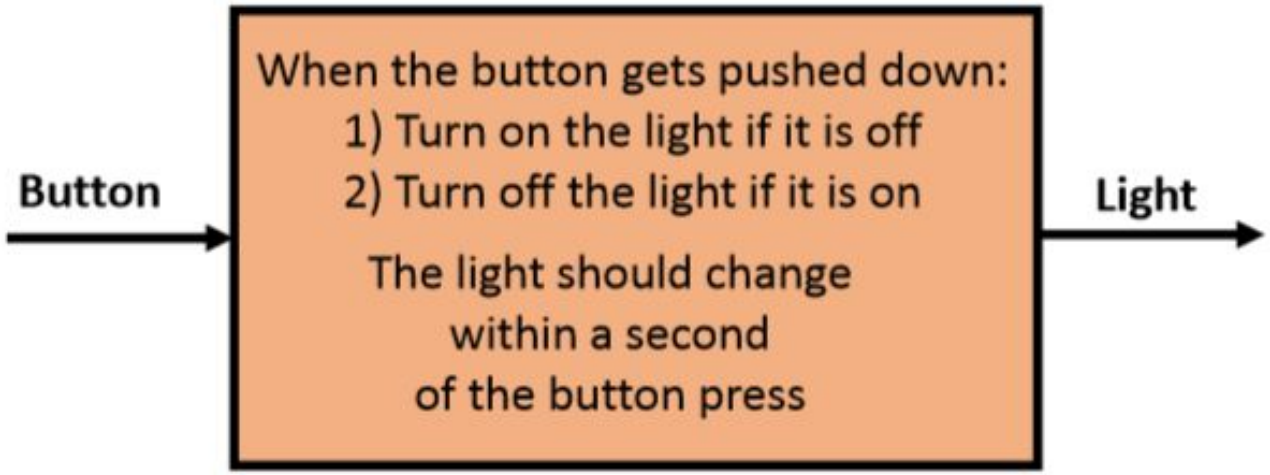
CSE460: VLSI Design

Lecture 8: Finite State Machines (Part 1)

Let’s change the problem!!

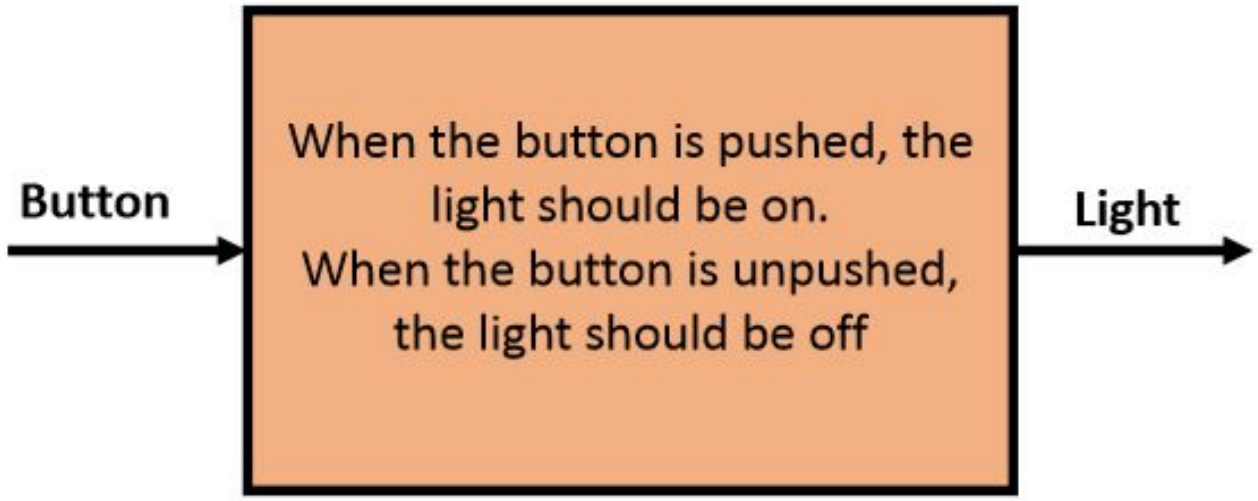
- What if you were given the following design specification:



What makes this circuit so different from those we’ve discussed before?

