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

$$z = y_1 y_2$$

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

$$z = y_1 y_2$$

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. Verilog code for the solution given in problem 8.3 is

```
module prob8_4 (Clock, Resetn, w, z);
    input Clock, Resetn, w;
    output z;
    reg z;
    reg [3:1] y, Y;
    parameter [3:1] A = 3'b000, B = 3'b001, C = 3'b010, D = 3'b011, E = 3'b100, F = 3'b101;
    // Define the next state and output combinational circuits
    always @(w or y)
        case (y)
            A:   if (w)   begin
                              Y = B;   z = 0;
                          end
                 else     begin
                              Y = A;   z = 0;
                          end
            B:   if (w)   begin
                              Y = C;   z = 0;
                          end
                 else     begin
                              Y = E;   z = 0;
                          end
            C:   if (w)   begin
                              Y = D;   z = 0;
                          end
                 else     begin
                              Y = E;   z = 0;
                          end
            D:   if (w)   begin
                              Y = D;   z = 1;
                          end
                 else     begin
                              Y = E;   z = 0;
                          end
            E:   if (w)   begin
                              Y = B;   z = 0;
                          end
                 else     begin
                              Y = F;   z = 0;
                          end
            F:   if (w)   begin
                              Y = B;   z = 1;
                          end
                 else     begin
                              Y = A;   z = 0;
                          end
            default:      begin
                              Y = 3'bxxx;   z = 0;
                          end
        endcase
    // Define the sequential block
    always @(negedge Resetn or posedge Clock)
        if  (Resetn == 0)   y <= A;
        else   y <= Y;

endmodule
```

8.5. A minimal state table is

Present state	Next State $w=0$	Next State $w=1$	Output z
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 $y_3y_2y_1$	Next state $w=0$ $Y_3Y_2Y_1$	Next state $w=1$ $Y_3Y_2Y_1$	Output z
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_2y_1	Next state $w=0$ Y_2Y_1	Next state $w=1$ Y_2Y_1	Output z
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 $y_4y_3y_2y_1$	Next state DN=00	01	10	11	Output z
		$Y_4Y_3Y_2Y_1$				
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

$$\begin{aligned}
Y_4 &= Dy_3 \\
Y_3 &= Dy_1 + Dy_2 + Ny_2 + \overline{D}y_3\overline{y}_2y_1 \\
Y_2 &= N\overline{y}_2 + y_3\overline{y}_1 + \overline{N}\overline{y}_3y_2\overline{y}_1 \\
Y_1 &= Ny_2 + D\overline{y}_2\overline{y}_1 + \overline{D}\overline{y}_2y_1 \\
z &= y_4 + y_1y_2 + \overline{y}_1y_3
\end{aligned}$$

Using the same approach for Figure 8.56 gives

	Present state $y_3y_2y_1$	Next state DN=00	01	10	11	Output z
		$Y_3Y_2Y_1$				
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:

$$\begin{aligned}
Y_3 &= D\overline{y}_2y_1 \\
Y_2 &= y_3 + \overline{N}y_2\overline{y}_1 + N\overline{y}_2 \\
Y_1 &= \overline{D}\overline{y}_2y_1 + Ny_2\overline{y}_1 + D\overline{y}_3\overline{y}_1 \\
z &= y_3 + y_2y_1
\end{aligned}$$

These expressions define a circuit that has considerably lower cost that 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	Next State		Output	
state	$k=0$	$k=1$	$k=0$	$k=1$
y_2y_1	Y_2Y_1	Y_2Y_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

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

8.10. Verilog code for the solution given in problem 8.9 is

```
module prob8_10 (Clock, Resetn, w1, w2, z);
    input Clock, Resetn, w1, w2;
    output z;
    reg z;
    reg [2:1] y, Y;
    wire k;
    parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;

    // Define the next state and output combinational circuits
    assign k = w1 ^ w2;
    always @(k or y)
        case (y)
            A:  if (k)  begin
                            Y = A;  z = 0;
                        end
                else    begin
                            Y = B;  z = 0;
                        end
            B:  if (k)  begin
                            Y = A;  z = 0;
                        end
                else    begin
                            Y = C;  z = 0;
                        end
            C:  if (k)  begin
                            Y = A;  z = 0;
                        end
                else    begin
                            Y = D;  z = 0;
                        end
            D:  if (k)  begin
                            Y = A;  z = 0;
                        end
                else    begin
                            Y = D;  z = 1;
                        end
        endcase

    // Define the sequential block
    always @(negedge Resetn or posedge Clock)
        if (Resetn == 0)  y <= A;
        else  y <= Y;

endmodule
```

