

Chapter 8

8.1. The expressions for the inputs of the flip-flops are

$$\begin{aligned} D_2 &= Y_2 = \overline{w}y_2 + \overline{y}_1\overline{y}_2 \\ D_1 &= Y_1 = w \oplus y_1 \oplus y_2 \end{aligned}$$

The output equation is

$$z = y_1y_2$$

8.2. The excitation table for JK flip-flops is

Present state y_2y_1	Flip-flop inputs				Output z
	$w = 0$		$w = 1$		
	J_2K_2	J_1K_1	J_2K_2	J_1K_1	
00	1d	0d	1d	1d	0
01	0d	d0	0d	d1	0
10	d0	1d	d1	0d	0
11	d0	d1	d1	d0	1

The expressions for the inputs of the flip-flops are

$$\begin{aligned} J_2 &= \overline{y}_1 \\ K_2 &= w \\ J_1 &= \overline{w}y_2 + w\overline{y}_2 \\ K_1 &= J_1 \end{aligned}$$

The output equation is

$$z = y_1y_2$$

8.3. A possible state table is

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	B	0	0
B	E	C	0	0
C	E	D	0	0
D	E	D	0	1
E	F	B	0	0
F	A	B	0	1

```

8.4.  LIBRARY ieee ;
      USE ieee.std_logic_1164.all ;

```

```

ENTITY prob8_4 IS
  PORT ( Clock   : IN   STD_LOGIC ;
        Resetn   : IN   STD_LOGIC ;
        w        : IN   STD_LOGIC ;
        z        : OUT  STD_LOGIC ) ;
END prob8_4 ;

```

```

ARCHITECTURE Behavior OF prob8_4 IS
  TYPE State_type IS ( A, B, C, D, E, F ) ;
  SIGNAL y : State_type ;
BEGIN
  PROCESS ( Resetn, Clock )
  BEGIN
    IF Resetn = '0' THEN
      y <= A ;
    ELSIF Clock'EVENT AND Clock = '1' THEN
      CASE y IS
        WHEN A =>
          IF w = '0' THEN y <= A ;
          ELSE y <= B ;
          END IF ;
        WHEN B =>
          IF w = '0' THEN y <= E ;
          ELSE y <= C ;
          END IF ;
        WHEN C =>
          IF w = '0' THEN y <= E ;
          ELSE y <= D ;
          END IF ;
        WHEN D =>
          IF w = '0' THEN y <= E ;
          ELSE y <= D ;
          END IF ;
        WHEN E =>
          IF w = '0' THEN y <= F ;
          ELSE y <= B ;
          END IF ;
        WHEN F =>
          IF w = '0' THEN y <= A ;
          ELSE y <= B ;
          END IF ;
      END CASE ;
    END IF ;
  END PROCESS ;

```

... con't

```

PROCESS ( y, w )
BEGIN
  IF (y = D AND w = '1') OR (y = F AND w = '1') THEN
    z <= '1' ;
  ELSE
    z <= '0' ;
  END IF ;
END PROCESS ;
END Behavior ;

```

8.5. A minimal state table is

Present state	Next State		Output z
	$w = 0$	$w = 1$	
A	A	B	0
B	E	C	0
C	D	C	0
D	A	F	1
E	A	F	0
F	E	C	1

8.6. An initial attempt at deriving a state table may be

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	B	0	0
B	D	C	0	0
C	D	C	1	0
D	A	E	0	1
E	D	C	0	0

States B and E are equivalent; hence the minimal state table is

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	B	0	0
B	D	C	0	0
C	D	C	1	0
D	A	B	0	1

8.7. For Figure 8.51 have (using the straightforward state assignment):

	Present state $y_3 y_2 y_1$	Next state		Output z
		$w = 0$	$w = 1$	
		$Y_3 Y_2 Y_1$	$Y_3 Y_2 Y_1$	
A	0 0 0	0 0 1	0 1 0	1
B	0 0 1	0 1 1	1 0 1	1
C	0 1 0	1 0 1	1 0 0	0
D	0 1 1	0 0 1	1 1 0	1
E	1 0 0	1 0 1	0 1 0	0
F	1 0 1	1 0 0	0 1 1	0
G	1 1 0	1 0 1	1 1 0	0

This leads to

$$\begin{aligned}
 Y_3 &= \overline{w}y_3 + \overline{y}_1y_2 + wy_1\overline{y}_3 \\
 Y_2 &= wy_3 + w\overline{y}_1\overline{y}_2 + wy_1y_2 + \overline{w}y_1\overline{y}_2\overline{y}_3 \\
 Y_1 &= \overline{y}_3\overline{w} + \overline{y}_1\overline{w} + wy_1\overline{y}_2 \\
 z &= y_1\overline{y}_3 + \overline{y}_2\overline{y}_3
 \end{aligned}$$

For Figure 8.52 have

	Present state $y_2 y_1$	Next state		Output z
		$w = 0$	$w = 1$	
		$Y_2 Y_1$	$Y_2 Y_1$	
A	0 0	0 1	1 0	1
B	0 1	0 0	1 1	1
C	1 0	1 1	1 0	0
F	1 1	1 0	0 0	0

This leads to

$$\begin{aligned}
 Y_2 &= \overline{w}y_2 + \overline{y}_1y_2 + w\overline{y}_2 \\
 Y_1 &= \overline{y}_1\overline{w} + wy_1\overline{y}_2 \\
 z &= \overline{y}_2
 \end{aligned}$$

Clearly, minimizing the number of states leads to a much simpler circuit.

