Chapter 6

Synchronous Sequential Circuits

In a combinational circuit, the values of the outputs are determined solely by the present values of its inputs.

In a sequential circuit, the values of the outputs depend on the past behavior of the circuit, as well as the present values of its inputs.

A sequential circuit has states, which in conjunction with the present values of inputs determine its behavior.

Sequential circuits can be:

- Synchronous – where flip-flops are used to implement the states, and a clock signal is used to control the operation

- Asynchronous – where no clock is used

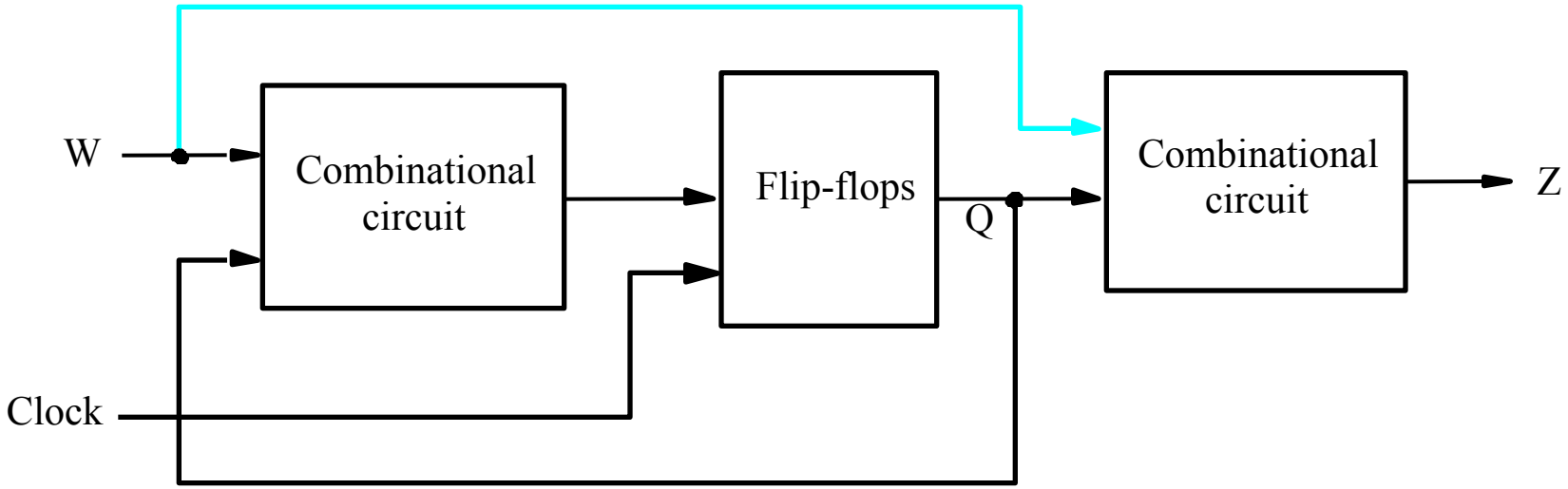


Figure 6.1. The general form of a synchronous sequential circuit.

If the outputs depend only on the present state, the circuit is said to be of Moore type.

If the outputs depend on both the present state and the present values of the inputs, the circuit is said to be of Mealy type.

Clockcycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	0	1	0	0	1	1	0

Figure 6.2. Sequences of input and output signals.

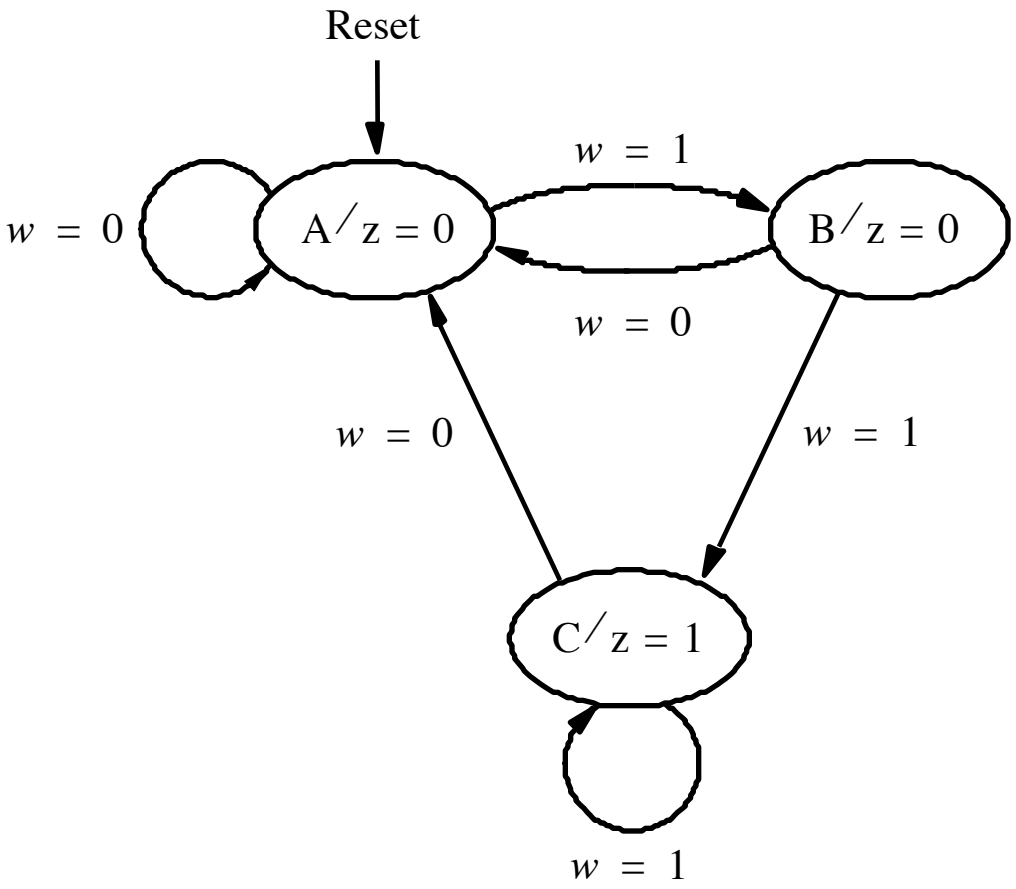


Figure 6.3. State diagram of a simple sequential circuit.

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

Figure 6.4. State table for the sequential circuit in Figure 6.3.

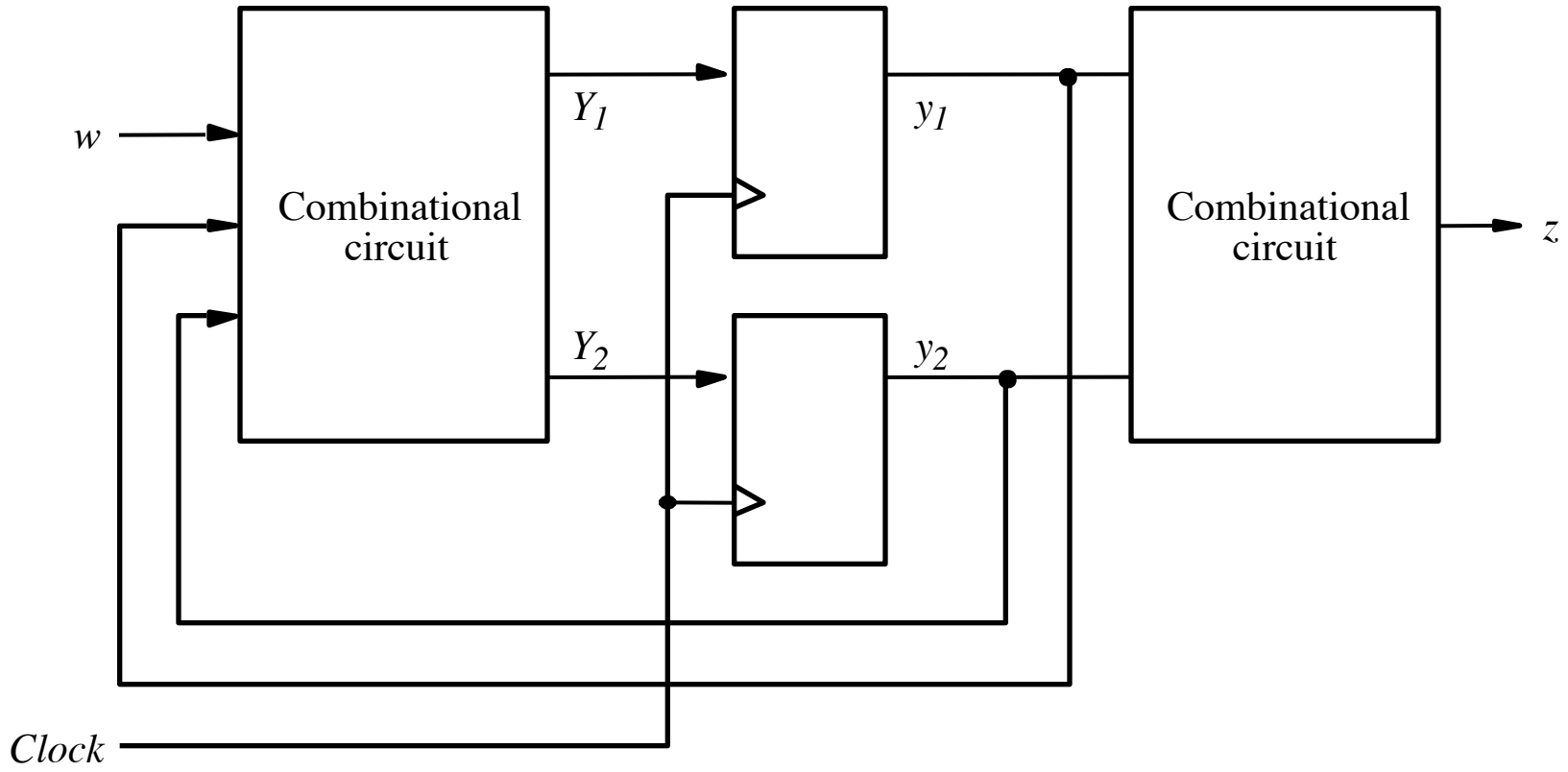


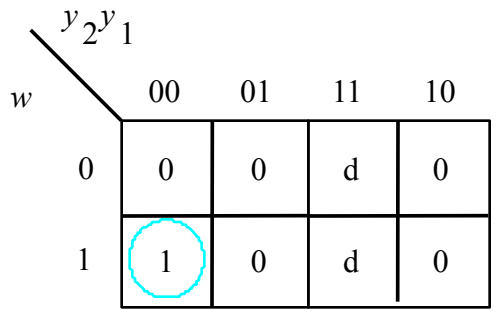
Figure 6.5. A general sequential circuit with input *w*, output *z*, and two state flip-flops.

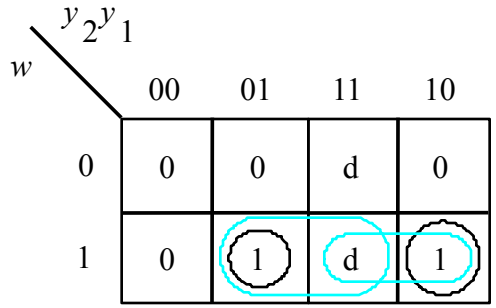
The present state variables, y_1 and y_2, determine the present state of the circuit.

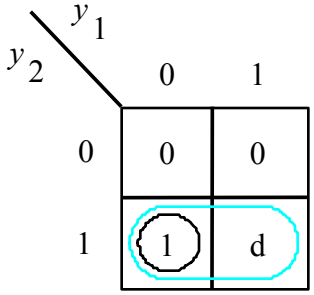
The next state variables, Y_1 and Y_2, determine the state into which the circuit will go after the next active edge of the clock signal.

	Present state y_2y_1	Next state $w = 0$ Y_2Y_1	Next state $w = 1$ Y_2Y_1	Output z
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

Figure 6.6. State-assigned table for the sequential circuit in Figure 6.4.







Ignoring don't cares

$$Y_1 = w\bar{y}_1\bar{y}_2$$

$$Y_2 = wy_1\bar{y}_2 + w\bar{y}_1y_2$$

$$z = \bar{y}_1y_2$$

Using don't cares

$$Y_1 = w\bar{y}_1\bar{y}_2$$

$$Y_2 = wy_1 + wy_2$$
$$= w(y_1 + y_2)$$

$$z = y_2$$

Figure 6.7. Derivation of logic expressions for the sequential circuit in Figure 6.6.

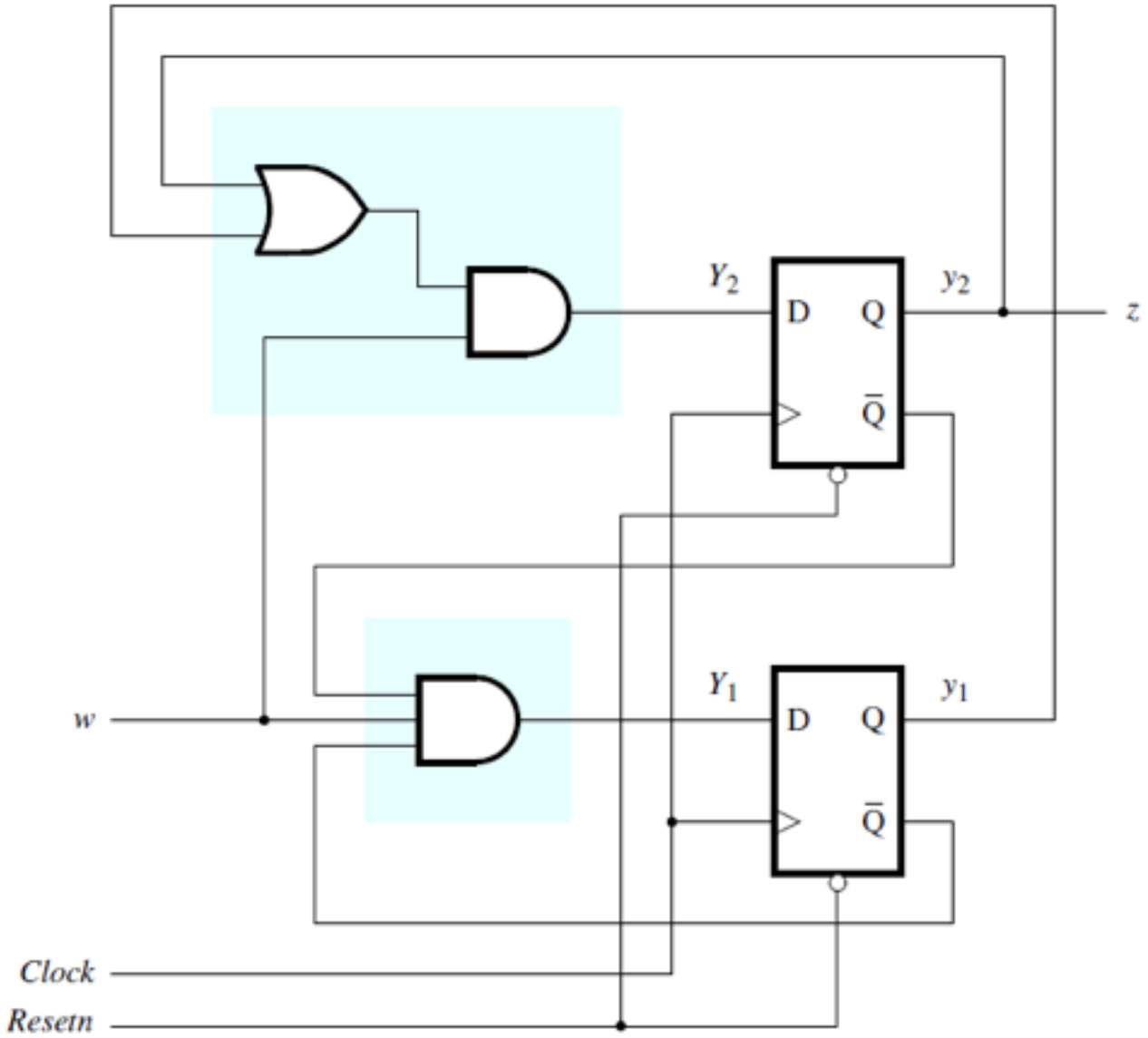


Figure 6.8. Final implementation of the sequential circuit derived in Figure 6.7.

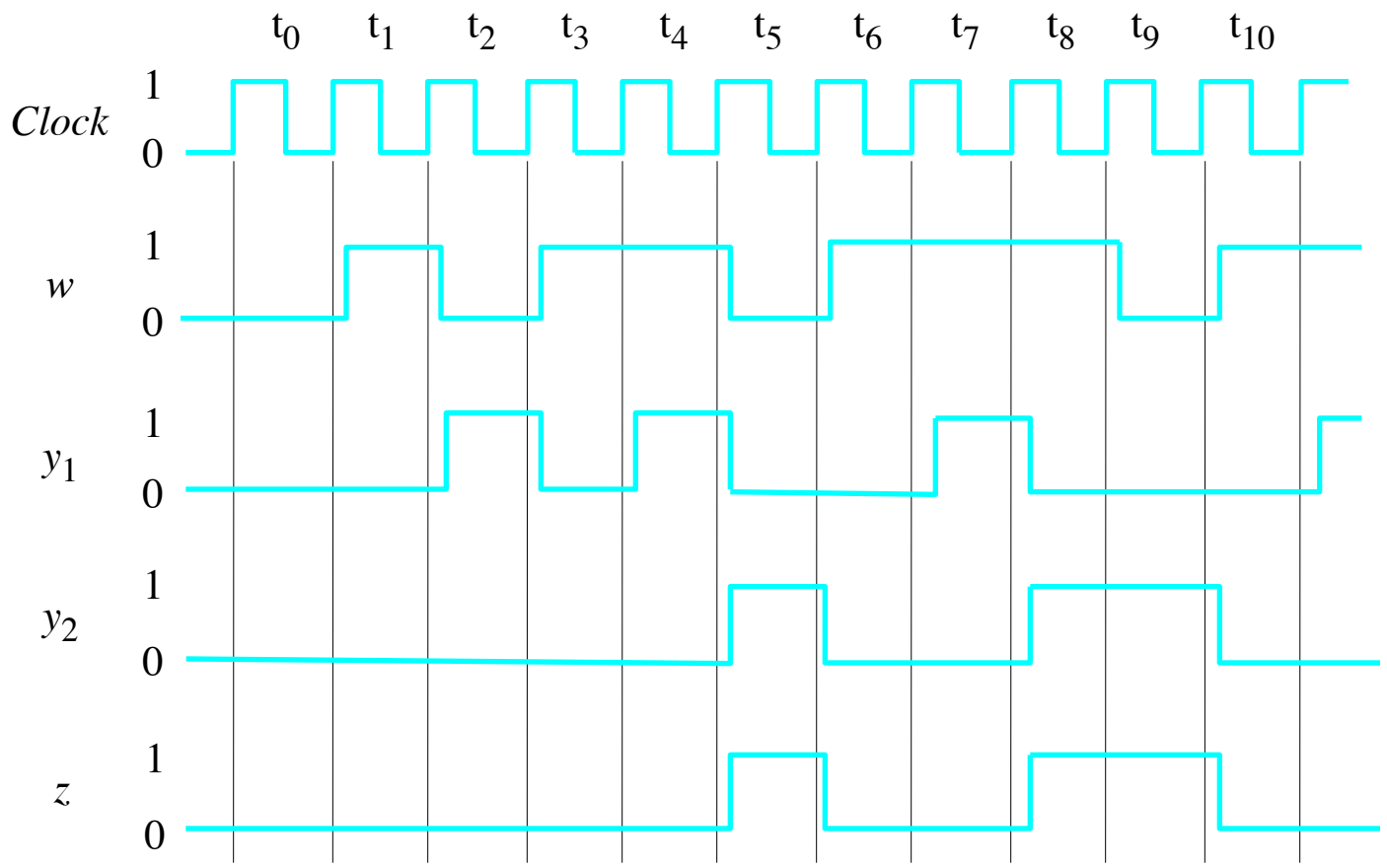


Figure 6.9. Timing diagram for the circuit in Figure 6.8.

Design steps:

1. Obtain the specification of the desired circuit.
2. Derive a state diagram.
3. Derive the corresponding state table.
4. Reduce the number of states if possible.
5. Decide on the number of state variables.
6. Choose the type of flip-flops to be used.
7. Derive the logic expressions needed to implement the circuit.

