Kohavi Solution Manual pdf

Basic electrical engineering (Delhi Technological University)

Solutions for the End-of-the-Chapter Problems in Switching and Finite Automata Theory, 3rd Ed.

Zvi Kohavi and Niraj K. Jha

Chapter 1

1.2.
(a) $(16)_{10} = (100)_4$
(b) $(292)_{10} = (1204)_6$

1.4.
(a) Number systems with base $b \geq 7$.
(b) Base $b = 8$.

1.5. The missing number is $(31)_5$. The series of integers represents number $(16)_{10}$ in different number systems.

1.7.
(a) In a positively weighted code, the only way to represent decimal integer 1 is by a 1, and the only way to represent decimal 2 is either by a 2 or by a sum of 1 + 1. Clearly, decimal 9 can be expressed only if the sum of the weights is equal to or larger than 9.
(b)

5211*	5321
4311*	6321
5311	7321
6311	4421
4221*	5421
5221	6421
6221	7421
3321*	8421
4321	

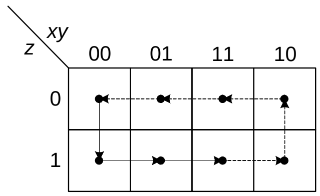
* denotes a self-complementing code. The above list exhausts all the combinations of positive weights that can be a basis for a code.

1.8.
(a) If the sum of the weights is larger than 9, then the complement of zero will be $w_1 + w_2 + w_3 + w_4 > 9$. If the sum is smaller than 9, then it is not a valid code.

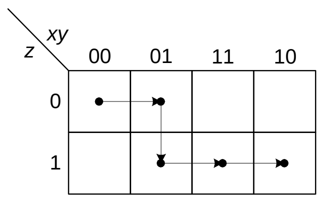
(b) $751-4$; $832-4$; $652-4$.

1.9.

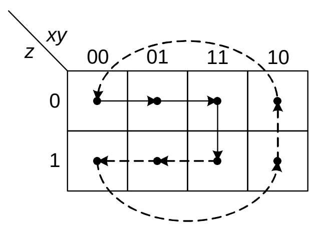
(a) From the map to below, it is evident that the code can be completed by adding the sequence of code words: 101, 100, 110, 010.



(b) This code cannot be completed, since 001 is not adjacent to either 100 or 110.



(c) This code can be completed by adding the sequence of code words: 011, 001, 101, 100.



(d) This code can be completed by adding code words: 1110, 1100, 1000, 1001, 0001, 0011, 0111, 0110, 0010.

1.13.

(a)

(i) A, C, and D detect single errors.

(ii) C and D detect double errors.

(iii) A and D detect triple errors.

(iv) C and D correct single errors.

(v) None of the codes can correct double errors.

(vi) D corrects single and detects double errors.

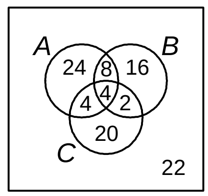
(b) Four words: 1101, 0111, 1011, 1110. This set is unique.

Chapter 2

2.1.

A: 40%	A,B: 12%	A,B,C: 4%
B: 30%	A,C: 8%	
C: 30%	B,C: 6%	

22% receive no credit.



2.2.

	Reflexive	Symmetric	Antisymmetric	Transitive	Relation name
(a)	yes	yes	no	yes	equivalence
(b)	yes	yes	no	no	compatibility
(c)	yes	yes	no	yes	equivalence
(d)	yes	no	yes*	yes	partial ordering
(e)	yes	no	no	yes	–
(f)	yes	no	no	yes	–

*depends on the interpretation of congruence.

2.4.

(a) $\pi_1 + \pi_2 = \{\overline{a,b,c,g,h,i,j,k}; \overline{d,e,f}\}$, $\quad \pi_1 \cdot \pi_2 = \{\overline{a,b}; \overline{c}; \overline{d,e}; \overline{f}; \overline{g,h}; \overline{i}; \overline{j,k}\}$

(b) $\pi_1 + \pi_3 = \pi_3$, $\pi_1 \cdot \pi_3 = \pi_1$

(c) $\pi_1 < \{\overline{a,b,c}; \overline{d,e}; \overline{f}; \overline{g,h,i,j,k}\} < \pi_3$

(d) No, since π_2 is not greater than or smaller than π_3.

2.6. Lattice 1 is not distributive, because $c(b+d) = ca = c$ while $cb + cd = e + d = d \neq c$. It is a complemented lattice, where the complemented pairs are (a, e), (b, c), and (b, d).

Lattice 2 is complemented, where $b' = c$ or d, $c' = b$ or d, $d' = b$ or c, and $a' = e$ while $e' = a$. It is not distributive by Prob. 2.5. Lattice 3 is distributive but not complemented, for none of the elements b, c, or d has a complement.

Lattice 4 is distributive and complemented. The unique complements are (a, d) and (b, c). (It is actually a Boolean lattice of four elements.) Lattice 5 is distributive, but not complemented, for element b has no complement. It corresponds to a total ordering.

Chapter 3

3.3.

(a) $x' + y' + xyz' = x' + y' + xz' = x' + y' + z'$

(b) $(x' + xyz') + (x' + xyz')(x + x'y'z) = x' + xyz' = x' + yz'$

(c) $xy + wxyz' + x'y = y(x + wxz' + x') = y$

(d)

$$\begin{aligned} a + a'b + a'b'c + a'b'c'd + \cdots &= a + a'(b + b'c + b'c'd + \cdots) \\ &= a + (b + b'c + b'c'd + \cdots) \\ &= a + b + b'(c + c'd + \cdots) \\ &= a + b + c + \cdots \end{aligned}$$

(e) $xy + y'z' + wxz' = xy + y'z'$ (by 3.19)

(f) $w'x' + x'y' + w'z' + yz = x'y' + w'z' + yz$ (by 3.19)

3.5.

(a) Yes, (b) Yes, (c) Yes, (d) No.

3.7.

$$\begin{aligned} A' + AB &= 0 \Rightarrow A' + B = 0 \Rightarrow A = 1, B = 0 \\ AB &= AC \Rightarrow 1 \cdot 0 = 1 \cdot C \Rightarrow C = 0 \\ AB + AC' + CD &= C'D \Rightarrow 0 + 1 + 0 = D \Rightarrow D = 1 \end{aligned}$$

3.8. Add $w'x + yz'$ to both sides and simplify.

3.13.

(a) $f'(x_1, x_2, \ldots, x_n, 0, 1, +, \cdot) = f(x_1', x_2', \ldots, x_n', 1, 0, \cdot, +)$

$f'(x_1', x_2', \ldots, x_n', 1, 0, +, \cdot) = f(x_1, x_2, \ldots, x_n, 0, 1, \cdot, +) = f_d(x_1, x_2, \ldots, x_n)$

(b) There are two distinct self-dual functions of three variables.

$$\begin{aligned} f_1 &= xy + xz + yz \\ f_2 &= xyz + x'y'z + x'yz' + xy'z' \end{aligned}$$

(c) A self-dual function g of three variables can be found by selecting a function f of two (or three) variables and a variable A which is not (or is) a variable in f, and substituting to $g = Af + A'f_d$. For example,

$$\begin{aligned} f &= xy + x'y' \quad \text{then} \quad f_d = (x + y)(x' + y') \\ g &= z(xy + x'y') + z'(x + y)(x' + y') \\ &= xyz + x'y'z + xy'z' + x'yz' = f_2 \end{aligned}$$

Similarly, if we select $f = x + y$, we obtain function f_1 above.

3.14.
(*a*) $f(x, x, y) = x'y' =$ NOR, which is universal.
(*b*)

$$\begin{aligned} f(x, 1) &= x' \\ f(x', y) &= xy \end{aligned}$$

These together are functionally complete.

3.15. All are true except (*e*), since

$$1 \oplus (0 + 1) = 0 \neq (1 \oplus 0) + (1 \oplus 1) = 1$$

3.16.
(*a*)

$$\begin{aligned} f(0,0) &= a_0 = b_0 = c_0 \\ f(0,1) &= a_1 = b_0 \oplus b_1 = c_1 \\ f(1,0) &= a_2 = b_0 \oplus b_2 = c_2 \\ f(1,1) &= a_3 = b_0 \oplus b_1 \oplus b_2 \oplus b_3 = c_3 \end{aligned}$$

Solving explicitly for the b's and c's

$$\begin{aligned} b_0 &= a_0 \\ b_1 &= a_0 \oplus a_1 \\ b_2 &= a_0 \oplus a_2 \\ b_3 &= a_0 \oplus a_1 \oplus a_2 \oplus a_3 \\ c_0 &= a_0 \\ c_1 &= a_1 \\ c_2 &= a_2 \\ c_3 &= a_3 \end{aligned}$$

(*b*) The proof follows from the fact that in the canonical sum-of-products form, at any time only one minterm assumes value 1.

3.20.

$$f(A, B, C, D, E) = (A + B + E)(C)(A + D)(B + C)(D + E) = ACD + ACE + BCD + CDE$$

Clearly "C" is the essential executive.

3.21.

$$xy'z' + y'z + w'xyz + wxy + w'xyz' = x + y'z$$

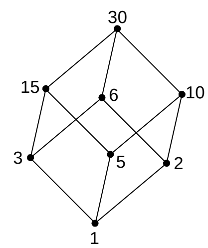
Married or a female under 25.

3.24. Element a must have a complement a', such that $a + a' = 1$. If $a' = 0$, then $a = 1$ (since $1' = 0$), which is contradictory to the initial assumption that a, 0, and 1 are distinct elements. Similar reasoning shows that $a' \neq 1$. Now suppose $a' = a$. However, then $a + a' = a + a = a \neq 1$.

3.25.
(a) $a + a'b = (a + a')(a + b) = 1 \cdot (a + b) = a + b$
(b) $b = b + aa' = (b + a)(b + a') = (c + a)(c + a') = c + aa' = c$
(c)

$$\begin{aligned} b &= aa' + b = (a+b)(a'+b) = (a+c)(a'+b) = aa' + a'c + ab + bc \\ &= a'c + ac + bc + c(a' + a + b) = c \end{aligned}$$

3.26. The (unique) complementary elements are: (30,1), (15,2), (10,3), (6,5)



Defining $a + b \cong lub(a, b)$ and $a \cdot b \cong glb(a, b)$, it is fairly easy to show that this is a lattice and is distributive. Also, since the lattice is complemented, it is a Boolean algebra.

Chapter 4

4.1
(a) $f_1 = x' + w'z'$ (unique)
(b) $f_2 = x'y'z' + w'y'z + xyz + wyz'$ or $f_2 = w'x'y' + w'xz + wxy + wx'z'$
(c) $f_3 = x'z' + w'z' + y'z' + w'xy'$ (unique)

4.2.
(a) MSP $= x'z' + w'yz + wx'y'$ (unique)
MPS $= (w + y + z')(w' + y' + z')(x' + z)(w' + x')$ or $(w + y + z')(w' + y' + z')(x' + z)(x' + y)$ (two minimal forms)
(b) $f = w'x'z' + xy'z' + x'y'z + xyz$ (unique)

4.4.

$$
\begin{aligned}
f &= \sum(4, 10, 11, 13) + \sum_{\phi}(0, 2, 5, 15) \\
&= wx'y + wxz + w'xy'
\end{aligned}
$$

There are four minimal forms.

4.5.
(a) $f_3 = f_1 \cdot f_2 = \sum(4) + \sum_{\phi}(8, 9)$ (four functions)
(b) $f_4 = f_1 + f_2 = \sum(0, 1, 2, 3, 4, 5, 7, 8, 9, 10, 11, 15) + \sum_{\phi}(6, 12)$

4.6. The intersection of f_1 and f_2' must be equal to $\sum(5, 6, 13)$.

$yz \backslash wx$	00	01	11	10
00				
01		1	1	
11				
10		1		

f

$yz \backslash wx$	00	01	11	10
00	1			1
01	1	1	1	1
11	1			1
10	1	1		1

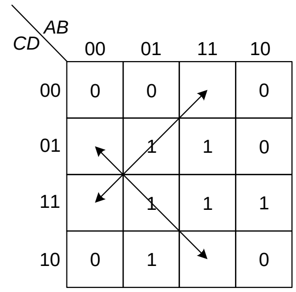
f_1

$yz \backslash wx$	00	01	11	10
00	0	ϕ	ϕ	0
01	0	1	1	0
11	0	ϕ	ϕ	0
10	0	1	ϕ	0

f_2'

Thus, $f_2 = \sum(0, 1, 2, 3, 8, 9, 10, 11) + \sum_{\phi}(4, 7, 12, 14, 15)$. f_2 represents 32 functions, the simplest of which is $f_2 = x'$.

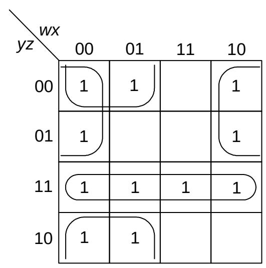
4.9. When $+$ and $\cdot$ are interchanged to form the dual f_d of f, for each 1 cell in the map of f, there must be a 0 at the complementary variable value in the map of f_d. For example, if f has a 1 in cell 1101, then f_d has a 0 in cell 0010. For a self-dual function, if f has, for example, term $ABCD$ in the sum-of-products form, it must have term $A + B + C + D$ in the product-of-sums form. The map of f without g is shown below. The zeros are entered following the above discussion.



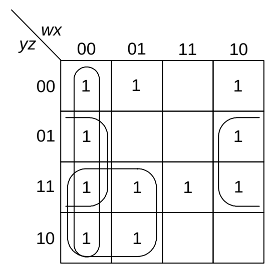
The two pairs of cells connected by arrows can be arbitrarily filled so long as one end of an arrow points to a 1 and the other end to a 0. Two possible minimal sum-of-products expressions are:

$$
\begin{aligned}
f &= AB + BD + BC + ACD \\
f &= CD + A'D + BD + A'BC
\end{aligned}
$$

4.13.



Essential prime implicants and unique minimum cover.
$T = yz + w'z' + x'y'$



Nonessential prime implicants.

4.14.

(a) Essential prime implicants: $\{w'xy', wxy, wx'y'\}$

Non-essential prime implicants: $\{w'z, x'z, yz\}$

(b) $T = w'xy' + wxy + wx'y' +$ {any two of $(w'z, x'z, yz)$}.

4.15.

(a) (i) $\sum(5, 6, 7, 9, 10, 11, 13, 14)$

(ii) $\sum(0, 3, 5, 6, 9, 10, 12, 15)$

(b) (i) $\sum(5, 6, 7, 9, 10, 11, 12, 13, 14)$

(ii) $\sum(5, 6, 7, 9, 10, 11, 13, 14, 15)$

(c) $\sum(0, 4, 5, 13, 15)$

(d) $\sum(0, 3, 6, 7, 8, 9, 13, 14)$

4.16.

(a) False. See, for example, function $\sum(3, 4, 6, 7)$.

(b) False. See, for example, Problem 4.2(a).

(c) True, for it implies that *all* the prime implicants are essential and, thus, f is represented uniquely by the sum of the essential prime implicants.

(d) True. Take all prime implicants except p, eliminate all the redundant ones; the sum of the remaining prime implicants is an irredundant cover.
(e) False. See the following counter-example.

$yz \backslash vwx$	000	001	011	010	110	111	101	100
00		1			1		1	1
01		1	1	1	1			
11	1	1	1					1
10			1		1	1		1

4.17.
(a)

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

(b) To prove existence of such a function is a simple extension of (a); it can be represented as a sum of 2^{n-1} product terms. Adding another true cell can only create an adjacency which will reduce the number of cubes necessary for a cover.
(c) From the above arguments, clearly $n2^{n-1}$ is the necessary bound.

4.18.
(a) Such a function has

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

1's, of which no two are adjacent. Thus, the number of prime implicants equals the number of 1's. All prime implicants are essential.
(b) Each of the cells of the function from part (a) will still be contained in a distinct prime implicant, although it may not be an essential prime implicant. Each cube now covers 2^{n-k} cells, and there are

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

such cubes.

4.19.
(a) T is irredundant and depends on D. Thus, no expression for T can be independent of D.

(b) Convert the minimal sum-of-products expression of the unate function to a form having no primed literals. A prime implicant in this function is of the form $x_1x_2\cdots x_k$, and covers minterm $x_1x_2\cdots x_kx'_{k+1}\cdots x'_n$. Any other implicant containing this minterm must:
(i) either contain a complemented variable from $x'_{k+1},\cdots,x'_n$, which violates unateness, or
(ii) be expressed only in terms of variables $x_1,\cdots,x_k$, in which case, it contains the given prime implicant, which violates the minimality assumption. The expression being minimal implies that it is a sum of prime implicants and the discussion above shows that each such prime implicant is essential. Thus, the minimal form of a unate function is the sum of all the essential prime implicants.
(c) No. Counter-example: $f = xy + x'z$

4.21.

$$f = vx'yz + vwxz + \left\{ \begin{array}{l} v'w'y'z' + vx'y'z' + wxy'z' \\ w'x'y'z' + v'xy'z' + vwy'z' \end{array} \right.$$

4.22.
(a) Use truth tables.
(b)

$$\begin{aligned} xyz' + xy'z + x'z &= (xyz' \oplus xy'z \oplus xyz'xy'z) + x'z = (xyz' \oplus xy'z \oplus 0) + x'z \\ &= (xyz' \oplus xy'z) \oplus x'z \oplus (xyz' \oplus xy'z)\cdot x'z = xyz' \oplus xy'z \oplus x'z \end{aligned}$$

To eliminate the negation, write

$$\begin{aligned} xyz' \oplus xy'z \oplus x'z &= xy(1\oplus z) \oplus x(1\oplus y)z \oplus (1\oplus x)z = xy \oplus xyz \oplus xz \oplus xyz \oplus z \oplus xz \\ &= xy \oplus z \end{aligned}$$

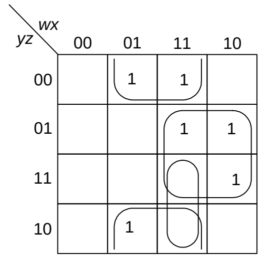
(c) Use $A \oplus B = AB' + A'B$.

$$\begin{aligned} (x\oplus y)\oplus z &= (xy' + x'y)\oplus z \\ &= (xy' + x'y)z' + (xy' + x'y)'z \\ &= xy'z' + x'yz' + (xy + x'y')z \\ &= xy'z' + x'yz' + xyz + x'y'z \end{aligned}$$

4.23.
(a) The following rules may be employed when forming cubes on the map:

$$\begin{aligned} x \oplus x &= 0 \\ xy \oplus xy' &= x \end{aligned}$$

(b)



$f_1 = wz \oplus xz' \oplus wxy$

From the map of f_2, we similarly find

$$f_2 = yz' \oplus wxz' \oplus w'x'y \oplus wy'z \oplus xy'z$$

4.24. Since prime implicant A has only three x's, it is obvious that minterm 2 is a don't care. Since we know that we are given *all* the prime implicants, it therefore follows that, for example, prime implicant $D = abd$ and hence the minterm that corresponds to the rightmost column is 13. We employ similar reasoning to find the other prime implicants. To determine B, we must establish the other unknown minterm. It is easy to verify that there are two possibilities; it is either 4 or 1. Thus, we can show that there are two possible functions corresponding to the above possibilities; namely,

$$\begin{aligned} f_1 &= \sum(0,4,7,8,10,13,15) + \sum_{\phi}(2,9) \\ f_2 &= \sum(0,1,7,8,10,13,15) + \sum_{\phi}(2,12) \end{aligned}$$

The minimal expressions are:

$$\begin{aligned} f_1 &= b'd' + a'c'd' + bcd + ac'd \\ f_2 &= b'd' + a'b'c' + bcd + abc' \end{aligned}$$

4.25.
(a) P_{min} is the union of the minterms of T_1 and T_2.
(b)

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

Q

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

R

(c)

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

P_{max}

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

Q

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

R

4.26.

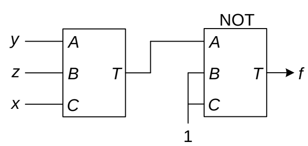
(*a*) $T(A, 1, 1) = A'$ and $T(A, B, B) = A'B \Rightarrow T(A', B, B) = AB$. Since set {AND,NOT} is functionally complete, so is $\{T, 1\}$.

(*b*)

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

$w'T'(y,z,x)$ $\quad$ $wT'(y,z,x)$

$$f(w,x,y,z) = w'T'(y,z,x) + wT'(y,z,x)$$
$$= T'(y,z,x)$$



Chapter 5

5.4. From the map, it is evident that T can be expressed in product-of-sums form as:

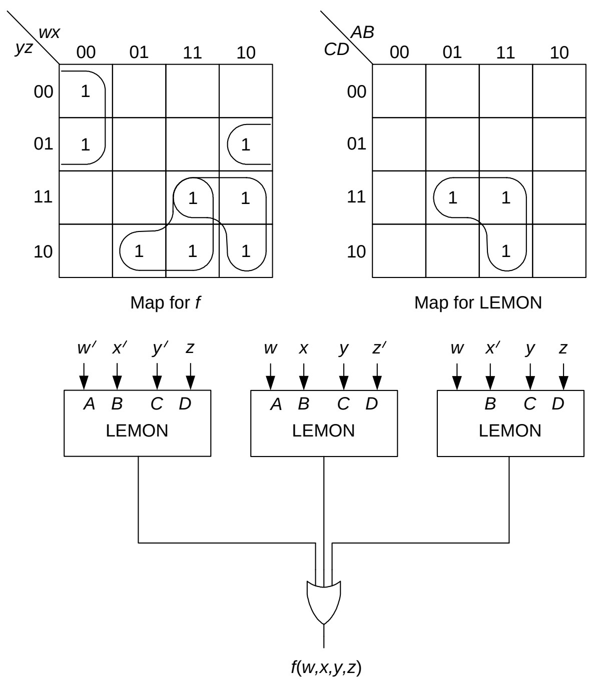
$$\begin{aligned} T &= (w' + x' + z')(x + y) \\ &= (wxz)'(x + y) \end{aligned}$$

5.5.

		A=0	A=1	A=2
B	0	2	2	2
	1	0	1	0
	2	2	2	0

(A∖B)⌐C = f(A,B)

5.6. To cover the function, it is only necessary to cover the 1's in the map for $f(w, x, y, z)$ using a number of L-shaped patches like that shown in the map on the right below; translations and rotations being permitted.



5.7. Connect inputs A and B together; this eliminates the possibility of an "explosion." The output of the modified gate is 1 if and only if its two inputs are 0's. Hence, it is a NOR gate. Alternatively, connecting B and C together avoids explosions and yields a NAND gate.

5.9.

(*a*) $X_4 > Y_4$ if either $G_3 = 1$ or if the first three significant bits of X_4 and Y_4 are identical, but $x_4 = 1$ while $y_4 = 0$. Thus,

$$G_4 = G_3 + S_3' x_4 y_4'$$

Using similar reasoning we find that,

$$S_4 = S_3 + G_3' x_4' y_4$$

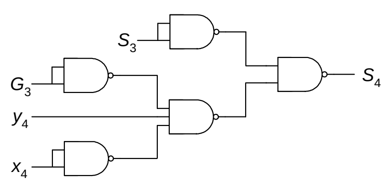
The above equations can also be found by means of the maps below

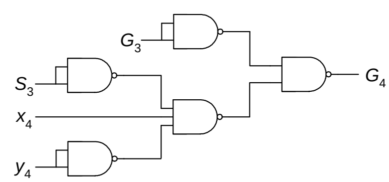
$x_4y_4 \backslash G_3S_3$	00	01	11	10
00	0	0	ϕ	1
01	0	0	ϕ	1
11	0	0	ϕ	1
10	1	0	ϕ	1

G_4

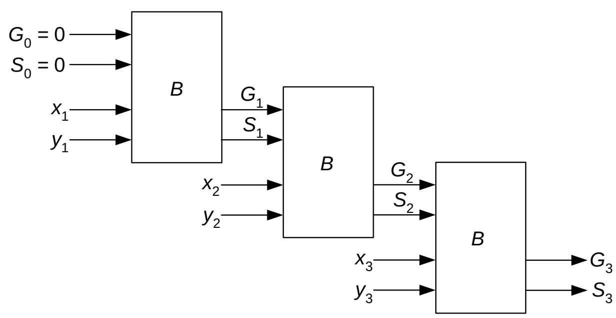
$x_4y_4 \backslash G_3S_3$	00	01	11	10
00	0	1	ϕ	0
01	1	1	ϕ	0
11	0	1	ϕ	0
10	0	1	ϕ	0

S_4

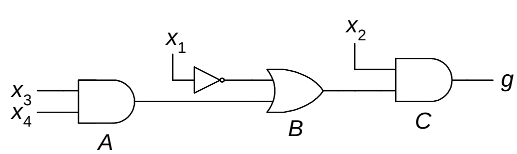




(b) Assume that $G_0 = 0$, $S_0 = 0$.



