Logic Design

Synchronous Sequential Circuits

McMaster University

Introduction

- Combinational circuits: value of each output depends only on the values of inputs
- Sequential Circuits: values of outputs depend on inputs and past behavior of the circuit
- In most cases a clock is used to control the operation of a sequential circuit
- These circuits are called synchronous sequential circuits

- Synchronous sequential circuits are realized using combinational logic and one or more flip-flops
- State: the value of outputs of flip-flops
- Under the control of clock signal, flip-flop outputs change their state as determined by the combinational logic

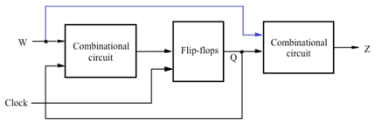


Figure 8.1. The general form of a sequential circuit.

- To ensure that only one transition from one state to another takes place during one clock cycle, flip-flops are edge-triggered
- Outputs are generated by another combinational circuit and are function of present state of the flip-flops and the inputs
- Outputs do not necessarily have to depend directly on the inputs
- Moore type: the output depends only on the state of the circuit
- Mealy type: outputs depend on both the state and the inputs
- Sequential circuits are also called finite state machines (FSM)

Design Example

- Design a circuit that:
  - Has one input (w) and one output (z)
  - All changes occur on the positive edge of the clock
  - Output z is equal to 1 if during the two immediately preceding clock cycles the input w was equal to 1. Otherwise z is equal to 0.

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.

Design Example

- First step in designing a FSM: determine how many states are needed and which transitions are possible from one state to another
- No set procedure for this task
- A good way is select a starting state (a state that the circuit enters when the power is turned on or a reset signal is applied)
- Starting state A
- As long as w is 0, the circuit should remain in A
- When w becomes 1, the machine should move to a different state (B)

State Diagram

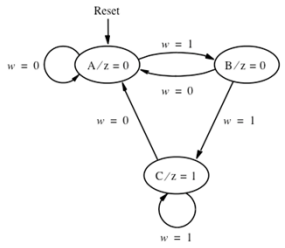


Figure 8.3. State diagram of a simple sequential circuit.

State Table

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

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

Design Example

- When implemented in logic circuits, each state is represented by a particular valuation (combination) of state variables
- Each state variable may be implemented in the form of a flip-flop
- Since there are three states in this example, two state variables are sufficient: y_1 and y_2

Design Example

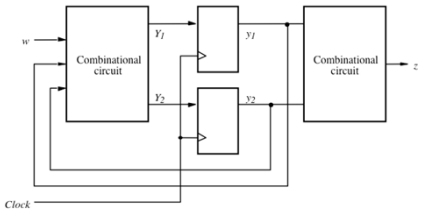


Figure 8.5. A general sequential circuit with input w, output z, and two state flip-flops.

Design Example

- We need to design a combinational circuit with inputs w, y1 and y2 such that for all valuations of these signals Y1 and Y2 will cause the machine to move to the next state
- We create a truth table by assigning specific valuation of variables y1 and y2 to each state

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

Design Example

- Choice of flip-flop:
- Most straightforward choice is to use D flip-flops because the values of Y1 and Y2 are simply clocked into the flip-flops to become the new values of y1 and y2

Design Example

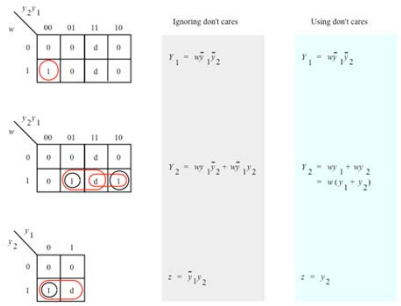


Figure 8.7. Derivation of logic expressions for the sequential circuit in Figure 8.6.

Design Example

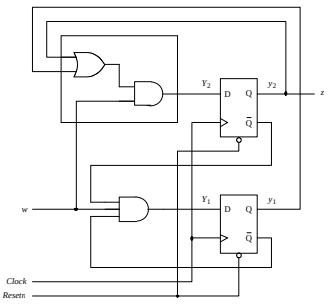


Figure 8.8. Final implementation of the sequential circuit derived in Figure 8.7.

