CSCI 355 - Lab 7 Report

Student: Luka Karanovic
Student #: 665778833
Instructor: Ajay Shrestha

1 - Pre-Lab Tasks

Compare Moore and Mealy model (in terms of advantages and disadvantages).

Moore models are more simple, making them easier to implement (conceptually) and their outputs are stable and predictable, as they change only on a clock edge.

Moore models require more states to implement the same function, increasing hardware cost and power consumption. Also, there is a slower response time since the output changes only after the clock edge, meaning that there is a delay between changes in input and output.

Mealy models have output values generated based on both the state of the circuit and its present inputs. This provides additional flexibility to the design of sequential circuits, as there are more possible actions you can take in a Mealy model. Additionally, the Mealy model requires less states, which is less hardware, making it cheaper to implement. The output for Mealy model is typically faster than a Moore model because it can change immediately when the input changes, without waiting for the next clock cycle (one cycle earlier).

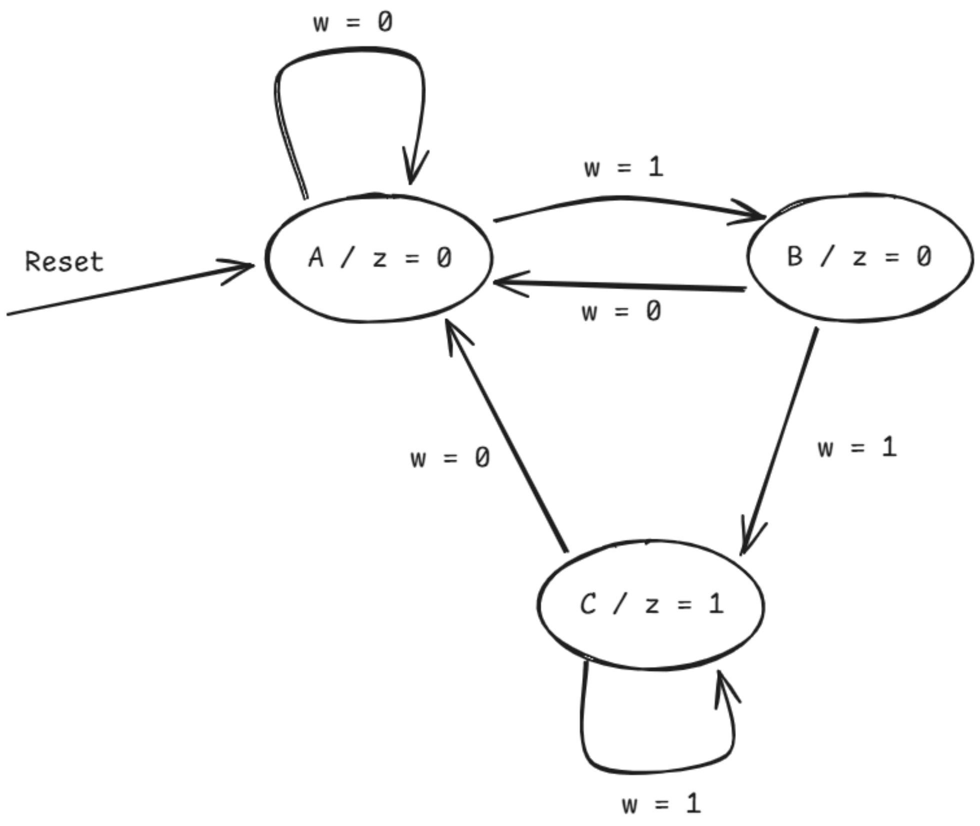
However, Mealy models can sometimes be unstable or glitchy, and debugging them is harder as the present input factor makes it more complex.

2 - Lab Procedures

6.1 Simple Moore Module

- Write the behavioral Verilog code for a Moore-type FSM that reads binary sequence w and sets z=1 if 11 pattern (with overlaps) is detected.
- Draw state diagram.
- Simulate using Vivado ML to verify the operation.
- Write Verilog testbench to simulate your design.
- Show screenshot of the Waveform window indicating correct operation.