8.8. For Figure 8.55 have (using straightforward state assignment):

	Present state $y_4 y_3 y_2 y_1$	Next state				Output z
		DN=00	01	10	11	
		$Y_4 Y_3 Y_2 Y_1$				
S1	0 0 0 0	0 0 0 0	0 0 1 0	0 0 0 1	—	0
S2	0 0 0 1	0 0 0 1	0 0 1 1	0 1 0 0	—	0
S3	0 0 1 0	0 0 1 0	0 1 0 1	0 1 1 0	—	0
S4	0 0 1 1	0 0 0 0	—	—	—	1
S5	0 1 0 0	0 0 1 0	—	—	—	1
S6	0 1 0 1	0 1 0 1	0 1 1 1	1 0 0 0	—	0
S7	0 1 1 0	0 0 0 0	—	—	—	1
S8	0 1 1 1	0 0 0 0	—	—	—	1
S9	1 0 0 0	0 0 1 0	—	—	—	1

The next-state and output expressions are

$$\begin{aligned}
 Y_4 &= D y_3 \\
 Y_3 &= D y_1 + D y_2 + N y_2 + \overline{D} y_3 \overline{y}_2 y_1 \\
 Y_2 &= N \overline{y}_2 + y_3 \overline{y}_1 + \overline{N} \overline{y}_3 y_2 \overline{y}_1 \\
 Y_1 &= N y_2 + D \overline{y}_2 \overline{y}_1 + \overline{D} \overline{y}_2 y_1 \\
 z &= y_4 + y_1 y_2 + \overline{y}_1 y_3
 \end{aligned}$$

Using the same approach for Figure 8.56 gives

	Present state $y_3y_2y_1$	Next state				Output z
		DN=00	01	10	11	
		$Y_3Y_2Y_1$				
S1	0 0 0	0 0 0	0 1 0	0 0 1	—	0
S2	0 0 1	0 0 1	0 1 1	1 0 0	—	0
S3	0 1 0	0 1 0	0 0 1	0 1 1	—	0
S4	0 1 1	0 0 0	—	—	—	1
S5	1 0 0	0 1 0	—	—	—	1

The next-state and output expressions are:

$$\begin{aligned}
 Y_3 &= D \overline{y}_2 y_1 \\
 Y_2 &= y_3 + \overline{N} y_2 \overline{y}_1 + N \overline{y}_2 \\
 Y_1 &= \overline{D} \overline{y}_2 y_1 + N y_2 \overline{y}_1 + D \overline{y}_3 \overline{y}_1 \\
 z &= y_3 + y_2 y_1
 \end{aligned}$$

These expressions define a circuit that has considerably lower cost than the circuit resulting from Figure 8.55.

8.9. To compare individual bits, let $k = w_1 \oplus w_2$. Then, a suitable state table is

Present state	Next state		Output z	
	$k = 0$	$k = 1$	$k = 0$	$k = 1$
A	B	A	0	0
B	C	A	0	0
C	D	A	0	0
D	D	A	1	0

The state-assigned table is

Present state $y_2 y_1$	Next State		Output	
	$k = 0$	$k = 1$	$k = 0$	$k = 1$
	$Y_2 Y_1$	$Y_2 Y_1$	z	z
00	01	00	0	0
01	10	00	0	0
10	11	00	0	0
11	11	00	1	0

The next-state and output expressions are

$$\begin{aligned} Y_2 &= \bar{k}y_1 + \bar{k}y_2 \\ Y_1 &= \bar{k}\bar{y}_1 + \bar{k}y_2 \\ z &= \bar{k}y_1y_2 \end{aligned}$$

8.10. `LIBRARY ieee ;`
 `USE ieee.std_logic_1164.all ;`

```
ENTITY prob8_10 IS
    PORT ( Clock   : IN   STD_LOGIC ;
          Resethn  : IN   STD_LOGIC ;
          w1, w2   : IN   STD_LOGIC ;
          z        : OUT  STD_LOGIC ) ;
END prob8_10 ;
```

```
ARCHITECTURE Behavior OF prob8_10 IS
    TYPE State_type IS ( A, B, C, D ) ;
    SIGNAL y : State_type ;
    SIGNAL k : STD_LOGIC ;

    ... con't
```

```

BEGIN
  k <= w1 XOR w2 ;
  PROCESS ( Resetn, Clock )
  BEGIN
    IF Resetn = '0' THEN
      y <= A ;
    ELSIF (Clock'EVENT AND Clock = '1') THEN
      CASE y IS
        WHEN A =>
          IF k = '0' THEN y <= B ;
          ELSE y <= A ;
          END IF ;
        WHEN B =>
          IF k = '0' THEN y <= C ;
          ELSE y <= A ;
          END IF ;
        WHEN C =>
          IF k = '0' THEN y <= D ;
          ELSE y <= A ;
          END IF ;
        WHEN D =>
          IF k = '0' THEN y <= D ;
          ELSE y <= A ;
          END IF ;
      END CASE ;
    END IF ;
  END PROCESS ;

  z <= '1' WHEN y = D AND k = '0' ELSE '0' ;
END Behavior ;

```

8.11. A possible minimum state table for a Moore-type FSM is

Present state	Next state		Output z
	w = 0	w = 1	
A	B	C	0
B	D	E	0
C	E	D	0
D	F	G	0
E	F	F	0
F	A	A	0
G	A	A	1

8.12. A minimum state table is shown below. We assume that the 3-bit patterns do not overlap.

Present state	Next state		Output p
	w = 0	w = 1	
A	B	C	0
B	D	E	0
C	E	D	0
D	A	F	0
E	F	A	0
F	B	C	1

8.13. LIBRARY ieee ;
 USE ieee.std_logic_1164.all ;

```
ENTITY prob8_13 IS
    PORT ( Clock   : IN   STD_LOGIC ;
           Resetn  : IN   STD_LOGIC ;
           w       : IN   STD_LOGIC ;
           p       : OUT  STD_LOGIC ) ;
END prob8_13 ;
```

```
ARCHITECTURE Behavior OF prob8_13 IS
    TYPE State_type IS ( A, B, C, D, E, F ) ;
    SIGNAL y : State_type ;
BEGIN
    PROCESS ( Resetn, Clock )
    BEGIN
        IF Resetn = '0' THEN
            y <= A ;
        ELSIF (Clock'EVENT AND Clock = '1') THEN
            CASE y IS
                WHEN A =>
                    IF w = '0' THEN y <= B ;
                    ELSE y <= C ;
                    END IF ;
                WHEN B =>
                    IF w = '0' THEN y <= D ;
                    ELSE y <= E ;
                    END IF ;
                WHEN C =>
                    IF w = '0' THEN y <= E ;
                    ELSE y <= D ;
                    END IF ;
            END CASE ;
        END IF ;
    END PROCESS ;
... con't
```