5.10.



5.11. (*a*)

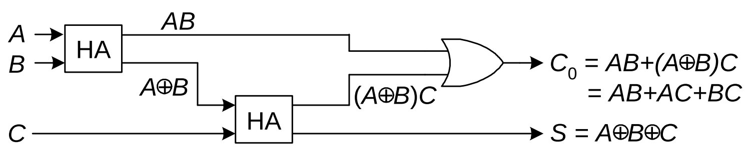
A	B	S	C_0
0	0	0	0
0	1	1	0
1	1	0	1
1	0	1	0

$$\begin{aligned} S &= AB' + A'B = A \oplus B \\ C_0 &= AB \end{aligned}$$

(*b*) $S = (A+B)(A'+B') = (A+B)(AB)'$. The circuit diagram is now evident.

(*c*) Similar to (*b*). Use a gate whose inputs are connected together to obtain complementation.

5.12.



5.13.

x	y	Difference (D)	Borrow (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$\begin{aligned} D &= x \oplus y \\ B &= x'y \end{aligned}$$

5.14.

x	y	B	D	B_0
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

$$\begin{aligned} D &= x \oplus y \oplus B \\ B_0 &= x'y + x'B + yB \end{aligned}$$

5.15. The circuit is a NAND realization of a full adder.

5.17. A straightforward solution by means of a truth table. Let $A = a_1a_0$, $B = b_1b_0$ and $C = A \cdot B = c_3c_2c_1c_0$.

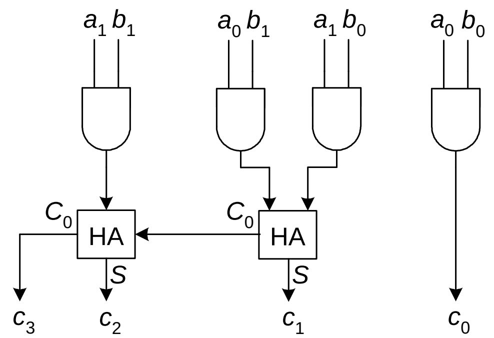
a_1	a_0	b_1	b_0	c_3	c_2	c_1	c_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

$$
\begin{aligned}
c_0 &= a_0b_0 \\
c_1 &= a_1b_0b_1' + a_0b_0'b_1 + a_0'a_1b_0 + a_0a_1'b_1 \\
c_2 &= a_0'a_1b_1 + a_1b_0'b_1 \\
c_3 &= a_0a_1b_0b_1
\end{aligned}
$$

A simpler solution is obtained by observing that

$$
\begin{array}{cccc}
 & & a_1 & a_0 \\
 & \times & b_1 & b_0 \\
\hline
 & + & a_1b_0 & a_0b_0 \\
 & a_1b_1 & a_0b_1 & \\
\hline
c_3 & c_2 & c_1 & c_0
\end{array}
$$

The c_i's can now be realized by means of half adders as follows:



5.19. Since the distance between A and B is 4, one code word cannot be permuted to the other due to noise in only two bits. In the map below, the minterms designated A correspond to the code word A and to all other words which are at a distance of 1 from A. Consequently, when any of these words is received, it implies that A has been transmitted. (Note that all these minterms are at a distance 3 from B and, thus, cannot correspond to B under the two-error assumption.) The equation for A is thus given by

$$A = x_1'x_2'x_3 + x_1'x_2'x_4' + x_1'x_3x_4' + x_2'x_3x_4'$$

Similarly, B is given by

$$B = x_1x_2x_3' + x_1x_2x_4 + x_2x_3'x_4 + x_1x_3'x_4$$

The minterms labeled C correspond to output C.

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00	A	C	B	C
01	C	B	B	B
11	A	C	B	C
10	A	A	C	A

5.20.

(a)

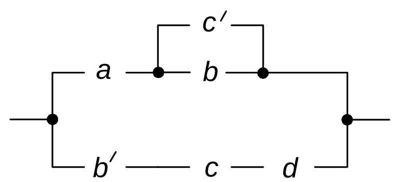
Tie sets: ac'; ad; abc; $b'cd$

Cut sets: $(a+b')$; $(a+b+c)$; $(b+c'+d)$

$T = (a+b')(a+b+c)(b+c'+d)$

(b) Let $d=0$ and choose values for a, b, and c, such that $T=0$; that is, choose $a=0$, $b=0$, $c=1$. Evidently, T cannot change from 0 to 1, unless a d branch exists and is changed from 0 to 1.

(c)



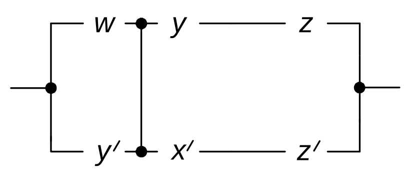
5.21.

(a) Tie sets:

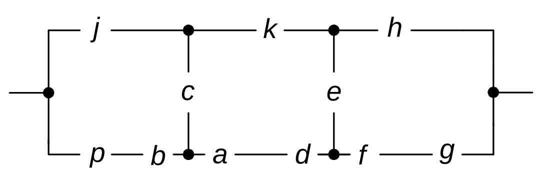
$$\begin{aligned} T &= wx'y' + wyz + xz + w'yz + x'yz' \\ &= xz + x'(w + y) \end{aligned}$$

requires five branches.

(b) The nMOS network of the complex CMOS gate can be obtained by directly replacing each branch in the network below by an nMOS transistor. Its pMOS network can be obtained by replacing a series (parallel) connection in the nMOS network by a parallel (series) connection in the pMOS network. Since a complex gate is inverting, it needs to be fed to an inverter to realize the same function. This requires a total of 6+6+2 = 14 transistors.



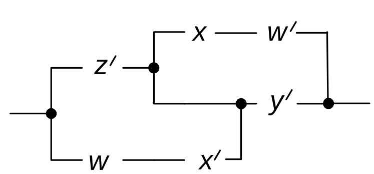
5.22. "Factor out" branches b, a, d, and f.



5.24.

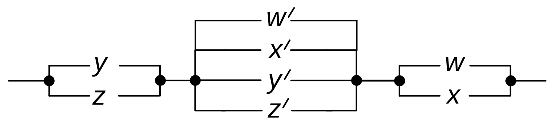
(a)

$$\begin{aligned} T &= y'z' + w'xz' + wx'y' \\ &= z'(y' + w'x) + wx'y' \end{aligned}$$



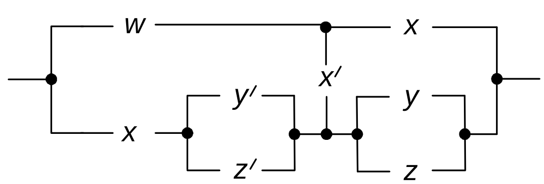
(c) Take the product-of-sums expression

$$T = (y+z)(w+x)(w'+x'+y'+z')$$

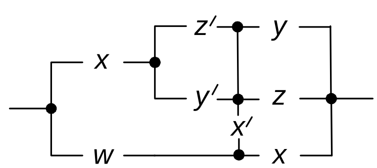


(d)

$$\begin{aligned} T &= wx + wy + wz + x(y'z + yz') \\ &= w(x+y+z) + x(y+z)(y'+z') \end{aligned}$$



(e) $T = x(y'z + yz') + w(x+y+z)$



Chapter 6

6.1.

(a) No, because of the wwx term

(b) Yes, even though the corresponding sum-of-products expression can be reduced using the consensus theorem.

(c) No, the sum-of-products expression contains yz and wyz. Since yz contains wyz, this is not an algebraic factored form.

6.2.

(a)

Algebraic divisors: $x + y$, $x + z$, $y + z$, $x + y + z$

Boolean divisors: $v + w$, $v + x$, $v + y$, $v + z$, $v + x + y$, $v + x + z$, $v + y + z$, $v + x + y + z$

(b) Algebraic divisor $x + y + z$ leads to the best factorization: $v + w(x + y + z)$ with only five literals.

6.3.

Level	*Kernel*	*Co-kernel*
0	$y' + z$	vw, x'
0	$vw + x'$	y', z
1	$vy' + vz + x$	w
2	$vwy' + vwz + x'y' + x'z + wx$	1

6.4.

(a)

Cube-literal incidence matrix

	Literal					
Cube	u	v	w	x	y	z'
wxz'	0	0	1	1	0	1
uwx	1	0	1	1	0	0
wyz'	0	0	1	0	1	1
uwy	1	0	1	0	1	0
v	0	1	0	0	0	0

(*b*)

Prime rectangles and co-rectangles

Prime rectangle	*Co-rectangle*
$(\{wxz', uwx\}, \{w, x\})$	$(\{wxz', uwx\}, \{u, v, y, z'\})$
$(\{wxz', wyz'\}, \{w, z'\})$	$(\{wxz', wyz'\}, \{u, v, x, y\})$
$(\{wxz', uwx, wyz', uwy\}, \{w\})$	$(\{wxz', uwx, wyz', uwy\}, \{u, v, x, y, z'\})$
$(\{uwx, uwy\}, \{u, w\})$	$(\{uwx, uwy\}, \{v, x, y, z'\})$
$(\{wyz', uwy\}, \{w, y\})$	$(\{wyz', uwy\}, \{u, v, x, z'\})$

(*c*)

Kernels and co-kernels

Kernel	*Co-kernel*
$u + z'$	wx
$x + y$	wz'
$xz' + ux + yz' + uy$	w
$x + y$	uw
$u + z'$	wy

6.6.

$$\begin{aligned} H &= x + y \\ G(H, w, z) &= 0 \cdot w'z' + 1 \cdot w'z + H' \cdot wz' + H \cdot wz \\ &= w'z + H'wz' + Hwz \\ &= w'z + H'wz' + Hz \end{aligned}$$

6.8.

(*a*) Consider don't-care combinations 1, 5, 25, 30 as true combinations (i.e., set them equal to 1). Then

$$\begin{aligned} H(v, x, z) &= \sum(0, 5, 6) = v'x'z' + vx'z + vxz' \\ G(H, w, y) &= H'w'y' + Hwy' + Hwy \\ &= H'w'y' + Hw \end{aligned}$$

6.9.

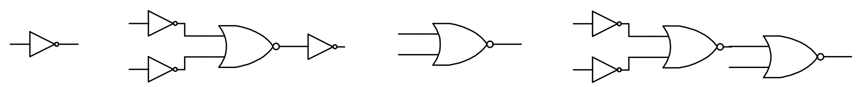
(*a*)

Permuting x and y yields: $yxz + yx'z' + y'xz' + y'x'z$, which is equal to f.
Permuting x and z yields: $zyx + zy'x' + z'yx' + z'y'x$, which is equal to f.
Permuting y and z yields: $xzy + xz'y' + x'zy' + x'z'y$, which is equal to f.
(*b*) Decomposition:

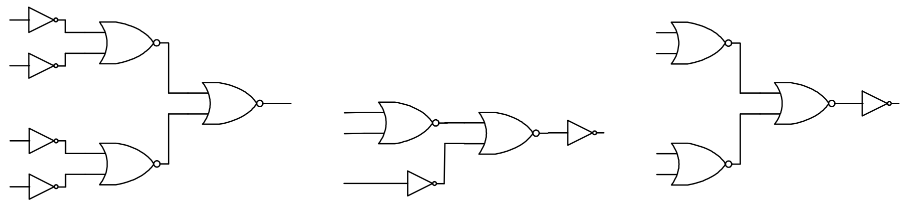
$$\begin{aligned} f_1 &= yz + y'z' \\ f &= xf_1 + x'f_1' \end{aligned}$$

6.10.

(*a*)



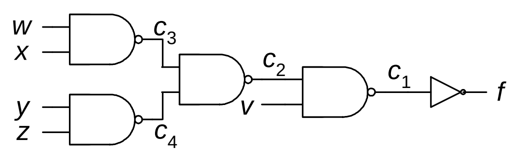
(*a*) INV. (*b*) NAND2. (*c*) NOR2. (*d*) AOI21.



(*e*) AOI22. (*f*) OAI21. (*g*) OAI22.

6.11.

(*a*)



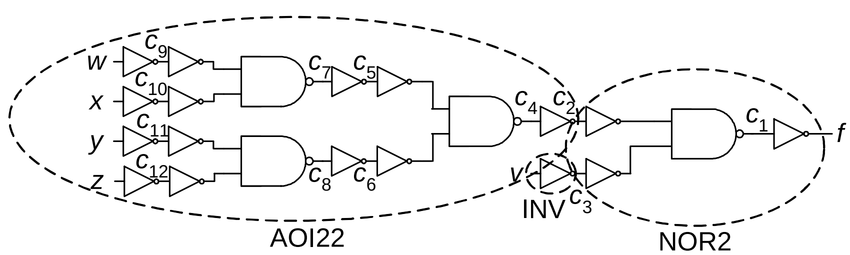
(*b*)

Matches

Node	*Match*
f	INV
c_1	NAND2
c_2	NAND2
c_3	NAND2
c_4	NAND2

Optimum-area network cover: $1+2\times 4 = 9$.

(*c*) Decomposed subject graph with INVP inserted (ignore the cover shown for now):



Matches

Node	*Match*
f	INV, NOR2
c_1	NAND2, OAI21
c_2	INV, NOR2, AOI22
c_3	INV
c_4	NAND2, OAI22
c_5	INV, NOR2
c_6	INV, NOR2
c_7	NAND2
c_8	NAND2
c_9	INV
c_{10}	INV
c_{11}	INV
c_{12}	INV

Covering of the subject graph

Node	*Match*	*Area cost*
c_9	INV(w)	1
c_{10}	INV(x)	1
c_{11}	INV(y)	1
c_{12}	INV(z)	1
c_7	NAND2(w, x)	2
c_8	NAND2(y, z)	2
c_5	INV(c_7)	3
	NOR2(c_9, c_{10})	4
c_6	INV(c_8)	3
	NOR2(c_{11}, c_{12})	4
c_4	NAND2(c_7, c_8)	6
	OAI22($c_9, c_{10}, c_{11}, c_{12}$)	8
c_3	INV(v)	1
c_2	INV(c_4)	7
	NOR2(c_5, c_6)	8
	AOI22(w, x, y, z)	4
c_1	NAND2(c_4, v)	8
	OAI21(c_5, c_6)	9
f	INV(c_1)	9
	NOR2(c_2, c_3)	7

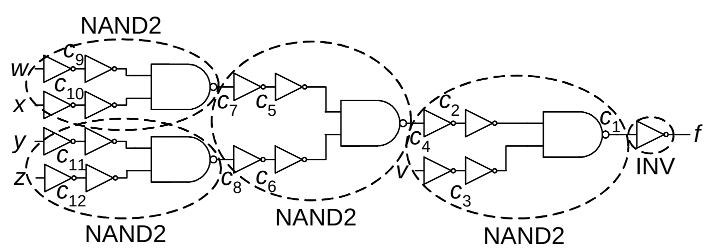
The optimum cover with area cost 7 is shown in the above figure. Thus, introducing INVP reduced overall area cost.

(*d*)

Covering of the subject graph

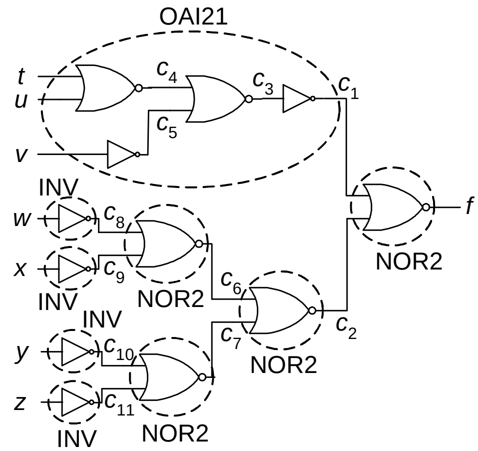
Node	*Match*	*Delay cost*
c_9	INV(w)	1
c_{10}	INV(x)	1
c_{11}	INV(y)	1
c_{12}	INV(z)	1
c_7	NAND2(w, x)	2
c_8	NAND2(y, z)	2
c_5	INV(c_7)	3
	NOR2(c_9, c_{10})	4
c_6	INV(c_8)	3
	NOR2(c_{11}, c_{12})	4
c_4	NAND2(c_7, c_8)	4
	OAI22($c_9, c_{10}, c_{11}, c_{12}$)	10
c_3	INV(v)	1
c_2	INV(c_4)	5
	NOR2(c_5, c_6)	6
	AOI22(w, x, y, z)	8
c_1	NAND2(c_4, v)	6
	OAI21(c_5, c_6)	10
f	INV(c_1)	7
	NOR2(c_2, c_3)	8

The optimum cover with delay cost of 7 is shown in the figure below.



6.12.

(a)



(b)

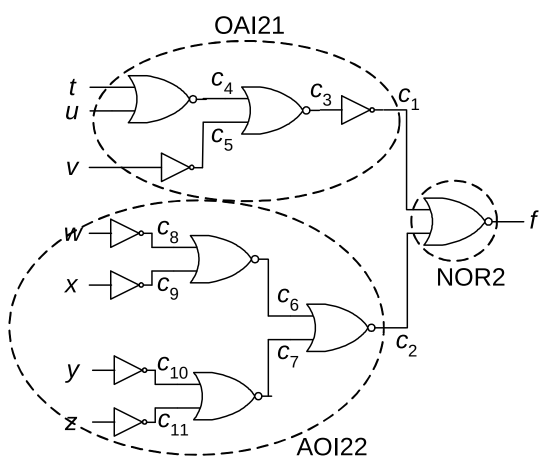
Matches

Node	*Match*
f	NOR2
c_1	INV, OAI21
c_2	NOR2, AOI22
c_3	NOR2
c_4	NOR2
c_5	INV
c_6	NOR2
c_7	NOR2
c_8	INV
c_9	INV
c_{10}	INV
c_{11}	INV

Covering of the subject graph

Node	*Match*	*Delay cost*
c_8	INV(w)	1
c_9	INV(x)	1
c_{10}	INV(y)	1
c_{11}	INV(z)	1
c_4	NOR2(t, u)	3
c_5	INV(v)	1
c_6	NOR2(c_8, c_9)	4
c_7	NOR2(c_{10}, c_{11})	4
c_3	NOR2(c_4, c_5)	6
c_2	NOR2(c_6, c_7)	7
	AOI22(w, x, y, z)	8
c_1	INV(c_3)	7
	OAI21(t, u, v)	7
f	NOR2(c_1, c_2)	10

There are two optimum-delay covers, the one with the lower area cost of 15 is shown in the above figure. (c) If the delay constraints is relaxed to 11, then AOI22(w, x, y, z) can be used to implement c_2, as shown in the figure below. Its area cost is only 9.



Chapter 7

7.1.

(*a*) The entries in the map at the left below indicate the weights associated with the corresponding input combinations. For example, the entry in cell 1001 is -4, since $w_1x_1 + w_4x_4 = -1 - 3 = -4$. The weight associated with each cell can be found by adding the weights of the corresponding row and column. Star symbol * marks those cells that are associated with weights larger than $T = -\frac{1}{2}$.

weights	$x_3x_4 \backslash x_1x_2$	00	01	11	10
	weights		2	1	-1
	00	0*	2*	1*	-1
-3	01	-3	-1	-2	-4
-1	11	-1	1*	0*	-2
2	10	2*	4*	3*	1*

$\sum w_i x_i$

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00	1	1	1	
01				
11		1	1	
10	1	1	1	1

Map of function

$$f(x_1, x_2, x_3, x_4) = x_2x_3 + x_3x_4' + x_2x_4' + x_1'x_4'$$

(*b*) Let g be the function realized by the left threshold element, then for $f(x_1, x_2, x_3, x_4)$, we obtain the following maps:

weights	$x_3x_4 \backslash x_1x_2$	00	01	11	10
	weights		2	4	2
	00	4*	2	4*	2
1	01	5*	3	5*	3
2	11	6*	8*	6*	8*
1	10	5*	3	5*	3

$\sum w_i x_i + 4g(x_1, x_2, x_3, x_4)$

$\xrightarrow{T = \frac{7}{2}}$

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00	1		1	
01	1		1	
11	1	1	1	1
10	1		1	

Map of function

$$f(x_1, x_2, x_3, x_4) = x_1'x_2' + x_1x_2 + x_3x_4$$

7.2.

(*a*) The following inequalities must be satisfied:

$$w_3 > T,\ w_2 > T,\ w_2 + w_3 > T,\ w_1 + w_2 < T,$$
$$w_1 + w_2 + w_3 > T,\ w_1 < T,\ w_1 + w_3 < T,\ 0 < T$$

These inequalities are satisfied by the element whose weight-threshold vector is equal to $\{-2,2,2;\ 1\}$. Hence, function $f_1(x_1, x_2, x_3)$ is a threshold function.

(*b*) The inequalities are:

$$0 > T,\ w_2 > T,\ w_3 < T,\ w_2 + w_3 < T$$
$$w_1 + w_2 > T,\ w_1 + w_2 + w_3 < T,\ w_1 > T,\ w_1 + w_3 > T$$

Function $f_2(x_1, x_2, x_3)$ is realizable by threshold element $\{1, -1, -2; -\frac{3}{2}\}$.
(c) Among the eight inequalities, we find the following four:

$$0 > T,\ w_3 < T,\ w_2 < T,\ w_2 + w_3 > T$$

The last three inequalities can only be satisfied if $T > 0$; but this contradicts the first inequality. Hence, $f_3(x_1, x_2, x_3)$ is not a threshold function.

7.4.
(a) Threshold element $\{1{,}1;\ \frac{1}{2}\}$ clearly generates the logical sum of the two other elements.

			1	2	1
	$x_3x_4 \backslash x_1x_2$	00	01	11	10
	00	0	1	2	1
3	01	3*	4*	5*	4*
2	11	2	3*	4*	3*
-1	10	-1	0	1	0

Top element

+

			2	3	1
	$x_3x_4 \backslash x_1x_2$	00	01	11	10
	00	0	2*	3*	1
-1	01	-1	1	2*	0
-3	11	-3	-1	0	-2
-2	10	-2	0	1	-1

Bottom element

=

$x_3x_4 \backslash x_1x_2$	00	01	11	10
00		1	1	
01	1	1	1	1
11		1	1	1
10				

f

$$f(x_1, x_2, x_3, x_4) = x_3'x_4 + x_2x_3' + x_2x_4 + x_1x_4$$

(b) Using the techniques of Sec. 7.2, we obtain

$$\Phi(x_1, x_2, x_3, x_4) = f(x_1, x_2, x_3', x_4) = x_3x_4 + x_2x_3 + x_2x_4 + x_1x_4$$

Minimal true vertices: 0011, 0110, 0101, 1001
Maximal false vertices: 1100, 1010, 0001

$$\left.\begin{array}{c} w_3 + w_4 \\ w_2 + w_3 \\ w_2 + w_4 \\ w_1 + w_4 \end{array}\right\} > \left\{\begin{array}{c} w_1 + w_2 \\ w_1 + w_3 \\ w_4 \end{array}\right. \Rightarrow \begin{array}{cc} w_4 > w_1 & \\ w_3 > w_1 & w_4 > w_2 \\ w_2 > w_1 & w_4 > w_3 \\ w_2 + w_3 > w_4 & \end{array}$$

In other words, $w_4 > \{w_2, w_3\} > w_1$. Choose $w_1 = 1$, $w_2 = w_3 = 2$, and $w_4 = 3$. This choice, in turn, implies $T = \frac{7}{2}$. Thus,

$$V_\Phi = \{1, 2, 2, 3; \frac{7}{2}\}$$

and

$$V_f = \{1, 2, -2, 3; \frac{3}{2}\}$$

7.5.
(a)

1. $\frac{T}{w} = 0$; $\sum x_i \geq 0$. Hence, f is identically equal to 1.

2. $\frac{T}{w} > n$; $\sum x_i \le n$. Hence, f is identically equal to 0.

3. $0 < \frac{T}{w} \le n$; then $f = 1$ if p or more variables equal 1 and $f = 0$ if $p-1$ variables or fewer variables equal 1, where $p - 1 < \frac{T}{w} \le p$ for $p = 0, 1, 2, \cdots$.

7.6.

