Synchronous Sequential Circuits

Chapter 8

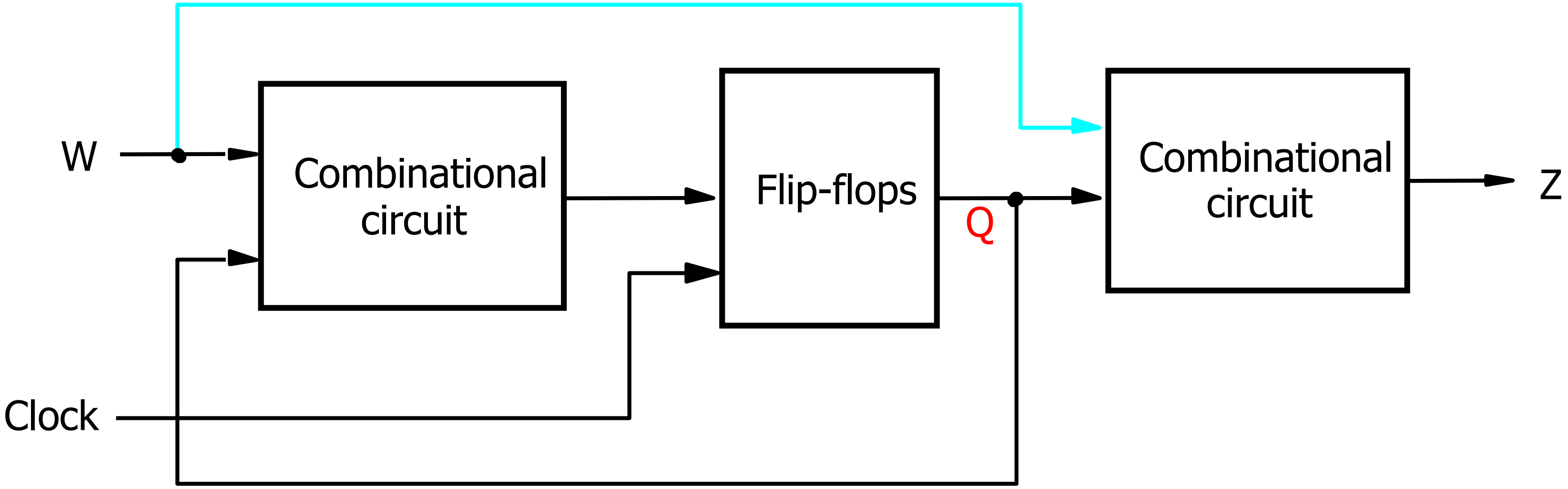
Chapter Objectives

- Design techniques for circuits that use flip-flops
- The concept of states that their implementation with flip-flops
- Synchronous control by using a clock signal
- A complete procedure for designing synchronous sequential circuits
- The concept of finite state machines

Contents

1. Basic Design Steps
2. State-Assignment Problem
3. Mealy State Model
4. Serial Adder Example
5. State Minimization
6. Design of a Counter Using the Sequential Circuit Approach
7. Analysis of Synchronous Sequential Circuit

The general form of a sequential circuit



Moore Type: Outputs depend only on the state of circuits
Mealy Type: Outputs depend on both the state and the inputs

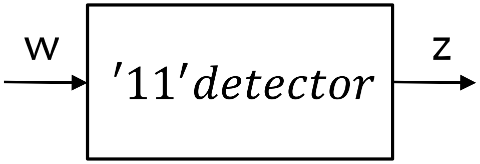
Figure 8.1. The general form of a sequential circuit.

BASIC DESIGN STEPS

Basic Design Steps

Moore-type Design

- Specification of *sequence detector*
  - One input w, one output z
  - All changes on the positive edge of a clock signal
  - The output z is equal to 1 if during two immediately preceding clock cycles the input w was equal to 1. Otherwise, the value of z is equal to 0.



w

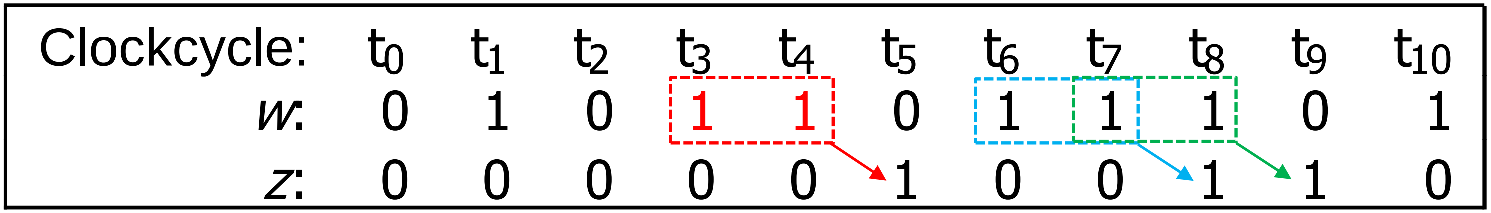
$'11'detector$

z

$'11'detector$

Sequences of input and output signals

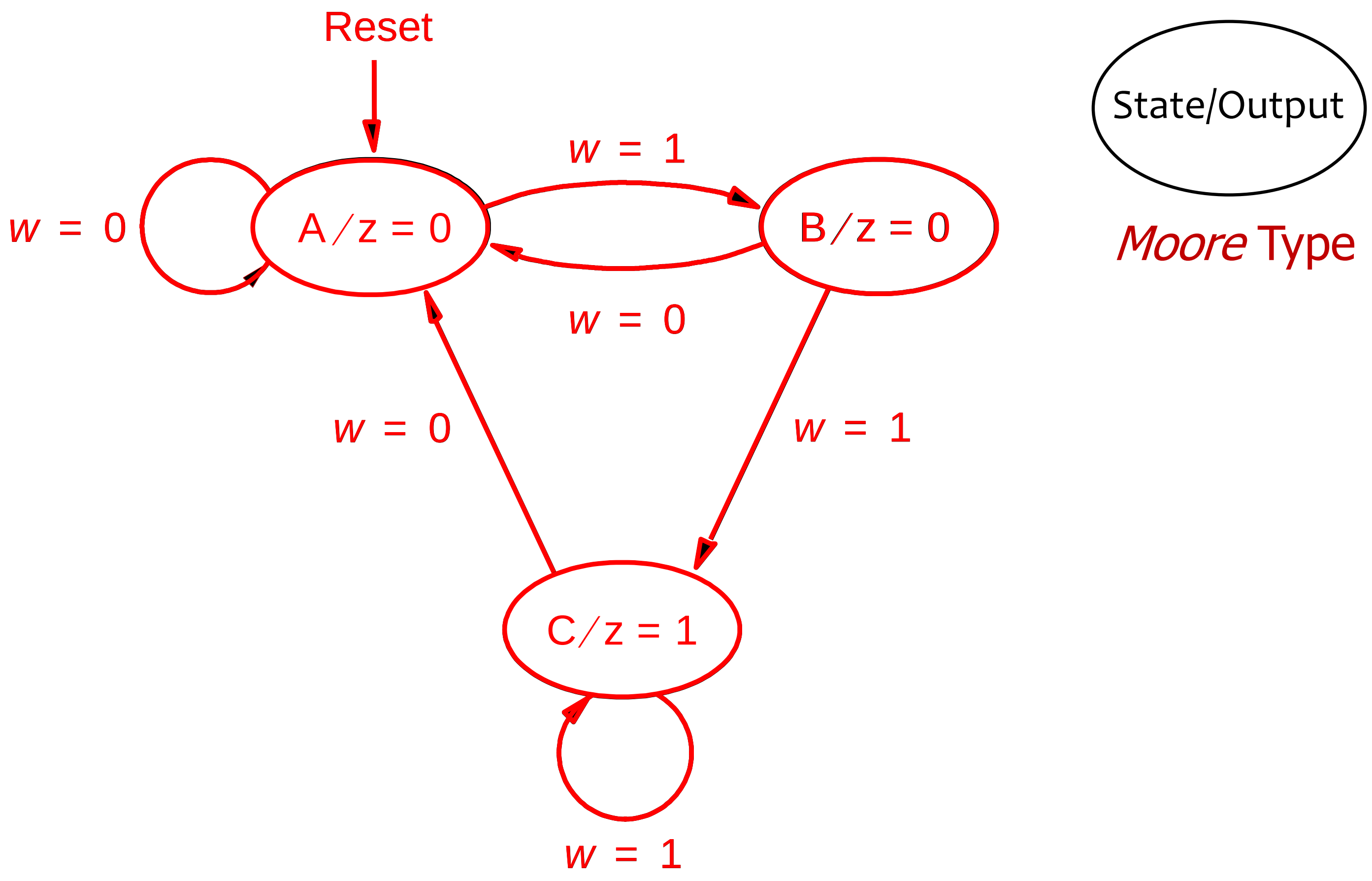
Moore-type Design



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 8.2. Sequences of input and output signals.

State diagram of a simple sequential circuit



Moore Type

Figure 8.3. State diagram of a simple sequential circuit.

State Table

In sequential logic, the next states are functions of the present states and inputs.
And, the outputs are functions of the present states and/or inputs.

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

W : input

$$f(w_1, w_2) = w_1' w_2 + w_1 w_2'$$

w_1	w_2	f
0	0	0
0	1	1
1	0	1
1	1	0

Truth table

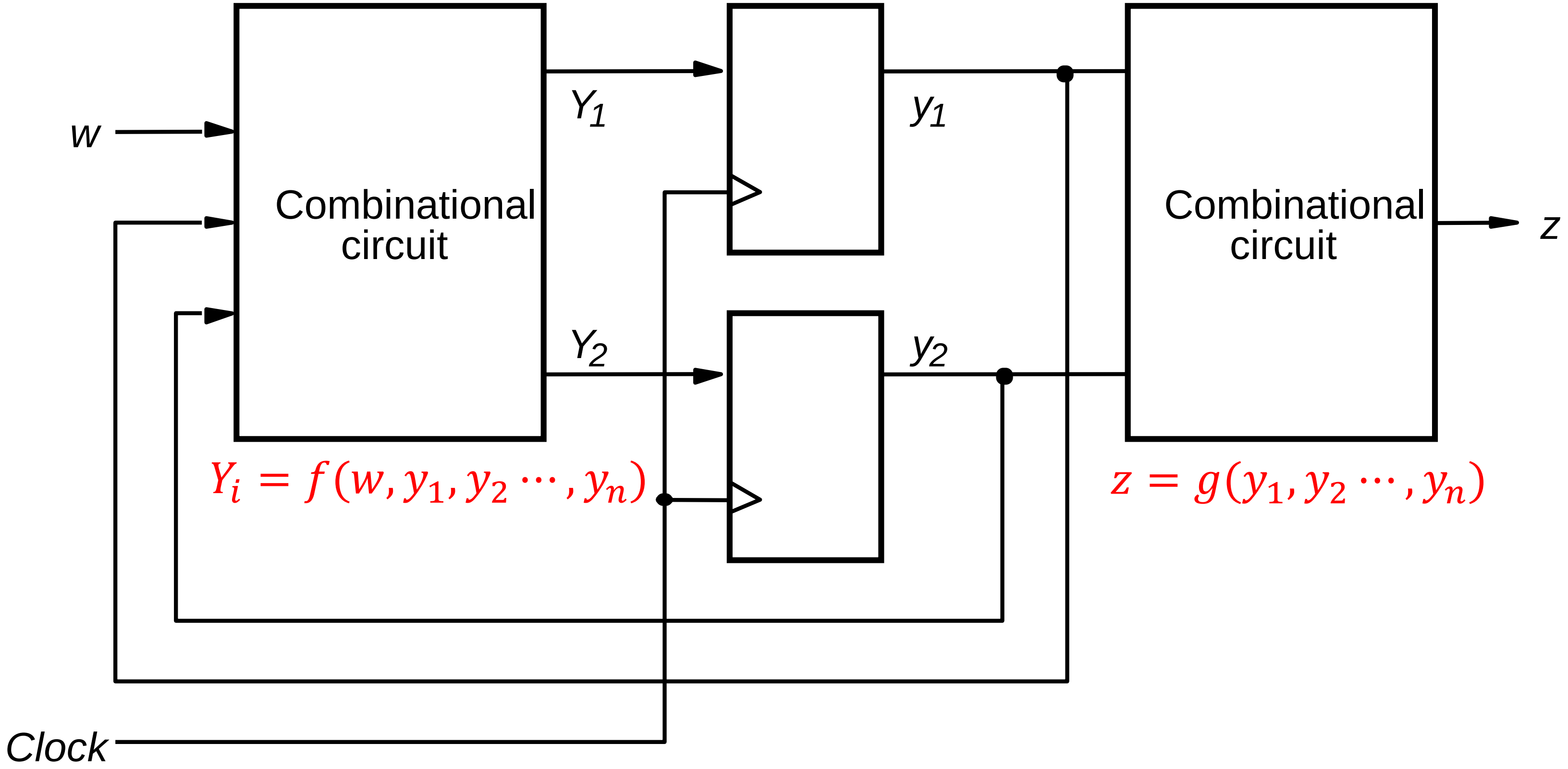
Difference in state table between Moore-type and Mealy type design is only the output.

In combinational logic,
output is function of inputs only.

Figure 8.4. State table for the sequential circuit in Figure 8.3.

A general sequential circuit

Moore-type Design



$Y_i = f(w, y_1, y_2 \cdots, y_n)$

$z = g(y_1, y_2 \cdots, y_n)$

Y_i : next state (input of F/F) y_i : present state

Figure 8.5. A general sequential circuit with input *w*, output *z*, and two state flip-flops.

State-Assigned Table

	Present state	Next state		Output
		$w = 0$	$w = 1$	z
	$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	
A	00	00	01	0
B	01	00	10	0
C	10	00	10	1
	11	*dd*	*dd*	*d*

Binary number code

Unused state

Figure 8.6. State-assigned table for the sequential circuit in Figure 8.4.

State-Assigned Table

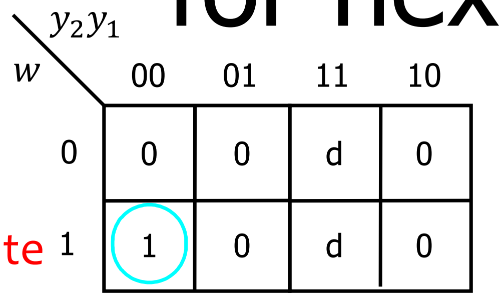
Present state & input $y_2 y_1 w$	Next state $Y_2 Y_1$
0 0 0	0 0
0 0 1	0 1
0 1 0	0 0
0 1 1	1 0
1 0 0	0 0
1 0 1	1 0
1 1 0	d d
1 1 1	d d

State-Assigned Table

Present state y_2y_1	Output z
00	0
01	0
10	1
11	d

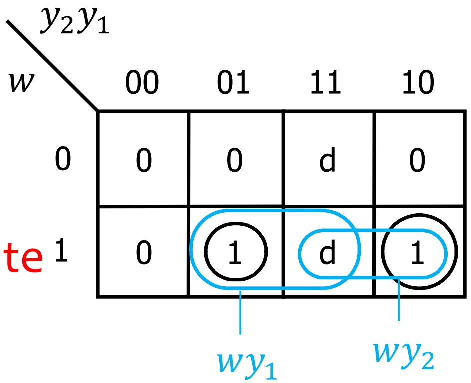
Derivation of logic expressions for next states and output

Y_1
Next state



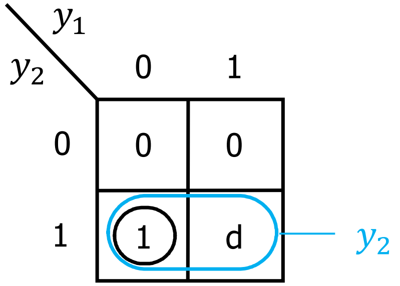
$w \backslash y_2y_1$	00	01	11	10
0	0	0	d	0
1	1	0	d	0

Y_2
Next state



$w \backslash y_2y_1$	00	01	11	10
0	0	0	d	0
1	0	1	d	1

z
output



$y_2 \backslash y_1$	0	1
0	0	0
1	1	d

Ignoring don't cares

$$Y_1 = wy_1'y_2'$$

$$Y_2 = wy_1y_2' + wy_1'y_2$$

$$z = y_1'y_2$$

Using don't cares

$$Y_1 = wy_1'y_2'$$

$$\begin{aligned} Y_2 &= wy_1 + wy_2 \\ &= w(y_1 + y_2) \end{aligned}$$

$$z = y_2$$

Figure 8.7. Derivation of logic expressions for the sequential circuit in Figure 8.6.

Implementation of the sequential circuit

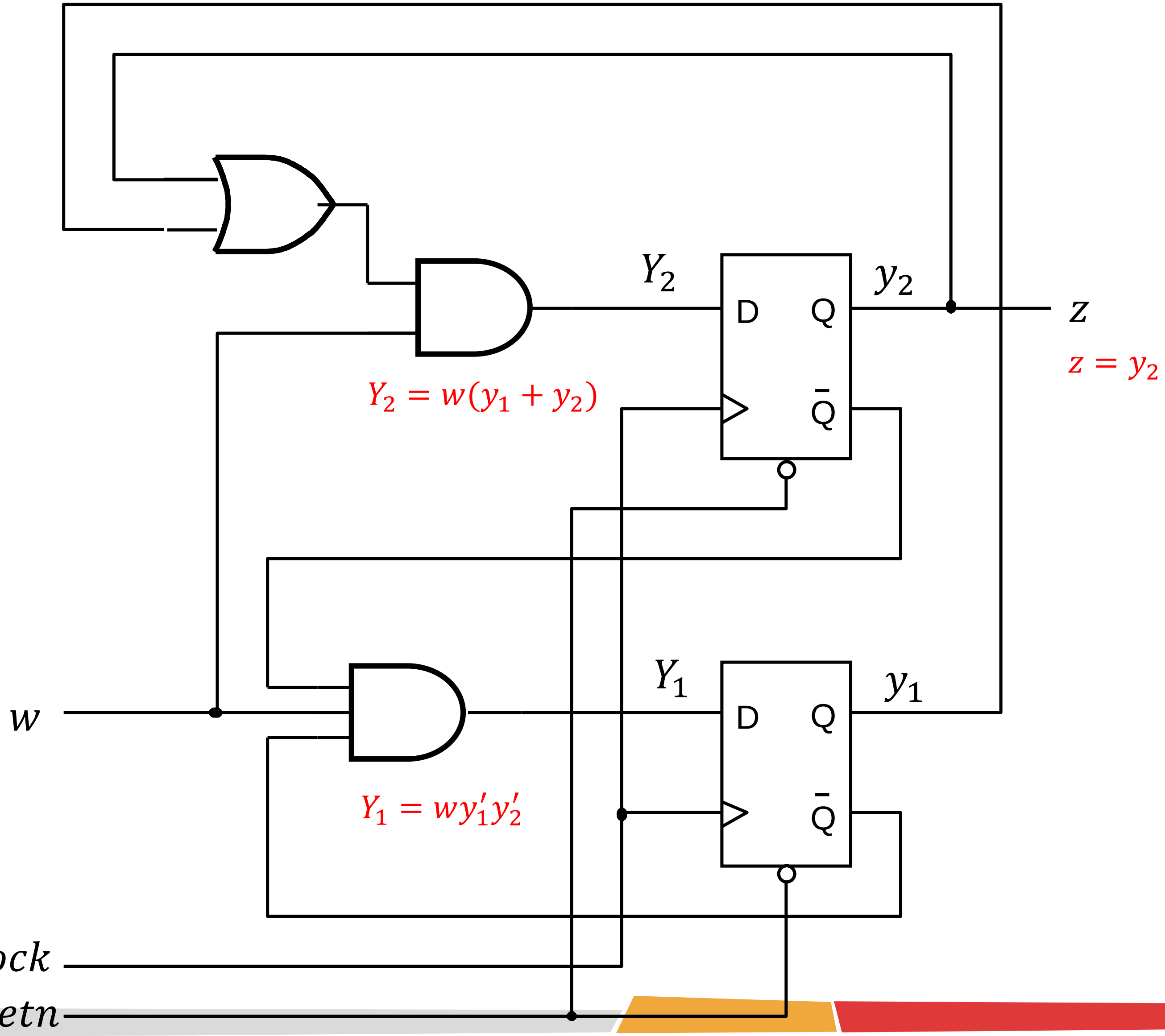


Figure 8.8. Final implementation of the sequential circuit derived in Figure 8.7.

Timing diagram

$z = y_2$

$Y_1 = w y_1' y_2'$

$Y_2 = w(y_1 + y_2)$

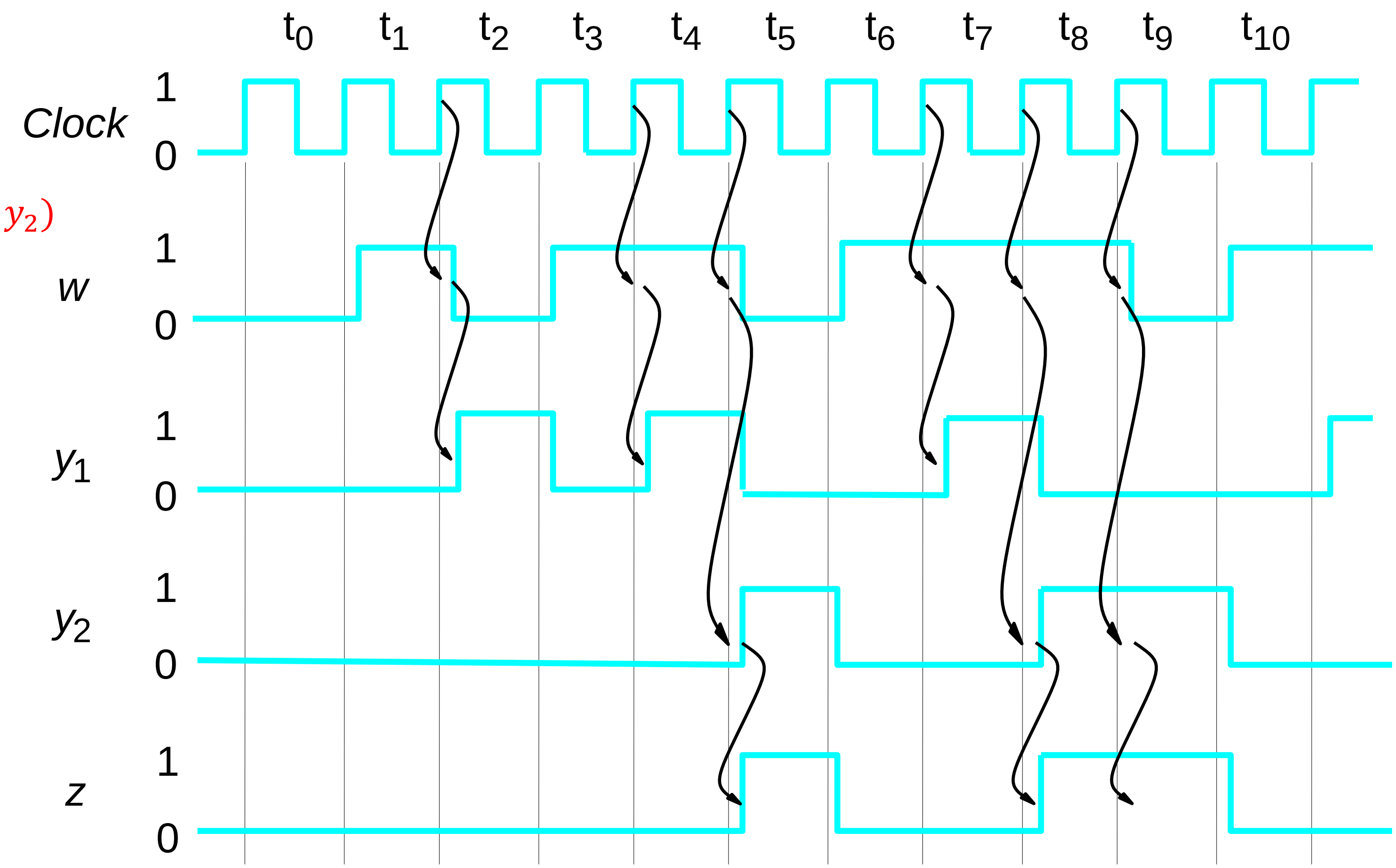


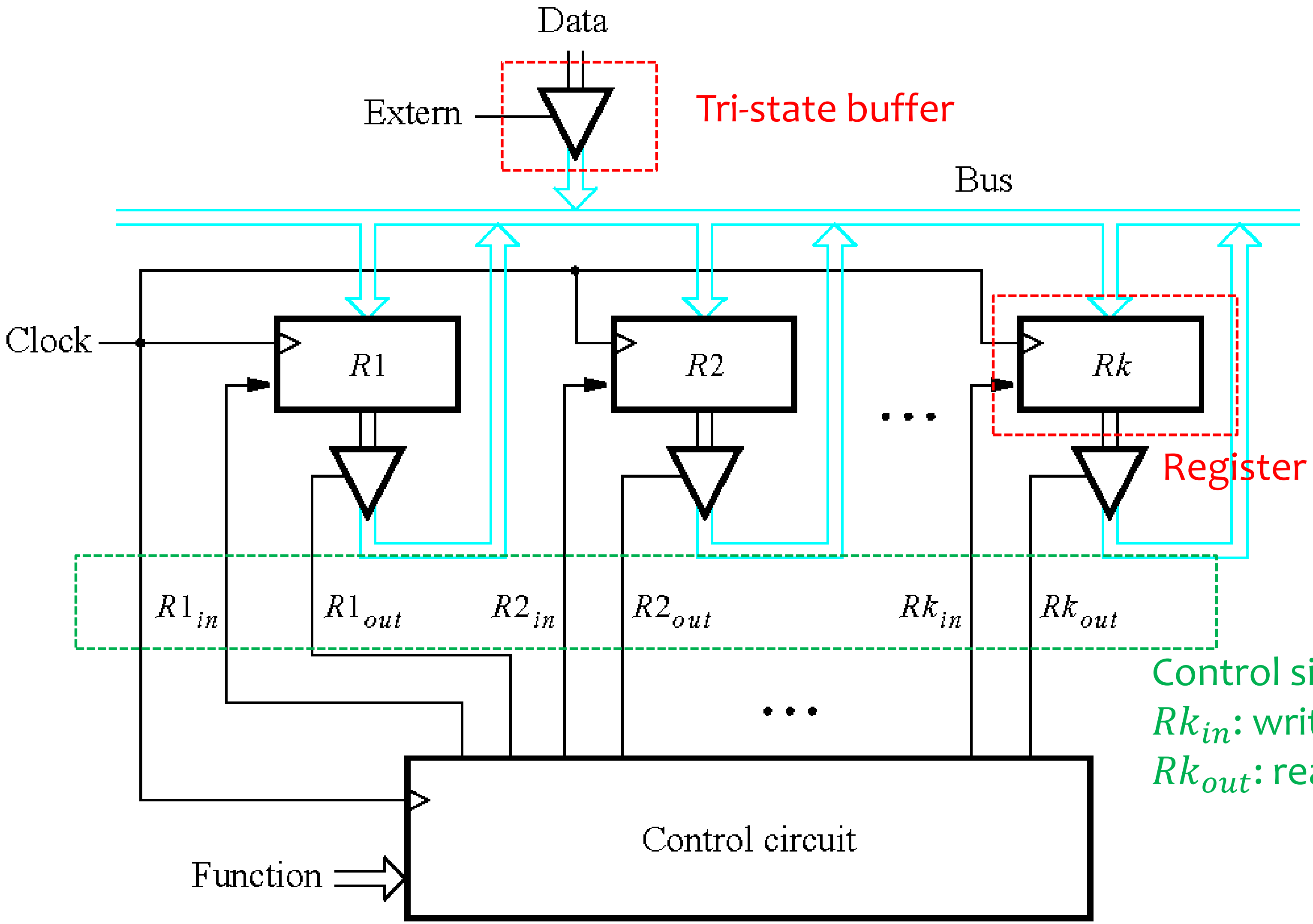
Figure 8.9. Timing diagram for the circuit in Figure 8.8.

Summary of Design Steps

1. Obtain the specification of the desired circuit.
2. Derive the state diagram.
3. Create a state table from the state diagram.
4. Minimize the number of states (The process of the state minimization).
5. Decide on the number of state variables.
6. Choose the type of flip-flops to be used in the circuit.
7. Implement the circuit.

Example 8.1

- The contents of R1 and R2 can be swapped using R3 as a temporary storage
  1. R2 ➔ R3
     - Control signal $R2_{out} = 1$ and $R3_{in} = 1$
  2. R1 ➔ R2
     - Control signal $R1_{out} = 1$ and $R2_{in} = 1$
  3. R3 ➔ R1
     - Control signal $R3_{out} = 1$ and $R1_{in} = 1$
     - Task is completed, $Done = 1$
- Swapping is performed in response to a pulse on an input signal called w



Tri-state buffer

Register

Control signal

Rk_{in}: write signal

Rk_{out}: read signal

Signal needed in Example 8.1

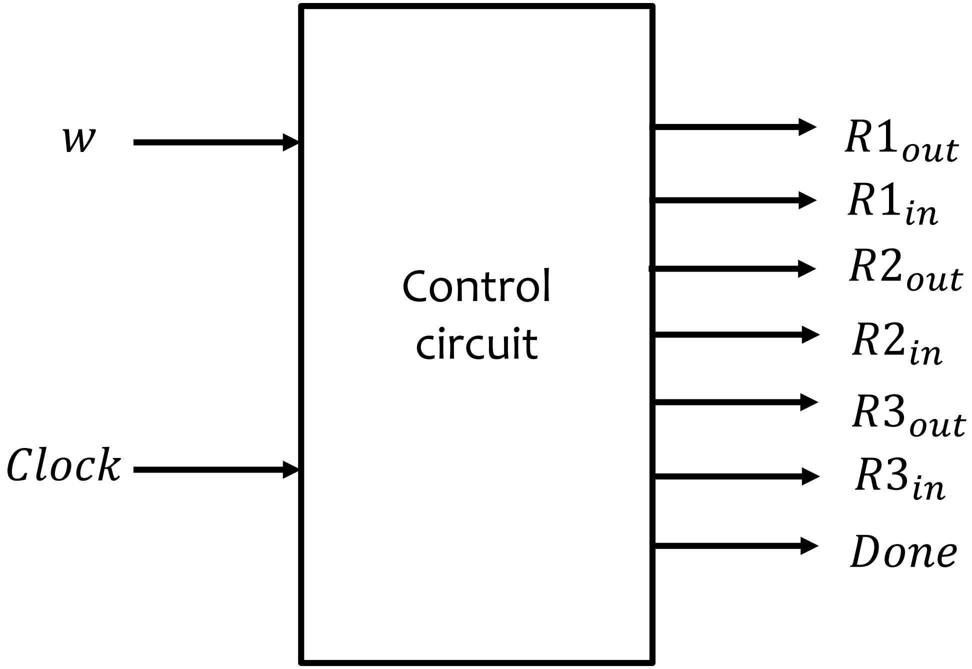
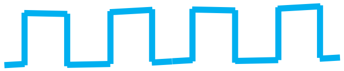


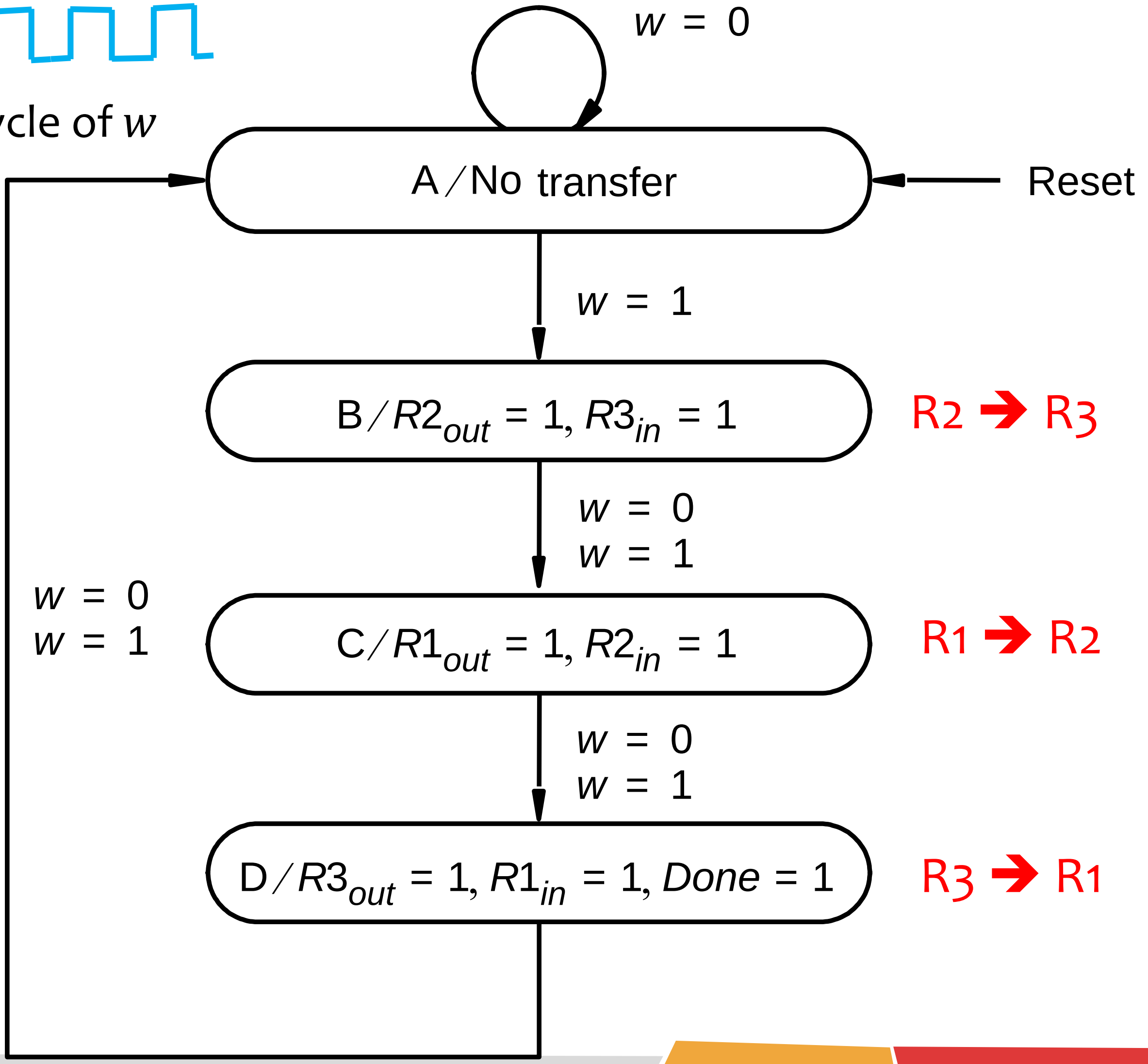
Figure 8.10. Signal needed in Example 8.1.