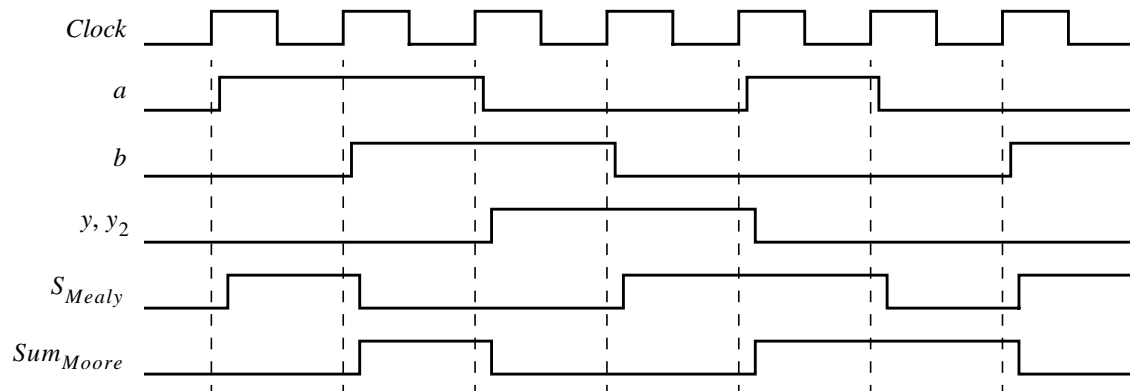
```

    WHEN D =>
        IF w = '0' THEN y <= A ;
        ELSE y <= F ;
        END IF ;
    WHEN E =>
        IF w = '0' THEN y <= F ;
        ELSE y <= A ;
        END IF ;
    WHEN F =>
        IF w = '0' THEN y <= B ;
        ELSE y <= C ;
        END IF ;
    END CASE ;
END IF ;
END PROCESS ;

p <= '1' WHEN y = F ELSE '0' ;
END Behavior ;

```

8.14. The timing diagram is



8.15. The state table corresponding to Figure P8.1 is

Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	C	D	0
B	B	A	0
C	D	A	0
D	C	B	1

Using one-hot encoding, the state-assigned table is

	Present state $y_4 y_3 y_2 y_1$	Next state		Output z
		$w = 0$	$w = 1$	
		$Y_4 Y_3 Y_2 Y_1$	$Y_4 Y_3 Y_2 Y_1$	
A	0 0 0 1	0 1 0 0	1 0 0 0	0
B	0 0 1 0	0 0 1 0	0 0 0 1	0
C	0 1 0 0	1 0 0 0	0 0 0 1	0
D	1 0 0 0	0 1 0 0	0 0 1 0	1

The next-state expressions are

$$\begin{aligned}
 D_4 &= Y_4 = \overline{w}y_3 + wy_1 \\
 D_3 &= Y_3 = \overline{w}(y_1 + y_4) \\
 D_2 &= Y_2 = \overline{w}y_2 + wy_4 \\
 D_1 &= Y_1 = w(y_2 + y_1)
 \end{aligned}$$

The output is given by $z = y_4$.

- 8.16. The state-assignment given in problem 8.15 can be used, except that the state variable y_1 should be complemented. Thus, the state assignment will be $y_4 y_3 y_2 y_1 = 0000, 0011, 0101$, and 1001 , for the states A, B, C , and D , respectively. The circuit derived in problem 8.15 can be used, except that the signal for the state variable y_1 should be taken from the \overline{Q} output of flip-flop 1, rather than from its Q output.

- 8.17. The partitioning process gives

$$\begin{aligned}
 P_1 &= (ABCDEFGG) \\
 P_2 &= (ABD)(CEFG) \\
 P_3 &= (ABD)(CEG)(F) \\
 P_4 &= (ABD)(CEG)(F)
 \end{aligned}$$

The minimum state table is

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	A	C	0	0
C	F	C	0	1
F	C	A	0	1

- 8.18. The partitioning process gives

$$\begin{aligned}
 P_1 &= (ABCDEFGG) \\
 P_2 &= (ADG)(BCEF) \\
 P_3 &= (AG)(D)(B)(CE)(F) \\
 P_4 &= (A)(G)(D)(B)(CE)(F)
 \end{aligned}$$

The minimized state table is

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A	B	C	0	0
B	D	—	0	1
C	F	C	0	1
D	B	G	0	0
F	C	D	0	1
G	F	—	0	0

- 8.19. An implementation for the Moore-type FSM in Figures 8.5.7 and 8.5.6 is given in the solution for problem 8.8. The Mealy-type FSM in Figure 8.58 is described in the form of a state table as

Present state	Next state				Output z			
	DN=00	01	10	11	00	01	10	11
S1	S1	S3	S2	—	0	0	0	1
S2	S2	S1	S3	—	0	1	1	—
S3	S3	S2	S1	—	0	0	1	—

The state-assigned table is

Present state $y_2 y_1$	Next state				Output			
	DN=00	01	10	11	00	01	10	11
	$Y_2 Y_1$	$Y_2 Y_1$	$Y_2 Y_1$	$Y_2 Y_1$	z	z	z	z
00	00	10	01	—	0	0	0	—
01	01	00	10	—	0	1	1	—
10	10	01	00	—	0	0	1	—

The next-state and output expressions are

$$\begin{aligned}
 Y_2 &= D y_1 + \overline{D} y_2 \overline{N} + N \overline{y}_2 \overline{y}_1 \\
 Y_1 &= N y_2 + \overline{D} y_1 \overline{N} + D \overline{y}_2 \overline{y}_1 \\
 z &= D y_1 + D y_2 + N y_1
 \end{aligned}$$

In this case, choosing the Mealy model results in a simpler circuit.

