, 9, 11	0-0-1✓		
10	01010✓	2, 6	00-10 A'B'DE'	~~1, 9, 3, 11~~	~~0-0-1~~		
17	10001✓	2, 10	0-010✓	1, 9, 17, 25	--001 C'D'E		
20	10100✓	8, 9	0100-✓	~~1, 17, 9, 25~~	~~--001~~		
11	01011✓	8, 10	010-0✓	2, 3, 10, 11	0-01-✓		
21	10101✓	3, 11	0-011✓	~~2, 10, 3, 11~~	~~0-01-~~		
25	11001✓	9, 11	010-1✓	8, 9, 10, 11	010--✓		
28	11100✓	9, 25	-1001✓	~~8, 10, 9, 11~~	~~010--~~		
23	10111✓	10, 11	0101-✓				
30	11110✓	17, 21	10-01 AB'D'E				
31	11111✓	17, 25	1-001✓				
		20, 21	1010- AB'CD'				
		20, 28	1-100 ACD'E'				
		21, 23	101-1 AB'CE				
		28, 30	111-0 ABCE'				
		23, 31	1-111 ACDE				
		30, 31	1111- ABCD				

		0	1	2	3	6	8	9	10	11	17	20	21	23	25	28	30	31
2, 6	A'B'DE'					⊗												
17, 21	AB'D'E										⊗							
20, 21	AB'CD'												⊗					
20, 28	ACD'E'															⊗		
21, 23	AB'CE													⊗				
28, 30	ABCE'																⊗	
23, 31	ACDE																	⊗
30, 31	ABCD																	⊗
1, 9, 17, 25	C'D'E		⊗															
0, 1, 2, 3, 8, 9, 10, 11	A'C'	⊗		⊗	⊗		⊗		⊗	⊗								

$f = A'C' + C'D'E + A'B'DE' + ACDE + ABCE' + AB'CD'$

~~$f = A'C' + C'D'E + A'B'DE' + ACD'E' + AB'CE + ABCD$~~

6.12 (b) $f = BD + CDE' + A'BC'E + AB'C'D' + B'CD'E + A'B'D'E'$

$f = BD + CDE' + A'BC'E + AB'D'E + A'B'CD' + B'C'D'E'$

6.13

$F = BD' + AB'D + A'B + A'C' + BC'$

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

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

6.14 Prime implicants: *ace, cd'e, b'ce, a'ce', a'b'c, a'cd', a'b'de, c'd'e', a'd'e'*

Essential prime implicants are underlined:

$F = \underline{ace} + \underline{c'd'e'} + \underline{a'ce'} + \underline{a'b'de} + a'cd'$

$F = \underline{ace} + \underline{c'd'e'} + \underline{a'ce'} + \underline{a'b'de} + cd'e$

6.15

1	000001✓	1, 3	0000-1✓	1, 3, 17, 19	0-00-1 A'C'D'F
2	000010✓	1, 17	0-0001✓	~~1, 17, 3, 19~~	~~0-00-1~~
16	010000✓	2, 3	00001-✓	2, 3, 18, 19	0-001- A'C'D'E
32	100000✓	2, 18	0-0010✓	~~2, 18, 3, 19~~	~~0-001-~~
3	000011✓	16, 17	01000-✓	16, 17, 18, 19	0100-- A'BC'D'
17	010001✓	16, 18	0100-0✓	~~16, 18, 17, 19~~	~~0100--~~
18	010010✓	16, 48	-10000 BC'D'E'F'		
48	110000✓	32, 48	1-0000 AC'D'E'F'		
19	010011✓	3, 19	0-0011✓		
26	011010✓	17, 19	0100-1✓		
28	011100✓	18, 19	01001-✓		
15	001111 A'B'CDEF	18, 26	01-010 A'BD'EF'		
29	011101✓	26, 30	011-10 A'BCEF'		
30	011110✓	28, 29	01110- A'BCDE'		
39	100111 AB'C'DEF	28, 30	0111-0 A'BCDF'		
63	111111 ABCDEF				

		1	2	3	16	17	18	19	26	32	39	48	63
15	A'B'CDEF												
39	AB'C'DEF										⊗		
63	ABCDEF											⊗	
16, 48	BC'D'E'F'												
32, 48	AC'D'E'F'									⊗		×	
18, 26	A'BD'EF'								×				
26, 30	A'BCEF'								×				
28, 29	A'BCDE'												
28, 30	A'BCDF'												
1, 3, 17, 19	A'C'D'F	⊗		×		×		×					
2, 3, 18, 19	A'C'D'E		⊗	×		×		×					
16, 17, 18, 19	A'BC'D'				×	×	×	×					

6.15 (a) $G = \underline{AB'C'DEF} + \underline{ABCDEF} + \underline{A'C'D'F} + \underline{A'C'D'E} + \underline{AC'D'E'F'} + A'BC'D' + A'BD'EF'$
$G = \underline{AB'C'DEF} + \underline{ABCDEF} + \underline{A'C'D'F} + \underline{A'C'D'E} + \underline{AC'D'E'F'} + A'BC'D' + A'BCEF'$

6.15 (b) Essential prime implicants are underlined in 6.15 (a).

6.15 (c) If there were no don't cares, prime implicants 15, (26, 30), (28, 29), and (28, 30) are omitted. There is only one minimum solution. Same as (a), except delete the second equation.

6.16 (a) Prime implicants: *ABD'EF, ABC'D'F, AB'CDE'F', ACD'EF, A'BCDE'F', BCEF, A'BDEF, BC'D'E'F, AB'CD'E, AB'D'EF', AB'C'EF', A'B'CDEF'*

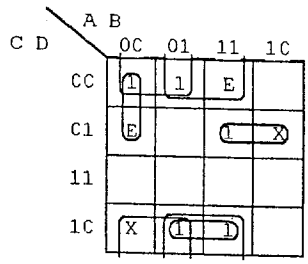
$G = \underline{BCEF} + \underline{AB'CDE'F'} + \underline{A'BCDE'F'} + \underline{A'BDEF} + \underline{BC'D'E'F} + ABD'EF + ACD'EF$
$G = \underline{BCEF} + \underline{AB'CDE'F'} + \underline{A'BCDE'F'} + \underline{A'BDEF} + \underline{BC'D'E'F} + ABD'EF + AB'CD'E$
$G = \underline{BCEF} + \underline{AB'CDE'F'} + \underline{A'BCDE'F'} + \underline{A'BDEF} + \underline{BC'D'E'F} + ABC'D'F + ACD'EF$
$G = \underline{BCEF} + \underline{AB'CDE'F'} + \underline{A'BCDE'F'} + \underline{A'BDEF} + \underline{BC'D'E'F} + ABC'D'F + AB'CD'E$

6.16 (b) Essential prime implicants are underlined in 6.16(a).

6.17 Prime implicants: *AC, AD', AB, CD, BD, A'D*

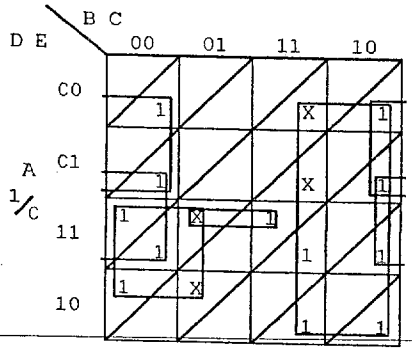
Minimum solutions: *(AD' + CD); (AD' + BD); (AB + BD); (AB + CD); (AB + A'D)*

6.18 (a)



$F = \underbrace{AC'D + BCD' + A'D'}_{MS_0} + E\ \underbrace{(A'B'C' + BD')}_{MS_1}$

6.19 (a) Each minterm of the four variables *A, B, C, D* expands to two minterms of the five variables *A, B, C, D, E*. For example, m_4(A,B,C,D) = A'BC'D'
= A'BC'D'E' + A'BC'D'E
= m_8(A,B,C,D,E) + m_9(A,B,C,D,E)



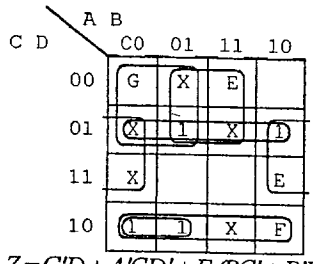
F = A'C'D' + A'B + A B'D + A'C'E + B C D E
F = A'C'D' + A'B + A B'D + A'C'E + A C D E

6.16 (c) If there are no don't cares, the prime implicants are:
ABD'EF, ABC'D'F, AB'CDE'F', ACD'EF, A'BCDE'F', BCEF, A'BDEF, BC'D'E'F

$G = \underline{BCEF} + \underline{AB'CDE'F'} + \underline{ACD'EF} + \underline{A'BCDE'F'} + \underline{A'BDEF} + \underline{BC'D'E'F} + ABC'D'F$

$G = \underline{BCEF} + \underline{AB'CDE'F'} + \underline{ACD'EF} + \underline{A'BCDE'F'} + \underline{A'BDEF} + \underline{BC'D'E'F} + ABD'EF$

6.18 (b)

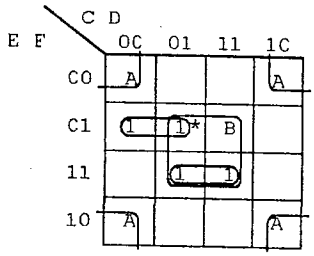


$Z = \underbrace{C'D + A'CD'}_{MS_0} + E\ \underbrace{(BC' + B'D)}_{MS_1} + F\underbrace{(CD')}_{MS_2} + G\underbrace{(A'C')}_{MS_3}$

6.19 (b) Prime implicants: *A'C'D', A'B, AB'D, A'C'E, ACDE, BCDE, B'C'DE*

$F = \underline{A'C'D'} + \underline{A'B} + \underline{AB'D} + A'C'E + ACDE$
$F = \underline{A'C'D'} + \underline{A'B} + \underline{AB'D} + A'C'E + BCDE$

6.20

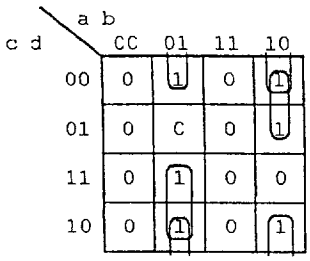


* This square contains 1 + *B*, which reduces to 1.

$G = \underbrace{C'E'F + DEF}_{MS_0} + A\ \underbrace{(D'F')}_{MS_1} + B\ \underbrace{(DF)}_{MS_2}$

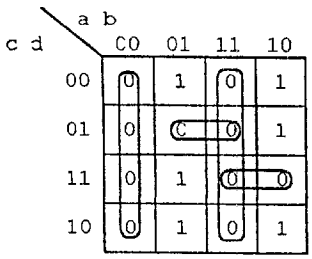
Unit 7 Problem Solutions

7.1 (a)



f = a b'd' + a b'c' + a'b c + a'b d'

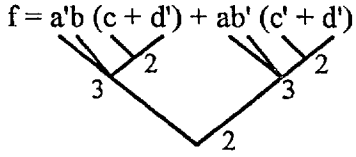
Sum of products solution requires 5 gates, 16 inputs



f = (a'+b') (a + b) (a + c + d') (b + c'+ d')
f = (a'+b') (a + b) (b + c'+ d') (b'+ c + d')
f = (a'+b') (a + b) (a + c + d') (a'+ c'+ d')
f = (a'+b') (a + b) (b'+ c + d') (a'+ c'+ d')

Product of sums solution requires 5 gates, 14 inputs, so product of sums solution is minimum.

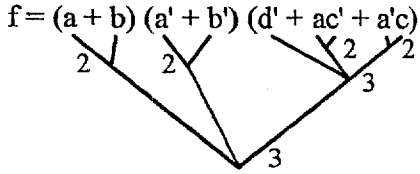
7.1 (b) Beginning with the minimum sum of products solution, we can get



5 gates, 12 inputs

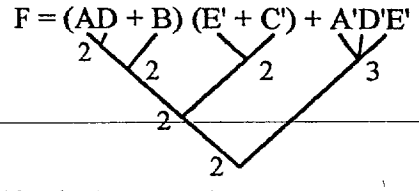
So sum of products solution is minimum.

Beginning with a minimum product of sums solution, we can get



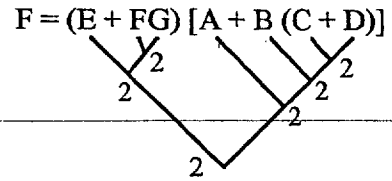
6 gates, 14 inputs

7.2 (a) *AC'D + ADE' + BE' + BC' + A'D'E'*
= E' (AD + B) + A'D'E' + C' (AD + B)



4 levels, 6 gates, 13 inputs

7.2 (b) *AE + BDE + BCE + BCFG + BDFG + AFG*
= AE + AFG + BE (C + D) + BFG (C + D)

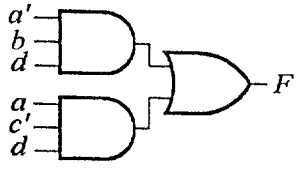


4 levels, 6 gates, 12 inputs

7.3 $F(a, b, c, d)n = a'bd + ac'd$ or $d(a'b + ac') = d(a + b)(a' + c')$

You can obtain this equation in the product of sums form using a Karnaugh map, as shown below:

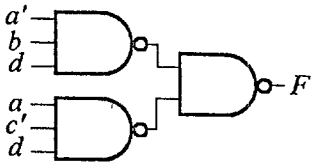
AND-OR



$F=a'bd+ac'd$

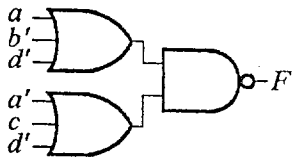
$(F')'=[(a'bd+ac'd)']'$

NAND-NAND



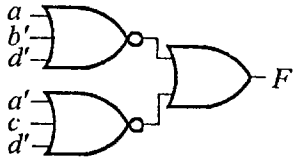
$(F')'=[(a'bd)'(ac'd)']'$

OR-NAND



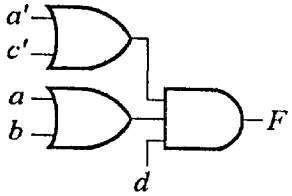
$(F')'=[(a+b'+d')(a'+c+d')]'$

NOR-OR



$(F')'=(a+b'+d')'+(a'+c+d')'$

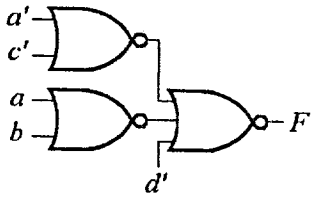
OR-AND



$F=d(a+b)(a'+c')$

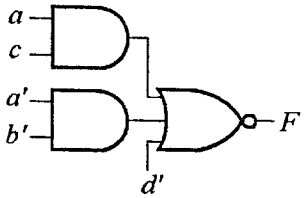
$(F')'=[[d(a+b)(a'+c'))']'$

NOR-NOR



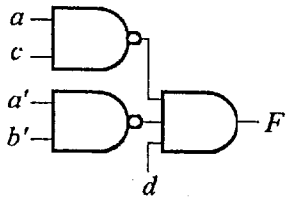
$(F')'=[d'+(a+b)'+(a'+c')']'$

AND-NOR



$(F')'=[d'+a'b'+ac]'$

NAND-AND



$(F')'=d(a'b')'(ac)'$

C D \ A B	CC	01	11	10
00	0	C	0	0
01	0	1	1	1
11	0	1	0	0
10	0	C	0	0

F = A C'D + A'B D

C D \ A B	00	01	11	1C
CC	C	0	0	C
C1	C	1	1	1
11	C	1	C	C
1C	C	0	C	C

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

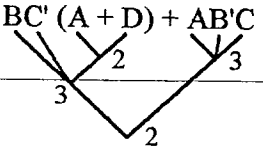
7.4 $F(A, B, C, D) = \sum m(5, 10, 11, 12, 13)$

C D \ A B	C0	01	11	10
00	0	C	1	0
01	0	1	1	0
11	0	C	0	1
10	0	C	0	1

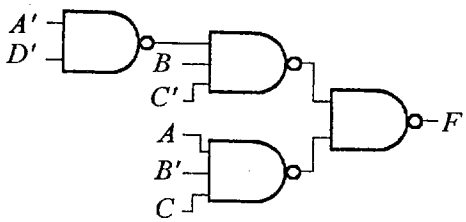
F = B C'D + A B'C + A B C'

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

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



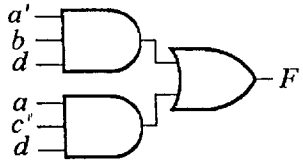
4 gates, 10 inputs



7.3 $F(a, b, c, d)n = a'bd + ac'd$ or $d(a'b + ac') = d(a + b)(a' + c')$

You can obtain this equation in the product of sums form using a Karnaugh map, as shown below:

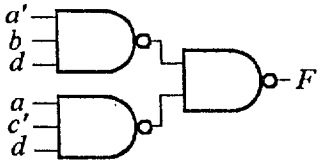
AND-OR



$F=a'bd+ac'd$

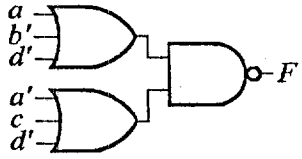
$(F')'=[(a'bd+ac'd)']'$

NAND-NAND



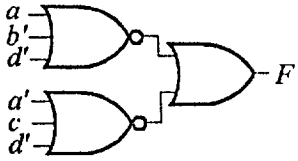
$(F')'=[(a'bd)'(ac'd)']'$

OR-NAND



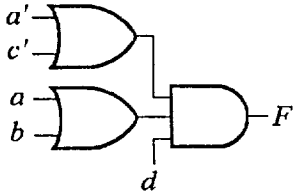
$(F')'=[(a+b'+d')(a'+c+d')]'$

NOR-OR



$(F')'=(a+b'+d')'+(a'+c+d')'$

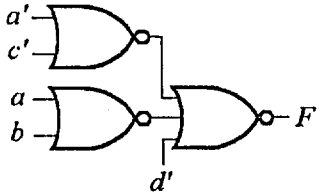
OR-AND



$F=d(a+b)(a'+c')$

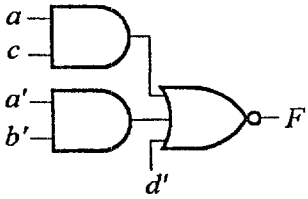
$(F')'=[[d(a+b)(a'+c'))']'$

NOR-NOR



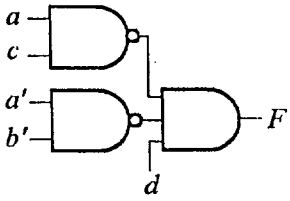
$(F')'=[d'+(a+b)'+(a'+c')']'$

AND-NOR



$(F')'=[d'+a'b'+ac]'$

NAND-AND



$(F')'=d(a'b')'(ac)'$

C D \ A B	00	01	11	10
00	0	C	0	0
01	0	1	1	1
11	0	1	0	0
10	0	C	0	0

$F = AC'D + A'BD$

C D \ A B	00	01	11	10
00	C	0	0	C
01	C	1	1	1
11	C	1	C	C
10	C	0	C	C

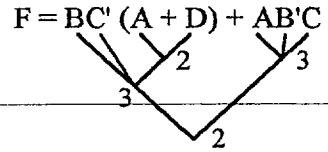
$F = (A + B)(D)(A' + C')$

7.4 $F(A, B, C, D) = \sum m(5, 10, 11, 12, 13)$

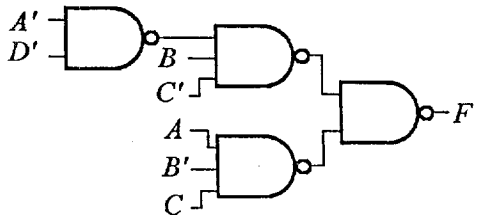
C D \ A B	00	01	11	10
00	0	C	1	0
01	0	1	1	0
11	0	C	0	1
10	0	C	0	1

$F = BC'D + AB'C + ABC'$

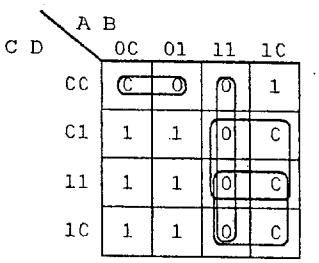
$F = ABC' + BC'D + AB'C = BC'(A + D) + AB'C$



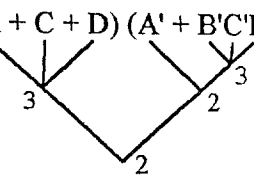
4 gates, 10 inputs



7.5

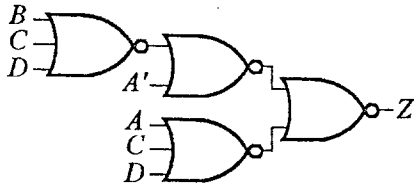


Z = (A + C + D) (A' + B'C'D')

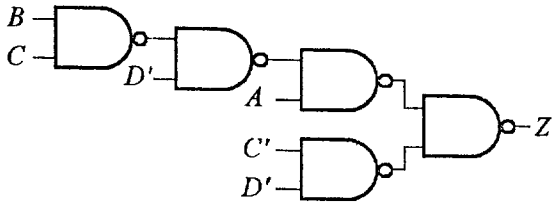


4 gates, 10 inputs

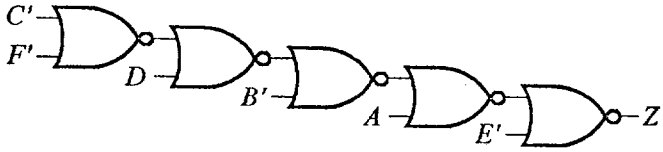
Z = (A + C + D) (A'+ D') (A'+ C') (A'+ B')



7.6 Z = ABC + AD + C'D'
= A (BC + D) + C'D'

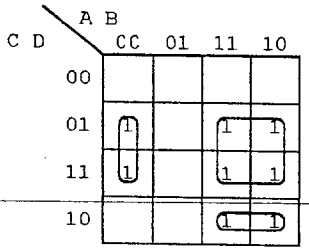


7.7 Z = AE + BDE + BCEF
= E (A + BD + BCF)
= E [A + B (D + CF)]

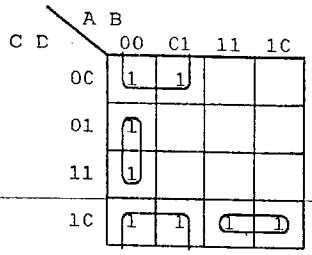


7.8 *For the solution to 7.8, see FLD P. 633*

7.9



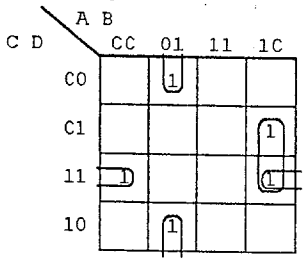
F1 = A C D' + A D + A'B'D



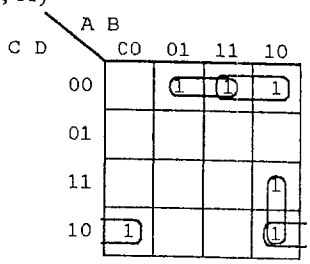
F2 = A'D' + A'B'D + A C D'

6 gates

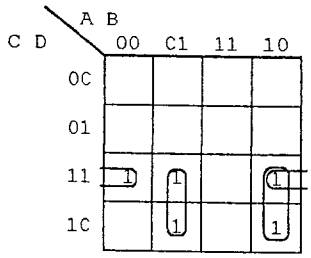
7.10 $f_1(A, B, C, D) = \sum m(3, 4, 6, 9, 11)$
$f_2(A, B, C, D) = \sum m(2, 4, 8, 10, 11, 12)$
$f_3(A, B, C, D) = \sum m(3, 6, 7, 10, 11)$



F1 = A B'D + B'C D + A'B D'

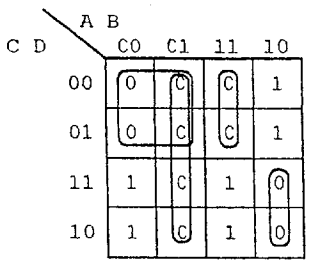


F2 = A B'C + B'C D' + B C'D' + A C'D'
F2 = A B'C + B'C D' + B C'D' + A B'D'
11 gates

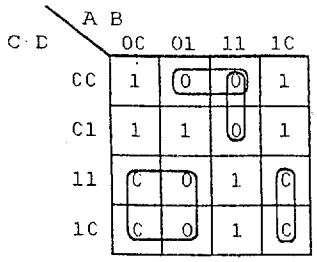


F3 = A B'C + B'C D + A'B C

7.11



F1 = (A + C) (A + B') (A'+B'+C)(A'+B+C')

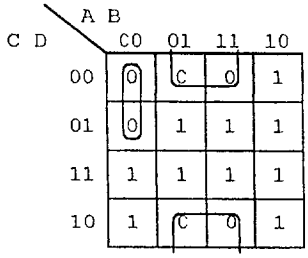


F2 = (B'+C+D)(A'+B'+C)(A+C')(A'+B+C')

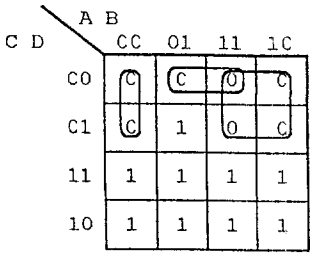
F2 = (A+B'+D)(A'+B'+C)(A+C')(A'+B+C')

8 gates

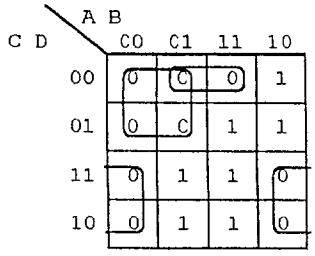
7.12



F1 = (A+B+C)(B'+D)



F2 = (A+B+C)(B'+C+D)(A'+C)

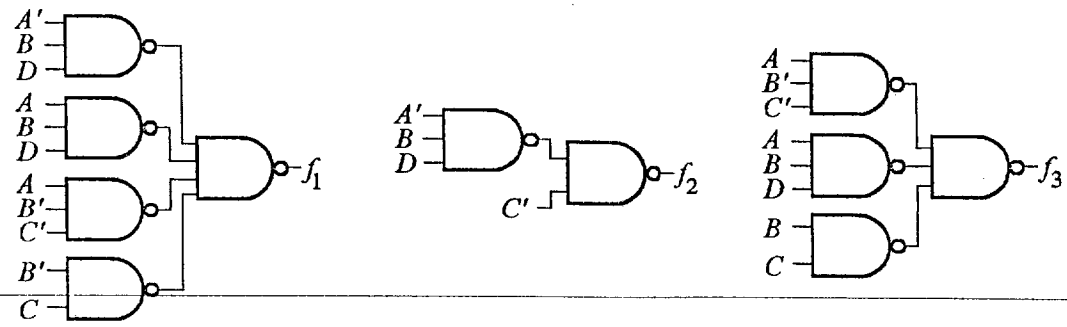


F3 = (B'+C+D)(A+C)(B+C')

9 gates

7.13 (a) Using F = (F')' from Equations (7-23(b)), p. 194:

$f_1 = [(A'BD)'\ (ABD)'\ (AB'C')'\ (B'C)']'$; $f_2 = [C'\ (A'BD)']'$; $f_3 = [(BC)'\ (AB'C')'\ (ABD)']'$

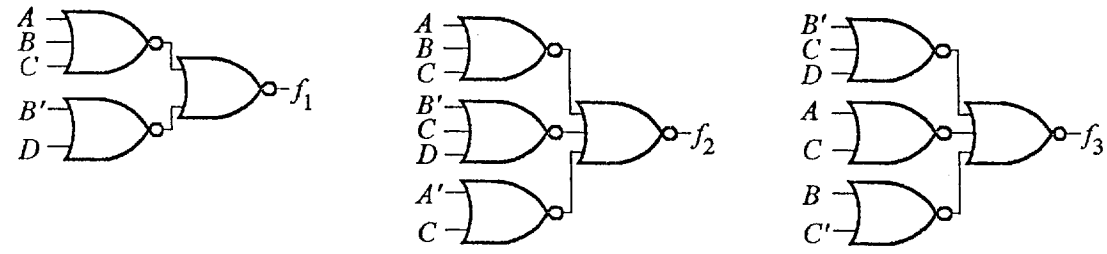


7.13 (b) Using F = (F')' from Equations derived in problem 7.12:

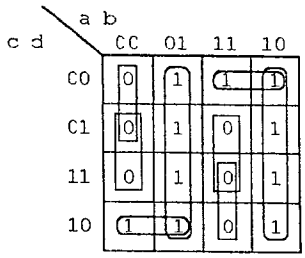
f_1 = [(A + B + C)' + (B' + D)']'

f_2 = [(A + B + C)' + (B' + C + D)' + (A' + C)']'

f_3 = [(B' + C + D)' + (A + C)' + (B + C')']'



7.14 (a)



f = (a+b+c)(a+b+d')(a'+b'+d')(a'+b'+c')

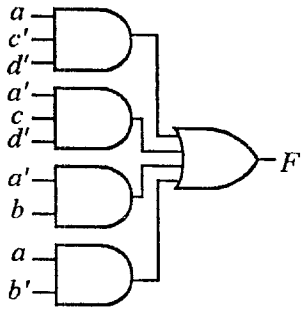
5 gates, 16 inputs

and $f = a'b + ab' + b'cd' + ac'd'$

$f = a'b + ab' + a'cd' + bc'd'$

(two other minimun solutions)

5 gates, 14 inputs *minimal*



7.14 (b) Beginning with the sum of products solution, we get

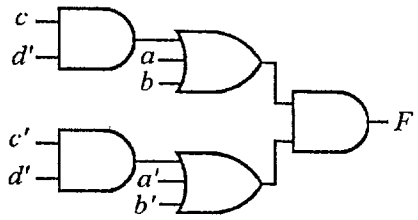
$f = a'b + ab' + d'(a'c + ac')$

$= a'b + ab' + d'(a' + c')(a + c)$ —

6 gates, 14 inputs

But, beginning with the product of sums solution above, we get

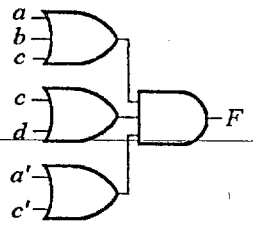
$f = (a + b + cd')(a' + b' + c'd')$ — 5 gates, 12 inputs, which is minimum



7.15 (a) From K-maps:

$F = a'c + bc'd + ac'd$ — 4 gates, 11 inputs

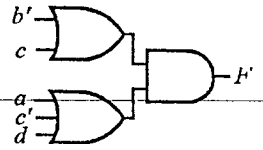
$F = (a + b + c)(c + d)(a' + c')$ — 4 gates, 10 inputs, *minimal*



7.15 (b) From K-maps:

$F = cd + ac + b'c'$ — 4 gates, 9 inputs

$F = (b' + c)(a + c' + d)$ — 3 gates, 7 inputs, *minimal*

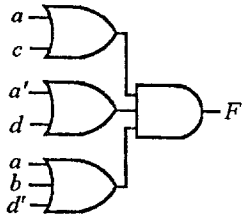


7.15 (c) From K-maps:

$F = ad + a'cd' + bcd$

$= ad + a'cd' + a'bc$ — 4 gates, 11 inputs

$F = (a + c)(a' + d)(a + b + d')$ — 4 gates, 10 inputs, *minimal*

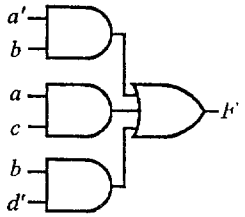


7.15 (d) From K-maps:

$F = a'b + ac + bd'$ — 4 gates, 9 inputs, *minimal*

$F = (a + b)(a' + c + d')(a' + b + c)$

$= (a + b)(a' + c + d')(b + c + d)$ — 4 gates, 11 inputs

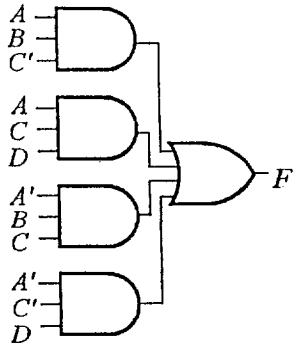


7.16 (a) In this case, multi-level circuits do not improve the solution. From K-maps:

$F = ABC' + ACD + A'BC + A'C'D$ — 5 gates, 16 inputs, ***minimal***

$F = (A' + B + C)\,(A + C + D)\,(A' + C' + D)\,(A + B + C')$ — 5 gates, 16 inputs, ***also minimal***

Either answer is correct.



7.16 (b) Too many variables to use a K-map; use algebra.

Add ACE by consensus, then use $X + XY = X$

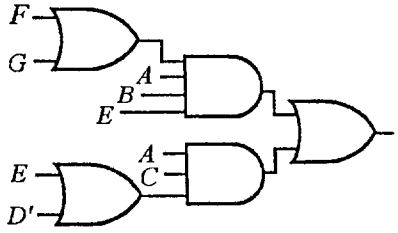
$$ABCE + ABEF + ACD' + ABEG + ACDE + ACE$$

$$= ABEF + ACD' + ABEG + ACE$$

$$F = ABE\,(F + G) + AC\,(D' + E)$$

2 2 4 3 2

5 gates, 13 inputs, ***minimal***



7.17 (a)

	$ABCD$	F
0	0 0 0 0	0
1	0 0 0 1	0
2	0 0 1 0	0
3	0 0 1 1	1
4	0 1 0 0	0
5	0 1 0 1	1
6	0 1 1 0	1
7	0 1 1 1	1
8	1 0 0 0	0
9	1 0 0 1	1
10	1 0 1 0	1
11	1 0 1 1	1
12	1 1 0 0	1
13	1 1 0 1	1
14	1 1 1 0	1
15	1 1 1 1	1

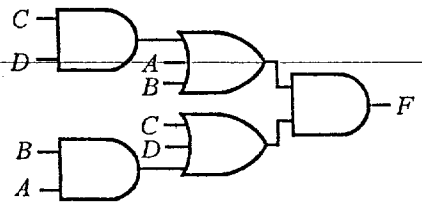
$F = \prod M(0, 1, 2, 4, 8)$

7.17 (b)

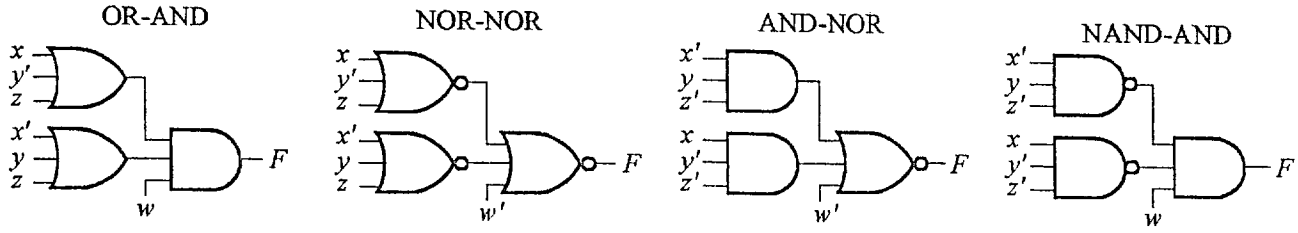
C D \ A B	00	01	11	10
00	0	0	1	0
01	0	1	1	1
11	1	1	1	1
10	0	1	1	1

$F = (A + C + D)\,(A + B + C)$
$(A + B + D)\,(B + C + D)$
$= (A + D + BC)(B + C + AD)$ or
$= (A + C + BD)(B + D + AC)$ or
$= (C + D + AB)(A + B + CD)$

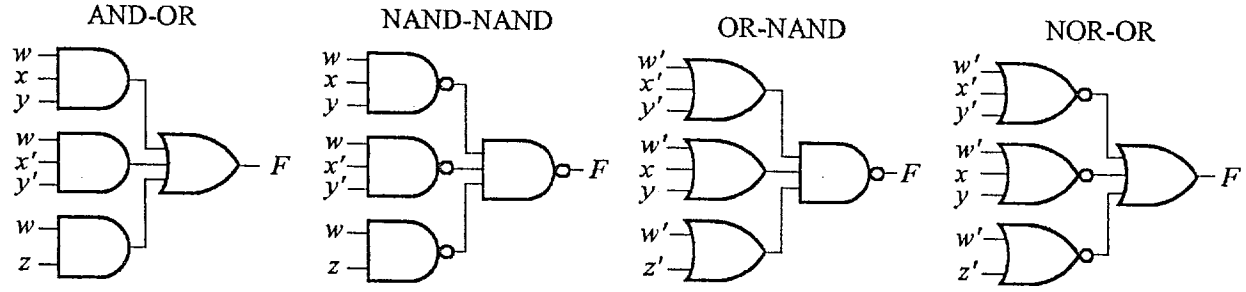
This solution has 5 gates, 12 inputs. Beginning with the sum of products requires 6 gates.



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



From Karnaugh map: $F = wxy + wx'y' + wz$



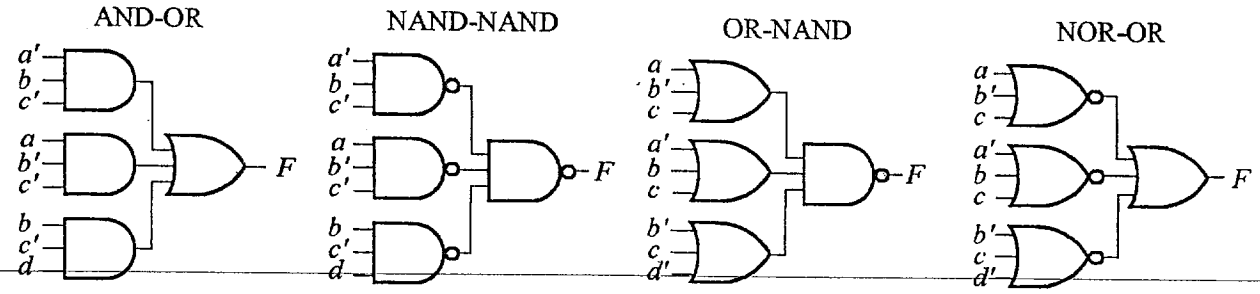
7.18 (b) $F(a, b, c, d) = \sum m(4, 5, 8, 9, 13)$

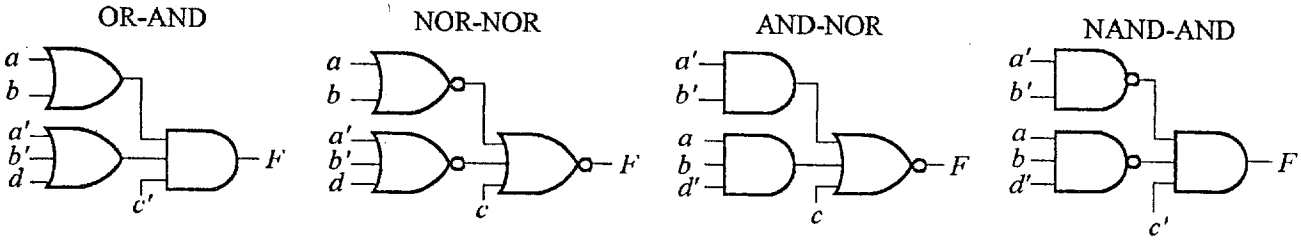
From Kmap:

$F = a'bc' + ab'c' + bc'd$

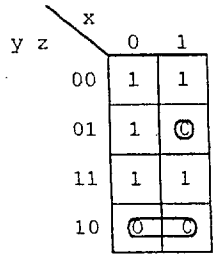
$F = a'bc' + ab'c' + ac'd$

$F = c'(a + b)(a' + b' + d)$



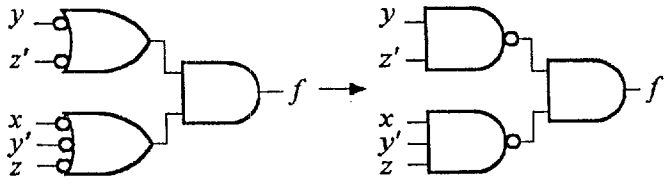


7.19 (a)

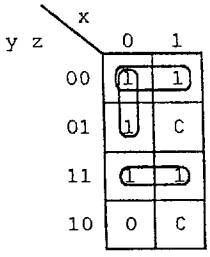


F = (y'+z)(x'+y+z')

From Kmap:
$F = (y' + z)(x' + y + z')$

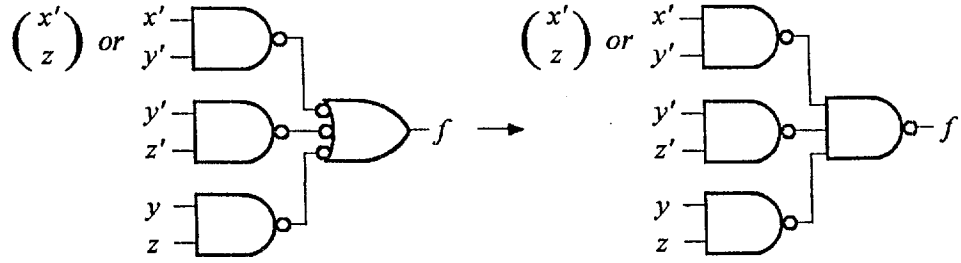


7.19 (b)

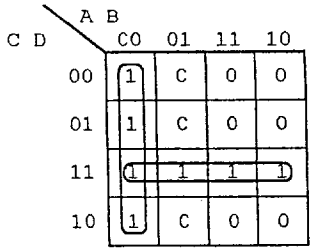


F = yz + y'z' + x'y'

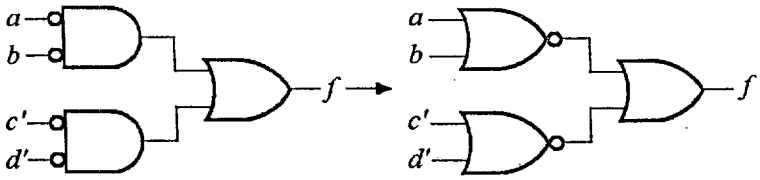
F = yz + y'z' + x'z



7.20 (a) Using OR and NOR gates:

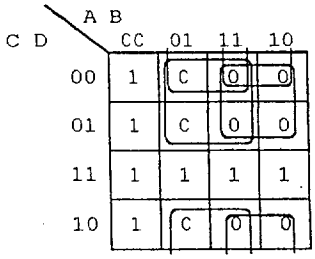


F = A'B' + CD

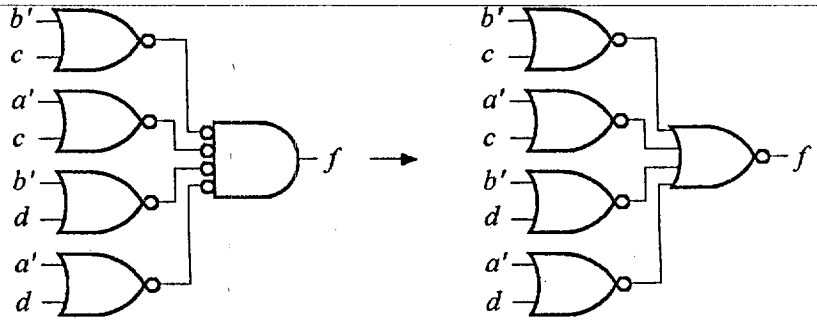


7.20 (b)

Using NOR gates only:



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



7.21 (a) NAND gates:

$F = D' + B'C + AB$

(Refer to problem 5.4 for K-map)

NOR gates:

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

7.21 (b) NAND gates:

$F = a'bc' + ac'd + b'cd'$

(Refer to problem 5.8(a) for K-map)

NOR gates:

$F = (b' + c')\ (c' + d')\ (a + b + c)\ (a' + c + d)$

7.21 (c) NAND gates:

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

(Refer to problem 5.8(b) for K-map)

NOR gates:

$F = (b + d)\ (b' + d')\ (a' + c)\ (b' + c')$

$F = (b + d)\ (b' + d')\ (a' + c)\ (c' + d)$

7.21 (d) NAND gates:

$F = AB'CD' + A'C'E + C'DE + A'DE + ABCDE' + A'C'D + B'C'E' + A'B'D$

$F = AB'CD' + A'C'E + C'DE + A'DE + ABCDE' + A'C'D + B'C'E' + A'B'E'$

(Refer to problem 5.9(a) for K-map)

NOR gates:

$F = (B' + D + E)\ (A + C' + D)\ (A' + B + C' + D')\ (A' + B' + C + E)\ (A + B' + C' + E)\ (A' + C + D + E')\ (A' + B' + C' + E')$

7.21 (e) NAND gates:

$F=A'CD'+A'BE'+CDE+AB'C'D'+B'D'E+AB'DE'$

$F=A'CD'+A'BE'+CDE+AB'C'E'+AB'CD+B'D'E$

$F=A'CD'+A'BE'+CDE+AB'DE'+AB'C'E'+B'D'E$

$F=A'CD'+A'BE'+CDE+AB'DE'+AB'C'E+B'CE$

(Refer to problem 5.9(b) for K-map)

NOR gates:

$F = (A' + C' + D + E)\ (C + D' + E')\ (A' + B' + E)\ (A + B + D' + E)\ (A + B + C)\ (B' + D + E')$

7.21 (f) NAND gates:

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

(Refer to problem 5.22(a) for K-map)

NOR gates:

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

7.21 (g) NAND gates:

$f = x'y' + wy' + w'z + wz'$

$f = x'y' + wy' + wx' + w'z$

$f = x'y' + wy' + y'z + wz'$

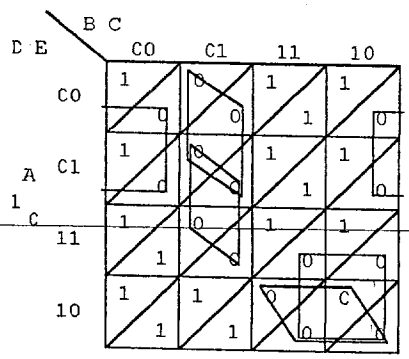
(Refer to problem 5.22(b) for K-map)

NOR gates:

$f = (w + x' + z)\ (w + y' + z)\ (w' + y' + z')$

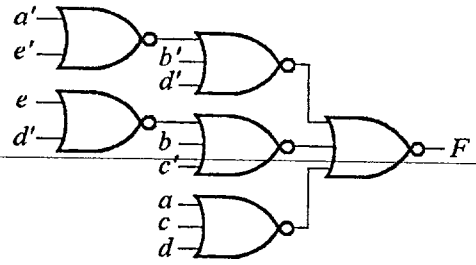
$f = (w + x' + z)\ (w + y' + z)\ (w' + x' + y')$

7.22 (a)

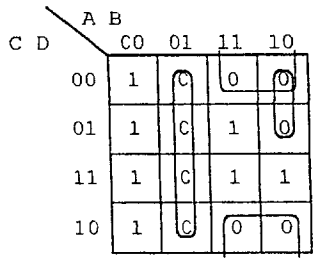


$f = (b + c' + d)\ (b + c' + e')\ (b' + d' + e)\ (a + c + d)\ (a + b' + d')$

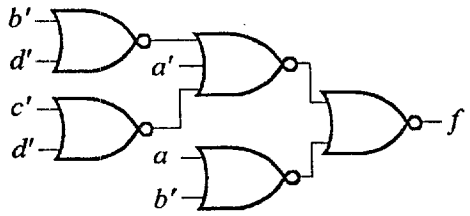
7.22 (b) $f = (b' + d' + ae)\ (b + c' + de')\ (a + c + d)$



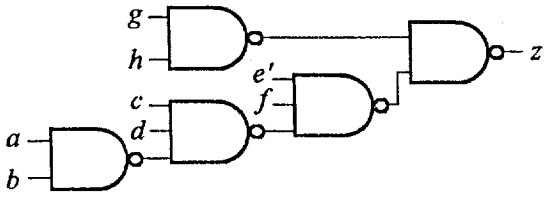
7.23



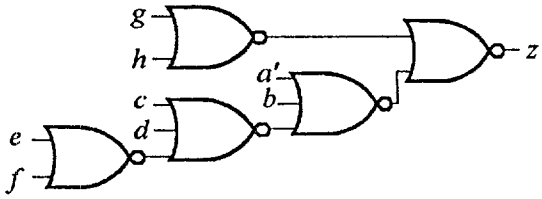
$f = (a' + d)\ (a' + b + c)\ (a + b')$
$= (a + b')\ [a' + d\ (b + c)]$
$= (a + b')\ (a' + bd + cd)$



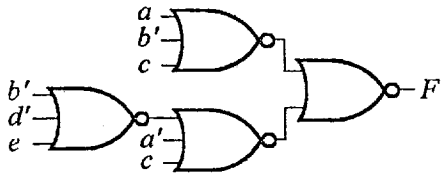
7.24 (a) $Z = abe'f + c'e'f + d'e'f + gh$
$= e'f\ (ab + c' + d') + gh$



7.24 (b) $Z = (a' + b + e + f)(c' + a' + b)(d' + a' + b)(g+h)$
$= [a' + b + c'd'\ (e + f)]\ (g + h)$

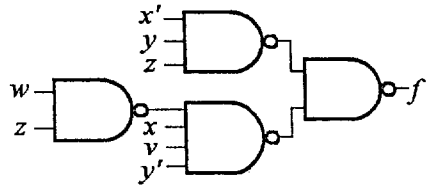


7.25 $F = abde' + a'b' + c$
$= (a + b')\ (a' + bde') + c$
$= (a + b' + c)\ (a' + c + bde')$

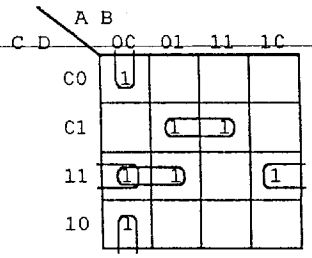


Alternate: $F = (a' + b + c)\ (b' + c + ade')$

7.26 $f = x'yz + xvy'w' + xvy'z'$
$= x'yz + xvy'\ (z' + w')$



7.27 (a)

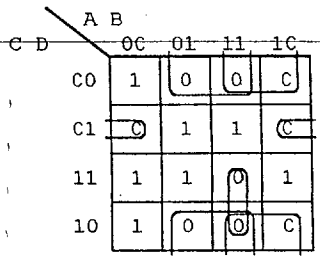


F = B C'D + B'C D + A'B'D' + A'C D

F = B C'D + B'C D + A'B'D' + A'B D

Draw AND-OR circuit and replace all gates with NANDs.

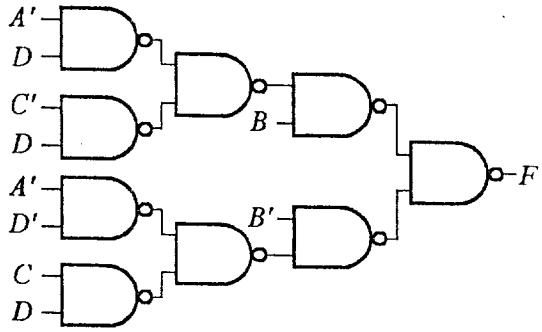
7.27 (b)



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

Draw OR-AND circuit and replace all gates with NORs.

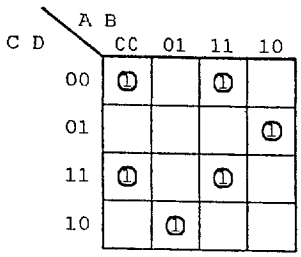
7.27 (c) $F = B\,(A'D + C'D) + B'\,(A'D' + CD)$



Alternative:

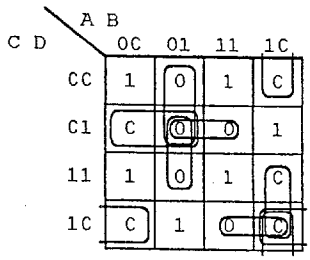
$F = A'\,(B'D' + BD) + D\,(B'C + BC')$
$= D\,(A'B + BC') + B'\,(A'D' + CD)$
$= A'\,(B'D' + CD) + D\,(B'C + BC')$
$= D\,(A'C + BC') + B'\,(A'D' + CD)$

7.28 (a)



F = ABCD + ABC'D' + AB'C'D + A'BCD' + A'B'CD + A'B'C'D'

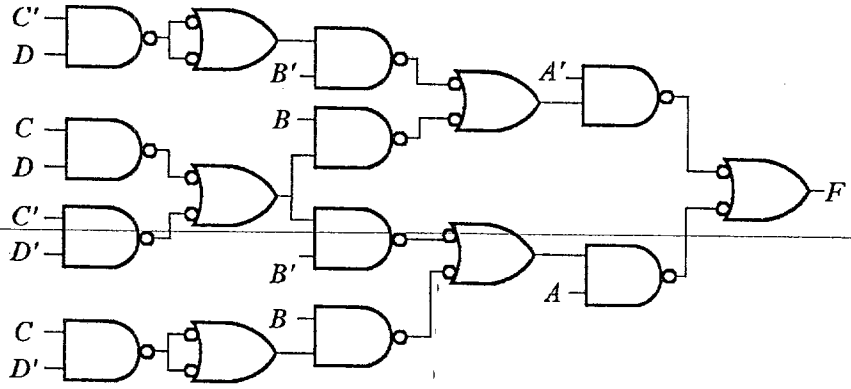
7.28 (b)



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

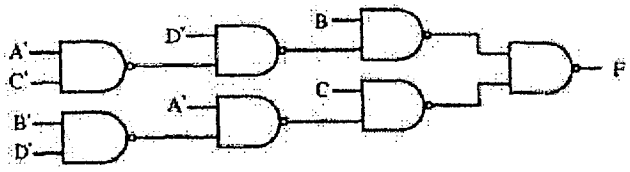
7.28 (c) Many solutions exist. Here is one, drawn with alternate gate symbols.

$F = A'\,(B'C'D' + B'CD + BCD') + A\,(B'C'D + BC'D' + BCD)$
$= A'\,(B'(C'D' + CD) + BCD') + A\,(B(C'D' + CD) + B'C'D)$



7.29 (a) $F = A'BC' + BD + AC + B'CD'$

$= B\,(D + A'C') + C\,(A + B'D')$



C D \ A B	0C	01	11	1C
C0	C	1	0	C
C1	C	1	1	C
11	C	1	1	1
10	1	0	1	1

Many NOR solutions exist. Here is one.

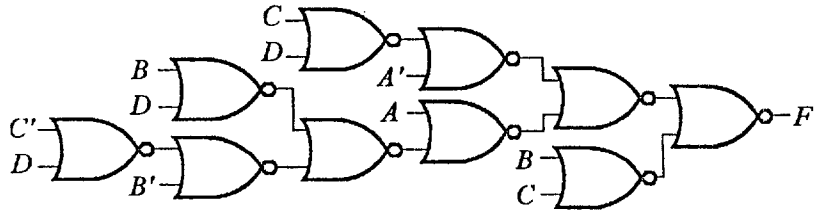
$F = (B + C)\,(A' + C + D)\,(A + B + D')\,(A + B' + C' + D)$

$= (B + C)\,[A + (B + D')\,(B' + C' + D)]\,(A' + C + D)$

$= (B + C)\,[A\,(C + D) + A'\,(B + D')\,(B' + C' + D)]$

$= (B + C)\,[A\,(C + D) + A'\,(B(C' + D) + B'D')]$

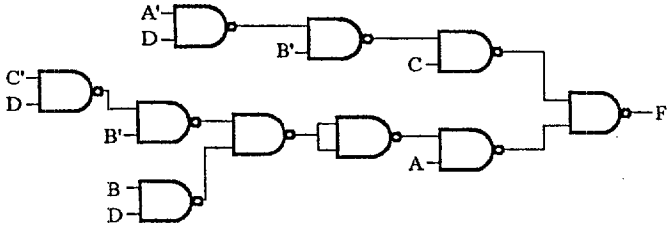
C D \ A B	0C	01	11	1C
C0	C	1	0	C
C1	C	1	1	C
11	C	1	1	1
10	1	0	1	1



7.29 (b)

C D \ A B	C0	01	11	10
00	0	C	1	0
01	0	C	0	1
11	1	1	1	0
10	0	1	1	0

F = A'C D + B C + A B'C'D + A B D'

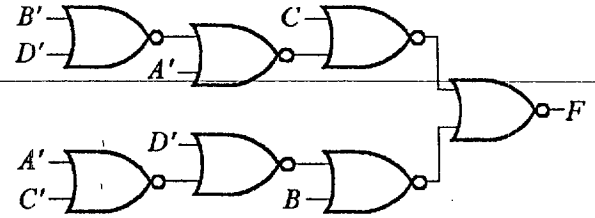


$F = C\,(B + A'D) + A\,(B'C'D + BD')$

$= C\,(B + A'D) + A[(B' + D')\,(B + C'D)]$

C D \ A B	00	C1	11	10
0C	0	0	1	C
01	0	0	C	1
11	1	1	1	C
1C	0	1	1	C

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



$F = (B' + C + D')\,(A' + B + C')\,(B + D)\,(A + C)$

$= (C + A\,(B' + D'))\,(B + D\,(A' + C'))$

7.30

C D \ A B	CC	01	11	10
00	1	1		
01	1	1	1	1
11	1	1	1	1
10	1		1	

F = A'B' + A'C' + D + A B C

$F = \sum m(0, 1, 2, 3, 4, 5, 7, 9, 11, 13, 14, 15)$

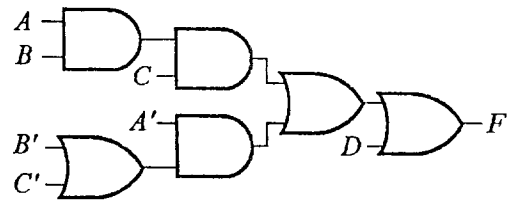
$F = D + A'B' + A'C' + ABC$

$= D + A'\,(B' + C') + ABC$

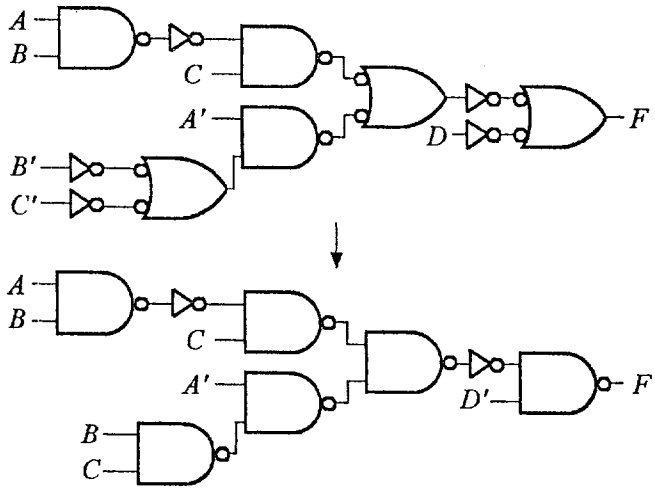
Alternate solution:

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

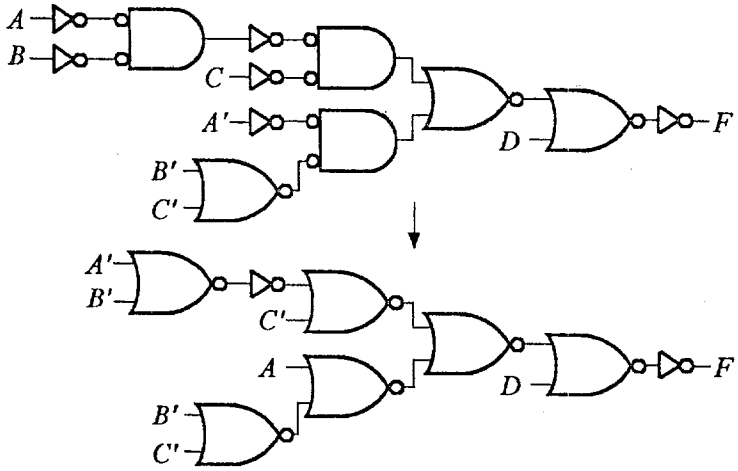
7.30 (a)



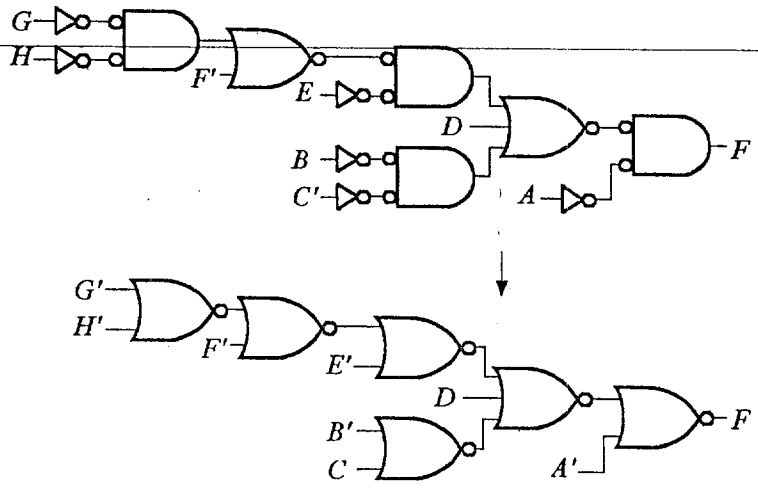
7.30 (b)



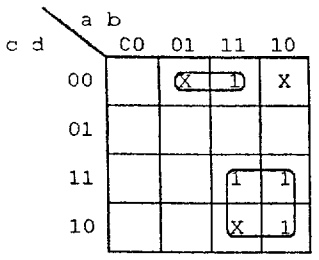
7.30 (c)



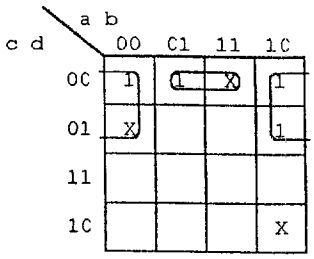
7.31 $Z = A\,[BC' + D + E(F' + GH)]$



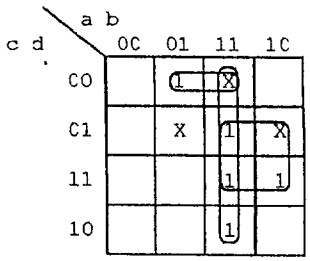
7.32



f1 = b c'd' + a c



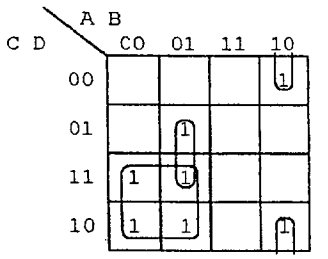
f2 = b'c' + b c'd'



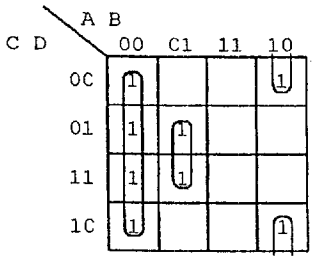
f3 = a b + a d + b c'd'

8 gates

7.33



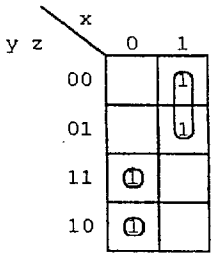
F = A'C + A'B D + A B'D'



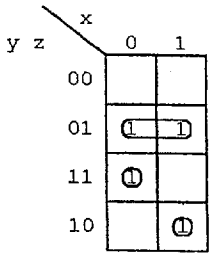
F = A'B' + A'B D + A B'D'

6 gates

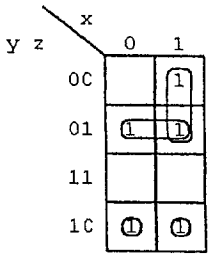
7.34



f1 = x'y z + x' y z' + x y'



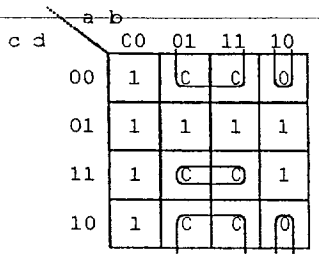
f2 = y' z + x'y z + x y z'



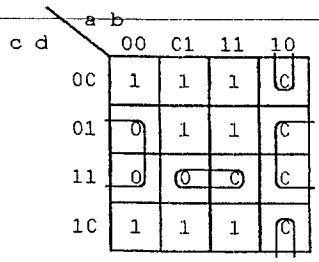
f3 = x y' + y'z + x'y z' + x y z'

8 gates

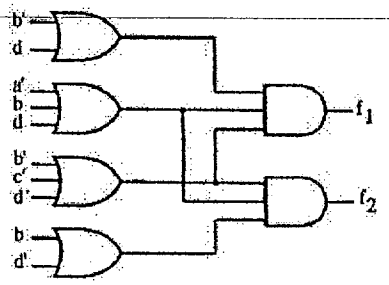
7.35 (a)



f1



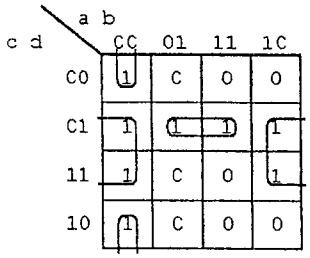
f2



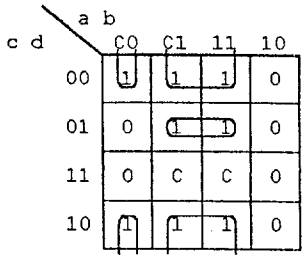
$f_1 = (a' + b + d)\ (b' + c' + d')\ (b' + d)$ — 6 gates

$f_2 = (a' + b + d)\ (b' + c' + d')\ (b + d')$

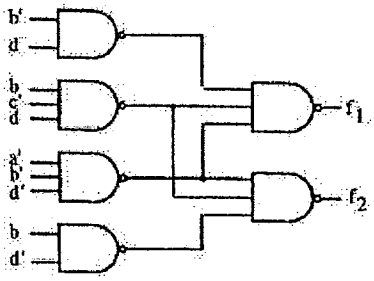
7.35 (b)



f1

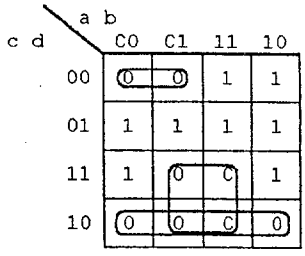


f2

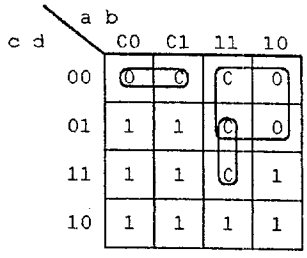


Circle 1's to get sum-of-products expressions:
$f_1 = \underline{bc'd} + \underline{a'b'd'} + b'd$ — 6 gates
$f_2 = \underline{bc'd} + \underline{a'b'd'} + bd'$
Then convert directly to NAND gates.

7.36 (a) Circle 0's

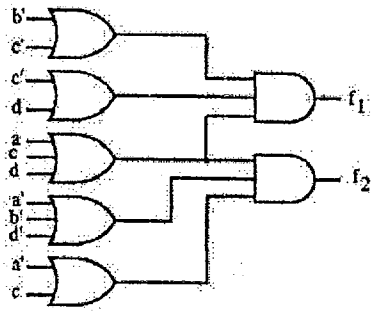


$f_1 = (a + c + d)(b' + c')(c' + d)$



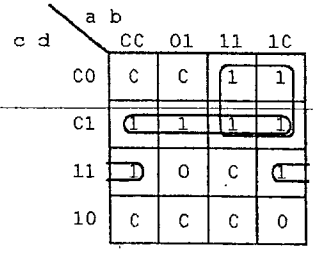
$f_2 = \underline{(a + c + d)}(a' + c)(a' + b' + d')$

7 gates

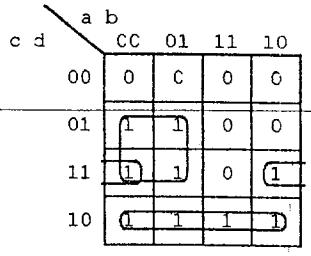


7.36 (b) Circle 1's to get sum-of-products expressions:

Then convert directly to NAND gates

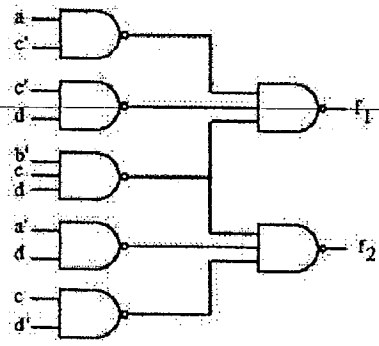


$f_1 = a\,c' + c'd + \underline{b'c\,d}$

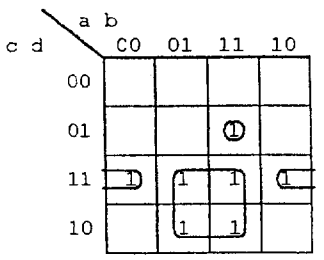


$f_2 = a'd + c\,d' + \underline{b'c\,d}$

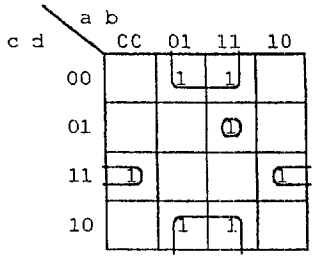
7 gates



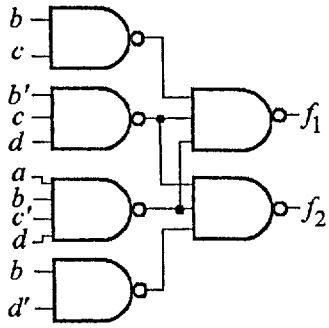
7.37 (a)



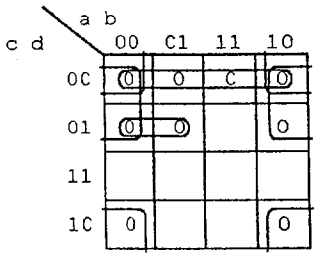
$f_1 = b\,c + \underline{b'c\,d} + \underline{a\,b\,c'd}$



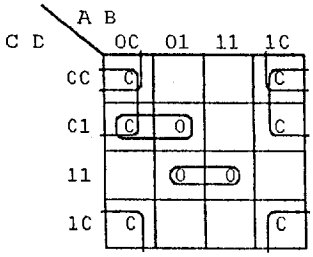
$f_2 = b\,d' + \underline{b'c\,d} + \underline{a\,b\,c'd}$



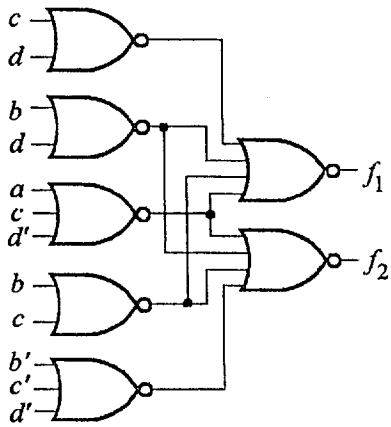
7.37 (b)



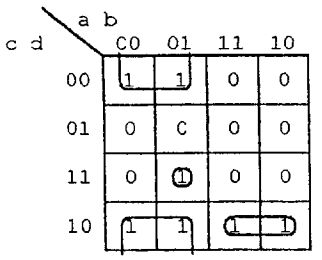
$f_1 = \underline{(b+d)}\,(c+d)\,\underline{(b+c)}\,\underline{\underline{(a+c+d')}}$



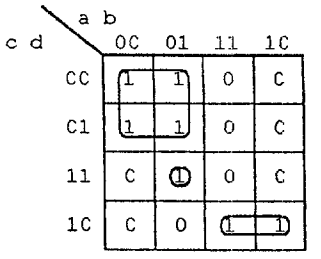
$f_2 = \underline{(b+d)}\,\underline{(b+c)}\,\underline{\underline{(a+c+d')}}\,(b'+c'+d')$



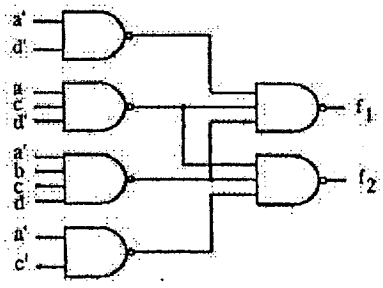
7.38 (a)



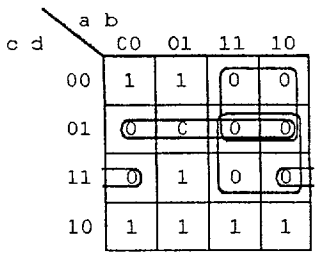
$f1 = a'd' + \underline{a'b\,c\,d} + \underline{\underline{a\,c\,d'}}$



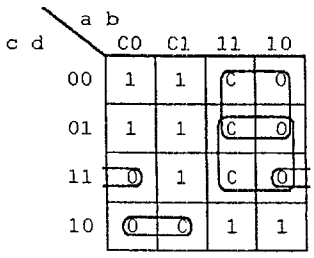
$f2 = a'c' + \underline{a'b\,c\,d} + \underline{\underline{a\,c\,d'}}$



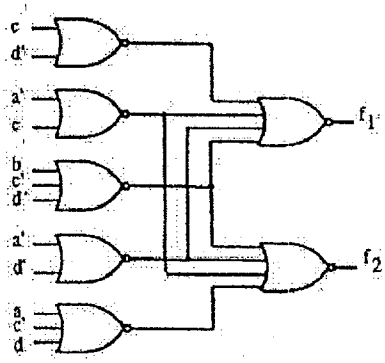
7.38 (b)



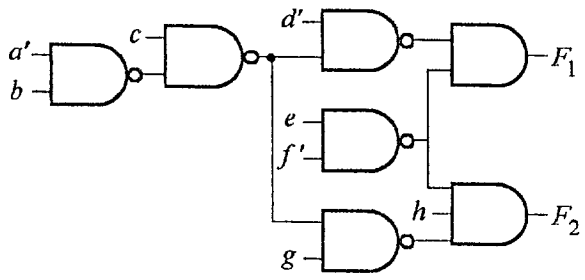
$f1 = \underline{(a'+c)}\,\underline{(a'+d')}\,(c+d')\,\underline{\underline{(b+c'+d')}}$



$f2 = \underline{(a'+c)}\,\underline{(a'+d')}\,\underline{(b+c'+d')}\,(a+c'+d)$



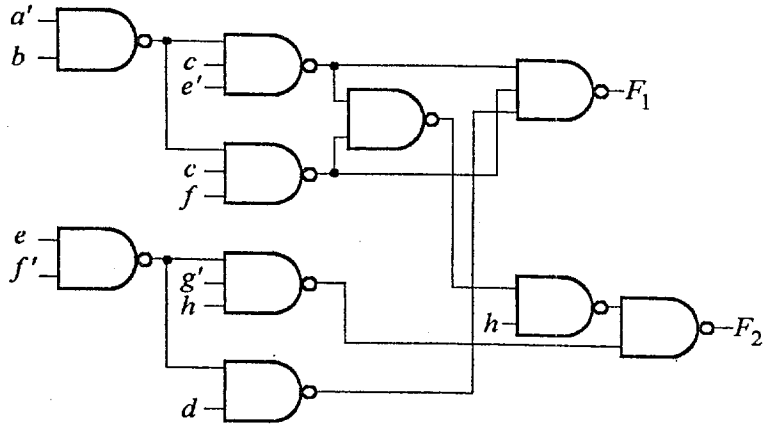
7.39 (a) The circuit consisting of levels 2, 3, and 4 has OR gate outputs. Convert this circuit to NAND gates in the usual way, leaving the AND gates at level 1 unchanged. The result is:



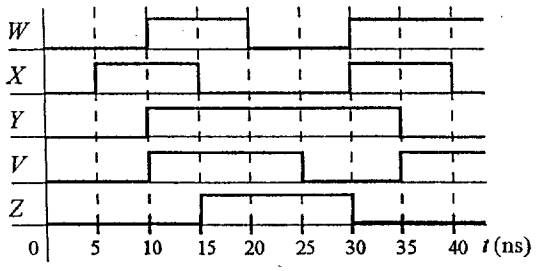
7.39 (b) One solution would be to replace the two AND gates in (a) with NAND gates, and then add inverters at the output. However, the following solution avoids adding inverters at the outputs:

$F_1 = [(a + b')\,c + d]\,(e' + f)$
$= ace' + b'ce' + de' + acf + b'cf + df$
$= \underline{ce'(a + b')} + d\,(e' + f) + \underline{cf(a + b')}$

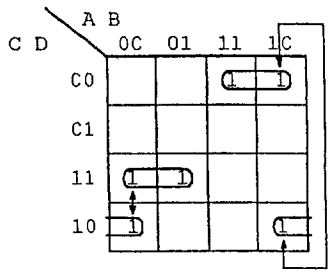
$F_2 = [(a + b')\,c + g']\,(e' + f)\,h$
$= h\,(ace' + b'ce' + acf + b'cf) + g'h\,(e' + f)$
$= h\,[\underline{ce'(a + b')} + \underline{cf(a + b')}] + g'h\,(e' + f)$



8.6



8.7



Z = A'C D + A C'D' + B'C D'

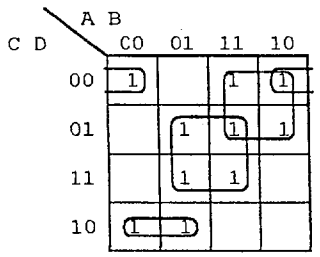
Static 1-hazards lie between 1000↔1010 and 0010↔0011

Without hazards: $Z' = AC'D' + A'CD + B'CD' + A'B'C + AB'D'$

8.8 $A = Z; B = 0; C = Z' = X; D = Z \cdot 0 = 0;$
$E = Z; F = 0 + 0 + X = X; G = (0 \cdot Z)' = 0' = 1;$
$H = (X + 1)' = 1' = 0$

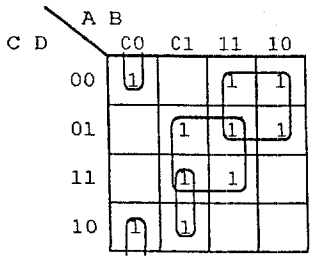
8.9 $A = B = C = 1$, so $F = (A + B' + C')\ (A' + B + C')\ (A' + B' + C) = 1$
But, in the figure, gate 4 outputs F = 0, indicating something is wrong. For the last NOR gate, F = 1 only when all its inputs are 0. But the output of gate 1 is 1. Therefore, gate 4 is working properly, but gate 1 is connected incorrectly or is malfunctioning.

8.10 (a) $F(A, B, C, D) = \sum m(0, 2, 5, 6, 7, 8, 9, 12, 13, 15)$
There are 3 different minimum AND-OR solutions to this problem. The problem asks for any two of these.



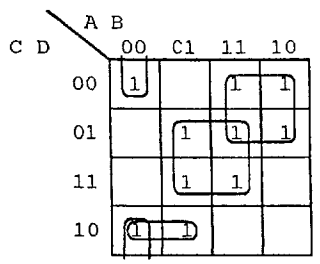
F = B D + A C' + A'C D' + B'C'D'

Solution 1: 1-hazards are between 0000↔0010 and 0111↔0110



F = B D + A C' + A'B'D' + A'B C

Solution 2: 1-hazards are between 0010↔0110 and 0000↔1000

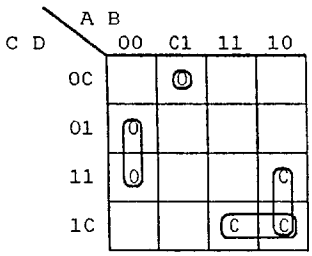


F = B D + A C' + A'B'D' + A'C D'

Solution 3: 1-hazards are between 0111↔0110 and 0000↔1000

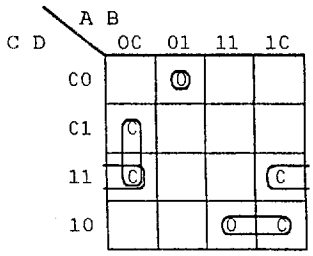
Without hazards:
$F' = BD + AC' + B'C'D' + A'CD' + A'B'D' + A'BC$

8.10 (b)



$F = (A + B + D')\ (A + B' + C + D)\ (A' + C' + D)\ (A' + B + C')$
0-hazard is between 1011↔0011

Either way, without hazard:
$F' = (A + B + D')\ (A + B' + C + D)\ (A' + C' + D)\ (B + C' + D')\ (A' + B + C')$



F = (A + B + D') (A + B' + C + D) (A' + C' + D) (B + C' + D')
0-hazard is between 1011↔1010

Unit 9 Problem Solutions

9.1 See FLD p. 636 for solution.

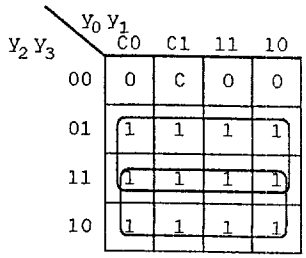
9.2 See FLD p. 636 for solution.