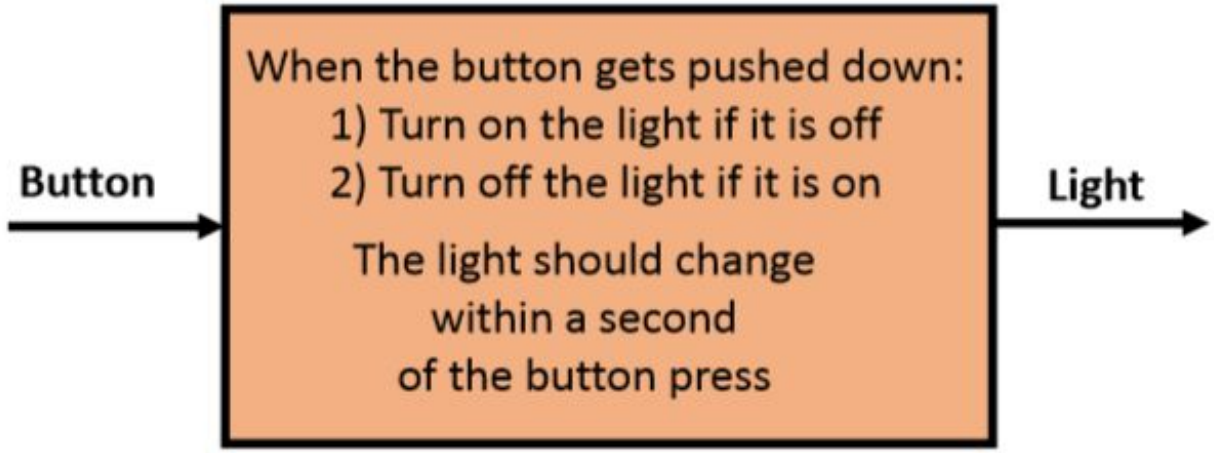
Something We Can Build (So Far)

- What if you were given the following design specification:



What makes this circuit so different from those we've discussed before?

Specifications:



1. "State" –i.e. **the circuit has memory** (become "state-ful").

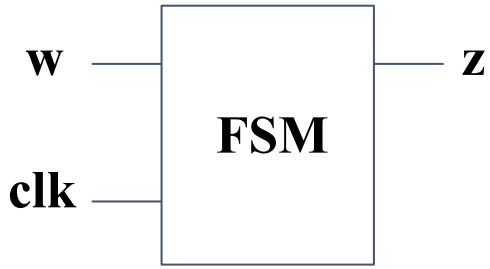
2. The output was changed by an input "event" (pushing a button) rather than an input "value" (output may depend on previous inputs).

Example 1

An automatically-controlled vehicle is designed to run at some predetermined speed. However, due to some operational conditions the speed may exceed the desirable limit, in which case the vehicle has to be slowed down. To determine when such action is needed, the speed is measured at regular intervals.

- Let a binary signal **w** indicate whether the speed exceeds the required limit, such that **w = 0 means that the speed is within acceptable range and w = 1 indicates excessive speed.**
- The desired control strategy is that if w = 1 during **two** or more consecutive measurements, **a control signal z must be 1** to cause the vehicle to slow down. Thus, z = 0 allows the current speed to be maintained, while z = 1 reduces the speed. Let a signal Clock define the required timing intervals, such that **the speed is measured once during each clock cycle.**

Example 1



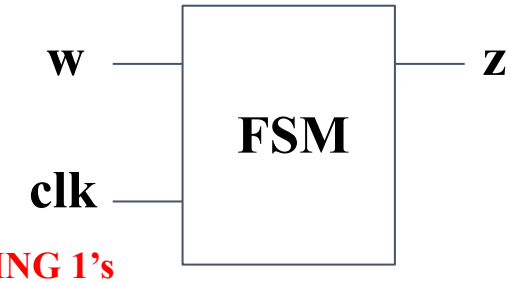
From previous example 1, we wish to design the circuit that meets the following specification:

1. The circuit has one input **w**, and one output **z**.
2. All changes in the circuit occur on the **positive edge** of a clock signal.
3. The output **z** is equal to 1 if during two immediately preceding clock cycle the input w was equal to **1**. Otherwise the value of 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 3 : Sequences of input and output signals.