(a) By definition, $f_d(x_1, x_2, \cdots, x_n) = f'(x_1', x_2', \cdots, x_n')$. Let $g(x_1, x_2, \cdots, x_n) = f(x_1', x_2', \cdots, x_n')$, then the weight-threshold vector for g is

$$V_g = \{-w_1, -w_2, \cdots, -w_n;\ T - (w_1 + w_2 + \cdots + w_n)\}$$

Now, since $f_d = g'$, we find

$$V_{f_d} = \{w_1, w_2, \cdots, w_n;\ (w_1 + w_2 + \cdots + w_n) - T\}$$

(b) Let V be the weight-threshold vector for f, where $V = \{w_1, w_2, \cdots, w_n;\ T\}$. Utilizing the result of part (a), we find that $g = x_i' f + x_i f_d$ equals 1 if and only if $x_i = 0$ *and* $\sum w_j x_j > T$, OR if and only if $x_i = 1$ *and* $\sum w_j x_j > (w_1 + w_2 + \cdots + w_n) - T$.

Establishing, similarly, the conditions for $g = 0$, we find weight w_i of x_i, namely

$$w_i = (w_1 + w_2 + \cdots + w_n) - 2T = \sum_f w_j - 2T$$

Thus, the weight-threshold vector for g is

$$V_g = \{w_1, w_2, \cdots, w_n, \sum_f w_j - 2T;\ T\}$$

7.7.

(a) In G, for $x_p = 0$, $G = f$. Thus, $T_g = T$. When $x_p = 1$, $\sum_n w_i + w_p$ should be larger than T, regardless of the values of $\sum_n w_i$. Define N as the most negative value that $\sum_n w_i$ can take, then $N + w_p > T$, which yields $w_p > T - N$. Choose $w_p = T - N + 1$, then

$$V_g = \{w_1, w_2, \cdots, w_n, T - N + 1;\ T\}$$

Similarly, we can find weight-threshold vector $V_h = \{w_1, w_2, \cdots, w_n, w_p;\ T_h\}$ of H, that is

$$V_h = \{w_1, w_2, \cdots, w_n, M - T + 1;\ M + 1\}$$

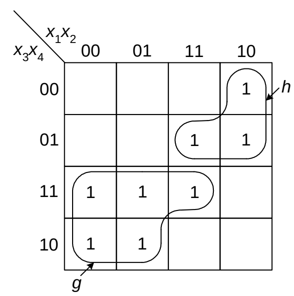
Note: If x_p is a member of set $\{x_1, x_2, \cdots, x_n\}$, then N and M must be defined, such that the contribution of w_p is excluded.

7.10.

(a) Proof that "common intersection" implies unateness: If the intersection of the prime implicants is nonzero, then in the set of all prime implicants, each variable appears only in primed or in unprimed

form, but not in both, since otherwise the intersection of the prime implicants would contain some pair $x_i x_i'$ and would be zero. Also, since f is a sum of prime implicants, it is unate by definition.
Proof that unateness implies "common intersection": Write f as a sum of *all* its prime implicants. Assume that f is positive (negative) in x_i and literal x_i' (x_i) appears in some prime implicant. By Problem 7.9, this literal is redundant and the corresponding term is not a prime implicant. However, this contradicts our initial assumption; hence, each variable appears only in primed or unprimed form in the set of all prime implicants. This implies that the intersection of the prime implicants is nonzero, and is an implicant of f which is covered by all the prime implicants.
(b) The proof follows from the above and from Problem 4.19.

7.12.
(a)



$$
\begin{aligned}
f_1 &= \sum(2,3,6,7,8,9,13,15) \\
V_g &= \{-2,1,3,1;\frac{5}{2}\} \text{ [see Fig. 7.10]} \\
h &= x_1x_3'(x_2'+x_4) \\
\Phi_h &= x_1x_3(x_2+x_4)
\end{aligned}
$$

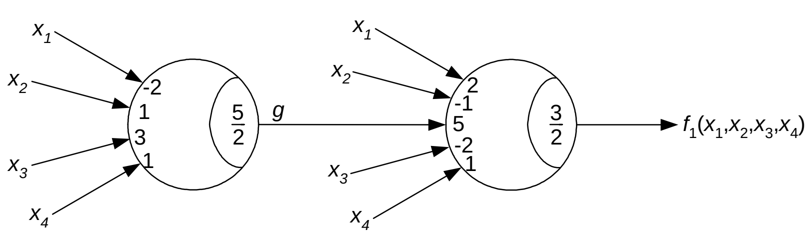
The weight-threshold vector for Φ_h is given by

$$V_{\Phi_h} = \{2,1,2,1;\frac{9}{2}\}$$

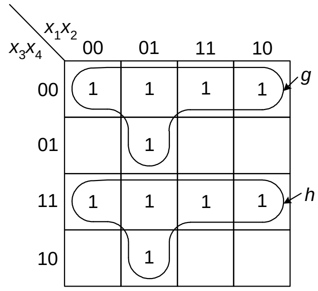
Thus,

$$V_h = \{2,-1,-2,1;\frac{3}{2}\}$$

The cascade realization of f_1 is shown below. The smallest $\sum w_i x_i$ for h, when $g=1$, is -3. Thus, $-3 + w_g \geq \frac{3}{2}$, which implies $w_g \geq \frac{9}{2}$. We, therefore, choose $w_g = 5$.



(b) $f_2 = \sum(0, 3, 4, 5, 6, 7, 8, 11, 12, 15)$



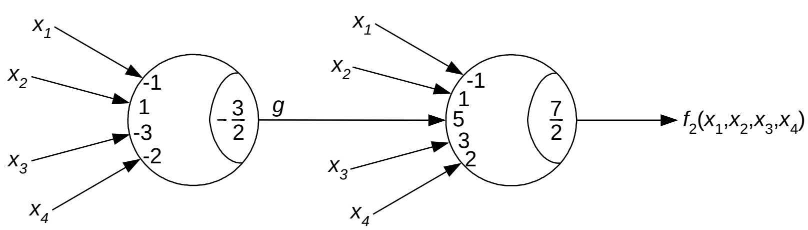
It is easy to verify that $g(x_1, x_2, x_3, x_4)$ can be realized by the element with weight-threshold vector

$$V_g = \{-1, 1, -3, -2; -\frac{3}{2}\}$$

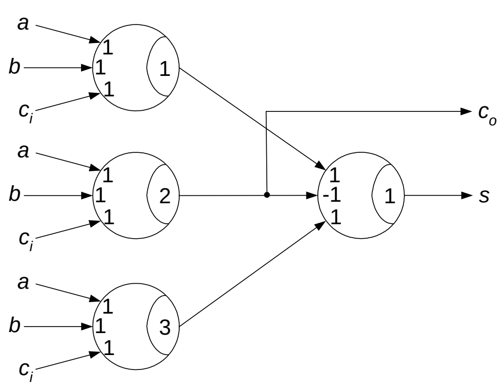
Also, since $h(x_1, x_2, x_3, x_4) = g(x_1, x_2, x_3', x_4')$, we conclude

$$V_h = \{-1, 1, 3, 2; \frac{7}{2}\}$$

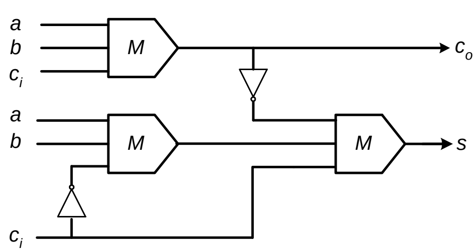
A possible realization of f_2 is shown below.



7.13. The implementation is given below.



7.15. The implementation is given below.



Chapter 8

8.1.

(a) The vector that activates the fault and sensitizes the path through gates G_5 and G_8 requires: (i) $c_1 = 1 \Rightarrow x_2 = 0$ and $x_3 = 0$, (ii) $G_6 = 0$ regardless of whether a fault is present, which implies $x_4 = 1$, (iii) $G_7 = 0$, which implies $G_3 = 1$ (since $x_3 = 0$), which in turn implies $x_4 = 0$, and (iv) $G_4 = 0$. Thus, we need to set conflicting requirements on x_4, which leads to a contradiction. By the symmetry of the circuit, it is obvious that an attempt to sensitize the path through gates G_6 and G_8 will also fail.
(b) Sensitizing both the paths requires: (i) $c_1 = 1 \Rightarrow x_2 = 0$ and $x_3 = 0$, (ii) $G_4 = 0$, which implies $G_1 = 1$ (since $x_2 = 0$), which in turn implies $x_1 = x_3 = 0$, and (iii) $G_7 = 0$, which implies $G_3 = 1$ (since $x_3 = 0$), which in turn implies $x_2 = x_4 = 0$. All these conditions can be satisfied by the test vector (0,0,0,0).

8.2.

(a) Label the output of the AND gate realizing $C'E$ as c_5 and the output of the OR gate realizing $B'+E'$ as c_6. To detect A' s-a-0 by D-algorithm, the singular cover will set $A' = D$, $c_5 = 0$ and $c_1 = c_2 = D$. Performing backward implication, we get $(C, E) = (1,\phi)$ or $(\phi,0)$. Performing D-drive on the upper path with $B = 1$ and $c_4 = 0$ allows the propagation of D to output f. $c_4 = 0$ can be justified through backward implication by $c_6 = 0$, which in turn implies $(B, E) = (1,1)$. All of the above values can now be justified by $(A, B, C, E) = (0,1,1,1)$, which is hence one of the test vectors.

The other test vectors can be obtained by performing the D-drive through the lower path with $c_6 = 1$ (implying $(B, E) = (0,\phi)$ or $(\phi,0)$) and $c_3 = 0$. $c_3 = 0$ can be justified by $B = 0$. The test vectors that meet all the above conditions are $\{(0,0,\phi,0), (0,0,1,\phi)\}$.

Since the two paths have unequal inversion parity emanating from A' and reconverging at f, we must ensure that only one of these paths will be sensitized at a time.
(b) We assume that a fault on input B' is independent of the value of input B; similarly for inputs E' and E; that is, a fault can occur on just one of these inputs. The faults detected are: s-a-1 on lines A', B', C', E', c_1, c_3, c_4, c_5, c_6 and f.

8.3.

(a) The key idea is that when $F = 0$, both inputs of the OR gates can be sensitized simultaneously and, therefore, both subnetworks can be tested simultaneously. Thus, the number of tests resulting in $F = 0$ is determined by the subnetwork requiring the larger number of tests, namely,

$$n_f^0 = max(n_x^0, n_y^0)$$

However, when $F = 1$, only one input of the OR gate can be sensitized at a time. Thus,

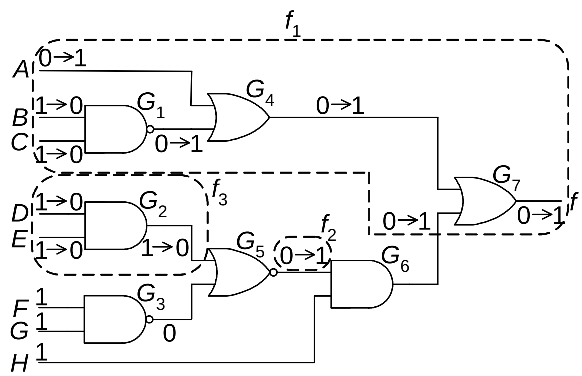
$$n_f^1 = n_x^1 + n_y^1$$

(*b*)

$$n_f^0 = max(n_x^1, n_y^1)$$

$$n_f^1 = n_x^0 + n_y^0$$

8.4.
(a, b) In the figure below, all possible single faults that result in an erroneous output value are shown. The equivalent faults are grouped into three equivalence classes f_1, f_2, and f_3.



8.5 Since f is always 0 in the presence of this multiple stuck-at fault, any vector that makes $f = 1$ in the fault-free case is a test vector: {(1,1,ϕ,ϕ), (ϕ,ϕ,1,1)}.

8.7. A minimal sequence of vectors that detects all stuck-open faults in the NAND gate: {(1,1), (0,1), (1,1), (1,0)}.

8.8. A minimal test set that detects all stuck-on faults in the NOR gate: {(0,0), (0,1), (1,0)}.

8.9. {(0,0,1), (0,1,1), (1,0,1), (1,1,0)}.

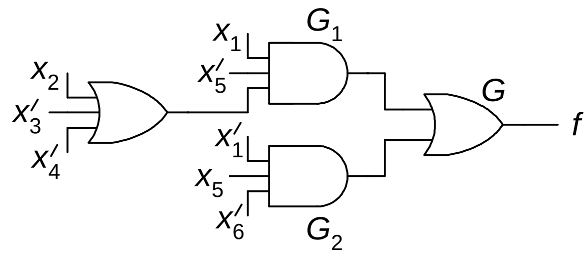
8.10.
(a) No. of nodes = 16. Therefore, no. of gate-level two-node BFs = 120 (i.e., $\binom{16}{2}$).
(b) Nodes that need to be considered = x_1, x_2, c_1, c_2, x_3, x_4, x_5, c_7, c_8, f_1, f_3. Therefore, no. of two-node BFs remaining = 55 (i.e., $\binom{11}{2}$).
(c)

Test set

x_1	x_2	x_3	x_4	x_5	c_1	c_2	c_7	c_8	f_1	f_3
1	0	0	1	0	1	0	0	0	1	0
1	1	0	0	0	0	1	1	1	0	1
0	1	1	1	1	1	0	0	0	1	0

Each member in the following set has identical values on each line in the member: $\{\{x_1\}, \{x_2\}, \{x_3, x_5\}, \{x_4, c_1, f_1\}, \{c_2, c_7, c_8, f_3\}\}$. Thus, the no. of BFs not detected = 0+0+1+3+6 = 10. Therefore, no. of BFs detected = 55−10 = 45.

8.11. BF $< c_1, c_2 >$ detected by $< c_1 = 0, c_2 = 1 >$ or $< c_1 = 1, c_2 = 0 >$.
(i) $< c_1 = 0, c_2 = 1 >$: $c_1 = 0$ for $x_1(x_2 + x_3' + x_4')$ and $c_2 = 1$ for x_5'. Thus, gate G_1 in the gate-level model below realizes $x_1(x_2 + x_3' + x_4')x_5'$.
(ii) $< c_1 = 1, c_2 = 0 >$: $c_1 = 1$ for x_1' and $c_2 = 0$ for x_5x_6'. Thus, gate G_2 realizes $x_1'x_5x_6'$.



Target fault: f SA0.

Tests

x_1	x_2	x_3	x_4	x_5	x_6
1	1	ϕ	ϕ	0	ϕ
1	ϕ	0	ϕ	0	ϕ
1	ϕ	ϕ	0	0	ϕ
0	ϕ	ϕ	ϕ	1	0

8.12.

Test set

Test	x_1	x_2	x_3	x_4	x_5	c_1	c_2	c_3	f
t_1	0	0	0	0	1	1	1	1	0
t_2	1	1	0	0	1	0	1	1	0
t_3	1	0	1	0	0	1	1	1	1
t_4	1	1	1	1	0	0	0	0	1
t_5	0	0	0	0	0	1	1	1	1
t_6	0	1	1	1	0	1	0	1	1

Essential vectors:
t_2: only vector to detect BF $< x_2, x_4 >$.
t_3: only vector to detect BF $< x_3, x_4 >$.
t_5: only vector to detect BF $< x_3, f >$.
t_6: only vector to detect BF $< c_2, c_3 >$.
Each member in the following set has identical values on each line in the member: $\{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{c_1, f\}, \{c_2\}, \{c_3\}\}$, when ($t_2$, t_3, t_5, t_6) is applied. Either t_1 or t_4 will distinguish between c_1 and

f. Therefore, there are two minimum test sets: $\{t_1, t_2, t_3, t_5, t_6\}$ or $\{t_2, t_3, t_4, t_5, t_6\}$.

8.13.

PDF and test

PDF	Test
$x_1c_2f \uparrow$	{(0,0), (1,0)}
$x_1c_2f \downarrow$	{(1,0), (0,0)}
$x_2c_3f \uparrow$	{(0,0), (0,1)}
$x_2c_3f \downarrow$	{(0,1), (0,0)}
$x_1c_1c_3f \downarrow$	{(0,1), (1,1)}
$x_2c_1c_2f \downarrow$	{(1,0), (1,1)}

8.14.

(a) The conditions for robustly detecting $\downarrow x_1'c_3f_1$ are: (i) $x_1' = U0$, (ii) $x_3' = S1$, (iii) $c_1 = c_2 = U0$. Robust two-pattern test: $\{(0,\phi,0), (1,0,0)\}$.

(b) To derive a robust two-pattern test for this fault, we need (i) $c_1 = U1$ and (ii) $c_2 = c_3 = S0$. Thus, the two-pattern test is {(1,0,0), (1,1,0)}.

8.17. Consider a two-output two-level AND-OR circuit that implements $f_1 = x_1x_2 + x_1x_2'x_3$ and $f_2 = x_1x_2'x_3 + x_2'x_3'$, in which the AND gate implementing $x_1x_2'x_3$ is shared between the two outputs. It is easy to check that neither output is based on an irredundant sum of products. However, the following test set detects all single stuck-at faults in the circuit: {(0,0,1), (0,1,0), (1,0,0), (1,0,1), (1,1,1)}.

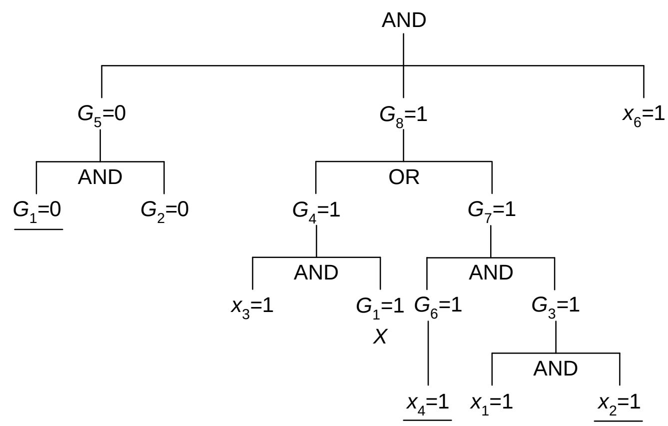
8.18. The multi-level circuit implements $f = (x_1x_2 + x_3)(x_4 + x_5) + (x_1x_2 + x_3)'x_6$.

Test set for the two-level circuit

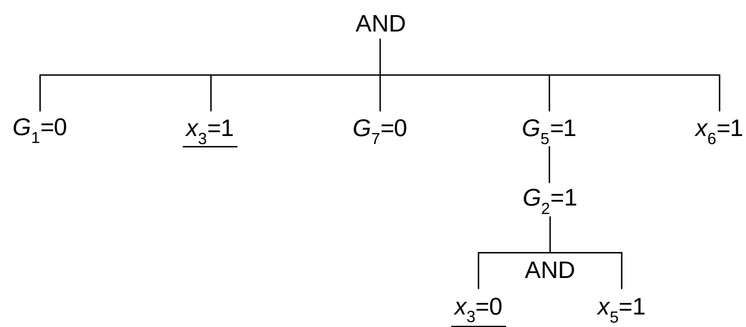
x_1	x_2	x_3	x_4	x_5	x_6	f
1	1	0	1	0	0	1
1	1	0	0	1	0	1
0	ϕ	1	1	0	0	1
0	ϕ	1	0	1	0	1
0	1	0	ϕ	ϕ	1	1
1	0	0	ϕ	ϕ	1	1
0	1	0	1	1	0	0
1	0	0	1	1	0	0
1	1	1	0	0	ϕ	0
1	1	0	0	0	1	0
0	0	1	0	0	1	0

The above test set can be shown to detect all single stuck-at faults in the above multi-level circuit as well.

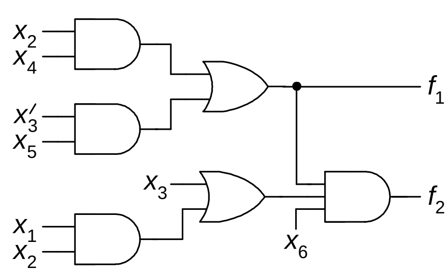
8.19. The mandatory assignments for an s-a-1 fault on the dashed connection are given in the figure below. The underlined assignments are impossible to justify simultaneously. Thus, this connection is redundant.



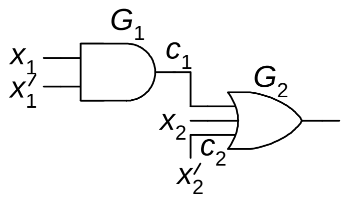
In the presence of the above redundant connection, c_1 s-a-1 and c_2 s-a-1 become redundant at f_2 (note that c_1 s-a-1 is not redundant at f_1). Redundancy of c_1 s-a-1 at f_2 can be observed from the mandatory assignments below (the underlined assignments are impossible to satisfy simultaneously).



Similarly, c_2 s-a-1 can be shown to be redundant. Even after removing the impact of c_1 s-a-1 at f_2 (basically, replacing G_4 by x_3), c_2 s-a-1 remains redundant. Thus, after removing these two faults, we get the following circuit.



8.20. Consider the redundant s-a-0 fault at c_1 in the following figure. Its region is only gate G_1. Next, consider the redundant s-a-1 fault at c_2. Its region is the whole circuit.



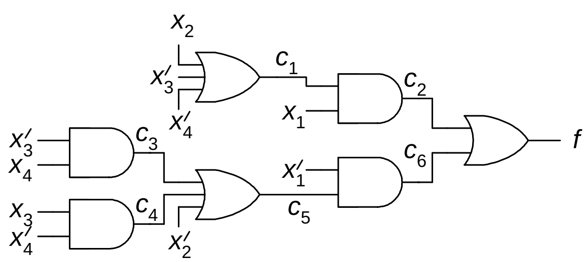
8.21. $f = x_1x_2 + x_1x_2x_3 + x_1x_2' = y_1 + y_2 + y_3$.

$DOBS_2 = [(y_1 + y_3) \oplus 1]' = (y_1 + y_3)$

Simplifying y_2 with don't cares $y_1 + y_3$, we get $y_2 = 0$ since don't care $y_1 = x_1x_2$ can be made 0. Therefore, $f = y_1 + y_3$ and $DOBS_1 = y_3$. Simplifying y_1 with don't cares in $y_3 = x_1x_2'$, we get $y_1 = x_1$ by making don't care $y_3 = 1$. Therefore, $f = y_1 + y_3$ where $y_1 = x_1$, $y_3 = x_1x_2'$. $DOBS_3 = y_1$. Simplifying y_3 with don't care y_1, we get $y_3 = 0$. Therefore, $f = y_1 = x_1$.

8.22. Expanding around x_1:

$f = x_1(x_2 + x_3'x_4 + x_3x_4' + x_3') + x_1'(x_2' + x_3'x_4 + x_3x_4') = x_1(x_2 + x_3' + x_4') + x_1'(x_2' + x_3'x_4 + x_3x_4')$. Its implementation is given below.



Partial test set

x_1	x_2	x_3	x_4
1	0	1	1
1	1	1	1
1	0	1	1
0	1	ϕ	ϕ
1	1	ϕ	ϕ
0	1	ϕ	ϕ

In the above table, the first two vectors constitute a robust two-pattern test for $\uparrow x_2c_1c_2f$ and the second and third vectors for $\downarrow x_2c_1c_2f$. The last three vectors are similarly robust two-pattern tests for rising and falling transitions along path x_1c_2f. Tests can be similarly derived for the other path delay faults.

8.23. $z = x_1x_2x_3^*x_5 + x_1x_3'x_4x_5 + x_1x_3x_4'x_5x_6 + x_1x_2'x_4x_5' + x_1'x_2x_4x_5' + x_1'x_2x_3'x_4' + x_2x_3'^{*}x_5^*x_6' + x_2x_3x_4x_5^* + x_2x_3'x_4'x_5' = (x_1 + x_4)x_2x_3x_5 + (x_1x_4 + x_2x_6')x_3'x_5 + x_1x_3x_4'x_5x_6 + x_1x_2'x_4x_5' + x_1'x_2x_4x_5' + x_1'x_2x_3'x_4' +$

$x_2x_3'x_4'x_5'$.

8.24. Proof available in Reference [8.11].

8.25. Proof available in Reference [8.11].

8.26. The threshold gate realizes the function $x_1x_2x_3 + x_2x_3x_4 + x_1x_2x_4$. The AND gate realizes the function $x_1'x_3'$. The test vector is: (0,1,1,0).

8.27. Proof available in Reference [8.11].

Chapter 9

9.1.

