CSE 460: VLSI Design

Lecture 9: Finite State Machines (part 2)

Example 2: A Simple Input Pattern (‘11’ Overlapping Sequence) Detection Circuit (Mealy Type)

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 6.22 Sequences of input and output signals.

Steps->

- ➢ State diagram
- ➢ State table
- ➢ State assigned table
- ➢ K-map
- ➢ Circuit

Example 2: A Simple Input Pattern (‘11’ Overlapping Sequence) Detection Circuit (Mealy Type)

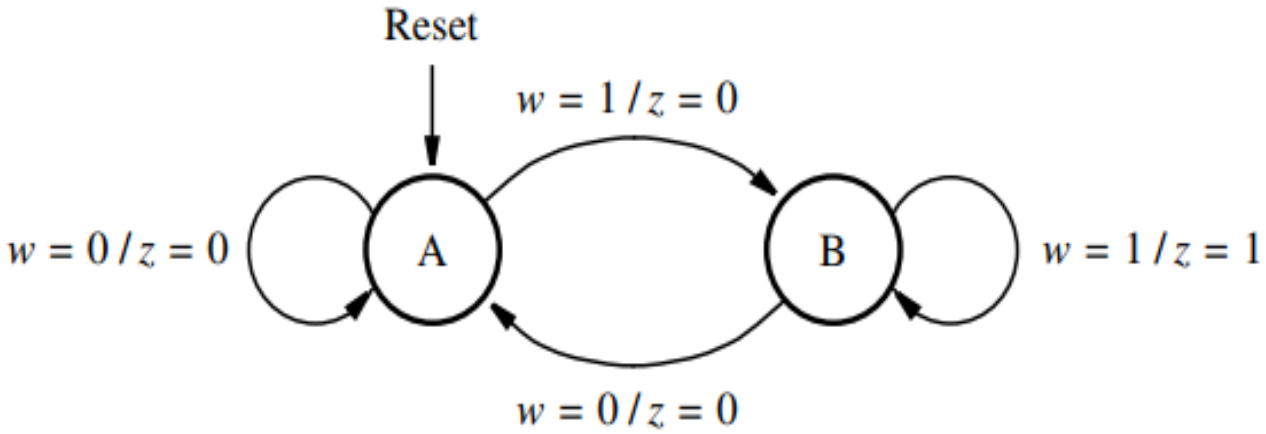


Figure 6.23 State diagram of an FSM that realizes the task in Figure 6.22.

Steps->

- **State diagram**
- State table
- State assigned table
- K-map
- Circuit

Example 2: A Simple Input Pattern (‘11’ Overlapping Sequence) Detection Circuit (Mealy Type)

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 6.24 State table for the FSM in Figure 6.23.

Steps->

- ➢ State diagram
- ➢ **State table**
- ➢ **State assigned table**
- ➢ K-map
- ➢ Circuit

	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

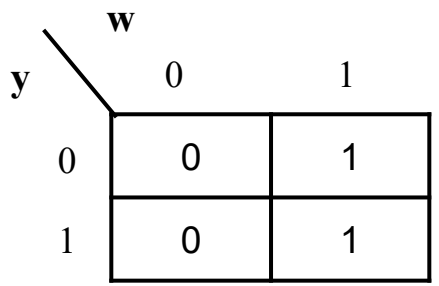
Figure 6.25 State-assigned table for the FSM in Figure 6.24.

Note that,

- ➢ Y = f(w,y)
- ➢ z = f(w,y)

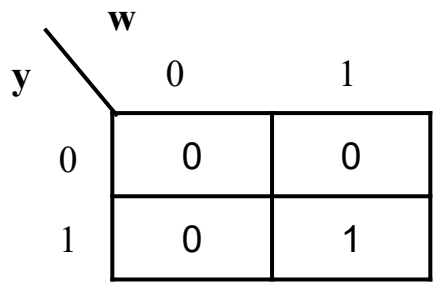
Example 2: A Simple Input Pattern (‘11’ Overlapping Sequence) Detection Circuit (Mealy Type)

