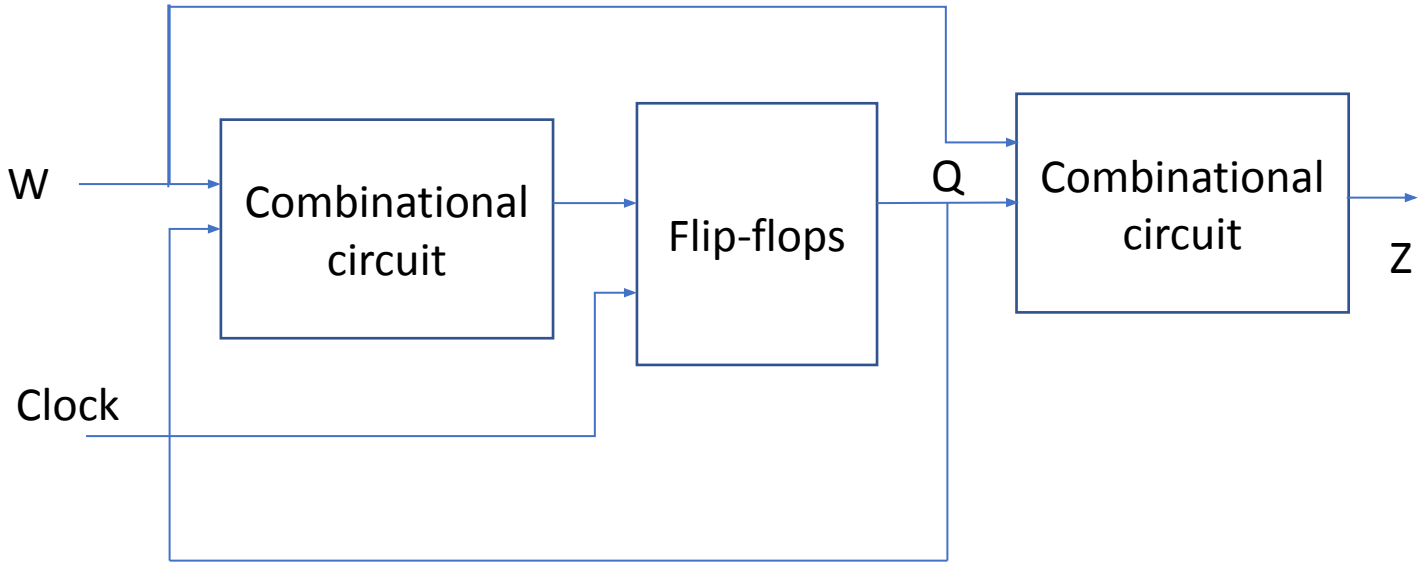
CSE 460: VLSI Design (Lab)

Experiment 3

Simulation of Mealy and Moore Type FSMs in Verilog

Finite State Machine

- In Combinational logic circuits, outputs are determined solely by the present values of the inputs.
- In **Sequential circuits, outputs depend both on the past behavior of the circuit and the present values of the inputs.** The storage elements in terms of flip-flops can reserve the state information of the logic circuit at any given time.
- The sequential circuits in which a clock signal is used to control the operation of the circuit, is called synchronous sequential circuits. These synchronous sequential circuits are known in general as Finite State Machines (FSMs).
- The **building blocks of a finite state machine are combinational circuits and one or more flip-flops**. Under the control of the clock signal, the flip-flops change their state as determined by the combinational logic.



BRAC UNIVERSITY
Inspiring Excellence

Finite State Machine

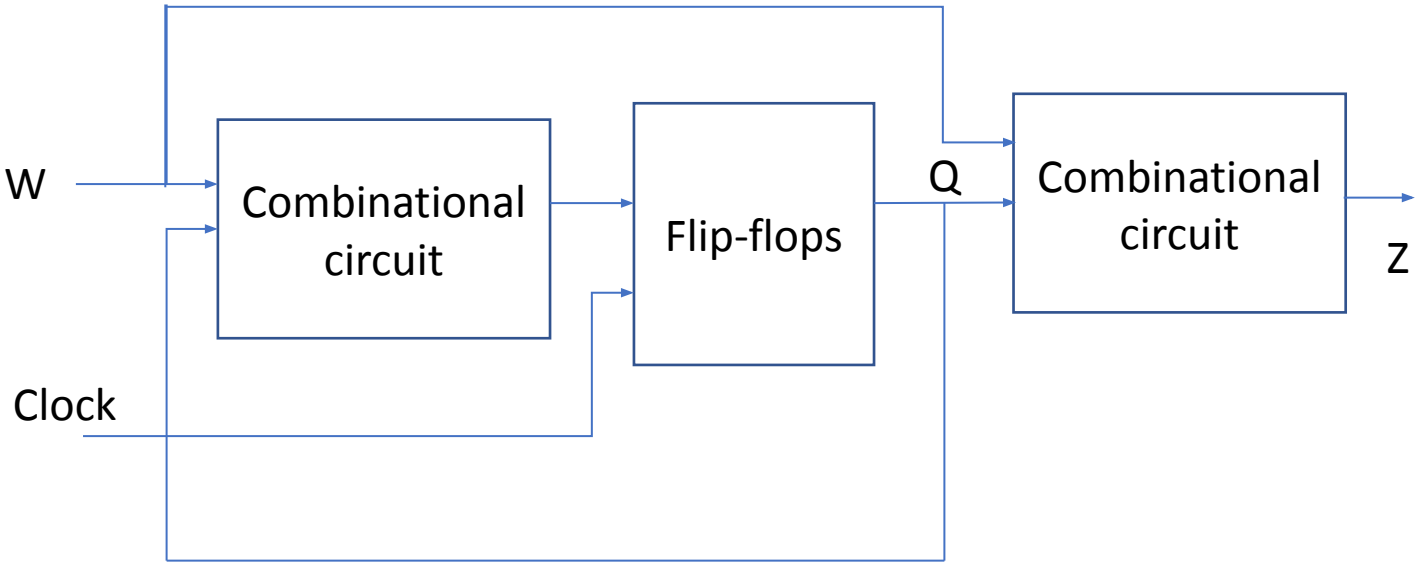
FSMs can be of two types:

1. Moore type FSM

   The sequential circuits whose outputs depend only on the states of the circuit are of Moore type.

