Logic Design

Synchronous Sequential Circuits

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 an done 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

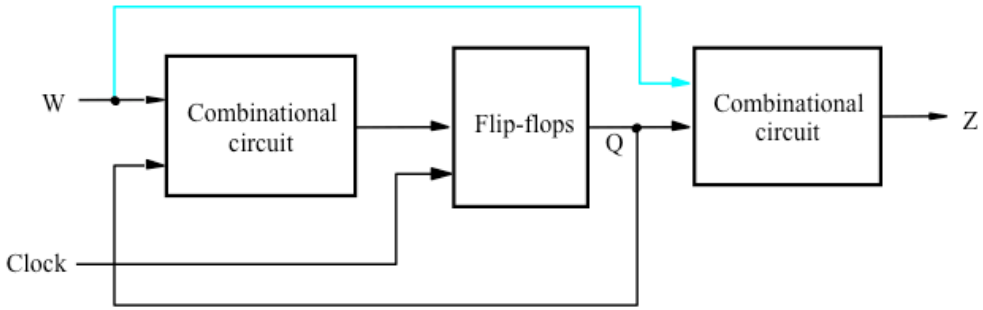


Figure 8.1. The general form of a sequential circuit.

McMaster University

- 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 sate 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.

McMaster University

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

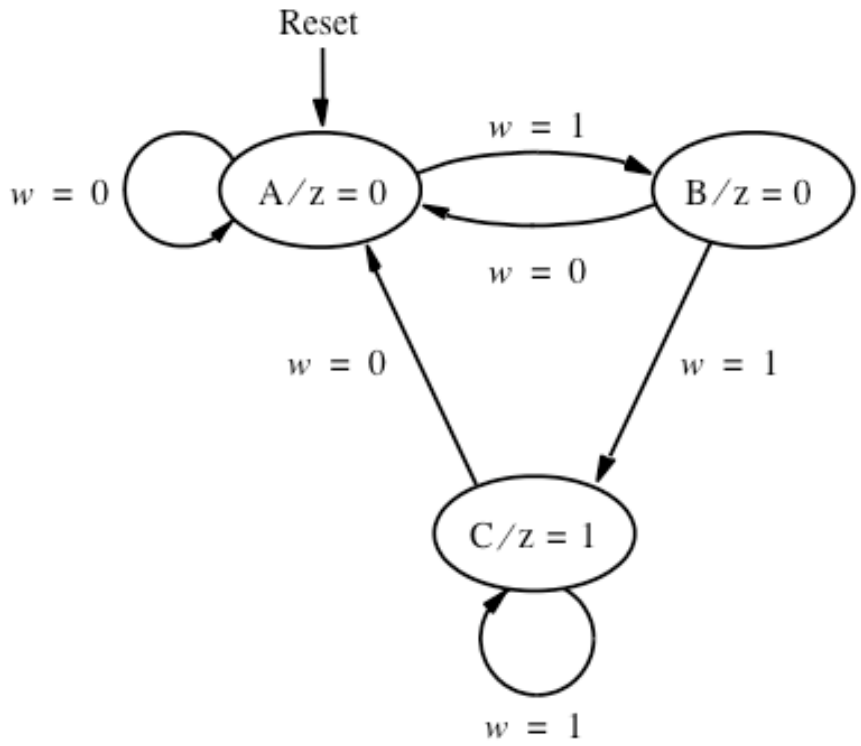


Figure 8.3. State diagram of a simple sequential circuit.

McMaster University

State Table

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

McMaster University

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

McMaster University

Design Example

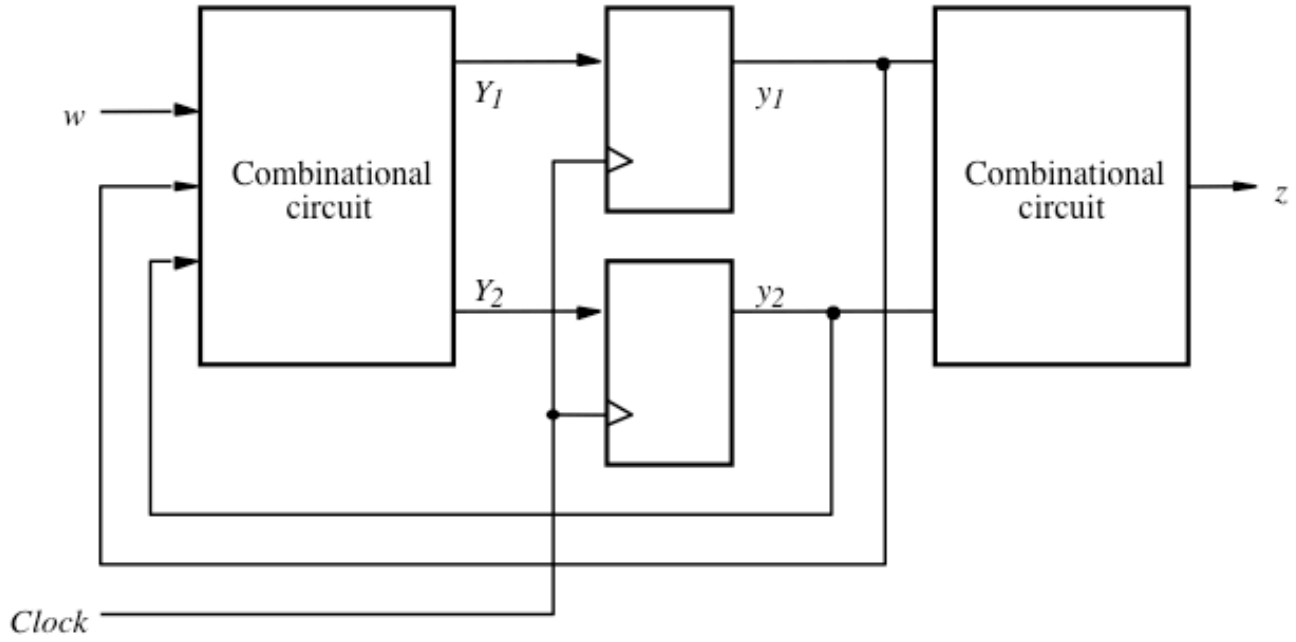


Figure 8.5. A general sequential circuit with input w, output z, and two state flip-flops.