Design Example

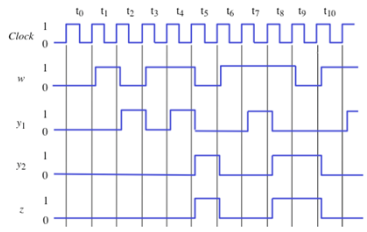


Figure 8.9. Timing diagram for the circuit in Figure 8.8.

Bus controller

- Digital systems often contain a set of registers to store data
- Each register is connected to a common set of n wires, used to transfer data into and out of registers
- This common set of wires is called a bus
- In addition to registers other types of circuits would be connected to the bus
- It is essential to ensure that only one circuit block attempts to place data onto the bus wires at any given time
- A control circuit is used to ensure that only one of the tri-state buffers enables is asserted at a given time

Bus Controller

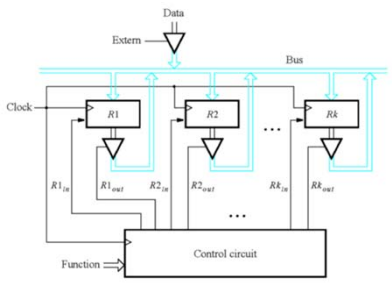


Figure 7.55. A digital system with *k* registers.

Bus controller

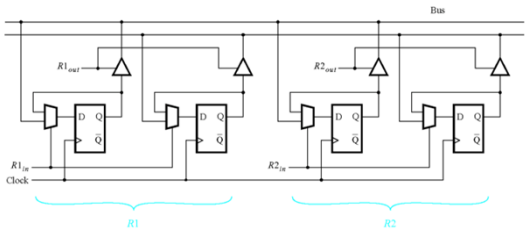


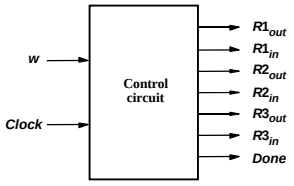
Figure 7.56. Details for connecting registers to a bus.

Bus controller

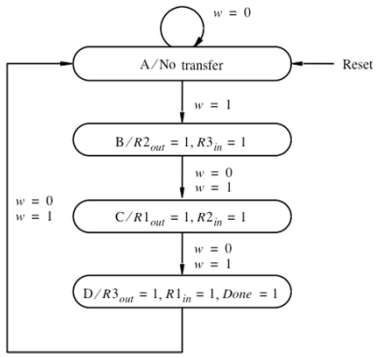
- An example: consider a system that has three registers, R1, R2 andR3. We want to swap the content of R1 and R2
- Steps:
  - Copy R2 to R3
  - Copy R1 to R2
  - Transfer R3 to R1

Bus controller

- Content of R2 is loaded into R3 using $R2_{out}$=1, $R3_{in}$=1
- Content of R1 is transferred into R2 using $R1_{out}$=1, $R2_{in}$=1
- Content of R3 is transferred into R1 using $R3_{out}$=1, $R1_{in}$=1
- We will indicate the completion of the task by setting a signal Done=1



Bus controller



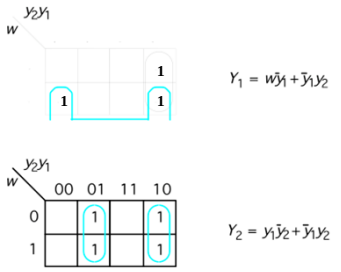
Bus controller

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

Bus controller

	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	01	0	0	0	0	0	0	0
B	01	10	10	0	0	1	0	0	1	0
C	10	11	11	1	0	0	1	0	0	0
D	11	00	00	0	1	0	0	1	0	1

Bus controller



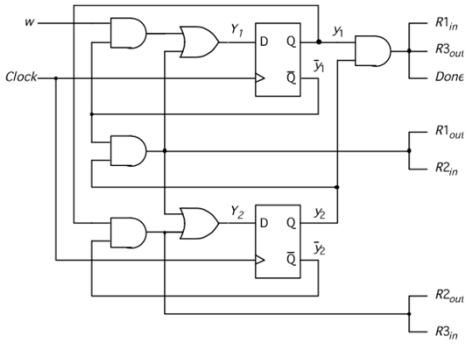
Bus controller

$$R1_{out} = R2_{in} = \bar{y}_1 y_2$$

$$R1_{in} = R3_{out} = Done = y_1 y_2$$

$$R2_{out} = R3_{in} = y_1 \bar{y}_2$$

Bus controller



State Assignment Problem

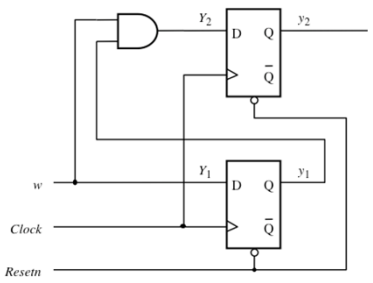
- Some state assignment might be better than the others
- It is often impossible to find the best state assignment for a large circuit
- Exhaustive search is not practical because the number of available state assignments is huge