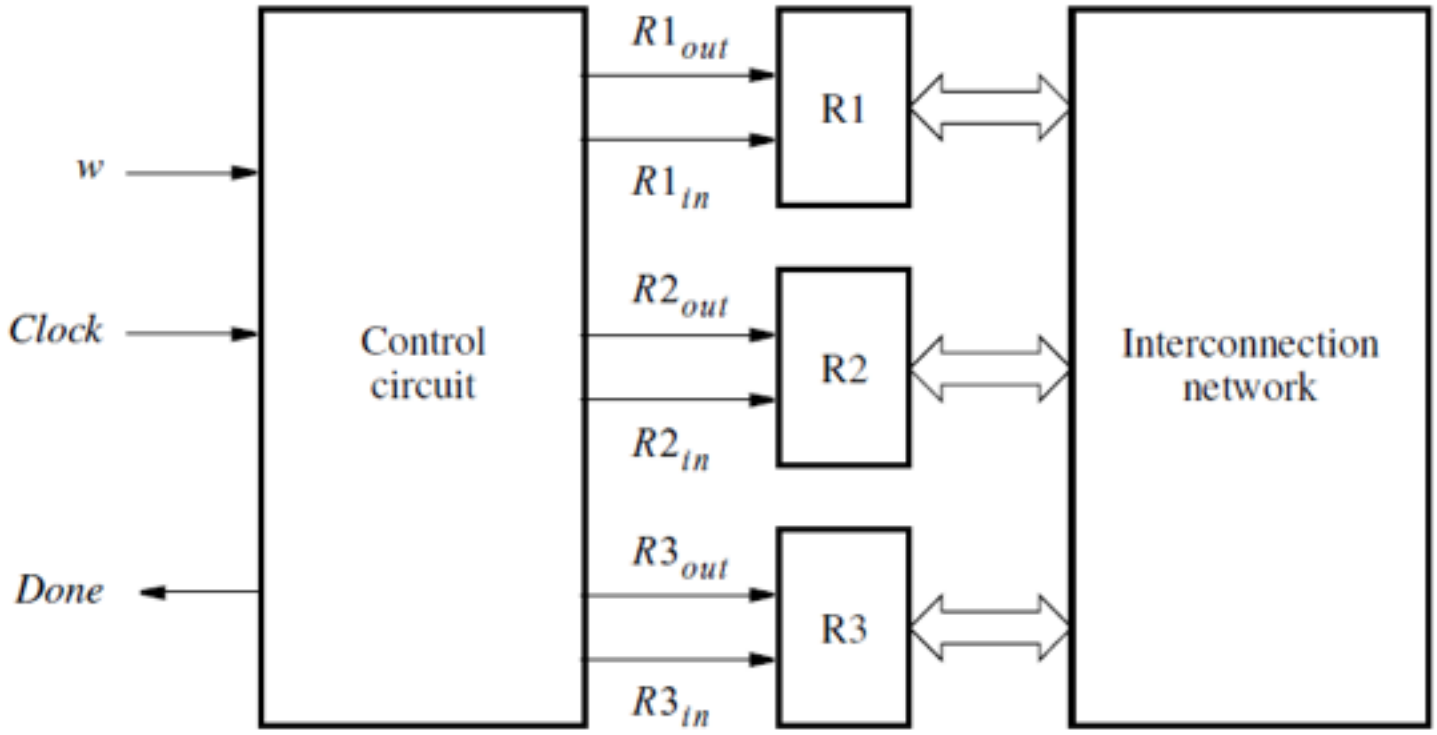


Figure 6.10. System for Example 6.1.

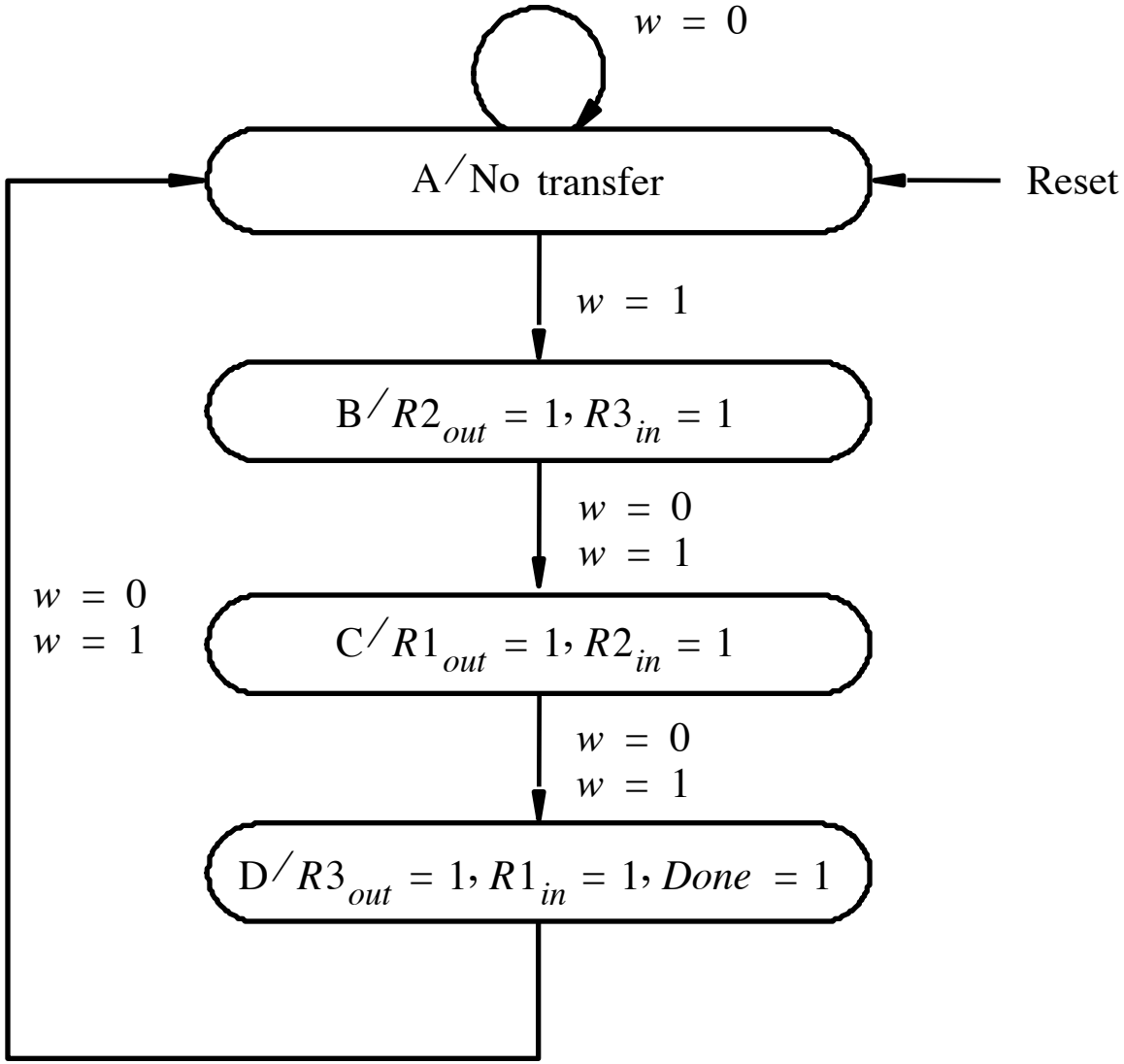


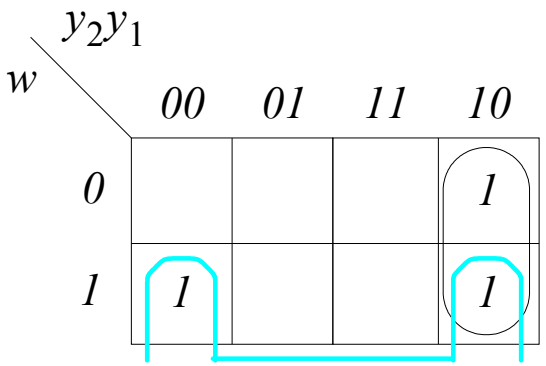
Figure 6.11. State diagram for Example 6.1.

Present state	Next state		Outputs						
	$w = 0$	$w = 1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	A	B	0	0	0	0	0	0	0
B	C	C	0	0	1	0	0	1	0
C	D	D	1	0	0	1	0	0	0
D	A	A	0	1	0	0	1	0	1

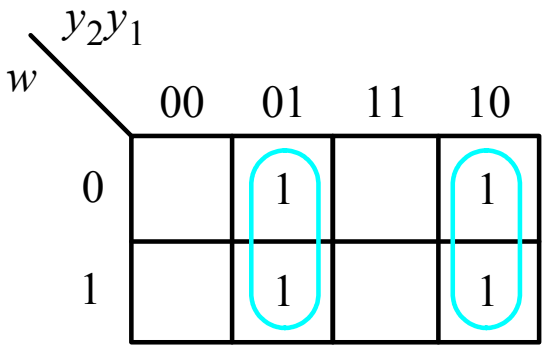
Figure 6.12. State table for Example 6.1.

	Present state	Next state		Outputs						
		$w = 0$	$w = 1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	0 1	0	0	0	0	0	0	0
B	01	10	1 0	0	0	1	0	0	1	0
C	10	11	1 1	1	0	0	1	0	0	0
D	11	00	0 0	0	1	0	0	1	0	1

Figure 6.13. State-assigned table for the sequential circuit in Figure 6.12.



$$Y_1 = w\bar{y}_1 + \bar{y}_1 y_2$$



$$Y_2 = y_1\bar{y}_2 + \bar{y}_1 y_2$$

Figure 6.14. Derivation of next-state expressions for the sequential circuit in Figure 6.13.

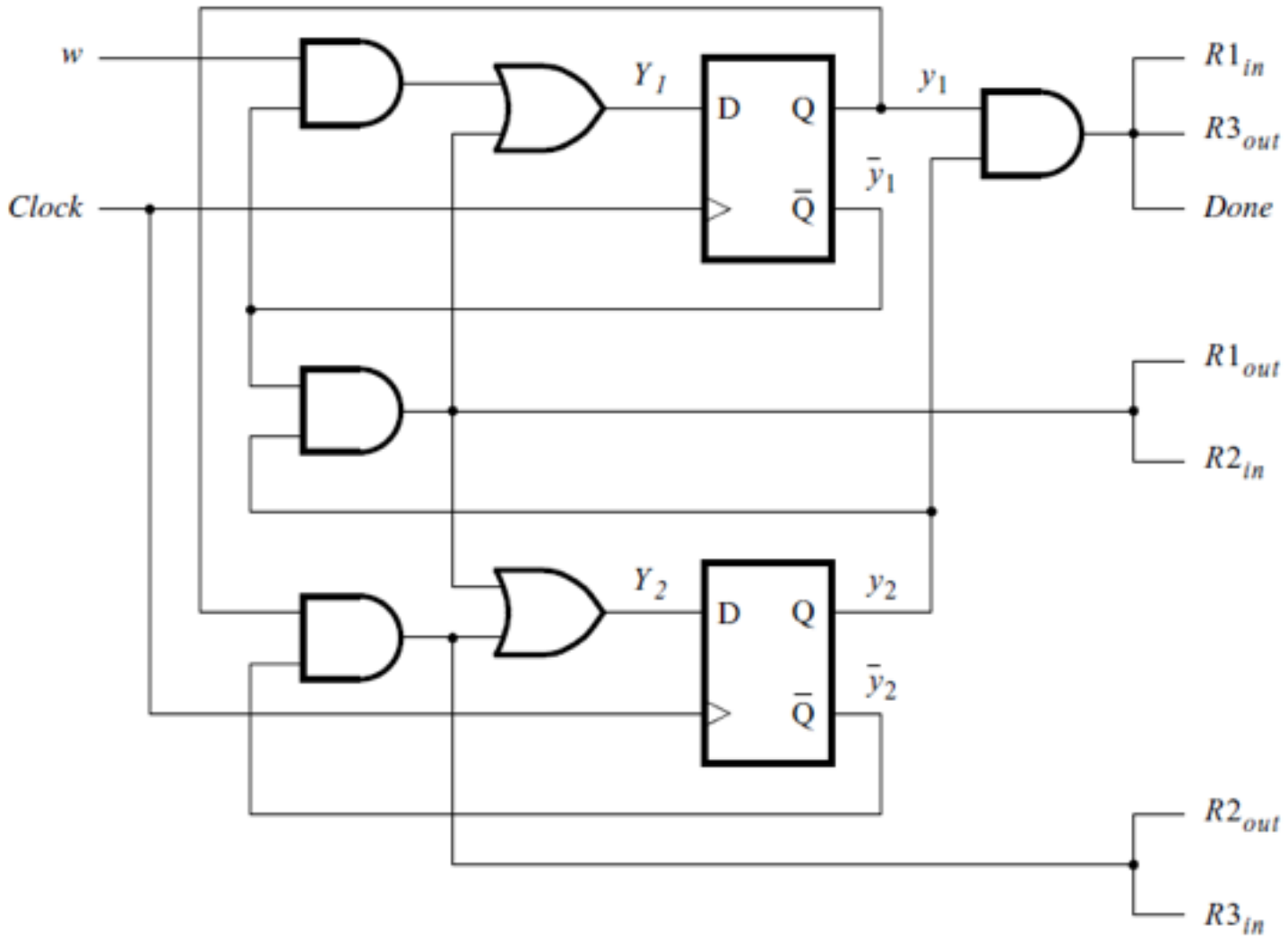


Figure 6.15. Final implementation of sequential circuit in Figure 6.13.

	Present state	Next state		Output
		$w = 0$	$w = 1$	
	y_2y_1	Y_2Y_1	Y_2Y_1	z
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	*dd*	*dd*	*d*

Figure 6.16. Improved state assignment for the sequential circuit in Figure 6.4.

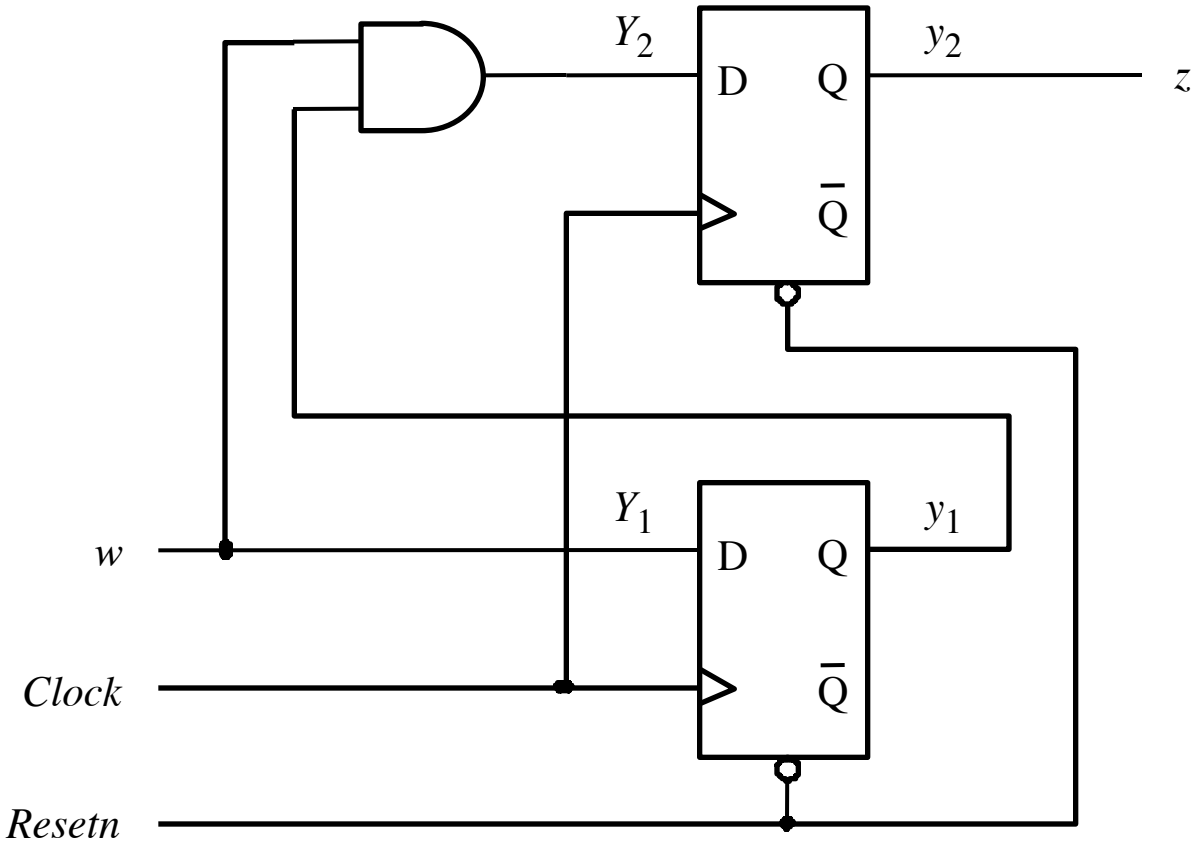
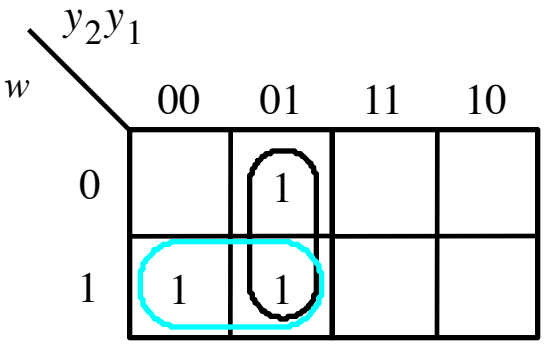


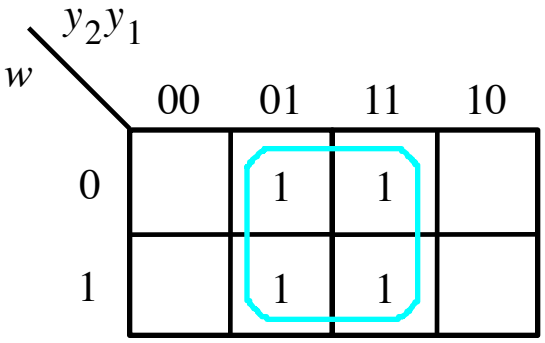
Figure 6.17. Final circuit for the improved state assignment in Figure 6.16.

	Present state	Nextstate		Outputs						
		$w = 0$	$w = 1$							
	$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	0 0	01	0	0	0	0	0	0	0
B	01	1 1	11	0	0	1	0	0	1	0
C	11	1 0	10	1	0	0	1	0	0	0
D	10	0 0	00	0	1	0	0	1	0	1

Figure 6.18. Improved state assignment for the sequential circuit in Figure 6.12.



$$Y_1 = w\bar{y}_2 + y_1\bar{y}_2$$



$$Y_2 = y_1$$

Figure 6.19. Derivation of next-state expressions for the sequential circuit in Figure 6.18.

	Present state	Nextstate		Output
		$w = 0$	$w = 1$	
	$y_3 y_2 y_1$	$Y_3 Y_2 Y_1$	$Y_3 Y_2 Y_1$	z
A	001	001	010	0
B	010	001	100	0
C	100	001	100	1

Figure 6.20. One-hot state assignment for the sequential circuit in Figure 6.4.

	Present state	Nextstate		Outputs						
		$w = 0$	$w = 1$							
	$y_4y_3y_2y_1$	$Y_4Y_3Y_2Y_1$	$Y_4Y_3Y_2Y_1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	0 001	0001	0010	0	0	0	0	0	0	0
B	0 010	0100	0100	0	0	1	0	0	1	0
C	0 100	1000	1000	1	0	0	1	0	0	0
D	1 000	0001	0001	0	1	0	0	1	0	1

Figure 6.21. One-hot state assignment for the sequential circuit in Figure 6.12.

Clock cycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w:	0	1	0	1	1	0	1	1	1	0	1
z:	0	0	0	0	1	0	0	1	1	0	0

Figure 6.22. Sequences of input and output signals.