(*a*)

$$J_1 = y_2 \quad K_1 = y_2' \quad J_2 = x \quad K_2 = x' \quad z = x'y_2 + y_1y_2' + xy_1'$$

(*b*)

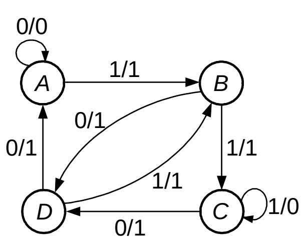
Excitation/output tables

	$x=0$		$x=1$		z	
y_1y_2	J_1K_1	J_2K_2	J_1K_1	J_2K_2	$x=0$	$x=1$
00	01	01	01	10	0	1
01	10	01	10	10	1	1
11	10	01	10	10	1	0
10	01	01	01	10	1	1

(*c*)

State table

	NS, z	
PS	$x=0$	$x=1$
$00 \to A$	$A, 0$	$B, 1$
$01 \to B$	$D, 1$	$C, 1$
$11 \to C$	$D, 1$	$C, 0$
$10 \to D$	$A, 1$	$B, 1$

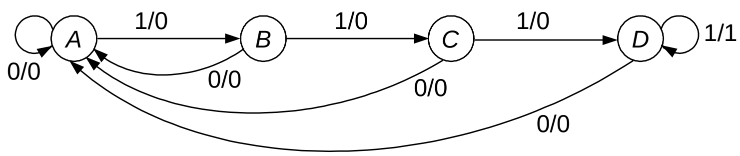


State diagram.

From the state diagram, it is obvious that the machine produces a zero output value if and only if the last three input values were identical (under the assumption that we ignore the first two output symbols).

9.2.

(a)



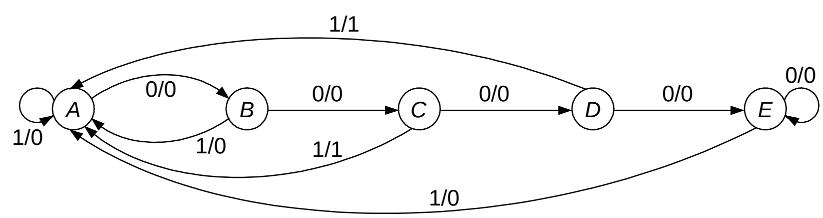
(b) Assignment: $A \to 00$, $B \to 01$, $C \to 10$, $D \to 11$

(c) Excitation and output functions:

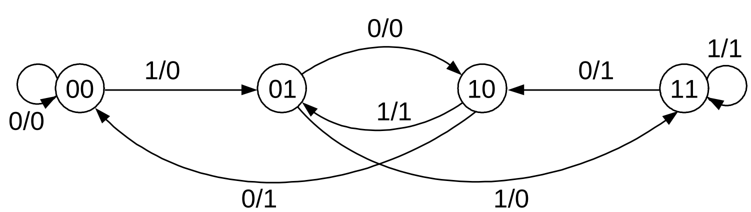
$$S_1 = xy_2, \quad R_1 = x', \quad S_2 = xy_2', \quad R_2 = x' + y_1'y_2, \quad z = xy_1y_2$$

9.5.

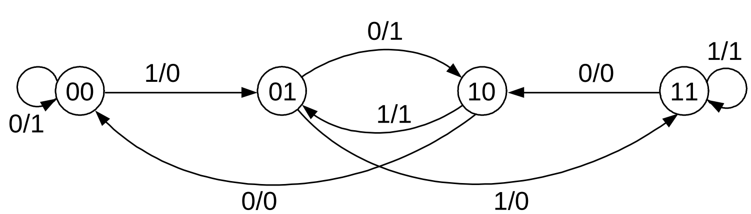
(a)



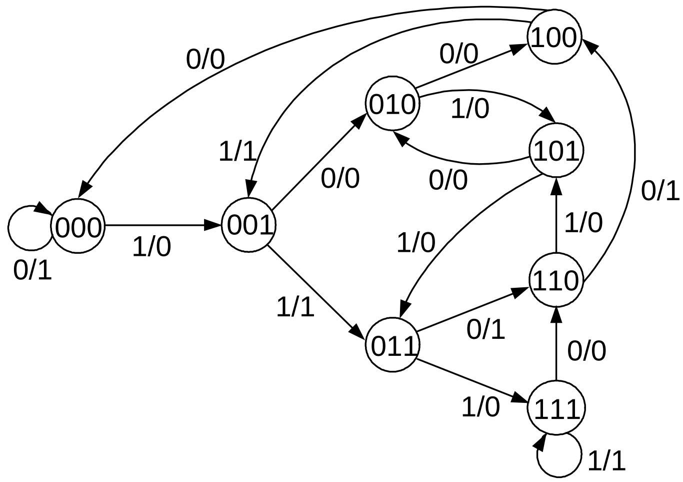
(b)



(c)

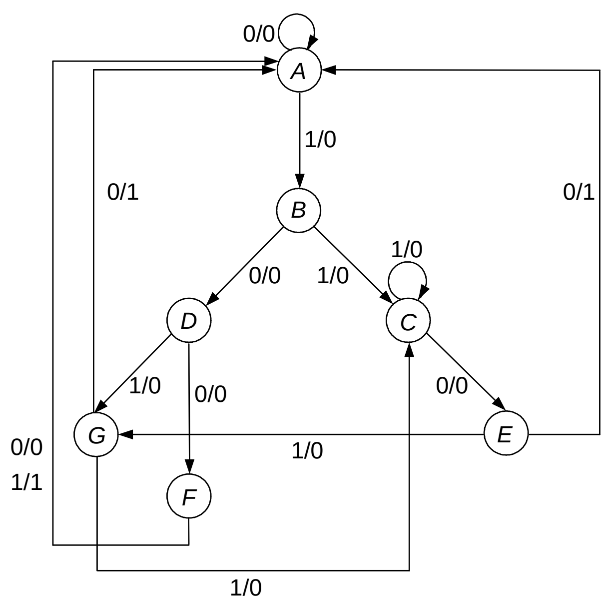


(*d*)

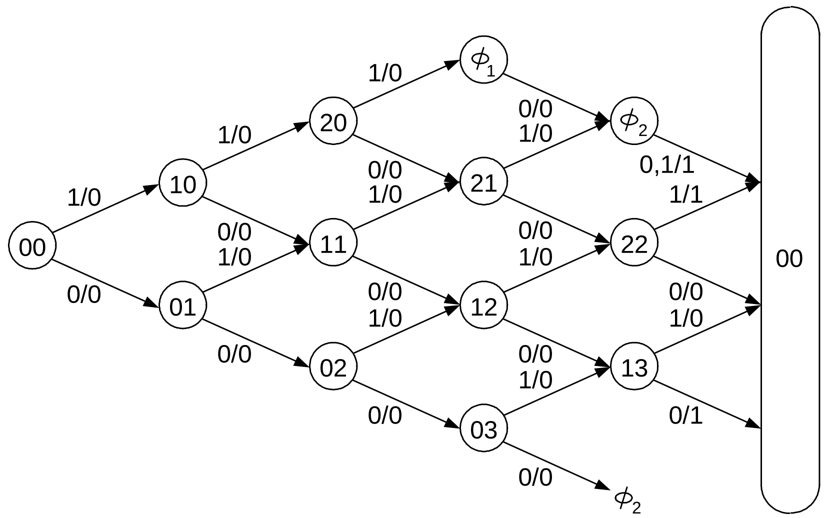


The labels of the states in (*b*), (*c*), and (*d*) were picked so that they will indicate the past input values, and make the construction of the state diagram easier.

9.6.



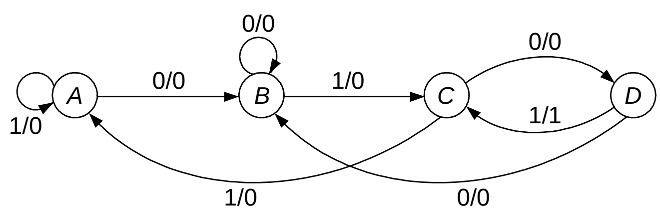
9.9.



The states are labeled in such a way that all the necessary information about the past input values is contained in the state label. The left digit indicates the number of 1's that have been received so far, and the right digit indicates the number of 0's.

9.11.

(a)



State table

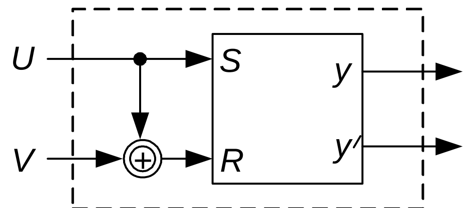
PS	NS, z $x=0$	$x=1$
A	$B, 0$	$A, 0$
B	$B, 0$	$C, 0$
C	$D, 0$	$A, 0$
D	$B, 0$	$C, 1$

(*b*)

Transition table

	Y_1Y_2		z	
y_1y_2	$x=0$	$x=1$	$x=0$	$x=1$
00	01	00	0	0
01	01	11	0	0
11	10	00	0	0
10	01	11	0	1

(*c*)



$$(S = U, R = U \oplus V) \Rightarrow (U = S, V = R \oplus S)$$

Thus, the excitation table of one flip-flop is:

State table

$y \to Y$	S	R	U	V
$0 \to 0$	0	ϕ	0	ϕ
$0 \to 1$	1	0	1	1
$1 \to 0$	0	1	0	1
$1 \to 1$	ϕ	0	0	0
	ϕ	0	1	1

Note that in the $1 \to 1$ case, we can choose U and V to be either 0's or 1's, but they must be identical.

Excitation table

	$x=0$		$x=1$		z	
y_1y_2	U_1V_1	U_2V_2	U_1V_1	U_2V_2	$x=0$	$x=1$
00	0ϕ	11	0ϕ	0ϕ	0	0
01	0ϕ	11	11	11	0	0
11	11	01	01	01	0	0
10	01	11	11	11	0	1

From the above table, we find one possible set of equations:

$$\begin{aligned} U_1 &= x'y_1y_2 + xy_1'y_2 + xy_1y_2' \\ V_1 &= 1 \end{aligned}$$

$$
\begin{aligned}
U_2 &= y_1'y_2 + y_1y_2' + x'y_1' \\
V_2 &= 1
\end{aligned}
$$

9.12.
(a) The excitation requirements for a single memory element can be summarized as follows:

Present state	*Next state*	*Required excitation*
0	0	0
0	1	1
1	0	ϕ
1	1	impossible

The state assignment must be chosen such that neither memory element will ever be required to go from state 1 to state 1. There are two such assignments:

Assignment α	*Assignment* β
$A \to 11$	$A \to 11$
$B \to 00$	$B \to 00$
$C \to 10$	$C \to 01$
$D \to 01$	$D \to 10$

Assignment α yields the following excitation table and logic equations.

	Y_1Y_2, z	
y_1y_2	$x = 0$	$x = 1$
11	$\phi\phi, 0$	$\phi\phi, 0$
00	$10, 0$	$11, 1$
10	$\phi 0, 0$	$\phi 1, 0$
01	$1\phi, 0$	$0\phi, 1$

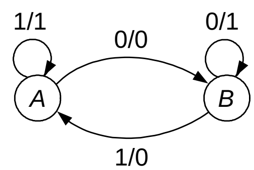
$$
\begin{aligned}
Y_1 &= x' + y_2' \\
Y_2 &= x \\
z &= xy_1'
\end{aligned}
$$

Assignment β yields the same equations except for an interchange of subscripts 1 and 2.
(b) The constraint that no memory element can make a transition from state 1 to state 1 means that not all state tables can be realized with a number of memory elements equal to the logarithm of the number of states. An n-state table can always be realized by using an n-variable assignment in which each state is represented by a "one-out-of-n" coding. Such assignments are not necessarily the best that can be achieved and, in general, it may be very difficult to find an assignment using the minimum number of state variables.

9.13. The reduced state table in standard form is as follows.

	NS, z	
PS	$x=0$	$x=1$
A	$A, 0$	$B, 1$
B	$A, 1$	$C, 1$
C	$A, 1$	$D, 1$
D	$E, 0$	$D, 1$
E	$F, 0$	$D, 0$
F	$A, 0$	$D, 0$

9.15.



Machine M

(b) An output of 0 from machine N identifies the final state as B, while an output of 1 identifies the final state as A. Once the state of N is known, it is straightforward to determine the input to N and the state to which N goes by observing its output. Consequently, except for the first bit, each of the subsequent bits can be decoded.

9.16. A simple strategy can be employed here whereby the head moves back and forth across the block comparing the end symbols and, whenever they are identical, replacing them with 2's and 3's (for 0's and 1's, respectively). At the end of the computation, the original symbols are restored.

		NS, *write*, *shift*			
PS	#	0	1	2	3
A	$B, \#, R$	$A, 0, R$	$A, 1, R$	$A, 0, R$	$A, 1, R$
B		$C, 2, R$	$D, 3, R$	$I, 2, L$	$I, 3, L$
C	$E, \#, L$	$C, 0, R$	$C, 1, R$	$E, 2, L$	$E, 3, L$
D	$F, \#, L$	$D, 0, R$	$D, 1, R$	$F, 2, L$	$F, 3, L$
E		$G, 2, L$	$H, 1, L$	$I, 2, L$	
F		$H, 0, L$	$G, 3, L$		$I, 3, L$
G		$G, 0, L$	$G, 1, L$	$B, 2, R$	$B, 3, R$
H	$A, \#, R$	$H, 0, L$	$H, 1, L$	$H, 2, L$	$H, 3, L$
I	$J, \#, R$	$I, 0, L$	$I, 1, L$	$I, 2, L$	$I, 3, L$
I	Halt	$J, 0, R$	$J, 1, R$	$J, 0, R$	$J, 1, R$
Halt	Halt	Halt	Halt		

state A. Starting state. The machine restores the original symbols of the present block and moves to test the next block.

state B. (columns 0,1): The head checks the leftmost original symbol of the block.

(columns 2,3): A palindrome has been detected.

states C and D. (#,0,1,2,3): The head searches for the rightmost original symbol of the block. State C is for the case that the leftmost symbol is a 0, and D for the case that it is a 1.

states E and F. (0,1): The head checks the rightmost symbol and compares it with the leftmost symbol. (E is for 0 and F is for 1.)

(2,3): At this point, the machine knows that the current symbol is the middle of an odd-length palindrome.

state G. The symbols that were checked so far can be members of a palindrome.

(0,1,2,3): The head searches for the leftmost original symbol of the block.

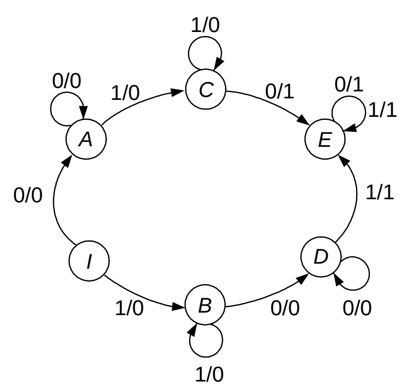
state H. At this point, the machine knows that the block is not a palindrome.

(0,1,2,3): The head goes to the beginning of the block, so that it will be ready to restore the original symbols (as indicated in state A).

state I. The machine now knows that the block is a palindrome. Therefore, the head goes to the beginning of the block, so that it will be ready to restore the original symbols (as indicated in state J).

states J and Halt: The original symbols are restored and the machine stops at the first # to the right of the block.

9.18.



State I is an initial state and it appears only in the first cell. It is not a necessary state for all other cells. We may design the network so that the first cell that detects an error will produce output value 1 and then the output values of the next cells are unimportant. In such a case, state E is redundant and the transitions going out from C and D with a 1 output value may be directed to any arbitrary state ϕ. The reduced state and excitation tables are as follows:

	NS, z_i	
PS	$x_i = 0$	$x_i = 1$
A	$A, 0$	$C, 0$
B	$D, 0$	$B, 0$
C	$\phi, 1$	$C, 0$
D	$D, 0$	$\phi, 1$

	$Y_{i1}Y_{i2}, z_i$	
$y_{i1}y_{i2}$	$x_i = 0$	$x_i = 1$
00	$00, 0$	$11, 0$
01	$10, 0$	$01, 0$
11	$\phi\phi, 1$	$11, 0$
10	$10, 0$	$\phi\phi, 1$

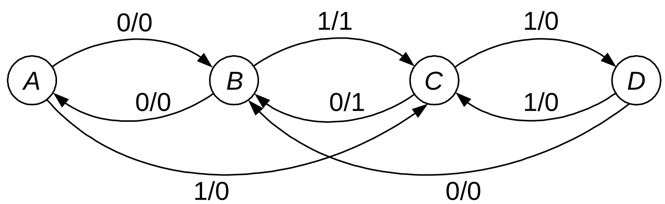
$$z_i = x_i' y_{i1} y_{i2} + x_i y_{i1} y_{i2}'$$

$$
\begin{aligned}
Y_{i1} &= x_i' y_{i2} + x_i y_{i2}' + y_{i1} \\
Y_{i2} &= x_i
\end{aligned}
$$

If state E is retained, the realization will require three state variables.

9.19.

(a)



State A corresponds to an even number of zeros, B to an odd number of zeros, C to an odd number of ones, and D to an even number of ones.

	NS, z_i	
PS	$x_i = 0$	$x_i = 1$
A	$B, 0$	$C, 0$
B	$A, 0$	$C, 1$
C	$B, 1$	$D, 0$
D	$B, 0$	$C, 0$

(b) The assignment and the corresponding equations are given next: $A \to 00$, $B \to 01$, $C \to 11$, $D \to 10$.

$$
\begin{aligned}
Y_{i1} &= x_i \\
Y_{i2} &= y_{i2}' + x_i y_{i1}' y_{i2} + x_i' y_{i1} y_{i2} \\
z_i &= x_i y_{i1}' y_{i2} + x_i' y_{i1} y_{i2}
\end{aligned}
$$

9.20.

	NS, z_i	
PS	$x_i = 0$	$x_i = 1$
A	$A, 0$	$B, 1$
B	$C, 0$	$D, 1$
C	$A, 1$	$B, 0$
D	$C, 1$	$D, 0$

	$Y_{i1}Y_{i2}, z_i$	
$y_{i1}y_{i2}$	$x_i = 0$	$x_i = 1$
00	00, 0	01, 1
01	11, 0	10, 1
11	00, 1	01, 0
10	11, 1	10, 0

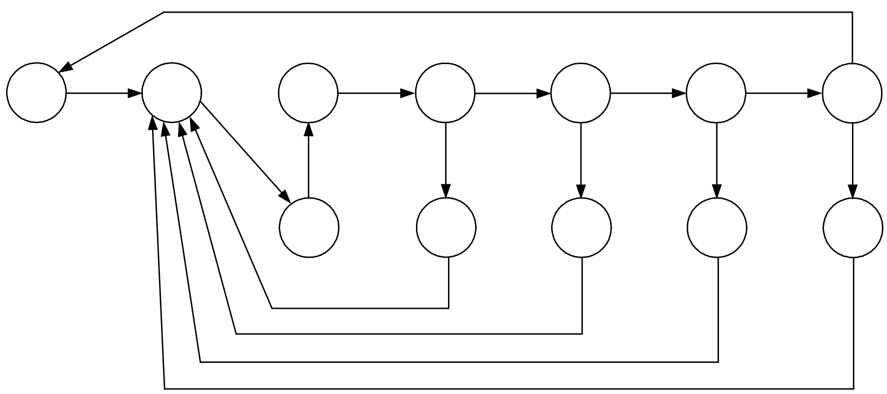
$$
\begin{aligned}
Y_{i1} &= y_{i1} \oplus y_{i2} \\
Y_{i2} &= x_i \oplus y_{i1} \oplus y_{i2} \\
z_i &= x_i \oplus y_{i1}
\end{aligned}
$$

Chapter 10

10.1.

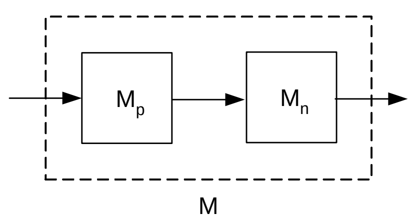
(a) Suppose the starting state is S_1, then one input symbol is needed to take the machine to another state, say S_2. Since the machine is strongly connected, *at most* two symbols are needed to take the machine to a third state, say S_3. (This occurs if S_3 can be reached from S_1 and not from S_2.) By the same line of reasoning, we find that the fourth distinct state can be reached from S_3 by at most three input symbols. Thus, for an n-state machine, the total number of symbols is at most $\sum_{i=1}^{n-1} i = \frac{n}{2}(n-1)$. This is not a least upper bound. It is fairly easy to prove, for example, that the least upper bound for a three-state machine is 2 and not 3, which is given by the above bound. In other words, there exists no three-state strongly connected machine that needs more than two input symbols to go through each of its states once.

(b)

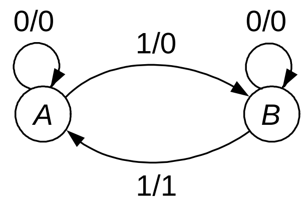


10.2.

(a) Let M_p be a p-state machine that responds to a sequence of 1's by producing a periodic output sequence with period p. (Clearly, such a machine can always be constructed.) Consider now the composite machine shown below, where the output of M_p is the only input to M_n. If M_n has n states, the composite machine M can have at most np states. Also, since its input is a string of 1's, its output will be periodic with period at most np. Machine M_n now receives an input sequence with period p and produces an output sequence with period np.



The proof that np is a least upper bound is straightforward. For example, the machine below responds to input sequence 001001001... by producing an output sequence with period 6.



(b) The output of M_1^* (Table 10.2) becomes periodic with period 1 after a transient period of eight symbols.

10.3. If there exists an n-state machine that accepts all palindromes, then it accepts the following sequence: $\underbrace{00\cdots00}_{n+1}1\underbrace{00\cdots00}_{n+1}$. However, during the first $n+1$ time units, the machine must have visited some state twice, say S_i. Thus, the machine will also accept sequence $001\overbrace{00\ldots00}_{n+1}$, which is formed by deleting from the original sequence the subsequence contained between the two visits to S_i.

10.4.

(a) Not realizable, since it must be capable of counting arbitrary numbers of 1's and 0's.

(b) Not realizable, since it must store number π which contains an infinite number of digits.

10.5.

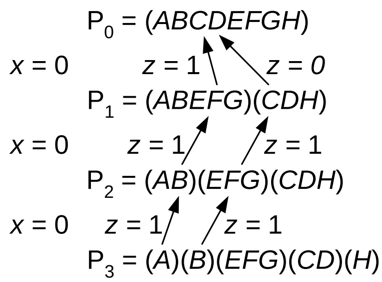
(a)

$$\begin{aligned}
P_0 &= (ABCDEFGH) \\
P_1 &= (ABEFG)(CDH) \\
P_2 &= (AB)(EFG)(CDH) \\
P_3 &= (A)(B)(EFG)(CD)(H) \\
P_4 &= (A)(B)(EFG)(CD)(H) = \text{Equivalence partition}
\end{aligned}$$

(b)

PS	NS, z $x=0$	$x=1$
$(A) \to \alpha$	$\beta, 1$	$\epsilon, 1$
$(B) \to \beta$	$\gamma, 1$	$\delta, 1$
$(EFG) \to \gamma$	$\delta, 1$	$\delta, 1$
$(CD) \to \delta$	$\delta, 0$	$\gamma, 1$
$(H) \to \epsilon$	$\delta, 0$	$\alpha, 1$

(c) The first partition in which A and B appear in different blocks is P_3; thus the shortest distinguishing sequence between A and B is of length 3. For the first symbol in the sequence, select any symbol I_j for which the successors of A and B lie in different blocks of P_2. In this case, $x = 0$ is appropriate, as shown below. The 0-successors of A and B are B and F, respectively. Select now a symbol I_k for which the I_k-successors of B and F are in different blocks of P_1. Again $x = 0$ is chosen. Finally, choose an input symbol that will produce different output symbols when applied to these two states in P_1. From the illustration below, it is evident that 000 is the minimum-length sequence that distinguishes state A from state B.



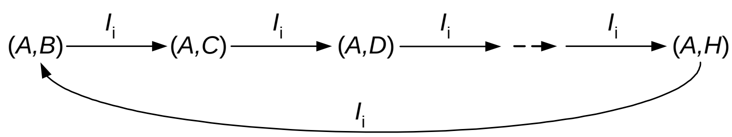
10.6.