State Assignment Problem

- Y1=D1=w
- Y2=D2=wy1
- z=y2

	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	11	0
C	11	00	11	1
	10	*dd*	*dd*	*d*

State Assignment Problem



State Assignment Problem

- We now consider a different state assignment for the bus controller example

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

State Assignment Problem

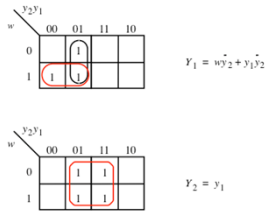


Figure 8.19. Derivation of next-state expressions for the sequential circuit in Figure 8.18.

One-hot coding

- One possibility is to use as many state variables as there are states
- For each state all but one of the state variables are equal to 0
- This approach is known as one-hot coding

One-hot coding

- Y1=w'
- Y2=wy_1
- Y3=wy_1'
- z=y_3

	Present state $y_3y_2y_1$	Nextstate $w=0$ $Y_3Y_2Y_1$	Nextstate $w=1$ $Y_3Y_2Y_1$	Output z
A	001	001	010	0
B	010	001	100	0
C	100	001	100	1

One-hot coding

	Present state $y_4y_3y_2y_1$	Nextstate $w=0$ $Y_4Y_3Y_2Y_1$	Nextstate $w=1$ $Y_4Y_3Y_2Y_1$	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
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

One-hot coding

$$Y_1 = w'y_1 + y_4$$
$$Y_2 = wy_1$$
$$Y_3 = y_2$$

$$R1_{out} = R2_{in} = y_3$$
$$R1_{in} = R3_{out} = Done = y_4$$
$$R2_{out} = R3_{in} = y_2$$

Mealy State Model

- Mealy state machine: output values are generated based on both the state and the inputs
- Example: design a sequential circuit that the output z is equal to 1 in the same clock cycle when the second occurrence of w (input) is detected

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

Mealy State Model

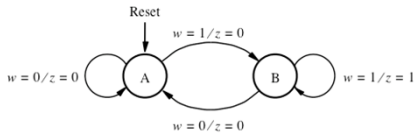


Figure 8.23. State diagram of an FSM that realizes the task in Figure 8.22.

Mealy State Model

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 8.24. State table for the FSM in Figure 8.23.

Mealy State Model

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

Mealy State Model

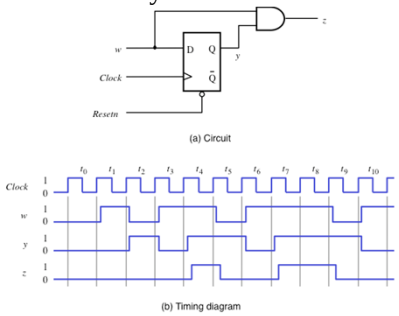


Figure 8.26. Implementation of FSM in Figure 8.25.

Mealy State Model

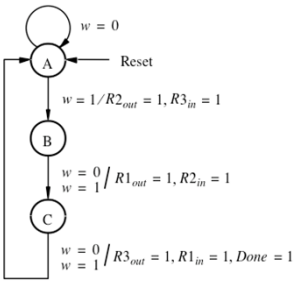


Figure 8.28. State diagram for Example 8.4.

Serial Adder

- If speed is not of great importance, a cost-effective option is to use a serial adder
- Serial adder: bits are added a pair at a time (in one clock cycle)
- A=an-1an-2…a0, B=bn-1bn-2…b0

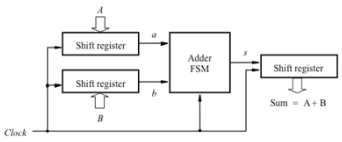


Figure 8.39. Block diagram for the serial adder.

