A Project Report

on

Sequence Detector

Melay State Machine
Sequence -1010

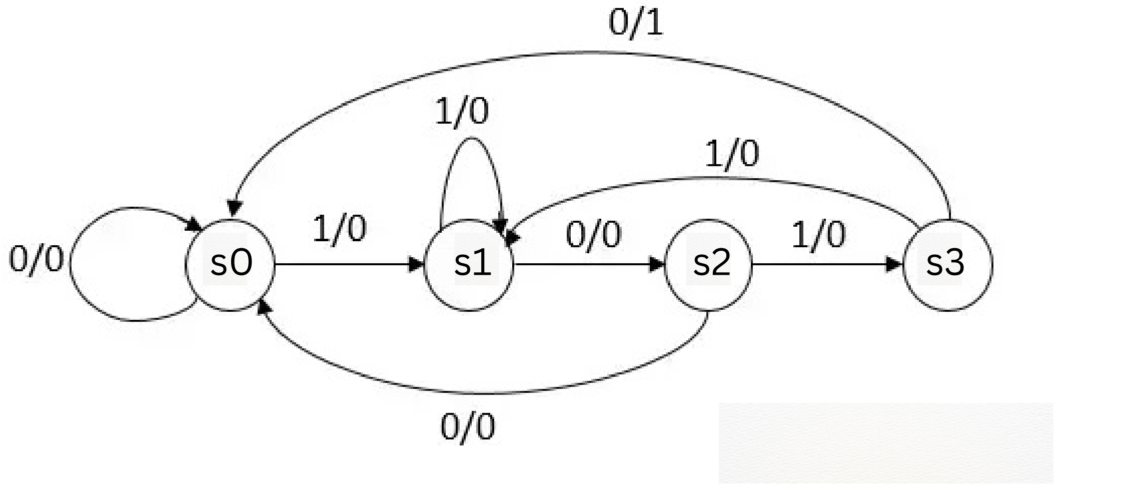
Created By -

Anirban Nath
anirbannath3@gmail.com

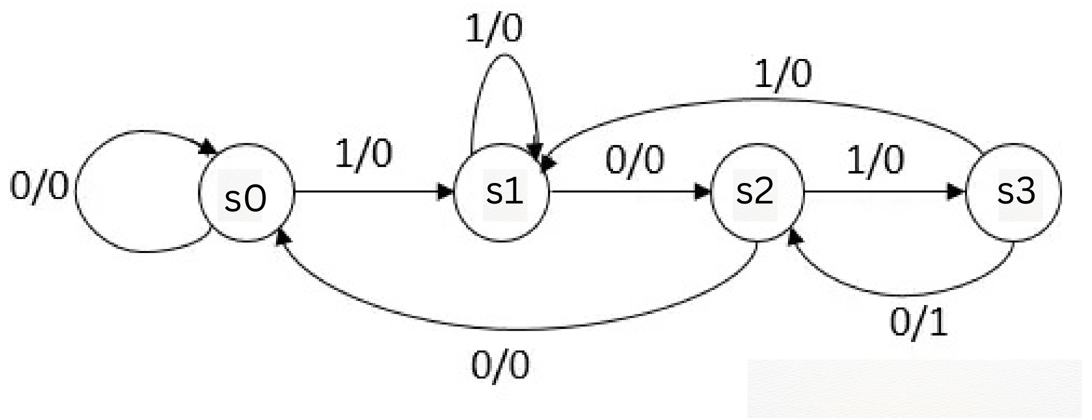
Melay State Machine

Sequence to be detected 1010

Design



1010 Non-Overlapping Mealy Sequence Detector



1010 Overlapping Mealy Sequence Detector

RTL Code (non-overlapping)

```
module melay_detector(input clk, rst, btn,output reg out);

parameter s0=3'b000;
parameter s1=3'b001;
parameter s2=3'b010;
parameter s3=3'b011;
parameter s4=3'b100;

reg [2:0] state, n_state;

// State transition logic
always @(posedge clk) begin
   if (rst)
      state <= s0;        // Use non-blocking assignment
 end

// Next state logic
always @(posedge clk) begin
   n_state <= state;
   if (state == s0) begin
   if (btn)
      state <= s1;
   else
      state <= s0;
end
```

```
else if (state == s1) begin
    if (btn)
        state <= s1;
    else
        state <= s2;
end
else if (state == s2) begin
    if (btn)
        state <= s3;
    else
        state <= s0;
end
else if (state == s3) begin
    if (btn)
        state <= s1;
    else
        state <= s0;
end
end

// Output logic
always@(posedge clk )
   if(rst)
    out<='bx;
   else begin
     if((state==s3) && (~btn))
       out<=1'b1;
     else
      out<=1'b0;
  end

endmodule
```