McMaster University

Design Example

- We need to design a combinational circuit with inputs w, y1 and y2 such that for all valuations of these signalsY1 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*

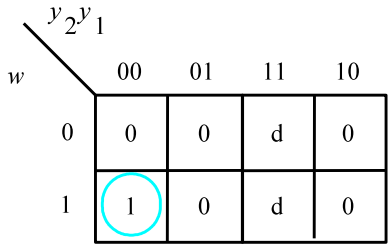
Figure 8.6. State-assigned table for the sequential circuit in Figure 8.4.

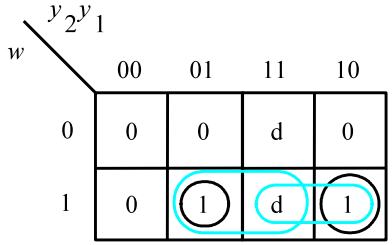
Design Example

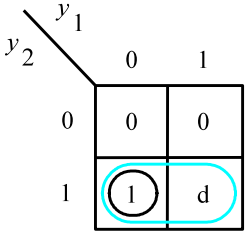
- Choice of flip-flop:
- Most straightforward choice is to use D flip-flops because the values of Y1 and Y2 are simply clocked into the flip-flops to become the new values of y1 and y2

McMaster University

Design Example







Ignoring don't cares	Using don't cares
$Y_1 = w\bar{y}_1\bar{y}_2$	$Y_1 = w\bar{y}_1\bar{y}_2$
$Y_2 = wy_1\bar{y}_2 + w\bar{y}_1y_2$	$Y_2 = wy_1 + wy_2$ $= w(y_1 + y_2)$
$z = \bar{y}_1y_2$	$z = y_2$

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

Design Example

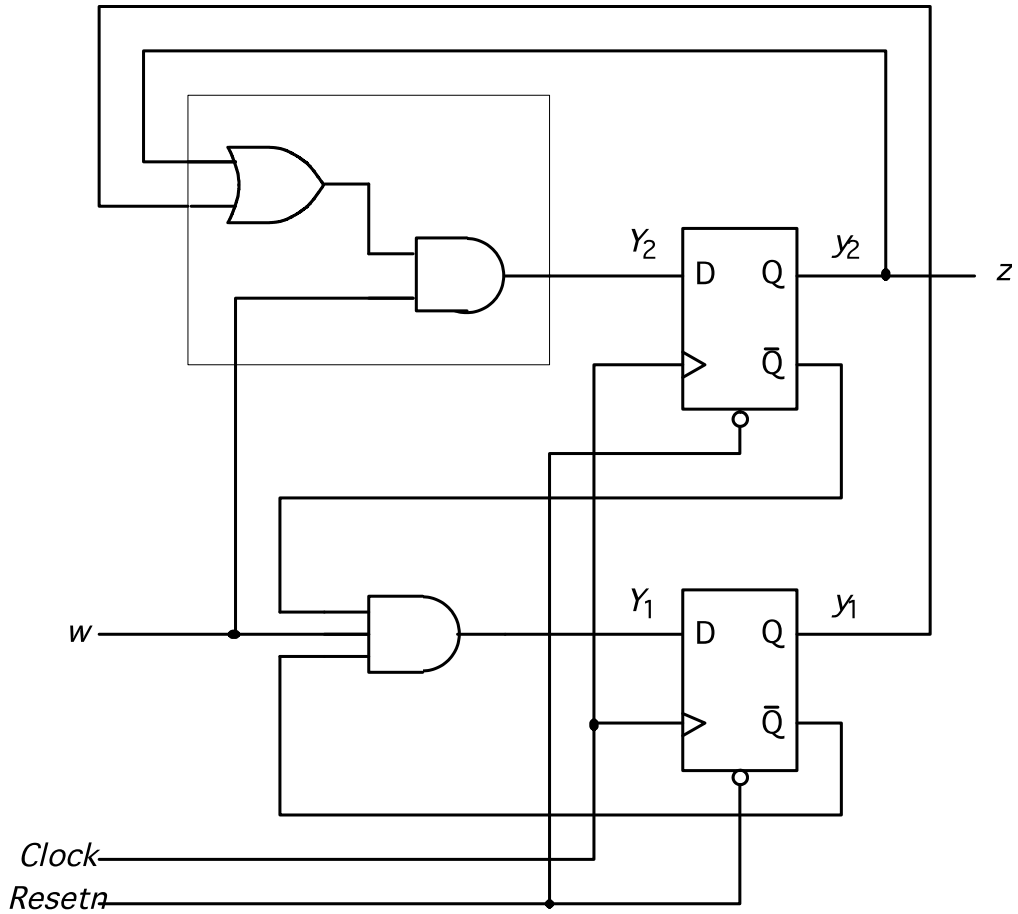


Figure 8.8. Final implementation of the sequential circuit derived in Figure 8.7.