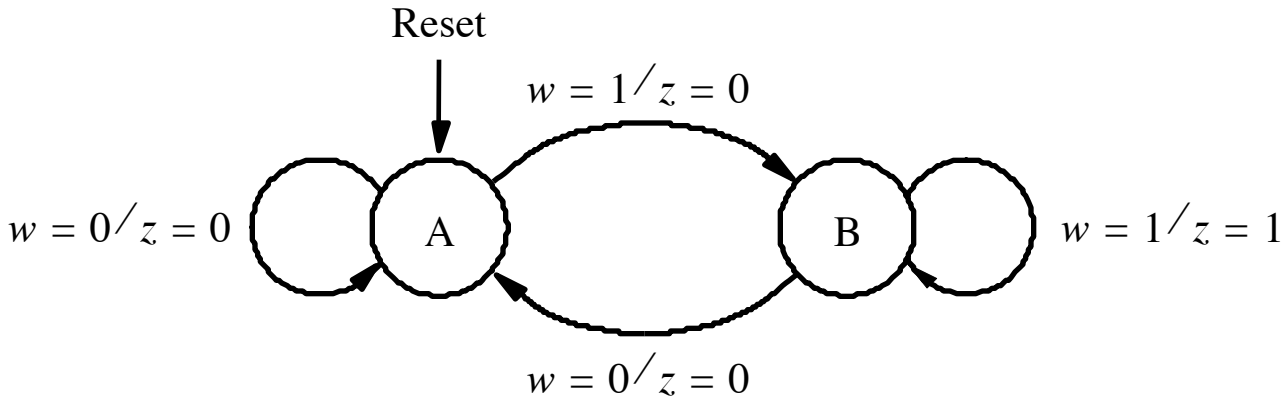


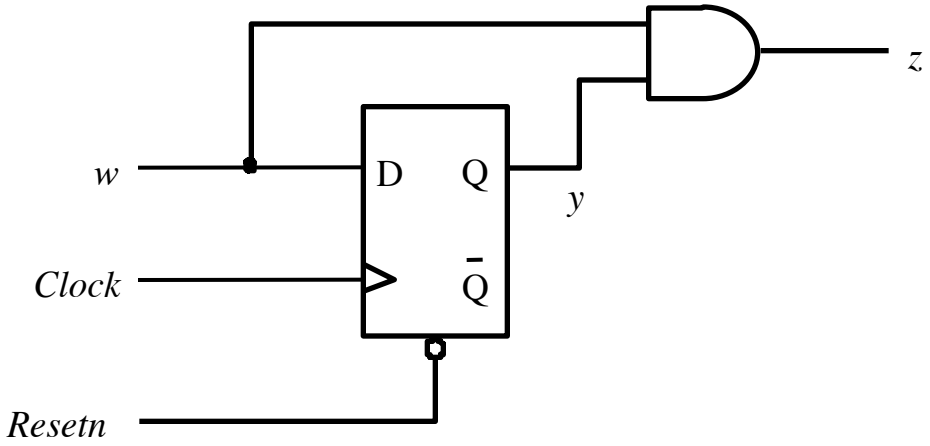
Figure 6.23. State diagram of an FSM that realizes the task in Figure 6.22.

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

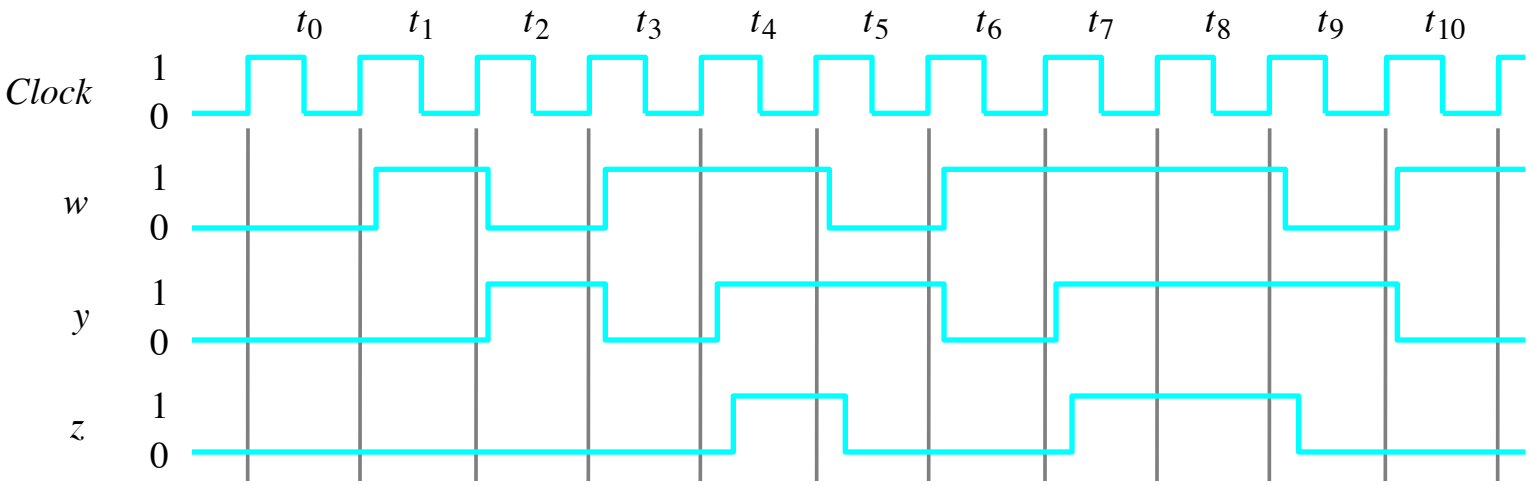
Figure 6.24. State table for the FSM in Figure 6.23.

	Present state	Next state		Output	
		$w = 0$	$w = 1$	$w = 0$	$w = 1$
	y	Y	Y	z	z
A	0	0	1	0	0
B	1	0	1	0	1

Figure 6.25. State-assigned table for the FSM in Figure 6.24.



(a) Circuit



(b) Timing diagram

Figure 6.26. Implementation of FSM in Figure 6.25.

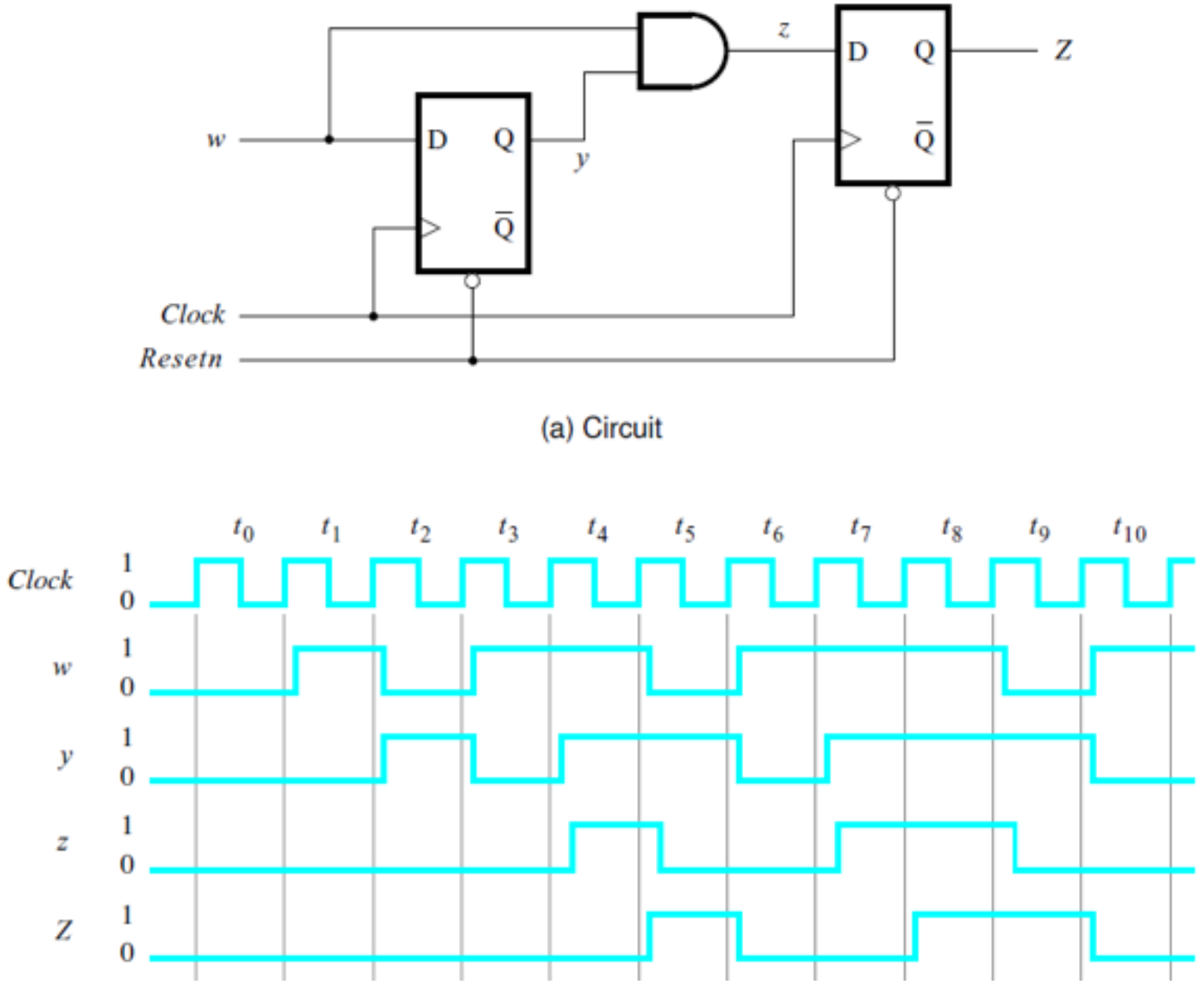


Figure 6.27. Circuit that implements the specification in Figure 6.2.

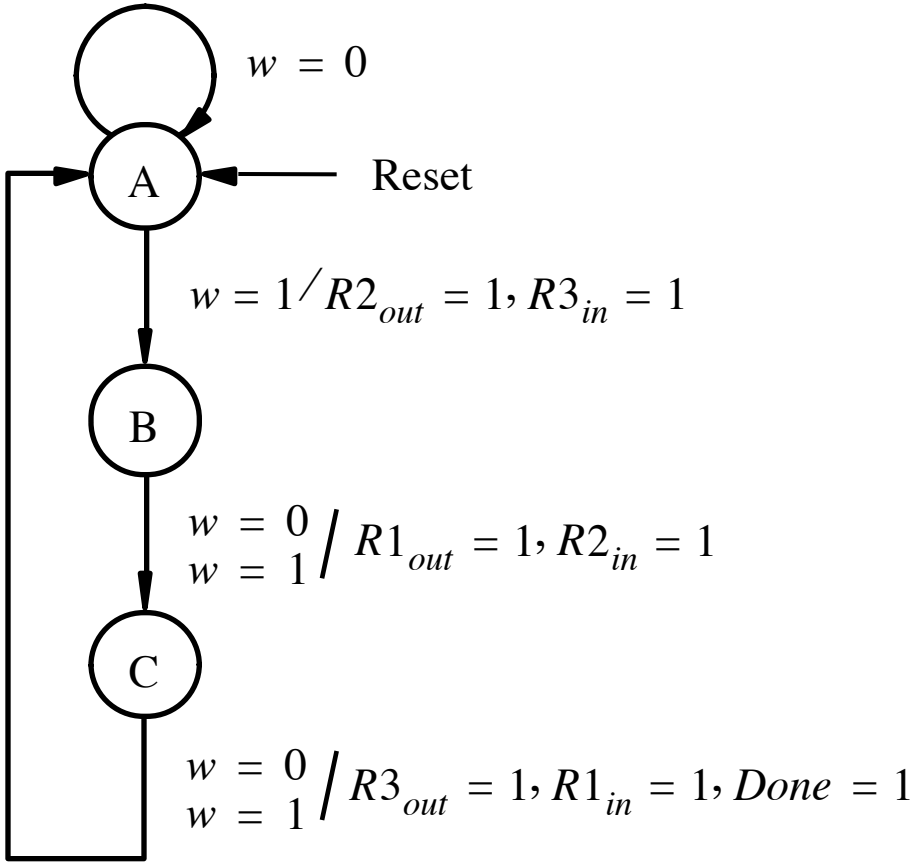


Figure 6.28. State diagram for Example 6.4.

```
module simple (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;

    // Define the next state combinational circuit
    always @(w, y)
        case (y)
            A: if (w)   Y = B;
               else     Y = A;
            B: if (w)   Y = C;
               else     Y = A;
            C: if (w)   Y = C;
               else     Y = A;
            default:    Y = 2'bxx;
        endcase

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

    // Define output
    assign z = (y = = C);

endmodule
```

Figure 6.29. Verilog code for the FSM in Figure 6.3.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.30. Implementation of the FSM of Figure 6.3 in a CPLD.

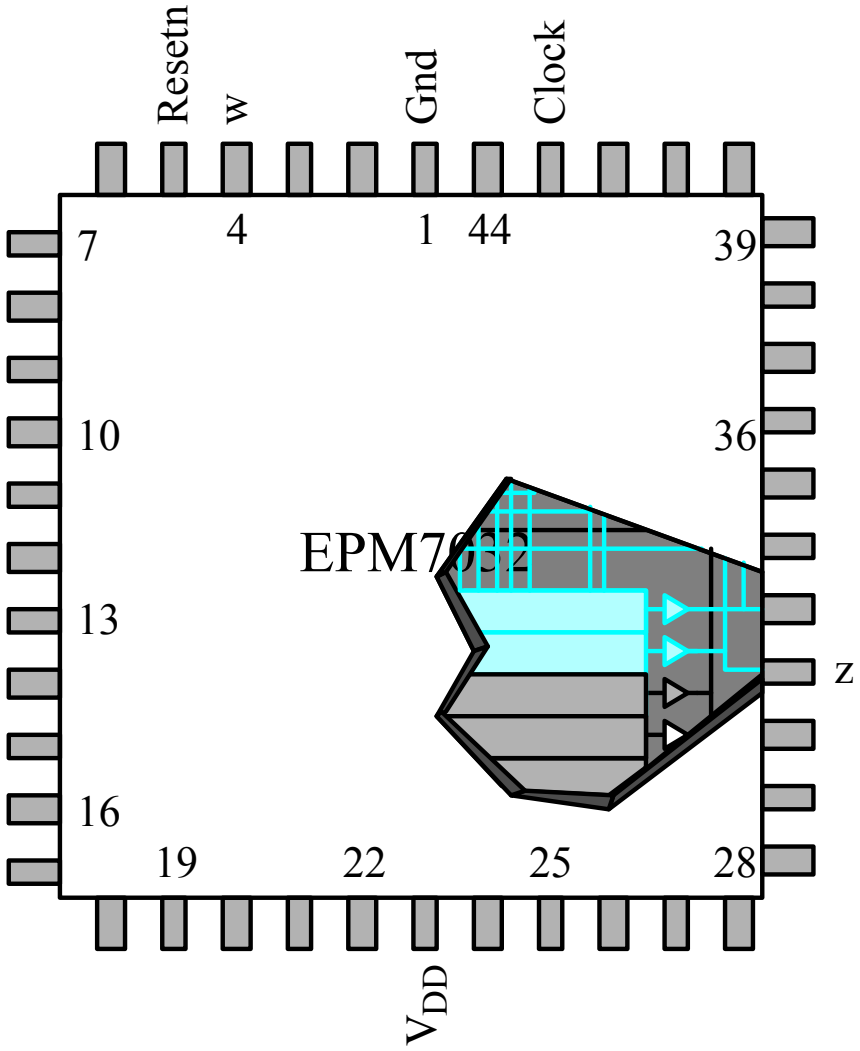


Figure 6.31. The circuit from Figure 6.30 in a small CPLD.

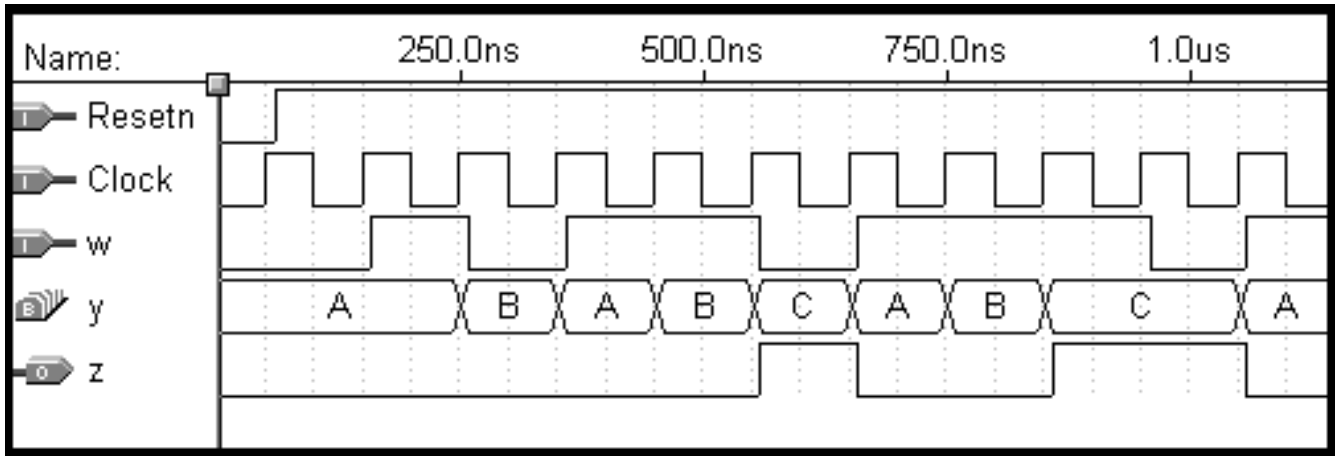


Figure 6.32. Simulation results for the circuit in Figure 6.30.

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

    // Define the next state combinational circuit
    always @(w, y)
    begin
        case (y)
            A: if (w)   Y = B;
               else     Y = A;
            B: if (w)   Y = C;
               else     Y = A;
            C: if (w)   Y = C;
               else     Y = A;
            default:    Y = 2'bxx;
        endcase
        z = (y = = C); //Define output
    end

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

endmodule
```

Figure 6.33. Second version of code for the FSM in Figure 6.3.

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

    // Define the sequential block
    always @(negedge Resetn, posedge Clock)
        if (Resetn = = 0)   y <= A;
        else
            case (y)
                A: if (w)   y <= B;
                   else     y <= A;
                B: if (w)   y <= C;
                   else     y <= A;
                C: if (w)   y <= C;
                   else     y <= A;
                default:    y <= 2'bxx;
            endcase

    // Define output
    assign z = (y = = C);

endmodule
```

Figure 6.34. Third version of code for the FSM in Figure 6.3.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.35. Verilog code for the FSM in Figure 6.11.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.36. Verilog code for the Mealy machine of Figure 6.23.

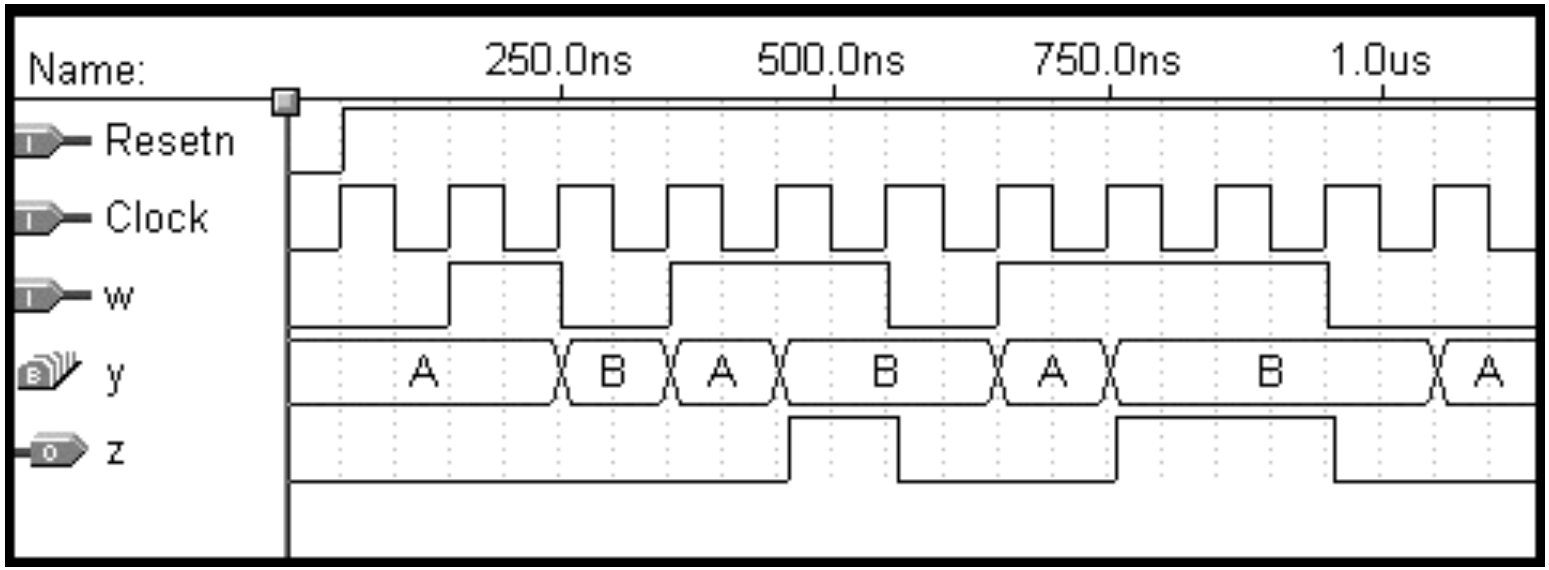


Figure 6.37. Simulation results for the Mealy machine.