Y =>



Y = w

z =>



z = wy

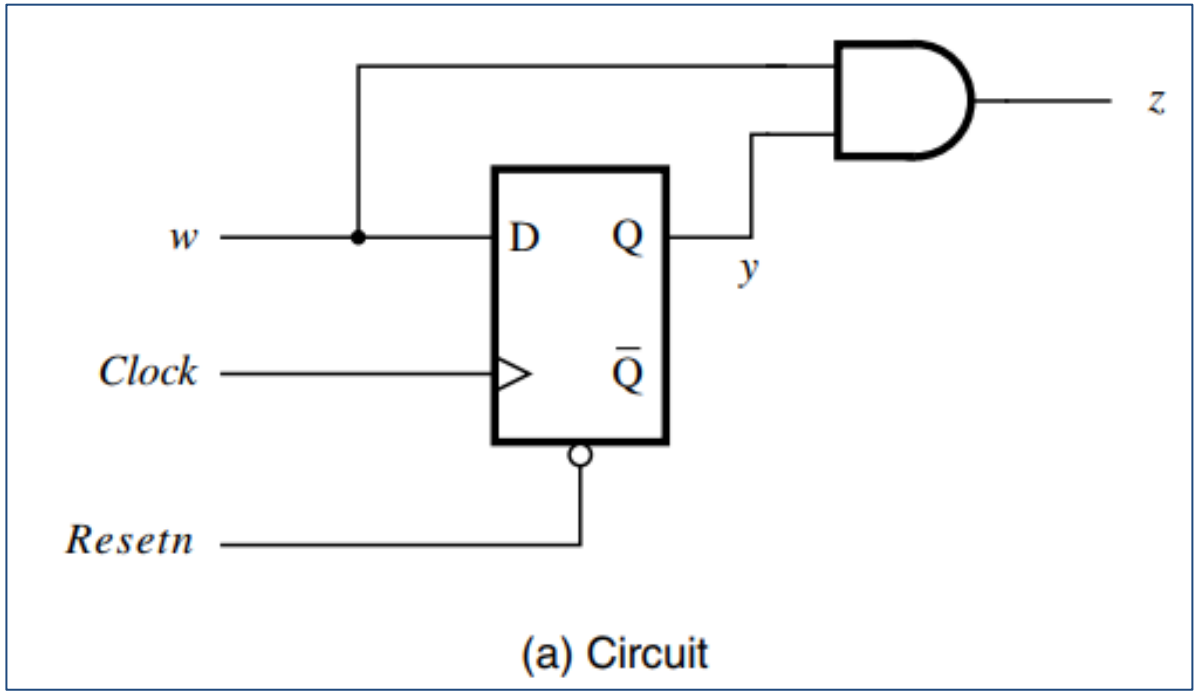
Steps->

- ➢ State diagram
- ➢ State table
- ➢ State assigned table
- ➢ **K-map** (Y and z)
- ➢ Circuit

Note that,

- ➢ Y = f(w,y)
- ➢ z = f(w,y)

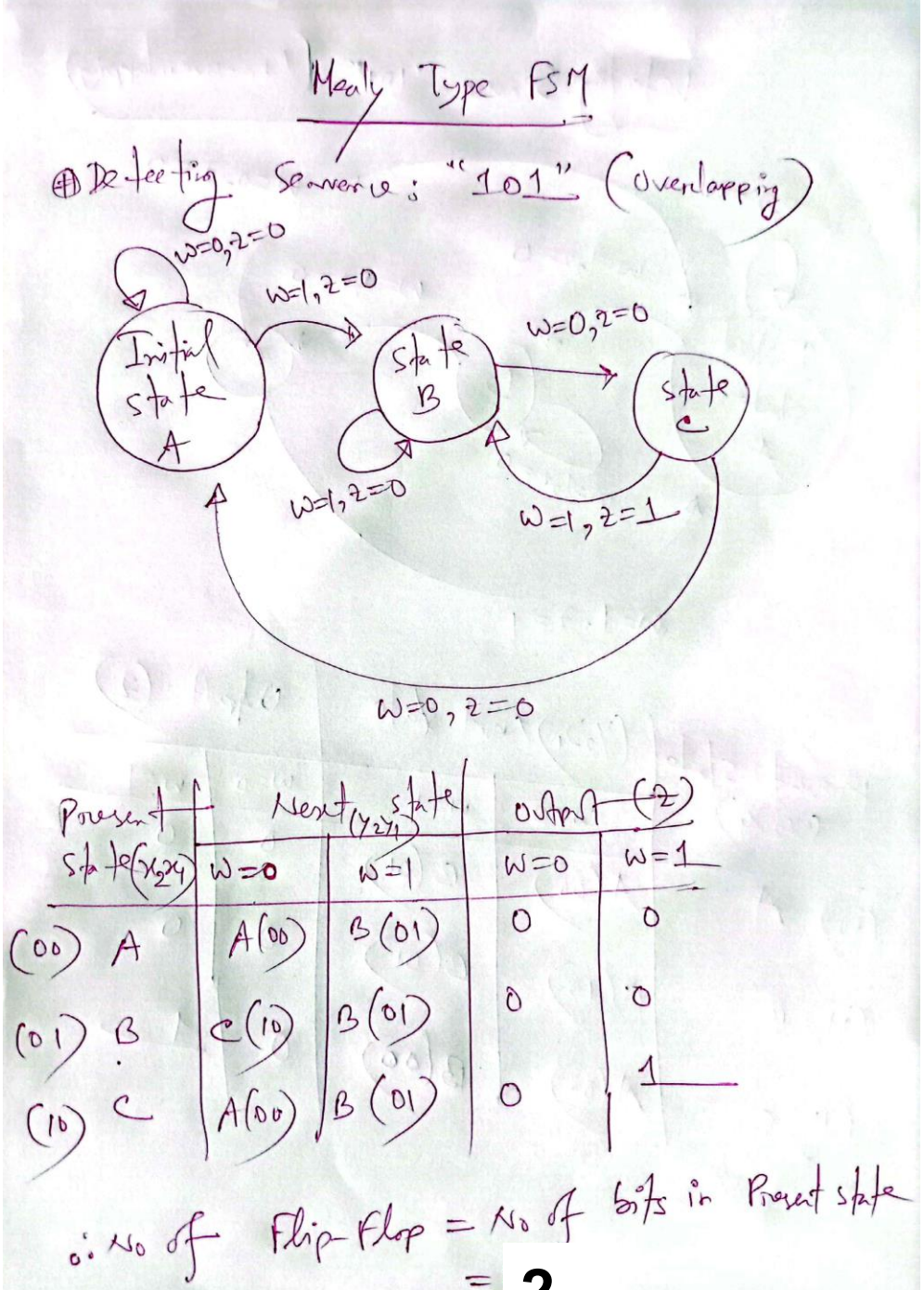
Example 2: A Simple Input Pattern (‘11’ Overlapping Sequence) Detection Circuit (Mealy Type)