Example 1: A Simple Input Pattern (‘11’ Overlapping Sequence)



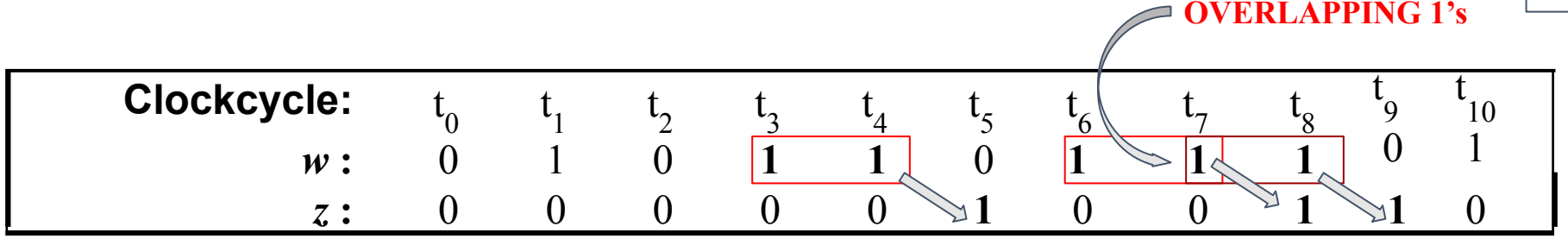
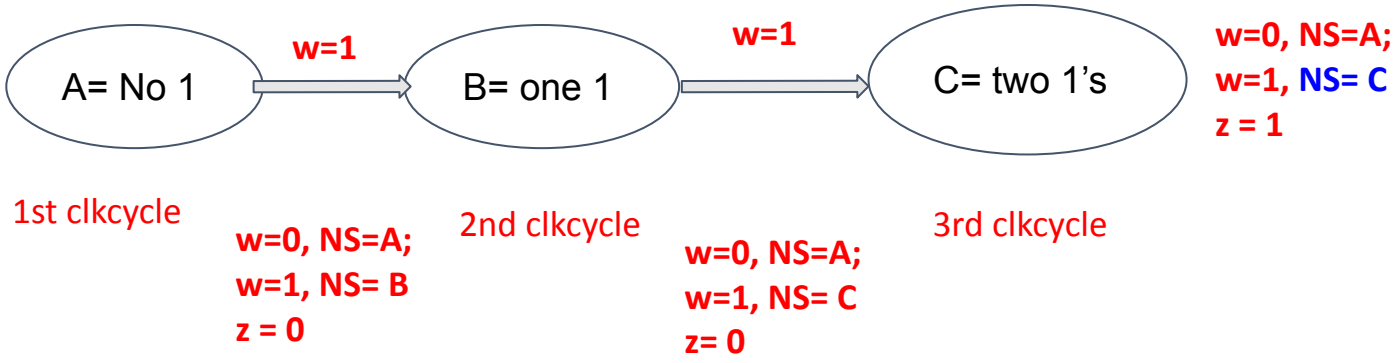


Figure 4 : Sequences of input and output signals.

The circuit detects two consecutive ‘1’. So, it must keep in memory how many ‘1’ it received in its input. So, a state diagram is introduced.

- **A** state- no ‘1’ in input,
- **B** state- one ‘1’ in input and
- **C** state- two ‘1’ in input.



Example 1: A Simple Input Pattern (‘11’ Overlapping Sequence) Detection Circuit - Moore Type