Design Example

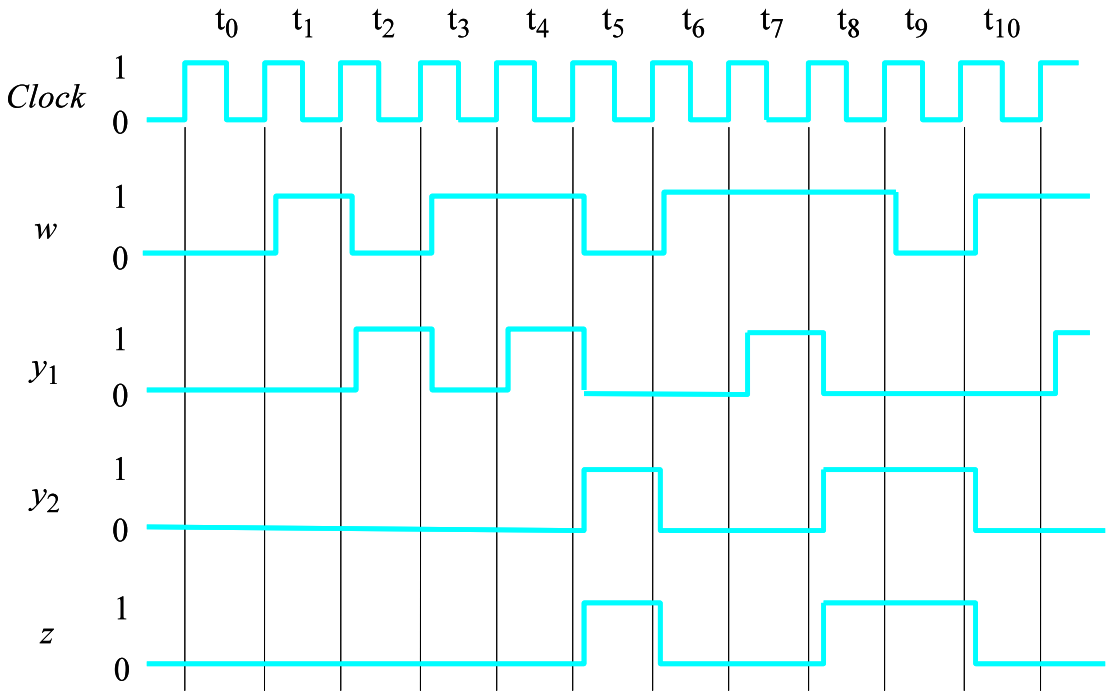


Figure 8.9. Timing diagram for the circuit in Figure 8.8.

McMaster University

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 tome
- A control circuit is used to ensure that only one of the tri-state buffers enables is asserted at a given time

McMaster University

Bus Controller

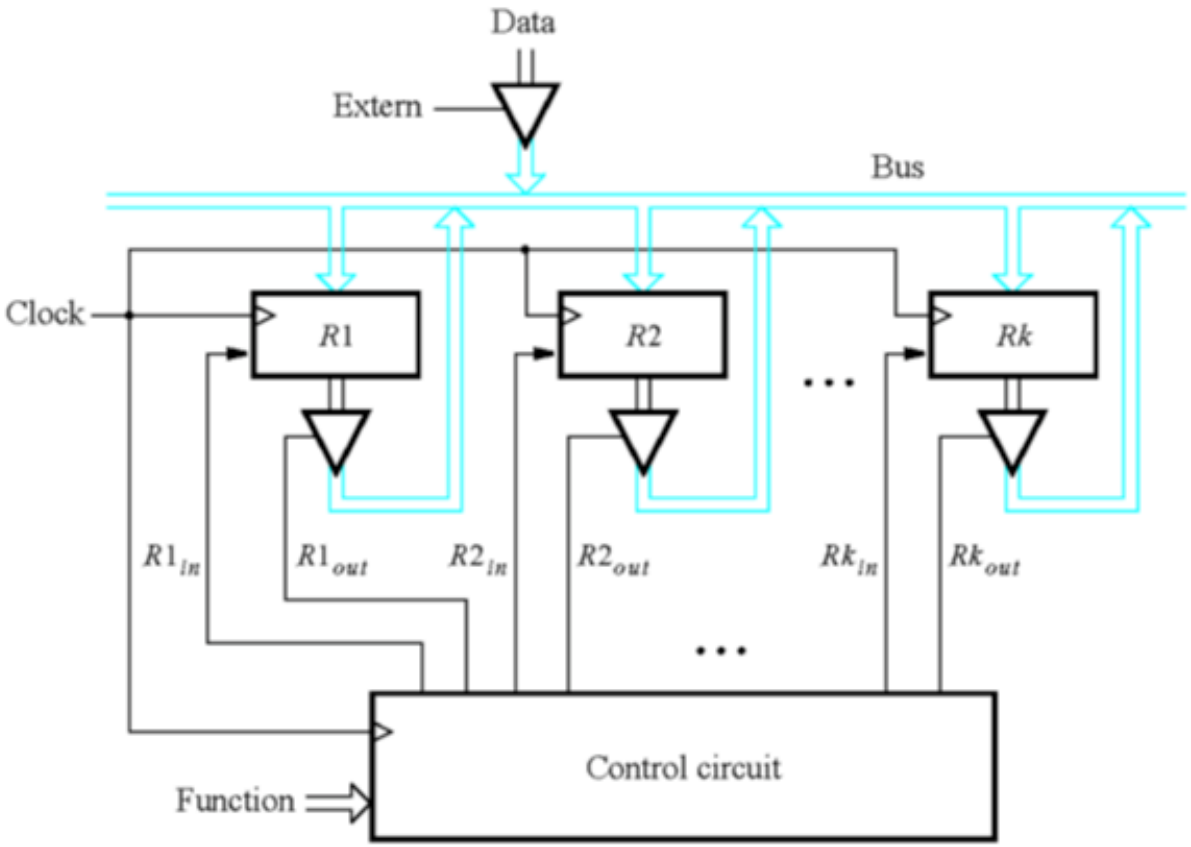


Figure 7.55. A digital system with *k* registers.