(a) Circuit

Steps->

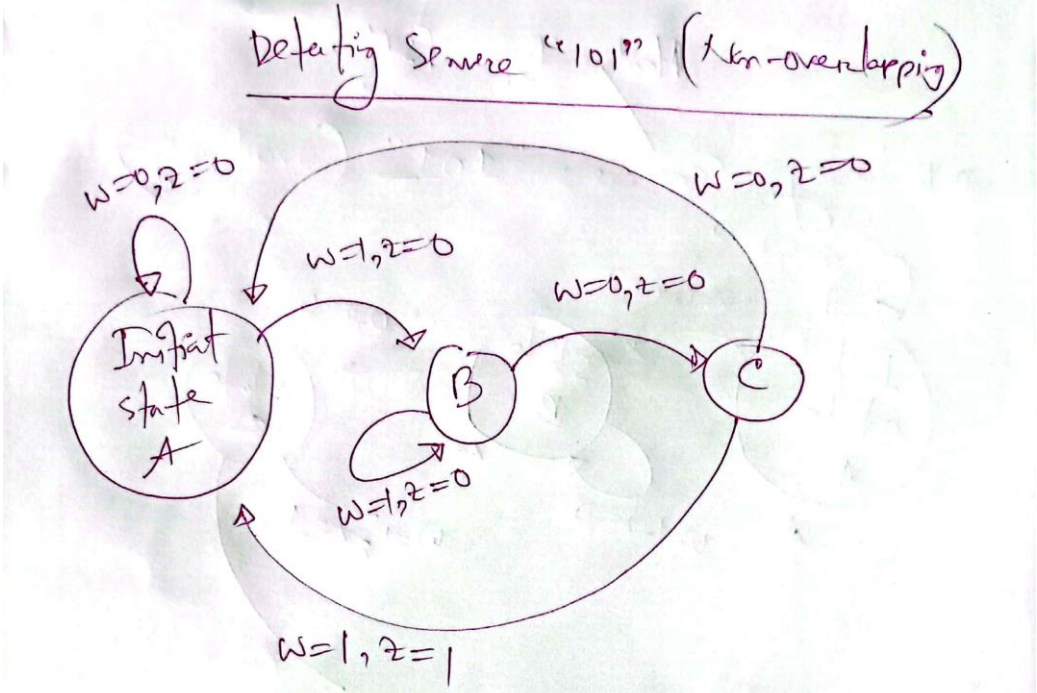
- ➢ State diagram
- ➢ State table
- ➢ State assigned table
- ➢ K-map
- ➢ **Circuit**



Example 3: A Simple Input Pattern ('101' Overlapping Sequence) Detection Circuit (Mealy Type)

Important:

In Mealy Type FSM, you do not need to declare a separate state for your final output (for this example: no separate state for '101' is required). After getting your output z = 1, you will start your 2nd loop (Start finding z=1 for the 2nd time).



Present state (x_2x_1)	(Y_2Y_1) Next state		Output (z)	
	w=0	w=1	w=0	w=1
(00) A	A(00)	B(01)	0	0
(01) B	C(10)	B(01)	0	0
(10) C	A(00)	A (00)	0	1

Example 4: A Simple Input Pattern ('101' Non-Overlapping Sequence) Detection Circuit (Mealy Type)

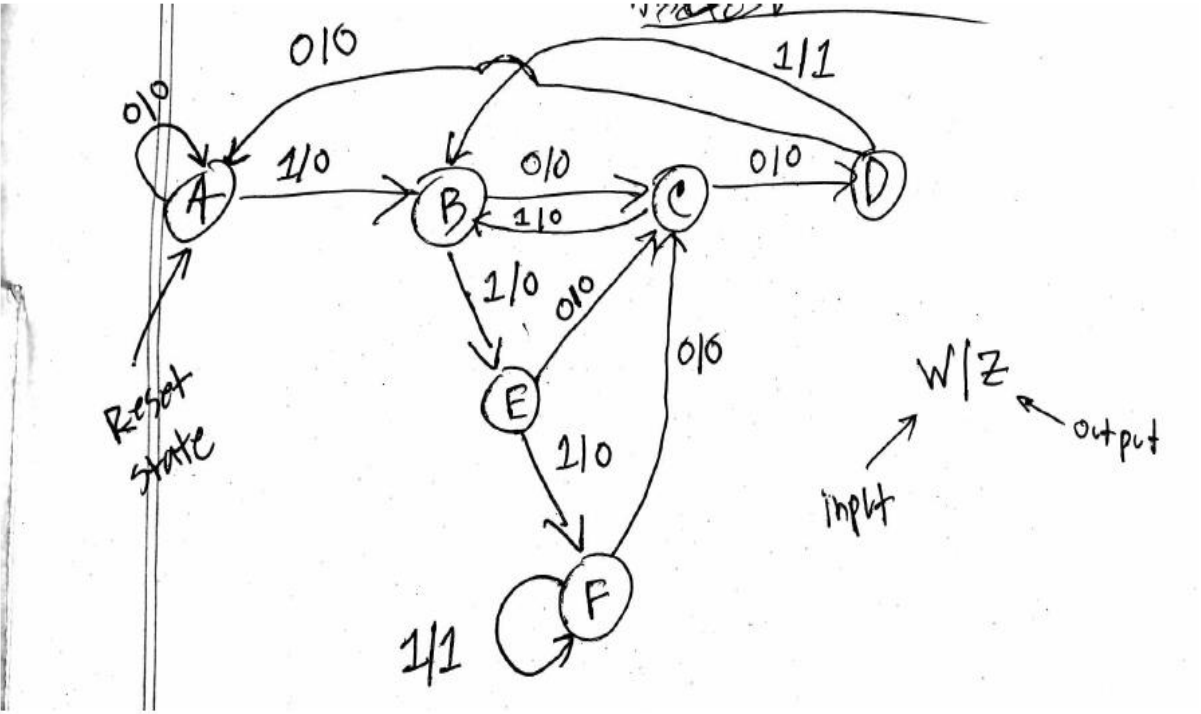
Example 5:

1.	1. A finite state machine that has to generate z = 1 when the previous four values of input w were 1001 or 1111; otherwise, z = 0. Overlapping input patters are allowed. The machine also has a negative reset. An example of the desired behavior is w : 0101 1110 0110 0111 11 z : 0000 0010 0100 0100 11	
(a)	**Design** the state diagram. Mark your reset states and transitions clearly. Is this machine Mealy type or Moore type?	**3**
(b)	**Derive** the assigned table and write the Verilog code to implement the following FSM.	**4**
(c)	**Validate** your answer with the appropriate input and output waveforms on a timing diagram. Include all possible combinations of the input (w).	**3**

Solution:

The Machine is Mealy type
My Reset state is A State
W = Input Sequence
Z = Output

Format is W/Z



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

State Assigned Table:

A = 000
B = 001
C = 010
D = 011
E = 100
F = 101

Present state	Next state		Out put	
	w=0	w=1	w=0	w=1
000	000	001	0	0
001	010	100	0	0
010	011	001	0	0
011	000	001	0	1
100	010	101	0	0
101	010	101	0	1

Example 6:

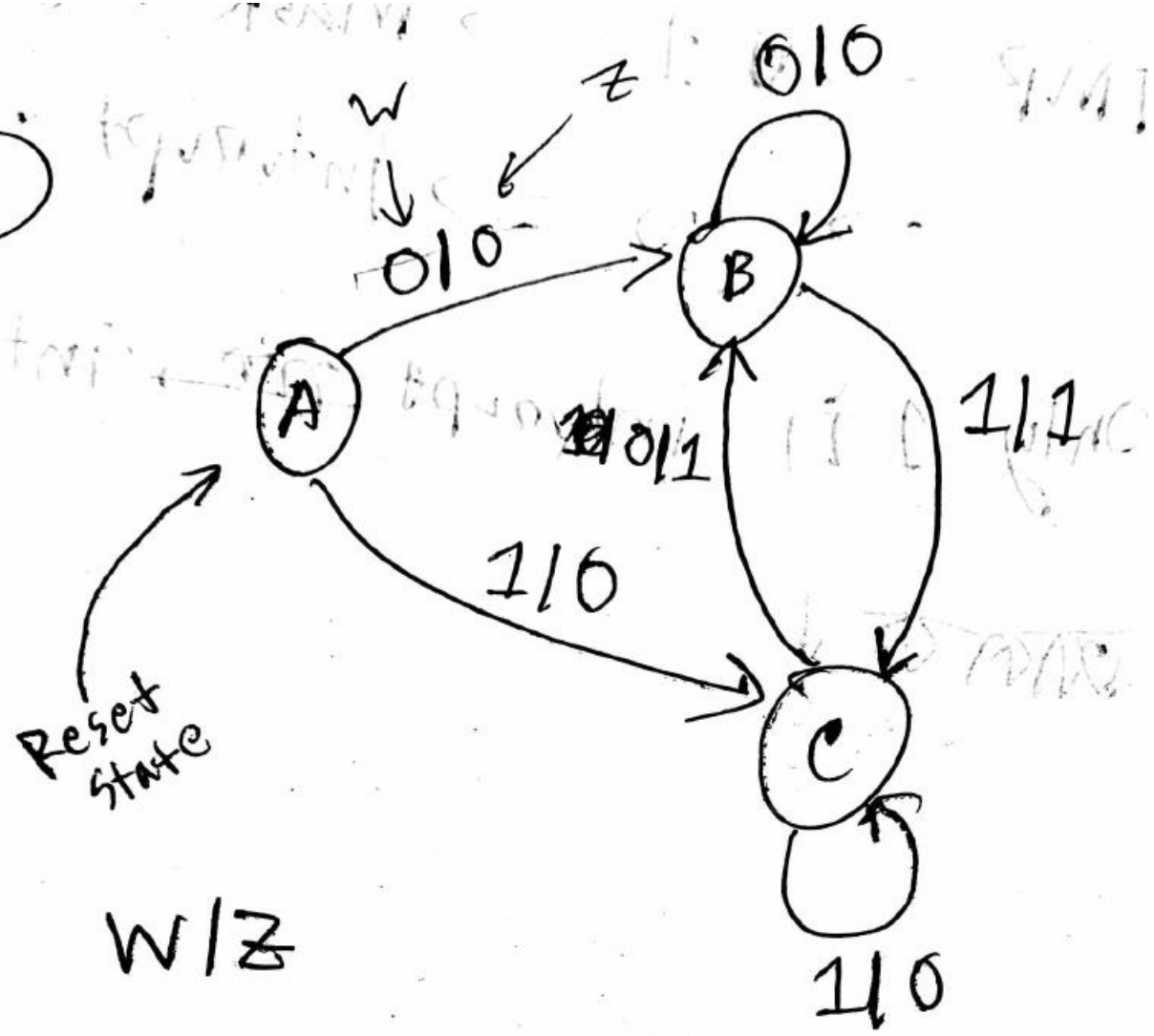
The input-output behavior of a sequence detector FSM is given below.

CLK	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}	t_{16}	t_{17}	t_{18}
w	**1**	**0**	**0**	**1**	1	**1**	**0**	0	0	0	0	0	0	**0**	**1**	**0**	0	0
z	0	**1**	0	**1**	0	0	**1**	0	0	0	0	0	0	0	**1**	**1**	0	0

(a)	**Identify** the sequence and the type of FSM for the above input-output combinations. [Hint: The above table contains two sequences]	**1+1**
(b)	**Design** the state diagram, the state table, and the ***gray-encoded*** state assigned table. **[Overlapping allowed].**	**3+2+2**
(c)	**Determine** the logic expressions of the next state and output variables using k-maps.	**6**

** Alternating 0 and 1
** Two consecutive inputs are different

Solution:



Present State	Next State		Output Z	
	w=0	w=1	w=0	w=1
A	B	C	0	0
B	B	C	0	1
C	B	C	1	0

Present state	Next State		output z	
	w=0	w=1	w=0	w=1
0 0	0 1	11	0	0
0 1	0 1	11	0	1
11	0 1	11	1	0