2. Mealy type FSM

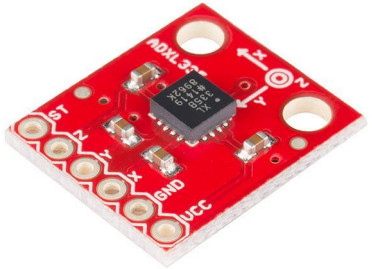
   The sequential circuits whose outputs depend on both the state and the present primary inputs are of Mealy type



BRAC UNIVERSITY
Inspiring Excellence

FSM based motor speed controller

Accelerometer



Motor speed (rpm)

Comparator

Compares the accelerometer reading with the preset reference speed and outputs $w = 1$ if the motor speed exceeds the reference value

w

$500\ rpm$

Finite State Machine (FSM)

Based on the comparator output the FSM will enable the speed controller if $\mathbf{w = 1}$ for two consecutive clock cycles

z

Speed Controller

Reduces the motor speed if the output from the FSM is high i.e. $z = 1$

DC motor

BRAC UNIVERSITY
Inspiring Excellence

Moore Type FSM

Problem:

Suppose that we wish to design a circuit that meets the following specification:

- The circuit has one input, **w**, and one output, **z**.
- All changes in the circuit occur on the positive edge of a clock signal.
- The output **z** is equal to **1** if during **two immediately preceding** clock cycles the input **w** was equal to 1. Otherwise, the value of z is equal to 0.

Input-output combination:

Clock Cycle	1	2	3	4	5	6	7	8	9	10
w	0	1	0	1	1	0	1	1	1	0
z	0	0	0	0	0	1	0	0	1	1

BRAC UNIVERSITY
Inspiring Excellence

Moore Type FSM

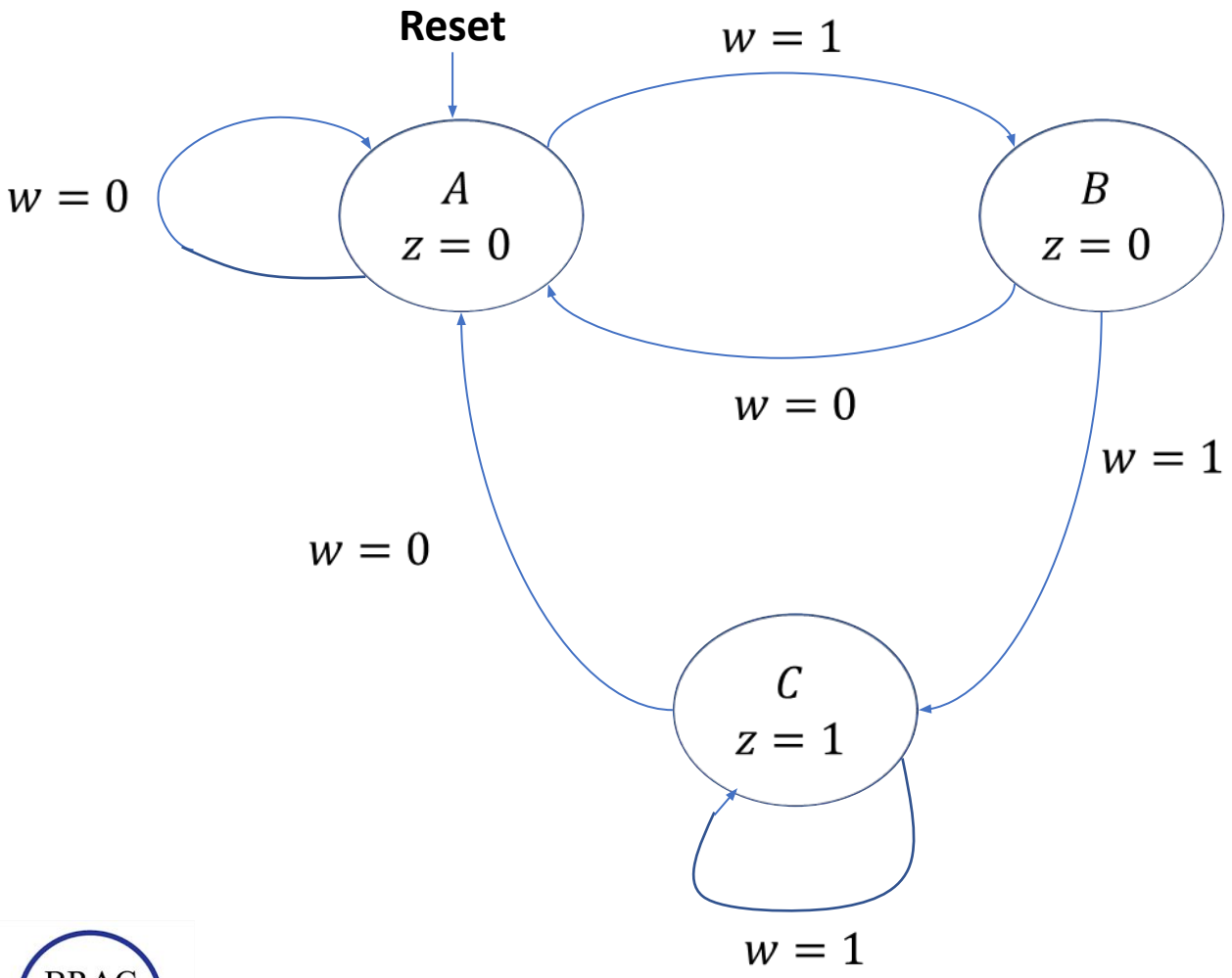


Figure : State Diagram

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

Figure : State Table

BRAC UNIVERSITY
Inspiring Excellence

Moore Type FSM

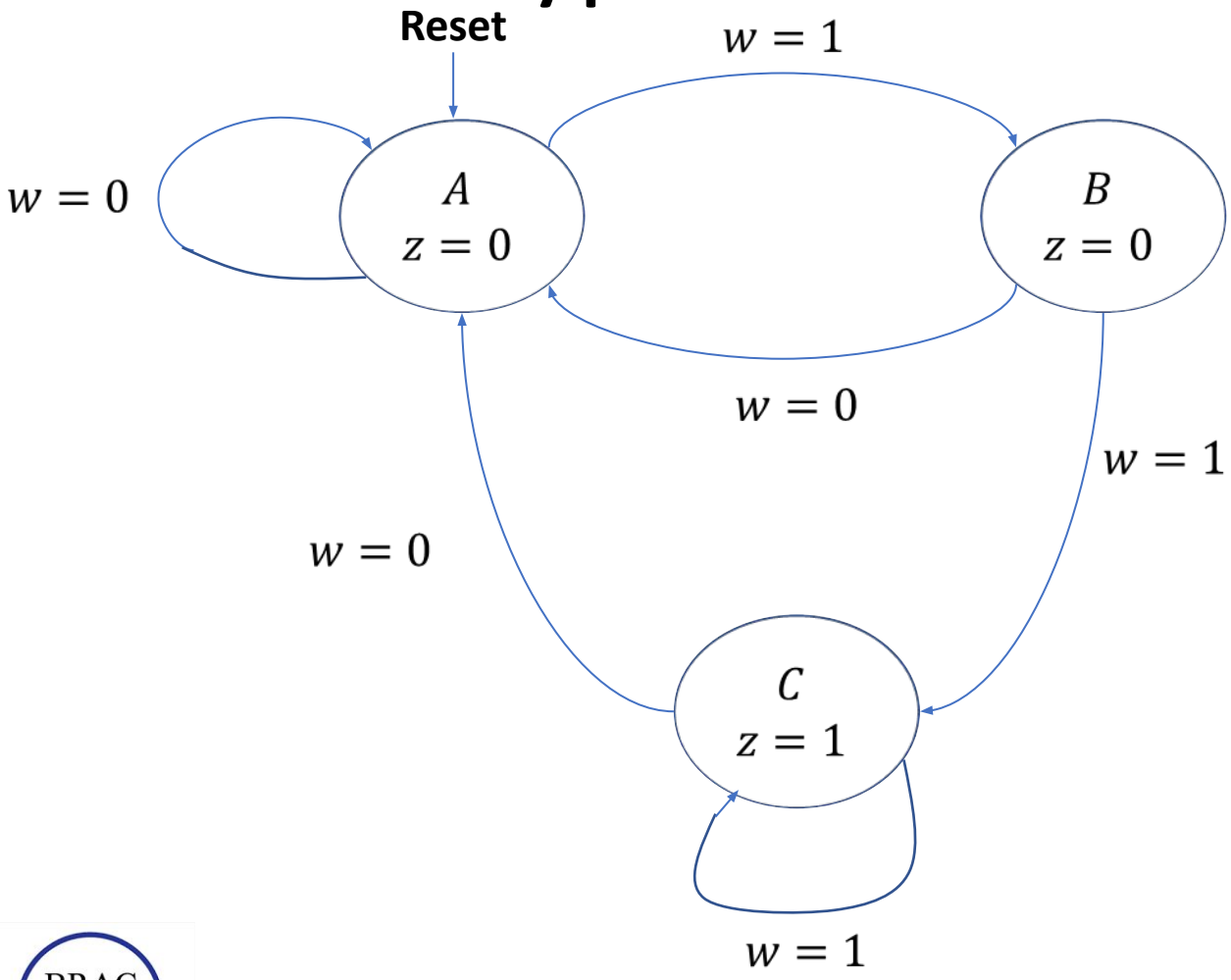


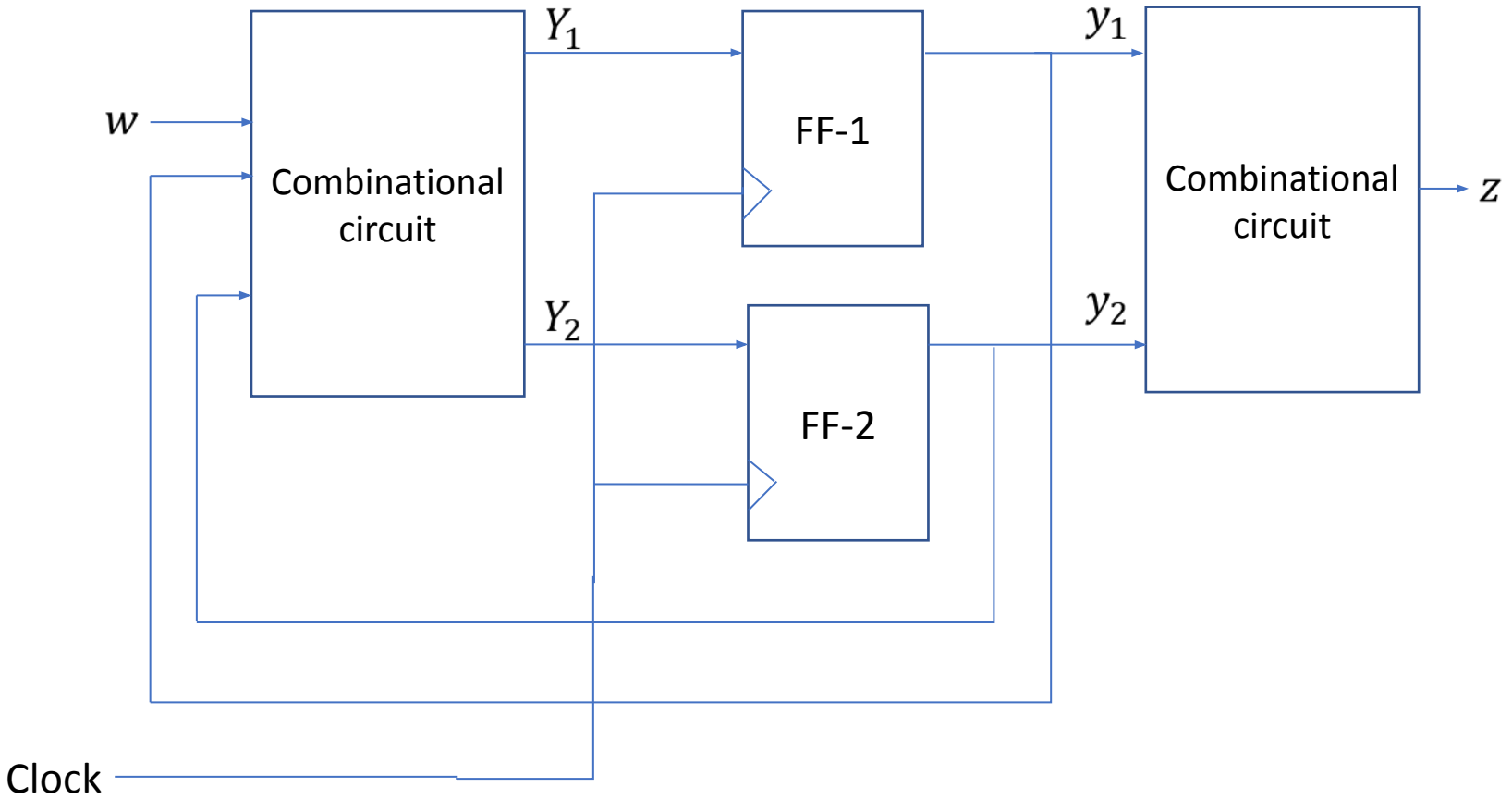
Figure : State Diagram

	Next State		Output
	w=0	w=1	z
00(A)	00(A)	01(B)	0
01(B)	00(A)	10(C)	0
10(C)	00(A)	10(C)	1
11	dd	dd	d

BRAC UNIVERSITY
Inspiring Excellence

General Circuit Diagram of the FSM

	Next State		Output
	w=0	w=1	z
00	00	01	0
01	00	10	0
10	00	10	1
11	dd	dd	d



Designing the FSM in Verilog

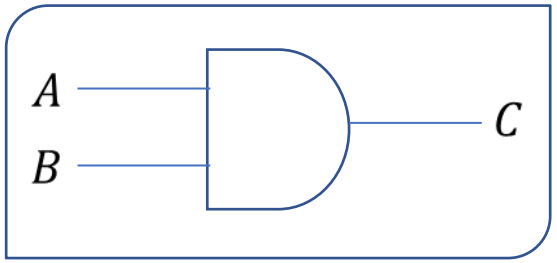
In order to design the FSM using Verilog we will need to perform the following steps:

- Assigning binary variables to states
- A mechanism for figuring out the current state (Sequential circuit block)
- Defining the logic for state transition (Combinational circuit block)
- Defining the output for the current state (Continuous assignment statement)

Combinational circuit synthesis in Verilog

`always@(*)` blocks are used to describe combinational logic in Verilog. *Blocking assignment* statements are to be used inside these blocks.

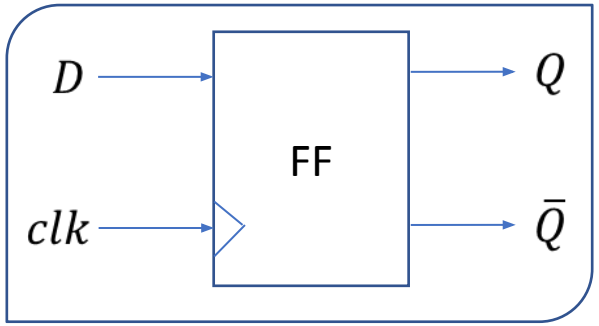
```
always@(*)
begin
    C = A & B;
end
```



Sequential circuit synthesis in Verilog