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

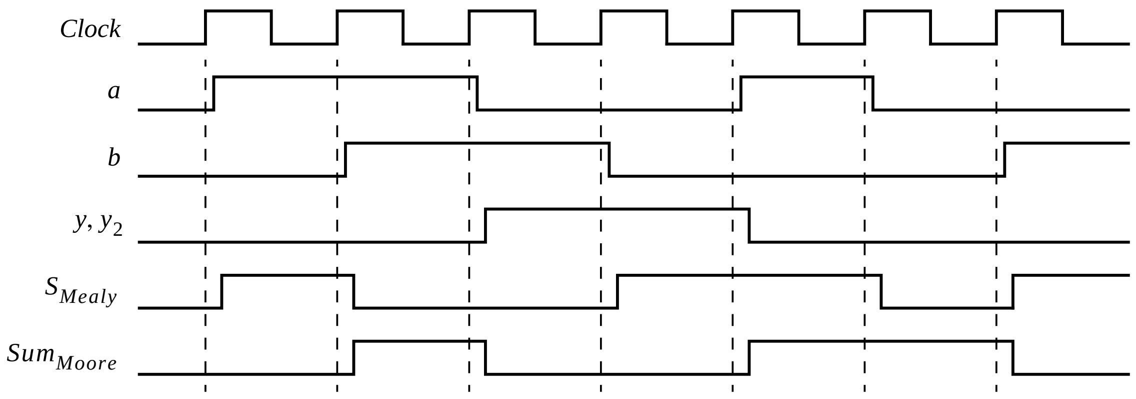
8.12. A minimum state table is shown below. We assume that the 3-bit patterns do not overlap.

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. Verilog code for the solution given in problem 8.12 is

```
module prob8_13 (Clock, Resetn, w, p);
    input Clock, Resetn, w;
    output p;
    reg [3:1] y, Y;
    parameter [3:1] A = 3'b000, B = 3'b001, C = 3'b010, D = 3'b011, E = 3'b100, F = 3'b101;

    // Define the next state combinational circuit
    always @(w or y)
        case (y)
            A:  if (w)    Y = C;
                else      Y = B;
            B:  if (w)    Y = E;
                else      Y = D;
            C:  if (w)    Y = D;
                else      Y = E;
            D:  if (w)    Y = F;
                else      Y = A;
            E:  if (w)    Y = A;
                else      Y = F;
            F:  if (w)    Y = C;
                else      Y = B;
            default:      Y = 3'bxxx;
        endcase

    // Define the sequential block
    always @(negedge Resetn or posedge Clock)
        if (Resetn == 0)  y <= A;
        else  y <= Y;

    // Define output
    assign p = (y == F);
endmodule
```

8.14. The timing diagram is



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

Present state	Next state		Output
	$w = 0$	$w = 1$	z
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	Next state		Output
		$w = 0$	$w = 1$	
	$y_4y_3y_2y_1$	$Y_4Y_3Y_2Y_1$	$Y_4Y_3Y_2Y_1$	z
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 $y_4y_3y_2y_1 = 0000, 0011, 0101,$ 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 $\overline{\text{Q}}$ output of flip-flop 1, rather than from its Q output.

8.17. The partitioning process gives

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

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

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

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	Next state				Output			
state	DN=00	01	10	11	00	01	10	11
y_2y_1	Y_2Y_1	Y_2Y_1	Y_2Y_1	Y_2Y_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

$$\begin{aligned} Y_2 &= Dy_1 + \overline{D}y_2\overline{N} + N\overline{y}_2\overline{y}_1 \\ Y_1 &= Ny_2 + \overline{D}y_1\overline{N} + D\overline{y}_2\overline{y}_1 \\ z &= Dy_1 + Dy_2 + Ny_1 \end{aligned}$$

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_1z_0
A	B	0 0
B	C	1 0
C	D	0 1
D	A	1 1

