Electrical Engineering

Digital Logic Design

A type of
“Synchronous Sequential Networks”
Sometimes referred to as
Finite State Machines
FSM

FSM

- Finite State Machines are systems that combine Combinational Logic Networks with Memory networks, i.e. flip-flops and are similar to the networks we have been looking at.

- The values of the outputs of the flip-flops are referred to as the *state* of the circuit.

- Under control of the clock signal, the flip-flop outputs change state as determined by the network inputs and the combinational logic that feeds the inputs of these flip-flops.

- In this way, the circuit output varies from one state to another.

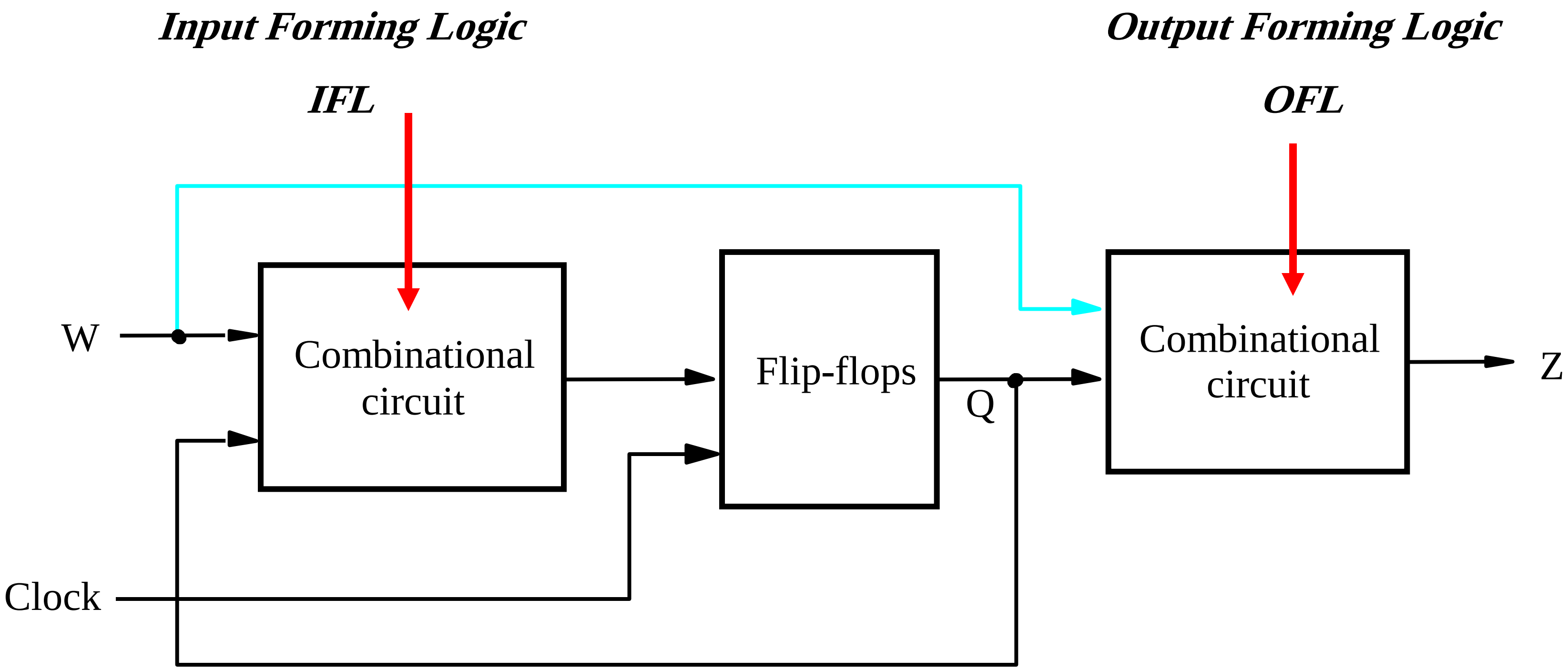
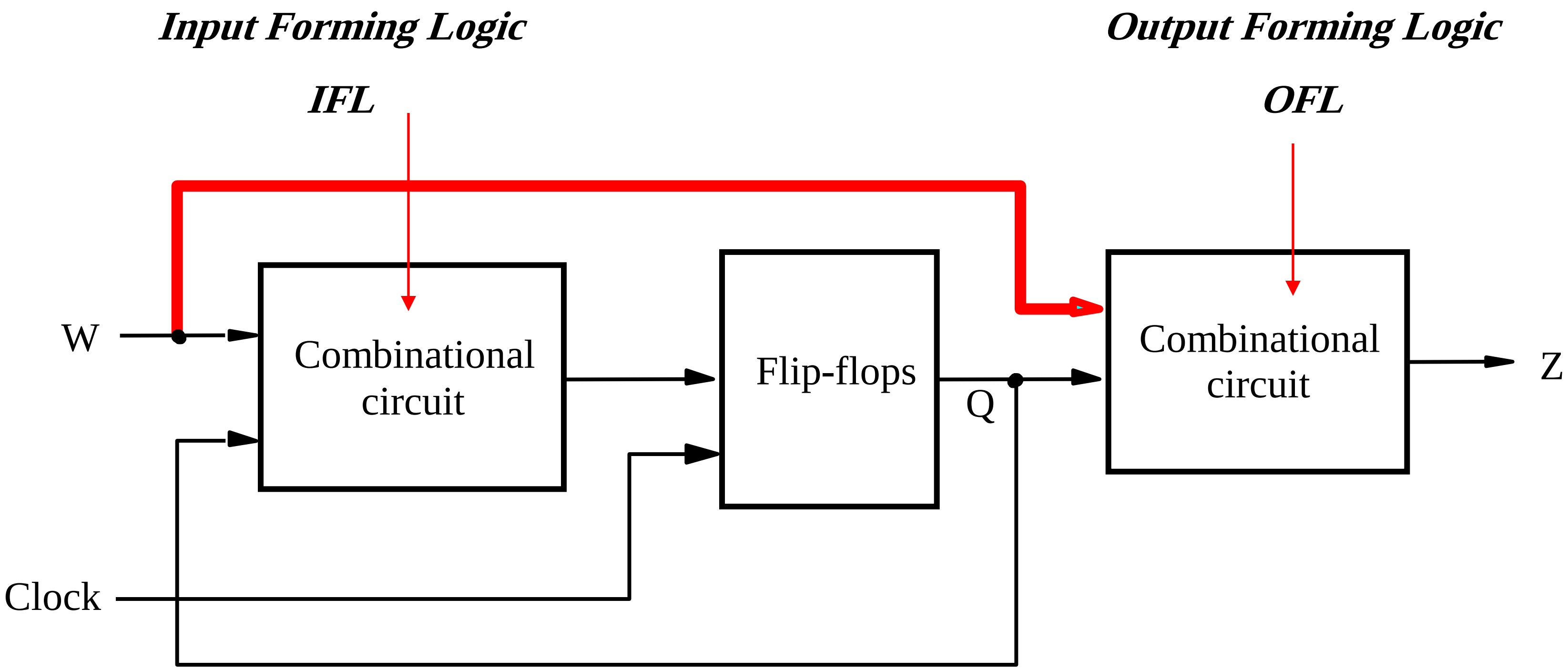