`always@(posedge clk)` or `always@(negedge clk)` blocks are used to describe combinational logic in Verilog. *Non-blocking assignment* statements are to be used inside these blocks.

```
always@(posedge clk)
begin
    Q <= D;
end
```



Moore Type FSM

	Next State		Output
			z
	w=0	w=1	
00(A)	00(A)	01(B)	0
01(B)	00(A)	10(C)	0
10(C)	00(A)	10(C)	1
11	dd	dd	d

```
1  module testmoore (Clock, Resetn, w, z);
2
3  input Clock, Resetn, w;
4  output z;
5  reg [2:1] y, Y;
6  parameter [2:1] A = 2'b00, B = 2'b01, C = 2'b10;
7
8  // Define the next state combinational circuit
9  always @(w, y)
10     case (y)
11         A:  if (w) Y = B;
12             else Y = A;
13         B:  if (w) Y = C;
14             else Y = A;
15         C:  if (w) Y = C;
16             else Y = A;
17         default: Y = 2'bxx;
18     endcase
19
20 // Define the sequential block
21 always @(negedge Resetn, posedge Clock)
22     if (Resetn == 0) y <= A;
23     else y <= Y;
24
25 // Define output
26 assign z = (y == C);
27 endmodule
```

BRAC UNIVERSITY
Inspiring Excellence

Moore Type FSM

BRAC UNIVERSITY
Inspiring Excellence

Mealy Type FSM

Problem:

We wish to design a circuit that meets the following specification:

- The circuit has one input, **w**, and one output, **z**.
- All changes in the circuit occur on the positive edge of a clock signal.
- The output **z** is equal to **1** if during **two immediate** clock cycles the input **w** was equal to 1. Otherwise, the value of z is equal to 0.