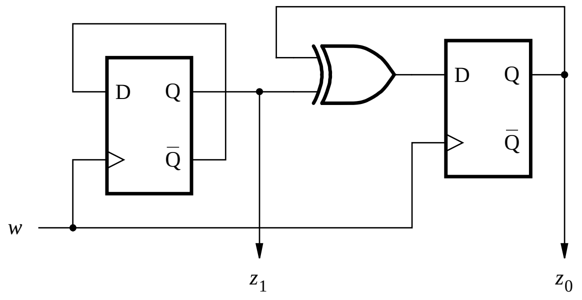
The state-assigned table is

Present state y_1y_0	Next state Y_1Y_0	Output z_1z_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

$$\begin{aligned} Y_1 &= \overline{y}_1 \\ Y_2 &= y_1 \oplus y_2 \end{aligned}$$

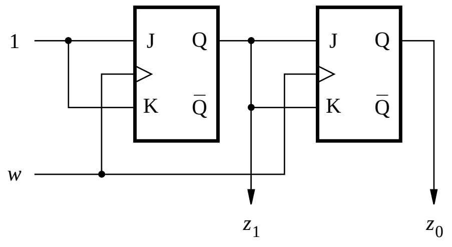
The resulting circuit is



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_1y_0	Flip-flop inputs J_1K_1	Flip-flop inputs J_0K_0	Output z_1z_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

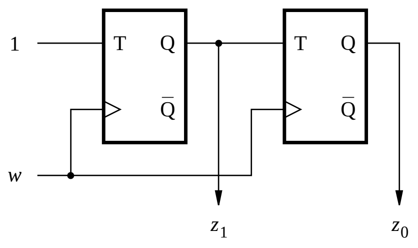
The flip-flop inputs are $J_1 = K_1 = 1$ and $J_2 = K_2 = y_1$. The resulting circuit is



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_1y_0	Flip-flop inputs T_1	Flip-flop inputs T_0	Output z_1z_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
	$w = 0$	$w = 1$	$z_2z_1z_0$
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	Next state		Output
	$w = 0$	$w = 1$	
$y_2y_1y_0$	$Y_2Y_1Y_0$		$z_2z_1z_0$
0 0 0	0 0 0	0 0 1	0 0 0
0 0 1	0 0 1	0 1 0	0 0 1
0 1 0	0 1 0	0 1 1	0 1 0
0 1 1	0 1 1	1 0 0	0 1 1
1 0 0	1 0 0	1 0 1	1 0 0
1 0 1	1 0 1	0 0 0	1 0 1

The next-state expressions are

$$\begin{aligned} Y_2 &= \overline{y}_0y_2 + \overline{w}y_2 + wy_0y_1 \\ Y_1 &= \overline{y}_0y_1 + \overline{w}y_1 + wy_0\overline{y}_1\overline{y}_2 \\ Y_0 &= \overline{w}y_0 + w\overline{y}_0 \end{aligned}$$

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	Flip-flop inputs						Outputs
	$w=0$			$w=1$			
$y_2y_1y_0$	J_2K_2	J_1K_1	J_0K_0	J_2K_2	J_1K_1	J_0K_0	$z_2z_1z_0$
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

$$\begin{aligned} J_2 &= wy_1y_0 \\ K_2 &= wy_2y_0 \\ J_1 &= w\overline{y}_2y_0 \\ K_1 &= wy_0 \\ J_0 &= w \\ K_0 &= w \end{aligned}$$

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	Flip-flop inputs		Outputs
	$w=0$	$w=1$	
$y_2y_1y_0$	$T_2T_1T_0$	$T_2T_1T_0$	$z_2z_1z_0$
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

$$\begin{aligned} T_2 &= wy_1y_0 + wy_2y_0 \\ T_1 &= w\overline{y}_2y_0 \\ T_0 &= w \end{aligned}$$

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 $w=0$	Next state $w=1$	Count
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	Next state		Output
		$w=0$	$w=1$	
	$y_2y_1y_0$	$Y_2Y_1Y_0$	$Y_2Y_1Y_0$	$z_2z_1z_0$
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

$$\begin{aligned} D_2 &= Y_2 &=& \; w\overline{y}_2y_1 + \overline{w}y_2y_1 + wy_2\overline{y}_1 + \overline{w}y_2y_0 + \overline{y}_2\overline{y}_1\overline{y}_0w \\ D_1 &= Y_1 &=& \; w\overline{y}_1 + \overline{y}_1\overline{y}_0 + \overline{w}y_1y_0 \\ D_0 &= Y_0 &=& \; \overline{y}_0\overline{w} + y_0w \end{aligned}$$