Figure 1. Here is the general form of a sequential circuit.

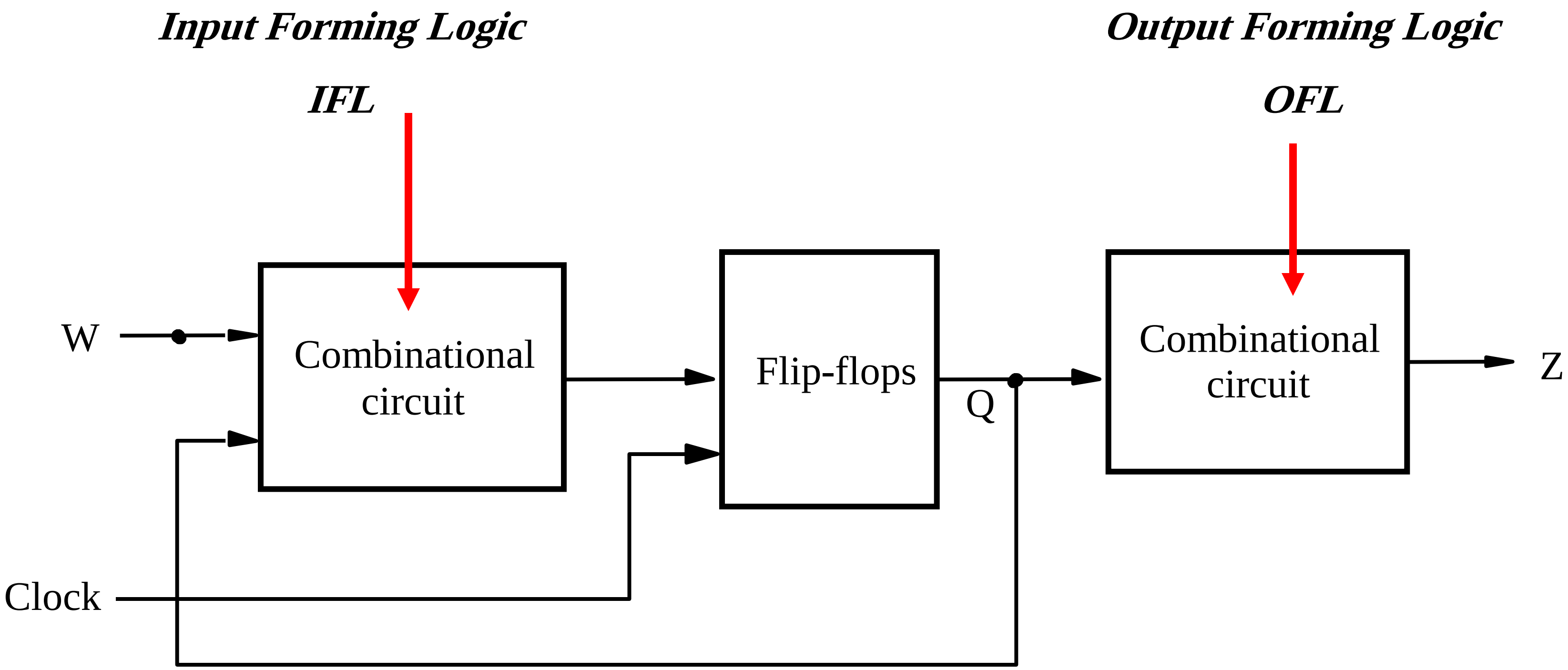
There are Two Types of State Machines

- Mealy Machine (named after George Mealy) in ’50’s
- Moore Machine (named after Edward Moore who expanded on Mealy concept).



Mealy Machine

Input signals **ARE** applied to both the input circuits and the output circuits.



Input Forming Logic

IFL

Output Forming Logic

OFL

W

Combinational circuit

Flip-flops

Q

Combinational circuit

Z

Clock

Moore Machine

Input signals **ARE NOT** applied to the output circuits – only to the input circuit .