State diagram for Example 8.1

w : input



State change/cycle of *w*



$w = 0$

A / No transfer

Reset

$w = 1$

B / $R2_{out} = 1$, $R3_{in} = 1$

R2 → R3

$w = 0$
$w = 1$

$w = 0$
$w = 1$

C / $R1_{out} = 1$, $R2_{in} = 1$

R1 → R2

$w = 0$
$w = 1$

D / $R3_{out} = 1$, $R1_{in} = 1$, $Done = 1$

R3 → R1

Figure 8.11. State diagram for Example 8.1.

State table for Example 8.1

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

R2 → R3 $R2_{out} = 1$ and $R3_{in} = 1$

R1 → R2 $R1_{out} = 1$ and $R2_{in} = 1$

R3 → R1 $R3_{out} = 1$ and $R1_{in} = 1, Done = 1$

Figure 8.12. State table for Example 8.1.

State-assigned table

	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	0 0	0 1	0	0	0	0	0	0	0
B	01	1 0	1 0	0	0	1	0	0	1	0
C	10	1 1	1 1	1	0	0	1	0	0	0
D	11	0 0	0 0	0	1	0	0	1	0	1

Binary number code

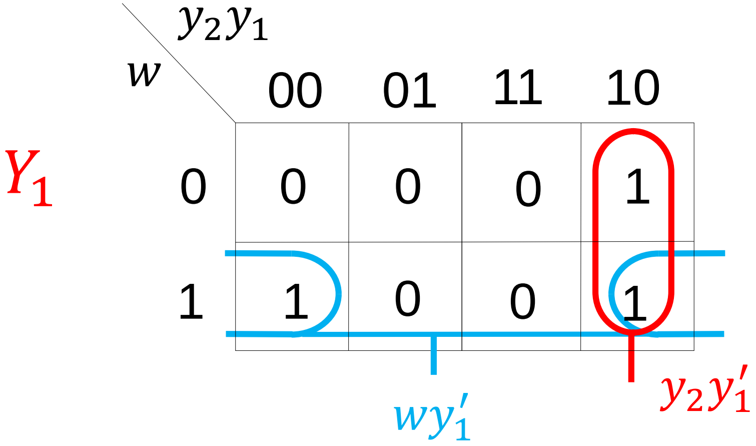
$$R2_{out} = R3_{in} = y_2'y_1$$

$$R3_{out} = R1_{in} = y_2y_1'$$

$$R1_{out} = R2_{in} = Done = y_2y_1$$

Figure 8.13. State-assigned table for the sequential circuit in Figure 8.12.

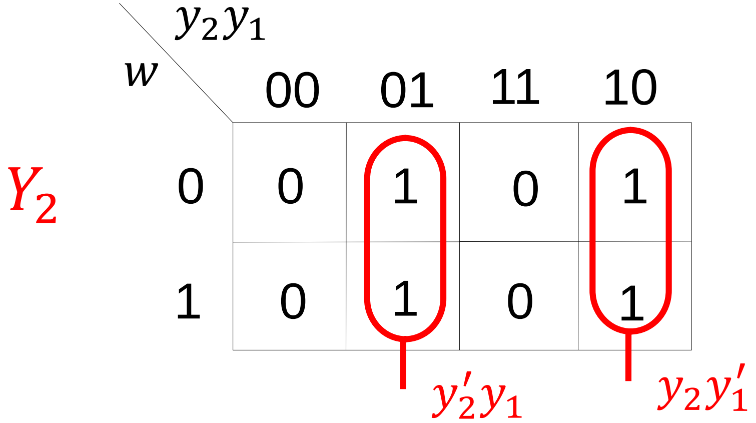
Derivation of next-state expressions



Y_1

w \ y_2y_1	00	01	11	10
0	0	0	0	1
1	1	0	0	1

$$Y_1 = wy_1' + y_2y_1'$$



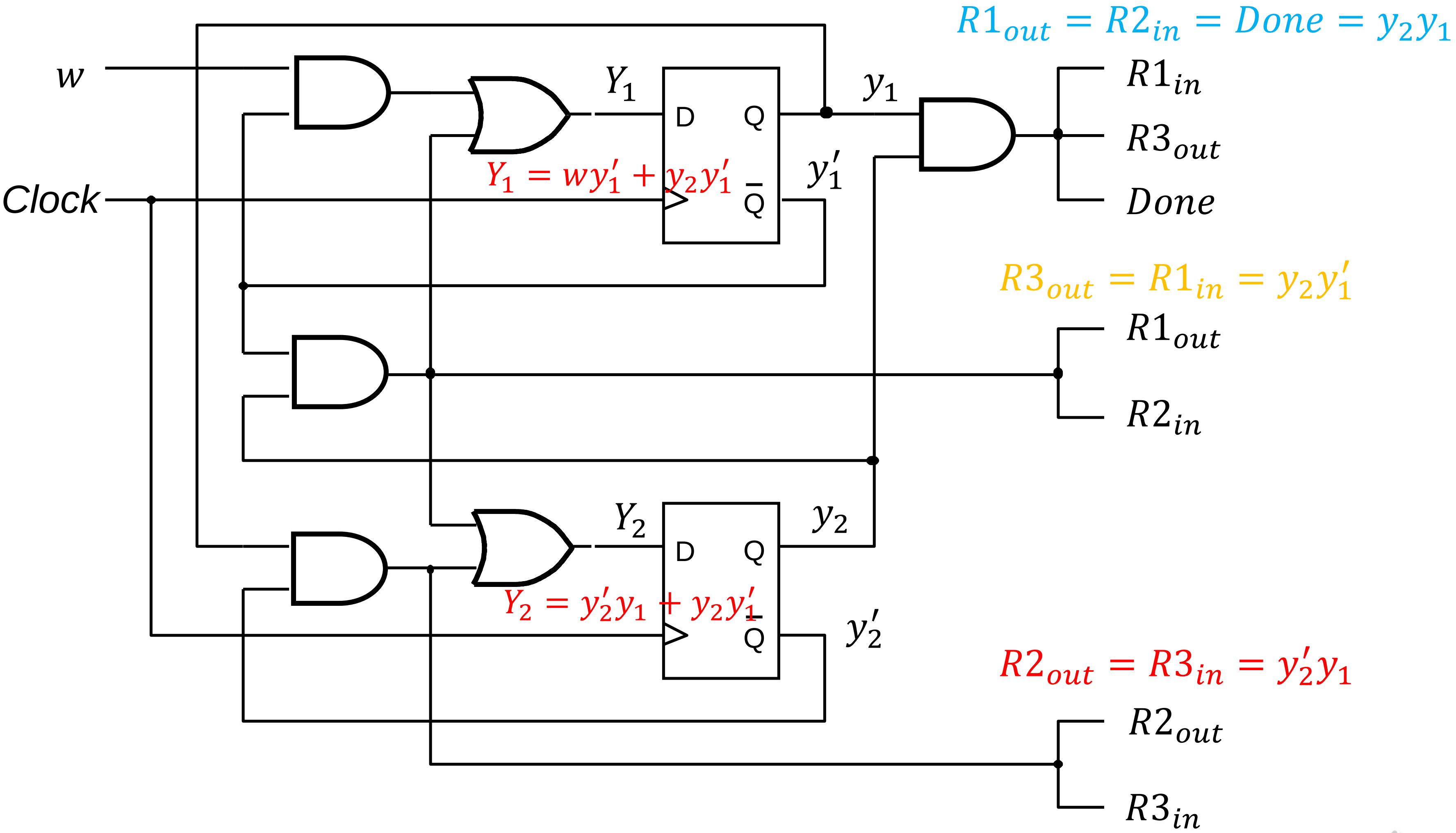
Y_2

w \ y_2y_1	00	01	11	10
0	0	1	0	1
1	0	1	0	1

$$Y_2 = y_2'y_1 + y_2y_1'$$

Figure 8.14. Derivation of next-state expressions for the sequential circuit in Figure 8.13.

Final implementation of sequential circuit in Figure 8.13



$R1_{out} = R2_{in} = Done = y_2y_1$

$Y_1 = wy_1' + y_2y_1'$

$R3_{out} = R1_{in} = y_2y_1'$

$Y_2 = y_2'y_1 + y_2y_1'$

$R2_{out} = R3_{in} = y_2'y_1$

Figure 8.15. Final implementation of sequential circuit in Figure 8.13.

STATE-ASSIGNMENT PROBLEM

State-Assignment Problem

	Present state	Next state		Output
		$w = 0$	$w = 1$	z
	$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	*dd*	*dd*	*d*

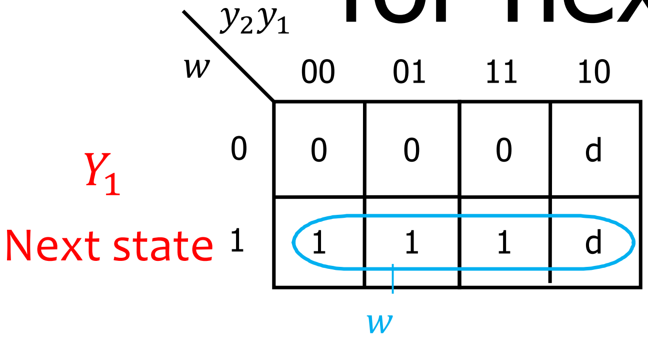
Gray code

Figure 8.16. Improved state assignment for the sequential circuit in Figure 8.4.

Derivation of logic expressions for next states and output

Y_1

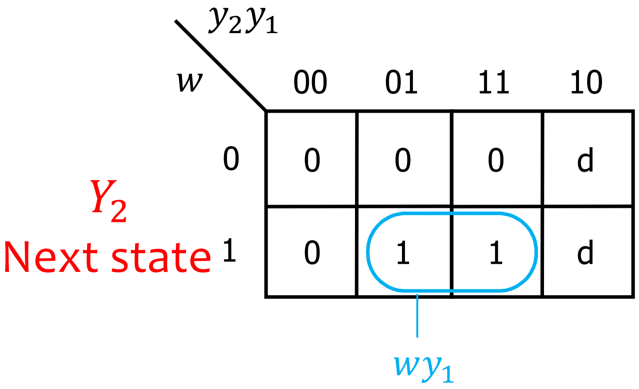
Next state



w \ y_2y_1	00	01	11	10
0	0	0	0	d
1	1	1	1	d

Y_2

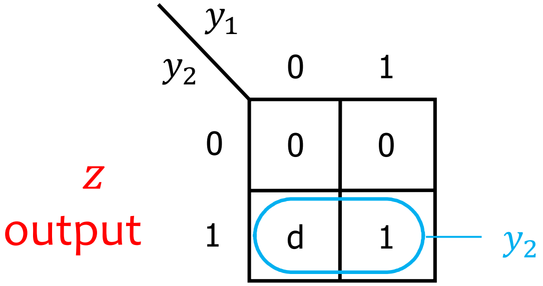
Next state



w \ y_2y_1	00	01	11	10
0	0	0	0	d
1	0	1	1	d

z

output



y_2 \ y_1	0	1
0	0	0
1	d	1

Gray code

$$Y_1 = w$$

$$Y_2 = wy_1$$

$$z = y_2$$

Binary code

$$Y_1 = wy_1'y_2'$$

$$\begin{aligned} Y_2 &= wy_1 + wy_2 \\ &= w(y_1 + y_2) \end{aligned}$$

$$z = y_2$$

Figure 8.7. Derivation of logic expressions for the sequential circuit in Figure 8.6.

Final circuit for the improved state assignment

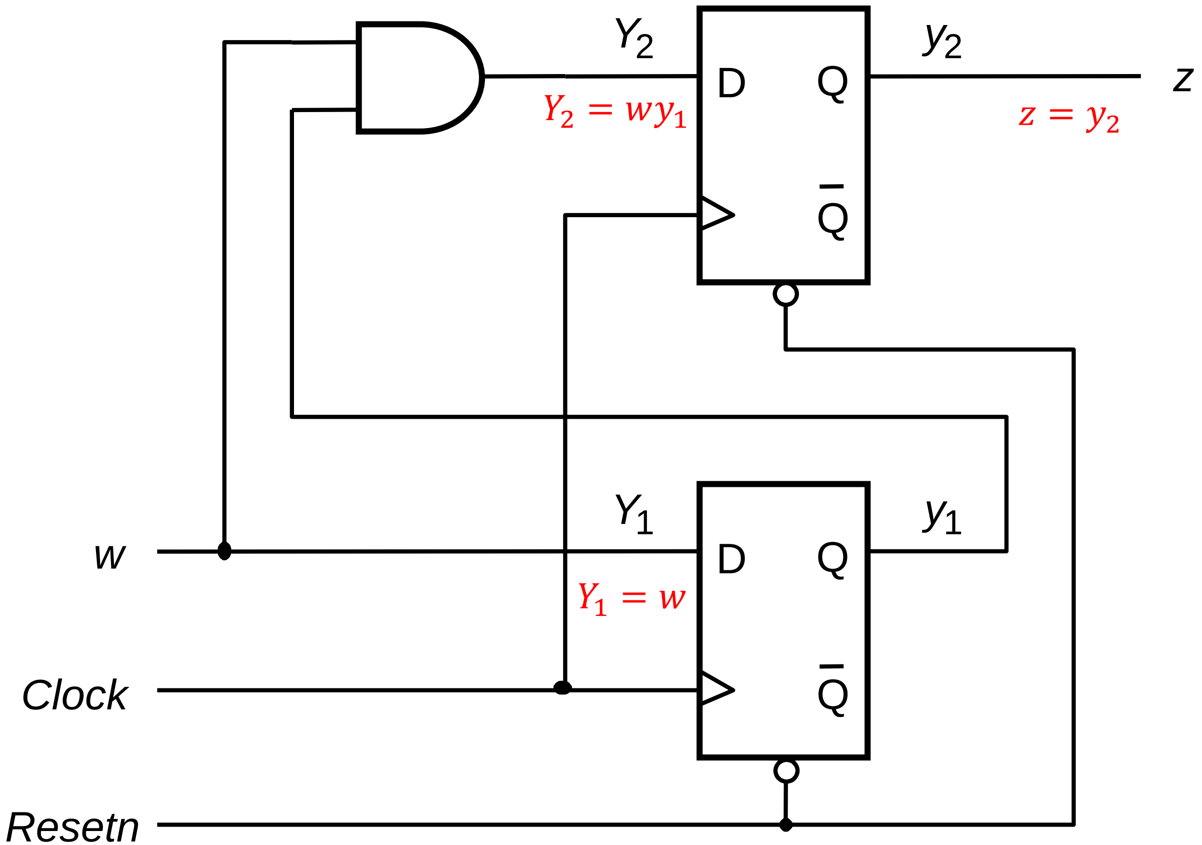


Figure 8.17. Final circuit for the improved state assignment in Figure 8.16.

Improved state assignment for the sequential circuit in Figure 8.12

	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	0 0	0 1	0	0	0	0	0	0	0
B	01	1 1	1 1	0	0	1	0	0	1	0
C	11	1 0	1 0	1	0	0	1	0	0	0
D	10	0 0	0 0	0	1	0	0	1	0	1

Gray code

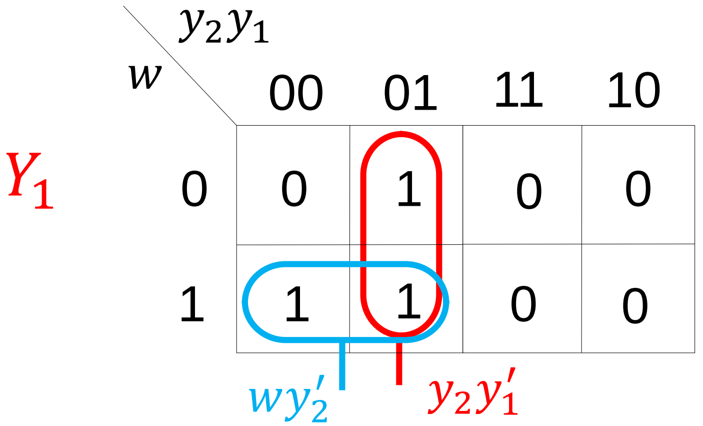
$$R2_{out} = R3_{in} = y_2'y_1$$

$$R1_{out} = R2_{in} = \boldsymbol{y_2y_1}$$

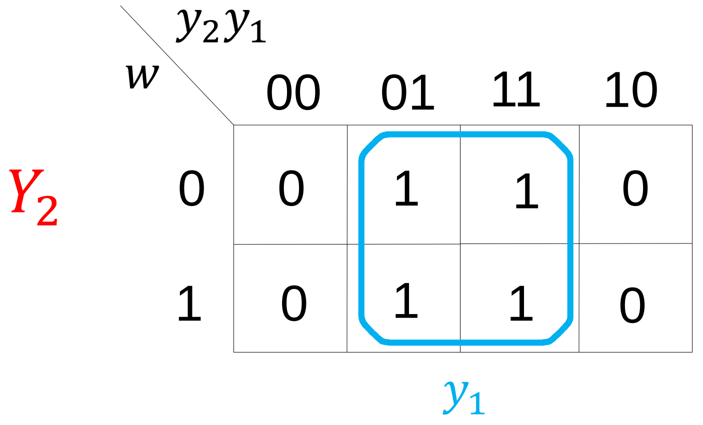
$$R3_{out} = R1_{in} = Done = \boldsymbol{y_2y_1'}$$

Figure 8.18. Improved state assignment for the sequential circuit in Figure 8.12.

Derivation of next-state expressions



w \ y_2y_1	00	01	11	10
0	0	1	0	0
1	1	1	0	0



w \ y_2y_1	00	01	11	10
0	0	1	1	0
1	0	1	1	0

Gray Code	Binary Code
$Y_1 = wy'_2 + y_2y'_1$	$Y_1 = wy'_1 + y_2y'_1$
$Y_2 = y_1$	$Y_2 = y'_2y_1 + y_2y'_1$

Figure 8.19. Derivation of next-state expressions for the sequential circuit in Figure 8.18.

One-Shot Encoding

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

$$Y_1 = w'$$

$$Y_2 = y_1 w$$

$$Y_3 = y_1' w$$

$$z = y_3$$

Figure 8.20. One-hot state assignment for the sequential circuit in Figure 8.4.

One-Shot Encoding

Y_3

$y_1w \backslash y_3y_2$	00	01	11	10
00	d	0	d	0
01	d	1	d	1
11	0	d	d	d
10	0	d	d	d

$y_1'w$

Y_2

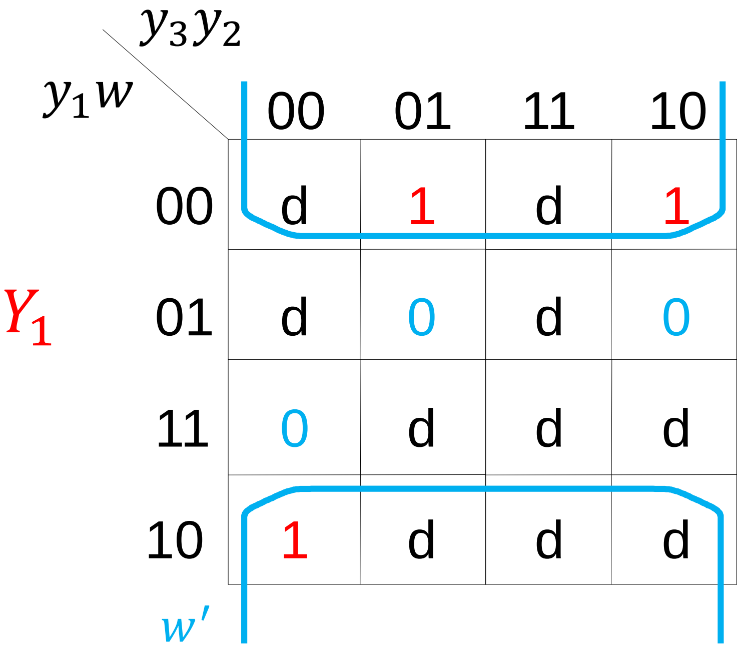
$y_1w \backslash y_3y_2$	00	01	11	10
00	d	0	d	0
01	d	0	d	0
11	1	d	d	d
10	0	d	d	d

y_1w

$$Y_3 = y_1'w$$

$$Y_2 = y_1w$$

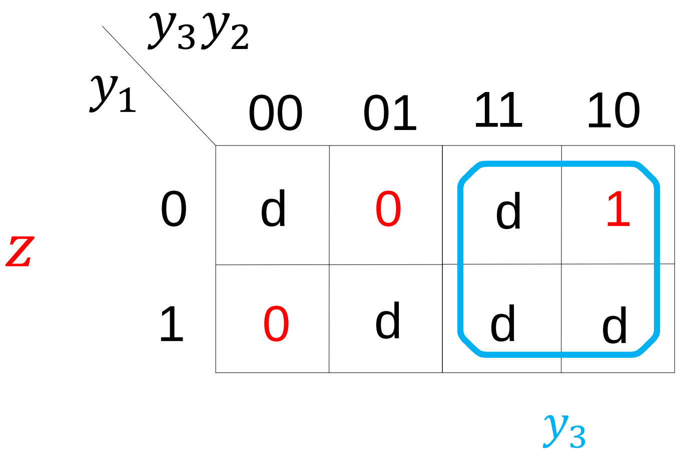
One-Shot Encoding



Y_1

y_1w \ y_3y_2	00	01	11	10
00	d	1	d	1
01	d	0	d	0
11	0	d	d	d
10	1	d	d	d

$$Y_1 = w'$$



z

y_1 \ y_3y_2	00	01	11	10
0	d	0	d	1
1	0	d	d	d

$$z = y_3$$

Example 8.3

	Present state	Next state		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 0 0 1	0 0 0 1	0 0 1 0	0	0	0	0	0	0	0
B	0 0 1 0	0 1 0 0	0 1 0 0	0	0	1	0	0	1	0
C	0 1 0 0	1 0 0 0	1 0 0 0	1	0	0	1	0	0	0
D	1 0 0 0	0 0 0 1	0 0 0 1	0	1	0	0	1	0	1

$$R2_{out} = R3_{in} = y_2$$

$$R3_{out} = R1_{in} = \mathbf{y_3}$$

$$R1_{out} = R2_{in} = Done = \mathbf{y_4}$$

Figure 8.21. One-hot state assignment for the sequential circuit in Figure 8.12.

One-hot state assignment for the sequential circuit in Figure 8.12

Y_4

y_2y_1 \ y_4y_3	00	01	11	10
00	d	1	d	0
01	0	d	d	d
11	d	d	d	d
10	0	d	d	d

$w = 0$

y_3

y_2y_1 \ y_4y_3	00	01	11	10
00	d	1	d	0
01	0	d	d	d
11	d	d	d	d
10	0	d	d	d

$w = 1$

$$Y_4 = y_3$$

One-hot state assignment for the sequential circuit in Figure 8.12

Y_3

y_2y_1 \ y_4y_3	00	01	11	10
00	d	0	d	0
01	0	d	d	d
11	d	d	d	d
10	1	d	d	d

$w = 0$

y_2

y_2y_1 \ y_4y_3	00	01	11	10
00	d	0	d	0
01	0	d	d	d
11	d	d	d	d
10	1	d	d	d

$w = 1$

$$Y_3 = y_2$$

One-hot state assignment for the sequential circuit in Figure 8.12

Y_2

y_2y_1 \ y_4y_3	00	01	11	10
00	d	0	d	0
01	0	d	d	d
11	d	d	d	d
10	0	d	d	d

$w = 0$

y_2y_1 \ y_4y_3	00	01	11	10
00	d	0	d	0
01	1	d	d	d
11	d	d	d	d
10	0	d	d	d

y_2w

$w = 1$

$$Y_2 = y_2w$$

One-hot state assignment for the sequential circuit in Figure 8.12

Y_1

y_2y_1 \ y_4y_3	00	01	11	10
00	d	0	d	1
01	1	d	d	d
11	d	d	d	d
10	0	d	d	d

$w = 0$

y_4

y_1w'

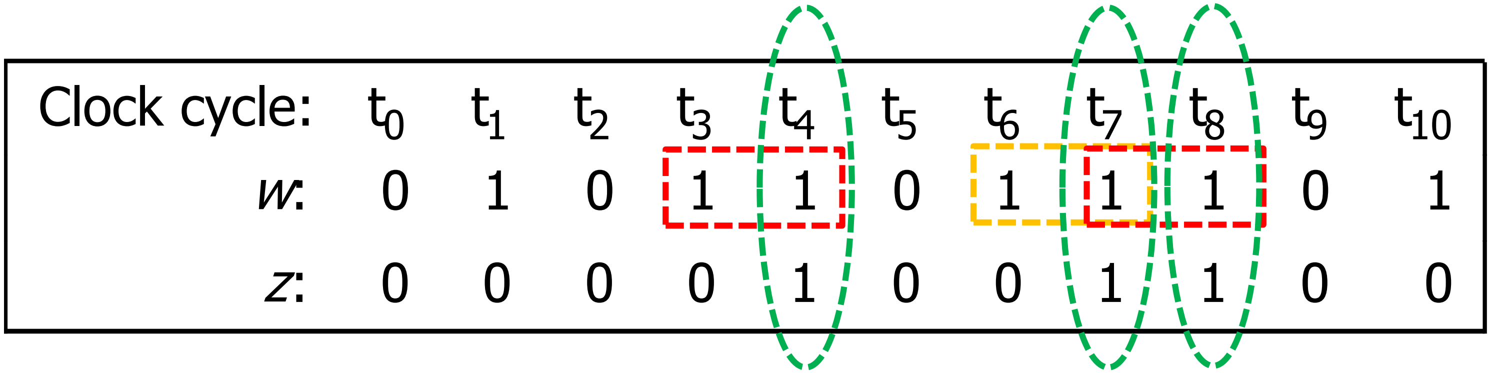
y_2y_1 \ y_4y_3	00	01	11	10
00	d	0	d	1
01	0	d	d	d
11	d	d	d	d
10	0	d	d	d

$w = 1$

$$Y_1 = y_4 + y_1w'$$

MEALY STATE MODEL

Mealy State Model



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 8.22. Sequences of input and output signals.

State diagram of an FSM

$input/output: w = 1/z = 0$

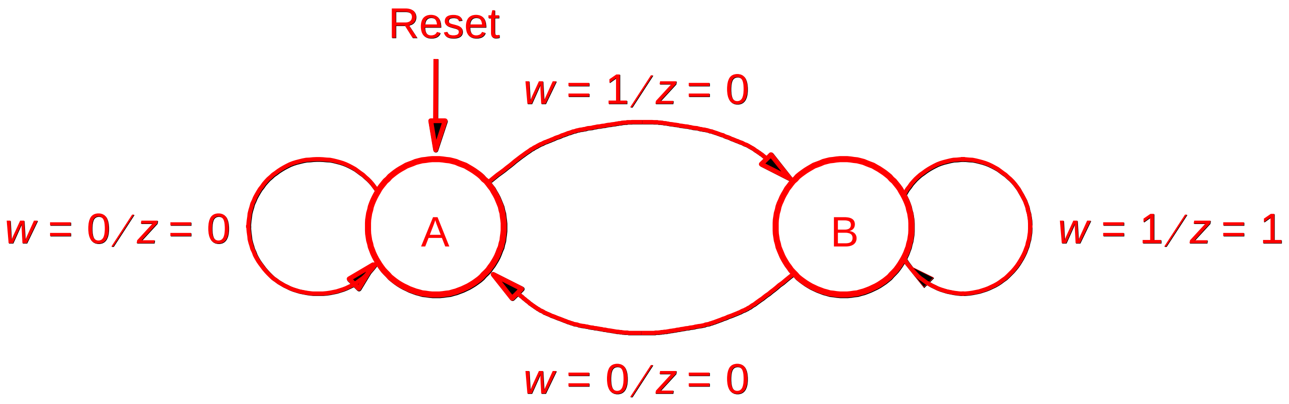
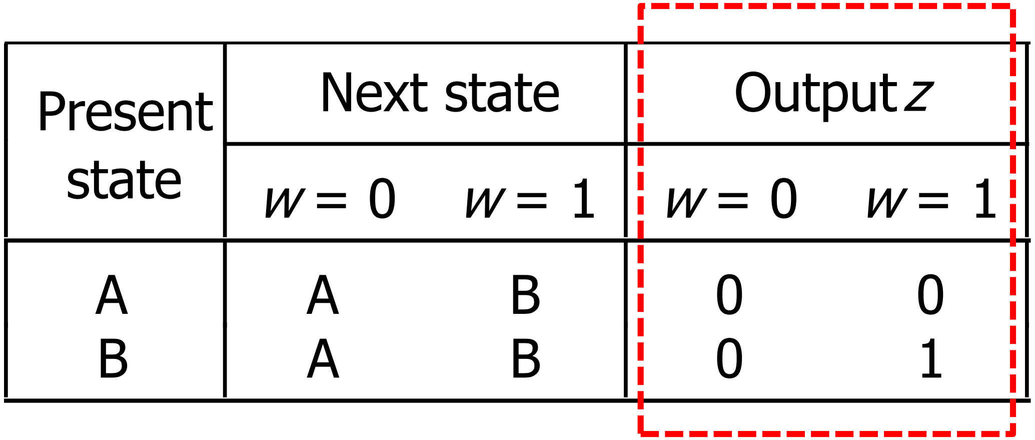


Figure 8.23. State diagram of an FSM that realizes the task in Figure 8.22.

