IMPLEMENTATION OF SEQUENCE
DETECTORS BY FINITE STATE MACHINES
USING VERILOG

FSM SEQUENCE DETECTORS

Table of Contents

| FSM DESIGN STYLES | 1 |
|--|----|
| SINGLE PROCEDURAL BLOCK | 1 |
| TWO PROCEDURAL BLOCK(METHOD-01) | 6 |
| TWO PROCEDURAL BLOCK(METHOD-02) | 11 |
| THREE PROCEDURAL BLOCK | 16 |
| | |
| 1010 SEQUENCE DETECTOR NON OVERLAPPING USING MEALY MACHINE | 21 |
| 1010 SEQUENCE DETECTOR OVERLAPPING USING MEALY MACHINE | 26 |
| 1010 SEQUENCE DETECTOR NON OVERLAPPING USING MOORE MACHINE | 31 |
| 1010 SEQUENCE DETECTOR OVERLAPPING USING MOORE MACHINE | 36 |
| 101X SEQUENCE DETECTOR OVERLAPPING USING MEALY MACHINE | 41 |
| 10X0 SEQUENCE DETECTOR OVERLAPPING USING MEALY MACHINE | 46 |

FSM SEQUENCE DETECTOR

SINGLE PROCEDURAL BLOCK

DESIGN MODULE

```
`timescale 1ns / 1ps
module mealy 1010 seq det non over(Clk,Rst,In,OP,state);
input Clk,Rst,In; // Input signals
output OP;  // Output signal
output [1:0]state;
reg op;  // register to store output
reg [1:0]State; // State register
parameter s0 = 0; // State registers
parameter s1 = 1;
parameter s10 = 2;
parameter s101 = 3;
always @(posedge Clk,negedge Rst) begin
     if(!Rst)
    begin
         State <= s0;
         op <= 0;
     end
    else begin
          case (State)
               s0 : begin
                         State <= In ? s1:s0 ;
                        op <= 0;
                     end
               s1 : begin
```

```
State <= In ? s1:s10 ;
                         op <= 0;
                    end
               s10 : begin
                         State <= In ? s101:s0 ;
                         op <= 0;
                    end
               s101: begin
                         State <= In ? s1:s0 ;
                         op <= In ? 0:1 ;
                    end
               default : begin
                            State <= s0 ;
                            op <= 0;
                            end
          endcase
     end
end
assign OP = op;
assign state = State;
endmodule
```

FSM SEQUENCE DETECTORS

TESTBENCH MODULE

```
`timescale 1ns / 1ps
module mealy 1010 seq det non over tb ();
// Signal Declarations
reg Clk,Rst,In;
wire OP;
wire [1:0]state;
//instantiating design to test
mealy 1010 seq det non over UUT(Clk,Rst,In,OP,state);
// Monitoring the simulation results
initial begin
     $monitor("Input = %b || State = %b || Output = %b", In, state, OP);
end
// clock signal generation
initial begin
     Clk = 1'b0;
     forever #5 Clk = ~Clk;
end
// input signals
initial begin
         Rst = 1'b0;
     #15 Rst = 1'b1; end
initial begin
         In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b0;
```

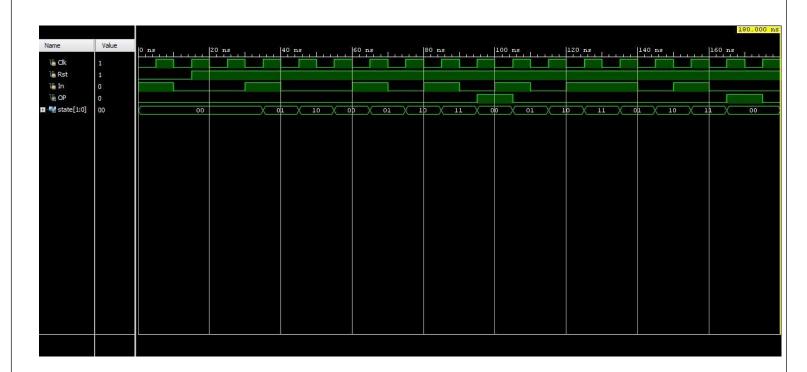
```
FSM
                                                             SEQUENCE DETECTORS
    #10 In = 1'b1;
    #10 In
            = 1'b0;
    #10 In
            = 1'b1;
    #10 In
            = 1'b0;
    #10 In
            = 1'b1;
    #10 In
            = 1'b0;
    #10 In = 1'b1;
    #10 In = 1'b1;
    #10 In = 1'b0;
    #10 In
            = 1'b1;
    #10 In = 1'b0;
    #10 In = 1'b0;
```

end

endmodule

SIMULATION WAVEFORMS

#10 \$finish();