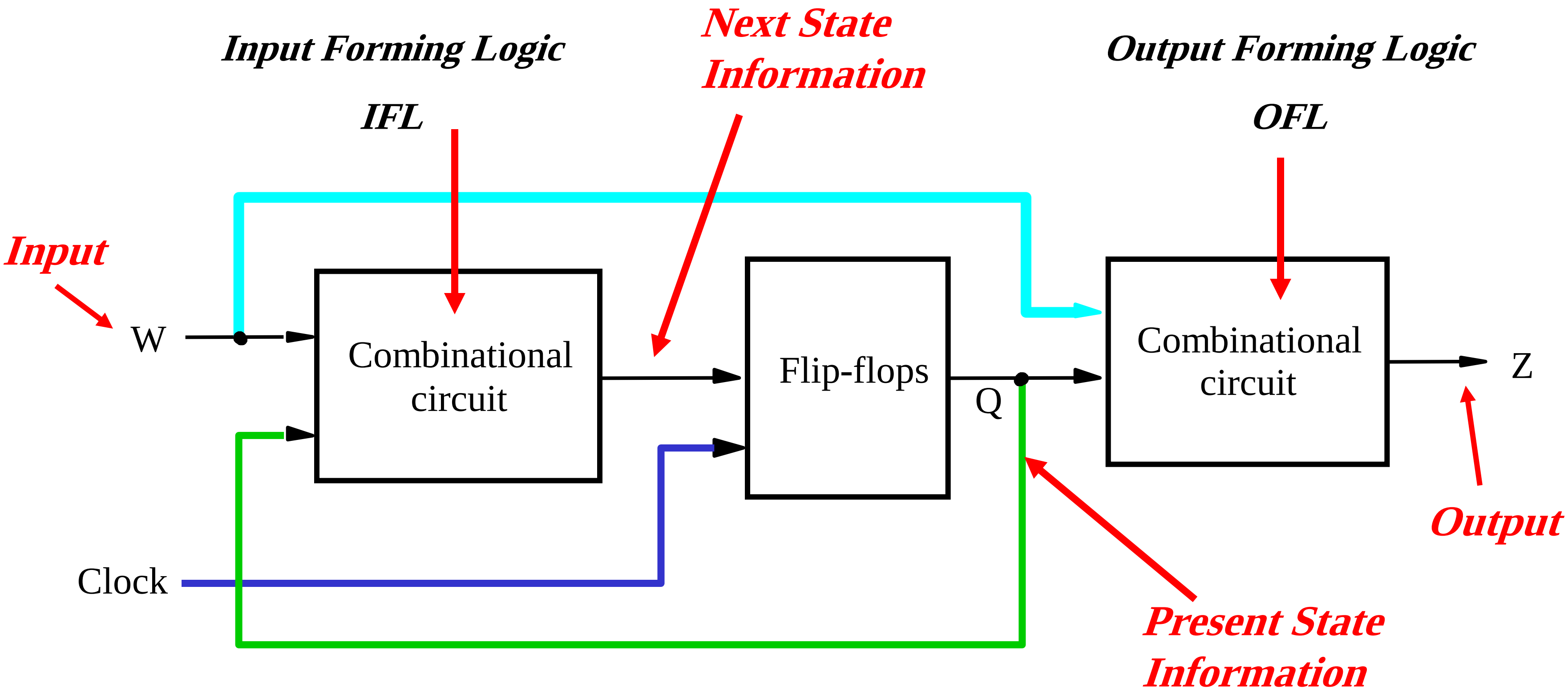
Operation

- To insure that only one transition, from one state to another state, takes place during one clock cycle, flip-flop’s must be EDGE TRIGGERED, and operate on either rising or falling edge.

- The output of the IFL (which is next state information) is formed by the input and current state of the memory

- Thus “state” changes depend on both Input and Present State signals.

- The output of the circuit is formed by either:
  - Inputs and present state of flip-flop’s (Mealy)
  - Present state of flip-flop’s (Moore)



Recall the general form of a sequential circuit.

Lets Look at an example

Before going through the formal process for FSM design/analysis, let’s look at an example just to get a little familiar with how these circuits operate and what these circuits do for us.

Begin with a problem definition and create a diagram that depicts the behavior that we want!

Consider a candy vending machine

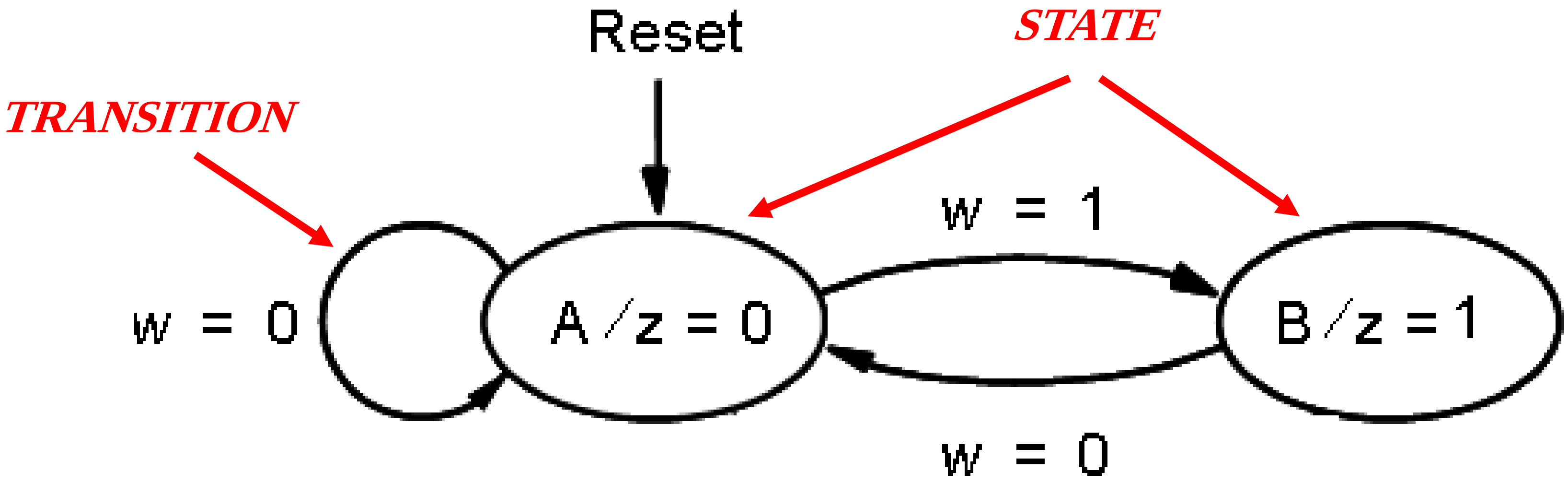
(think of this as a requirements document)

- Requirements for the machine:
  - Dispenses a piece of candy if 15¢ is input
  - Accepts nickels and dimes
  - No change is given
  - Two input signals: Nickels (x) and Dimes (y)
  - Only one coin entered at a time between clock ticks (clock in this case is a signal resulting from the operation of a gate used to sense coins.

Vending machine

- On each clock tick, a transition must occur synchronous with the clock.
  - Let a circle represent the state
  - Let an arc represent a transition between states.

Here is how a STATE diagram might look.



w is an input signal

z is an output signal

State diagram of a simple sequential circuit.

Candy vending machine.