McMaster University

Bus controller

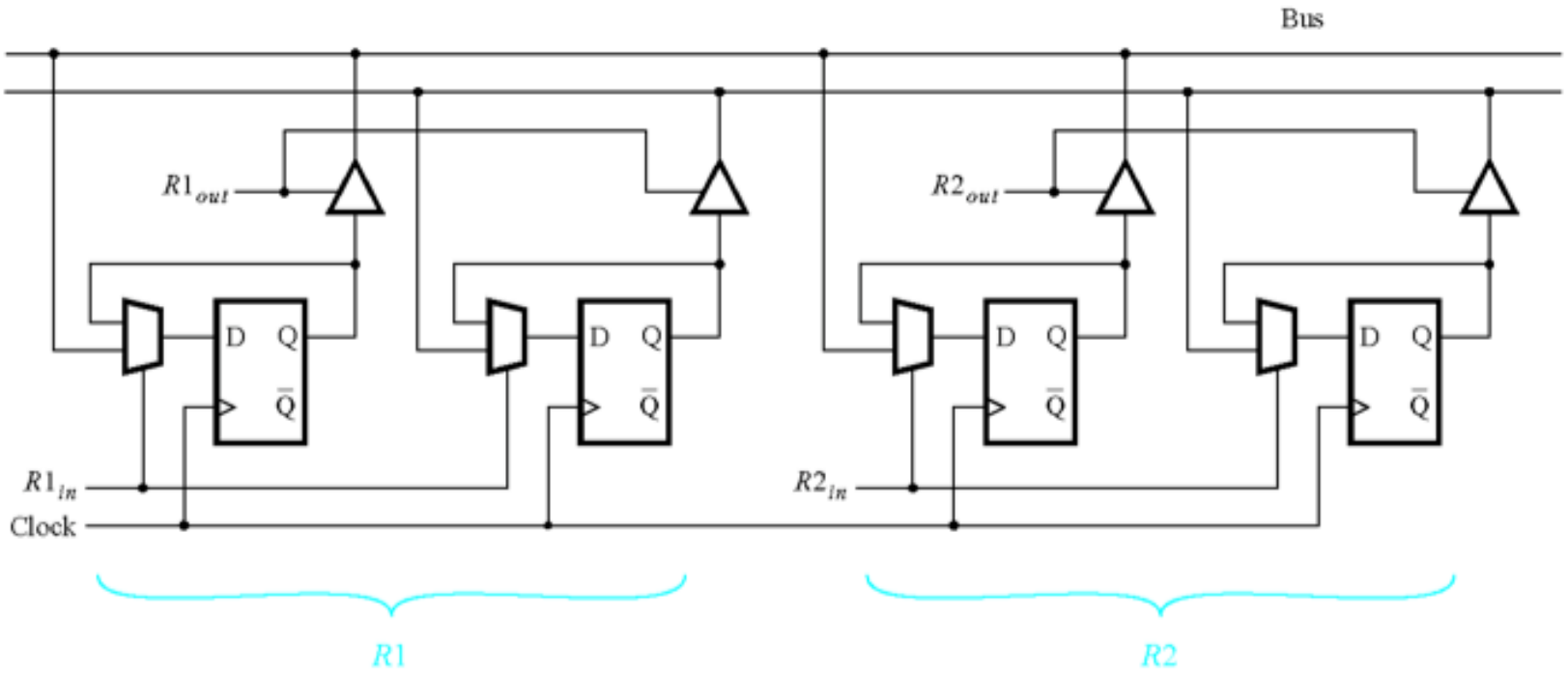


Figure 7.56. Details for connecting registers to a bus.