(a) $P = (A)(BE)(C)(D)$

(b) $P = (AB)(C)(D)(E)(FG)$

(c) $P = (A)(B)(CD)(EFG)(H)$

10.7. From column I_i, we find

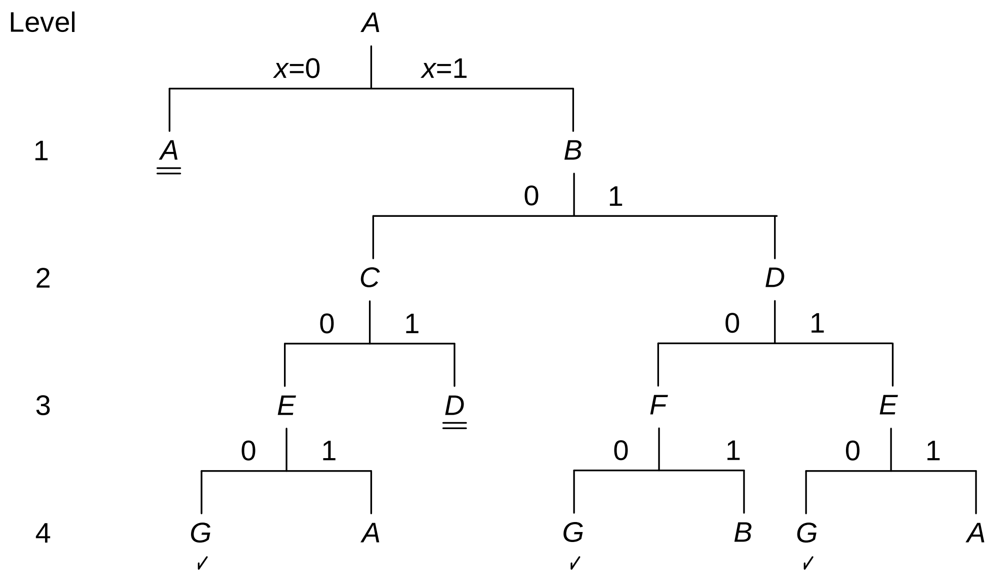


Thus, if A is equivalent to any one of the other states, it must be equivalent to all of them. From column I_j, we conclude that if state B is equivalent to any other state, then A must also be equivalent to some state, which means that all the states are equivalent. Using the same argument, we can show that no two states are equivalent unless all the states are equivalent.

10.8.

(a) Construct a tree which starts with S_i and lists in the first level the 0- and 1-successors of S_i, in the second level the successors of the states in the first level, and so on. A branch is terminated if it is associated with a state already encountered at a previous level. The shortest transfer sequence $T(S_i, S_j)$ is obtained from the path in the tree leading from S_i to an appearance of S_j in the lowest-numbered level.

(*b*)

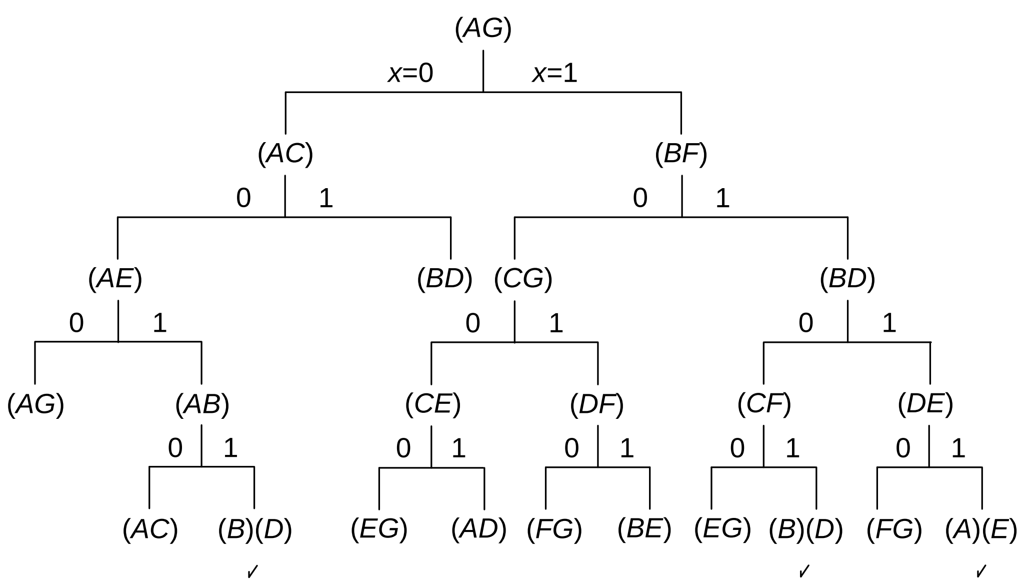


There are three minimal transfer sequences $T(A, G)$, namely 1110, 1100, and 1000.

10.9.

(*a*) Construct a tree which starts with the pair (S_iS_j). Then, for every input symbol, the two successor states will be listed separately if they are associated with different output symbols, and will form a new pair if the output symbols are the same. A branch of the tree is terminated if it is associated with a pair of states already encountered at a previous level, or if it is associated with a pair of identical states, i.e., (S_kS_k). A sequence that distinguishes S_i from S_j is one that can be described by a path in the tree that leads from (S_iS_j) to any two separately listed states, e.g., $(S_p)(S_q)$.

(*b*)



There are five sequences of four symbols each that can distinguish A from G: 1111, 1101, 0111, 0101, 0011.

10.10.

(a)

$$
\begin{aligned}
P_0 &= (ABCDEFGH) \\
P_1 &= (AGH)(BCDEF) \\
P_2 &= (AH)(G)(BDF)(CE) \\
P_3 &= (AH)(G)(BDF)(CE)
\end{aligned}
$$

Thus, A is equivalent to H.

(b) From P_3, we conclude that for each state of M_1, there exists an equivalent state in M_2 that is, H is equivalent to A, F is equivalent to B and D, and E is equivalent to C.

(c) M_1 and M_2 are equivalent if their initial states are either A and H, or B and F, or D and F, or C and E.

10.13.

$$
\begin{array}{ll}
X: & 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 0 \\
S: & \underline{A}\ \underline{B}\ \underline{A}\ \underline{\underline{B}}\ \underline{\underline{A}}\ \underline{\underline{\underline{A}}}\ B\ \underline{C}\ \underline{C}\ \underline{\underline{C}}\ \underline{\underline{C}}\ \underline{\underline{\underline{A}}} \\
Z: & 1\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1
\end{array}
$$

Observing the input and output sequences, we note that during the experiment the machine responded to 00 by producing 10, 01, and 00. This indicates that the machine has three distinct states, which we shall denote A, B, and C, respectively. (These states are marked by one underline). At this point, we can determine the transitions from A, B, and C under input symbol 0, as indicated by the entries with double underlines. In addition, we conclude that A is the only state that responds by producing a 1 output symbol to a 0 input symbol. Thus, we can add two more entries (A with three underlines) to the state-transition row above. The machine in question indeed has three states and its state transitions and output symbols are given by the state table below:

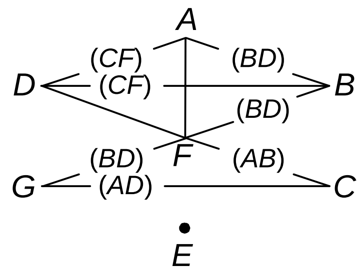
	NS, z	
PS	$x = 0$	$x = 1$
A	$B, 1$	$A, 0$
B	$A, 0$	$C, 1$
C	$C, 0$	$A, 0$

10.16. Find the direct sum M of M_1 and M_2. M has $n_1 + n_2$ states and we are faced with the problem of establishing a bound for an experiment that distinguishes between S_i and S_j in M. From Theorem

10.2, the required bound becomes evident.

10.17. Modify the direct sum concept (see Problem 10.10) to cover incompletely specified machines.

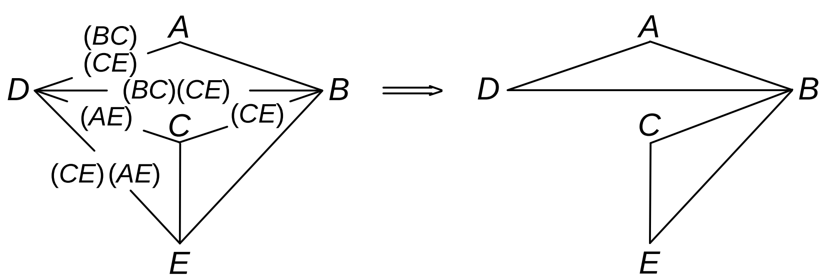
10.18.



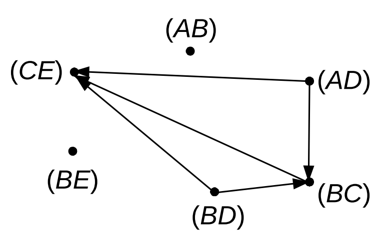
The set of maximal compatibles is $\{(ABDF), (CFG), (E)\}$. This set is a minimal closed covering. Note that the set $\{(ABD), (CFG), (E)\}$ is also a minimal closed covering.

	NS, z	
PS	$x = 0$	$x = 1$
$(ABDF) \to \alpha$	$\alpha, 0$	$\beta, 1$
$(CFG) \to \beta$	$\alpha, 0$	$\gamma, 0$
$(E) \to \gamma$	$\beta, 1$	α or β ,0

10.19.
(a)



The maximal compatibles are $\{(ABD)$,$(BCE)\}$. The compatibility graph shown below can be covered by vertices (AD), (BC), and (CE). However, (BC) and (CE) may be combined to yield (BCE).

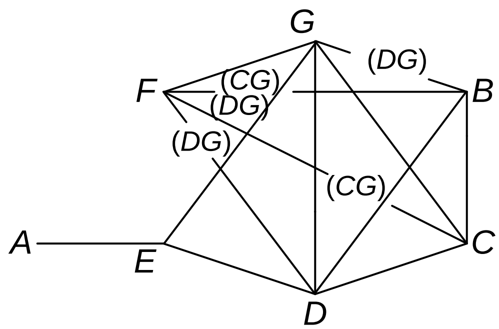


Let (AD) be denoted by α and (BCE) by β, then the reduced table is given by

PS	NS, z		
	I_1	I_2	I_3
α	$\beta, 0$	$\beta, 1$	$\beta, -$
β	$\beta, 0$	$\beta, 0$	$\alpha, -$

(b) The minimal set of compatibles is $\{(AB), (AE), (CF), (DF)\}$.

10.21.



The set of maximal compatibles is $\{(AE), (BCDFG)\}$. Thus, the minimal machine that contains the given one is

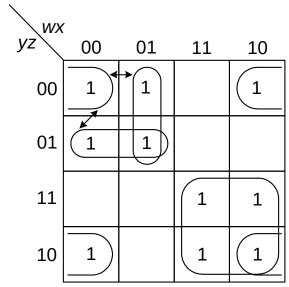
PS	NS, z_1z_2			
	00	01	11	10
$(AE) \rightarrow \alpha$	$\alpha, 00$	$\alpha, 01$	$\beta, 00$	$\alpha, 01$
$(BCDFG) \rightarrow \beta$	$\alpha, 00$	$\beta, 10$	$\beta, 00$	$\beta, 11$

y	SR, z_1z_2			
	00	01	11	10
0	$0\phi, 00$	$0\phi, 01$	$10, 00$	$0\phi, 01$
1	$01, 00$	$\phi 0, 10$	$\phi 0, 00$	$\phi 0, 11$

$$
\begin{aligned}
S &= x_1x_2 \\
R &= x_1'x_2' \\
z_1 &= yx_1'x_2 + yx_1x_2' \\
z_2 &= x_1x_2' + y'x_1'x_2
\end{aligned}
$$

Chapter 11

11.1.



$f_a(w,x,y,z) = x'z' + wy + w'xy' + w'y'z$

The hazards are indicated by the small double arrows. (Since the function is realized in sum-of-products form, there are no hazards among the zeros.) To eliminate the hazards, simply remove the first OR gate (that computes $x + z$).

11.2. Add a connection from the AND gate realizing xz' (yz') to the input of the first (second) OR gate.

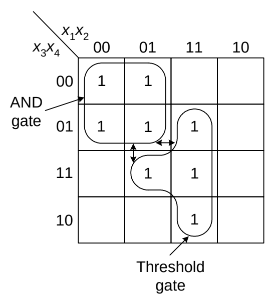
11.3.

(a) Counter-example: $f_1 = xy' + x'y$, $f_2 = yz' + y'z$.

(b) Counter-example: the two circuits in Fig. P11.2.

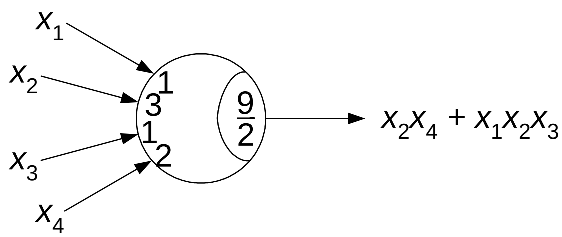
11.5.

(a) The threshold gate realizes the function $x_1x_2x_3+x_2x_3x_4+x_1x_2x_4$; the AND gate realizes the function $x_1'x_3'$. The function $f(x_1, x_2, x_3, x_4)$ is, therefore, as shown in the map below:



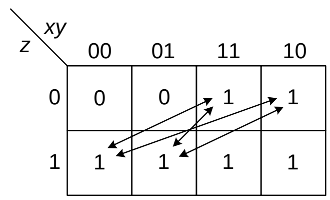
As indicated by the arrows, there are two static hazards. (Since the function is realized in a sum-of-products form, there are no hazards among the zeros.)

(b) To eliminate the hazards, the threshold element must be redesigned to also cover the cell $x_1'x_2x_3'x_4$, that is, it should realize the function $x_2x_4 + x_1x_2x_3$. One possible solution is:

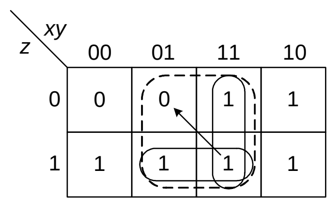


11.7.

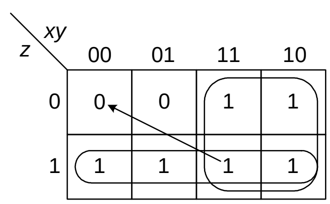
(a) The four bidirectional arrow in the K-map below show the eight MIC transitions that have a function hazard.



(b) The required cubes $\{xy,\, yz\}$ and privileged cube $\{y\}$ are shown below.

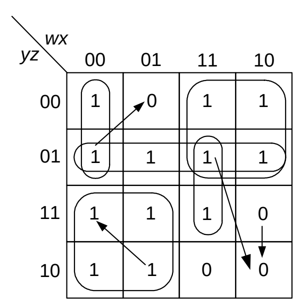


(c) The required cubes $\{x,\, z\}$ are shown below.



11.8.

(a) The required cubes, privileged cubes and dhf-prime implicants are shown below.



(b) Hazard-free sum of products: $w'x'y' + y'z + wy' + w'y + wxz$.

11.12. The reduced table shown below is derived from the following primitive table and the corresponding merger graph.

State, output			
x_1x_2			
00	01	11	10
1,0	2	–	3
1	**2**,0	4	–
1	–	5	**3**,0
–	2	**4**,0	3
–	2	**5**,1	6
7	–	5	**6**,1
7,1	8	–	6
7	**8**,1	5	–

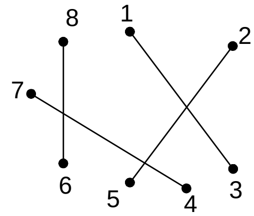
Merge: (1,2,4), (3), (5), (6,7,8)

y_1y_2	Y_1Y_2, z			
	x_1x_2			
	00	01	11	10
$(5) \to 00$	–,0	2,0	**5**,1	6,1
$(3) \to 01$	1,0	–,0	5,1	**3**,0
$(1,2,4) \to 11$	**1**,0	**2**,0	**4**,0	3,0
$(6,7,8) \to 10$	**7**,1	**8**,1	5,1	**6**,1

$$
\begin{aligned}
Y_1 &= x_1'y_2 + x_1'y_1 + x_2y_1y_2 + x_1x_2'y_2' + x_2'y_1y_2' \\
Y_2 &= x_1'y_1' + x_1'y_2 + y_1y_2 + x_2'y_2 \\
z &= y_2'(x_1 + y_1) + x_1x_2y_1'
\end{aligned}
$$

11.13. The primitive and reduced flow tables are as follows:

State, output			
x_1x_2			
00	01	11	10
1,10	2	–	3
5	**2**,01	4	–
1	–	6	**3**,10
–	7	**4**,10	8
5,01	2	–	3
–	7	**6**,01	8
5	**7**,10	4	–
1	–	6	**8**,01



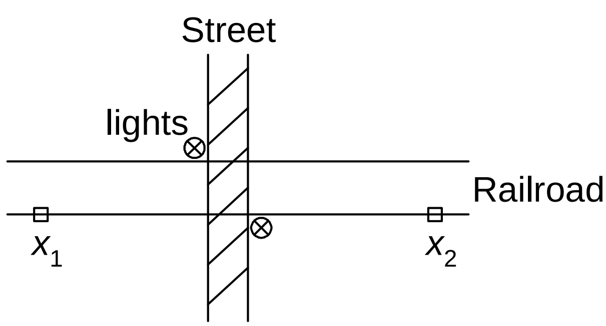
y_1y_2	Y_1Y_2, z_1z_2 x_1x_2 00	01	11	10
$(1,3) \to 00$	**1**,10	2	6	**3**
$(2,5) \to 01$	**5**,01	**2**,01	4	3
$(4,7) \to 11$	5	**7**,10	**4**,10	8
$(6,8) \to 10$	1	7	**6**,01	**8**,01

$$\begin{aligned} Y_1 &= x_1x_2 + x_2y_1 + x_1y_1 \\ &= [(x_1x_2)'(x_2y_1)'(x_1y_1)']' \\ Y_2 &= x_1'x_2 + x_2y_2 + x_1'y_2 \\ &= [(x_1'x_2)'(x_2y_2)'(x_1'y_2)']' \end{aligned}$$

If we regard the unspecified outputs as don't-care combinations, we obtain

$$\begin{aligned} z_1 &= y_1'y_2' + y_1y_2 = [(y_1'y_2')'(y_1y_2)']' \\ z_2 &= y_1'y_2 + y_1y_2' = [(y_1'y_2)'(y_1y_2')']' \end{aligned}$$

11.15.



(a) Switches x_1 and x_2 are each placed 1500 feet from the intersection. The primitive flow table that describes the light control is as follows:

State, output				
x_1x_2				
00	01	11	10	
1,0	2	–	6	No train in intersection
3	**2**,1	–	–	A train pressed x_1
3,1	5	–	4	A train is between switches x_1 and x_2
1	–	–	**4**,1	The train now leaves the intersection
1	**5**,1	–	–	The train now leaves the intersection
3	–	–	**6**,1	A train pressed x_2

The states that can be merged are: (1), (2,6), (3), (4,5)
The reduced flow table can now be written as

y_1y_2	Y_1Y_2, z			
	x_1x_2			
	00	01	11	10
$(1) \rightarrow 00$	**1**,0	2	–	6
$(2,6) \rightarrow 01$	3	**2**,1	–	**6**,1
$(3) \rightarrow 11$	**3**,1	5	–	4
$(4,5) \rightarrow 10$	1	**5**,1	–	**4**,1

$$
\begin{aligned}
Y_1 &= (x_1 + x_2)y_1 + x_1'x_2'y_2 \\
Y_2 &= (x_1 + x_2)y_1' + x_1'x_2'y_2 \\
z &= y_1 + y_2
\end{aligned}
$$

The design of the output is made so as to achieve minimal complexity.

11.16.

State, output				
x_1x_2				
00	01	11	10	
1,0	–	–	2	Empty room
3	–	–	**2**,1	One enters room
3,1	6	–	4	One in room
5	–	–	**4**,1	Second enters room
5,1	7	–	–	Two in room
1	**6**,1	–	–	Room being vacated
3	**7**,1	–	–	One (of two) leaves room

Merge rows: (1,6), (2,7), (3), (4,5)

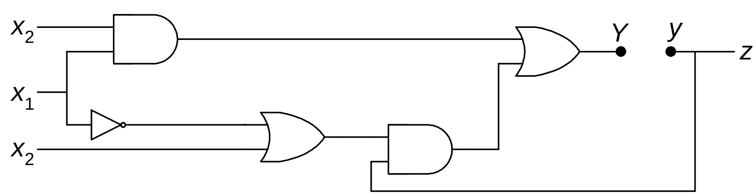
11.17. A reduced flow table that accomplishes the control in question is shown below:

State, output P_1P_2 00	01	11	10
A,1	D	C	**A**,1
B,0	D	C	**B**,0
B	**C**,0	**C**,0	B
A	**D**,0	**D**,0	A

11.19.

(a) Whenever the inputs assume the values $x_1 = x_2 = 1$, the circuit output becomes 1 (or remains 1 if it was already 1). Whenever the inputs assume the values $x_1 = 1$, $x_2 = 0$, the output becomes (or remains) 0. Whenever the inputs assume any other combination of values, the output retains its previous value.

(b) The flow table can be derived by breaking the feedback loop as shown below:



The excitation and output equations are then

$$
\begin{aligned}
Y &= x_1x_2 + (x_1' + x_2)y \\
z &= y
\end{aligned}
$$

(c) The same function for Y can be achieved by eliminating the x_2y term. This, however, creates a static hazard in Y when $y = 1$, $x_2 = 1$ and x_1 changes between 0 and 1. Hence, proper operation would require the simultaneous change of two variables. A simple solution is to place an inertial or smoothing delay between Y and y in the figure shown above.

11.20.

$y_1y_2y_3$	$Y_1Y_2Y_3$ x_1x_2 00	01	11	10
$a \to 000$	000	001	000	001
$a \to 001$	001	101	001	111
$a \to 011$	011	001	011	111
$b \to 010$	{001,011}	010	110	010
$b \to 100$	000	100	101	100
$b \to 101$	101	101	101	111
$c \to 111$	111	110	011	111
$d \to 110$	000	110	100	110

The assignment in column 00 row 010 is made so as to allow minimum transition time to states c and d. In column 00, row 110 and column 10, row 001, we have noncritical races.

Chapter 12

12.1. Since the machine has n states, $k = \lceil \log_2 n \rceil$ state variables are needed for an assignment. The problem is to find the number of distinct ways of assigning n states to 2^k codes. There are 2^k ways to assign the first state, $2^k - 1$ ways to assign the second state, etc., and $2^k - n + 1$ ways to assign the nth state. Thus, there are

$$2^k \cdot (2^k - 1) \cdot \ldots \cdot (2^k - n + 1) = \frac{2^k!}{(2^k - n)!}$$

ways of assigning the n states.

There are $k!$ ways to permute (or rename) the state variables. In addition, each state variable can be complemented, and hence there are 2^k ways of renaming the variables through complementation. Thus, the number of distinct state assignments is

$$\frac{2^k!}{(2^k - n)!} \cdot \frac{1}{k! 2^k} = \frac{(2^k - 1)!}{(2^k - n)! k!}$$

12.2.

Assignment α:

$$\begin{aligned}
Y_1 &= x'y_1'y_2'y_3 + x'y_2y_3' + xy_2y_3 = f_1(x, y_1, y_2, y_3) \\
Y_2 &= y_1'y_2'y_3' + xy_2' + y_1y_3 = f_2(x, y_1, y_2, y_3) \\
Y_3 &= x'y_1'y_2' + x'y_2y_3 + xy_2y_3' + xy_1 = f_3(x, y_1, y_2, y_3) \\
z &= xy_1'y_2'y_3 + x'y_2y_3 + x'y_1 + y_1y_3' = f_0(x, y_1, y_2, y_3)
\end{aligned}$$

Assignment β:

$$\begin{aligned}
Y_1 &= xy_1 + x'y_1' = f_1(x, y_1) \\
Y_2 &= x'y_3 = f_2(x, y_3) \\
Y_3 &= y_2'y_3' = f_3(y_2, y_3) \\
z &= x'y_1' + xy_3 = f_0(x, y_1, y_3)
\end{aligned}$$

12.3. No, since $\pi_1 + \pi_2$ is not included in the set of the given closed partitions.

12.5. Suppose π is a closed partition. Define a new partition

$$\pi' = \sum \{\pi_{S_iS_j} | S_i \text{ and } S_j \text{ are in the same block of} \pi\}$$

Since π' is the sum of closed partitions, it is itself closed. (Note that π' is the sum of some basic partitions.) Furthermore, since $\pi \geq \pi_{S_iS_j}$, for all S_i and S_j which are in the same block of π, we must have $\pi \geq \pi'$. To prove that in fact $\pi = \pi'$, we note that if $\pi > \pi'$, then π would identify some pair S_iS_j which is not identified by π', but this contradicts the above definition of π'. Thus, $\pi' = \pi$.

Since any closed partition is the sum of some basic partitions, by forming all possible sums of the basic partitions we obtain all possible closed partitions. Thus, the procedure for the construction of the π-lattice, outlined in Sec. 12.3, is verified.

12.7.

(a) Let us prove first that if π is a closed and output-consistent partition, then all the states in the same block of π are equivalent. Let A and B be two states in some block of π. Since π is closed, the k-successors of A and B, where $k = 1, 2, 3, \cdots$, will also be in the same block of π. Also, since π is output consistent, just one output symbol is associated with every block of π. Thus, A and B must be k-equivalent for all k's.