State table for the FSM



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

Different from Moore type model

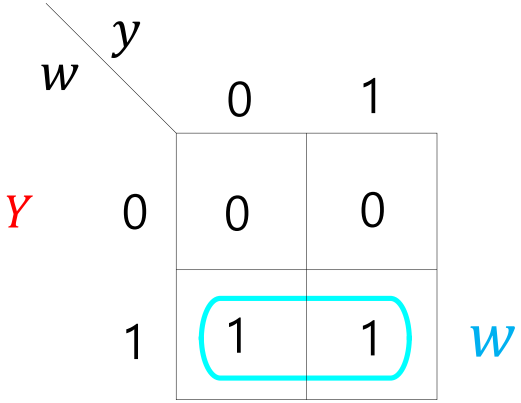
Figure 8.24. State table for the FSM in Figure 8.23.

State table for the FSM

	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

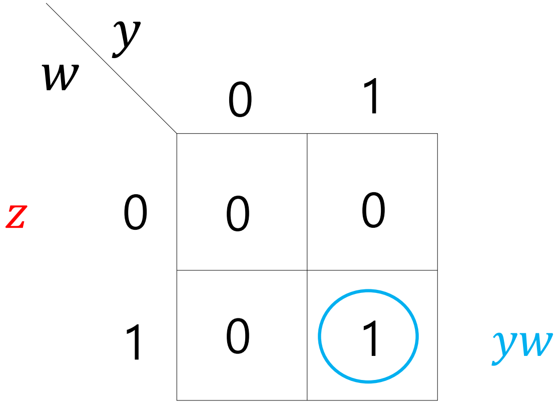
$$Y = w$$

$$z = yw$$



Y	$w \backslash y$	0	1
	0	0	0
	1	1	1

w

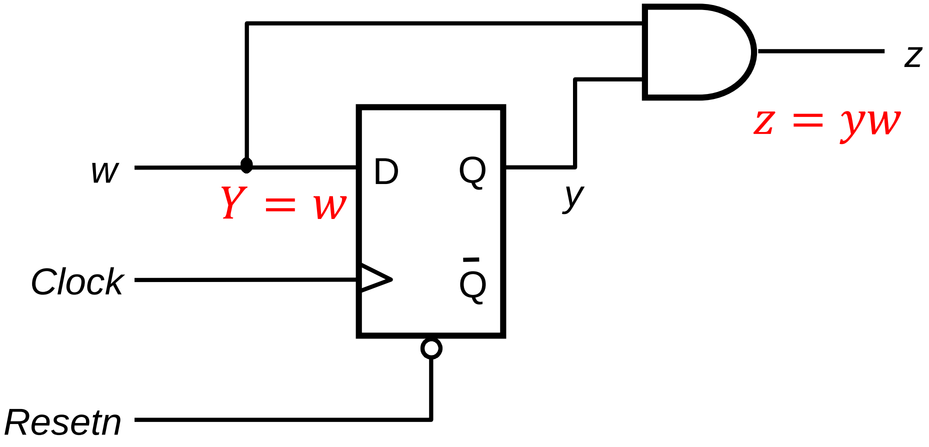


z	$w \backslash y$	0	1
	0	0	0
	1	0	1

yw

Figure 8.25. State-assigned table for the FSM in Figure 8.24.

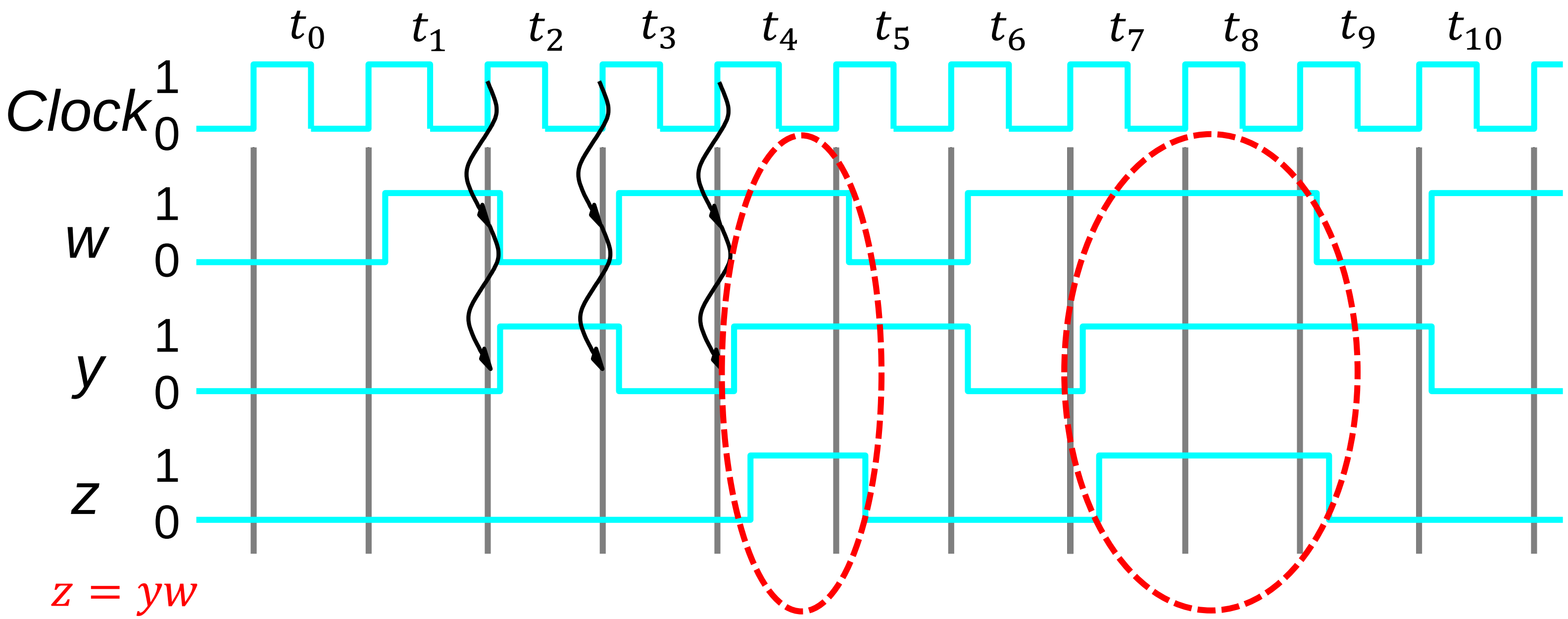
Implementation of FSM in Figure 8.25



(a) Circuit

Figure 8.26. Implementation of FSM in Figure 8.25.

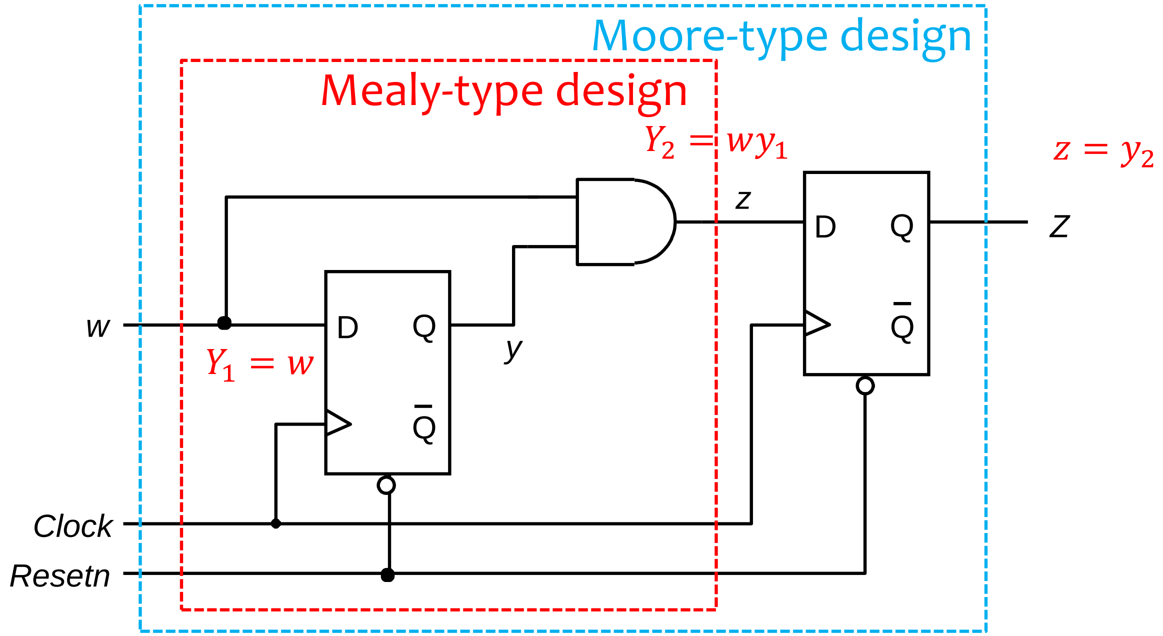
Implementation of FSM in Figure 8.25



(b) Timing diagram

Figure 8.26. Implementation of FSM in Figure 8.25.

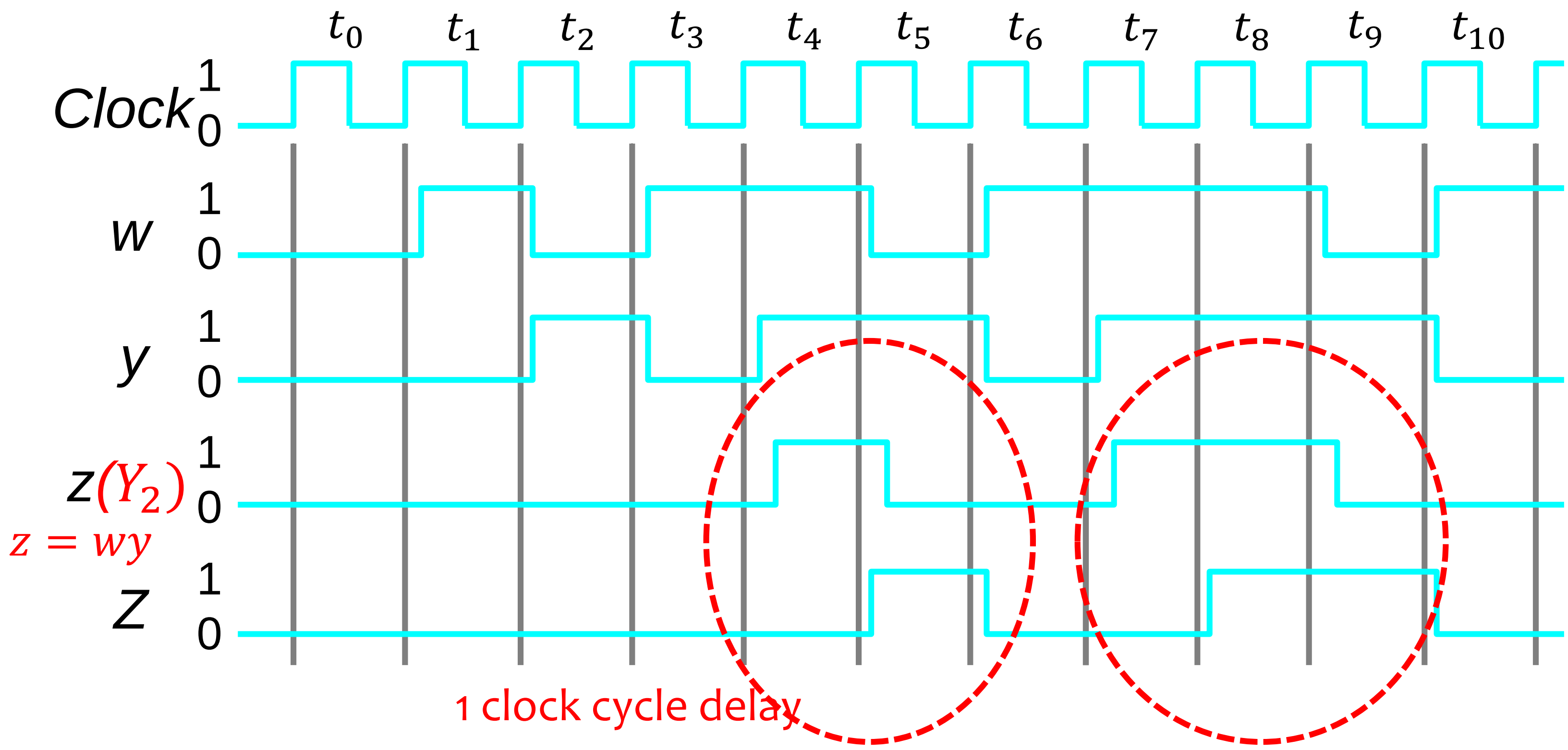
Circuit that implements the specification in Figure 8.2



(a) Circuit

Figure 8.27. Circuit that implements the specification in Figure 8.2.

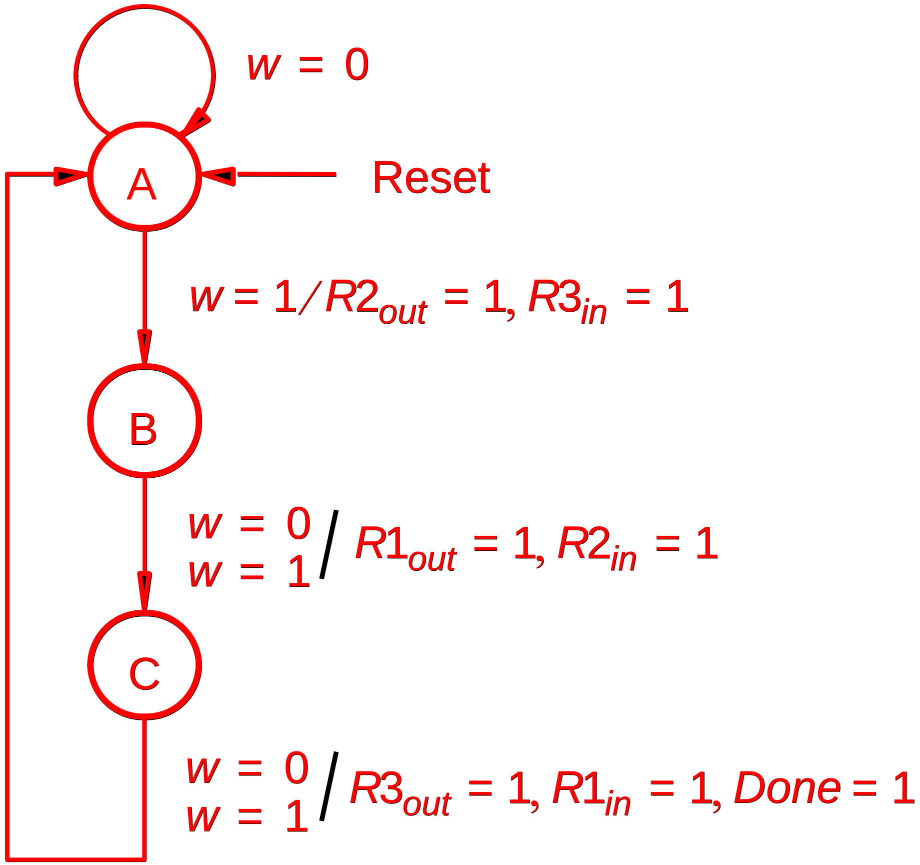
Circuit that implements the specification in Figure 8.2



(b) Timing diagram

Figure 8.27. Circuit that implements the specification in Figure 8.2.

State diagram for Example 8.4



$w = 0$
Reset
A
$w = 1 / R2_{out} = 1, R3_{in} = 1$
B
$w = 0$
$w = 1$
$/ R1_{out} = 1, R2_{in} = 1$
C
$w = 0$
$w = 1$
$/ R3_{out} = 1, R1_{in} = 1, Done = 1$

Figure 8.28. State diagram for Example 8.4.

State assignment for the sequential circuit in Figure 8.28

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

$$R2_{out} = R3_{in} = y_1'w$$

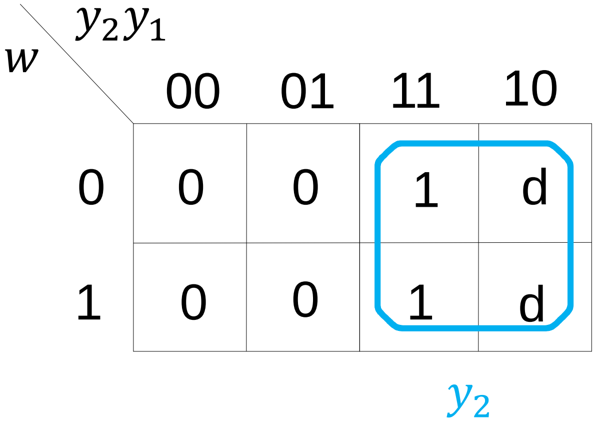
$$R1_{out} = R2_{in} = y_2'y_1$$

$$R3_{out} = R1_{in} = Done = y_2$$

Figure 8.18. Improved state assignment for the sequential circuit in Figure 8.12.

Karnaugh map for Figure 8.28

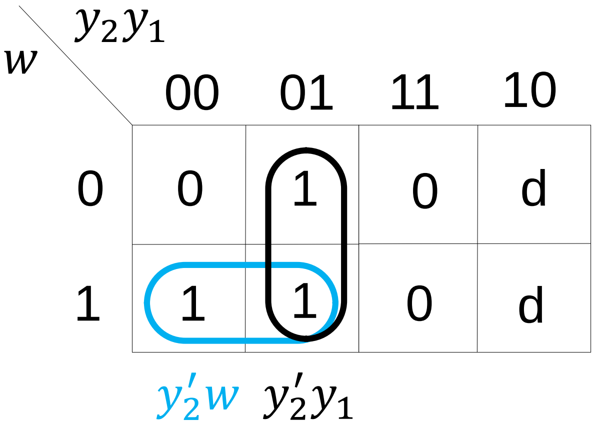
Y_2



w \ y_2y_1	00	01	11	10
0	0	0	1	d
1	0	0	1	d

$$Y_2 = y_2$$

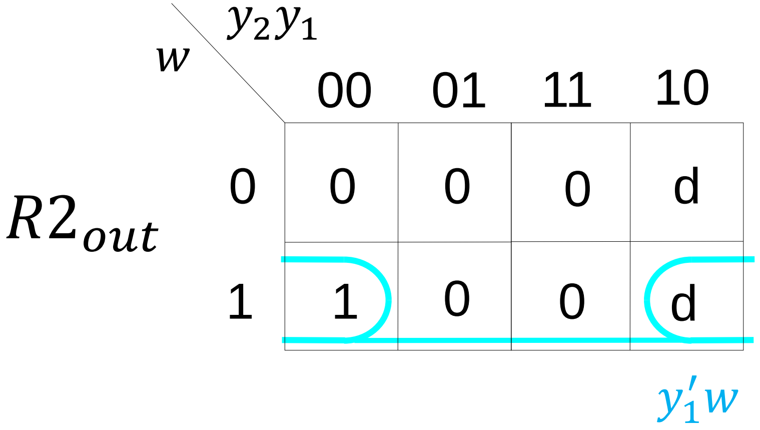
Y_1



w \ y_2y_1	00	01	11	10
0	0	1	0	d
1	1	1	0	d

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

Karnaugh map for Figure 8.28

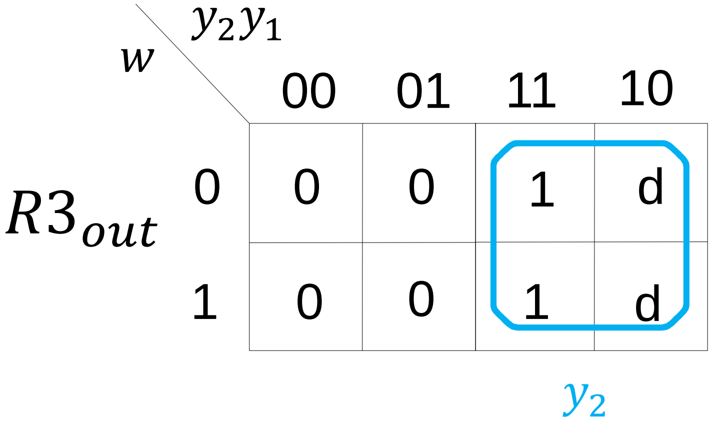


$R2_{out}$

$w \backslash y_2y_1$	00	01	11	10
0	0	0	0	d
1	1	0	0	d

$y_1'w$

$$R2_{out} = y_1'w$$

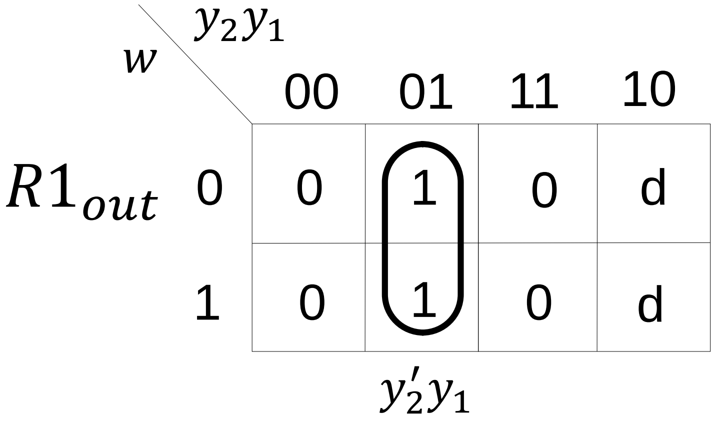


$R3_{out}$

$w \backslash y_2y_1$	00	01	11	10
0	0	0	1	d
1	0	0	1	d

y_2

$$R3_{out} = y_2$$



$R1_{out}$

$w \backslash y_2y_1$	00	01	11	10
0	0	1	0	d
1	0	1	0	d

$y_2'y_1$

$$R1_{out} = y_2'y_1$$

SERIAL ADDER EXAMPLE

Block diagram for the serial adder

$A = a_3a_2a_1a_0$

$B = b_3b_2b_1b_0$

In Full Adder

$s = a \oplus b \oplus c_i$

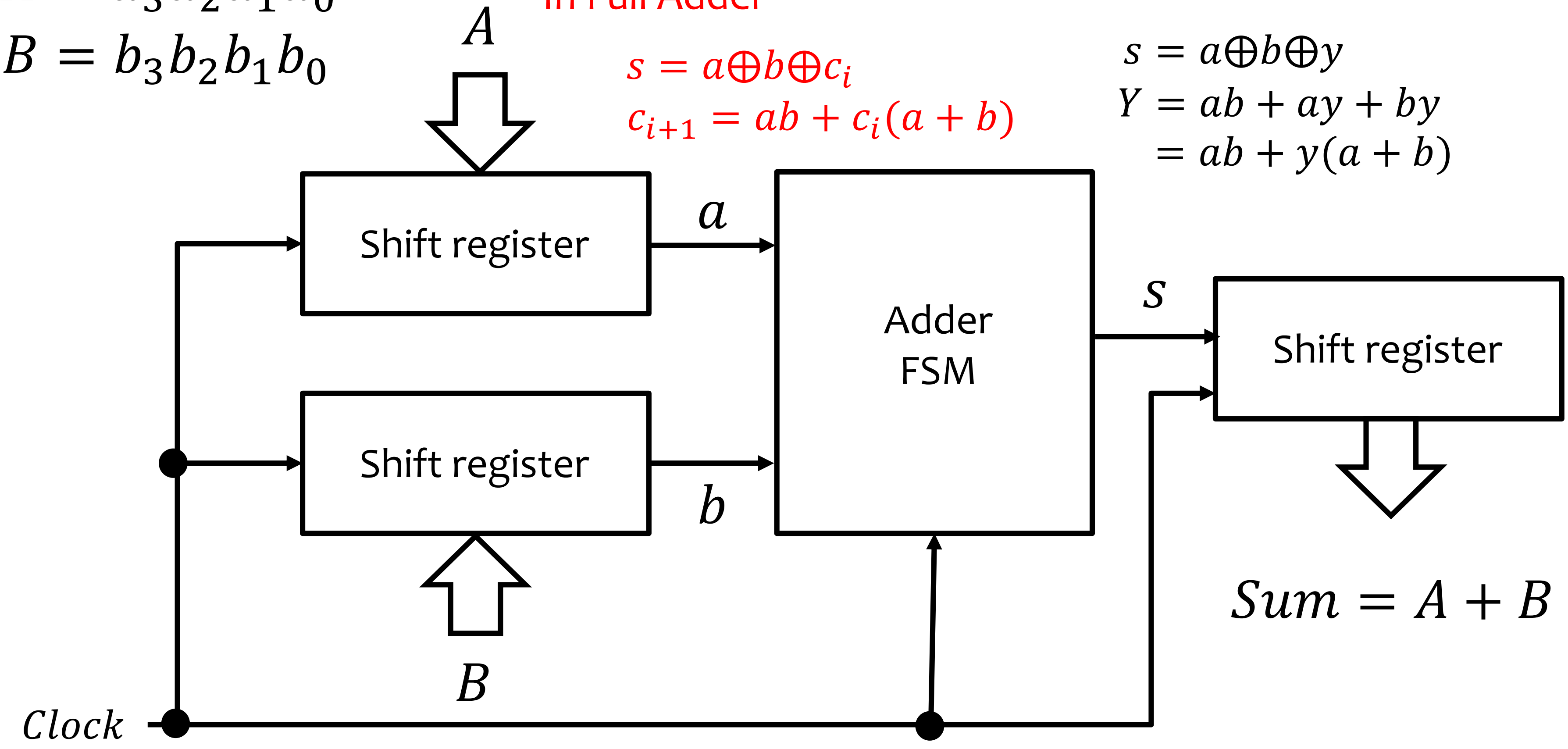
$c_{i+1} = ab + c_i(a+b)$

In adder FSM

$s = a \oplus b \oplus y$

$Y = ab + ay + by$

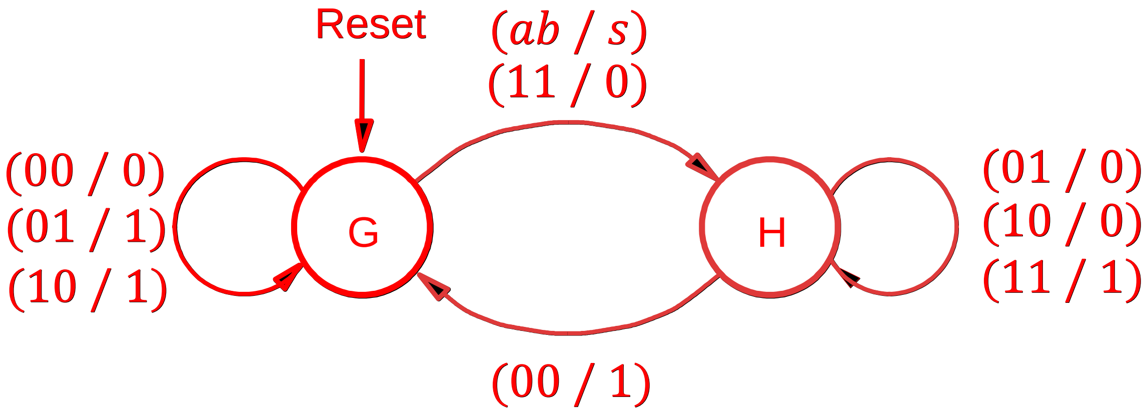
$= ab + y(a+b)$



$Sum = A + B$

Figure 8.39. Block diagram for the serial adder.

State diagram for the serial adder FSM

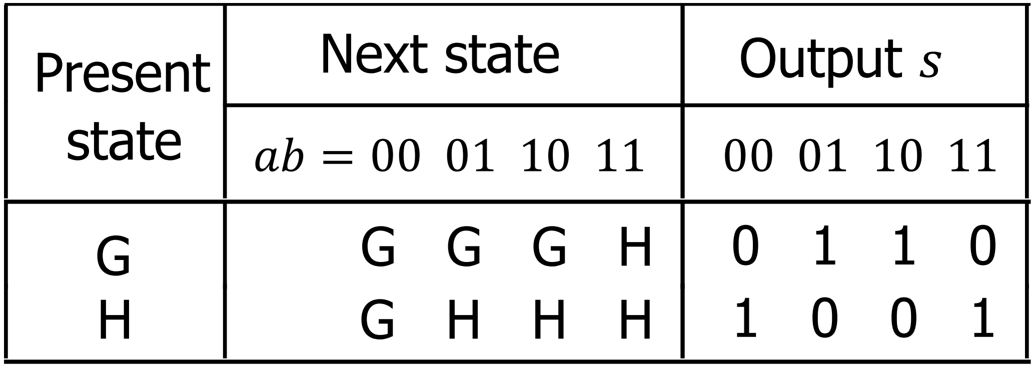


G : carry-in $= 0$ $(y = 0)$

H : carry-in $= 1$ $(y = 1)$

Figure 8.40. State diagram for the serial adder FSM.

State table 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

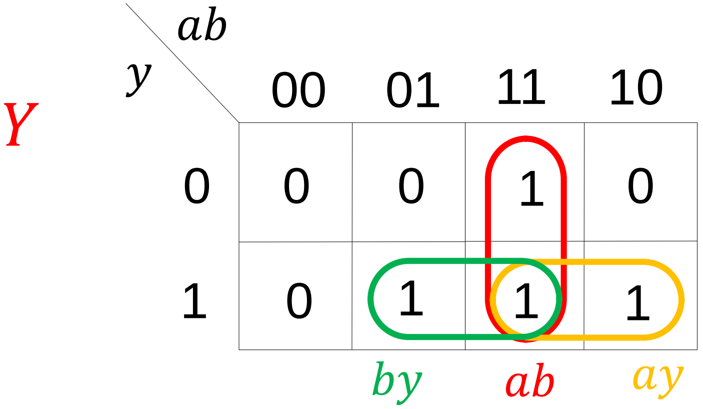
Figure 8.41. State table for the serial adder FSM.

State-assigned table for Figure 8.41

Present state	Next state				Output			
	$ab = 00$	01	10	11	00	01	10	11
$y\ (c_i)$	$Y\ (c_{i+1})$				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Figure 8.42. State-assigned table for Figure 8.41.

Karnaugh Map for State Table



Y

y \ ab	00	01	11	10
0	0	0	1	0
1	0	1	1	1

s

y \ ab	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Output of Full Adder

$$
\begin{aligned}
Y &= ab + ay + by \\
&= ab + y(a + b)
\end{aligned}
$$

$$s = a \oplus b \oplus y$$

Circuit for the adder FSM for Figure 8.39

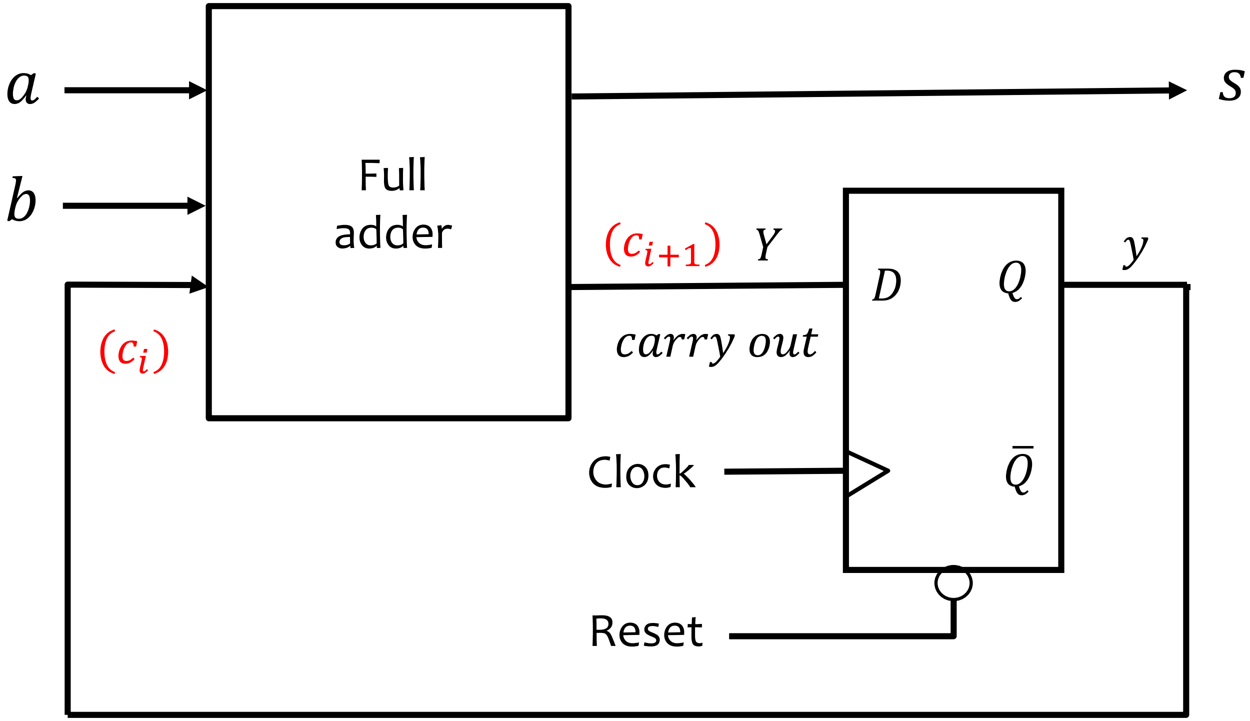


Figure 8.43. Circuit for the adder FSM for Figure 8.39.