(sketch)

- State Definition:
  - ***A*** – POWER ON State – no money entered
  - ***B*** – Nickel entered – no candy
  - ***C*** – 2 nickels or a dime entered – no candy
  - ***D*** – 15 ¢ entered – dispense candy
- Assume only one coin can be entered at a time.
- If both conditions X and Y could occur at one clock tick, then other paths would have to be considered.

Now lets work through the formal process for a typical FSM design

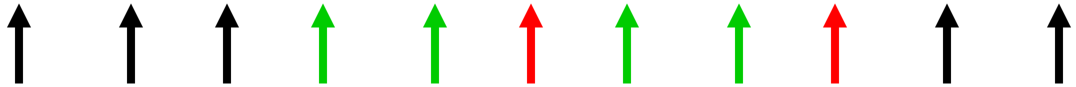
- Consider a circuit that
  - Has one input,w
  - Has one output, z
  - Z=1 if w=1 for 2 or more immediately proceeding consecutive clock cycles (rising edge)
  - Z = 0 if w does not equal 1 for 2 immediately proceeding consecutive clock cycles

- Circuits that detect a specific bit pattern on the input are referred to as *sequence detectors*.

- It is apparent that z cannot depend solely on present state of w.

- ***Our job is to design a circuit that satisfies the above requirement.***

Here is the formal description of the sequence detector



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

If z depended on *w* alone, then it would not be possible for z to be equal to two different values for a single value of the input, or the same value of z for different values of *w*

Figure 2. Sequences of input and output signals.

Here is the STATE diagram that we’ve come up with that satisfies our specification for the sequence detector.

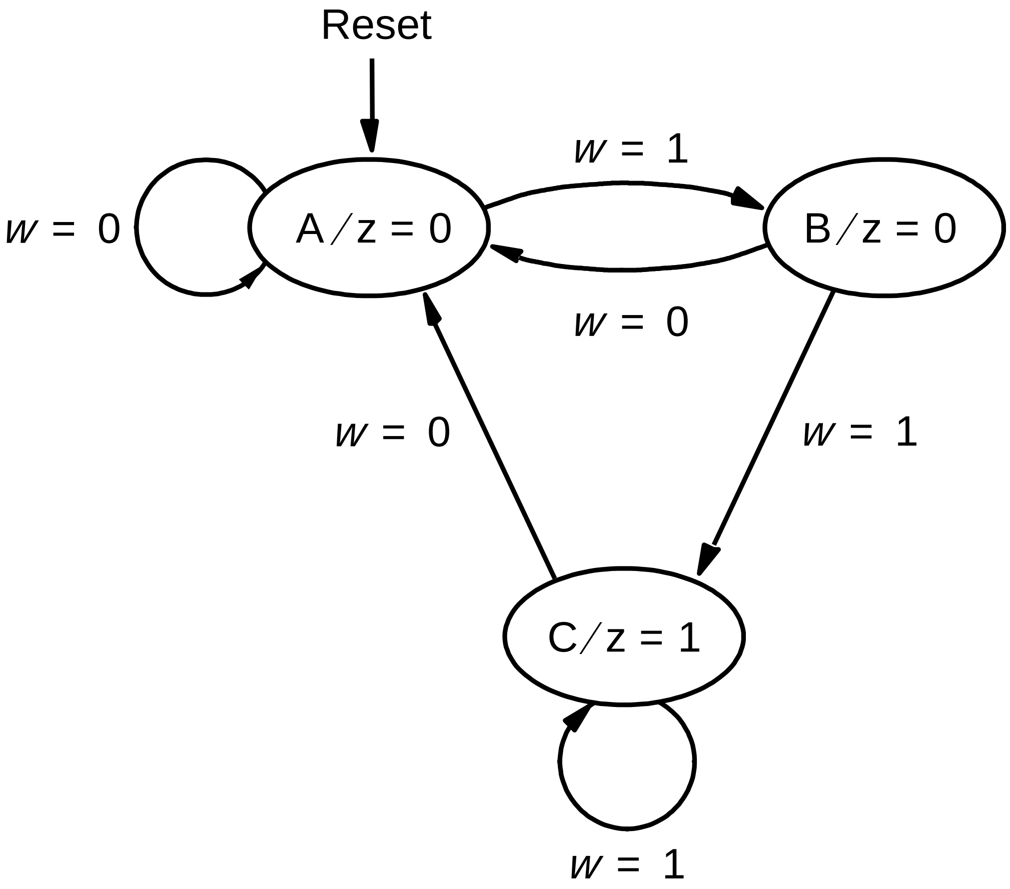


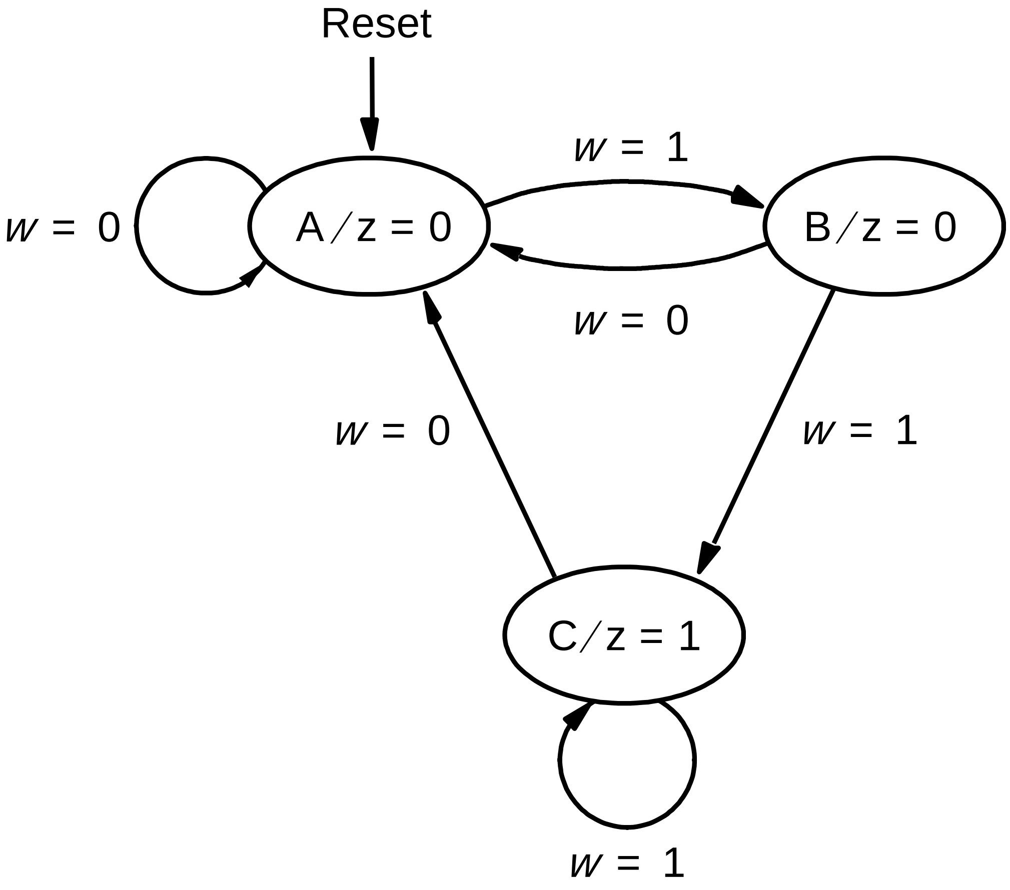
Figure 3. State diagram of a simple sequential circuit.