RTL Code (Overlapping)

```
module melay_detector(input clk, rst, btn,output
reg out);

parameter s0=3'b000;
parameter s1=3'b001;
parameter s2=3'b010;
parameter s3=3'b011;
parameter s4=3'b100;

reg [2:0] state, n_state;

// State transition logic
always @(posedge clk) begin
   if (rst)
      state <= s0;        // Use non-blocking assignment
 end

/ Next state logic
always @(posedge clk) begin
   n_state <= state;
   if (state == s0) begin
   if (btn)
      state <= s1;
   else
      state <= s0;
end
```

```
else if (state == s1) begin
    if (btn)
        state <= s1;
    else
        state <= s2;
end
else if (state == s2) begin
    if (btn)
        state <= s3;
    else
        state <= s0;
end
else if (state == s3) begin
    if (btn)
        state <= s1;
    else
        state <= s2;
end

end

// Output logic
always@(posedge clk )
   if(rst)
    out<='bx;
   else begin
     if((state==s3) && (~btn))
       out<=1'b1;
     else
      out<=1'b0;
  end

endmodule
```

Testbench

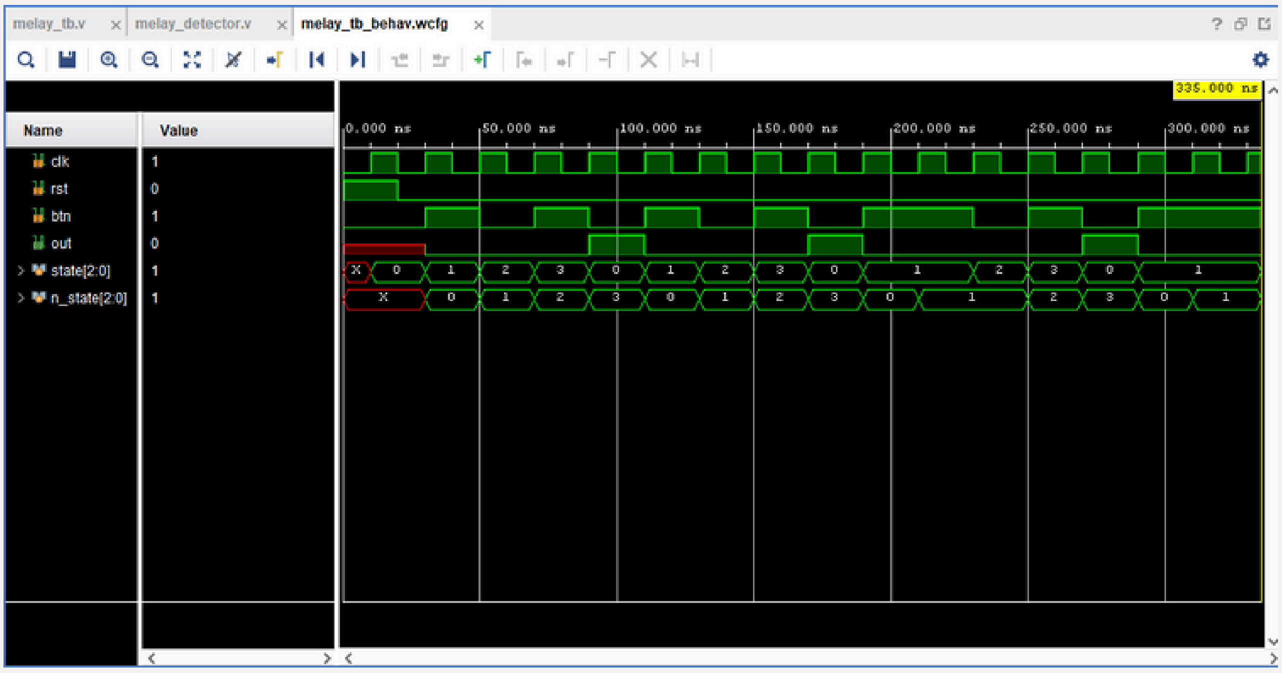
(Same for Overlapping & Non-Overlapping)

```
module melay_tb();
   reg clk, rst, btn;
   wire out;

   // Instantiate the sequence detector module
   melay_detector uut (
      .clk(clk),
      .rst(rst),
      .btn(btn),
      .out(out)
   );

   // Generate clock signal with a period of 10 time units
   always
      #10 clk = ~clk

// Initial block to initialize signals and apply stimulus
   initial begin
      clk = 0;
      rst = 1;     // Start with reset asserted (high)
      btn = 0;

      // Deassert reset after a few clock cycles
      #20 rst = 0;    // Deassert reset to start normal operation
```