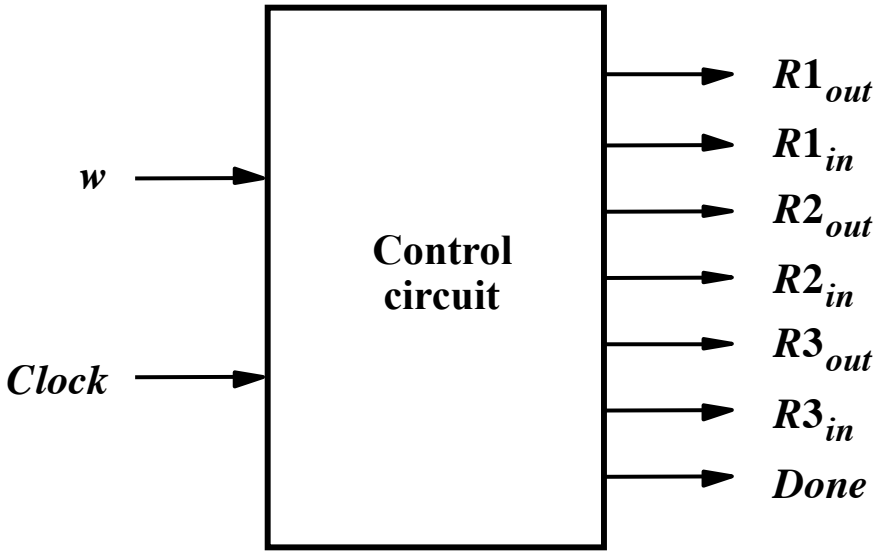
McMaster University

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

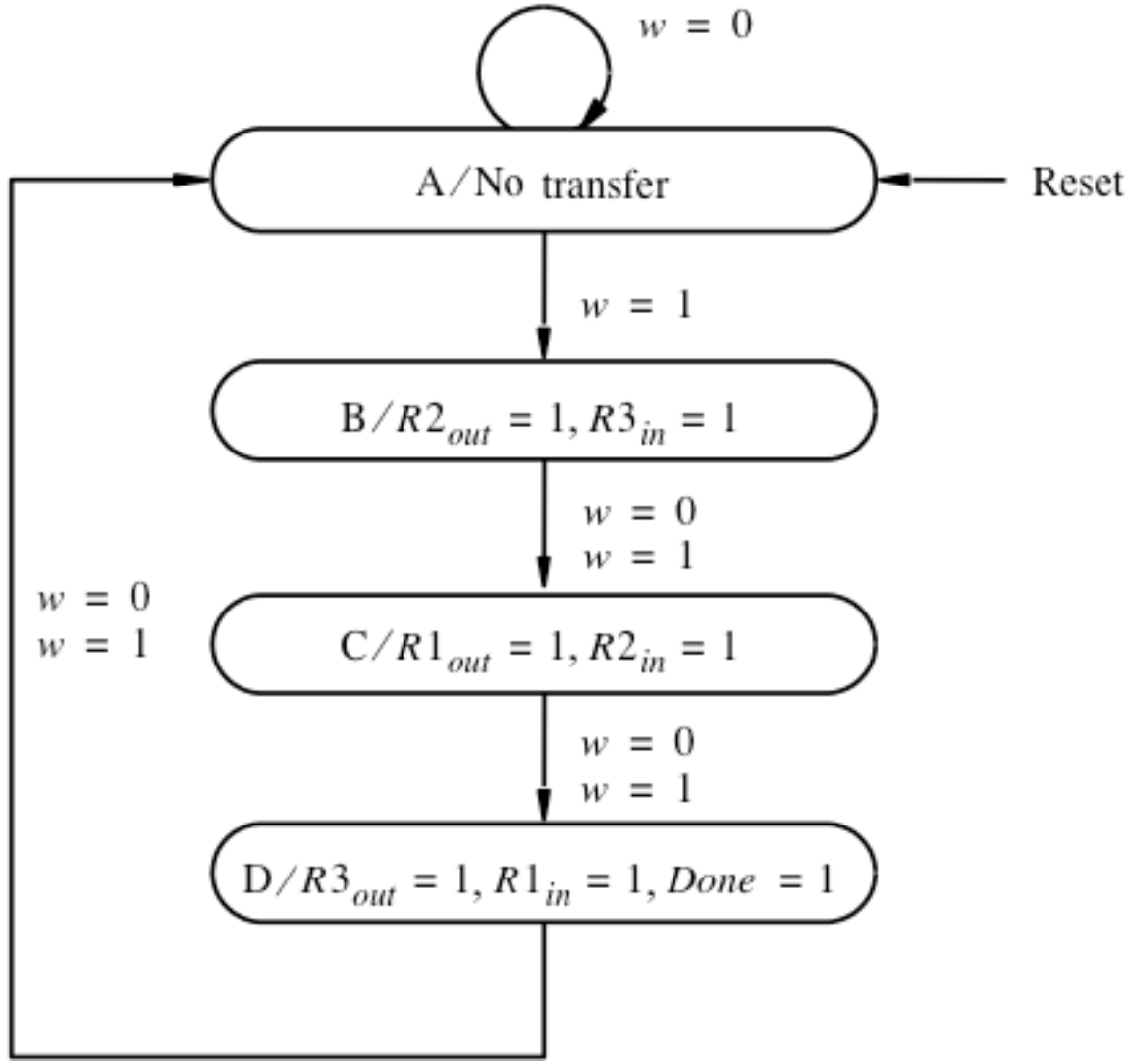
Bus controller

- Content of R2 is loaded into R3 using R2out=1, R3in=1
- Content of R1 is transferred into R2 using R1out=1, R2in=1
- Content of R3 is transferred into R1 using R3out=1, R1in=1
- We will indicate the completion of the task by setting a signal Done=1



McMaster University

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

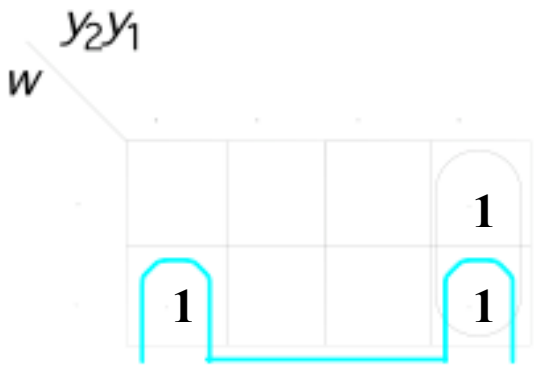
McMaster University

Bus controller

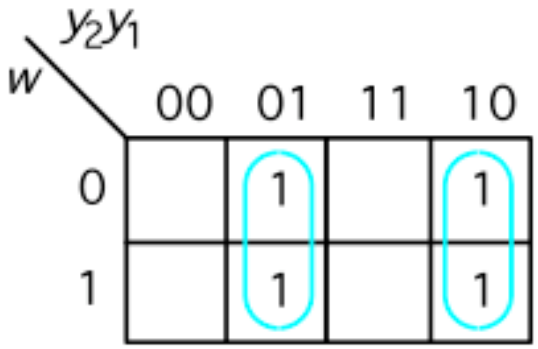
	Present state	Next state		Outputs						
		$w = 0$	$w = 1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	0 1	0	0	0	0	0	0	0
B	01	10	1 0	0	0	1	0	0	1	0
C	10	11	1 1	1	0	0	1	0	0	0
D	11	00	0 0	0	1	0	0	1	0	1