Remember, we have to design a circuit that satisfies the description we came up with earlier.

Next Step

- The next step in the design is to create a tabular listing of the action that is depicted in the state diagram.
- This is called a STATE TABLE

Here is the STATE diagram that we came up with



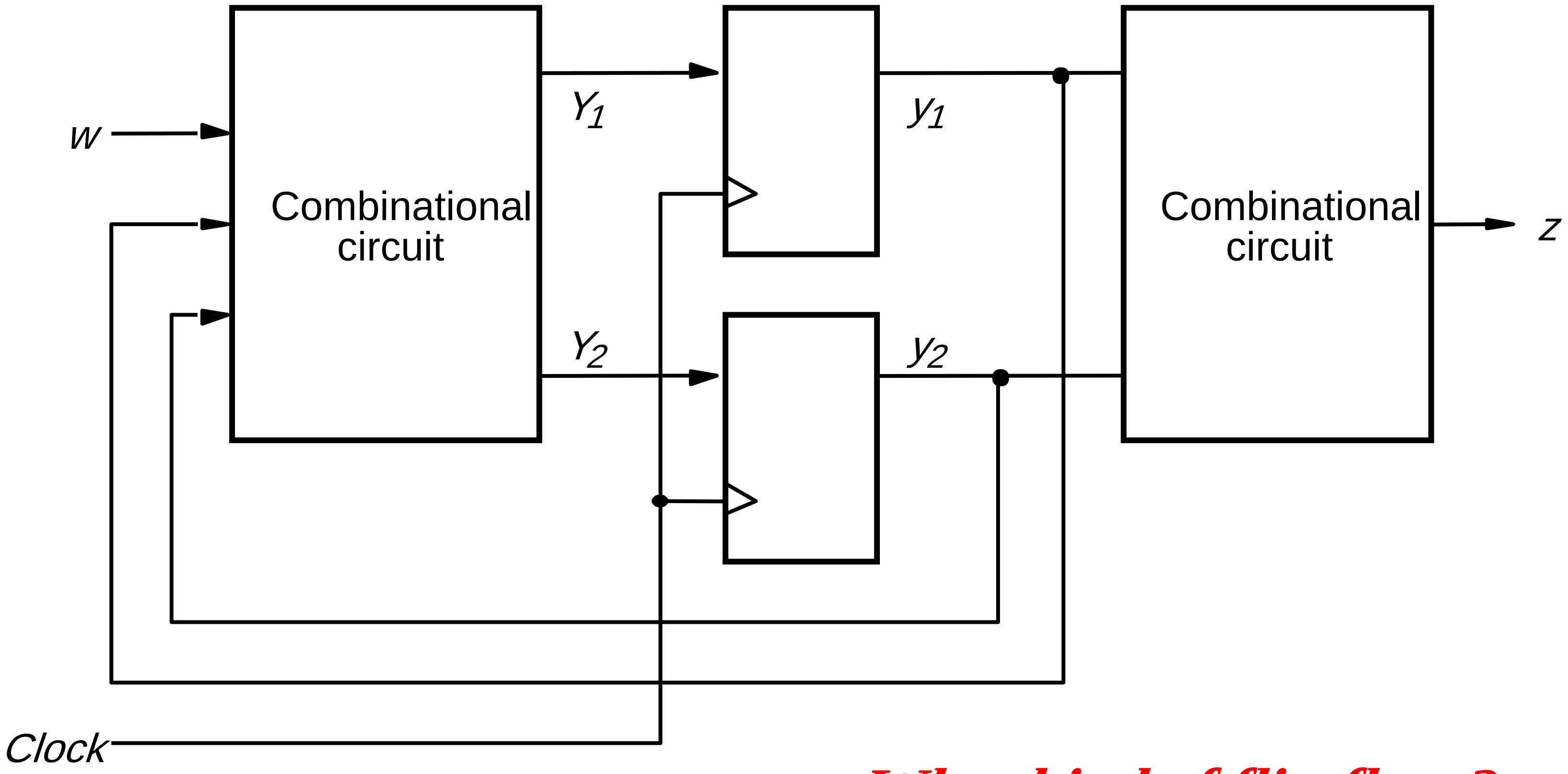
Present state	Next state		Output z
	$w = 0$	$w = 1$	
A	A	B	0
B	A	C	0
C	A	C	1

Figure 4. State table for the sequential circuit in Figure 3.

Number of Flip-Flops

- Since each Flip=flop can represent 2 states, since we have 3 states, we need 2 flip-flops.