Gray Encoding

Example: 07: Double input FSM

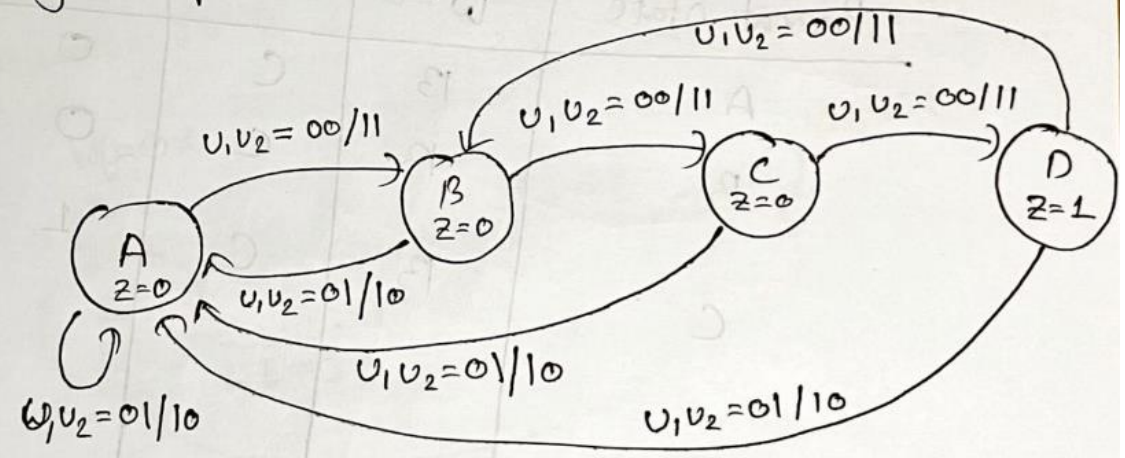
Consider a sequential circuit with two inputs (***u1*** & ***u2***) and a single output, ***z***.
CLEO is a prestigious conference where researchers from all over the world submit their research articles. These articles are handled by an **Editor** and a **Reviewer**. They review each article *independently,* and either approve a submitted article (denoted by 1) or reject it (denoted by 0). The decisions of the **Editor** are given by a bit stream, ***u1***, whereas the decisions of the **Reviewer** are given by ***u2***. **If for three consecutive articles, both of their decisions are the same (i.e. *u1* = *u2* for three consecutive clock cycles), then they receive a consistency reward (*z* = 1) from the conference committee *after* a period.**
Here, the decisions of the Editor & the Reviewer for a number of submitted articles & their corresponding reward instances are given below:

u1	0	1	1	0	1	1	1	0	0	0	1
u2	1	1	1	0	0	0	1	0	0	1	1
z	0	0	0	0	1	0	0	0	0	1	0

(a)	What type of FSM would you need for the given problem?	**1**
(b)	**Design** the state diagram for the corresponding FSM.	**4**
(c)	**Derive** the state-assigned table from your state diagram in (a).	**4**
(d)	**Evaluate** the next state & output logic expressions to implement the FSM & **calculate** the number of flip-flops required for the circuit implementation. (You don't need to draw the circuit implementation.)	**5 + 1**

a) We need Moore type FSM for this given problem.

b)



c)

Present state Y_2Y_1	Next state $u_1u_2=00$ Y_2Y_1	$u_1u_2=01$ Y_2Y_1	$u_1u_2=10$ Y_2Y_1	$u_1u_2=11$ Y_2Y_1	Output, z
A 00	(B)01	(A)00	(A)00	(B)01	0
B 01	(C)11	(A)00	(A)00	(C)11	0
C 11	(D)10	(A)00	(A)00	(D)10	0
D 10	(B)01	(A)00	(A)00	(B)01	1

Example: 08: Double input FSM

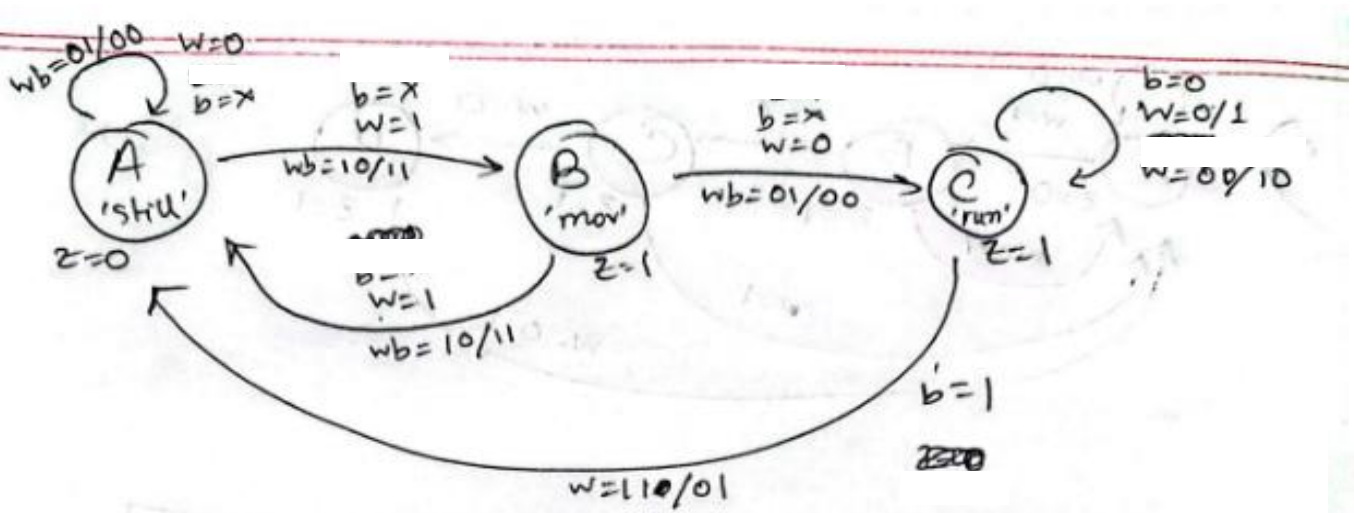
A car was in a stand still condition. To *start* the car, a start/stop button, ***w*** should be pressed. An output, $z = 1$ confirms that the car is *moving (accelerating) or running (optimum constant speed)*.