9.3 See FLD p. 637 for solution.

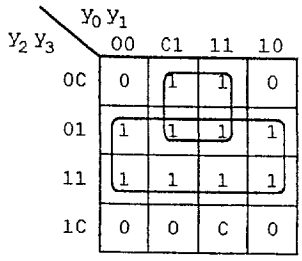
9.4 See FLD p. 637 and Figure 4-4 on FLD p.99.

9.5

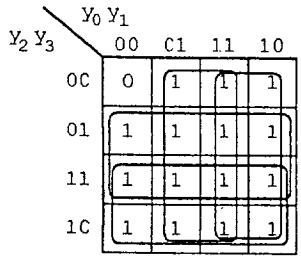
$y_0y_1y_2y_3$	a b c
0 0 0 0	0 0 0
1 0 0 0	0 0 1
X 1 0 0	0 1 1
X X 1 0	1 0 1
X X X 1	1 1 1



$a = y_3 + y_2$



$b = y_3 + y_2' y_1$



$c = y_3 + y_2 + y_1 + y_0$

9.6 See FLD p. 638 for solution.

9.7 See FLD p. 638 for solution.

9.8 See FLD p. 638-639 for solution.

9.9 The equations derived from Table 4-6 on FLD p. 101 are:
$D = x'y'b_{in} + x'yb_{in}' + xy'b_{in}' + xyb_{in}$
$bout = x'b_{in} + x'y + yb_{in}$
See FLD p. 639 for PAL diagram.

9.10 Note: $A_6 = A_4'$ and $A_5 = A_4$. Equations for A_4 through A_0 can be found using Karnaugh maps. See FLD p. 640-641 for answers.

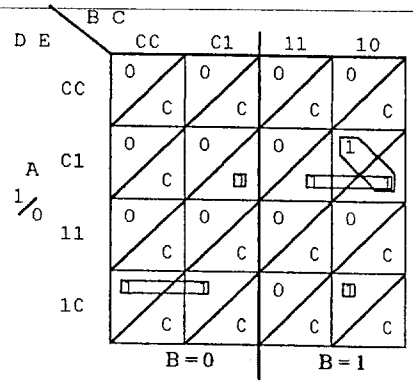
9.11 (a) $F = C'D' + BC' + A'C \rightarrow$ Use 3 AND gates
$F' = [C'D' + BC' + A'C]' = [C' (B + D') + CA']'$
$= [(C + B + D') (A' + C')]'$
$= B'C'D + AC \rightarrow$ Use 2 AND gates

9.11 (b) $F = A'B' + C'D' \rightarrow$ Use 2 AND gates
$F' = (A'B' + C'D')'$
$= [(A' + C') (A' + D') (B' + C') (B' + D')]'$
$= AC + AD + BC + BD \rightarrow$ Use 4 AND gates

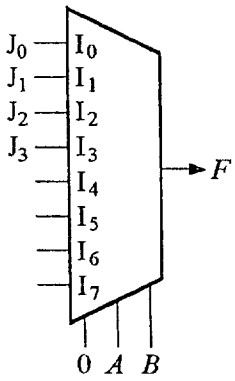
9.12 (a) See FLD p. 641, use the answer for 9.12 (b), but leave off all connections to 1 and 1'.

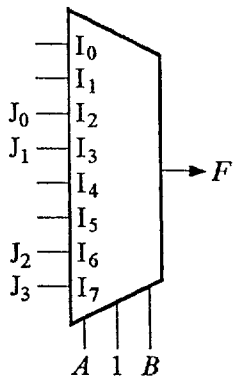
9.12 (b) See FLD p. 641 for solution.

9.13 Using Shannon's expansion theorem:
$F = ab'cde' + bc'd'e + a'cd'e + ac'de'$
$= b' (acde' + a'cd'e + ac'de') + b (c'd'e + a'cd'e + ac'de')$
$= b' [ade' (c + c') + a'cd'e] + b [(c' + a'c) d'e + ac'de']$
$= b' (ade' + a'cd'e) + b (c'd'e + a'd'e + ac'de')$
The same result can be obtained by splitting a Karnaugh map, as shown to the right.



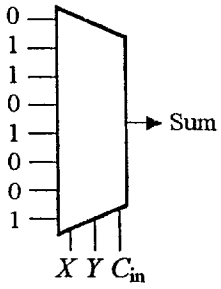
9.14 There are many solutions. For example:

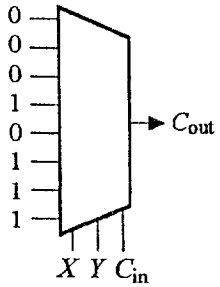




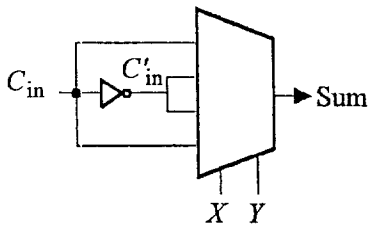
9.15 (a)

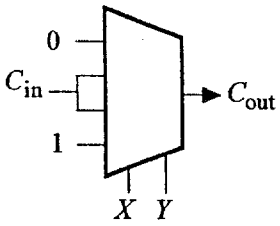
x y c_{in}	Sum C_{out}
0 0 0	0 0
0 0 1	1 0
0 1 0	1 0
0 1 1	0 1
1 0 0	1 0
1 0 1	0 1
1 1 0	0 1
1 1 1	1 1



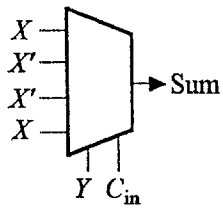


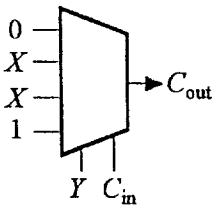
9.15 (b)





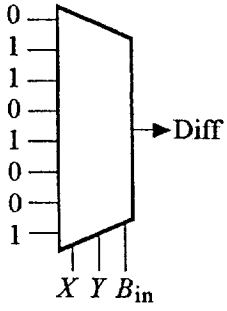
9.15 (c)

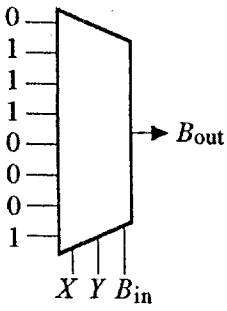




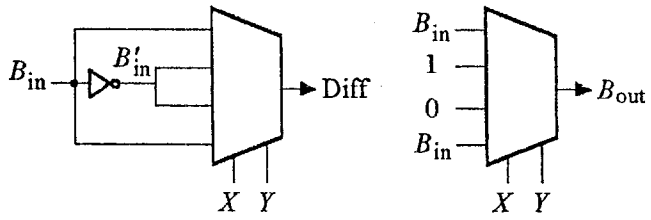
9.16 (a)

x y b_{in}	Diff B_{out}
0 0 0	0 0
0 0 1	1 1
0 1 0	1 1
0 1 1	0 1
1 0 0	1 0
1 0 1	0 0
1 1 0	0 0
1 1 1	1 1

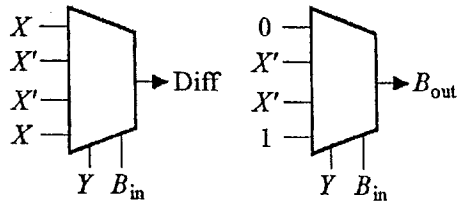




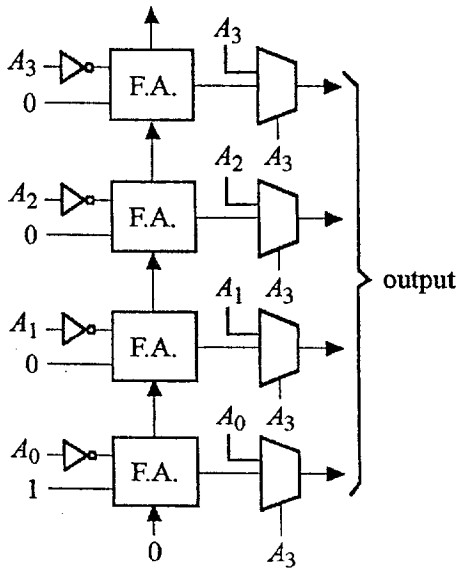
9.16 (b)



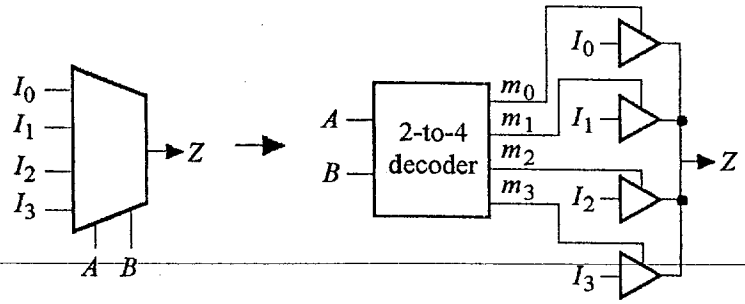
9.16 (c)



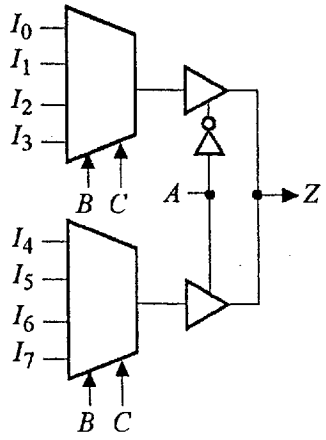
9.17 For a positive number A, |A| = A and for a negative number A, |A| = −A. Therefore, if the number is negative, that is A[3] is 1, then the output should be the 2's complement (that is, invert and add 1) of the input A.



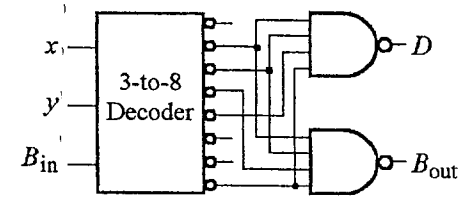
9.18



9.19

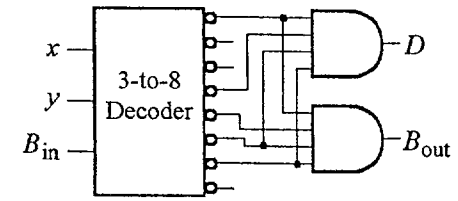


9.20 (a)



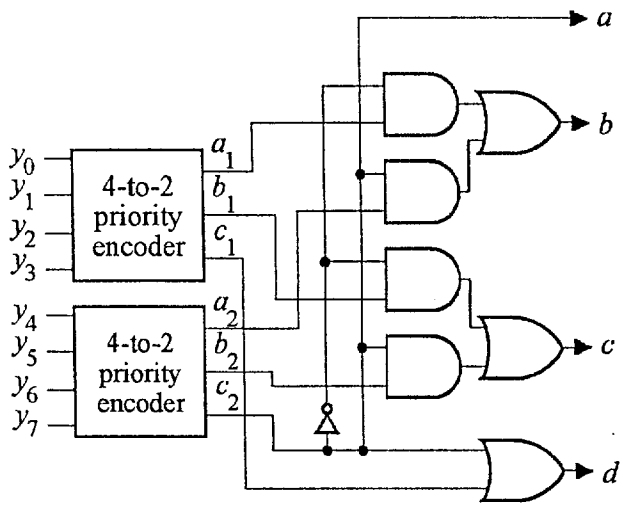
$D = \sum m(1, 2, 4, 7)$; $Bout = \sum m(1, 2, 3, 7)$

9.20 (b)



$D = \prod M(0, 3, 5, 6)$; $B_{out} = \prod M(0, 4, 5, 6)$

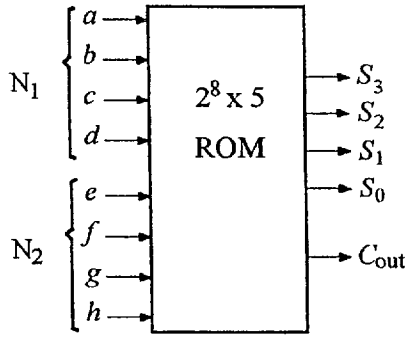
9.21



If any of the inputs y_0 through y_7 is 1, then d of the 8-to-3 decoder should be 1. But in that case, c_1 or c_2 of one of the 4-to-2 decoders will be 1. So $d = c_1 + c_2$.

If one of the inputs y_4, y_5, y_6, and y_7 is 1, then a should be 1, and b and c should correspond to a_2 and b_2, respectively. Otherwise, a should be 0, and b and c should coresond to a_1 and b_1, respectively. So $a = c_2$, $b = c_2a_2 + c_2'a_1$, and $c = c_2b_2 + c_2'b_1$.

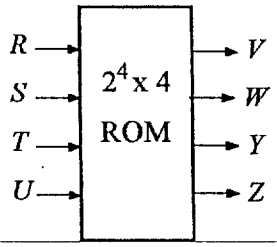
9.22



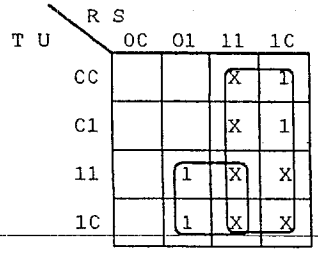
a	b	c	d	e	f	g	h	S_3	S_2	S_1	S_0	C_{out}	Meaning
0	0	0	0	0	0	0	0	X	X	X	X	X	(0000 is a not valid input)
0	0	1	1	0	0	1	1	0	0	1	1	0	(0 + 0 = 0)
0	1	0	1	0	1	1	0	1	0	0	0	0	(2 + 3 = 5)
1	0	1	0	0	1	1	1	0	1	0	0	1	(7 + 4 = 11)

9.23 (a)

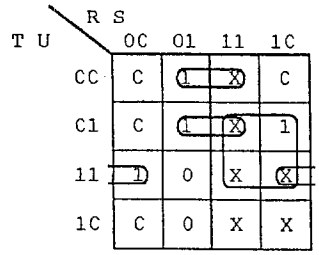
R S T U	V W Y Z
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 0
0 0 1 1	0 1 0 0
0 1 0 0	0 1 0 1
0 1 0 1	0 1 1 0
0 1 1 0	1 0 0 0
0 1 1 1	1 0 0 1
1 0 0 0	1 0 1 0
1 0 0 1	1 1 0 0
1 0 1 0	XXXX
1 0 1 1	XXXX
1 1 0 0	XXXX
1 1 0 1	XXXX
1 1 1 0	XXXX
1 1 1 1	XXXX



9.23 (b)



V = S T + R



W = S'T U + S T'U + R U + S T'U'

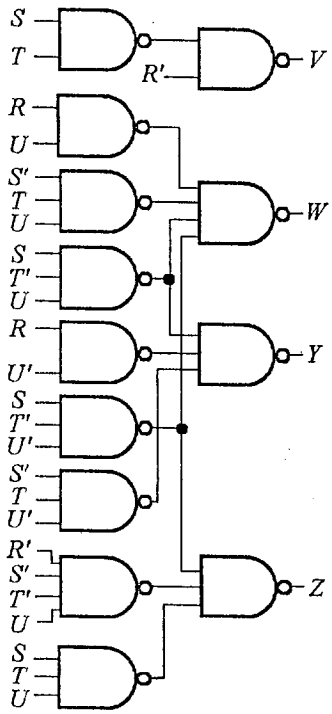
9.23 (b) (contd)

T U \ R S	00	01	11	10
00	0	C	X	1
01	0	1	X	0
11	0	C	X	X
10	1	C	X	X

$Y = S'T U' + \underline{S T'U} + R U'$

T U \ R S	00	01	11	10
00	0	1	X	0
01	1	C	X	0
11	0	1	X	X
10	0	C	X	X

$Z = R'S'TU + \underline{S TU'} + S TU$



9.23 (c)

R S T U	*V W Y Z*
- 1 1 -	1 0 0 0
1 0 0 -	1 0 0 0
1 - - 0	0 0 1 0
1 - - 1	0 1 0 0
- 0 1 1	0 1 0 0
- 1 0 1	0 1 1 0
- 1 0 0	0 1 0 1
- 0 1 0	0 0 1 0
0 0 0 1	0 0 0 1
- 1 1 1	0 0 0 1

9.24 (a)

R S T U	*V W Y Z*
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 1 0 0
0 0 1 1	0 0 1 0
0 1 0 0	X X X X
0 1 0 1	X X X X
0 1 1 0	0 1 0 1
0 1 1 1	X X X X
1 0 0 0	1 1 0 0
1 0 0 1	1 0 1 0
1 0 1 0	1 0 0 0
1 0 1 1	1 0 0 1
1 1 0 0	X X X X
1 1 0 1	X X X X
1 1 1 0	0 1 1 0
1 1 1 1	X X X X

9.24 (b)

T U \ R S	CC	01	11	10
00	0	X	X	1
01	0	X	X	1
11	0	X	X	1
10	0	C	0	1

V = RS'

T U \ R S	CC	01	11	1C
CO	C	X	X	1
C1	C	X	X	0
11	C	X	X	0
10	1	1	1	0

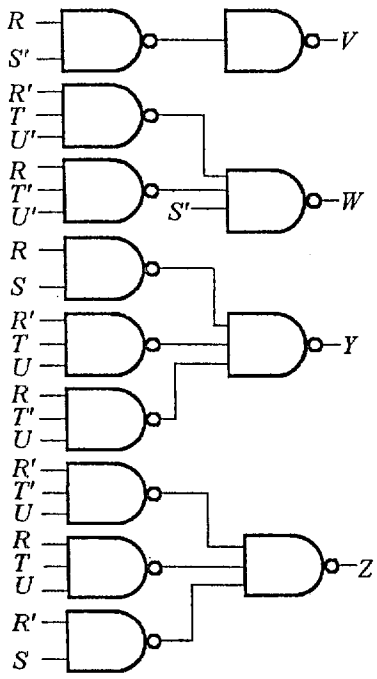
W = R'TU' + RTU' + S

T U \ R S	CC	01	11	10
00	0	X	X	0
01	0	X	X	1
11	1	X	X	0
10	0	C	1	0

Y = R'TU + RTU + RS

T U \ R S	OC	01	11	1C
CC	C	X	X	C
C1	1	X	X	C
11	C	X	X	1
1C	C	1	0	C

Z = R'TU + R'S + RTU



9.24 (c)

RSTU	VWYZ
10--	1000
-1--	0100
0-10	0100
1-00	0100
11--	0010
0-11	0010
1-01	0010
01--	0001
0-01	0001
1-11	0001

or

RSTU	VWYZ
10--	1000
01--	0101
11--	0110
0-10	0100
1-00	0100
0-11	0010
1-01	0010
0-01	0001
1-11	0001

9.25 (a)

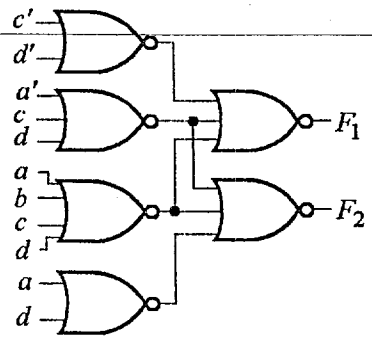
C D \ A B	OC	01	11	1C
CC	0	1	1	1
C1	1	1	0	0
11	0	0	0	0
1C	1	1	1	1

$F_1 = (A + B + C + D)(A' + C + D')(C' + D')$

9.25 (a) (contd)

C D \ A B	CC	01	11	10
00	0	1	1	1
01	0	0	0	0
11	0	0	1	1
10	1	1	1	1

$F_2 = (A + B + C + D)(A' + C + D')(A + D')$



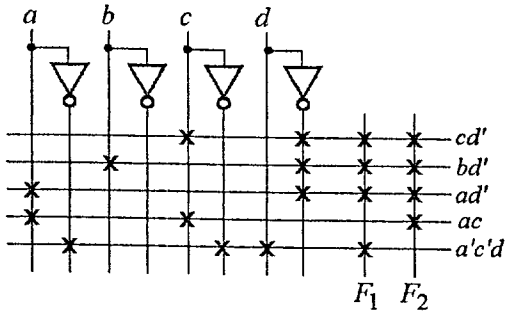
Alternate solution:

$F_1 = (a + b + c + d)\ (a + c' + d')\ (a' + d')$

$F_2 = (a + b + c + d)\ (a + b' + d')\ (c + d')$

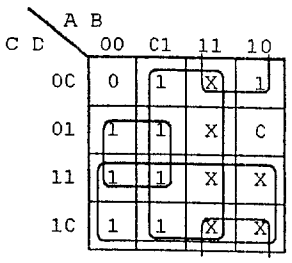
9.25 (b)

	$a\ b\ c\ d$	F_1F_2
(cd')	- - 1 0	1 1
(bd')	- 1 - 0	1 1
(ad')	1 - - 0	1 1
(ac)	1 - 1 -	0 1
$(a'c'd)$	0 - 0 1	1 0

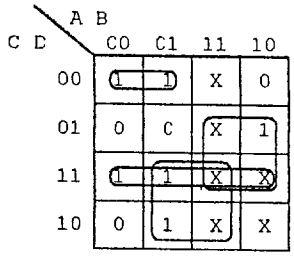


9.26 (a)

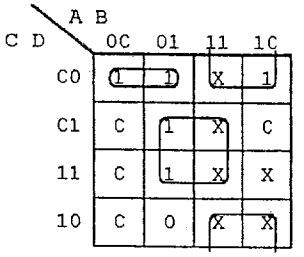
$A\,B\,C\,D$	$W\,X\,Y\,Z$
0 0 0 0	0 1 1 1
0 0 0 1	1 0 0 0
0 0 1 0	1 0 0 1
0 0 1 1	1 1 0 0
0 1 0 0	1 1 1 0
0 1 0 1	1 0 1 0
0 1 1 0	1 1 0 1
0 1 1 1	1 1 1 1
1 0 0 0	1 0 1 1
1 0 0 1	0 1 0 1



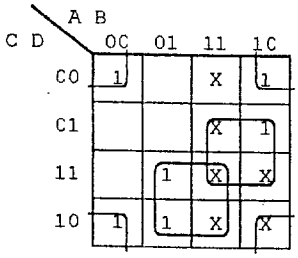
W = A'D + C + B + A D'



X = A'C'D' + C D + A D + B C

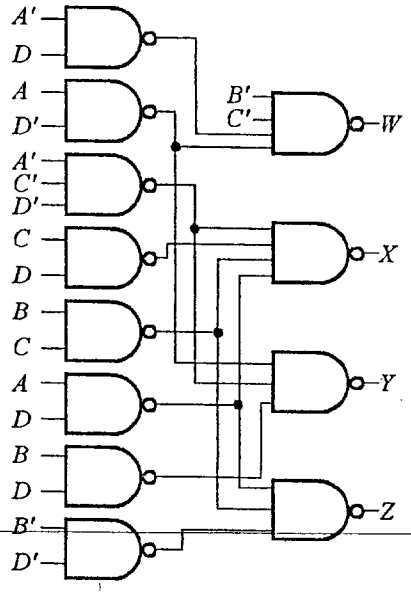


Y = A D' + B D + A'C'D'



Z = A D + B C + B'D'

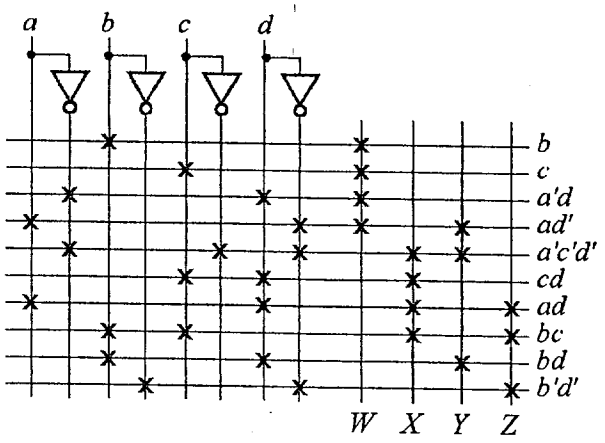
Alt: Z = A + B C + B'D'



9.26 (b)

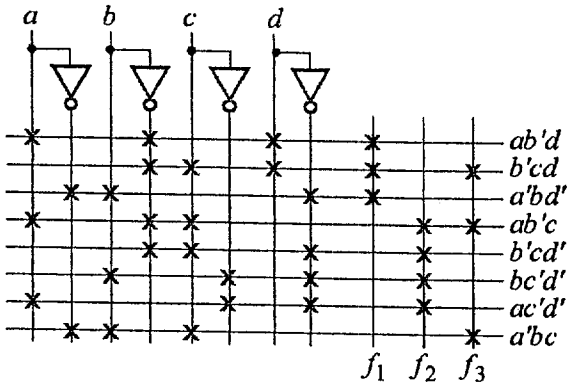
$a\ b\ c\ d$	$W\,X\,Y\,Z$
- 1 - -	1 0 0 0
- - 1 -	1 0 0 0
0 - - 1	1 0 0 0
1 - - 0	1 0 1 0
0 - 0 0	0 1 1 0
- - 1 1	0 1 0 0
1 - - 1	0 1 0 1
- 1 1 -	0 1 0 1
- 1 - 1	0 0 1 0
- 0 - 0	0 0 0 1

9.26 (c)



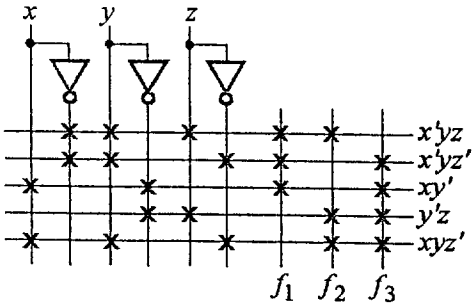
9.27 (a) See solution for 7.10

$a\ b\ c\ d$	$f_1\ f_2\ f_3$
1 0 - 1	1 0 0
- 0 1 1	1 0 1
0 1 - 0	1 0 0
1 0 1 -	0 1 1
- 0 1 0	0 1 0
- 1 0 0	0 1 0
1 - 0 0	0 1 0
0 1 1 -	0 0 1



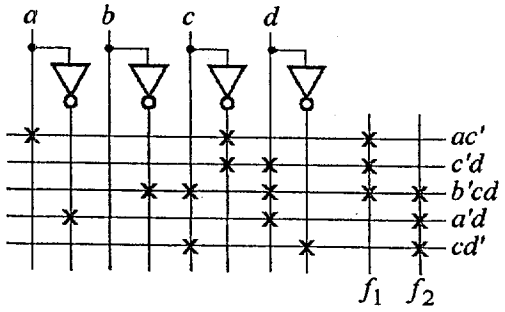
9.27 (b) See solution for 7.34

$x\ y\ z$	$f_1\ f_2\ f_3$
0 1 1	1 1 0
0 1 0	1 0 1
1 0 -	1 0 1
- 0 1	0 1 1
1 1 0	0 1 1

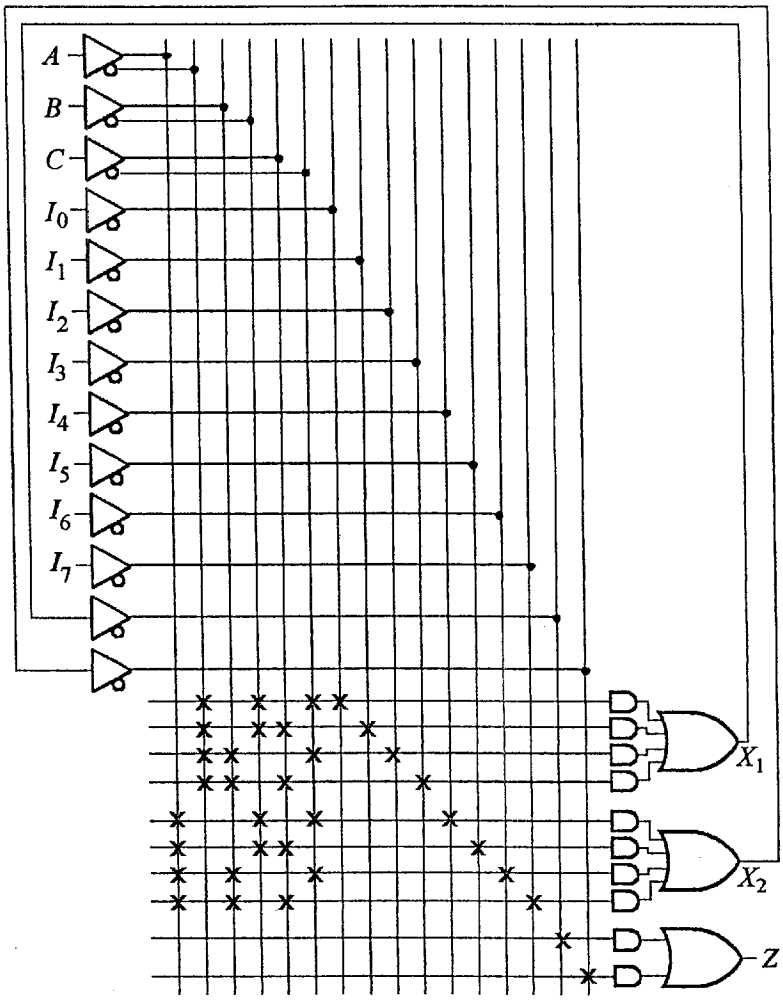


9.27 (c) Because a PLA works with a sum-of-products expression, see solution for 7.36(b), not (a).

$a\ b\ c\ d$	$f_1\ f_2$
1 - 0 -	1 0
- - 0 1	1 0
- 0 1 1	1 1
0 - - 1	0 1
- - 1 0	0 1



9.28 $Z = I_0A'B'C' + I_1A'B'C + I_2A'BC' + I_3A'BC + I_4AB'C' + I_5AB'C + I_6ABC' + I_7ABC$
$= X_1A' + X_2A$ where $X_1 = I_0B'C' + I_1B'C + I_2BC' + I_3BC$ and $X_2 = I_4B'C' + I_5B'C + I_6BC' + I_7BC$



Note: Unused inputs, outputs, and wires have been omitted from this diagram.

9.29 For an 8-to-3 encoder, using the truth table given in FLD Figure 9-16, we get

$a = y_4 + y_5 + y_6 + y_7$

$b = y_2y_3'y_4'y_5'y_6'y_7' + y_3y_4'y_5'y_6'y_7' + y_6y_7' + y_7$

$c = y_1y_2'y_3'y_4'y_5'y_6'y_7' + y_3y_4'y_5'y_6'y_7' + y_5y_6'y_7' + y_7$

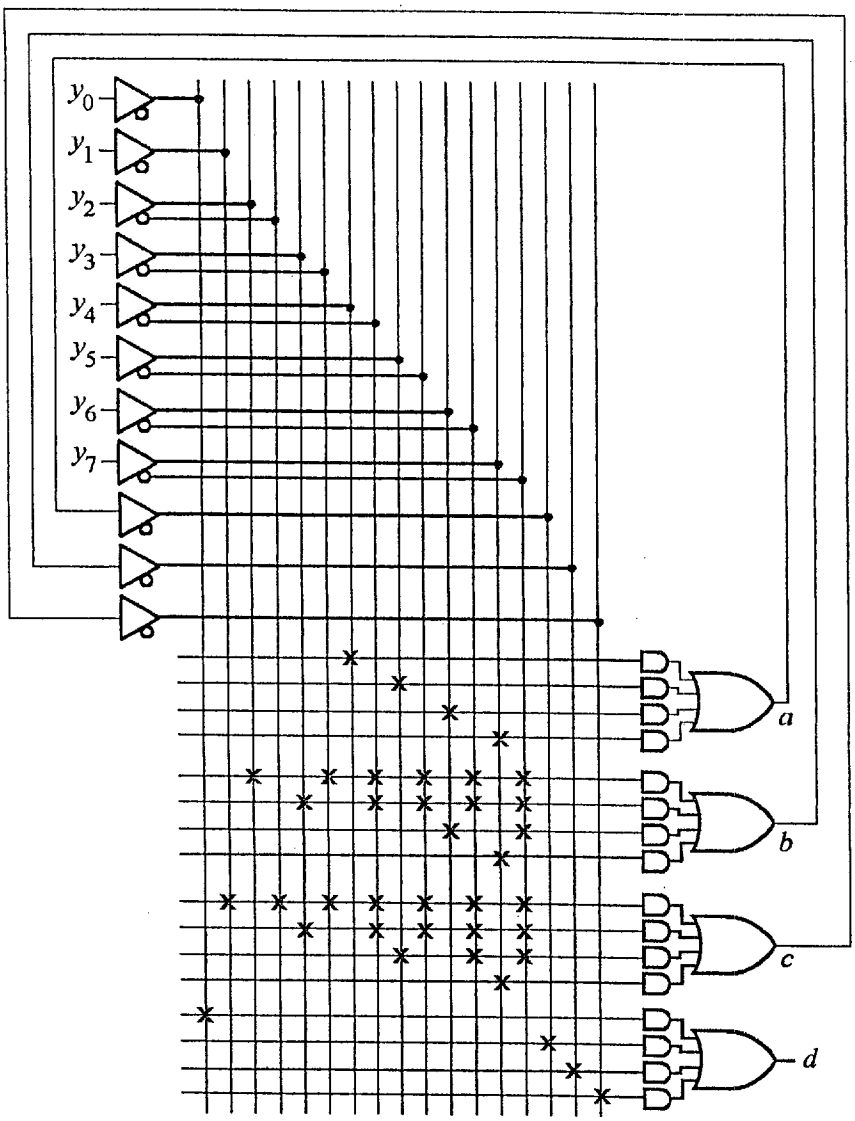
$d = a + b + c + y_0$

Alternative solution for simplified expressions:

$b = y_2y_4'y_5' + y_3y_4'y_5' + y_6 + y_7$

$c = y_1y_2'y_4'y_6' + y_3y_4'y_6' + y_5y_6' + y_7$

9.29 (contd)



Note: Unused inputs, outputs, and wires have been omitted from this diagram.

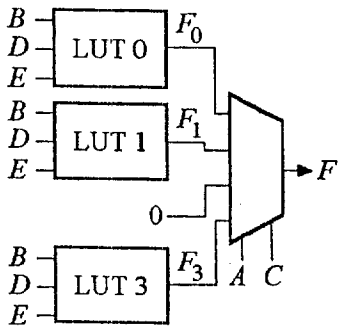
9.30 $F = CD'E + CDE + A'D'E + A'B'DE' + BCD$

9.30 (a) $F = A'B'(CD'E + CDE + D'E + DE') + A'B\,(CD'E + CDE + D'E + CD) + AB'\,(CD'E + CDE) + AB\,(CD'E + CDE + CD)$

9.30 (b) $F = B'C'\,(A'D'E + A'DE') + B'C\,(D'E + DE + A'D'E + A'DE') + BC'\,(A'D'E) + BC\,(D'E + DE + A'D'E + D)$

9.30 (c) $F = A'C'\,(D'E + B'DE') + A'C\,(D'E + DE + D'E + B'DE' + BD) + AC'(0) + AC(D'E + DE + BD)$

9.30 (d) Use the expansion about A and C
$F = A'C'(F_0) + A'C(F_1) + AC(F_3)$
where F_0, F_1, F_3 are implemented in lookup tables:



BDE	$F_0F_1F_3$
0 0 0	0 0 0
0 0 1	1 1 1
0 1 0	1 1 0
0 1 1	0 1 1
1 0 0	0 0 0
1 0 1	1 1 1
1 1 0	0 1 1
1 1 1	0 1 1

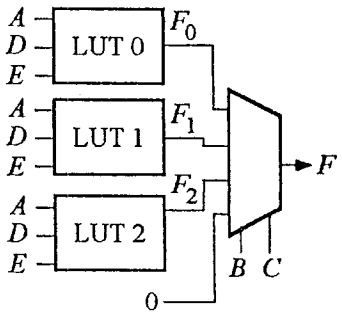
9.31 $F = B'D'E' + AB'C + C'DE' + A'BC'D$

9.31 (a) $F = A'B'(D'E' + C'DE') + A'B(C'DE' + C'D) + AB'(D'E' + C + C'D) + AB(C'DE')$

9.31 (b) $F = B'C'(D'E' + DE') + B'C(D'E' + A) + BC'(DE' + A'D) + BC(0)$

9.31 (c) $F = A'C'(B'D'E' + DE' + BD) + A'C(B'D'E') + AC'(B'D'E' + DE') + AC(B'D'E' + B')$

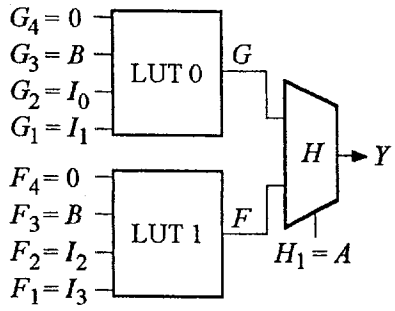
In this case, use the expansion about B and C to implement the function in 3 LUTs:
$F = B'C'(F_0) + B'C(F_1) + BC'(F_2) + BC(0)$
Here we use the LUTs to implement F_0, F_1, F_2 which are functions of A, D, E



A D E	$F_0 F_1 F_2$
0 0 0	1 1 0
0 0 1	0 0 0
0 1 0	1 0 1
0 1 1	0 0 1
1 0 0	1 1 0
1 0 1	0 1 0
1 1 0	1 1 1
1 1 1	0 1 0

9.32 For a 4-to-1 MUX:
$Y = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3$
$= A'(B'I_0 + BI_1) + A(B'I_2 + BI_3)$
$= A'G + AF$, where $G = B'I_0 + BI_1$; $F = B'I_2 + BI_3$

Set programmable MUX so that Y is the output of MUX H.



Unit 10 Problem Solutions

10.1 *See FLD p. 642 for solution.*

10.2 *See FLD p. 642 for solution.*

10.3 *See FLD p. 643 for solution.*

10.4 *See FLD p. 643 for solution.*

10.5 *See FLD p. 643 for solution.*
Notes: The function vec2int is found in bit_pack, which is in the library bitlib, so the following declarations are needed to use vec2int:

```
library bitlib;
use bitlib.bit_pack.all;
```

If std_logic is used instead of bits, then the index can be computed as:

index <= conv_integer(A&B&Cin); where A, B, and Cin are std_logic.

conv_integer is found in the std_logic_arith package.

10.6 *See FLD p. 643 for solution.*

10.7 *See FLD p. 644 for solution.*
Notes: In line 8, "00"&a converts a to a 18-bit std_logic_vector. The overloaded "+" operators automatically extend b, c, and d to 18 bits so that the sum is 18 bits. In line 9, sum(17 downto 2) drops the lower 2 bits of sum, which effectively divides by 4 to give the average. Adding sum(1) rounds up the value of f if sum(1) = 1.

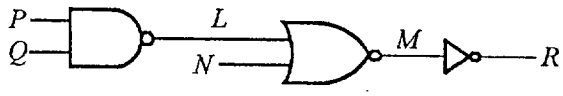
10.8 *See FLD p. 644 for solution.*

Add the following to the answer given on FLD p. 644:

```
Addout <= '0' & E + Bus;
Sum <= Addout(3 downto 0);
Cout <= Addout(4);
```

10.9 *See FLD p. 644 for solution.*

10.10 The network represented by the given code is:



(1) Statement (a) will execute as soon as either P or Q change. Hence, it will *execute* at 4 ns.

(2) Since the NAND gate has a delay of 10 ns, L will be *updated* at 14 ns.

(3) Statement (c) will execute when the value M changes. It will *execute* at 19 ns.

(4) R will be *updated* at 19 + Δ ns, since Δ is the default delay time when no delay is explicitly specified.

10.11 (a)

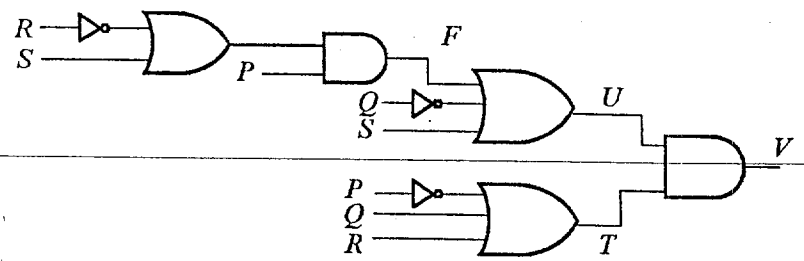
```
H <= not A nand B nor not D nand E;
```

(*Note:* **not** happens first, then it proceeds from left to right)

10.11(b)

```
AN <= not A after 5 ns;
C <= AN nand B after 10ns;
F <= not D after 5ns;
G <= C nor F after 15ns;
H <= G nand E after 10ns;
```

10.12



10.13 L = X (Since 1 and 0 in the resolution function yields X)
M = 0
N = 1 (1 overrides Z in the resolution function)

10.14 (a) The expression can be rewritten as:

```
F <= (((not E) & "011") or "000100") and (not D);
```

Evaluating in this order, we get:
F = 000110

10.14 (b) LHS: **not**("101" & "011") = "010100"
RHS: ("100" & "101" **and** "010" & "101") = "000101"
Since LHS > RHS, the expression evaluates to FALSE

10.15
```
library bitlib;
use bitlib.bit_pack.all;

entity myrom is
    port (A, B, C, D: in bit; W, X, Y: out bit);
end myrom;

architecture table of myrom is
type ROM16_3 is array(0 to 15)of bit_vector(0 to 2);
constant ROM1: ROM16_3 := ("010", "111", "100", "110", "011", "110",
    "001", "000", "000", "111", "100", "010", "001", "100", "101", "000");
signal index: integer range 0 to 15;
signal temp: bit_vector(0 to 2);
begin
    index <= vec2int(A&B&C&D);
    temp <= ROM1(index);
    W <= temp(0);
    X <= temp(1);
    Y <= temp(2);
end table;
```

10.16 (a)
```
databus <= membus when mRead = '1'
    else "ZZZZZZZZ";
databus <= probus when mWrite = '1'
    else "ZZZZZZZZ";
```

10.16 (b) The value will be determined by the std_logic resolution function. For example, if membus = "01010101" and probus = "00001111", then databus = "0X0XX1X1"

10.17 (a)
```
with C&D select
    F <= not A after 15ns when "00",
    B after 15ns when "01",
    not B after 15ns when "10",
    '0' after 15ns when "11";
```

10.17 (b)
```
F <= not A after 15ns when C&D = "00"
    else B after 15ns when C&D = "01"
    else not B after 15ns when C&D = "10"
    else '0' after 15ns;
```

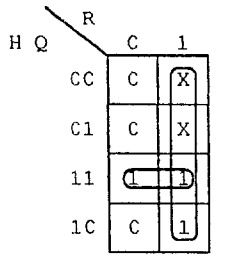
10.18 (a)
```
entity mynand is
    port(X, Y: in bit; Z: out bit);
end mynand;

architecture eqn of mynand is
begin
    Z <= X nand Y after 4 ns;
end eqn;
```

10.18 (b)
```
entity main is
    port(A, B, C, D: in bit; F: out bit);
end main;

architecture eqn of main is
component mynand is
    port(X, Y: in bit; Z: out bit);
end component;

signal E, G: bit;
begin
    n1: mynand port map(A, B, E);
    n2: mynand port map(C, D, G);
    n3: mynand port map(E, G, F);
end eqn;
```

Unit 11 Problem Solutions

11.1 Z responds to X and to Y after 10 ns; Y responds to Z after 5 ns. See FLD p. 646 for answer.

11.2 See FLD p. 646 for solution. For part (b), also use the following Karnaugh map. Don't cares come from the restriction in part (a).



$Q^+ = R + HQ$

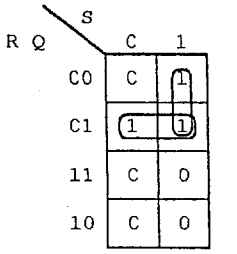
11.3 P and Q will oscillate. See FLD p. 646 for timing chart.

11.4 See FLD p. 647 for solution.

11.5 See FLD p. 647 for solution.

11.6 (a)

SRQ	Q^+
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	1



$Q^+ = R'Q + S R'$

11.6 (b) See FLD p. 647 for solution.

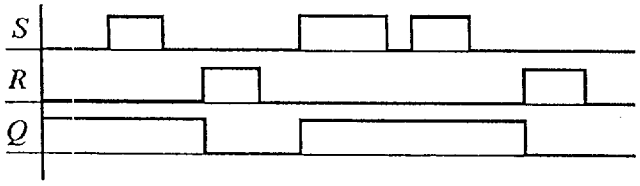
11.7 See FLD p. 647 for solution.

11.8 See FLD p. 647 for solution.

11.9 See FLD p. 648 for solution.

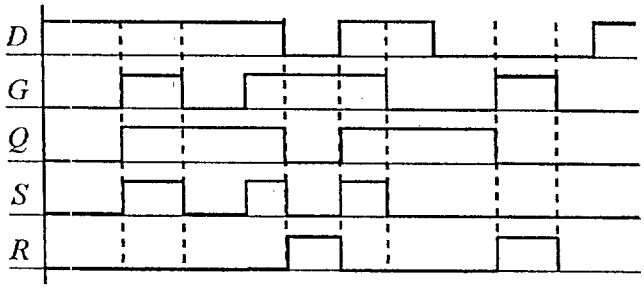
11.10 See FLD p. 648 for solution.

11.11

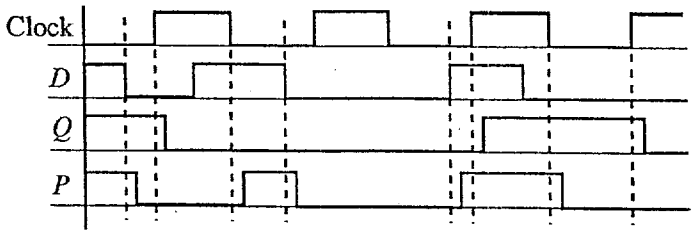


11.12 For every input/state combination with the condition $SR = 0$ holding, each circuit obeys the next-state equation $Q^+ = S + R'Q$. When $S = R = 1$, in (a), both outputs are 1, and in (b), the latch holds its state.

11.13

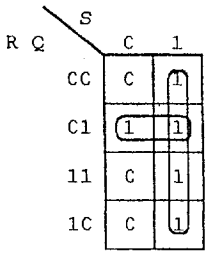


11.14



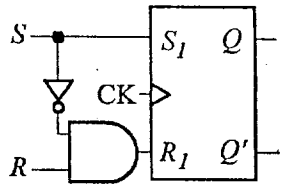
11.15 (a)

S R Q	Q^+
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	1
1 1 1	1

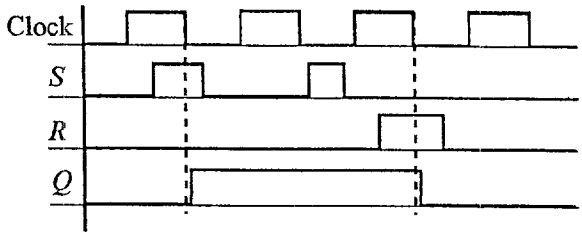


$Q^+ = S + R'Q$

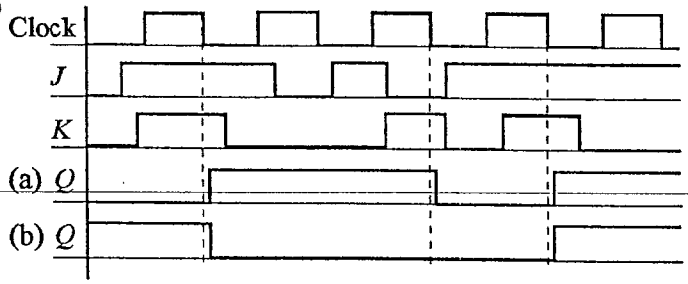
11.15 (b) A set-dominant FF from an S-R FF—The arrangement will ensure that when $S = R = 1$, $S_1 = 1$, $R_1 = 0$, and $Q^+ = 1$.



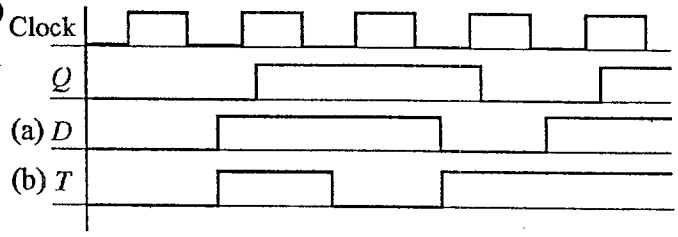
11.16



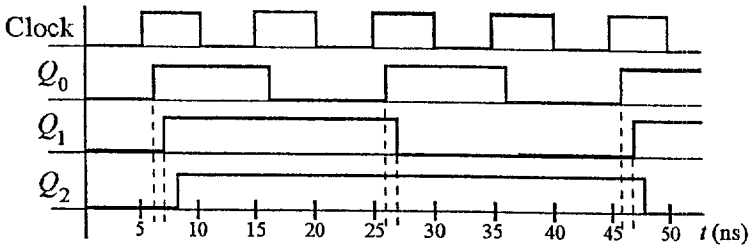
11.17 (a)-(b)



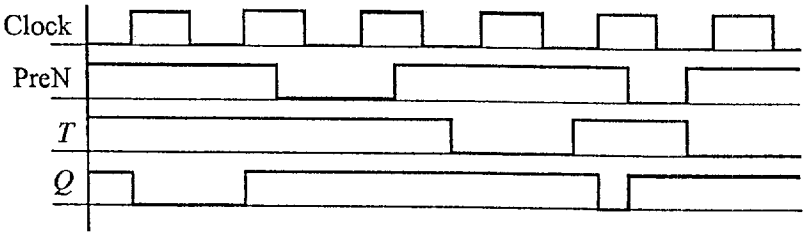
11.18 (a)-(b)



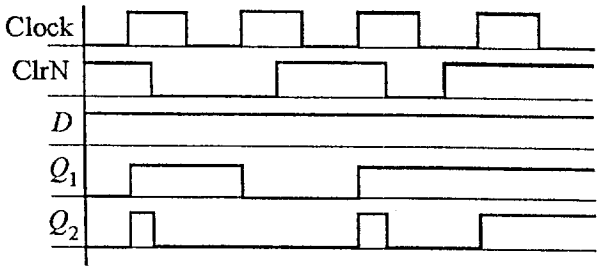
11.19



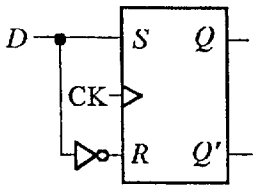
11.20



11.21



11.22 (a)

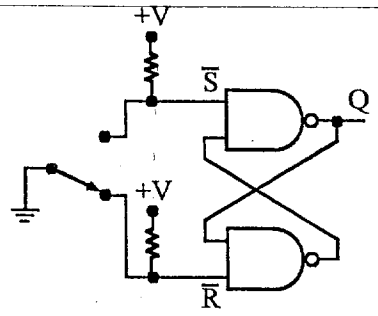


When $D = 0$, then $S = 0$, and $R = 1$, so $Q^+ = 0$.
When $D = 1$, then $S = 1$, and $R = 0$, so $Q^+ = 1$.

11.22 (b) R will not be ready until D goes through the inverter, so we must add the delay of the inverter to the setup time:
Setup time = 1.5 + 1 = 2.5 ns

Propagation delay for the DFF:
2.5 ns (same as for the S-R flip-flop, since the propagation delay is measured with respect to the clock)

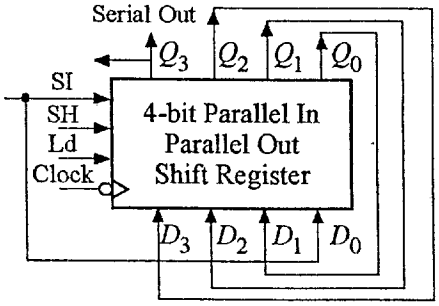
11.23



Unit 12 Problem Solutions

12.1 Consider $3 \times Y = Y + Y + Y$, that is, we need to add Y to itself 3 times. First, clear the accumulator before the first rising clock edge so that the X-register is 000000. Let the *Ad* pulse be 1 for 3 rising clock edges and let the Y register contain the desired number ($y_5y_4y_3y_2y_1y_0$) which is to be added three times. The timing diagram is on FLD p. 650. *Note: ClrN* should go to 0 and back to 1 before the first rising clock edge. *Ad* should be 1 before the same clock edge. However, it does not matter in what order, that is, *Ad* could go to 1 before *ClrN* returns to 1, or even before it goes to 0.

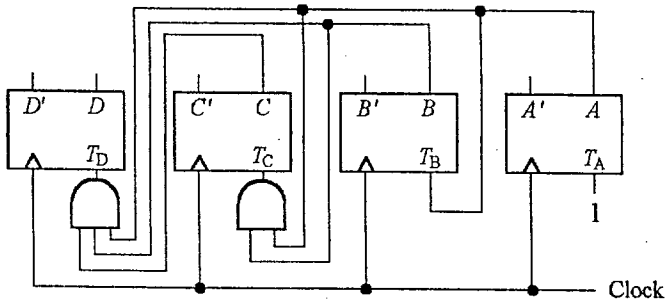
12.2 Serial input connected to D_0 for left shift.
Sh = 0, *L* = 1 causes a left shift.
Sh = 1, *L* = 1 or 0 causes a right shift



12.3 See FLD p. 650 for solution.

12.4 (a)

Present State $DCBA$	Next State $D^+C^+B^+A^+$	Flip-Flop Inputs $T_DT_CT_BT_A$
0 0 0 0	0 0 0 1	0 0 0 1
0 0 0 1	0 0 1 0	0 0 1 1
0 0 1 0	0 0 1 1	0 0 0 1
0 0 1 1	0 1 0 0	0 1 1 1
0 1 0 0	0 1 0 1	0 0 0 1
0 1 0 1	0 1 1 0	0 0 1 1
0 1 1 0	0 1 1 1	0 0 0 1
0 1 1 1	1 0 0 0	1 1 1 1
1 0 0 0	1 0 0 1	0 0 0 1
1 0 0 1	1 0 1 0	0 0 1 1
1 0 1 0	1 0 1 1	0 0 0 1
1 0 1 1	1 1 0 0	0 1 1 1
1 1 0 0	1 1 0 1	0 0 0 1
1 1 0 1	1 1 1 0	0 0 1 1
1 1 1 0	1 1 1 1	0 0 0 1
1 1 1 1	0 0 0 0	1 1 1 1



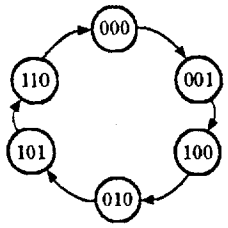
As explained in Section 12.3, it can be seen that A changes on every pulse: $T_A = 1$
B changes only when $A = 1$: $T_B = A$
C changes only when both B and $A = 1$: $T_C = AB$
D changes only when A, B, and $C = 1$: $T_D = ABC$

12.4 (b) The binary counter using D flip-flops is obtained by converting each T flip-flop to a D flip-flop by adding an XOR gate.

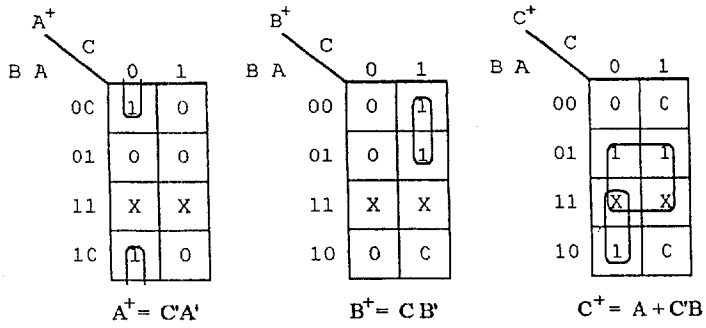
See FLD p. 650 and Figure 12-15 on FLD p. 335.

12.5 Equations for C, B, and A are from Equations (12-2) on FLD p. 335. Beginning with (b) of Problem 12.4 solutions,
$D^+ = D \oplus CBA = D'CBA + D(CBA)'$
$= D'CBA + D(C' + B' + A')$
$= D'CBA + DC' + DB' + DA'$

12.6 In the following state graph, the first flip-flop (C) takes on the required sequence 0, 0, 1, 0, 1, 1, (repeat).

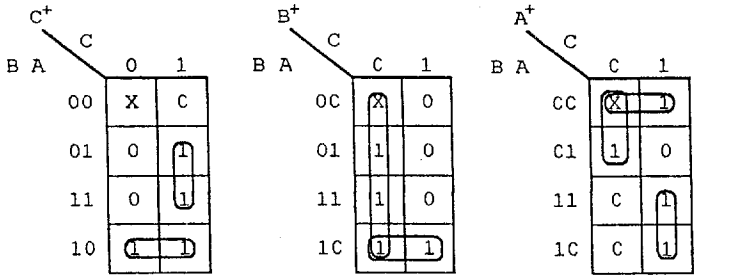


C B A	$C^+B^+A^+$
0 0 0	0 0 1
0 0 1	1 0 0
0 1 0	1 0 1
0 1 1	X X X
1 0 0	0 1 0
1 0 1	1 1 0
1 1 0	0 0 0
1 1 1	X X X



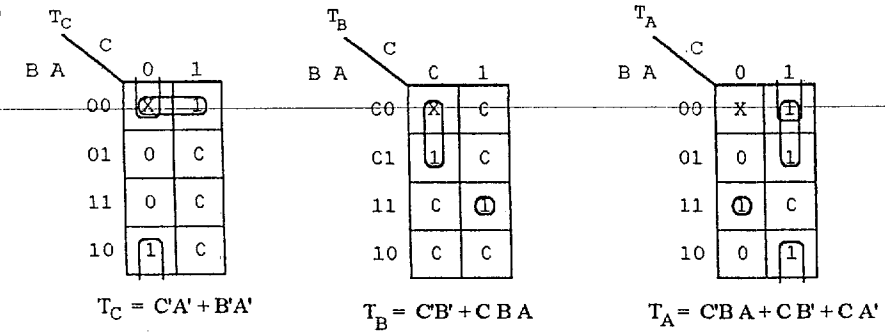
12.7 (a)

C B A	$C^+B^+A^+$
0 0 0	X X X
0 0 1	0 1 1
0 1 0	1 1 0
0 1 1	0 1 0
1 0 0	0 0 1
1 0 1	1 0 0
1 1 0	1 1 1
1 1 1	1 0 1



For D flip-flop: 000 goes to 011 because $D_C D_B D_A = 011$

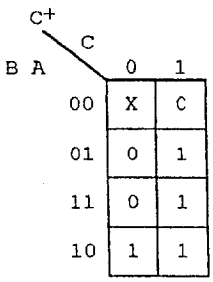
12.7 (b)

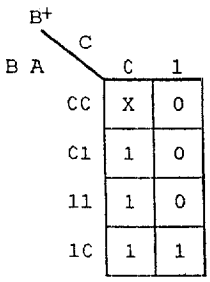


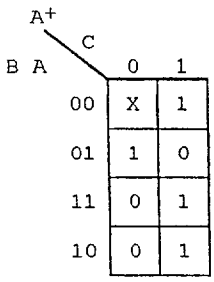
For T flip-flop: 000 goes to 110 because $T_A T_B T_C = 110$

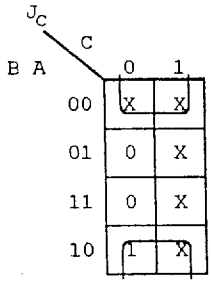
12.8 (a)

CBA	$C^+B^+A^+$
000	XXX
001	011
010	110
011	010
100	001
101	100
110	111
111	101

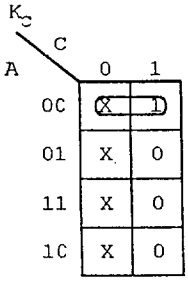




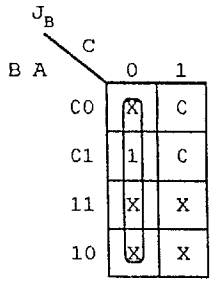




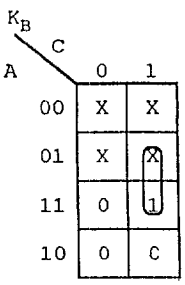
$J_C = A'$



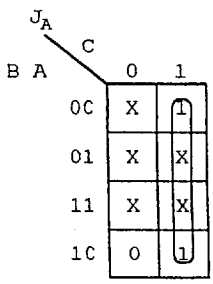
$K_C = B'A'$



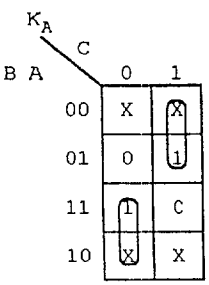
$J_B = C'$



$K_B = C\,A$



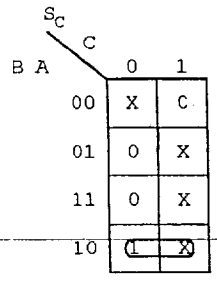
$J_A = C$



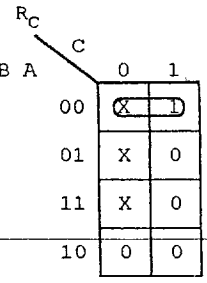
$K_A = C'B + C\,B'$

In state 000,
$J_C = A' = 1, K_C = B'A' = 1, C^+ = C' = 1$
$J_B = C' = 1, K_B = CA = 0, B^+ = 1$
$J_A = C = 0, K_A = CB' + C'B = 0, A^+ = A = 0$
So the next state is $C^+B^+A^+ = 110$

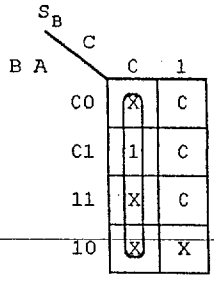
12.8 (b)



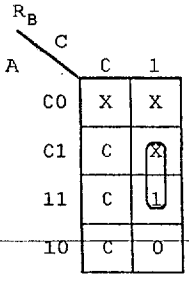
$S_C = B\,A'$



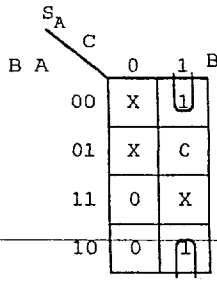
$R_C = B'A'$



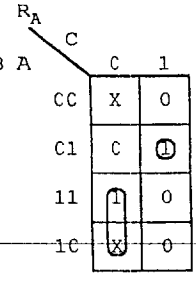
$S_B = C'$



$R_B = C\,A$



$S_A = C\,A'$



$R_A = C'B + C\,B'A$

$S_C = C'A'$

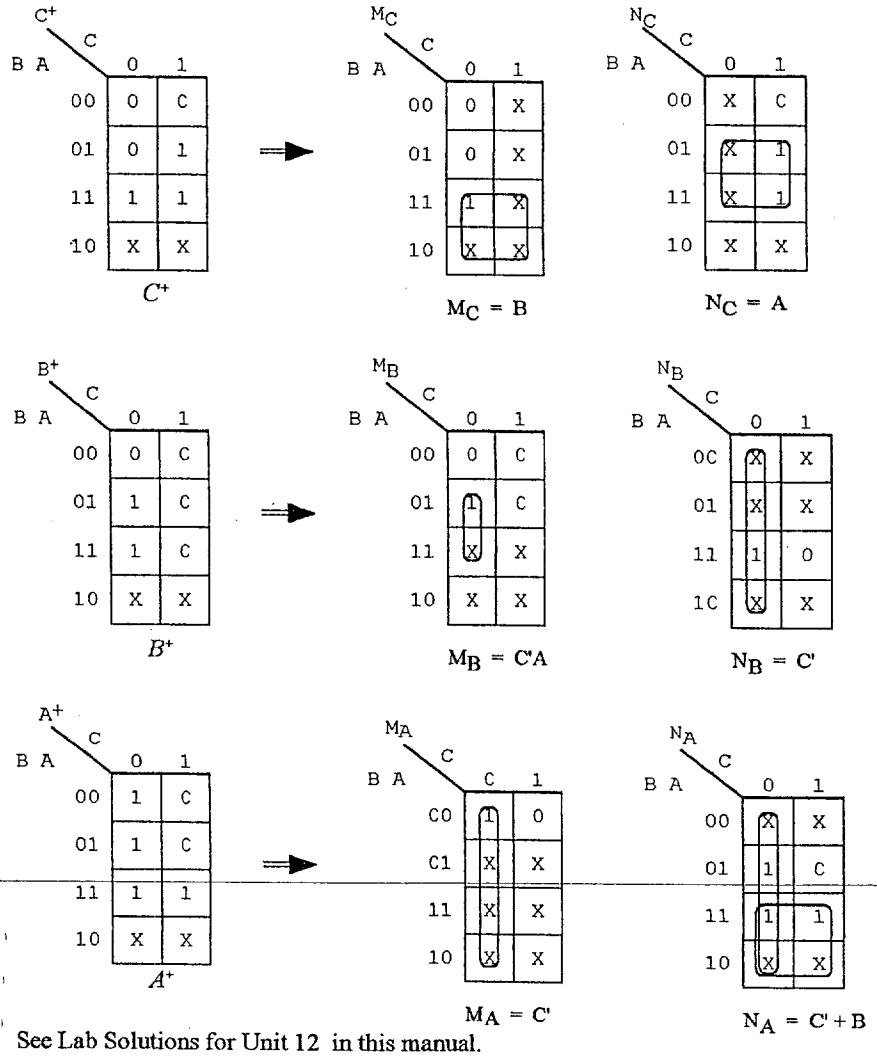
In state 000,
$S_C = BA' = 0, R_C = B'A' = 1, C^+ = 0$
$S_B = C' = 1, R_B = CA = 0, B^+ = 1$
$S_A = CA' = 0, R_A = C'B + C'BA = 0, A^+ = A = 0$
So the next state is $C^+B^+A^+ = 010$

12.9 (a)

QQ^+	MN	
00	00	0X
	01	
01	10	1X
	11	
10	10	X0
	00	
11	01	X1
	11	

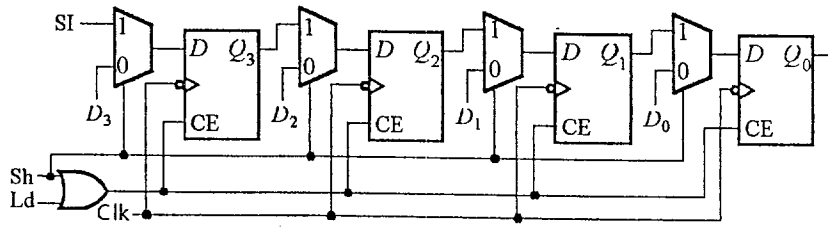
12.9 (b)

CBA	$C^+B^+A^+$
000	001
001	011
011	111
111	101
101	100
100	000

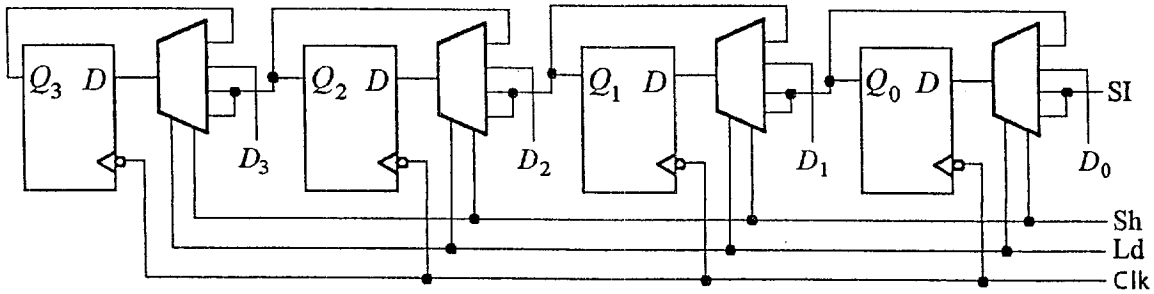


12.10 See Lab Solutions for Unit 12 in this manual.

12.11 The flip-flops change state only when *Ld* or *Sh* = 1. So $CE = Sh + Ld$. Now only a 2-to-1 MUX is required to select the input to the D flip-flop.

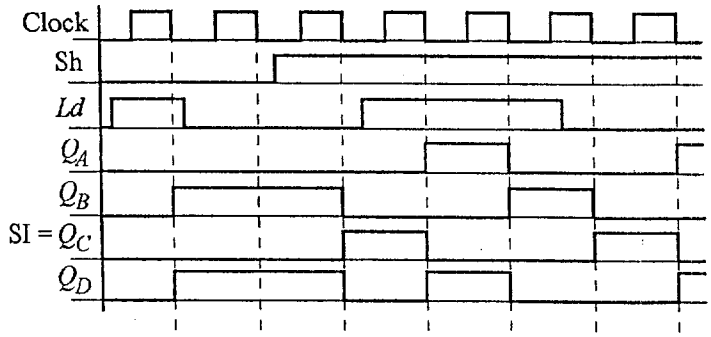


12.12 (a) When $ShLd = 00$, the MUX for flip-flop i selects Q_i to hold its state
When $ShLd = 01$, the MUX for flip-flop i selects D_i to load.
When $ShLd = 10$ or 11, the MUX for flip-flop i selects Q_{i-1} to shift left.

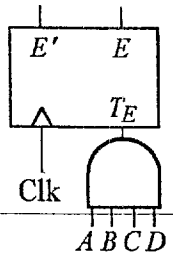


12.12 (b) $Q_3^+ = Ld'Sh'Q_3 + LdSh'D_3 + ShQ_2$; $Q_2^+ = Ld'Sh'Q_2 + LdSh'D_2 + ShQ_1$; $Q_1^+ = Ld'Sh'Q_1 + LdSh'D_1 + ShQ_0$
$Q_0^+ = Ld'Sh'Q_0 + LdSh'D_0 + ShSI$

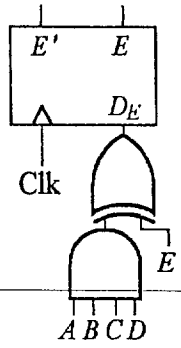
12.13 Notice that Sh overrides Ld when $Sh = Ld = 1$



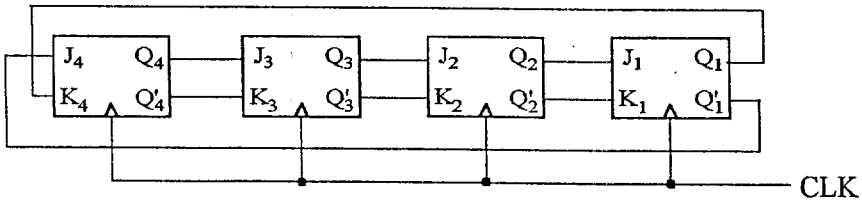
12.14 (a) Similar to problem 12.4 (a), T_E = ABCD.
T_D, T_C, T_B, and T_A, remain unchanged.



12.14 (b)
Similar to problem 12.4 (b),
$D_E = E \oplus DBCA$. D_D, D_C, D_B, and D_A, remain unchanged.



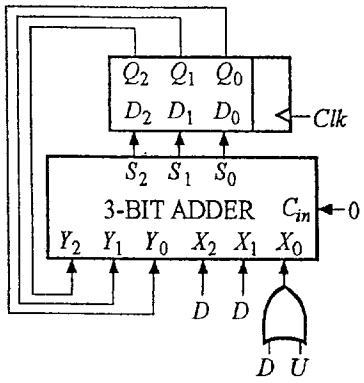
12.15 4-bit Johnson counter using J-K flip-flops:



Starting in 0000: 0000, 1000, 1100, 1110, 1111, 0111, 0011, 0001, (repeat) 0000, ...

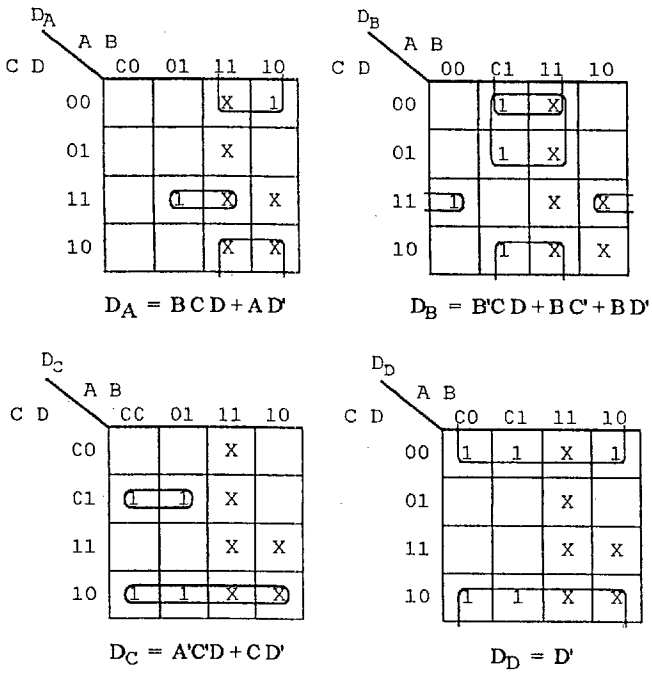
Starting in 0110: 0110, 1011, 0101, 0010, 1001, 0100, 1010, 1101, (repeat) 0110, ...

12.16 When $U = 1, D = 0$, add 001. When $U = 0, D = 1$, subtract 1: add 111.
When $U = 0, D = 0$, no change: add 000.
$U = 1, D = 1$, can never occur.
So add the contents of the register to $X_2X_1X_0$, where $X_2 = X_1 = D$ and $X_0 = D + U$. (*Note*: to save the OR gate, let $X_0 = D$ and $C_{in} = U$.)

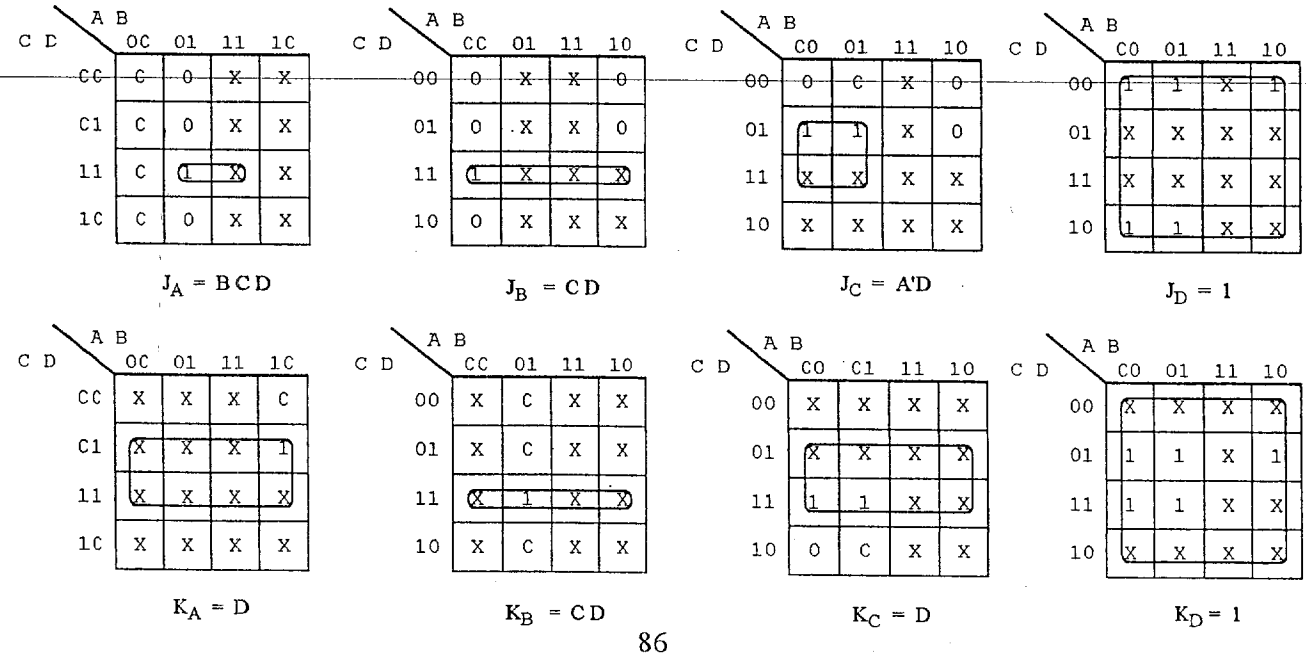


12.17 (a)

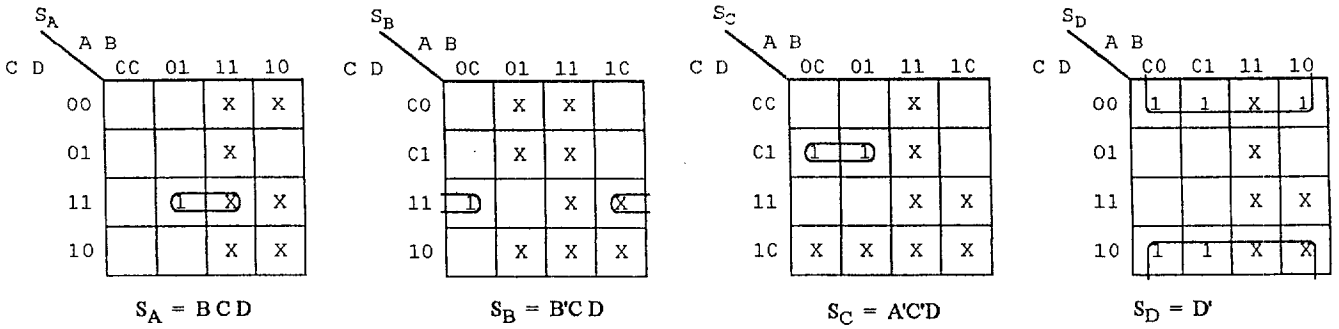
A B C D	A⁺B⁺C⁺D⁺
0 0 0 0	0 0 0 1
0 0 0 1	0 0 1 0
0 0 1 0	0 0 1 1
0 0 1 1	0 1 0 0
0 1 0 0	0 1 0 1
0 1 0 1	0 1 1 0
0 1 1 0	0 1 1 1
0 1 1 1	1 0 0 0
1 0 0 0	1 0 0 1
1 0 0 1	0 0 0 0
1 0 1 0	X X X X
1 0 1 1	X X X X
1 1 0 0	X X X X
1 1 0 1	X X X X
1 1 1 0	X X X X
1 1 1 1	X X X X

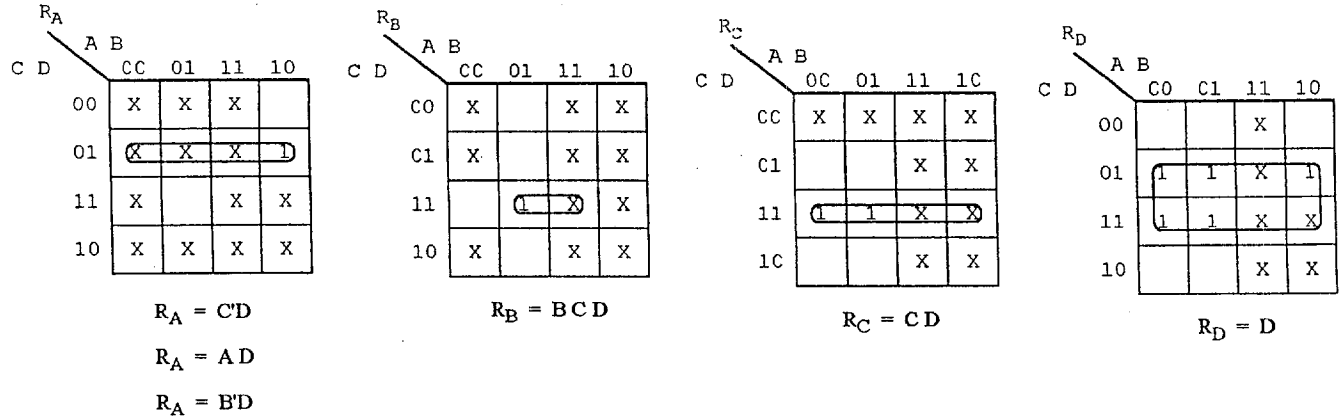


12.17 (b) See table 12-7 (c) on FLD p. 345.