8.20. Use w as the clock. Then the state table is

Present state	Next state	Output $z_1 z_0$
A	B	0 0
B	C	1 0
C	D	0 1
D	A	1 1

The state-assigned table is

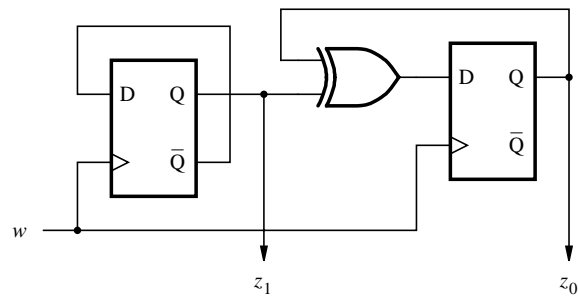
Present state $y_1 y_0$	Next state $Y_1 Y_0$	Output $z_1 z_0$
0 0	1 0	0 0
1 0	0 1	1 0
0 1	1 1	0 1
1 1	0 0	1 1

The next-state expressions are

$$Y_1 = \bar{y}_1$$

$$Y_2 = y_1 \oplus y_2$$

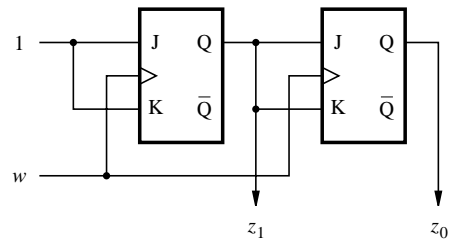
The resulting circuit is



8.21. From the state-assigned table given in the solution to Problem 8.20, the excitation table for JK flip-flops is

Present state $y_1 y_0$	Flip-flop inputs		Output $z_1 z_0$
	$J_1 K_1$	$J_0 K_0$	
0 0	1 d	0 d	0 0
1 0	d 1	1 d	1 0
0 1	1 d	d 0	0 1
1 1	d 1	d 1	1 1

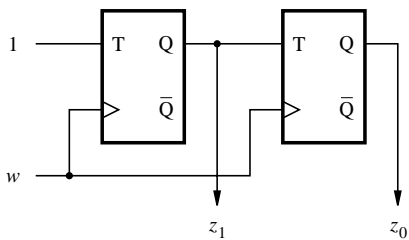
The flip-flop inputs are $J_1 = K_1 = 1$ and $J_2 = K_2 = y_1$. The resulting circuit is



8.22. From the state-assigned table given in the solution to Problem 8.20, the excitation table for T flip-flops is

Present state $y_1 y_0$	Flip-flop inputs		Output $z_1 z_0$
	T_1	T_0	
0 0	1	0	0 0
1 0	1	1	1 0
0 1	1	0	0 1
1 1	1	1	1 1

The flip-flop inputs are $T_1 = 1$ and $T_2 = y_1$. The resulting circuit is



8.23. The state diagram is

Present state	Next state		Output $z_2 z_1 z_0$
	$w = 0$	$w = 1$	
A	A	B	0 0 0
B	B	C	0 0 1
C	C	D	0 1 0
D	D	E	0 1 1
E	E	F	1 0 0
F	F	A	1 0 1

The state-assigned table is

Present state $y_2y_1y_0$	Next state		Output $z_2z_1z_0$
	$w = 0$	$w = 1$	
	$Y_2Y_1Y_0$		
000	000	001	000
001	001	010	001
010	010	011	010
011	011	100	011
100	100	101	100
101	101	000	101

The next-state expressions are

$$\begin{aligned} Y_2 &= \bar{y}_0 y_2 + \bar{w} y_2 + w y_0 y_1 \\ Y_1 &= \bar{y}_0 y_1 + \bar{w} y_1 + w y_0 \bar{y}_1 \bar{y}_2 \\ Y_0 &= \bar{w} y_0 + w \bar{y}_0 \end{aligned}$$

The outputs are: $z_2 = y_2$, $z_1 = y_1$, and $z_0 = y_0$.

8.24. Using the state-assigned table given in the solution for problem 8.23, the excitation table for JK flip-flops is

Present state $y_2 y_1 y_0$	Flip-flop inputs						Outputs $z_2 z_1 z_0$
	$w = 0$			$w = 1$			
	$J_2 K_2$	$J_1 K_1$	$J_0 K_0$	$J_2 K_2$	$J_1 K_1$	$J_0 K_0$	
000	0 d	0 d	0 d	0 d	0 d	1 d	000
001	0 d	0 d	d 0	0 d	1 d	d 1	001
010	0 d	d 0	0 d	0 d	d 0	1 d	010
011	0 d	d 0	d 0	1 d	d 1	d 1	011
100	d 0	0 d	0 d	d 0	0 d	1 d	100
101	d 0	0 d	d 0	d 1	0 d	d 1	101

The expressions for the inputs of the flip-flops are

$$\begin{aligned} J_2 &= w y_1 y_0 \\ K_2 &= w y_2 y_0 \\ J_1 &= w \bar{y}_2 y_0 \\ K_1 &= w y_0 \\ J_0 &= w \\ K_0 &= w \end{aligned}$$

The outputs are: $z_2 = y_2$, $z_1 = y_1$, and $z_0 = y_0$.

8.25. Using the state-assigned table given in the solution for problem 8.23, the excitation table for T flip-flops is

Present state $y_2 y_1 y_0$	Flip-flop inputs		Outputs $z_2 z_1 z_0$
	$w = 0$	$w = 1$	
	$T_2 T_1 T_0$	$T_2 T_1 T_0$	
000	000	001	000
001	000	011	001
010	000	001	010
011	000	111	011
100	000	001	100
101	000	101	101

The expressions for T inputs of the flip-flops are

$$T_2 = w y_1 y_0 + w y_2 y_0$$

$$T_1 = w \overline{y}_2 y_0$$

$$T_0 = w$$

The outputs are: $z_2 = y_2$, $z_1 = y_1$, and $z_0 = y_0$.