To prove the converse, let us show that if M is *not* reduced, there exists some closed partition on M which is also output consistent. Since M is not reduced, it has a nontrivial equivalence partition P_k. Clearly, P_k is closed and output consistent. Thus, the above assertion is proved.

(b) $\lambda_o = \{\overline{A, C, F}; \overline{B, D, E}\}$

The only closed partition $\pi \leq \lambda_o$ is

$$\pi = \{\overline{A, F}; \overline{B, E}; \overline{C}; \overline{D}\} = \{\alpha; \beta; \gamma; \delta\}$$

The reduced machine is

	NS		
PS	$x = 0$	$x = 1$	z
α	β	γ	0
β	β	α	1
γ	β	δ	0
δ	β	γ	1

12.8. For the machine in question:

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

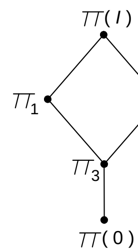
We can also show that the partition

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

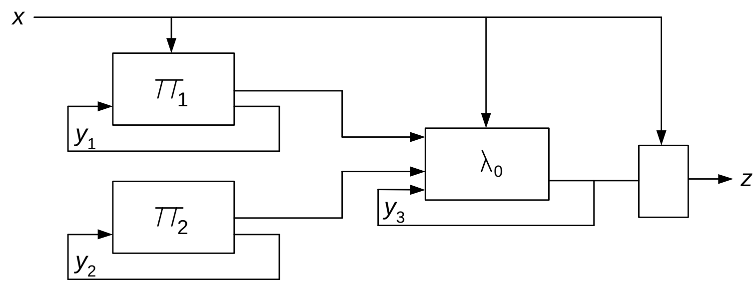
is closed. However, there is no way of specifying the unspecified entries in such a way that the partition $\{\overline{A, B}; \overline{C, D}\}$ will be input consistent and closed. (Note that, in general, in the case of incompletely specified machines, if π_1 and π_2 are closed partitions, $\pi_3 = \pi_1 + \pi_2$ is not necessarily a closed partition.)

12.9.

(i) Find $\pi_3 = \pi_1 \cdot \pi_2 = \{\overline{A, F}; \overline{B, E}; \overline{C, H}; \overline{D, G}\}$. The π-lattice is given by



Assign y_1 to π_1, y_2 to π_2, and y_3 to λ_o. Since $\pi_2 \geq \lambda_i$, y_2 is input independent. The schematic diagram is given as follows:

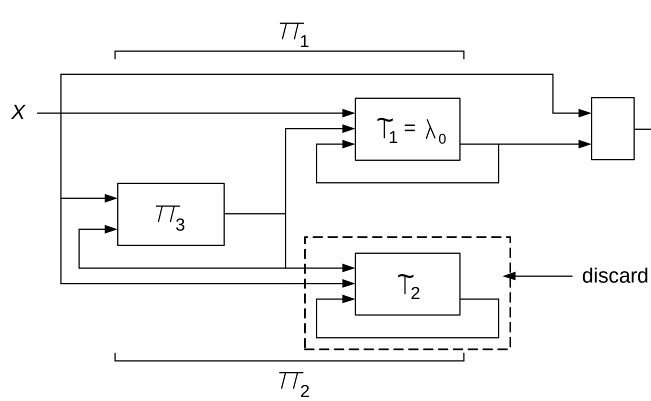


(ii)

$$
\begin{aligned}
\pi_1 &= \{\overline{A,B};\overline{C,D};\overline{E,F};\overline{G,H}\} \\
\pi_2 &= \{\overline{A,E};\overline{B,F};\overline{C,G};\overline{D,H}\} \\
\pi_3 &= \pi_1 + \pi_2 = \{\overline{A,B,E,F};\overline{C,D,G,H}\} \\
\lambda_o &= \lambda_i = \{\overline{A,B,C,D};\overline{E,F,G,H}\}
\end{aligned}
$$

Note that $\pi_1 \cdot \pi_2 = 0$ is a parallel decomposition, but requires four state variables. However, both π_1 and π_2 can be decomposed to yield the same predecessor, i.e.,

$$
\begin{aligned}
\pi_3 \cdot \tau_1 &= \pi_1 \\
\pi_3 \cdot \tau_2 &= \pi_2 \\
\tau_1 &= \{\overline{A,B,C,D};\overline{E,F,G,H}\} = \lambda_o = \lambda_i \\
\tau_2 &= \{\overline{A,C,E,G};\overline{B,D,F,H}\}
\end{aligned}
$$



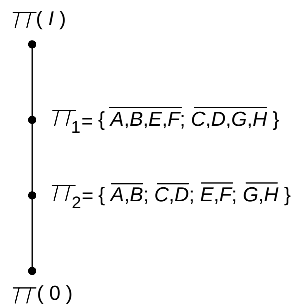
Since $\tau_1 = \lambda_o$, the output can be obtained from the τ_1 machine, and evidently the τ_2 machine is unnecessary. (Note that $\pi_1 \leq \lambda_o$; thus the machine can be reduced, as shown by the discarded submachine.)

12.10.

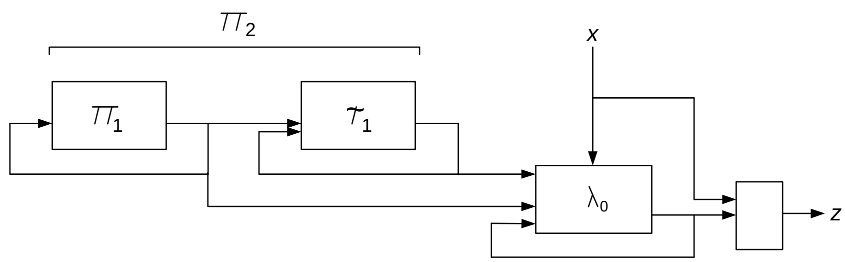
(a)

$$\begin{aligned} \lambda_i &= \{\overline{A,B};\overline{C,D};\overline{E,F};\overline{G,H}\} \\ \lambda_o &= \{\overline{A,C,E,G};\overline{B,D,F,H}\} \end{aligned}$$

The π-lattice is found to be



(b) $\pi_1 \geq \pi_2 \geq \lambda_i$, so that both π_1 and π_2 are autonomous clocks of periods 2 and 4, respectively. Choose τ_1, such that $\tau_1 \cdot \pi_1 = \pi_2$, and the following schematic diagram results.

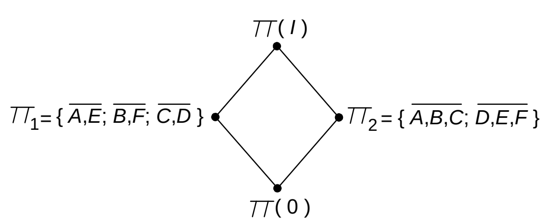


12.11.

(a)

$$\begin{aligned} \lambda_i &= \{\overline{A,B,C};\overline{D,E,F}\} \\ \lambda_o &= \{\overline{A,B,C,D,E};\overline{F}\} \end{aligned}$$

The following closed partitions are found using the procedure in Sec. 12.3.



(b) To find f_1 independent of x, we must assign y_1 to $\lambda_i = \pi_2$. Since f_2 and f_3 are functions of x, y_2, and y_3, the partitions corresponding to y_2 and y_3 are neither input consistent nor closed. However, the partition corresponding to y_2 and y_3 together is closed; in fact it is π_1. The assignment for y_2 and y_3 is induced by partitions τ_2 and τ_3, such that

$$\tau_1 \cdot \tau_2 \cdot \pi_2 = 0$$

where $\tau_1 \cdot \tau_2 = \pi_1$.

One possible choice is

$$\begin{aligned} \tau_1 &= \{\overline{A, C, D, E}; \overline{B, F}\} \\ \tau_2 &= \{\overline{A, E}; \overline{B, C, D, F}\} \end{aligned}$$

In this case, $\pi_2 \cdot \tau_1 \leq \lambda_o$, and z depends only on y_1 and y_2.

(c) $A \to 000$, $B \to 011$, $C \to 001$, $D \to 101$, $E \to 100$, $F \to 111$,

$$\begin{aligned} Y_1 &= y_1' \\ Y_2 &= xy_2'y_3 + x'y_2'y_3' \\ Y_3 &= x'y_2 + xy_2' + y_3' \\ z &= y_1y_2 \end{aligned}$$

12.14.

(a) There is no closed partition π', such that $\pi \cdot \pi' = \pi(0)$.

(b) From the implication graph, it is easy to show that if state E is split into E' and E'', a machine M' is obtained for which

$$\begin{aligned} \pi_2 &= \{\overline{A, E'}; \overline{B, C}; \overline{D, E''}; \overline{F, G}\} \\ \pi_1 &= \{\overline{A, C, D, F}; \overline{B, E', E'', G}\} = \lambda_i \end{aligned}$$

are closed partitions, and

$$\lambda_o = \{\overline{A, G}; \overline{B}; \overline{C, E', E''}; \overline{D, F}\}$$

is an output-consistent partition. Clearly,

$$\pi_1 \cdot \pi_2 = \pi(0)$$

(c)

	NS, z	
PS	$x=0$	$x=1$
A	F,1	C,0
B	E'',0	B,1
C	D,0	C,0
D	F,1	C,1
E'	G,0	B,0
E''	G,0	B,0
F	A,1	F,1
G	E',1	G,0

Machine M'

Let y_1 be assigned to blocks of π_1, y_2 to blocks of $\tau(y_2) = \{\overline{A, D, E', E''}; \overline{B, C, F, G}\}$, and y_3 to $\tau(y_3) = \{\overline{A, B, C, E'}; \overline{D, E'', F, G}\}$. Clearly, $\tau(y_2) \cdot \tau(y_3) = \pi_2$.
The state tables of the component machines are:

PS	NS
$(A, C, D, F) \rightarrow P$	P
$(B, E', E'', G) \rightarrow Q$	Q

M_1

	NS	
PS	$x=0$	$x=1$
$(A, D, E', E'') \rightarrow R$	S	S
$(B, C, F, G) \rightarrow S$	R	S

M_2

	NS			
	$x=0$		$x=1$	
PS	R	S	R	S
$(A, B, C, E') \rightarrow U$	V	V	U	U
$(D, E'', F, G) \rightarrow V$	V	U	U	V

M_3

(d) The state assignment induced by above partitions: $A \rightarrow 000$, $B \rightarrow 110$, $C \rightarrow 010$, $D \rightarrow 001$, $E' \rightarrow 100$, $E'' \rightarrow 101$, $F \rightarrow 011$, $G \rightarrow 111$.

12.15.

(a) We shall prove that $M\{m[M(\tau)]\} = M(\tau)$, leaving the proof that $m\{M[m(\tau)]\} = m(\tau)$ to the reader.

$[M(\tau), \tau]$ and $\{M(\tau), m[M(\tau)\}$ are partition pairs, and therefore (by the definition of an m-partition) $m[M(\tau)] \leq \tau$. By monotonicity, $M\{m[M(\tau)]\} \leq M(\tau)$. On the other hand, $\{M(\tau), m[M(\tau)]\}$ is a partition pair, and hence, $M\{m[M(\tau)]\} \geq M(\tau)$. Thus, the first equality is valid.

(b) $\{M(\tau), m[M(\tau)]\}$ is a partition pair. It is also an Mm pair, since (from part (a)), $M\{m[M(\tau)]\} = M(\tau)$. Similarly, $\{M[m(\tau)], m(\tau)\}$ is an Mm pair, since $m\{M[m(\tau)]\} = m(\tau)$.

12.19.

(a) m-partitions

$$\begin{aligned}
m(\tau_{AB}) &= \{\overline{A,C};\overline{B};\overline{D};\overline{E}\} = \tau'_1\\
m(\tau_{AC}) &= \{\overline{A,D};\overline{B};\overline{C};\overline{E}\} = \tau'_2\\
m(\tau_{AD}) &= \{\overline{A,B};\overline{C};\overline{D};\overline{E}\} = \tau'_3\\
m(\tau_{AE}) &= \tau(I)\\
m(\tau_{BC}) &= \{\overline{A,C,D};\overline{B};\overline{E}\} = \tau'_4\\
m(\tau_{BD}) &= \{\overline{A,B,C};\overline{D};\overline{E}\} = \tau'_5\\
m(\tau_{BE}) &= \{\overline{A,B,D};\overline{C,E}\} = \tau'_6\\
m(\tau_{CD}) &= \{\overline{A,B,D};\overline{C};\overline{E}\} = \tau'_7\\
m(\tau_{CE}) &= \{\overline{A,B,C};\overline{D,E}\} = \tau'_8\\
m(\tau_{DE}) &= \{\overline{A,C,D};\overline{B,E}\} = \tau'_9
\end{aligned}$$

From all the sums of the $m(\tau_{ab})$'s, only one additional m-partition results, namely

$$m(\tau_{AB}) + m(\tau_{CD}) = \{\overline{A,B,C,D};\overline{E}\} = \tau'_{10}$$

(b) We are looking for an assignment with three state variables. Thus, we look for three m-partitions, of two blocks each, such that their intersection is zero. The only such partitions are τ'_6, τ'_8, τ'_9. The corresponding M-partitions are:

$$\begin{aligned}
M(\tau'_6) &= \tau_{AC} + \tau_{AD} + \tau_{BE} + \tau_{CD} = \{\overline{A,C,D};\overline{B,E}\} = \tau_6\\
M(\tau'_8) &= \tau_{AB} + \tau_{AD} + \tau_{BD} + \tau_{CE} = \{\overline{A,B,D};\overline{C,E}\} = \tau_8\\
M(\tau'_9) &= \tau_{AB} + \tau_{AC} + \tau_{BC} + \tau_{DE} = \{\overline{A,B,C};\overline{D,E}\} = \tau_9
\end{aligned}$$

We now have three Mm pairs: (τ_6, τ'_6); (τ_8, τ'_8); (τ_9, τ'_9), where $\tau_6 = \tau'_9$, $\tau_8 = \tau'_6$, $\tau_9 = \tau'_8$. Consequently, we obtain the functional dependencies listed below:

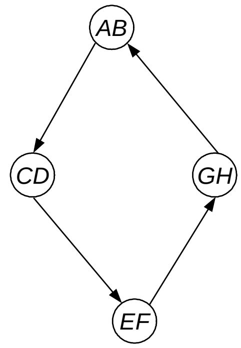
$$\begin{aligned}
y_1 \leftrightarrow \tau'_6 &\qquad Y_1 = f_1(y_2, x_1, x_2)\\
y_2 \leftrightarrow \tau'_8 &\qquad Y_2 = f_2(y_3, x_1, x_2)\\
y_3 \leftrightarrow \tau'_9 &\qquad Y_3 = f_3(y_1, x_1, x_2)
\end{aligned}$$

12.21.

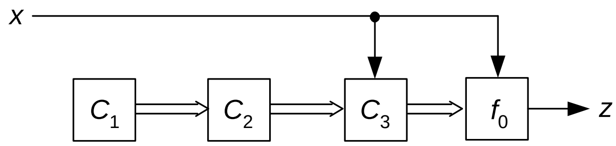
(a) The following closed partitions are found in the usual manner;

$$\begin{aligned}
\pi_1 &= \{\overline{A,B,E,F};\overline{C,D,G,H}\}\\
\pi_2 &= \{\overline{A,B};\overline{C,D};\overline{E,F};\overline{G,H}\}
\end{aligned}$$

where $\pi_1 > \pi_2$. The maximal autonomous clock is generated by $\lambda_i = \{\overline{A,B}; \overline{C,D}; \overline{E,F}; \overline{G,H}\} = \pi_2$. Its period is 4 and its state diagram is given by



(b)



$$\begin{aligned} M_1 &\leftrightarrow \pi_1 \\ M_2 \leftrightarrow \tau_2 &= \{\overline{A,B,C,D}; \overline{E,F,G,H}\}, \text{where } \pi_1 \cdot \tau_2 = \pi_2. \\ M_3 \leftrightarrow \tau_3 &= \{\overline{A,C,E,G}; \overline{B,D,F,H}\}, \text{where } \pi_2 \cdot \tau_3 = \pi(0). \end{aligned}$$

(c)

PS	NS
$(A,B,E,F) \to P$	Q
$(C,D,G,H) \to Q$	P

M_1

	NS	
PS	P	Q
$(A,B,C,D) \to R$	R	S
$(E,F,G,H) \to S$	S	R

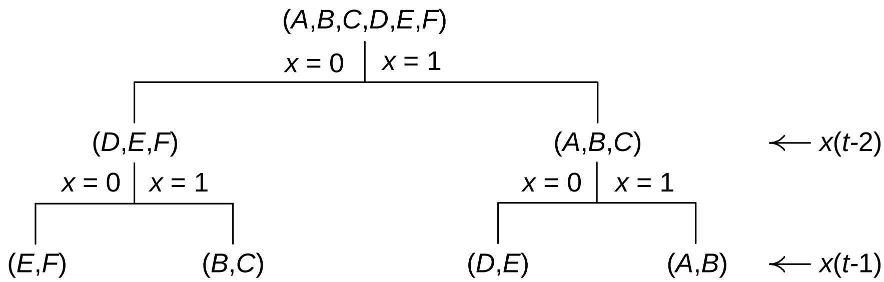
M_2

	NS,z							
	$x=0$				$x=1$			
PS	PR	QR	PS	QS	PR	QR	PS	QS
$(A,C,E,G) \to U$	V,0	U,0	U,0	V,0	U,0	V,0	V,0	U,0
$(B,D,F,H) \to V$	U,0	V,0	V,0	U,0	V,1	U,1	V,1	V,1

M_3

12.23. From the tree below, we conclude that the two delays actually distinguish the blocks of the cover

$\varphi = \{\overline{B, C, E, F}; \overline{A, B, D, E}\}$.



If the last two input values are 00, then machine is in (E, F)
If the last two input values are 10, then machine is in (B, C)
If the last two input values are 01, then machine is in (D, E)
If the last two input values are 11, then machine is in (A, B)

However, since only $x(t-2)$ [and not $x(t-1)$] is available to M_s, it can only distinguish (B, C, E, F) from (A, B, D, E), depending on whether $x(t-2)$ is equal to 0 or 1, respectively. Hence, we must select a partition τ such that $\tau \cdot \{\overline{B, C, E, F}; \overline{A, B, D, E}\} = \pi(0)$. To reduce the complexity of the output, choose $\tau = \lambda_o = \{\overline{A, C}; \overline{B}; \overline{D, F}; \overline{E}\}$. The state table of M_s is given by

	$x(t-2)=0$		$x(t-2)=1$	
PS	$x(t)=0$	$x(t)=1$	$x(t)=0$	$x(t)=1$
$(A,C) \to P$	R,0	Q,0	S,0	P,0
$(B) \to Q$	R,0	Q,1	R,0	Q,1
$(D,F) \to R$	R,1	Q,1	R,1	P,1
$(E) \to S$	S,1	P,0	S,1	P,0

12.26. Generate the composite machine of M and M_a.

	NS	
PS	$x=0$	$x=1$
AG	BH	CG
BH	CG	DH
CG	DH	EG
DH	EG	FH
EG	FH	AG
FH	AG	BH

Composite machine

The composite machine generates the closed partition $\pi_1 = \{\overline{A, C, E}; \overline{B, D, F}\}$.

To generate M_b, a second closed partition π_2 is needed, such that $\pi_1 \cdot \pi_2 = \pi(0)$. Evidently, $\pi_2 = \{\overline{A, D}; \overline{B, E}; \overline{C, F}\}$, and the corresponding state table for M_b, as well as the logic for z, are found to be:

PS	NS, $x=0$	$x=1$
$(A,D) \rightarrow \alpha$	β	γ
$(B,E) \rightarrow \beta$	γ	α
$(C,F) \rightarrow \gamma$	α	β

M_b

	$x=0$		$x=1$	
PS	G	H	G	H
α	0	0	0	1
β	1	0	0	1
γ	1	1	1	1

z logic

12.27.

(a) Composite machine of M_1 and M_2:

PS	NS, $x=0$	$x=1$	Z^1Z^2
PA	QB	RD	0 0
QB	PE	QC	0 0
RD	QB	PA	1 1
PE	QC	RE	0 1
QC	PA	QB	0 0
RE	QC	PE	1 1

(b) The partition $\{\overline{PA,PE};\overline{QB,QC};\overline{RE,RD}\}$ on the composite machine is closed. It clearly corresponds to $\{\overline{P};\overline{Q};\overline{R}\}$ on M_1 and $\{\overline{A,E};\overline{B,C};\overline{D,E}\}$ on M_2.

The state table of M_c is shown below. To find the state table of M_{2s}, choose the partition $\{\overline{PA,QB,RE};\overline{PE,QC,RD}\}$ which corresponds to $\{\overline{A,B,E};\overline{C,D,E}\}$ on M_2.

PS	NS, $x=0$	$x=1$
$(PA,PE) \rightarrow S_1$	S_2	S_3
$(QB,QC) \rightarrow S_2$	S_1	S_2
$(RE,RD) \rightarrow S_3$	S_2	S_1

	NS,z					
	$x=0$			$x=1$		
PS	S_1	S_2	S_3	S_1	S_2	S_3
$(PA,QB,RE) \rightarrow R_1$	R_1,0	R_1,1	R_2,0	R_2,1	R_1,0	R_1,1
$(PE,QC,RD) \rightarrow R_2$	R_2,0	R_1,0	R_1,0	R_1,1	R_1,0	R_1,0

Solutions for Problems 12.28, 12.29, and 12.30 are available in Reference 12.13 in the text.

Chapter 13

13.1.

(a) Shortest homing sequences:

M_1: 000,011,110,111

M_2: 01

M_3: 00,01

(b) Shortest synchronizing sequences:

M_1: 000 (to state A).

M_2: 000 (to B), 111 (to A).

M_3: 01010 (to C).

13.2. If one precludes taking adaptive decisions in which a new input symbol is applied after looking at the previous output symbols, then the procedure is simply that of deriving a synchronizing sequence. A minimal synchronizing sequence is 00101000. The corresponding path through the synchronizing tree is shown below.

$$(ABCDEF) \xrightarrow{0} (ABCEF) \xrightarrow{0} (ABCE) \xrightarrow{1} (DEF) \xrightarrow{0} (BF) \xrightarrow{1} (CD) \xrightarrow{0} (EF) \xrightarrow{0} (B) \xrightarrow{0} (A)$$

If an adaptive experiment is allowed, then the minimal homing sequence 001 can be first derived. Then depending on the response to this homing sequence, as shown in the table below, a transfer sequence can be used to reach state A. This requires only five input symbols in the worst case.

Response to homing sequence 001

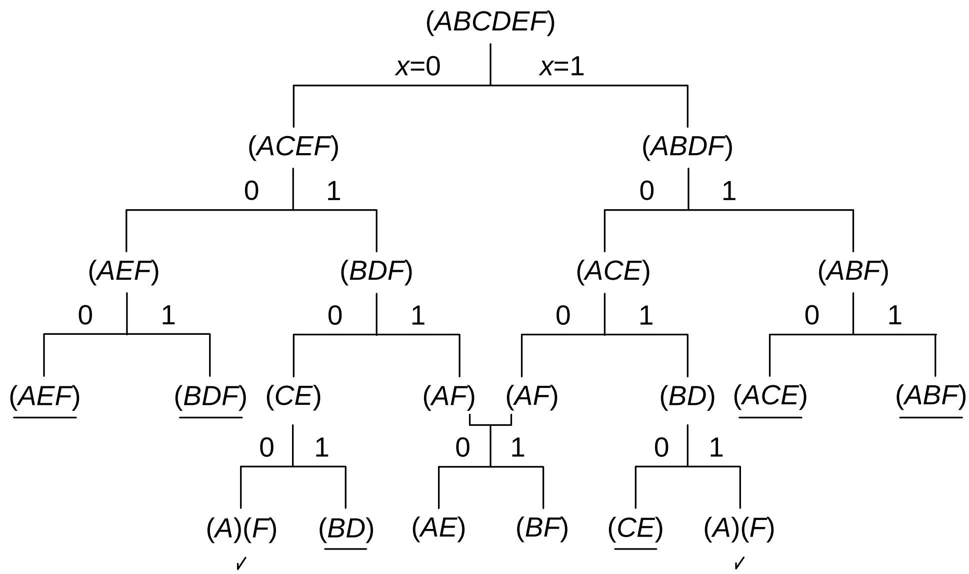
Initial state	*Response to* 001	*Final state*	*Transfer sequence*
A	100	F	$T(F,A) = 00$
B	011	D	$T(D,A) = 1$
C	011	D	$T(D,A) = 1$
D	111	D	$T(D,A) = 1$
E	101	E	$T(E,A) = 00$
F	101	E	$T(E,A) = 00$

13.3.

