Brac University

EEE 412/ ECE 412/ CSE 460

Verilog Design Laboratory

Experiment Title: Implementing Finite State Machines using Verilog HDL

Finite State Machines:

A finite state machine is a computation model that can be implemented with hardware or software and can be used to simulate sequential logic. A system where particular inputs cause particular changes in state can be represented using finite state machines.

There can be two types of Finite State Machines (FSM) based on the dependence of output on input:

Moore Type Machines: The sequential circuits whose outputs depend only on the states of the circuit are of Moore type. It has been named after Edward Moore.

Example: Suppose that we wish to design a 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 cycles** the input w was equal to 1. Otherwise, the value of z is equal to 0.

Thus the system follows the following 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

We can see that in 6^{th} clock cycle, input is 0 and output is 1, again in clock cycle 9, input is 1 and output is also 1. Thus the output is not depending on present input, rather they are depending on the past two states. This type of machine is Moore Type Machine.

Mealy Type Machines: The sequential circuits whose outputs depend on both the state and the present primary inputs are of Mealy type. It has been named after George Mealy.

Example:

We can modify the example of Moore Machine in the following way:

We wish to design a 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 immediate clock cycles** the input w is equal to 1. Otherwise, the value of z is equal to 0.

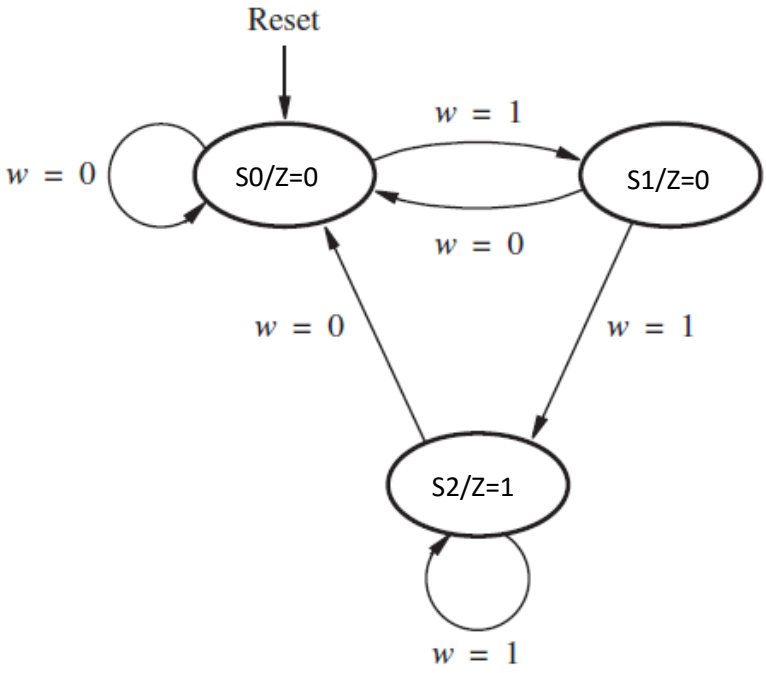
Thus the system follows the following input output combination.

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

We can see that for the output to be 1, the input must be 1 and the previous state must also be 1. This it is a Mealy machine.

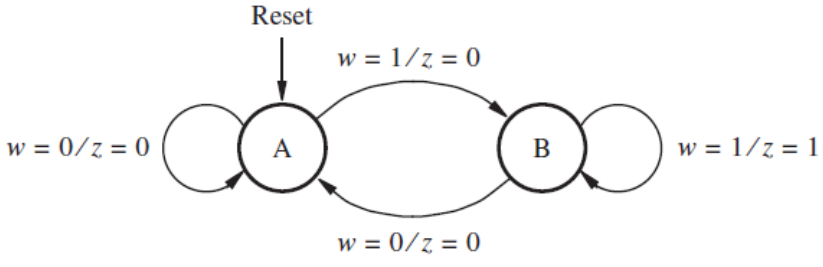
State Diagram:

The Moore machine of the example can be represented by the following state diagram:



The initial state is S0. When the input is 0, the system remains in state S0 and output is 0. When input is 1, at the next clock cycle, system moves to state S1 but the output still remains 0. If at the next clock cycle, input is still 1, the system moves to state S2 and at the next clock cycle, output turns 1. If the input is 0, the system returns to state S0. At state S2, the system remains in state S2 if input is 1 and the output continues to be 1. But if input is 0, system moves back to state S0 and output turns 0 at next clock cycle.

The Mealy machine of the example can be represented by the following state diagram:



Initially the system is at state S0 and remains at S0 as long as input is 0. When input is 1, state changes to S1 at next clock cycle. When at S1, if input is 1, output turns 1 at next clock cycle and the system still remains in S1. If input is 0, the output turns 0 and state moves to S0 at next clock.