8.26. The state diagram is

Present state	Next state		Count
	$w = 0$	$w = 1$	
A	H	C	0
B	A	D	1
C	B	E	2
D	C	F	3
E	D	G	4
F	E	H	5
G	F	A	6
H	G	B	7

The state-assigned table is

	Present state $y_2 y_1 y_0$	Next state		Output $z_2 z_1 z_0$
		$w = 0$	$w = 1$	
		$Y_2 Y_1 Y_0$	$Y_2 Y_1 Y_0$	
A	0 0 0	1 1 1	0 1 0	0 0 0
B	0 0 1	0 0 0	0 1 1	0 0 1
C	0 1 0	0 0 1	1 0 0	0 1 0
D	0 1 1	0 1 0	1 0 1	0 1 1
E	1 0 0	0 1 1	1 1 0	1 0 0
F	1 0 1	1 0 0	1 1 1	1 0 1
G	1 1 0	1 0 1	0 0 0	1 1 0
H	1 1 1	1 1 0	0 0 1	1 1 1

The next-state expressions (inputs to D flip-flops) are

$$D_2 = Y_2 = w\overline{y}_2 y_1 + \overline{w} y_2 y_1 + w y_2 \overline{y}_1 + \overline{w} y_2 y_0 + \overline{y}_2 \overline{y}_1 \overline{y}_0 w$$

$$D_1 = Y_1 = w\overline{y}_1 + \overline{y}_1 \overline{y}_0 + \overline{w} y_1 y_0$$

$$D_0 = Y_0 = \overline{y}_0 \overline{w} + y_0 w$$

The outputs are: $z_2 = y_2$, $z_1 = y_1$, and $z_0 = y_0$.

8.27. From the state-assigned table given in the solution to problem 8.26, the excitation table for JK flip-flops is

Present state $y_2 y_1 y_0$	Flip-flop inputs						Outputs $z_2 z_1 z_0$
	$w = 0$			$w = 1$			
	$J_2 K_2$	$J_1 K_1$	$J_0 K_0$	$J_2 K_2$	$J_1 K_1$	$J_0 K_0$	
000	1 d	1 d	1 d	0 d	1 d	0 d	000
001	0 d	0 d	d 1	0 d	1 d	d 0	001
010	0 d	d 1	1 d	1 d	d 1	0 d	010
011	0 d	d 0	d 1	1 d	d 1	d 0	011
100	d 1	1 d	1 d	d 0	1 d	0 d	100
101	d 0	0 d	d 1	d 0	1 d	d 0	101
110	d 0	d 1	1 d	d 1	d 1	0 d	110
111	d 0	d 0	d 1	d 1	d 1	d 0	111

The expressions for J and K inputs to the three flip-flops are

$$J_2 = y_1 w + \overline{y}_1 \overline{y}_0 \overline{w}$$

$$K_2 = J_2$$

$$J_1 = w + \overline{y}_0$$

$$K_1 = J_1$$

$$J_0 = \overline{w}$$

$$K_0 = J_0$$

The outputs are: $z_2 = y_2$, $z_1 = y_1$, and $z_0 = y_0$.

8.28. From the state-assigned table given in the solution to problem 8.26, the excitation table for T flip-flops is

Present state $y_2 y_1 y_0$	Flip-flop inputs		Outputs $z_2 z_1 z_0$
	$w = 0$	$w = 1$	
	$T_2 T_1 T_0$	$T_2 T_1 T_0$	
000	111	010	000
001	001	010	001
010	011	110	010
011	001	110	011
100	111	010	100
101	001	010	101
110	011	110	110
111	001	110	111

The expressions for T inputs of the flip-flops are

$$T_2 = \overline{y_1} \overline{y_0} \overline{w} + y_1 w$$

$$T_1 = w + \overline{y_0}$$

$$T_0 = \overline{w}$$

The outputs are: $z_2 = y_2$, $z_1 = y_1$, and $z_0 = y_0$.

8.29. The next-state and output expressions are

$$D_1 = Y_1 = w(y_1 + y_2)$$

$$D_2 = Y_2 = w(\overline{y_1} + \overline{y_2})$$

$$z = y_1 \overline{y_2}$$

The corresponding state-assigned table is

Present state $y_2 y_1$	Next state		Output z
	$w = 0$	$w = 1$	
	$Y_2 Y_1$	$Y_2 Y_1$	
0 0	0 0	1 0	0
0 1	0 0	1 1	1
1 0	0 0	1 1	0
1 1	0 0	0 1	0

This leads to the state table

Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	A	C	0
B	A	D	1
C	A	D	0
D	A	B	0

The circuit produces $z = 1$ whenever the input sequence on w comprises a 0 followed by an even number of 1s.

```

8.30.  LIBRARY ieee ;
        USE ieee.std_logic_1164.all ;

        ENTITY prob8_30 IS
            PORT ( Clock   : IN    STD_LOGIC ;
                  Resetn   : IN    STD_LOGIC ;
                  N, D     : IN    STD_LOGIC ;
                  z        : OUT   STD_LOGIC ) ;
        END prob8_30 ;

        ARCHITECTURE Behavior OF prob8_30 IS
            TYPE State_type IS ( S1, S2, S3, S4, S5 ) ;
            SIGNAL y : State_type ;
        BEGIN
            PROCESS ( Resetn, Clock )
            BEGIN
                IF Resetn = '0' THEN
                    y <= S1 ;
                ELSIF (Clock'EVENT AND Clock = '1') THEN
                    CASE y IS
                        WHEN S1 =>
                            IF N = '1' THEN y <= S3 ;
                            ELSIF D = '1' THEN y <= S2 ;
                            ELSE y <= S1 ;
                            END IF ;
                        WHEN S2 =>
                            IF N = '1' THEN y <= S4 ;
                            ELSIF D = '1' THEN y <= S5 ;
                            ELSE y <= S2 ;
                            END IF ;
                        WHEN S3 =>
                            IF N = '1' THEN y <= S2 ;
                            ELSIF D = '1' THEN y <= S4 ;
                            ELSE y <= S3 ;
                            END IF ;
                        WHEN S4 =>
                            y <= S1 ;
                        WHEN S5 =>
                            y <= S3 ;
                    END CASE ;
                END IF ;
            END PROCESS ;

            z <= '1' WHEN y = S4 OR y = S5 ELSE '0' ;
        END Behavior ;

