30 Days of RTL Coding

-By T Tushar Shenoy

Day 20

Problem Statement: Implementing 101 Sequence Detector Mealy machine.

Theory:

A sequence detector is a sequential state machine that takes an input string of bits and generates an output 1 whenever the target sequence has been detected. In a Mealy machine, output depends on the present state and the external input (x). Hence, in the diagram, the output is written outside the states, along with inputs. Sequence detector is of two types:

1. Overlapping
2. Non-Overlapping

In an overlapping sequence detector, the last bit of one sequence becomes the first bit of the next sequence. However, in a non-overlapping sequence detector, the last bit of one sequence does not become the first bit of the next sequence.

Examples:

For non overlapping case

Input :0110101011001

Output:0000100010000

For overlapping case

Input :0110101011001

Output:0000101010000

Mealey Machine 101 Sequence Detector non Overlapping

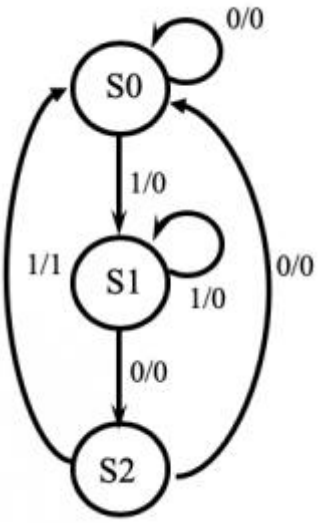


FIG: State Diagram of Mealey Machine 101 Sequence Detector non Overlapping

Verilog Code:

```
//Verilog Code for 101 Sequence Detector (Mealey Machine non Overlapping)
module fsm_detector_mealey(
    input in,
    input clk,
    input reset,
    output reg out
    );
 // Registers to hold the current and next states
  reg [1:0]currentstate;
  reg [1:0]nextstate;

 //parameters representing different states
    parameter s0=2'b00;
```

```
  parameter s1=2'b01;
  parameter s2=2'b10;


always@(posedge clk)
  begin
    if(reset)
     begin
      out=1'b0;
      currentstate=1'b0;
      nextstate=1'b0;
     end
     else
      begin
      currentstate=nextstate;

      case(currentstate)

      s0: if(in) // Check if input is true
          begin
           out=1'b0; // Set output to 0
           nextstate=s1; // Transition to state s1
          end
          else
          begin
           out=1'b0; // Set output to 0
           nextstate=s0; // Stay in state s0
          end
```

```
        s1: if(in) // Check if input is true
            begin
             out=1'b0; // Set output to 0
             nextstate=s1; // Stay in state s1
            end
            else
            begin
             out=1'b0; // Set output to 0
             nextstate=s2; // Transition to state s2
            end
        s2: if(in) // Check if input is true
            begin
             out=1'b1; // Set output to 1
             nextstate=s0; // Transition to state s0
            end
            else
            begin
             out=1'b0; // Set output to 0
             nextstate=s0; // Transition to state s0
            end
        endcase
    end
end
endmodule
```