Input-output combination:

Clock Cycle	1	2	3	4	5	6	7	8	9	10
w	0	1	0	1	1	0	1	1	1	0
z	0	0	0	0	1	0	0	1	1	0

BRAC UNIVERSITY
Inspiring Excellence

Mealy Type FSM

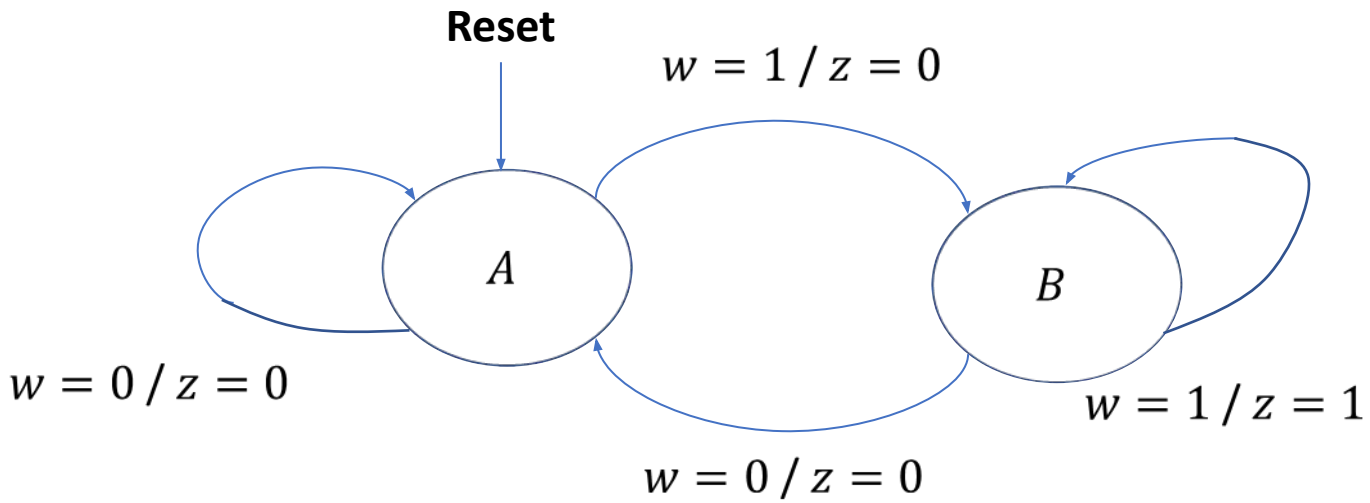


Figure : State Diagram

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 : State Table

Mealy Type FSM

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 : State Table

Present State y	Next State		Output z	
	w=0	w=1	w=0	w=1
	Y	Y		
0(A)	0(A)	1(B)	0	0
1(B)	0(A)	1(B)	0	1

Figure : State-assigned Table

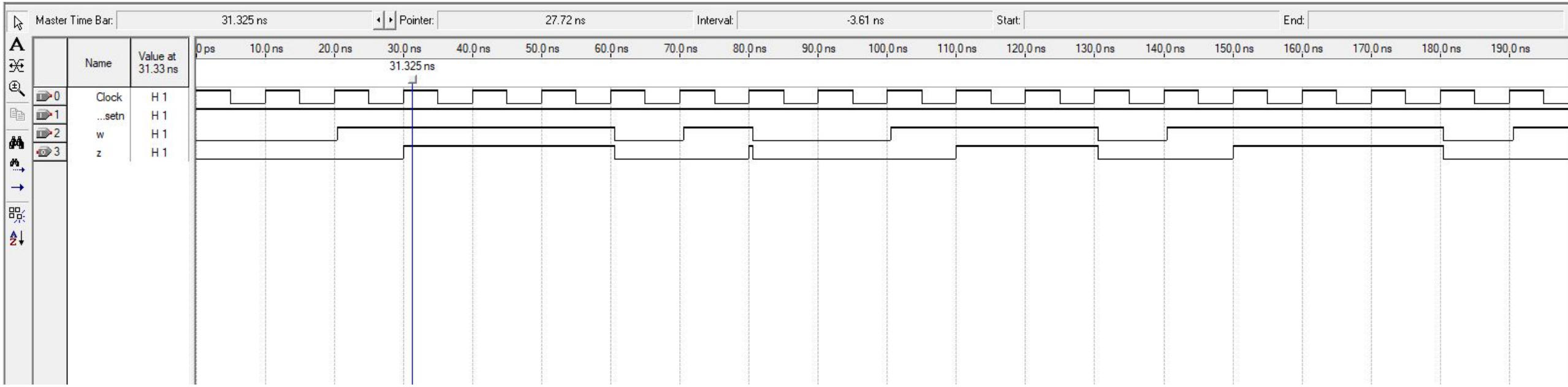
Mealy Type FSM

Present State	Next State		Output z	
y	w=0	w=1	w=0	w=1
	Y	Y		
0(A)	0(A)	1(B)	0	0
1(B)	0(A)	1(B)	0	1

```
1  module testmealy (Clock, Resetn, w, z);
2
3  input Clock, Resetn, w;
4  output reg z;
5  reg y, Y;
6  parameter A = 0, B = 1;
7  // Define the next state and output combinational circuits
8  always @(w, y)
9      case (y)
10         A: if (w)
11         begin
12         z = 0;
13         Y = B;
14         end
15         else
16         begin
17         z = 0;
18         Y = A;
19         end
20         B: if (w)
21         begin
22         z = 1;
23         Y = B;
24         end
25         else
26         begin
27         z = 0;
28         Y = A;
29         end
30     endcase
31
32 // Define the sequential block
33 always @(negedge Resetn, posedge Clock)
34     if (Resetn == 0) y <= A;
35     else y <= Y;
36 endmodule
```