```

```

8.31.  LIBRARY ieee ;
        USE ieee.std_logic_1164.all ;

        ENTITY prob8_32 IS
            PORT ( Resetn, Clock : IN    STD_LOGIC ;
                  N, D          : IN    STD_LOGIC ;
                  z              : OUT  STD_LOGIC ) ;
        END prob8_32 ;

        ARCHITECTURE Behavior OF prob8_32 IS
            TYPE State_type IS ( S1, S2, S3 ) ;
            SIGNAL y : State_type ;
        BEGIN
            PROCESS ( Resetn, Clock )
            BEGIN
                IF Resetn = '0' THEN
                    y <= S1 ;
                ELSIF Clock'EVENT AND Clock = '1' THEN
                    CASE y IS
                        WHEN S1 =>
                            IF N = '1' THEN y <= S3 ;
                            ELSIF D = '1' THEN y <= S2 ;
                            ELSE y <= S1 ; END IF ;
                        WHEN S2 =>
                            IF N = '1' THEN y <= S1 ;
                            ELSIF D = '1' THEN y <= S3 ;
                            ELSE y <= S2 ; END IF ;
                        WHEN S3 =>
                            IF N = '1' THEN y <= S2 ;
                            ELSIF D = '1' THEN y <= S1 ;
                            ELSE y <= S3 ; END IF ;
                    END CASE ;
                END IF ;
            END PROCESS ;

            z <= '1' WHEN (y = S2 AND (D = '1' OR N = '1')) OR (y = S3 AND D = '1') ELSE '0' ;
        END Behavior ;

```

```

8.32.  LIBRARY ieee ;
        USE ieee.std_logic_1164.all ;

        ENTITY prob8_32 IS
            PORT ( Clock : IN    STD_LOGIC ;
                  Resetn : IN    STD_LOGIC ;
                  N, D   : IN    STD_LOGIC ;
                  z       : OUT  STD_LOGIC ) ;
        END prob8_32 ;

        ... con't

```

ARCHITECTURE Behavior OF prob8_32 IS

TYPE State_type IS (S1, S2, S3);

SIGNAL y : State_type ;

BEGIN

PROCESS (Resetn, Clock)

BEGIN

IF Resetn = '0' THEN

y <= S1 ;

ELSIF Clock'EVENT AND Clock = '1' THEN

CASE y IS

WHEN S1 =>

IF N = '1' THEN y <= S3 ;

ELSIF D = '1' THEN y <= S2 ;

ELSE y <= S1 ;

END IF ;

WHEN S2 =>

IF N = '1' THEN y <= S1 ;

ELSIF D = '1' THEN y <= S3 ;

ELSE y <= S2 ;

END IF ;

WHEN S3 =>

IF N = '1' THEN y <= S2 ;

ELSIF D = '1' THEN y <= S1 ;

ELSE y <= S3 ;

END IF ;

END CASE ;

END IF ;

END PROCESS ;

z <= '1' WHEN (y = S2 AND (D = '1' OR N = '1')) OR (y = S3 AND D = '1') ELSE '0' ;

END Behavior ;

8.33. LIBRARY ieee ;

USE ieee.std_logic_1164.all ;

ENTITY prob8_33 IS

PORT (Clock : IN STD_LOGIC ;

Resetn : IN STD_LOGIC ;

N, D : IN STD_LOGIC ;

z : OUT STD_LOGIC) ;

END prob8_33 ;

ARCHITECTURE Behavior OF prob8_33 IS

TYPE State_type IS (S1, S2, S3) ;

SIGNAL y_present, y_next : State_type ;

... con't

```

BEGIN
  PROCESS ( N, D, y_present)
  BEGIN
    CASE y_present IS
      WHEN S1 =>
        IF N = '1' THEN y_next <= S3 ;
        ELSIF D = '1' THEN y_next <= S2 ;
        ELSE y_next <= S1 ;
        END IF ;
      WHEN S2 =>
        IF N = '1' THEN y_next <= S1 ;
        ELSIF D = '1' THEN y_next <= S3 ;
        ELSE y_next <= S2 ;
        END IF ;
      WHEN S3 =>
        IF N = '1' THEN y_next <= S2 ;
        ELSIF D = '1' THEN y_next <= S1 ;
        ELSE y_next <= S3 ;
        END IF ;
    END CASE ;
  END PROCESS ;

  PROCESS ( Clock, Resetn )
  BEGIN
    IF Resetn = '0' THEN
      y_present <= S1 ;
    ELSIF Clock'EVENT AND Clock = '1' THEN
      y_present <= y_next ;
    END IF ;
  END PROCESS ;

  z <= '1' WHEN (y_present = S2 AND (D = '1' OR N = '1')) OR
    (y_present = S3 AND D = '1') ELSE '0' ;
END Behavior ;

```

8.34. LIBRARY ieee ;
 USE ieee.std_logic_1164.all ;

```

ENTITY prob8_34 IS
  PORT ( Clock   : IN   STD_LOGIC ;
        Resetn   : IN   STD_LOGIC ;
        w        : IN   STD_LOGIC ;
        z        : OUT  STD_LOGIC ) ;
END prob8_34 ;