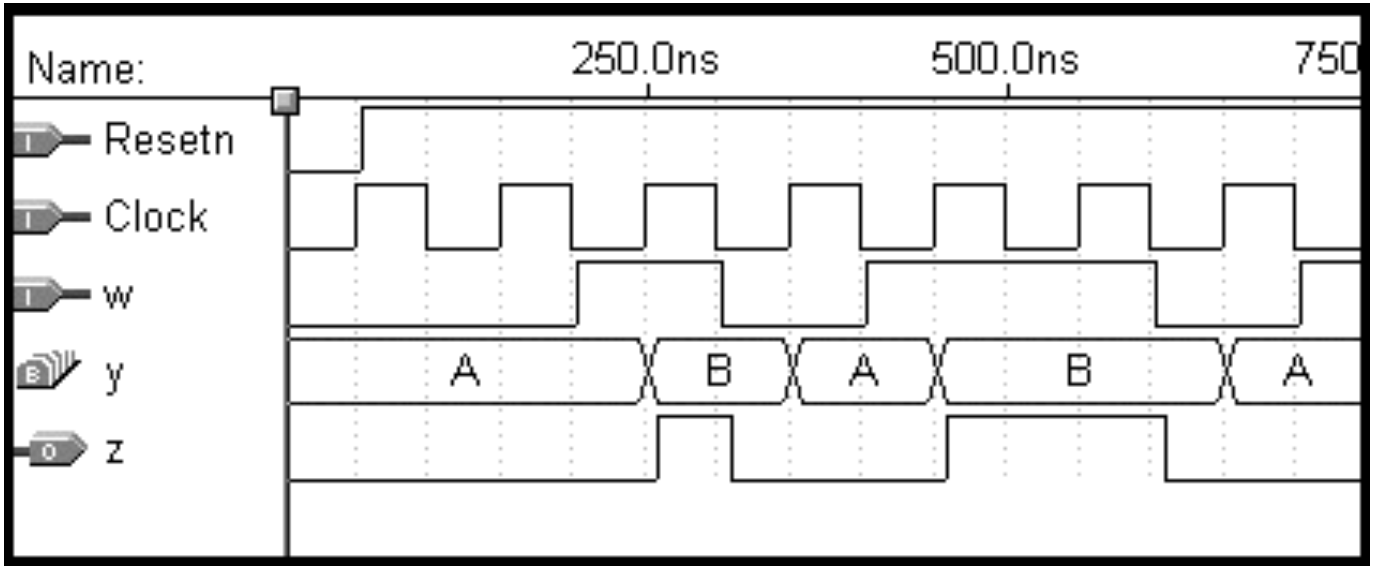


Figure 6.38. Potential problem with asynchronous inputs to a Mealy FSM.

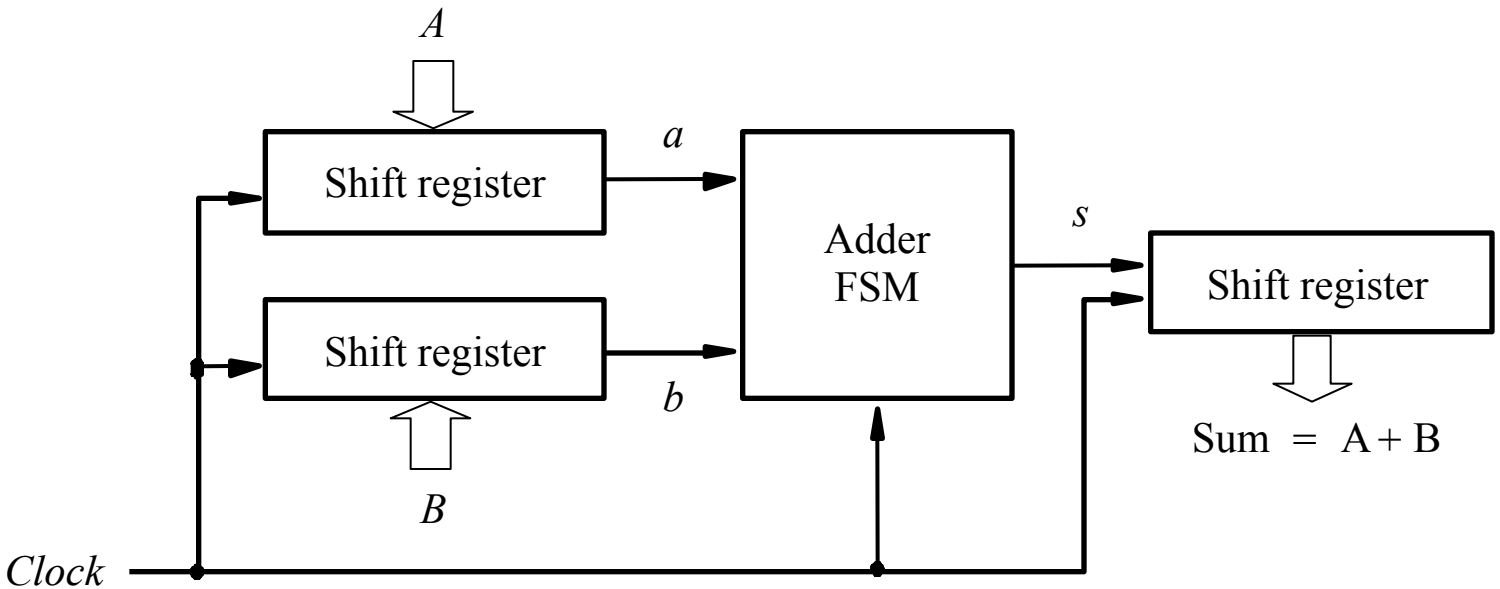


Figure 6.39. Block diagram for the serial adder.

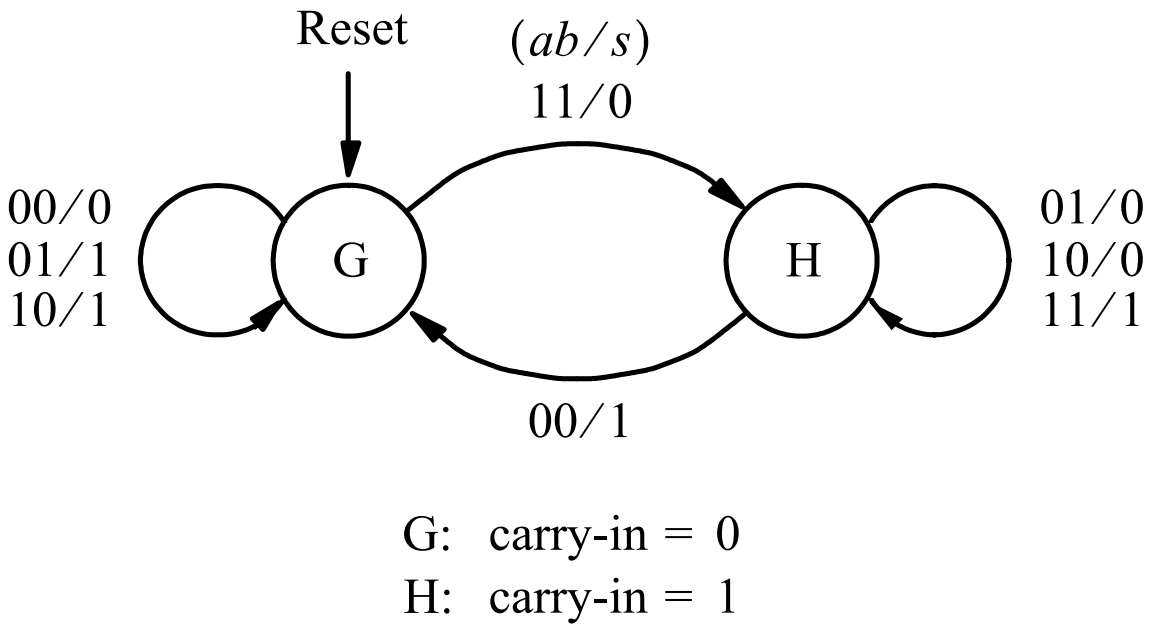


Figure 6.40. State diagram for the serial adder FSM.

Present state	Next state				Output s			
	ab =00	01	10	11	00	01	10	11
G	G	G	G	H	0	1	1	0
H	G	H	H	H	1	0	0	1

Figure 6.41. State table for the serial adder FSM.

Present state	Next state				Output			
	ab =00	01	10	11	00	01	10	11
y	*Y*				*s*			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Figure 6.42. State-assigned table for Figure 6.41.

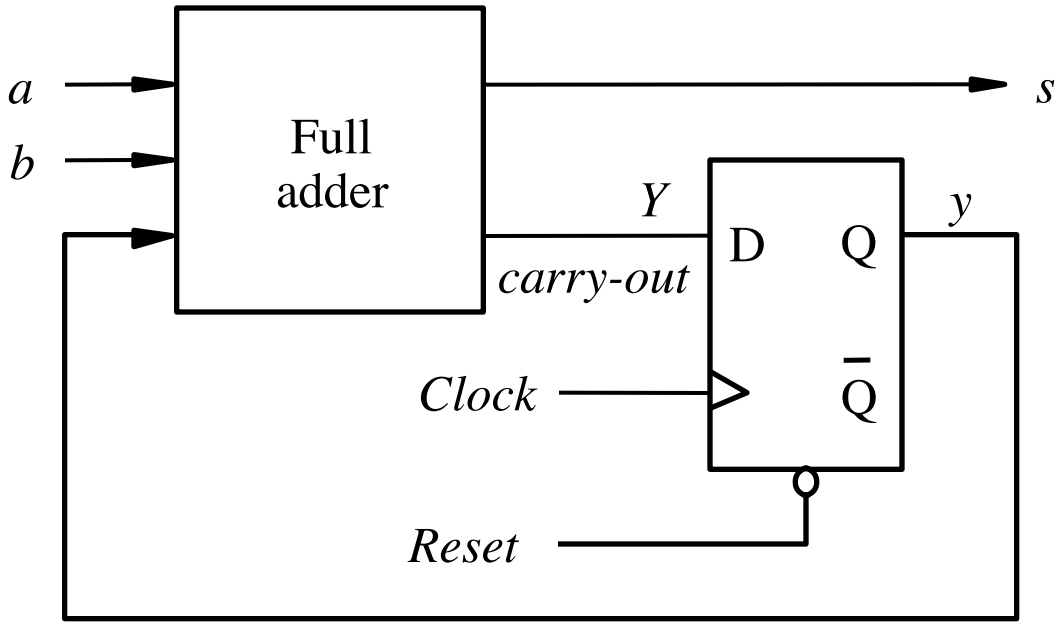


Figure 6.43. Circuit for the adder FSM in Figure 6.39.

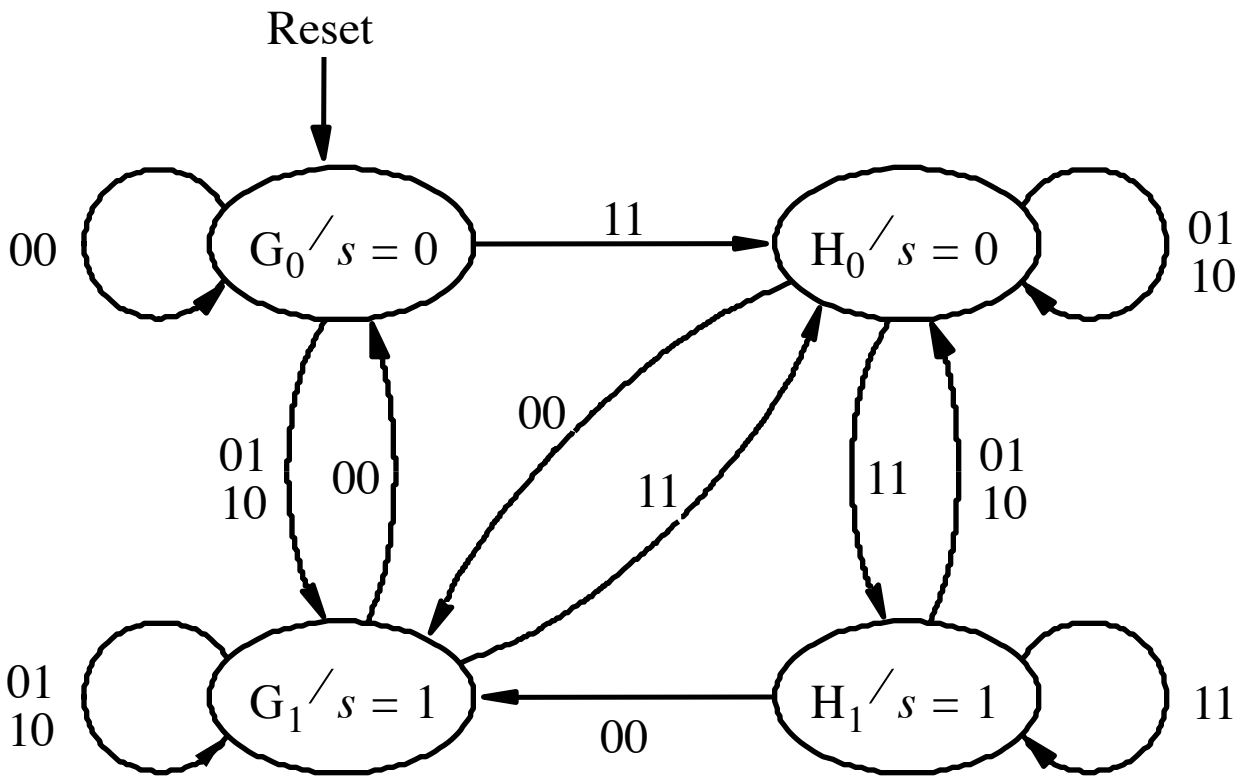


Figure 6.44. State diagram for the Moore-type serial adder FSM.

Present state	Nextstate				Output
	ab =00	01	10	11	s
G_0	G_0	G_1	G_1	H_0	0
G_1	G_0	G_1	G_1	H_0	1
H_0	G_1	H_0	H_0	H_1	0
H_1	G_1	H_0	H_0	H_1	1

Figure 6.45. State table for the Moore-type serial adder FSM.

Present state y_2y_1	Nextstate				Output s
	ab =00	01	10	11	
	Y_2Y_1				
00	0 0	01	0 1	10	0
01	0 0	01	0 1	10	1
10	0 1	10	1 0	11	0
11	0 1	10	1 0	11	1

Figure 6.46. State-assigned table for Figure 6.45.

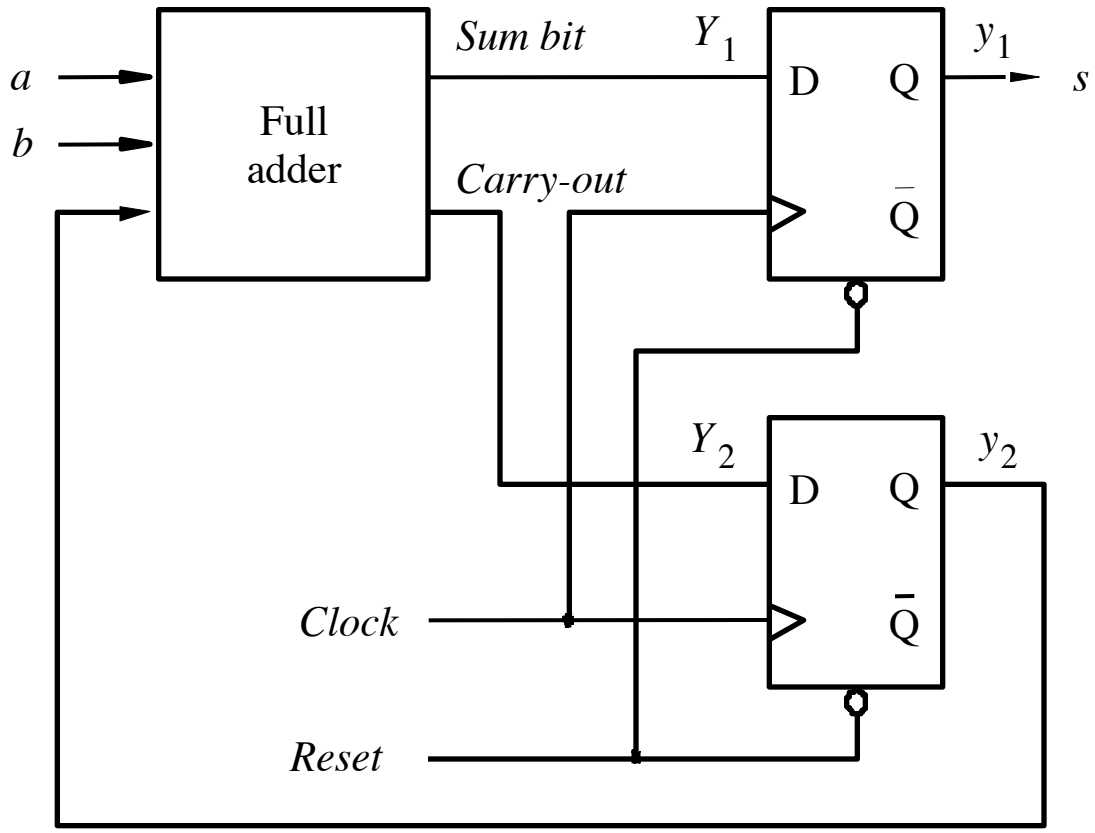


Figure 6.47. Circuit for the Moore-type serial adder FSM.

```
module shiftrne (R, L, E, w, Clock, Q);
    parameter n = 8;
    input [n-1:0] R;
    input L, E, w, Clock;
    output reg [n-1:0] Q;
    integer k;

    always @(posedge Clock)
        if (L)
            Q <= R;
        else if (E)
        begin
            for (k = n-1; k > 0; k = k-1)
                Q[k-1] <= Q[k];
            Q[n-1] <= w;
        end

endmodule
```

Figure 6.48. Code for a left-to-right shift register with an enable input.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.49. Verilog code for the serial adder.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.50. Synthesized serial adder.

Equivalence of states

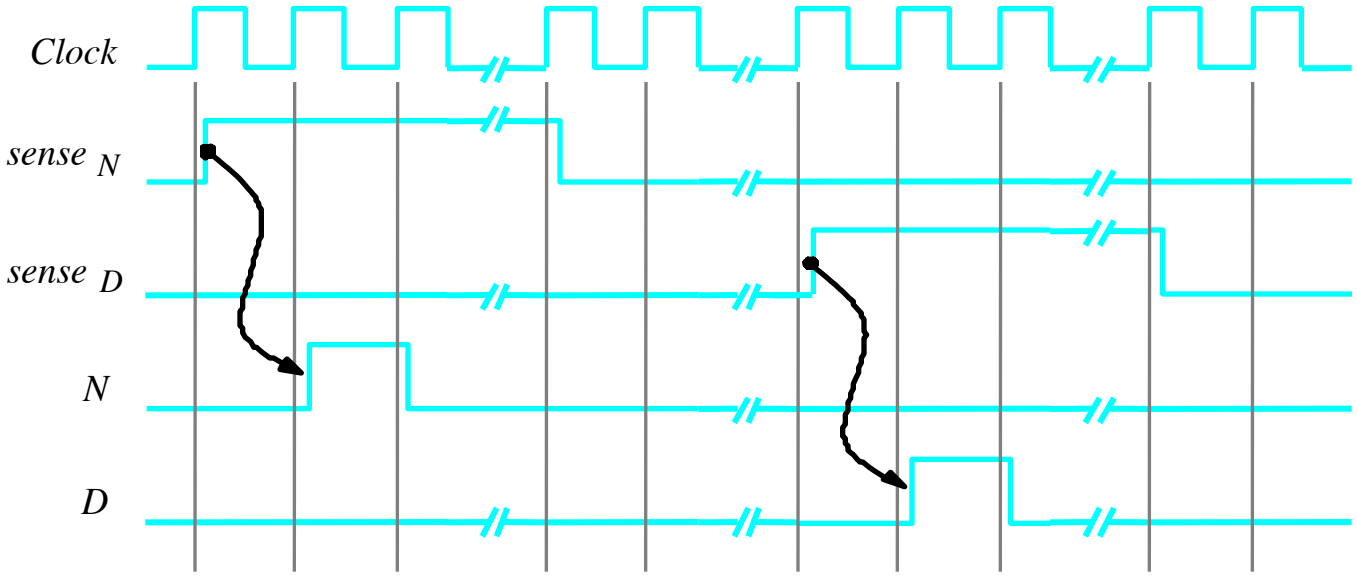
Two states S_i and S_j are said to be equivalent if and only if for every possible input sequence, the same output sequence will be produced regardless of whether S_i or S_j is the initial state.