State diagram for the serial adder FSM

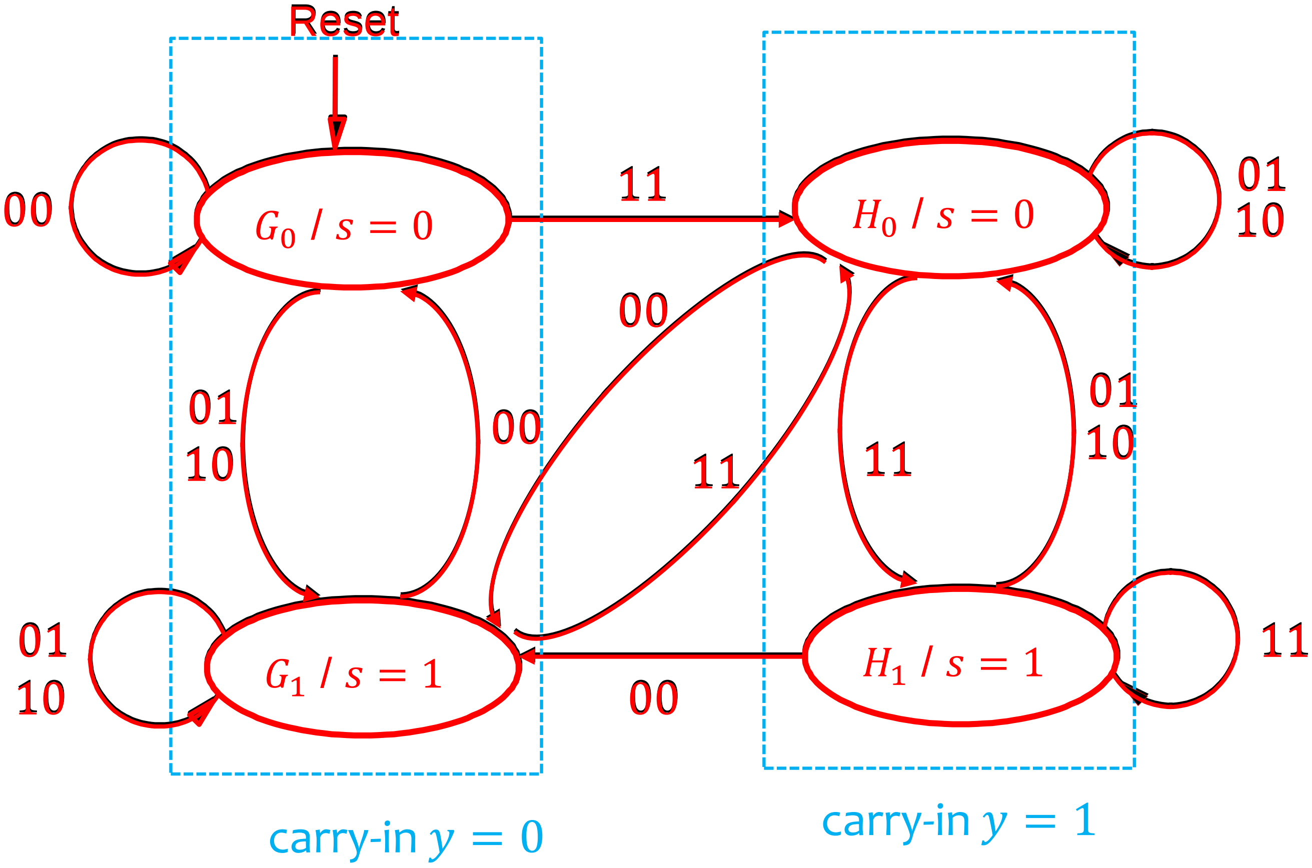
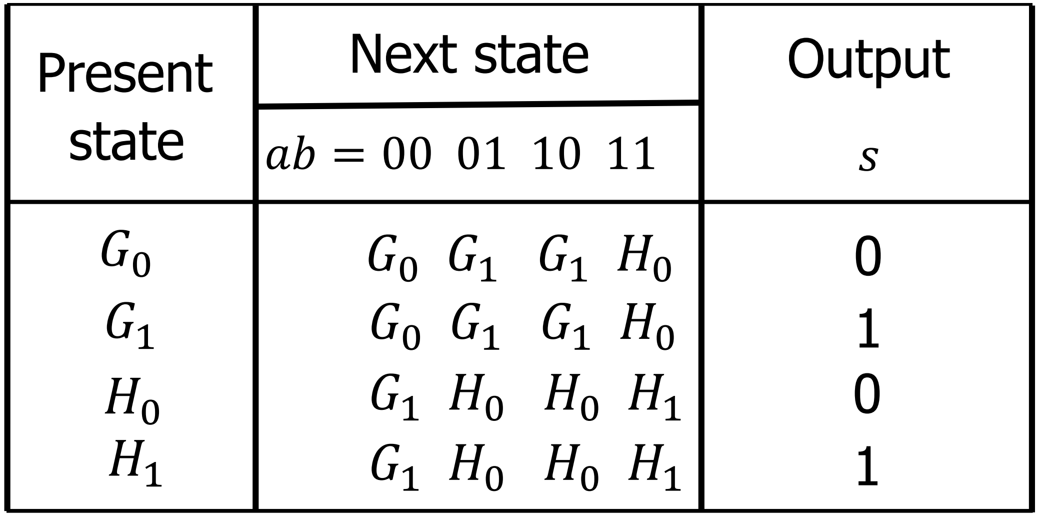


Figure 8.44. State diagram for the Moore-type serial adder FSM.

State table for the serial adder FSM



Present state	Next state				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 8.45. State table for the Moore-type serial adder FSM.

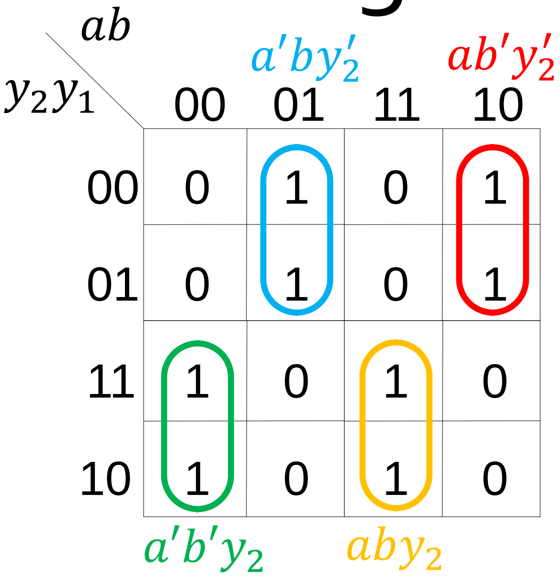
State-assigned table for Figure 8.45

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

Figure 8.46. State-assigned table for Figure 8.45.

Karnaugh Map for State Table

Y_1

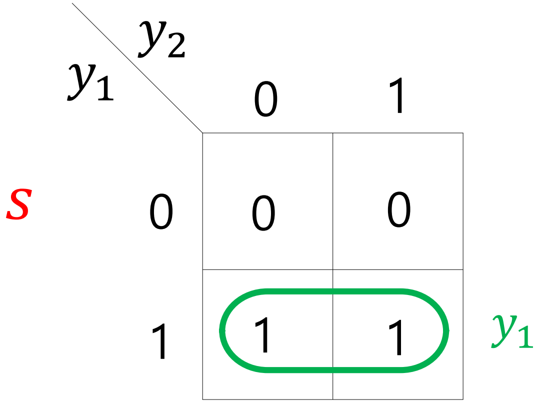


y_2y_1 \ ab	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	1	0	1	0
10	1	0	1	0

$a'by_2'$ $ab'y_2'$

$a'b'y_2$ aby_2

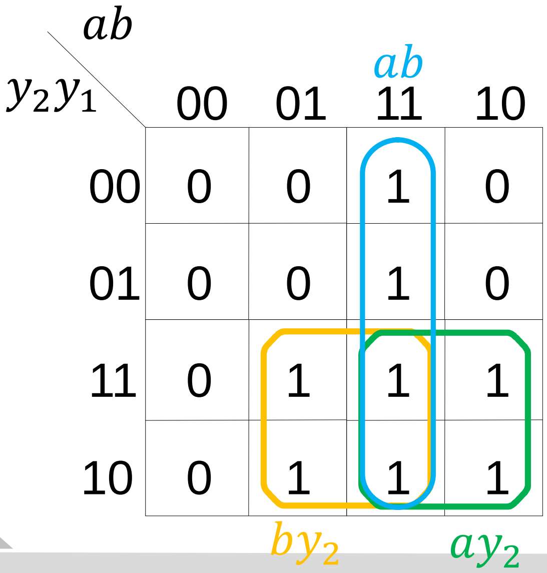
s



y_1 \ y_2	0	1
0	0	0
1	1	1

y_1

Y_2



y_2y_1 \ ab	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	0	1	1	1
10	0	1	1	1

ab

by_2 ay_2

$$Y_1 = a \oplus b \oplus y_2$$

$$Y_2 = ab + ay_2 + by_2$$

$$s = y_1$$

$$\begin{aligned} Y_1 &= a'by_2' + a'b'y_2 + ab'y_2' + aby_2 \\ &= a'(by_2' + b'y_2) + a(b'y_2' + by_2) \\ &= a'(b \oplus y_2) + a(b \oplus y_2)' \\ &= a \oplus b \oplus y_2 \end{aligned}$$

State-assigned table for Figure 8.41

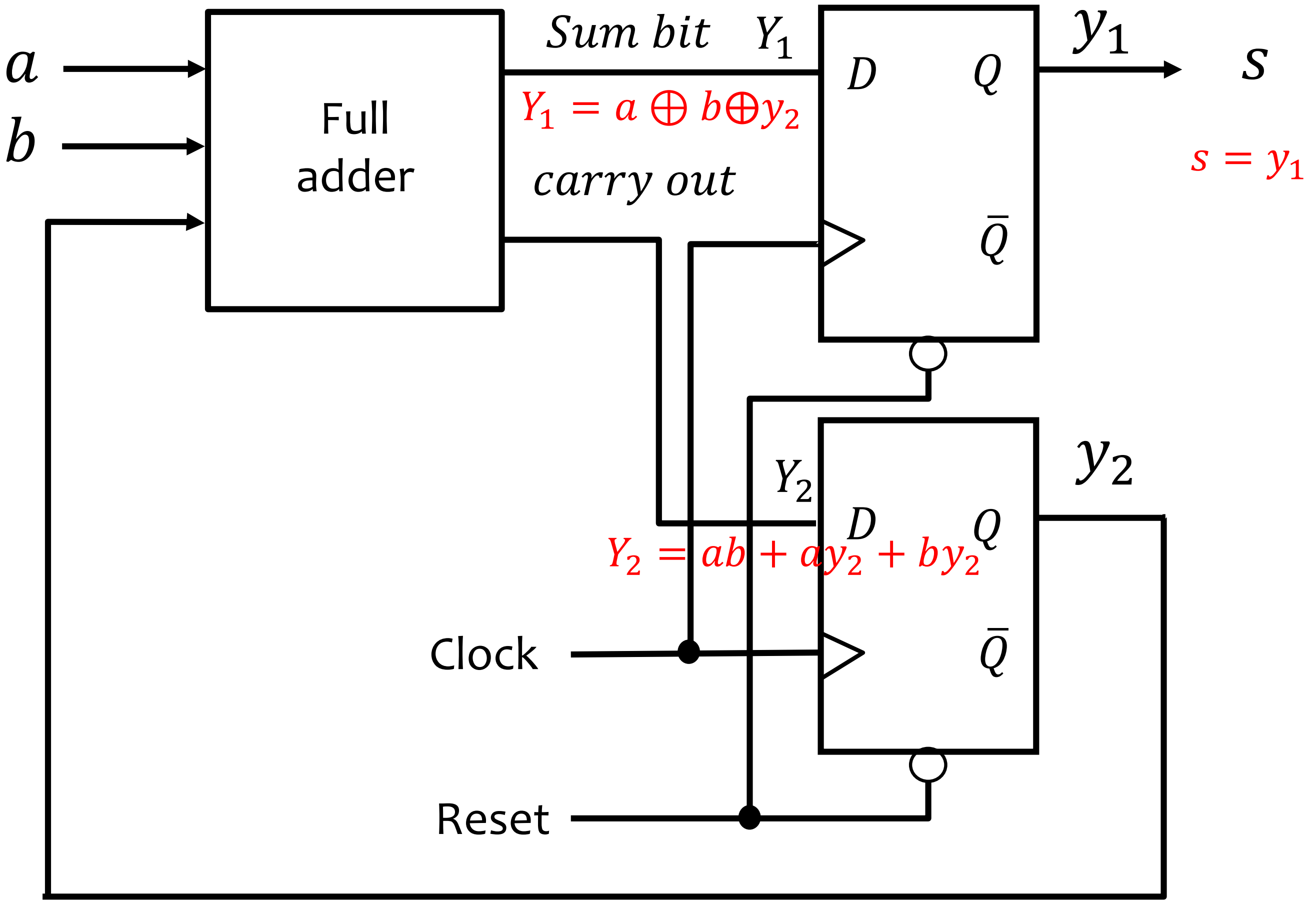


Figure 8.47. Circuit for the the Moore-type serial adder FSM.

STATE MINIMIZATION

State Minimization

- If the number of states in an FSM can be reduced, then some states in the original design must be equivalent to other states in their contribution to the overall behavior of the FSM.
- Definition 8.1 – *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.*
- Definition 8.2 – *A partition consists of one or more blocks, where each block comprises a subset of states that may be equivalent, but the states in a given block are definitely not equivalent to the states in other blocks*.

State table for Example 8.5

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

P_1=(ABCDEFG)

P_2=(ABD)(CEFG) depends on the value of output z

Group I Group II

Figure 8.51. State table for Example 8.5.

State table for Example 8.5

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

$P2=\underbrace{(ABD)}_{\text{Group I}}\underbrace{(CEFG)}_{\text{Group II}}$ depends on the value of output z

Figure 8.51. State table for Example 8.5.

State table for Example 8.5

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

$$P3=\underbrace{(ABD)}_{\text{Group I}}\underbrace{(CEG)}_{\text{Group II}}\underbrace{(F)}_{\text{Group III}}$$

depends on 0-successors and 1-successors

Figure 8.51. State table for Example 8.5.

State table for Example 8.5

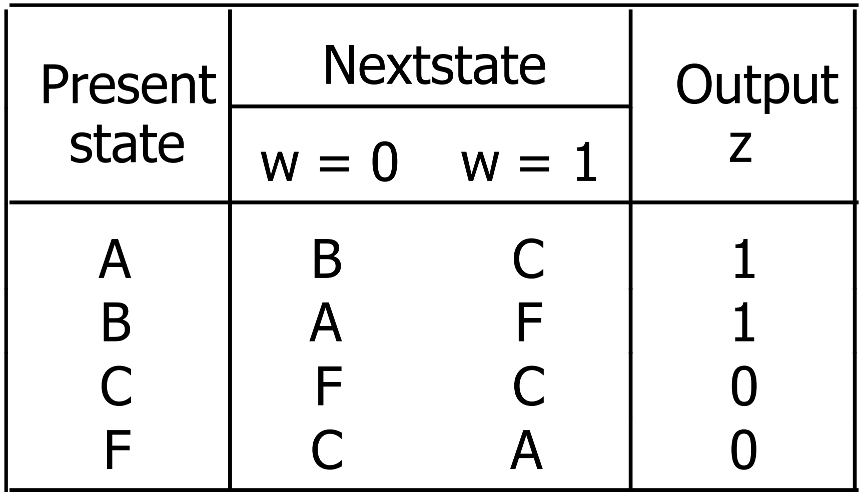
Present state	Next state		Output
	$w = 0$	$w = 1$	z
A Group I	B Group II	C Group III	1
B Group II	D Group I	F Group IV	1
C Group III	F Group IV	E Group III	0
D Group I	B Group II	G Group III	1
E Group III	F Group IV	C Group III	0
F Group IV	E Group III	D Group I	0
G Group III	F Group IV	G Group III	0

P4=(AD)(B)(CEG)(F) depends on 0-successors and 1-successors

Group I Group II Group III Group IV

Figure 8.51. State table for Example 8.5.

Minimized state table for Example 8.5



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 8.52. Minimized state table for Example 8.5.

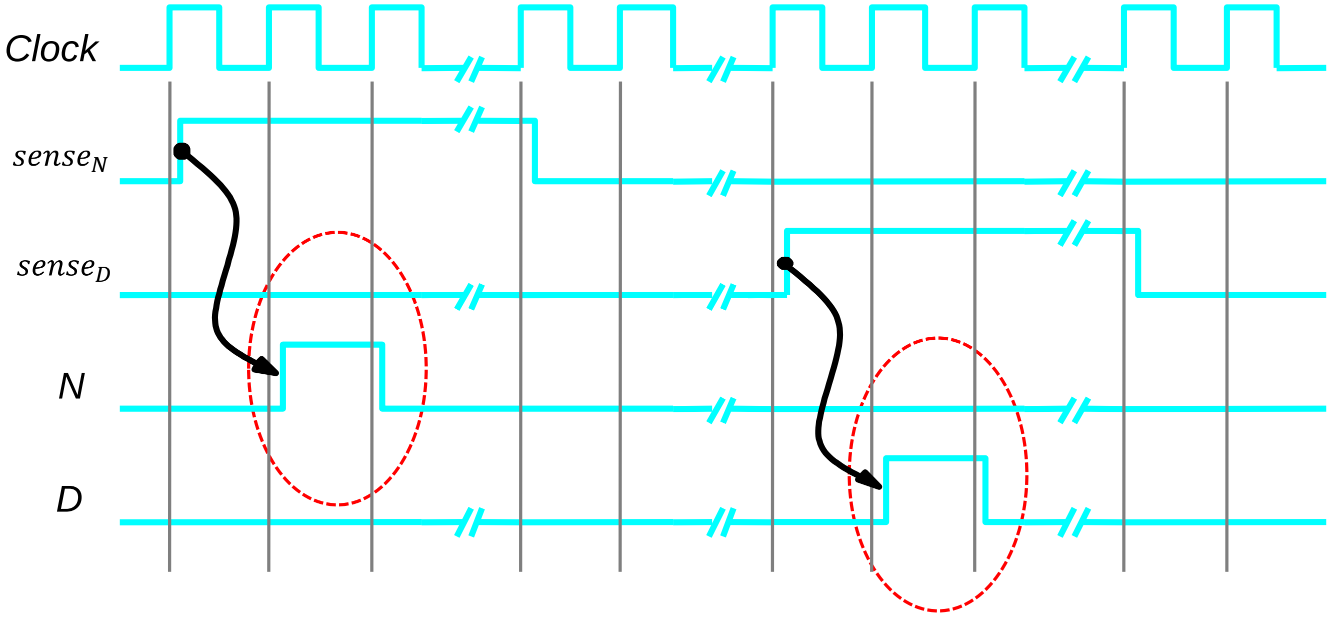
Example 8.5

- State Table in Figure 8.51
- To minimize the number of states
    - Initial partition: P1=(ABCDEFG)
    - New partition: P2=(ABD)(CEFG)
        - Depends on the value of output z
    - New partition: P3=(ABD)(CEG)(F)
        - Depends on 0-successors and 1-successors
    - New partition: P4=(AD)(B)(CEG)(F)
        - Repeats the process on 0-successors and 1-successors

Example 8.6

- Another example of minimization
- A coin-operated vending machine
  - The machine accepts nickels and dimes
  - It takes 15 cents for a piece of candy
  - It 20 cents is deposited, the machine will not return the change, but it will credit the buyer with 5 cents and wait for the buyer to make a second purchase.

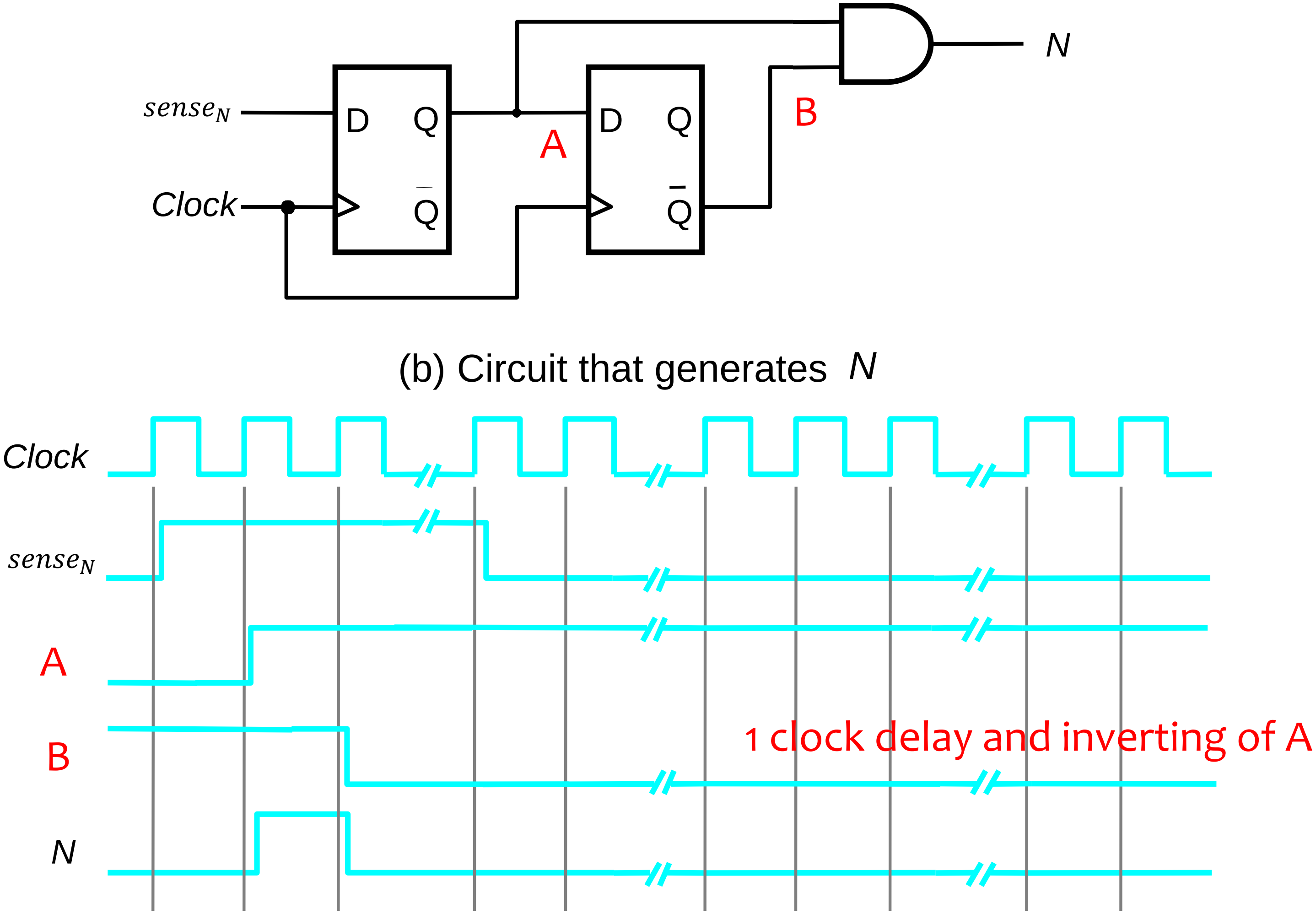
Signals for the vending machine



(a) Timing diagram

Figure 8.53. Signals for the vending machine.

Signals for the vending machine

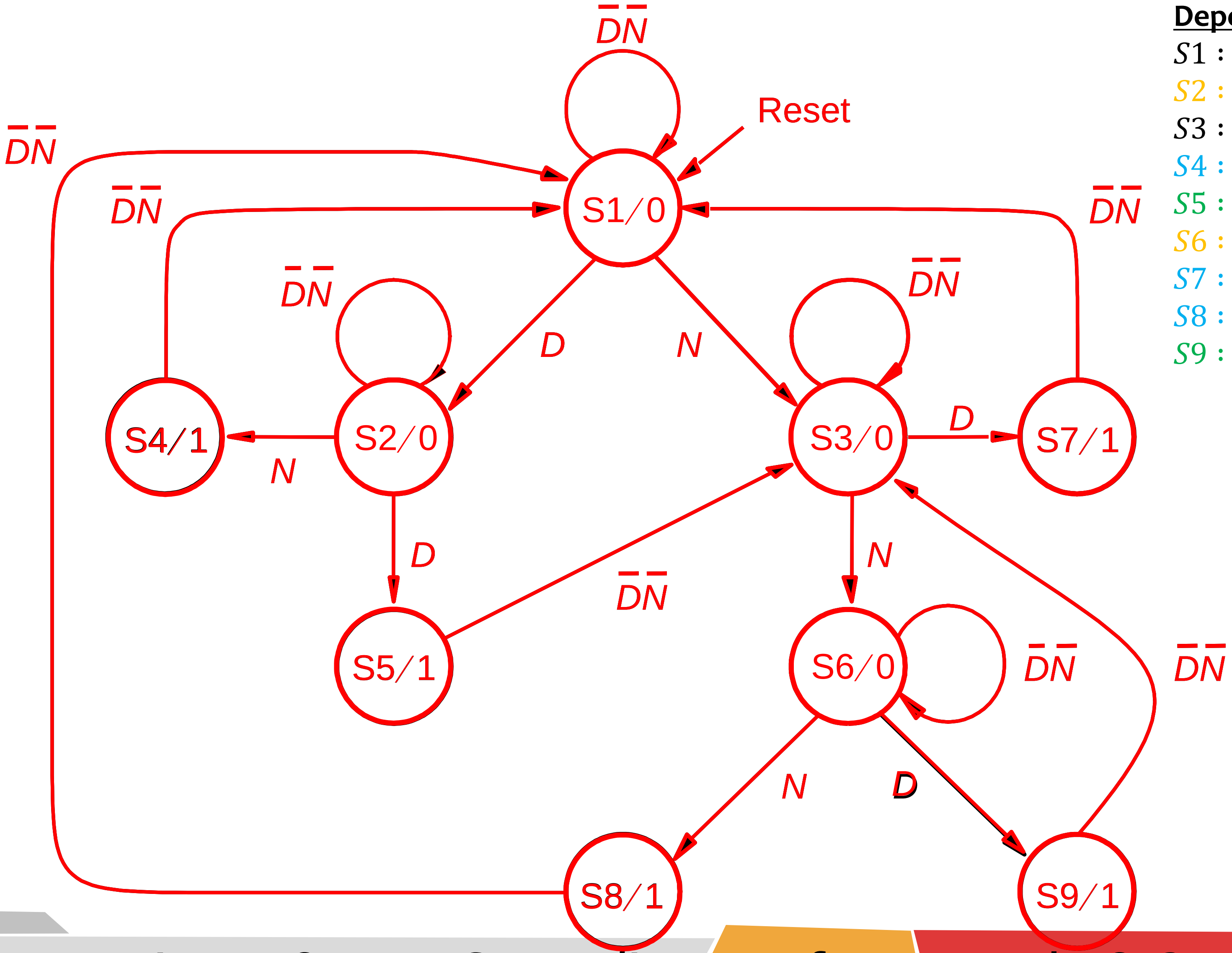


(b) Circuit that generates N

1 clock delay and inverting of A

Figure 8.53. Signals for the vending machine.

State diagram for Example 8.6



Deposited money

$S1 : 0\ cent$
$S2 : 10\ cent$
$S3 : 5\ cent$
$S4 : 15\ cent$
$S5 : 20\ cent$
$S6 : 10\ cent$
$S7 : 15\ cent$
$S8 : 15\ cent$
$S9 : 20\ cent$

Figure 8.54. State diagram for Example 8.6.

State table for Example 8.6

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

P1=(S1, S2, S3, S4, S5, S6, S7, S8, S9)

Figure 8.55. State table for Example 8.6.

State table for Example 8.6

Present state	Next state				Output
	DN =00	01	10	11	*z*
S1 g1	S1 g1	S3 g1	S2 g1	–	0
S2 g1	S2 g1	S4 g2	S5 g2	–	0
S3 g1	S3 g1	S6 g1	S7 g2	–	0
S4 g2	S1 g1	–	–	–	1
S5 g2	S3 g1	–	–	–	1
S6 g1	S6 g1	S8 g2	S9 g2	–	0
S7 g2	S1 g1	–	–	–	1
S8 g2	S1 g1	–	–	–	1
S9 g2	S3 g1	–	–	–	1

$P2=\underset{g1}{(S1, S2, S3, S6)}\underset{g2}{(S4, S5, S7, S8, S9)}$ depends on the value of output z

Figure 8.55. State table for Example 8.6.

State table for Example 8.6

Present state	Next state				Output *z*
	DN =00	01	10	11	
S1 g1	S1 g1	S3 g1	S2 g1	–	0
S2 g1	S2 g1	S4 g2	S5 g2	–	0
S3 g1	S3 g1	S6 g1	S7 g2	–	0
S4 g2	S1 g1	–	–	–	1
S5 g2	S3 g1	–	–	–	1
S6 g1	S6 g1	S8 g2	S9 g2	–	0
S7 g2	S1 g1	–	–	–	1
S8 g2	S1 g1	–	–	–	1
S9 g2	S3 g1	–	–	–	1

$P2=\underbrace{(S1, S2, S3, S6)}_{g1}\underbrace{(S4, S5, S7, S8, S9)}_{g2}$ depends on the value of output z

$P3=\underbrace{(S1)}_{g1}\underbrace{(S2 , S6)}_{g2}\underbrace{(S3)}_{g3}\underbrace{(S4, S5, S7, S8, S9)}_{g4}$ depends on DN-successors.

Figure 8.55. State table for Example 8.6.

State table for Example 8.6

Present state	Next state				Output *z*
	DN =00	01	10	11	
S1 g1	S1 g1	S3 g3	S2 g2	–	0
S2 g2	S2 g2	S4 g4	S5 g4	–	0
S3 g3	S3 g3	S6 g2	S7 g4	–	0
S4 g4	S1 g1	–	–	–	1
S5 g4	S3 g3	–	–	–	1
S6 g2	S6 g2	S8 g4	S9 g4	–	0
S7 g4	S1 g1	–	–	–	1
S8 g4	S1 g1	–	–	–	1
S9 g4	S3 g3	–	–	–	1

P3=(S1)(S2 , S6)(S3)(S4, S5, S7, S8, S9) depends on DN-successors.
g1 g2 g3 g4

P4=(S1)(S2 , S6)(S3)(S4, S7, S8)(S5, S9)depends on DN-successors.
g1 g2 g3 g4 g5

Figure 8.55. State table for Example 8.6.