McMaster University

Bus controller



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



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

McMaster University

Bus controller

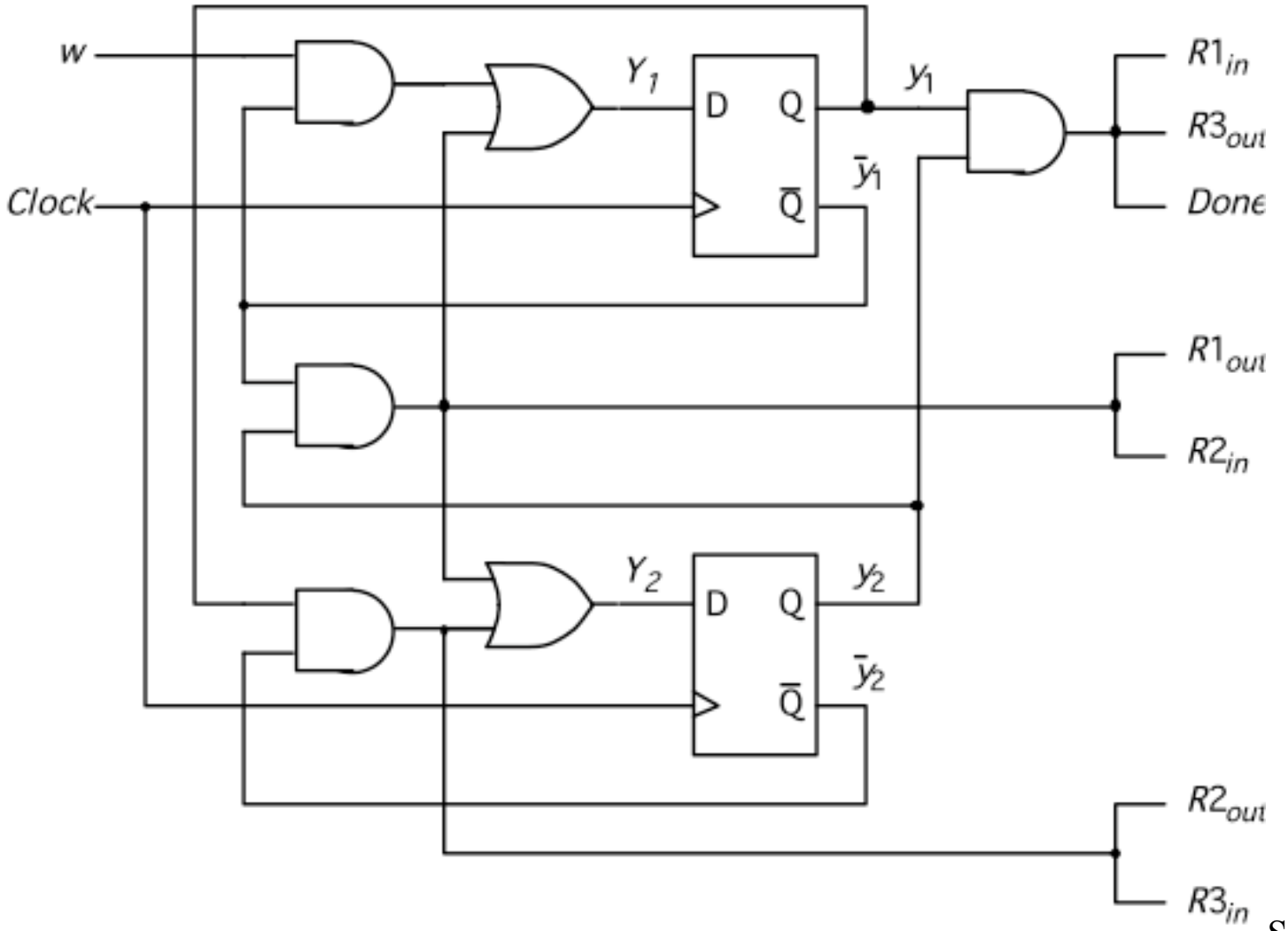
$$R1out = R2in = \overline{y_1} y_2$$

$$R1in = R3out = Done = y_1 y_2$$

$$R2out = R3in = y_1 \overline{y_2}$$

McMaster University

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

McMaster University

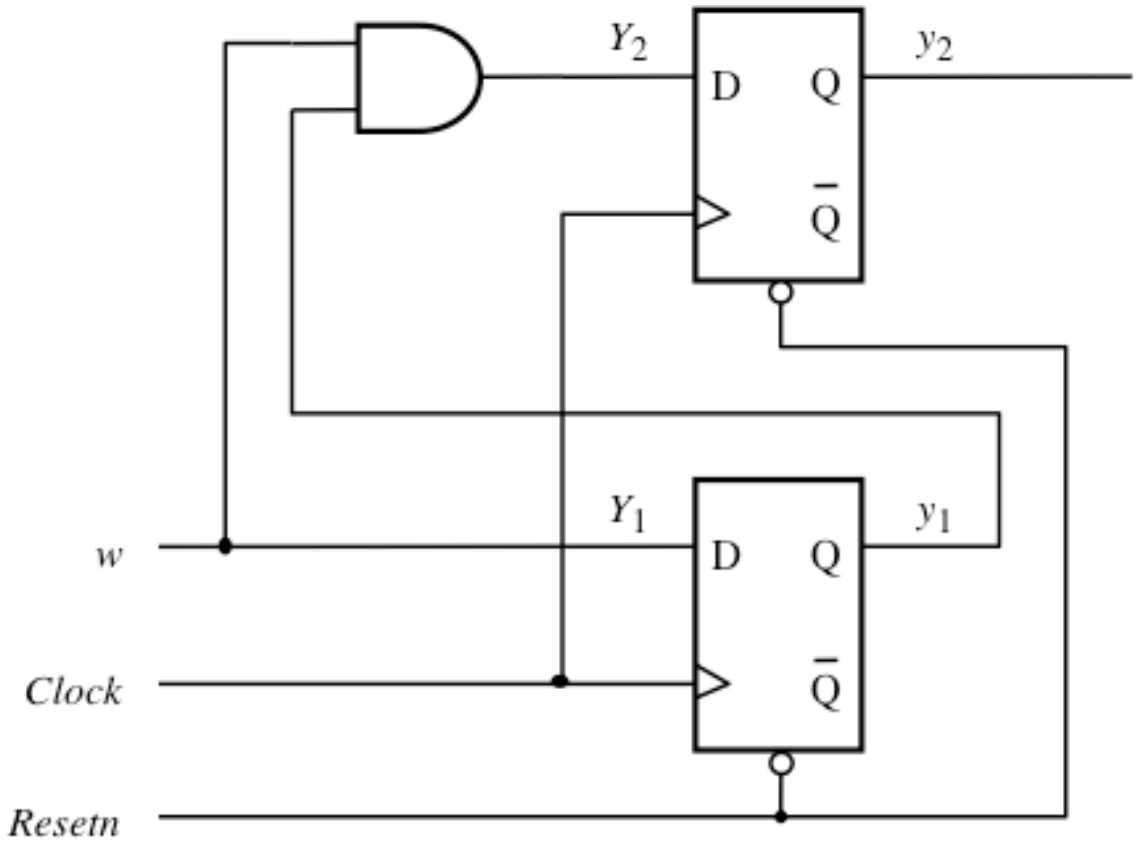
State Assignment Problem

- 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*

McMaster University

State Assignment Problem



McMaster University

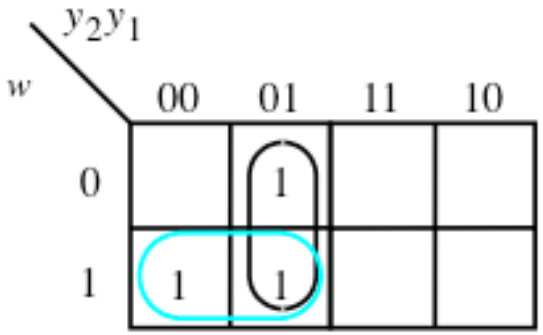
State Assignment Problem

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

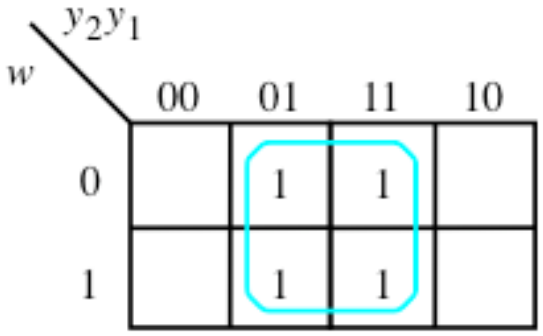
	Present state	Nextstate		Outputs						
		$w=0$	$w=1$							
	y_2y_1	Y_2Y_1	Y_2Y_1	$R1_{out}$	$R1_{in}$	$R2_{out}$	$R2_{in}$	$R3_{out}$	$R3_{in}$	$Done$