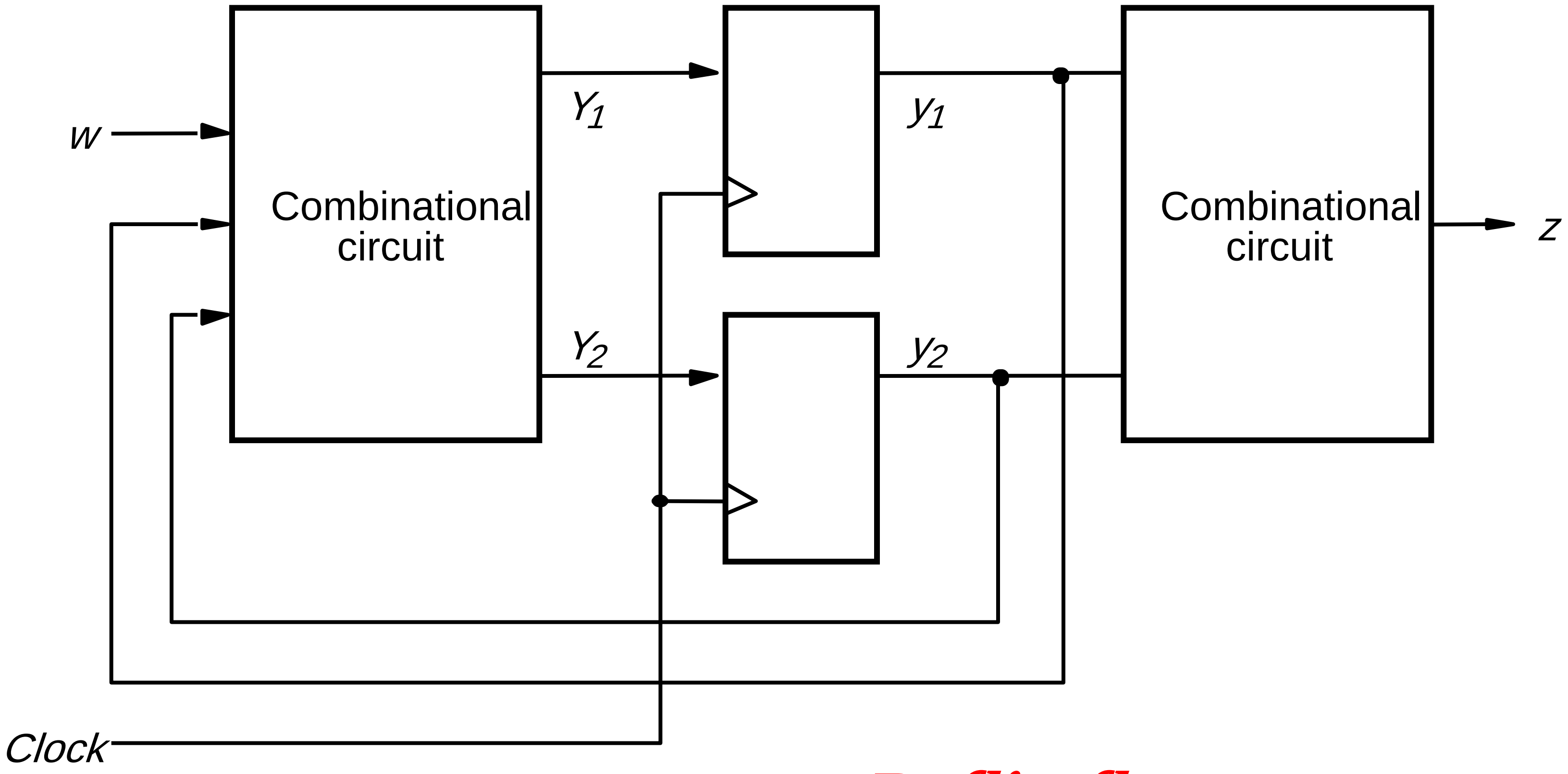
Mealy or Moore ?



What kind of flip-flops?

Figure 5. A general sequential circuit with input *w*, output *z*, and two state flip-flops.

Moore



D flip-flops

Figure 5. A general sequential circuit with input *w*, output *z*, and two state flip-flops.

Next Step

- Design the combinational logic circuit that will generate the

NEXT STATE VARIABLES

and the

OUTPUT

Since the output is solely determined by the output of the memory FlipFlops (present state information) it is a Moore Machine.