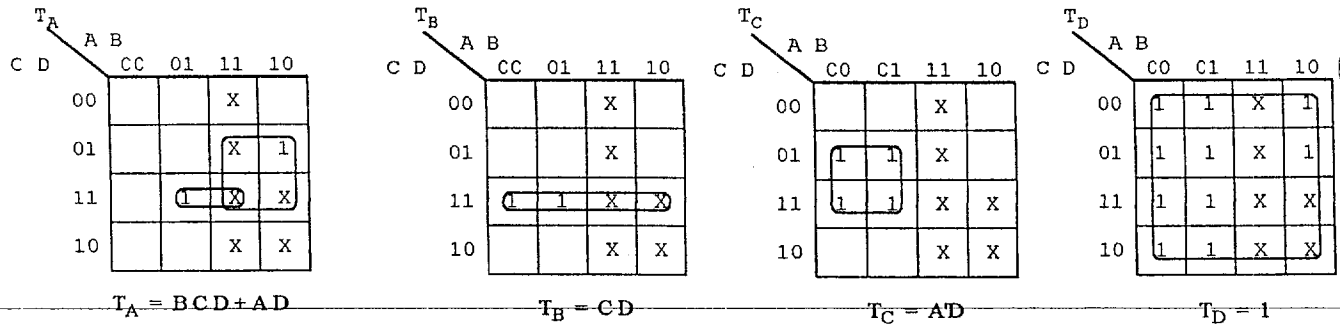


12.17 (c) *See table 12-5 (c) on FLD p. 342.*





12.17 (d) *See table 12-4 on FLD p. 339.*



12.17 (e) Use equations to find next states for unused states.

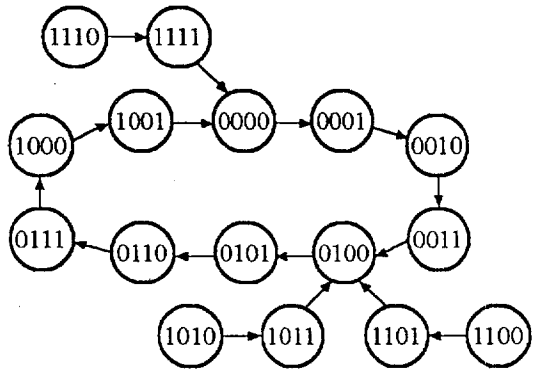
State 1101: $J_A = BCD = 0, K_A = D = 1, A^+ = 0$

$J_B = CD = 0, K_B = CD = 0, B^+ = B = 1$

$J_C = A'D = 0, K_C = D = 1, C^+ = 0$

$J_D = 1, K_D = 1, D^+ = D' = 0$

So the next state is 0100. Other next states can be found in a similar way.



12.18

$ABCD$	$A^+B^+C^+D^+$
0 0 0 0	1 0 0 1
0 0 0 1	0 0 0 0
0 0 1 0	0 0 0 1
0 0 1 1	0 0 1 0
0 1 0 0	0 0 1 1
0 1 0 1	0 1 0 0
0 1 1 0	0 1 0 1
0 1 1 1	0 1 1 0
1 0 0 0	0 1 1 1
1 0 0 1	1 0 0 0
1 0 1 0	X X X X
1 0 1 1	X X X X
1 1 0 0	X X X X
1 1 0 1	X X X X
1 1 1 0	X X X X
1 1 1 1	X X X X

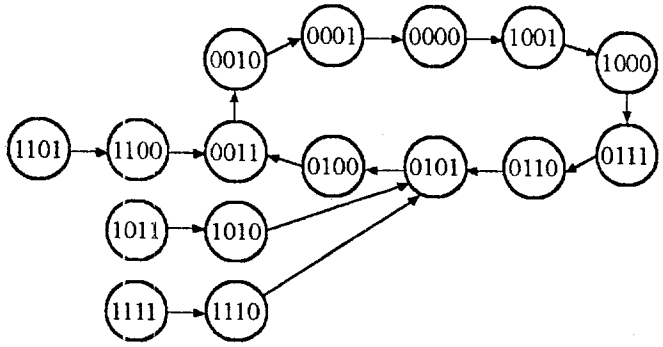
12.18 (a) $D_A = A'B'C'D' + AD$; $D_B = BD + BC + AD'$; $D_C = CD + BC'D' + AD'$; $D_D = D'$

12.18 (b) $J_A = B'C'D'$, $K_A = D'$; $J_B = AD'$, $K_B = C'D'$; $J_C = BD' + AD'$, $K_C = D'$; $J_D = 1$, $K_D = 1$

12.18 (c) $S_A = A'B'C'D'$, $R_A = AD'$; $S_B = AC'D'$, $R_B = BC'D'$ or $A'C'D'$; $S_C = BD'C' + AD'$, $R_C = CD'$, $S_D = D'$, $R_D = D$

12.18 (d) $T_A = B'C'D'$; $T_B = BC'D' + AC'D'$; $T_C = CD' + BD' + AD'$; $T_D = 1$

12.18 (e)



12.19

ABC	$A^+B^+C^+$
0 0 0	X X X
0 0 1	1 0 0
0 1 0	0 1 1
0 1 1	0 0 1
1 0 0	1 0 1
1 0 1	1 1 1
1 1 0	0 1 0
1 1 1	1 1 0

12.19 (a) $D_A = B' + AC$; $D_B = AC + BC'$; $D_C = A'B + AB'$

12.19 (b) $J_A = B'$, $K_A = BC'$; $J_B = AC$, $K_B = A'C$; $J_C = A' + B'$, $K_C = A'B' + AB$

12.19 (c) $T_A = A'B' + ABC'$; $T_B = A'BC + AB'C$; $T_C = A'B' + A'C' + B'C' + ABC$

12.19 (d) $S_A = B'$, $R_A = BC'$; $S_B = AC$, $R_B = A'C$; $S_C = A'B + AB'$, $R_C = A'B' + AB$

12.19 (e) State 000 goes to 100, because $D_A D_B D_C = 100$.

12.20 (a)

QQ^+	LM	
0 0	0 1 1 1	X1
0 1	0 0 1 0	X0
1 0	1 0 1 1	1X
1 1	0 0 0 1	0X

12.20 (b)

ABC	$A^+B^+C^+$
0 0 0	1 0 0
0 0 1	0 0 0
0 1 0	X X X
0 1 1	0 0 1
1 0 0	1 0 1
1 0 1	1 1 1
1 1 0	X X X
1 1 1	0 1 1

A^+

B C \ A	C	1
CC	1	1
C1	C	1
11	C	0
1C	X	X

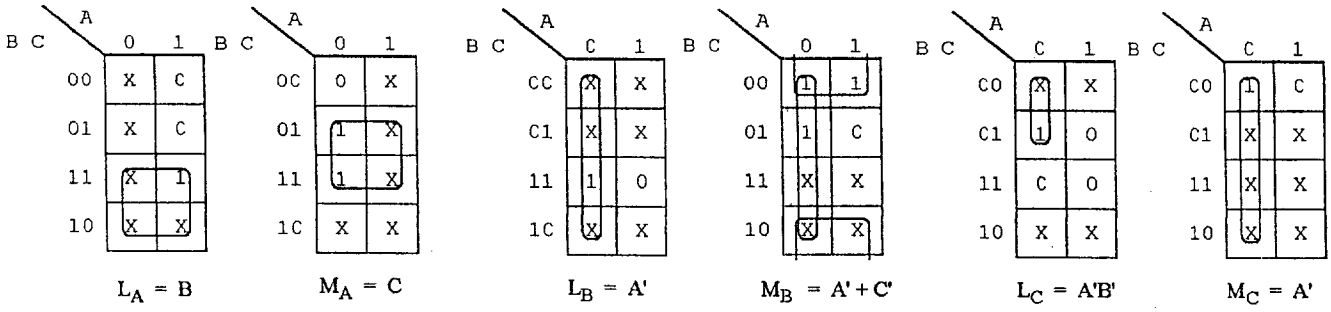
B^+

B C \ A	C	1
C0	C	C
C1	C	1
11	C	1
10	X	X

C^+

B C \ A	C	1
CC	C	1
C1	C	1
11	1	1
1C	X	X

12.20 (b) $L_A = B, M_A = C; L_B = A', M_B = A' + C'; L_C = A'B',$
(contd) $M_C = A'$



12.21

$ABCD$	$A^+B^+C^+D^+$	$J_A K_A J_B K_B J_C K_C J_D K_D$
0 0 0 0	0 0 1 1	0 X 0 X 1 X 1 X
0 0 0 1	0 1 0 0	0 X 1 X 0 X X 1
0 0 1 0	0 1 0 1	0 X 1 X X 1 1 X
0 0 1 1	0 1 1 0	0 X 1 X X 0 X 1
0 1 0 0	0 1 1 1	0 X X 0 1 X 1 X
0 1 0 1	1 0 0 0	1 X X 1 0 X X 1
0 1 1 0	1 0 0 1	1 X X 1 X 1 1 X
0 1 1 1	1 0 1 0	1 X X 1 X 0 X 1
1 0 0 0	1 0 1 1	X 0 0 X 1 X 1 X
1 0 0 1	1 1 0 0	X 0 1 X 0 X X 1
1 0 1 0	1 1 0 1	X 0 1 X X 1 1 X
1 0 1 1	1 1 1 0	X 0 1 X X 0 X 1
1 1 0 0	1 1 1 1	X 0 X 0 1 X 1 X
1 1 0 1	X X X X	X X X X X X X X
1 1 1 0	X X X X	X X X X X X X X
1 1 1 1	X X X X	X X X X X X X X

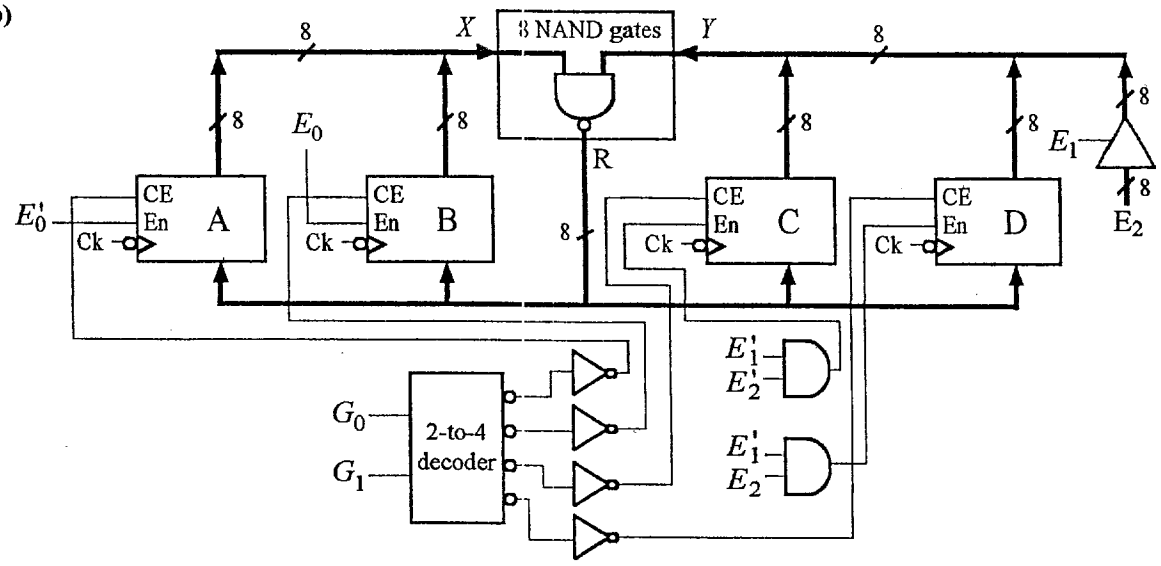
Using Karnaugh maps:
$J_A = A + BD + BC$, $K_A = 0$; $J_B = C + D$, $K_B = C + D$;
$J_C = D'$, $K_C = D'$; $J_D = 1$, $K_D = 1$

12.22

Clock Cycle	Input Data	EnIn	EnAd	LdAc	LdAd	Accumulator Register	Addend Register	Bus	Description
0	18	1	0	1	0	0	0	18	Input to accumulator
1	13	1	0	0	1	18	0	13	Input to addend
2	15	0	1	1	0	18	13	31	Sum to accumulator
3	93	1	0	0	1	31	13	93	Input to addend
4	47	0	1	1	0	31	93	124	Sum to accumulator
5	22	1	0	0	1	124	93	22	Input to addend
6	0	0	1	0	0	124	22	146	Sum on bus

Note: Register values change *after* the clock edge. So a value loaded from the bus appears in the register on the next clock cycle after the load signal and bus value are present.

12.23 (a, b)



12.23 (c) Call the values beginning in the *A* & *D* registers *X* and *Y*, respectively. We want $C = X + Y = (X'Y')'$. Invert using $M' = 1$ NAND M. To invert a value on the right side, in register *C* or *D*, we will need a 1 on the left side, in register *A* or *B*. This can be accomplished using 1 = 0 NAND (anything.)

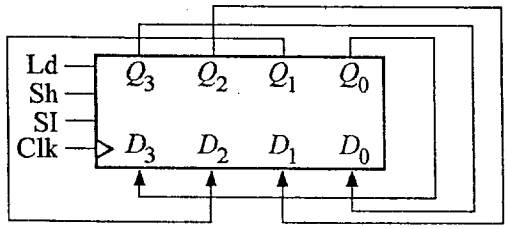
There are several solutions using different registers. Here is an example:

Clock Cycle	G_0G_1	E_0	E_1E_2	Description
1	0 0	0	1 1	1 NAND $A = A' = X' \rightarrow A$
2	0 1	1	1 0	0 NAND $B = 1 \rightarrow B$
3	1 1	1	0 1	B NAND $D = 1$ NAND $Y = Y' \rightarrow D$
4	1 0	0	0 1	A NAND $D = X'$ NAND $Y' = X + Y \rightarrow C$

Alternate three-cycle solution:
Use $X + Y = X + X'Y = (X' (X'Y)')'$

Clock Cycle	G_0G_1	E_0	E_1E_2	Description
1	0 0	0	1 1	1 NAND $A = A' = X' \rightarrow A$
~~2~~	~~1 1~~	~~0~~	~~0 1~~	~~A NAND $D = (X'Y)' \rightarrow D$~~
3	1 0	0	0 1	A NAND $D = (X' (X'Y)')' = X + Y \rightarrow C$

12.24 (a) For bit reversal using the D inputs of the shift register: $Sh = 0, Ld = 1$



12.24 (b) Same as Figure 12-10 (b) on FLD p. 331, except that for the "11" input of each MUX, instead of *SI*, Q_3, Q_2, or Q_1, use Q_0, Q_1, Q_2, or Q_3, respectively. Also, replace *Sh* with *A* and *Ld* with *B*.

Unit 13 Problem Solutions

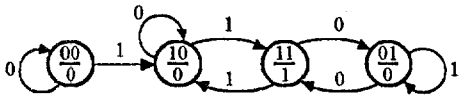
13.2 Notice that this is a shift register. At each falling clock edge, Q_3 takes on the value Q_2 had right before the clock edge, Q_2 takes on the value Q_1 had right before the clock edge, and Q_1 takes on the value X had right before the clock edge. For example, if the initial state is 000 and the input sequence is $X = 1100$, the state sequence is = 100, 110, 011, 001, and the output sequence is $Z = (0)0011$. Z is always Q_3, which does not depend on the present value of X. So it's a Moore machine. *See FLD p. 653 for the state graph.*

13.3 (a) $A^+ = AK_A' + A'J_A = A(B' + X) + A'(BX' + B'X)$
$B^+ = B'J_B + BK_B' = AB'X + B(A' + X')$
$Z = AB$

A+	X=0	X=1
00	0	1
01	1	C
11	0	1
10	1	1

B+	X=0	X=1
CC	C	0
C1	1	1
11	1	0
1C	C	1

Present State AB	Next State (A^+B^+) $X=0$	$X=1$	Z
00	00	10	0
01	11	01	0
11	01	10	1
10	10	11	0



13.3 (b) $X =$ 0 1 1 0 0
$AB =$ 00 00 10 11 01 11
$Z =$ (0) 0 0 1 0 1

13.3 (c) *See FLD p. 653 for solution.*

13.4 (a)

Q2 Q3 \ X Q1	00	C1	11	10
00	0	0	C	0
01	0	0	C	0
11	0	0	C	0
10	1	1	1	1

$Q_1^+ = D_1$

Q2 Q3 \ X Q1	00	C1	11	1C
CC	C	0	0	C
C1	1	1	1	1
11	1	1	1	1
1C	C	0	0	C

$Q_2^+ = D_2$

Q2 Q3 \ X Q1	0C	01	11	1C
CC	1	1	1	1
C1	1	1	1	1
11	C	0	1	1
1C	C	0	1	1

$Q_3^+ = D_3$

Q2 Q3 \ X Q1	C0	C1	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	C	0
10	1	1	C	0

Z

Z depends on the input X, so this is a Mealy machine. Because there are more than 2 state variables, we cannot put the state table in Karnaugh Map order (i.e. 00, 01, 11, 10), but we can still read the next state and output from the Karnaugh map. For example, when the input is X = 1 and the state is $Q_1Q_2Q_3 = 110$, we can read the next state and output from the $XQ_1Q_2Q_3 = 1110$ position in the Karnaugh maps for Q_1^+, Q_2^+, Q_2^+, and Z. So in this case, the next state is $Q_1^+Q_2^+Q_3^+ = 101$ and the output is $Z = 0$. The entire table can be derived from the Karnaugh maps in this manner. *Note*: We can also fill in the state table directly from the equations, without using Karnaugh maps. See FLD p. 653 for the state table and state graph.

13.4 (b - d) *See FLD p. 654 for solutions.*

13.5 (a) Mealy machine, because the output, Z, depends on the input X as well as the present state.

13.5 (c - d) *See FLD p. 654 for solutions.*

13.5 (b)

Z	X A			
B C	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	0	0
10	1	1	0	0

Note: Not all Karnaugh map entries are needed. See FLD p. 654 for the state table.

13.6 (a) After a rising clock edge, it takes 4 ns for the flip-flop outputs to change. Then the ROM will take 8 ns to respond to the new flip-flop outputs. The ROM outputs must be correct at the flip-flop inputs for at least the setup time of 2 ns before the next rising clock edge. So the minimum clock period is (4 + 8 + 2) ns = 14 ns.

13.6 (b) The correct output sequence is 0101. See FLD p. 655 for the timing diagram.

13.6 (c) Read the state transition table from ROM truth table. See FLD p. 655 for the state graph and table.

Present State	Next State $(Q_1^+Q_2^+)$		Z	
Q_1Q_2	$X=0$	$X=1$	$X=0$	$X=1$
00	10	10	0	0
01	00	11	0	0
10	11	01	0	1
11	01	11	1	1

Alternate solution: Using Karnaugh map order, swap states S_2 and S_3 in the graph and table.

13.7 (a) Notice that Z depends on the input X, so this is a Mealy machine.

$Q_1^+ = J_1Q_1' + K_1'Q_1 = XQ_1'Q_2 + X'Q_1$

$Q_2^+ = J_2Q_2' + K_2'Q_2 = XQ_1'Q_2' + X'Q_2$

$Z = Q_2 \oplus X = XQ_2' + X'Q_2$

State	Present State	Next State $(Q_1^+Q_2^+)$		Z	
	Q_1Q_2	$X=0$	$X=1$	$X=0$	$X=1$
S_0	00	00	01	0	1
S_1	01	01	10	1	0
S_2	11	11	00	1	0
S_3	10	10	00	0	1

Alternate solution: Swap states S_2 and S_3.

Q1 Q2 \ X	0	1
0C	0	0
01	0	1
11	1	0
1C	1	0

$Q1^+$

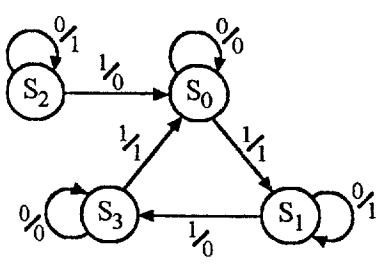
Q1 Q2 \ X	C	1
CC	C	1
C1	1	0
11	1	0
1C	C	0

$Q2^+$

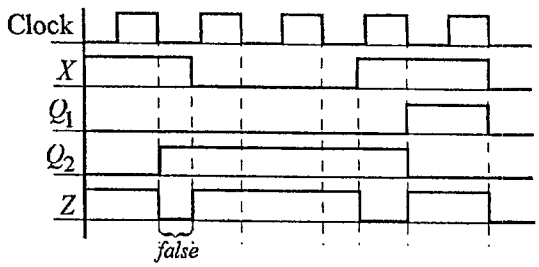
Q1 Q2 \ X	C	1
0C	C	1
01	1	0
11	1	0
1C	C	1

Z

13.7 (a) (contd)



13.7 (b)



13.8 (a) Notice that Z does not depend on this input X, so this is a Moore machine.

$Q_1^+ = X_1X_2Q_1 + Q_1Q_2 + X_1Q_2$

$Q_2^+ = Q_1'(X_1 + X_2) + Q_2(X_1' + X_2') = X_1Q_1' + X_2Q_1' + X_1'Q_2 + X_2'Q_2$

$Z = Q_1Q_2'$

Q1 Q2 \ X1 X2	C0	01	11	10
00	0	C	C	0
01	0	C	1	1
11	1	1	1	1
10	0	C	1	0

$Q1^+$

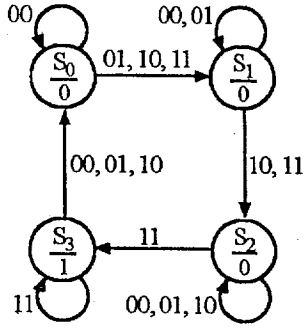
Q1 Q2 \ X1 X2	C0	01	11	10
00	0	1	1	1
01	1	1	1	1
11	1	1	0	1
10	0	C	0	0

$Q2^+$

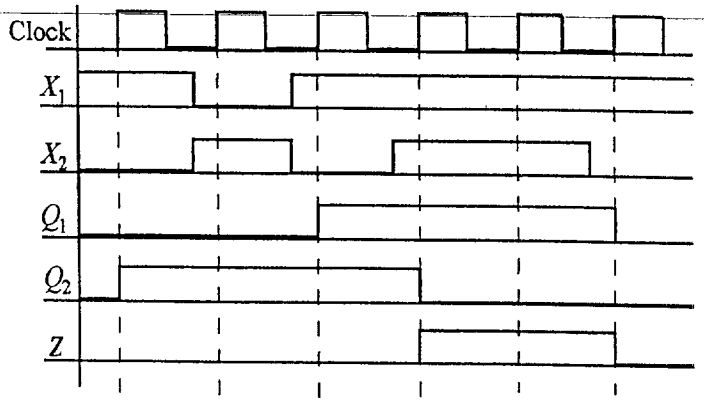
Q2 \ Q1	0	1
C	0	1
1	0	0

Z

State	Present State Q_1Q_2	Next State X_1X_2 00	01	11	10	Z
S_0	00	00	01	01	01	0
S_1	01	01	01	11	11	0
S_2	11	11	11	10	11	0
S_3	10	00	00	10	00	1

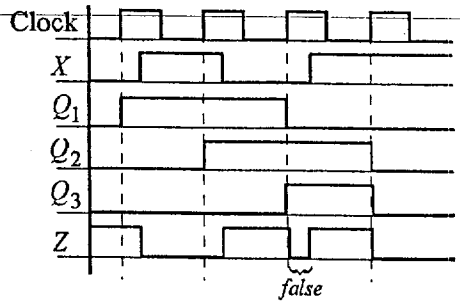


13.8 (b)



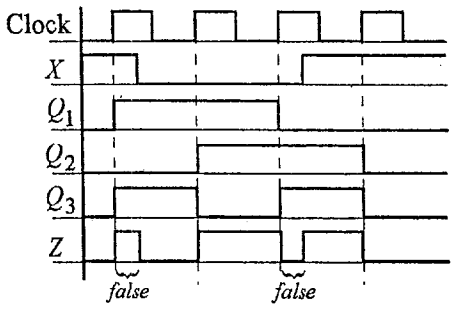
Correct output: $Z = 0, 0, 0, 1, 1, 0$

13.9



Correct output: $Z = 1, 0, 1, 1$

13.10



Correct output: $Z = 0, 0, 1, 1$

13.11 (a)

Q2 Q3 \ X Q1	CC	01	11	1C
C0	C	C	0	1
C1	C	C	0	1
11	C	C	0	1
10	1	1	1	1

$D_1 - Q_1^+$

Q2 Q3 \ X Q1	00	C1	11	10
0C	0	0	C	C
01	1	1	1	1
11	1	1	1	1
1C	1	1	C	C

$D_2 - Q_2^+$

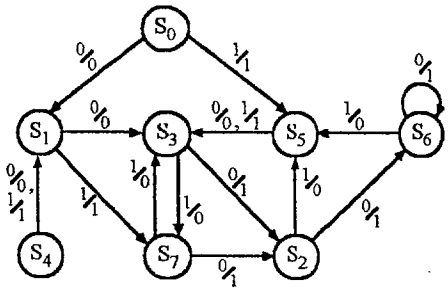
Q2 Q3 \ X Q1	00	01	11	1C
CC	1	1	1	1
C1	1	1	1	1
11	C	0	1	1
1C	C	0	1	1

$D_3 - Q_3^+$

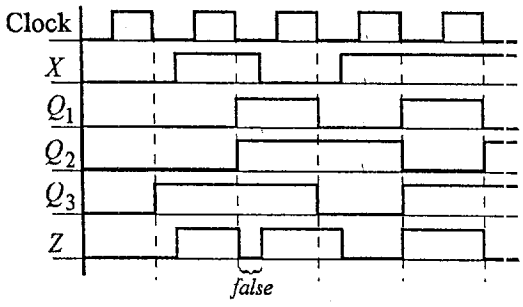
Q2 Q3 \ X Q1	C0	01	11	10
00	0	C	1	1
01	0	C	1	1
11	1	1	0	0
10	1	1	0	0

Z

State	Present State $Q_1Q_2Q_3$	Next State $(Q_1^+Q_2^+Q_3^+)$ $X=0$	$X=1$	Z $X=0$	$X=1$
S_0	000	001	101	0	1
S_1	001	011	111	0	1
S_2	010	110	101	1	0
S_3	011	010	111	1	0
S_4	100	001	001	0	1
S_5	101	011	011	0	1
S_6	110	110	101	1	0
S_7	111	010	011	1	0



13.11 (b)



13.11 (c) From diagram: 0, 1, (0), 1, 0, 1
From graph: 0, 1, 1, 0, 1
(they are the same, except for the false output)

13.11 (d) Change the input on the falling edge of the clock (assuming negligible circuit delays).

13.12 (a)

Q2 Q3 \ X Q1	CC	01	11	10
00	1	1	0	0
01	0	C	0	0
11	0	C	0	0
10	1	1	0	0

$D_1 = Q_1^+$

Q2 Q3 \ X Q1	C0	C1	11	10
00	0	1	1	1
01	0	C	1	1
11	0	C	C	0
10	0	1	1	0

$D_2 = Q_2^+$

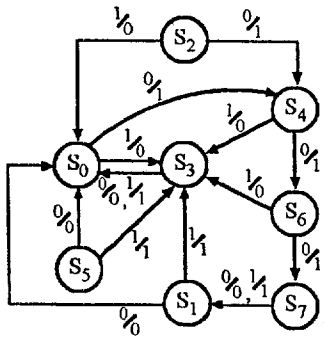
Q2 Q3 \ X Q1	00	C1	11	1C
CC	C	0	1	1
C1	C	0	1	1
11	C	1	1	C
1C	C	1	1	C

$D_3 = Q_3^+$

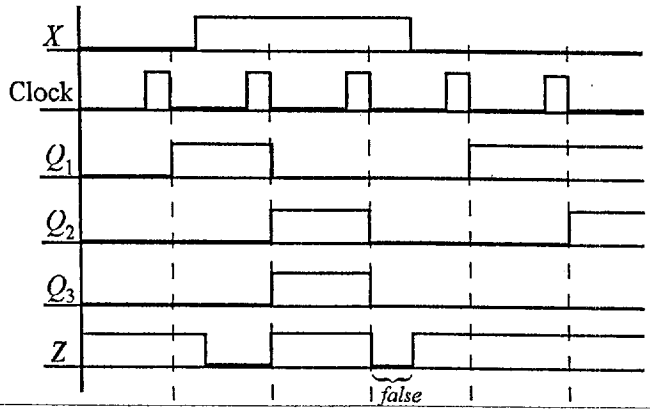
Q2 Q3 \ X Q1	CC	01	11	10
00	1	1	0	0
01	0	C	1	1
11	0	C	1	1
10	1	1	0	0

Z

State	Present State	Next State $(Q_1^+Q_2^+Q_3^+)$		Z	
	$Q_1Q_2Q_3$	$X=0$	$X=1$	$X=0$	$X=1$
S_0	000	100	011	1	0
S_1	001	000	011	0	1
S_2	010	100	000	1	0
S_3	011	000	000	0	1
S_4	100	110	011	1	0
S_5	101	000	011	0	1
S_6	110	111	011	1	0
S_7	111	001	001	0	1



13.12 (b)



13.12 (c)

From diagram: 1 0 1 (0) 1 1

From graph: 1 0 1 1 1

(they are the same, except for the false output)

13.12 (d)

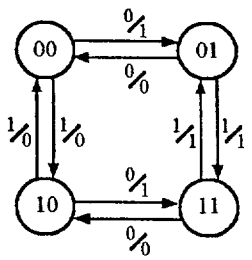
Change the input on the falling edge of the clock (assuming negligible circuit delays).

13.13

Clock Cycle	Information Gathered
1	$Q_1Q_2 = 00, X=0 \Rightarrow Z=1, Q_1^+Q_2^+ = 01$
2	$Q_1Q_2 = 01, X=0 \Rightarrow Z=0; X=1 \Rightarrow Z=1, Q_1^+Q_2^+ = 11$
3	$Q_1Q_2 = 11, X=1 \Rightarrow Z=1; X=0 \Rightarrow Z=0, Q_1^+Q_2^+ = 10$
4	$Q_1Q_2 = 10, X=0 \Rightarrow Z=1; X=1 \Rightarrow Z=0, Q_1^+Q_2^+ = 00$
5	$Q_1Q_2 = 00, X=1 \Rightarrow Z=0, Q_1^+Q_2^+ = 10$
6	$Q_1Q_2 = 10, X=1 \Rightarrow (Z=0); X=0 \Rightarrow (Z=1), Q_1^+Q_2^+ = 11$
7	$Q_1Q_2 = 11, X=0 \Rightarrow (Z=0); X=1 \Rightarrow (Z=1), Q_1^+Q_2^+ = 01$
8	$Q_1Q_2 = 01, X=1 \Rightarrow (Z=1); X=0 \Rightarrow (Z=0), Q_1^+Q_2^+ = 00$
9	$Q_1Q_2 = 00, X=0 \Rightarrow (Z=1)$

Note: Information inside parentheses was already obtained in a previous clock cycle.

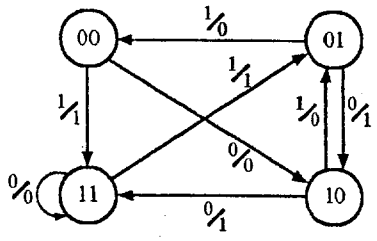
Present State	Next State $(Q_1^+Q_2^+)$		Z	
Q_1Q_2	$X=0$	$X=1$	0	1
00	01	10	1	0
01	00	11	0	1
10	11	00	1	0
11	10	01	0	1



13.14

Clock Cycle	Information Gathered
1	$Q_1Q_2 = 00, X = 0 \Rightarrow Z = 0, Q_1^+Q_2^+ = 10$
2	$Q_1Q_2 = 10, X = 0 \Rightarrow Z = 1; X = 1 \Rightarrow Z = 0, Q_1^+Q_2^+ = 01$
3	$Q_1Q_2 = 01, X = 1 \Rightarrow Z = 0; X = 0 \Rightarrow Z = 1, Q_1^+Q_2^+ = 10$
4	$Q_1Q_2 = 10, X = 0 \Rightarrow (Z = 1), Q_1^+Q_2^+ = 11$
5	$Q_1Q_2 = 11, X = 0 \Rightarrow Z = 0, Q_1^+Q_2^+ = 11$
6	$Q_1Q_2 = 11, X = 0 \Rightarrow (Z = 0); X = 1 \Rightarrow Z = 1, Q_1^+Q_2^+ = 01$
7	$Q_1Q_2 = 01, X = 1 \Rightarrow (Z = 0), Q_1^+Q_2^+ = 00$
8	$Q_1Q_2 = 00, X = 1 \Rightarrow Z = 1, Q_1^+Q_2^+ = 11$
9	$Q_1Q_2 = 11, X = 1 \Rightarrow (Z = 1)$
Note: Information inside parentheses was already obtained in a previous clock cycle.	

Present State	Next State $(Q_1^+Q_2^+)$		Z	
Q_1Q_2	$X = 0$	$X = 1$	0	1
00	10	11	0	1
01	10	00	1	0
10	11	01	1	0
11	11	01	0	1



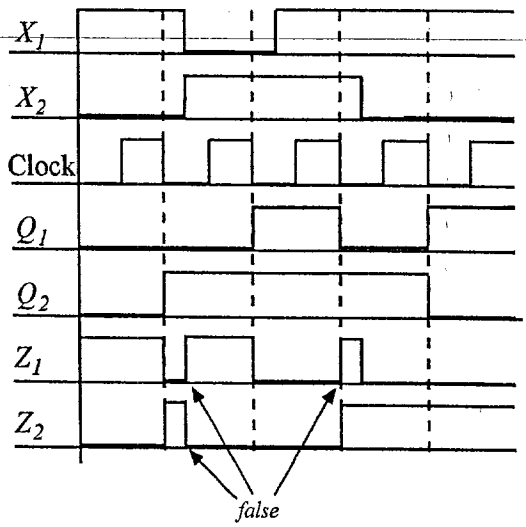
13.15

Clock Cycle	Information Gathered
1	$Q_1Q_2 = 00, X_1X_2 = 01 \Rightarrow Z_1Z_2 = 10, Q_1^+Q_2^+ = 01$
2	$Q_1Q_2 = 01, X_1X_2 = 01 \Rightarrow Z_1Z_2 = 01; X_1X_2 = 10 \Rightarrow Z_1Z_2 = 10, Q_1^+Q_2^+ = 10$
3	$Q_1Q_2 = 10, X_1X_2 = 10 \Rightarrow Z_1Z_2 = 00; X_1X_2 = 11 \Rightarrow Z_1Z_2 = 00, Q_1^+Q_2^+ = 01$
4	$Q_1Q_2 = 01, X_1X_2 = 11 \Rightarrow Z_1Z_2 = 11; X_1X_2 = 01 \Rightarrow (Z_1Z_2 = 01), Q_1^+Q_2^+ = 11$
5	$Q_1Q_2 = 11, X_1X_2 = 01 \Rightarrow Z_1Z_2 = 01$

Note: When $Q_1Q_2 = 01$, the outputs Z_1Z_2 vary depending on the inputs X_1X_2, so this is a Mealy machine.

Present State	$Q_1^+Q_2^+$ $X_1X_2=$				Z_1Z_2 $X_1X_2=$			
Q_1Q_2	00	01	11	10	00	01	11	10
00	?	01	?	?	?	10	?	?
01	?	11	?	10	?	01	11	10
11	?	?	?	?	?	01	?	?
10	?	?	01	?	?	?	00	00

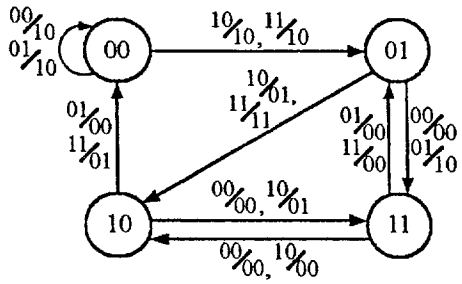
13.16 (a)



Correct output: $Z_1Z_2 = 10, 10, 00, 01$

13.16 (b)

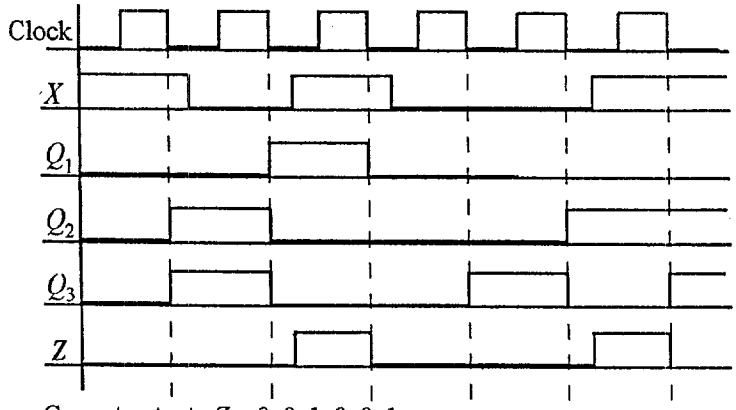
Present State	$Q_1^+Q_2^+$ $X_1X_2=$				Z_1Z_2 $X_1X_2=$			
Q_1Q_2	00	01	10	11	00	01	10	11
00	00	00	01	01	10	10	10	10
01	11	11	10	10	00	10	01	11
10	11	00	11	00	00	00	01	01
11	10	01	10	01	00	00	00	00



13.17

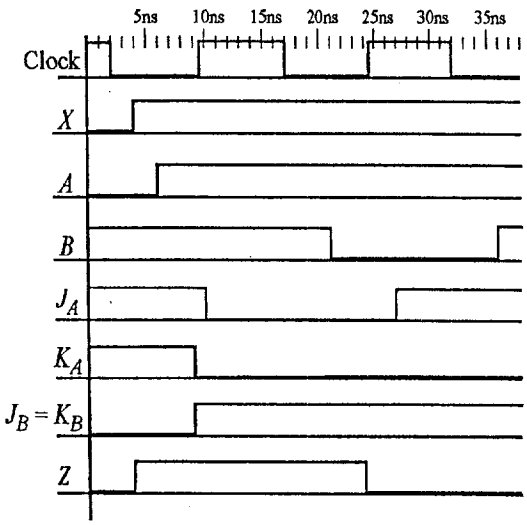
Transition table using a straight binary state assignment:

State	Present State	Next State $(Q_1^+Q_2^+Q_3^+)$		Z	
	$Q_1Q_2Q_3$	$X=0$	$X=1$	$X=0$	$X=1$
S_0	000	001	011	0	0
S_1	001	010	011	0	0
S_2	010	001	011	0	1
S_3	011	100	000	0	0
S_4	100	011	000	0	1



Correct output: $Z = 0, 0, 1, 0, 0, 1$

13.18 (a)



All flip-flop inputs are stable for more than the setup time before each falling clock edge. So the circuit is operating properly.

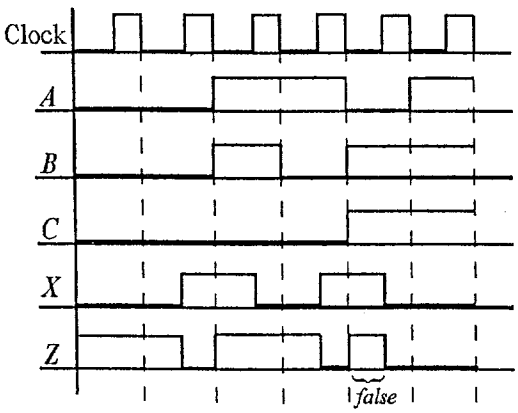
13.18 (b) If X is changed early enough:

Minimum clock period = Flip-flop propagation delay + Two NAND-gate delays + Setup time

$= 4 + (3 + 3) + 2 = 12$ ns

X can change as late as 8 ns (two NAND-gate delays plus the setup time) before the next falling edge without causing improper operation.

13.19



Correct output: $Z = 1\ 0\ 1\ 0\ 1$

Deriving the State Table:

JK flip-flop equation: $Q^+ = JQ' + K'Q$

$\therefore A^+ = (X'C + XC')A' + X'A$

[As, $J_A = X'C + XC'$, $K_A = X$, $Q = A$]

$= A'X'C + A'XC' + X'A$

Similarly, $B^+ = XC' + XA + X'A'C$

$C^+ = 0' \cdot C + (XB'A) \cdot C' = C + XB'A$

$Z = XB + X'C' + X'B'A$

A+

B C \ X A	CC	01	11	1C
C0	C	1	0	1
C1	1	1	0	0
11	1	1	0	0
10	C	1	0	1

B+

B C \ X A	00	C1	11	10
0C	0	0	1	1
01	1	0	1	C
11	1	0	1	C
1C	0	0	1	1

C+

B C \ X A	00	01	11	1C
CC	C	0	1	C
C1	1	1	1	1
11	1	1	1	1
1C	C	0	0	C

Z

B C \ X A	C0	01	11	10
00	1	1	C	0
01	0	1	C	0
11	0	C	1	1
10	1	1	1	1

From the Karnaugh maps, we can get the state table that follows:

State	Present State ABC	Next State $(A^+B^+C^+)$ $X=0$	$X=1$	Z $X=0$	$X=1$
S_0	000	000	110	1	0
S_1	001	111	001	0	0
S_2	010	000	110	1	1
S_3	011	111	001	0	1
S_4	100	100	011	1	0
S_5	101	101	011	1	0
S_6	110	100	010	1	1
S_7	111	101	011	0	1

13.20

$R = X_2(X_1' + B)$

$S = X_2'(X_1' + B')$

$A^+ = A[(X_2)(X_1' + B)]' + X_2'(X_1' + B')$

$= A(X_2' + X_1B') + X_2'X_1' + X_2'B'$

$A^+ = AX_2' + AX_1B' + X_2'X_1' + X_2'B'$

$T = X_1'BA + X_1'B'A'$

$B^+ = BT' + B'T$

$= B(X_1'BA + X_1'B'A')'B' (X_1'BA + X_1'B'A')'$

$= B[(X_1'BA)'(X_1'B'A')'] + X_1'B'A'$

$= B[(X_1 + B' + A')(X_1 + B + A)] + X_1'B'A'$

$= (BX_1 + BA')(X_1 + B + A) + X_1'B'A'$

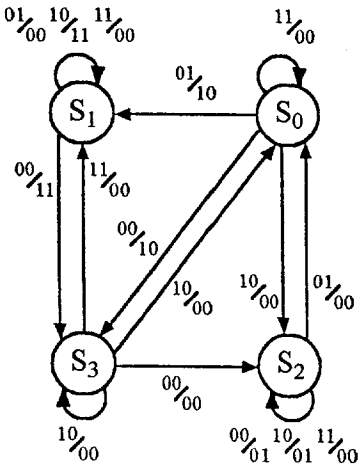
$= BX_1 + BX_1 + BX_1A + BA'X_1 + BA' + X_1'B'A'$

$= X_1B(1 + 1 + A + A') + A'(B + X_1'B)$

$= X_1B + A'B + X_1'A'$

13.20 (contd)

State	Present State AB	A^+B^+ $X_1X_2=$ 00	01	10	11	Z_1Z_2 $X_1X_2=$ 00	01	10	11
S_0	00	11	01	10	00	10	10	00	00
S_1	01	11	01	01	01	11	00	11	00
S_2	10	10	00	10	10	01	00	01	00
S_3	11	10	00	11	01	00	00	00	00



Unit 14 Problem Solutions

14.4 Typical input and output sequences:

X = 0 1 0 0 0 0 0 1 0 1 0 1 1 ...
Z = (0) 0 0 0 0 0 0 0 1 1 1 ... (output remains 1)
X = 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 0 1 ...
Z = (0) 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 ... (output remains 1)
X = 0 1 0 1 0 1 ...
Z = (0) 0 0 0 1 1 1 ... (output remains 1)

See FLD p. 656 for state graph.

The state meanings are given in the following table:

Name	Meaning
S_0	Reset
S_1	One 0, no 1's
S_2	≥ Two 0's, no 1's
S_3	≥ Two 0's and one 1
S_4	≥ Two 0's and ≥ Two 1's
S_5	≥ One 1, no 0's
S_6	≥ Two 1's, no 0's
S_7	≥ Two 1's and one 0
S_8	One 0 and one 1

14.5 Typical input and output sequence:

X = 0 0 1 0 1 0 1 1 0 0 1 0 1 0 0 ...
Z_1 = 0 0 0 1 0 1 0 0 0 0 0 0 0 0 ... (output remains 0 after 100 received)
Z_2 = 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 ... (at this point, the sequence 01 has occurred, so Z_1 = 0 from now on)

The graph needs two distinct parts. The first checks for 010 and 100. If 100 is received, we proceed to the second part of the graph, which checks only for 100. The two parts are joined by a one-way arc, so once in the second part it is impossible to go back to the first.
See FLD p. 656 for state table and graph.

The state meanings are given in the following table:

Name	Meaning
S_0	Reset
S_1	Last input was 0, 100 has never occurred
S_2	Last input was 01, 100 has never occurred
S_3	Last input was 1, 100 has never occurred
S_4	Last input was 10, 100 has never occurred
S_5	Last input was 0, 100 has occurred at least once
S_6	Last input was 1, 100 has occurred at least once
S_7	Last input was10, 100 has occurred at least once

14.6 This should be solved in the same way as Example 3 on FLD p. 406. Assign a state to each possible input (00, 01, 11, 10) with an output of 0, and another state to each input with an output of 1. This gives eight states.
See FLD p. 657 for the state table.

State	Z = 0
S_0	Last input was 00
S_1	Last input was 01
S_2	Last input was 11
S_3	Last input was 10

State	Z = 1
S_4	Last input was 00
S_5	Last input was 01
S_6	Last input was 11
S_7	Last input was 10

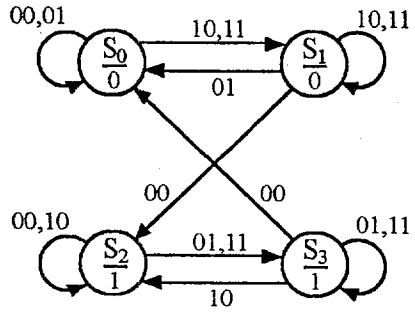
Each input takes you to the state defined by that input (e.g. an input of 01 takes you to either S_1 or S_5). The only thing in question is whether the output is 0 or 1. Determine the output by checking whether the last two inputs correspond to the three input sequences.

Alternate Solution: Notice that when $Z = 0$, "causes the output to become 0" is the same as remaining constant, and "causes the output to become 1" is the same as toggling the output. The situation is similar when $Z = 1$. So we can use only four states, as follows:

State	Meaning
S_0	Z= 0 and last input was either 00 or 01
S_1	Z = 0 and last input was either 10 or 11
S_2	Z = 1 and last input was either 00 or 10
S_3	Z = 1 and last input was either 01 or 11

State	Next State X_1X_2 = *00 01 11 10*	Z
S_0	S_0 S_0 S_1 S_1	0
S_1	S_2 S_0 S_1 S_1	0
S_2	S_2 S_3 S_3 S_2	1
S_3	S_0 S_3 S_3 S_2	1

Note: The state table with 8 states reduces to this 4-state table using methods in Unit 15.



14.7 (a) Typical input and output sequence:

$X = 0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,1\,1\,0\,1\,0\,0\,1$ ...
$Z = 1\,1\,0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,1\,1\,1\,0$ ...

See FLD p. 657 for state graph.

State	Meaning
S_0	Number of 1's is divisible by three
S_1	Number of 1's is one more than divisible by 3
S_2	Number of 1's is two more than divisible by 3

14.7 (b) Typical input and output sequence:

$X = 0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,1\,1\,0\,1\,0\,1\,1\,1\,1$ ...
$Z = 0\,1\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0$ ...

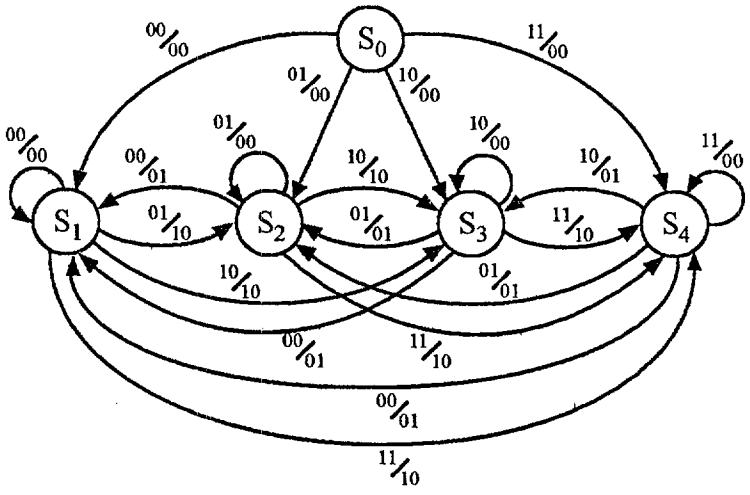
See FLD p. 657 for state graph.

State	Meaning
S_0	Number of 1's is divisible by three, no 0's
S_1	Number of 1's is one more than divisible by 3, no 0's
S_2	Number of 1's is two more than divisible by 3, no 0's
S_3	Number of 1's is divisible by three, number of 0's is odd
S_4	Number of 1's is one more than divisible by 3, number of 0's is odd
S_5	Number of 1's is two more than divisible by 3, number of 0's is odd
S_6	Number of 1's is divisible by three, number of 0's is even and < 0
S_7	Number of 1's is one more than divisible by 3, number of 0's is even and < 0
S_8	Number of 1's is two more than divisible by 3, number of 0's is even and < 0

14.8 (a) Typical input and output sequence:

$X_1 = 1\,0\,0\,1\,0\,0\,1\,1\,1\,0$ ...
$X_2 = 1\,0\,0\,0\,1\,1\,0\,0\,1\,1$ ...
$Z_1 = 0^*0\,0\,1\,0\,0\,1\,0\,1\,0$ ...
$Z_2 = 0^*1\,0\,0\,1\,0\,0\,0\,0\,1$ ...
* Regardless of any value of N.

See FLD p. 657 for state table.



State	Meaning
S_0	Reset
S_1	Previous input was 00
S_2	Previous input was 01
S_3	Previous input was 10
S_4	Previous input was 11

14.8 (b) Similar to part (a), but we need a separate state for each possible output and previous input.

See FLD p. 658 for state table.

State	Meaning
S_0	Reset state / current output is = 00
S_1	Previous input was 00 / current output is = 00
S_2	Previous input was 00 / current output is = 01
S_3	Previous input was 01 / current output is = 10
S_4	Previous input was 01 / current output is = 00
S_5	Previous input was 01 / current output is = 01
S_6	Previous input was 10 / current output is = 10
S_7	Previous input was 10 / current output is = 00
S_8	Previous input was 10 / current output is = 01
S_9	Previous input was 11 / current output is = 10
S_{10}	Previous input was 11 / current output is = 00

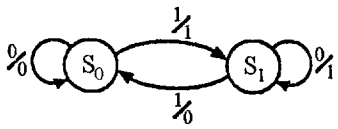
14.9 (a)

State	Meaning
S_0	Previous output bit was 0
S_1	Previous output bit was 1

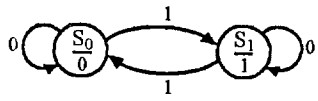
14.9 (b)

State	Meaning
S_0	Output bit is 0
S_1	Output bit is 1

14.9 (c) A false output occurs in NRZI just before the input NRZ goes from 1 to 0.



See FLD p. 658 for state table.



See FLD p. 658 for state table.

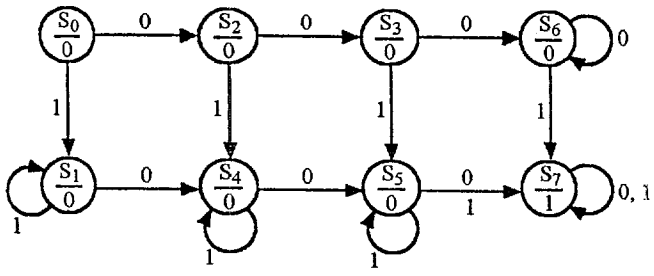
14.9 (d) Notice that the Moore output is delayed to the next clock cycle.

14.10 *See FLD p. 658 for solution.*

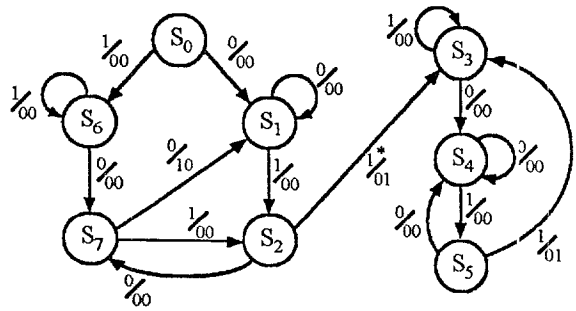
14.11 *See FLD p. 659 for state graph.*

State	Meaning
S_0	Reset
S_1	Button pressed. First full clock cycle with $Z = 1$.
S_2	Second full clock cycle with $Z = 1$.
S_3	Third full clock cycle with $Z = 1$.
S_4	Fourth full clock cycle with $Z = 1$.
S_5	X has not yet returned to 0.

14.12 Plot 0's horizontally. Plot 1's vertically. Receiving a 0 takes us one state to the right. Receiving a 1 takes us one state down. The output is a 1 only in the "three 0's or more, one 1 or more" state:



14.13

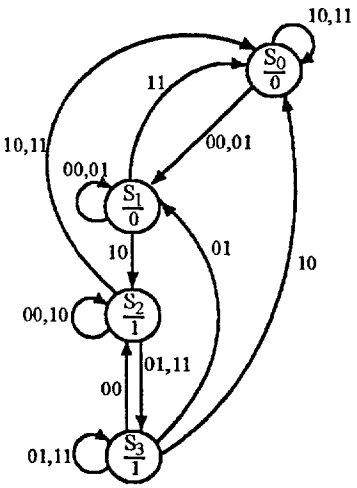


* When this point in the graph is reached, 011 has been received, and we are only looking for 011 to occur again.

State	Meaning
S_0	Reset
S_1	Previous input was 0 / 011 has not occurred
S_2	Previous input was 01 / 011 has not occurred
S_3	(No sequence) / 011 *has* occurred
S_4	Previous input was 0 / 011 *has* occurred
S_5	Previous input was 01 / 011 *has* occurred
S_6	Previous input was 1 / 011 has not occurred
S_7	Previous input was 10 / 011 has not occurred

State	Next State $X=0$	Next State $X=1$	Z_1Z_2 $X=0$	Z_1Z_2 $X=1$
S_0	S_1	S_6	00	00
S_1	S_1	S_2	00	00
S_2	S_7	S_3	00	01
S_3	S_4	S_3	00	00
S_4	S_4	S_5	00	00
S_5	S_4	S_3	00	01
S_6	S_7	S_6	00	00
S_7	S_1	S_2	10	00

14.14



State	Next State $X_1X_2 = 00\ 01\ 10\ 11$	Z
S_0	$S_1\ S_1\ S_0\ S_0$	0
S_1	$S_1\ S_1\ S_2\ S_0$	0
S_2	$S_2\ S_3\ S_2\ S_3$	1
S_3	$S_2\ S_3\ S_0\ S_3$	1

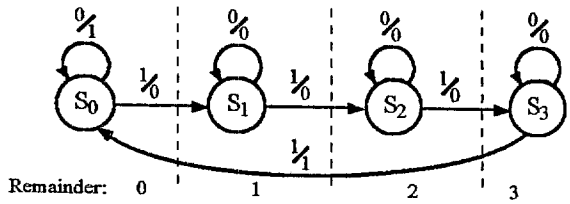
State	Meaning
S_0	$Z = 0$, last input was 10 or 11
S_1	$Z = 0$, last input was 00 or 01
S_2	$Z = 1$, last input was 00 or 10
S_3	$Z = 1$, last input was 01 or 11

Alternate solution has 8 states, similar to problem 14.6:

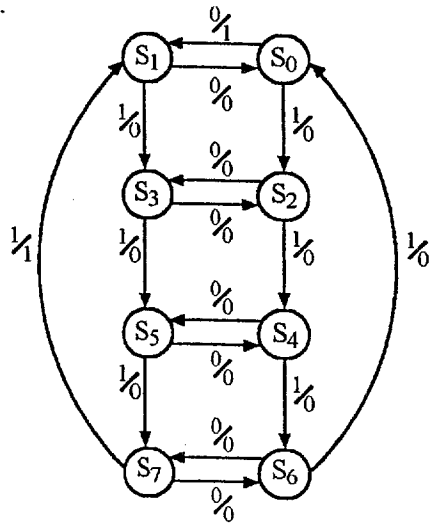
State	Meaning
S_0	$Z = 0$, last input was 10 (reset)
S_1	$Z = 0$, last input was 00
S_2	$Z = 0$, last input was 01
S_3	$Z = 0$, last input was 11
S_4	$Z = 1$, last input was 10
S_5	$Z = 1$, last input was 00
S_6	$Z = 1$, last input was 01
S_7	$Z = 1$, last input was 11

State	Next State $X_1X_2 = 00\ 01\ 10\ 11$	Z
S_0	$S_1\ S_2\ S_0\ S_3$	0
S_1	$S_1\ S_2\ S_4\ S_3$	0
S_2	$S_1\ S_2\ S_4\ S_3$	0
S_3	$S_1\ S_2\ S_0\ S_3$	0
S_4	$S_5\ S_6\ S_4\ S_7$	1
S_5	$S_5\ S_6\ S_4\ S_7$	1
S_6	$S_5\ S_6\ S_0\ S_7$	1
S_7	$S_5\ S_6\ S_0\ S_7$	1

14.15 (a) We need four states to describe the 1's received, as there are four possible remainders when dividing by four. An input of 1 takes us to the next state in cyclic fashion. An input of zero leaves us in the same state.



14.15 (b) Now, expand the state graph into two dimensions: one for 1's and the other for 0's. We need two states to describe the zeros, odd and even.



Left column: odd zeros
Right column: even zeros

First row: Remainder = 0
Second row: Remainder = 1
Third row: Remainder = 2
Fourth row: Remainder = 3
} As part (a)

14.16 (a) We need four states, one for each of the possible past inputs. The next state is just the one that describes that input. The output Z_1 is formed by adding the value of the present state to the present input. Z_2 is found in a similar way:

State	Next State 00	01	10	11	Z_1Z_2 00	01	10	11
S_0	S_0	S_1	S_2	S_3	00	00	00	10
S_1	S_0	S_1	S_2	S_3	00	00	10	11
S_2	S_0	S_1	S_2	S_3	00	10	11	11
S_3	S_0	S_1	S_2	S_3	10	11	11	11

State	Meaning
S_0	Previous input was 00 (0)
S_1	Previous input was 01 (1)
S_2	Previous input was 10 (2)
S_3	Previous input was 11 (3)

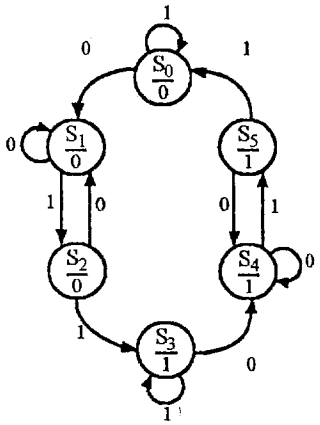
14.16 (b) The Moore version is less intuitive. Again, we need a state for each past input. We do not, however, need a state for every possible output (this would give $4 \times 4 = 16$ states) since some outputs never occur. For instance, if the last input was zero, Z_2 can never be 1, because anything multiplied by zero is zero. In fact, only ten states are needed:

Note: The output can never be 01. If two integers between 0 and 3 multiply to a number greater than 2, their sum is also greater than 2, i.e. $(Z_2 = 1) \Rightarrow (Z_1 = 1)$

Previous Input	State	X_1X_2 00	01	10	11	Z_1Z_2
00	S_0	S_0	S_2	S_5	S_8	00
00	S_1	S_0	S_2	S_5	S_8	10
01	S_2	S_0	S_2	S_6	S_9	00
01	S_3	S_0	S_2	S_6	S_9	10
01	S_4	S_0	S_2	S_6	S_9	11
10	S_5	S_0	S_3	S_7	S_9	00
10	S_6	S_0	S_3	S_7	S_9	10
10	S_7	S_0	S_3	S_7	S_9	11
11	S_8	S_1	S_4	S_7	S_9	10
11	S_9	S_1	S_4	S_7	S_9	11

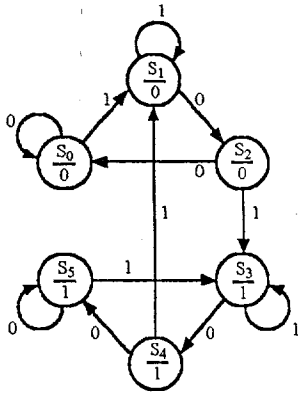
14.17 There are two identical parts: one with an output of 0 and one with an output of 1.

State	Meaning
S_1, S_4	Previous input was 0
S_2, S_5	Previous inputs were 01
S_3, S_0	Previous input was 1 / Reset (S_0)

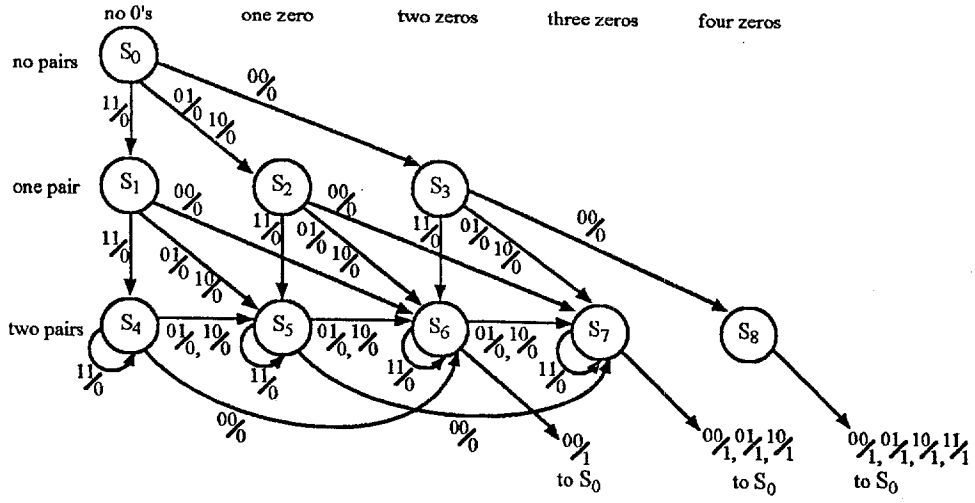


14.18 There are two identical parts: one with an output of 0 and one with an output of 1.

State	Meaning
S_0	Reset
S_1	Previous input was 1
S_2	Previous inputs were 10
S_3	Previous inputs were 101 (first 101)
S_4	Previous inputs were 10 (start of second 101)
S_5	Previous inputs were 00



14.19 This is another problem similar to 14.10. Plot the number of 0's horizontally and the number of pairs vertically:

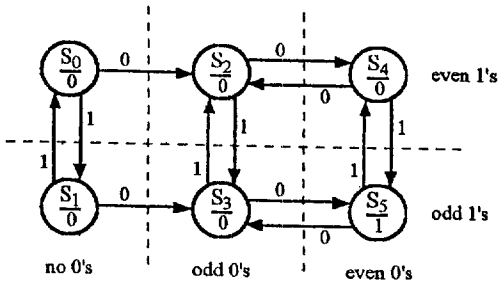


Pairs	0's	Present State	Next State 00 01 10 11	Z_1Z_2 00 01 10 11
0	0	S_0	S_3 S_2 S_2 S_1	0 0 0 0
1	0	S_1	S_6 S_5 S_5 S_4	0 0 0 0
1	1	S_2	S_7 S_6 S_6 S_5	0 0 0 0
1	2	S_3	S_8 S_7 S_7 S_6	0 0 0 0
2	0	S_4	S_6 S_5 S_5 S_4	0 0 0 0
2	1	S_5	S_7 S_6 S_6 S_5	0 0 0 0
2	2	S_6	S_0 S_7 S_7 S_6	1 0 0 0
2	3	S_7	S_0 S_0 S_0 S_7	1 1 1 0
2	4	S_8	S_0 S_0 S_0 S_0	1 1 1 1

Note: There is a seven-state solution.

14.20 0's are plotted horizontally. 1's are plotted vertically.

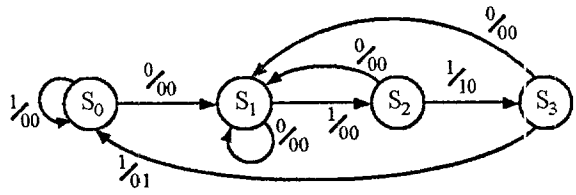
State	Next State $X=0$	$X=1$	Z
S_0	S_2	S_1	0
S_1	S_3	S_0	1
S_2	S_4	S_3	0
S_3	S_5	S_2	1
S_4	S_2	S_5	0
S_5	S_3	S_4	1



14.21

State	Next State $X=0$	Next State $X=1$	Z_1Z_2 $X=0$	Z_1Z_2 $X=1$
S_0	S_1	S_0	00	00
S_1	S_1	S_2	00	00
S_2	S_1	S_3	00	10
S_3	S_1	S_0	00	01

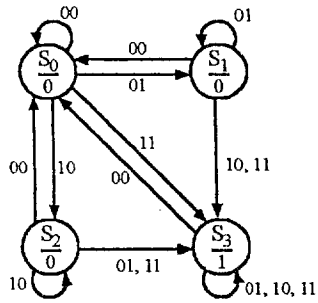
State	Meaning
S_0	Reset, 0111
S_1	0
S_2	01
S_3	011



14.22

State	X_1X_2 00	01	10	11	Z
S_0	S_0	S_1	S_2	S_3	0
S_1	S_0	S_1	S_2	S_3	0
S_2	S_0	S_3	S_2	S_3	0
S_3	S_0	S_3	S_3	S_3	1

State	Meaning
S_0	Reset
S_1	Previous input was 01, $Z=0$
S_2	Previous input was 10, $Z=0$
S_3	$Z=1$ (Until input 00)

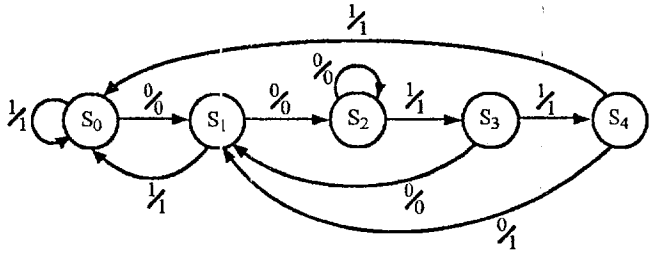


14.23 Example: $X=0\,0\,1\,1\,0\,0\,1\,1\,0\,1\,0\,1$
$Z=0\,0\,1\,1\,1\,0\,1\,1\,1\,1\,0\,1$

Note: Overlapping sequences are allowed.

State	Meaning
S_0	No sequence
S_1	0
S_2	00
S_3	001
S_4	0011

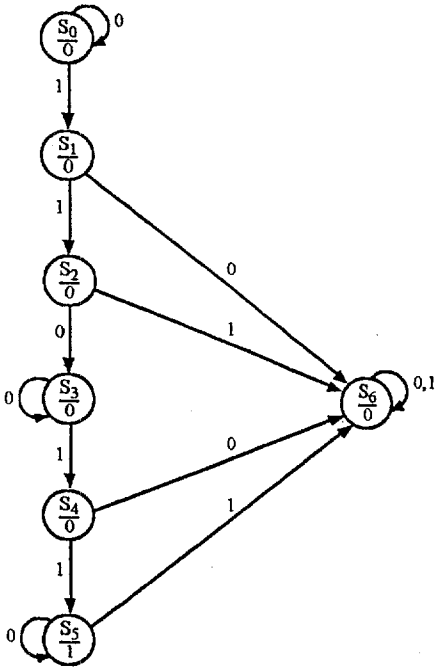
State	Next State $X=0$	Next State $X=1$	Z $X=0$	Z $X=1$
S_0	S_1	S_0	0	1
S_1	S_2	S_0	0	1
S_2	S_2	S_3	0	1
S_3	S_1	S_4	0	1
S_4	S_1	S_0	1	1



14.24

State	Next State $X=0$	Next State $X=1$	Z
S_0	S_0	S_1	0
S_1	S_6	S_2	0
S_2	S_3	S_6	0
S_3	S_3	S_4	0
S_4	S_6	S_5	0
S_5	S_5	S_6	1
S_6	S_6	S_6	0

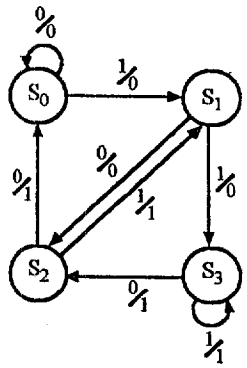
State	Meaning
S_0	No 1's
S_1	One 1 in first group
S_2	Two 1's in first group
S_3	First group 11 complete, had exactly two 1's
S_4	One 1 in second group
S_5	Two 1's in second group ($Z = 1$)
S_6	"Disqualified" state ($Z = 0$)



14.25 To delay by two clock periods, we need to remember the previous two inputs. So we have four states, one for each combination of two inputs:

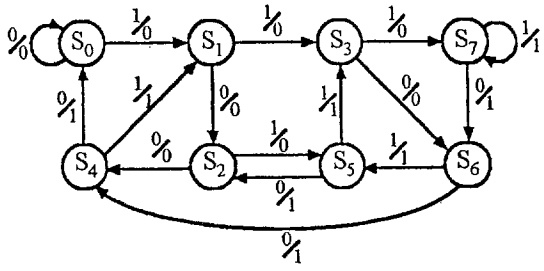
State	Next State $X=0$	Next State $X=1$	Z $X=0$	Z $X=1$
S_0	S_0	S_1	0	0
S_1	S_2	S_3	0	0
S_2	S_0	S_1	1	1
S_3	S_2	S_3	1	1

State	Meaning
S_0	Previous two inputs were 00
S_1	Previous two inputs were 01
S_2	Previous two inputs were 10
S_3	Previous two inputs were 11



Note: Just go to the state that represents the last two inputs.

14.26 This is the same as 14.25, except that we need to remember the last three inputs. So we have eight states:

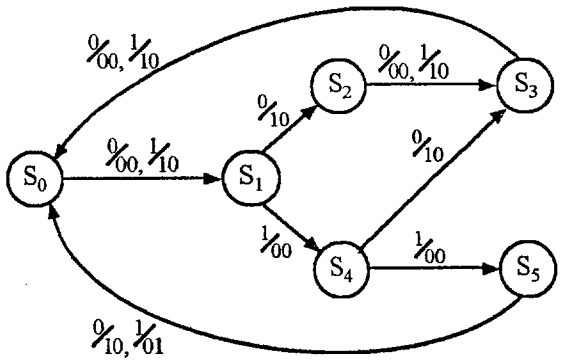


State	Next State $X=0$	Next State $X=1$	Z $X=0$	Z $X=1$
S_0	S_0	S_1	0	0
S_1	S_2	S_3	0	0
S_2	S_4	S_5	0	0
S_3	S_6	S_7	0	0
S_4	S_0	S_1	1	1
S_5	S_2	S_3	1	1
S_6	S_4	S_5	1	1
S_7	S_6	S_7	1	1

Note: The state number expressed in binary gives the last 3 inputs.

14.27

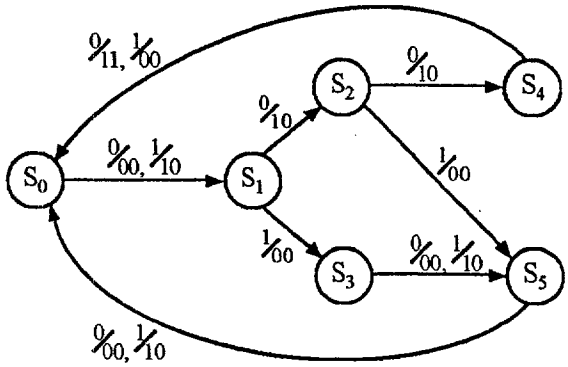
State	Next State $X=0$	Next State $X=1$	SV $X=0$	SV $X=1$
S_0	S_1	S_1	00	10
S_1	S_2	S_4	10	00
S_2	S_3	S_3	00	10
S_3	S_0	S_0	00	10
S_4	S_3	S_5	10	00
S_5	S_0	S_0	10	01



State	Meaning
S_0	No bits received
S_1	One bit received
S_2	Two bits received; Carry-in = 0
S_4	Two bits received; Carry-in = 1
S_3	Three bits received; Carry-in = 0
S_5	Three bits received; Carry-in = 1

14.28

State	Next State $X=0$	Next State $X=1$	DB $X=0$	DB $X=1$
S_0	S_1	S_1	00	10
S_1	S_2	S_3	10	00
S_2	S_4	S_5	10	00
S_3	S_5	S_5	00	10
S_4	S_0	S_0	11	00
S_5	S_0	S_0	00	10

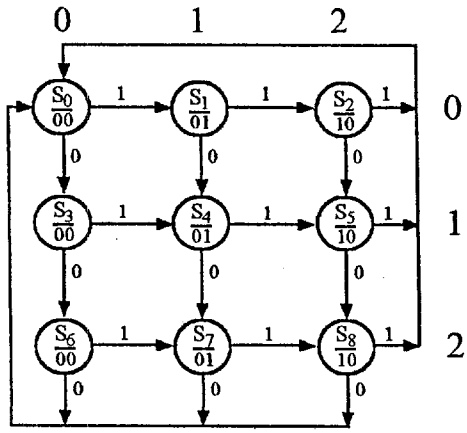


State	Meaning
S_0	No bits received
S_1	One bit received
S_2	Two bits received; Borrow-in = 1
S_4	Two bits received; Borrow-in = 0
S_3	Three bits received; Borrow-in = 1
S_5	Three bits received; Borrow-in = 0

14.29 This is similar to 14-15, and should be answered in the same way. See the solution to 14-15 for more information.

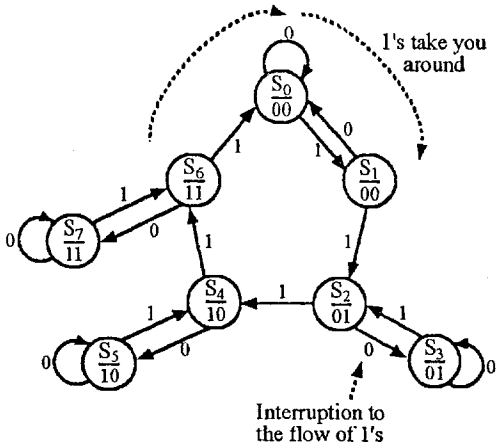
Horizontally: Number of 1's modulo 3
Vertically: Number of 0's modulo 3.

State	Next State $X=0$	Next State $X=1$	YZ
S_0	S_3	S_1	00
S_1	S_4	S_2	01
S_2	S_5	S_0	10
S_3	S_6	S_4	00
S_4	S_7	S_5	01
S_5	S_8	S_0	10
S_6	S_0	S_7	00
S_7	S_0	S_8	01
S_8	S_0	S_0	10



14.30 This problem is essentially a circular counting exercise. Pairs of 1's take you further around the state graph. Pairs can overlap, so if the last input was a 1, and the present input is a 1, you move on. If the sequence is interrupted, you branch off while you wait for the next 1. Then, you go back to the cycle of counting.

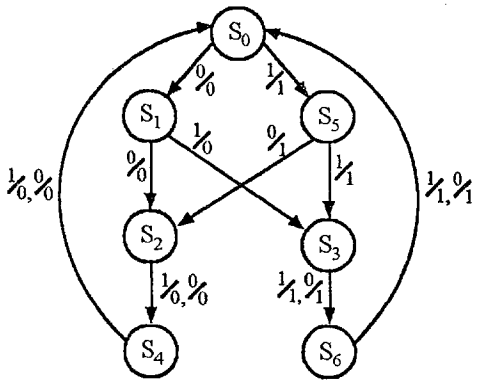
State	Next State $X=0$	Next State $X=1$	YZ
S_0	S_0	S_1	00
S_1	S_0	S_2	00
S_2	S_3	S_4	01
S_3	S_3	S_2	01
S_4	S_5	S_6	10
S_5	S_5	S_4	10
S_6	S_7	S_0	11
S_7	S_7	S_6	11



14.31 We notice that input *ABXX* becomes output *AABB*. It can be seen that it is not necessary to remember both *A* and *B* at once. We remember *A* for the first two clocks and *B* for the next two. Notice that if the output were, say, *ABAB*, we could not do this.

State	Next State $X=0$	Next State $X=1$	Z $X=0$	Z $X=1$
S_0	S_1	S_5	0	1
S_1	S_2	S_3	0	0
S_2	S_4	S_4	0	0
S_3	S_6	S_6	1	1
S_4	S_0	S_0	0	0
S_5	S_2	S_3	1	1
S_6	S_0	S_0	1	1

State	Meaning
S_0	Reset
S_1	$A=0$
S_5	$A=1$
S_2, S_4	$B=0$
S_3, S_6	$B=1$



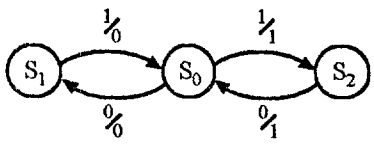
14.32 This problem is simply addition. We need a state to describe every possible sum of money entered, i.e., 0¢ to 45¢ in 5¢ intervals.

Just go to the state with the correct sum. The 25¢ state dispenses the product ($R = 1$) and resets. States above this in value cascade down to S_5 by giving out a nickel. When they get to S_5, the product is dispensed.

$	Present State	*NDQ* 000	100	010	001	*RC*
.00	S_0	S_0	S_1	S_2	S_5	00
.05	S_1	S_1	S_2	S_3	S_6	00
.10	S_2	S_2	S_3	S_4	S_7	00
.15	S_3	S_3	S_4	S_5	S_8	00
.20	S_4	S_4	S_5	S_6	S_9	00
.25	S_5	S_0	-	-	-	10
.30	S_6	S_5	-	-	-	01
.35	S_7	S_6	-	-	-	01
.40	S_8	S_7	-	-	-	01
.45	S_9	S_8	-	-	-	01

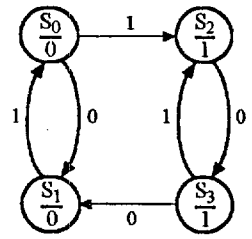
14.33 (a) Look at Figure 14-19, FLD p. 408, to see that
Manchester 01 gives NRZ 00
Manchester 10 gives NRZ 11

Other Manchester inputs are presumed not to occur.



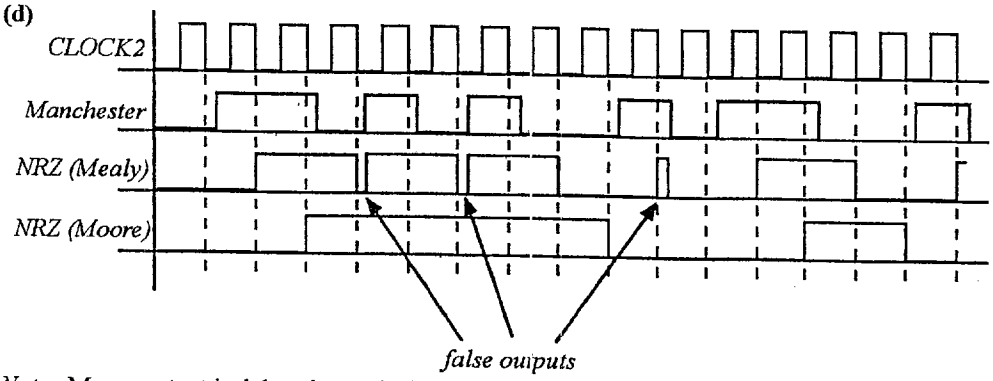
State	Next State $X=0$	$X=1$	Z $X=0$	$X=1$
S_0	S_1	S_2	0	1
S_1	-	S_0	0*	0
S_2	S_0	-	1	1*
* Filled in to prevent False outputs.				

14.33 (b) This is the same as the Mealy, except that we need two reset states, one with an output of zero, the other with an output of 1. Invalid inputs never occur.



State	Next State $X=0$	$X=1$	Z
S_0	S_1	S_2	0
S_1	-	S_0	0
S_2	S_3	-	1
S_3	S_1	S_2	1

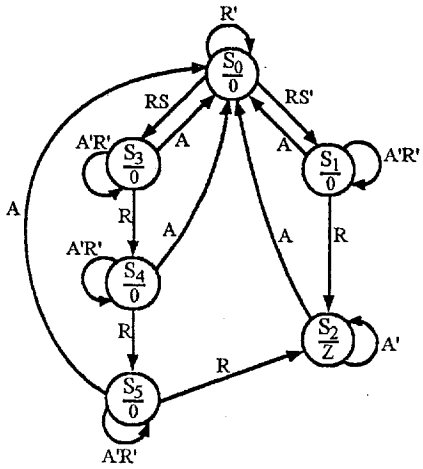
14.33 (c), (d)



Note: Moore output is delayed one clock cycle of *CLOCK2*.