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	Flip-flop inputs						Outputs
	$w=0$			$w=1$			
$y_2y_1y_0$	J_2K_2	J_1K_1	J_0K_0	J_2K_2	J_1K_1	J_0K_0	$z_2z_1z_0$
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

$$\begin{aligned} J_2 &= y_1w + \overline{y}_1\overline{y}_0\overline{w} \\ K_2 &= J_2 \\ J_1 &= w + \overline{y}_0 \\ K_1 &= J_1 \\ J_0 &= \overline{w} \\ K_0 &= J_0 \end{aligned}$$

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	Flip-flop inputs		Outputs
	$w=0$	$w=1$	
$y_2y_1y_0$	$T_2T_1T_0$	$T_2T_1T_0$	$z_2z_1z_0$
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

$$\begin{aligned} T_2 &= \overline{y}_1\overline{y}_0\overline{w} + y_1w \\ T_1 &= w + \overline{y}_0 \\ T_0 &= \overline{w} \end{aligned}$$

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

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

The corresponding state-assigned table is

Present state	Next state		Output
	$w = 0$	$w = 1$	
y_2y_1	Y_2Y_1	Y_2Y_1	z
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
	$w = 0$	$w = 1$	z
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. The Verilog code based on the style of code in Figure 8.29 is

```
module  prob8_30 (Clock, Resetn, D, N, z);
   input  Clock, Resetn, D, N;
   output  z;
   reg  [3:1] y, Y;
   wire  [1:0] K;
   parameter  [3:1] S1 = 3'b000, S2 = 3'b001, S3 = 3'b010, S4 = 3'b011, S5 = 3'b100;

   // Define the next state combinational circuit
   assign K = {D, N};
   always @(K or y)
      case (y)
         S1: if (K == 2'b00)   Y = S1;
             else if (K == 2'b01)   Y = S3;
             else if (K == 2'b10)   Y = S2;
             else   Y = 3'bxxx;
         S2: if (K == 2'b00)   Y = S2;
             else if (K == 2'b01)   Y = S4;
             else if (K == 2'b10)   Y = S5;
             else   Y = 3'bxxx;
         S3: if (K == 2'b00)   Y = S3;
             else if (K == 2'b01)   Y = S2;
             else if (K == 2'b10)   Y = S4;
             else   Y = 3'bxxx;
         S4: if (K == 2'b00)   Y = S1;
             else   Y = 3'bxxx;
         S5: if (K == 2'b00)   Y = S3;
             else   Y = 3'bxxx;
         default:   Y = 3'bxxx;
      endcase

   // Define the sequential block
   always @(negedge Resetn or posedge Clock)
      if  (Resetn == 0)   y <= S1;
      else   y <= Y;

   // Define output
   assign  z = (y == S4) | (y == S5);

endmodule
```

8.31. The Verilog code based on the style of code in Figure 8.34 is

```
module prob8_31 (Clock, Resetn, D, N, z);
    input Clock, Resetn, D, N;
    output z;
    reg [3:1] y;
    wire [1:0] K;
    parameter [3:1] S1 = 3'b000, S2 = 3'b001, S3 = 3'b010, S4 = 3'b011, S5 = 3'b100;

    assign K = {D, N};
    // Define the sequential block
    always @(negedge Resetn or posedge Clock)
        if (Resetn == 0)   y <= S1;
        else
            case (y)
                S1: if (K == 2'b00)   y <= S1;
                      else if (K == 2'b01)   y <= S3;
                      else if (K == 2'b10)   y <= S2;
                      else   y <= 3'bxxx;
                S2: if (K == 2'b00)   y <= S2;
                      else if (K == 2'b01)   y <= S4;
                      else if (K == 2'b10)   y <= S5;
                      else   y <= 3'bxxx;
                S3: if (K == 2'b00)   y <= S3;
                      else if (K == 2'b01)   y <= S2;
                      else if (K == 2'b10)   y <= S4;
                      else   y <= 3'bxxx;
                S4: if (K == 2'b00)   y <= S1;
                      else   y <= 3'bxxx;
                S5: if (K == 2'b00)   y <= S3;
                      else   y <= 3'bxxx;
                default:   y <= 3'bxxx;
            endcase

    // Define output
    assign z = (y == S4) | (y == S5);

endmodule
```