(i) Form the direct sum of the two machines. (ii) Construct a preset homing experiment that will determine the *machine* at the end of the experiment.

	NS,z	
PS	$x = 0$	$x = 1$
A	$A,0$	$B,0$
B	$C,0$	$A,0$
C	$A,1$	$B,0$
D	$E,0$	$F,1$
E	$F,0$	$D,0$
F	$E,0$	$F,0$

Direct sum



There are two experiments of length four that will identify the machine, namely 1011 and 0100.

13.6. For every n-state machine, there exists a homing sequence of length $l = \frac{n(n-1)}{2}$. Thus, any sequence that contains all possible sequences of length l will be a homing sequence for all n-state machines. A trivial bound on the length of this sequence is $l2^l$. (Although one can show that the least upper bound is $2^l + l - 1$.) In the case of $n = 3$, $l = \frac{3(3-1)}{2} = 3$, and the least upper bound is $2^3 + 3 - 1 = 10$. The following is an example of such a sequence: 0001011100.

13.7. Recall the minimization procedure for completely specified machines. At each step, the partition P_i has at least one more block than P_{i-1}.

$$
\begin{aligned}
P_0 &= (n \text{ states}) \\
P_1 &= (n-1 \text{ states})(\ ) \\
&\cdots \\
&\cdots \\
P_k &= (n-k \text{ states})(\ )(\ )\cdots()
\end{aligned}
$$

After k symbols, at most $n-k$ states are indistinguishable. And after the $(k+1)$st symbol, some pair of states must be distinguishable. (See Ref. 13.11).

13.9.

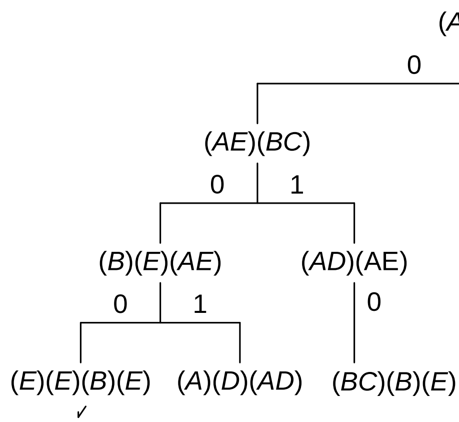
M_1: Shortest distinguishing sequence is 0100.

M_2: Shortest distinguishing sequence is 010100.

M_3: All sequences of length 3.

13.10.

(a)



The required sequence is 000.

(b) No.

13.11. The * entry in column 0 must be specified as D, while that in column 1 must be specified as C.

13.13.

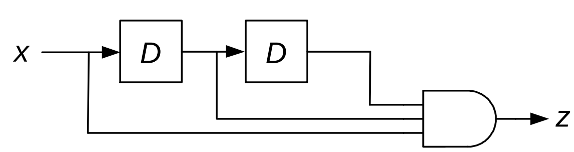
(a) All branches in the distinguishing tree will terminate in Level 1 because of Rule 2 (Sec. 13.3).

(b) No branch will terminate because of Rule 2 and since the machine is reduced, the tree will terminate according to Rule 3 and the distinguishing sequence follows. This in fact implies that every homing sequence in this case is also a distinguishing sequence, and since every machine has a homing sequence whose length is at most $\frac{n(n-1)}{2}$, it also has such a distinguishing sequence.

13.14.

(a) Any sequence of length k, where k is the number of delays, is a synchronizing sequence, since it determines the state of all the delays, and thus of the machine.

(b) No.



(c) No. (See Sec. 14.2). Any machine with no synchronizing sequence is a counter example, e.g.,

	NS,z	
PS	$x=0$	$x=1$
A	$A,0$	$B,1$
B	$B,1$	$A,0$

Another counter example, with a synchronizing sequence is

	NS	
PS	$x = 0$	$x = 1$
A	A	B
B	B	B

13.16. 01 is a homing sequence for the machine in question. A response 10 indicates a final state B. The first 1 symbol of the next input sequence takes the machine to state A; the correct output symbol is 0. It is easy to verify that the rest of the sequence takes the machine through every transition and verifies its correctness. The correct output and state sequences are:

X :		0		1		0		1		0		1		0		1	
	A		A		B		C		C		B		A		A		B
Z :		1		0		0		1		0		0		1		0	

X :		0		0		1		0		0		1		1		0	
	B		C		B		A		A		A		B		A		A
Z :		0		0		0		1		1		0		0		1	

X :		1		0		0		0		1	
	A		B		C		B		C		C
Z :		0		0		0		0		1	

13.18. The state table for which the given experiment is a checking experiment is

	NS,z	
PS	$x = 0$	$x = 1$
A	$B, 0$	$B, 1$
B	$C, 1$	$D, 1$
C	$A, 2$	$A, 1$
D	$E, 3$	$A, 2$
E	$D, 3$	$B, 2$

If the shortest distinguishing prefix of the distinguishing sequence 010 is used to identify the states, the given checking experiment can be obtained. The shortest distinguishing prefixes and the corresponding responses are summarized below:

Initial state	*Shortest distinguishing prefix*	*Response to shortest distinguishing prefix*	*Final state*
A	0	0	B
B	0	1	C
C	0	2	A
D	010	321	C
E	010	320	B

13.19. The sequence 000 is a synchronizing sequence (to state D). Both 000 and 111 are distinguishing sequences. To perform the experiment, apply 000 first and regardless of the machine's response to this sequence, construct, in the usual manner, a checking experiment, using the shortest distinguishing prefix of either distinguishing sequence. An experiment that requires 25 symbols is given below:

$X:$		0		0		0		0		0		1		0		0	
	$-$		$-$		$-$		D		D		D		A		D		D
$Z:$		$-$		$-$		$-$		1		1		1		0		1	

$X:$		1		1		0		0		0		1		1		1	
	D		A		C		A		D		D		A		C		B
$Z:$		1		0		0		0		1		1		0		0	

$X:$		0		0		0		0		1		1		1		1		0	
	B		C		A		D		D		A		C		B		D		D
$Z:$		0		0		0		1		1		0		0		0		1	

13.20. The given sequences do not constitute a checking experiment, since the response in question can be produced by the two distinct machines shown below.

	NS, z	
PS	$x = 0$	$x = 1$
A	$A, 2$	$B, 2$
B	$C, 0$	$A, 1$
C	$D, 1$	$E, 0$
D	$E, 2$	$A, 0$
E	$B, 1$	$C, 2$

Machine M^*

	NS, z	
PS	$x = 0$	$x = 1$
1	$1, 2$	$3, 2$
2	$5, 1$	$2, 2$
3	$4, 0$	$1, 0$
4	$3, 1$	$2, 0$
5	$2, 2$	$1, 1$

Machine M

Input :		0		0		1		0		0		1		0		1	
State in M	1		1		1		3		4		3		1		1		3
State in M^*	A		A		A		B		C		D		A		A		B
Output :		2		2		2		0		1		0		2		2	

Input :		0		1		1		0		0		0		1		0		0	
State in M	3		4		2		2		5		2		5		1		1		1
State in M^*	B		C		E		C		D		E		B		A		A		A
$Z:$		0		0		2		1		2		1		1		2		2	

(Actually, there exist eight distinct, five-state machines which produce the above output sequence in response to the given input sequence.)

13.21.

(a) The distinguishing sequences are: 00, 01.

Response to 00	*Initial state*
00	A
01	B
11	C
10	D

Response to 01	*Initial state*
00	A
01	B
11	C or D
10	D or C

(b)

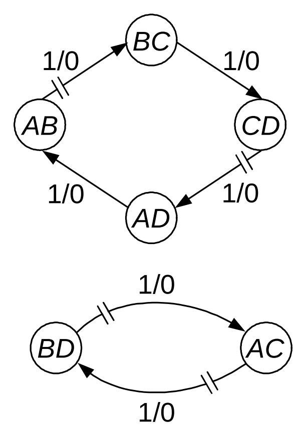
PS	NS,z $x=0$	$x=1$
A	$B,0$	$B,0$
B	$C,0$	$C,0$
C	$D,1$	$D,1$
D	$A,1$	$D,1$

OR

PS	NS,z $x=0$	$x=1$
A	$B,0$	$C,1$
B	$C,0$	$D,0$
C	$D,1$	$C,1$
D	$A,1$	$B,0$

13.23

(a)



(b)

PS	NS,zz_1 $x=0$	$x=1$
A	$A,0\phi$	$B,01$
B	$A,0\phi$	$C,00$
C	$A,0\phi$	$D,00$
D	$A,1\phi$	$A,01$

(c) Assuming an initial state A and all don't-care combinations equal to 0, we obtain the following checking experiment.

$$\begin{array}{lcccccccccccc} X: & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ Z: & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 2 \end{array}$$

$$
\begin{array}{lcccccccccccc}
X: & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\
Z: & 0 & 0 & 2 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 2 & 0
\end{array}
$$

(Note that an output symbol 2 is used to denote 10, 1 to denote 01, and 0 to denote 00.)

13.24. Let $S(t)$ denote the state at time t, that is, just before the application of the t^{th} input symbol (e.g., the initial state is $S(1)$). There seems to be no input sequence that yields three different responses at three different points in the experiment. Thus, more sophisticated reasoning must be used.

To begin with, the machine has at least one state that responds to input symbol 0 by producing an output symbol 1. Consequently, the machine has at most two states that respond to an input symbol 0 by producing an output symbol 0. Now consider the portion of the experiment between $t = 4$ and $t = 11$. Each of the states $S(4)$, $S(6)$, and $S(8)$ respond to a 0 by producing a 0. However, at most two of these states can be distinct. Therefore, either $S(4) = S(6)$ or $S(4) = S(8)$ or $S(6) = S(8)$. If $S(4) = S(6)$, then $S(6) = S(8)$; thus $S(8)$ is identical to either $S(4)$ or $S(6)$. This in turn implies that $S(10)$ is identical to either $S(6)$ or $S(8)$. Therefore S(10) is a state that responds to an input symbol 0 by producing an output symbol 0, and to an input symbol 1 by producing an output symbol 1.

Using similar reasoning, when examining the portion of the experiment between $t = 14$ and $t = 18$, we conclude that $S(17)$ must be identical to either $S(15)$ or $S(16)$. Thus, $S(17)$ is a state that responds to a 0 by producing a 0, and to a 1 by producing a 0. Therefore, the circuit does have two distinct states that respond to a 0 by producing a 0: $S(10)$ and $S(17)$.

A careful analysis of the experiment yields the state table below:

	NS,z	
PS	$x = 0$	$x = 1$
A	$B, 0$	$C, 1$
B	$B, 0$	$A, 0$
C	$C, 1$	$A, 0$

13.25.

(a) For each fault-free state transition, there are three possible SST faults. The collapsed set of SSTs for each such transition is given in the following table. The collapsed set is not unique, but is of minimum cardinality. For fault collapsing, we use the fact that:

$$
\begin{array}{c}
\text{SPDS}(A, B) = \text{SPDS}(A, C) = \text{SPDS}(B, D) = \text{SPDS}(C, D) = 0 \\
\text{SPDS}(A, B) = \text{SPDS}(A, D) = \text{SPDS}(B, C) = \text{SPDS}(C, D) = 1
\end{array}
$$

Fault-free transition	Set of collapsed faults
$<0, A, C, 0>$	$<0, A, A, 0>$, $<0, A, B, 0>$
$<1, A, C, 0>$	$<1, A, A, 0>$, $<1, A, B, 0>$
$<0, B, C, 1>$	$<0, B, A, 1>$, $<0, B, B, 1>$
$<1, B, B, 1>$	$<1, B, A, 1>$, $<1, B, D, 1>$
$<0, C, D, 1>$	$<0, C, A, 1>$, $<0, C, B, 1>$
$<1, C, A, 0>$	$<1, C, B, 0>$, $<1, C, D, 0>$
$<0, D, A, 0>$	$<0, D, B, 0>$, $<0, D, D, 0>$
$<1, D, B, 1>$	$<1, D, A, 1>$, $<1, D, C, 1>$

Whenever both 0 and 1 are available as SPDS of a pair of states, we assume in the above table that 0 is used as SPDS.

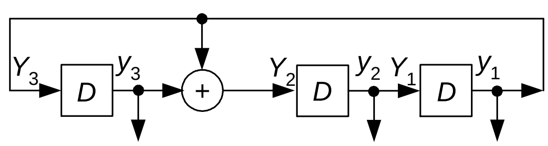
(b) To derive a test sequence for this SST fault, we first need $T(A, B) = 001$ or 101. Then the activation vector $x = 1$ is applied. Finally, SPDS$(B, C) = 1$ is applied. Hence, one possible test sequence is 00111.

13.27.

(a) A minimal test set: $(x_1, x_2, y_1, y_2) = \{$(0,0,1,1), (1,1,0,0), (1,1,1,1)$\}$.

(b) Minimum no. of clock cycles: $3(2 + 1) + 2 - 1 = 10$.

13.30. The three-stage LFSR for the polynomial $x^3 + x + 1$ is shown below. Its state sequence that repeats on $y_1y_2y_3$ (not on $y_3y_2y_1$ as in Fig. 13.16) is: $111 \rightarrow 101 \rightarrow 001 \rightarrow 010 \rightarrow 100 \rightarrow 011 \rightarrow 110$. This is exactly the reverse of the cycle shown in Fig. 13.16 for the polynomial $x^3 + x^2 + 1$.



13.31. Test set: $(x_3, x_2, x_1) = \{$(0,1,1), (1,0,0), (0,1,0), (0,0,1)$\}$.

Chapter 14

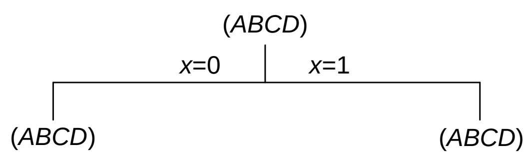
14.1.

(a) Not finite memory

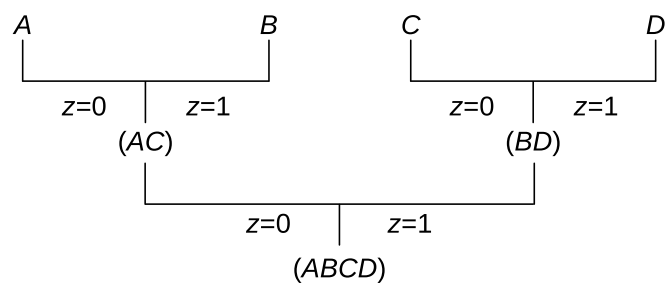
(b) Finite memory of order 3.

(c) Finite memory of maximal order, that is, $\mu = \frac{5(5-1)}{2} = 10$.

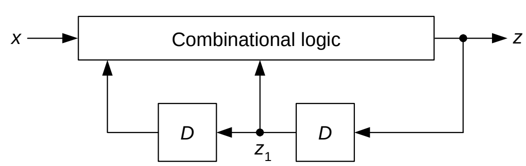
14.2. The machine in question can be shown to be finite memory of order 2. Let us determine the information supplied by the "input" and "output" delay registers. The synchronizing tree is shown below:



It clearly supplies no information, and thus the input delays are redundant in this case. The output synchronizing tree is shown below.



From the above tree, it is evident that the machine in question can be realized in the form:



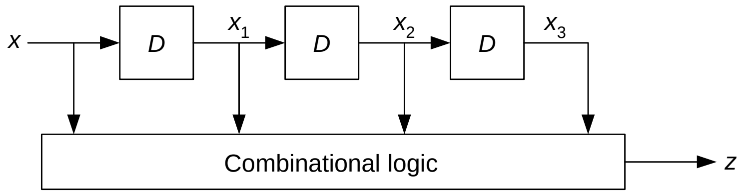
$x \backslash z_2z_1$	00	01	11	10
0	0	0	1	1
1	1	1	0	0

$z = x \oplus z_2$

The map of the combinational logic reveals that it is independent of the value of z_1.

14.4. The canonical realization of a finite-memory machine, shown in Fig. P14.2, has $(p)^\mu \cdot (q)^\mu = (pq)^\mu$ states. (Note that the input delay register may have $(p)^\mu$ output combinations, etc.). If such a realization corresponds to a finite-memory n-state machine, then clearly $(pq)^\mu \geq n$.

14.5. The machine in question is definite of order $\mu = 3$. The canonical realization and combinational logic are shown below.



$x_3x_2x_1$	x = 0	x = 1	State
000	0	0	C
001	0	1	B
011	0	1	B
010	1	0	A
110	1	0	A
111	0	1	B
101	0	0	E
100	1	1	D

Truth table for z

14.6.

(a)

PS	NS, $x = 0$	NS, $x = 1$
A	A	B
B	E	B
C	E	B or D or F
D	E	F
E	A	D
F	E	B

(For a systematic procedure of specifying the entries of an incompletely specified machine, so that the resulting machine is definite, the reader is referred to C. L. Liu, "A Property of Partially Specified Automata," Information and Control, vol. 6, pp. 169-176, 1963.)

(b) Impossible.

14.7.

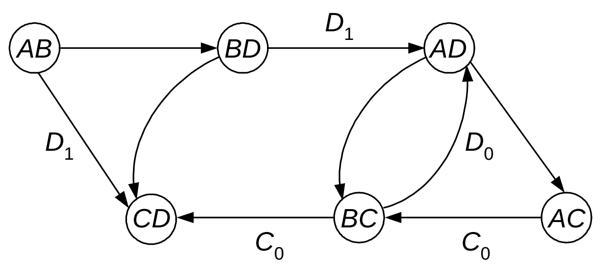
(a) $\mu = 2$

(b) Not finite output memory.

(c) $\mu = 4$

14.8. Write the testing table with two entries for each unspecified entry, namely, C_0 and C_1 denote the two possible output assignments for the 0-successor of state C. Similarly, use D_0 and D_1 to denote the two possible output assignments to the 1-successor of state B.

PS	0/0	0/1	1/0	1/1
A	B	–	–	C
B	D	–	D_0	D_1
C	C_0	C_1	A	–
D	C	–	–	A
AB	BD	–	–	CD_1
AC	BC_0	–	–	–
AD	BC	–	–	AC
BC	C_0D	–	AD_0	–
BD	CD	–	–	AD_1
CD	CC_0	–	–	–



From the testing graph, it is evident that D_0 results in a loop and must be eliminated. Consequently, D_1 must be selected. It is also evident that C_0 results in the longest path, i.e., $l = 5$ and $\mu = 6$.

14.10.

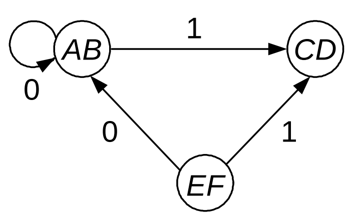
(a) Test the output successor table, whose entries must all be single-valued, for definiteness.

(b) See machine in Fig. P14.2. Its output successor table is definite of order $\mu = 2$.

14.11.

(a) Lossy, since D implies (AC), which implies (BC), which in turn implies (AA).

(b) The machine has three compatible pairs, as shown by the testing graph below. Since the graph is not loop-free, the machine is lossless of infinite order.



(c) Machine is lossless of order $\mu = 3$.

(d) Machine is lossy, since (DE) implies (EE).

14.14. In rows C and D, column $x = 0$, output symbol should be 11. In row B, column $x = 1$, output symbol should be 10. Such a (unique) specification yields a lossless machine of order $\mu = 4$.

14.16.

(a) If we disregard z_2 and consider the machine with output z_1 only, we can show that it is information lossless. Consequently, the input sequence X can always be retrieved from the output sequence Z_1 and the initial and final states. In this case, the input sequence can be determined from the output successor and output predecessor tables, i.e., $X = 011011$.

The determination of z_2 for the given X is straightforward, i.e., $Z_2 = 010010$.

(b) Yes, by using the foregoing procedure.

14.18.

(a) From the output sequence Z and the final state of M_2, we shall determine (by performing backward tracing) the intermediate sequence Y. From Y, by means of the output predecessor table of M_1, we shall determine the initial state of M_1. The state sequence of M_2 and the Y sequence are obtained from the output predecessor table of M_2.

$z=0$	$z=1$	NS
$B, 0$	$C, 1$	A
$A, 0$	$D, 0$	B
$B, 1$	$A, 1$	C
$D, 1$	$C, 0$	D

Y :		1		1		0		1		0		0	
	A		C		A		B		C		D		B
Z :		1		1		0		0		1		1	

The state sequence of M_1 is obtained from the output predecessor table of M_1 as follows.

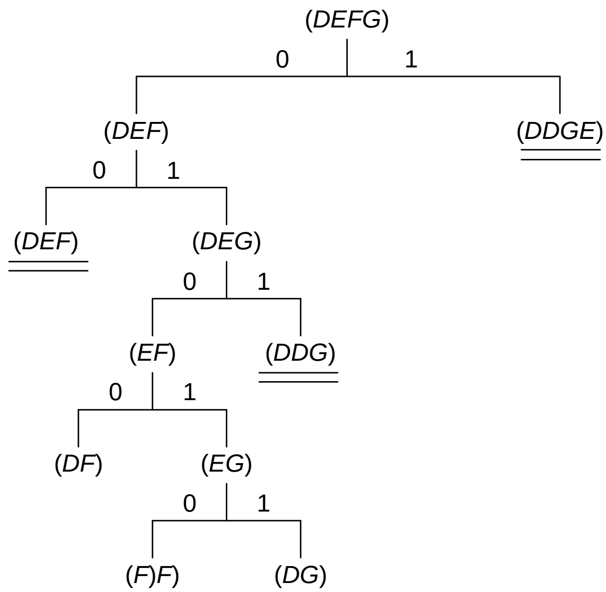
$y=0$	$y=1$	NS
–	D	A
A	B	B
B	A	C
(CD)	–	D

									A		A		A
					A		A						
			A						B		B		B
	D				C		B						
			D						C		C		C
					D		D						
									D		D		D
$y =$		1		1		0		1		0		0	

The initial state of M_1 is thus D. The input sequence X is either 110100 or 111100.

(b) From the output predecessor table of M_2, it is obvious that, given the output sequence Z and final state of M_2, we can always determine the input sequence Y. And since M_1 can be shown to be finite output memory of length $\mu = L = 6$, the statement is proved.

14.19. To determine the initial state of M_2, we must determine its distinguishing sequences. These sequences should then be produced as output sequences by M_1. The distinguishing tree for M_2 is:



The shortest distinguishing sequence for M_2 is: 01010. Now, perform backward tracing on the output successor of M_1 to determine which input sequence X will yield $Y = 01010$.

	NS, x	
PS	$y = 0$	$y = 1$
A	$(B, 0)$	$(C, 1)$
B	$(A, 1)$	$(C, 0)$
C	$(A, 0)(B, 1)$	$-$

From the output successor table, we find that X can be any one of the following sequences: 00010, 00011, 00100, 00101.

14.20. (a) From the output predecessor table shown below, we find the following state and input sequences:

PS, x		
$z = 0$	$z = 1$	NS
$B, 0$	$C, 1$	A
$A, 0$	$D, 0$	B
$B, 1$	$A, 1$	C
$D, 1$	$C, 0$	D

Input :		1	0	1	0	0
State :	C	A	B	C	D	B
Output :		1	0	0	1	1

(b) The proof is evident when we observe that all the entries of the output predecessor table are single-valued and the two transitions in each row are associated with different values of x.
(c) Use output predecessor tables and generalize the arguments in (b). It may be helpful to define a "lossless" output predecessor table of the μ^{th} order.

14.21.
(a) The output successor and predecessor tables are shown as follows:

PS	$z=0$	$z=1$
A	C	B
B	$-$	BD
C	B	E
D	AE	$-$
E	F	D
F	D	A

Output successor table

$z=0$	$z=1$	NS
D	F	A
C	AB	B
A	$-$	C
F	BE	D
D	C	E
E	$-$	F

Output predecessor table

Following the procedure for retrieving the input sequence, illustrated in Fig. 14.11, we find

Successors to A	A	B	BD	BD	AE	CF	BD	AE	CF	AE	CF
Z		1	1	1	0	0	0	0	0	1	0
Predecessors to F	AB	AB	BE	D	E	F	D	A	C	E	F
State sequence	A	B	B	D	E	F	D	A	C	E	F
Input sequence		0	1	0	0	0	0	0	1	0	0

(b) No, since the 11011000-successor of A may be either B or D, but *not* A.

14.23. The machine is lossless of order $\mu = 2$. Hence, the inverse state can be defined by the initial state and one successor output symbol. The possible states are $(A, 1)$; $(B, 0)$; $(C, 1)$; $(D, 0)$; $(E, 1)$; $(F, 0)$. The state table for the inverse machine is as follows.