State table for Example 8.6

Present state	Next state				Output z
	DN =00	01	10	11	
S1 g1	S1 g1	S3 g3	S2 g2	–	0
S2 g2	S2 g2	S4 g4	S5 g5	–	0
S3 g3	S3 g3	S6 g2	S7 g4	–	0
S4 g4	S1 g1	–	–	–	1
S5 g5	S3 g3	–	–	–	1
S6 g2	S6 g2	S8 g4	S9 g5	–	0
S7 g4	S1 g1	–	–	–	1
S8 g4	S1 g1	–	–	–	1
S9 g5	S3 g3	–	–	–	1

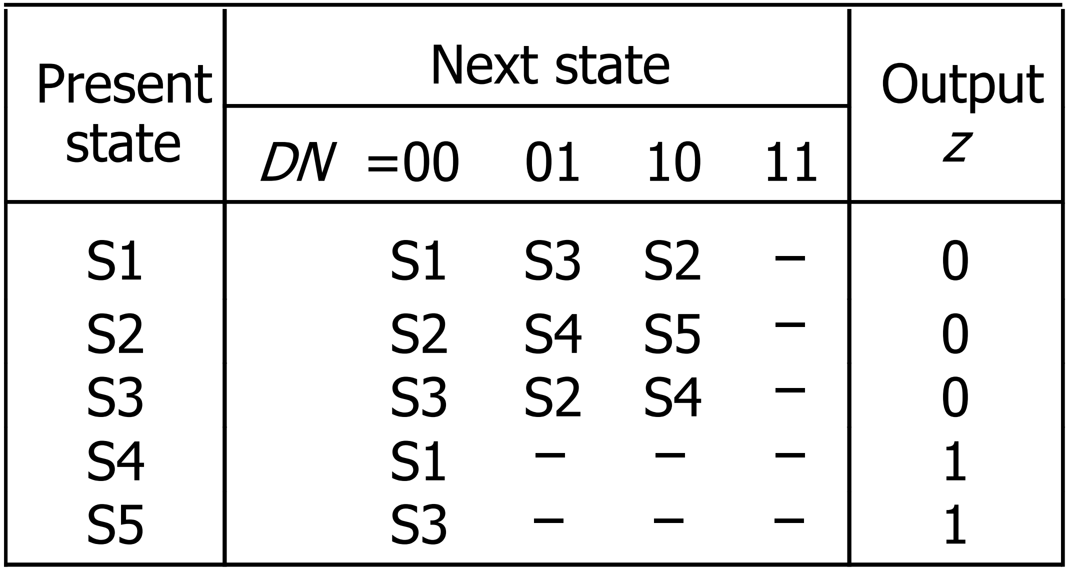
Deposited money

$S1$:	$0\ cent$
$S2$:	$10\ cent$
$S3$:	$5\ cent$
$S4$:	$15\ cent$
$S5$:	$20\ cent$
$S6$:	$10\ cent$
$S7$:	$15\ cent$
$S8$:	$15\ cent$
$S9$:	$20\ cent$

P4=$\underset{g1}{(S1)}\underset{g2}{(S2\ ,S6)}\underset{g3}{(S3)}\underset{g4}{(S4, S7, S8)}\underset{g5}{(S5, S9)}$ depends on DN-successors.

Figure 8.55. State table for Example 8.6.

Minimized state table for Example 8.6



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 8.56. Minimized state table for Example 8.6.

Minimized state diagram for Example 8.6

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

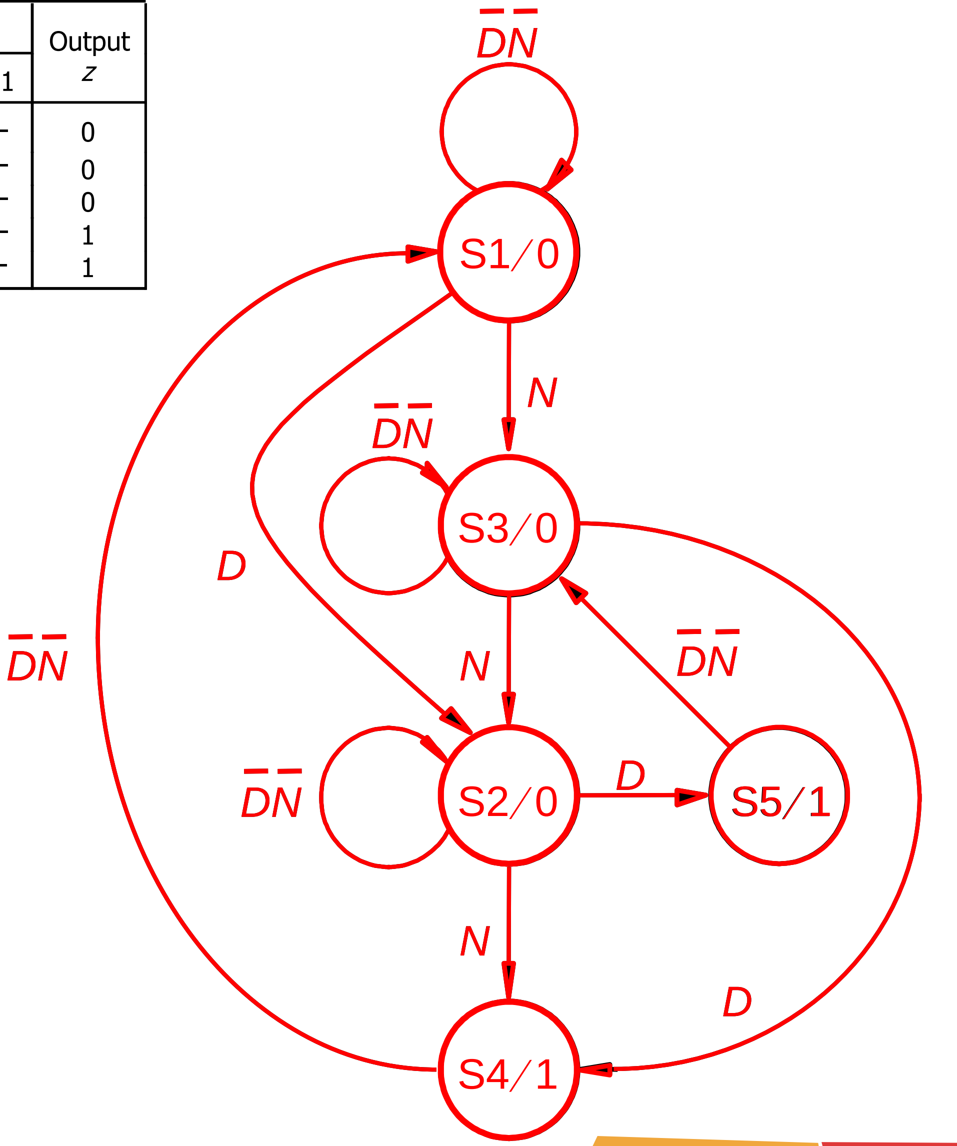


Figure 8.57. Minimized state diagram for Example 8.6.

Mealy type state table for Example 8.6

Deposit	Present state	Next state				Output z			
		DN =00	01	10	11	*DN* =00	01	10	11
0	S1	S1	S3	S2	–	0	0	0	–
10	S2	S2	S1	S3	–	0	1	1	–
5	S3	S3	S2	S1	–	0	0	1	–
15 → 0	S4	S1	–	–	–	–	–	–	–
20 → 5	S5	S3	–	–	–	–	–	–	–

discard

Figure 8.56. Mealy type state table for Example 8.6.

Mealy type state table for Example 8.6

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

Figure 8.56. Mealy type state table for Example 8.6.

Mealy-type FSM for Example 8.6

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

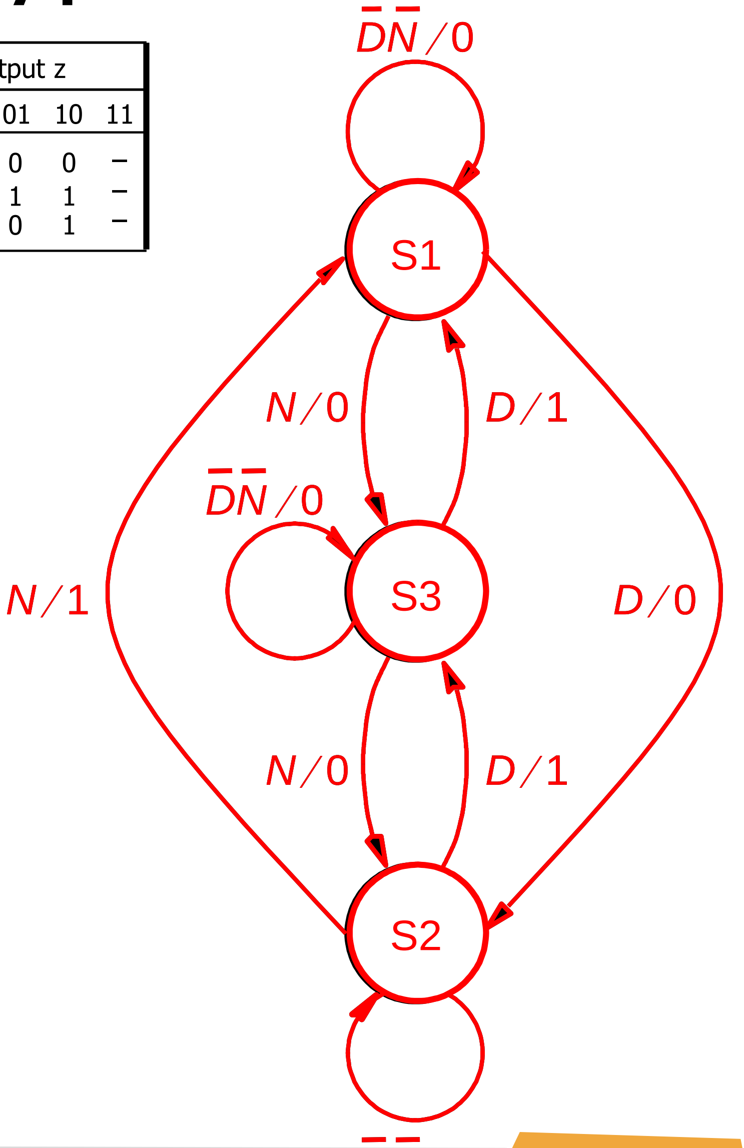
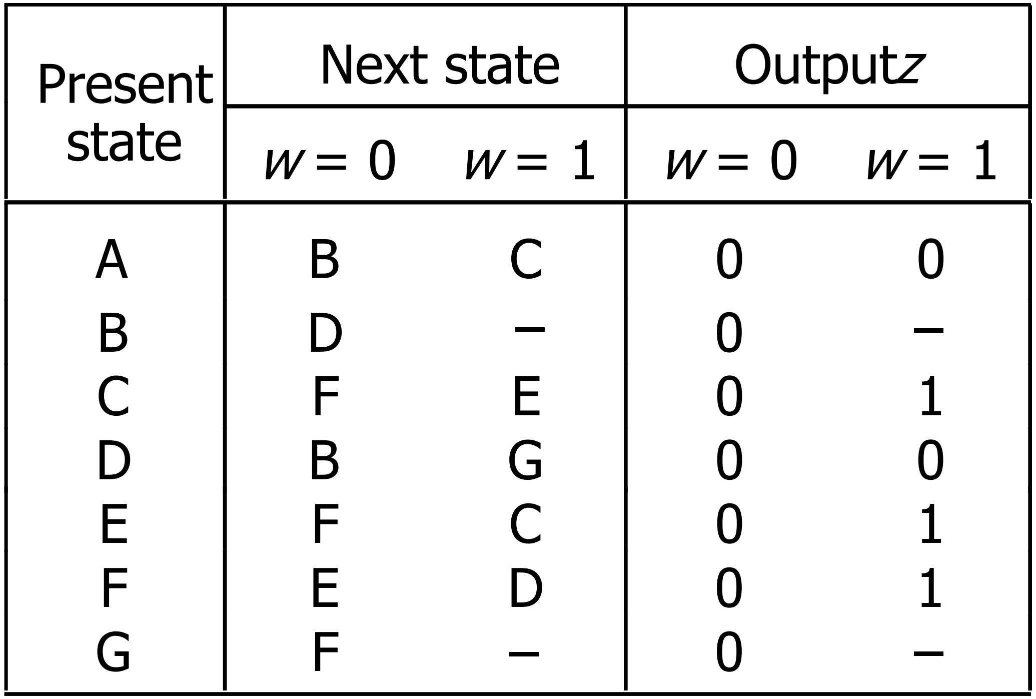

$\overline{D}\,\overline{N}/0$
S1
$N/0$
$D/1$
$\overline{D}\,\overline{N}/0$
$N/1$
S3
$D/0$
$N/0$
$D/1$
S2
$\overline{D}\,\overline{N}/0$

Figure 8.58. Mealy-type FSM for Example 8.6.

Incompletely Specified FSMs

Incompletely Specified FSMs: have don’t care terms in state table.



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	–

P1=(ABCDEFG)

Figure 8.59. Incompletely specified state table for Example 8.7

Incompletely Specified FSMs

Present state	Next state		Output z	
	w = 0	w = 1	w = 0	w = 1
A g1	B g1	C g2	0	0
B g1	D g1	–	0	0
C g2	F g2	E g2	0	1
D g1	B g1	G g1	0	0
E g2	F g2	C g2	0	1
F g2	E g2	D g1	0	1
G g1	F g2	–	0	0

$P2=\underset{g1}{(ABDG)}\underset{g2}{(CEF)}$ depends on the value of output z

Figure 8.59. Incompletely specified state table for Example 8.7.

Incompletely Specified FSMs

Present state	Next state		Output *z*	
	w = 0	*w* = 1	*w* = 0	*w* = 1
A g1	B g1	C g4	0	0
B g1	D g2	–	0	0
C g4	F g5	E g4	0	1
D g2	B g1	G g3	0	0
E g4	F g5	C g4	0	1
F g5	E g4	D g2	0	1
G g3	F g5	–	0	0

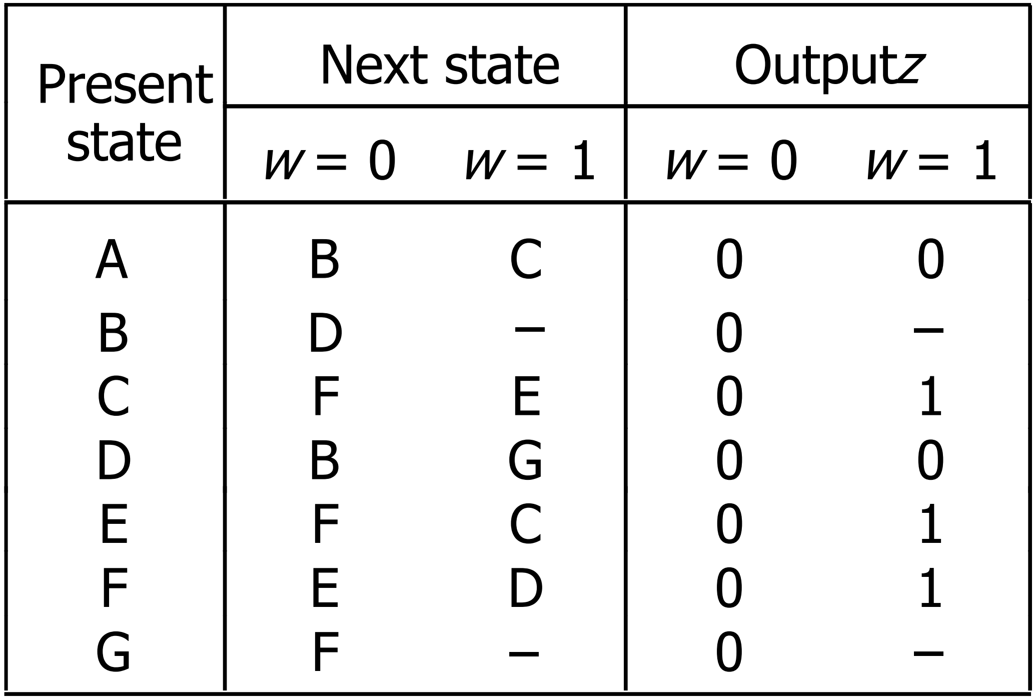
$P3=(\underset{g1}{AB})(\underset{g2}{D})(\underset{g3}{G})(\underset{g4}{CE})(\underset{g5}{F})$ depends on 0-successors and 1-successors

$P4=(A)(B)(D)(G)(CE)(F)$

Figure 8.59. Incompletely specified state table for Example 8.7.

Incompletely Specified FSMs

Incompletely Specified FSMs: have don’t care terms in state table.



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	–

P1=(ABCDEFG)

Figure 8.59. Incompletely specified state table for Example 8.7

Incompletely Specified FSMs

Present state	Next state		Output z	
	$w = 0$	$w = 1$	$w = 0$	$w = 1$
A g1	B g2	C g2	0	0
B g2	D g1	–	0	1
C g2	F g2	E g2	0	1
D g1	B g2	G g2	0	0
E g2	F g2	C g2	0	1
F g2	E g2	D g1	0	1
G g2	F g2	–	0	1

$P_2 = (\underset{g_1}{AD})(\underset{g_2}{BCEFG})$ depends on the value of output z

Figure 8.59. Incompletely specified state table for Example 8.7.

Incompletely Specified FSMs

Present state	Next state		Output *z*	
	w = 0	*w* = 1	*w* = 0	*w* = 1
A g1	B g2	C g3	0	0
B g2	D g1	–	0	1
C g3	F g4	E g3	0	1
D g1	B g2	G g3	0	0
E g3	F g4	C g3	0	1
F g4	E g3	D g1	0	1
G g3	F g4	–	0	1

$P3=\underset{g1}{(AD)}\underset{g2}{(B)}\ \underset{g3}{(CEG)}\ \underset{g4}{(F)}$ depends on 0-successors and 1-successors

Figure 8.59. Incompletely specified state table for Example 8.7.

Example 8.7

- Assuming the unspecified outputs have a value of 0
  - P1=(ABCDEFG), P2=(ABDG)(CEF)
  - P3=(AB)(D)(G)(CE)(F),P4=(A)(B)(D)(G)(CE)(F), P5=P4 ➔ 6 states
- Assuming the unspecified outputs have a value of 1
  - P1=(ABCDEFG), P2=(AD)(BCEFG)
  - P3=(AD)(B)(CEFG),P4=(AD)(B)(CEG)(F), P5=P4 ➔ 4 states

DESIGN OF A COUNTER USING THE SEQUENTIAL CIRCUIT APPROACH

State Diagram for a Modulo-8 Counter

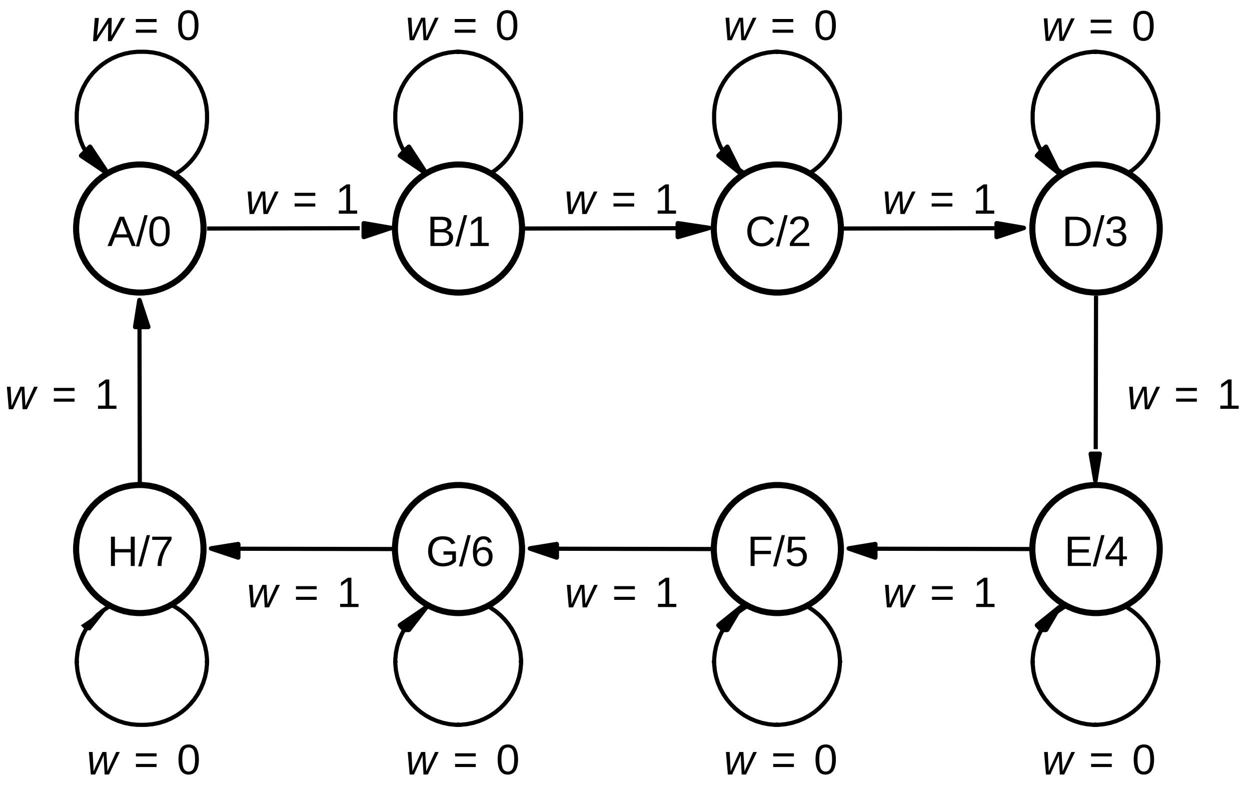
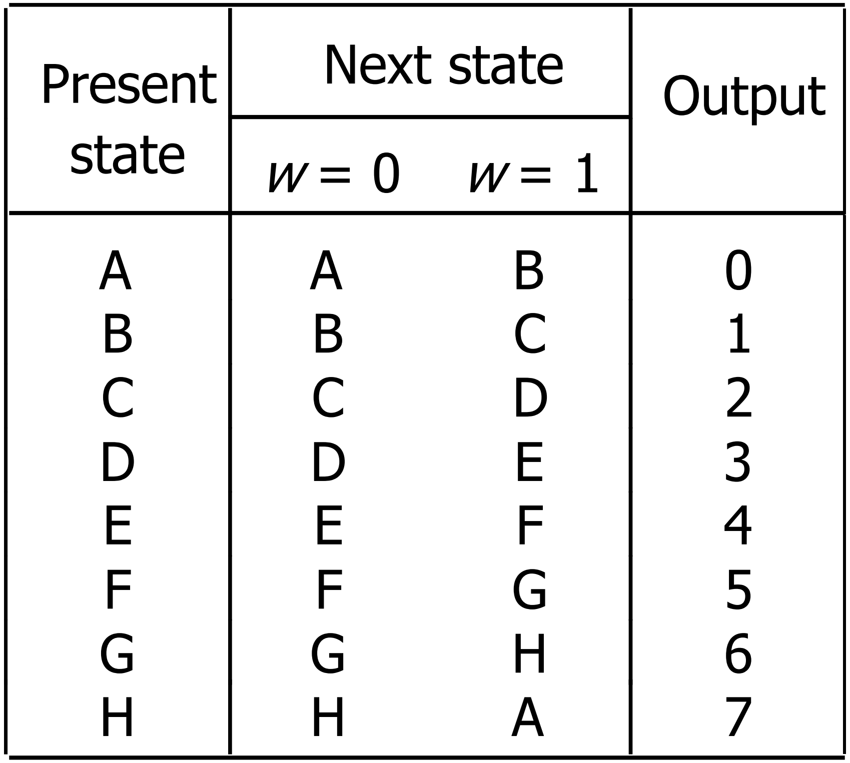


Figure 8.60. State diagram for the counter.

State table for a Modulo-8 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 8.61. State table for the counter.

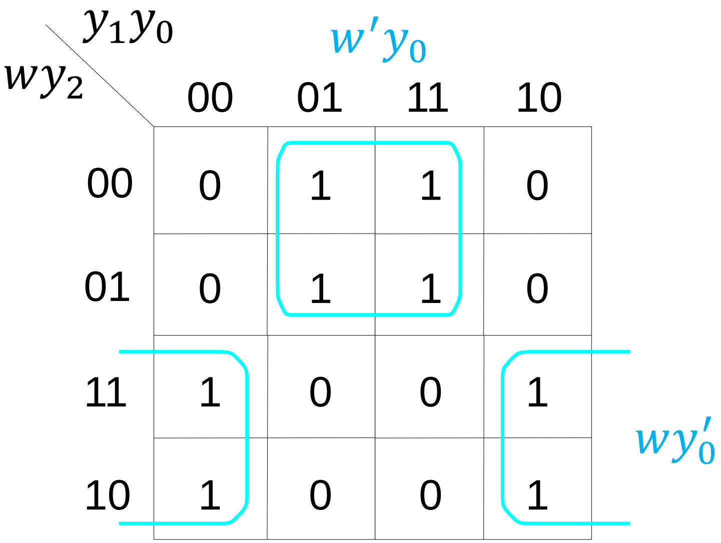
State-assigned table for the counter

Moore-type design

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

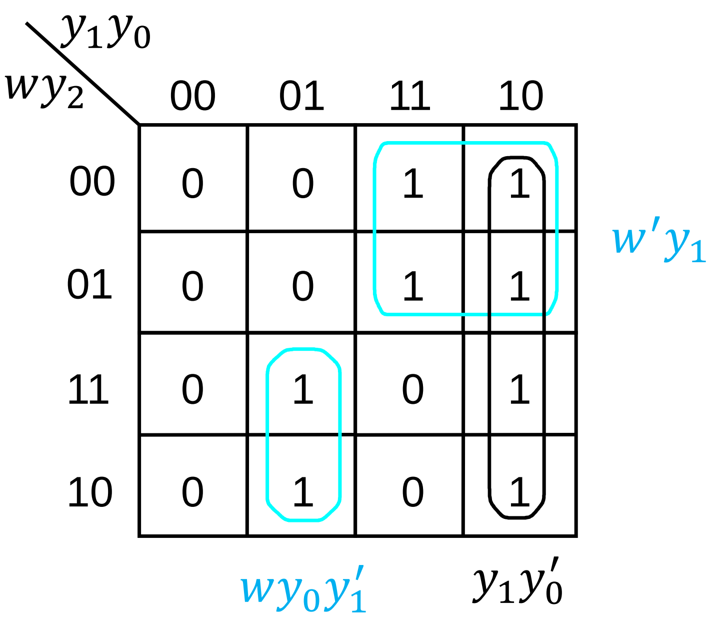
Figure 8.62. State-assigned table for the counter.

Implementation Using D-Type Flip-Flops



$wy_2 \backslash y_1y_0$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	0	0	1
10	1	0	0	1

$$Y_0 = w'y_0 + wy_0'$$
$$= w \oplus y_0$$

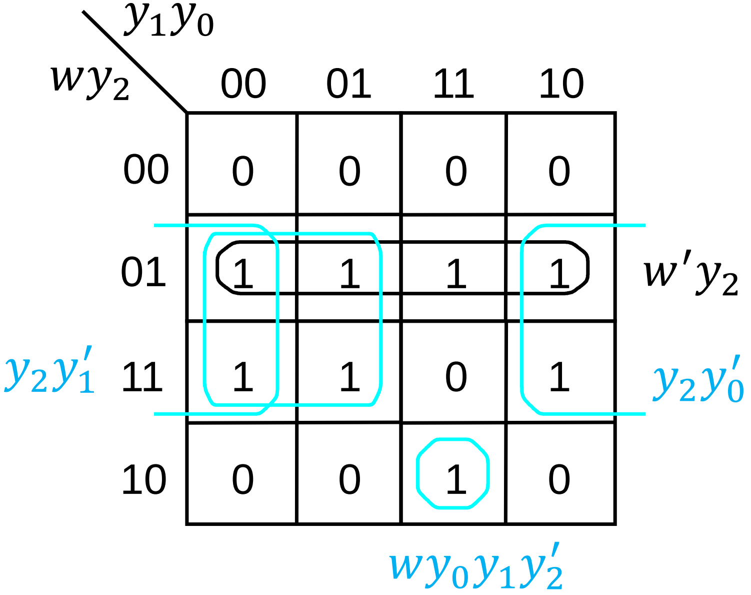


$wy_2 \backslash y_1y_0$	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	1	0	1
10	0	1	0	1

$$Y_1 = w'y_1 + y_1y_0' + wy_0y_1'$$
$$= (w' + y_0')y_1 + wy_0y_1'$$
$$= (wy_0)'y_1 + wy_0y_1'$$
$$= wy_0 \oplus y_1$$

Figure 8.63. Karnaugh maps for D flip-flops for the counter.

Implementation Using D-Type Flip-Flops



wy_2 \ y_1y_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	0	1
10	0	0	1	0

$$Y_2 = w'y_2 + y_2y_0' + y_2y_1' + wy_0y_1y_2'$$

$$= (w' + y_0' + y_1')y_2 + wy_0y_1y_2'$$

$$= (wy_0y_1)'y_2 + wy_0y_1y_2'$$

$$= wy_0y_1 \oplus y_2$$

Figure 8.63. Karnaugh maps for D flip-flops for the counter.

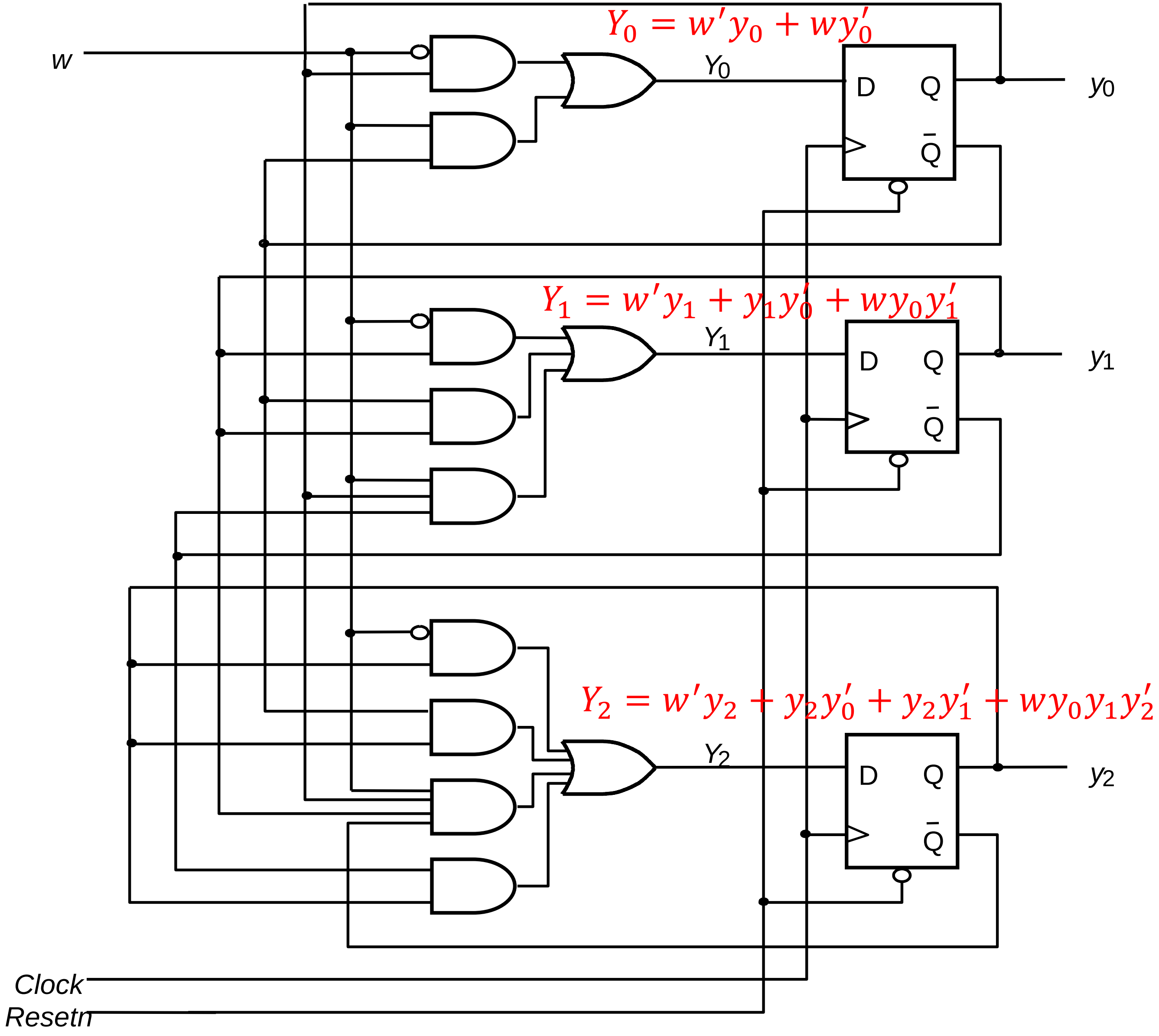


Figure 8.64. Circuit diagram for the counter implemented with D flip-flops.