Serial Adder

- G: state that the carry-in is 0
- H: state that the carry-in is1

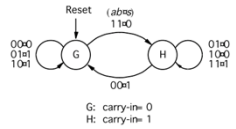


Figure 8.40. State diagram for the serial adder FSM.

Serial Adder

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.

Serial Adder

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 8.42. State-assigned table for Figure 8.41.

$$Y = ab + ay + by$$

$$s = a \oplus b \oplus c$$

Serial Adder

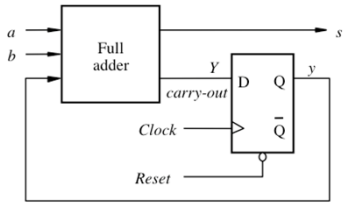


Figure 8.43. Circuit for the adder FSM in Figure 8.39.

Moore-type serial adder

- Since in both states G and H, it is possible to generate two outputs depending on the input, a Moore-type FSM will need more than two states
- G0 and G1: carry is 0 sum is 0 or 1
- H0 andH1: carry is 1 sum is 0 or 1

Moore-type serial adder

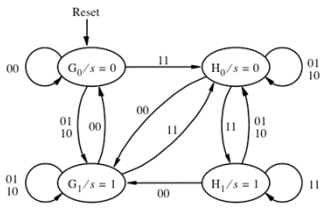


Figure 8.44. State diagram for the Moore-type serial adder FSM.

Moore-type serial adder

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

Moore-type serial adder

Present state y_2y_1	Nextstate				Output
	ab =00	01	10	11	s
	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.

$$Y_2 = ab + ay_2 + by_2$$
$$Y_1 = a \oplus b \oplus c$$
$$s = y_1$$

Moore-type serial adder

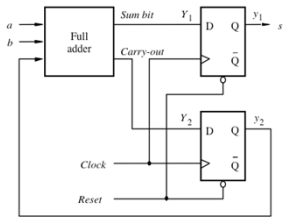


Figure 8.47. Circuit for the Moore-type serial adder FSM.

Counter design using sequential circuits

- Counting sequence: 0,1,2,3,4,5,6,7,0,1,..
- Input signal w: if w=1 count is incremented, if w=0 count is frozen

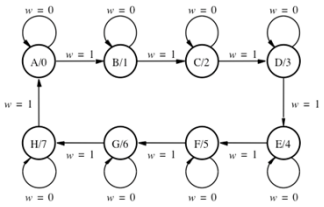


Figure 8.60. State diagram for the counter.

State table

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

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

Figure 8.62. State-assigned table for the counter.

Implementation using D flip-flop

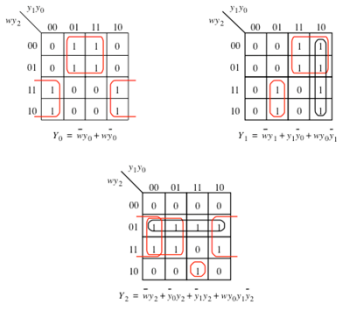


Figure 8.63. Karnaugh maps for D flip-flops for the counter.

$$D_0 = Y_0 = w'y_0 + wy_0'$$
$$D_1 = Y_1 = w'y_1 + y_1y_0' + wy_0y_1'$$
$$D_2 = Y_2 = w'y_2 + y_2y_0' + y_2y_1' + wy_0y_1y_2'$$

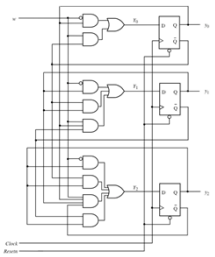


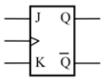
Figure 8.64. Circuit diagram for the counter implemented with D flip-flops.

Implementation using JK flip-flop

- For a JK flip-flop:
  - If state=0, to remains in 0 J=0, K=d
  - If state=0, to change to 1 J=1, K=d
  - If state=1, to remains in 1 J=d, K=0
  - If state=1, to remains in 0 J=d, K=1

J	K	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q}$(t)

(b) Truth table



(c) Graphical symbol

	Present state $y_2y_1y_0$	Flip-flop inputs $w=0$ $Y_2Y_1Y_0$	J_2K_2	J_1K_1	J_0K_0	$w=1$ $Y_2Y_1Y_0$	J_2K_2	J_1K_1	J_0K_0	Count $z_2z_1z_0$
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.