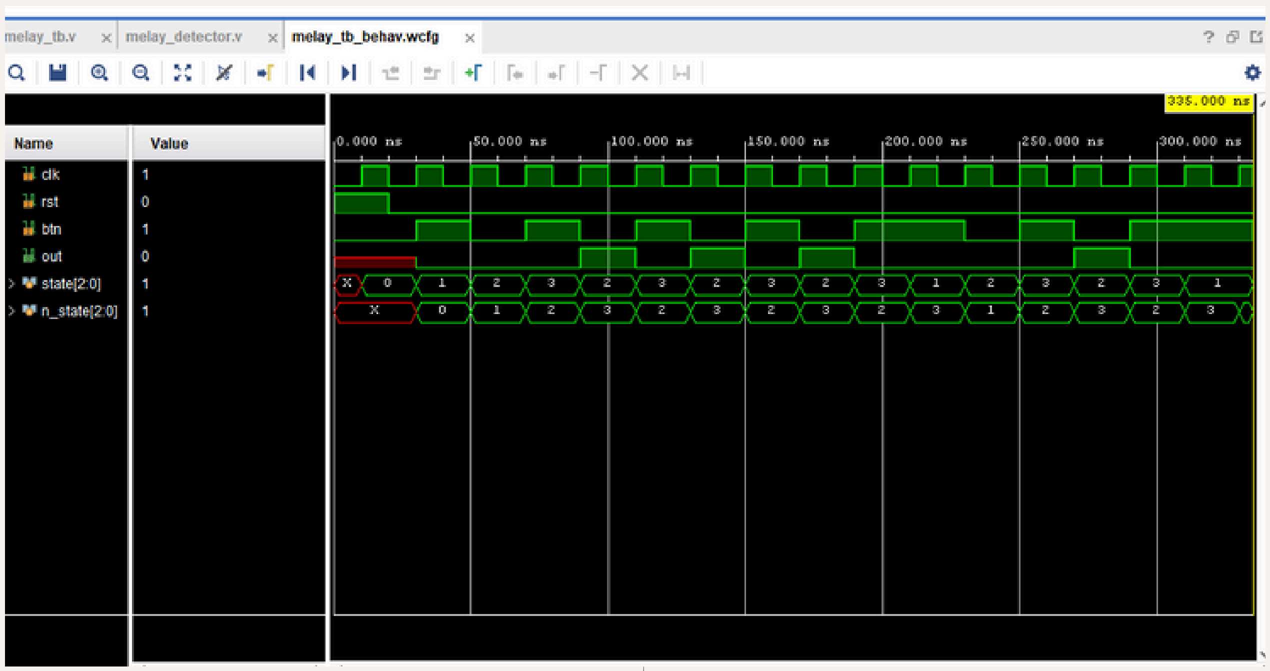
```
// Sequence to test the 1010 pattern
        #10 btn = 1;
        #20 btn = 0;
        #20 btn = 1;
        #20 btn = 0;
        #20 btn = 1;
        #20 btn = 0;
        #20 btn = 1;
        #20 btn = 0;
        #20 btn = 1;
        #20 btn = 1;
        #20 btn = 0;
        #20 btn = 1;
        #20 btn = 0;
        #20 btn = 1;

        // Finish the simulation after a delay
        #45 $finish;
    end

endmodule
```

Waveforms

Non-Overlapping



Overlapping



Thank You