Implementation Using JK-Type Flip-Flops

J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$Q'(t)$

Truth table

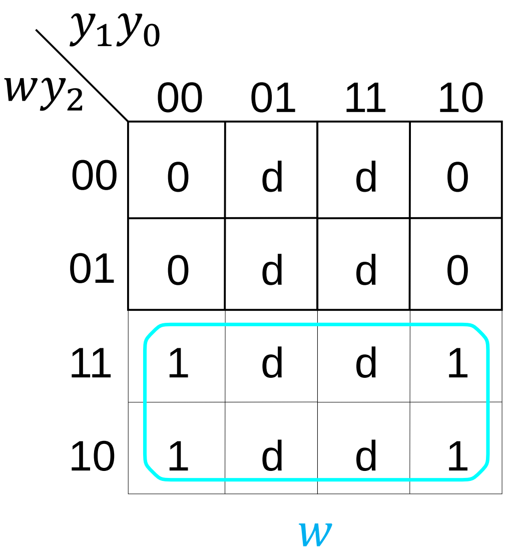
Present state $Q(t)=y$	Next state $Q(t+1)=Y$	J	K		
0	0	0 0	0 1	0	d
0	1	1 1	0 1	1	d
1	0	0 1	1 1	d	1
1	1	1 0	0 0	d	0

Implementation Using JK-Type Flip-Flops

	Present state $y_2 y_1 y_0$	Flip-flop inputs								Count $z_2 z_1 z_0$
		$w = 0$				$w = 1$				
		$Y_2 Y_1 Y_0$	$J_2 K_2$	$J_1 K_1$	$J_0 K_0$	$Y_2 Y_1 Y_0$	$J_2 K_2$	$J_1 K_1$	$J_0 K_0$	
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

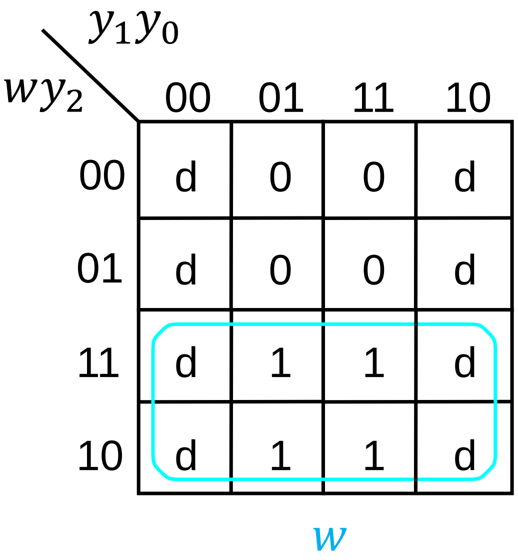
Figure 8.65. Excitation table for the counter with JK flip-flops.

Implementation Using JK-Type Flip-Flops



wy_2 \ y_1y_0	00	01	11	10
00	0	d	d	0
01	0	d	d	0
11	1	d	d	1
10	1	d	d	1

$$J_0 = w$$

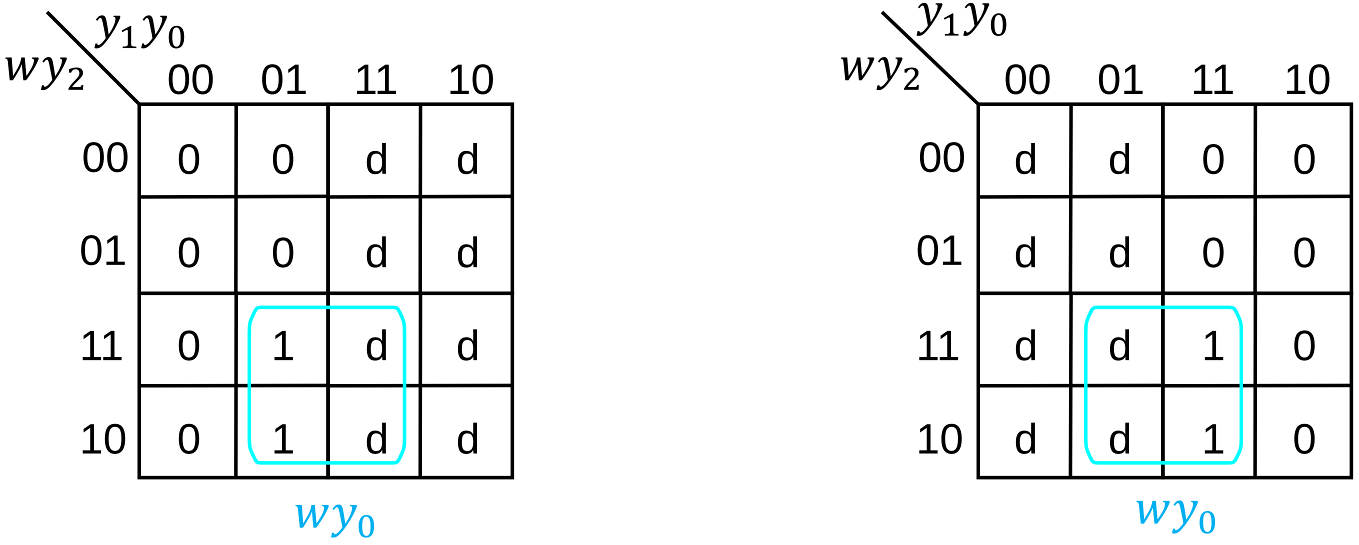


wy_2 \ y_1y_0	00	01	11	10
00	d	0	0	d
01	d	0	0	d
11	d	1	1	d
10	d	1	1	d

$$K_0 = w$$

Figure 8.66. Karnaugh maps for JK flip-flops in the counter.

Implementation Using JK-Type Flip-Flops



wy_2 \ y_1y_0	00	01	11	10
00	0	0	d	d
01	0	0	d	d
11	0	1	d	d
10	0	1	d	d

wy_0

$J_1 = wy_0$

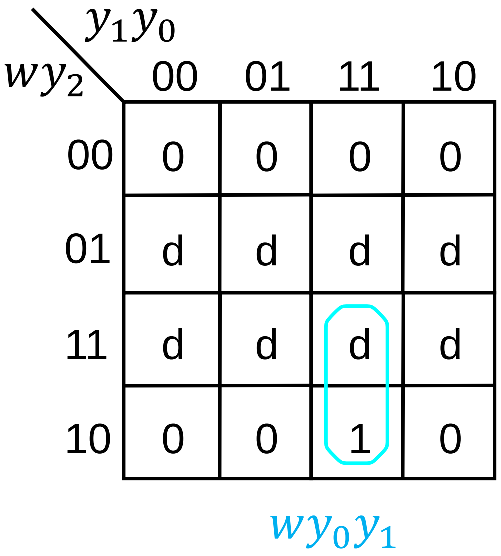
wy_2 \ y_1y_0	00	01	11	10
00	d	d	0	0
01	d	d	0	0
11	d	d	1	0
10	d	d	1	0

wy_0

$K_1 = wy_0$

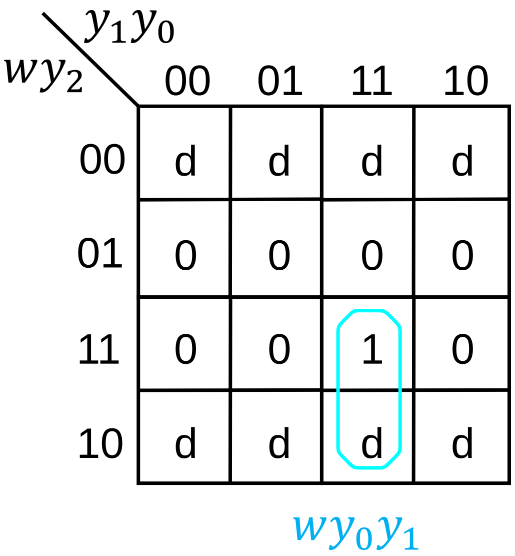
Figure 8.66. Karnaugh maps for JK flip-flops in the counter.

Implementation Using JK-Type Flip-Flops



$wy_2 \backslash y_1y_0$	00	01	11	10
00	0	0	0	0
01	d	d	d	d
11	d	d	d	d
10	0	0	1	0

$$J_2 = wy_0y_1$$



$wy_2 \backslash y_1y_0$	00	01	11	10
00	d	d	d	d
01	0	0	0	0
11	0	0	1	0
10	d	d	d	d

$$K_2 = wy_0y_1$$

Figure 8.66. Karnaugh maps for JK flip-flops in the counter.

Circuit diagram using JK flip-flops

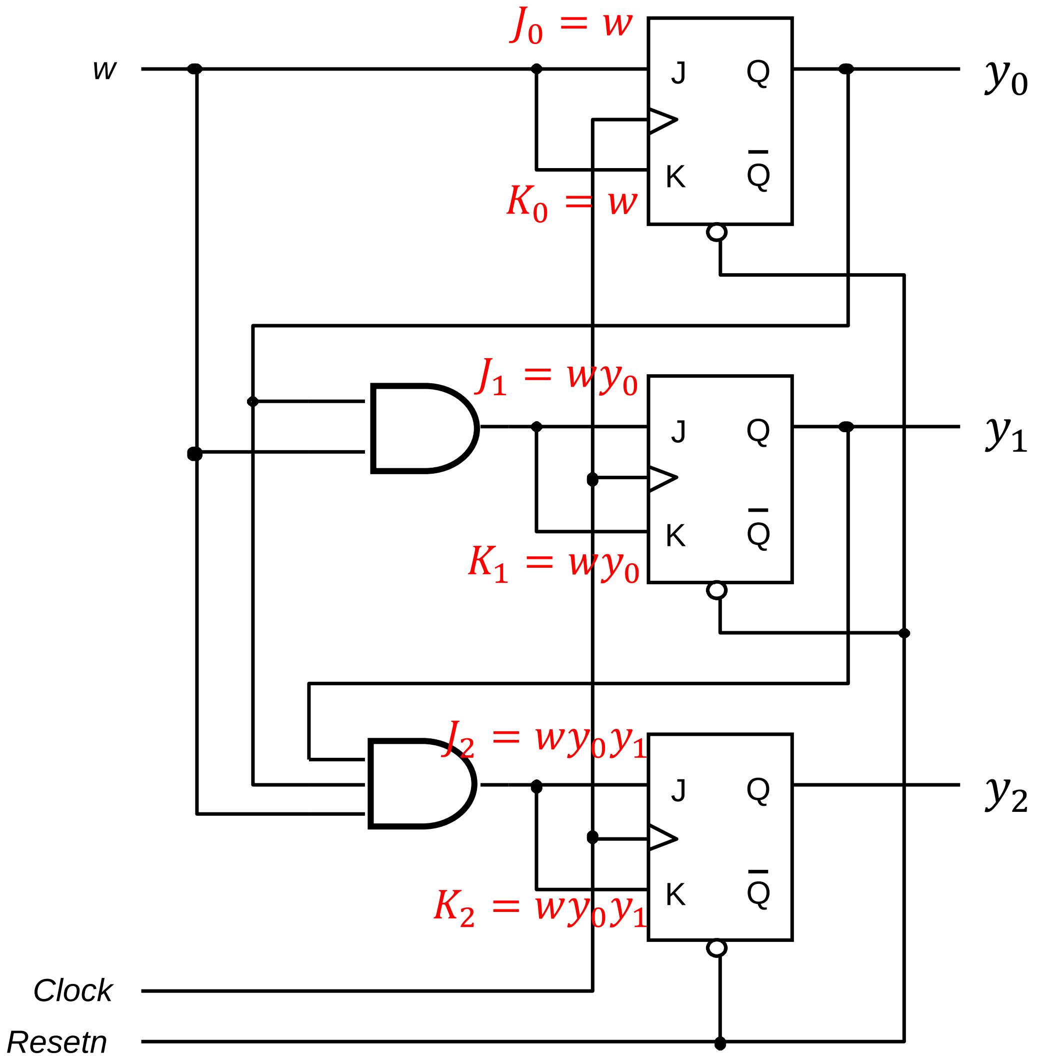


Figure 8.67. Circuit diagram using JK flip-flops.

Factored-form implementation of the counter

$$J_2 = K_2 = (wy_0)y_1 \qquad = J_1y_1$$

$$J_n = K_n = (wy_0 \cdots y_{n-2})y_{n-1} = J_{n-1}y_{n-1}$$

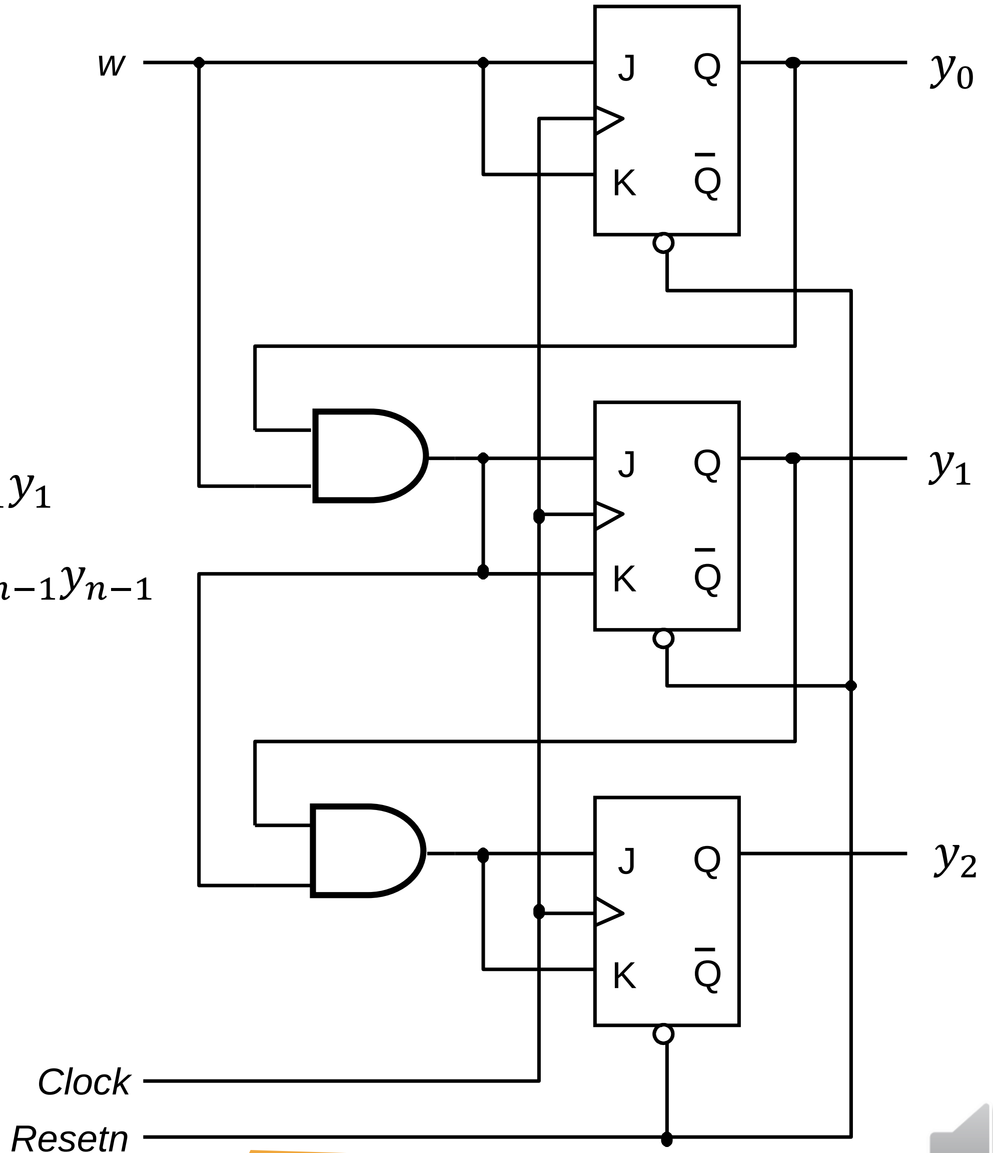
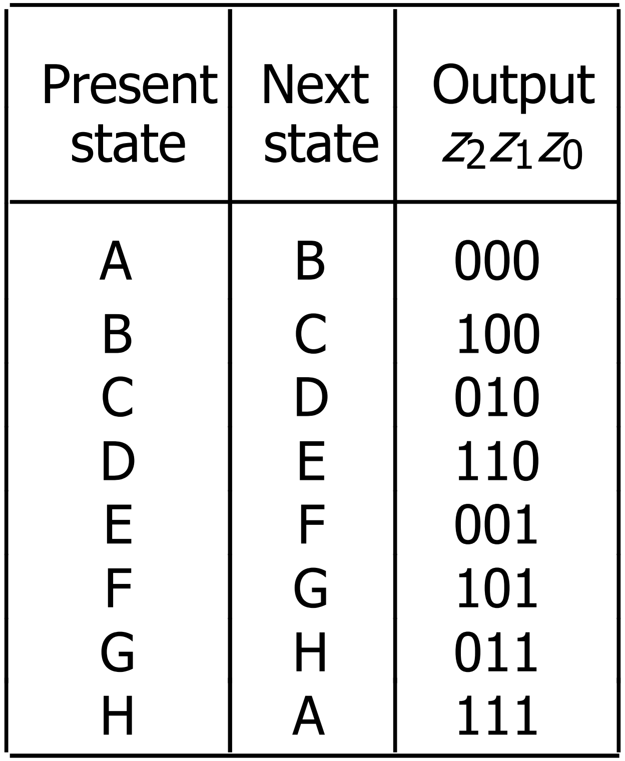


Figure 8.68. Factored-form implementation of the counter.

Example-A Different Counter



Present state	Next state	Output $z_2 z_1 z_0$
A	B	000
B	C	100
C	D	010
D	E	110
E	F	001
F	G	101
G	H	011
H	A	111

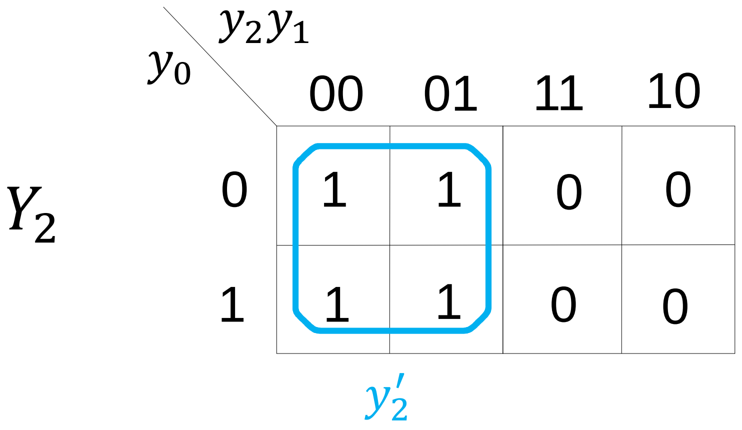
Figure 8.69. State table for the counter like example.

State-assigned table for Figure 8.69

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

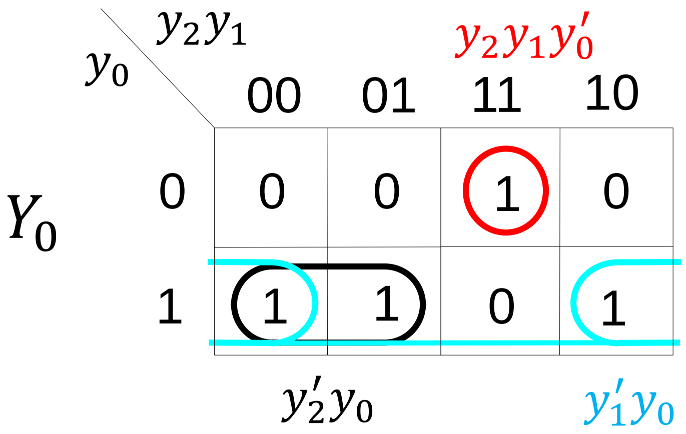
Figure 8.70. State-assigned table for Figure 8.69.

Karnaugh map for Figure 8.70



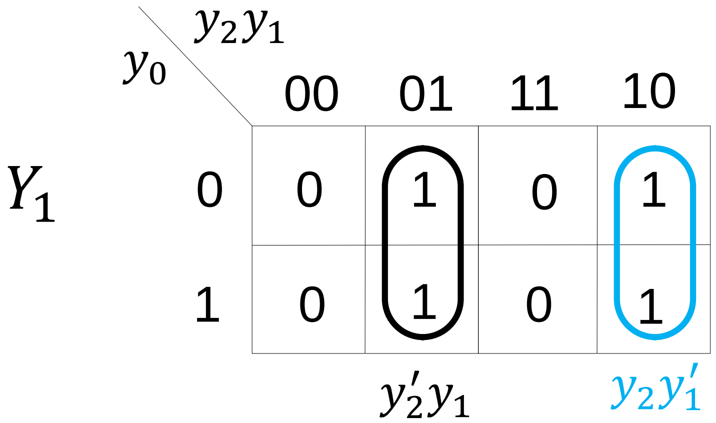
y_0 \ y_2y_1	00	01	11	10
0	1	1	0	0
1	1	1	0	0

$$Y_2 = y_2'$$



y_0 \ y_2y_1	00	01	11	10
0	0	0	1	0
1	1	1	0	1

$$\begin{aligned} Y_0 &= y_2'y_0 + y_1'y_0 + y_2y_1y_0' \\ &= (y_2' + y_1')y_0 + y_2y_1y_0' \\ &= y_2y_1 \oplus y_0 \end{aligned}$$



y_0 \ y_2y_1	00	01	11	10
0	0	1	0	1
1	0	1	0	1

$$Y_1 = y_2'y_1 + y_2y_1' = y_2 \oplus y_1$$

Circuit for Figure 8.70

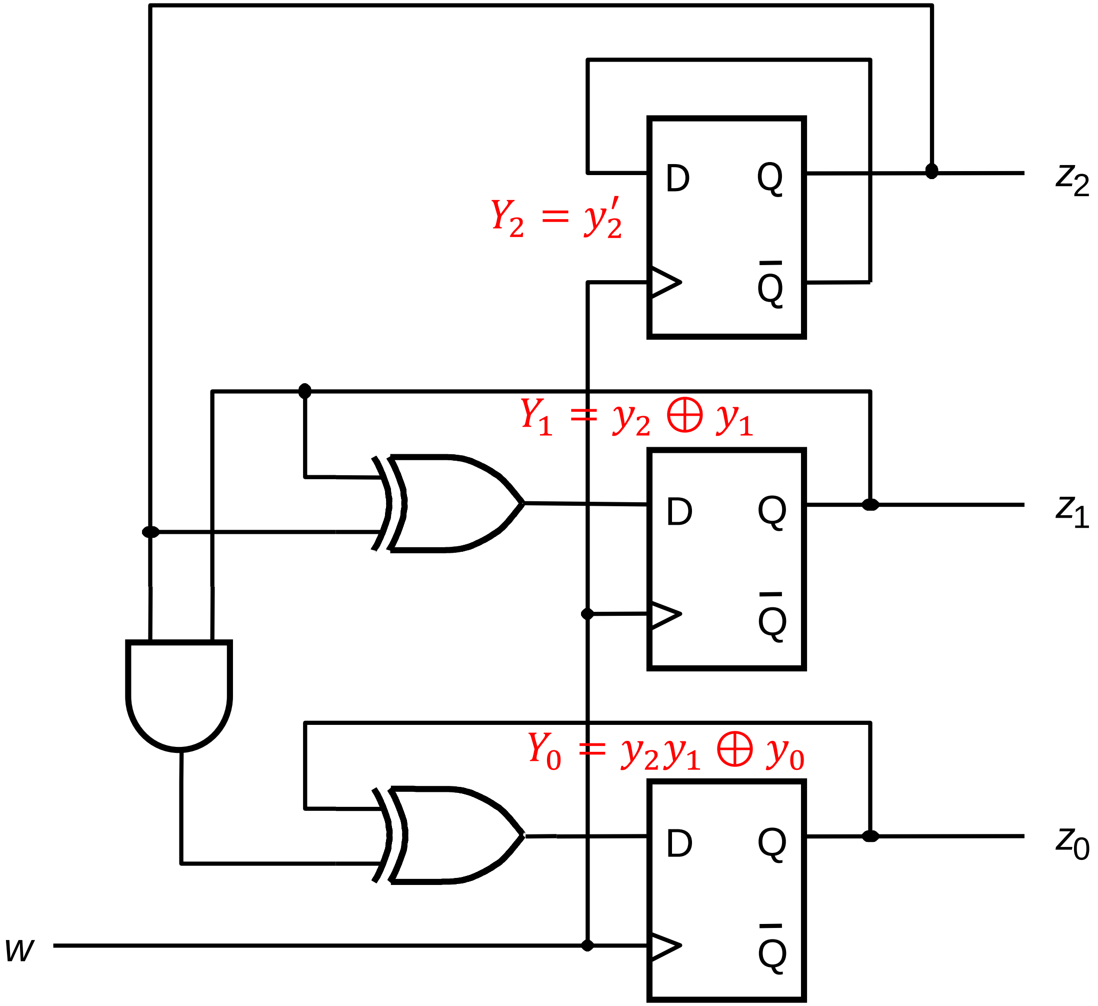


Figure 8.71. Circuit for Figure 8.70.

ANALYSIS OF SYNCHRONOUS SEQUENTIAL CIRCUITS

Example 8.8

Moore-type design

The output of the flip-flops represent the present-state variables.
Their inputs determine the next state that the circuit will enter.

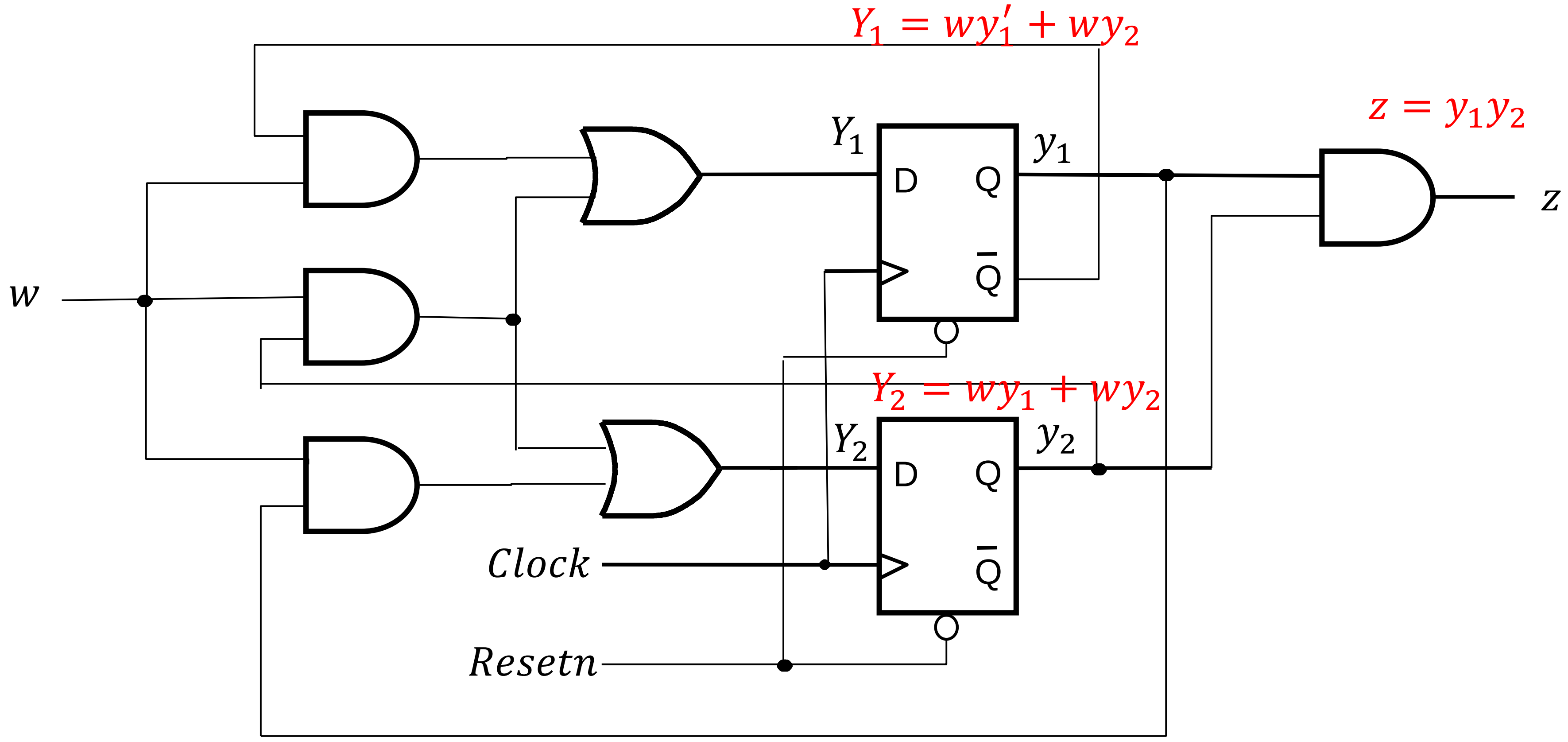


Figure 8.80. Circuit for Example 8.8.

Example 8.8

$$z = y_1 y_2$$
$$Y_1 = w y_1' + w y_2$$
$$Y_2 = w y_1 + w y_2$$

$y_2 y_1 = 00, w = 0 \quad \rightarrow \quad Y_2 Y_1 = 00, z = 0$

$y_2 y_1 = 00, w = 1 \quad \rightarrow \quad Y_2 Y_1 = 01$

	Present state	Next state		Output
		$w = 0$	$w = 1$	
	$y_2 y_1$	$Y_2 Y_1$	$Y_2 Y_1$	z
A	00	00	01	0
B	01	00	10	0
C	10	00	11	0
D	11	00	11	1

(a) State-assigned table

Figure 8.81. Tables for the circuit in Figure 8.80.

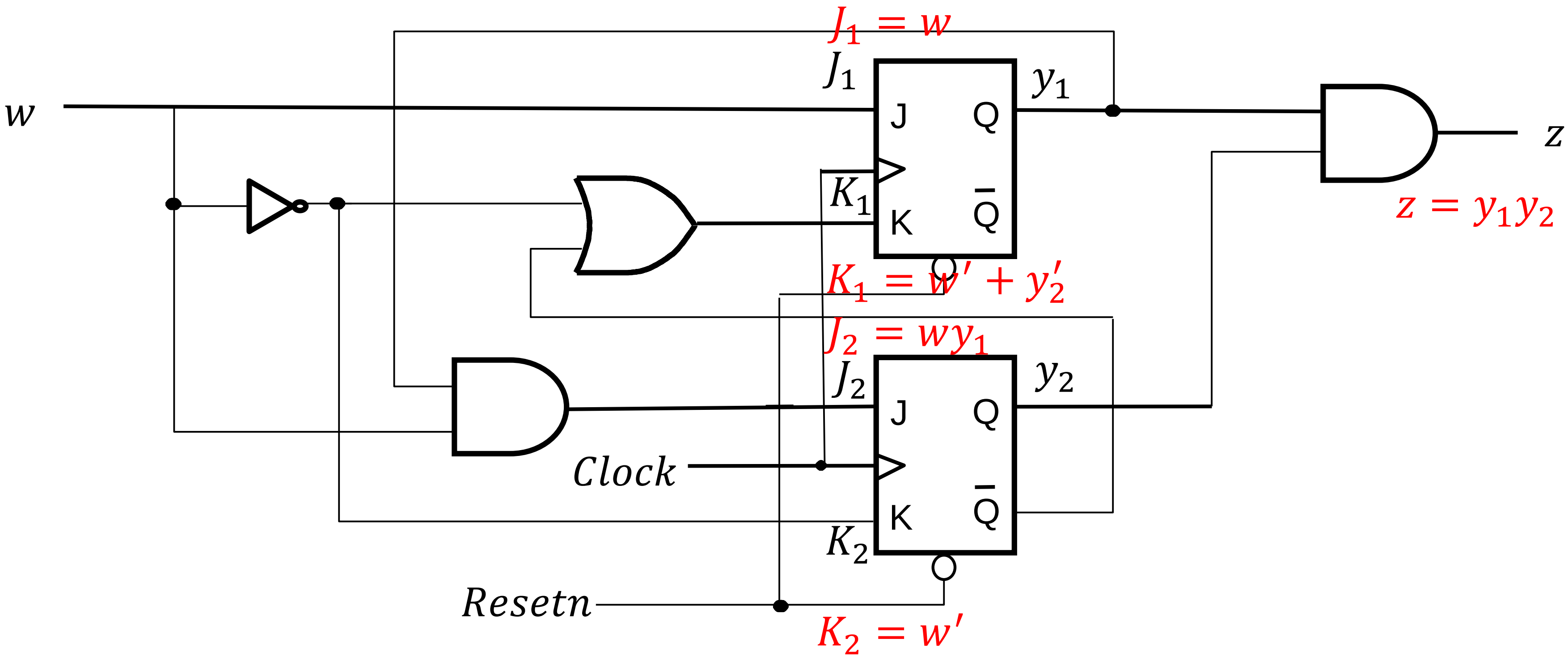
Example 8.8

Present state	Nextstate		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 8.81. Tables for the circuit in Figure 8.80.