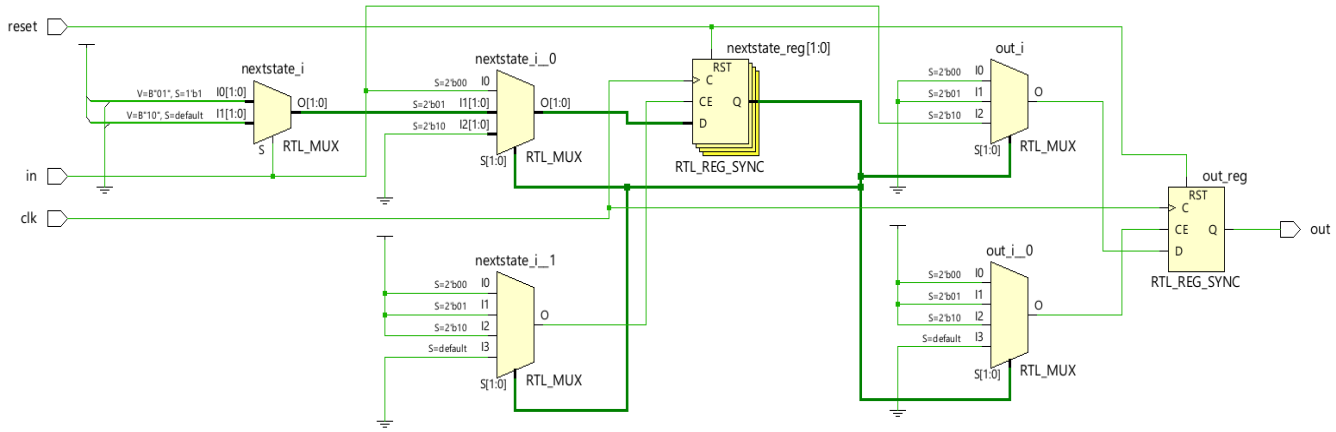
Testbench Code:

```
//Testbench Code for 101 Sequence Detector (Mealey Machine non Overlapping)

module fsm_detector_mealey_tb();

reg in;
reg clk;
reg reset;

wire out;

fsm_detector_mealey dut(.in(in),.clk(clk),.reset(reset),.out(out));

  initial begin
    in=1'b0;  // Initialize input
    clk=1'b0; // Initialize clock
    reset=1'b0; // Initialize reset

    #10;
    reset=1'b1; // Assert reset
    #10;
    reset=1'b0; // Deassert reset

    // Giving the inputs
    # 10 in=1'b0;
    # 10 in=1'b1;
    # 10 in=1'b0;
```

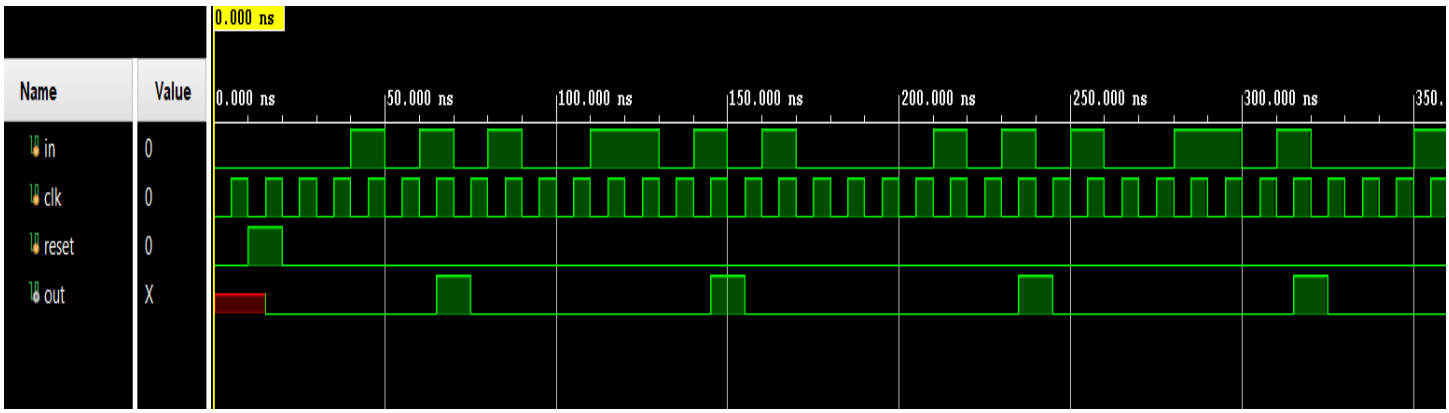
```
# 10 in=1'b1;
# 10 in=1'b0;
# 10 in=1'b1;
# 10 in=1'b0;
# 10 in=1'b0;
# 10 in=1'b1;
# 10 in=1'b1;
# 10 in=1'b0;
# 10 in=1'b1;
# 10 in=1'b0;
# 10 in=1'b1;
# 10 in=1'b0;
# 10 in=1'b0;
# 10 in=1'b0;
# 10 in=1'b0;
# 10 in=1'b1;
# 10 in=1'b0;
# 10 in=1'b1;
# 10 in=1'b0;
# 10 in=1'b1;
# 10 in=1'b0;
# 10 in=1'b0;
# 10 in=1'b1;
# 10 in=1'b1;
# 10 in=1'b0;
# 10 in=1'b1;
# 10 in=1'b0;
# 10 in=1'b0;
```

```
    # 10 in=1'b0;
    # 10 in=1'b1;
    # 10 in=1'b0;

      $finish;
end

always #5 clk=~clk; // Toggle clock
endmodule
```