State Diagram

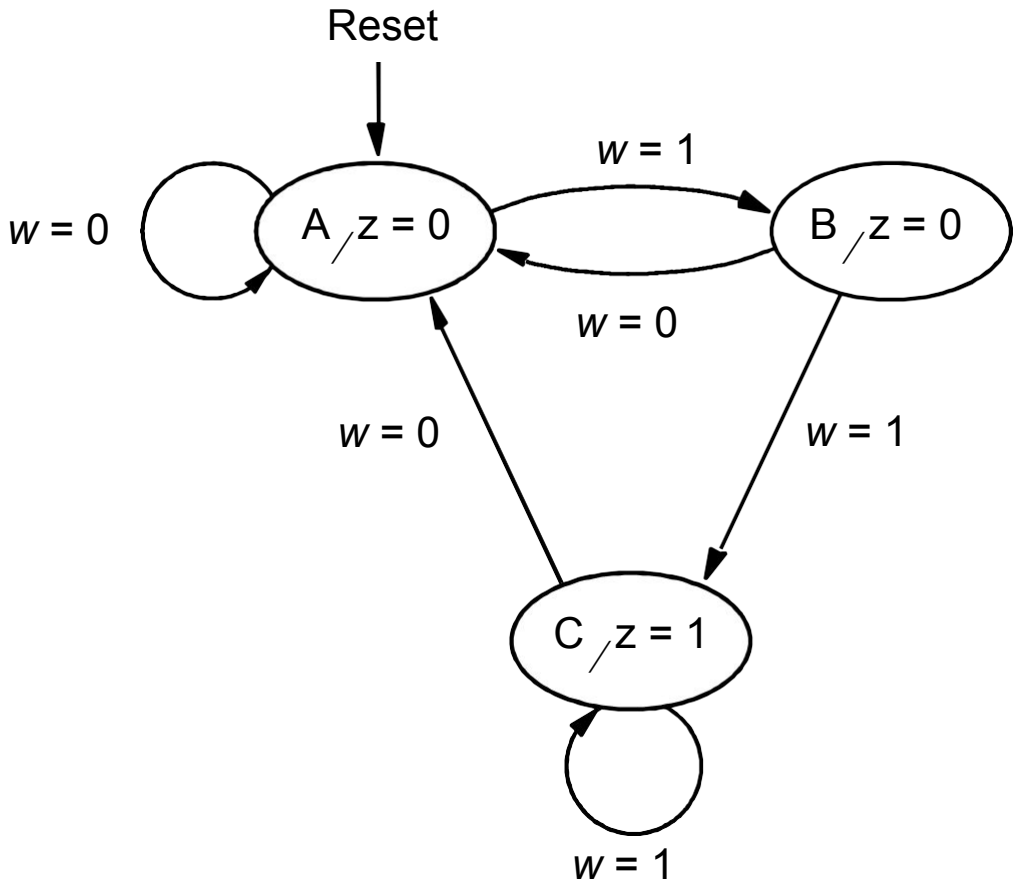


Figure 5: State diagram of a simple sequential circuit.

Example 1: A Simple Input Pattern ('11' Overlapping Sequence) Detection Circuit [State Table]

State Table

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

Note:
n bits can represent 2^n states.

Figure 6: State table for the sequential circuit in Figure 5.

Example 1: A Simple Input Pattern ('11' Overlapping Sequence) Detection Circuit [State Assigned Table]

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

We need to find expressions (K-map) for

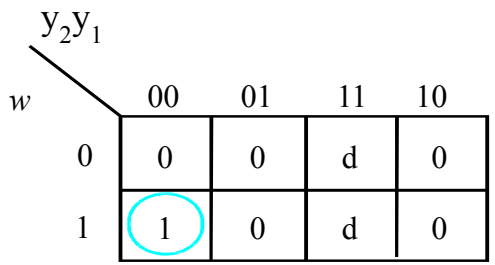
- next state (Y_2 ,Y_1)
- output (z).

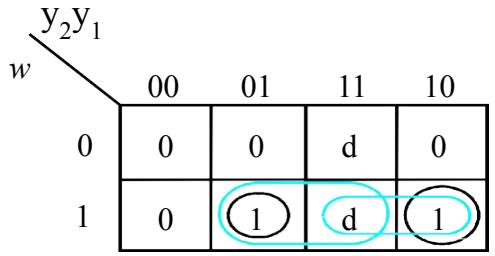
Note that,

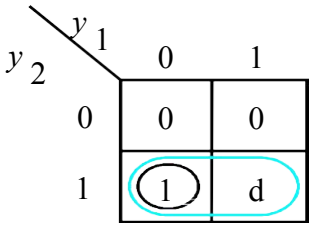
- ➢ $Y_1 = f(w,y_1,y_2)$
- ➢ $Y_2 = f(w,y_1,y_2)$
- ➢ $z = f(y_1,y_2)$

Figure 7: State-assigned table for the sequential circuit in Figure 7.

Example 1: A Simple Input Pattern ('11' Overlapping Sequence) Detection Circuit [K-Map]







Ignoring don't cares

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

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

$$z = y_1 \bar{y}_2$$

Using don't cares

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

$$Y_2 = w\bar{y}_1 + w y_2$$
$$= w(y_1 + y_2)$$

$$z = y_2$$

Figure 8: Derivation of logic expressions for the sequential circuit in Figure 7.