A	00	00	01	0	0	0	0	0	0	0
B	01	11	11	0	0	1	0	0	1	0
C	11	10	10	1	0	0	1	0	0	0
D	10	00	00	0	1	0	0	1	0	1

McMaster University

State Assignment Problem



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



$$Y_2 = y_1$$

McMaster University

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

McMaster University

One-hot coding

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

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

McMaster University

One-hot coding

	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

McMaster University

One-hot coding

$$Y1 = \overline{w}y1 + y4$$

$$Y2 = wy1$$

$$Y3 = y2$$

$$Y4 = y3$$

$$R1out = R2in = y3$$

$$R1in = R3out = Done = y4$$

$$R2out = R3in = y2$$

McMaster University

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

McMaster University

Mealy State Model

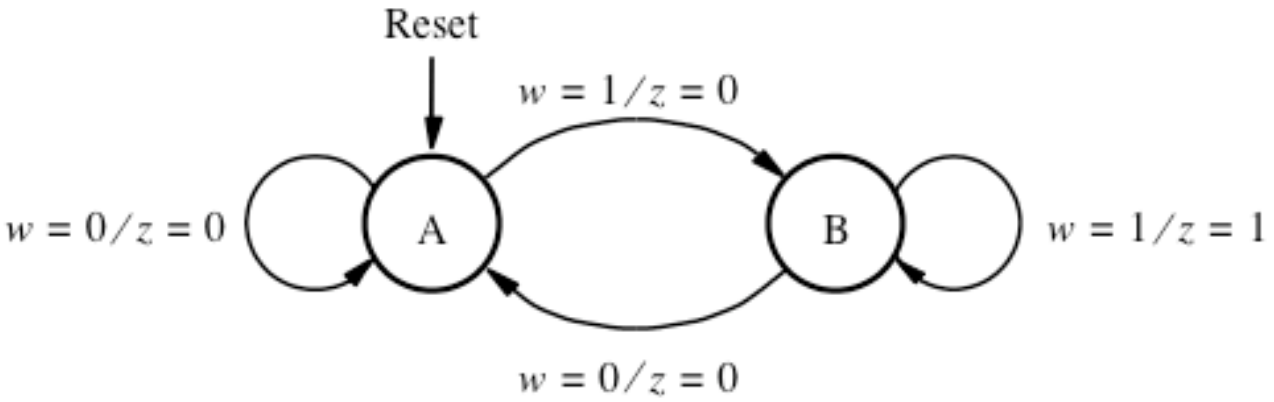


Figure 8.23. State diagram of an FSM that realizes the task in Figure 8.22.