8.32. The Verilog code based on the style of code in Figure 8.29 is

```
module prob8_32 (Clock, Resetn, D, N, z);
    input Clock, Resetn, D, N;
    output z;
    reg z;
    reg [2:1] y, Y;
    wire [1:0] K;
    parameter [2:1] S1 = 2'b00, S2 = 2'b01, S3 = 2'b10;
```

cont'd

```
    // Define the next state and output combinational circuits
    assign K = {D, N};
    always @(K or y)
        case (y)
            S1:  if (K == 2’b00)         begin
                                             Y = S1;   z = 0;
                                         end
                 else if (K == 2’b01)    begin
                                             Y = S3;   z = 0;
                                         end
                 else if (K == 2’b10)    begin
                                             Y = S2;   z = 0;
                                         end
                 else                    begin
                                             Y = 2’bxx;   z = 1’bx;
                                         end
            S2:  if (K == 2’b00)         begin
                                             Y = S2;   z = 0;
                                         end
                 else if (K == 2’b01)    begin
                                             Y = S1;   z = 1;
                                         end
                 else if (K == 2’b10)    begin
                                             Y = S3;   z = 1;
                                         end
                 else                    begin
                                             Y = 2’bxx;   z = 1’bx;
                                         end
            S3:  if (K == 2’b00)         begin
                                             Y = S3;   z = 0;
                                         end
                 else if (K == 2’b01)    begin
                                             Y = S2;   z = 0;
                                         end
                 else if (K == 2’b10)    begin
                                             Y = S1;   z = 1;
                                         end
                 else                    begin
                                             Y = 2’bxx;   z = 1’bx;
                                         end
            default:                     begin
                                             Y = 2’bxx;   z = 1’bx;
                                         end
        endcase

    // Define the sequential block
    always @(negedge Resetn or posedge Clock)
        if  (Resetn == 0)   y <= S1;
        else   y <= Y;
endmodule
```

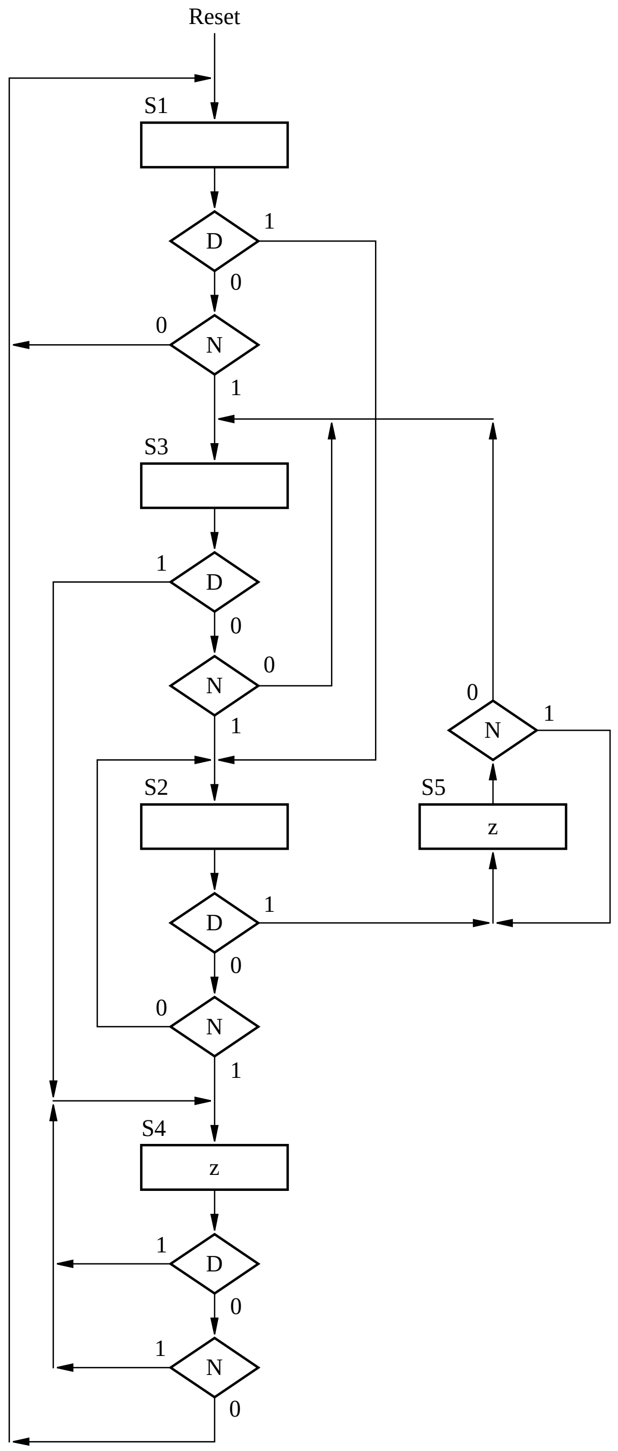
8.33. The Verilog code based on the style of code in Figure 8.34 is

```
module prob8_33 (Clock, Resetn, D, N, z);
    input Clock, Resetn, D, N;
    output z;
    reg [2:1] y;
    wire [1:0] K;
    parameter [2:1] S1 = 2'b00, S2 = 2'b01, S3 = 2'b10;

    assign K = {D, N};
    // Define the sequential block
    always @(negedge Resetn or posedge Clock)
        if (Resetn == 0)   y <= S1;
        else
            case (y)
                S1: if (K == 2'b00)   y <= S1;
                      else if (K == 2'b01)   y <= S3;
                      else if (K == 2'b10)   y <= S2;
                      else   y <= 2'bxx;
                S2: if (K == 2'b00)   y <= S2;
                      else if (K == 2'b01)   y <= S1;
                      else if (K == 2'b10)   y <= S3;
                      else   y <= 2'bxx;
                S3: if (K == 2'b00)   y <= S3;
                      else if (K == 2'b01)   y <= S2;
                      else if (K == 2'b10)   y <= S1;
                      else   y <= 2'bxx;
                default:   y <= 2'bxx;
            endcase

    // Define output
    assign z = ((y == S2) & ((K == 2'b01) | (K == 2'b10))) | ((y == S3) & (K == 2'b10));

endmodule
```