Example 1: A Simple Input Pattern ('11' Overlapping Sequence) Detection Circuit [Circuit]

Note that,

- # flipflops = # bits to represent current state (#y)
- Outputs of flipflops correspond to current state bits (y_1, y_2)
- Inputs of flipflops correspond to next state bits (Y_1, Y_2)

In this problem, we need **two flipflops** as we have used **two bits** (y_1, y_2) to assign all the present states (A, B, C).

Example 1: A Simple Input Pattern (‘11’ Overlapping Sequence) Detection Circuit [Circuit]

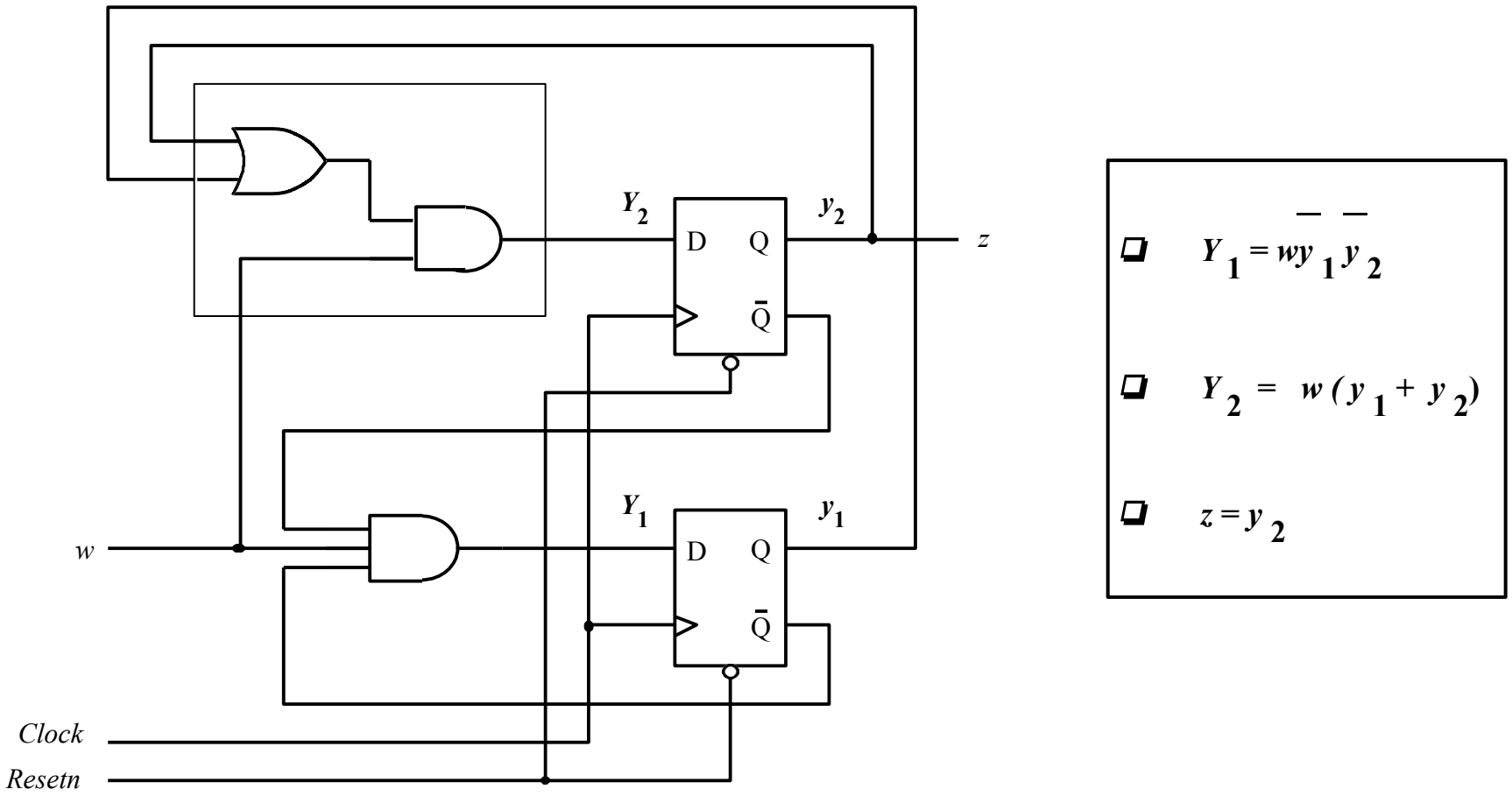


Figure 9. Final implementation of the sequential circuit derived in Figure 8.