- If $w = 0$, the car should remain *still* initially. For, $w = 1$, the car should start *moving* (if it was *still*) or stop (if it has just started *moving*, but not *running* yet).
- Once moving, the car should reach the optimal speed within two consecutive clock cycles without pressing the start/stop button, ***w***.
- After two cycles, it will automatically gain the optimal speed to keep *running*. The car should maintain its speed independent of the value of ***w*** while *running*.
- The break, ***b*** can only be pressed for an emergency stop while *running*. If the car is *still* or *moving*, break, ***b*** *can be ignored.*

[Hint: The system has two inputs: ***w, b*** and one output: ***z***]

(a)	**Design** the state diagram for the given FSM. [The FSM should be Moore type.]	**5**
(b)	**Derive** a state table and the state-assigned table from **(a)** using the *gray*-encoding technique.	**4**
(c)	**Determine** the logic expressions of the next state and output variables using k-maps.	**6**

Example: 08: Double input FSM



b = X means, b is in a don't care logic

Present Y_2Y_1	Next State M_2M_1 w=00	w=01	w=11	w=10	Output z
A=00	00	00	01	01	0
B=01	11	11	00	00	1
C=11	11	00	00	11	1
D=10	dd	dd	dd	dd	dd

$Y = M_1$

W_2W_1 \ Y_2Y_1	00	01	11	10
00	0	1	1	dd
01	0	1	0	dd
11	1	0	0	dd
10	1	0	1	dd

$M_1 = Y_2\overline{W_1} + \overline{W_2}\,\overline{Y_2}Y_1 + W_2\overline{Y_2}\,\overline{Y_1}$

$Y = M_2$

W_2W_1 \ Y_2Y_1	00	01	11	10
00	0	1	1	dd
01	0	1	0	dd
11	0	0	0	dd
10	0	0	1	dd

$M_2 = Y_2\overline{W_1} + \overline{W_2}\,\overline{Y_2}Y_1$

$Y = z$

Y_1 \ Y_2	0	1
0	0	dd
1	1	1

$z = Y_1$

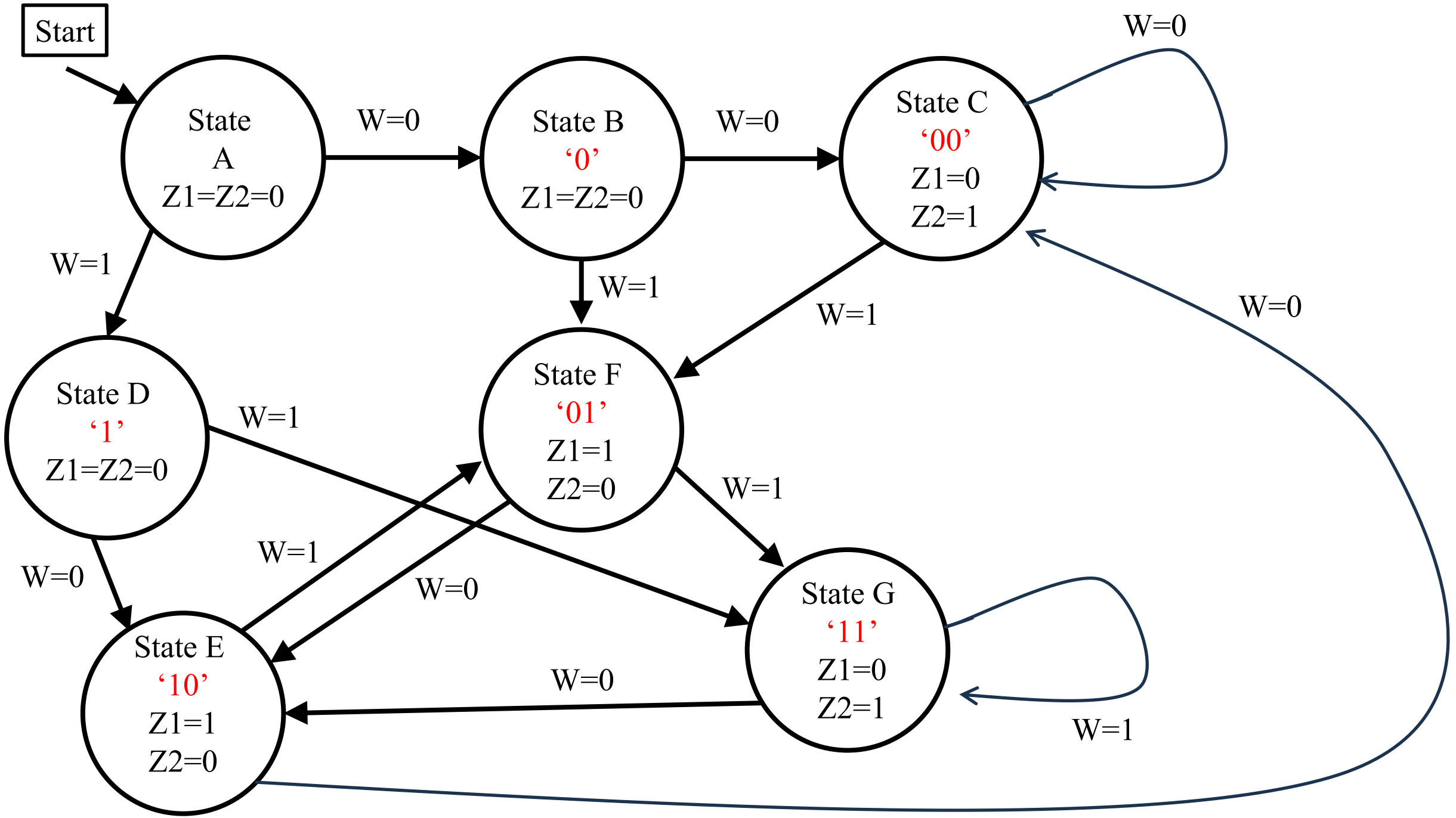
Example: 09: Double output FSM

CLK	1	2	3	4	5	6	7	8	9	10	11	12	13
w	0	1	0	1	0	0	0	0	1	1	0	0	1
Z1	0	0	**1**	**1**	**1**	**1**	0	0	0	1	**0**	1	0
Z2	0	0	0	0	0	0	**1**	**1**	**1**	0	**1**	0	**1**