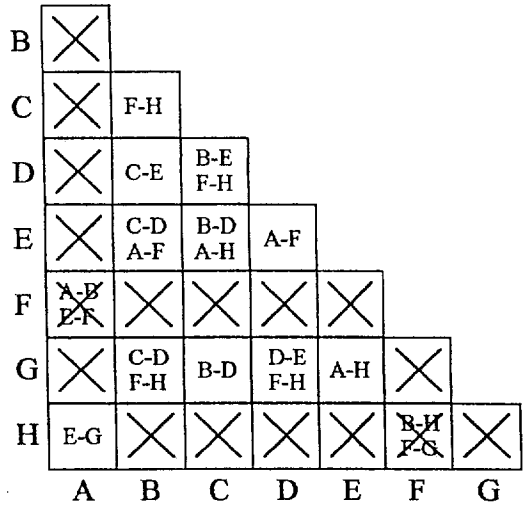
14.34

State	Meaning
S_0	Reset
S_1	One ring, waiting for two (or answer)
S_3, S_4, S_5	One, two, or three rings, respectively; waiting for four (or answer)
S_2	Activate answering machine; wait for it to answer

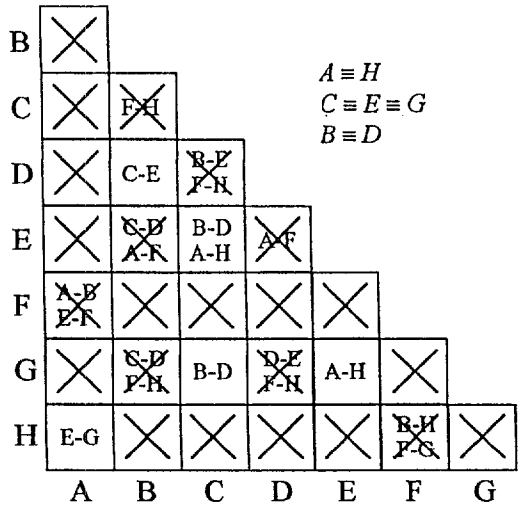


Unit 15 Problem Solutions

15.1 (a) Implication chart after one pass:



Complete implication chart



$A \equiv H$
$C \equiv E \equiv G$
$B \equiv D$

Reduced state table:

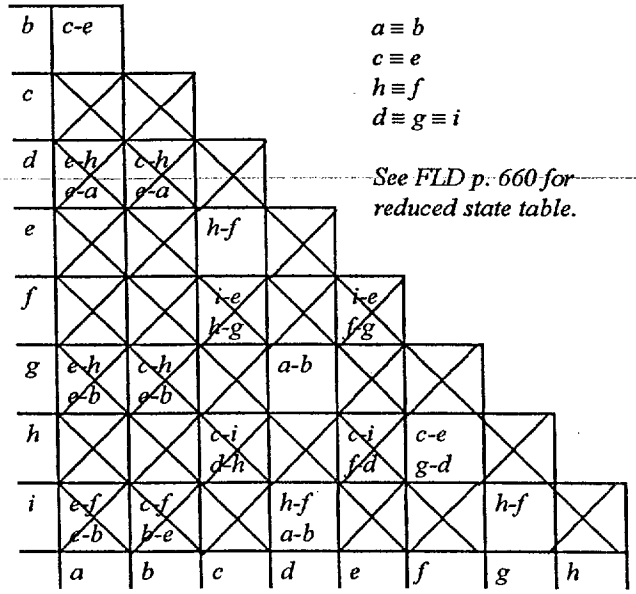
State	Next State $X=0$	Next State $X=1$	Output $X=0$	Output $X=1$
A	A	C	1	0
B	C	F	0	0
C	B	A	0	0
F	B	F	1	0

15.1 (b) $B \equiv C$ because $F \equiv H$, (and also because $C \equiv D$)
$F \equiv H$ because $B \equiv H$, (and also because $F \equiv G$), and
$B \equiv H$ because the output differs for $X = 0$.
So use the sequence $\underline{X} = 100$.

Input:	X:	1	0	0
Starting in B:	Z:	0	1	0
	State: (B)	F	B	
Starting in G:	Z:	0	1	1
	State: (G)	H	H	

So $\lambda_1(B, 100) = 010 \neq 011 = \lambda_2(G, 100)$, and $B \equiv G$.
(Alternative: $\lambda_1(B, 110) = 001 \neq 000 = \lambda_2(G, 110)$.
Also, $\lambda_1(B, 00101) \equiv \lambda_2(G, 00101)$, but this requires an $\underline{X}$ of length 5.

15.2



$a \equiv b$
$c \equiv e$
$h \equiv f$
$d \equiv g \equiv i$

See FLD p. 660 for reduced state table.

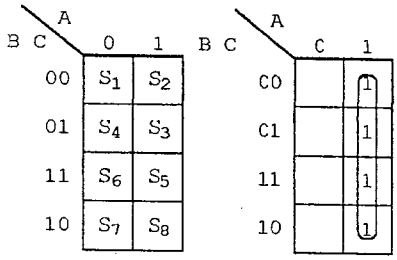
15.5 (b) Excluding 0000, there are 7 possible columns. All possible non-repeating combinations are given here. Those with repeating rows are crossed out; 29 assignments remain to try.

~~0 0 0~~	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
~~0 0 0~~	0 0 1	0 0 1	0 0 1	0 0 1	0 0 1	0 0 1	0 0 1
~~0 1 1~~	0 1 0	0 1 0	0 1 1	0 1 1	0 1 0	0 1 0	0 1 1
~~1 0 1~~	1 0 0	1 0 1	1 0 0	1 0 1	1 1 0	1 1 1	1 1 0
(1 2 3)	(1 2 4)	(1 2 5)	(1 2 6)	(1 2 7)	(1 3 4)	(1 3 5)	(1 3 6)
0 0 0	~~0 0 0~~	0 0 0	0 0 0	0 0 0	0 0 0	~~0 0 0~~	0 0 0	0 0 0
0 0 1	~~0 1 1~~	0 1 1	0 1 1	0 1 1	0 1 1	~~0 1 1~~	0 0 1	0 0 1
0 1 1	~~0 0 0~~	0 0 1	0 0 1	0 0 1	0 0 1	~~0 1 1~~	1 1 0	1 1 0
1 1 1	~~1 0 1~~	1 0 0	1 0 1	1 1 0	1 1 1	~~1 0 1~~	0 1 0	0 1 1
(1 3 7)	(1 4 5)	(1 4 6)	(1 4 7)	(1 5 6)	(1 5 7)	(1 6 7)	(2 3 4)	(2 3 5)
0 0 0	0 0 0	0 0 0	~~0 0 0~~	0 0 0	0 0 0	~~0 0 0~~	0 0 0	0 0 0
0 0 1	0 0 1	0 1 1	~~0 1 1~~	0 1 1	0 1 1	~~0 1 1~~	0 1 1	0 1 1
1 1 1	1 1 1	1 0 0	~~1 0 1~~	1 0 1	1 0 1	~~1 0 1~~	1 1 1	1 0 0
0 1 0	0 1 1	0 0 1	~~0 0 0~~	0 0 1	0 1 0	~~0 1 1~~	0 0 1	1 0 1
(2 3 6)	(2 3 7)	(2 4 5)	(2 4 6)	(2 4 7)	(2 5 6)	(2 5 7)	(2 6 7)	(3 4 5)
0 0 0	~~0 0 0~~	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
0 1 1	~~0 1 1~~	0 1 1	0 1 1	0 1 1	1 1 1	1 1 1	1 1 1	1 1 1
1 0 1	~~1 0 1~~	1 0 1	1 0 1	1 1 1	0 0 1	0 0 1	0 1 1	0 1 1
1 0 0	~~1 0 1~~	1 1 0	1 1 1	1 0 1	0 1 0	0 1 1	0 0 1	1 0 1
(3 4 6)	(3 4 7)	(3 4 6)	(3 5 7)	(3 6 7)	(4 5 6)	(4 5 7)	(4 6 7)	(5 6 7)

15.6 (a) Group (S_1, S_4, S_6, S_7) and (S_2, S_3, S_5, S_8).

One possible assignment:

$S_1 = 000$	$S_5 = 111$
$S_2 = 100$	$S_6 = 011$
$S_3 = 101$	$S_7 = 010$
$S_4 = 001$	$S_8 = 110$

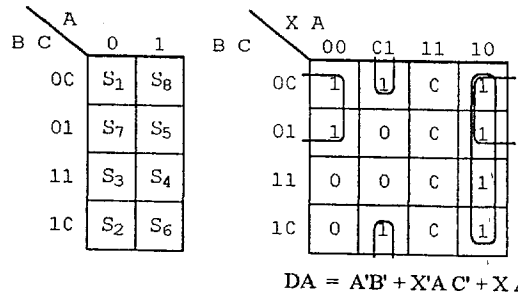


Z = A

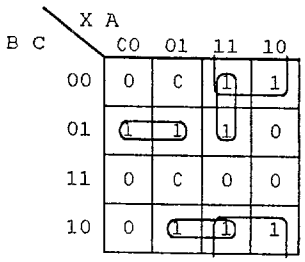
15.6 (b) I: (S_3, S_4)✓ (S_1, S_8)✓ (S_3, S_7)✓ (S_5, S_8)✓
II: (S_4, S_5)✓ (S_1, S_6) (S_7, S_8) (S_1, S_7)✓ (S_2, S_3)✓ (S_2, S_4) (S_6, S_8)✓ (S_3, S_5)
Adjacencies that are satisfied are checked (✓)

One possible assignment:

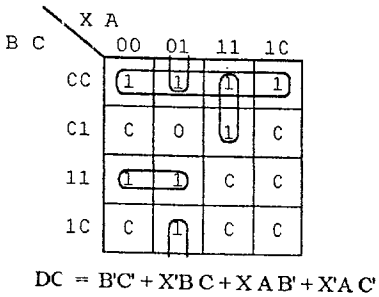
$S_1 = 000$	$S_5 = 101$
$S_2 = 010$	$S_6 = 110$
$S_3 = 011$	$S_7 = 001$
$S_4 = 111$	$S_8 = 100$



DA = A'B' + X'A C' + X A'



DB = X'B'C + A B C' + X C' + X A B'



DC = B'C' + X'B C + X A B' + X'A C'

State	ABC	$A^+B^+C^+$ $X=0$	$X=1$
S_1	000	101	111
S_7	001	110	100
S_2	010	000	110
S_3	011	001	100
S_8	100	101	011
S_5	101	010	011
S_6	110	111	010
S_4	111	001	000

15.7 (a) Guidelines:

1. $(A, D, F)\ (C, E)\ (A, D)\ (C, E)\ (B, F)$
2. $(F, D)\times 2\ (D, B)\ (A, C)\times 2\ (B, F)$
3. $(A, B, D, F)\ (C, E)$

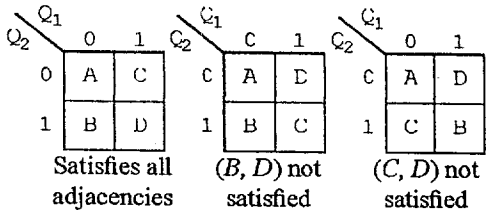
See FLD p. 661 for one good solution.

15.7 (b) *See FLD p. 661 for solution.*

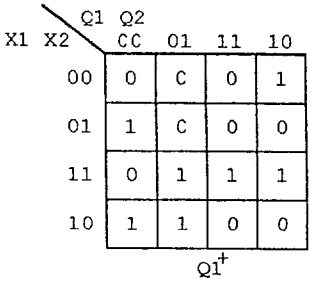
15.8 (a) Guidelines:

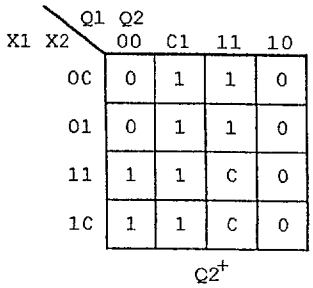
1. $(B, D)\times 2\ (C, D)\times 2\ (A, B)$
2. $(B, D)\ (A, C)\ (A, C, B)\ (A, B, C, D)$
3. $(A, B)\times 2\ (B, D)\times 2\ (C, D)\times 2$

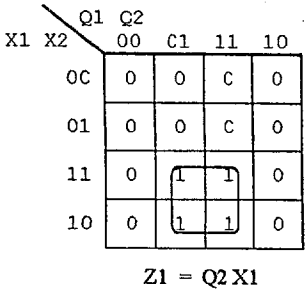
Best assignment: $A = 00, B = 01, C = 10, D = 11$

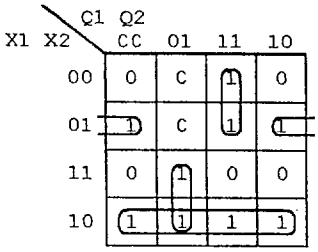


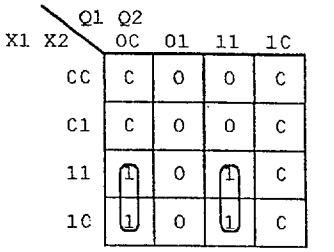
15.8 (b)

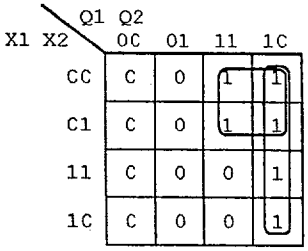








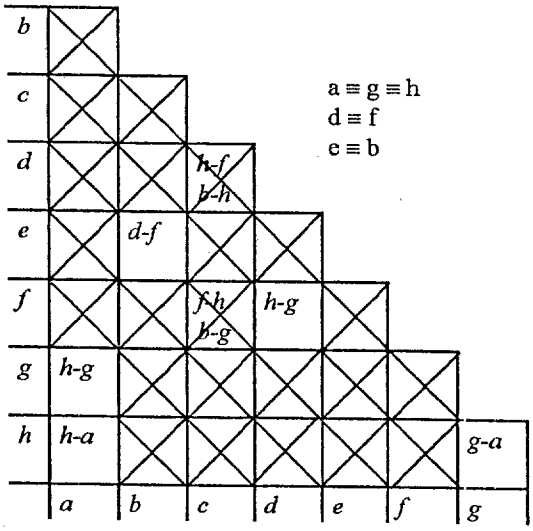




15.9 *See FLD p. 661 for solution using* Q_1, Q_2, *and* Q_3.
Alternate solution using Q_0, Q_1 and Q_2:

$D_0 = X'Q_0 + XY'Q_2$
$D_1 = XQ_0 + XQ_2 + X'Q_1$
$D_2 = YQ_1 + X'Y'Q_2$
$P = XQ_0 + X'Q_2 + XQ_1$
$\quad = XQ_0 + Q_2$
$S = X'Q_0 + XY'Q_2$

15.10 (a)



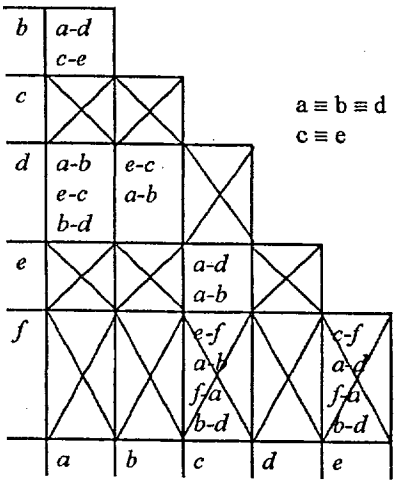
State	Next State $X=0$	Next State $X=1$	Output $X=0$	Output $X=1$
a	a	c	1	0
b	c	d	0	1
c	a	b	0	0
d	d	a	0	0

15.10 (b) Input: 00

Output starting in state c: 01 (state $c \xrightarrow{0}$ state $a \xrightarrow{0}$ state a)

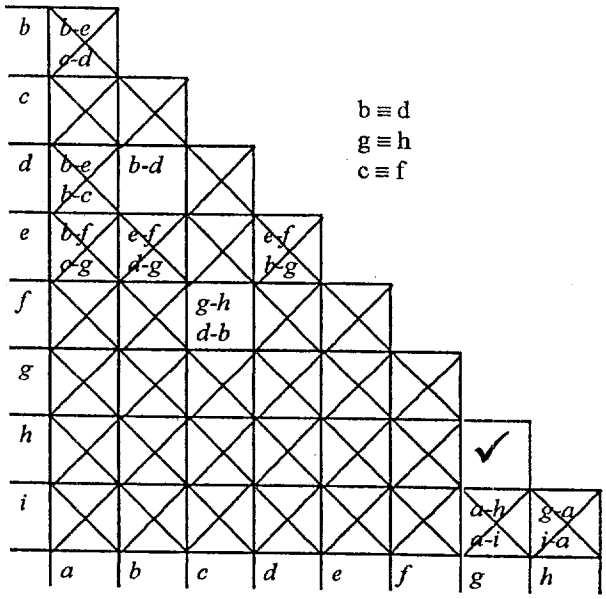
Output starting in state d: 00 (state $d \xrightarrow{0}$ state $d \xrightarrow{0}$ state d)

15.11 (a)



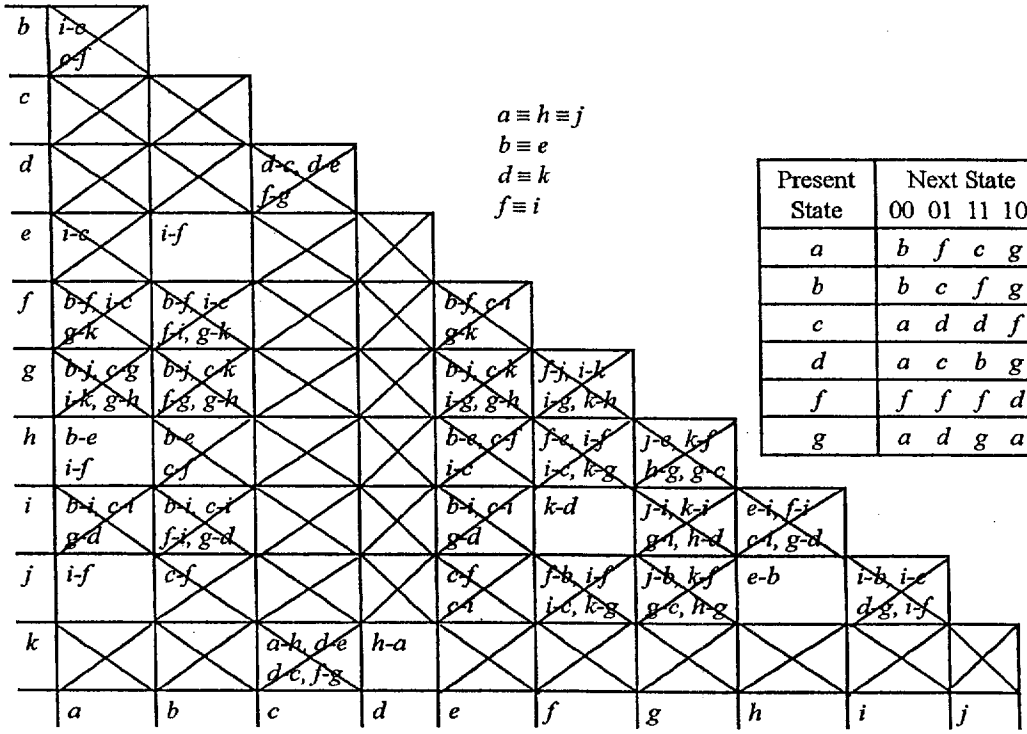
Present State	Next State 00	01	11	10	Z
a	a	c	c	a	0
c	c	a	f	a	1
f	f	a	a	a	1

15.11 (b)



State	Next State $X=0$	Next State $X=1$	Z $X=0$	Z $X=1$
a	b	c	1	0
b	e	b	1	0
c	g	b	1	1
e	c	g	1	0
g	g	i	0	1
i	a	a	0	1

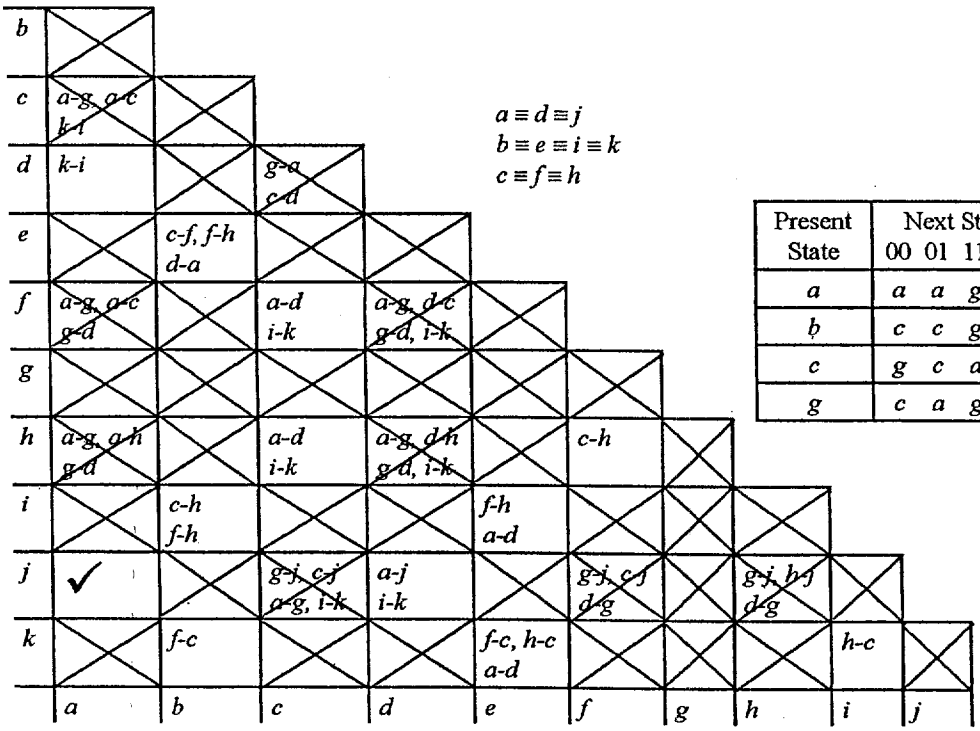
15.12 (a)



$a \equiv h \equiv j$
$b \equiv e$
$d \equiv k$
$f \equiv i$

Present State	Next State 00 01 11 10	Z
a	*b f c g*	0
b	*b c f g*	0
c	*a d d f*	1
d	*a c b g*	1
f	*f f f d*	0
g	*a d g a*	0

15.12 (b)



$a \equiv d \equiv j$
$b \equiv e \equiv i \equiv k$
$c \equiv f \equiv h$

Present State	Next State 00 01 11 10	Z 00 01 11 10
a	*a a g b*	1 0 0 0
b	*c c g a*	0 0 0 0
c	*g c a b*	1 0 0 0
g	*c a g b*	0 1 0 0

15.13 (a) $S_0 \equiv e \equiv f, S_1 \equiv c \equiv d, S_2 \equiv S_3 \equiv a \equiv b$

Since every state in N has an equivalent state in M, and vice versa, N and M are equivalent.

S_0			$E-S_3$ $D-S_1$	$E-S_3$ $C-S_1$	$B-S_3$ $D-S_1$	$B-S_3$ $C-S_1$
S_1			$E-S_0$ $D-S_1$	$E-S_0$ $C-S_1$	$B-S_0$ $D-S_1$	$B-S_0$ $C-S_1$
S_2	$E-S_0$ $A-S_2$	$F-S_0$ $B-S_2$				
S_3	$E-S_0$ $A-S_3$	$F-S_0$ $B-S_3$				
	A	B	C	D	E	F

15.13 (b)

M

	X=0	1	
S_0	S_2	S_1	0
S_1	S_0	S_1	0
S_2	S_0	S_2	1

$S_2 \equiv S_3$

N

	X=0	1	
A	E	A	1
C	E	C	0
E	A	C	0

$E \equiv F, C \equiv D, A \equiv B$

Note: $S_2 \equiv A$
$S_1 \equiv C$
$S_0 \equiv E$

15.14 (a) Set don't care to S_3 so $S_4 \equiv S_3$:

Present State	Next State X=0	1	Output
S_0	S_1	S_0	0
S_1	S_0	S_2	0
S_2	S_3	S_4	1
S_3	S_0	S_3	0

Set don't care to S_2 so $S_4 \equiv S_1$:

Present State	Next State X=0	1	Output
S_0^1	S_1^1	S_0^1	0
S_1^1	S_0^1	S_2^1	0
S_2^1	S_3^1	S_2^1	1
S_3^1	S_0^1	S_3^1	0

15.14 (b)

S_0	$S_1-S_1^1$ $S_0-S_0^1$	$S_1-S_0^1$ $S_0-S_2^1$		$S_1-S_0^1$ $S_0-S_3^1$
S_1	$S_0-S_1^1$ $S_2-S_0^1$	$S_0-S_0^1$ $S_2-S_2^1$		$S_0-S_0^1$ $S_2-S_3^1$
S_2			$S_3-S_3^1$ $S_3-S_2^1$	
S_3	$S_0-S_1^1$ $S_3-S_0^1$	$S_0-S_0^1$ $S_3-S_2^1$		$S_0-S_0^1$ $S_3-S_3^1$
	S_0^1	S_1^1	S_2^1	S_3^1

S_2 and S_2^1 have no corresponding states,
$\therefore$ N and N' are not equivalent.

15.14 (c) $X =$ 0 1 1
$Z =$ (0) 0 1 1
$Z^1 =$ (0) 0 1 0

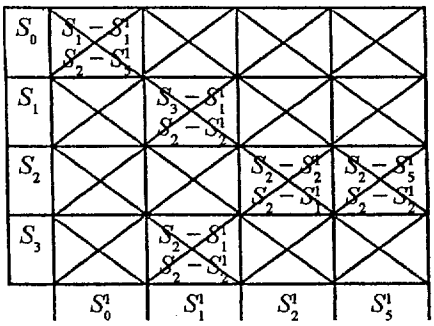
15.15 (a) Set don't care to 0 so $S_2 \equiv S_4 \equiv S_5$:

Present State	Next State X=0	1	Output $X=0$	$X=1$
S_0	S_1	S_2	0	0
S_1	S_3	S_2	1	1
S_2	S_2	S_2	0	1
S_3	S_2	S_2	1	1

Set don't care to 0 so $S_1 \equiv S_3 \equiv S_4$:

Present State	Next State X=0	1	Output $X=0$	$X=1$
S_0^1	S_1^1	S_5^1	0	0
S_1^1	S_1^1	S_2^1	1	1
S_2^1	S_2^1	S_1^1	0	1
S_5^1	S_5^1	S_2^1	0	1

15.15 (b)



No equivalent states.

15.15 (c) $X = 1\ 0$
$Z = 0\ 1$
$Z^1 = 0\ 0$

15.16 (a) Invert all three columns of assignment (*iv*), and then swap the first and last columns. Then (*iii*) and (*iv*) are the same, therefore, Assignment (*iii*) ≡ Assignment (*iv*).

15.16 (b) Equivalent assignments to each column having 000 as the starting state. Invert any column with 1 in the first row.

	(*ii*) - (c_2')	*iii* - c_1'	*iv* - $c_1'c_2'$	*v* - c_3'
S_0	000	000	000	000
S_1	101	001	100	110
S_2	011	100	001	100
S_3	100	101	101	010
S_4	010	011	110	001
S_5	110	010	010	011

15.16 (c) Many state assignments are not equivalent to (i) through (v), for example:

101	or	011
000		101
011		000
100		100
010		010
110		110

15.17 (a)

Straight Binary Assignment	Equivalent State Assignments (any three)				
	$c_2 \leftrightarrow c_3$	$c_1 \leftrightarrow c_3$	$c_1 \leftrightarrow c_2$	$c_1 \rightarrow c_3 \rightarrow c_2 \rightarrow c_1$	$c_1 \rightarrow c_2 \rightarrow c_3 \rightarrow c_1$
000	000	000	000	000	000
001	001	100	010	010	100
010	100	010	001	100	001
011	101	110	011	110	101
100	010	001	100	001	010
101	011	101	110	011	110
110	110	011	101	101	011
111	111	111	111	111	111

15.17 (b) Many state assignments are not equivalent to the straight binary assignment, for example:

111	111	etc.
101	001	
110	010	
100	011	
011	100	
010	101	
001	000	
000	110	

15.18 (a) 1. (*A, H*) (*B, G*) (*A, D*) (*E, G*)
2. (*D, G*) (*E, H*) (*B, F*) (*F, G*) (*C, A*) (*H, C*) (*E, A*) (*D, B*)
3. (*A, C, E, G*) (*B, D, F, H*)

Consider Guideline #3 only:

Q_2Q_3 \ Q_1	0	1
00	B	A
01	D	C
11	F	E
10	H	G

Q_2Q_3 \ Q_1	0	1
00	0	1
01	0	1
11	0	1
10	0	1

$Z = Q_1$

15.18 (b) Consider Guidelines #1, 2:
A = 000, B = 111, C = 110, D = 001, E = 010, F = 101, G = 011, H = 100

D_2D_3 \ D_1	0	1
00	A	H
01	D	F
11	G	B
10	E	C

Q2 Q3 \ X1 Q1	00	01	11	10
00	C	C	1	0
01	1	1	1	0
11	C	C	1	0
10	1	1	1	0

D1 = X1'Q2'Q3 + X1'Q2 Q3' + X1 Q1

Q2 Q3 \ X Q1	00	01	11	10
00	C	0	1	1
01	C	0	1	1
11	1	1	0	C
10	1	1	0	C

D2 = X'Q2 + X Q2'

Q2 Q3 \ X Q1	00	01	11	10
00	1	1	1	1
01	1	0	0	1
11	C	0	0	C
10	C	1	1	C

D3 = Q1'Q2' + Q1 Q3'

15.19 (a) 1. (*A, C*)×2✓ (*B, C*)×2✓ (*A, D*)✓
2. (*A, C*)✓ (*B, D*)✓ (*A, B, D*)✓ (*A, B, C, D*)✓
3. (*A, D*)✓
Adjacencies that are satisfied are checked (✓)

Q2 \ Q1	0	1
0	A	C
1	D	B

Q_1Q_2	$Q_1^+Q_2^+$ 00 01 11 10	Z_1Z_2 00 01 11 10
00	00 00 10 10	01 01 01 01
11	11 01 11 01	11 11 11 11
10	00 00 11 01	11 11 00 00
01	01 11 00 10	01 01 01 01

15.19 (b)

Q1 Q2 \ X1 X2	00	01	11	10
00	C	C	1	1
01	C	1	0	1
11	1	C	1	0
10	C	C	1	0

$Q1^+$

Q1 Q2 \ X1 X2	00	01	11	10
00				
01	1	1		
11	1	1	1	1
10			1	1

$Q2^+$

15.19 (b) (contd)

Q1 Q2 \ X1 X2	00	01	11	10
00	0	C	1	1
01	0	1	0	1
11	X	X	X	X
10	X	X	X	X

J1 = X1'X2 Q2 + X1 X2' + X1 Q2'

Q1 Q2 \ X1 X2	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	C	1	0	1
10	1	1	0	1

K1 = X1'X2 + X1 X2' + X2'Q2'
K1 = X1'X2 + X1 X2' + X1'Q2'

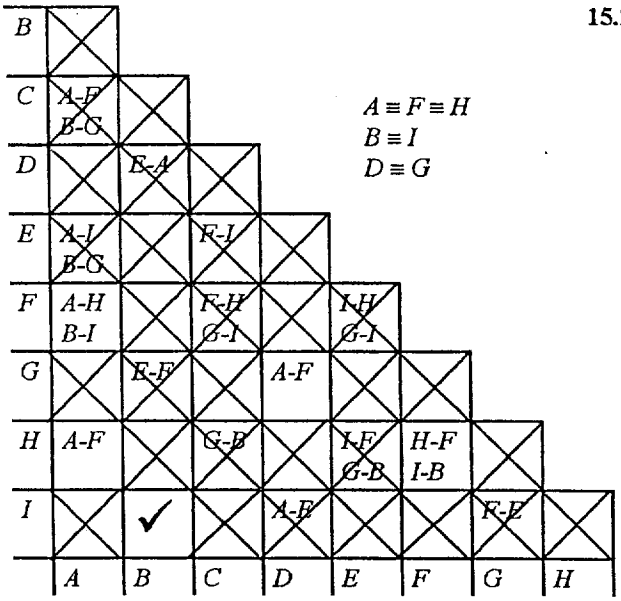
Q1 Q2 \ X1 X2	00	01	11	10
00	0	C	0	0
01	X	X	X	X
11	X	X	X	X
10	0	C	1	1

J2 = X1 Q1

Q1 Q2 \ X1 X2	00	01	11	10
00	X	X	X	X
01	0	C	1	1
11	0	C	0	0
10	X	X	X	X

K2 = X1 Q1'

15.20 (a)



$A \equiv F \equiv H$
$B \equiv I$
$D \equiv G$

State	Next State $X=0$	Next State $X=1$	X
A	A	B	1
B	C	E	0
C	A	D	1
D	C	A	0
E	B	D	1

15.20 (b) 1. (A, C)✓ (B, D)✓ (C, E)✓
2. (A, B)✓ (C, E)✓ (A, D) (A, C)✓ (B, D)✓
3. (A, C, E)✓ (B, D)✓

Adjacencies that are satisfied are checked (✓)
$A = 000, B = 100, C = 001, D = 101, E = 011$
All are satisfied except (A, D)

Alternate:

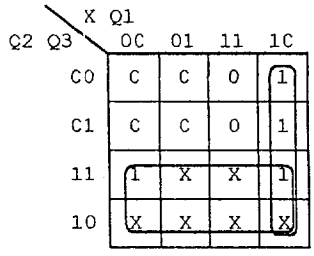
Q_2Q_3 \ Q_1	0	1
00	A	B
01	C	D
11	E	
10		

Q_2Q_3 \ Q_1	0	1
00	A	
01	C	
11	E	
10	B	D

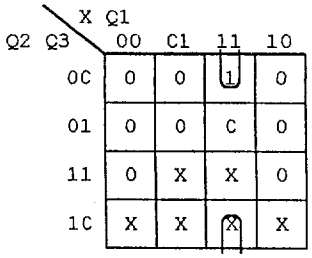
15.20 (c)

$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ $X=0$	$X=1$	Z
000	000	100	1
100	001	011	0
001	000	101	1
101	001	000	0
011	100	101	1

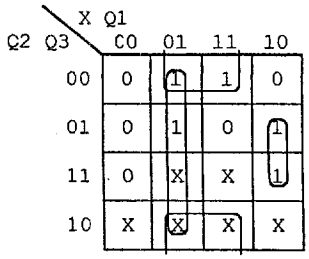
15.20 (c) (contd)



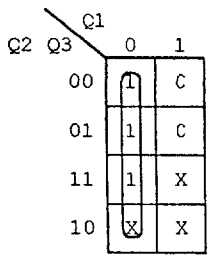
$Q1^+ = Q2 + X\,Q1'$



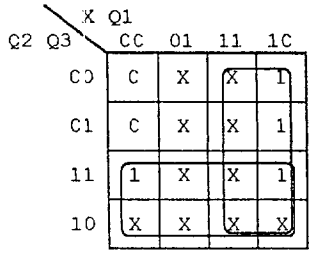
$Q2^+ = X\,Q1\,Q3'$



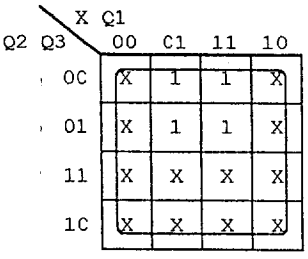
$Q3^+ = X'Q1 + X\,Q1'Q3 + Q1\,Q3'$



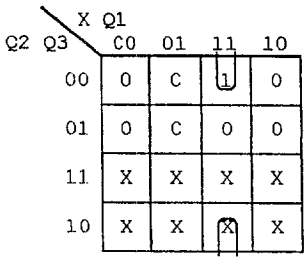
$Z = Q1'$



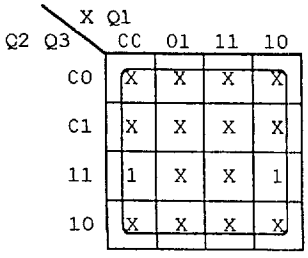
$J1 = Q2 + X$



$K1 = 1$



$J2 = X\,Q1\,Q3'$



$K2 = 1$

15.20 (d)

Q2 Q3 \ X Q1	00	01	11	10
00	C	1	1	C
01	X	X	X	X
11	X	X	X	X
10	X	X	X	X

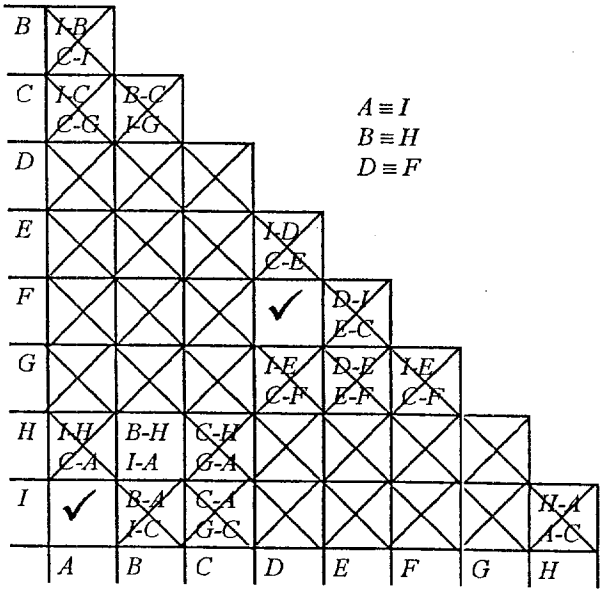
J3 = Q1

Q2 Q3 \ X Q1	00	01	11	10
00	X	X	X	X
01	1	0	1	0
11	1	X	X	0
10	X	X	X	X

K3 = X'Q1' + X Q1

Output Z equation is the same for D and J-K flip-flops.
(Actually, it is the same for any flip-flop.)

15.21 (a)



$A \equiv I$
$B \equiv H$
$D \equiv F$

Present State	Next State X = 0	Next State X = 1	Output
A	A	C	1
B	B	A	1
C	C	G	1
D	A	C	0
E	D	E	0
G	E	D	0

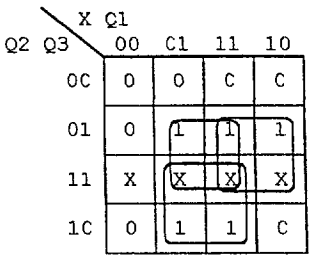
15.21 (b) 1. $(A, D)\times 2$
2. $(A, C)\times 2$ (A, B) (C, G) $(D, E)\times 2$
3. (A, B, C) (D, E, G)

There are several solutions. Here is one satisfying all guidelines:
$A = 000$, $B = 010$, $C = 001$, $D = 100$, $E = 110$, $G = 101$

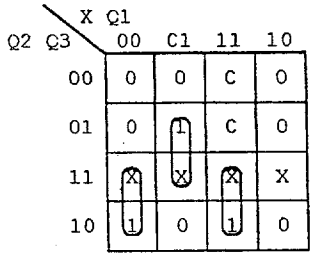
Q2 Q3 \ Q1	0	1
00	A	D
01	C	G
11		
10	B	E

15.21 (c)

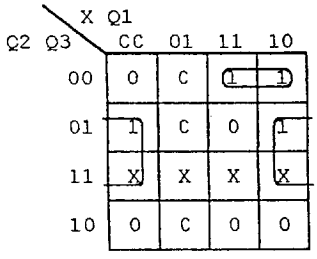
$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ $X=0$ 1	Z
000	000 001	1
010	010 000	1
001	001 101	1
100	000 001	0
110	100 110	0
101	110 100	0



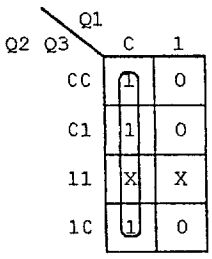
D1 = Q1 Q3 + Q1 Q2 + X Q3



D2 = X'Q1'Q2 + X'Q1 Q3 + X Q1 Q2

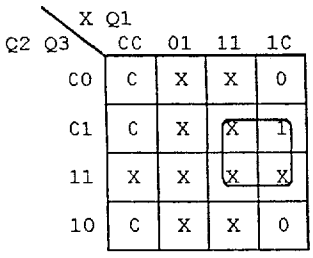


D3 = Q1'Q3 + X Q2'Q3'

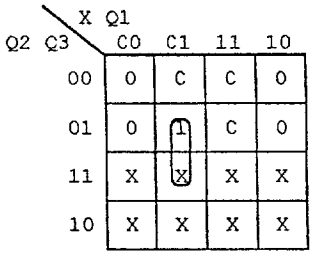


Z = Q1'

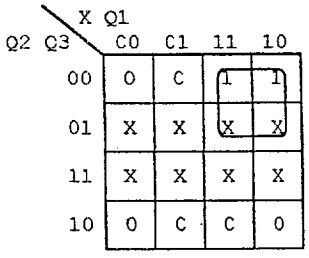
15.21 (d) Again, $Z = Q_1'$:



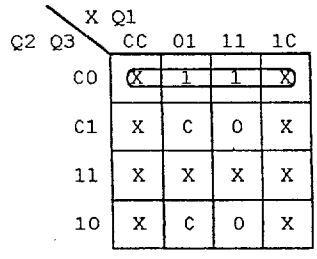
J1 = X Q3



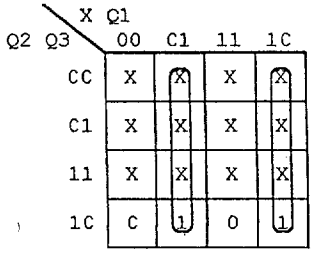
J2 = X'Q1 Q3



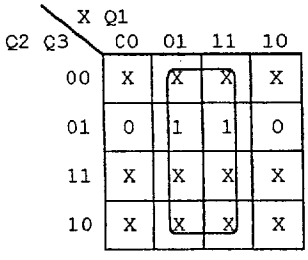
J3 = X Q2'



K1 = Q2'Q3'



K2 = X'Q1 + X Q1'



K3 = Q1

15.22

Present State	Next State X=0	Next State 1	Output X=0	Output X=1
S_0	S_1	S_4	0	0
S_1	S_1	S_2	0	0
S_2	S_3	S_4	1	0
S_3	S_5	S_2	0	0
S_4	S_3	S_4	0	0
S_5	S_1	S_2	0	1

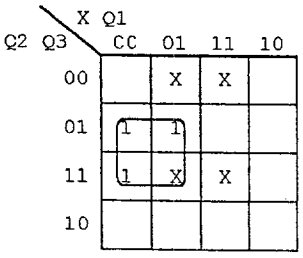
Q_2Q_3 \ Q_1	0	1
00	S_0	
01	S_4	S_3
11	S_2	
10	S_1	S_5

1. (S_0, S_1, S_5) (S_0, S_2, S_4) (S_1, S_3, S_5)
2. (S_1, S_4) $(S_1, S_2)\times 2$ $(S_3, S_4)\times 2$ (S_2, S_5)
3. (S_0, S_1, S_3, S_4)

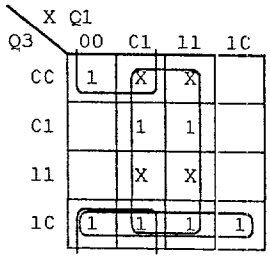
$S_0 = 000$. $S_1 = 010$, $S_2 = 011$, $S_3 = 101$, $S_4 = 001$, $S_5 = 110$

$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ X=0	$Q_1^+Q_2^+Q_3^+$ 1	Z X=0	Z X=1
000	010	001	0	0
010	010	011	0	0
011	101	001	1	0
101	110	011	0	0
001	101	001	0	0
110	010	011	0	1

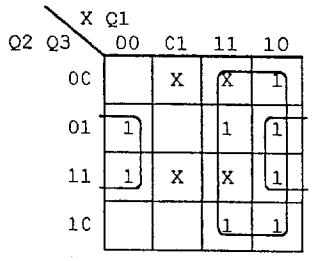
15.22 (a)



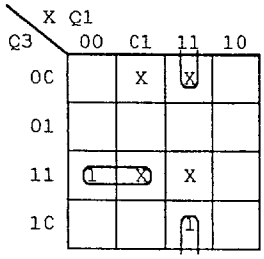
$Q1^+ = X'Q3$



$Q2^+ = X'Q3' + Q1 + Q2\ Q3'$

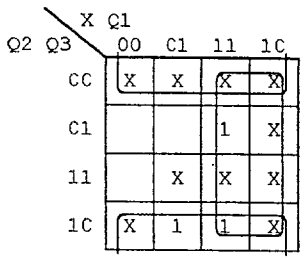


$Q3^+ = Q1'Q3 + X$

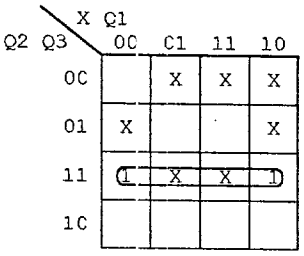


$Z = X'Q2\ Q3 + X\ Q1\ Q3'$

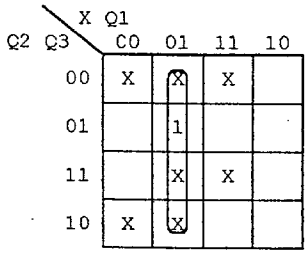
15.22 (b)



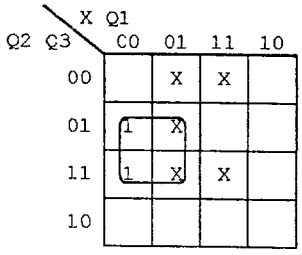
$R1 = X + Q3'$



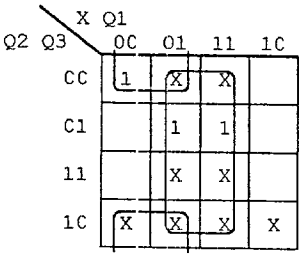
$R2 = Q2\ Q3$



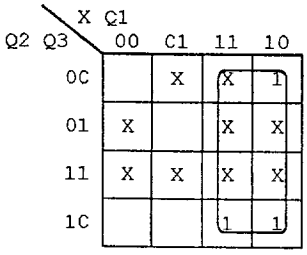
$R3 = X'Q1$



$S1 = X'Q3$



$S2 = X'Q3' + Q1$



$S3 = X$

One alternative assignment:

Q_1 \ Q_2Q_3	00	01	11	10
C	0	1		2
1	3	5		4

(a) $D_1 = XQ_1'Q_2' + Q_2 + X'Q_1Q_3'$; $D_2 = X$; $D_3 = X'Q_2'$
$Z = X'Q_1'Q_2 + XQ_1Q_3$

(b) $S_1 = XQ_1'Q_3' + Q_2$; $R_1 = XQ_1Q_2' + Q_3$; $S_2 = X$;
$R_2 = X'$; $S_3 = X'Q_2'$; $R_3 = X$; $Z = X'Q_1'Q_2 + XQ_1Q_3$

15.23

Present State	Next State $X=0$	1	Z
S_0	S_2	S_1	0
S_1	S_5	S_0	0
S_2	S_3	S_1	0
S_3	S_3	S_4	0
S_4	S_4	S_3	1
S_5	S_4	S_0	0

$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ $X=0$	1	Z
000	010	001	0
001	011	000	0
010	110	001	0
110	110	111	0
111	111	110	1
011	111	000	0

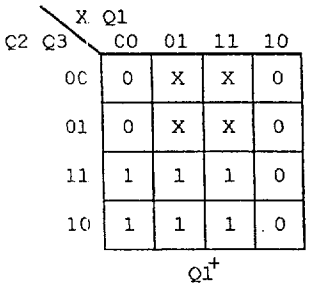
Q_3 \ Q_1Q_2	00	01	11	10
0	0	2	3	
1	1	5	4	

1. (S_2, S_3) (S_4, S_5) (S_0, S_2) (S_1, S_5)
2. (S_1, S_2) (S_0, S_5) (S_1, S_3) $(S_3, S_4)\times 2$ (S_0, S_4)

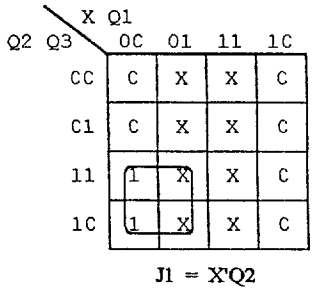
$S_0 = 000$, $S_1 = 001$, $S_2 = 010$, $S_3 = 110$, $S_4 = 111$, $S_5 = 011$

Guideline 3 is of no use for this state table.

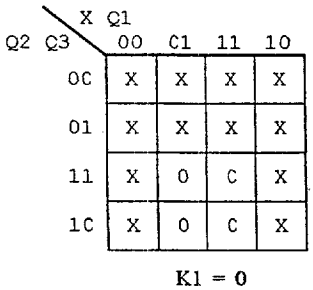
15.23 (a)



$Q1^+$

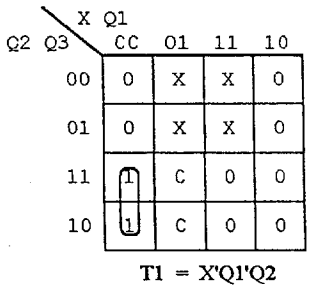


J1 = X'Q2

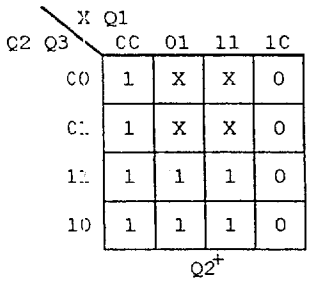


K1 = 0

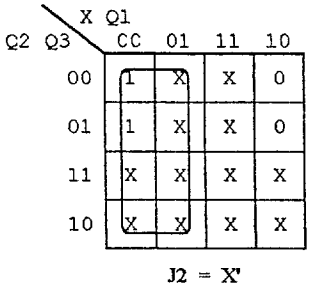
15.23 (b)



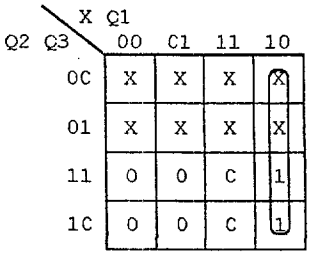
T1 = X'Q1'Q2



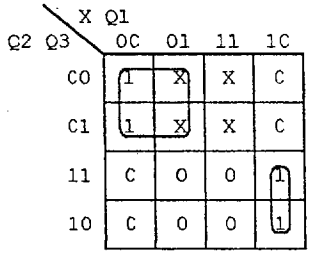
$Q2^+$



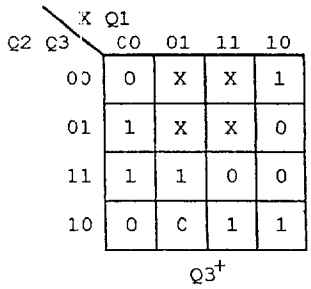
J2 = X'



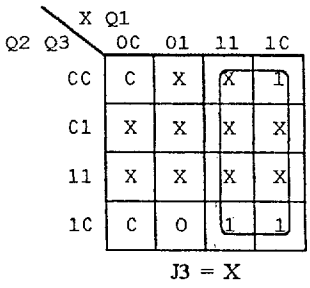
K2 = X Q1'



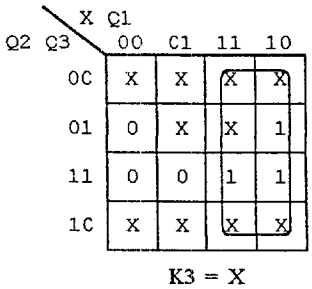
T2 = X'Q2' + X Q1'Q2



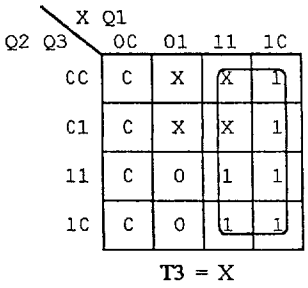
$Q3^+$



J3 = X



K3 = X



T3 = X

15.24 *See solutions to 14.13 for the state table.*

I. $(S_0, S_1, S_7)\ (S_2, S_6)\ (S_3, S_4, S_5)\ (S_0, S_6)\ (S_1, S_7)\ (S_2, S_3, S_5)$

II. $(S_1, S_6)\ (S_6, S_7)\ (S_1, S_2)\times 2\ (S_3, S_7)\ (S_3, S_4)\times 2\ (S_4, S_5)$

III. $(S_0, S_1, S_3, S_4, S_6)\ (S_2, S_5)$

$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ $X=0$	$X=1$	Z $X=0$	$X=1$
000	001	100	00	00
001	001	110	00	00
110	010	011	00	01
011	111	011	00	00
111	111	101	00	00
101	111	011	00	01
100	010	100	00	00
010	001	110	10	00

Q_2Q_3 \ Q_1	C	1
CC	S_0	S_6
C1	S_1	S_5
11	S_3	S_4
1C	S_7	S_2

Q_2Q_3 \ X Q_1	C0	01	11	10
00	0	C	1	1
01	0	1	0	1
11	1	1	1	0
10	0	C	0	1

$Q_1^+ = X'Q_2Q_3 + X'Q_1Q_3 + XQ_1'Q_2' + XQ_1'Q_3' + XQ_2'Q_3' + Q_1Q_2Q_3$

Q_2Q_3 \ X Q_1	0C	01	11	1C
CC	C	1	0	C
C1	C	1	1	1
11	1	1	0	1
1C	C	1	1	1

$Q_2^+ = X'Q_1 + Q_1'Q_2Q_3 + XQ_2'Q_3 + XQ_2Q_3'$

Q_2Q_3 \ X Q_1	CC	01	11	10
C0	1	C	0	0
C1	1	1	1	0
11	1	1	1	1
10	1	C	1	0

$Q_3^+ = X'Q_1' + Q_2Q_3 + Q_1Q_3 + XQ_1Q_2$

Q_2Q_3 \ X Q_1	0C	01	11	1C
CC	C	0	0	C
C1	C	0	0	C
11	C	0	0	C
1C	1	0	0	C

$Z_1 = X'Q_1'Q_2Q_3'$

Q_2Q_3 \ X Q_1	00	C1	11	10
0C	0	0	C	0
01	0	0	1	0
11	0	0	C	0
1C	0	0	1	0

$Z_2 = X\,Q_1Q_2'Q_3 + X\,Q_1Q_2Q_3'$

15.25 *See FLD p. 656 for the state table.*

I. $(S_0, S_1)\ (S_2, S_3)\ (S_4, S_5, S_7)\ (S_0, S_2, S_3)\ (S_1, S_4')\ (S_5, S_6, S_7)$

II. $(S_1, S_3)\ (S_1, S_2)\ (S_3, S_4)\times 2\ (S_2, S_5)\ (S_5, S_6)\times 2\ (S_6, S_7)$

III. $(S_0, S_1, S_3, S_5, S_6)\ (S_4, S_7)$

$S_0 = 000, S_1 = 001, S_2 = 010, S_3 = 011, S_4 = 111, S_5 = 110, S_6 = 100, S_7 = 101$

$Q_1Q_2Q_3$	$Q_1^+Q_2^+Q_3^+$ $X=0$	$X=1$	Z $X=0$	$X=1$
000	001	011	00	00
001	001	010	00	00
010	111	011	10	00
011	111	011	00	00
111	110	010	01	00
110	110	100	00	00
100	101	100	00	00
101	110	100	01	00

15.25
(contd)

Q_2Q_3 \ X Q_1	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	1	0	0
10	1	1	1	0

Q_1^+

Q_2Q_3 \ X Q_1	00	01	11	10
00	0	X	X	0
01	0	X	X	0
11	1	X	X	0
10	1	X	X	0

$J_1 = X'Q_2$

Q_2Q_3 \ X Q_1	00	01	11	10
00	X	0	0	X
01	X	0	0	X
11	X	0	1	X
10	X	0	0	X

$K_1 = X Q_2 Q_3$

Q_2Q_3 \ X Q_1	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	1	0	0	0

$Z_1 = X'Q_1'Q_2Q_3'$

Q_2Q_3 \ X Q_1	00	01	11	10
00	0	0	0	1
01	0	1	0	1
11	1	1	1	1
10	1	1	0	1

Q_2^+

Q_2Q_3 \ X Q_1	00	01	11	10
00	0	0	0	1
01	0	1	0	1
11	X	X	X	X
10	X	X	X	X

$J_2 = X'Q_1Q_3 + XQ_1'Q_2'$

Q_2Q_3 \ X Q_1	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	0	0
10	0	0	1	0

$K_2 = X Q_1 Q_3'$

Q_2Q_3 \ X Q_1	00	01	11	10
00	0	0	0	0
01	0	1	0	0
11	0	1	0	0
10	0	0	0	0

$Z_2 = X'Q_1Q_3$

Q_2Q_3 \ X Q_1	00	01	11	10
00	1	1	0	1
01	1	0	0	0
11	1	0	0	1
10	1	0	0	1

Q_3^+

Q_2Q_3 \ X Q_1	00	01	11	10
00	1	1	0	1
01	X	X	X	X
11	X	X	X	X
10	1	0	0	1

$J_3 = Q_1' + X'Q_2'$

Q_2Q_3 \ X Q_1	00	01	11	10
00	X	X	X	X
01	0	1	1	1
11	0	1	1	0
10	X	X	X	X

$K_3 = Q_1 + X Q_2'$

15.26 Row reduction of the solution to 14.6 given on FLD p. 657 easily gives 4 states. Renaming them gives:

Present State	Next State 00 01 11 10	Z
S_0	S_0 S_1 S_1 S_0	0
S_1	S_0 S_1 S_3 S_3	0
S_2	S_2 S_3 S_2 S_0	1
S_3	S_2 S_3 S_2 S_3	1

See p. 100 in this manual for the state table.

I. $(S_0, S_1)\times 3\ (S_2, S_3)\times 2\ (S_0, S_2)\ (S_1, S_3)$
II. $(S_0, S_1)\ (S_0, S_1, S_3)\ (S_0, S_2, S_3)\ (S_2, S_3)$
III. $(S_0, S_1)\ (S_2, S_3)$
$S_0 = 00, S_1 = 01, S_2 = 10, S_3 = 11$

Q_1Q_2	$Q_1^+Q_2^+$ 00 01 11 10	Z
00	00 01 01 00	0
01	00 01 01 11	0
10	10 11 10 00	1
11	10 11 10 11	1

Q_2 \ Q_1	0	1
0	S_0	S_2
1	S_1	S_3

Q_1Q_2 \ X_1X_2	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	1	1	1
10	1	1	0	1

$D_1 = X_1X_2Q_2 + X_1'Q_1 + X_2'Q_1$

Q_1Q_2 \ X_1X_2	00	01	11	10
00	0	1	0	1
01	0	1	1	1
11	0	1	1	0
10	0	1	0	0

$D_2 = X_1'X_2 + X_1X_2'Q_1' + X_2Q_2$

Q_2 \ Q_1	0	1
0	0	1
1	0	1

$Z = Q_1$

15.27 *See answers to 14.14 for the state table. The four-state table is minimum.*

I. $(S_0, S_1)\times 3\ (S_0, S_3)\ (S_1, S_2)\ (S_2, S_3)\times 3$

II. $(S_0, S_1)\ (S_0, S_1, S_2)\ (S_2, S_3)\ (S_0, S_2, S_3)$

III. $(S_0, S_1)\ (S_2, S_3)$

Q_1Q_2	$Q_1^+Q_2^+$ 00	01	11	10	Z
00	01	01	00	00	0
01	01	01	11	00	0
11	11	10	11	10	1
10	11	10	00	10	1

Q_2 \ Q_1	0	1
0	S_0	S_3
1	S_1	S_2

Q_1Q_2 \ X_1X_2	00	01	11	10
00	0	C	0	0
01	0	C	0	1
11	1	1	1	1
10	1	1	1	0

$D_1 = X_1'Q_1 + X_1X_2'Q_2 + X_2Q_1$

Q_1Q_2 \ X_1X_2	00	01	11	10
00	1	1	0	C
01	1	1	0	C
11	1	0	0	1
10	1	0	0	C

$D_2 = X_1'X_2' + X_1'Q_1' + X_2'Q_1Q_2$

Q_2 \ Q_1	0	1
0	0	1
1	0	1

$Z = Q_1$

15.28

B C \ W A	00	01	11	10
00	C	0	0	C
01	C	1	1	1
11	1	0	1	C
10	1	1	0	1

T_A

B C \ W A	00	01	11	10
00	0	1	1	1
01	1	0	C	C
11	1	0	C	1
10	1	0	1	C

T_B

B C \ W A	00	01	11	10
00	1	1	C	1
01	0	1	C	0
11	0	1	1	1
10	0	0	C	1

T_C

B C \ W A	00	01	11	10
00	C	1	1	C
01	C	0	0	1
11	1	1	0	C
10	1	0	1	1

A^+

B C \ W A	00	01	11	10
00	C	1	1	1
01	1	0	0	C
11	C	1	1	C
10	C	1	0	1

B^+

B C \ W A	00	01	11	10
00	1	1	C	1
01	1	0	1	1
11	1	0	C	0
10	0	0	C	1

C^+

ABC	$A^+B^+C^+$ $W=0$	$W=1$	Z 0	1
000	001	011	0	0
001	011	101	0	0
010	100	111	1	0
011	101	000	0	0
100	111	110	0	0
101	000	001	0	0
110	010	100	1	0
111	110	010	0	0

Present State	Next State $W=0$	$W=1$	Z 0	1
0	1	3	0	0
1	3	5	0	0
2	4	7	1	0
3	5	0	0	0
4	7	6	0	0
5	0	1	0	0
6	2	4	1	0
7	6	2	0	0

$0 \equiv 1 \equiv 3 \equiv 5$

15.28 (contd) I. None

II. $(4, 7)\checkmark$ $(6, 7)\checkmark$ $(2, 4)\checkmark$ $(2, 6)\checkmark$

Assignment:

$S_0 = 000, S_2 = 100, S_4 = 111, S_6 = 110, S_7 = 101$

B C \ A	0	1
0C	S_0	S_2
01		S_7
11		S_4
1C		S_6

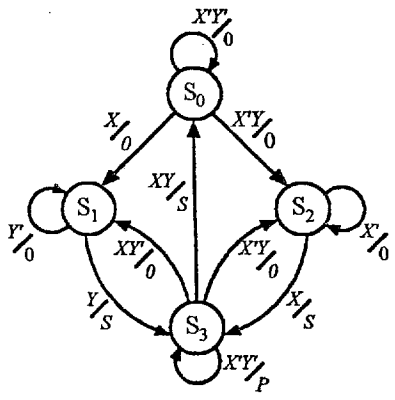
Present State	Next State $W=0$	Next State 1	Output 0	Output 1
S_0	S_0	S_0	0	0
S_2	S_4	S_7	1	0
S_4	S_7	S_6	0	0
S_6	S_2	S_4	0	0
S_7	S_6	S_2	0	0

Present State	Next State $W=0$	Next State 1	Output 0	Output 1
000	000	000	0	0
100	111	101	1	0
111	101	110	0	0
110	100	111	0	0
101	110	100	0	0

T input equations derived from the transition table using Karnaugh maps:

$T_A = 0; \quad T_B = W'A; \quad T_C = WB + AB'; \quad Z = W'AB'C'$

15.29



By inspecting incoming arrows, we get:

$D_0 = Q_0^+ = X'Y'Q_0 + XYQ_3$

$D_1 = Q_1^+ = XQ_0 + Y'Q_1 + XY'Q_3$

$D_2 = Q_2^+ = X'YQ_0 + X'Q_2 + X'YQ_3$

$D_3 = Q_3^+ = YQ_1 + XQ_2 + X'Y'Q_3$

$S = YQ_1 + XQ_2$

$P = X'Y'Q_3$

15.30 By inspecting incoming arrows, we get:

$Q_0^+ = D_0 = X'YQ_0 + Y'Q_1 + X'YQ_2$

$Q_1^+ = D_1 = XY'Q_0 + XYQ_1 + Y'Q_2$

$Q_2^+ = D_2 = XYQ_0 + X'Y'Q_0 + X'YQ_1 + XYQ_2$

$Z = X'YQ_1 + XYQ_2 + X'YQ_2 = X'YQ_1 + YQ_2$

Unit 16 Problem Solutions

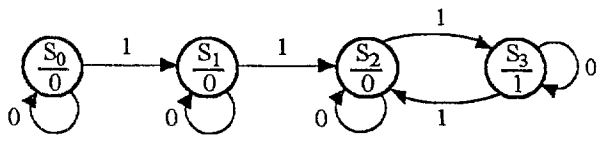
16.1–16.14 *See Lab Solutions in this manual.*

16.15 *See FLD p. 662 for solution.*

16.16 *See FLD p. 662 for solution.*

16.17 (a) The state meanings are given in the following table:

Name	Meaning
S_0	No 1's have occurred
S_1	One 1 has occurred (an odd number < 2)
S_2	Two 1's or an even number of 1's > 2 have occurred
S_3	An odd number of 1's > 2 has occurred.



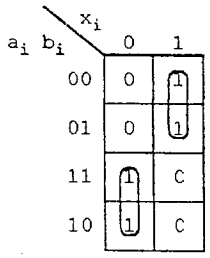
16.17 (b)

State	Next State $X=0$	Next State $X=1$	Z
S_0	S_0	S_1	0
S_1	S_1	S_2	0
S_2	S_2	S_3	0
S_3	S_3	S_2	1

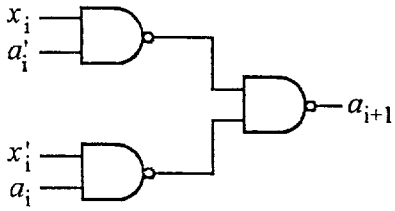
I: (1, 3)
II: $(0, 1)(1, 2)(2, 3)_Z$

a_i / b_i	0	1
0	S_0	S_1
1	S_2	S_3

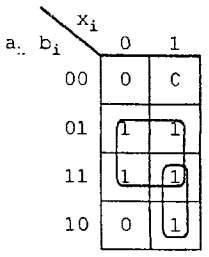
State	$a_i b_i$	$a_{i+1}b_{i+1}$ $X=0$	$X=1$	Z
S_0	00	00	10	0
S_1	10	10	01	0
S_2	01	01	11	0
S_3	11	11	01	1



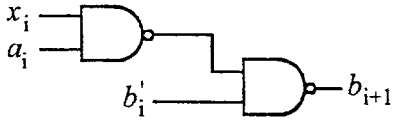
$a_{i+1} = x_i'a_i + x_i a_i'$



$a_{i+1} = x_i a_i' + x_i' a_i$



$b_{i+1} = b_i + x_i a_i$



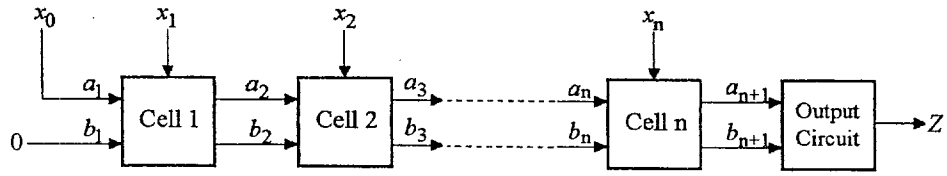
$b_{i+1} = b_i' + x_i a_i$

$z = a_{n+1}b_{n+1}$

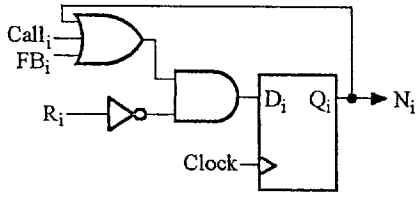
Note: Solution on FLD p. 622 uses state assignment $S_0 = 00$, $S_1 = 01$, $S_2 = 10$, $S_3 = 11$.

16.17 (c) Since no 1's have occurred, a_0 and b_0 are the same as S_0 or, $a_0 = 0$; $b_0 = 0$;
$a_i = x_0a_0' + x_0'a_0 = x_0$;
$b_i = b_0 + x_0a_0 = 0$ } first cell

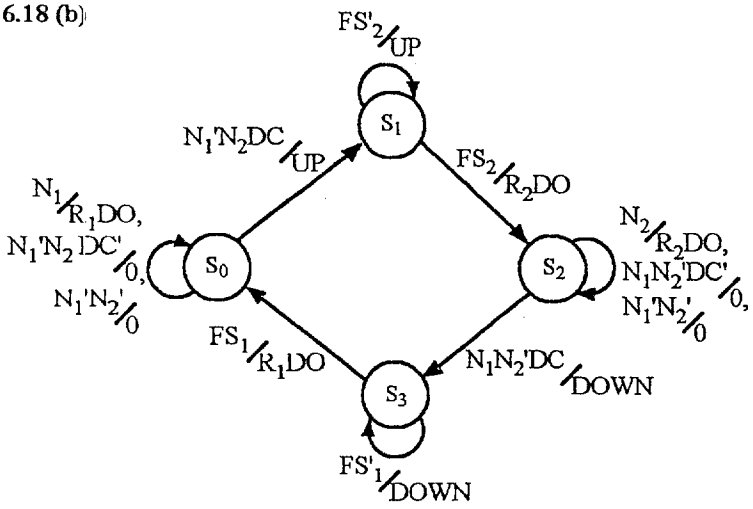
16.17 (d)



16.18 (a) $N_i = Q_i^+ = (Q_i + FB_i + CALL_i)R_i' = Q_iR_i' + FB_iR_i' + CALL_iR_i'$



16.18 (b)



Name	Meaning
S_0	Staying on first floor
S_1	Moving from first to second floor
S_2	Staying on second floor
S_3	Moving from second to first floor

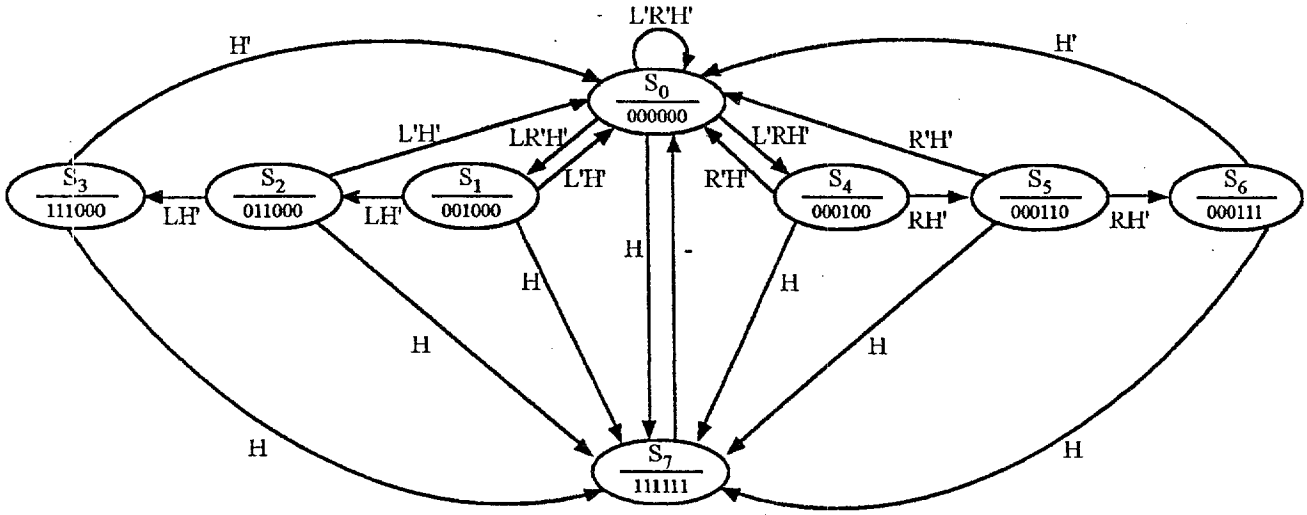
16.18 (c) With the state assignment $S_0 = 00$, $S_1 = 01$, $S_2 = 10$, $S_3 = 11$, we have:

$D_1 = FS_2Q_1'Q_2 + FS_1'Q_1 + Q_1Q_2'$; $D_2 = FS_2'Q_1'Q_2 + FS_1'Q_1Q_2 + N_1'N_2DCQ_1'Q_2' + N_1N_2'DCQ_1Q_2'$

$R_1 = FS_1Q_1Q_2 + N_1Q_1'Q_2'$; $R_2 = FS_2Q_1'Q_2 + N_2Q_1Q_2'$; $UP = FS_2'Q_1'Q_2 + N_1'N_2DCQ_1'Q_2'$

$DOWN = FS_1'Q_1Q_2 + N_1N_2'DCQ_1Q_2'$; $DO = FS_2Q_1'Q_2 + FS_1Q_1Q_2 + N_1Q_1'Q_2' + N_2Q_1Q_2'$

16.19 (a)



Outputs: *LC, LB, LA, RA, RB, RC*

16.19 (b) First, assign $LC = Q_1$, $LB = Q_2$, $LA = Q_3$, $RA = Q_4$, $RB = Q_5$, $RC = Q_6$. So $S_0 = 000000$, $S_1 = 001000$, $S_2 = 011000$, etc.

This state machine has too many state variables to use Karnaugh maps. Instead, we will write down equations for each flip-flop by inspection.

First consider Q_1. $Q_1 = 1$ in states S_3 or S_7 only.

- S_7 is reached whenever $H = 1$ and we are not already in S_7: $H(Q_1Q_2Q_3Q_4Q_5Q_6)'$. But S_7 is the only state in which both $Q_3 = 1$ and $Q_4 = 1$, so assuming we are always in a valid state, we can use $H(Q_3Q_4)' = HQ_3' + HQ_4'$. *Note*: Any combination of one left light and one right light will also work, i.e. $HQ_1' + HQ_5'$.

- S_3 is reached whenever we are in S_2 and $L = 1$ while $H = 0$: $LH'Q_1'Q_2Q_3Q_4'Q_5'Q_6'$. But $Q_3 = 1$ whenever $Q_2 = 1$, and $Q_4 = Q_5 = Q_6 = 0$ whenever $Q_1 = 0$. So we can use $LH'Q_1'Q_2$.

16.19 (b) (contd)

- So $D_1 = LH'Q_1'Q_2 + HQ_3' + HQ_4' = LQ_1'Q_2 + HQ_3' + HQ_4'$ (using $X + X'Y = X + Y$)
  Similarly $Q_2 = 1$ in states S_3, S_2, and S_7 only.

- S_3 and S_2 are reached whenever we are in S_2 or S_1 and $L = 1$ while $H = 0$.
  $LH'Q_1'Q_2Q_3Q_4'Q_5'Q_6' + LH'Q_1'Q_2'Q_3Q_4'Q_5'Q_6' = LH'Q_1'Q_3Q_4'Q_5'Q_6'$
  But again, $Q_4 = Q_5 = Q_6 = 0$ whenever $Q_1 = 0$, so $D_2 = LQ_1'Q_3 + HQ_3' + HQ_4'$
  We can also get by inspection: $D_3 = LQ_1'Q_4' + HQ_3' + HQ_4'$; $D_4 = RQ_3'Q_6' + HQ_3' + HQ_4'$;
  $D_5 = RQ_4Q_6' + HQ_3' + HQ_4'$; $D_6 = RQ_5'Q_6' + HQ_3' + HQ_4'$

16.19 (c)

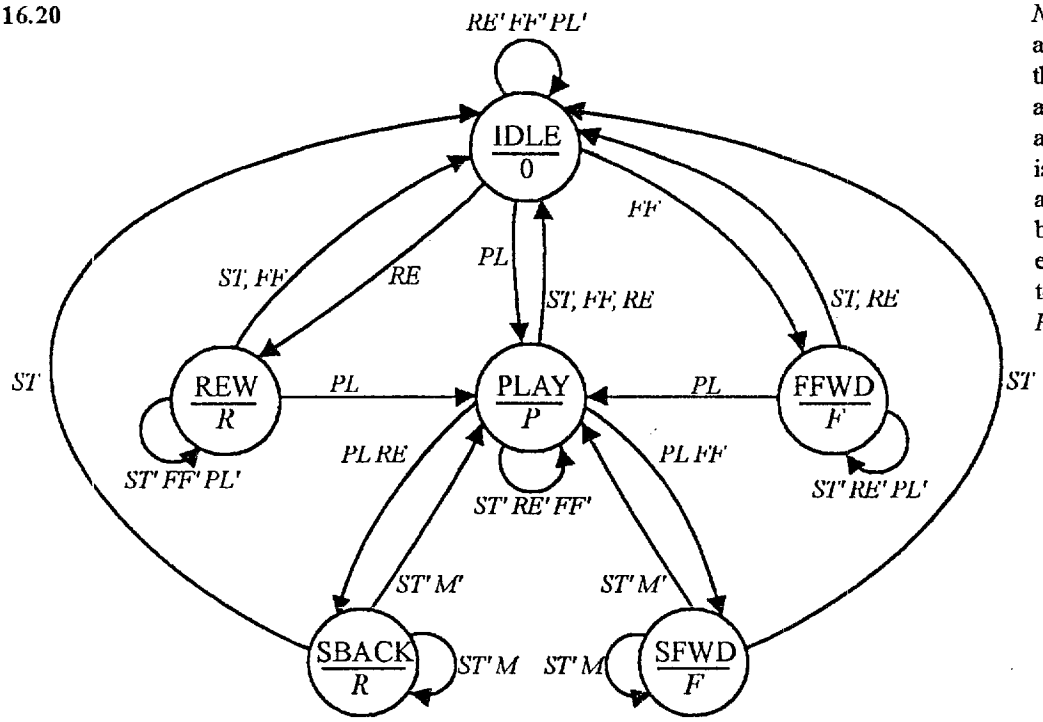
State	LRH = 000	001	010	011	100	101	110	111	LC	LB	LA	RA	RB	RC
S_0	S_0	S_7	S_4	S_7	S_1	S_7	-	-	0	0	0	0	0	0
S_1	S_0	S_7	S_0	S_7	S_2	S_7	-	-	0	0	1	0	0	0
S_2	S_0	S_7	S_0	S_7	S_3	S_7	-	-	0	1	1	0	0	0
S_3	S_0	S_7	S_0	S_7	S_0	S_7	-	-	1	1	1	0	0	0
S_4	S_0	S_7	S_5	S_7	S_0	S_7	-	-	0	0	0	1	0	0
S_5	S_0	S_7	S_6	S_7	S_0	S_7	-	-	0	0	0	1	1	0
S_6	S_0	S_7	S_0	S_7	S_0	S_7	-	-	0	0	0	1	1	1
S_7	S_0	S_0	S_0	S_0	S_0	S_0	-	-	1	1	1	1	1	1

I. $(S_0, S_1, S_2, S_3, S_4, S_5, S_6)$ for S_7 in LRH = 001, 011, 101
$(S_1, S_2, S_3, S_6, S_7)$ for S_0 in LRH = 010
$(S_3, S_4, S_5, S_6, S_7)$ for S_0 in LRH = 100
II. Every state matches S_0 and S_7. But S_0 and S_7 match the best, so (S_0, S_7)×(many times)
III. (S_1, S_2, S_3, S_7) (S_4, S_5, S_6, S_7) etc.

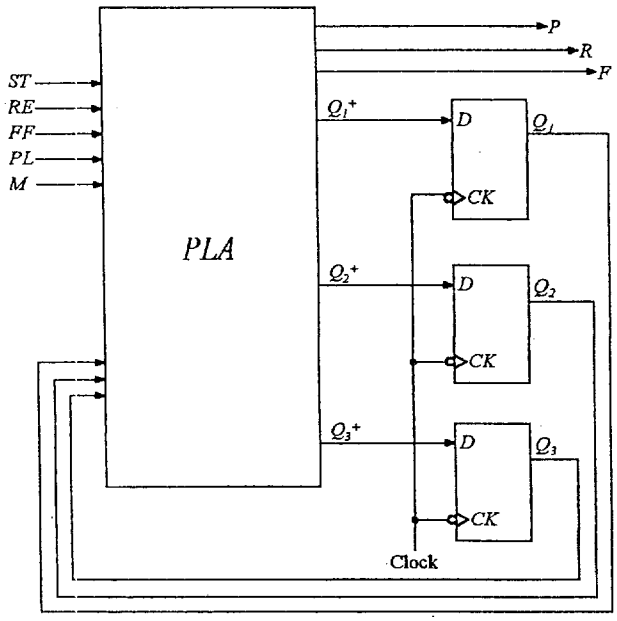
Q_2 Q_3 \ Q_1	0	1
00	S_0	S_7
01	S_2	S_3
11	S_6	S_5
10	S_1	S_4

From LogicAid:
So $D_1 = HQ_2 + RQ_1Q_2Q_3' + HQ_3 + LQ_1'Q_2'Q_3 + HQ_1' + RQ_1'Q_2'Q_3'$
$D_2 = RH'Q_1'Q_2'Q_3' + RH'Q_1Q_2 + LH'Q_1'Q_2'Q_3'$
$D_3 = LH'Q_1'Q_2Q_3' + LH'Q_1'Q_2'Q_3 + RH'Q_1Q_2$
$LC = Q_1Q_2'$; $LB = Q_1Q_2' + Q_2'Q_3$; $LA = Q_1Q_2' + Q_2'Q_3 + Q_1'Q_2Q_3'$
$RC = Q_1Q_2'Q_3' + Q_1'Q_2Q_3$; $RB = Q_1Q_2'Q_3' + Q_2Q_3$; $RA = Q_1Q_3' + Q_2Q_3$
Other minimum solutions can be found for D_2 and D_3 with this assignment.