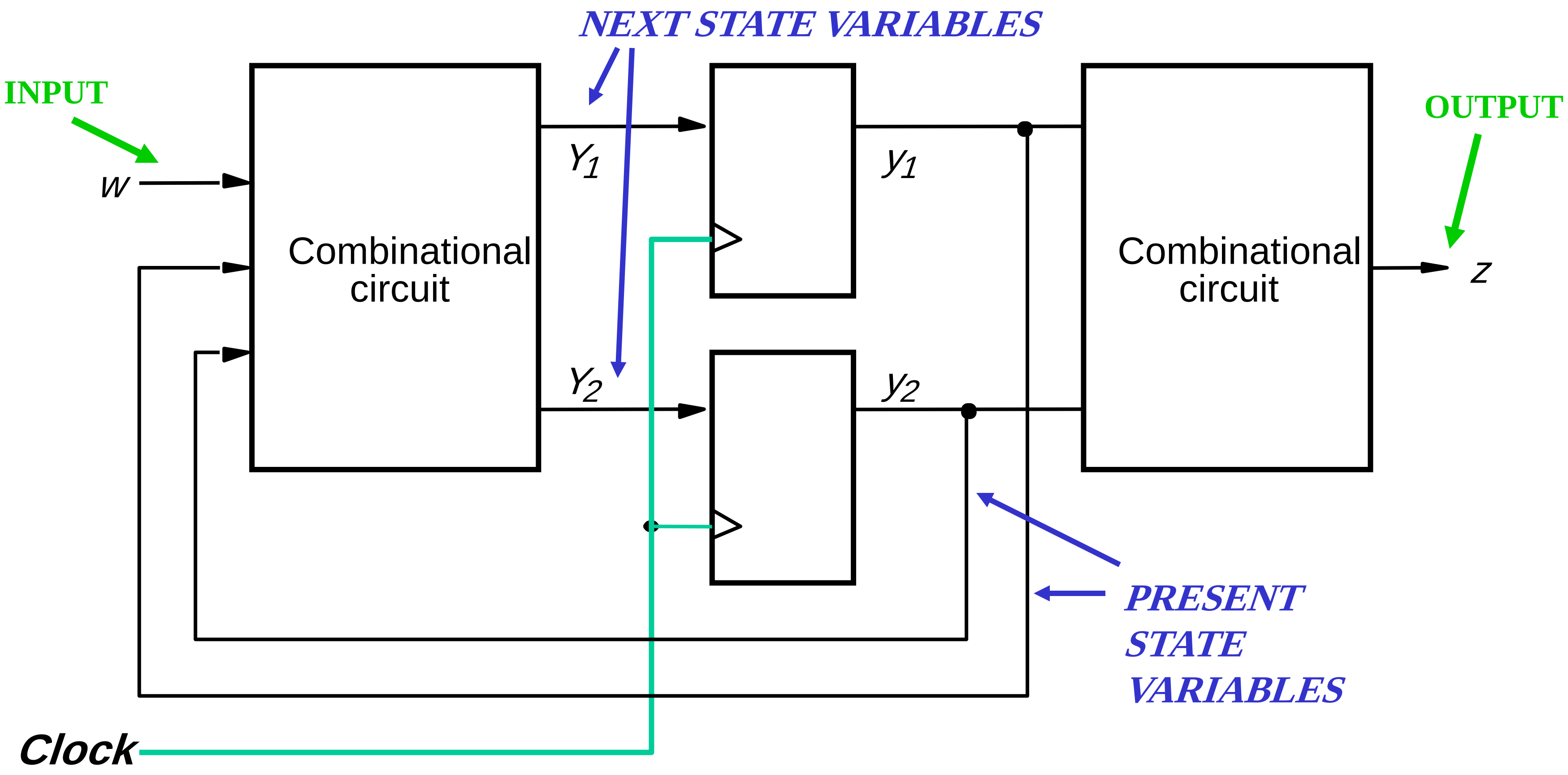


Figure 5. A general sequential circuit with input *w*, output *z*, and two state flip-flops.

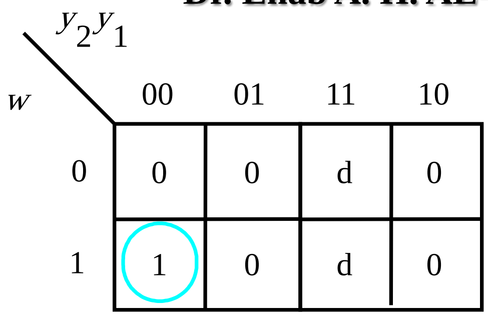
Here is where we assign the flip-flop outputs to the various states.

	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	10	1
	11	*dd*	*dd*	*d*

WHY?

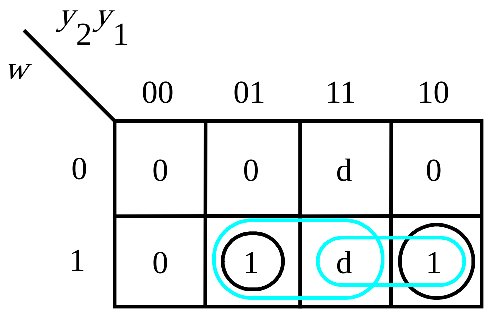
Note that this is just like the truth tables we developed for the combinational logic circuits.

Figure 6. State-assigned table for the sequential circuit in Figure 4.



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

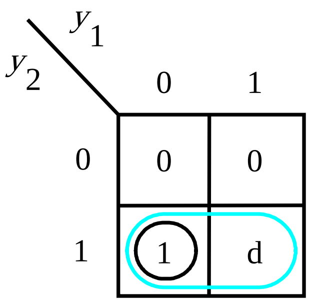
Ignoring don't cares	Using don't cares
$Y_1 = w\bar{y}_1\bar{y}_2$	$Y_1 = w\bar{y}_1\bar{y}_2$



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

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

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



y_2 \ y_1	0	1
0	0	0
1	1	d

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

$$z = y_2$$

Figure 7. Derivation of logic expressions for the sequential circuit in Figure 6.