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

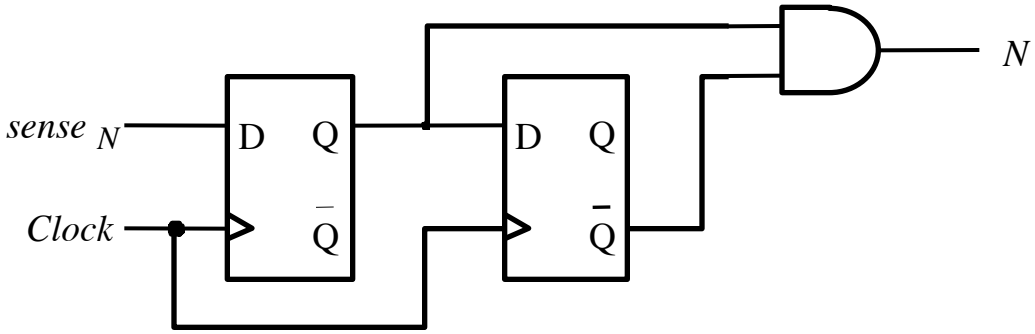
Figure 6.51. State table for Example 6.6.

Present state	Nextstate		Output z
	w = 0	w = 1	
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

Figure 6.52. Minimized state table for Example 6.6.



(a) Timing diagram



(b) Circuit that generates *N*

Figure 6.53. Signals for the vending machine.

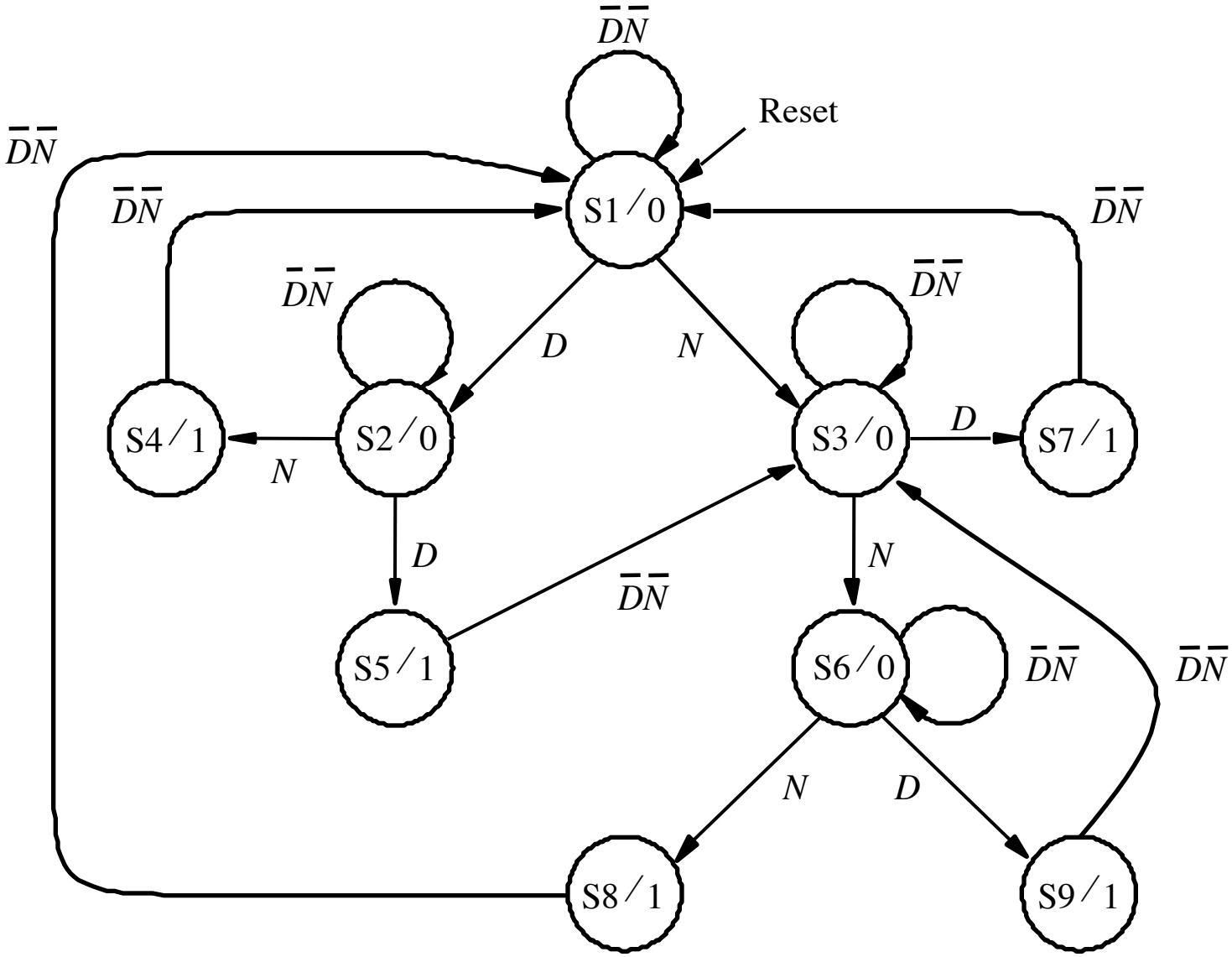


Figure 6.54. State diagram for Example 6.7.

Present state	Next state				Output z
	DN =00	01	10	11	
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S6	S7	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1
S6	S6	S8	S9	–	0
S7	S1	–	–	–	1
S8	S1	–	–	–	1
S9	S3	–	–	–	1

Figure 6.55. State table for Example 6.7.

Present state	Next state				Output
	DN =00	01	10	11	*z*
S1	S1	S3	S2	–	0
S2	S2	S4	S5	–	0
S3	S3	S2	S4	–	0
S4	S1	–	–	–	1
S5	S3	–	–	–	1

Figure 6.56. Minimized state table for Example 6.7.

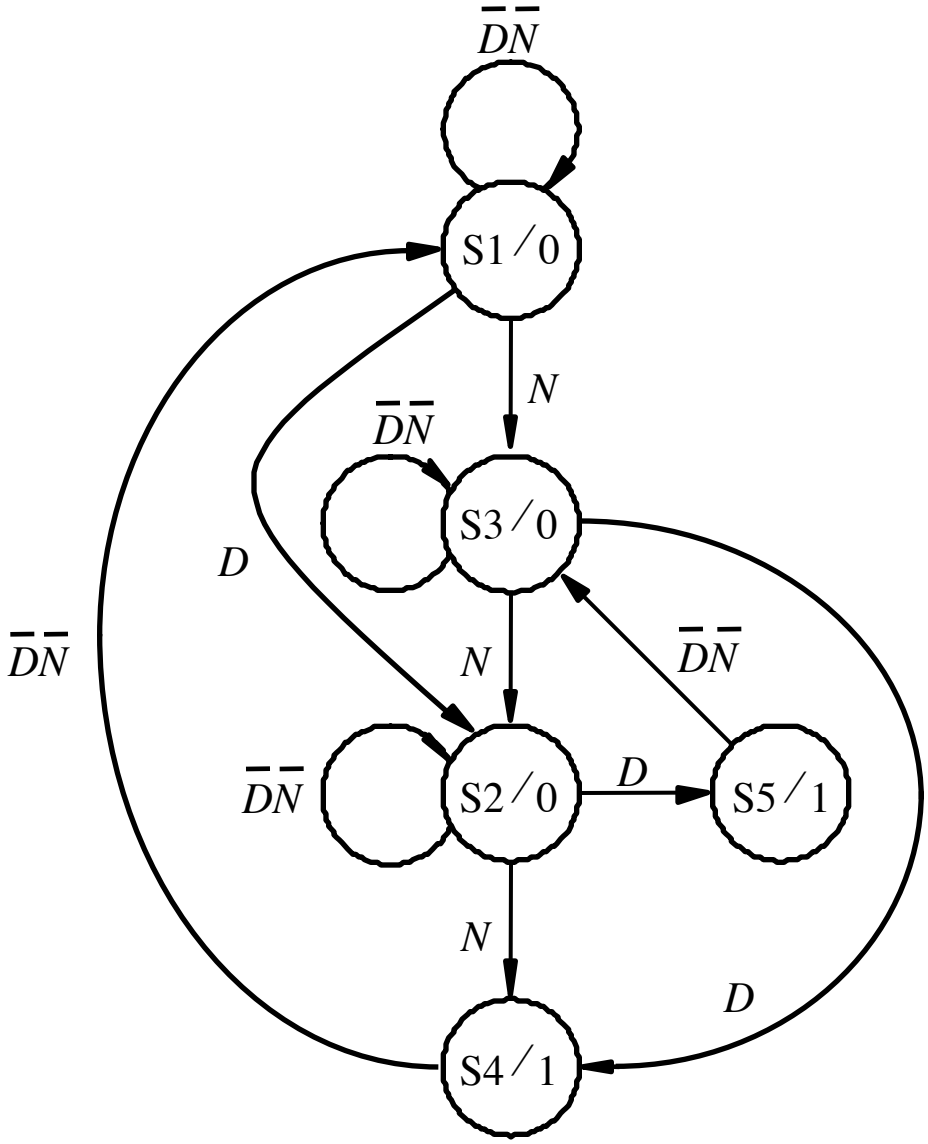


Figure 6.57. Minimized state diagram for Example 6.7.

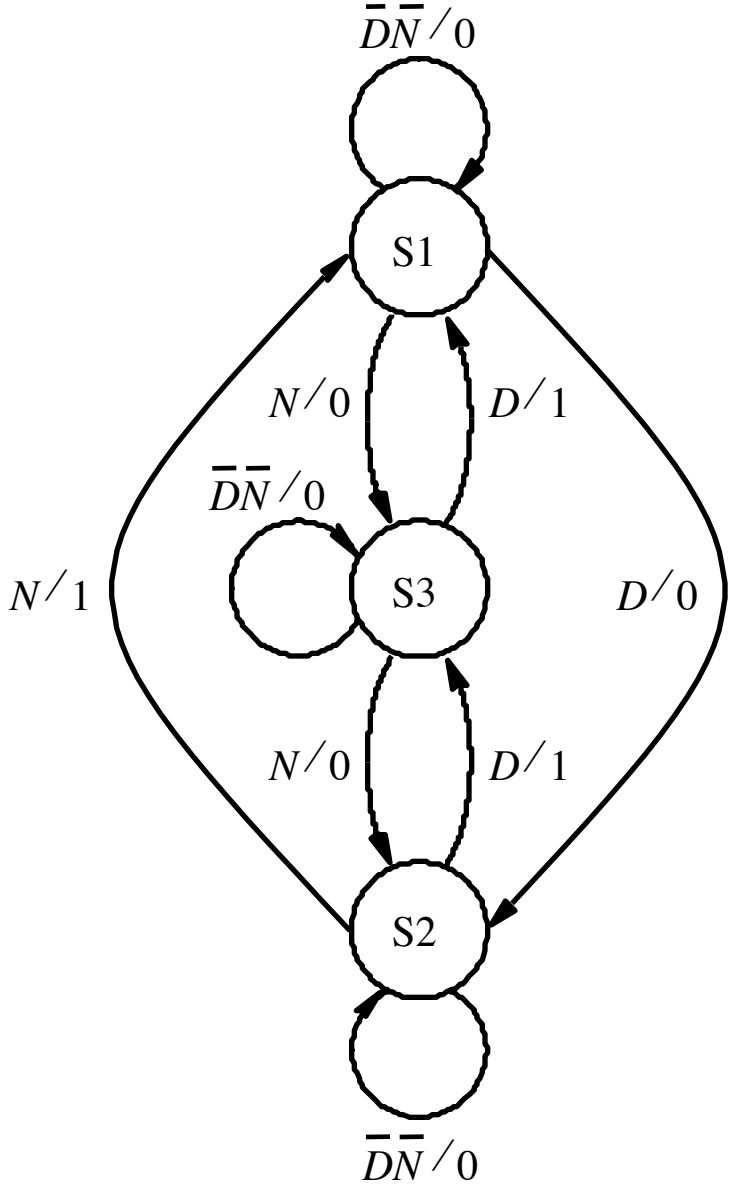


Figure 6.58. Mealy-type FSM for Example 6.7.

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

Figure 6.59. Incompletely specified state table for Example 6.8.

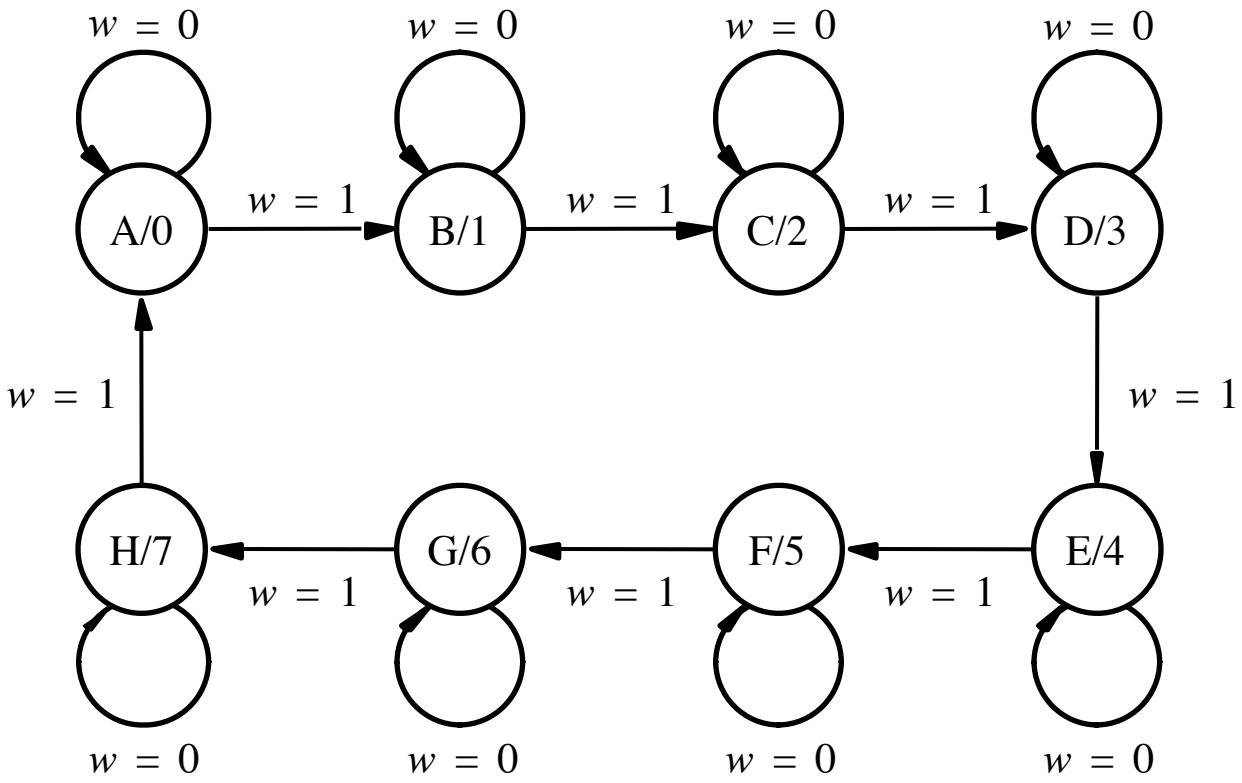


Figure 6.60. State diagram for the counter.

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

Figure 6.61. State table for the counter.

	Present state $y_2y_1y_0$	Next state		Count $z_2z_1z_0$
		$w = 0$	$w = 1$	
		$Y_2Y_1Y_0$	$Y_2Y_1Y_0$	
A	000	000	001	000
B	001	001	010	001
C	010	010	011	010
D	011	011	100	011
E	100	100	101	100
F	101	101	110	101
G	110	110	111	110
H	111	111	000	111

Figure 6.62. State-assigned table for the counter.

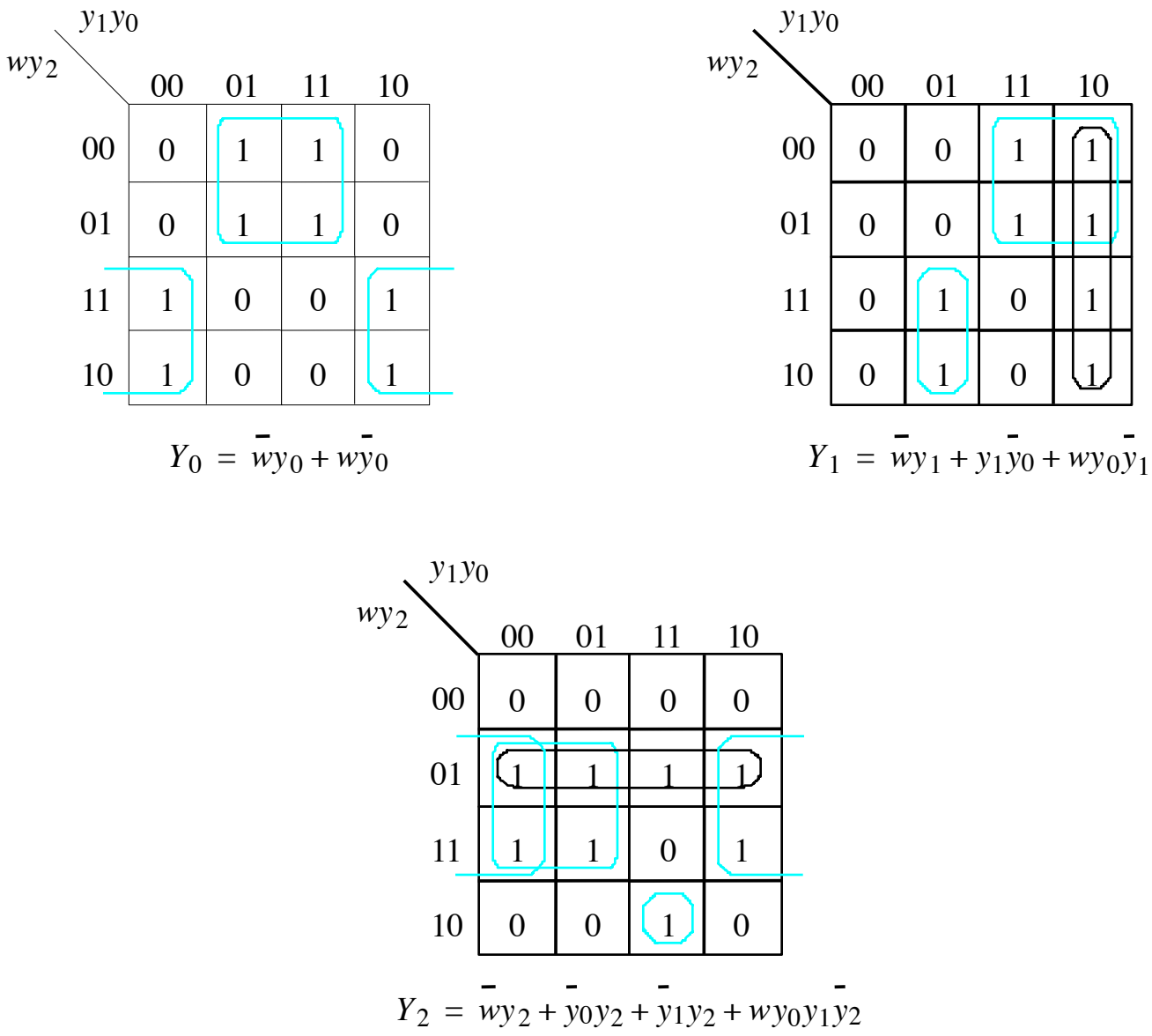


Figure 6.63. Karnaugh maps for D flip-flops for the counter.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.64. Circuit diagram for the counter implemented with D flip-flops.

	Present state $y_2y_1y_0$	Flip-flop inputs								Count $z_2z_1z_0$
		$w = 0$				$w = 1$				
		$Y_2Y_1Y_0$	J_2K_2	J_1K_1	J_0K_0	$Y_2Y_1Y_0$	J_2K_2	J_1K_1	J_0K_0	
A	000	000	0d	0d	0d	001	0d	0d	1d	000
B	001	001	0d	0d	d0	010	0d	1d	d1	001
C	010	010	0d	d0	0d	011	0d	d0	1d	010
D	011	011	0d	d0	d0	100	1d	d1	d1	011
E	100	100	d0	0d	0d	101	d0	0d	1d	100
F	101	101	d0	0d	d0	110	d0	1d	d1	101
G	110	110	d0	d0	0d	111	d0	d0	1d	110
H	111	111	d0	d0	d0	000	d1	d1	d1	111

Figure 6.65. Excitation table for the counter with JK flip-flops.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.66. Karnaugh maps for JK flip-flops in the counter.