Summary

Steps->

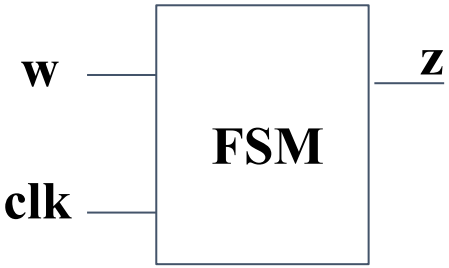
- State diagram
- State table
- State assigned table
- K-map (next states & outputs)
- Circuit

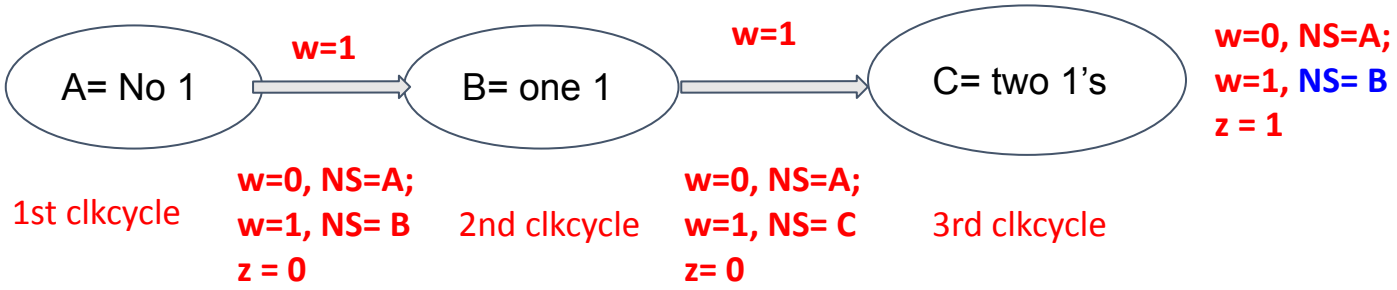
Example 1(cont.): A Simple Input Pattern ('11' Non-Overlapping Sequence) (Detection Circuit - Moore Type)

NON-OVERLAPPING 1's

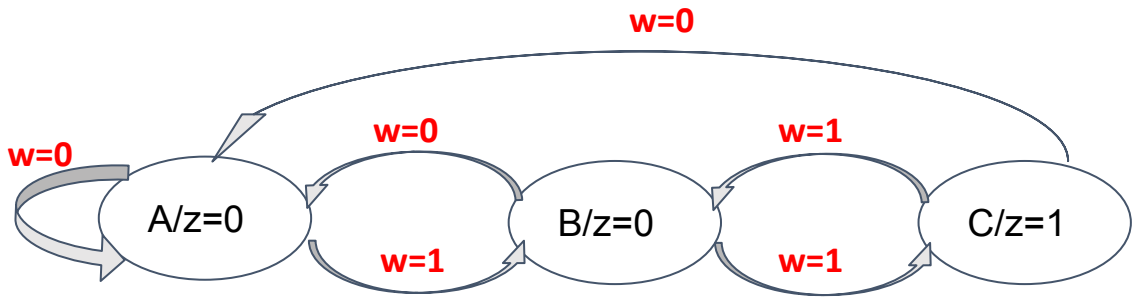
Clockcycle:	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
w :	0	1	0	**1**	**1**	0	**1**	**1**	**1**	0	1
z :	0	0	0	0	0	**1**	0	0	**1**	**0**	0

Figure 4 : Sequences of input and output signals.





STATE DIAGRAM



General form of a sequential circuit

- W represents input/s
- Z represents output/s
- Flip-flops are used to save the “present” state of the circuit, Q
- Also called Finite State Machine (FSM) or simply, machine
- 2 types: depending on the connection drawn in pink on the circuit