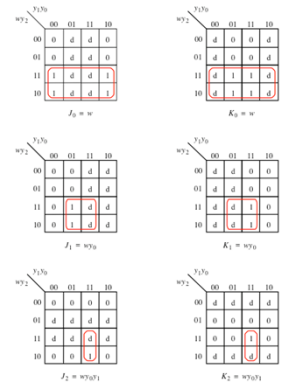


Figure 8.66. Karnaugh maps for JK flip-flops in the counter.

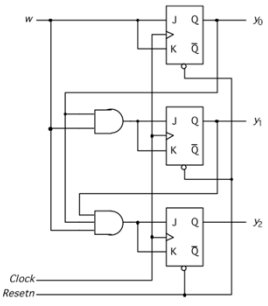


Figure 8.67. Circuit diagram using JK flip-flops.

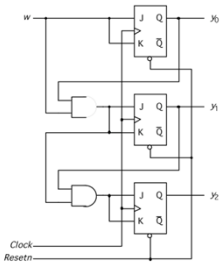


Figure 8.68. Factored-form implementation of the counter.

Analysis of Synchronous Sequential Circuits

- Outputs of flip-flops represent the current state
- Inputs of flip-flops determine the next state
- We can build the state transition table
- Then we build state diagram

Example1

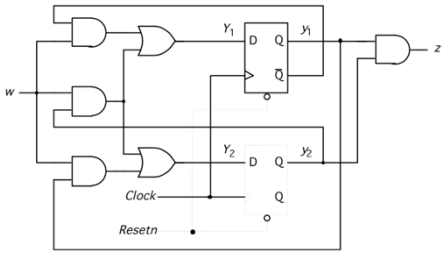


Figure 8.80. Circuit for Example 8.8.

Example 1

$$Y_1 = wy_2 + wy_1'$$
$$Y_2 = wy_1 + wy_2$$
$$z = y_2y_1$$

Present state y_2y_1	Next State $w=0$ Y_2Y_1	Next State $w=1$ Y_2Y_1	Output z
0 0	0 0	0 1	0
0 1	0 0	1 0	0
1 0	0 0	1 1	0
1 1	0 0	1 1	1

(a) State-assigned table

Present state	Next state $w=0$	Next state $w=1$	Output z
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 Example 8.80.

Example 2

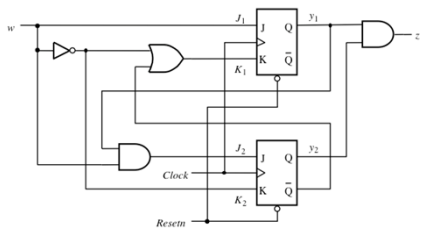


Figure 8.82. Circuit for Example 8.9.

Example 2

$$J_1 = w$$
$$K_1 = w' + y_2$$
$$J_2 = wy_1$$
$$K_2 = w'$$
$$z = y_2y_1$$

Present state y_2y_1	Flip-flop inputs $w=0$ J_2K_2	$w=0$ J_1K_1	$w=1$ J_2K_2	$w=1$ J_1K_1	Output z
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 8.83. The excitation table for the circuit in Figure 8.82.

Example 2

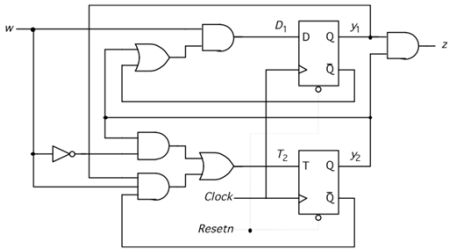
Present state y_2y_1	Next State w = 0 Y_2Y_1	Next State w = 1 Y_2Y_1	Output z
0 0	0 0	01	0
0 1	0 0	10	0
1 0	0 0	11	0
1 1	0 0	11	1

(a) State-assigned table

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

(b) State table

Example 3



Example 3

$D_1 = w(y_2 + y_1')$
$T_2 = wy_1y_2' + w'y_2$
$z = y_2y_1$

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

Example 3

Present state y_2y_1	Next State w = 0 Y_2Y_1	Next State w = 1 Y_2Y_1	Output z
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 w = 0	Next state w = 1	Output z
A	A	B	0
B	A	C	0
C	A	D	0
D	A	D	1

(b) State table