16.20

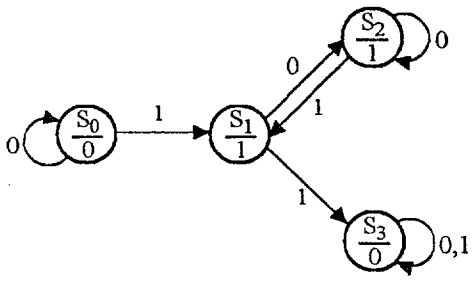


Note: This state graph assumes that only one of the buttons *ST*, *PL*, *RE*, and *FF* can be pressed at any given time. The graph is incompletely specified and must be augmented before using LogicAid. For example, the arc from *REW* to *PLAY* should be labeled *PL ST' FF'*.



$$D_1 = ST'\,FF\,PS\,Q_1'Q_2'Q_3 + ST'\,RE\,PL\,Q_1'Q_2'Q_3 + ST'\,M\,Q_1$$
$$D_2 = ST'\,FF\,Q_1'Q_2'Q_3' + ST'\,RE\,Q_1'Q_2'Q_3' + ST'\,RE'\,PL'\,Q_2\,Q_3 + ST'\,FF'\,PL'\,Q_2\,Q_3'$$
$$D_3 = ST'\,RE'\,FF\,Q_1'Q_2'Q_3' + ST'\,RE'\,FF'\,Q_3 + ST'\,FF'\,PL\,Q_2\,Q_3' + ST'\,RE'\,Q_2\,Q_3 + ST'\,M'\,Q_1 + ST'\,Q_1\,Q_3 + ST'\,RE'\,PL\,Q_1'Q_2'$$
$$P = Q_1'Q_2'Q_3; \quad R = Q_2Q_3' + Q_1Q_3'; \quad F = Q_2Q_3 + Q_1Q_3$$

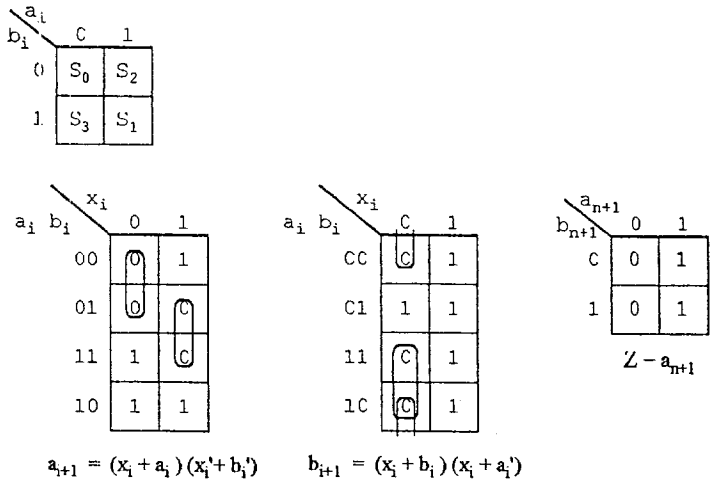
16.21 (a)



16.21 (b)

State	Next State $x_i = 0$	Next State $x_i = 1$	Z
S_0	S_0	S_1	0
S_1	S_2	S_3	1
S_2	S_2	S_1	1
S_3	S_3	S_3	0

$a_i b_i$	$a_{i+1}b_{i+1}$ $x_i = 0$	$a_{i+1}b_{i+1}$ $x_i = 1$	Z
00	00	11	0
11	10	01	1
10	10	11	1
01	01	01	0



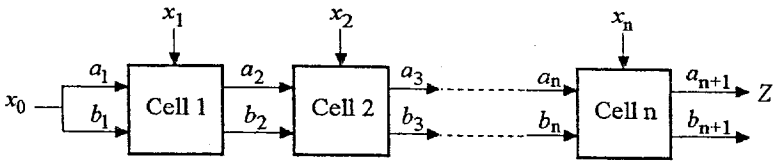
$a_{i+1} = (x_i + a_i)(x_i' + b_i')$ $\quad b_{i+1} = (x_i + b_i)(x_i + a_i')$

16.21 (c) $a_0 = b_0 = 0$

$a_1 = (x_0 + 0)(x_0' + 1) = x_0$

$b_1 = (x_0 + 1)(x_0 + 0) = x_0$

16.21 (d)

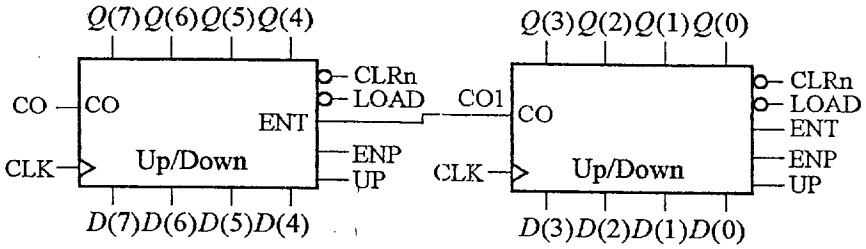


Unit 17 Problem Solutions

17.1 *See FLD p. 664 for solution.*

17.2 *See FLD p. 665 for solution.*

17.3 (a, b)

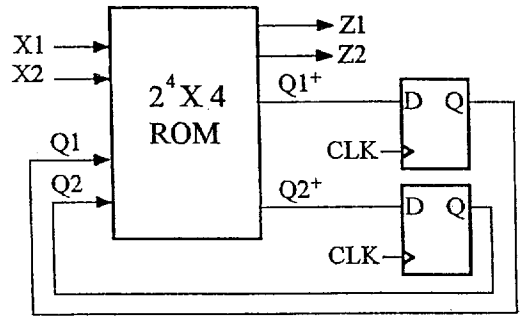


See FLD p. 665-666 for solutions.

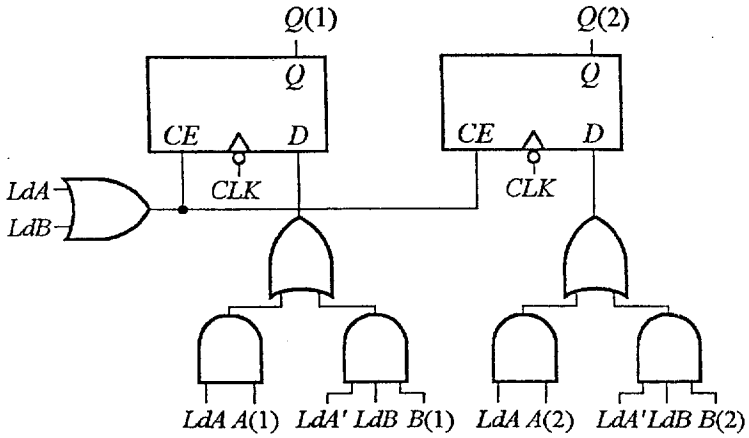
17.4 *See FLD p. 666-667 for solution.*

17.5 *See FLD p. 667 for solution.*

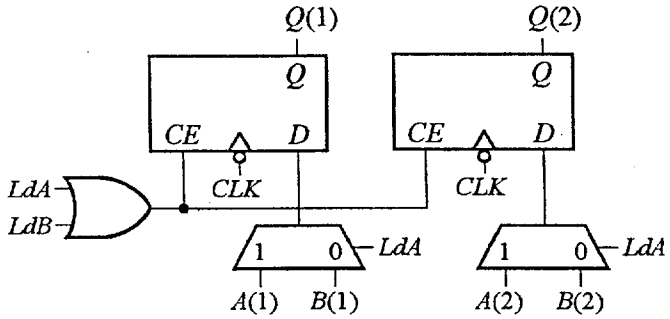
17.6 (a, b) *See FLD p. 667-668 for solutions.*



17.7 (a) *See FLD p. 668 for solution.*



17.7 (b)



17.8 *See FLD p. 668 for solution.*

17.9

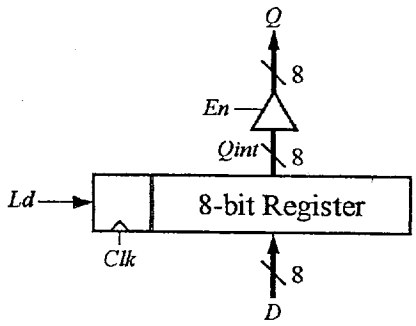
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity srff is
    port (clk, s, r : in std_logic;
        q, qn : out std_logic);
end srff;
architecture Behavioral of srff is
signal qint : std_logic:='0';
    begin
    q <= qint;
    qn <= not qint;
    process(clk)
        begin
        if clk'event and clk='1' then
            if (not s and r)='1' then qint <= '0';
            elsif (s and not r)='1' then qint<='1';
            elsif (s and r)='1' then qint<='X'; end if;
        end if;
    end process;
end Behavioral;
```

17.10

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
-- D-G Latch
entity dglatch is
    port (d, g : in bit;
        q : out bit);
end dglatch;
architecture Behavioral of dglatch is
    begin
    process(g, d)
        begin
        if g='1' then q <= d; end if;
    end process;
end Behavioral;
-- D flip flop using D-G latches
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity dff is
    port (d, clk : in bit;
        q : out bit);
end dff;
architecture Behavioral of dff is
component dglatch is
    port (d, g: in bit;
        q : out bit);
end component;
signal p, clkn : bit;
    begin
    clkn <= not clk;
    dg1 : dglatch port map(d, clkn, p);
    dg2 : dglatch port map(p, clk, q);
end Behavioral;
```

17.11 A rising edge triggered D-CE flip flop with asynchronous clear and preset.

17.12




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity myreg is
    port(en, ld, clk : in std_logic;
        d : in std_logic_vector(7 downto 0);
        q : out std_logic_vector(7 downto 0));
end myreg;
architecture Behavioral of myreg is
signal qint : std_logic_vector(7 downto 0):="00000000";
    begin
    q <= qint when en ='1' else "ZZZZZZZZ";
    process(clk)
        begin
        if clk' event and clk='1' then
            if ld='1' then qint <= d; end if;
        end if;
    end process;
end Behavioral;
```

17.13

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity encoder is
    port (y0, y1, y2, y3 : in bit;
        a, b, c : out bit);
end encoder;
architecture Behavioral of encoder is
    begin
    process(y0, y1, y2, y3)
        begin
        if y3='1' then a <= '1'; b <= '1'; c <= '1';
            -- y3 has highest priority
        elsif y2='1' then
            a <= '1'; b <= '0'; c <= '1';
        elsif y1='1' then
            a <= '0'; b <= '1'; c <= '1';
        elsif y0='1' then
            a <= '0'; b <= '0'; c <= '1';
        else a <= '0'; b <= '0'; c <= '0'; end if;
    end process;
end Behavioral;
```

17.14

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity comparator is
    port (a, b : in std_logic_vector(3 downto 0);
        agb, alb, aeb : out std_logic);
end comparator;
architecture Behavioral of comparator is
    begin
    process(a, b)
        begin
        if a > b then agb <= '1'; alb <= '0'; aeb <= '0';
        elsif a < b then agb <= '0'; alb <= '1'; aeb <= '0';
        else agb <= '0'; alb <= '0'; aeb <= '1'; end if;
    end process;
end Behavioral;
```

17.15

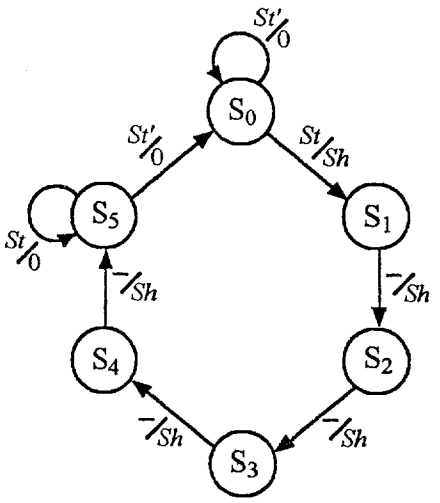
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity super is
    port (a: in std_logic_vector(2 downto 0);
        d : in std_logic_vector(5 downto 0);
        rsi, lsi, clk : in std_logic;
        q : out std_logic_vector(5 downto 0));
end super;
architecture Behavioral of super is
signal qint: std_logic_vector(5 downto 0);
    begin
    q <= qint;
    process(clk)
        begin
        if clk' event and clk='1' then
            case a is
            when "111"=> qint <= d;
            when "110"=> qint <= qint-1;
            when "101"=> qint <= qint+1;
            when "100"=> qint <= "111111";
            when "011"=> qint <= "000000";
            when "010"=> qint <= rsi&qint(5 downto 1);
            when "001"=> qint <= qint(4 downto 0)&lsi;
            when others=> NULL;
            end case;
        end if;
    end process;
end Behavioral;
```

17.16

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity bcd_seven is
    port (bcd : in bit_vector(3 downto 0);
        seven : out bit_vector(7 downto 1));
end bcd_seven;
architecture Behavioral of bcd_seven is
    begin
    process(bcd)
        begin
        case bcd is
        when "0000"=> seven <= "0111111";
        when "0001"=> seven <= "0000110";
        when "0010"=> seven <= "1011011";
        when "0011"=> seven <= "1001111";
        when "0100"=> seven <= "1100110";
        when "0101"=> seven <= "1101101";
        when "0110"=> seven <= "1111101";
        when "0111"=> seven <= "0000111";
        when "1000"=> seven <= "1111111";
        when "1001"=> seven <= "1101111";
        end case;
    end process;
end Behavioral;
```

Unit 18 Problem Solutions

18.3 *See FLD p. 669 for circuit. Notice that the Q output of the flip-flop is b_{in}, while the D input is b_{out}.*



18.4 *See FLD p. 670.* AND-ing with x_i is like *M/Ad* if x_i is 1. Shifting is like moving from AND gates involving x_1 to those involving x_2, or from x_2 to x_3.

18.5 *See FLD p. 670. Compare to divider state graph of FLD Figure 18-11.*

18.6 *See FLD p. 670.*

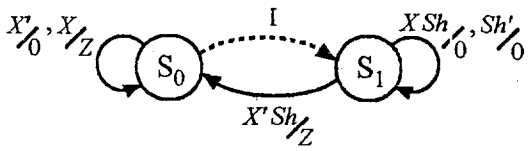
18.7 (a) Overflow occurs only on division by 0, so $V = y_0'y_1'y_2'y_3'y_4' = (y_0 + y_1 + y_2 + y_3 + y_4)'$

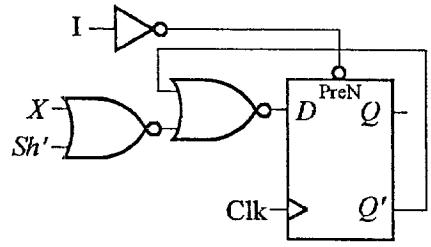
18.7 (b) - (d) *See FLD p. 671.*

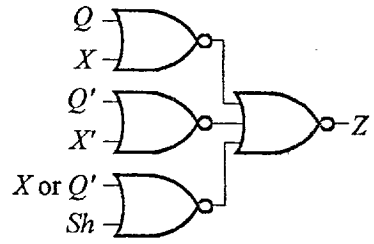
18.8 *See FLD p. 671.*

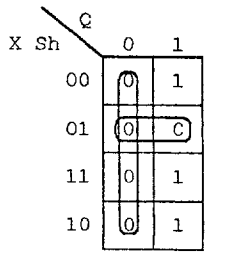
18.9 The ONE ADDER is similar to a serial adder, except that there is only one input. This means that the carry will be added to *X*. Thus, if the carry flip-flop is initially set to 1, 1 will be added to the input. The signal I can be used to preset the carry flip-flop to 1.

Let S_0 represent *Carry* = 0, and let S_1 represent *Carry* = 1. The state graph is as follows:

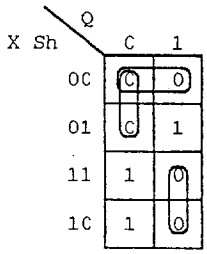






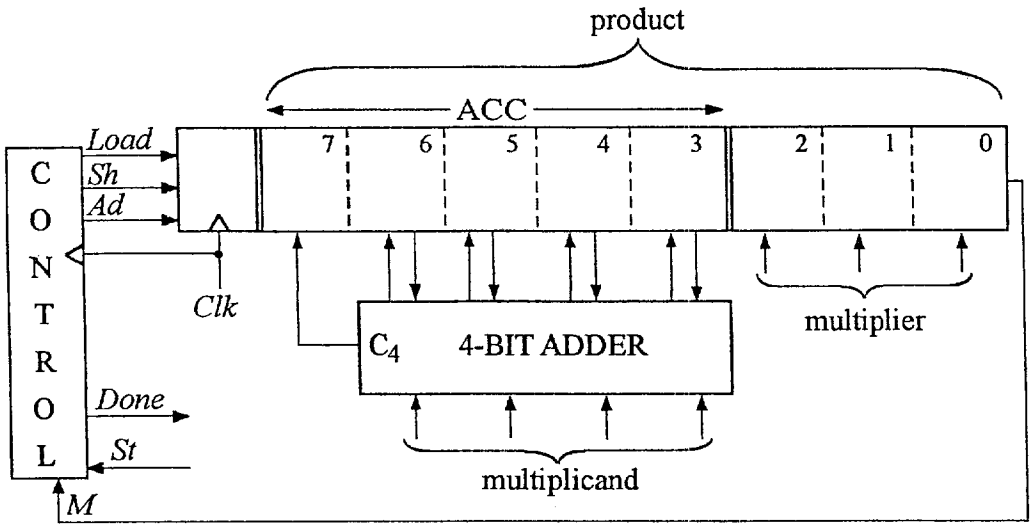


$Q^+ = Q(Sh' + X)$

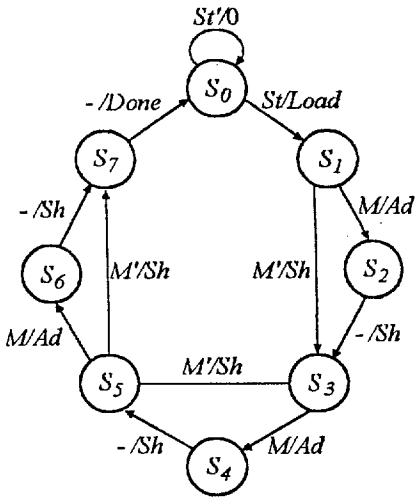


$Z = (Q + X)(Q' + X')(X + Sh)$
$Z = (Q + X)(Q' + X')(Q' + Sh)$

18.10 (a)



18.10 (b)



18.10 (c)

0	0	0	0	0	1	0	1		add
	1	0	1	1					
0	1	0	1	1	1	0	1		shift
0	0	1	0	1	1	1	0		shift
0	0	0	1	0	1	1	1		add
	1	0	1	1					
0	1	1	0	1	1	1	1		shift
0	0	1	1	0	1	1	1		

18.10 (d)

Present State	Next State *StM:* 00	01	10	11	*Ad Sh Load Done* 00	01	10	11
S_0	S_0	S_0	S_1	S_1	0000	0000	0010	0010
S_1	S_3	S_2	S_3	S_2	0100	1000	0100	1000
S_2	S_3	S_3	S_3	S_3	0100	0100	0100	0100
S_3	S_5	S_4	S_5	S_4	0100	1000	0100	1000
S_4	S_5	S_5	S_5	S_5	0100	0100	0100	0100
S_5	S_7	S_6	S_7	S_6	0100	1000	0100	1000
S_6	S_7	S_7	S_7	S_7	0100	0100	0100	0100
S_7	S_0	S_0	S_0	S_0	0001	0001	0001	0001

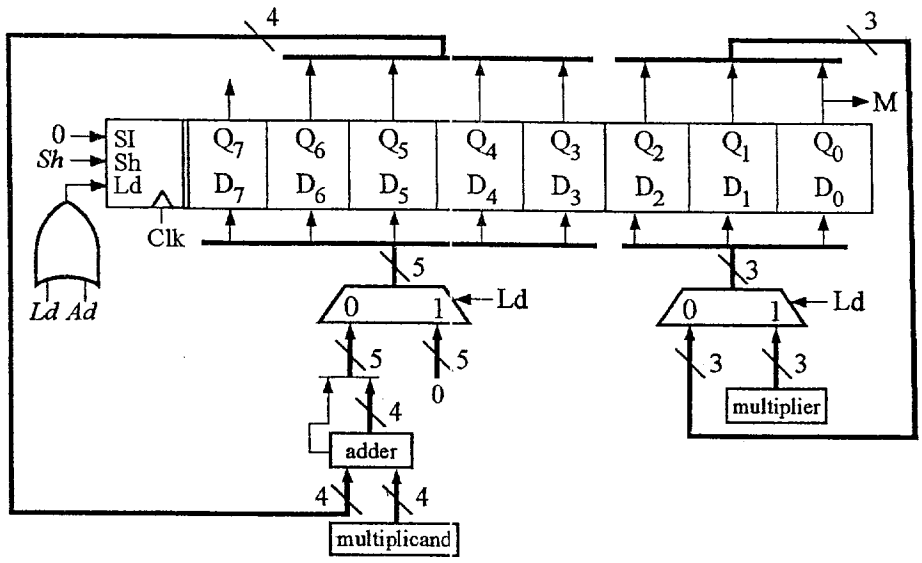
I. (S_0, S_7) (S_1, S_2) (S_3, S_4) (S_5, S_6)
II. (S_0, S_1) (S_2, S_3) (S_4, S_5) (S_6, S_7)
III. (S_1, S_3, S_5) (S_2, S_4, S_6) etc.

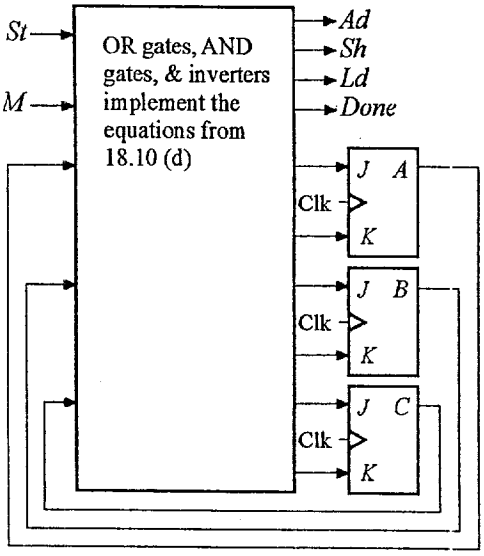
B C \ A	0	1
00	S_0	S_1
01	S_3	S_2
11	S_5	S_4
10	S_7	S_6

(Other assignments are possible.)

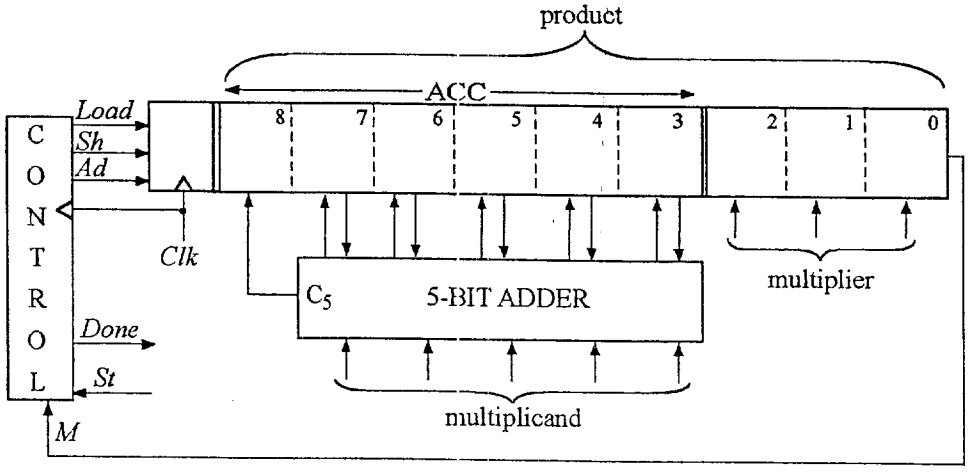
18.10 (d) For this assignment, from LogicAid:

(contd) $J_A = StB'C' + MC$; $K_A = M' + B + C$; $J_B = A'C$; $K_B = A'C'$; $J_C = AB'$; $K_C = A'B$; $Ad = MAB'C' + MA'C$; $Sh = M'A + M'C + AB + AC$; $Load = StA'B'C'$; $Done = A'BC'$





18.11 (a)



18.11 (b) *See solution to 18.10 (b).*

18.11 (c)

0	0	0	0	0	0	1	1	0	shift
0	0	0	0	0	0	0	1	1	add
	1	0	1	0	0				
0	1	0	1	0	0	0	1	1	shift
0	0	1	0	1	0	0	0	1	add
	1	0	1	0	0				
0	1	1	1	1	0	0	0	1	shift
0	0	1	1	1	1	0	0	0	

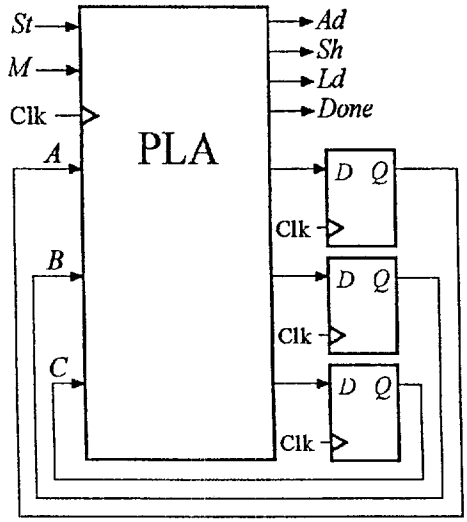
18.11 (d)

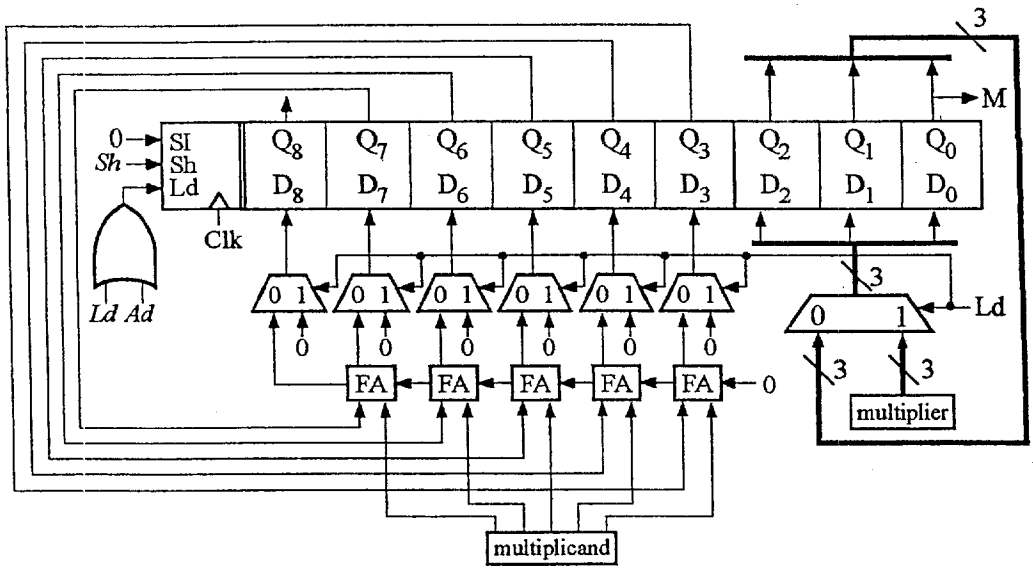
Graph is same as 18.10, so from LogicAid, using the same state assignment:

$D_A = StA'B'C' + MAB'C' + MA'C$
$D_B = A'C + AB$
$D_C = AB' + B'C + AC$

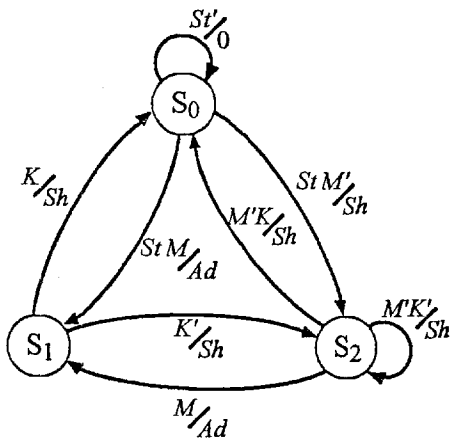
Ad, Sh, Ld, Done: See solution to 18.10 (d)

St	*M*	*A*	*B*	*C*	D_A	D_B	D_C	*Ad*	*Sh*	*Ld*	*Done*
1	-	0	0	0	1	0	0	0	0	1	0
-	1	1	0	0	1	0	0	1	0	0	0
-	1	0	-	1	1	0	0	1	0	0	0
-	-	0	-	1	0	1	0	0	0	0	0
-	-	1	1	-	0	1	0	0	1	0	0
-	-	1	0	-	0	0	1	0	0	0	0
-	-	-	0	1	0	0	1	0	0	0	0
-	-	1	-	1	0	0	1	0	1	0	0
-	0	1	-	-	0	0	0	0	1	0	0
-	0	-	-	1	0	0	0	0	1	0	0
-	-	0	1	0	0	0	0	0	0	0	1





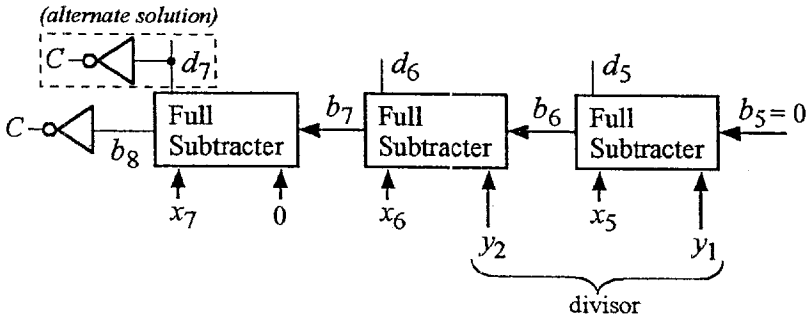
18.12 (a)



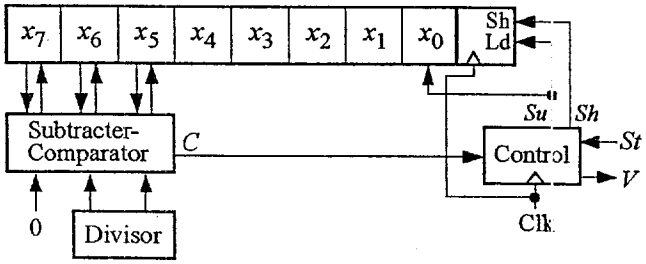
18.12 (b)

State	Counter	X	St M K Ad Sh
S_0	00	000000111	1 1 0 1 0
S_1	00	011001111	0 1 0 0 1
S_2	01	001100111	0 1 0 1 0
S_1	01	100101111	0 1 0 0 1
S_2	10	010010111	0 1 1 1 0
S_1	10	101011111	0 1 1 0 1
S_0	00	010101111	0 1 0 0 0

18.13 (a)



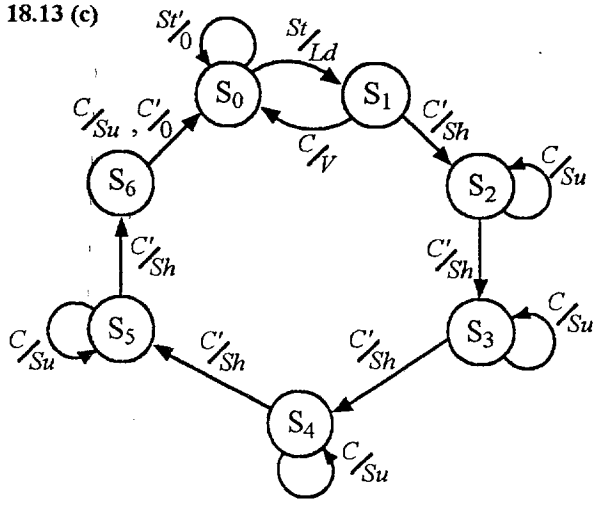
18.13 (b)



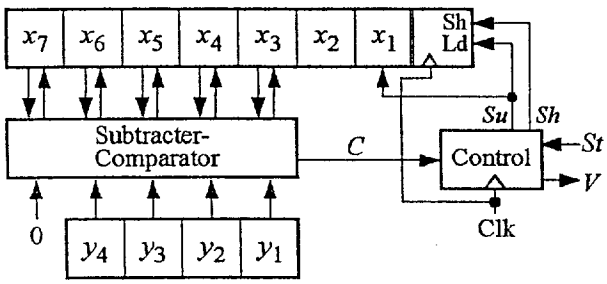
18.13 (d)

```
0 1 0 1 0 0 1 1
  1 1                       shift C = 0
1 0 1 0 0 1 1 0
  1 1                       sub. C = 1
0 1 0 0 0 1 1 1             shift C = 0
1 0 0 0 1 1 1 0
  1 1                       sub. C = 1
0 0 1 0 1 1 1 1             shift C = 0
0 1 0 1 1 1 1 0
  1 1                       shift C = 0
1 0 1 1 1 1 0 0
  1 1                       sub. C = 1
0 1 0 1 1 1 0 1             shift C = 0
1 0 1 1 1 0 1 0
  1 1                       sub. C = 1
0 1 0 1 1 0 1 1               C = 0
remainder  quotient
```

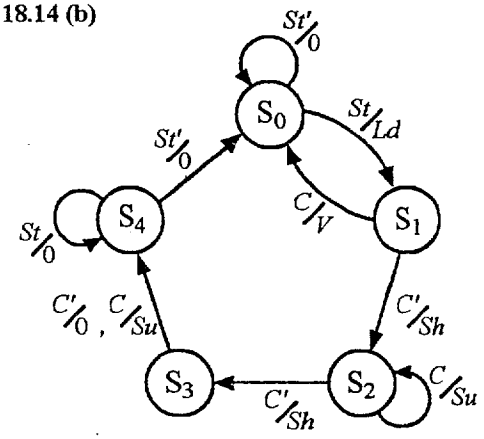
18.13 (c)



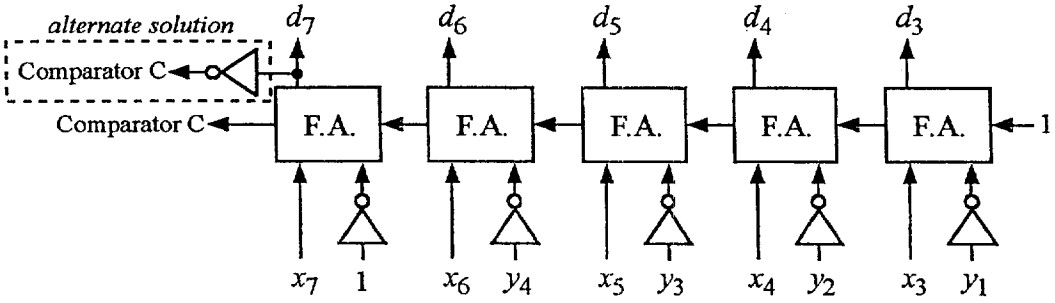
18.14 (a)



18.14 (b)



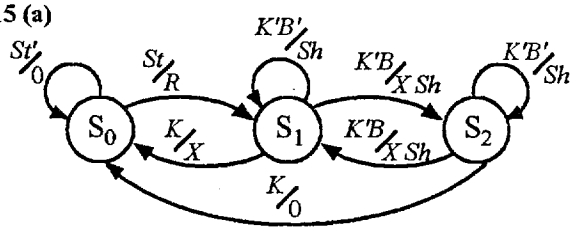
18.14 (c)



18.14 (d)

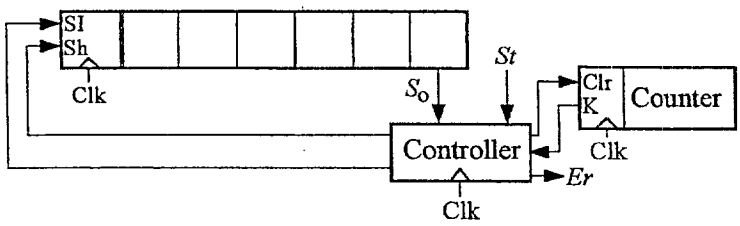
0 1 0 1 1 0 1		
1 1 0 1		shift C = 0
1 0 1 1 0 1	0	
1 1 0 1		sub. C = 1
0 1 0 0 1 1	1	
1 1 0 1		shift C = 0
1 0 0 1 1	1 0	
1 1 0 1		sub. C = 1
0 0 1 1 0	1 1	
remainder	quotient	

18.15 (a)

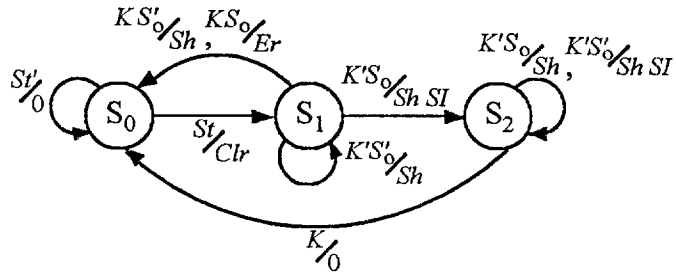


18.15 (b) $D_0 = St'Q_0 + KQ_1 + KQ_2$; $D_1 = StQ_0 + K'B'Q_1 + K'BQ_2$; $D_2 = K'BQ_1 + K'B'Q_2$; $R = StQ_0$
$Sh = K'B'Q_1 + K'BQ_1 + K'BQ_2 + K'B'Q_2 = K'Q_1 + K'Q_2$; $X = KQ_1 + K'BQ_1 + K'BQ_2 = KQ_1 + BQ_1 + K'BQ_2$

18.16 (a)

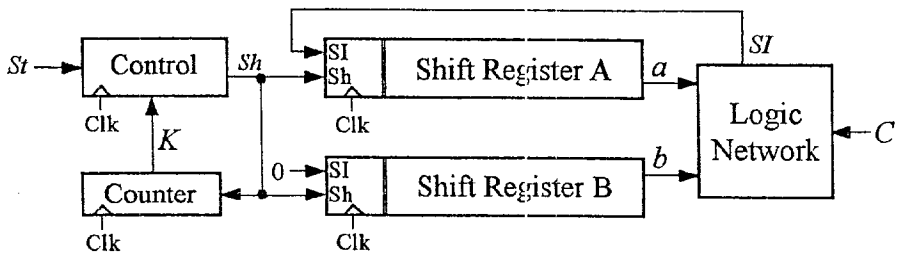


18.16 (b)

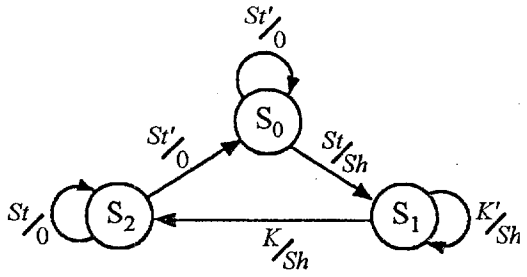


18.16 (c) $D_0 = St'Q_0 + KQ_1 + KQ_2'$; $D_1 = K'X_0'Q_1 + StQ_0$; $D_2 = K'X_0Q_1 + K'Q_2'$; $Clr = StQ_0$
$Sh = K'X_0'Q_1 + KX_0'Q_1 + K'X_0Q_1 + K'Q_2'$; $Er = KX_0'Q_1'$; $SI = K'X_0Q_1 + K'X_0'Q_2$

18.17 (a)



18.17 (b)



Present State	StK 00	01	11	10	Sh 00	01	11	10
S_0	S_0	S_0	S_1	S_1	0	0	1	1
S_1	-	-	S_2	S_1	-	-	1	1
S_2	S_0	S_0	S_2	S_2	0	0	0	0

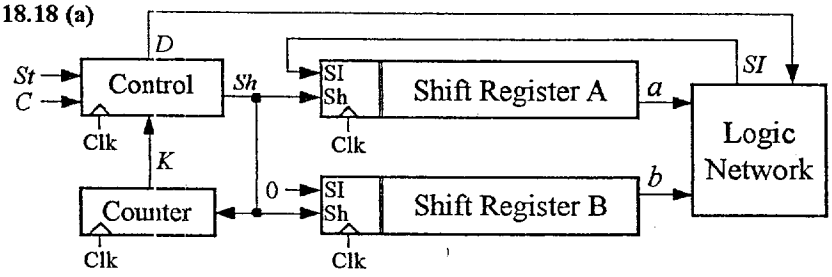
18.17 (c) I. $(S_0, S_2)\times 2$ (S_1, S_2) (S_0, S_1)
II. $(S_0, S_2)\times 2$ (S_1, S_2) $(S_0, S_1)\times 2$
From Karnaugh maps:
$D_0 = Q_0^+ = StQ_0 + KQ_0'Q_1$
$D_1 = Q_1^+ = St$; $Sh = StQ_0'$
Alternative: $Q_0^+ = StQ_0 + StKQ_1$

C \	0	1
0	S_0	
1	S_1	S_2

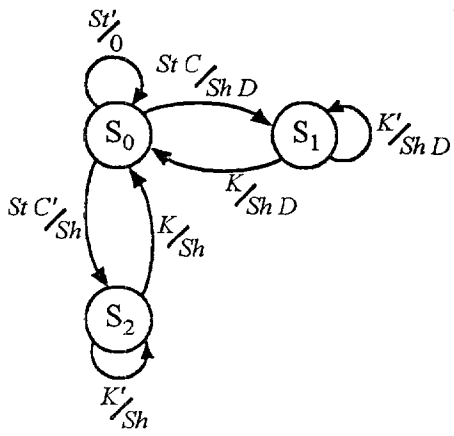
18.17 (d) $SI = C'ab + Cab' + Ca'b$

St	K	Q_0	Q_1	D_0	D_1	Sh
1	-	1	-	1	0	0
-	1	0	1	1	0	0
1	-	-	-	0	1	0
1	-	0	-	0	0	1

18.18 (a)



18.18 (b)

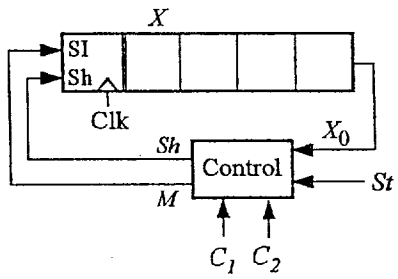


State	Meaning
S_0	Reset
S_1	Find AND of A & B
S_2	Find XOR of A & B

18.18 (c) $Q_0^+ = St'Q_0 + KQ_1 + KQ_2$; $Q_1^+ = StCQ_0 + K'Q_1$;
$Q_2^+ = StC'Q_0 + K'Q_2$;
$Sh = StCQ_0 + StC'Q_0 + K'Q_1 + KQ_1 + K'Q_2 + KQ_2$
$D = StCQ_0 + K'Q_1 + KQ_1$

St	C	K	Q_0	Q_1	Q_2	Q_1^+	Q_2^+	Q_3^+	Sh	D
0	-	-	1	-	-	1	0	0	0	0
-	-	1	-	1	-	1	0	0	1	1
-	-	1	-	-	1	1	0	0	1	0
1	1	-	1	-	-	0	1	0	1	1
-	-	0	-	1	-	0	1	0	1	1
1	0	-	1	-	-	0	0	1	1	0
-	-	0	-	-	1	0	0	1	1	0

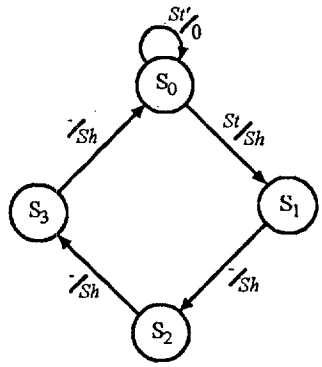
18.19 (a)



18.19 (c) $J_A = B$; $K_A = B$; $J_B = St + A$; $K_B = 1$;
$Sh = St + A + B$; $M = C_1'C_2 + X_0'C_1C_2$

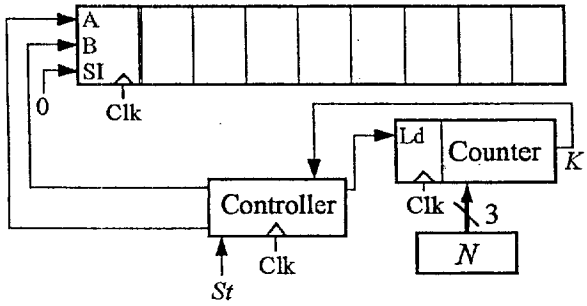
18.18 (d) Change C' to D' in 18.17 (d)
$SI = D'ab + Dab' + Da'b$

18.19 (b)

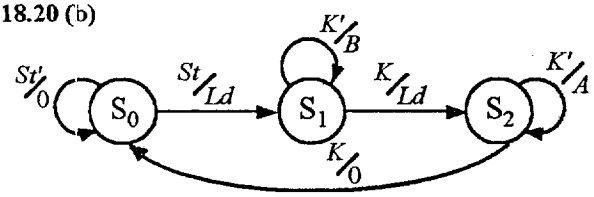


Note: M can be determined independently of the state of the system, so it is not included in the state graph.

18.20 (a)



18.20 (b)

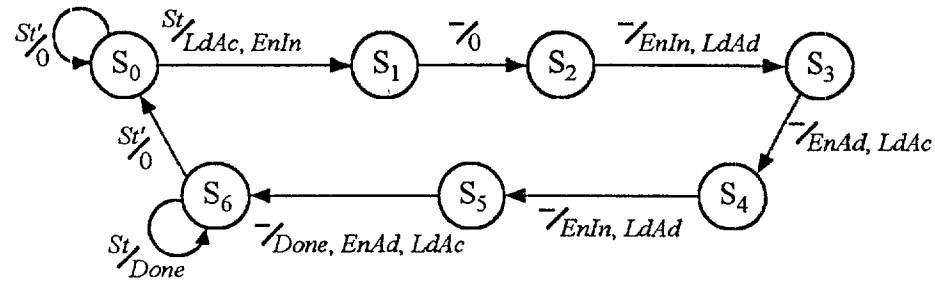


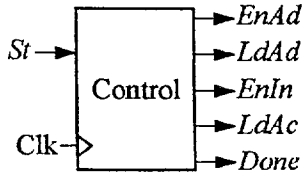
18.20 (c)

State	StK 00	01	11	10	ABLd 00	01	11	10
S_0	S_0	S_0	S_1	S_1	000	000	001	001
S_1	S_1	S_2	-	-	010	001	-	-
S_2	S_2	S_0	-	-	100	000	-	-

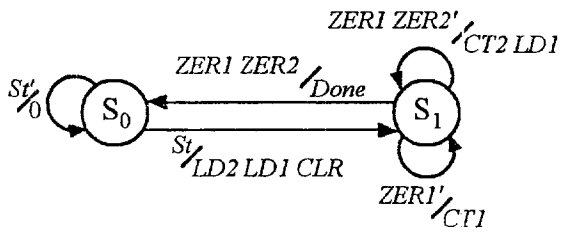
$D_1 = KQ_2 + K'Q_1$; $D_2 = St + K'Q_2$; $A = K'Q_1$;
$B = K'Q_2$; $Ld = St + KQ_2$

18.21





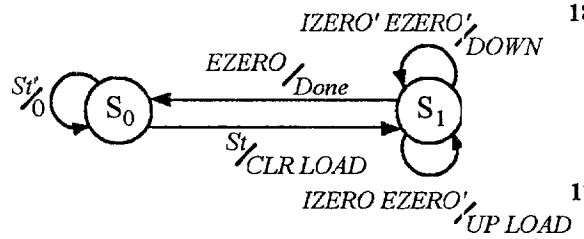
18.22 (a)



18.22 (b) *J = ST; K = ZER1 ZER2;*
Done = ZER1 ZER2 Q; CLR = STQ';
LD2 = STQ'; LD1 = STQ' + ZER1 ZER2' Q;
CT1 = ZER1' Q; CT2 = ZER1 ZER2' Q

18.22 (c) $(N_1 + 1)N_2$ cycles

18.23 (a)



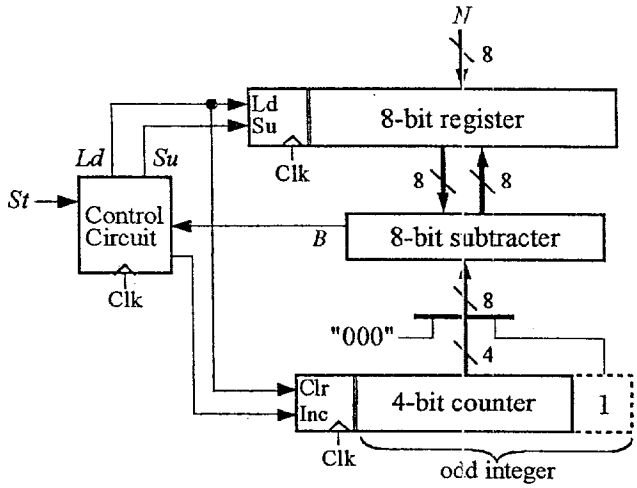
18.23 (b) *D = EZERO' Q + StQ'; Done = EZERO Q;*
CLR = StQ';
LOAD = StQ' + IZERO EZERO' Q
DOWN = IZERO' EZERO' Q
UP = IZERO EZERO' Q

18.23 (c) $N_1 + (N_1/N_2)$ cycles (round down)

18.23 (d) The quotient counter reaches 1111, and *UP* = 1 again.

18.23 (e) The quotient will count upward forever, and *Done* will never be 1.

18.24



St - start

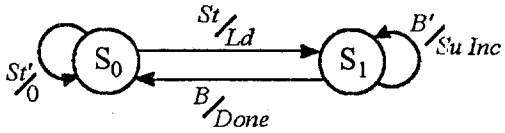
Ld - load N into register and clear counter

Su - load subtracter output into register

Inc - increment counter

B - borrow

When the done signal comes on, square root is in the 4-bit counter



Unit 19 Problem Solutions

19.1 *See FLD p. 672 for solution.*

19.2 *See FLD p. 672 for solution.*

19.3 *See FLD p. 672 for solution.*

19.4 *See FLD p. 673 for solution.*

19.5 *See FLD p. 673 for solution.*

19.6 *See FLD p. 674 for solution.*

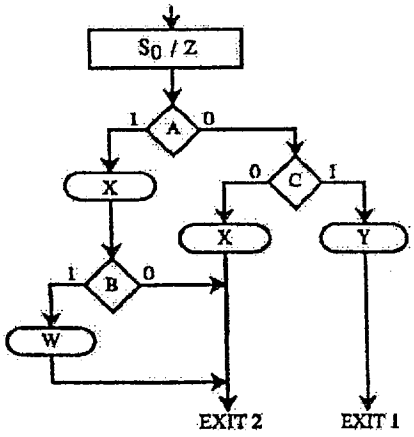
19.7 *See FLD p. 674 for solution.*

19.8 *See FLD p. 674 for solution.*

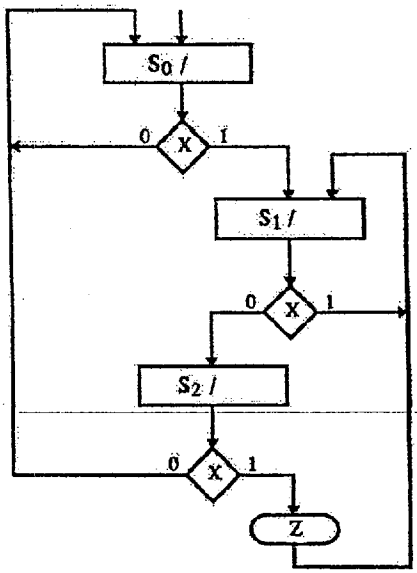
19.9 *See FLD p. 674 for solution.*

19.10 *See FLD p. 675 for solution.*

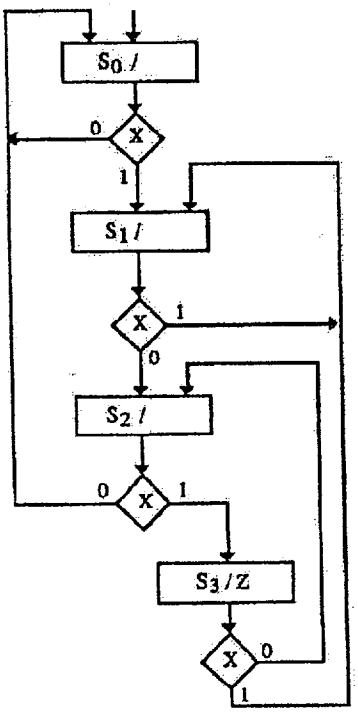
19.11



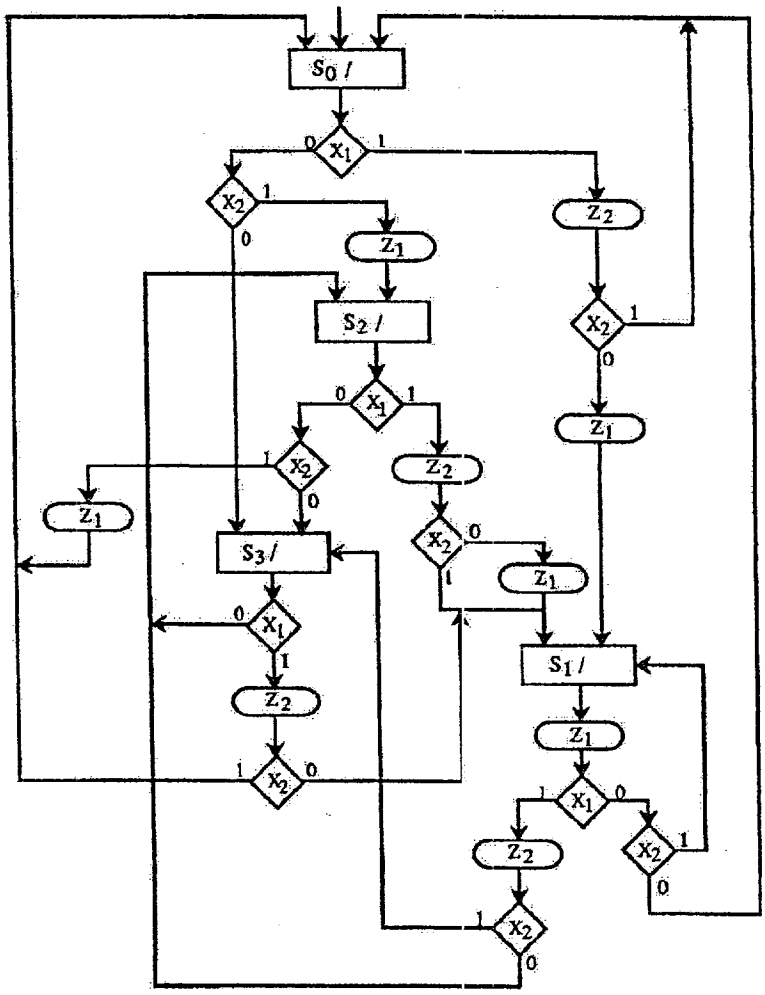
19.12 (a)



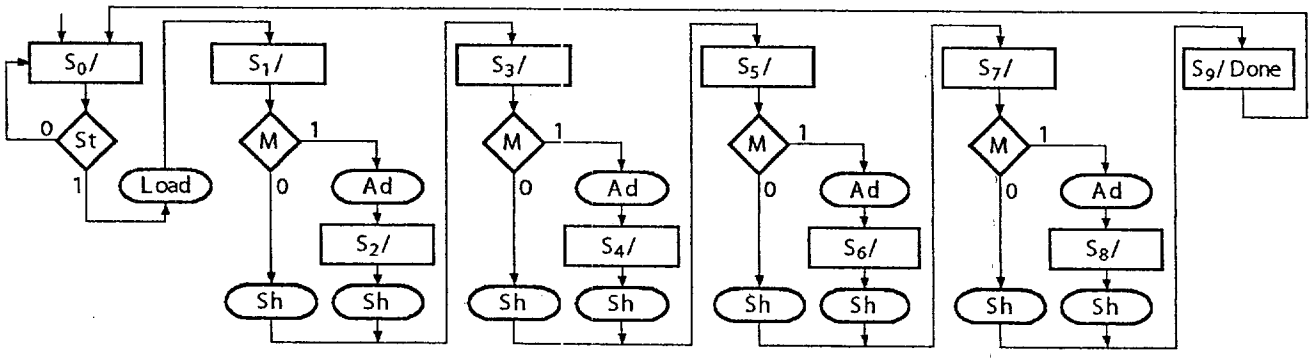
19.12 (b)



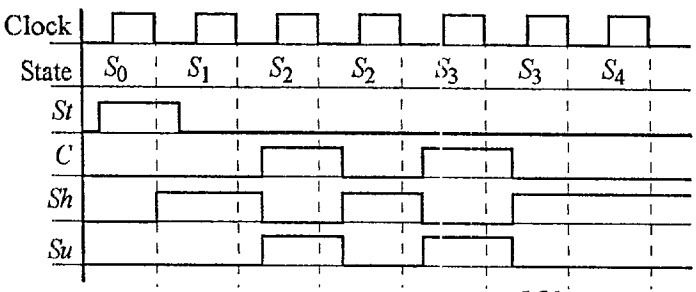
19.13



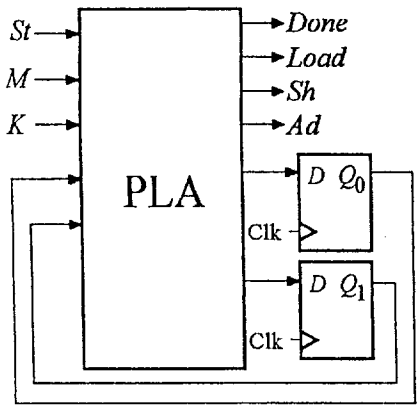
19.14



19.15

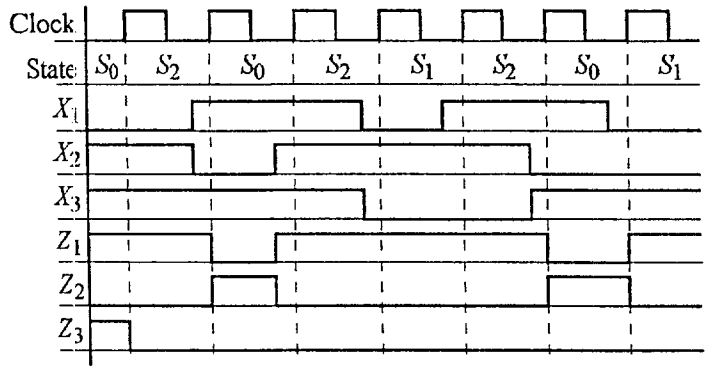


19.16



State	$Q_0 Q_1$	St	M	K	Q_0^+	Q_1^+	Ad	Sh	Load	Done
S_0	0 0	0	-	-	0	0	0	0	0	0
S_0	0 0	1	-	-	0	1	0	0	1	0
S_1	0 1	-	0	0	0	1	0	1	0	0
S_1	0 1	-	0	1	1	0	0	1	0	0
S_1	0 1	-	1	-	1	1	1	0	0	0
S_2	1 1	-	-	0	0	1	0	1	0	0
S_2	1 1	-	-	1	1	0	0	1	0	0
S_3	1 0	-	-	-	0	0	0	0	0	1

19.17 (a)



19.17 (b)

$A^+ = A'BX_2 + A'B'X_2 (X_1' + X_3) + \{AB\}$
$= BX_2 + A'X_2 (X_1' + X_3)$
$B^+ = A'B' (X_2' + X_1X_3') + AB'X_1' + A'BX_2' + \{AB\}$
$= AX_1' + A'B'X_1X_3' + A'X_2'$
$Z_1 = A + B + X_2$; $Z_2 = A'B'X_2'$; $Z_3 = A'B'X_1'X_2$

In the preceding equations, curly brackets ({}) indicate a don't care term.

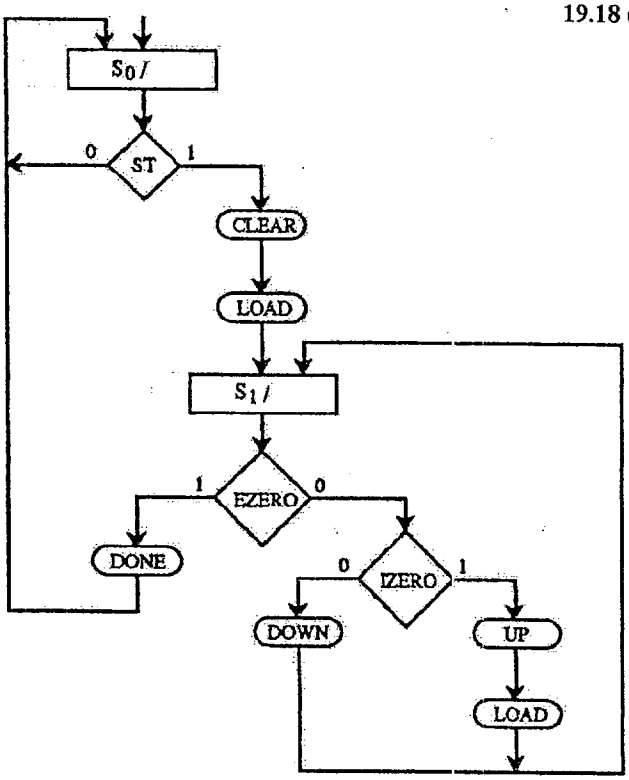
19.17 (c) PLA table obtained by tracing link paths:

State	AB	$X_1X_2X_3$	A^+B^+	$Z_1Z_2Z_3$
S_0	00	- 0 -	01	010
	00	01 -	10	101
	00	110	01	100
	00	111	10	100
S_1	01	- 0 -	01	100
	01	- 1 -	10	100
S_2	10	0 - -	01	100
	10	1 - -	00	100

19.17 (d) $2^5 \times 5$ ROM

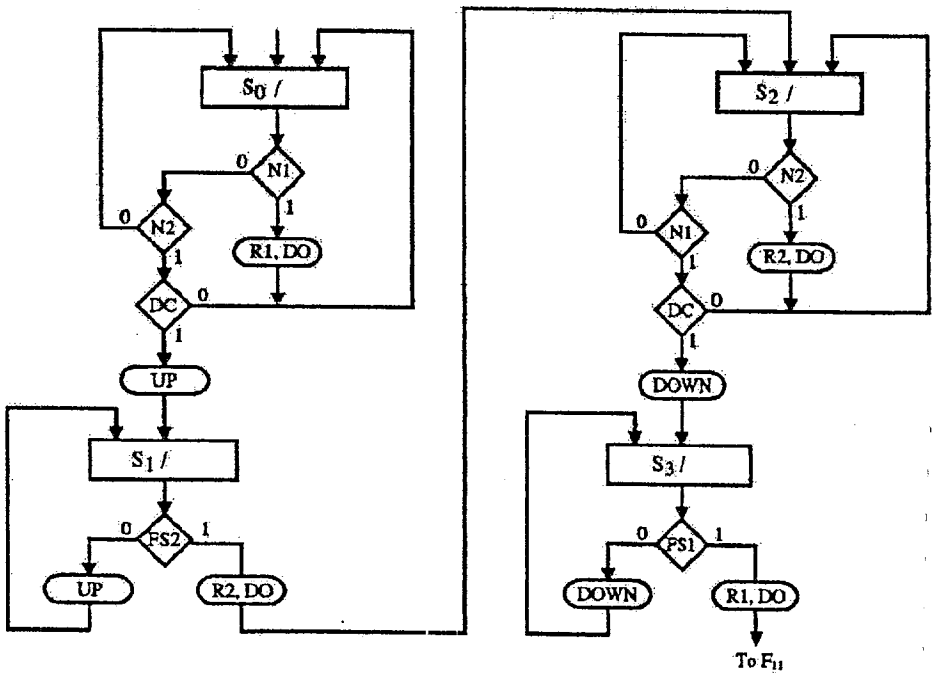
AB	$X_1X_2X_3$	A^+B^+	$Z_1Z_2Z_3$
00	000	01	010
00	001	01	010
00	010	10	001
00	011	10	101
00	100	01	010

19.18 (a)



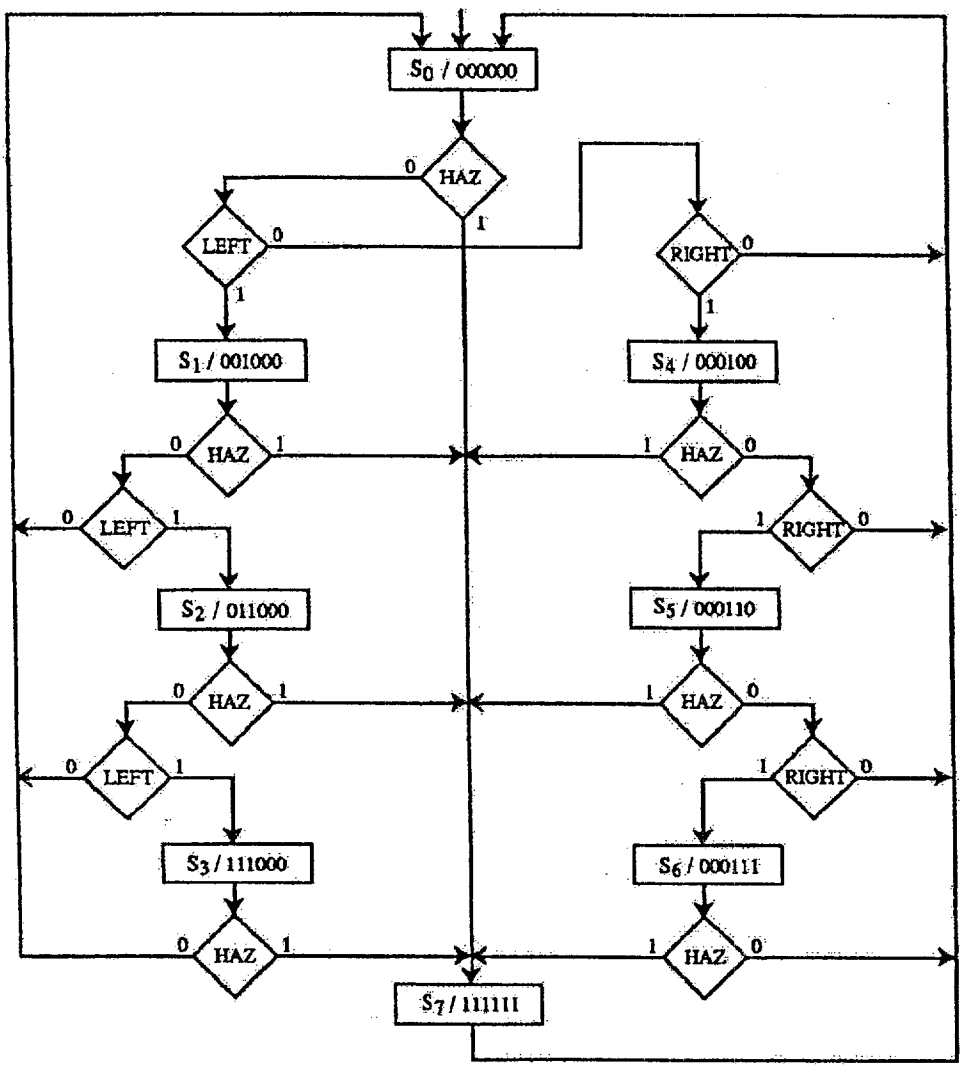
19.18 (b) *See answer to 18.23 (b) on page 149.*

19.19 (a)

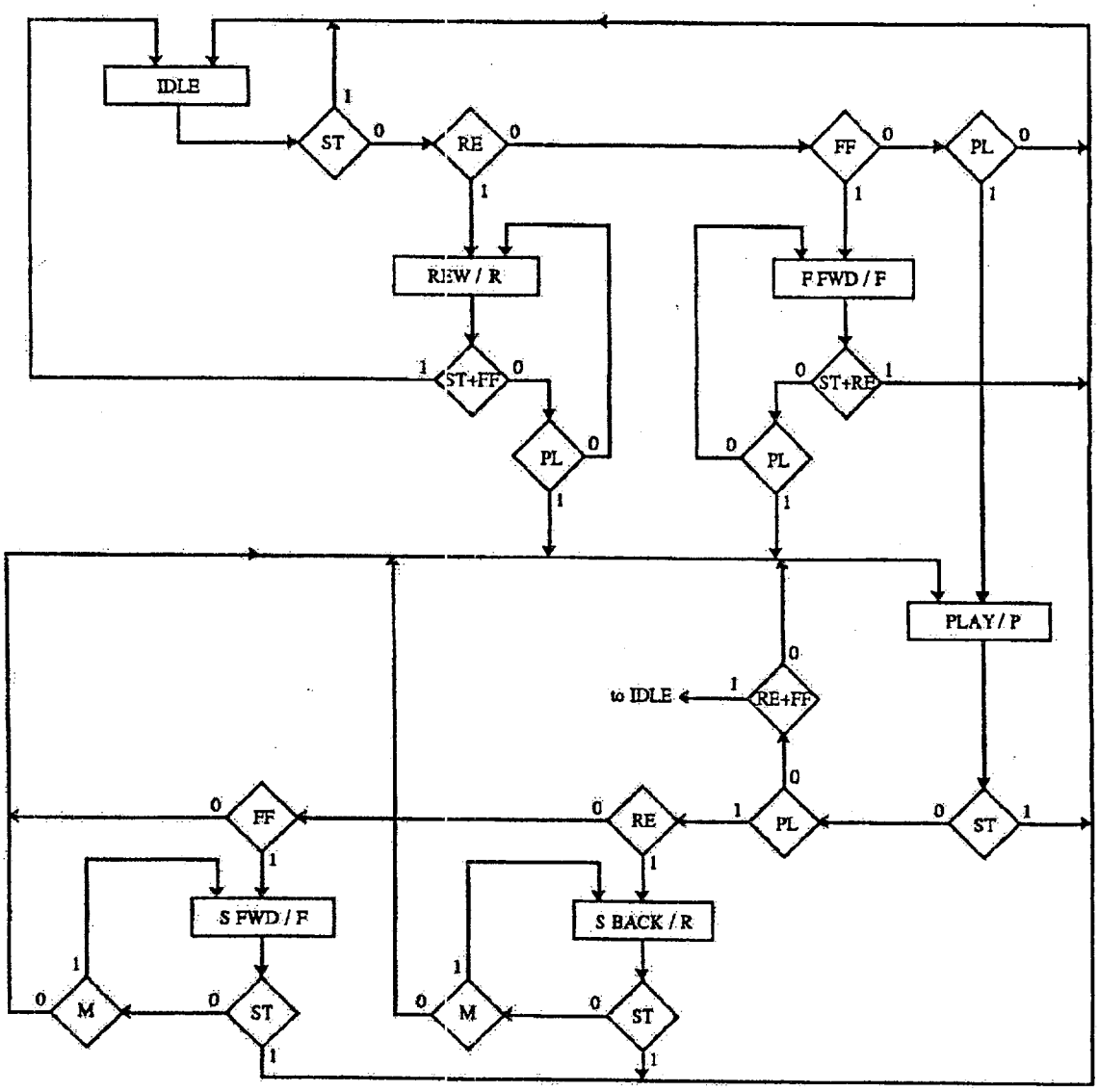


19.19 (b) *See answer to 16.18 (c) on page 132.*

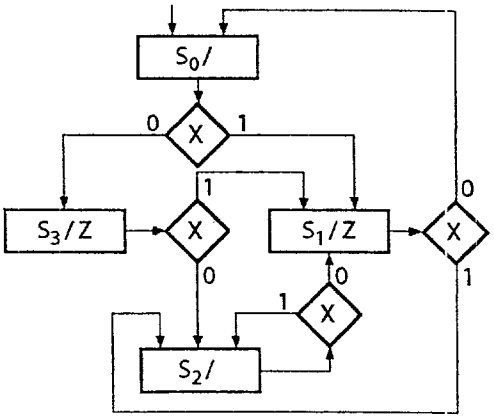
19.20


S0 / 000000
HAZ
LEFT
RIGHT
S1 / 001000
S4 / 000100
HAZ
HAZ
LEFT
RIGHT
S2 / 011000
S5 / 000110
HAZ
HAZ
LEFT
RIGHT
S3 / 111000
S6 / 000111
HAZ
HAZ
S7 / 111111

19.21


IDLE
ST
RE
FF
PL
REW / R
F FWD / F
ST+FF
ST+RE
PL
PL
PLAY / P
to IDLE
RE+FF
FF
RE
PL
ST
S FWD / F
S BACK / R
M
ST
M
ST

19.22


S0/
X
S3/Z
X
S1/Z
X
X
S2/

Unit 20 Problem Solutions

20.1 *See FLD p. 676 for solution.*

20.2 *See FLD p. 676-677 for solution.*

20.3 Replace line 12 with:

```
signal State, Nextstate: integer range 0 to 5;
```

Replace lines 27 - 33 with:

```
when 1 | 2 | 3 | 4 =>
    if M = '1' then Ad <='1';
    Nextstate <= State;
    else Sh <='1'; Nextstate <= State + 1; end if;
when 5 =>   Done <= '1'; Nextstate <= 0;
```

Replace lines 39 - 41 with:

```
if Load = '1' then  ACC <= "00000" & MPlier; end if;
if Ad = '1' then ACC(8 downto 4) <= addout; ACC(0) <= '0'; end if;
```

20.4 See FLD p. 677-678 for solution.

20.5 See FLD p. 678 for solution.

20.6 See FLD p. 679 for solution.

20.7 Replace line 14 with:

```
signal Counter: integer range 0 to 4;
signal State, NextState: integer range 0 to 3;
```

After line 22, add:

```
K<='1' when Counter=3 else '0';
```

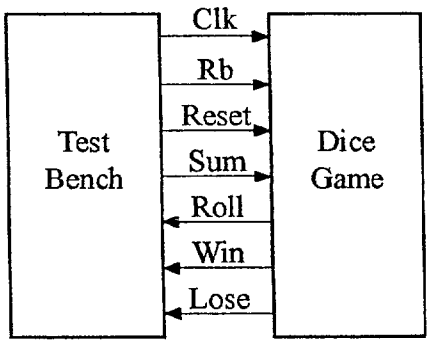
Replace lines 33 - 36 with:

```
when 2 =>
    if C = '1' then Su <= '1'; NextState <= 2;
    elsif K = '1' then Sh <='1'; NextState <= 3;
    else Sh <= '1'; NextState <= 2; end if;
when 3 =>
```

Replace line 47 with:

```
if Sh = '1' then Dividend <= Dividend (7 downto 0) & '0';
    Counter <= Counter + 1; end if;
```

20.8



20.8 (contd) Entity and architecture for DiceGame goes here.