Schematic:



Simulation Output:



Mealey Machine 101 Sequence Detector Overlapping

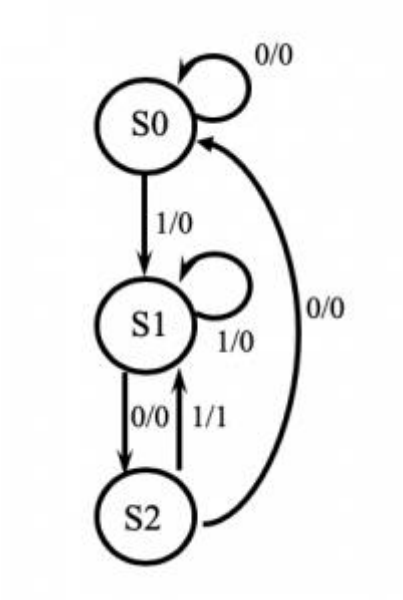


FIG: State Diagram of Mealey Machine 101 Sequence Detector Overlapping

Verilog Code:

```
//Verilog Code for 101 Sequence Detector (Mealey Machine Overlapping)
module fsm_detector_mealeyo(
    input in,
    input clk,
    input reset,
    output reg out
    );
 // Registers to hold the current and next states
  reg [1:0]currentstate;
  reg [1:0]nextstate;

 //parameters representing different states
    parameter s0=2'b00;
    parameter s1=2'b01;
```

```
   parameter s2=2'b10;


always@(posedge clk)
   begin
     if(reset)
      begin
       out=1'b0;
       currentstate=1'b0;
       nextstate=1'b0;
      end
      else
       begin
       currentstate=nextstate;

       case(currentstate)

       s0: if(in) // Check if input is true
           begin
            out=1'b0; // Set output to 0
            nextstate=s1; // Transition to state s1
           end
           else
           begin
            out=1'b0; // Set output to 0
            nextstate=s0; // Stay in state s0
           end
```

```
        s1: if(in) // Check if input is true
            begin
             out=1'b0; // Set output to 0
             nextstate=s1; // Stay in state s1
            end
            else
            begin
             out=1'b0; // Set output to 0
             nextstate=s2; // Transition to state s2
            end
        s2: if(in) // Check if input is true
            begin
             out=1'b1; // Set output to 1
             nextstate=s1; // Transition to state s1
            end
            else
            begin
             out=1'b0; // Set output to 0
             nextstate=s0; // Transition to state s0
            end
        endcase
    end
end
endmodule
```

Testbench Code:

```
//Testbench Code for 101 Sequence Detector (Mealey Machine Overlapping)

module fsm_detector_mealey_tb();

reg in;
reg clk;
reg reset;

wire out;

fsm_detector_mealeyo dut(.in(in),.clk(clk),.reset(reset),.out(out));

  initial begin
    in=1'b0;  // Initialize input
    clk=1'b0; // Initialize clock
    reset=1'b0; // Initialize reset

    #10;
    reset=1'b1; // Assert reset
    #10;
    reset=1'b0; // Deassert reset

    // Giving the inputs
    # 10 in=1'b0;
    # 10 in=1'b1;
    # 10 in=1'b0;
    # 10 in=1'b1;
```

10 in=1'b0;

10 in=1'b1;

10 in=1'b0;

10 in=1'b0;

10 in=1'b1;

10 in=1'b1;

10 in=1'b0;

10 in=1'b1;

10 in=1'b0;

10 in=1'b1;

10 in=1'b0;

10 in=1'b0;

10 in=1'b0;

10 in=1'b0;

10 in=1'b0;

10 in=1'b1;

10 in=1'b0;

10 in=1'b1;

10 in=1'b0;

10 in=1'b0;

10 in=1'b1;

10 in=1'b1;

10 in=1'b0;

10 in=1'b1;