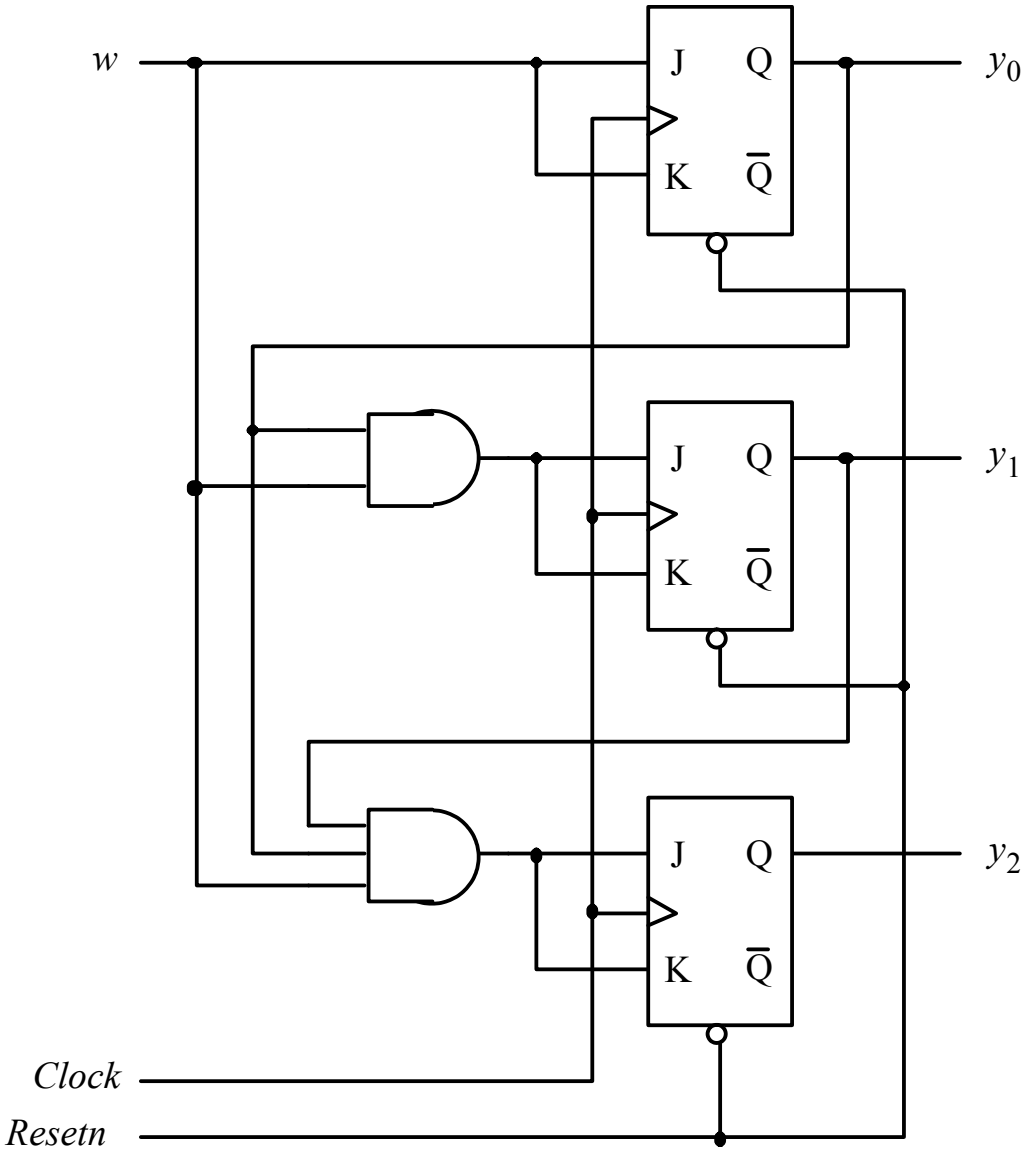


Figure 6.67. Circuit diagram using JK flip-flops.

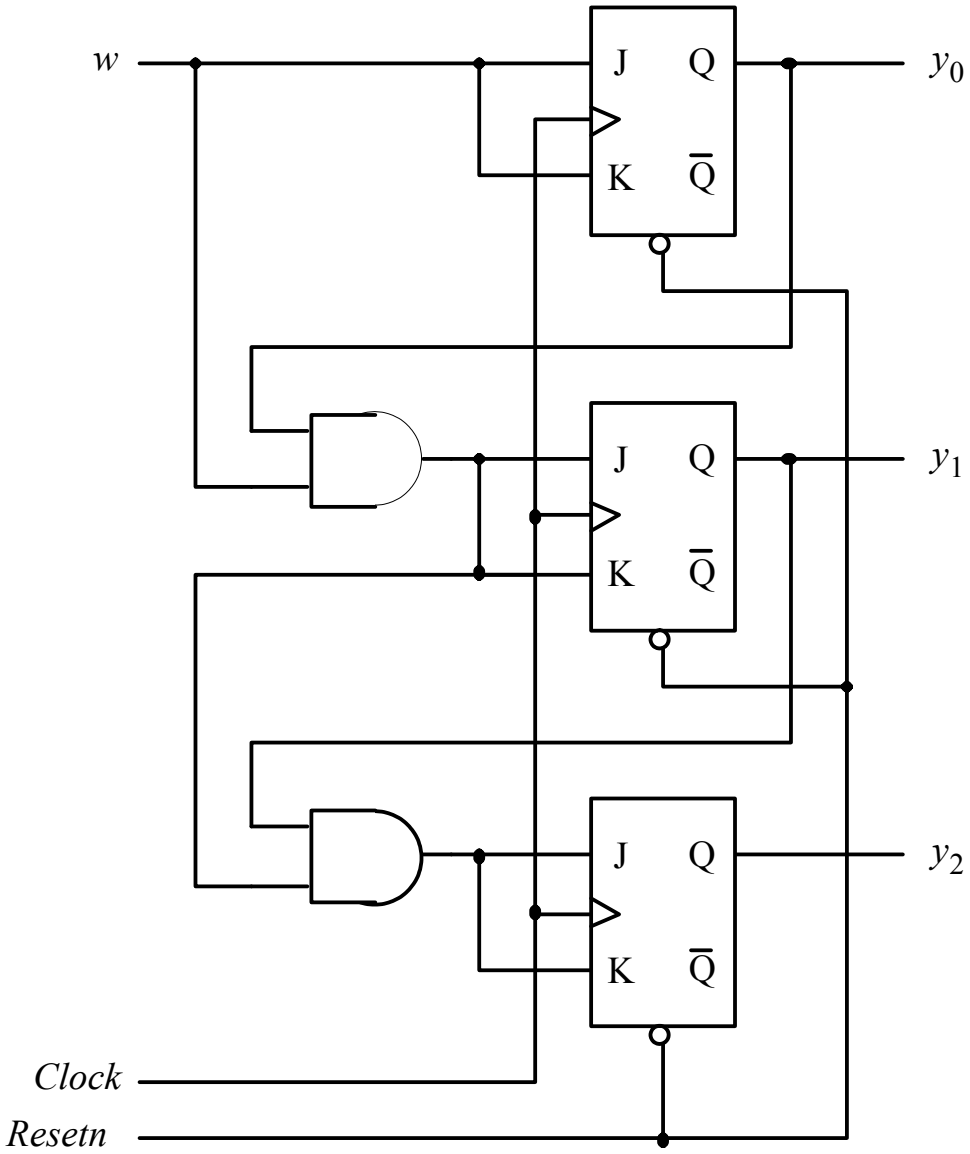


Figure 6.68. Factored-form implementation of the counter.

Present state	Next state	Output $z_2z_1z_0$
A	B	000
B	C	100
C	D	010
D	E	110
E	F	001
F	G	101
G	H	011
H	A	111

Figure 6.69. State table for the counterlike example.

Present state $y_2y_1y_0$	Next state $Y_2Y_1Y_0$	Output $z_2z_1z_0$
000	100	000
100	010	100
010	110	010
110	001	110
001	101	001
101	011	101
011	111	011
111	000	111

Figure 6.70. State-assigned table for Figure 6.69.

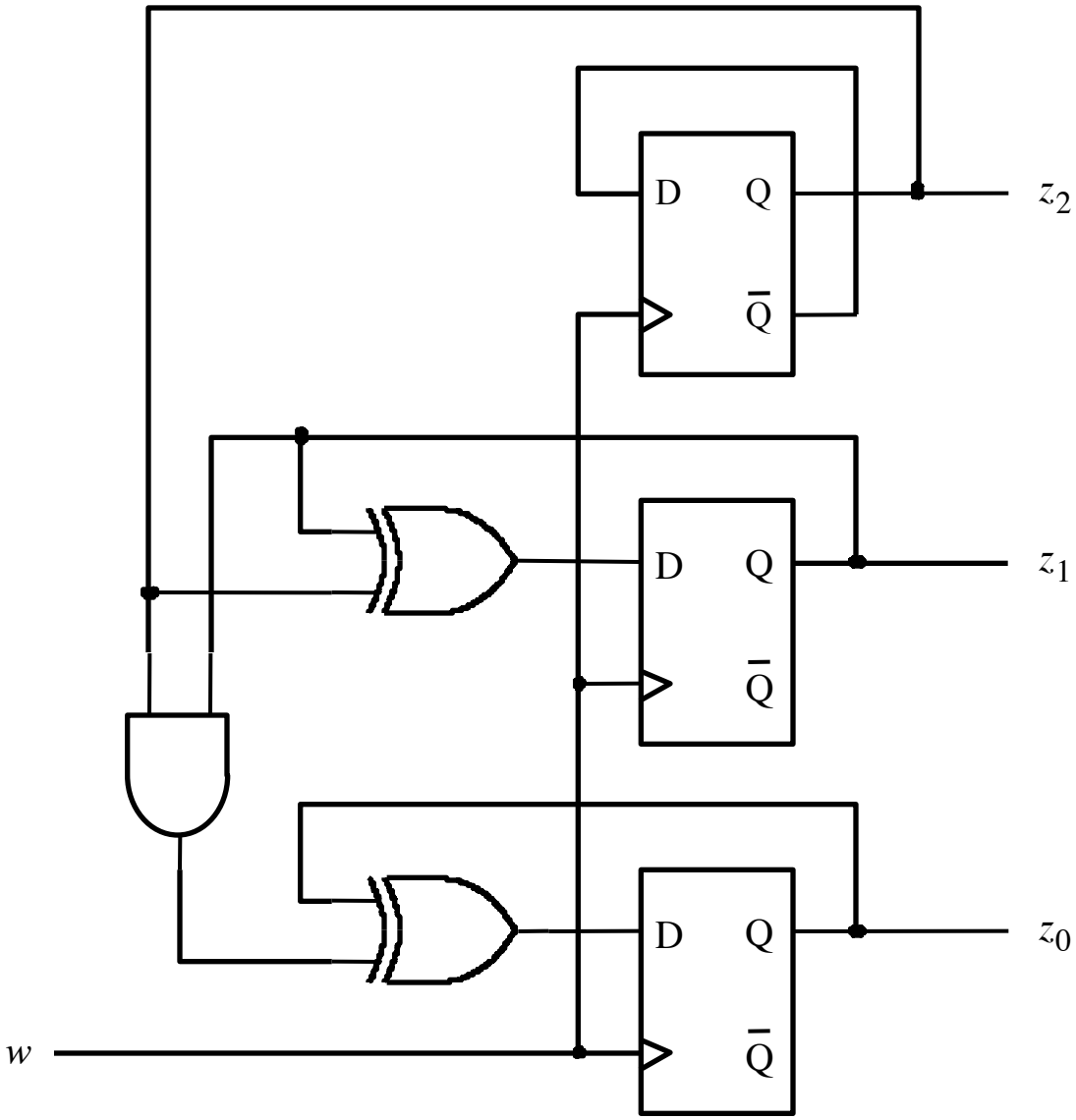


Figure 6.71. Circuit for Figure 6.70.

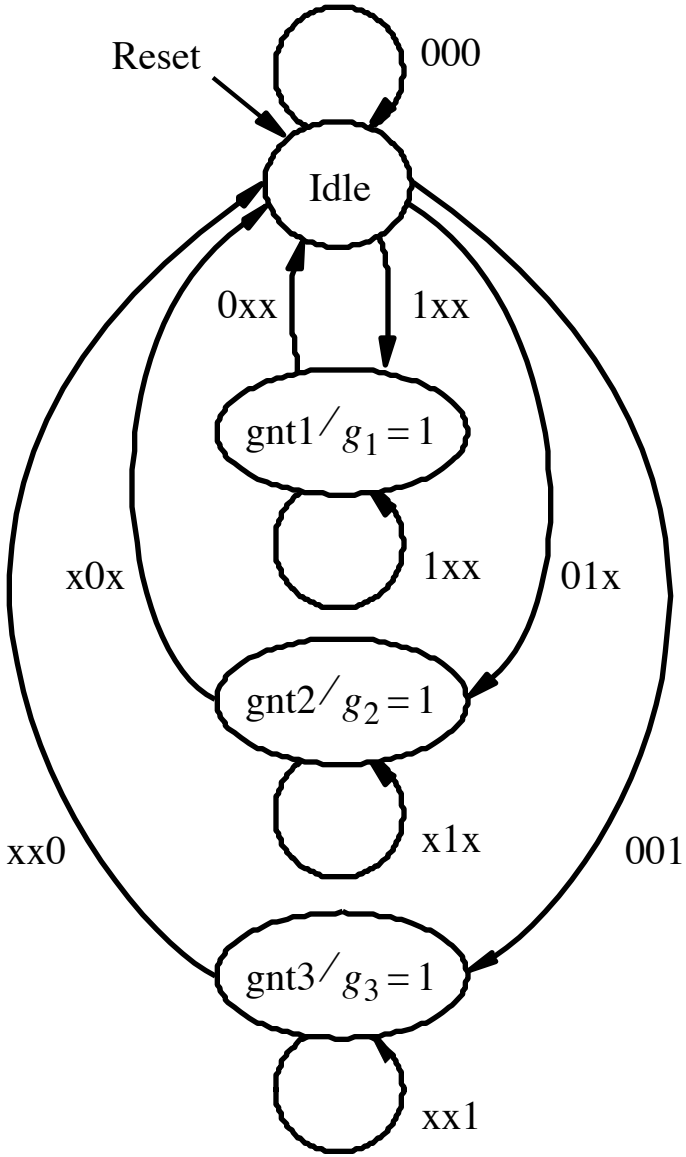


Figure 6.72. State diagram for the arbiter.

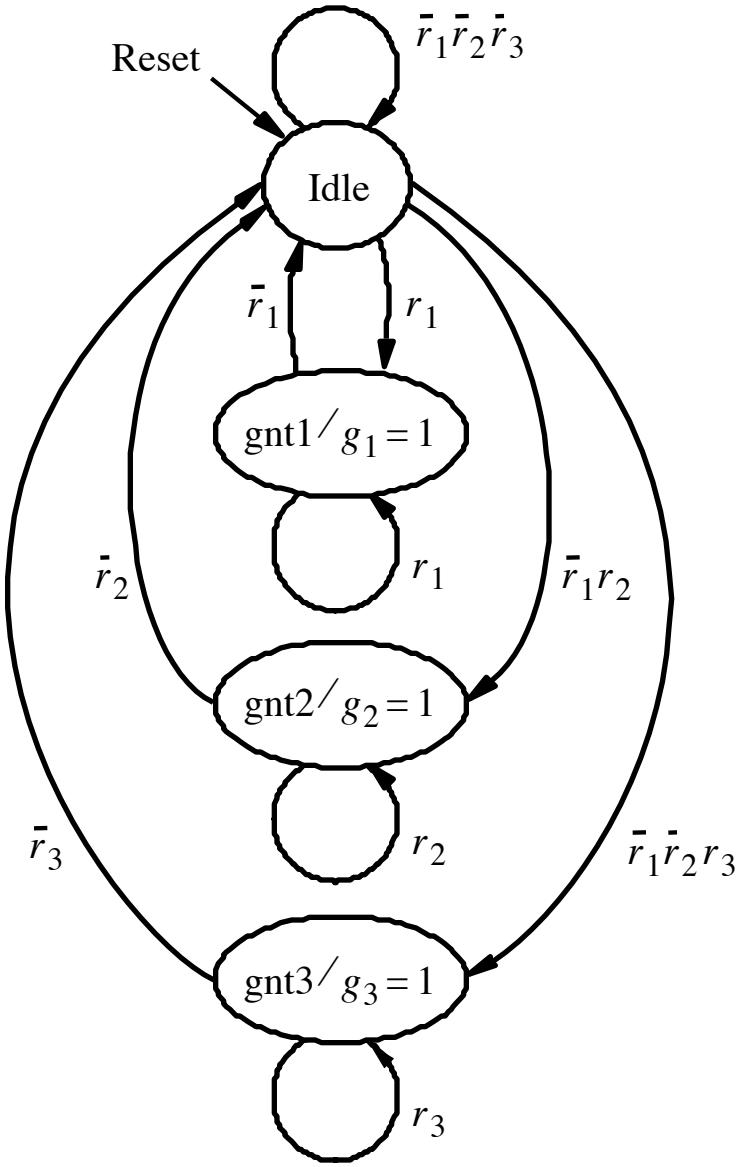


Figure 6.73. Alternative style of state diagram for the arbiter.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.74. Verilog code for the arbiter.

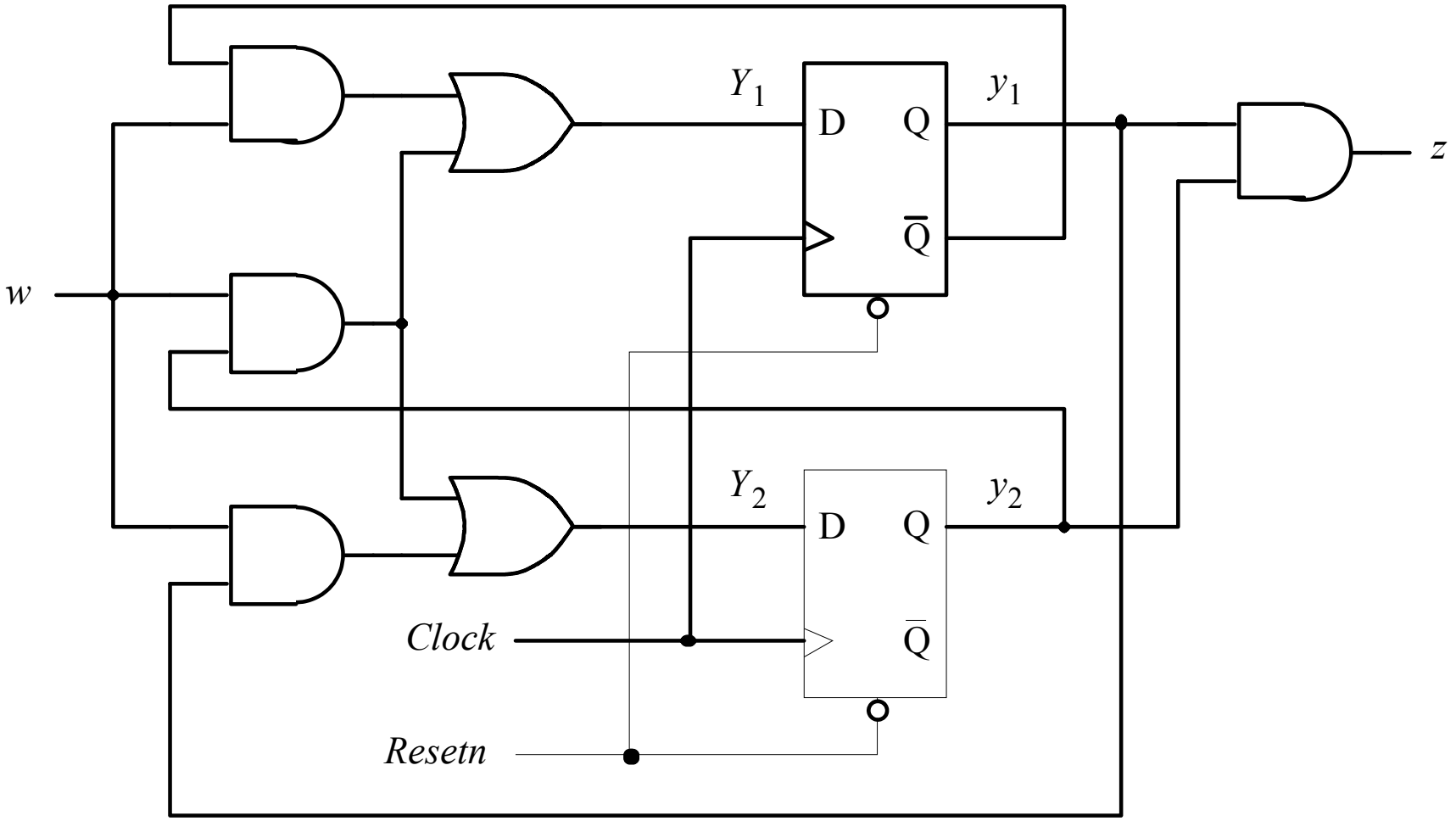


Figure 6.75. Circuit for Example 6.9.

Present state y_2y_1	Next State		Output z
	w = 0	w = 1	
	Y_2Y_1	Y_2Y_1	
0 0	0 0	01	0
0 1	0 0	10	0
1 0	0 0	11	0
1 1	0 0	11	1

(a) State-assigned table

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

(b) State table

Figure 6.76. Tables for the circuit in Example 6.75.