State Diagram:



Verilog Code:

Filename: simpleMoore.v

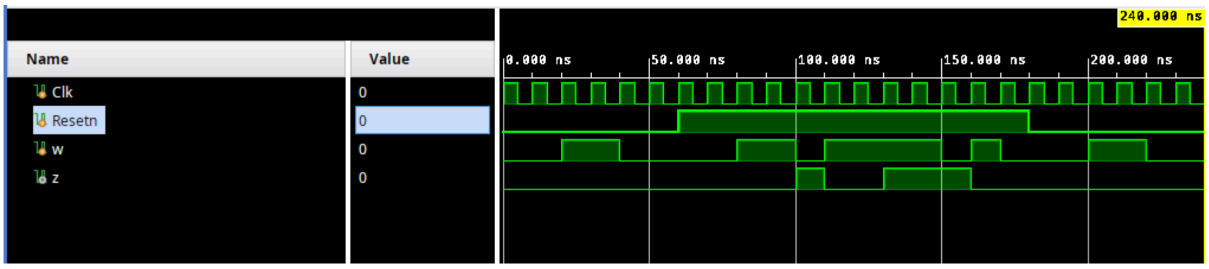
```
// Module to implement a simple Moore FSM that outputs 1 if there is a 11 pattern in input w.
module simpleMoore(Clk, Resetn, w, z);

   input wire Clk, Resetn, w;
   output reg z;

   reg [1:0] y, Y; // 2-bit current and next state variables
   parameter A = 2'b00, B=2'b01, C = 2'b10; // Define states

   // Determine next state based on current state and w
   always @(w, y) begin
      case (y)
         A: if (w) Y = B;
            else Y = A;
         B: if (w) Y = C;
            else Y = A;
         C: if (w) Y = C;
            else Y = A;
         default: Y = 2'bxx; // Don't care

      endcase
      if (y == C) z = 1;
```

```
        else z = 0;
    end


    // Update current state + handle async reset
    always @(negedge Resetn, posedge Clk) begin
        #10;
        if (Resetn == 0) y <= A;
        else y <= Y;
    end

endmodule
```

Testbench Verilog Code:
Filename: simpleMoore_tb.v

```
module simpleMoore_tb();
    reg Clk = 0;
    reg Resetn = 0;
    reg w = 0;
    wire z;

    simpleMoore UT(Clk, Resetn, w, z);

    always begin
        Clk = ~Clk;
        #5;
    end

    initial begin
        w = 0; #10;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        w = 0; #10;
        w = 0; #10;
        Resetn = 1;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        w = 1; #10;
        w = 1; #10;
        w = 0; #10;
        w = 1; #10;
        w = 0; #10;
        Resetn = 0;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        w = 0; #10;
        w = 0; #10;
        $finish;
    end

endmodule
```

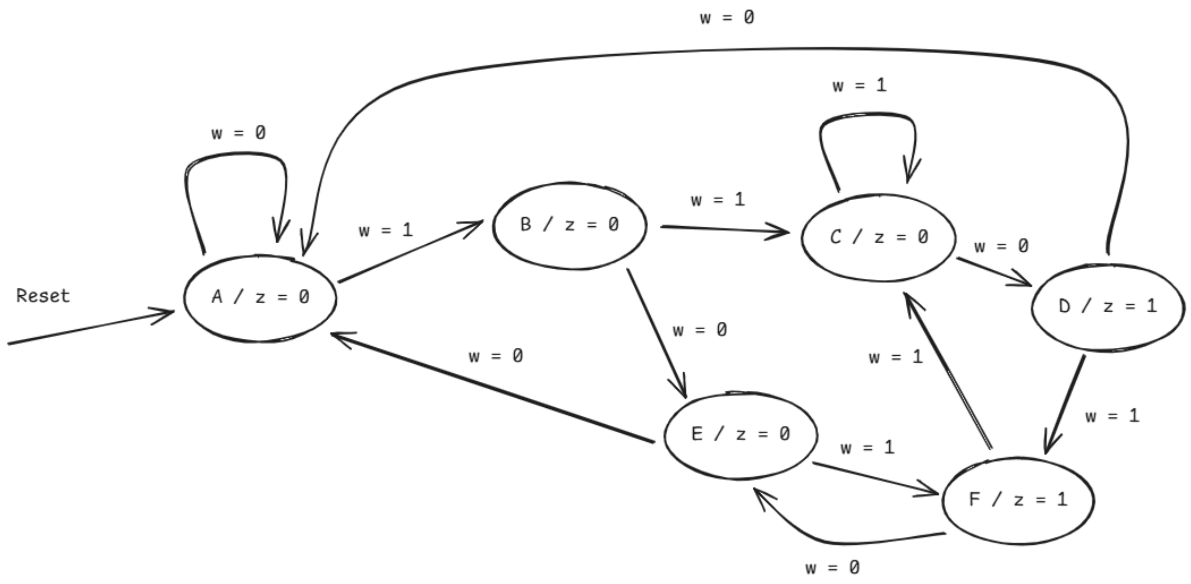
Waveform:



6.2 Moore-Type FSM

- Write the behavioral Verilog code for a Moore-type FSM that reads binary sequence w and sets z=1 if either a 110 or 101 pattern (with overlaps) is detected.
- Draw state diagram.
- Simulate using Vivado ML to verify the operation.
- Write Verilog testbench to simulate your design.
- Show screenshot of the Waveform window indicating correct operation

State Diagram:



Verilog Code:

<u>Filename: complexMoore.v</u>

// Module to implement a more complex Moore FSM that outputs 1 if there is a 110 or 101 pattern in input w.

```
module complexMoore(Clk, Reset, w, z);
   input wire Clk, Reset, w;
   output reg z;


   reg [2:0] y, Y; // 3-bit current and next state variables
   parameter A = 3'b000, B=3'b001, C = 3'b011, D = 3'b110, E = 3'b010, F = 3'b101; // Define
states

   // Determine next state based on current state and w
   always @(w, y) begin
      case (y)
         A: if (w) Y = B;
            else Y = A;
         B: if (w) Y = C;
            else Y = E;
         C: if (w) Y = C;
            else Y = D;
         D: if (w) Y = F;
            else Y = A;
         E: if (w) Y = F;
            else Y = A;
         F: if (w) Y = C;
            else Y = E;
         default: Y = 3'bxxx; // Don't care

      endcase
      if (y == D | y == F) z = 1;
      else z = 0;
   end


   // Update current state + handle async reset
   always @(negedge Reset, posedge Clk) begin
      #10; // Clock delay for Moore model
      if (Reset == 0) y <= A;
      else y <= Y;
   end

endmodule
```

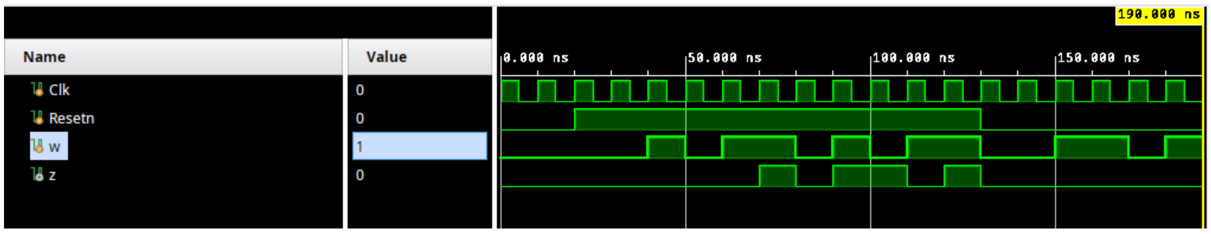
Testbench Verilog Code:

Filename: complexMoore_tb.v

```
module complexMoore_tb();

   reg Clk = 0;
   reg Resetn = 0;
   reg w;
```

```
    wire z;

    complexMoore UT(Clk, Resetn, w, z);

    always begin
        Clk = ~Clk;
        #5;
    end

    initial begin
        w = 0; #10;
        w = 0; #10;
        Resetn = 1;
        w = 0; #10;
        w = 0; #10;
        w = 1; #10;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        w = 0; #10;
        w = 1; #10;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        Resetn = 0;
        w = 0; #10;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        w = 0; #10;
        w = 1; #10;
        $finish;
    end

endmodule
```

Waveform:

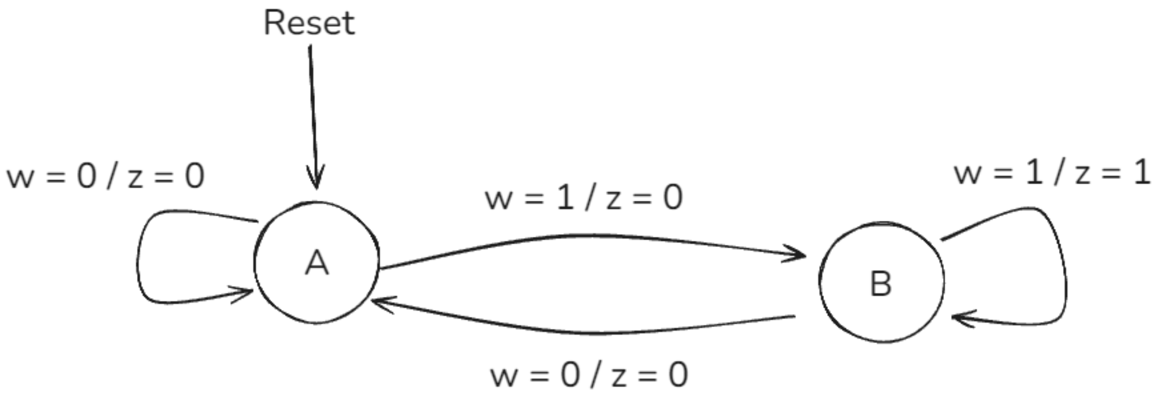


6.3 Simple Mealy Module

- Write the behavioral Verilog code for a Mealy-type FSM that reads binary sequence w and sets z=1 if 11 pattern (with overlaps) is detected.

- Draw state diagram.
- Simulate using Vivado ML to verify the operation. Write Verilog testbench to simulate your
- design.
- Show screenshot of the Waveform window indicating correct operation

State Diagram:



Verilog Code:

<u>Filename: simpleMealy.v</u>

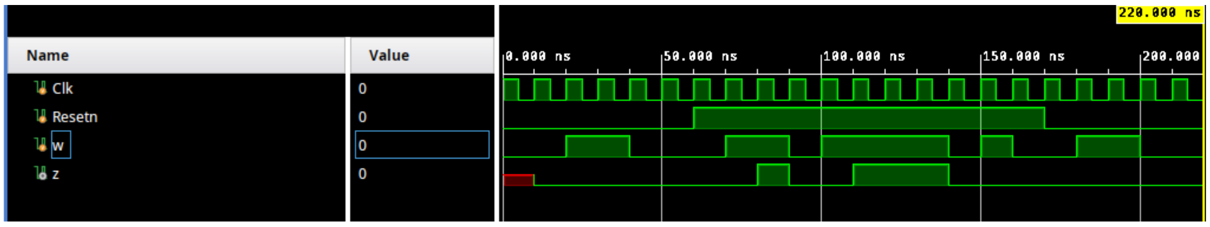
```
module simpleMealy(Clk, Resetn, w, z);

   input wire Clk, Resetn, w;
   output reg z;

   reg y, Y;
   parameter A=1'b0, B=1'b1;

   // Determine next state and output z based on input w.
   always @(w, y) begin
      case (y)
         A: if (w) begin
            Y = B;
            z = 0;
         end
         else begin
            Y = A;
            z = 0;
         end
         B: if (w) begin
            Y = B;
            z = 1;
         end
         else begin
            Y = A;
            z = 0;
         end
```

```
        endcase
    end

    // Handle clock updates and reset, reset is async and happens when it is set to 0.
    always @(negedge Resetn, posedge Clk) begin
        #10;
        if (Resetn == 0) y <= A;
        else y <= Y;
    end

endmodule
```

Testbench Verilog Code:
Filename: simpleMealy_tb.v

```
module simpleMealy_tb();
    reg Clk = 0;
    reg Resetn = 0;
    reg w;
    wire z;

    simpleMealy UT(Clk, Resetn, w, z);

    always begin
        Clk = ~Clk;
        #5;
    end

    initial begin
        w = 0; #10;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        w = 0; #10;
        w = 0; #10;
        Resetn = 1;;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        w = 1; #10;
        w = 1; #10;
        w = 0; #10;
        w = 1; #10;
        w = 0; #10;
        Resetn = 0;
        w = 0; #10;
        w = 1; #10;
        w = 1; #10;
        w = 0; #10;
        w = 0; #10;
        $finish;
    end

endmodule
```