```
entity GameTest is
end GameTest;
architecture dicetest of GameTest is
component DiceGame
    port (CLK, Rb, Reset : in bit;
        Sum: in integer range 2 to 12 ;
        Roll, Win, Lose: out bit);
end component;
signal rb, reset, clk, roll, win, lose: bit;
signal sum: integer range 2 to 12;
type arr is array(0 to 11) of integer;
constant Sumarray:arr := (7,11,2,4,7,5,6,7,6,8,9,6);
    begin
    CLK <= not CLK after 20 ns;
    Dice: Dicegame port map(rb,reset,clk,sum,roll,win,lose);
```

Contintued next column

```
process
    begin
    for i in 0 to 11 loop
        Rb <= '1';  -- push roll button
        wait until roll = '1';
        wait until clk'event and clk = '1';
        Rb <= '0';  -- release roll button
        wait until roll <= '0';
        sum <= Sumarray(i);
        -- read roll of dice from array
        wait until clk'event and clk = '1';
        wait until clk'event and clk = '1';
        if win = '1' or lose = '1' then reset <= '1';
        end if;
        wait until clk'event and clk = '1';
        reset <= '0';
    end loop;
  wait;   -- test completed, do not execute process
  again
  end process;
end dicetest;
```

20.9 Replace lines 6 - 11 with:

```
Port (Dividend_in: in std_logic_vector(4 downto 0);
      Divisor: in std_logic_vector(4 downto 0);
      St, Clk: in std_logic;
      Quotient: out std_logic_vector(4 downto 0);
      Remainder: out std_logic_vector(4 downto 0);
```

Replace lines 14 - 17 with:

```
signal State, NextState: integer range 0 to 6;
signal C, Load, Su, Sh, V: std_logic;
signal Subout : std_logic_vector (4 downto 0);
signal Dividend: std_logic_vector (9 downto 0);
```

Replace lines 19 - 23 with:

```
Subout <= Dividend(9 downto 5) - Divisor;
C <= not Subout (4);
Remainder <= Dividend (9 downto 5);
V <= '1' when Divisor = "00000" else '0';
Quotient <= Dividend  (4 downto 0);
State_Graph: process (State, St, C, V)
```

Replace line 25 with:

```
Load <= '0'; Sh <= '0'; Su <= '0';
```

Replace lines 28 - 33 with:

```
if (St = '1') then
    if (V='0') then Load <='1'; NextState <= 1;
    else Nextstate <= 0; end if;
else Nextstate <= 0; end if;
when 1 => Sh <='1'; NextState <= 2;
when 2 | 3 | 4 | 5 =>
```

Replace line 36 with:

```
when 6 =>
```

Replace lines 45 - 47 with:

```
if Load = '1' then Dividend <= "00000" & Dividend_in; end if;
if Su = '1' then Dividend(9 downto 5) <= Subout; Dividend(0) <= '1'; end if;
if Sh = '1' then Dividend <= Dividend (8 downto 0) & '0'; end if;
```

III. SOLUTIONS TO DESIGN, SIMULATION, AND LAB EXERCISES

Solutions to Unit 8 Design Problems

Problems 8.A through 8.S are combinational logic design problems using NAND and NOR gates. Problems 8.A through 8.R are of approximately equal difficulty so that different students in the class can be assigned different problems. We ask our students to use the following procedure:

(1) Derive a truth table for the assigned problem.
(2) Use Karnaugh maps to derive logic equations in sum-of-products or product-of-sums form depending on whether NAND gates or NOR gates are required.
(3) Enter the truth table into LogicAid, derive the logic equations, and check the answers against the results of step (2).
(4) Draw a circuit of AND and OR gates, trying to minimize the number of gates required by using common gates where appropriate. Factoring or multiplying out is required in some cases.
(5) Convert to NAND or NOR gates as specified.
(6) Simulate your answer to (5) using SimUaid, and verify that the circuit works correctly. Use switches as inputs and probes or a 7-segment indicator as outputs.

In Unit 10, we ask our students to implement the same design problem using VHDL, synthesize it and download it to a CPLD on a hardware board that has switches, LEDs, and 7-segment indicators.

For each design problem, the solutions that follow show a SimUaid circuit that meets the problem specifications, but the solution does not necessarily use the minimum number of gates. Each solution shows the truth table and the equations derived using LogicAid, and in several cases the Karnaugh maps are shown to help identify common terms.

8.A

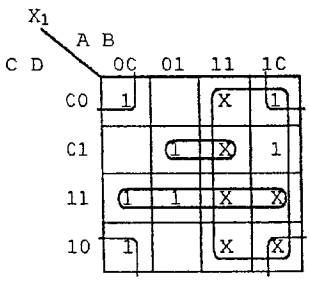
$ABCD$	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	0	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1

$X_1 = B'D' + BD + A + CD = \underline{B'D'} + \underline{BC'D} + A + \underline{CD}$ (used in circuit)
$X_1 = B'D' + BD + A + B'C$
$X_2 = B' + \underline{C'D'} + \underline{CD}$
$X_3 = C' + D + B$
$X_4 = \underline{B'D'} + \underline{B'C} + \underline{BC'D} + \underline{CD'}$
$X_5 = \underline{B'D'} + \underline{CD'}$
$X_6 = \underline{C'D'} + \underline{BC'} + BD' + A$
$X_7 = \underline{B'C} + \underline{BC'} + A + \underline{CD'}$ (used in circuit)
$X_7 = B'C + BC' + A + BD'$

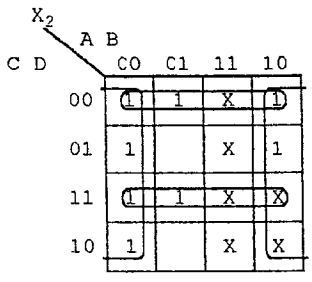
This solution uses 15 gates and 41 gate inputs.

Students are allowed to use a maximum of 18 gates.

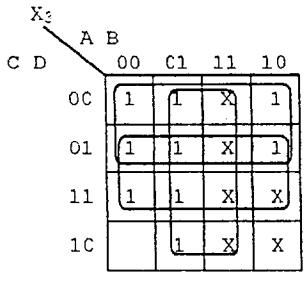
8.A, continued



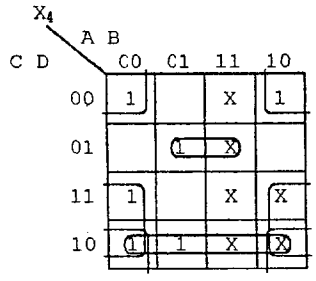
$X_1 = B'D' + BC'D + A + CD$



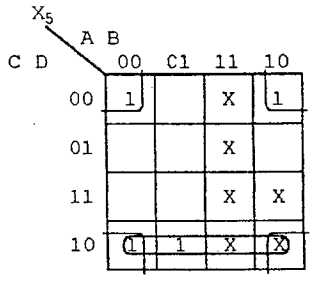
$X_2 = B' + C'D' + CD$



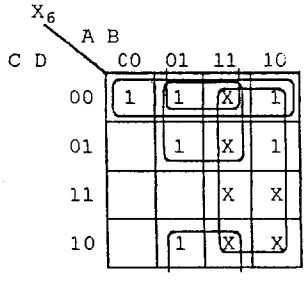
$X_3 = C' + D + B$



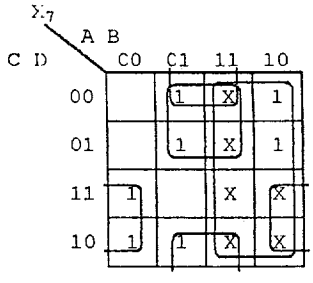
$X_4 = B'D' + B'C + BC'D + CD'$



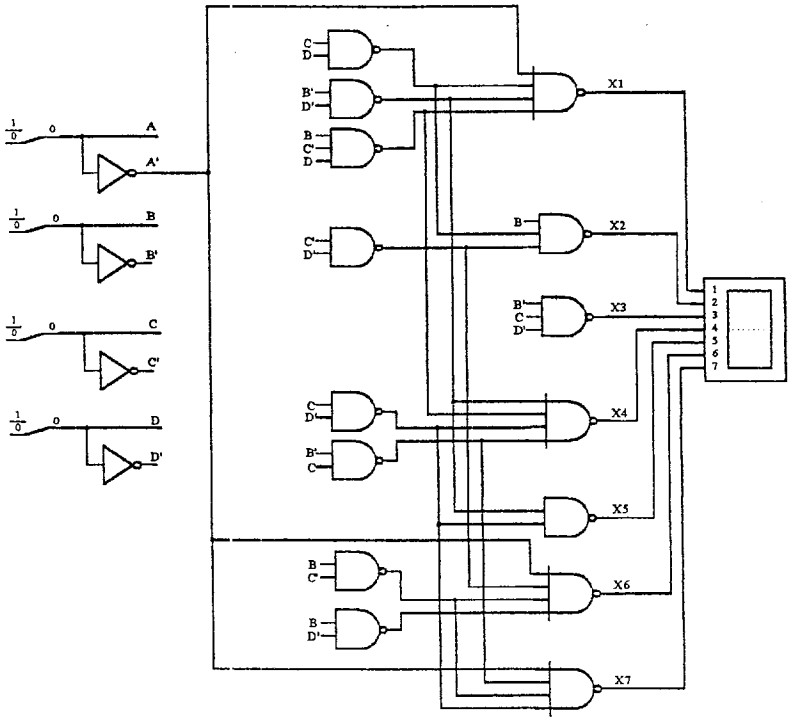
$X_5 = B'D' + CD'$



$X_6 = C'D' + BC' + BD' + A$



$X_7 = B'C + BC' + A + BD'$



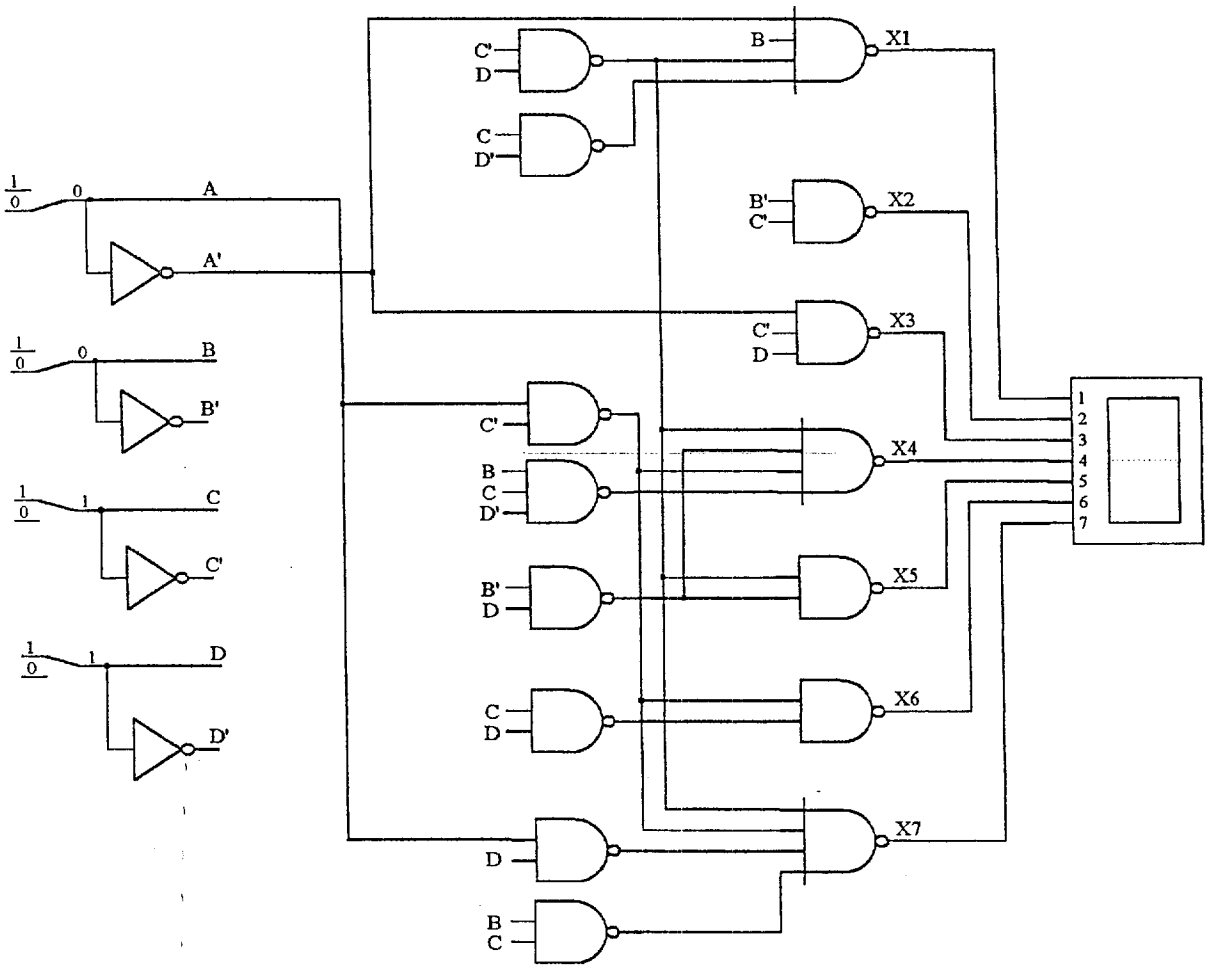
8.B

$ABCD$	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0000	X	X	X	X	X	X	X
0001	X	X	X	X	X	X	X
0010	X	X	X	X	X	X	X
0011	1	1	1	1	1	1	0
0100	0	1	1	0	0	0	0
0101	1	1	0	1	1	0	1
0110	1	1	1	1	0	0	1
0111	0	1	1	0	0	1	1
1000	1	0	1	1	0	1	1
1001	1	0	1	1	1	1	1
1010	1	1	1	0	0	0	0
1011	1	1	1	1	1	1	1
1100	1	1	1	1	0	1	1

$X_1 = B' + \underline{C'D} + C\,D' + A$
$X_2 = C + B$
$X_3 = D' + C + A$ (used in circuit)
$X_3 = D' + C + B'$
$X_4 = \underline{C'D} + \underline{B'D} + B\,C\,D' + \underline{A\,C'}$ (used in circuit)
$X_4 = C'D + A'C\,D' + B'D + A\,C'$
$X_5 = \underline{C'D} + \underline{B'D}$
$X_6 = C\,D + \underline{A\,C'}$
$X_7 = A\,D + B\,C + \underline{C'D} + \underline{A\,C'}$ (used in circuit)
$X_7 = A\,D + B\,C + A\,C' + B\,D$

This solution uses 15 gates and 38 gate inputs.

Students are allowed to use a maximum of 16 gates.



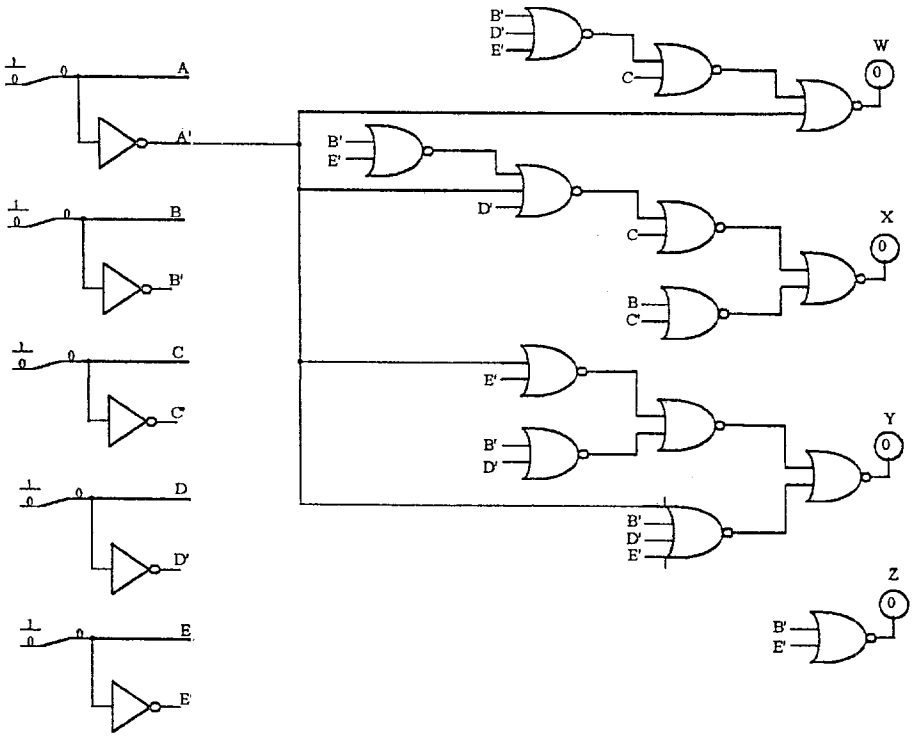
8.C

ABCDE	*W*	*X*	*Y*	*Z*
00000	0	0	0	0
00001	0	0	0	0
00010	0	0	0	0
00011	0	0	0	0
00100	0	0	0	0
00101	0	0	0	0
00110	X	X	X	X
00111	X	X	X	X
01000	0	0	0	0
01001	0	0	0	1
01010	0	0	1	0
01011	0	0	1	1
01100	0	1	0	0
01101	0	1	0	1
01110	X	X	X	X
01111	X	X	X	X
10000	0	0	0	0
10001	0	0	1	0
10010	0	1	0	0
10011	0	1	1	0
10100	1	0	0	0
10101	1	0	1	0
10110	X	X	X	X
10111	X	X	X	X
11000	0	0	0	0
11001	0	0	1	1
11010	0	1	1	0
11011	1	0	0	1
11100	1	1	0	0
11101	1	1	1	1

$W = A(C + D)(B + C)(C + E) = A(C + BDE)$ (used in circuit)
$W = A(C + D)(B + C)(D' + E)$
$W = A(C + D)(C + E)(B + D')$
$W = A(C + D)(B + D')(D' + E)$
$X = (C + D)(B' + C + E')(A + C)(B + C') =$
$(B + C')(C + AD(B' + E'))$ (used in circuit)
$X = (C + D)(B' + C + E')(B + C')(A + D')$
$X = (C + D)(B + C')(A + D')(B' + D' + E')$
$X = (C + D)(B' + C + E')(B + D)(A + D')$
$X = (C + D)(B + D)(A + D')(B' + D' + E')$
$X = (C + D)(A + C)(B + C')(B' + D' + E')$
$X = (C + D)(B' + C + E')(A + C)(B + D)$
$X = (C + D)(A + C)(B + D)(B' + D' + E')$
$Y = (A + B)(A + D)(B + E)(D + E)(A' + B' + D' + E') = (A + BD)(E + BD)$
$(A' + B' + D' + E') = (AE + BD)(A' + B' + D' + E')$
$Z = BE$

This solution uses 14 gates and 32 gate inputs.

Student are allowed to use a maximum of 15 gates.



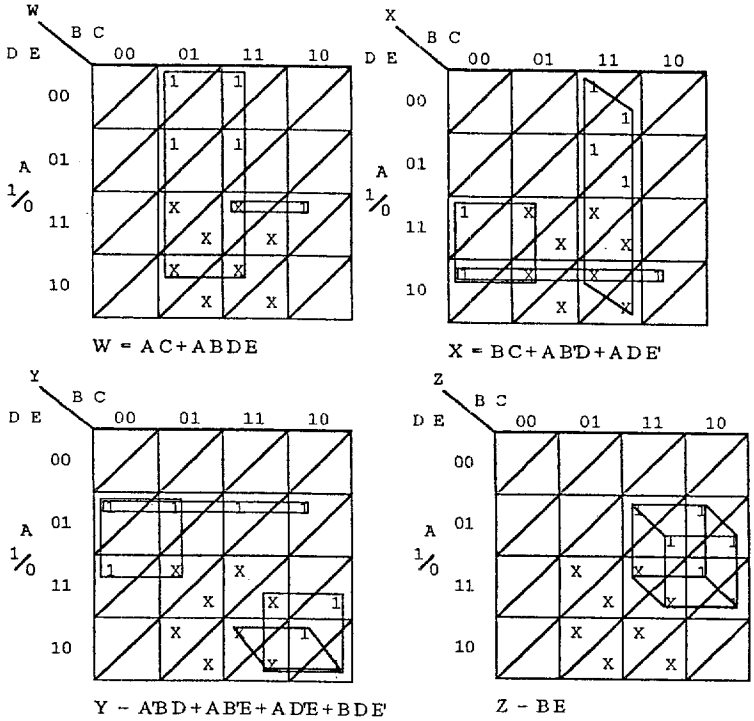
8.D

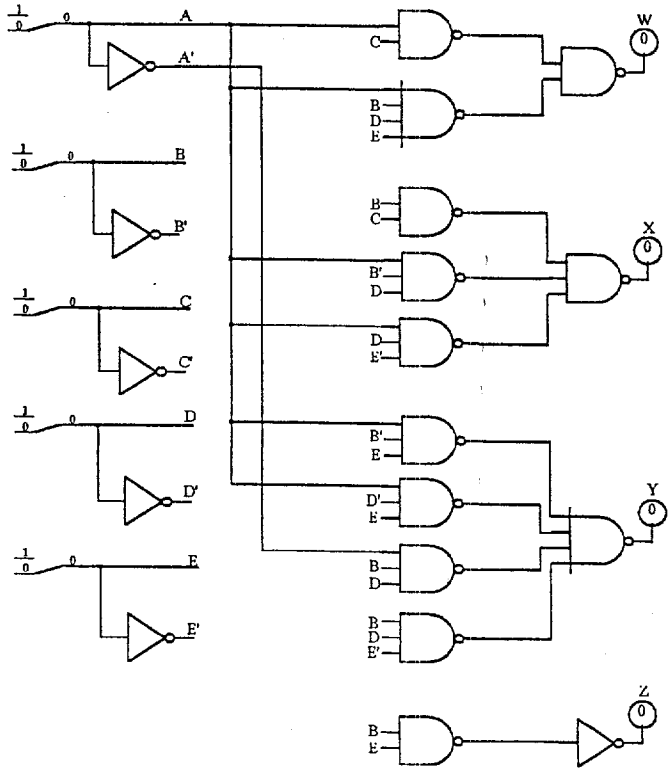
ABCDE	W X Y Z
00000	0 0 0 0
00001	0 0 0 0
00010	0 0 0 0
00011	0 0 0 0
00100	0 0 0 0
00101	0 0 0 0
00110	X X X X
00111	X X X X
01000	0 0 0 0
01001	0 0 0 1
01010	0 0 1 0
01011	0 0 1 1
01100	0 1 0 0
01101	0 1 0 1
01110	X X X X
01111	X X X X
10000	0 0 0 0
10001	0 0 1 0
10010	0 1 0 0
10011	0 1 1 0
10100	1 0 0 0
10101	1 0 1 0
10110	X X X X
10111	X X X X
11000	0 0 0 0
11001	0 0 1 1
11010	0 1 1 0
11011	1 0 0 1
11100	1 1 0 0
11101	1 1 1 1

$W = AC + ABDE$
$X = BC + AB'D + ADE'$
$Y = A'BD + AB'E + AD'E + BDE'$
$Z = BE$

This solution uses 14 gates and 38 gate inputs.

Students are allowed to use a maximum of 14 gates.

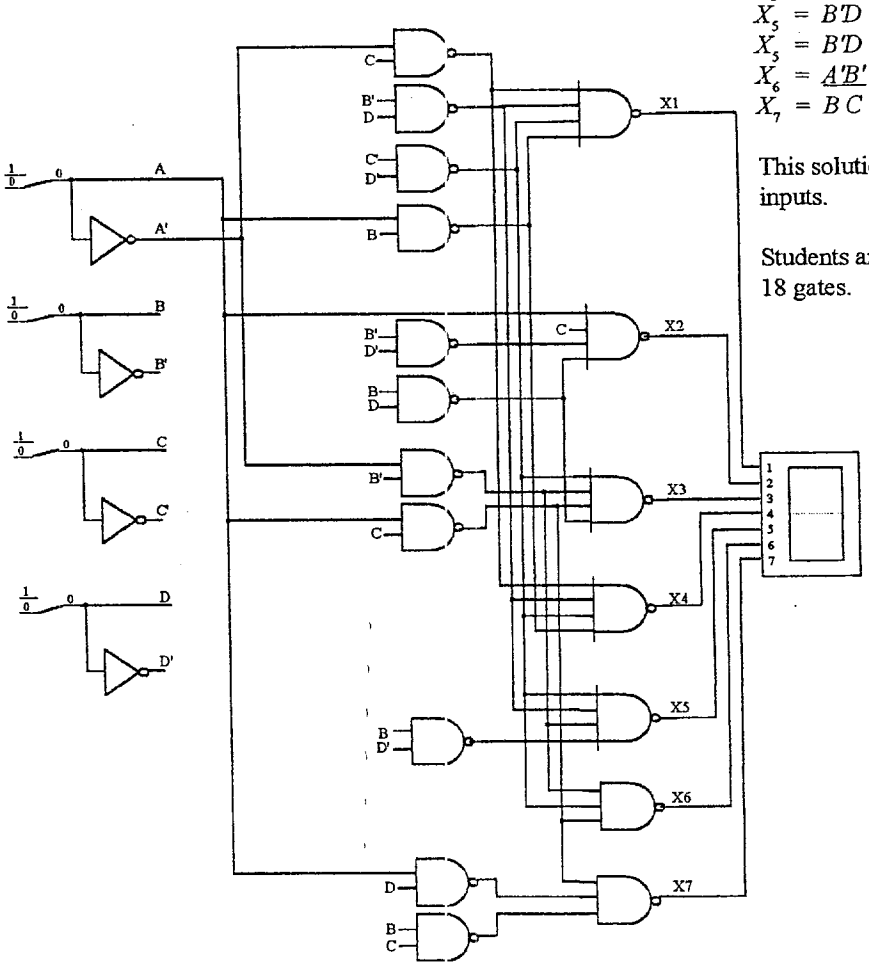




8.E

$ABCD$	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0000	1	1	1	1	1	1	0
0001	1	1	1	1	1	1	0
0010	1	1	1	1	1	1	0
0011	1	1	1	1	1	1	0
0100	1	1	1	1	1	1	0
0101	0	1	1	0	0	0	0
0110	1	1	0	1	1	0	1
0111	1	1	1	1	0	0	1
1000	1	1	1	1	1	1	0
1001	1	1	0	1	1	0	1
1010	0	1	1	0	0	1	1
1011	1	0	1	1	1	1	1
1100	1	1	1	1	1	1	0
1101	1	1	1	1	0	0	1
1110	1	0	1	1	1	1	1
1111	1	1	1	1	0	1	1

$X_1 = \underline{AB} + \underline{A'C} + \underline{B'D} +$
$\underline{C'D'}$ (used in circuit)
$X_1 = AC' + BC + B'D + A'D'$
$X_1 = AD + BC + C'D' + A'B'$
$X_1 = AC' + BD' + CD + A'B'$
$X_1 = AB + CD + A'D' + B'C'$
$X_1 = AD + BD' + A'C + B'C'$
$X_2 = A' + C' + B'D' + \underline{BD}$
$X_3 = \underline{AC} + \underline{BD} + \underline{C'D'} +$
$\underline{A'B'}$ (used in circuit)
$X_3 = AB + B'C + A'D + C'D'$
$X_3 = AB + CD + A'C' + B'D'$
$X_3 = AD' + BD + B'C + A'C'$
$X_3 = AD' + BC' + CD + A'B'$
$X_3 = AC + BC' + A'D + B'D'$
$X_4 = \underline{AB} + \underline{A'C} + \underline{B'D} +$
$\underline{C'D'}$ (used in circuit)
$X_4 = X_1$
$X_5 = \underline{B'D} + BD' + \underline{C'D'} +$
$\underline{A'B'}$ (used in circuit)
$X_5 = B'D + BD' + C'D' + A'D'$
$X_5 = B'D + BD' + A'D' + B'C'$
$X_5 = B'D + BD' + A'B' + B'C'$
$X_6 = \underline{A'B'} + \underline{C'D'} + \underline{AC}$
$X_7 = BC + AD + AC$



This solution uses 17 gates and 44 gate inputs.

Students are allowed to use a <u>maximum</u> of 18 gates.

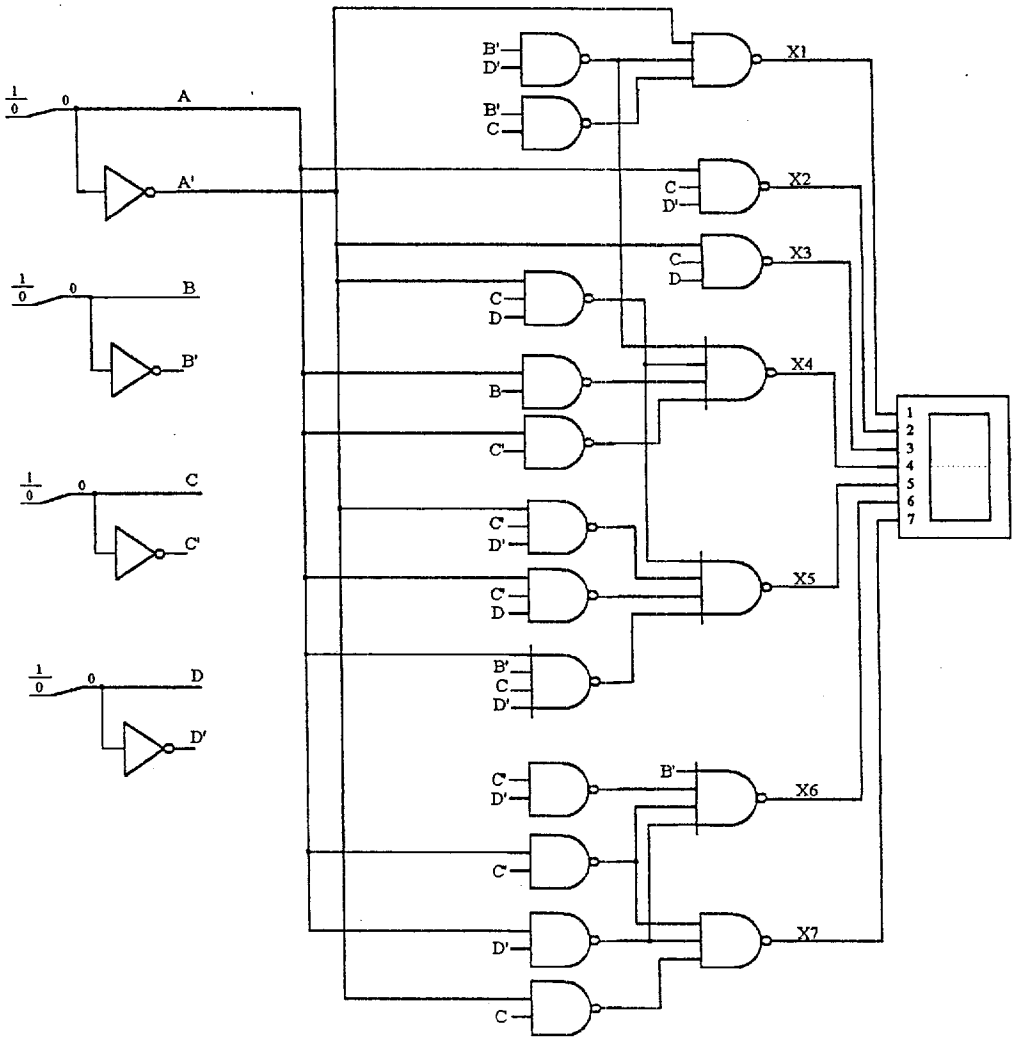
8.F

ABCD	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	1	1	0	0	1
0011	1	1	0	1	1	0	1
0100	X	X	X	X	X	X	X
0101	X	X	X	X	X	X	X
0110	0	1	1	0	0	1	1
0111	X	X	X	X	X	X	X
1000	1	1	1	1	0	1	1
1001	1	1	1	1	1	1	1
1010	1	0	1	1	1	1	1
1011	1	1	1	0	0	0	0
1100	X	X	X	X	X	X	X
1101	X	X	X	X	X	X	X
1110	1	0	1	1	0	1	1
1111	X	X	X	X	X	X	X

$X_1 = A + B'C + \underline{B'D'}$ (used in circuit)
$X_1 = A + B'C + C'D'$
$X_1 = A + B'D' + C\,D$
$X_2 = A' + C' + D$
$X_3 = C' + D' + A$
$X_4 = \underline{A\,C'} + \underline{B'D'} + A\,B + \underline{A'C\,D}$ (used in circuit)
$X_4 = A\,C' + A'B'C + A\,D' + C'D'$
$X_4 = A\,C' + A'B'C + B'D' + A\,D'$
$X_4 = A\,C' + A'B'C + B'D' + A\,B$
$X_4 = A\,C' + B'D' + A\,D' + A'C\,D$
$X_5 = A'C'D' + \underline{A'C\,D} + A\,C'D + A\,B'C\,D'$
$X_6 = C'D' + B + \underline{A\,C'} + \underline{A\,D'}$
$X_7 = A'C + \underline{A\,C'} + \underline{A\,D'}$ (used in circuit)
$X_7 = A'C + A\,C' + C\,D'$

This solution uses 18 gates and 51 gate inputs.

Students are allowed to use a maximum of 20 gates.



8.G

K N3 N2 N1 N0	M3 M2 M1 M0
0 0 0 0 0	0 0 0 1
0 0 0 0 1	0 0 1 0
0 0 0 1 0	0 0 1 1
0 0 0 1 1	0 1 0 0
0 0 1 0 0	0 1 0 1
0 0 1 0 1	0 1 1 0
0 0 1 1 0	0 1 1 1
0 0 1 1 1	1 0 0 0
0 1 0 0 0	1 0 0 1
0 1 0 0 1	1 0 1 0
0 1 0 1 0	1 0 1 1
0 1 0 1 1	1 1 0 0
0 1 1 0 0	1 1 0 1
0 1 1 0 1	1 1 1 0
0 1 1 1 0	1 1 1 1
0 1 1 1 1	X X X X
1 0 0 0 0	0 0 1 0
1 0 0 0 1	0 0 1 1
1 0 0 1 0	0 1 0 0
1 0 0 1 1	0 1 0 1
1 0 1 0 0	0 1 1 0
1 0 1 0 1	0 1 1 1
1 0 1 1 0	1 0 0 0
1 0 1 1 1	1 0 0 1
1 1 0 0 0	1 0 1 0
1 1 0 0 1	1 0 1 1
1 1 0 1 0	1 1 0 0
1 1 0 1 1	1 1 0 1
1 1 1 0 0	1 1 1 0
1 1 1 0 1	1 1 1 1
1 1 1 1 0	X X X X
1 1 1 1 1	X X X X

$M3 = N2\,N1\,N0 + N3 + K\,N2\,N1 = N3 + N2\,N1(\underline{K + N0})$

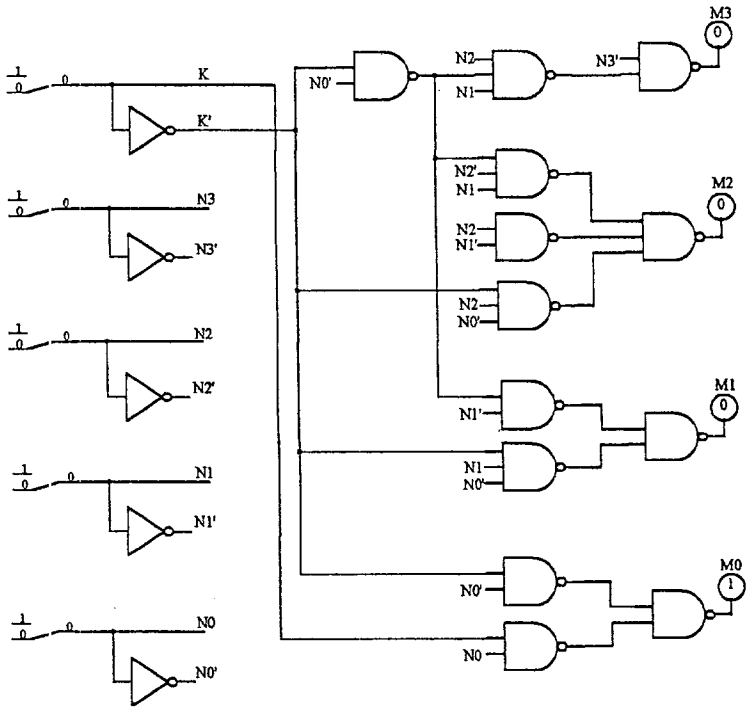
$M2 = N2\,N1' + K'N2\,N0' + N2'N1\,N0 + K\,N2'\,N1 =$
$N2\,N1' + K'\,N2\,N0' + N2'\,N1(\underline{K + N0})$

$M1 = K'N1\,N0' + K\,N1' + N1'N0 =$
$K'\,N1\,N0' + N1'(\underline{K + N0})$

$M0 = K'\,N0' + K\,N0$

This solution uses 13 gates and 31 gate inputs.

Students are allowed to use a maximum of 13 gates.



8.H

ABCD	X_1 X_2 X_3 X_4 X_5 X_6 X_7
0000	1 1 1 1 1 1 0
0001	1 1 1 1 1 1 1
0010	0 1 1 0 0 0 0
0011	1 1 1 1 0 1 1
0100	1 1 0 1 1 0 1
0101	X X X X X X X
0110	1 1 1 1 0 0 1
0111	X X X X X X X
1000	0 1 1 0 0 1 1
1001	X X X X X X X
1010	1 0 1 1 0 1 1
1011	X X X X X X X
1100	1 0 1 1 1 1 1
1101	X X X X X X X
1110	1 1 1 0 0 0 0
1111	X X X X X X X

$X_1 = \underline{A'C'} + D + B + A\,C$

$X_2 = A' + \underline{B'C'} + B\,C$

$X_3 = B' + C + A$

$X_4 = X_5 + D + \underline{A'B} + A\,B'C$

$X_5 = \underline{A'C'} + \underline{B\,C'}$

$X_6 = \underline{B'C'} + D + \underline{A\,B'} + A\,C'$

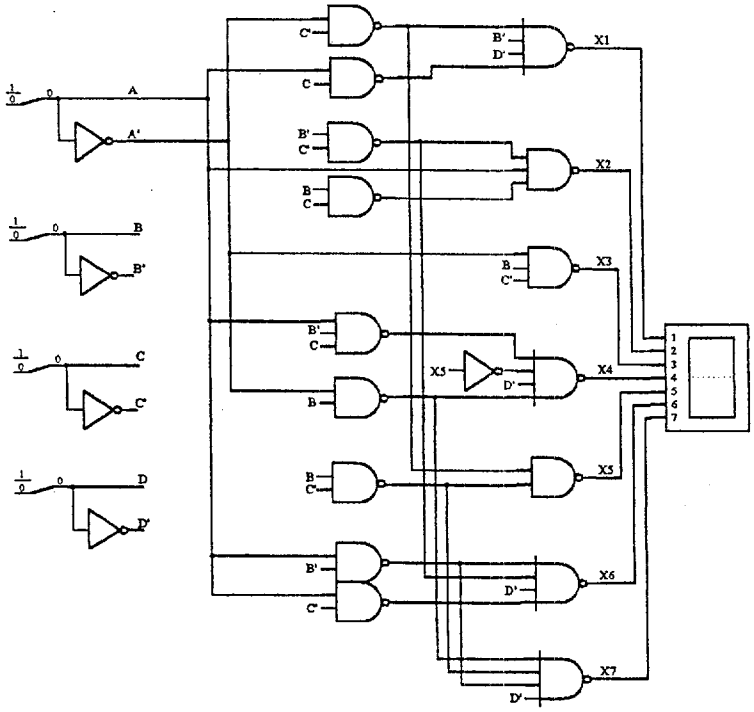
$X_7 = D + \underline{A'B} + \underline{A\,B'} + \underline{B\,C'}$ (used in circuit)

$X_7 = D + A'B + A\,B' + A\,C'$

This solution uses 17 gates and 45 gate inputs.

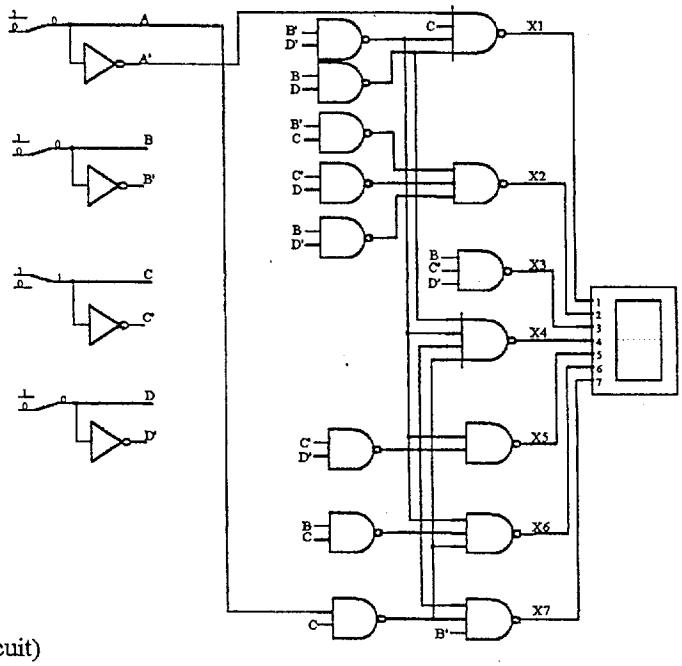
Students are allowed to use a maximum of 20 gates.

8.H, continued



8.I

$ABCD$	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0000	X	X	X	X	X	X	X
0001	X	X	X	X	X	X	X
0010	1	1	1	1	1	1	0
0011	0	1	1	0	0	0	0
0100	1	1	0	1	1	0	1
0101	1	1	1	1	0	0	1
0110	0	1	1	0	0	1	1
0111	1	0	1	1	0	1	1
1000	1	0	1	1	1	1	1
1001	1	1	1	0	0	0	0
1010	1	1	1	1	1	1	1
1011	1	1	1	1	0	1	1



$X_1 = \underline{B'D'} + C' + \underline{BD} + A$

$X_2 = B'C + C'D + BD'$ (used in circuit)

$X_2 = B'C + A'D' + C'D$

$X_2 = B'D + CD' + BC'$

$X_2 = B'D + A'C' + CD'$

$X_3 = D + C + B'$ (used in circuit)

$X_3 = D + C + A$

$X_4 = \underline{B'D'} + \underline{BD} + \underline{AC} + \underline{C'D'}$ (used in circuit)

$X_4 = B'D' + BD + AC + BC'$

$X_4 = B'D' + BD + AC + A'C'$

$X_5 = \underline{B'D'} + \underline{C'D'}$

$X_6 = BC + \underline{AC} + \underline{B'D'}$

$X_7 = B + \underline{AC} + \underline{C'D'}$ (used in circuit)

$X_7 = B + AC + AD'$

This solution uses 15 gates and 38 gate inputs.

Students are allowed to use a maximum of 17 gates.

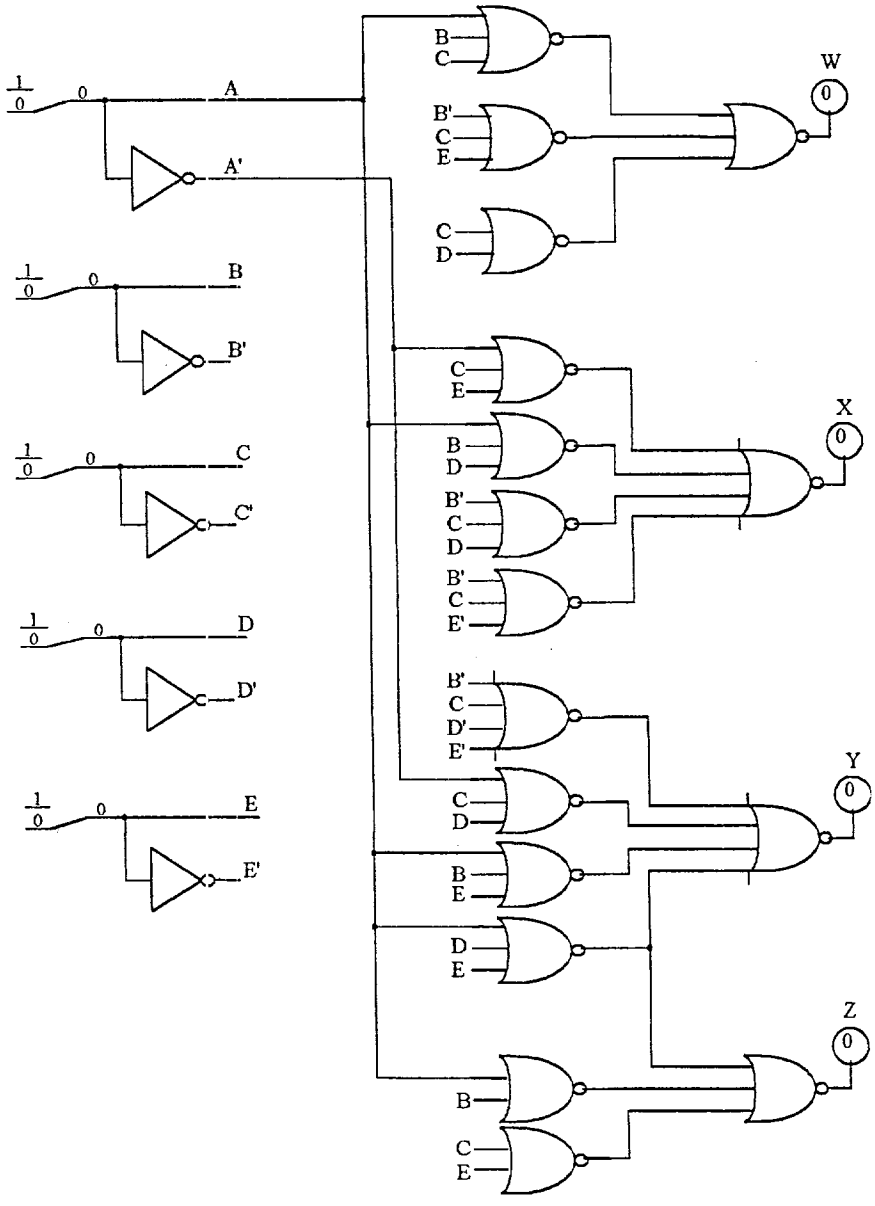
8.J

ABCDE	*W*	*X*	*Y*	*Z*
00000	0	0	0	0
00001	0	0	1	0
00010	0	1	0	0
00011	0	1	1	0
00100	1	0	0	0
00101	1	0	1	0
00110	1	1	0	0
00111	1	1	1	0
01000	0	0	0	0
01001	0	0	1	1
01010	0	1	1	0
01011	1	0	0	1
01100	1	1	0	0
01101	1	1	1	1
01110	1	1	1	1
01111	1	1	1	1
10000	0	0	0	0
10001	0	1	0	1
10010	1	0	1	0
10011	1	1	1	1
10100	1	1	1	1
10101	1	1	1	1
10110	1	1	1	1
10111	1	1	1	1

$W = (A + B + C)\ (C + D)\ (B' + C + E)$ (used in circuit)
$W = (A + B + C)\ (C + D)\ (A + C + E)$
$X = (A + B + D)\ (E' + C + E')\ (A' + C + E)\ (B' + C + D)$ (used in circuit)
$X = (A + B + D)\ (E' + C + E')\ (A' + C + E)\ (C + D + E)$
$X = (A + B + D)\ (E' + C + E')\ (A' + C + E)\ (A + C + D)$
$Y = (A + B + E)\ (B' + C + D' + E')\ (A' + C + D)\ \underline{(A + D + E)}$ (used in circuit)
$Y = (A + B + E)\ (B' + C + D' + E')\ (A' + C + D)\ (B' + D + E)$
$Z = (A + B)\ (C + E')\ \underline{(A + D + E)}$ (used in circuit)
$Z = (A + B)\ (C + E')\ (B' + D + E)$

This solution uses 17 gates and 51 gate inputs.

Students are allowed to use a <u>maximum</u> of 19 gates.



8.K

ABCDE	W	X	Y	Z
00000	0	0	0	0
00001	0	0	0	0
00010	0	0	0	0
00011	0	0	0	1
00100	0	0	0	1
00101	0	0	0	1
00110	0	0	1	0
00111	0	0	1	0
01000	0	0	1	0
01001	0	0	1	1
01010	0	0	1	1
01011	0	0	1	1
01100	0	1	0	0
01101	0	1	0	0
01110	0	1	0	0
01111	0	1	0	1
10000	0	1	0	1
10001	0	1	0	1
10010	0	1	1	0
10011	0	1	1	0
10100	0	1	1	0
10101	0	1	1	1
10110	0	1	1	1
10111	0	1	1	1
11000	1	0	0	0
11001	1	0	0	0
11010	1	0	0	0
11011	1	0	0	1

$W = A\,B$

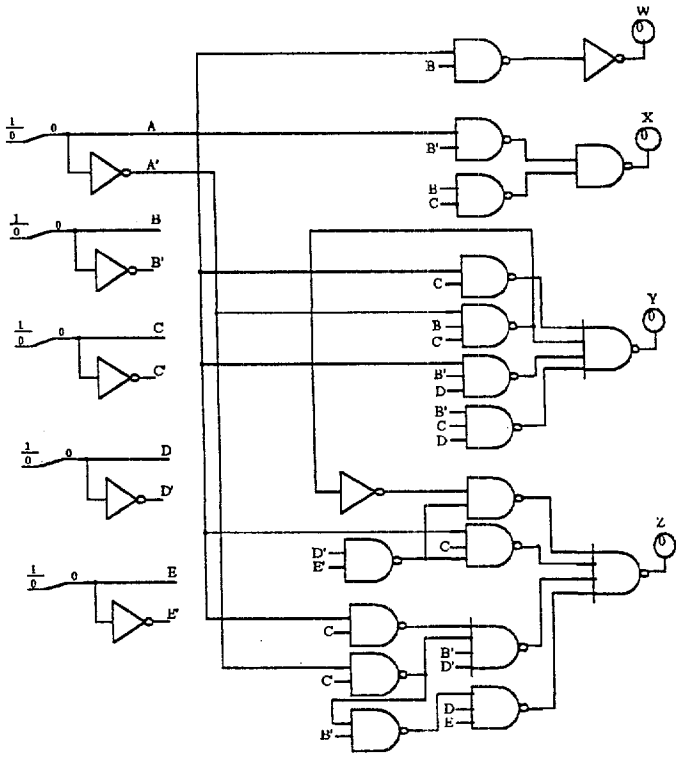
$X = B\,C + A\,B'$

$Y = B'C\,D + \underline{A'B\,C'} + A\,B'D + A\,C$

$Z = A'C'D\,E + B\,D\,E + A'B'C\,D' + A\,B'C'D' + A'B\,C'D + A'B\,C'E +$
$A\,C\,D + A\,C\,E = DE(B + A'C') + B'D'(A'C + AC') +$
$A'BC'(D + E) + AC(D + E) =$
$DE(B + A'C') + B'D'(A + C)(A' + C') + \underline{A'BC'\,(D + E)} + AC\underline{(D + E)}$

This solution uses 19 gates and 47 gate inputs.

Students are allowed to use a <u>maximum</u> of 22 gates.



8.L

ABCD	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0000	1	0	0	1	1	1	1
0001	1	0	0	1	1	1	1
0010	1	0	0	1	1	1	1
0011	1	1	1	1	1	1	0
0100	0	1	1	0	0	0	0
0101	1	1	0	1	1	0	1
0110	1	1	1	1	0	0	1
0111	0	1	1	0	0	1	1
1000	1	0	1	1	0	1	1
1001	1	0	1	1	1	1	1
1010	1	1	1	0	0	0	0
1011	1	1	1	1	1	1	1
1100	1	1	1	1	0	1	1

$X_1 = \underline{(A + B' + C + D)\,(B' + C' + D')}$

$X_2 = (B + C)\,\underline{(A + B + D)}$

$X_3 = \underline{(A + B + D)}\,(A + C + D')$

$X_4 = \underline{(A + B' + C + D)\,(B' + C' + D')\,(A' + C' + D)}$

$X_5 = (B' + D)\,(B' + C')\,(A' + D) = (B' + D)\,\underline{(B' + C' + D')}\,(A' + D)$

$X_6 = \underline{(A' + C' + D)}\,(A + B' + D)\,(A + B' + C)$ (used in circuit)

$X_6 = (A' + C' + D)\,(A + B' + C)\,(B' + C' + D)$

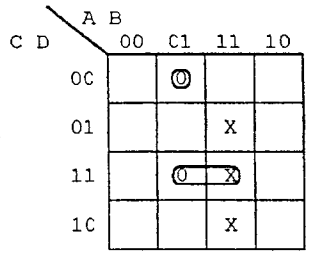
$X_6 = (A' + C' + D)\,(A + B' + D)\,(B' + C + D')$

$X_7 = (A + B + C' + D')\,\underline{(A + B' + C + D)}\,\underline{(A' + C' + D)}$

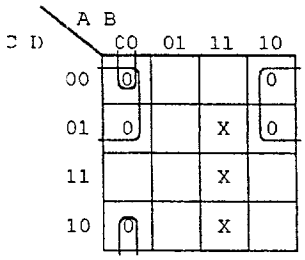
This solution uses 18 gates and 50 gate inputs.

Students are allowed to use a <u>maximum</u> of 18 gates.

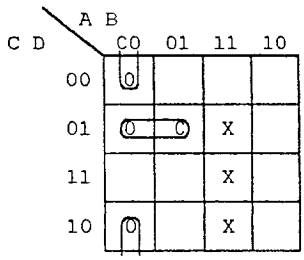
8.L, continued



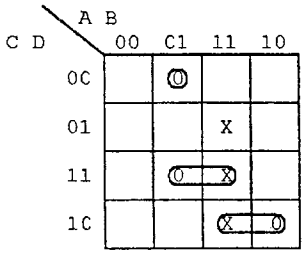
$X_1 = (A + B' + C + D)(B' + C' + D')$



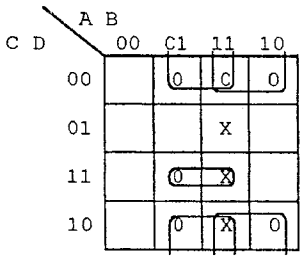
$X_2 = (B + C)(A + B + D)$



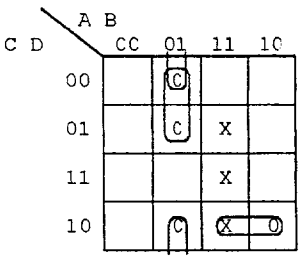
$X_3 = (A + B + D)(A + C + D')$



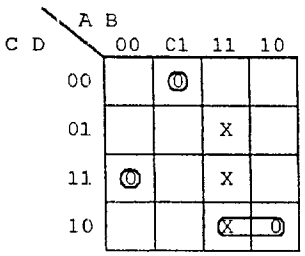
$X_4 = (A' + C' + D)(B' + C' + D')(A + B' + C + D)$



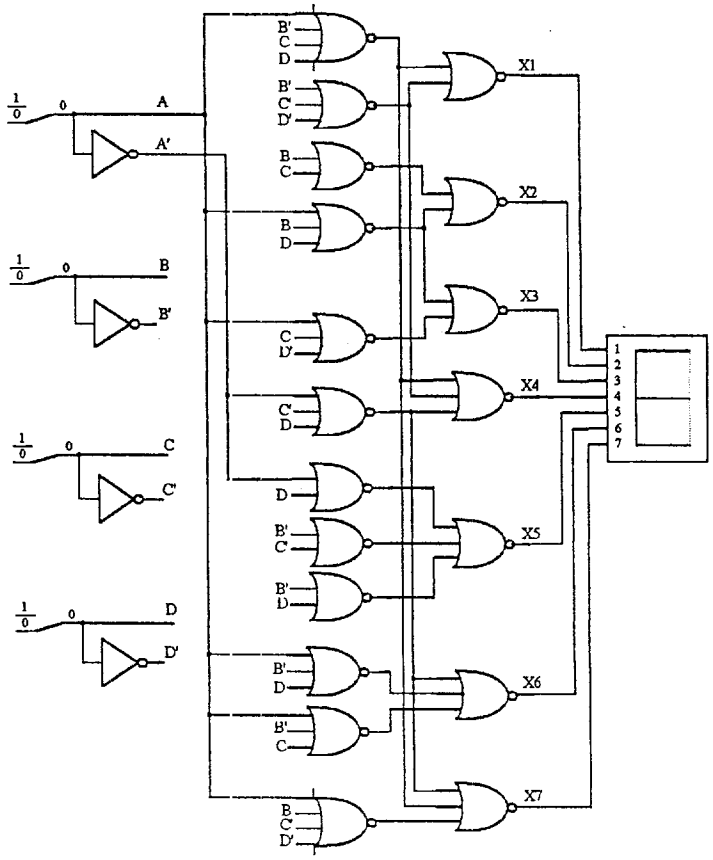
$X_5 = (B' + D)(B' + C' + D')(A' + D)$



$X_6 = (A' + C' + D)(A + B' + C)(A + B' + D)$



$X_7 = (A' + C' + D)(A + B' + C + D)(A + B + C' + D')$



8.M

$ABCD$	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0000	X	X	X	X	X	X	X
0001	1	1	1	0	1	1	1
0010	0	0	1	1	1	1	1
0011	1	0	0	1	1	1	0
0100	0	1	1	1	1	0	1
0101	1	0	0	1	1	1	1
0110	1	0	0	0	1	1	1
0111	1	1	1	1	0	1	1
1000	0	1	1	0	1	1	1
1001	0	0	0	0	1	1	0
1010	0	1	1	1	1	0	0

$X_1 = W'(X + Z)(Y + Z)$

$X_2 = \underline{(X' + Y + Z')}\ \underline{(W' + Z')}\ (W + Y' + Z)\ (W + X + Y')$ (used in circuit)

$X_2 = (X' + Y + Z')\ (W' + Z')\ (W + X + Y')\ (X' + Y' + Z)$

$X_2 = (X' + Y + Z')\ (W' + Z')\ (W + Y' + Z)\ (X + Y' + Z')$

$X_3 = \underline{(X + Y' + Z')}\ \underline{(X' + Y + Z')}\ (X' + Y' + Z)\ \underline{(W' + Z')}$

$X_4 = (X + Y)\ \underline{(X' + Y' + Z)}$

$X_5 = (X' + Y' + Z')$

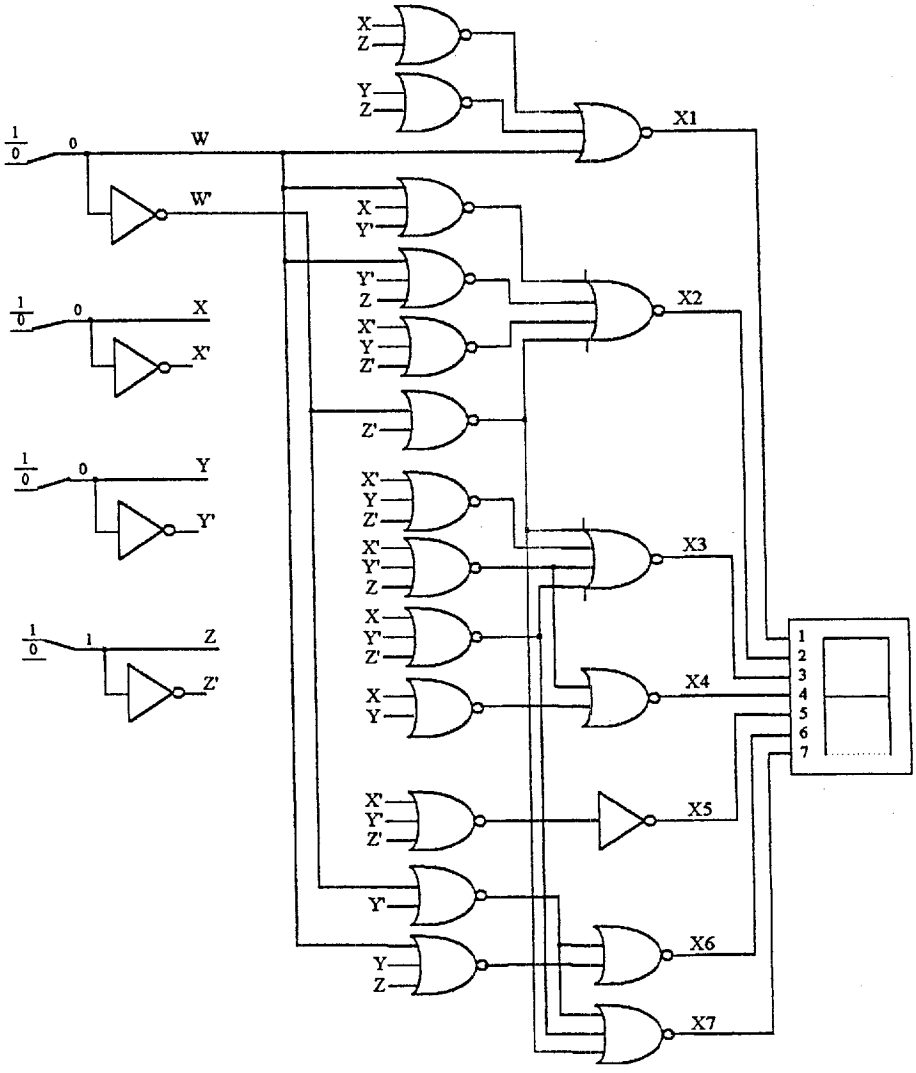
$X_6 = \underline{(W' + Y')}\ (W + Y + Z)$ (used in circuit)

$X_6 = (W' + Y')\ (X' + Y + Z)$

$X_7 = \underline{(X + Y' + Z')}\ \underline{(W' + Z')}\ \underline{(W' + Y')}$

This solution uses 19 gates and 50 gate inputs.

Students are allowed to use a maximum of 22 gates.



8.N

ABCDE	*X*	*Y*	*Z*
00000	0	0	0
00001	0	0	0
00010	0	0	0
00011	0	0	0
00100	0	0	0
00101	0	0	0
00110	0	0	0
00111	0	0	0
01000	0	0	1
01001	0	0	1
01010	0	0	1
01011	0	0	1
01100	0	0	1
01101	1	0	0
01110	1	0	0
01111	0	0	1
10000	0	0	1
10001	0	0	1
10010	0	0	1
10011	0	0	1
10100	0	0	1
10101	0	0	1
10110	0	0	1
10111	0	0	1
11000	0	0	1
11001	0	0	1
11010	0	0	1
11011	0	1	0
11100	0	0	1
11101	0	1	0
11110	0	0	1
11111	0	0	1

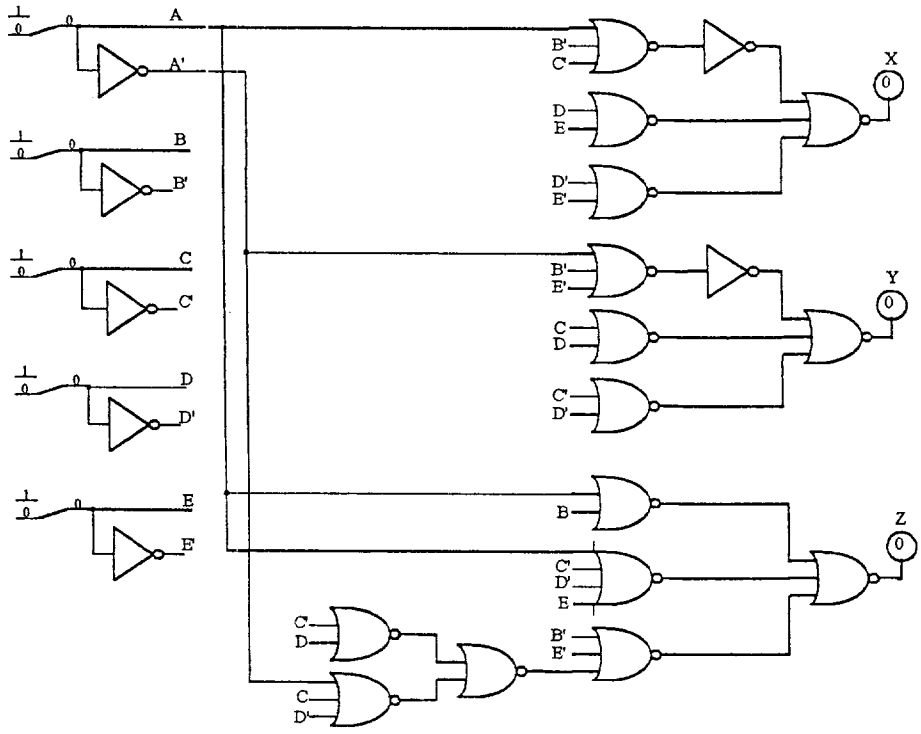
$X = A'BC(D + E)\ (D' + E')$

$Y = ABE(C + D)\ (C' + D')$

$Z = (A + B)\ (A + C' + D' + E)\ (A' + B' + C + D' + E')\ (B' + C' + D + E') =$
$(A + B)\ (A + C' + D' + E)\ (B' + E')\ (A' + C + D')\ (C' + D)$

This solution uses 17 gates and 41 gate inputs.

Students are allowed to use a maximum of 19 gates.



8.O

ABCD	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	0	1	1	0	1
0011	1	1	1	1	0	0	1
0100	0	1	1	0	0	1	1
0101	1	0	1	1	0	1	1
0110	0	0	1	1	1	1	1
0111	1	1	1	0	0	0	0
1000	1	1	1	1	1	1	1
1001	1	1	1	0	0	1	1

$X_1 = (A + B + C + D')\ (B' + D) = (A + B + C + D')\ \underline{(B' + C' + D)\ (B' + C + D)}$

$X_2 = (B' -- C + D')\ \underline{(B' + C' + D)}$

$X_3 = (B + C' + D)$

$X_4 = (B + C + D')\ \underline{(B' + C + D)\ (B' + C' + D')}$

$X_5 = D'(B' + C) = D'\ \underline{(B' + C + D)}$

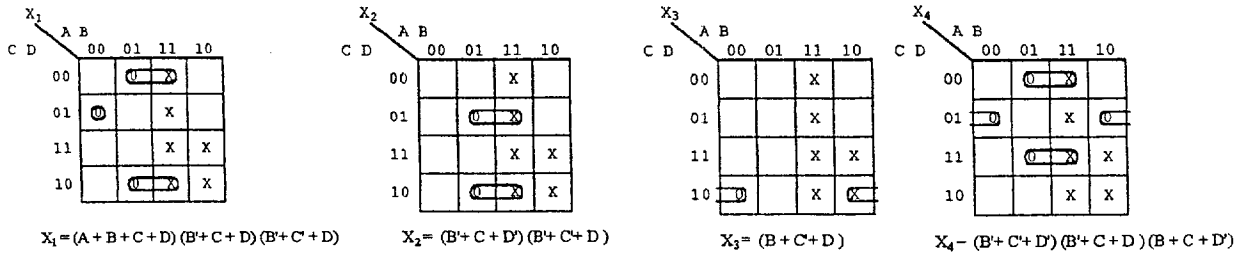
$X_6 = (A + B + D')\ (B + C')\ (C' + D')$

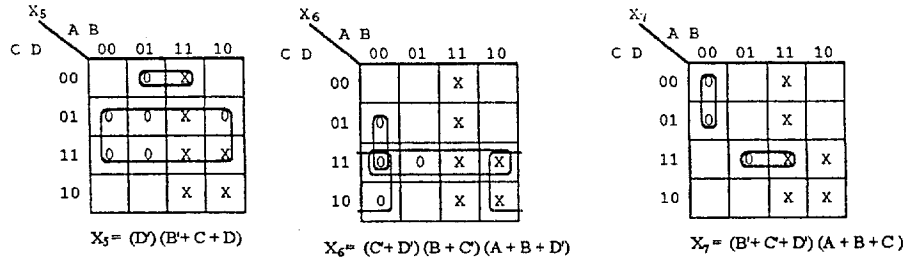
$X_7 = (A + B + C)\ \underline{(B' + C' + D')}$

This solution uses 18 gates and 48 gate inputs.

Students are allowed to use a maximum of 19 gates.

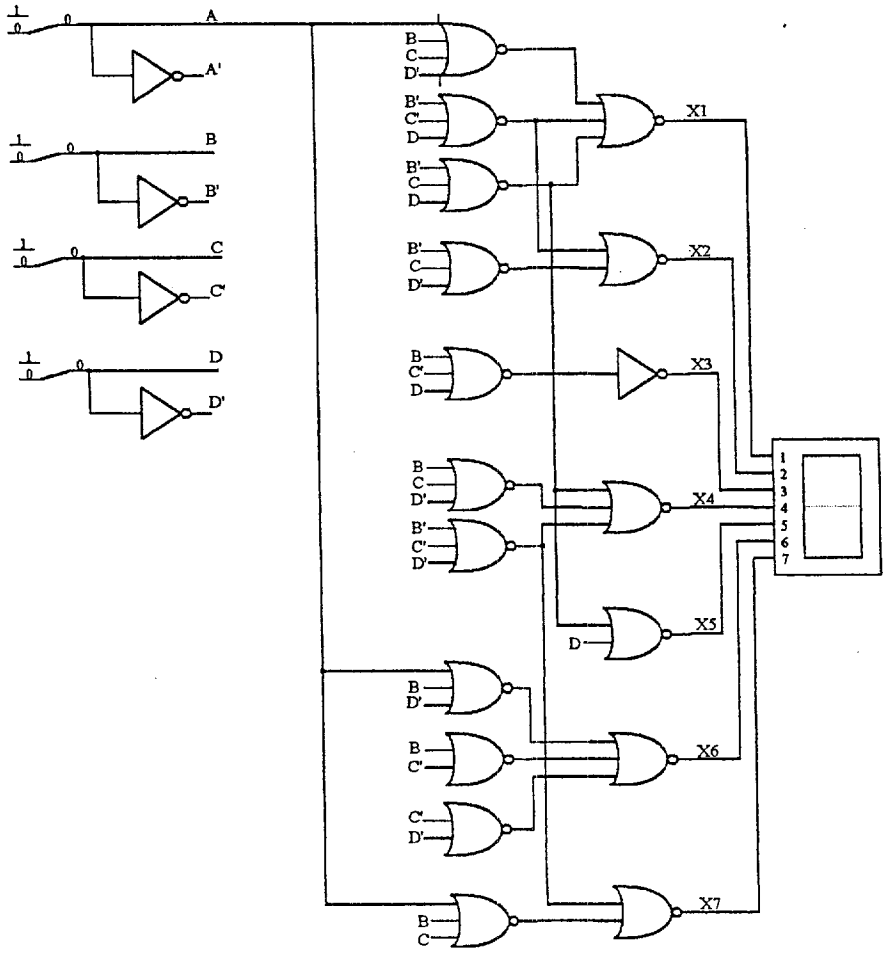
8.O, continued





To save one gate, use:

$X_6 = (B' + C' + D')(B + C')(A + B + D')$



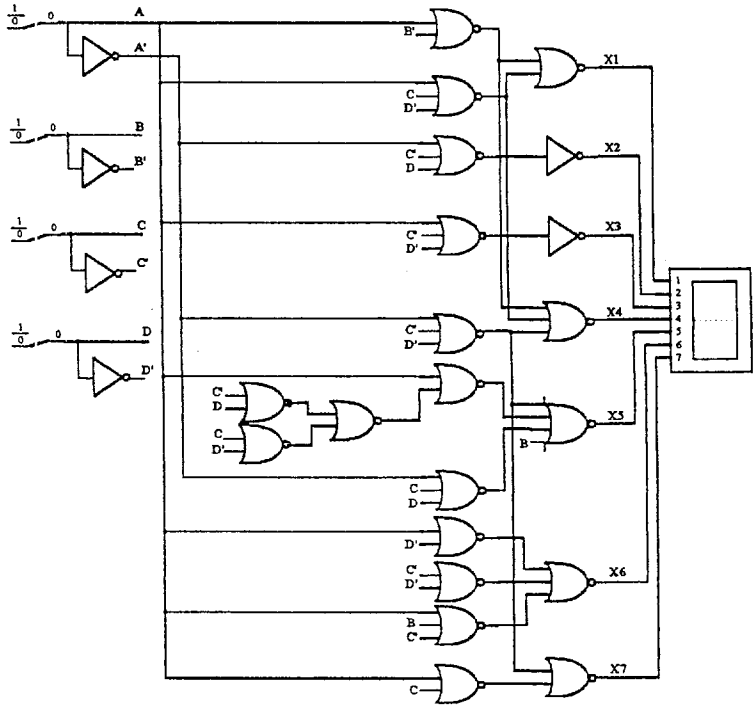
8.P

ABCD	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0000	1	1	1	1	1	1	0
0001	0	1	1	0	0	0	0
0010	1	1	1	1	0	0	1
0011	1	1	0	1	1	0	1
0100	X	X	X	X	X	X	X
0101	X	X	X	X	X	X	X
0110	0	1	1	0	0	1	1
0111	X	X	X	X	X	X	X
1000	1	1	1	1	0	1	1
1001	1	1	1	1	1	1	1
1010	1	0	1	1	1	1	1
1011	1	1	1	0	0	0	0
1100	X	X	X	X	X	X	X
1101	X	X	X	X	X	X	X
1110	1	0	1	1	0	1	1
1111	X	X	X	X	X	X	X

$X_1 = \underline{(A + C + D')\ (A + B')}$
$X_2 = (A' + C' + D)$
$X_3 = (A + C' + D')$
$X_4 = \underline{(A + C + D')\ (A + B')\ (A' + C' + D')}$
$X_5 = B'(A + C + D')\ (A + C' + D)\ (A' + C + D)\ (A' + C' + D') =$
$B'\ (A + (C + D')\ (C' + D))\ (A' + C + D)\ \underline{(A' + C' + D')}$
$X_6 = (A + D')\ (A + B + C')\ (C' + D')$
$X_7 = (A + C)\ \underline{(A' + C' + D')}$

This solution uses 21 gates and 50 gate inputs.

Students are allowed to use a maximum of 21 gates.



8.Q

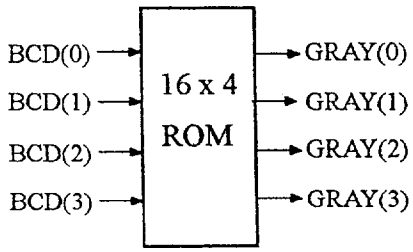
ABCD	X_1	X_2	X_3	X_4	X_5	X_6	X_7
0000	1	1	1	1	1	1	0
0001	1	1	1	1	1	1	1
0010	0	1	1	0	0	0	0
0011	1	1	1	1	0	1	1
0100	1	1	0	1	1	0	1
0101	X	X	X	X	X	X	X
0110	1	1	1	1	0	0	1
0111	X	X	X	X	X	X	X
1000	0	1	1	0	0	1	1
1001	X	X	X	X	X	X	X
1010	1	0	1	1	0	1	1
1011	X	X	X	X	X	X	X
1100	1	0	1	1	1	1	1
1101	X	X	X	X	X	X	X
1110	1	1	1	0	0	0	0
1111	X	X	X	X	X	X	X

$X_1 = (A + B + C' + D)\ \underline{(A' + B + C)}$
$X_2 = (A' + B + C')\ (A' + B' + C)$
$X_3 = \underline{(A + B' + C)}$
$X_4 = (A + B + C' + D)\ \underline{(A' + B + C)}\ \underline{(A' + B' + C')}$
$X_5 = C'(A' + B) = C'\underline{(A' + B + C)}$
$X_6 = (A + C' + D)\ \underline{(A + B' + C)}\ \underline{(A' + B' + C')}$
$X_7 = (A + B + D)\ \underline{(A' + B' + C')}$

This solution uses 15 gates and 40 gate inputs.

Students are allowed to use a maximum of 17 gates.

10.B



Test Sequence: BCD = 0010, 0101, 1001.

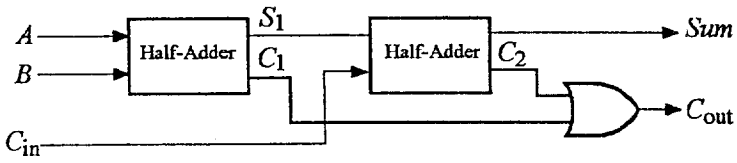
Command Sequence:
force bcd 0010
run 5ns
force bcd 0101
run 5ns
force bcd 1001
run 5ns

Output (from DirectVHDL):

```
Time  bcd  gray index
0 ns  0010 0011 2
5 ns  0101 1110 5
10 ns 1001 1000 9
```

```
library bitlib;
use bitlib.bit_pack.all;

entity rom is
    port (bcd : in bit_vector(3 downto 0);
    gray : out bit_vector(3 downto 0));
end rom;
architecture eqn of rom is
    type rom16_4 is array (0 to 15) of bit_vector (3 downto 0);
    constant rom1 : rom16_4 := ("0000", "0001", "0011", "0010",
    "0110", "1110", "1010", "1011", "1001", "1000", others =>
    "1111");
    signal index : integer range 0 to 15;
    begin
    index <= vec2int(bcd);
    gray <= rom1(index);
end eqn;
```

10.C Note: See solution to Problem 4.33 for derivation of the half-adder



Test Sequence: a b cin = 0 0 1, 0 1 1, 1 1 1, 1 1 0, 1 0 0.

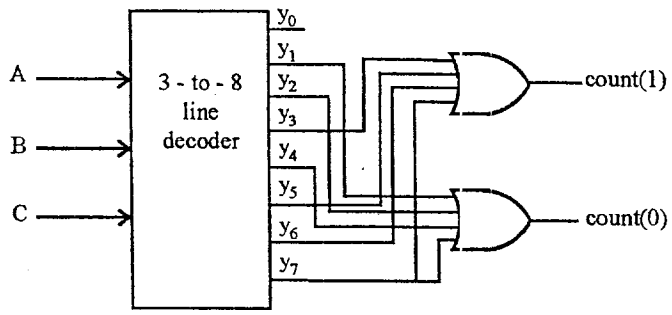
Command Sequence:
force a 0
force b 0
force cin 1
run 5ns
force b 1
run 5ns
force a 1
run 5ns
force cin 0
run 5ns
force b 0
run 5ns

Output (from DirectVHDL):

```
Time  a   b   cin sum cout c2  c1  s1
0 ns  '0' '0' '1' '1' '0'  '0' '0' '0'
5 ns  '0' '1' '1' '0' '1'  '1' '0' '1'
10 ns '1' '1' '1' '1' '1'  '0' '1' '0'
15 ns '1' '1' '0' '0' '1'  '0' '1' '0'
20 ns '1' '0' '0' '1' '0'  '0' '0' '1'
```

```
-- Code for the half adder
entity ha is
    port (a, b : in bit;
    s, c : out bit);
end ha;
architecture eqn of ha is
    begin
    s <= a xor b;
    c <= a and b;
end eqn;
-- Code for the full adder
entity fa is
    port (a, b, cin : in bit;
    sum, cout : out bit);
end fa;
architecture eqn of fa is
    component ha is
        port (a, b : in bit;
        s, c : out bit);
    end component;
    signal s1, c1, c2 : bit;
    begin
    ha1 : ha port map(a, b, s1, c1);
    ha2 : ha port map(s1, cin, sum, c2);
    cout <= c1 or c2;
end eqn;
```

10.D



Test Sequence: a b c = 0 0 0, 0 1 0, 1 1 0, 1 1 1, 0 1 1.

Command Sequence:

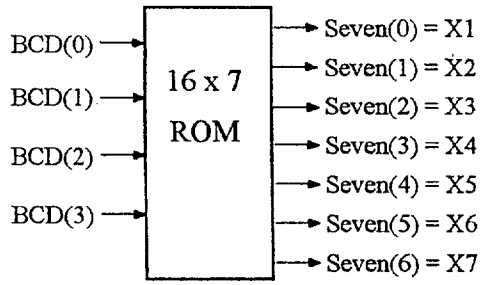
```
force a 0
force b 0
force c 0
run 5ns
force b 1
run 5ns
force a 1
run 5ns
force c 1
run 5ns
force a 0
run 5ns
```

```
-- Code for the 3 to 8 decoder
entity decoder is
    port (a, b, c : in bit;
    y0, y1, y2, y3, y4, y5, y6, y7 : out bit);
end decoder;
architecture eqn of decoder is
    begin
    y0 <= not a and not b and not c;
    y1 <= not a and not b and c;
    y2 <= not a and b and not c;
    y3 <= not a and b and c;
    y4 <= a and not b and not c;
    y5 <= a and not b and c;
    y6 <= a and b and not c;
    y7 <= a and b and c;
end eqn;
-- Code for the main module
entity fa is
    port (a, b, c : in bit;
    count : out bit_vector(1 downto 0));
end fa;
architecture eqn of fa is
    component decoder is
        port (a, b, c : in bit;
        y0, y1, y2, y3, y4, y5, y6, y7 : out bit);
    end component;
    signal y0, y1, y2, y3, y4, y5, y6, y7 : bit;
    begin
    d1 : decoder port map(a, b, c, y0, y1, y2, y3, y4, y5,
         y6, y7);
    count(0) <= y1 or y2 or y4 or y7;
    count(1) <= y3 or y5 or y6 or y7;
end eqn;
```

Output (from DirectVHDL):

```
Time   a   b   c   count y7  y6  y5  y4  y3  y2  y1  y0
0 ns   '0' '0' '0' 00    '0' '0' '0' '0' '0' '0' '0' '1'
5 ns   '0' '1' '0' 01    '0' '0' '0' '0' '0' '1' '0' '0'
10 ns  '1' '1' '0' 10    '0' '1' '0' '0' '0' '0' '0' '0'
15 ns  '1' '1' '1' 11    '1' '0' '0' '0' '0' '0' '0' '0'
20 ns  '0' '1' '1' 10    '0' '0' '0' '0' '1' '0' '0' '0'
```

10.E



Test Sequence: BCD = 0000, 0001, 1000, 1001.

Command Sequence:

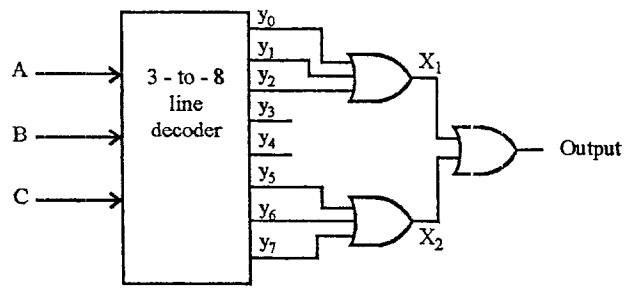
```
force bcd 0000
run 5ns
force bcd 0001
run 5ns
force bcd 1000
run 5ns
force bcd 1001
run 5ns
```

Output (from DirectVHDL):

```
Time  bcd  seven index
0 ns  0000 3f    0
5 ns  0001 6     1
10 ns 1000 7f    8
15 ns 1001 6f    9
```

```
library bitlib;
use bitlib.bit_pack.all;
entity rom is
    port (bcd : in bit_vector(3 downto 0);
    seven : out bit_vector(6 downto 0));
end rom;
architecture eqn of rom is
    type rom16_7 is array (0 to 15) of bit_vector (6
    downto 0);
    constant rom1 : rom16_7 := ("0111111", "0000110",
    "1011011", "1001111", "1100110", "1101101",
    "1111101", "0000111", "1111111", "1101111", others
    => "0000000");
    signal index : integer range 0 to 15;
    begin
    index <= vec2int(bcd);
    seven <= rom1(index);
end eqn;
```

10.F




```
entity decoder is
    port (a, b, c : in bit;
    y0, y1, y2, y3, y4, y5, y6, y7 : out bit);
end decoder;
architecture eqn of decoder is
    begin
    y0 <= not a and not b and not c;
    y1 <= not a and not b and c;
    y2 <= not a and b and not c;
    y3 <= not a and b and c;
    y4 <= a and not b and not c;
    y5 <= a and not b and c;
    y6 <= a and b and not c;
    y7 <= a and b and c;
end eqn;
-- Code for the main module
entity main is
    port (a, b, c : in bit;
    output : out bit);
end main;
architecture eqn of main is
    component decoder is
        port (a, b, c : in bit;
        y0, y1, y2, y3, y4, y5, y6, y7 : out bit);
    end component;
    signal y0, y1, y2, y3, y4, y5, y6, y7 : bit;
    signal x1, x2 : bit;
    begin
    d1 : decoder port map(a, b, c, y0, y1, y2, y3, y4, y5,
    y6, y7);
    x1 <= y0 or y1 or y2;
    x2 <= y5 or y6 or y7;
    output <= x1 or x2;
end eqn;
```

Test Sequence:
a b c = 0 0 0, 1 0 0, 1 0 1, 0 0 1, 0 1 1.

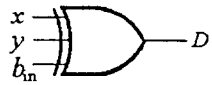
Command Sequence:

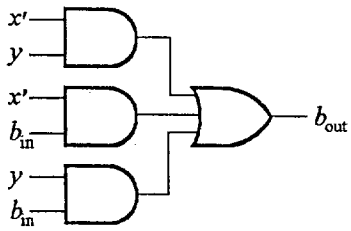
```
force a 0
force b 0
force c 0
run 5ns
force a 1
run 5ns
force c 1
run 5ns
force a 0
run 5ns
force b 1
run 5ns
```

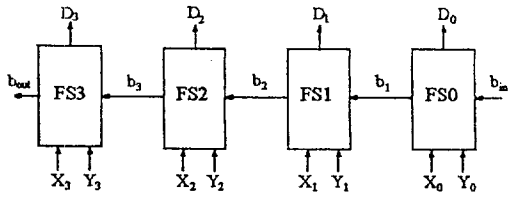
Output (from DirectVHDL):

Time	a	b	c	output	y7	y6	y5	y4	y3	y2	y1	y0	x2	x1
0 ns	'0'	'0'	'0'	'1'	'0'	'0'	'0'	'0'	'0'	'0'	'0'	'1'	'0'	'1'
5 ns	'1'	'0'	'0'	'0'	'0'	'0'	'0'	'1'	'0'	'0'	'0'	'0'	'0'	'0'
10 ns	'1'	'0'	'1'	'1'	'0'	'0'	'1'	'0'	'0'	'0'	'0'	'0'	'1'	'0'
15 ns	'0'	'0'	'1'	'1'	'0'	'0'	'0'	'0'	'0'	'0'	'1'	'0'	'0'	'1'
20 ns	'0'	'1'	'1'	'0'	'0'	'0'	'0'	'0'	'1'	'0'	'0'	'0'	'0'	'0'

10.G