```

... con't

```

ARCHITECTURE Behavior OF prob8_34 IS
    TYPE State_type IS ( A, B, C, D );
    ATTRIBUTE ENUM_ENCODING : STRING ;
    ATTRIBUTE ENUM_ENCODING OF State_type : TYPE IS "00 01 10 11" ;
    SIGNAL y : State_type ;
BEGIN
    PROCESS ( Resetn, Clock )
    BEGIN
        IF Resetn = '0' THEN
            y <= A ;
        ELSIF Clock'EVENT AND Clock = '1' THEN
            CASE y IS
                WHEN A =>
                    IF w = '0' THEN y <= C ;
                    ELSE y <= D ;
                    END IF ;
                WHEN B =>
                    IF w = '0' THEN y <= B ;
                    ELSE y <= A ;
                    END IF ;
                WHEN C =>
                    IF w = '0' THEN y <= D ;
                    ELSE y <= A ;
                    END IF ;
                WHEN D =>
                    IF w = '0' THEN y <= C ;
                    ELSE y <= B ;
                    END IF ;
            END CASE ;
        END IF ;
    END PROCESS ;

    z <= '1' WHEN y = D ELSE '0' ;
END Behavior ;

```

8.35. LIBRARY ieee ;
 USE ieee.std_logic_1164.all ;

```

ENTITY prob8_35 IS
    PORT ( Clock   : IN   STD_LOGIC ;
          Resetn   : IN   STD_LOGIC ;
          w        : IN   STD_LOGIC ;
          z        : OUT  STD_LOGIC );
END prob8_35 ;

... con't

```

```

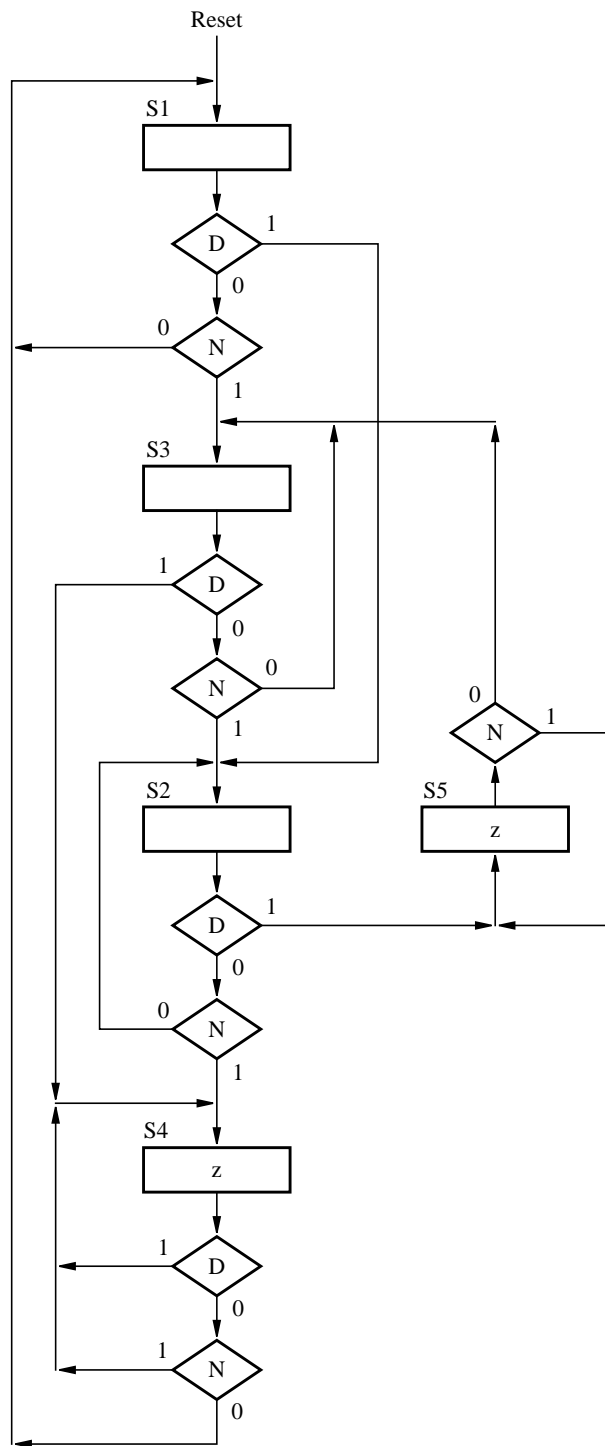
ARCHITECTURE Behavior OF prob8_35 IS
    SIGNAL y_present, y_next : STD_LOGIC_VECTOR(1 DOWNTO 0) ;
    CONSTANT A : STD_LOGIC_VECTOR(1 DOWNTO 0) := "00" ;
    CONSTANT B : STD_LOGIC_VECTOR(1 DOWNTO 0) := "01" ;
    CONSTANT C : STD_LOGIC_VECTOR(1 DOWNTO 0) := "10" ;
    CONSTANT D : STD_LOGIC_VECTOR(1 DOWNTO 0) := "11" ;
BEGIN
    PROCESS ( w, y_present )
    BEGIN
        CASE y_present IS
            WHEN A =>
                IF w = '0' THEN y_next <= C ;
                ELSE y_next <= D ;
                END IF ;
            WHEN B =>
                IF w = '0' THEN y_next <= B ;
                ELSE y_next <= A ;
                END IF ;
            WHEN C =>
                IF w = '0' THEN y_next <= D ;
                ELSE y_next <= A ;
                END IF ;
            WHEN OTHERS =>
                IF w = '0' THEN y_next <= C ;
                ELSE y_next <= B ;
                END IF ;
        END CASE ;
    END PROCESS ;

    PROCESS ( Clock, Resetn )
    BEGIN
        IF Resetn = '0' THEN
            y_present <= A ;
        ELSIF Clock'EVENT AND Clock = '1' THEN
            y_present <= y_next ;
        END IF ;
    END PROCESS ;

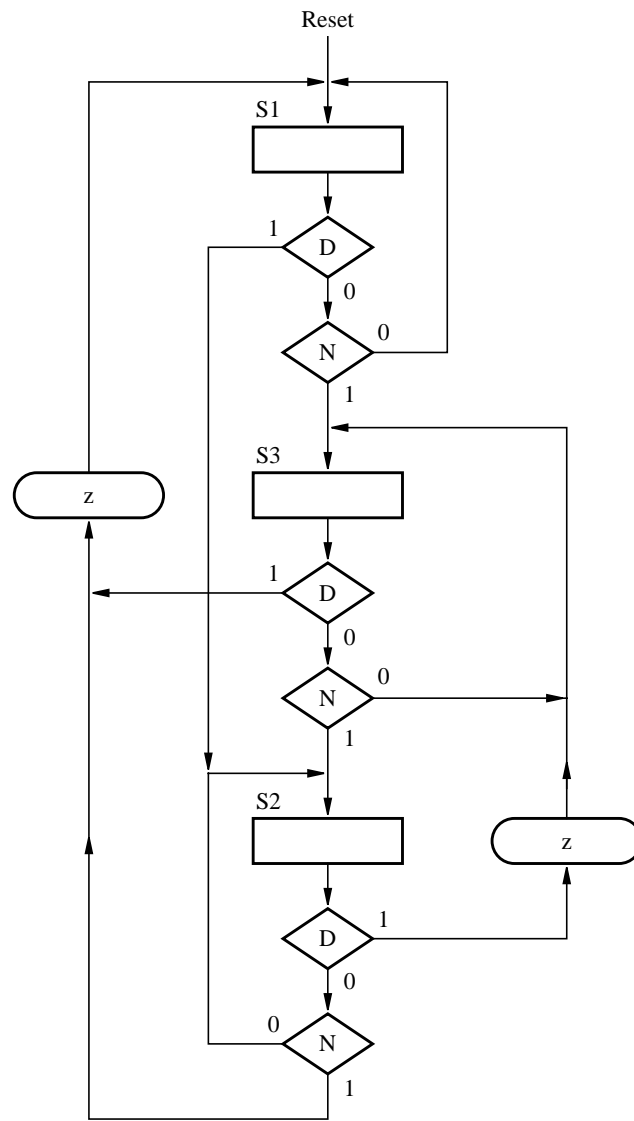
    z <= '1' WHEN y_present = D ELSE '0' ;
END Behavior ;