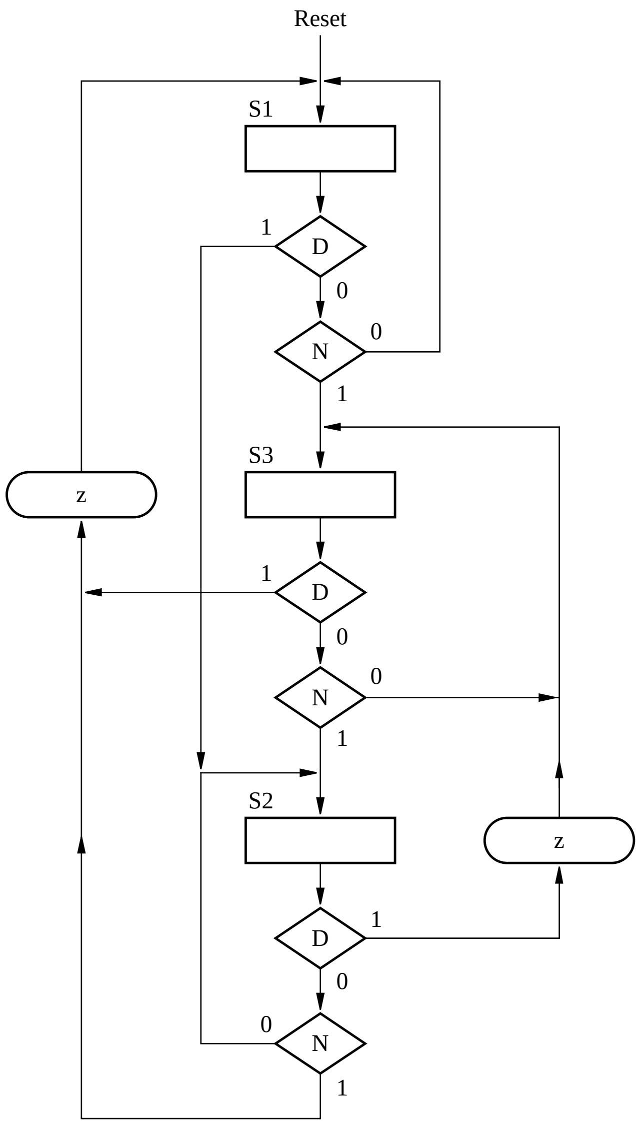
8.34. Verilog code for the FSM in Figure P8.2 is

```
module prob8_34 (Clock, Resetn, w, z);
    input Clock, Resetn, w;
    output z;
    reg [2:1] y, Y;
    parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10, D = 2'b11;

    // Define the next state combinational circuit
    always @(w or y)
        case (y)
            A: if (w)    Y = C;
               else      Y = A;
            B: if (w)    Y = D;
               else      Y = A;
            C: if (w)    Y = D;
               else      Y = A;
            D: if (w)    Y = B;
               else      Y = A;
        endcase

    // Define the sequential block
    always @(negedge Resetn or posedge Clock)
        if (Resetn == 0)  y <= A;
        else  y <= Y;

    // Define output
    assign z = (y == B);
endmodule
```