```
-- Code for the full subtracter
entity sub1 is
    port (X, Y, bin : in bit;
    D, bout : out bit);
end sub1;
architecture eqn of sub1 is
    begin
    D <= X xor Y xor bin after 5 ns;
    bout <= (not X and bin) or (not X and Y) or (bin and
    Y) after 5 ns;
end eqn;
-- Code for the 4 bit subtracter
entity sub4 is
    port (X, Y : in bit_vector(3 downto 0);
    bin : in bit;
    D : out bit_vector(3 downto 0);
    bout : out bit);
end sub4;
architecture eqn of sub4 is
    component sub1 is
        port (X, Y, bin : in bit;
        D, bout : out bit);
    end component;
    signal b : bit_vector(3 downto 1);
    begin
    FS0 : sub1 port map (X(0), y(0), bin, D(0), b(1));
    FS1 : sub1 port map (X(1), y(1), b(1), D(1), b(2));
    FS2 : sub1 port map (X(2), y(2), b(2), D(2), b(3));
    FS3 : sub1 port map (X(3), y(3), b(3), D(3), bout);
end eqn;
```

Test Sequence: 1100-0101, 0110-1011

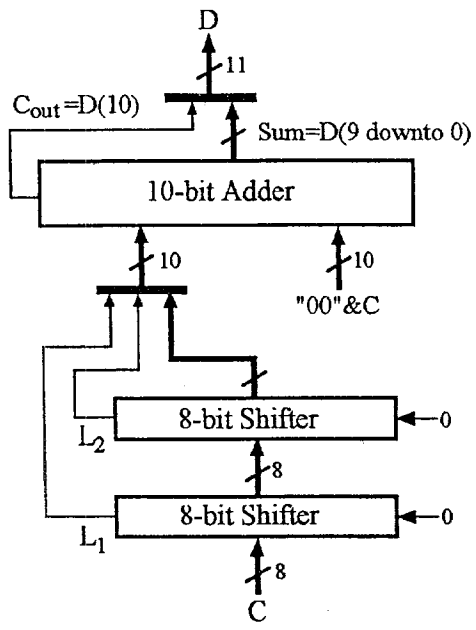
Command Sequence:

```
force X 1100
force Y 0101
force bin 0
run 25ns
force X 0110
force Y 1011
run 25ns
```

Output (from DirectVHDL):

```
Time   X     Y     bin D     bout b
0 ns   1100  0101  '0' 0000  '0'  000
5 ns   1100  0101  '0' 1001  '0'  001
10 ns  1100  0101  '0' 1011  '0'  011
15 ns  1100  0101  '0' 1111  '0'  111
20 ns  1100  0101  '0' 0111  '0'  111
25 ns  0110  1011  '0' 0111  '0'  111
30 ns  0110  1011  '0' 0011  '1'  011
35 ns  0110  1011  '0' 1011  '1'  011
```

10.H



Test Sequence: C = 10100101, 11111111

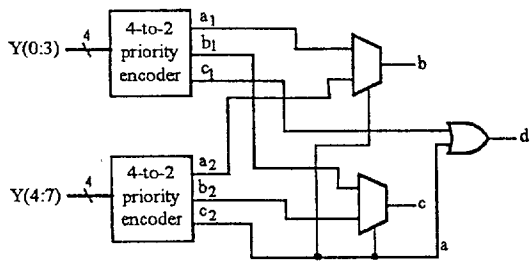
Command Sequence:
force C 10100101
run 10ns
force C 11111111
run 10ns

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity shifter is
    port (Rin : in std_logic;
    A : in std_logic_vector(7 downto 0);
    B : out std_logic_vector(7 downto 0);
    Lout : out std_logic);
end shifter;
architecture Behavioral of shifter is
    begin
    B(0) <= Rin;
    B(7 downto 1) <= A(6 downto 0);
    Lout <= A(7);
end Behavioral;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mult3 is
    port (C : in std_logic_vector(7 downto 0);
    D : out std_logic_vector(10 downto 0));
end mult3;
architecture Behavioral of mult3 is
    component shifter
        port (Rin : in std_logic;
        A : in std_logic_vector(7 downto 0);
        B : out std_logic_vector(7 downto 0);
        Lout : out std_logic);
    end component;
    signal E, F : std_logic_vector(7 downto 0);
    signal L1, L2 : std_logic;
    begin
    Shifter1 : shifter port map('0', C, E, L1);
    Shifter2 : shifter port map('0', E, F, L2);
    D <= ('0' & L1 & L2 & F) + C;
end Behavioral;
```

Output (from DirectVHDL):

```
Time     C        D            F        E        L2  L1
0 ns     10100101 01100111001 10010100 01001010 '0' '1'
10 ns    11111111 10011111011 11111100 11111110 '1' '1'
```

10.I



Test Sequence: Y = 00000000, 10000000, 11000000, ... , 11111111

Command Sequence:

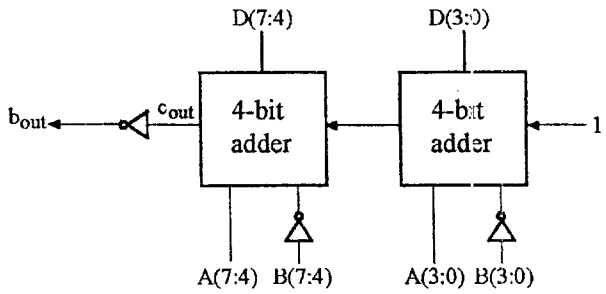
```
force y 00000000
run 5ns
force y 10000000
run 5ns
force y 11000000
run 5ns
force y 11100000
run 5ns
force y 11110000
run 5ns
force y 11111000
run 5ns
force y 11111100
run 5ns
force y 11111110
run 5ns
force y 11111111
run 5ns
```

```
-- Code for the 4 to 2 priority encoder
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity four_to_two_pe is
    port (y : in std_logic_vector(0 to 3);
    a1, b1, c1 : out std_logic);
end four_to_two_pe;
architecture equation of four_to_two_pe is
    begin
    a1 <= y(2) or y(3);
    b1 <= y(3) or (y(1) and not y(2));
    c1 <= y(0) or y(1) or y(2) or y(3);
end equation;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
-- Code for the 8 to 3 priority encoder
entity eight_to_three_pe is
    port (y : in std_logic_vector(0 to 7);
    a, b, c, d : out std_logic);
end eight_to_three_pe;
architecture Structure of eight_to_three_pe is
    component four_to_two_pe
        port (y : in std_logic_vector(0 to 3);
        a1, b1, c1 : out std_logic);
    end component;
    signal a1, b1, c1, a2, b2, c2 : std_logic;
    begin
    four_to_two_pe1 : four_to_two_pe
        port map (y(0 to 3), a1, b1, c1);
    four_to_two_pe2 : four_to_two_pe
        port map (y(4 to 7), a2, b2, c2);
    a <= c2;
    b <= a1 when c2 = '0' else a2;
    c <= b1 when c2 = '0' else b2;
    d <= y(0) or y(1) or y(2) or y(3) or y(4) or y(5) or
    y(6) or y(7);
    -- d <= '0' when y=0 else '1'; (alternate solution for d)
end Structure;
```

Output (from DirectVHDL):

```
Time  y        A   b   c   d   c2  b2  a2  c1  b1  a1
0 ns  00000000 '0' '0' '0' '0' '0' '0' '0' '0' '0' '0'
5 ns  10000000 '0' '0' '0' '1' '0' '0' '0' '1' '0' '0'
10 ns 11000000 '0' '0' '1' '1' '0' '0' '0' '1' '1' '0'
15 ns 11100000 '0' '1' '0' '1' '0' '0' '0' '1' '0' '1'
20 ns 11110000 '0' '1' '1' '1' '0' '0' '0' '1' '1' '1'
25 ns 11111000 '1' '0' '0' '1' '1' '0' '0' '1' '1' '1'
30 ns 11111100 '1' '0' '1' '1' '1' '1' '0' '1' '1' '1'
35 ns 11111110 '1' '1' '0' '1' '1' '0' '1' '1' '1' '1'
40 ns 11111111 '1' '1' '1' '1' '1' '1' '1' '1' '1' '1'
```

10.J



Command Sequence:
force A 11011011
force B 01110110
run 10ns
force A 01110110
force B 11011011
run 10ns

Test Sequence: 11011011-01110110,
01110110-11011011

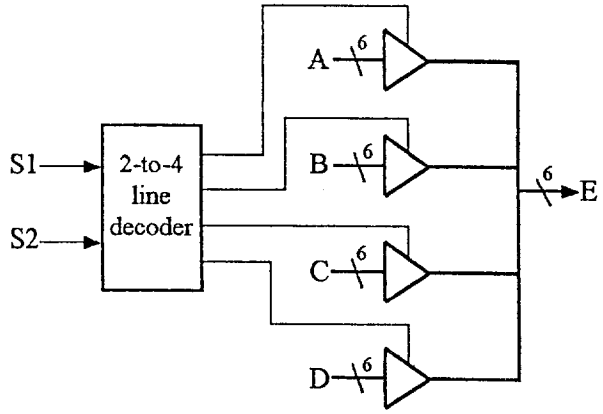
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity adder4 is
    port (X, Y : in std_logic_vector(3 downto 0);
    in : in std_logic;
    S : out std_logic_vector(3 downto 0);
    cout : out std_logic);
end entity;
architecture addeqn of adder4 is
    signal sum : std_logic_vector(4 downto 0);
    begin
    sum <= '0' & X + Y + cin;
    S <= sum(3 downto 0);
    cout <= sum(4);
end addeqn;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity subtracter8 is
    port(A, B : in std_logic_vector(7 downto 0);
    D : out std_logic_vector(7 downto 0);
    bout : out std_logic);
end entity;
architecture subeqn of subtracter8 is
    component adder4
        port (X, Y : in std_logic_vector(3 downto 0);
        cin : in std_logic;
        S : out std_logic_vector(3 downto 0);
        cout : out std_logic);
    end component;
    signal b1, cout : std_logic;
    signal notB : std_logic_vector(7 downto 0);
    signal D1, D2 : std_logic_vector(3 downto 0);
    begin
    notB <= not B;
    ADDERA : adder4 port map(A(3 downto 0),
        notB(3 downto 0), '1', D2, b1);
    ADDERB : adder4 port map(A(7 downto 4),
        notB(7 downto 4), b1, D1,cout);
    D <= D1 & D2;
    bout <= not cout;
end subeqn;
```

Output (from DirectVHDL):

Time	A	b	D	bout	bout1	b1	notB	D2	D1
0 ns	11011011	01110110	01100101	'0'	'1'	'1'	10001001	0101	0110
10 ns	01110110	11011011	10011011	'1'	'0'	'0'	00100100	1011	1001

10.K



Test Sequence: s1 s2 = 00, 01, 10, 11

Command Sequence:

```
force a 000111
force b 101010
force c 111000
force d 010101
force s1 0
force s2 0
run 5ns
force s2 1
run 5ns
force s1 1
force s2 0
run 5ns
force s2 1
run 5ns
```

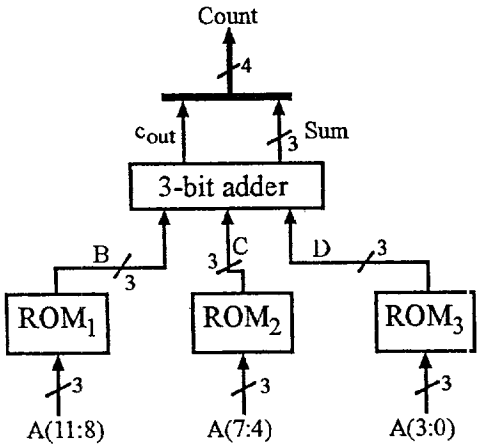
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity tsb is
    port (A : in std_logic_vector(5 downto 0);
    B : in std_logic;
    Z : out std_logic_vector(5 downto 0));
end tsb;
architecture Behavioral of tsb is
    begin
    Z <= A when B = '1' else "ZZZZZZ";
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity pkc is
    port (A, B, C, D : in std_logic_vector(5 downto 0);
    E : out std_logic_vector(5 downto 0);
    s1, s2 : in std_logic);
end pkc;
architecture Behavioral of pkc is
    component tsb
        port (A : in std_logic_vector(5 downto 0);
        B : in std_logic;
        Z : out std_logic_vector(5 downto 0));
    end component;
    signal internal : std_logic_vector (3 downto 0);
    begin
    internal(0) <= '1' when (s1 = '0' and s2='0') else '0';
    internal(1) <= '1' when (s1 = '0' and s2='1') else '0';
    internal(2) <= '1' when (s1 = '1' and s2='0') else '0';
    internal(3) <= '1' when (s1 = '1' and s2='1') else '0';
    tsb0 : tsb port map(A, internal(0), E);
    tsb1 : tsb port map(B, internal(1), E);
    tsb2 : tsb port map(C, internal(2), E);
    tsb3 : tsb port map(D, internal(3), E);
end Behavioral;
```

Output (from DirectVHDL):

```
Time   A      b      C      D      E      s1  s2  internal
0 ns   000111 101010 111000 010101 000111 '0' '0' 0001
5 ns   000111 101010 111000 010101 101010 '0' '1' 0010
10 ns  000111 101010 111000 010101 111000 '1' '0' 0100
15 ns  000111 101010 111000 010101 010101 '1' '1' 1000
```

10.L




```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity ROM4_3 is
    port (ROMin : in std_logic_vector(0 to 3);
    ROMout : out std_logic_vector(0 to 2));
end ROM4_3;
architecture Behavioral of ROM4_3 is
    type ROM16X3 is array (0 to 15) of std_logic_
    vector(0 to 2);
    constant ROM1: ROM16X3 := ("000", "001", "001",
    "010", "001", "010", "010", "011", "001", "010", "010",
    "011", "010", "011", "011", "100");
    signal index : integer range 0 to 15;
    begin
    index <= conv_integer(ROMin);
    ROMout <= ROM1(index);
end Behavioral;
```

Test Sequence: A = 111111111111,
010110101101, 100001011100

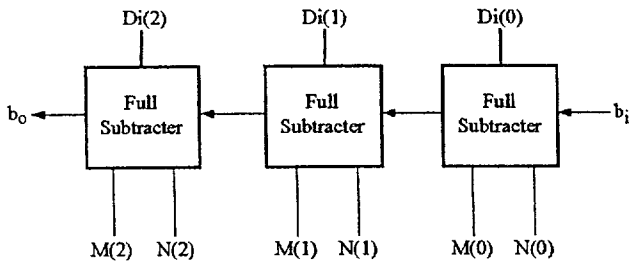
Command Sequence:
force A 111111111111
run 5ns
force A 010110101101
run 5ns
force A 100001011100
run 5ns

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity lab10l_soln is
    port (A : in std_logic_vector(11 downto 0);
    count : out std_logic_vector(3 downto 0));
end lab10l_soln;
architecture Behavioral of lab10l_soln is
    component ROM4_3
        port (ROMin: in std_logic_vector(0 to 3);
        ROMout: out std_logic_vector(0 to 2));
    end component;
    signal B, C, D : std_logic_vector(0 to 2);
    begin
    RO1 : ROM4_3 port map (A(11 downto 8), B);
    RO2 : ROM4_3 port map (A(7 downto 4), C);
    RO3 : ROM4_3 port map (A(3 downto 0), D);
    count <= "0" & B + C + D;
end Behavioral;
```

Output (from DirectVHDL):

Time	A	COUNT	D	C	b
0 ns	111111111111	1100	100	100	100
5 ns	010110101101	0111	011	010	010
10 ns	100001011100	0101	010	010	001

10.M



Test Sequence: 110-010 w/ borrow input of 1,
011-101 w/ borrow input of 0

Command Sequence:

```
force M 110
force N 010
force BI 1
run 5ns
force M 011
force N 101
force BI 0
run 5ns
```

```
library bitlib;
use bitlib.bit_pack.all;
entity FSub is
    port (X, Y, Bin : in bit;
    Bout, Dout : out bit);
end FSub;
architecture Behavioral of FSub is
    type ROM8x2 is array ( 0 to 7 ) of bit_vector(0 to 1 );
    constant ROM1 : ROM8x2 := ("00", "11", "11", "01",
    "10", "00", "00", "11");
    signal index : integer range 0 to 7;
    signal F : bit_vector(0 to 1);
    begin
    index <= vec2int(X&Y&Bin);
    F <= ROM1(index);
    Bout <= F(1);
    Dout <= F(0);
end Behavioral;
library bitlib;
use bitlib.bit_pack.all;
entity lab10m is
    port (M, N : bit_vector (2 downto 0);
    BI : in bit;
    BO : out bit;
    DI : out bit_vector(2 downto 0));
end lab10m;
architecture arch of lab10m is
    component FSub
        port (X, Y, Bin : in bit;
        Bout, Dout : out bit);
    end component;
    signal C : bit_vector (2 downto 1);
    begin
    SB0 : FSub port map (M(0), N(0), BI, C(1), DI(0));
    SB1 : FSub port map (M(1), N(1), C(1), C(2), DI(1));
    SB2 : FSub port map (M(2), N(2), C(2), BO, DI(2));
end arch;
```

Output (from DirectVHDL):

```
Time M   n   BI  BO  DI  C
0 ns 110 010 '1' '0' 011 11
5 ns 011 101 '0' '1' 110 00
```

Solutions to Unit 12 Design and Simulation Problems

Problem 12.10 is a simulation exercise where students are required to design and simulate a counter. The problem has 14 parts of equal difficulty, so that different students can be assigned different parts. We ask students to do the following preparation and lab work:

1. Read Unit 12 in the course textbook, completing Study Guide parts 1 through 5.

2. Read Section 2.2, "Simulating Flip-Flops with SimUaid" in the *SimUaid User's Guide* on the CD.

3. Answer the following questions:

   (a) How can a D flip-flop be set to logic 0 without using the clock input?

   (b) How can it be set to logic 1 without using the clock input?

   (c) Explain the term *Asynchronous Input.*

4. Design a counter that counts in the sequence assigned to you. Use D flip-flops, NAND gates, and inverters. Draw your circuit explicitly showing all connections to gate and flip-flop inputs. **Explicitly means that you should draw in all wires, don't just label the inputs and outputs. Show switches connected to the *Preset* and *Clear* inputs of the flip-flops. Use one switch for all clears and a separate switch for each preset.**

5. Explain in detail how you can set the flip-flops to the two missing states not in the prescribed counting sequence without using the clock input. Your explanation should describe each change you make to a switch position. After you have cleared or set a flip-flop, in what position (0 or 1) should you leave the switches?

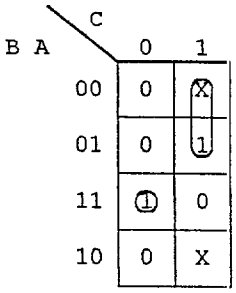
6. After a proctor has approved your preparation, go to one of the computer labs and work through the exercise for simulating a D flip-flop using SimUaid, found in Section 2.2 "Simulating Flip Flops with SimUaid" of the *SimUaid User's Guide* on the CD.

7. Enter your circuit from part 3 into SimUaid. In the space below, draw the **complete** state graph ***determined experimentally*** using your SimUaid circuit. Include the 6 states in the counting sequence **and the 2 states not in the sequence.**

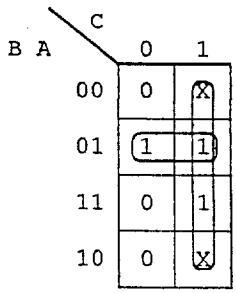
The complete solution for problem 12.10(a) follows. The solutions for parts (b) through (n) are similar, so only the state table, D flip-flop input equations (derived using Karnaugh maps), and the state graphs determined in part 6 are given. The D flip-flop input equations can also be derived using LogicAid by entering a state table with zero input variables.

12.10(a)

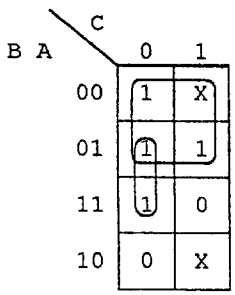
C B A	C^+ B^+ A^+
0 0 0	0 0 1
0 0 1	0 1 1
0 1 0	0 0 0
0 1 1	1 0 1
1 0 0	X X X
1 0 1	1 1 1
1 1 0	X X X
1 1 1	0 1 0



$D_C = C'B\,A + C\,B'$

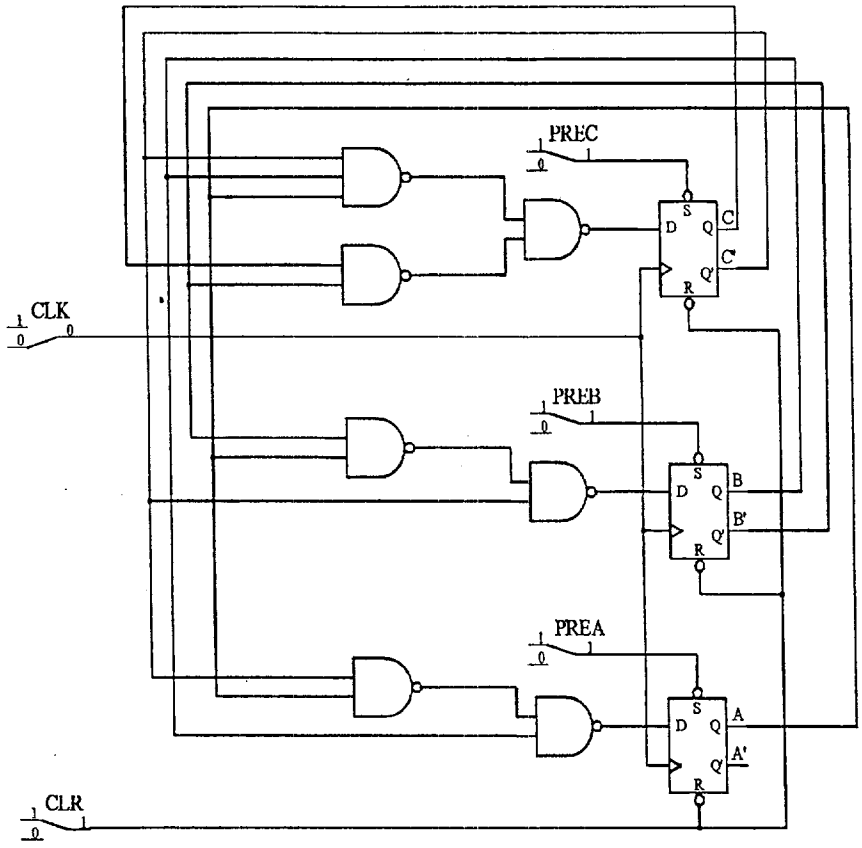


$D_B = B'A + C$

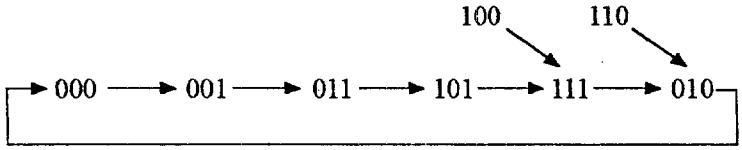


$D_A = B' + C'A$

12.10(a) (continued)



State graph determined experimentally:



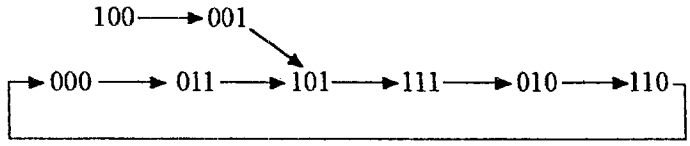
12.10(b)

C B A	C^+ B^+ A^+
0 0 0	0 1 1
0 0 1	X X X
0 1 0	1 1 0
0 1 1	1 0 1
1 0 0	X X X
1 0 1	1 1 1
1 1 0	0 0 0
1 1 1	0 1 0

$*D_C = C'B + B'A$ $\quad D_C = C'B + C\,B'$

$*D_B = C'A' + C\,A$ $\quad *D_A = B' + C'A$

Circuit based on equations marked * was used to obtain the following state graph



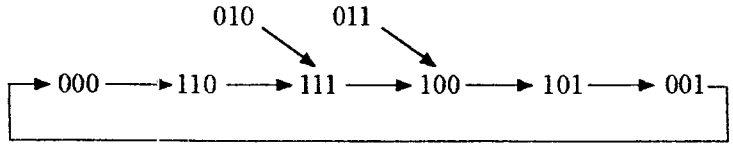
12.10(c)

C B A	C^+ B^+ A^+
0 0 0	1 1 0
0 0 1	0 0 0
0 1 0	X X X
0 1 1	X X X
1 0 0	1 0 1
1 0 1	0 0 1
1 1 0	1 1 1
1 1 1	1 0 0

$*D_C = A' + B$ $\quad *D_B = C'A' + BA'$

$*D_A = CB' + BA'$ $\quad D_A = CB' + CA'$

Circuit based on equations marked * was used to obtain the following state graph



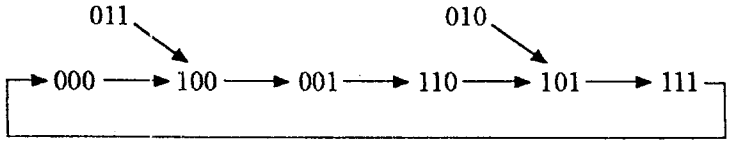
12.10(d)

C B A	C^+ B^+ A^+
0 0 0	1 0 0
0 0 1	1 1 0
0 1 0	X X X
0 1 1	X X X
1 0 0	0 0 1
1 0 1	1 1 1
1 1 0	1 0 1
1 1 1	0 0 0

$*D_C = C' + B'A + BA'$ $\quad *D_B = B'A$

$*D_A = CB' + BA'$ $\quad D_A = CB' + CA'$

Circuit based on equations marked * was used to obtain the following state graph



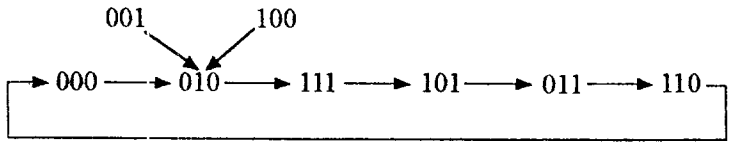
12.10(e)

C B A	C^+ B^+ A^+
0 0 0	0 1 0
0 0 1	X X X
0 1 0	1 1 1
0 1 1	1 1 0
1 0 0	X X X
1 0 1	0 1 1
1 1 0	0 0 0
1 1 1	1 0 1

$D_C = C'B + BA$ $\quad D_B = C' + B'$

$D_A = C'BA' + CA$

State graph determined experimentally:



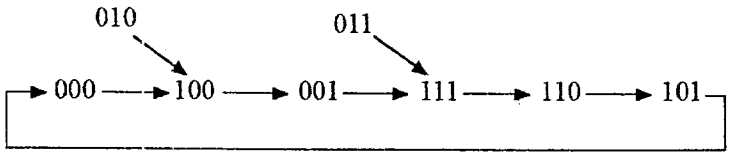
12.10(f)

C B A	C^+ B^+ A^+
0 0 0	1 0 0
0 0 1	1 1 1
0 1 0	X X X
0 1 1	X X X
1 0 0	0 0 1
1 0 1	0 0 0
1 1 0	1 0 1
1 1 1	1 1 0

$D_C = C' + B$ $\quad D_B = C'A + BA$

$D_A = C'A + CA'$

State graph determined experimentally:



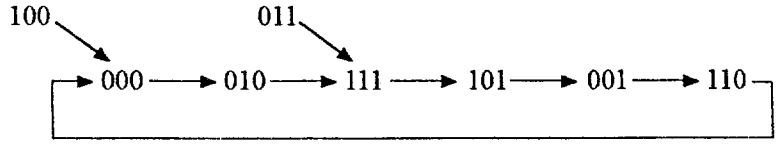
12.10(g)

C B A	C^+ B^+ A^+
0 0 0	0 1 0
0 0 1	1 1 0
0 1 0	1 1 1
0 1 1	X X X
1 0 0	X X X
1 0 1	0 0 1
1 1 0	0 0 0
1 1 1	1 0 1

$D_C = C'A + C'B + BA$ $\quad D_B = C'$

$D_A = C'B + CA$

State graph determined experimentally:



12.10(h)

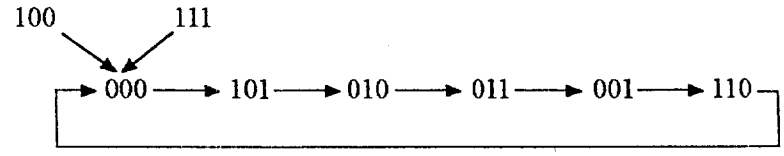
C B A	C^+ B^+ A^+
0 0 0	1 0 1
0 0 1	1 1 0
0 1 0	0 1 1
0 1 1	0 0 1
1 0 0	X X X
1 0 1	0 1 0
1 1 0	0 0 0
1 1 1	X X X

$*D_C = C'B'$ $\quad *D_B = B'A + C'BA'$

$*D_A = C'B + C'A'$ $\quad D_A = C'B + B'A'$

$D_A = C'A' + BA$

Circuit based on equations marked * was used to obtain the following state graph



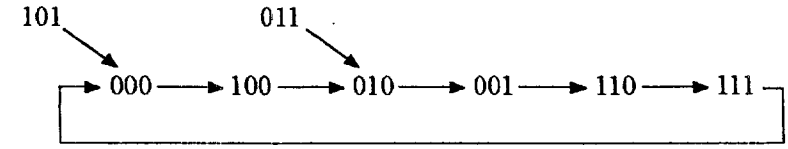
12.10(i)

C B A	C^+ B^+ A^+
0 0 0	1 0 0
0 0 1	1 1 0
0 1 0	0 0 1
0 1 1	X X X
1 0 0	0 1 0
1 0 1	X X X
1 1 0	1 1 1
1 1 1	0 0 0

$*D_C = C'B' + CBA'$ $\quad *D_B = CA' + C'A$

$D_B = CA' + B'A$ $\quad *D_A = BA'$

Circuit based on equations marked * was used to obtain the following state graph



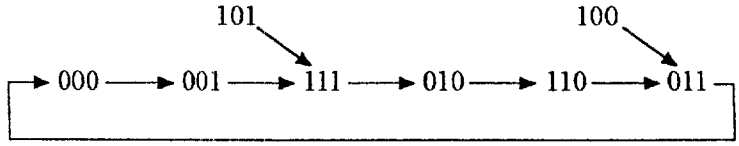
12.10(j)

C B A	C^+ B^+ A^+
0 0 0	0 0 1
0 0 1	1 1 1
0 1 0	1 1 0
0 1 1	0 0 0
1 0 0	X X X
1 0 1	X X X
1 1 0	0 1 1
1 1 1	0 1 0

$D_C = B'A + C'BA'$ $\quad D_B = B'A + BA' + C$

$D_A = B' + CA'$

State graph determined experimentally:



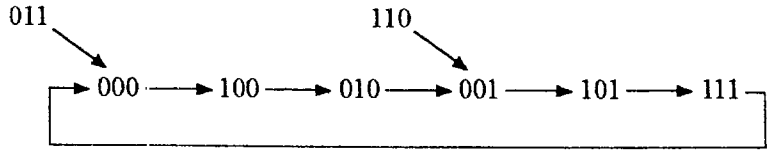
12.10(k)

C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	0	1
0	1	1	X	X	X
1	0	0	0	1	0
1	0	1	1	1	1
1	1	0	X	X	X
1	1	1	0	0	0

$*D_C = C'B' + B'A$ $\quad *D_B = C\,B'$

$*D_A = B'A + B\,A'$ $\quad D_A = B'A + C'B$

Circuit based on equations marked * was used to obtain the following state graph



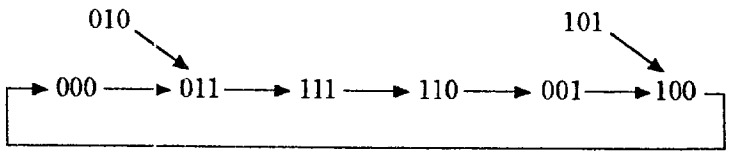
12.10(l)

C	B	A	C^+	B^+	A^+
0	0	0	0	1	1
0	0	1	1	0	0
0	1	0	X	X	X
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	X	X	X
1	1	0	0	0	1
1	1	1	1	1	0

$D_C = A$ $\quad D_B = C'A' + B\,A$

$D_A = C'A' + C'B + B\,A'$

State graph determined experimentally:



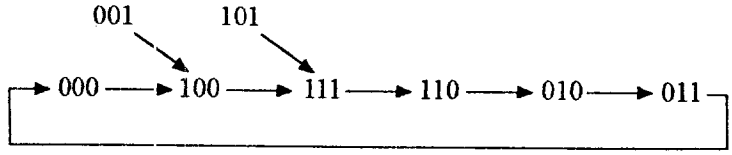
12.10(m)

C	B	A	C^+	B^+	A^+
0	0	0	1	0	0
0	0	1	X	X	X
0	1	0	0	1	1
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	X	X	X
1	1	0	0	1	0
1	1	1	1	1	0

$D_C = B' + C\,A$ $\quad D_B = B\,A' + C$

$D_A = C'B\,A' + C\,B'$

State graph determined experimentally:



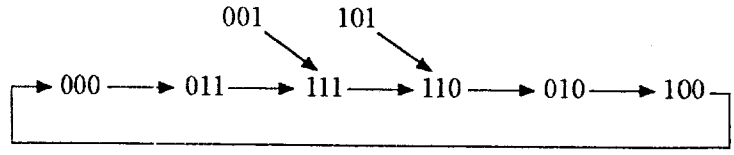
12.10(n)

C	B	A	C^+	B^+	A^+
0	0	0	0	1	1
0	0	1	X	X	X
0	1	0	1	0	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	1	X	X	X
1	1	0	0	1	0
1	1	1	1	1	0

$D_C = C'B + A$ $\quad D_B = C'B' + A + C\,B$

$D_A = C'B' + C'A$

State graph determined experimentally:



Solutions to Unit 16 Design and Simulation Problems

Problems 16.1 through 16.14 are Mealy sequential circuit design and simulation problems. These problems are of approximately equal difficulty so that different students can be assigned different problems. In this exercise, students first complete their designs using gates and D flip-flops. They then test their designs using the SimUaid simulator. Next, they convert their design to VHDL, synthesize the VHDL code, and download it to a CPLD board to test their design using hardware.

We ask our students to use the following procedure:

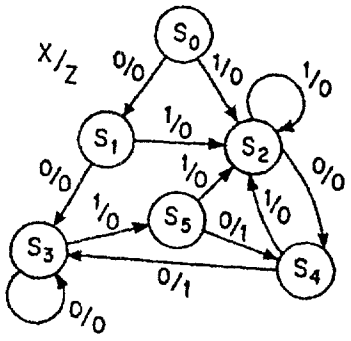
(1) Derive a state graph and state table for the assigned problem. Reduce the table to a minimum number of states. Check the reduced table using the LogicAid state table checker. Encoded solution files are found in the Lab16 folder on the CD.
(2) Make a state assignment using the guidelines. Derive the transition table, and then derive the D flip-flop input equations and output equation(s) using Karnaugh maps.
(3) Use LogicAid to derive the same equations and verify that the equations derived in step (2) are correct. Then derive one or more sets of equations for different state assignments using LogicAid.
(4) Design the circuit using NAND gates, NOR gates, and three D flip-flops. Choose the equations from step (3) that lead to the lowest cost circuit, and make sure that it meets the specifications.
(5) Input the logic circuit into SimUaid using switches for the X input, clock, reset, and preset inputs. Use probes for the Z output and flip-flop outputs. Use SimUaid to verify the transition table by presetting the flip-flops to each state and observing the next state and outputs. Then use SimUaid to manually test the operation of the circuit by applying the required test sequences and observing the outputs, being very careful to read the outputs at the proper time.
(6) Replace the clock and X input switches with a clock module and an input device. Program the input device to produce the proper test waveform. Display the simulator timing waveforms for clock, X, Z, and the flip-flop outputs. Print the waveforms and mark the times to read the Z output. Verify that the output sequence is correct.
(7) Replace the X and reset switches and the Z probe with a checker module (found on the SimUaid device menu), run the checker and verify that the circuit passes the test. The checker automatically generates input sequences and verifies that the output sequences are correct. Test files to use with the checker will be provided in the SimUaid/Checkers directory on the updated version of the CD.

After a proctor has verified that your design is correct and meets specifications, you will implement your design in hardware using the following steps:

(6) Use SimUaid to generate a VHDL file from your circuit file.
(7) Use the Xilinx ISE software to synthesize the circuit from the VHDL file.
(8) Generate a programming file and download it to the XCR 3064 on the hardware board.
(9) Test your circuit manually using the switches on the hardware board for the clock, reset, and X inputs. Use the same test sequence as you did in step (5).
(10) If you have an automatic hardware tester available, connect it to the circuit board and verify that your circuit passes.

The solution for 16.1 includes the complete SimUaid circuit and the VHDL code generated by SimUaid. The other solutions only give the logic equations for the flip-flop inputs and for Z.

16.1



	X = 0	1	X = 0	1
S_0	S_1	S_2	0	0
S_1	S_3	S_2	0	0
S_2	S_4	S_2	0	0
S_3	S_3	S_5	0	0
S_4	S_3	S_2	1	0
S_5	S_4	S_2	1	0

Assignment by guidelines:

I. (1, 3, 4) (2, 5) (0, 1, 2, 4, 5)

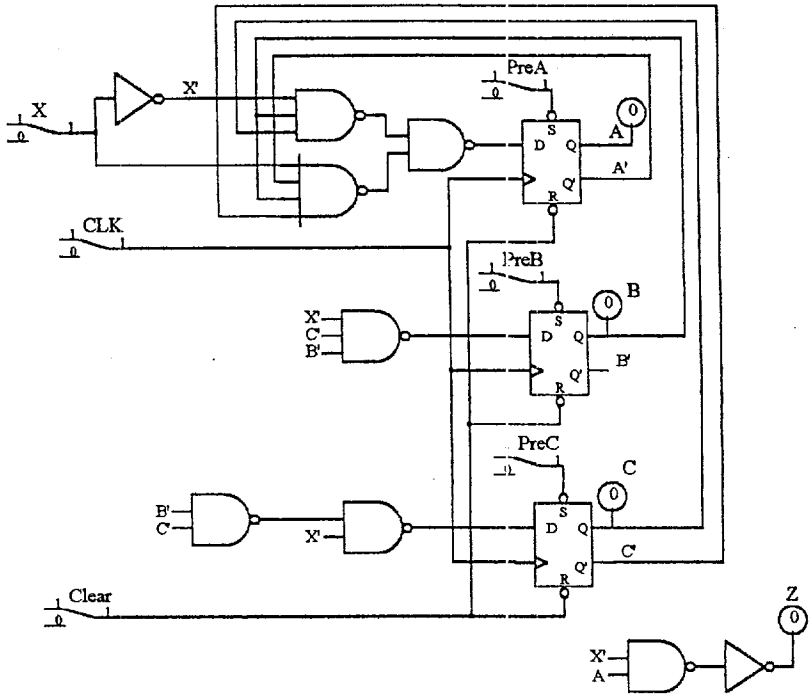
II. (1, 2) $(2, 3)_2$ $(2, 4)_2$ (3, 5)

III. (0, 1, 2, 3) (4, 5)

B C \ A	0	1
00	S_0	
01	S_1	
11	S_2	S_5
10	S_3	S_4

From Q^+ maps:

$A^+ = X'BC + XA'BC'$ $B^+ = X + C + B$ $C^+ = B'C' + X$ $Z = X'A$



See p. 204 for VHDL code generated by SimUaid.

Test sequences:

a) X = 001101001010100010010010

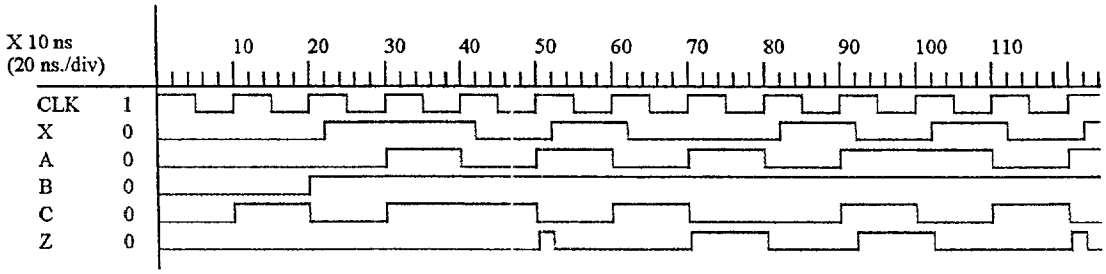
Z = 000000010100001001101101

b) X = 110011001010100101010010

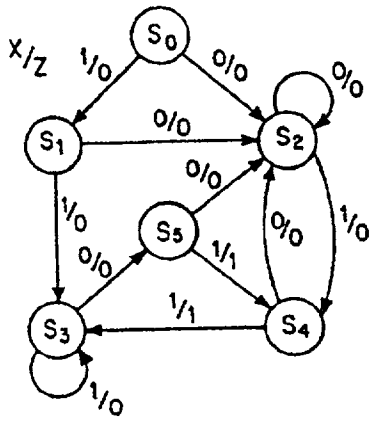
Z = 000100010100001010000101

To generate waveforms, replace the X switch with ⎍⎍ and the CLK switch with ⎍⎍

The input device should be programmed to generate the corresponding waveform.



16.2



	X = 0	1	X = 0	1
S_0	S_2	S_1	0	0
S_1	S_2	S_3	0	0
S_2	S_2	S_4	0	0
S_3	S_5	S_3	0	0
S_4	S_2	S_3	0	1
S_5	S_2	S_4	0	1

Guidelines and state assignments are the same as for16.1

Flip-flop and output equations and logic circuit are the same as 16.1, except interchange X and X' throughout

Test sequences a) X = 1100101101010111011011 01

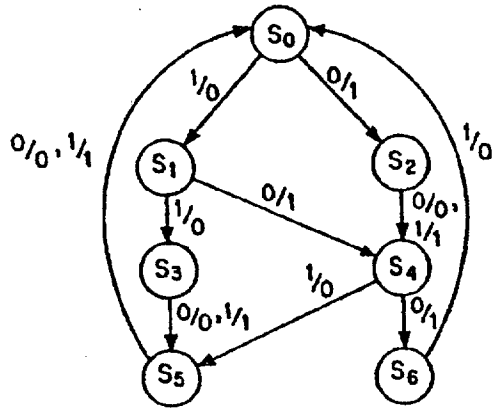
Z = 0000000101000010011011 01

b) X = 0011001101010110101011 01

Z = 0001000101000010100001 01

16.3

X				Z			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1



	X = 0	1	X = 0	1
S_0	S_2	S_1	1	0
S_1	S_4	S_3	1	0
S_2	S_4	S_4	0	1
S_3	S_5	S_5	0	1
S_4	S_6	S_5	1	0
S_5	S_0	S_0	0	1
S_6	-	S_0	-	0

Assignment by guidelines:

I. (1✓, 2) (5✓, 6) (3✓, 4)

II. (1✓, 2) (3✓, 4) (5✓, 6)

III. (0, 1✓, 4) (2, 3✓, 5)

B C \ A	0	1
00	S_0	
01	S_1	S_2
11	S_4	S_3
10	S_6	S_5

From Q+ maps:

$A^+ = XA'C + ABC + X'A'C'$ $B^+ = C$ $C^+ = B'$ $Z = X'A' + XA$

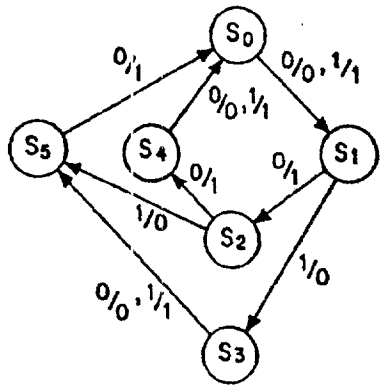
or $A^+ = XA'C + ABC + X'B'C'$

Test sequence: X = 1100 0010 1010 0110 1110 0001 1001 0101 1101 0011

Z = 0000 1000 0100 1100 0010 1010 0110 1110 0001 1001

16.4

X				Z			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	1	1	0
0	0	0	1	0	1	1	1
0	0	1	0	1	0	0	0
0	0	1	1	1	0	0	1
0	1	0	0	1	0	1	0
0	1	0	1	1	0	1	1
0	1	1	0	1	1	0	0
0	1	1	1	1	1	0	1
1	0	0	0	1	1	1	0
1	0	0	1	1	1	1	1



B C \ A	0	1
00	S_0	
01		S_1
11	S_3	S_2
10	S_4	S_5

	X = 0	1	X = 0	1
S_0	S_1	S_1	0	1
S_1	S_2	S_3	1	0
S_2	S_4	S_5	1	0
S_3	S_5	S_5	0	1
S_4	S_0	S_0	0	1
S_5	S_0	-	1	-

Assignment by guidelines:

I. (4, 5) (2, 3)

II. (2, 3) (4, 5)

III. (0, 3, 4) (1, 2, 5)

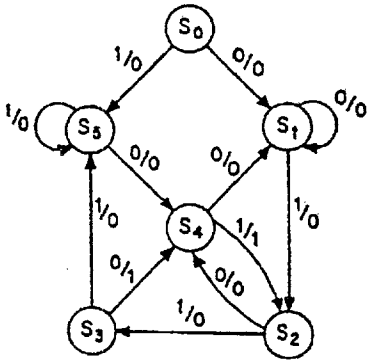
From Q^+ maps:

$A^+ = X'B' + A'C + A'B' + XAB \quad B^+ = C \quad C^+ = B' \quad Z = XA' + X'A$

Test sequence: X = 0000 1000 0100 1100 0010 1010 0110 1110 0001 1001

Z = 0110 1110 0001 1001 0101 1101 0011 1011 0111 1111

16.5



	X = 0	1	X = 0	1
S_0	S_1	S_5	0	0
S_1	S_1	S_2	0	0
S_2	S_4	S_3	0	0
S_3	S_4	S_5	1	0
S_4	S_1	S_2	0	1
S_5	S_4	S_5	0	0

Assignment by guidelines:

I. (0, 1, 4) (2, 3, 5) (0, 3, 5) (1, 4)

II. (1, 5) $(1, 2)_2$ (3, 4) $(4, 5)_2$

B C \ A	0	1
00	S_0	S_3
01		
11	S_1	S_2
10	S_4	S_5

From Q^+ maps:

$A^+ = X \quad B^+ = A' + C' + X' \quad C^+ = X'A' + A'B \quad Z = X'AB' + XA'BC'$

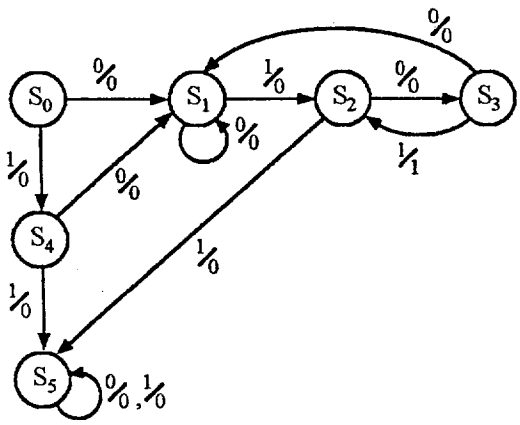
Test sequences: a) X = 0011011110010100

Z = 0000110000000100

b) X = 1010001111011000

Z = 0010000000010100

16.6



	X = 0	1	X = 0	1
S_0	S_1	S_4	0	0
S_1	S_1	S_2	0	0
S_2	S_3	S_5	0	0
S_3	S_1	S_2	0	1
S_4	S_1	S_5	0	0
S_5	S_5	S_5	0	0

B C \ A	0	1
00	S_0	S_2
01	S_1	S_4
11	S_3	S_5
10		

Assignment by guidelines:

I. (0, 1, 3, 4) (1, 3) (2, 4, 5)

II. (1, 4) $(1, 2)_2$ (3, 5) (1, 5)

III. (0, 1, 2, 4, 5)

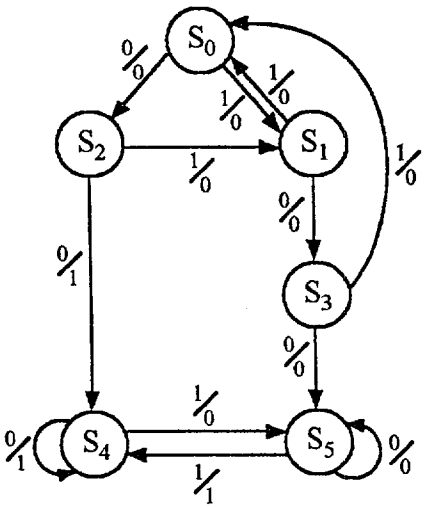
From Q^+ maps:

$A^+ = X + AB \quad B^+ = AC' + XA + AB \quad C^+ = X' + A + C' \quad Z = XA'B$

Test sequences: a) X = 010100010110
Z = 000100000100

b) X = 101010110101
Z = 000010100000

16.7



	X = 0	1	X = 0	1
S_0	S_2	S_1	0	0
S_1	S_3	S_0	0	0
S_2	S_4	S_1	1	0
S_3	S_5	S_0	0	0
S_4	S_4	S_5	1	0
S_5	S_5	S_4	0	1

B C \ A	0	1
00	S_0	S_4
01	S_1	S_5
11	S_3	
10	S_2	

Assignment by guidelines:

I. (0, 2) (1, 3) (2, 4) (3, 5)

II. (2, 1) (3, 0) (4, 1) (5, 0) $(4, 5)_2$

III. (0, 1, 3) (2, 4)

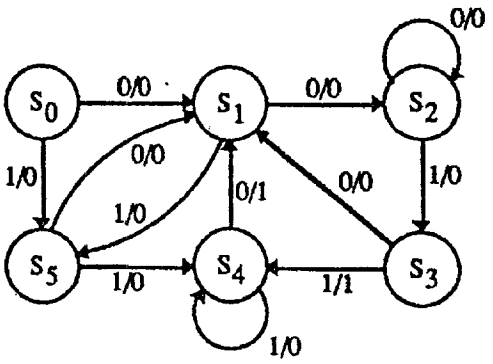
From Q^+ maps:

$A^+ = A + X'B \quad B^+ = X'A'B' \quad C^+ = X'C + XC' \quad Z = XAC + X'AC' + X'BC'$

Test sequences: a) X = 0110010100
Z = 0000100111

b) X = 101111001110
Z = 000000001011

16.8



	X = 0	1	X = 0	1
S_0	S_1	S_5	0	0
S_1	S_2	S_5	0	0
S_2	S_2	S_3	0	0
S_3	S_1	S_4	0	1
S_4	S_1	S_4	1	0
S_5	S_1	S_4	0	0

B C \ A	0	1
00	S_0	
01	S_3	
11	S_4	S_1
10	S_5	S_2

Assignment by guidelines:

I. (0, 3, 4, 5) (1, 2) (0, 1) (3, 4, 5)

II. (2, 3) $(1, 4)_3$ (2, 5) (1, 5)

III. (0, 1, 2, 5)

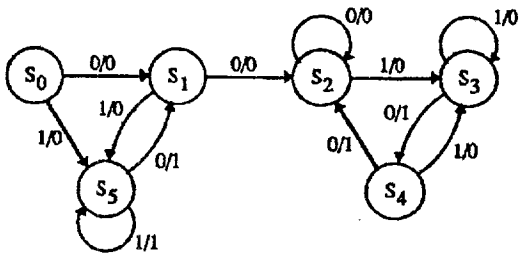
From Q^+ maps:

$A^+ = X'$ $B^+ = X' + A' + C$ $C^+ = X'A' + XBC' + A'C$ $Z = XB'C + X'A'BC$

Test sequences: a) X = 000100011010
Z = 000000001100

b) X = 111001000110
Z = 000100000011

16.9



	X = 0	1	X = 0	1
S_0	S_1	S_5	0	0
S_1	S_2	S_5	0	0
S_2	S_2	S_3	0	0
S_3	S_4	S_3	1	0
S_4	S_2	S_3	1	0
S_5	S_1	S_5	1	1

B C \ A	0	1
00	S_0	S_5
01		S_1
11	S_4	S_3
10		S_2

Assignment by guidelines:

I. (0, 5) (1, 2, 4) (0, 1, 5) (2, 3, 4)

II. (3, 4) $(1, 5)_2$ (2, 5) $(2, 3)_2$

III. (0, 1, 2) (3, 4)

From Q^+ maps:

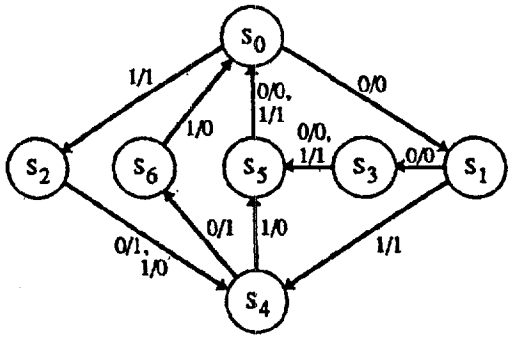
$A^+ = A' + B' + C' + X$ $B^+ = B + X'C$ $C^+ = XB + ABC + X'B'C'$ $Z = X'BC + AB'C'$

Test sequences: a) X = 100100100011
Z = 010011011000

b) X = 011000011011
Z = 001100000100

16.10

8	4	-2	-1	8	4	2	1
0	0	0	0	0	0	0	0
0	0	0	1	-	-	-	-
0	0	1	0	-	-	-	-
0	0	1	1	-	-	-	-
0	1	0	0	0	1	0	0
0	1	0	1	0	0	1	1
0	1	1	0	0	0	1	0
0	1	1	1	0	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	-	-	-	-
1	1	0	1	-	-	-	-
1	1	1	0	-	-	-	-
1	1	1	1	1	0	0	1



B C \ A	0	1
00	S_0	
01	S_1	S_2
11	S_3	S_4
10	S_5	S_6

Assignment by guidelines:

I. (1, 2) (3, 4) (5, 6)

II. (1, 2) (3, 4) (5, 6)

III. (0, 1, 3, 5) (2, 4)

	X = 0	1	X = 0	1
S_0	S_1	S_2	0	1
S_1	S_3	S_4	0	1
S_2	S_4	S_4	1	0
S_3	S_5	S_5	0	1
S_4	S_6	S_5	1	0
S_5	S_0	S_0	0	1
S_6	S_0	S_0	-	0

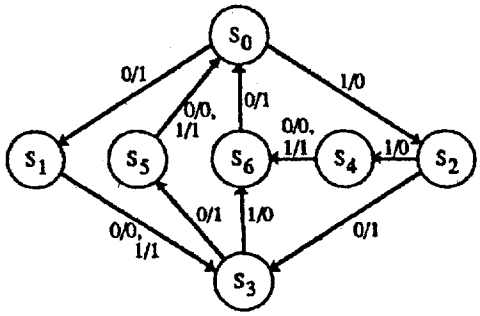
From Q^+ maps:

$A^+ = XB' + X'AC \quad B^+ = C \quad C^+ = B' \quad Z = XA' + X'A$

Test sequence: X = 0000 0010 1010 0110 1110 0001 1001 0101 1101 1111

Z = 0000 0010 1100 0100 1000 0001 1110 0110 1010 1001

16.11



	X = 0	1	X = 0	1
S_0	S_1	S_2	1	0
S_1	S_3	S_3	0	1
S_2	S_3	S_4	1	0
S_3	S_5	S_6	1	0
S_4	S_6	S_6	0	1
S_5	S_0	S_0	0	1
S_6	S_0	S_0	1	-

B C \ A	0	1
00	S_0	
01	S_6	S_5
11	S_3	S_4
10	S_2	S_1

Assignment by guidelines:

I. (1, 2) (5, 6) (3, 4)

II. (1, 2) (3, 4) (5, 6)

III. (0, 2, 3) (1, 4, 5)

From Q^+ maps:

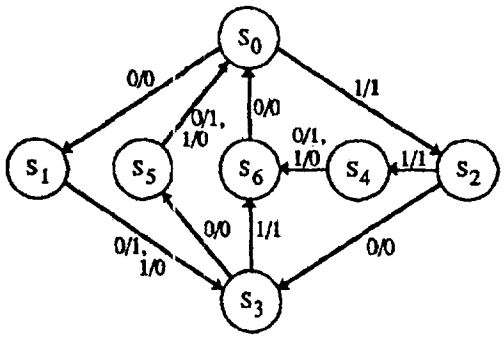
$A^+ = X'B'C' + XA'BC' + X'A'BC \quad B^+ = C' \quad C^+ = B \quad Z = XA + X'A'$

Test sequence: X = 0000 1000 0100 1100 0010 1010 0110 1110 0001 1001 0101

Z = 1010 0110 1110 0001 1001 0101 1101 0011 1011 0111 1111

16.12

X				Z			
0	0	0	0	1	0	1	0
0	0	0	1	1	0	0	1
0	0	1	0	1	0	0	0
0	0	1	1	0	1	1	1
0	1	0	0	0	1	1	0
0	1	0	1	0	1	0	1
0	1	1	0	0	1	0	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	1	0
1	0	0	1	0	0	0	1
1	0	1	0	0	0	0	0



	X = 0	1	X = 0	1
S_0	S_1	S_2	0	1
S_1	S_3	S_3	1	0
S_2	S_3	S_4	0	1
S_3	S_5	S_6	0	1
S_4	S_6	S_6	1	0
S_5	S_0	S_0	1	0
S_6	S_0	S_0	0	-

B C \ A	0	1
00	S_0	
01	S_6	S_5
11	S_3	S_4
10	S_2	S_1

Assignment by guidelines:

I. (1, 2) (5, 6) (3, 4)

II. (1, 2) (3, 4) (5, 6)

III. (0, 2, 3) (1, 4, 5)

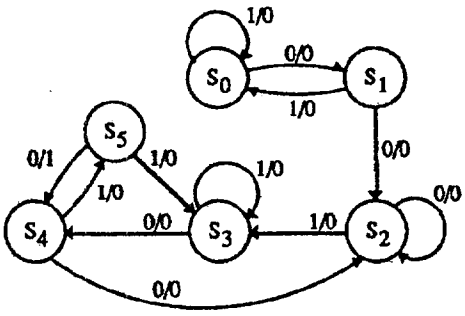
From Q^+ maps:

$A^+ = X'B'C' + XA'BC' + X'A'BC \quad B^+ = C' \quad C^+ = B \quad Z = X'A + XA'$

Test sequence: X = 0000 1000 0100 1100 0010 1010 0110 1110 0001 1001 0101

Z = 0101 1001 0001 1110 0110 1010 0010 1100 0100 1000 0000

16.13



	X = 0	1	X = 0	1
S_0	S_1	S_0	0	0
S_1	S_2	S_0	0	0
S_2	S_2	S_3	0	0
S_3	S_4	S_3	0	0
S_4	S_2	S_5	0	0
S_5	S_4	S_3	1	0

B C \ A	0	1
00	S_0	S_2
01	S_1	S_4
11		S_3
10		S_5

Assignment by guidelines:

I. (1, 2, 4) (3, 5) (0, 1) (2, 3, 5)

II. (0, 1) (0, 2) (2, 3) (3, 4)$_2$ (2, 5)

III. (0, 1, 2, 3, 4)

From Q^+ maps:

$A^+ = X'C + A \quad B^+ = XA \quad C^+ = X'A'C' + XAC' + B \quad Z = X'BC'$

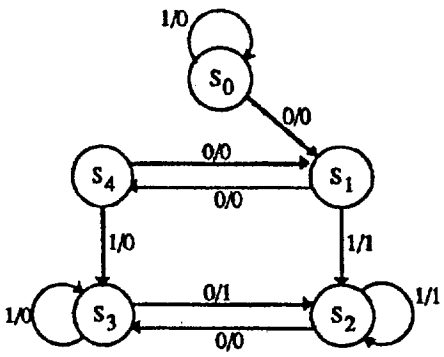
Test sequences: a) X = 100100110101

Z = 000000000010

b) X = 101000101010

Z = 000000000101

16.14



	X=0	1	X=0	1
S_0	S_1	S_0	0	0
S_1	S_4	S_2	0	1
S_2	S_3	S_2	0	1
S_3	S_2	S_3	1	0
S_4	S_1	S_3	0	0

B C \ A	0	1
00	S_0	S_4
01		S_2
11		S_1
10		S_3

Assignment by guidelines:

I. (0, 4)✓ (1, 2)✓ (3, 4)✓

II. (0, 1) (2, 4)✓ (2, 3)$_2$ (1, 3)✓

III. (0, 4)✓ (1, 2)✓

From Q^+ maps:

$A^+ = X' + A \quad B^+ = X'B' + XAC' \quad C^+ = XC + X'C' \quad Z = XC + X'BC'$

Test sequences: a) X = 10001101001
Z = 00001100100

b) X = 00001010001
Z = 00000110100

VHDL code for Problem 16.1 automatically generated by SimUaid follows. This code can be synthesized and downloaded to a CPLD or FPGA board for testing

```
-- This file has been automatically generated by SimUAid.
library ieee;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
library SimUAid_synthesis;
use SimUAid_synthesis.SimuAid_synthesis_pack.all;
entity lab16p1 is
port(CLK, PreA, PreB, PreC, Clear, X: in STD_LOGIC;
        Z, A, B, C, X_0, CLK_0: out STD_LOGIC
        );
end lab16p1;
architecture Structure of lab16p1 is
signal B_p, Vnet_0, Vnet_1, Vnet_2, Vnet_3, Vnet_4, C_p, Vnet_5, Vnet_6, Vnet_7,
        X_p, Vnet_8, Vnet_9, Vnet_10, Vnet_11, Vnet_12, Vnet_13, A_p, Vnet_14, Vnet_15: STD_LOGIC;
begin
        VHDL_Device_0: Dflipflop port map (CLK, Vnet_5, PreB, Clear, Vnet_3, B_p);
        VHDL_Device_1: Dflipflop port map (CLK, Vnet_6, PreC, Clear, Vnet_4, C_p);
        VHDL_Device_2: nand2 port map (Vnet_0, Vnet_1, Vnet_2);
        VHDL_Device_3: nand4 port map (X, A_p, Vnet_3, C_p, Vnet_1);
        VHDL_Device_4: nand3 port map (X_p, C_p, B_p, Vnet_5);
        VHDL_Device_5: nand2 port map (Vnet_7, X_p, Vnet_6);
        VHDL_Device_6: nand2 port map (B_p, C_p, Vnet_7);
        VHDL_Device_7: inverter port map (X, X_p);
        VHDL_Device_8: nand2 port map (X_p, Vnet_9, Vnet_11);
        VHDL_Device_9: inverter port map (Vnet_11, Z);
        VHDL_Device_10: nand3 port map (X_p, Vnet_3, Vnet_4, Vnet_0);
        VHDL_Device_11: Dflipflop port map (CLK, Vnet_2, PreA, Clear, Vnet_9, A_p);
        B <= Vnet_3;
        C <= Vnet_4;
        A <= Vnet_9;
        CLK_0 <= CLK;
        X_0 <= X;
end Structure;
```

Solutions to Unit 17 Simulation and Lab Problems

Problems 17.A through 17.M are relatively easy VHDL problems that use a register, counter, or other clocked device. We ask students to write VHDL code for their assigned problem, and then simulate, test, and debug their code. We have provided appropriate test sequences for each of these problems in the solutions that follow. We ask students to turn in simulation waveforms that demonstrate the operation of their code.

In addition to solving one of the above problems, we ask students to perform the following lab exercise:

(1) Write behavioral VHDL code that implements the state machine that you designed in Unit 16 (one of problems 16.1 through 16.14). Use a case statement to represent the state table as illustrated in FLD Figure 17-16. Use two processes – one for the combinational logic and one for the state register. Add an asynchronous reset input.

(2) Simulate the VHDL code and verify that it works correctly. Use the same test sequences that you used in Unit 16.

(3) Synthesize the VHDL and download it to a hardware board for testing. (We use the Xilinx ISE software for synthesizing the code and programming an XCR 3064 CPLD.)

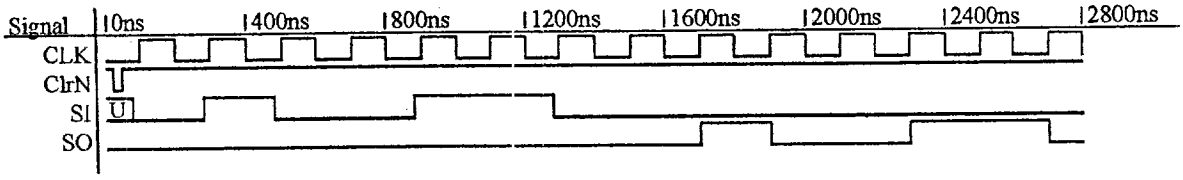
(4) Verify the correct operation of the hardware implementation of the state machine using the same procedures as in Unit 16.

17.A Test data:
- reset
- set Si = 0 1 0 0 1 1 0 0 0 0 0 0

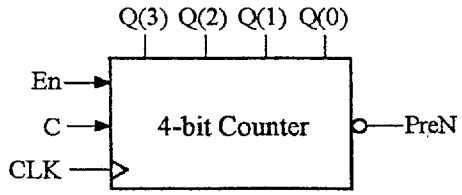
Command sequence:

```
force CLK 0 0, 1 100 -repeat 200
force ClrN 1 0, 0 25, 1 50
force SI 0 80, 1 280, 0 480, 0 680, 1 880,
    1 1080, 0 1280, 0 1480, 0 1680, 0 1880,
    0 2080, 0 2280
run 2800ns
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Lab17A is
    port (CLK, ClrN, SI : in std_logic;
    SO : out std_logic);
end Lab17A;
architecture Behavioral of Lab17A is
signal Q : std_logic_vector(7 downto 0):="00000000";
    begin
    SO <= Q(0);
    process(ClrN, CLK)
        begin
        if ClrN='0' then Q <= "00000000";
        elsif CLK' event and CLK='1' then
          Q <= SI & Q(7 downto 1); end if;
    end process;
end Behavioral;
```



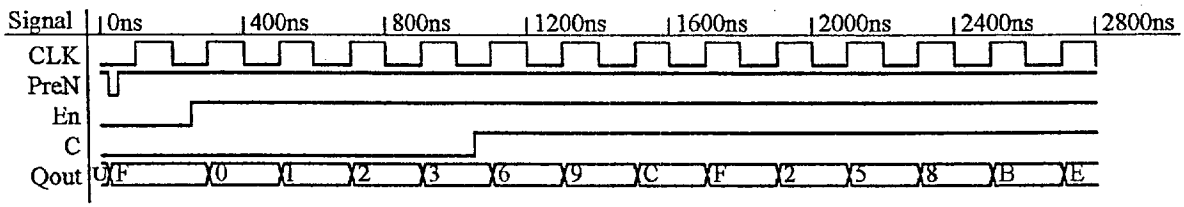
17.B



Test data:
- preset
- set En = 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
- set C = 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1

Command sequence:
force CLK 0 0, 1 100 -repeat 200
force PreN 1 0, 0 25, 1 50
force C 0
force En 0
run 200
force En 1 50
run 800
force C 1 50
run 1800

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Lab17B is
    port (CLK, PreN, En, C : in std_logic;
    Qout : out std_logic_vector (3 downto 0));
end Lab17B;
architecture count4bit of Lab17B is
signal Q : std_logic_vector (3 downto 0);
    begin
    Qout <= Q;
    process(PreN,CLK)
        begin
            if PreN = '0' then Q <= "1111";
            elsif CLK'event and CLK = '1' then
               if En = '0' then Q <= Q;
               elsif En = '1' and C = '0' then Q <= Q + 1;
               elsif En = '1' and C = '1' then Q <= Q + 3;
               end if;
            end if;
    end process;
end count4bit;
```

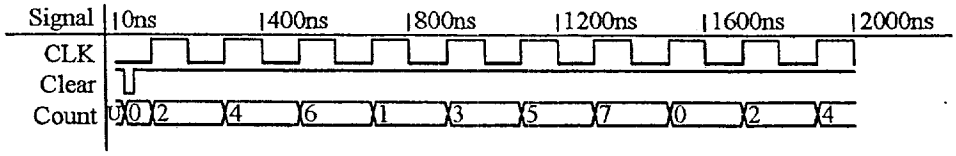


17.C

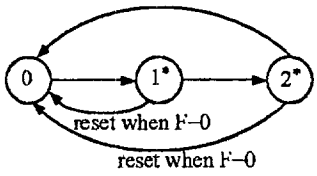
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity counter is
    port (CLK, ClrN : in std_logic;
    Count : out std_logic_vector(0 to 2));
end counter;
architecture Behavioral of counter is
type ROM8X3 is array(0 to 7) of
        std_logic_vector(0 to 2);
constant ROM1 : ROM8X3 := ("010", "011", "100",
"101", "110", "111", "001", "000");
signal Q, ROM_out, Index, D : std_logic_vector(0 to 2);
    begin
    Index <= Q;
    ROM_out <= ROM1(conv_integer(Index));
    Count <= Q;
    D <= ROM_out;
    process(CLK, ClrN)
        begin
        if ClrN='0' then Q <= "000";
        elsif CLK'event and CLK = '1' then Q <= D; end if;
    end process;
end Behavioral;
```

Test data:
- Counter sequence: 000, 010, 100, 110, 001, 011, 101, 111, 000 ...

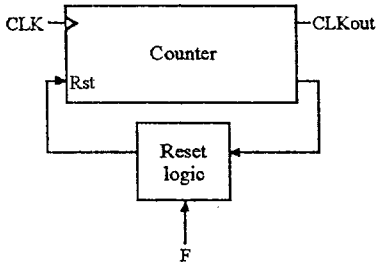
Command sequence:
force CLK 0 0, 1 100 -repeat 200
force ClrN 1 0, 0 25, 1 50
run 2000ns



17.D



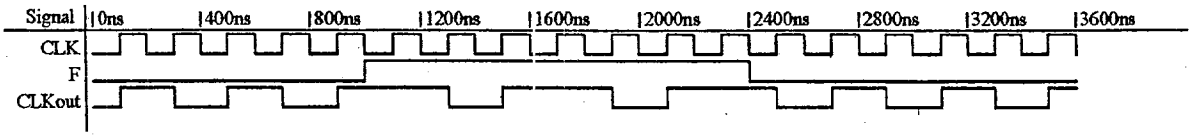
*CLKout – 1 in states 1 and 2



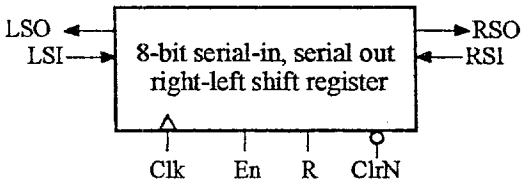
Test data:
- reset
- set F = 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0

Command sequence:
force CLK 0 0, 1 100 -repeat 200
force F 0 0, 1 1000, 0 2400
run 3600ns

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Lab17D is
    port (CLK, F : in std_logic;
    CLKout : out std_logic);
end Lab17D;
architecture Behavioral of Lab17D is
signal count : integer range 0 to 2 := 0;
signal Rst : std_logic;
    begin
    CLKout <= '0' when count = 0 else '1';
    Rst <= '1' when F = '0' and (count = 1 or count = 2)
        else '0';
    process(CLK)
        begin
        if ((CLK'event) and (CLK = '1')) then
            if Rst = '1' then count <= 0;
            elsif count = 2 then count <= 0;
            else count <= count + 1; end if;
        end if;
    end process;
end Behavioral;
```



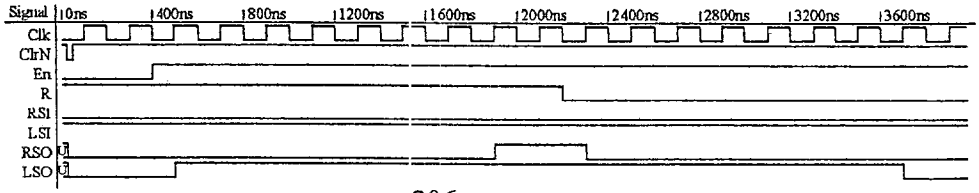
17.E



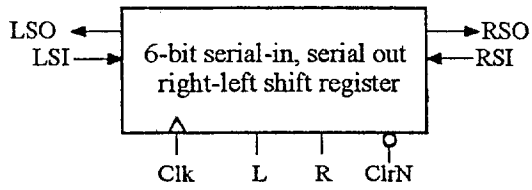
Test data:
- reset
- set LSI = 1; RSI = 0
- set En = 0 for 2 clock cycles
- set En = 1 for the rest of the test
- Shift Right 9x
- Shift Left 9x

Command sequence:
force Clk 0 0, 1 100 -repeat 200
force ClrN 1 0, 0 25, 1 50
force LSI 1
force RSI 0
force En 0 0, 1 400
force R 1 0, 0 2200, 1 4000
run 4000ns

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Lab17E is
    port (Clk, ClrN, En, R, RSI, LSI : in std_logic;
    RSO, LSO : out std_logic);
end Lab17E;
architecture Behavioral of Lab17E is
signal qint: std_logic_vector(7 downto 0);
    begin
    RSO <= qint(0);
    LSO <= qint(7);
    process(ClrN, Clk)
    begin
        if ClrN='0' then qint <= "00000000";
        elsif Clk' event and Clk='1' then
          if En='1' and R='1' then qint <= LSI&qint(7 downto 1);
          elsif En='1' and R='0' then
            qint <= qint(6 downto 0)&RSI; end if;
        end if;
    end process;
end Behavioral;
```



17.F

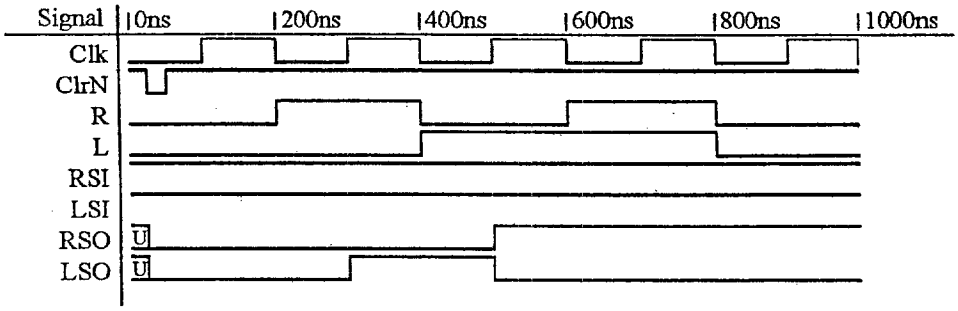


Test data:
- reset
- set LSI = 1; RSI = 1
- set R = 0 1 0 1 0 0 0
- set L = 0 0 1 1 0 0 0

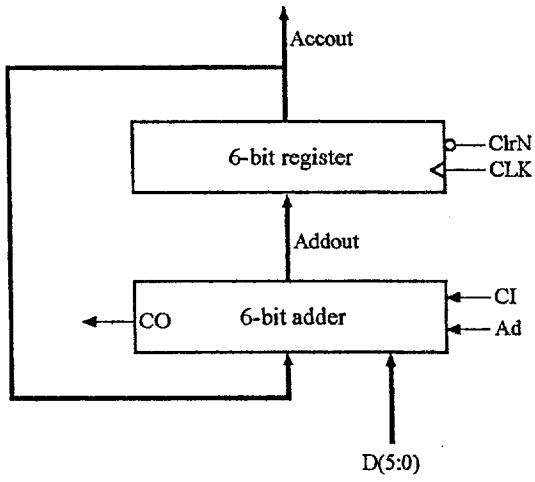
Command sequence:

```
force Clk 0 0, 1 100 -repeat 200
force ClrN 1 0, 0 25, 1 50
force LSI 1
force RSI 1
force R 0 0, 1 200, 0 400, 1 600, 0 800
force L 0 0, 1 400, 0 800
run 1000ns
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Lab17F is
    port (Clk, ClrN, R, L, RSI, LSI : in std_logic;
    RSO, LSO: out std_logic);
end Lab17F;
architecture Behavioral of Lab17F is
signal Q: std_logic_vector(5 downto 0);
    begin
    RSO <= Q(0);
    LSO <= Q(5);
    process(ClrN,Clk)
        begin
        if ClrN = '0' then Q <= "000000";
        elsif Clk' event and Clk='1' then
          if L='0' and R='1' then
            Q <= RSI&Q(5 downto 1);
          elsif L='1' and R='0' then
            Q <= Q(4 downto 0)&LSI; end if;
        end if;
    end process;
end Behavioral;
```



17.G

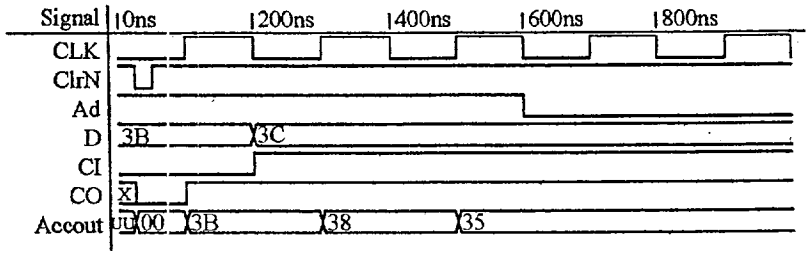


Test data:
- reset
-set D 111100 for one clock cycle
- set D = 111100 for the rest of the test
- set CI = 0 for the first clock cycle, 1 for the rest of the test
- set Ad = 1 for 3 clock cycles
  Ad = 0 for the next 2 clock cycles

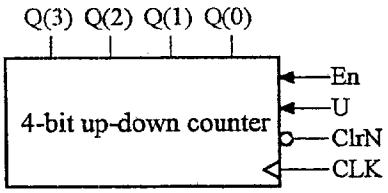
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity acc_6bit is
    port (CLK, ClrN, Ad, CI : in std_logic;
    D : in std_logic_vector(5 downto 0);
    CO : out std_logic;
    Accout : out std_logic_vector(5 downto 0));
end acc_6bit;
architecture Behavioral of acc_6bit is
signal Acc : std_logic_vector (5 downto 0);
signal Addout : std_logic_vector(6 downto 0);
    begin
    Addout <= ('0'&Acc) + D + CI;
    CO <= Addout(6);
    Accout <= Acc;
    process(CLK, ClrN)
        begin
        if ClrN = '0' then Acc <= "000000";
        elsif CLK'event and CLK = '1' then
          if Ad = '1' then
          Acc <= Addout(5 downto 0); end if;
        end if;
    end process;
end Behavioral;
```

17.G, continued

Command sequence:
force CLK 0 0, 1 100 -repeat 200
force ClrN 1 0, 0 25, 1 50
force Ad 1
force CI 0
force D 111011
run 200ns
force CI 1
force D 111100
force Ad 1 0, 0 400
run 800ns



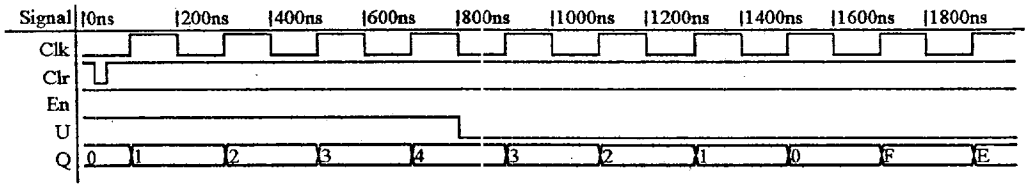
17.H



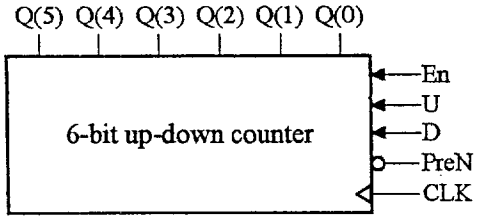
Test data:
- reset
- set En = 1 for 10 clock cycles
- set U = 1 for 4 clock cycles then U = 0 for the rest of the test

Command sequence:
force CLK 0 0, 1 100 -repeat 200
force ClrN 1 0, 0 25, 1 50
force En 1 0, 0 2000
force U 1 0, 0 800
run 2000ns