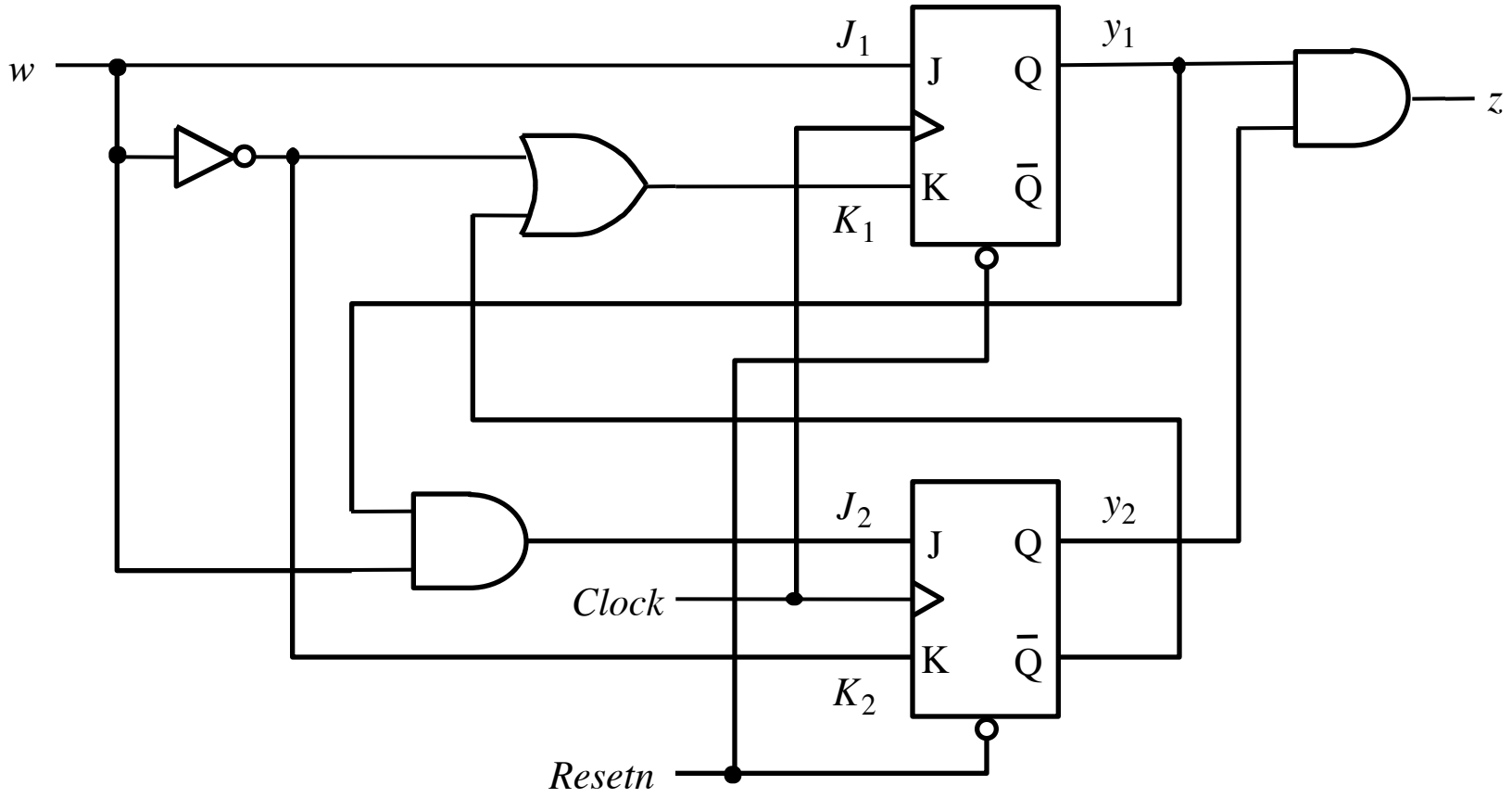


Figure 6.77. Circuit for Example 6.10.

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	01	0 1	0 0	1 1	0
01	01	0 1	1 0	1 1	0
10	01	0 1	0 0	1 0	0
11	01	0 1	1 0	1 0	1

Figure 6.78. The excitation table for the circuit in Figure 6.77.

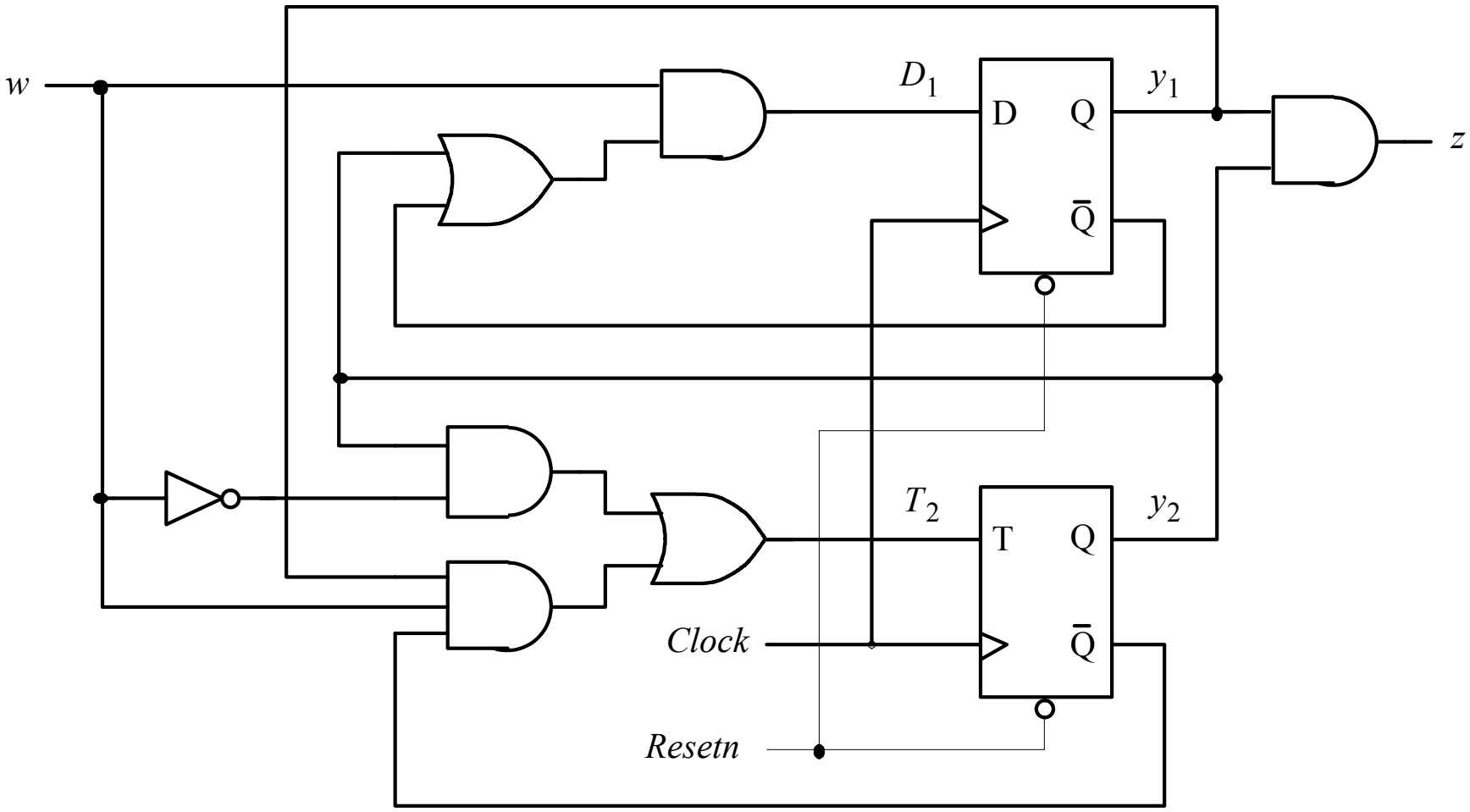


Figure 6.79. Circuit for Example 6.11.

Present state y_2y_1	Flip-flop inputs		Output z
	$w = 0$	$w = 1$	
	T_2D_1	T_2D_1	
0 0	0 0	01	0
0 1	0 0	10	0
1 0	1 0	01	0
1 1	1 0	01	1

Figure 6.80. Excitation table for the circuit in Figure 6.79.

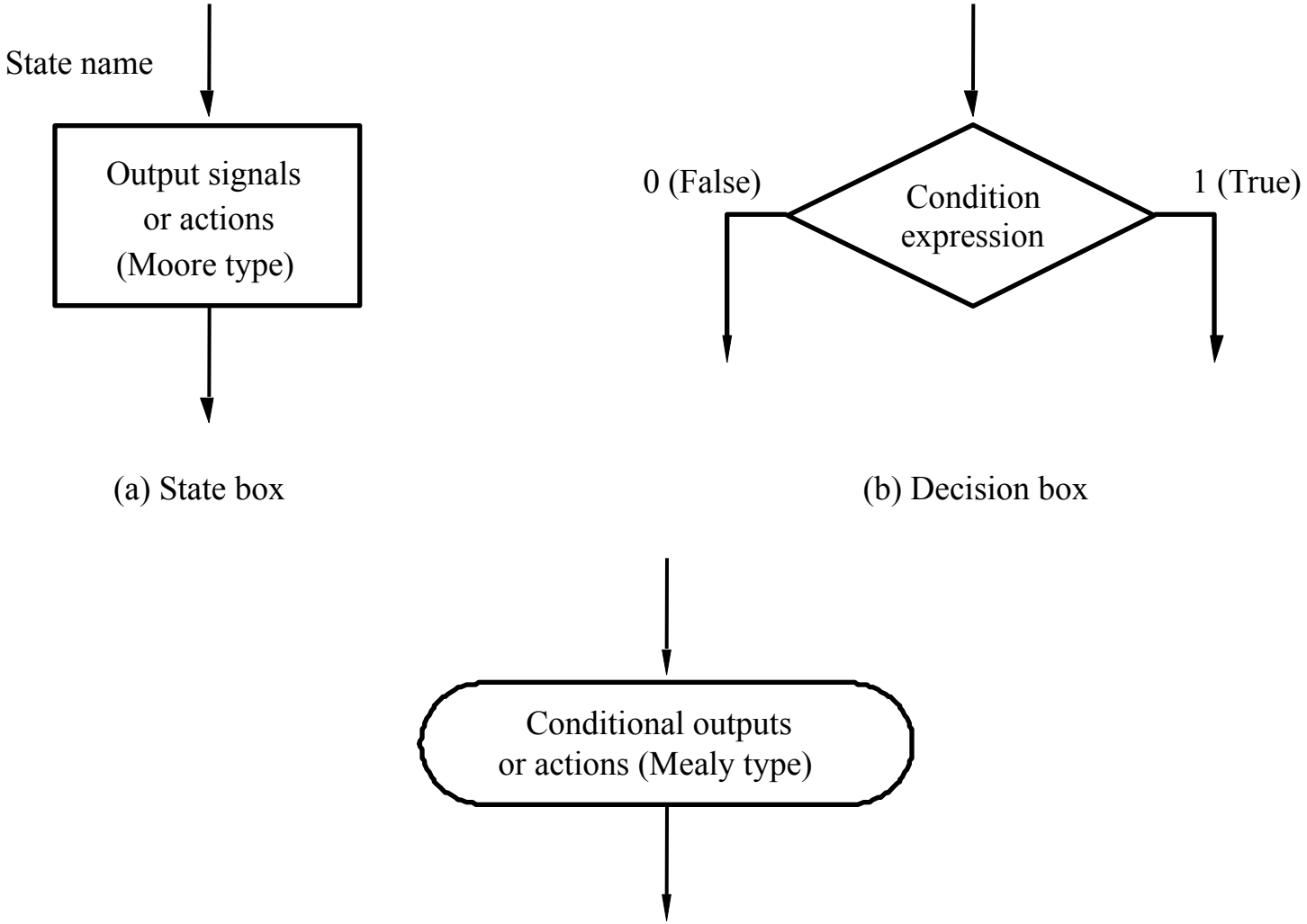


Figure 6.81. Elements used in ASM charts.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.82. ASM chart for the FSM in Figure 6.3.

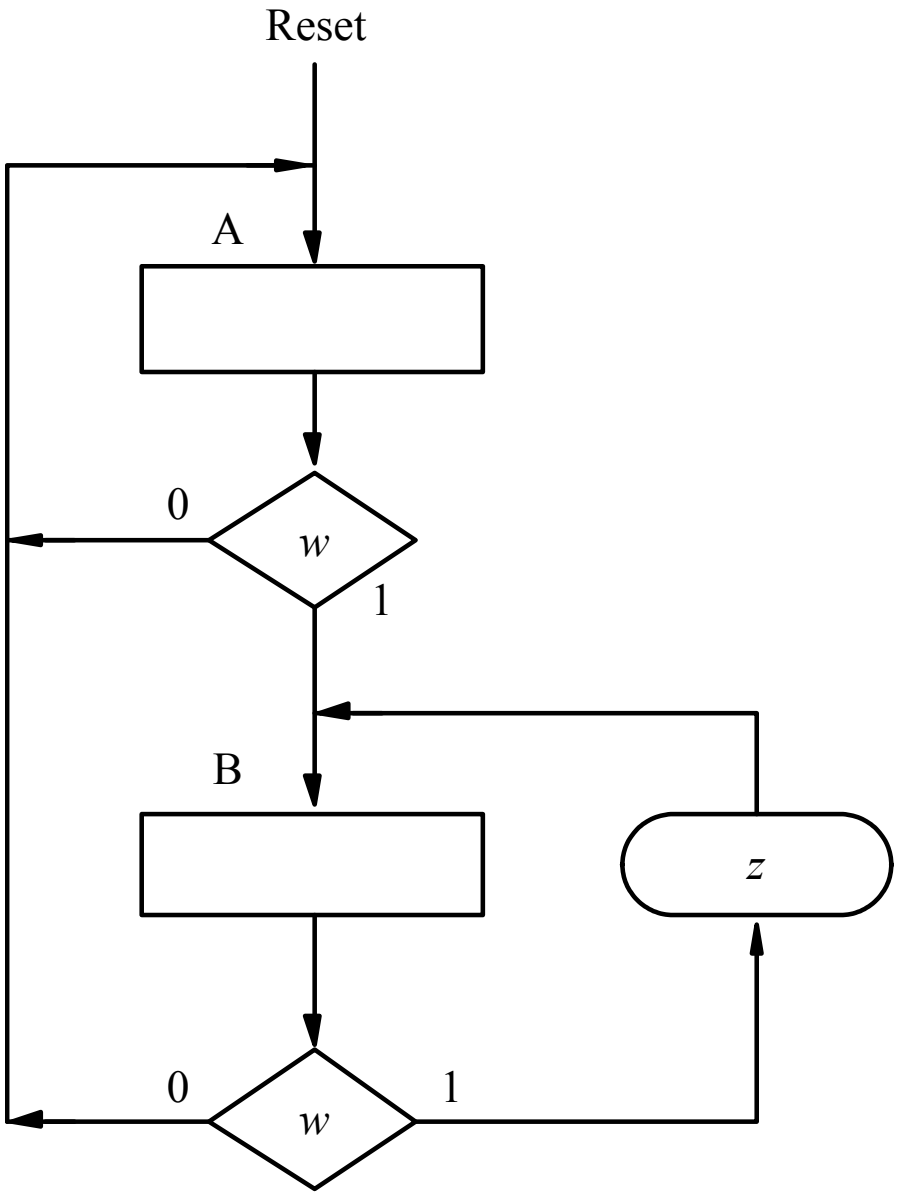


Figure 6.83. ASM chart for the FSM in Figure 6.23.

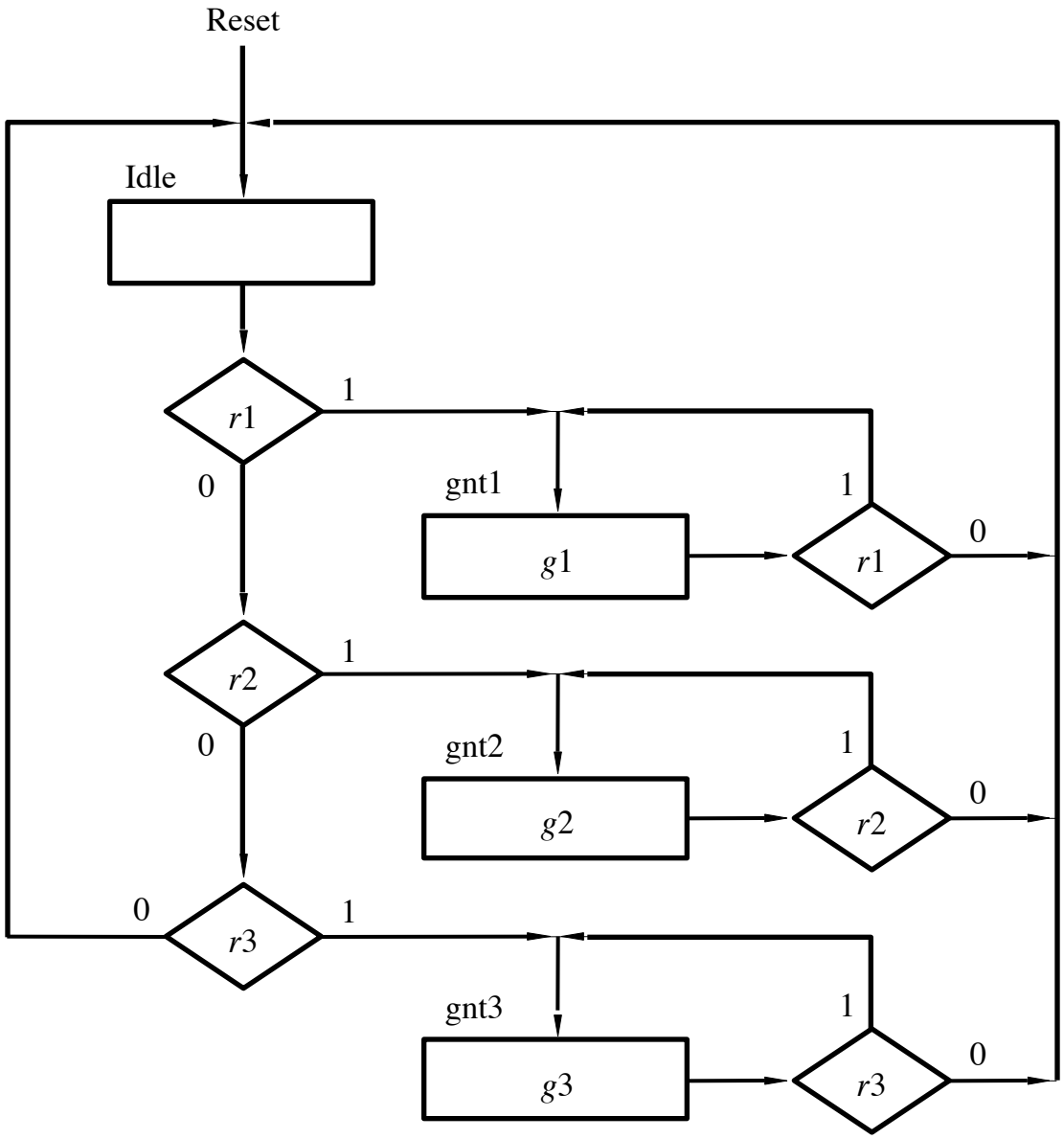


Figure 6.84. ASM chart for the arbiter FSM in Figure 6.73.