BRAC UNIVERSITY
Inspiring Excellence

Mealy Type FSM



BRAC UNIVERSITY
Inspiring Excellence

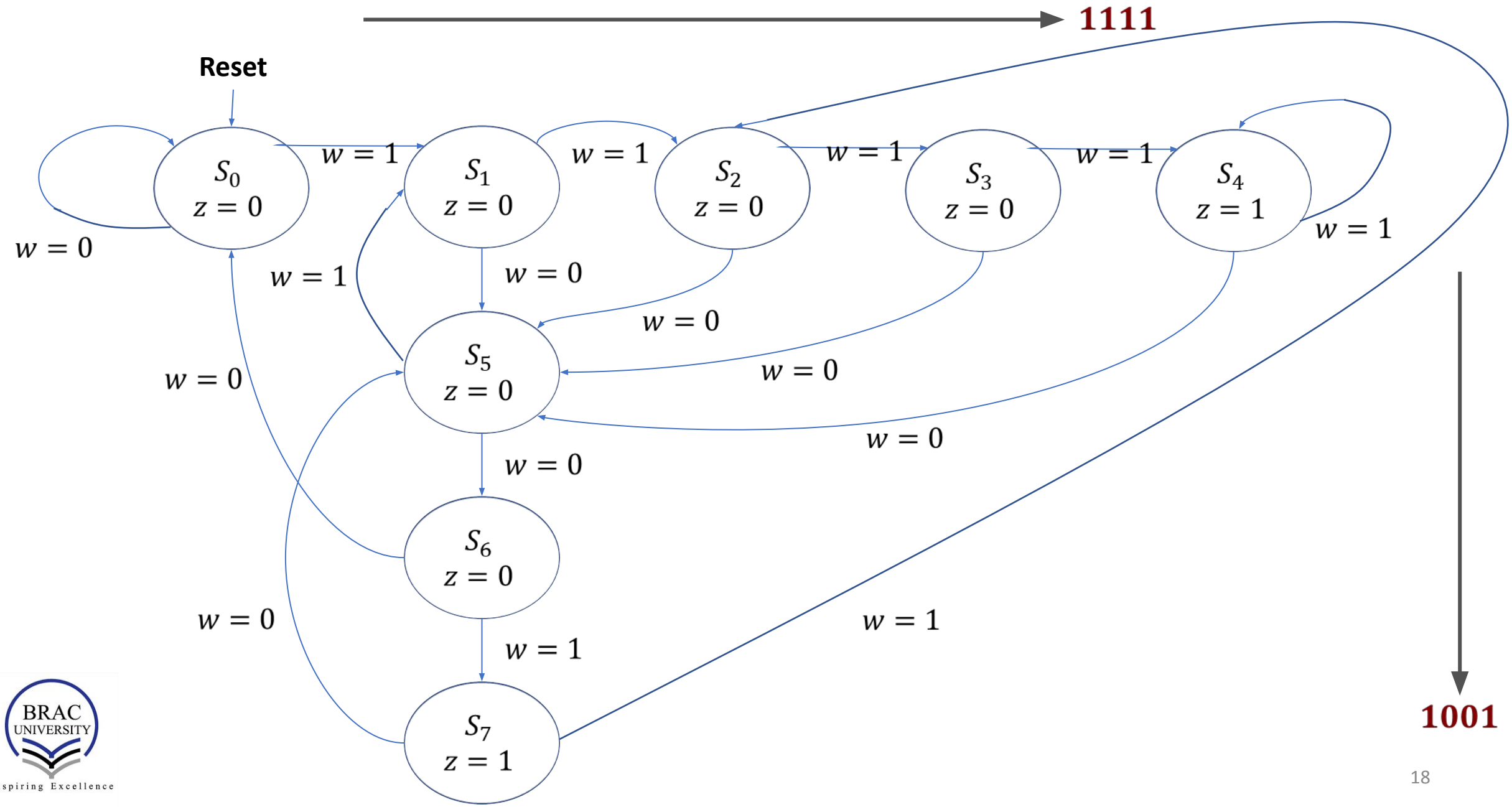
An Example

Problem: Derive the state diagram for an FSM that has an input w and an output z. The machine has to generate z = 1 when the previous four values of w were 1001 or 1111; otherwise, z = 0.Overlapping input patterns are allowed. An example of the desired behavior is

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
w	0	1	0	1	1	1	1	0	0	1	1	0	0	1	1	1	1
z	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0

BRAC UNIVERSITY
Inspiring Excellence

State diagram(Moore type)



State table (Moore type)

Present State	Next State		Output
	w=0	w=1	z
S0	S0	S1	0
S1	S5	S2	0
S2	S5	S3	0
S3	S5	S4	0
S4	S5	S4	1
S5	S6	S1	0
S6	S0	S7	0
S7	S5	S2	1

Figure : State Table

	Next State		Output
	w=0	w=1	z
000(S0)	000(S0)	001(S1)	0
001(S1)	101(S5)	010(S2)	0
010(S2)	101(S5)	011(S3)	0
011(S3)	101(S5)	100(S4)	0
100(S4)	101(S5)	100(S4)	1
101(S5)	110(S6)	001(S1)	0
110(S6)	000(S0)	111(S7)	0
111(S7)	101(S5)	010(S2)	1

Figure : State-assigned Table

State assigned table (Moore Type)

	Next State		Output
	w=0	w=1	z
000(S0)	000(S0)	001(S1)	0
001(S1)	101(S5)	010(S2)	0
010(S2)	101(S5)	011(S3)	0
011(S3)	101(S5)	100(S4)	0
100(S4)	101(S5)	100(S4)	1
101(S5)	110(S6)	001(S1)	0
110(S6)	000(S0)	111(S7)	0
111(S7)	101(S5)	010(S2)	1

examplemoore.v | Compilation Report - Flow Summary | examplemoo

```
1  module examplemoore(Clock, Resetn, w,z);
2
3  input Clock, Resetn, w;
4  output z;
5  reg[2:0] y;
6  parameter [2:0] S0=0, S1=1, S2=2, S3=3, S4=4, S5=5, S6=6, S7=7;
7
8  always@(posedge Clock, negedge Resetn)
9  begin
10     if(Resetn==0) y<=S0;
11     else
12     begin
13         case(y)
14         S0: if(w) y<=S1;
15             else y<=S0;
16         S1: if(w) y<=S2;
17             else y<=S5;
18         S2: if(w) y<=S3;
19             else y<=S5;
20         S3: if(w) y<=S4;
21             else y<=S5;
22         S4: if(w) y<=S4;
23             else y<=S5;
24         S5: if(w) y<=S1;
25             else y<=S6;
26         S6: if(w) y<=S7;
27             else y<=S0;
28         S7: if(w) y<=S2;
29             else y<=S5;
30         endcase
31     end
32 end
33 assign z=(y==S4)|(y==S7);
34 endmodule
```