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity counter is
    port (CLK, ClrN, En, U, : in std_logic;
    Q : out std_logic_vector(3 downto 0));
end counter;
architecture Behavioral of counter is
signal W: std_logic_vector(3 downto 0):="0000";
    begin
    Q <= W;
    process(ClrN, Clk)
        begin
        if ClrN = '0'then
        W <= "0000";
        elsif CLK'event and CLK='1' then
          if En ='1' then
             if U = '1' then W <= W+1;
             elsif U = '0' then W <= W-1; end if;
          end if;
        end if;
    end process;
end Behavioral;
```



17.I



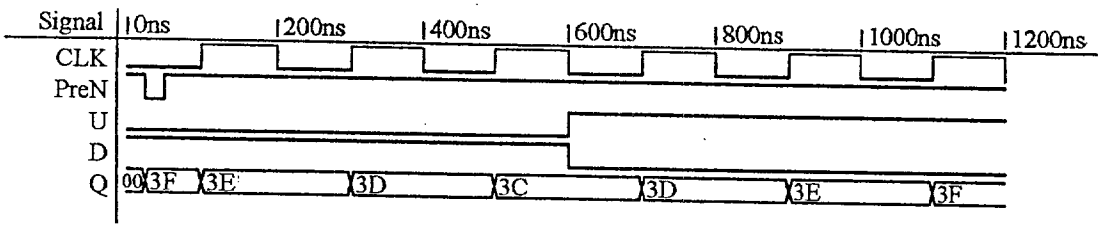

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity counter is
    port (CLK, PreN, U, D : in std_logic;
    Q : out std_logic_vector(5 downto 0));
end counter;
```

Test data:
- preset
- set U = 0 0 0 1 1 1
- set D = 1 1 1 0 0 0

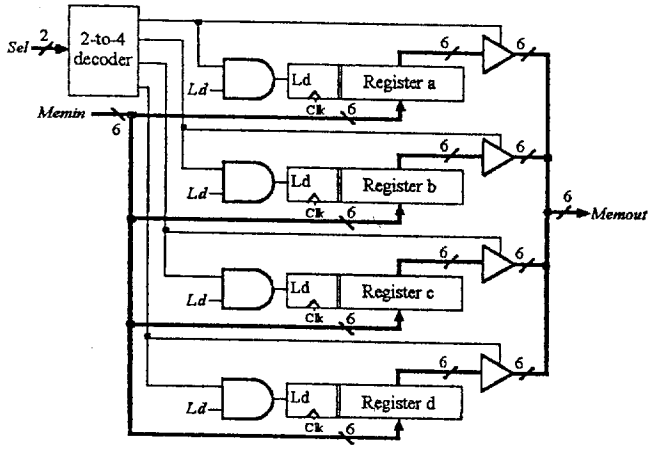
```
architecture Behavioral of counter is
signal W : std_logic_vector(5 downto 0):="000000";
    begin
    Q <= W;
    process(PreN, CLK)
        begin
        if PreN = '0' then
        W <= "111111";
        elsif CLK'event and CLK='1' then
          if U = '1' and D = '0' then W <= W+1;
          elsif U = '0' and D = '1' then W <= W-1; end if;
        end if;
    end process;
end Behavioral;
```

Command sequence:
force CLK 0 0, 1 100 -repeat 200
force PreN 1 0, 0 25, 1 50
force U 0 0, 1 600
force D 1 0, 0 600
run 1200ns

17.I, continued



17.J

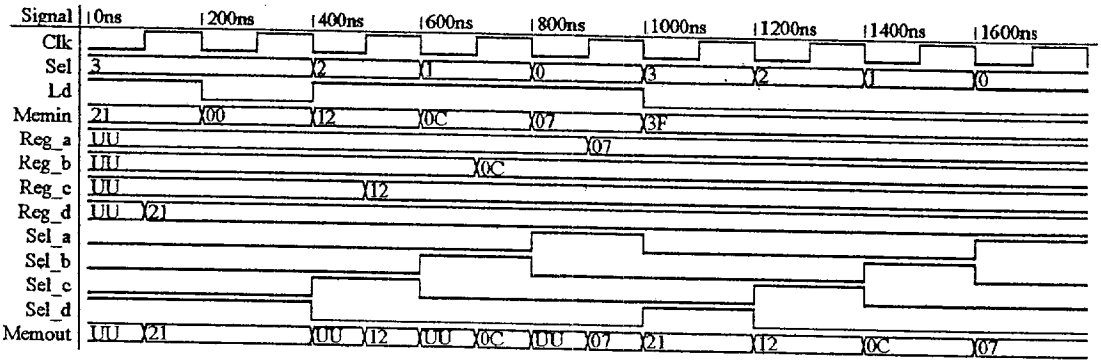


Test data:
- set Memin = 100001, Ld = 1, Sel = 11
- set Memin = 000000, Ld = 0, Sel = 11
- set Ld = 1
- set Memin = 010010, 001100, 000111 while Sel = 10, 01, 00
- set Ld = 0, Memin = 111111
- set Sel = 11, 10, 01, 00

Command sequence:

```
force CLK 0 0, 1 100 -repeat 200
force Memin 100001 0, 000000 200, 010010 400,
   001100 600, 000111 800, 111111 1000
force Ld 1 0, 0 200, 1 400, 0 1000
force Sel 11 0, 11 200, 10 400, 01 600, 00 800,
   11 1000, 10 1200, 01 1400, 00 1600
run 1800ns
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL
entity Lab17J is
    port ( Memin : in std_logic_vector(5 downto 0);
        clk, Ld : in std_logic;
        Sel : in std_logic_vector(1 downto 0);
        Memout : out std_logic_vector(5 downto 0));
end Lab17J;
architecture Behavioral of Lab17J is
signal Sel_a, Sel_b, Sel_c, Sel_d: std_logic;
signal Reg_a, Reg_b, Reg_c, Reg_d:
        std_logic_vector(5 downto 0);
    begin
    --Code for 2-to-4 decoder
    Sel_a <= not Sel(1) and not Sel(0);
    Sel_b <= not Sel(1) and Sel(0);
    Sel_c <= Sel(1) and not Sel(0);
    Sel_d <= Sel(1) and Sel(0);
    --Code for tri-state buffers
    Memout <= Reg_a when Sel_a = '1' else "ZZZZZZ";
    Memout <= Reg_b when Sel_b = '1' else "ZZZZZZ";
    Memout <= Reg_c when Sel_c = '1' else "ZZZZZZ";
    Memout <= Reg_d when Sel_d = '1' else "ZZZZZZ";
    --Code for Registers
    process(clk)
        begin
        if clk'event and clk = '1' then
          if (Sel_a and Ld) = '1' then
              Reg_a <= Memin; end if;
          if (Sel_b and Ld) = '1' then
              Reg_b <= Memin; end if;
          if (Sel_c and Ld) = '1' then
              Reg_c <= Memin; end if;
          if (Sel_d and Ld) = '1' then
              Reg_d <= Memin; end if;
        end if;
    end process;
end Behavioral;
```



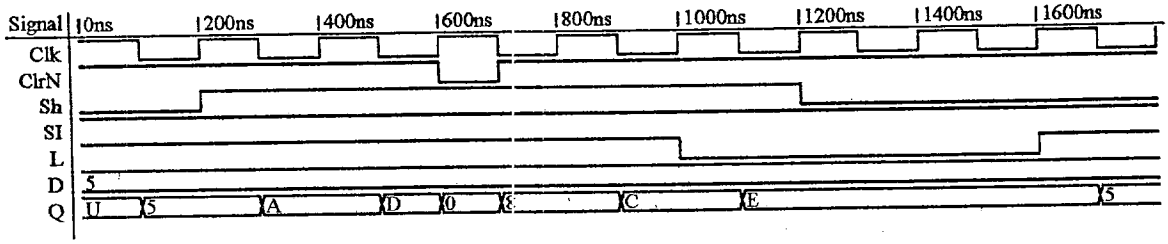
17.K Test data:

- set ClrN = 1 for 3.5 clock cycles
  = 0 for the next half clock cycle
  = 1 for the rest of the test
- set L = 1 for 5 clock cycles
  = 0 for the next 3 clock cycles
  = 1 for the rest of the test
- set SI = 1
- set D = 0101
- set Sh = 0 for 1 clock cycle
  = 1 for the next 5 clock cycles
  = 0 for the rest of the test

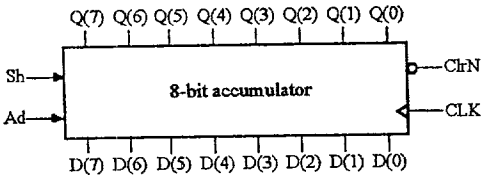
Command sequence:

```
force clk 1 0, 0 100 -repeat 200
force clrN 1 0, 0 600, 1 700
force l 1 0, 0 1000, 1 1600
force si 1
force d 0101
force sh 0 0, 1 200, 0 1200
run 1800
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity shift_reg1210 is
    port (CLK, ClrN, Sh, SI, L : in std_logic;
    D : in std_logic_vector (3 downto 0);
    Q : out std_logic_vector (3 downto 0));
end shift_reg1210;
architecture parallel_ShReg of shift_reg1210 is
signal Qint : std_logic_vector (3 downto 0);
    begin
    Q <= Qint;
    process(CLK, ClrN)
        begin
        if ClrN = '0' then Qint <= "0000";
        elsif CLK'event and CLK = '0' then
          if Sh = '1' then Qint <= SI & Qint (3 downto 1);
          elsif Sh = '0' and L = '1' then Qint <= D; end if;
        end if;
    end process;
end parallel_ShReg;
```



17.L



Test data:

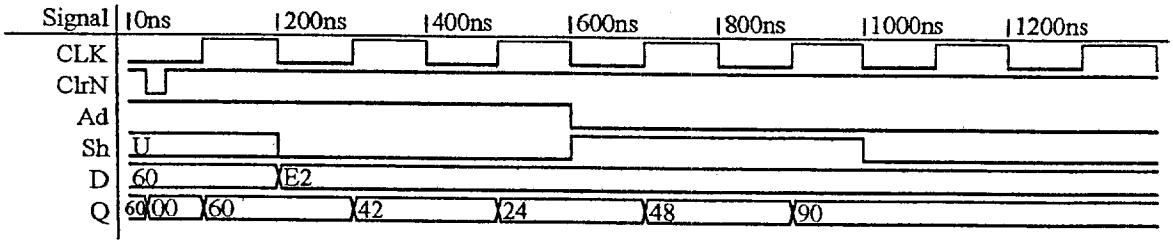
- reset
- set Acc = 01100000
- set D = 11100010
- set Ad = 1 for 2 clock cycles
  = 0 for the rest of the test
- set Sh = 0 for 2 clock cycles
  = 1 for 2 clock cycles
  = 0 for the rest of the test

Command sequence:

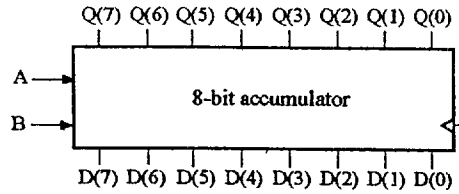
```
force CLK 0 0, 1 100 -repeat 200
force ClrN 1 0, 0 25, 1 50
force D 01100000
force Ad 1
run 200ns
force D 11100010
force Ad 1 0, 0 400
force Sh 0 0, 1 400, 0 800
run 1200ns
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity sregister is
    port (CLK, ClrN, Ad, Sh : in std_logic;
    D: in std_logic_vector(7 downto 0);
    Q: out std_logic_vector(7 downto 0));
end sregister;
architecture Behavioral of sregister is
signal Acc: std_logic_vector(7 downto 0):= "01100000";
    begin
    Q <= Acc;
    process(CLK, ClrN)
        begin
        if ClrN ='0' then Acc <= "00000000";
        elsif CLK'event and CLK = '1' then
          if Ad = '1' then Acc <= Acc + D;
          elsif Sh = '1' then Acc <= Acc(6 downto 0)&'0';
          else Acc <= Acc; end if;
        end if;
    end process;
end Behavioral;
```

17.L, continued



17.M

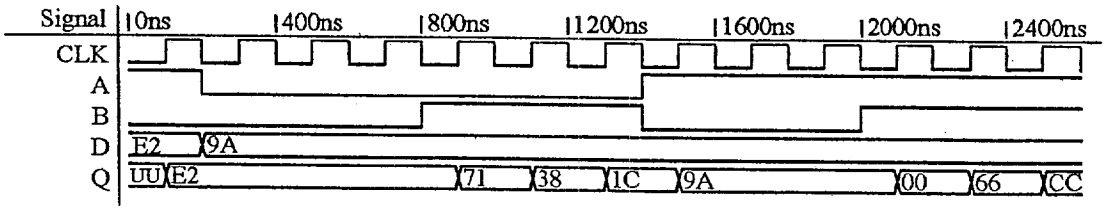


Test data:
- set D = 10011010
- set Acc = 11100010
- set AB = 00 for 3 clock cycles
  = 01 for the next 3 clock cycles
  = 10 for the next 3 clock cycles
  = 11 for the rest of the test

Command sequence:

```
force CLK 0 0, 1 100 -repeat 200
force D 11100010
force A 1
force B 0
run 200
force D 10011010
force A 0 0, 1 1200
force B 0 0, 1 600, 0 1200, 1 1800
run 2400
```

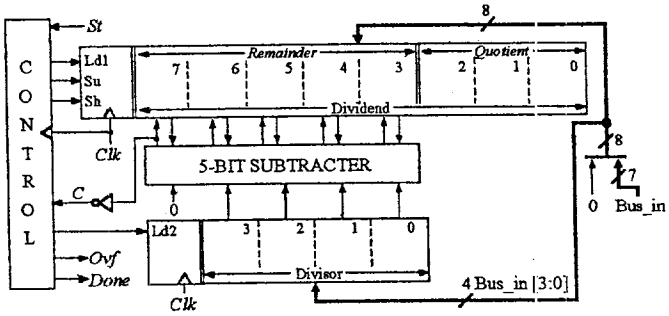
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Lab17M is
    port(CLK, A, B : in std_logic;
    D : in std_logic_vector(7 downto 0)
    Q : out std_logic_vector(7 downto 0));
end Lab17M;
architecture Behavioral of Lab17M is
signal Acc : std_logic_vector(7 downto 0);
    begin
    Q <= Acc;
    process(CLK)
        begin
        if CLK'event and CLK='1' then
          if (A and B)='1' then Acc <= Acc-D;
          elsif (A and not B)='1' then Acc <= D;
          elsif (not A and B)='1' then
            Acc <= '0'&Acc(7 downto 1); end if;
        end if;
    end process;
end Behavioral;
```

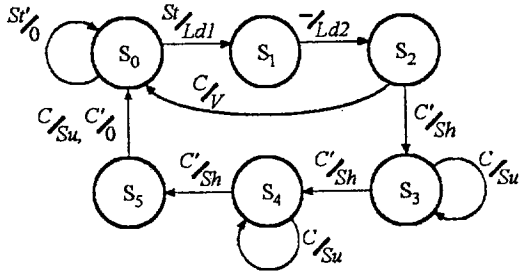


Solutions to Unit 20 Lab Design Problems

As a final lab assignment in our course, we ask students to solve one of the problems 20.A through 20.L Each of these problems is designed to fit on a small CPLD or FPGA circuit board with 8 input switches, 4 pushbuttons, and 8 LEDs. We ask our students to follow the procedure given on FLD p. 608. In addition to the given specifications, they should include an asynchronous reset input to set the state machine controller to the proper initial state. Students are asked to use a de-bounced pushbutton to manually clock their circuit and observe the step-by-step operation. Some problems require all 8 switches for input data. In this case, one of the pushbuttons can be used for the start signal. For problems 20.A through 20.D, the 7 or 8 switches serve as an input bus for loading both the dividend and the divisor.

20.A





Note: commented code is for 20.B

One process:

```
 library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity divider is
    port ( Bus_in: in std_logic_vector(6 downto 0);
         St, Clk, Reset: in std_logic;
         Quotient: out std_logic_vector(2 downto 0);
         --Quotient: out std_logic_vector(3 downto 0);
         Remainder: out std_logic_vector(4 downto 0);
         --Remainder: out std_logic_vector(3 downto 0);
         Overflow : out std_logic; --V
         Done : out std_logic) ;
end divider;
architecture Behavioral of divider is
signal State : integer range 0 to 6;
--signal State : integer range 0 to 7;
signal C : std_logic;
signal Subout :std_logic_vector(4 downto 0);
--signal Subout :std_logic_vector(3 downto 0);
signal Dividend: std_logic_vector(7 downto 0);
signal Divisor: std_logic_vector(3 downto 0);
--signal Divisor: std_logic_vector(2 downto 0);
    begin
    Subout<=Dividend(7 downto 3)-('0' & Divisor);
    --Subout<=Dividend(7 downto 4)-('0' & Divisor);
    C<=not Subout(4);
    --C<=not Subout(3);
    Remainder<=Dividend(7 downto 3);
    --Remainder<=Dividend(7 downto 4);
    Quotient<=Dividend(2 downto 0);
    --Quotient<=Dividend(3 downto 0);
    process(CLK, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif CLK'event and CLK='1' then
        case State is
        when 0 =>
            if St ='1' then
                Dividend<='0'& Bus_in;
                State<=1; Overflow <='0';
            else State <=0; end if;
        when 1 =>
            Divisor <= Bus_in (3 downto 0);
            --Divisor <= Bus_in (2 downto 0);
            State <= 2;
        when 2=>
            if C='1' then
                State <=0;
            else
            Dividend <= Dividend(6 downto 0) &'0';
            State<=3; end if;
        when 3|4 =>
        --when 3|4|5 =>
            if C='1' then
                Dividend(7 downto 3)<=Subout;
                --Dividend(7 downto 4)<=Subout;
                Dividend(0) <='1';
                State<=State;
            else
                Dividend<=Dividend(6 downto 0) &'0';
                State<=State+1; end if;
        when 5=>
        --when 6=>
            if C='1' then
                Dividend(7 downto 3)<=Subout;
                --Dividend(7 downto 4)<=Subout;
                Dividend(0) <='1'; end if;
            State <=6;
            -- State <= 7;
        when 6=> -- done state is optional
        --when 7 =>
            State <= 0;
        end case;
    end if;
end process;
```

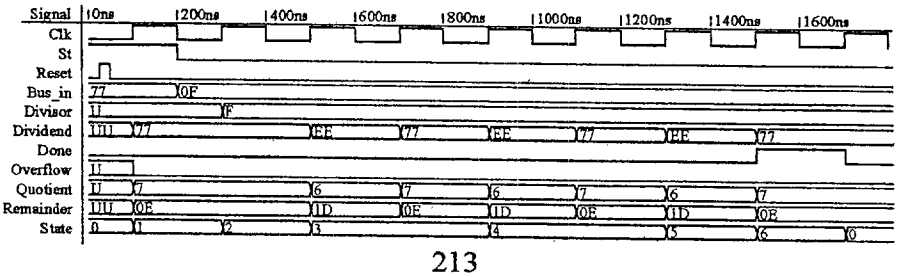
One process continued:

```
    Done <= '1' when State=6 else '0';
    --Done <= '1' when State=7 else '0';
    Overflow <= '1' when State = 2 and C = '1' else '0';
end Behavioral;
```

Two process:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity divider is
    port (Bus_in : in std_logic_vector (6 downto 0);
        St, Clk, Reset : in std_logic;
        Done, Overflow : out std_logic;
        Quotient : out std_logic_vector(2 downto 0);
        --Quotient : out std_logic_vector(3 downto 0);
        Remainder : out std_logic_vector(4 downto 0));
        --Remainder : out std_logic_vector(3 downto 0));
end divider;
architecture Behavioral of divider is
signal State, NextState : integer range 0 to 6;
--signal State, NextState : integer range 0 to 7;
signal C,Ld1, Ld2, Su, Sh : std_logic;
signal Subout :std_logic_vector(4 downto 0);
--signal Subout :std_logic_vector(3 downto 0);
signal Dividend: std_logic_vector(7 downto 0);
signal Divisor : std_logic_vector (3 downto 0);
--signal Divisor : std_logic_vector (2 downto 0);
    begin
    Subout<=Dividend(7 downto 3) - ('0' & Divisor);
    --Subout<=Dividend(7 downto 4) - ('0' & Divisor);
    C<=not Subout(4);
    --C<=not Subout(3);
    Remainder<=Dividend(7 downto 3);
    --Remainder<=Dividend(7 downto 4);
    Quotient<=Dividend(2 downto 0);
    --Quotient<=Dividend(3 downto 0);
    process(State,St,C)
        begin
        Ld1 <='0'; Ld2 <= '0';  Sh <='0'; Su <='0';
        case State is
          when 0 =>
            if St ='1' then
                Ld1 <='1';
                NextState<=1;
            else
                NextState <=0; end if;
          when 1=>
            Ld2 <= '1';
            NextState <= 2;
          when 2=>
            if C='1' then
                NextState <=0;
            else
            Sh <='1';
            NextState<=3; end if;
          when 3|4 =>
          --when 3|4|5 =>
            if C='1' then
                Su <='1';
                NextState<=State;
            else
                Sh <='1';
                NextState<=State+1;
            end if;
          when 5 =>
          --when 6 =>
            if C='1' then
                Su <='1'; end if;
            NextState <=6;
          when 6 =>
            NextState <= 0;
        end case;
    end process;
    process(CLK, Reset)
        begin
          if Reset = '1' then State <= 0;
          elsif CLK'event and CLK='1' then
            State<=NextState;
            if Ld1='1' then Dividend<='0'& Bus_in; end if;
            if Ld2='1' then
                Divisor <= Bus_in (3 downto 0); end if;
                --Divisor <= Bus_in (2 downto 0); end if;
            if Su='1' then
                Dividend(7 downto 3)<=Subout;
                --Dividend(7 downto 4)<=Subout;
                Dividend(0) <='1'; end if;
            if Sh='1' then
                Dividend<=Dividend(6 downto 0) &'0'; end if;
          end if;
    end process;
    Done<='1' when State=6 else '0';
    --Done<='1' when State=7 else '0';
    Overflow <= '1' when State=2 and C = '1' else '0';
end Behavioral;
```

Waveform for 1110111 / 1111 = 111 remainder 01110:



20.B For the corresponding code for the one and two process solutions, refer to problem 20.A. Commented lines are used for this problem and the lines above the commented ones are used for 20.A.

20.C **Note:** commented code is for 20.D

One process:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity divider20c_1 is
    port ( Bus_in : in std_logic_vector (7 downto 0);
        St, Clk, Reset : in std_logic;
        Done, Overflow : out std_logic;
        Quotient : out std_logic_vector (4 downto 0);
        --Quotient : out std_logic_vector (2 downto 0);
        Remainder : out std_logic_vector (3 downto 0));
        --Remainder : out std_logic_vector (5 downto 0));
end divider20c_1;
architecture Behavioral of divider20c_1 is
signal State : integer range 0 to 8;
--signal State : integer range 0 to 6;
signal C ; std_logic;
signal subout : std_logic_vector (3 downto 0);
--signal subout : std_logic_vector (5 downto 0);
signal Dividend: std_logic_vector (8 downto 0);
signal Divisor : std_logic_vector (2 downto 0);
--signal Divisor : std_logic_vector (4 downto 0);
    begin
    subout <= Dividend(8 downto 5) - ('0'&Divisor);
    --subout <= Dividend(8 downto 3) - ('0'&Divisor);
    C <= not subout (3);
    --C <= not subout (5);
    Remainder <= Dividend (8 downto 5);
    --Remainder <= Dividend (8 downto 3);
    Quotient <= Dividend (4 downto 0);
    --Quotient <= Dividend (2 downto 0);
    process (Clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif Clk'event and Clk = '1' then
          case State is
          when 0 =>
              if St = '1' then
                  Dividend <= '0' & Bus_in;
                  State <= 1;
              else
                  State <= 0; end if;
          when 1 =>
              Divisor <= Bus_in (2 downto 0);
              --Divisor <= Bus_in (4 downto 0);
              State <= 2;
          when 2 =>
              if C = '1' then State <= 0;
              else
                  Dividend <= Dividend(7 downto 0) & '0';
                  State <= 3; end if;
          when 3|4|5|6 =>
          --when 3|4 =>
          if C = '1' then
                  Dividend (8 downto 5) <= subout;
```

20.C, continued

```
                  --Dividend (8 downto 3) <= subout;
                Dividend (0) <= '1'; State <= State;
            else
                Dividend <= Dividend (7 downto 0) & '0';
                State <= State + 1; end if;
          when 7 =>
          --when 5 =>
              if C = '1' then
                  Dividend (8 downto 5) <= subout;
                  --Dividend (8 downto 3) <= subout;
                  Dividend (0) <= '1'; end if;
                  State <= 8;
                  --State <= 6;
          when 8 =>
          --when 6 =>
              State <= 0;
          end case;
        end if;
    end process;
    Done <= '1' when State = 8 else '0';
    --Done <= '1' when State = 6 else '0';
    Overflow <= '1' when State = 2 and C = '1' else '0';
end Behavioral;
```

Two process:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity divider20c_1 is
    port (Bus_in : in std_logic_vector (7 downto 0);
        St, Clk, Reset : in std_logic;
        Done, Overflow : out std_logic;
        Quotient : out std_logic_vector (4 downto 0);
        --Quotient : out std_logic_vector (2 downto 0);
        Remainder : out std_logic_vector (3 downto 0));
        --Remainder : out std_logic_vector (5 downto 0));
end divider20c_1;
architecture Behavioral of divider20c_1 is
signal State, NextState : integer range 0 to 8;
--signal State, NextState : integer range 0 to 6;
signal C, Sh, Su, Ld1, Ld2 : std_logic;
signal subout : std_logic_vector (3 downto 0);
--signal subout : std_logic_vector (5 downto 0);
signal Dividend: std_logic_vector (8 downto 0);
signal Divisor : std_logic_vector (2 downto 0);
--signal Divisor : std_logic_vector (4 downto 0);
    begin
    subout <= Dividend(8 downto 5) - ('0'&Divisor);
    --subout <= Dividend(8 downto 3) - ('0'&Divisor);
    C <= not subout (3);
    --C <= not subout (5);
    Remainder <= Dividend (8 downto 5);
    --Remainder <= Dividend (8 downto 3);
    Quotient <= Dividend (4 downto 0);
    --Quotient <= Dividend (2 downto 0);
    process (State, C, St)
        begin
        Sh <= '0'; Su <= '0'; Ld1 <= '0'; Ld2 <= '0';
        Overflow <= '0'; Done <= '0';
        case State is
          when 0 =>
              if St = '1' then Ld1 <= '1'; NextState <= 1;
```

20.C, continued

Two process continued:

```
            else NextState <= 0; end if;
          when 1 =>
            Ld2 <= '1'; NextState <= 2;
          when 2 =>
            if C = '1' then NextState <= 0; Overflow <= '1';
            else Sh <= '1'; NextState <= 3; end if;
          when 3|4|5|6 =>
          --when 3|4 =>
            if C = '1' then Su <= '1'; NextState <= State;
            else Sh <= '1'; NextState <= State + 1; end if;
          when 7 =>
          --when 5 =>
            if C = '1' then Su<= '1'; end if;
            NextState <= 8;
            --NextState <= 6;
          when 8 =>
          --when 6 =>
            Done <= '1'; NextState <= 0;
        end case;
      end process;
      process (Clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif Clk'event and Clk = '1' then
          State <= NextState;
          if Ld1 = '1' then
            Dividend <= '0' & Bus_in; end if;
          if Ld2 = '1' then
            Divisor <= Bus_in (2 downto 0); end if;
            --Divisor <= Bus_in (4 downto 0); end if;
          if Sh = '1' then
            Dividend <= Dividend (7 downto 0) & '0'; end if;
          if Su = '1' then
            Dividend (8 downto 5) <= subout;
            --Dividend (8 downto 3) <= subout;
            Dividend (0) <= '1'; end if;
          end if;
      end process;
    end Behavioral;
```

20.D For the corresponding code for the one and two process solutions, refer to problem 20.C. Commented lines are used for this problem and the lines above the commented ones are used for 20.C.

20.E Block diagram and state graph for 20.E-20.H are similar to Fig 18.7 and 18.8, respectively in FLD5. However, modifications to the block diagram and state graph need to be made for each problem to obtain the correct solution.

20.E, continued

One process:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity multiply is
    port (Clk, St, Reset : in std_logic;
        Mplier : in std_logic_vector(3 downto 0);
        Mcand : in std_logic_vector(2 downto 0);
        Done: out std_logic;
        Product: out std_logic_vector(6 downto 0));
end multiply;
architecture Behavioral of multiply is
signal state: integer range 0 to 9;
signal ACC: std_logic_vector(7 downto 0);
alias M: std_logic is ACC(0);
    begin
    Product<= ACC(6 downto 0);
    process(Clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif Clk'event and Clk='1' then
          case state is
          when 0 =>
              if St='1' then
                  ACC(7 downto 4) <= "0000";
                  ACC(3 downto 0) <= Mplier;
                  state <= 1;
              end if;
          when 1 | 3 | 5 | 7 =>
              if M = '1' then
                  ACC(7 downto 4)<=
                      ('0'&ACC(6 downto 4))+Mcand;
                  state <= state + 1;
              else
                  ACC <= '0' & ACC(7 downto 1);
                  state <= state + 2;
              end if;
          when 2 | 4 | 6 | 8 =>
              ACC <= '0' & ACC(7 downto 1);
              state <= state + 1;
          when 9 =>
              state <= 0;
          end case;
        end if;
    end process;
    Done <= '1' when state = 9 else '0';
end Behavioral;
```

Two process:

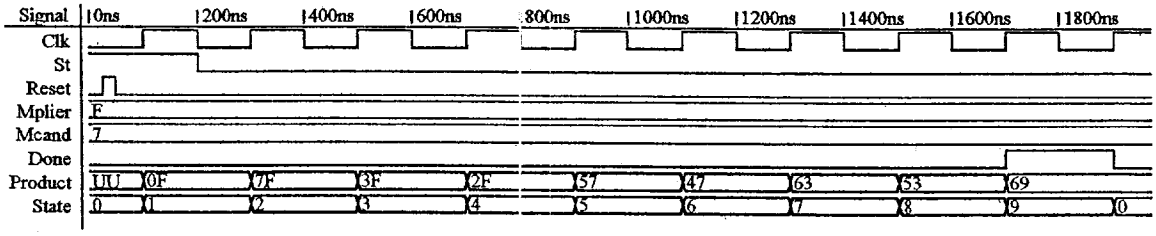
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity multiply is
    port ( Clk, St, Reset : in std_logic;
        Mplier : in std_logic_vector(3 downto 0);
        Mcand : in std_logic_vector(2 downto 0);
        Done: out std_logic;
        Product: out std_logic_vector(6 downto 0));
end multiply;
```

20.E, continued

Two process continued:

```
architecture Behavioral of multiply is
signal state, nextstate: integer range 0 to 9;
signal ACC: std_logic_vector(7 downto 0);
alias M: std_logic is ACC(0);
signal addout : std_logic_vector(3 downto 0);
signal Load, Ad, Sh: std_logic;
signal test: std_logic := '0';
    begin
    Product<= ACC(6 downto 0);
    addout<=('0' & ACC(6 downto 4)) + Mcand ;
    process(state,St,M)
        begin
        Load <= '0'; Ad <= '0'; Sh <= '0'; Done <= '0';
        case state is
        when 0 =>
          if St='1' then Load <= '1' ;nextstate <= 1;
          else nextstate <= 0; end if;
        when 1|3|5|7 =>
            if M = '1' then Ad <= '1';
                nextstate <= state + 1;
            else Sh <='1'; nextstate <= state+ 2; end if;
        when 2|4|6|8 =>
            Sh <= '1'; nextstate <= state + 1;
        when 9 =>
          Done <= '1'; nextstate <= 0;
        end case;
    end process;
    process(Clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif Clk'event and Clk='1' then
          if Load = '1' then
            ACC(7 downto 4) <= "0000";
            ACC(3 downto 0) <= Mplier; end if;
          if Ad ='1' then
            ACC(7 downto 4) <= addout; end if;
          if Sh = '1' then
            ACC <= '0' & ACC(7 downto 1); end if;
          state <= nextstate;
        end if;
    end process;
end Behavioral;
```

Waveform for 1111 X 011 = 1101001:



20.F *One process:*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity 20F_1proc is
    port (Clk, St, Reset : in std_logic;
        Mplier : in std_logic_vector (2 downto 0);
        Mcand : in std_logic_vector (3 downto 0);
        Product : out std_logic_vector (6 downto
        0);
        Done : out std_logic );
end 20F_1proc;
architecture Behave of 20F_1proc is
signal Acc : std_logic_vector (7 downto 0);
signal State : integer range 0 to 7;
signal M : std_logic;
    begin
    M <= Acc (0);
    Product <= Acc (6 downto 0);
    process (Clk, Reset)
    begin
        if Reset = '1' then State <= 0;
        elsif Clk'event and Clk = '1' then
        case State is
        when 0 =>
          if St = '1' then
            Acc (7 downto 3) <= "00000";
            Acc (2 downto 0) <= Mplier;
            State <= 1;
          else State <= 0; end if;
        when 1|3|5 =>
          if M = '1' then
            Acc (7 downto 3) <=
             ('0'&Acc(6 downto 3) + Mcand);
            State <= State + 1;
          else
            Acc (7 downto 0) <=
             '0' & Acc (7 downto 1);
            State <= State + 2;
          end if;
```

20.F, continued

One process continued:

```
        when 2|4|6 =>
          Acc (7 downto 0) <=
            '0' & Acc (7 downto 1);
          State <= State + 1;
        when 7 => State <= 0;
        end case;
    end if;
    end process;
Done <= '1' when State = 7 else '0';
end Behave;
```

Two process:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity 20F_2proc is
    port (Clk, St, Reset : in std_logic;
        Mcand : in std_logic_vector (3 downto 0);
        Mplier : in std_logic_vector (2 downto 0);
        Product : out std_logic_vector (6 downto 0);
        Done : out std_logic );
end 20F_2proc;
architecture Behave2 of 20F_2proc is
signal NextState, State : integer range 0 to 7;
signal Ld, Ad, Sh : std_logic;
signal Acc : std_logic_vector (7 downto 0);
signal Addout : std_logic_vector (4 downto 0);
signal M : std_logic;
    begin
    M <= Acc (0);
    Product <= Acc (6 downto 0);
    Addout <= Acc (7 downto 3) + ('0'&Mcand);
    process (State, St, M)
        begin
        Sh <= '0'; Ad <= '0'; Ld <= '0';
        case State is
        when 0 =>
          if St = '1' then Ld <= '1'; NextState <= 1;
          else NextState <= 0; end if;
        when 1|3|5 =>
          if M = '1' then NextState <= State + 1; Ad <= '1';
          else NextState <= State + 2; Sh <= '1'; end if;
        when 2|4|6 => NextState <= State + 1; Sh <= '1';
        when 7 => NextState <= 0;
        end case;
    end process;
    process (Clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif Clk'event and Clk = '1' then
          State <= NextState;
          if Ld = '1' then
              Acc (7 downto 3) <= "00000";
              Acc (2 downto 0) <= Mplier;
          elsif Ad = '1' then
            Acc (7 downto 3) <= Addout;
          elsif Sh = '1' then
            Acc (7 downto 0) <=
              '0' & Acc (7 downto 1);
          end if;
```

20.F, continued

```
        end if;
    end process;
    Done <= '1' when State = 7 else '0';
end Behave2;
```

20.G *One process:*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity multiplier is
    port(Clk, St : in std_logic;
        Mplier : in std_logic_vector(2 downto 0);
        Mcand : in std_logic_vector(4 downto 0);
        Done : out std_logic;
        Product: out
          std_logic_vector(7 downto 0):="00000000");
end multiplier;
architecture behavioral of multiplier is
signal State: integer range 0 to 7;
signal Acc:
        std_logic_vector(8 downto 0):="000000000";
alias M: std_logic is Acc(0);
    begin
    Product<=Acc(7 downto 0);
    process(Clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif Clk'event and Clk='1' then
          case State is
          when 0 =>
            if St='1' then
              Acc(8 downto 3)<="000000";
              Acc(2 downto 0)<=Mplier;
              State<=1;
            end if;
          when 1|3|5 =>
            if M='1' then
              Acc(8 downto 3)<=
                '0'&Acc(7 downto 3)+Mcand;
              State<=State+1;
            else
              Acc<='0'&Acc(8 downto 1);
              State<=State+2;
            end if;
          when 2|4|6 =>
            Acc<='0'&Acc(8 downto 1);
            State<=State+1;
          when 7 =>
            State<=0;
          end case;
        end if;
    end process;
    Done<='1' when State =7 else '0';
end behavioral;
```

20.G, continued

Two process:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity multiplier2 is
    port (Clk, St : in std_logic;
        Mplier : in std_logic_vector(2 downto 0);
        Mcand : in std_logic_vector(4 downto 0);
        Done : out std_logic;
        Product : out
            std_logic_vector(7 downto 0) := "00000000");
end multiplier2;
architecture Behavioral of multiplier2 is
signal State, NextState : integer range 0 to 7;
signal Acc :
        std_logic_vector(8 downto 0) := "000000000";
alias M : std_logic is Acc(0);
signal Load, Ad, Sh : std_logic;
signal Addout : std_logic_vector(5 downto 0);
    begin
    Product <= Acc(7 downto 0);
    addout <= ('0' & Acc(7 downto 3))+Mcand;
    process(State, St, M)
        begin
        Load <= '0'; Ad <= '0'; Sh <= '0'; Done <= '0';
        case State is
        when 0 =>
        if St = '1' then Load <= '1'; NextState <= 1;
        else NextState <= 0;
        end if;
        when 1|3|5 =>
        if M = '1' then Ad <= '1'; NextState <= State+1;
        else Sh <= '1'; NextState <= State+2;
        end if;
        when 2|4|6 =>
        Sh <= '1'; NextState <= State+1;
        when 7 =>
        Done <= '1'; NextState <= 0;
        end case;
    end process;
    process(Clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif Clk'event and Clk='1' then
        if Load = '1' then Acc(8 downto 3) <= "000000";
        Acc(2 downto 0) <= Mplier; end if;
        if Ad = '1' then Acc(8 downto 3) <= addout; end if;
        if Sh = '1' then Acc <= '0' & Acc(8 downto 1); end if;
        State <= NextState;
        end if;
    end process;
end Behavioral;
```

20.H

One process:

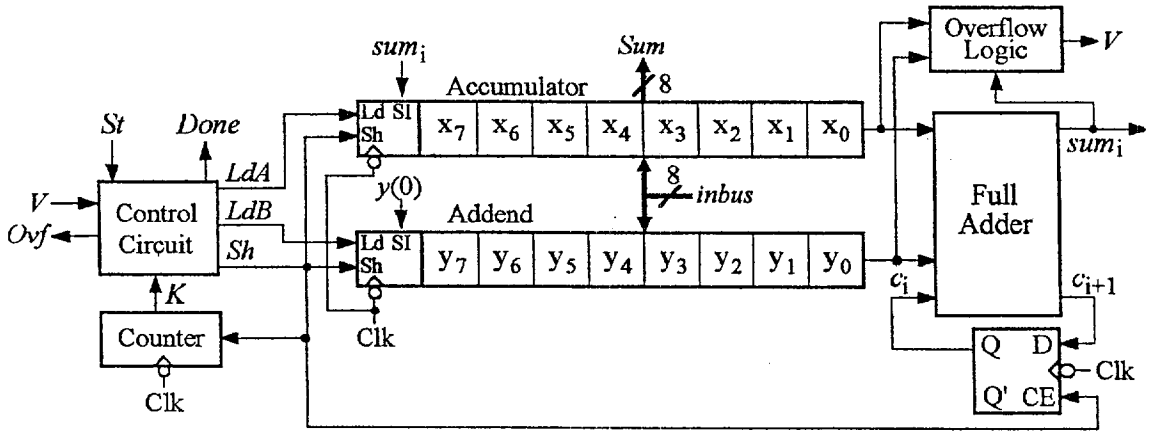
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity multiplier358 is
    port (clk, st, Reset: in std_logic;
        mplier: in std_logic_vector(4 downto 0);
        mcand: in std_logic_vector(2 downto 0);
        product: out std_logic_vector(7 downto
        0);
        done: out std_logic);
end multiplier358;
architecture Behavioral of multiplier358 is
signal state, nextstate: integer range 0 to 11;
signal acc: std_logic_vector(8 downto 0);
alias m: std_logic is acc(0);
    begin
    product <= acc(7 downto 0);
    process(clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif clk'event and clk = '1' then
        case state is
        when 0 =>
        if st = '1' then
        acc(8 downto 5) <= "0000";
        acc(4 downto 0) <= mplier;
        state <= 1;
        else
        state <= 0;
        end if;
        when 1|3|5|7|9 =>
        if m = '1' then
        acc(8 downto 5) <=
          ('0'&acc(7 downto 5)) + mcand;
        state <= state + 1;
        else
        acc <= '0'&acc(8 downto 1);
        state <= state + 2;
        end if;
        when 2|4|6|8|10 =>
        acc <= '0'&acc(8 downto 1);
         state <= state + 1;
        when 11 =>
        done <= '1'; state <= 0;
        end case;
        end if;
    end process;
end Behavioral;
```

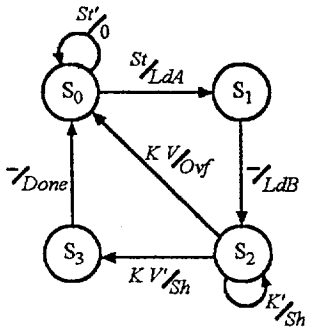
20.H, continued

Two process:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity multiplier358 is
    port (clk, st, Reset: in std_logic;
          mplier: in std_logic_vector(4 downto 0);
          mcand: in std_logic_vector(2 downto 0);
          product: out std_logic_vector(7 downto 0);
          done: out std_logic);
end multiplier358;
architecture Behavioral of multiplier358 is
signal state, nextstate: integer range 0 to 11
signal acc: std_logic_vector(8 downto 0);
alias m: std_logic is acc(0);
signal addout: std_logic_vector(3 downto 0);
signal ld, ad, sh: std_logic;
    begin
    product <= acc(7 downto 0);
    addout <= ('0'&acc(7 downto 5)) + mcand;
    process(state, st, m)
        begin
        ld <= '0'; ad <= '0'; sh <= '0';
        case state is
        when 0 =>
        if st = '1' then ld <= '1'; nextstate <= 1;
        else nextstate <= 0; end if;
        when 1|3|5|7|9 =>
        if m = '1' then ad <= '1'; nextstate <= state + 1;
        else sh <= '1'; nextstate <= state + 2; end if;
        when 2|4|6|8|10 =>
        sh <= '1'; nextstate <= state + 1;
        when 11 =>
        done <= '1'; nextstate <= 0;
        end case;
    end process;
    process(clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif clk'event and clk = '1' then
        if ld = '1' then
        acc(8 downto 5) <= "0000";
        acc(4 downto 0) <= mplier;
        end if;
        if ad = '1' then
          acc(8 downto 5) <= addout; end if;
        if sh = '1' then
          acc <= '0'&acc(8 downto 1); end if;
        state <= nextstate;
        end if;
    end process;
end Behavioral;
```

20.I





One process:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity lab20l_1p is
    port ( clk, st, Reset : in std_logic;
           inbus : in std_logic_vector(7 downto 0);
           sum : out std_logic_vector(7 downto 0);
           done, ovf : out std_logic);
end lab20l_1p;
architecture Behavioral of lab20l_1p is
-- X = Acc
signal X,Y : std_logic_vector(7 downto 0);
signal Ci,Ciplus,Sumi,K,V,cat1,cat2 : std_logic;
signal State : integer range 0 to 3;
```

20.I, continued

One process continued:

```
signal Count : std_logic_vector(2 downto 0);
    begin
    cat1 <= '0'; -- Needed to use the 7 segment display
    cat2 <= '1'; -- to provide a 9th LED
    Sumi <= X(0) xor Y(0) xor Ci; -- 1-bit adder
    Ciplus <= (Ci and X(0)) or
      (Ci and Y(0)) or (X(0) and Y(0));
    sum <= X;
    done <= '1' when state = 3 else '0';
    V <= (sumi and not Y(0) and not X(0))
      or (not sumi and Y(0) and X(0));
    K <= '1' when Count = 7 else '0';
    -- Control state machine
    process (clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif clk'event and clk='0' then
          case State is
          when 0 =>
            if St = '1' then
              X <= inbus; -- load accumulator from bus
              Ci <='0'; -- Clear the carry bit
              Count <= "000";
              State <= 1; end if;
          when 1 =>
            Y <= inbus;
            State <= 2;
          when 2 =>
            if K = '0' or V = '0' then -- shift
              X <= Sumi & X(7 downto 1);
              Y <= Y(0) & Y(7 downto 1);
              Ci <= Ciplus;
              Count <= Count + 1; end if;
            if K = '0' then State <= 2;
            elsif V = '0' then State <= 3;
            else State <= 0; end if;
          when 3 =>
            State <= 0;
          end case;
        end if;
    end process;
    Ovf <= '1' when V = '1' and State = 2 and K = '1'
          else '0';
end Behavioral;
```

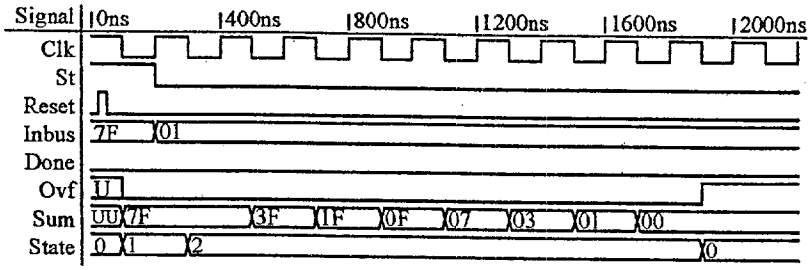
Two process:

```
ibrary IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity lab20I_2p is
    port ( clk, st, Reset : in std_logic;
        inbus : in std_logic_vector(7 downto 0);
        sum : out std_logic_vector(7 downto 0);
        ovf, done : out std_logic);
end lab20I_2p;
```

20.I, continued

```
architecture Behavioral of lab20I_2p is
-- X = Accumulator
signal X,Y : std_logic_vector(7 downto 0);
signal Sh,Ci,Ciplus,Sumi,K,LdA,LdB,V,cat1,cat2 : std_logic;
signal State, NextState : integer range 0 to 3;
signal count : std_logic_vector(2 downto 0);
    begin
    cat1 <= '0'; -- Needed to use the 7 segment display
    cat2 <= '1'; -- To provide a 9th LED
    Sumi <= X(0) xor Y(0) xor Ci; -- 1-bit adder
    Ciplus <= (Ci and X(0))
      or (Ci and Y(0)) or (X(0) and Y(0));
    sum <= X;
    V <= (Sumi and not Y(0) and not X(0))
      or (not Sumi and Y(0) and X(0));
    K <= '1' when count=7 else '0';-- Control state machine
    process (State,St,K,V)
        begin
        LdA <= '0'; LdB <= '0'; Sh <= '0';
        Ovf <= '0'; Done <= '0';
        case State is
            when 0 =>
                if St = '1' then
                    LdA <= '1'; -- load accumulator from bus
                    NextState <= 1;
                else
                    NextState <= 0; end if;
            when 1 =>
                LdB <= '1'; NextState <= 2;
            when 2 =>
                if K = '0' then Sh <= '1'; else
                    if V = '0' then Sh <= '1'; NextState <= 3;
                    else ovf <= '1'; NextState <= 0; end if;
                end if;
            when 3 =>
                done <= '1'; NextState <= 0;
        end case;
    end process;
    process(clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif clk'event and clk='0' then
            State <= NextState;
            if LdA = '1' then
                X<=inbus;
                count <= "000"; -- Initialize counter
                Ci <= '0'; -- Initialize carry
            elsif LdB = '1' then Y<=inbus;
            elsif Sh='1' then
                -- Shifting both ACC and Addend
                X <= Sumi & X(7 downto 1);
                Y <= Y(0) & Y(7 downto 1);
                Ci <= Ciplus; -- store carry
                count <= count + 1; end if;
        end if;
    end process;
end Behavioral;
```

20.I, continued Waveform for 01111111 + 00000001 = 10000000 (overflow):



20.J Same as solution for 20.I except change all 7's to 6's.

20.K ***One process:***

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity lab20K_1p is
    port ( clk, st, Reset : in std_logic;
        inbus : in std_logic_vector(7 downto 0);
        diff : out std_logic_vector(7 downto 0);
        done, ovf : out std_logic);
end lab20K_1p;
architecture Behavioral of lab20K_1p is
signal X,Y : std_logic_vector(7 downto 0); -- X =
Accumulator
signal Bi,Biplus,diffi,K,V,cat1,cat2 : std_logic;
signal State : integer range 0 to 3;
signal Count : std_logic_vector(2 downto 0);
    begin
    cat1 <= '0'; -- Needed to use the 7 segment display
    cat2 <= '1'; -- To provide a 9th LED
    diffi <= X(0) xor Y(0) xor Bi; -- 1-bit adder
    Biplus <= (Bi and not X(0)) or (Bi and Y(0)) or (not
    X(0) and Y(0));
    diff <= X;
    done <= '1' when state = 3 else '0';
    V <= (diffi and Y(0) and not X(0)) or (not diffi and
    not Y(0) and X(0));
    K <= '1' when Count = 7 else '0';
    -- Control state machine
    process (clk, Reset)
        begin
        if Reset = '1' then State <= 0;
        elsif clk'event and clk='0' then
        case State is
        when 0 =>
        if St = '1' then
        X <= inbus; -- load accumulator from bus
        Bi <='0'; -- Clear the borrow bit
        Count <= "000"; State <= 1;
        end if;
        when 1 =>
        Y <= inbus; State <= 2;
        when 2 =>
            if K = '0' or V = '0' then -- shift
                X <= Diffi & X(7 downto 1);
                Y <= Y(0) & Y(7 downto 1);
                Bi <= Biplus;
                Count <= Count + 1; end if;
            if K = '0' then State <= 2;
            elsif V = '0' then State <= 3;
            else State <= 0; end if;
        when 3 =>
        State <= 0;
        end case;
        end if;
    end process;
    Ovf <= '1' when V = '1' and State = 2 and K = '1'
        else '0';
end Behavioral;
```

Two process:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity lab20K_2p is
    port ( clk, st, Reset : in std_logic;
        inbus : in std_logic_vector(7 downto 0);
        diff : out std_logic_vector(7 downto 0);
        ovf, done : out std_logic);
end lab20K_2p;
architecture Behavioral of lab20K_2p is
signal X,Y : std_logic_vector(7 downto 0); -- X = Accumulator
signal Sh,Bi,Biplus,diffi,K,LdA,LdB,V : std_logic;
signal State, NextState : integer range 0 to 3;
signal count : std_logic_vector(2 downto 0);
signal cat1, cat2 : std_logic;
    begin
    cat1 <= '0'; -- Needed to use the 7 segment display
    cat2 <= '1'; -- To provide a 9th LED
    diffi <= X(0) xor Y(0) xor Bi; -- 1-bit adder
    Biplus <= (Bi and not X(0)) or (Bi and Y(0) and X(0)) or
    (not X(0) and Y(0));
    diff <= X;
    V <= (diffi and Y(0) and not X(0)) or (not diffi and not
    Y(0) and X(0));
    K <= '1' when count=7 else '0';
```

20.K, continued

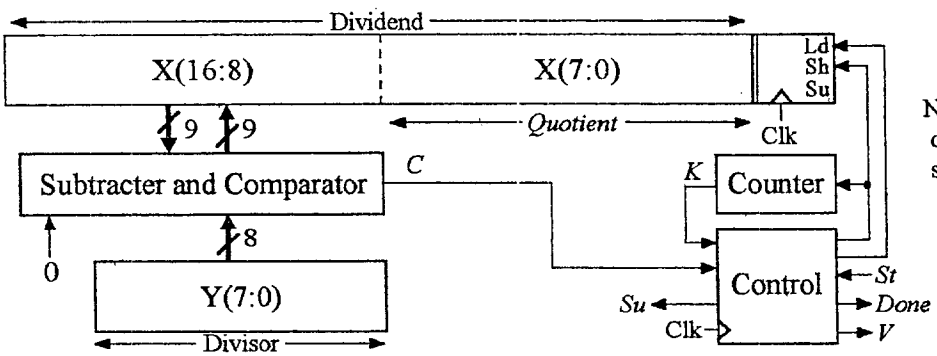
Two process continued:

```
-- Control state machine
process (State,St,K,V)
    begin
    LdA <= '0'; LdB <= '0'; Sh <= '0';
    Ovf <= '0'; Done <= '0';
    case State is
      when 0 =>
        if St = '1' then
          LdA <= '1'; -- load accumulator from bus
          NextState <= 1;
        else
          NextState <= 0; end if;
      when 1 =>
        LdB <= '1';
        NextState <= 2;
      when 2 =>
        if K = '0' then Sh <= '1'; else
        if V = '0' then Sh <= '1'; NextState <= 3;
        else ovf <= '1'; NextState <= 0; end if;
        end if;
       when 3 =>
          done <= '1'; NextState <= 0;
     end case;
  end process;
```

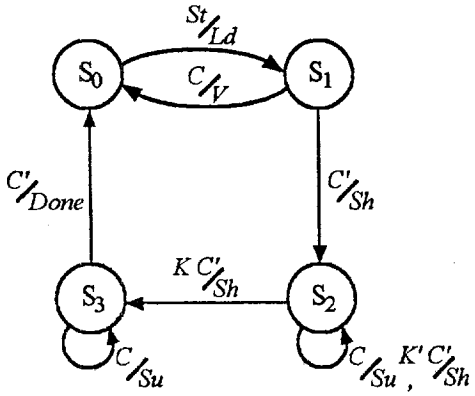
```
process(clk, Reset)
    begin
    if Reset = '1' then State <= 0;
    elsif clk'event and clk='0' then
      State <= NextState;
      if LdA = '1' then
        X<=inbus;
        count <= "000"; -- Initialize counter
        Bi <= '0'; -- Initialize borrow
      elsif LdB = '1' then Y<=inbus;
      elsif Sh='1' then
        -- Shifting both ACC and Addend
        X <= diffi & X(7 downto 1);
        Y <= Y(0) & Y(7 downto 1);
        Bi <= Biplus;  -- store borrow
        count <= count + 1;
      end if;
    end if;
  end process;
end Behavioral;
```

20.L Same as solution to 20.K except change all 7's to 6's.

20.M



Note: *Su* loads subtracter output into X(16:8) and sets X_0 to 1.



Two process:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Div16 is
    port (CLK, St. Reset : in std_logic;
        Dvend : in std_logic_vector(15 downto 0);
        Dvsor : in std_logic_vector(7 downto 0);
        Quotient : out std_logic_vector(7 downto 0);
        Remainder : out std_logic_vector(8 downto 0);
        V, Done : out std_logic);
end Div16;
architecture Behavioral of Div16 is
signal state, nextstate : integer range 0 to 2;
signal X : std_logic_vector(16 downto 0);  -- dividend register
signal Y : std_logic_vector(8 downto 0);  -- divisor register
signal C, Sh, Su, Load, K : std_logic;
signal Count : std_logic_vector(2 downto 0);
signal subout : std_logic_vector(8 downto 0);
```

20.M, continued

Two process continued:

```
begin
Quotient <= X(7 downto 0);
Remainder <= X(16 downto 8);
K <= '1' when Count = 7 else '0';
Subout <= X(16 downto 8) - Y;
C <= not Subout(8);
process (state, St, C, K)
    begin
    Load <= '0'; Sh <= '0'; Su <= '0';
    V <= '0'; Done <= '0';
    case State is
      when 0 => if St = '1' then Load <= '1';
      nextstate <= 1; end if;
    when 1 =>
      if C = '1' then V <= '1'; nextstate <= 0;
      else Sh <= '1'; nextstate <= 2; end if;
    when 2 =>
      if C = '1' then Su <= '1'; nextstate <= 2;
      else
        Sh <= '1';
        if K = '0' then nextstate <= 2;
        else nextstate <= 3; end if;
      end if;
    when 3 =>
      if C = '1' then Su <= '1'; nextstate <= 3;
      else Done <= '1'; nextstate <= 0; end if;
    end case;
 end process;
 process(clk, Reset)
    begin
    if Reset = '1' then State <= 0;
    elsif clk'event and clk = '1' then
      if Load = '1' then
        Count <= "000"; X <= '0'&Dvend; Y <= '0'&Dvsor; end if;
      if Sh = '1' then
        X <= X(15 downto 0) & '0'; Count <= Count + 1; end if;
      if Su = '1' then
        X(16 downto 8) <= Subout; X(0) <= '1'; end if;
      state <= nextstate;
      end if;
 end process;
end Behavioral;
```

IV. SAMPLE UNIT TESTS

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 1 - FORM A

Please answer the following questions carefully without any reference to outside notes. Check your answers before you turn in your paper since an essentially perfect score is required before you go on to the next unit.

1. Multiply in binary: 11011
   × 1011

2. Convert $(7813.400)_9$ to hexadecimal (base 16). Carry out your answer to 3 places past the "point".

3. Add the following numbers in binary. Use 2's complement to represent negative numbers. Use a word length of 5 bits (including sign). Indicate if an overflow occurs.

   (a) 10 + 6 (b) −12 + (−14) (c) −3 + (−12) (d) 12 + (−15)

4. If possible, construct an 1-4-2-3 weighted code for decimal digits. If not possible, explain why. If possible, express 673 decimal in 1-4-2-3 code.

5. Why are binary numbers used in digital computers?

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 2 - FORM A
(Issue a theorem sheet with this test)

1. Using a truth table, prove that
   $(XY)\,(X' + Z') = XYZ'$

2. Simplify each of the following expressions
   $(A'C + B' + DF')[A + C' + B' + DF'] =$

   $(AC + B')(G + F + DE)\,(AC + B')' =$

   $(X' + Y'Z' + 1)\,(X + Y)\,(X + Z) =$

   $(A + B'C + 1)\,(B' + C' + BCF) =$

   $PQ\,(M'N'P' + NR + R'S')\,R' =$

3. Factor as much as possible: (Your answer should be in product-of-sums form)
   $Z + XYQ + XYP =$

4. Find D' if $D = (F' + G'H)E$

5. Illustrate the following theorem using a circuit of switches.

   $X+YZ = (X+Y)(X+Z)$

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 3 - FORM A

1. Is the following statement always true? Justify your answer.

   If $ab' + [b + b'(a + bc)]' = [a + a'(ac + b)]\,(a + b')$, then $a = b'$

2. Factor as much as possible to obtain a product of sums:

   $WYZ + W'UV + W'Y' + WV'$

3. Each time you apply a theorem, mark your paper clearly to indicate which terms were used to add or delete other terms, or indicate which terms were combined. Simplify algebraically: (answer should be a sum of *3 terms*)

   $(A + B' + C' + D')\,(B + C + D)\,(B' + C' + D')\,(A' + B + C + D)$

4. Simplify to obtain a sum of 3 terms:
   $(A \equiv B')\,(CD \oplus B') + ABCD$

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 4 - FORM A

1. A switching circuit has three inputs (A, B, C) and one output (Z). If $A = 0$, the output Z is the exclusive-OR of B and C. If $A = 1$, the output is the equivalence of B and C.
   a) Find the truth table for Z.
   b) Write the minterm expansion for Z in decimal form and in terms of A, B, C.
   c) Write the maxterm expansion for Z in decimal form and in terms of A, B, C.

2. Without using a truth table, find (a) the minterm and maxterm expansions for F in algebraic and decimal forms, and (b) the minterm and maxterm expansions for F' in decimal form.

   $F(A,B,C) = A + B'C' + BC$

3. Write an equation for the following sequence (use only 3 variables for the right-hand side) : The overflow indicator V will turn on iff X is negative, Y is positive and D is positive, or if X is positive, Y is negative and D is negative.

4. Design a 4-bit adder/subtracter using four full adders and four exclusive-OR gates. When $Su = 1$, the circuit should output $A - B$, otherwise it should output $A + B$. Remember that $B' = B \oplus 1$. (Assume that negative numbers are represented in 2's complement).

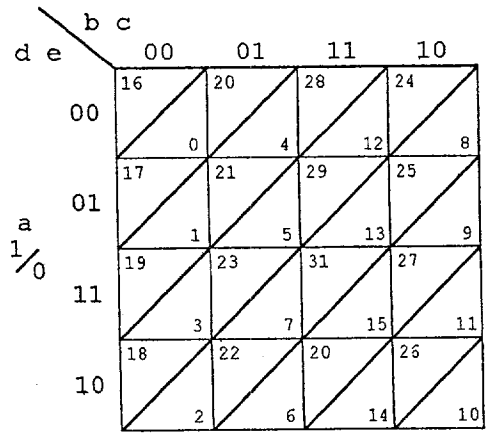
NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 5 - FORM A

1. $F(a, b, c, d, e) = \Sigma\, m(0, 1, 2, 6, 7, 8, 16, 17, 19, 20, 25, 26, 29, 31) + \Sigma\, d(3, 5, 18, 27)$

   (a) Find the essential prime implicants of F and indicate the minterm which makes each one essential.

   (b) Find a minimum sum-of-products expression for F.



2. $F(a, b, c, d) = \Sigma\, m(7, 11, 12, 14) + \Sigma\, d(0, 10)$

   (a) Find all of the prime implicants of F'.
   (b) Find a minimum product-of-sums for F.
   Note: F' is specified in (a) and F in (b).

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 6 - FORM A

1. Find two different minimum sum-of-products expresstions for F using the Quine-McCluskey procedure:

   $F = \sum m(2, 4, 5, 10, 11, 13, 14, 15) + \sum d(1, 8)$

   Circle the essential prime implicants in your answer and specify the minterm(s) that make each one essential.

2. What theorem is used when combining terms in the Quine-McCluskey procedure?

   Why is is unnecessary to compare two terms which have the same number of 1's (that is, why is it unnecessary to compare two terms in the same group)?

3. Find a minimum sum of products for G using a 4-variable Karnaugh map.
   $G = A'BD' + A'C'D' + A'BDE + ABD'F + AC'D'F$
   ($A'B'CD'$ is a don't care term)

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 7 - FORM A

1. Find a minimum *3-level* **NOR** circuit to realize
   $f(A,B,C,D) = \sum m(0,2,4,6,7,9,11,12,13,15)$ (5 gates)
   Do *not* use gate symbols with *input* bubbles in your final answer.

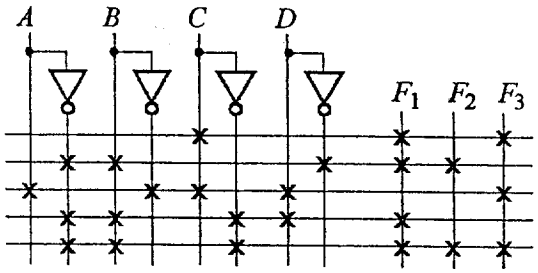
2. (a) Design a *minimum* 2-level **NOR-NOR** circuit to realize
   $f(X,Y,Z,W) = \Sigma\, m(5,13,15)$
   (b) Convert the circuit from part (a) to a 2-level **NAND-AND** circuit.
   Do *not* use gate symbols with *input* bubbles in your final answer.

3. Find a *minimum* 2-level **NAND** gate circuit to simultaneously realize
   $F_1(A,B,C,D) = \Sigma\, m(1,3,5,7,8,9,13)$
   $F_2(A,B,C,D) = \Sigma\, m(1,3,5,7,8,10,13)$
   (Hint: Minimum solution has 6 gates)

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 9 - FORM A

1. (a) Given the following PLA connection diagram, specify the PLA contents in table form.



(b) Give the logic equations realized by the PLA
(c) What size ROM would be required to realize the equations?

2. (a) Write the logic equation for the output of an 8-to-1 MUX with control inputs B, C, D.
(b) Implement a 4-to-1 MUX having control inputs A, B using gates.
(c) Implement a 4-to-1 MUX using a 2-to-4 decoder and tri-state buffers to select one of the MUX inputs.

3. $Z(a, b, c, d, e) = a'b'cd' + ac'de' + a'bd'e + bde$
(a) Expand the above expression about the variable a
(b) Use the expansion in part (a) to realize the function using two 4-variable LUTs and a 2-to-1 MUX. Specify the LUT inputs.
(c) Show the truth table for one of the LUTs.

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT R1 - FORM A
TIME LIMIT 60 MINUTES

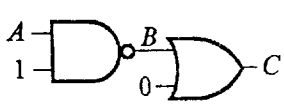
1. For the following function, find the simplest sum-of-products form algebraically. (Hint: Minimum solution has three terms)
   $F = (A + B' + D)\,(A' + B + D)\,(C + D)\,(C' + D')$

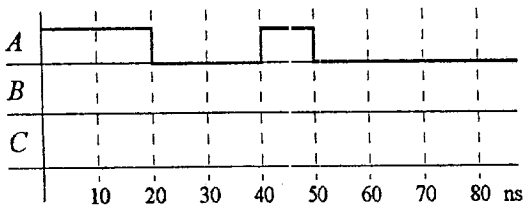
2. Find a minimum 2-level multiple output NOR-NOR realization for the two functions given by the PLA table below. (Hint: Minimum solution has seven gates)

w x y z	f_1 f_2
0 1 1 -	1 0
1 - 0 0	1 1
1 0 1 1	1 1
1 0 - 0	1 1
- 1 - 0	0 1
1 - 1 -	0 1
0 0 - 1	0 0

3. Given $F = A'B'C + B'C'D' + A'C'D + A'B'D + B'C\,D'$
   (a) Find the maxterm expansion in decimal notation.
   (b) Find a minimum 3-level circuit of AND and OR gates (no particular order of gates implied). The minimum solution has 4 gates and 8 inputs.

4. B and C are initially zero. Assume 10 ns gate delays and complete the timing diagram below.

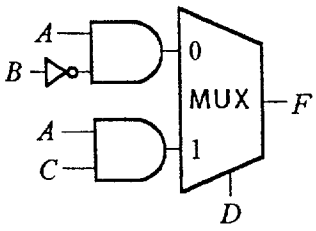




NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 10 - FORM A

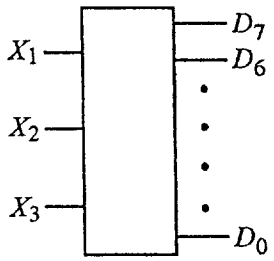
1. Write a conditional signal assignment statement that represents the following circuit.



2. Draw a circuit that implements the following VHDL code:

```
A <= B or not C and D;
D <= not D after 5 ns;
```

3. A 3-to-8 decoder can be represented by a truth table with inputs $X_1 X_2 X_3$ and outputs $D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0$. Write a complete VHDL module that implements the decoder using an 8 words × 8 bits ROM. Include the array type declaration and the constant declaration that defines the contents of the ROM



4. Evaluate the following expression for A = "010" and B = "110":
(TEST is of type Boolean)

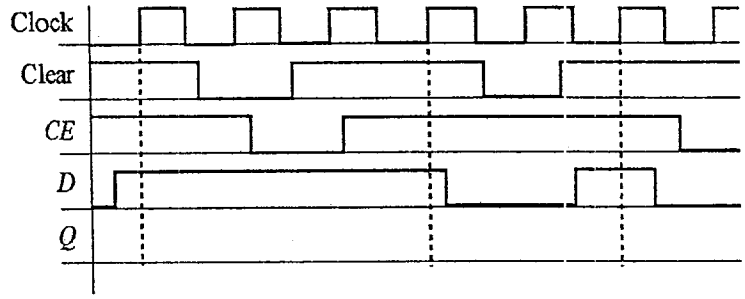
```
TEST <= (A & B or not (B & A)) < (B & A and not A & A)
```

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 11 - FORM A

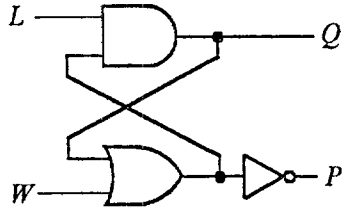
1. An M-N flip-flop behaves as follows:
   If $MN = 11$, the flip-flop is set to $Q = 0$.
   If $MN = 01$, the flip-flop is set to $Q = 1$.
   If $MN = 10$, no change of state occurs.
   If $MN = 00$, the flip-flop changes states.
   Derive the characteristic (next state) equation for this flip-flop.

2. Complete the following timing diagram for a rising edge triggered D-CE flip-flop with a clear input.



3. For the latch given below,

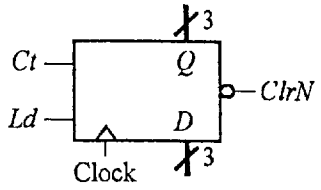
   a) If $L = W = 0$, what value will P and Q assume?
   b) If L is now changed to 1, what value will P and Q assume?
   c) If W is now changed to 1, what value will P and Q assume?
   d) If W is now changed to 0, what value will P and Q assume?
   e) What restriction must be placed on L and W so that Q will always equal P'?



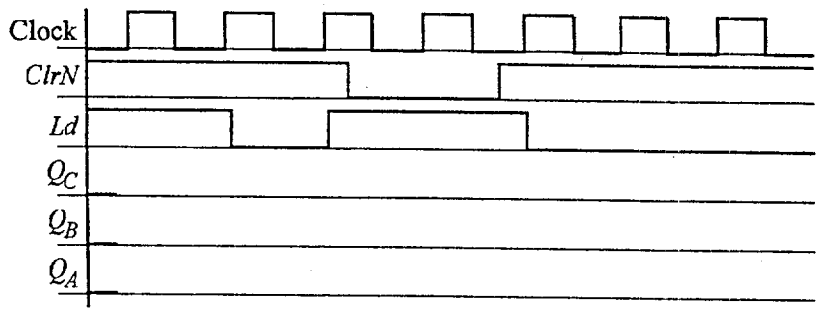
NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 12 - FORM A

1. Consider the following Loadable Counter with an Asynchronous Clear. Assuming that Count (*Ct*) is 1 throughout, and that $D_C D_B D_A$ are 010, complete the timing diagram.



ClrN	Ld	Ct	C+	B+	A+
0	X	X	0	0	0
1	1	X	D_C	D_B	D_A
1	0	0	C	B	A
1	0	1	Present State + 1		



2. A Y-Z flip-flop behaves as follows:
   If $YZ = 01$, the flip-flop does not change state.
   If $YZ = 11$, the flip-flop is set to $Q = 1$.
   If $YZ = 10$, the flip-flop changes state.
   The input combination $YZ = 00$ is not allowed.
   a) Complete the table below using don't cares where possible.
   b) Realize the following next state equation for Q using a YZ flip-flop. $Q^+ = Q'AB' + A'B + QB$. Find equations for Y and Z.

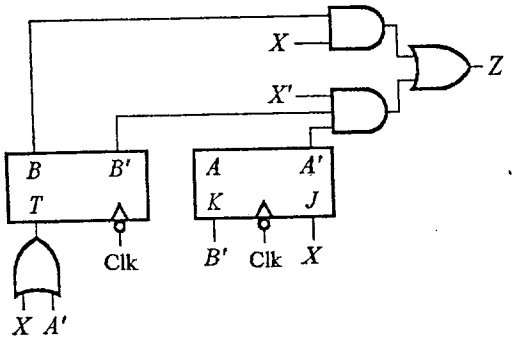
Q	Q+	Y	Z
0	0		
0	1		
1	0		
1	1		

3. A 3-bit counter uses three different types of flip-flops as follows:
   Flip-flop *A* is a D flip-flop.
   Flip-flop *B* is a T flip-flop.
   Flip-flop *C* is a J-K flip-flop.
   Design the counter so that it counts in the following sequence:
   ABC = 000, 010, 100, 101, 111, 110, 011, 000, ...

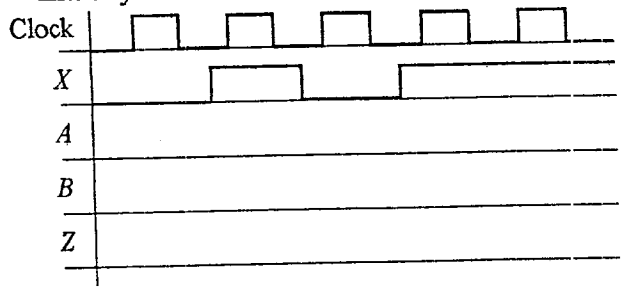
NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 13 - FORM A

1. a) For the given circuit, construct a transition table and a state graph.



b) Complete the following timing chart. Indicate at what times Z has the correct value and specify the correct output sequence.
Initially $A = B = 0$.



2. Draw a general model of a Moore circuit with m inputs, n outputs, and k D flip-flops.

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 14 - FORM A

1. A Mealy sequential circuit has one input and two outputs. When the fourth input is received, the circuit outputs $Z_1 = 1$ if the majority of the inputs are 0's, $Z_2 = 1$ if the majority are 1's, and $Z_1 = Z_2 = 1$ if the number of 0's and 1's is the same. The circuit then resets. Find a state graph and state table. Also specify what each state in your graph corresponds to. (Any solution with 12 or fewer states is acceptable). *Hint*: The outputs depend only on the total number of 0's and 1's received and not on the sequence. Thus 0 1 and 1 0 should go to the same state.

2. A Moore circuit has one input and one output. The output is initially zero. The output is the same as the input just received, unless the input sequence 1011 occurs, in which case the output stays 1 thereafter. Derive the state graph and table for the circuit.

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 15 - FORM A

1. State tables for circuits N_1 and N_2 are given below. Are the circuits equivalent? Justify your answer.

N_1	0	1	0	1
S_0	S_3	S_1	0	0
S_1	S_2	S_1	1	0
S_2	S_0	S_0	0	0
S_3	S_4	S_1	0	0
S_4	S_3	S_1	0	0

N_2	0	1	0	1
A	C	D	0	0
B	A	A	0	0
C	C	D	0	0
D	B	D	1	0

2. Reduce the following Moore state table to a minimum number of states.

	X_1X_2				
	00	01	11	10	Z
A	C	D	D	F	0
B	D	F	E	A	1
C	A	B	B	A	0
D	B	C	E	F	1
E	E	A	F	C	1
F	F	B	D	C	0

3. (a) The following state table is to be realized using D flip-flops. Make a suitable state assignment which will lead to an economical circuit. *Assign 000 to S_0 and 010 to S_2.* Show how you arrived at your state assignment.
   (b) For the assignment obtained in (a) derive the input equations for the flip-flops and the output equation.

	0	1	0	1
S_0	S_1	S_5	0	0
S_1	S_1	S_4	1	0
S_2	S_1	S_2	0	1
S_3	S_0	S_2	0	1
S_4	S_2	S_3	1	1
S_5	S_1	S_4	1	1

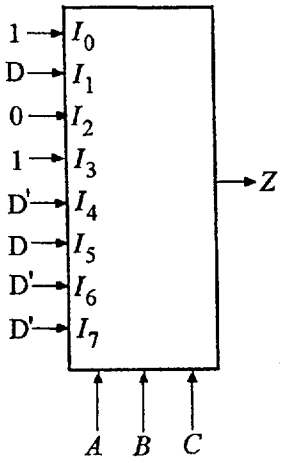
NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 17 - FORM A

1. Draw a diagram to show the hardware that the following VHDL code represents. Use a D-CE flip-flop and gates.

```
if CLK'event and CLK = '1' then
    if LDA = '1' then C <= A or B;
    elsif LDB = '1' then C <= B;
    end if;
end if;
```

2. Write a VHDL process using a case statement that represents the following MUX:



3. Write a VHDL code module that implements the following state table. Assume that all state changes occur on the rising edge of the clock. Use two processes. Use a case statement with if-then-else statements.

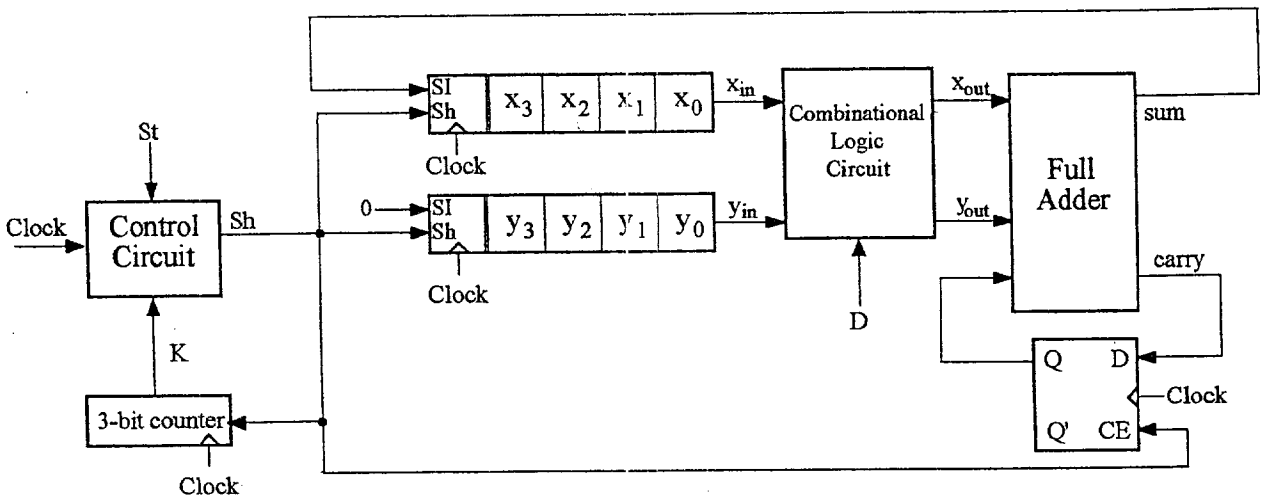
Present State	Next State		Output
	X = 0	X = 1	
S0	S0	S2	0
S1	S0	S2	1
S2	S2	S3	1
S3	S3	S1	0

4. Write a VHDL process to represent a 4-bit counter that will increment on every falling edge of the clock. The counter has an asynchronous, active low reset that overrides the clock. Specify the type of the signal that represents the counter output.

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 18 - FORM A

1. This problem concerns the design of a serial adder/subtracter with accumulator for 4-bit numbers as shown below.



The combinational logic circuit has an input D which determines whether addition or subtraction will take place. When $D = 1$, the contents of registers X and Y are added together. When $D = 0$, the contents of registers X and Y are subtracted (i.e., $X - Y$). The control circuit has one output (Sh) and two inputs—a start signal St and a counter output signal K. The counter counts the number of shifts, and outputs a signal of $K = 1$ when it is 111 (after 7 shifts). Assume that D is not changed during the operation of the adder/subtracter. Assume negative numbers are represented in 1's complement. Subtraction should be done by adding the 1's complement. Note that 8 shifts are required: 4 to do the initial addition and 4 more to add on the end-around carry.

(a) Construct a state graph for the control circuit (3 states). The operation should begin when St is set to 1. St may be 1 for one clock cycle, or it might remain 1 for many clock cycles, even after the addition is completed. Make sure that the circuit does not begin again after it is finished! The start signal must be changed back to 0 to reset the control circuit.
(b) Design the control unit using D flip-flops and any kinds of gates.
(c) Design the combinational logic circuit.

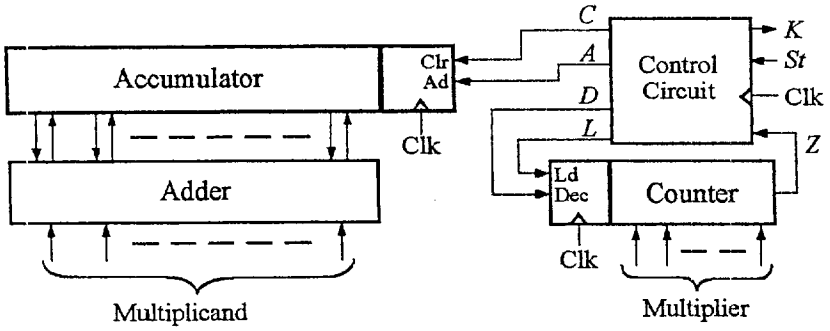
2. (a) Draw a block diagram for a parallel divider which is capable of dividing a positive 8-bit binary number by a positive 4-bit binary number to give a 4-bit quotient. Use a 9-bit dividend register, a 4-bit divisor register, a subtracter-comparator block and a control block.
(b) Show the contents of the dividend register and the value of C at each clock time when 49 is divided by 6. (C is the comparator output and should have a value of $C = 1$ if subtraction can take place without a negative result).
(c) Under what conditions will overflow take place?

NAME_____ PASSWORD_____ DATE_____
(signature)

READINESS TEST - UNIT 19 - FORM A

1. This problem concerns the design of a digital system which multiplies two binary numbers by the repeated addition method. The block diagram of the system is shown below. When a start signal (*St*) is set to 1, the accumulator is cleared and the multiplier is loaded into a counter. The multiplier is then added to the accumulator and the counter is decremented until the multiplication is complete. The control then goes to a stop state and turns on a completion signal (*K*). The circuit then resets when *St* is changed back to 0. Draw an SM chart for the control circuit which has 3 states. Control signals are defined as follows:

C = clear accumulator
L = load multiplier
D = decrement counter
A = add multiplicand to accumulator
Z = 1 when the count is 0



2 (a) Realize the following SM chart using a PLA and D flip-flops that trigger on the falling edge of the clock pulse. Draw a block diagram and give the PLA table. (Do not simplify the equations.)
(b) Complete the timing diagram.