Verilog Representation:

Following is the Verilog representation of the example presented in Moore Type FSM:

```
module moore(Clock, Resetn, w, z);
input Clock, Resetn, w;
output reg z;
reg [2:1] y;
parameter [1:0] S0 = 0, S1 = 1, S2 = 2;
always @(posedge Clock, negedge Resetn)
begin
      if (Resetn == 0) y <= S0;
      else
      begin
            z <= (y == S2);
            case (y)
            S0: if (w) y<= S1;
            else y<= S0;
            S1: if (w) y<= S2;
            else y<= S0;
            S2: if (w) y<= S2;
            else y<= S0;
            default: y<= 2'bxx;
            endcase
      end
end
endmodule
```

Following is the Verilog representation of the example presented in Mealy Type FSM:

```
module mealy(z,w,clk,rst);
input clk, rst, w;
output reg z;
reg y;
parameter S0=0, S1=1;

always @(posedge clk, posedge rst)
      if (rst) y<=S0;
      else
            case(y)
            S0: if (w)
            begin
                  z<=0;
                  y<=S1;
            end
            else
            begin
                  z<=0;
                  y<=S0;
            end
```

```
            S1: if (w)
            begin
                  z<=1;
                  y<=S1;
            end
            else
            begin
                  z<=0;
                  y<=S0;
            end
      endcase
endmodule
```

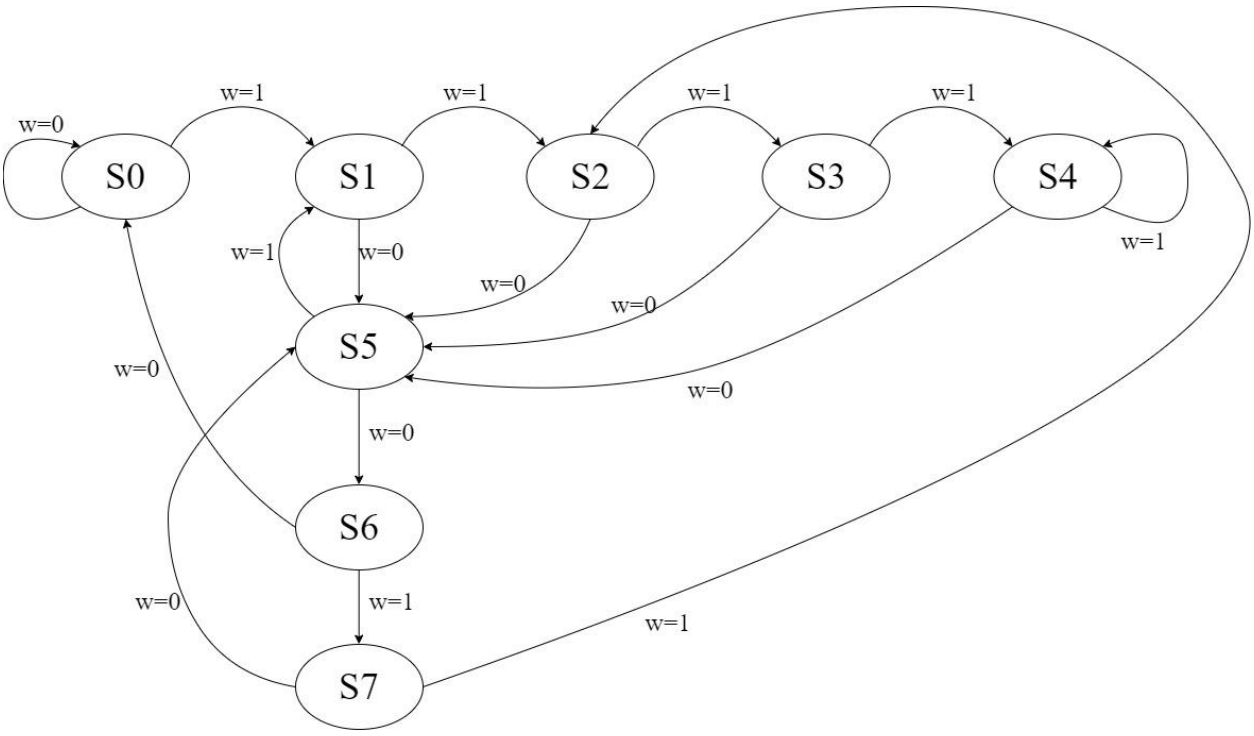
Example 1:

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

w : 0 1 0 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1

z : 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 1 1

State Diagram:



Verilog Representation:

```
module example1 (Clock, Resetn, w, z);
input Clock, Resetn, w;
output z;
reg [2:0] y;
parameter [2:0] S0 = 0, S1 = 1, S2 = 2, S3=3, S4=4, S5=5, S6=6, S7=7;
```

```
always @(posedge Clock, negedge Resetn)
begin
      if (Resetn == 0) y <= S0;
      else
      begin
            case (y)
            S0: if (w) y<=S1;
            else y<=S0;
            S1: if (w) y<=S2;
            else y<=S5;
            S2: if (w) y<=S3;
            else y<=S5;
            S3: if (w) y<=S4;
            else y<=S5;
            S4: if (w) y<=S4;
            else y<=S5;
            S5: if (w) y<=S1;
            else y<=S6;
            S6: if (w) y<=S7;
            else y<=S0;
            S7: if (w) y<=S2;
            else y<=S5;
            endcase
      end
end
assign z=(y==S4)|(y==S7);
endmodule
```

Homework:

1. A sequential circuit has two inputs, w1 and w2, and an output, z. Its function is to compare the input sequences on the two inputs. If w1 = w2 during any four consecutive clock cycles, the circuit produces z = 1; otherwise, z = 0. For example

w1 : 0 1 1 0 1 1 1 0 0 0 1 1 0
w2 : 1 1 1 0 1 0 1 0 0 0 1 1 1
z : 0 0 0 0 1 0 0 0 0 1 1 1 0

2. An FSM is defined by the state-assigned table in Figure:

Present State	Next State		Output (z)
	Input (w=0)	Input (w=1)	
S0	S2	S3	0
S1	S1	S0	0
S2	S3	S0	0
S3	S1	S1	1

Draw the state diagram and implement the system using Verilog code.