	NS, x	
PS	$z=0$	$z=1$
$(A,1)$	$(B,0)$,0	$(C,1)$,1
$(B,0)$	$(D,0)$,0	$(E,1)$,1
$(C,1)$	$(F,0)$,1	$(A,1)$,0
$(D,0)$	$(B,0)$,1	$(C,1)$,0
$(E,1)$	$(F,0)$,0	$(A,1)$,1
$(F,0)$	$(D,0)$,1	$(E,1)$,0

M is in initial state A, therefore, M^{-1} must be in $(A,1)$ one unit time later. The states from which $(A,1)$ can be reached in one unit time with $z=1$ are $(C,1)$ and $(E,1)$. Similarly, we obtain:

Initial state in M	Possible initial states in M^{-1}
A	$(C,1)$; $(E,1)$
B	$(A,1)$; $(D,0)$
C	$(A,1)$; $(D,0)$
D	$(B,0)$; $(F,0)$
E	$(B,0)$; $(F,0)$
F	$(C,1)$; $(E,1)$

14.26.

$A \rightarrow S1S$

$B \rightarrow S1B_11B_20S$

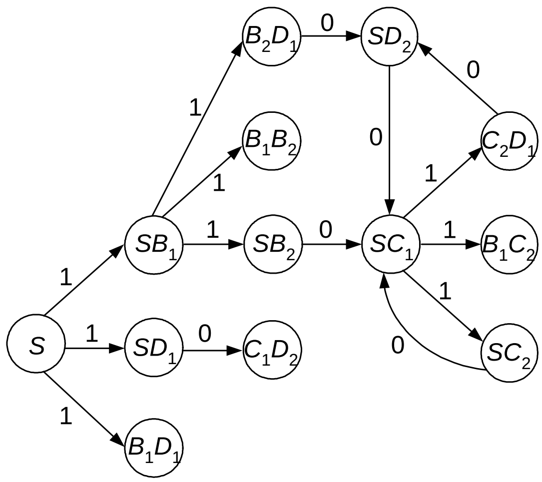
$C \rightarrow S0C_11C_20S$

$D \rightarrow S1D_10D_20S$

The testing table for $\gamma = \{1, 110, 010, 100\}$ is shown below:

	0	1
S	$-$	(SB_1), (SD_1), (B_1D_1)
(SB_1)	$-$	(B_2S), (B_2B_1), (B_2D_1)
(SD_1)	(C_1D_2)	$-$
(B_1D_1)	$-$	$-$
(B_2S)	(SC_1)	$-$
(B_2B_1)	$-$	$-$
(B_2D_1)	(SD_2)	$-$
(C_1D_2)	$-$	$-$
(SC_1)	$-$	(C_2S), (C_2B_1), (C_2D_1)
(SD_2)	(C_1S)	$-$
(SC_2)	(SC_1)	$-$
(B_1C_2)	$-$	$-$
(D_1C_2)	(SD_2)	$-$

Since the pair (SS) was not generated in the table, the code is uniquely decipherable. Let us now construct the testing graph to determine the delay μ



The graph is clearly not loop-free; one of the loops being $(SC_1) \to (SC_2) \to (SC_1)$. The infinite message $1101010\cdots$ cannot be deciphered in a finite time, since it can be interpreted as $AACACAC\cdots$ or $BACAC\cdots$.

14.27. Following the deciphering procedure in Fig. 14.16 we find

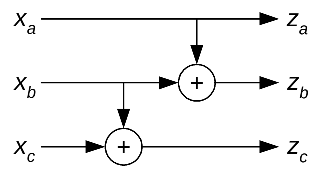
0 ¦ 0 ¦ 1 , 0 , 1 ¦ 0,0 , 1 ¦ 1 , 0 , 1 ¦ 0 ¦ 0 , 1 ,1 ¦ 0 ¦ 0 , 0 , 1

The message is 0; 0; 101; 001; 101; 0; 011; 0; 001.

Chapter 15

15.1.

(*a*)



(*b*)

z_a :	0	1	0	1	1	1	1	0	0	0	1	0	1	1
z_b :	1	0	0	0	1	1	1	0	1	0	0	1	1	0
z_c :	1	1	1	0	0	1	1	0	0	0	1	1	0	0

(*c*)

$$\begin{aligned} x_a &= z_a \\ x_b &= z_a + z_b \\ x_c &= z_a + z_b + z_c \end{aligned}$$

15.2.

(*a*)

$T = \frac{z}{x} = 1 + D + D^3 + D^4 + D^5$ (modulo 2)

(*b*)

$(1 + D + D^3 + D^4 + D^5)X_0 = 0$

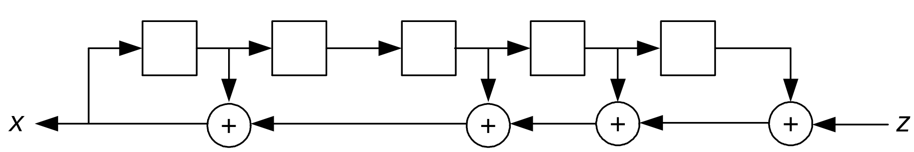
$X_0 = (D + D^3 + D^4 + D^5)X_0$

X_0 : (00001)1100110111110100010010101100001

The sequence consists of 31 symbols, and thus it is maximal. (Note that $2^5 - 1 = 31$.)

(*c*)

$T^{-1} = \frac{1}{1+D+D^3+D^4+D^5}$



15.5. The transfer function of the left circuit is

$$T_1 = D^3 + 1$$

while that of the right circuit is

$$T_2 = (D^2 + 2D + 1)(D + 1)$$

However,

$$T_2 = (D^2 + 2D + 1)(D + 1) = D^3 + D^2 + 2D^2 + 2D + D + 1 = D^3 + 1 = T_1$$

15.6. Proof is obvious, using the identity

$$(1 + 2D)^{-1} = 1 + 14D + 4D^2 + 8D^2 \quad \text{(modulo 16)}$$

15.7.

$(2D^3 + D^2 + 2)X_0 = 0$

$2D^3X_0 + D^2X_0 = X_0$

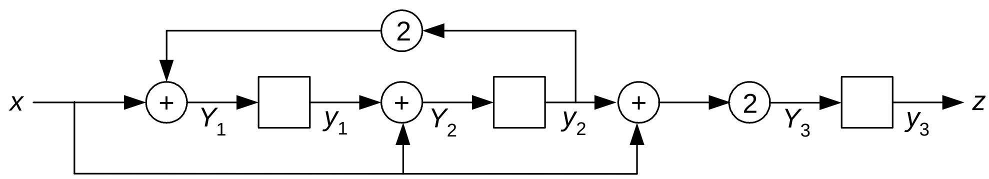
Thus, the present digit of the null sequence is equal to the modulo-3 sum of the second previous digit and the third previous digit. The sequence

(001) 0 1 2 1 1 2 0 1 1 1 0 0 2 0 2 1 2 2 1 0 2 2 2 0 0 1

is the desired null sequence. It is a maximal sequence, since it contains all $3^3 - 1 = 26$ possible nonzero subsequences of three ternary digits.

15.8. Proof can be found in Reference 15.10 or Reference 15.14, pp. 72-73.

15.9.



$$\begin{aligned} z &= DY_3 = D(2x + 2y_2) = 2Dx + 2Dy_2 & (1) \\ y_2 &= DY_2 = D(y_1 + x) = D(DY_1 + x) = D^2Y_1 + DX & (2) \\ Y_1 &= x + 2y_2 = x + 2D^2Y_1 + 2Dx & (3) \end{aligned}$$

Eq. (3) can be rewritten as

$$Y_1 + D^2Y_1 = x + 2Dx$$

or

$$Y_1 = \frac{x(1 + 2D)}{1 + D^2} \tag{4}$$

Substituting Eq. (4) into Eq. (2), we obtain

$$y_2 = \frac{x(D^2 + 2D^3)}{1 + D^2} + Dx \tag{5}$$

Substituting Eq. (5) into Eq. (1), we have

$$z = 2Dx + 2D^2x + \frac{2Dx(D^2 + 2D^3)}{1 + D^2}$$

Thus,

$$T = \frac{z}{x} = \frac{2D + 2D^2 + D^3}{1 + D^2}$$

15.10.

(a)

$$\begin{aligned}
T &= T_1 + T_1T_2T_1 + T_1T_2T_1T_2T_1 + T_1T_2T_1T_2T_1T_2T_1 + \cdots \\
&= T_1[1 + T_1T_2 + (T_1T_2)^2 + (T_1T_2)^3 + \cdots] \\
&= \frac{T_1}{1 - T_1T_2}
\end{aligned}$$

(b)

$$\begin{aligned}
T &= 2D + \frac{2D^2}{1 - 2D^2} + \frac{2D^3}{1 - 2D^2} \\
&= \frac{2D + 2D^3 + 2D^2 + 2D^3}{1 + D^2} \\
&= \frac{2D + 2D^2 + D^3}{1 + D^2}
\end{aligned}$$

15.11. The transfer function is

$$T = \frac{1 + D}{1 + D^2 + D^3}$$

15.12.

(b) Impulse responses:

for T_1: $(1101001)\cdots$

for T_2: $0(01220211)\cdots$

The sequences in parenthesis repeat periodically.

(c) T_1 has a delayless inverse

$$T_1^{-1} = \frac{1 + D + D^3}{1 + D^2}$$

T_2 has no constant term in its numerator, consequently it does not have an instantaneous inverse. It does, however, have a *delayed* inverse.

15.13.

(a) It can be shown, by the Euclidean algorithm, that the greatest common divisor to the numerator and denominator is $D^2 + D + 1$. Thus,

$D^7 + D^4 + D^2 + D + 1 = (D^2 + D + 1)(D^5 + D^4 + 1)$

$D^{10} + D^9 + D^8 + D^7 + D = (D^2 + D + 1)(D^8 + D^5 + D^4 + D^2 + D)$

So that

$$T = \frac{D^8 + D^5 + D^4 + D^2 + D}{D^5 + D^4 + 1}$$

(*b*) Straightforward.

15.15. The impulse response

$h = 1111100(1011100)$

can be decomposed as:

$$\begin{aligned}
h_p &= 1011100 \quad 1011100 \quad 1011100 \\
h_t &= 0100000 \quad 0000000 \quad 0000000 \\
T_p &= (1 + D^2 + D^3 + D^4)(1 + D^7 + D^{14} + D^{21} + \cdots) \\
&= \frac{1 + D^2 + D^3 + D^4}{1 + D^7} \\
T_t &= D \\
T &= T_p + T_t = \frac{1 + D^2 + D^3 + D^4}{1 + D^7} + D
\end{aligned}$$

However, since

$$\frac{1 + D^2 + D^3 + D^4}{1 + D^7} = \frac{1 + D^2 + D^3 + D^4}{(1 + D^2 + D^3 + D^4)(1 + D^2 + D^3)} = \frac{1}{1 + D^2 + D^3}$$

Then,

$$T = \frac{1}{1 + D^2 + D^3} + D = \frac{1 + D + D^3 + D^4}{1 + D^2 + D^3}$$

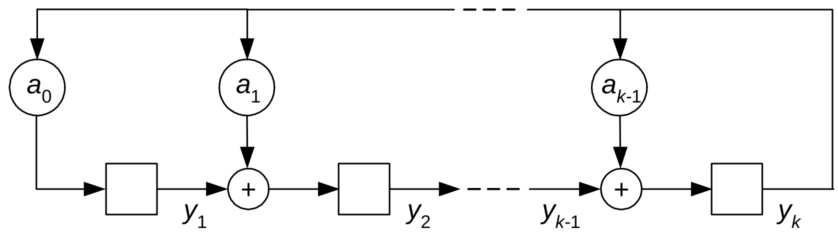
15.17.

(*a*)

$$\begin{bmatrix}
0 & 1 & & & & & \\
 & & 1 & & & & \\
 & & & \cdot & & & \\
 & & & & \cdot & & \\
 & & & & & \cdot & \\
 & & & & & & 1 \\
a_0 & a_1 & a_2 & \cdot & \cdot & \cdot & a_{k-1}
\end{bmatrix}$$

(*b*)

$$\begin{bmatrix}
0 & & & & & a_0 \\
1 & & & & & a_1 \\
 & 1 & & & & a_2 \\
 & & \cdot & & & \cdot \\
 & & & \cdot & & \cdot \\
 & & & & \cdot & \cdot \\
 & & & & 1 & a_{k-1}
\end{bmatrix}$$



15.18. A proof is available in Reference 15.4. (Note that the definition of definiteness is slightly different in the above-mentioned paper, from the one given in the present text.)

15.19.

(a) Straightforward.

(b)

$$\mathbf{K} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \\ \mathbf{CA}^3 \\ \mathbf{CA}^4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

(Note that a letter denotes a matrix.)

$$\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{A}^* = \mathbf{TAR} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \qquad \mathbf{B}^* = \mathbf{TB} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

$$\mathbf{C}^* = \mathbf{CR} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{D}^* = \mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

The machine's independence from x_2 is evident from the fact that in both $\mathbf{B}^*$ and $\mathbf{D}^*$, the second column consists of 0's.

15.20.

$$\mathbf{K}_4 = \begin{bmatrix} z(0) \\ z(1) \\ z(2) \\ z(3) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

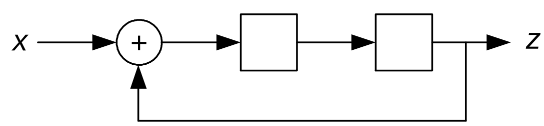
Clearly, the rank of $\mathbf{K}_4$ is two, and thus

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}$$

Therefore,

$$\mathbf{A}^* = \mathbf{TAR} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \mathbf{B}^* = \mathbf{TB} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathbf{C}^* = \mathbf{CR} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \mathbf{D}^* = \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

The schematic diagram of the reduced linear machine is shown below:



15.21.

(a) Straightforward.

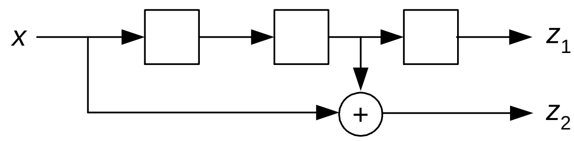
(b)

$$\mathbf{K}_3 = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \mathbf{CA}^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

The rank of $\mathbf{K}_3$ is 3; hence no reduction in the machine's dimension is possible.

$$\mathbf{T} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad \mathbf{R} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$\mathbf{A}^* = \mathbf{TAR} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \qquad \mathbf{B}^* = \mathbf{TB} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad \mathbf{C}^* = \mathbf{CR} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{D}^* = \mathbf{D} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$



15.22. A proof is available in Gill [15.9], Sec. 2.11.

15.23. Proofs are available in Cohn and Even [15.5], and Gill [15.9] in Sec. 2.15.

15.24. A proof can be found in Gill [15.9], Sec. 2.20, and in Cohn [15.3].

15.26. The distinguishing table is

	A	B	C	D	
$\mathbf{z}(0)$	0	0	1	1	$\surd$
$\mathbf{z}(1)$	0	1	0	1	$\surd$
$\mathbf{z}(2)$	0	0	0	0	

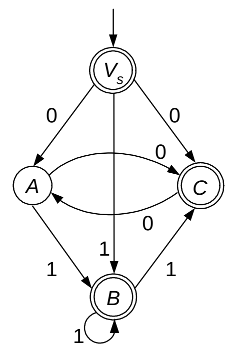
From the table we find

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} 0 \end{bmatrix}$$

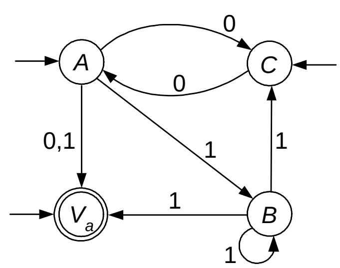
Matrices **A** and **B** can be verified to satisfy all state-input combinations. However, matrices **C** and **D** do not conform to the state table of the machine. (In particular, the output symbol in state D, column 1, must be 1 for the machine to be linear.) Thus, the machine in question is state-linear, but output-nonlinear.

Chapter 16

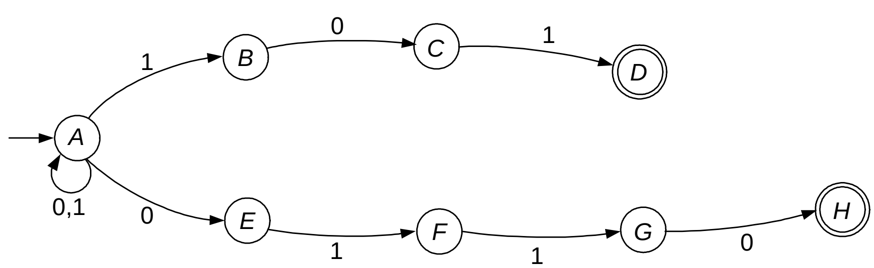
16.2.
(a) Add a new vertex V_s and designate it as the starting vertex. Draw arcs leading from V_s in such a way that every α-successor of one of the original starting vertices is also made an α-successor of V_s. Vertex V_s is designated as an accepting vertex if and only if one or more of the original starting vertices were accepting vertices.



(b) Add a new vertex V_a and designate it as the accepting vertex. An arc labeled α is drawn from vertex V_i to vertex V_a if and only if at least one of the original accepting vertices was an α-successor of V_i. Vertex V_i is designated as a starting vertex if and only if one or more of the original accepting vertices was a starting vertex.



16.5. The nondeterministic graph is shown below; it actually consists of a self-loop around vertex A and two chains of transitions. The self-loop provides for an arbitrary number of symbols preceding the sequences 101 and 0110.



The conversion of the graph into deterministic form is a routine matter.

This approach, although requiring a conversion of a graph from nondeterministic to deterministic form, is quite straightforward; most of the work is purely mechanical. On the other hand, the direct design of a state diagram is intricate and requires careful analysis at each step.

16.6.

(a) The set of strings starting with 11, followed by an arbitrary number (possibly zero) of 0's, and ending with either 0 or 1.

(b) The set of strings that start with a 1 and end with 101.

(c) The set of strings that start with an arbitrary number of 10 substrings, followed by an arbitrary number of 01 substrings, followed in turn by a concatenation of any combination of 00 and 11 substrings.

16.8.

(a)

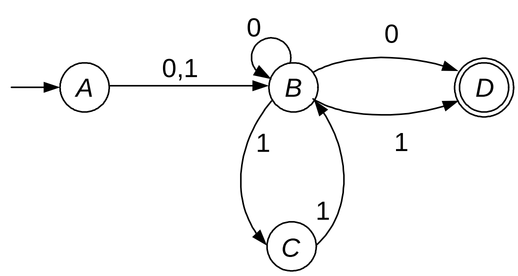
$$\begin{aligned} 10 + (1010)^*[\lambda^* + \lambda(1010)^*] &= 10 + (1010)^*[\lambda^* + (1010)^*] \\ &= 10 + (1010)^* + (1010)^*(1010)^* \\ &= 10 + (1010)^* \end{aligned}$$

(c)

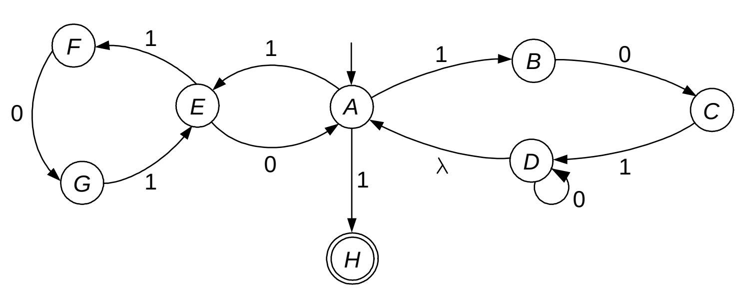
$$\begin{aligned} [(1^*0)^*01^*]^* &= [((1+0)^*0+\lambda)01^*]^* \qquad \text{(by 16.15)} \\ &= [((1+0)^*00+0)1^*]^* \qquad \text{(by 16.15)} \\ &= [(1+0)^*00+0][(1+0)^*00+0+1]^* + \lambda \\ &= [(1+0)^*00+0](1+0)^* + \lambda \\ &= \lambda + 0(0+1)^* + (0+1)^*00(0+1)^* \end{aligned}$$

16.10.

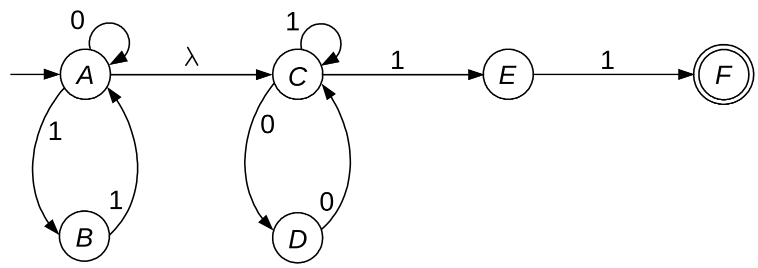
(a)



(b)



(c)



16.15. Only the set described in (b) is regular, since in all other cases, a machine that recognizes the set in question must be capable of counting an arbitrary number of 1's – a task that no finite-state machine can accomplish.

16.17.

(a) The reverse of an expression that contains no star operations is simply the expression written backwards. For example, the reverse of (10+1)01(011+10) is (01+110)10(1+01). The reverse of the set A^* is $(A^r)^*$. For example, the reverse of $(1 + 01^*)^*$ is $(1^*0 + 1)^*$. Thus, the reverse R^r of a regular expression R can be obtained by shifting each star immediately to the left of the subexpression to which it applies and reversing the resulting expression

(b) $R^r = (0^*11^*0)^*0^*1 + (0^*1 + 0)^*(00)^*$

16.18.

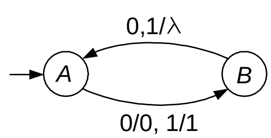
(a) Yes, since every *finite* set of strings is regular and can be recognized by a finite-state machine.

(b) No. Since, for example, P does not contain the string λ.

(c) No. Let R denote the set of all strings $(0+1)^*$. Clearly, the set of palindromes is a subset of $(0+1)^*$, but is not regular (see Problem 10.3).

(d) No. See answer to (c).

16.21. Let M be the finite-state machine whose state diagram is shown below (assuming an alphabet {0,1}). Suppose that M receives its input from another finite-state machine whose outputs are strings from P. If the inputs to M are strings from P, then the outputs of M (with λ's deleted) are strings of Q. However, since the set of strings that can be produced as output strings by a finite-state machine M, starting from its initial state, is regular, then Q must be regular. The generalization of this result to nonbinary alphabets is straightforward.



16.22. Let M be a finite-state machine that recognizes P, and suppose that M has n states $S_1, S_2, \cdots, S_n$, where S_1 is the starting state. Define U_j to be the set of strings that take M from S_1 to S_j. U_j is clearly regular.

Define next V_j to be the set of strings that take M from S_j to an accepting state. V_j is clearly also regular. And since the intersection of regular sets is regular, we conclude that $U_j \cap V_j$ is regular. Finally, Q can be expressed as

$$Q = P \cap (U_1 \cap V_1 + U_2 \cap V_2 + \cdots U_n \cap V_n) = P \cap \sum_{j=1}^{n} U_j \cap V_j$$

which is clearly regular.

16.23.

(a) Let M be a finite-state machine that recognizes the set R. We shall now show how M can be modified so that it recognizes R_x. Determine the x-successor of the initial state S_0 and let this state be the starting state of the modified machine. From the definition of R_x, it is evident that it consists of all strings, and only those strings, that correspond to paths leading from the new starting state to the accepting states of M. Thus, R_x is regular.

(b) There is only a finite number of ways in which M can be modified, in the above manner, since M has a finite number of states. Thus, there is a finite number of distinct derivatives R_x. In fact, if R can be recognized by a transition graph G with k vertices, there are at most 2^k ways of modifying this graph, corresponding to the 2^k subsets of the vertices of G and, consequently, there are at most 2^k distinct derivatives. (Note that in the case of non-deterministic graphs, a *set* of vertices must, in general, be assigned as the new starting vertices.)

16.26. Since there are np combinations of states and squares on the tape, the machine must be in a cycle if it moves more than np times.

16.28. Note that A, B, and C are regular sets. A is clearly regular; B is regular because it can be recognized by the automaton that recognizes A, if the outputs associated with the states are complemented; C is regular because it is the complement of $A + B$, and the set $A + B$ is regular.

We can now construct three machines, one to recognize each of the sets A, B, and C. These machines can be viewed as submachines of a composite machine whose final action (to accept, reject, or go into a cycle) is based on which submachine accepts the tape. This machine can be made to accept, reject, or cycle on any combinations of the sets A, B, and C.