BRAC UNIVERSITY
Inspiring Excellence

Timing diagrams (Moore Type)

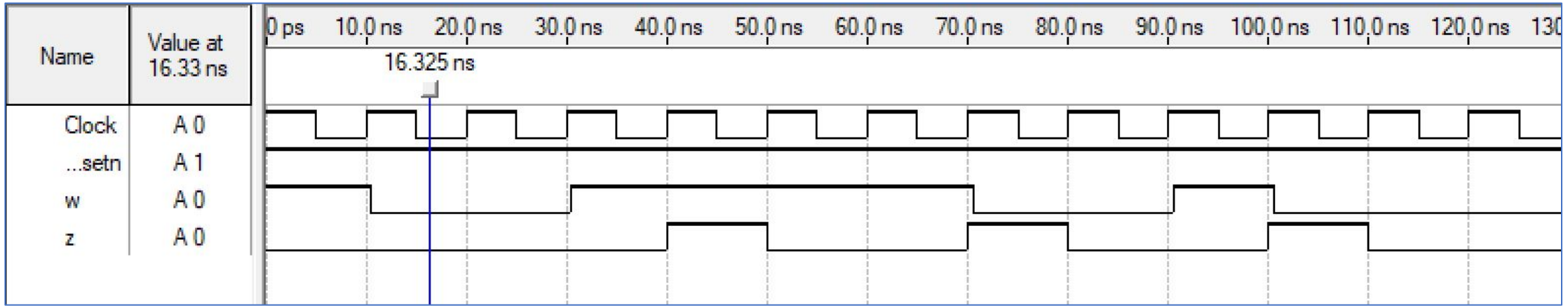


Figure : Output waveform

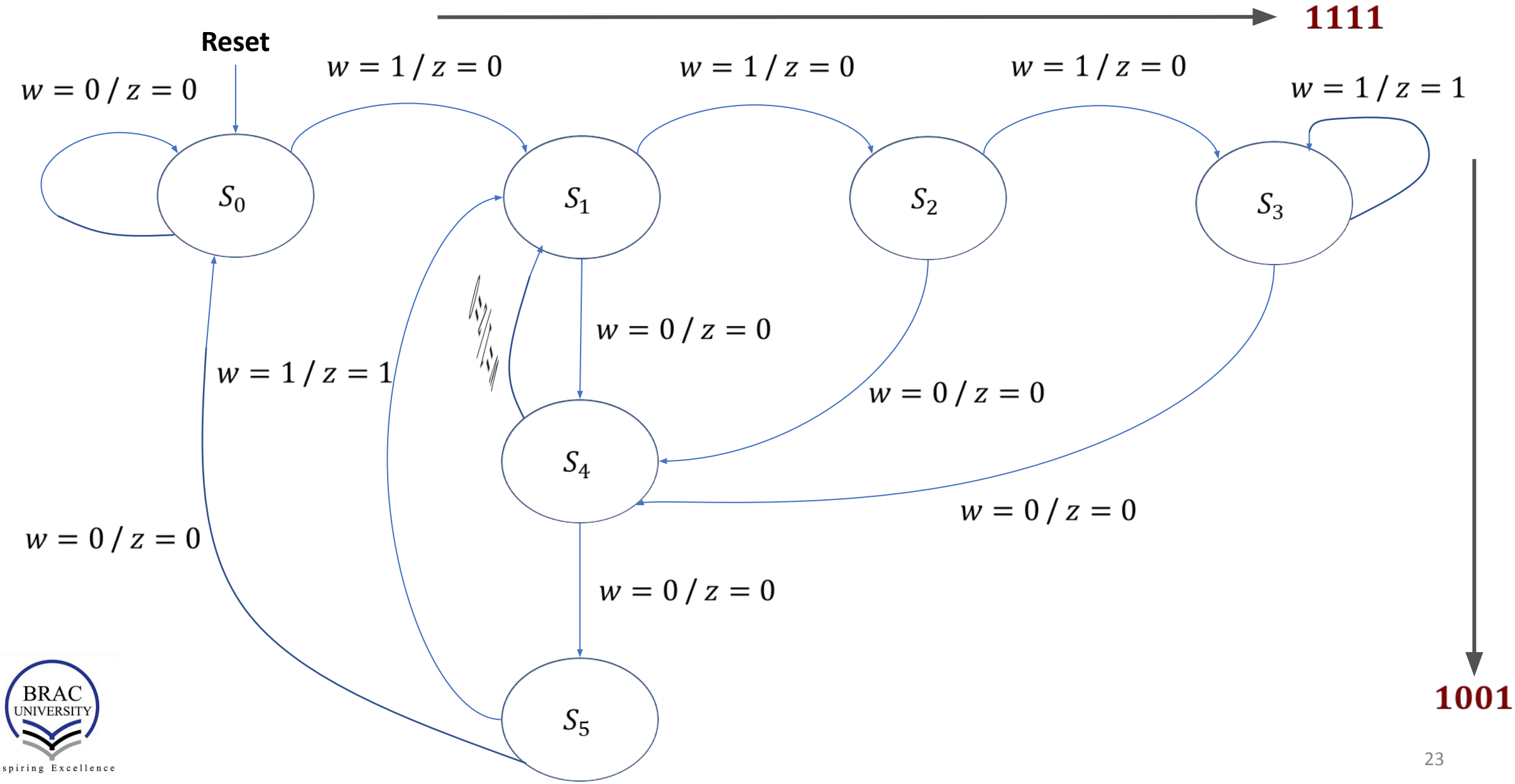
An Example

- Problem: Derive the state diagram of the Mealy version for the example FSM that is, the FSM will generate $z = 1$ at the instant the input values of w have the pattern 1111 or 1001. Overlapping of the input patterns are allowed.