McMaster University

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.

McMaster University

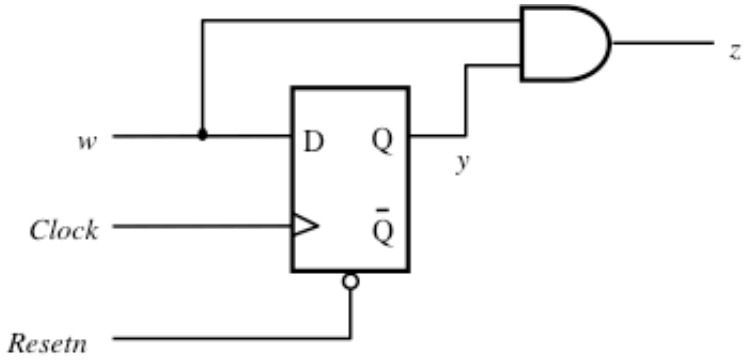
Mealy State Model

	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

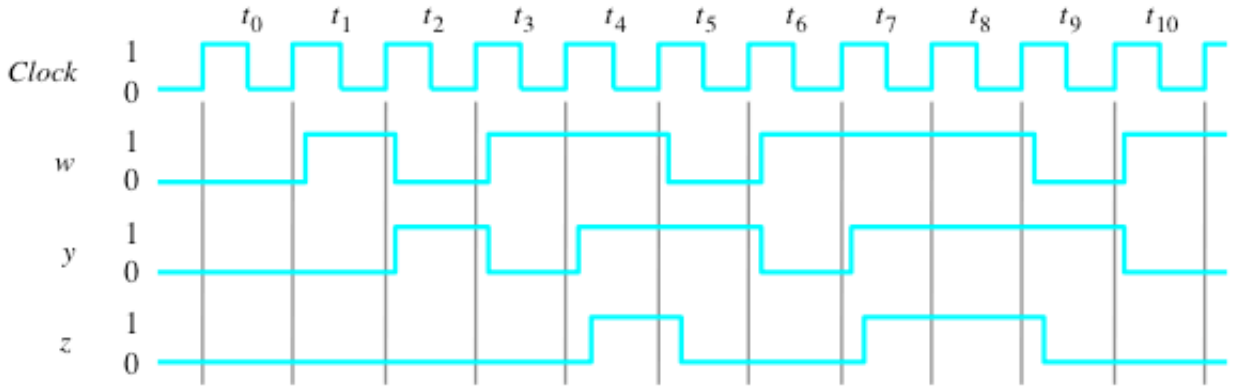
Figure 8.25. State-assigned table for the FSM in Figure 8.24.

McMaster University

Mealy State Model



(a) Circuit



(b) Timing diagram

Figure 8.26. Implementation of FSM in Figure 8.25.

Mealy State Model

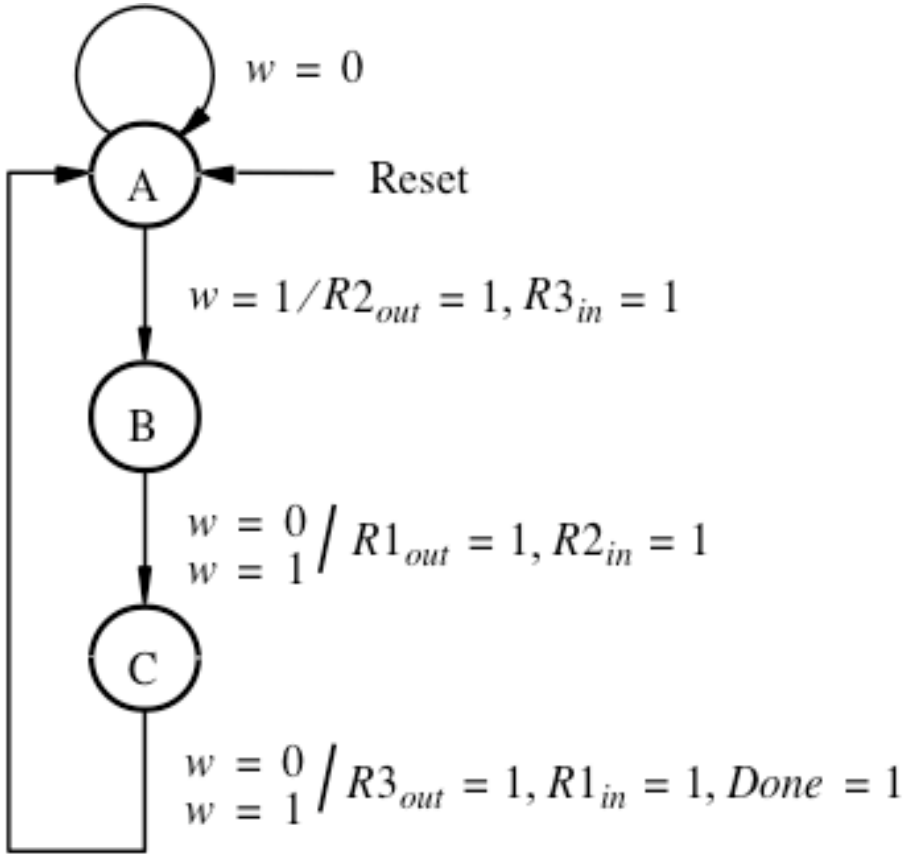


Figure 8.28. State diagram for Example 8.4.

McI University

Mealy State Model

$$Y1 = \overline{w}y1 + y3$$

$$Y2 = wy1$$

$$Y3 = y2$$

McMaster University