```

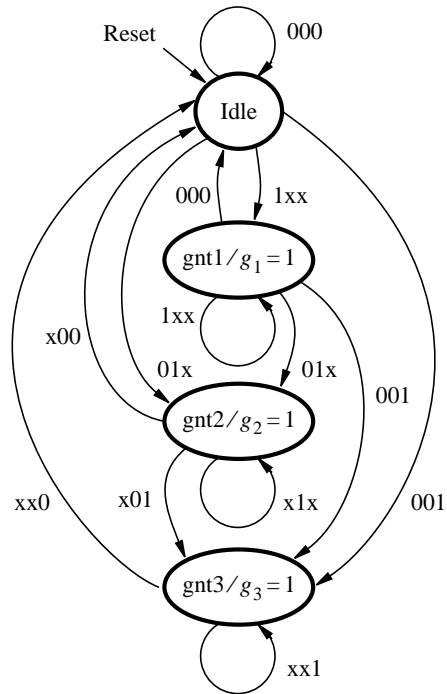
8.36. An ASM chart for the FSM in Figure 8.57 is



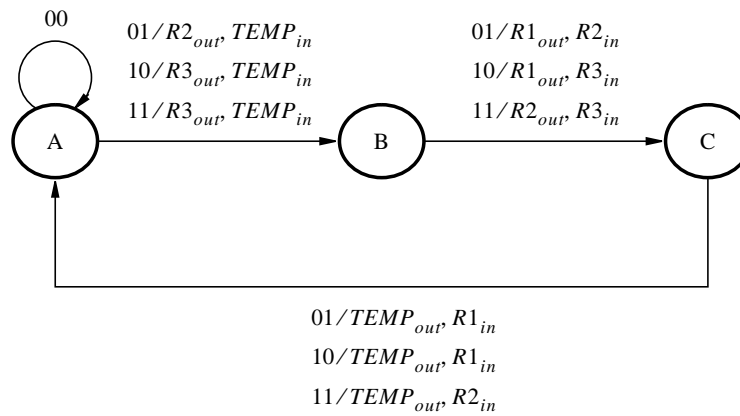
8.37. An ASM chart for the FSM in Figure 8.58 is



8.38. To ensure that the device 3 will get serviced the FSM in Figure 8.72 can be modified as follows:



8.40. The required control signals can be generated using the following FSM:



Let $k = w_2 + w_1$. Then the next-state transitions can be defined as

Present state	Next state	
	$k = 0$	$k = 1$
A	A	B
B	B	C
C	C	A

Using one-hot encoding, the state-assigned table becomes

Present state $y_3 y_2 y_1$	Next state	
	$k = 0$	$k = 1$
	$Y_3 Y_2 Y_1$	$Y_3 Y_2 Y_1$
001	001	010
010	010	100
100	100	001

The next-state expressions are

$$\begin{aligned} Y_3 &= \bar{k} y_3 + k y_2 \\ Y_2 &= \bar{k} y_2 + k y_1 \\ Y_1 &= \bar{k} y_1 + k y_3 \end{aligned}$$

The output expressions are

$$\begin{aligned} TEMP_{in} &= k y_1 \\ TEMP_{out} &= k y_3 \\ R1_{out} &= y_2 (w_2 \oplus w_1) \\ R1_{in} &= y_3 (w_2 \oplus w_1) \\ R2_{out} &= y_1 \bar{w}_2 w_1 + y_2 w_2 w_1 \\ R2_{in} &= y_2 \bar{w}_2 w_1 + y_3 w_2 w_1 \\ R3_{out} &= y_1 w_2 \\ R3_{in} &= y_2 w_2 \end{aligned}$$