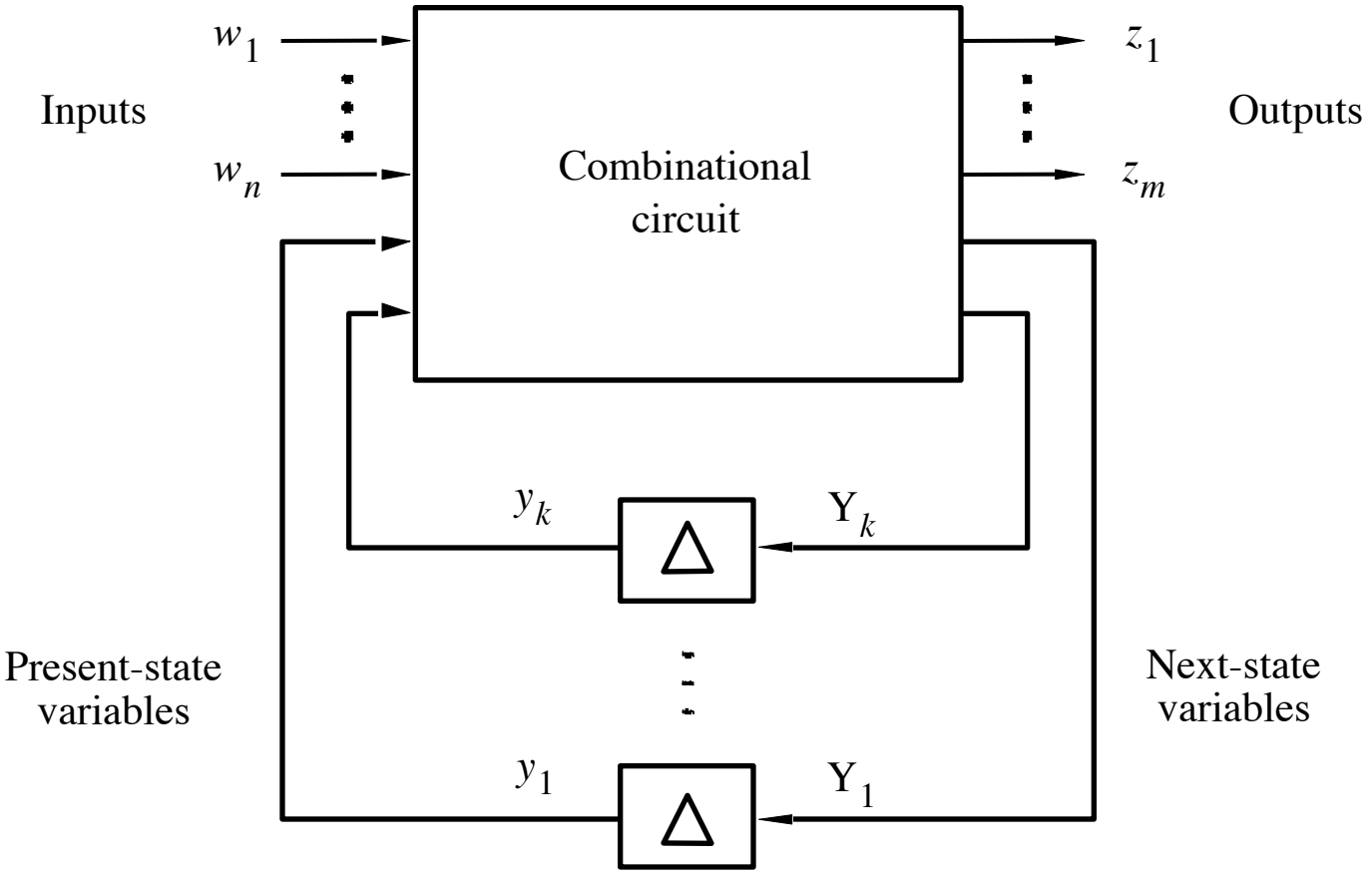


Figure 6.85. The general model for a sequential circuit.

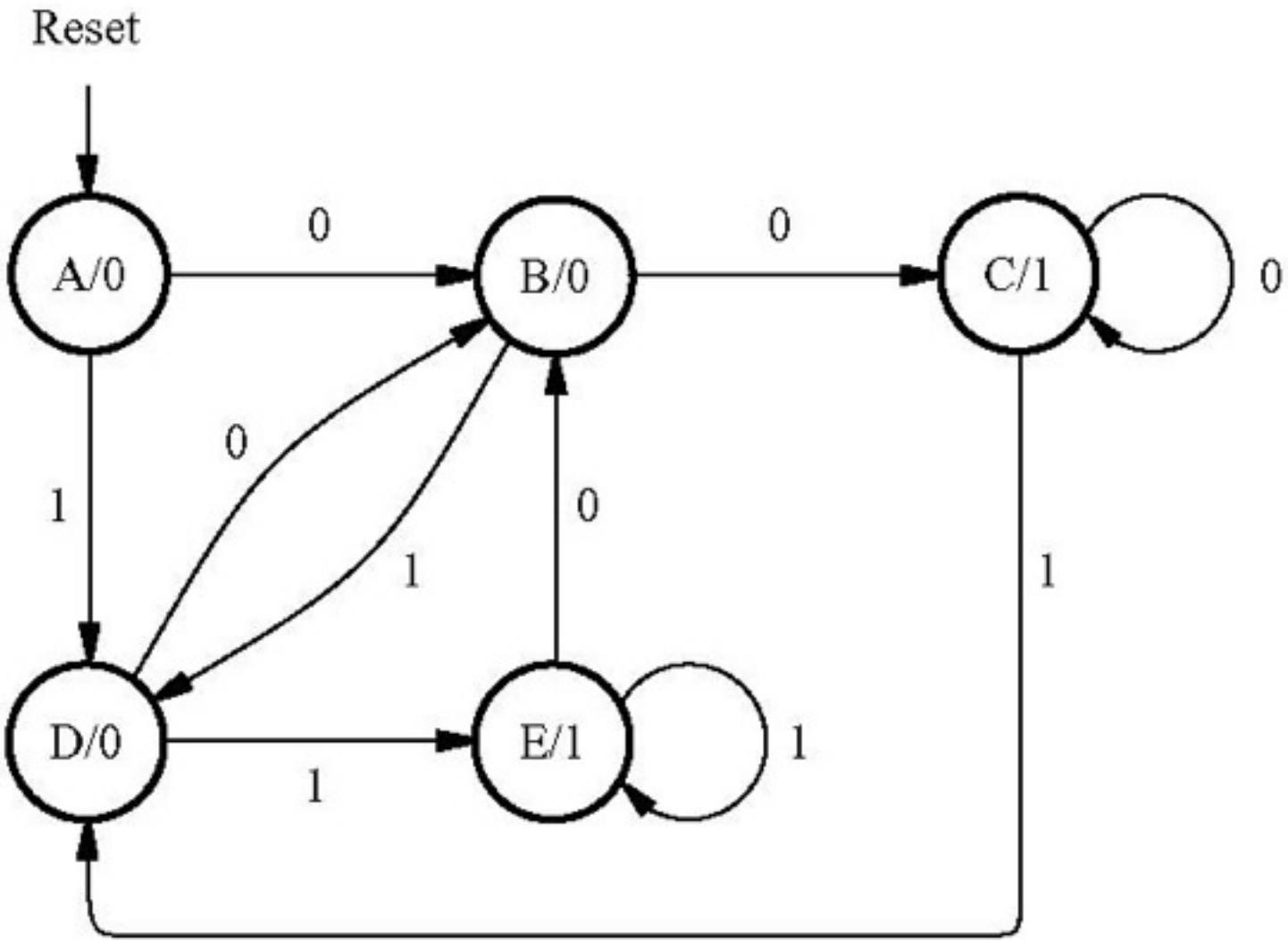


Figure 6.86. State diagram for Example 6.12.

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

Figure 6.87. State table for the FSM in Figure 6.86.

	Present state $y_3y_2y_1$	Next state		Output z
		$w = 0$	$w = 1$	
		$Y_3Y_2Y_1$	$Y_3Y_2Y_1$	
A	000	001	011	0
B	001	010	011	0
C	010	010	011	1
D	011	001	100	0
E	100	001	100	1

Figure 6.88. State-assigned table for the FSM in Figure 6.87.

	Present state $y_3y_2y_1$	Next state		Output z
		$w = 0$	$w = 1$	
		$Y_3Y_2Y_1$	$Y_3Y_2Y_1$	
A	000	100	110	0
B	100	101	110	0
C	101	101	110	1
D	110	100	111	0
E	111	100	111	1

Figure 6.89. An improved state assignment for the FSM in Figure 6.87.

Present state	Ne xt state		Output
	$w = 0$	$w = 1$	z_{zeros}
D	E	D	0
E	F	D	0
F	F	D	1

(a) State table

	Present state	Next state		Output
		$w = 0$	$w = 1$	
	y_4y_3	Y_4Y_3	Y_4Y_3	z_{zeros}
D	00	01	00	0
E	01	11	00	0
F	11	11	00	1
	10	dd	dd	d

Figure 6.90. FSM that detects a sequence of two zeros.

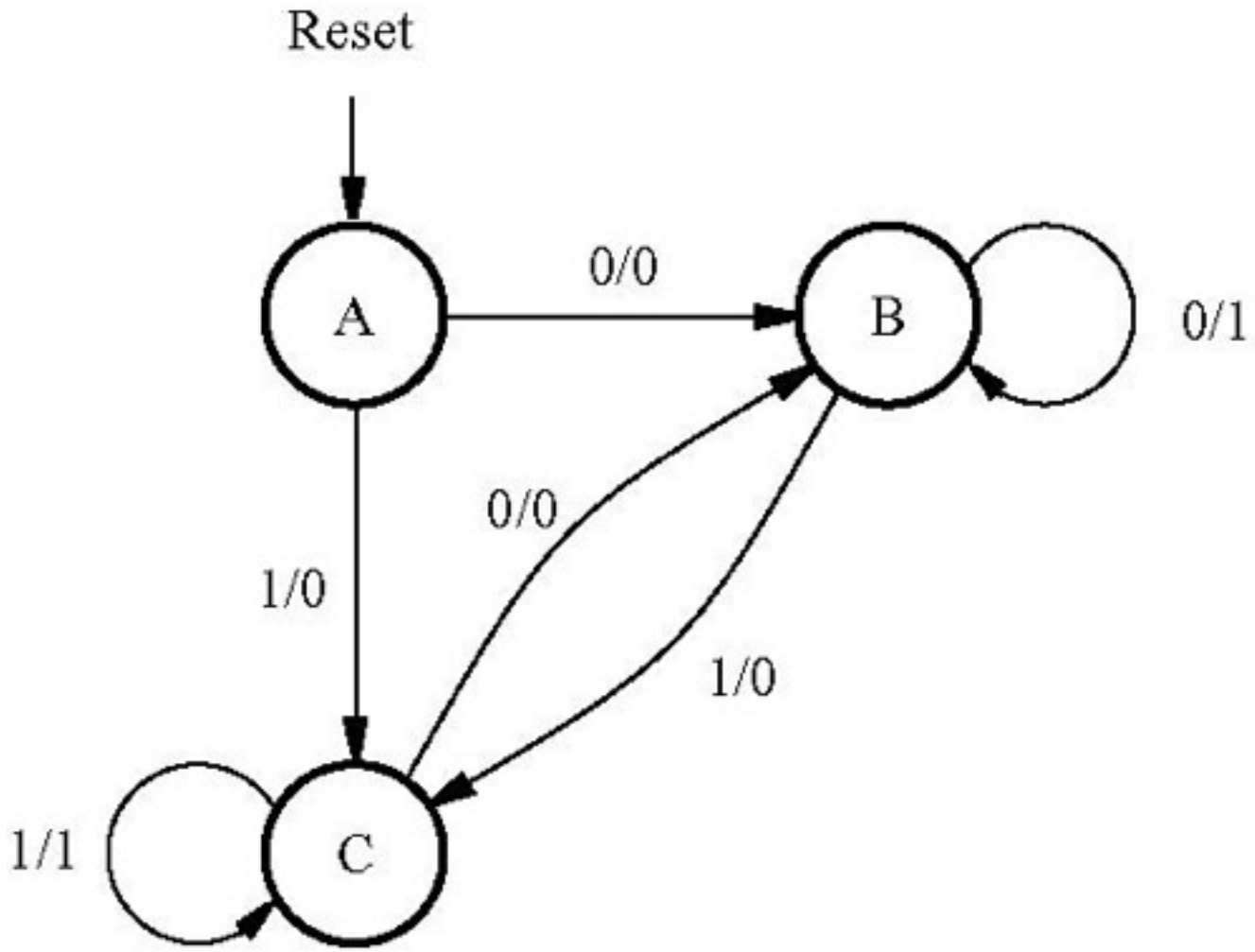


Figure 6.91. State diagram for Example 6.14.

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

Figure 6.92. State table for the FSM in Figure 6.91.

	Present state	Next state		Output	
		$w = 0$	$w = 1$	$w = 0$	$w = 1$
	y_2y_1	Y_2Y_1	Y_2Y_1	z	z
A	00	01	11	0	0
B	01	01	11	1	0
C	11	01	11	0	1

Figure 6.93. State-assigned table for the FSM in Figure 6.92.

	Present state $y_3y_2y_1$	Flip-flop inputs								Output z
		$w = 0$				$w = 1$				
		$Y_3Y_2Y_1$	J_3K_3	J_2K_2	J_1K_1	$Y_3Y_2Y_1$	J_3K_3	J_2K_2	J_1K_1	
A	000	100	$1d$	$0d$	$0d$	110	$1d$	$1d$	$0d$	0
B	100	101	$d0$	$0d$	$1d$	110	$d0$	$1d$	$0d$	0
C	101	101	$d0$	$0d$	$d0$	110	$d0$	$1d$	$d1$	1
D	110	100	$d0$	$d1$	$0d$	111	$d0$	$d0$	$1d$	0
E	111	100	$d0$	$d1$	$d1$	111	$d0$	$d0$	$d0$	1

Figure 6.94. Excitation table for the FSM in Figure 6.89 with JK flip-flops.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.95. Verilog code for the FSM in Figure 6.86.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.96. Verilog code for the FSM in Figure 6.91.

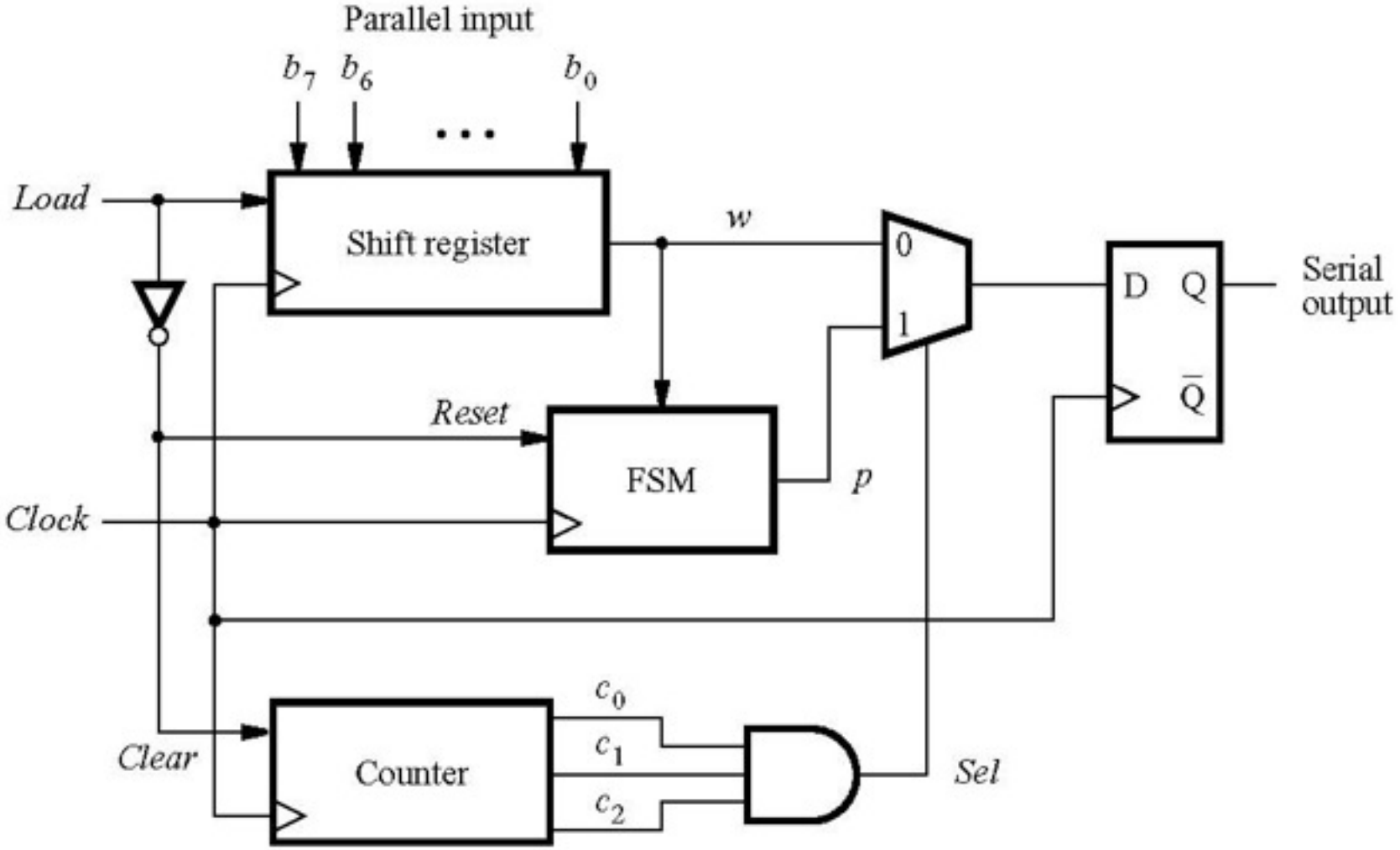


Figure 6.97. Parallel-to-serial converter.

Please see “portrait orientation” PowerPoint file for Chapter 6

Figure 6.98. FSM for parity generation.

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

Figure P6.1. State-assigned table for Problems 6.1 and 6.2.

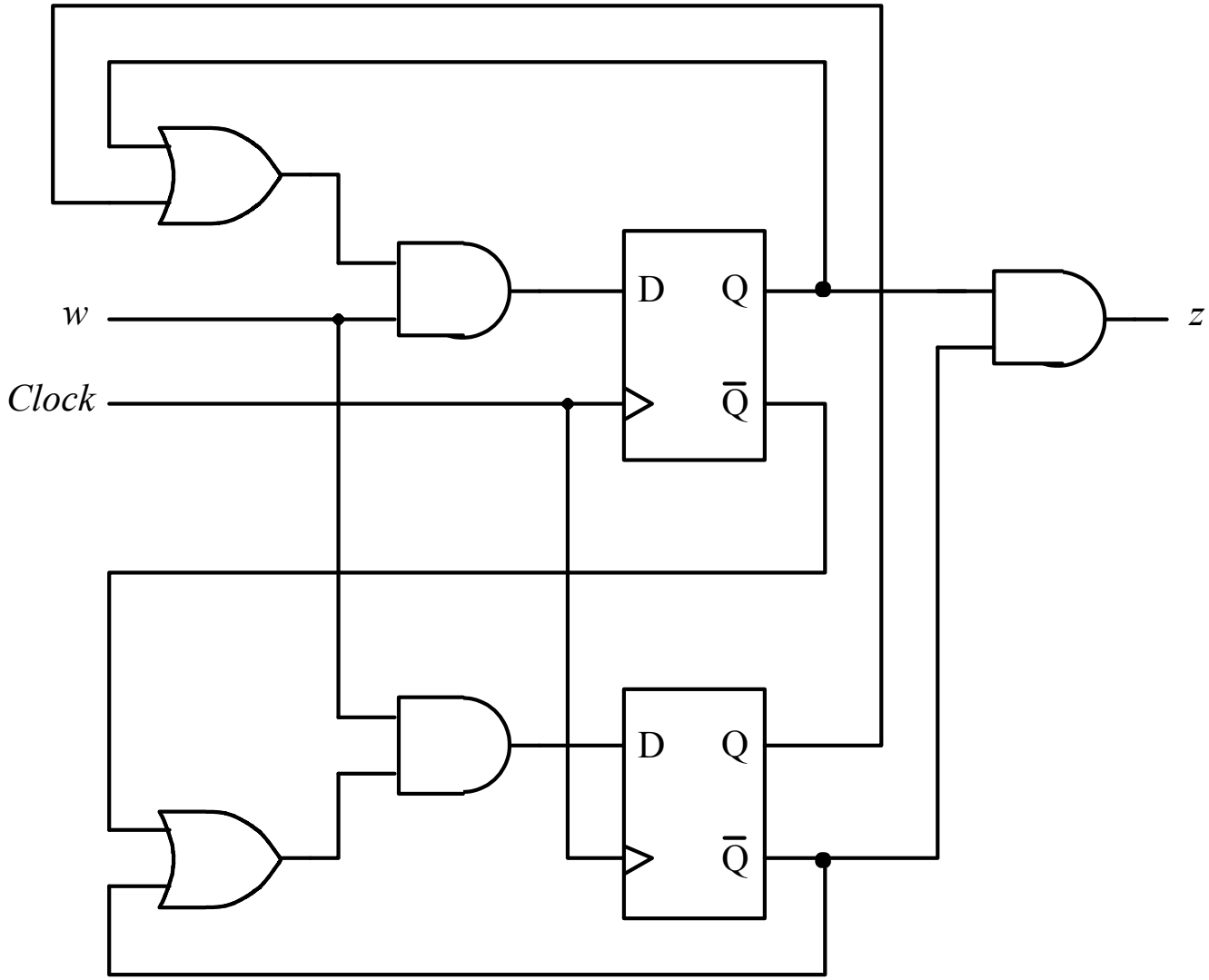


Figure P6.2. Circuit for Problem 6.29.