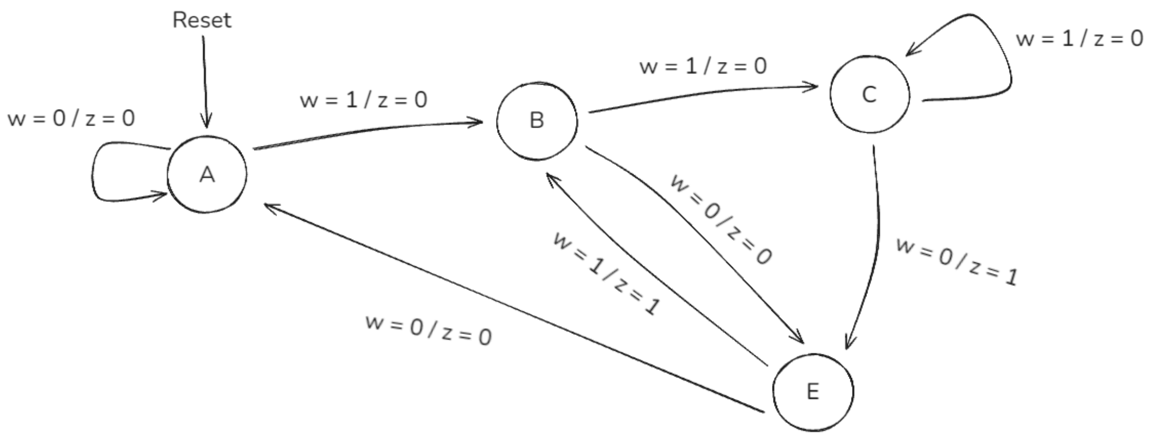
Waveform:



6.4 Mealy-Type FSM

- Write the behavioral Verilog code for a Mealy-type FSM that reads binary sequence w and sets z=1 if either a 110 or 101 pattern (with overlaps) is detected.
- Draw state diagram.
- Simulate using Vivado ML to verify the operation. Write Verilog testbench to simulate your design.
- Show screenshot of the Waveform window indicating correct operation.

State Diagram:



- Note state E should be state D

Verilog Code:

Filename: complexMealy.v

```
module complexMealy(Clk, Resetn, w, z);

    input wire Clk, Resetn, w;
    output reg z;
```

```
reg [1:0] y, Y;
parameter A=2'b00, B=2'b01, C=2'b10, D=2'b11;

// Determine next state and output z based on input w.
always @(w, y) begin
   case (y)
      A: if (w) begin
         Y = B;
         z = 0;
      end
      else begin
         Y = A;
         z = 0;
      end
      B: if (w) begin
         Y = C;
         z = 0;
      end
      else begin
         Y = D;
         z = 0;
      end
      C: if (w) begin
         Y = C;
         z = 0;
      end
      else begin
         Y = D;
         z = 1;
      end
      D: if (w) begin
         Y = B;
         z = 1;
      end
      else begin
         Y = A;
         z = 0;
      end
   endcase
end

// Handle clock updates and reset, reset is async and happens when it is set to 0.
always @(negedge Resetn, posedge Clk) begin
   #10;
   if (Resetn == 0) y <= A;
   else y <= Y;
end
```

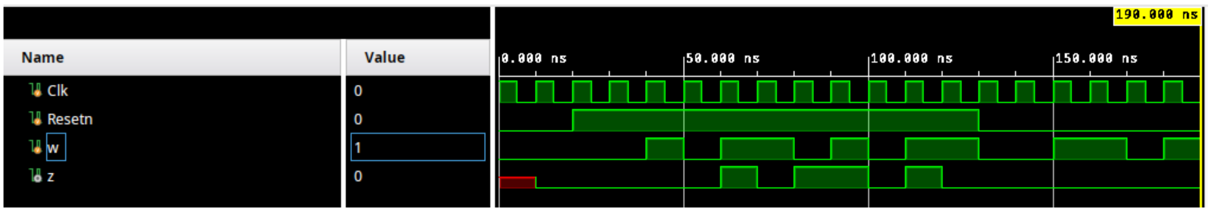
```
endmodule
```

Testbench Verilog Code:

Filename: complexMealy_tb.v

```
module complexMealy_tb();

   reg Clk = 0;
   reg Resetn = 0;
   reg w;
   wire z;

   complexMealy UT(Clk, Resetn, w, z);

   always begin
      Clk = ~Clk;
      #5;
   end

   initial begin
      w = 0; #10;
      w = 0; #10;
      Resetn = 1;
      w = 0; #10;
      w = 0; #10;
      w = 1; #10;
      w = 0; #10;
      w = 1; #10;
      w = 1; #10;
      w = 0; #10;
      w = 1; #10;
      w = 0; #10;
      w = 1; #10;
      w = 1; #10;
      Resetn = 0;
      w = 0; #10;
      w = 0; #10;
      w = 1; #10;
      w = 1; #10;
      w = 0; #10;
      w = 1; #10;
      $finish;
   end

endmodule
```

Waveform:


Name
Value
Clk
0
Resetn
0
w
1
z
0
190.000 ns
0.000 ns
50.000 ns
100.000 ns
150.000 ns