Example 8.9



$J_1 = w$

$K_1 = w' + y_2'$

$J_2 = wy_1$

$K_2 = w'$

$z = y_1 y_2$

Figure 8.82. Circuit for Example 8.9.

Example 8.9

$J_1 = w \qquad K_1 = w' + y_2'$

$J_2 = wy_1 \qquad K_2 = w' \qquad z = y_1 y_2$

$y_2 y_1 = 00, w = 0 \quad \rightarrow J_2 K_2 = 01, J_1 K_1 = 01, z = 0$
$y_2 y_1 = 00, w = 1 \quad \rightarrow J_2 K_2 = 00, J_1 K_1 = 11$

Present state	Next state				Output
	$w = 0$		$w = 1$		z
$y_2 y_1$	$J_2 K_2$	$J_1 K_1$	$J_2 K_2$	$J_1 K_1$	
00	01	01	00	11	0
01	01	01	10	11	0
10	01	01	00	01	0
11	01	01	10	01	1

Figure 8.83. The excitation tables for the circuit in Figure 8.82.

Example 8.10

$D_1 = w(y_1' + y_2)$

$T_2 = w'y_2 + wy_1y_2'$

$z = y_1y_2$

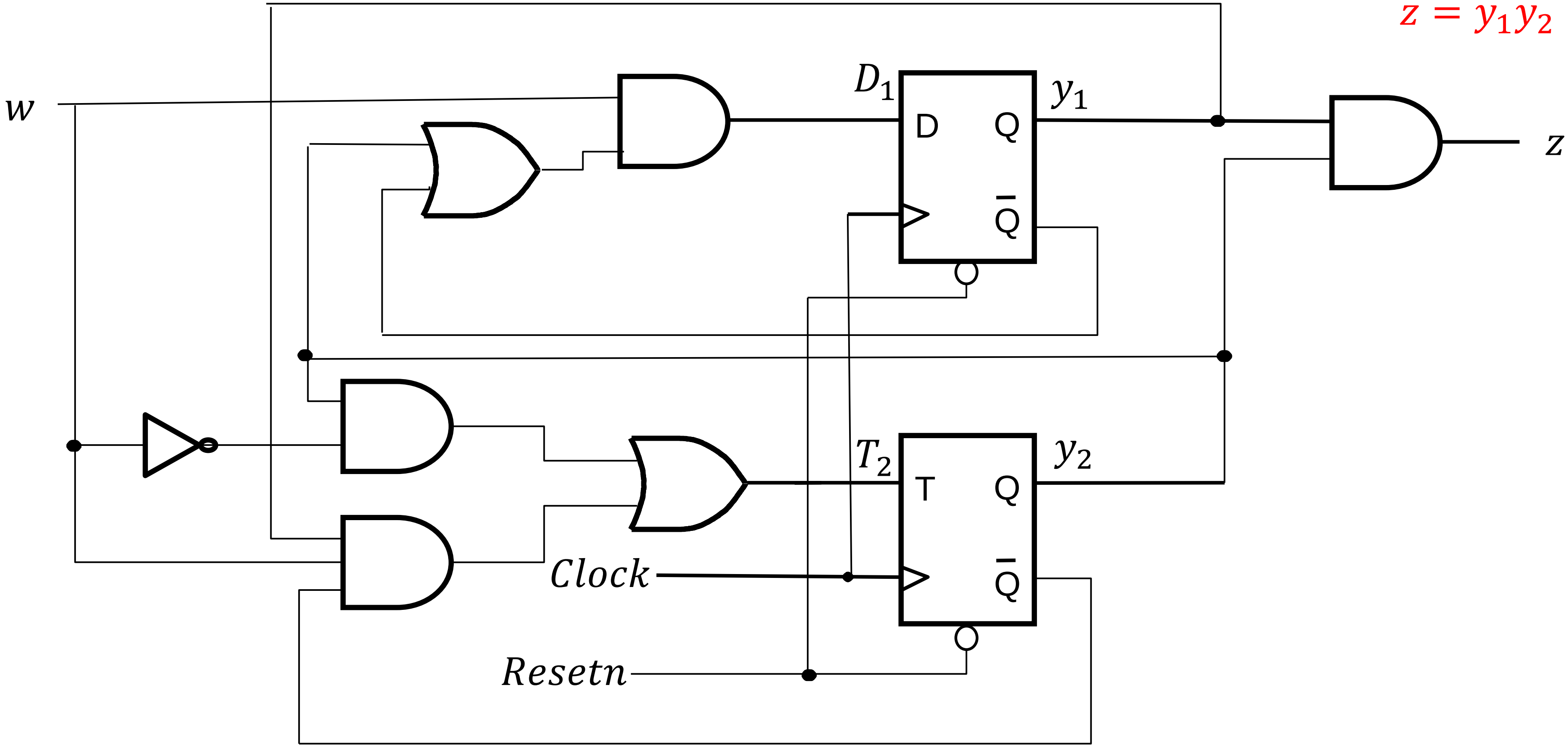


Figure 8.84. Circuit for Example 8.10.

Example 8.10

$D_1 = w(y_1' + y_2)$

$T_2 = w'y_2 + wy_1y_2'$ $z = y_1y_2$

$y_2y_1 = 00, w = 0 \rightarrow T_2D_1 = 00, z = 0$

$y_2y_1 = 00, w = 1 \rightarrow T_2D_1 = 01$

Present state	Next state		Output
	$w = 0$	$w = 1$	
y_2y_1	T_2D_1	T_2D_1	z
00	00	01	0
01	00	10	0
10	10	01	0
11	10	01	1

Figure 8.85. The excitation table for the circuit in Figure 8.84.

EXAMPLES OF SOLVED PROBLEMS

Example 8.11

Problem: Design an FSM that has an input w and an output z. The machine is a sequence detector that produces $z = 1$ when the previous two values of w were 00 or 11; otherwise $z = 0$.

State diagram for Example 8.11

A: reset

B: 0

C: more than 00

D: 1

C: more than 11

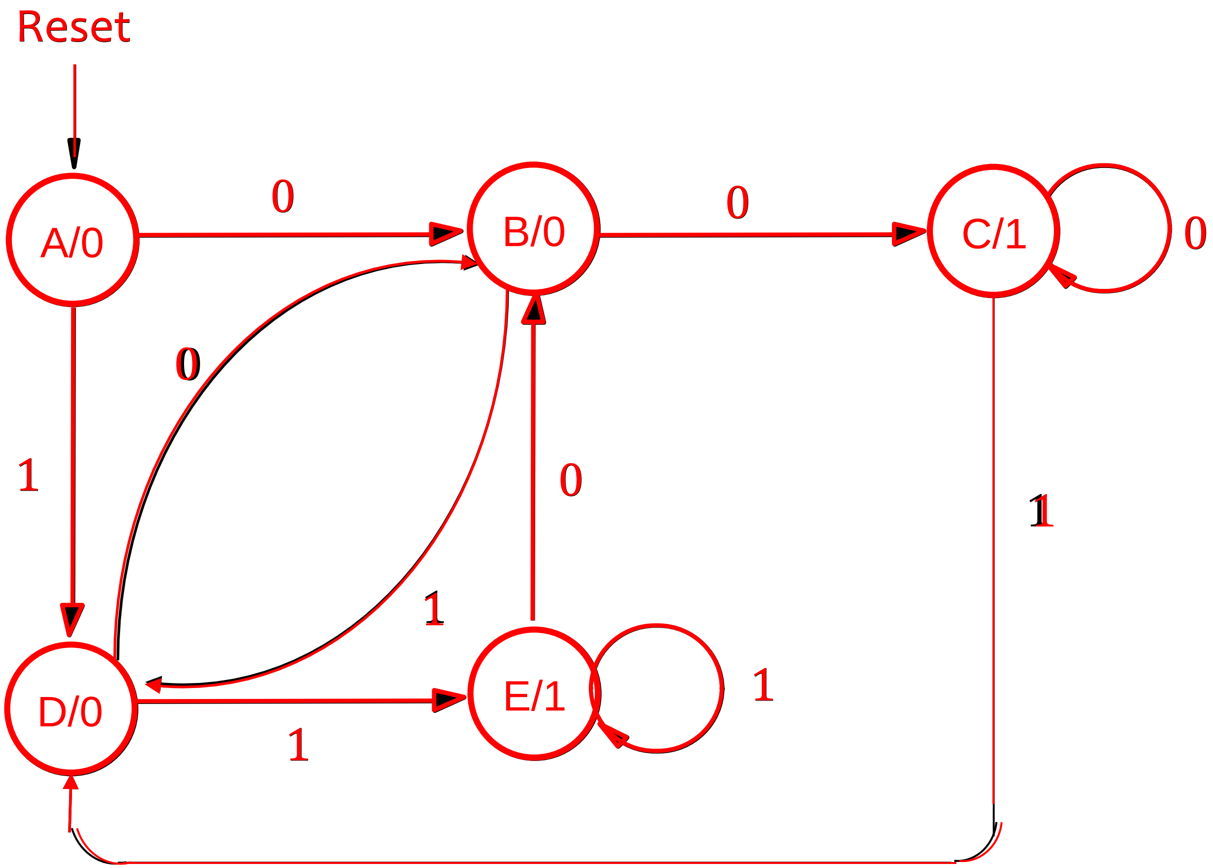


Figure 8.91. State diagram for Example 8.11.

Stable table

Present state	Next state		Output
	w = 0	*w* = 1	z
A g1	B g1	D g1	0
B g1	C g2	D g1	0
C g2	C g2	D g1	1
D g1	B g1	E g2	0
E g2	B g1	E g2	1

P1=(ABCDE)

P2=(ABD)(CE)

g1 g2

P3=(A)(B)(D)(C)(E)

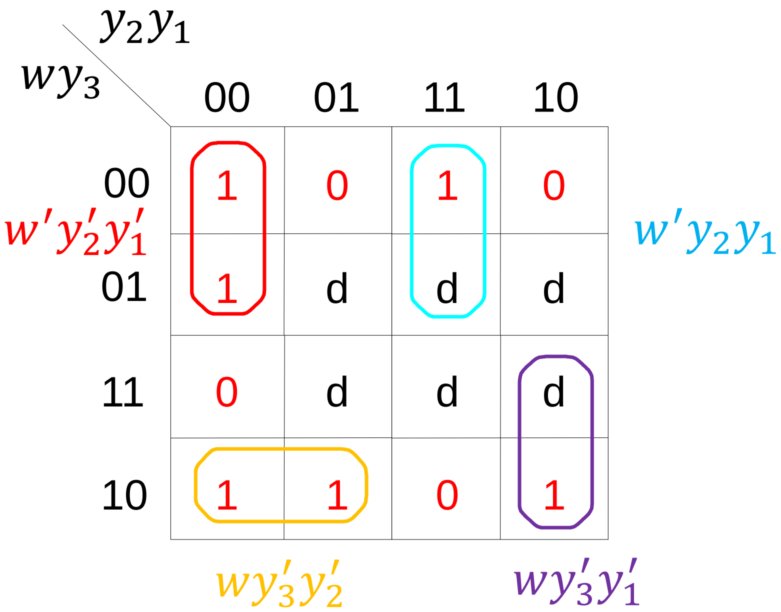
Figure 8.92. State table for the FSM in Example 8.11.

Stable table for Example 8.11

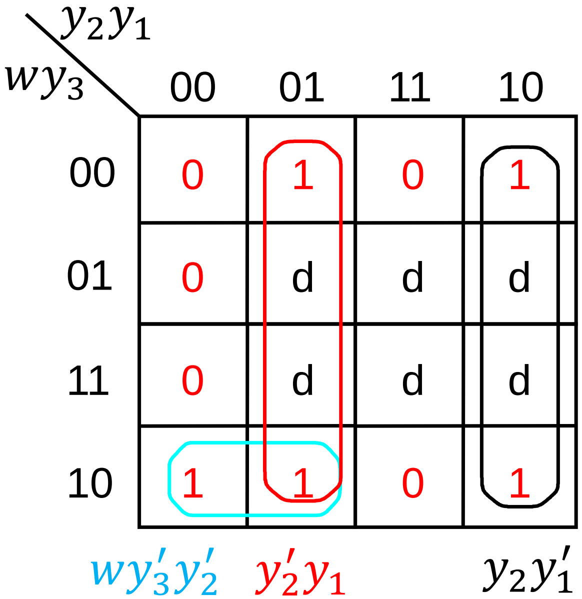
	Present state	Next state		Output
		$w = 0$	$w = 1$	
	$y_3y_2y_1$	$Y_3Y_2Y_1$	$Y_3Y_2Y_1$	z
A	000	000	001	0
B	001	001	010	0
C	010	010	011	1
D	011	011	100	0
E	100	100	101	1

Figure 8.93. State-assigned table for the FSM in Figure 8.92.

Karnaugh Map for Example 8.11



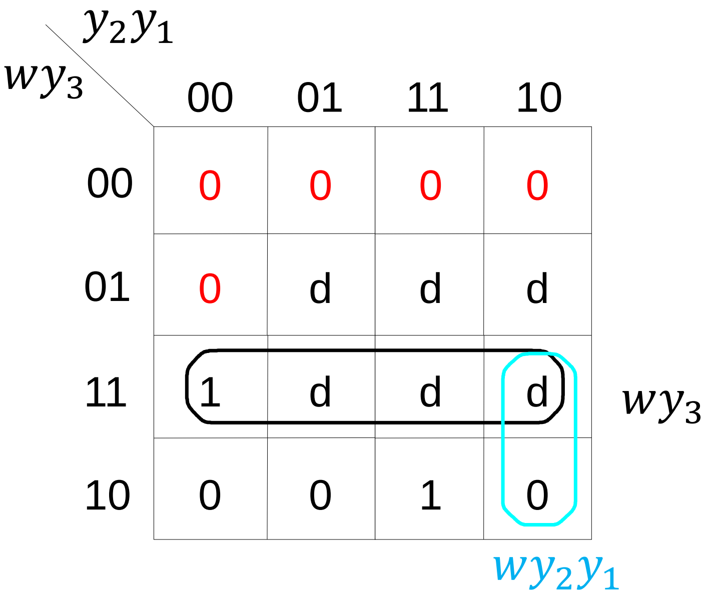
wy_3 y_2y_1	00	01	11	10
00	1	0	1	0
01	1	d	d	d
11	0	d	d	d
10	1	1	0	1



wy_3 y_2y_1	00	01	11	10
00	0	1	0	1
01	0	d	d	d
11	0	d	d	d
10	1	1	0	1

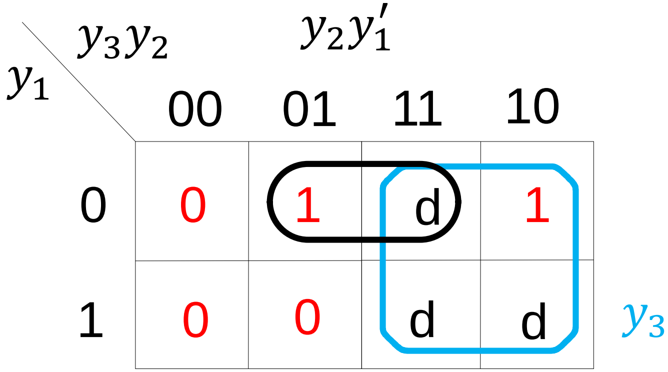
$$Y_1 = wy_3'y_1' + wy_3'y_2' + w'y_2y_1 + w'y_2'y_1' \qquad Y_2 = y_2'y_1 + y_2y_1' + wy_3'y_2'$$

Karnaugh Map for Example 8.11



wy_3 \ y_2y_1	00	01	11	10
00	0	0	0	0
01	0	d	d	d
11	1	d	d	d
10	0	0	1	0

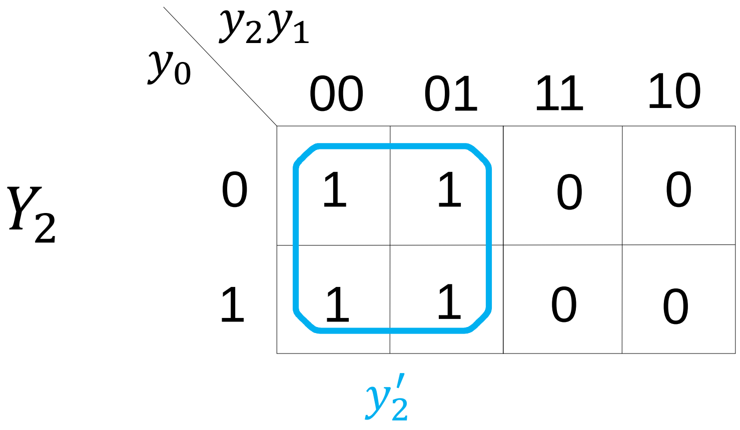
$$Y_3 = wy_3 + wy_2y_1$$



y_1 \ y_3y_2	00	01	11	10
0	0	1	d	1
1	0	0	d	d

$$z = y_3 + y_2y_1'$$

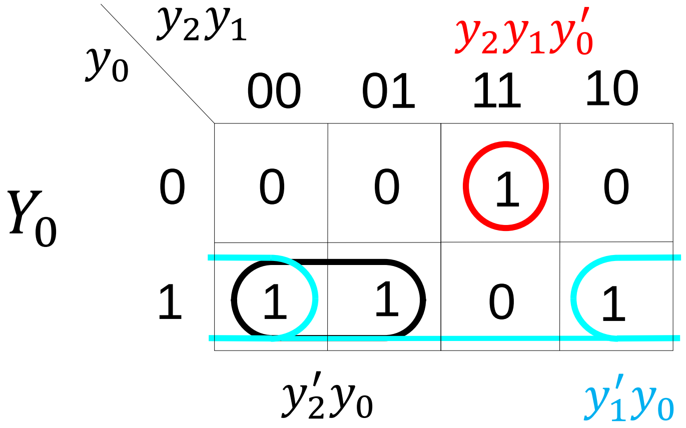
Karnaugh map for Figure 8.70



Y_2

y_0 \ y_2y_1	00	01	11	10
0	1	1	0	0
1	1	1	0	0

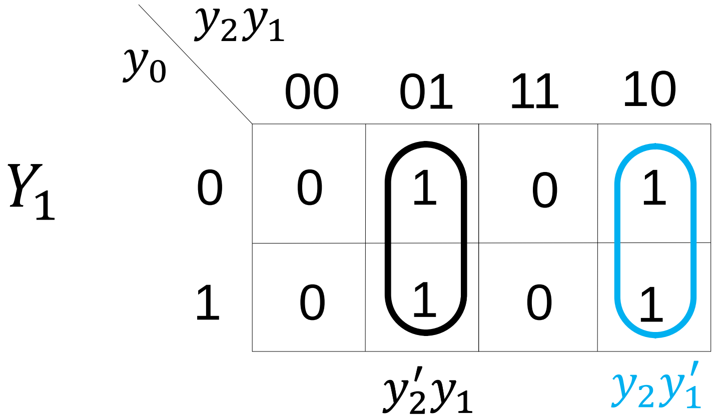
$$Y_2 = y_2'$$



Y_0

y_0 \ y_2y_1	00	01	11	10
0	0	0	1	0
1	1	1	0	1

$$\begin{aligned} Y_0 &= y_2'y_0 + y_1'y_0 + y_2y_1y_0' \\ &= (y_2' + y_1')y_0 + y_2y_1y_0' \\ &= y_2y_1 \oplus y_0 \end{aligned}$$

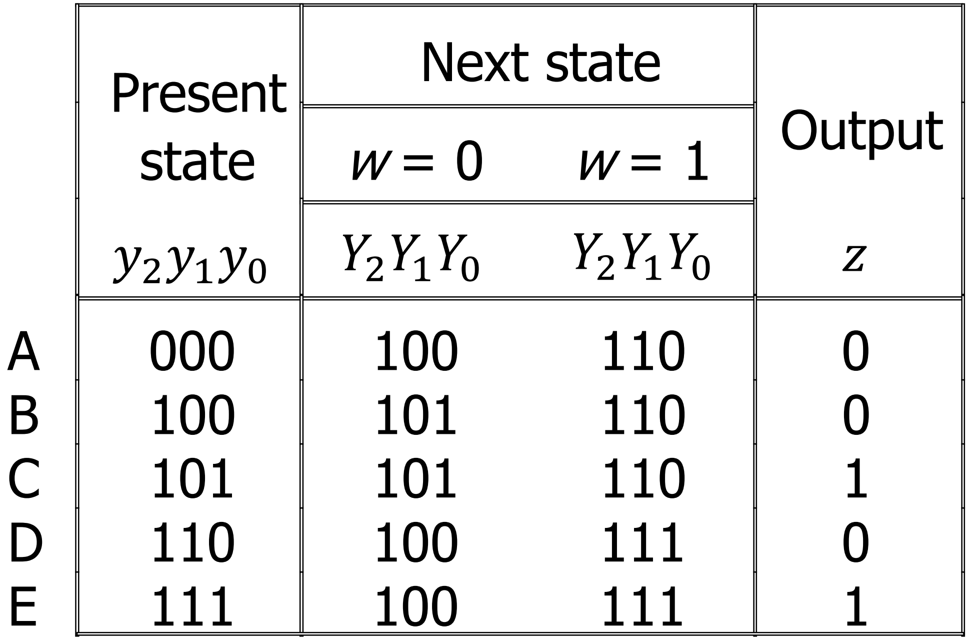


Y_1

y_0 \ y_2y_1	00	01	11	10
0	0	1	0	1
1	0	1	0	1

$$Y_1 = y_2'y_1 + y_2y_1' = y_2 \oplus y_1$$

Stable table



	Present state	Next state		Output
		$w = 0$	$w = 1$	
	$y_2y_1y_0$	$Y_2Y_1Y_0$	$Y_2Y_1Y_0$	z
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 8.94. An improved state assignment for the FSM in Figure 8.92.

Karnaugh Map for Example 8.11

wy_3 \ y_2y_1	00	01	11	10
00	0	d	d	d
01	1	1	0	0
11	0	0	1	1
10	0	d	d	d

$w'y_3y_2'$

wy_2

wy_3 \ y_2y_1	00	01	11	10
00	0	d	d	d
01	0	0	0	0
11	1	1	1	1
10	1	d	d	d

w

$$Y_1 = wy_2 + w'y_3y_2'$$

$$Y_2 = w$$

Karnaugh Map for Example 8.11

wy_3 \ y_2y_1	00	01	11	10
00	1	d	d	d
01	1	1	1	1
11	1	1	1	1
10	1	d	d	d

$$Y_3 = 1$$

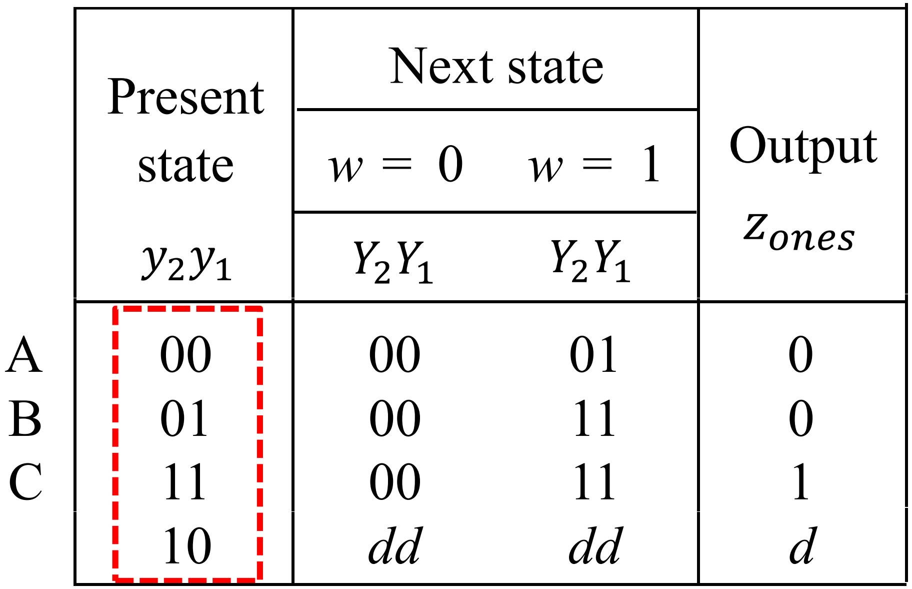
y_1 \ y_3y_2	00	01	11	10	
0	0	d	0	0	
1	d	d	1	1	y_1

$$z = y_1$$

Example 8.12

Problem: Implement the sequence detector of Example 8.11 by using two FSMs. One FSM detects the occurrence of consecutive 1s, while the other detects consecutive 0’s.

State-Assignment Problem



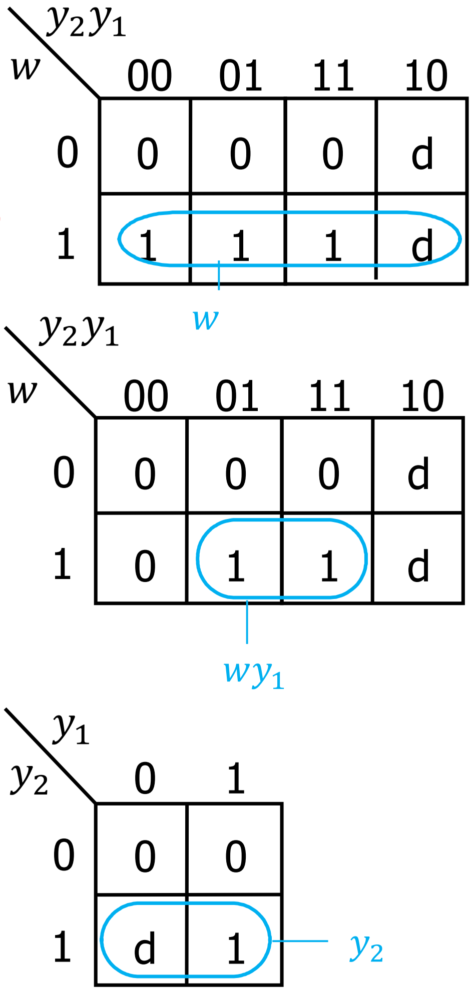
	Present state	Next state		Output
		$w = 0$	$w = 1$	
	y_2y_1	Y_2Y_1	Y_2Y_1	z_{ones}
A	00	00	01	0
B	01	00	11	0
C	11	00	11	1
	10	*dd*	*dd*	*d*

Gray code

Figure 8.16. Improved state assignment for the sequential circuit in Figure 8.4.

Derivation of logic expressions for next states and output

Gray code



Y_1

Next state

w \ y_2y_1	00	01	11	10
0	0	0	0	d
1	1	1	1	d

Y_2

Next state

w \ y_2y_1	00	01	11	10
0	0	0	0	d
1	0	1	1	d

z_{ones}

output

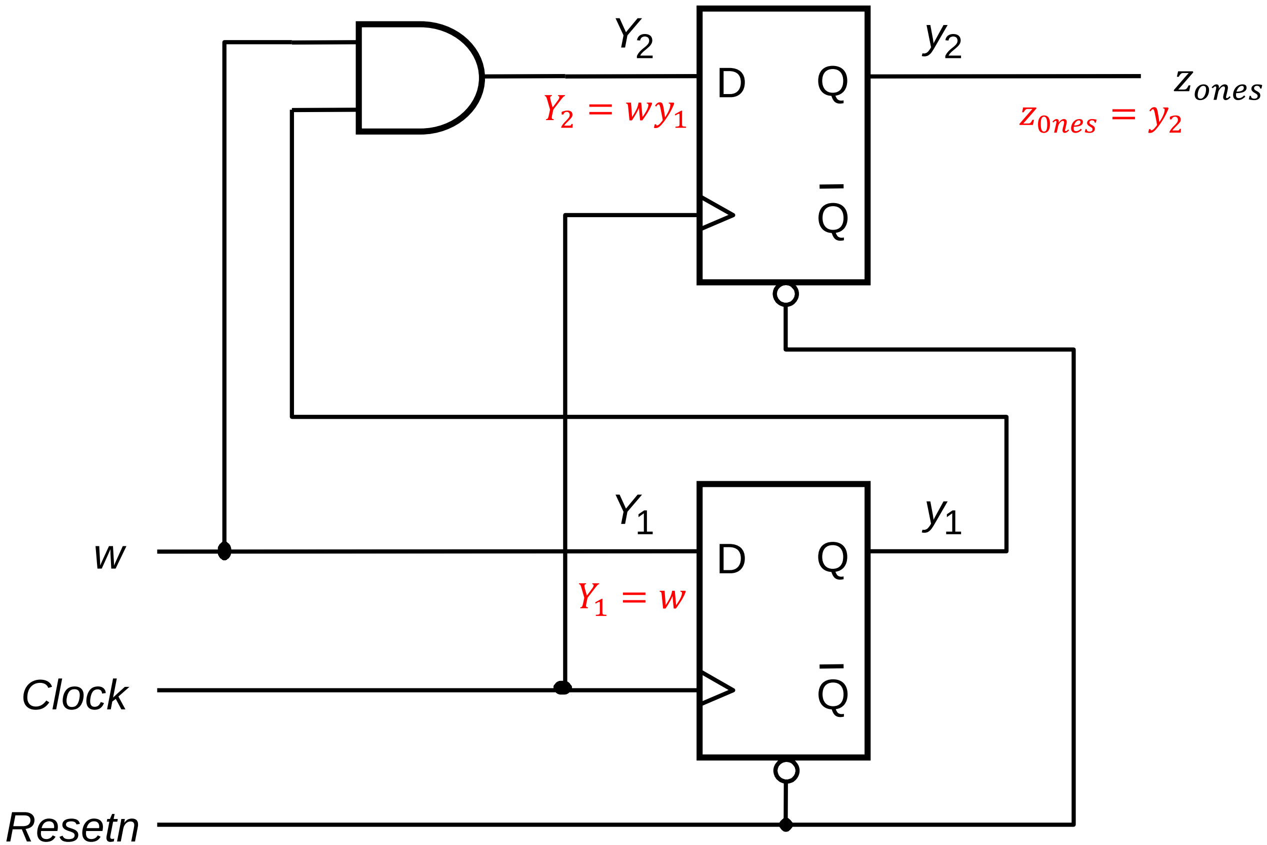
y_2 \ y_1	0	1
0	0	0
1	d	1

$$Y_1 = w$$

$$Y_2 = wy_1$$

$$z_{ones} = y_2$$

Circuit for the output z_{ones}



$Y_2 = wy_1$

$z_{ones} = y_2$

$Y_1 = w$

Example 8.12

Present state	Next 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

(b) State-assigned table

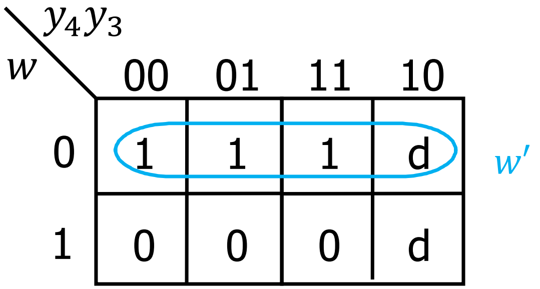
Figure 8.95. FSM that detects a sequence of two zeros.