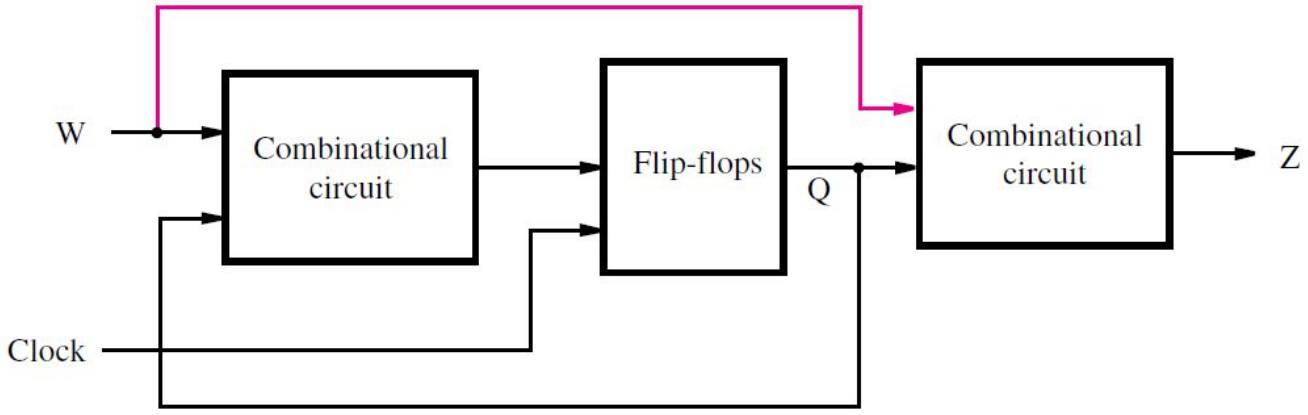


Figure: General form of a sequential circuit

FSM types

- Moore type FSM

  Output (Z) depends only on the “present” state (Q) of the circuit

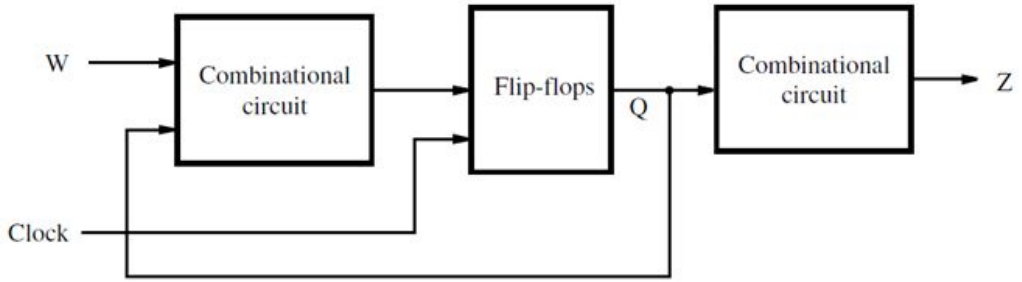


Figure: Moore type FSM

- Mealy type FSM

  Output (Z) depends on both the primary inputs (W) and the “present” state (Q) of the circuit

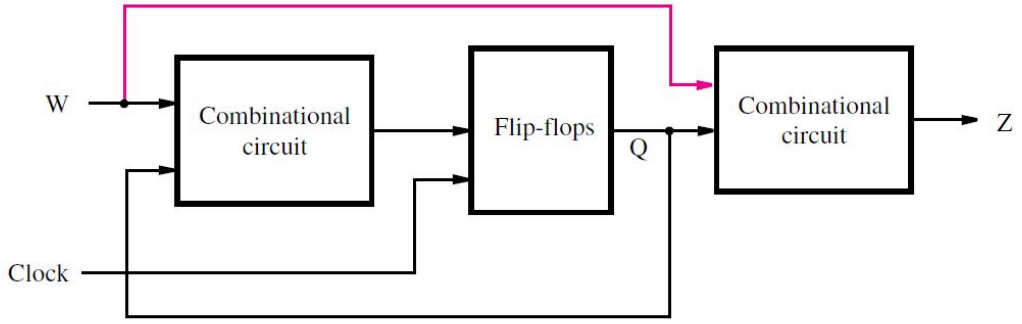


Figure: Mealy type FSM

Moore vs. Mealy type machines

Moore Machine	Mealy Machine
▪ **Output** depends only on the ***present state***.	▪ **Output** depends both on the ***present state*** and the *present* ***input***.
▪ Generally, it has *more states* than Mealy Machine.	▪ Generally, it has *fewer states* than Moore Machine.
▪ The output is a function of the current state and changes *at clock edges*	▪ The output is a function of the transitions and can change any instant
▪ In Moore machines, more logic is required to decode the outputs resulting in more circuit delays. They generally react one clock cycle later.	▪ Mealy machines react faster to inputs. They generally react in the same clock cycle.