Solution: Moore Type FSM

For,
w = 01 or 10, Z1 = 1 and Z2 = 0

For,
w = 00 or 11, Z1 = 0 and Z2 = 1


Start
State
A
Z1=Z2=0
W=0
State B
‘0’
Z1=Z2=0
W=0
State C
‘00’
Z1=0
Z2=1
W=0
W=1
W=1
W=1
W=0
State D
‘1’
Z1=Z2=0
W=1
State F
‘01’
Z1=1
Z2=0
W=1
W=0
W=1
W=0
State E
‘10’
Z1=1
Z2=0
State G
‘11’
Z1=0
Z2=1
W=0
W=1

Present State ($Y_2Y_1Y_0$)	Next State ($M_2M_1M_0$)		Output	
	W = 0	W = 1	Z_2	Z_1
A(000)	B	D	0	0
B(001)	C	F	0	0
C(010)	C	F	1	0
D(011)	E	G	0	0
E(100)	C	F	0	1
F(101)	E	G	0	1
G(110)	E	G	1	0
H(111)	d	d	d	d

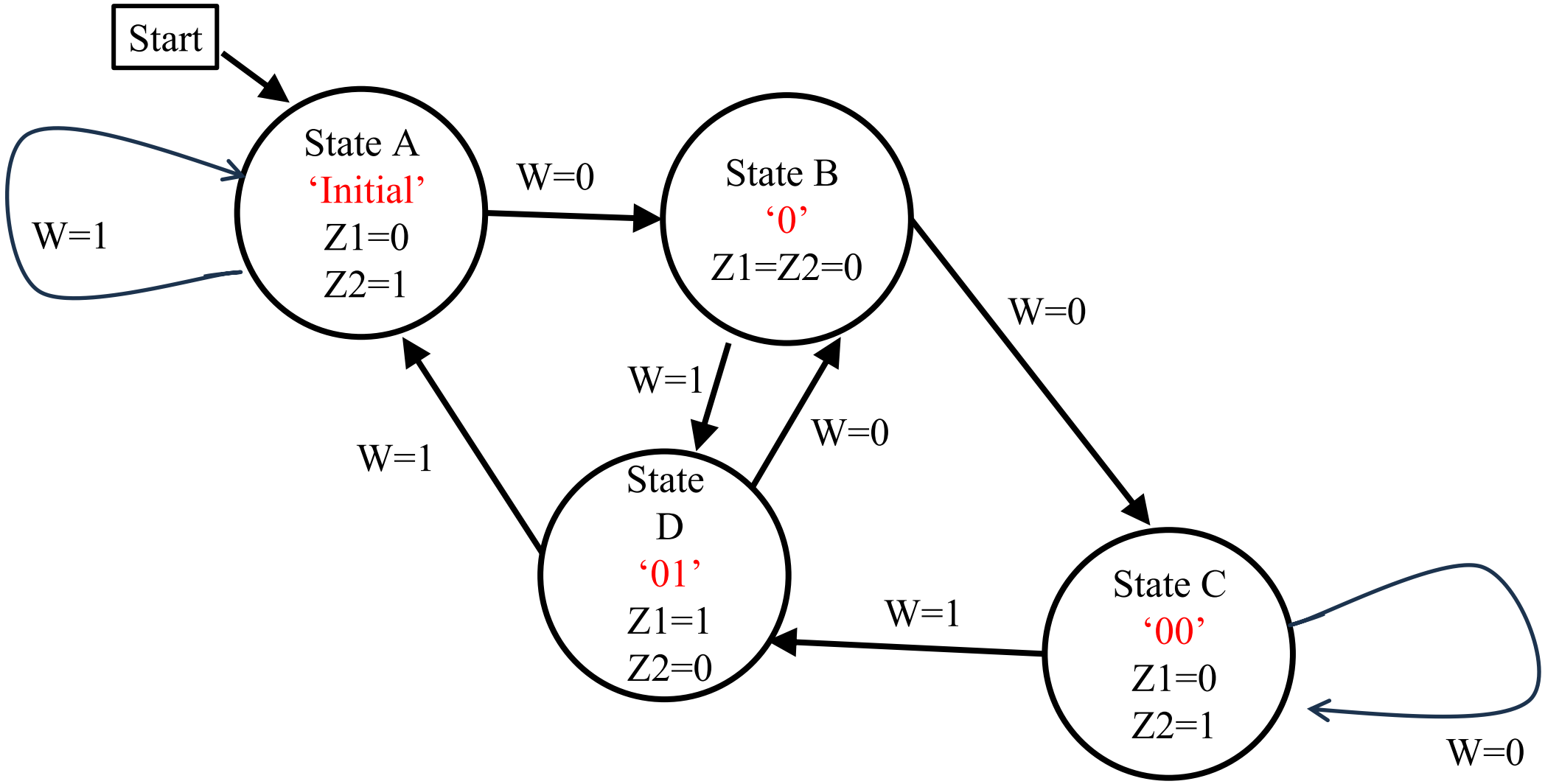
Example: 10: Double output FSM

CLK	1	2	3	4	5	6	7	8	9	10	11	12	13
w	0	1	0	1	0	0	0	0	1	1	0	0	1
Z1	0	0	**1**	0	**1**	0	0	0	0	1	0	0	0
Z2	0	0	0	0	0	0	**1**	**1**	**1**	0	0	0	**1**

Solution: Moore Type FSM

For,
w = 01, Z1 = 1 and Z2 = 0

For,
w = 00, Z1 = 0 and Z2 = 1


Start
State A
'Initial'
Z1=0
Z2=1
W=1
W=0
State B
'0'
Z1=Z2=0
W=0
W=1
W=0
W=1
State
D
'01'
Z1=1
Z2=0
W=1
State C
'00'
Z1=0
Z2=1
W=0