Derivation of logic expressions for next states and output

Gray code

Y_3

Next state

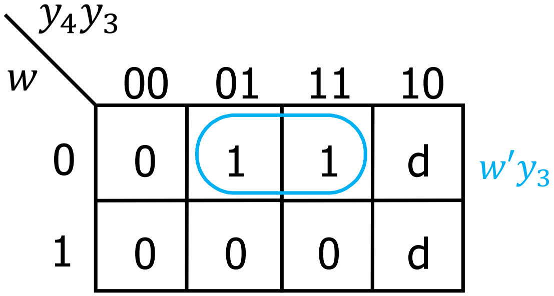


$w \backslash y_4y_3$	00	01	11	10
0	1	1	1	d
1	0	0	0	d

w'

Y_4

Next state

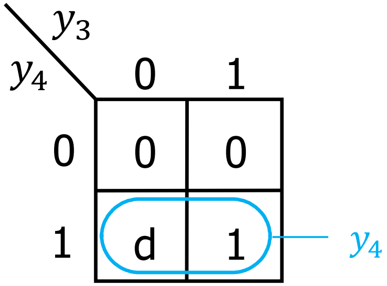


$w \backslash y_4y_3$	00	01	11	10
0	0	1	1	d
1	0	0	0	d

$w'y_3$

z_{zeros}

output



$y_4 \backslash y_3$	0	1
0	0	0
1	d	1

y_4

$$Y_1 = w'$$

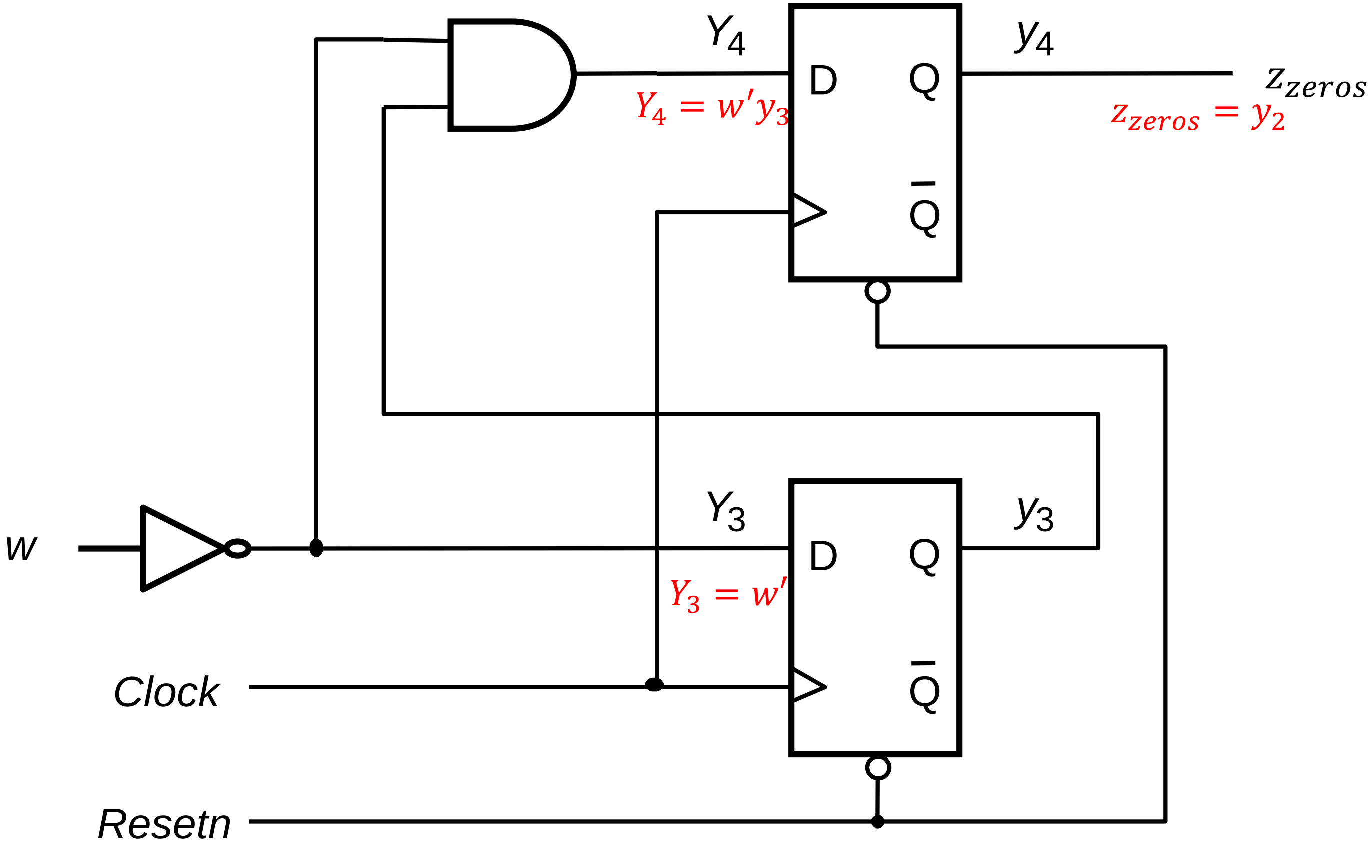
$$Y_2 = w'y_3$$

$$z_{zeros} = y_4$$

Circuit for the output z_{zeros}

The output of the combined circuit is

$z = z_{ones} + z_{zeros}$



Example 8.13

Problem: Derive a Mealy-type FSM that can act as a sequence detector described in Example 8.11.

State diagram

A: reset

B: ≥ 0

C: ≥ 1

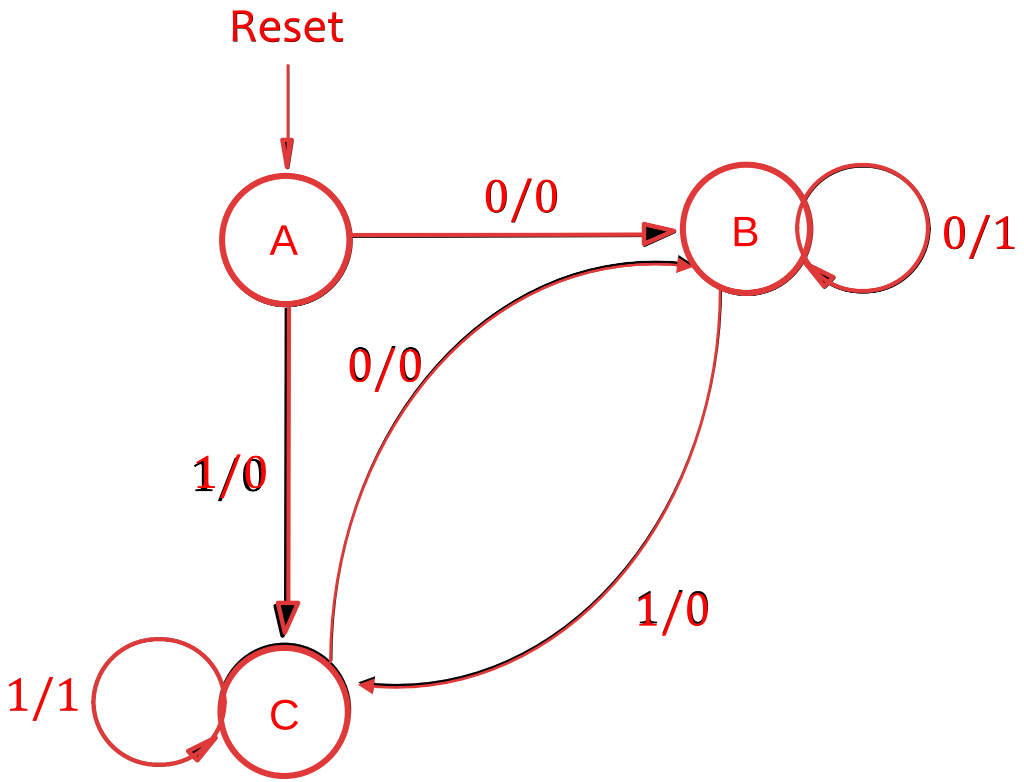


Figure 8.96. State diagram for Example 8.13.

Stable table

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 8.97. State table for the FSM in Figure 8.96.

Stable table

	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 8.98. State-assigned table for the FSM in Figure 8.92.

Karnaugh map for Figure 8.98

Y_2

$w \backslash y_2y_1$	00	01	11	10
0	0	0	0	d
1	1	1	1	d

w

$$Y_2 = w$$

z

$y_2'y_1w'$

$w \backslash y_2y_1$	00	01	11	10
0	0	1	0	d
1	0	0	1	d

y_2w

$$z = y_2'y_1w' + y_2w$$

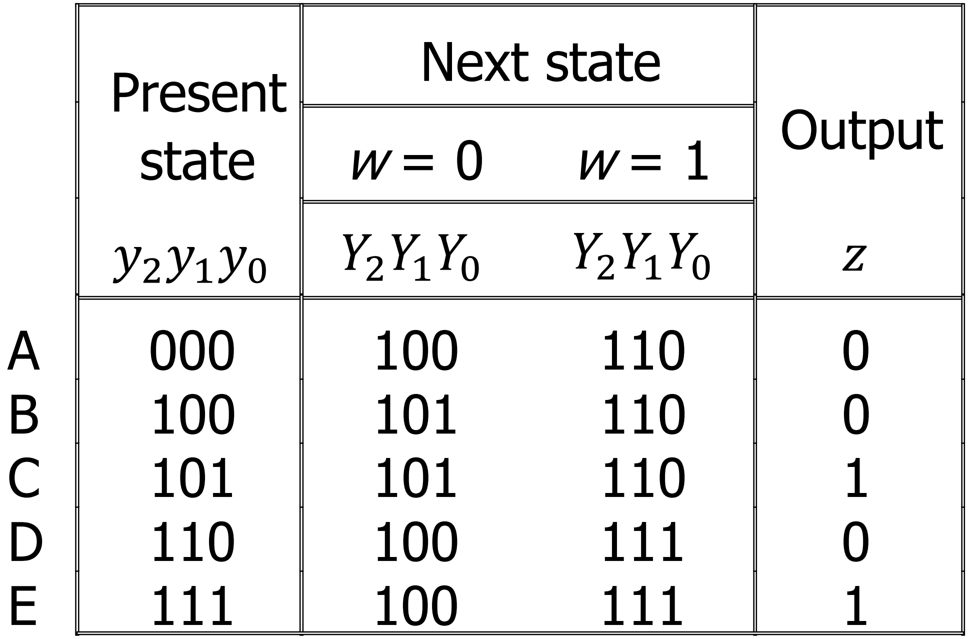
Y_1

$w \backslash y_2y_1$	00	01	11	10
0	1	1	1	d
1	1	1	1	d

$$Y_1 = 1$$

Example 8.14

Problem: Implement the FSM in the Figure 8.94 using JK-type flip-flop.



	Present state	Next state		Output
		$w = 0$	$w = 1$	
	$y_2y_1y_0$	$Y_2Y_1Y_0$	$Y_2Y_1Y_0$	z
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 8.94. An improved state assignment for the FSM in Figure 8.92.

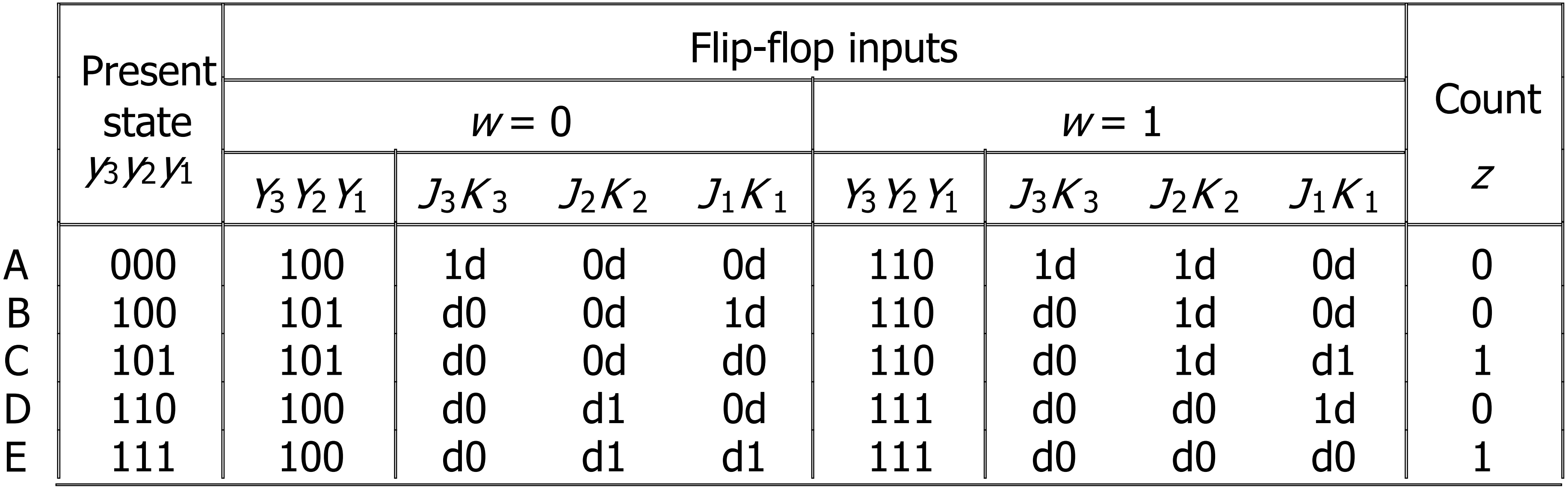
Implementation Using JK-Type Flip-Flops

J	K	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$Q'(t)$

Truth table

Present state $Q(t) = y$	Next state $Q(t+1) = Y$	J	K		
0	0	0 0	0 1	0	d
0	1	1 1	0 1	1	d
1	0	0 1	1 1	d	1
1	1	1 0	0 0	d	0

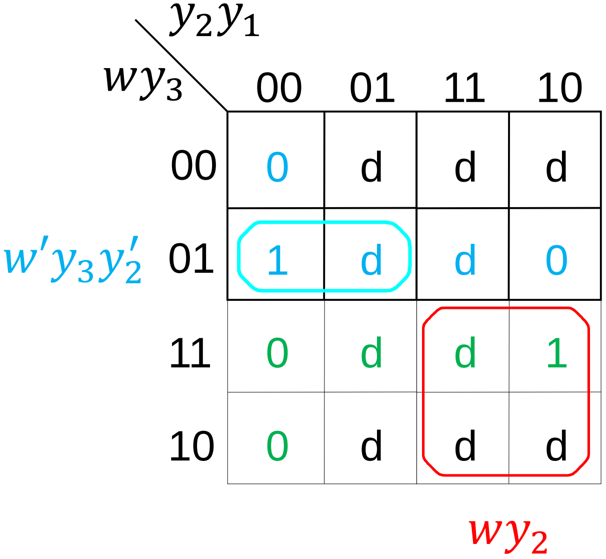
Example 8.14



	Present state	Flip-flop inputs								Count
		$w = 0$				$w = 1$				
	$y_3 y_2 y_1$	$Y_3 Y_2 Y_1$	$J_3 K_3$	$J_2 K_2$	$J_1 K_1$	$Y_3 Y_2 Y_1$	$J_3 K_3$	$J_2 K_2$	$J_1 K_1$	z
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 8.99. Excitation table for the FSM in Figure 8.94 with JK-type flip-flops.

Implementation Using JK-Type Flip-Flops

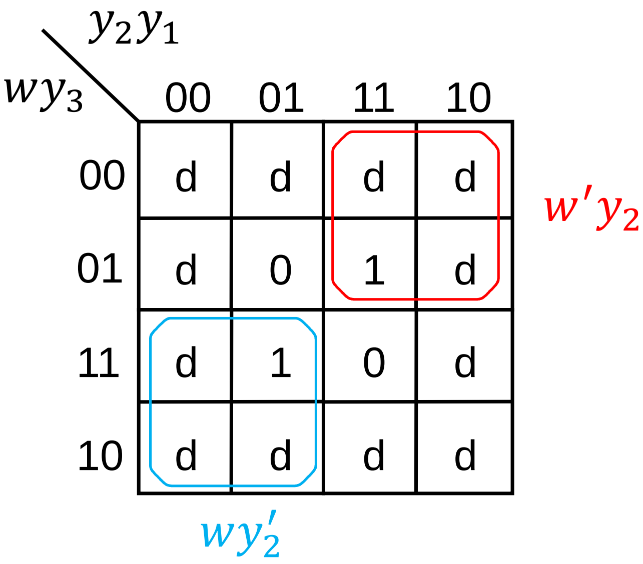


$wy_3 \backslash y_2y_1$	00	01	11	10
00	0	d	d	d
01	1	d	d	0
11	0	d	d	1
10	0	d	d	d

$w'y_3y_2'$

wy_2

$$J_1 = wy_2 + w'y_3y_2'$$



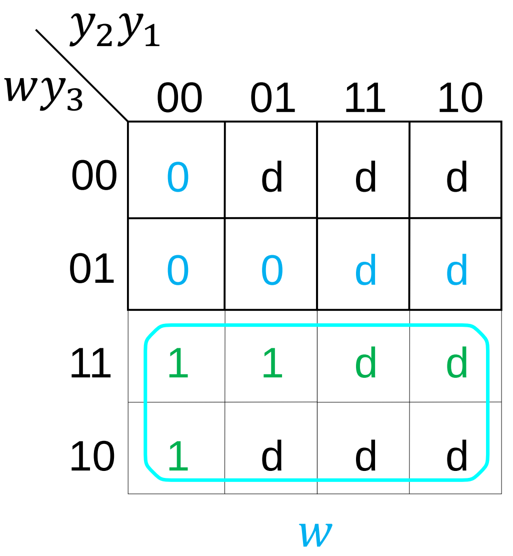
$wy_3 \backslash y_2y_1$	00	01	11	10
00	d	d	d	d
01	d	0	1	d
11	d	1	0	d
10	d	d	d	d

$w'y_2$

wy_2'

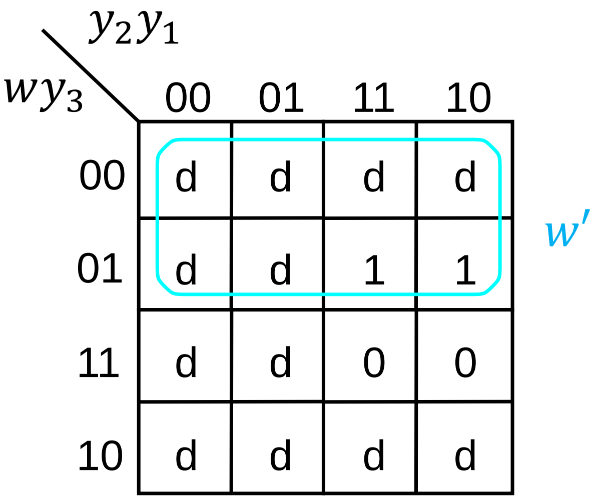
$$K_1 = w'y_2 + wy_2'$$

Implementation Using JK-Type Flip-Flops



$wy_3 \backslash y_2y_1$	00	01	11	10
00	0	d	d	d
01	0	0	d	d
11	1	1	d	d
10	1	d	d	d

$J_2 = w$



$wy_3 \backslash y_2y_1$	00	01	11	10
00	d	d	d	d
01	d	d	1	1
11	d	d	0	0
10	d	d	d	d

$K_2 = w'$

Implementation Using JK-Type Flip-Flops

wy_3 \ y_2y_1	00	01	11	10
00	1	d	d	d
01	d	d	d	d
11	d	d	d	d
10	1	d	d	d

$J_3 = 1$

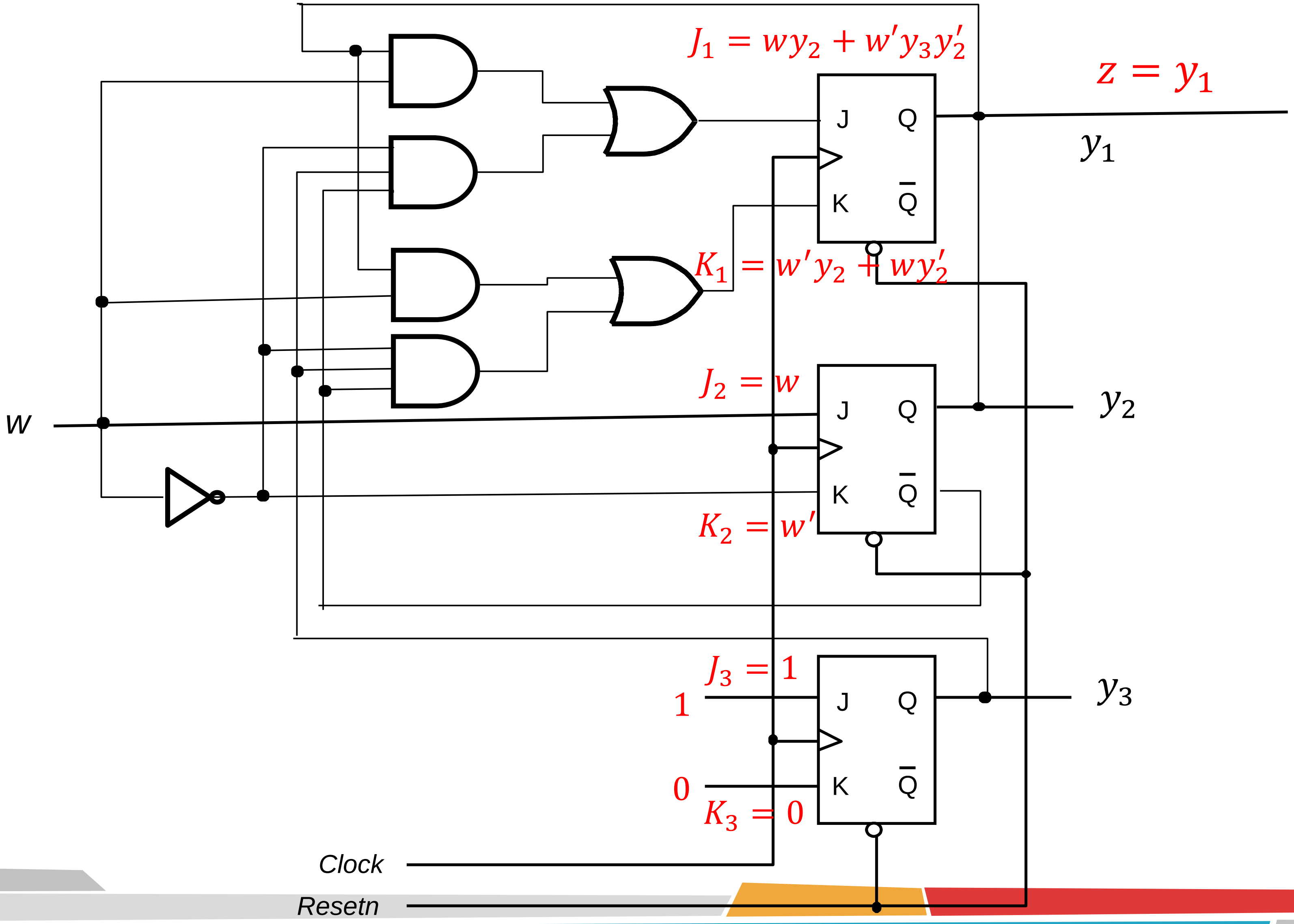
wy_3 \ y_2y_1	00	01	11	10
00	d	d	d	d
01	0	0	0	0
11	0	0	0	0
10	d	d	d	d

$K_3 = 0$

y_1 \ y_3y_2	00	01	11	10	
0	0	d	0	0	
1	d	d	1	1	y_1

$z = y_1$

Circuit diagram using JK flip-flops



$J_1 = wy_2 + w'y_3y_2'$

$z = y_1$

$K_1 = w'y_2 + wy_2'$

$J_2 = w$

$K_2 = w'$

$J_3 = 1$

$K_3 = 0$

Example 8.17

Problem: In computer systems it is often desirable to transmit data serially, namely, one bit at a time, to save on the cost of interconnecting cables. This means that parallel data at one end must be transmitted serially, and at the other end the received serial data has to be turned back into parallel form. Suppose that we wish to transmit ASCII characters in this manner. Usually, a character occupies one byte, in which case the eighth bit can either be set to 0 or it can be used to indicate the parity of the other bits to ensure a more reliable transmission.

Example 8.17

Parallel-to-serial conversion can be done by means of a shift register. Assume that a circuit accepts parallel data, $B = b_7, b_6, \cdots, b_0$, representing ASCII characters. Assume also that b_7 is set to 0. The circuit is supposed to generate a parity bit, p, and send it instead of b_7 as a part of the serial transfer. Figure 8.102 gives a possible circuit. An FSM is used to generate the parity bit, which is included in the output stream by using a multiplexer. A three-bit counter is used to determine when the p bit is transmitted, which happens when the count reaches 7. Design the desired FSM.

Example 8.17

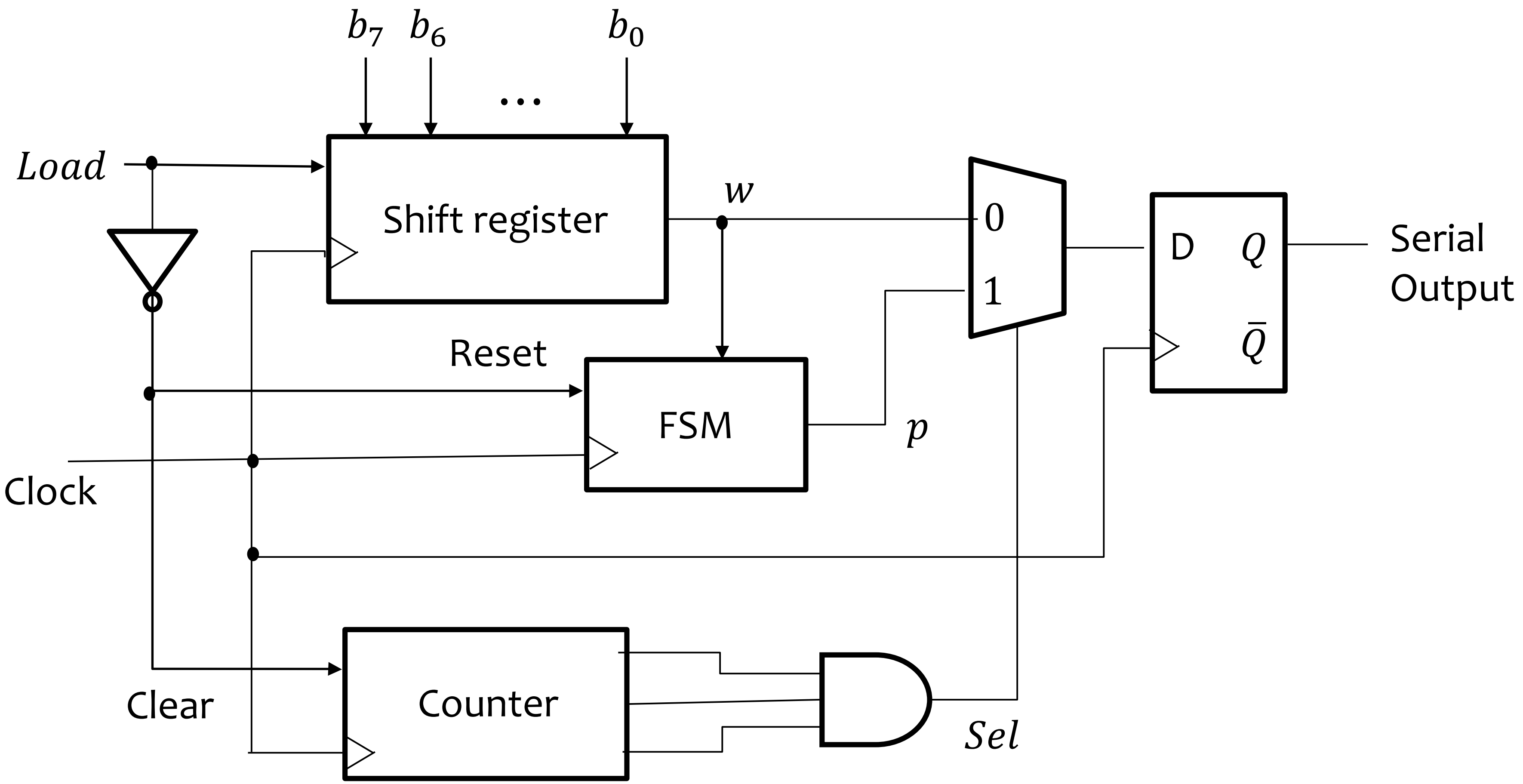
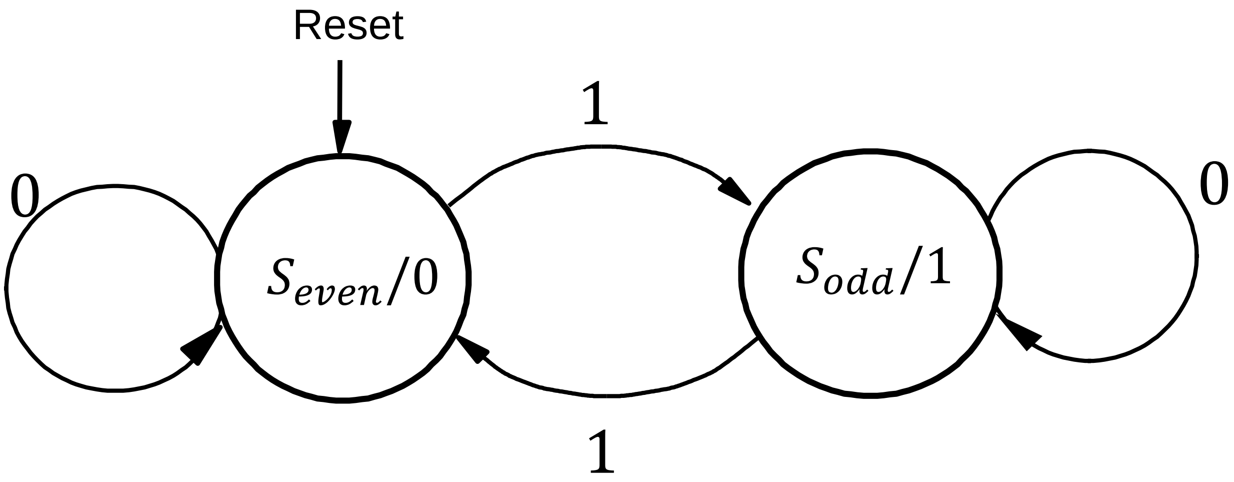


Figure 8.102. Parallel-to-serial converter.

State diagram for a parity generator FSM



Example 8.17

Present state	Next state		Output
	$w = 0$	$w = 1$	p
S_{even}	S_{even}	S_{odd}	0
S_{odd}	S_{odd}	S_{even}	1

(a) State table

Present state	Next state		Output
	$w = 0$	$w = 1$	
y	Y	Y	p
0	0	1	0
1	1	0	1

(b) State-assigned table

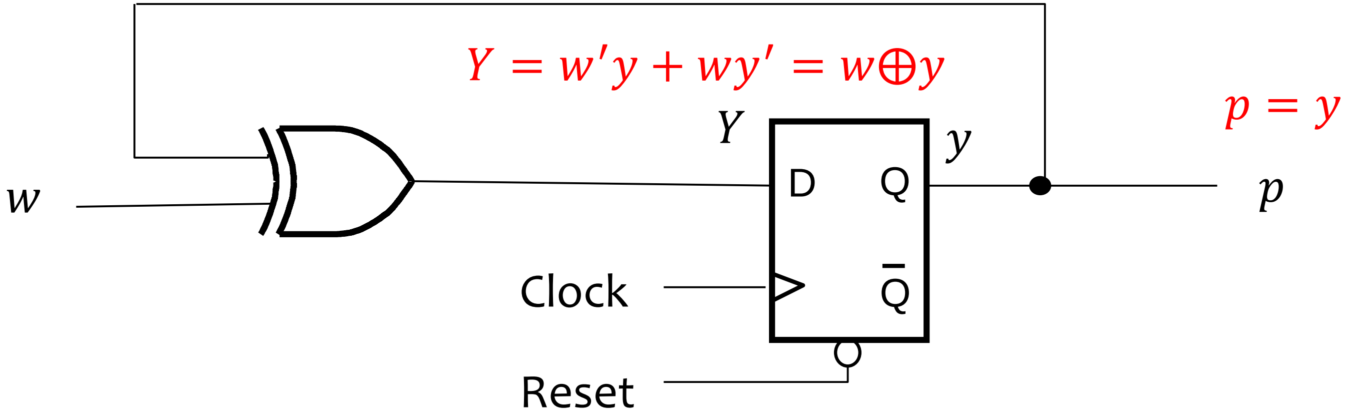
$$\begin{array}{c|cc} {}_{y}\diagdown^{w} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array}$$

$$Y = w'y + wy' = w \oplus y$$

$$p = y$$

Figure 8.103. FSM for parity generation.

Example 8.17



(c) Circuit

Figure 8.103. FSM for parity generation.