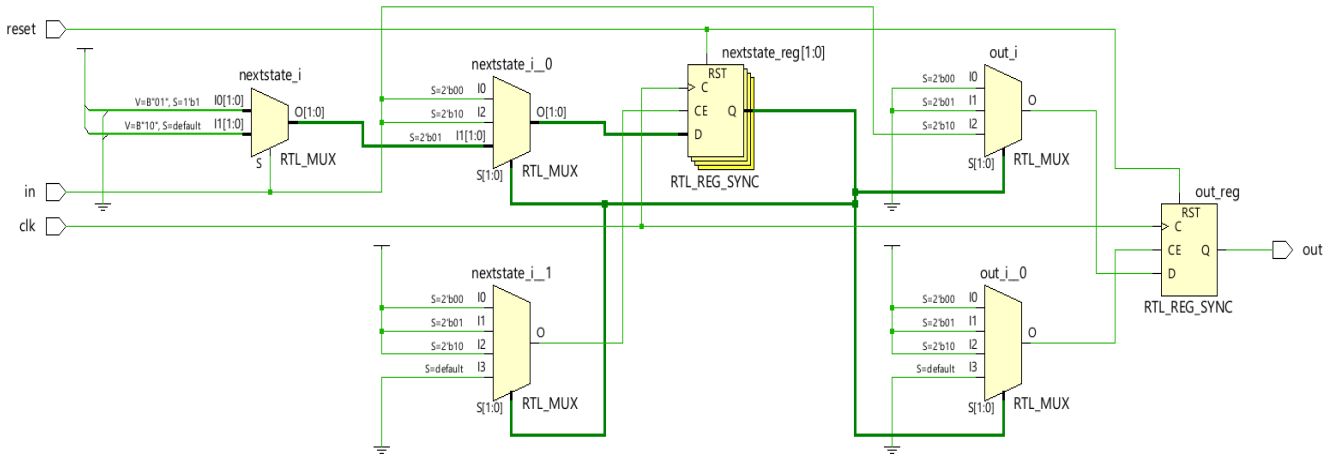
10 in=1'b0;

10 in=1'b0;

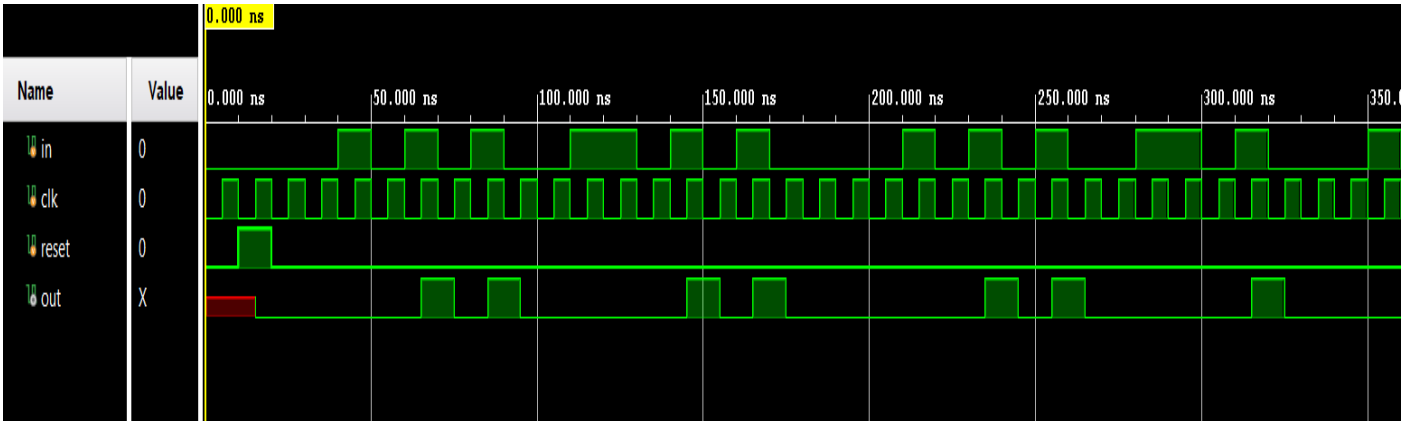
10 in=1'b0;

```
    # 10 in=1'b1;
    # 10 in=1'b0;

      $finish;
end

always #5 clk=~clk; // Toggle clock
endmodule
```

Schematic:



Simulation Output:



GitHub Repository URL:https://github.com/tusharshenoy/RTL-Day-20-Mealey-Machine