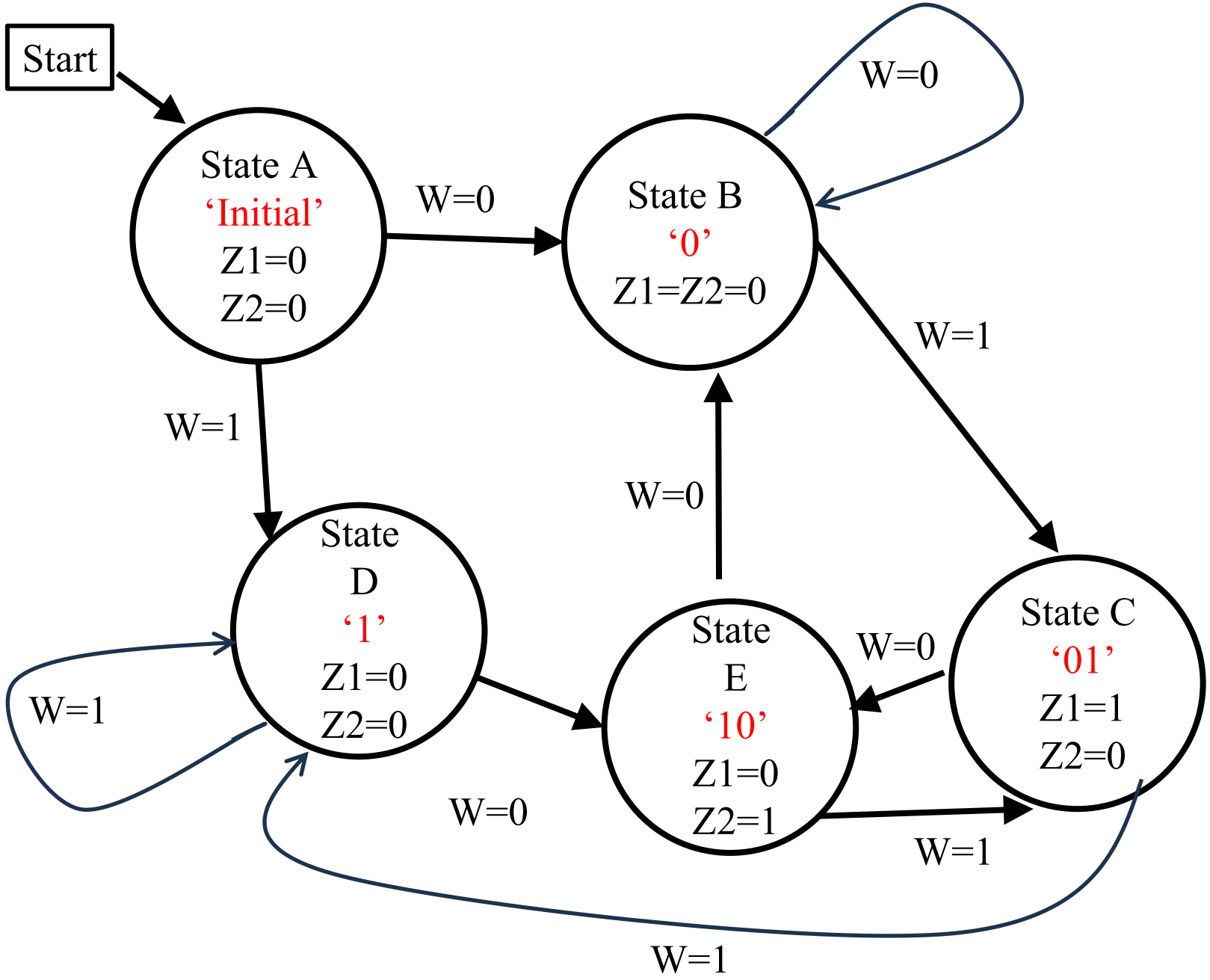
Example: 11: Double output FSM

CLK	1	2	3	4	5	6	7	8	9	10	11	12	13
w	0	1	0	1	0	0	0	0	1	1	0	0	1
Z1	0	0	**1**	0	**1**	0	0	0	0	1	0	0	0
Z2	0	0	0	**1**	0	**1**	0	0	0	0	0	**1**	0

Solution: Moore Type FSM

For,
w = 01, Z1 = 1 and Z2 = 0

For,
w = 10, Z1 = 0 and Z2 = 1


Start
State A
'Initial'
Z1=0
Z2=0
W=0
State B
'0'
Z1=Z2=0
W=0
W=1
W=1
W=0
State
D
'1'
Z1=0
Z2=0
State
E
'10'
Z1=0
Z2=1
State C
'01'
Z1=1
Z2=0
W=0
W=1
W=0
W=1

Encoding Schemes (State Assignment)

Consider a state assigned table below:

	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*

Schemes->

- **Binary encoding**
- Gray encoding
- One-hot encoding

Encoding Schemes (State Assignment)

Consider another state assigned table below:

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

Schemes->

- ➢ Binary encoding
- ➢ **Gray encoding**
- ➢ One-hot encoding

Decimal Number	4 bit Binary Number ABCD	4 bit Gray Code $G_1G_2G_3G_4$
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 1
3	0 0 1 1	0 0 1 0
4	0 1 0 0	0 1 1 0
5	0 1 0 1	0 1 1 1
6	0 1 1 0	0 1 0 1
7	0 1 1 1	0 1 0 0
8	1 0 0 0	1 1 0 0
9	1 0 0 1	1 1 0 1
10	1 0 1 0	1 1 1 1
11	1 0 1 1	1 1 1 0
12	1 1 0 0	1 0 1 0
13	1 1 0 1	1 0 1 1
14	1 1 1 0	1 0 0 1
15	1 1 1 1	1 0 0 0

Encoding Schemes (State Assignment)

Consider another state assigned table below:

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

Schemes->

- ➢ Binary encoding
- ➢ Gray encoding
- ➢ **One-hot encoding**

How many flipflops are required? **Ans: 3**