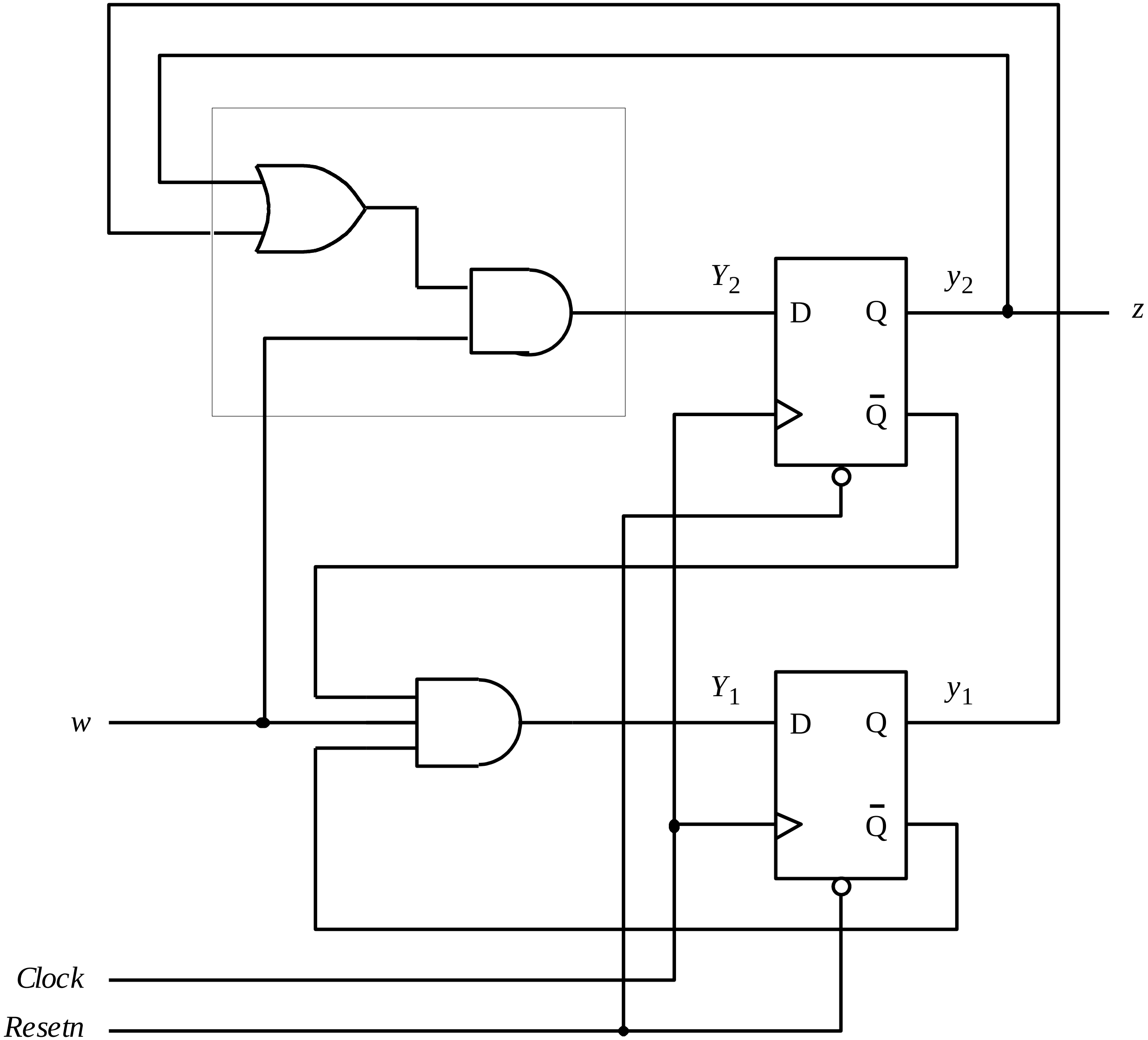


Figure 8. Final implementation of the sequential circuit derived in Figure 7.

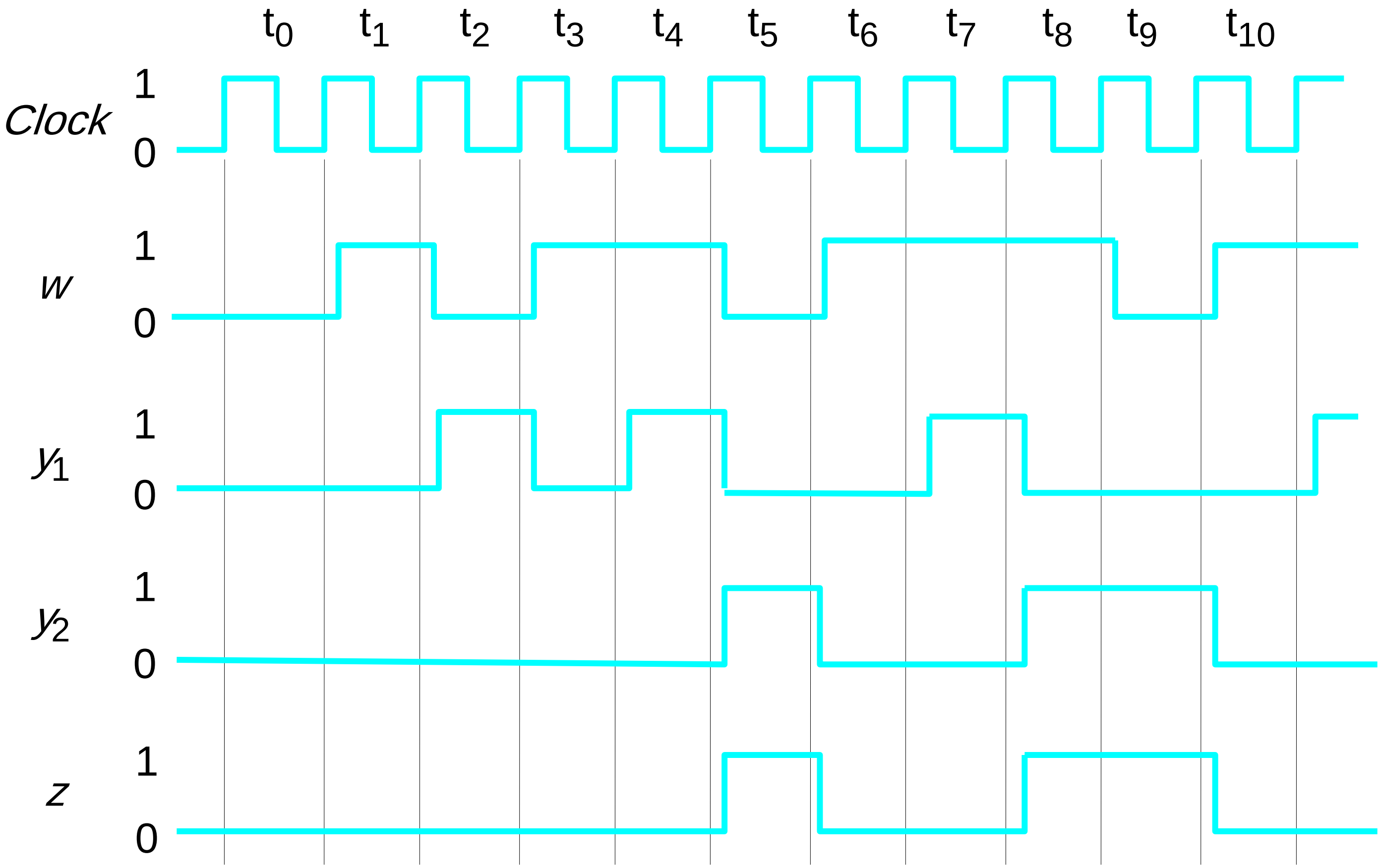


Figure 9. Timing diagram for the circuit in Figure 8.

Summary of Design Steps in the Design of a FSM

1. ***Obtain specification of desired behavior***
2. ***Select start state***
3. ***Determine number of states needed***
4. ***Use a state diagram to keep track of states***
5. ***Account for all I/O conditions***
6. ***Identify when machine moves from state to state***
7. ***Create state table from state diagram***
8. ***Minimize number of states***
9. ***Decide on number of state variables***
10. ***Perform state assignments***

Summary of Design Steps in the Design of a FSM (con’t)

11. ***Chose type of flip-flop***
12. ***Derive next state logic expressions (inputs to flip-flops)***
13. ***Derive logic expressions for outputs***
14. ***Implement circuits***