Clock cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
w	0	1	0	1	1	1	1	0	0	1	1	0	0	1	1	1	1
z	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0	1

BRAC UNIVERSITY
Inspiring Excellence

State diagram(Mealy type)



State table (Mealy type)

Present State	Next State		Output z	
	w=0	w=1	w=0	w=1
S0	S0	S1	0	0
S1	S4	S2	0	0
S2	S4	S3	0	0
S3	S4	S3	0	1
S4	S5	S1	0	0
S5	S0	S1	0	1

Figure : State Table

	Next State		Output z	
	w=0	w=1	w=0	w=1
000(S0)	000(S0)	001(S1)	0	0
001(S1)	100(S4)	010(S2)	0	0
010(S2)	100(S4)	011(S3)	0	0
011(S3)	100(S4)	011(S3)	0	1
100(S4)	101(S5)	001(S1)	0	0
101(S5)	000(S0)	001(S1)	0	1

Figure : State-assigned Table

BRAC UNIVERSITY
Inspiring Excellence

State assigned table (Mealy type)

	Next State		Output z	
	w=0	w=1	w=0	w=1
000(S0)	000(S0)	001(S1)	0	0
001(S1)	100(S4)	010(S2)	0	0
010(S2)	100(S4)	011(S3)	0	0
011(S3)	100(S4)	011(S3)	0	1
100(S4)	101(S5)	001(S1)	0	0
101(S5)	000(S0)	001(S1)	0	1

Figure : State-assigned Table

```
1  module examplemealy (Clock, Resetn, w, z);
2
3  input Clock, Resetn, w;
4  output reg z;
5  reg [3:1]y, Y;
6  parameter S0 = 3'b000, S1 = 3'b001, S2=3'b010, S3 = 3'b011 , S4=3'b100 , S5 = 3'b101;
7
8  // Define the next state and output combinational circuits
9  always @(w, y)
10     case (y)
11         S0: if (w)
12                 begin
13                     z = 0;
14                     Y = S1;
15                 end
16             else
17                 begin
18                     z = 0;
19                     Y = S0;
20                 end
21
22         S1: if (w)
23                 begin
24                     z = 0;
25                     Y = S2;
26                 end
27             else
28                 begin
29                     z = 0;
30                     Y = S4;
31                 end
32
33         S2: if (w)
34                 begin
35                     z = 0;
36                     Y = S3;
37                 end
38             else
39                 begin
40                     z = 0;
41                     Y = S4;
42                 end
43
```

Figure : Verilog code (part 1)

BRAC UNIVERSITY
Inspiring Excellence

State assigned table (Mealy type)

	Next State		Output z	
	w=0	w=1	w=0	w=1
000(S0)	000(S0)	001(S1)	0	0
001(S1)	100(S4)	010(S2)	0	0
010(S2)	100(S4)	011(S3)	0	0
011(S3)	100(S4)	011(S3)	0	1
100(S4)	101(S5)	001(S1)	0	0
101(S5)	000(S0)	001(S1)	0	1

Figure : State-assigned Table

```
43
44          S3: if (w)
45              begin
46                  z = 1;
47                  Y = S3;
48              end
49          else
50              begin
51                  z = 0;
52                  Y = S4;
53              end
54
55          S4: if (w)
56              begin
57                  z = 0;
58                  Y = S1;
59              end
60          else
61              begin
62                  z = 0;
63                  Y = S5;
64              end
65
66          S5: if (w)
67              begin
68                  z = 1;
69                  Y = S1;
70              end
71          else
72              begin
73                  z = 0;
74                  Y = S0;
75              end
76          default:
77              begin
78                  z=0;
79                  Y=3'bxxx;
80                  end
81
82      endcase
83
84  // Define the sequential block
85  always @(negedge Resetn, posedge Clock)
86      if (Resetn == 0) y <= S0;
87      else y <= Y;
88  endmodule
```

Figure : Verilog code (part 2)

Timing diagrams (Mealy Type)

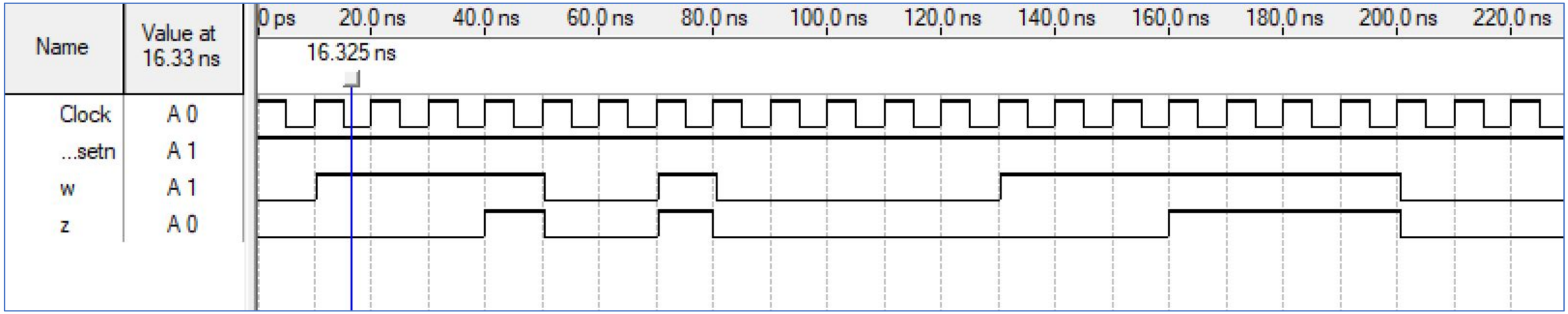


Figure : Output waveform