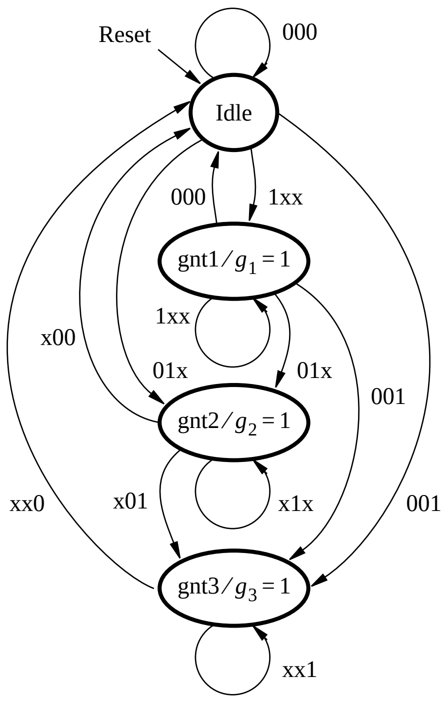
8.35. An ASM chart for the FSM in Figure 8.57 is



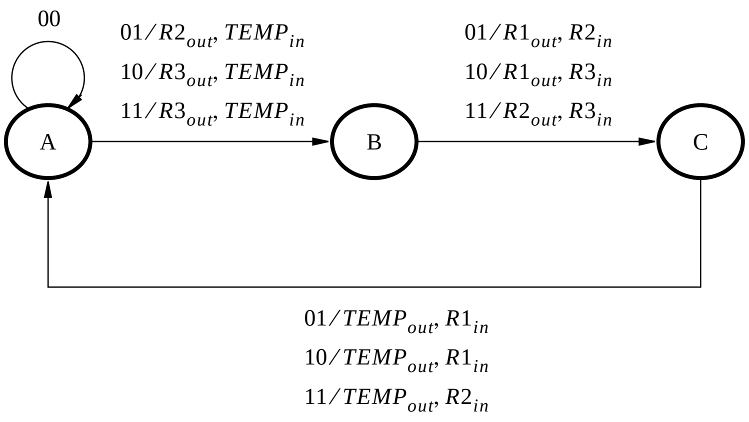
8.36. An ASM chart for the FSM in Figure 8.58 is



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



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



00
01/$R2_{out}$, $TEMP_{in}$
10/$R3_{out}$, $TEMP_{in}$
11/$R3_{out}$, $TEMP_{in}$
01/$R1_{out}$, $R2_{in}$
10/$R1_{out}$, $R3_{in}$
11/$R2_{out}$, $R3_{in}$
A
B
C
01/$TEMP_{out}$, $R1_{in}$
10/$TEMP_{out}$, $R1_{in}$
11/$TEMP_{out}$, $R2_{in}$

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	Next state	
	$k = 0$	$k = 1$
$y_3y_2y_1$	$Y_3Y_2Y_1$	$Y_3Y_2Y_1$
001	001	010
010	010	100
100	100	001

The next-state expressions are

$$\begin{aligned} Y_3 &= \overline{k}y_3 + ky_2 \\ Y_2 &= \overline{k}y_2 + ky_1 \\ Y_1 &= \overline{k}y_1 + ky_3 \end{aligned}$$

The output expressions are

$$\begin{aligned} TEMP_{in} &= ky_1 \\ TEMP_{out} &= ky_3 \\ R1_{out} &= y_2(w_2 \oplus w_1) \\ R1_{in} &= y_3(w_2 \oplus w_1) \\ R2_{out} &= y_1\overline{w}_2w_1 + y_2w_2w_1 \\ R2_{in} &= y_2\overline{w}_2w_1 + y_3w_2w_1 \\ R3_{out} &= y_1w_2 \\ R3_{in} &= y_2w_2 \end{aligned}$$