FSM SEQUENCE DETECTORS

SIMULATION OUTPUTS

```
Input = 1 || State = 00 || Output = 0
Input = 0 || State = 00 || Output = 0
Input = 1 || State = 00 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 0 || State = 00 || Output = 0
Input = 1 || State = 00 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 1 || State = 10 || Output = 0
Input = 1 || State = 11 || Output = 0
Input = 0 || State = 11 || Output = 0
Input = 0 || State = 00 || Output = 1
Input = 1 || State = 00 || Output = 1
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 1 || State = 10 || Output = 0
Input = 1 || State = 11 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 1 || State = 10 || Output = 0
Input = 1 || State = 11 || Output = 0
Input = 0 || State = 11 || Output = 0
Input = 0 || State = 00 || Output = 1
Input = 0 || State = 00 || Output = 0
```

TWO PROCEDURAL BLOCK(METHOD-01)

DESIGN MODULE

```
`timescale 1ns / 1ps
module mealy_1010_seq_det_non_over(Clk,Rst,In,OP);
input Clk,Rst,In; // Input signals
output OP;  // Output signal
reg op; // register to store output
reg [1:0]C State, N State; // state register
parameter s0 = 0, // state registers
           s1 = 1,
            s10 = 2,
            s101 = 3;
always @(posedge Clk,negedge Rst) begin
     if(!Rst)
    begin
         C State <= s0;
         op <= 0;
     end
    else begin
       C State <= N State;
     end
end
always @(C State, In) begin
          case(C State)
```

FSM SEQUENCE DETECTORS

```
s0 :
                          begin N State <= In ? s1:s0 ; op <= 0 ; end</pre>
                           begin N State <= In ? s1:s10 ; op <= 0 ; end</pre>
                s1
                s10 :
                           begin N State <= In ? s101:s0 ; op <= 0 ; end</pre>
               s101 :
                          begin N State <= In ? s1:s0 ; op <= In ? 0:1 ;</pre>
end
               default : begin N State <= s0; op <= 0; end</pre>
          endcase
     end
assign OP = op;
endmodule
```

TESTBENCH MODULE

```
"timescale 1ns / 1ps

module mealy_1010_seq_det_non_over_tb ();

// Signal Declarations

reg Clk,Rst,In;

wire OP;

//instantiating design to test

mealy_1010_seq_det_non_over UUT(Clk,Rst,In,OP);

// Monitoring the simulation results

initial begin
```

```
FSM
                                                             SEQUENCE DETECTORS
     $monitor("CS = %d ||Input = %b || NS = %b || Output
= %b",UUT.C State,In,UUT.N State,OP);
end
// clock signal generation
initial begin
     Clk = 1'b0;
     forever #5 Clk = ~Clk;
end
// input signals
initial begin
         Rst = 1'b0;
     #15 Rst = 1'b1;
end
initial begin
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #7 In = 1'b0;
     #13 In = 1'b1;
     #8
         In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b1;
     #10 In = 1'b0;
```

PABBULETI VENU 8

#10 In = 1'b1;

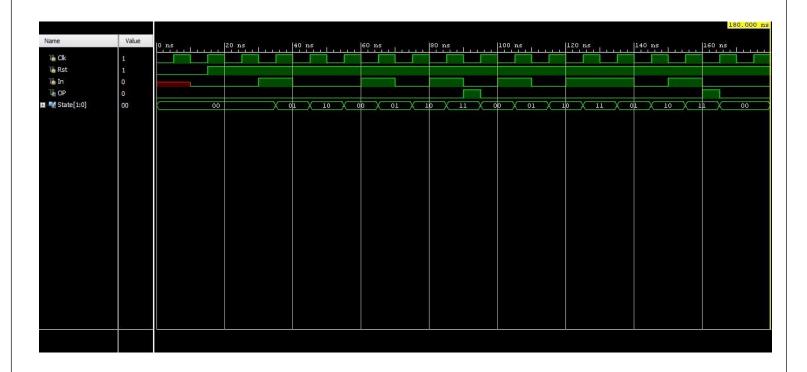
#10 In = 1'b0;

SEQUENCE DETECTORS

```
#10 In = 1'b0;
#10 $finish();
end
```

endmodule

SIMULATION WAVEFORMS



FSM

SIMULATION OUTPUTS

```
Input = x || State = 00 || Output = 0
Input = 0 || State = 00 || Output = 0
Input = 1 || State = 00 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 0 || State = 00 || Output = 0
Input = 1 || State = 00 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 1 || State = 10 || Output = 0
Input = 1 || State = 11 || Output = 0
Input = 0 || State = 11 || Output = 1
Input = 0 || State = 00 || Output = 0
Input = 1 || State = 00 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 1 || State = 10 || Output = 0
Input = 1 || State = 11 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 1 || State = 10 || Output = 0
Input = 1 || State = 11 || Output = 0
Input = 0 || State = 11 || Output = 1
Input = 0 || State = 00 || Output = 0
```

TWO PROCEDURAL BLOCK(METHOD-02)

DESIGN MODULE

```
`timescale 1ns / 1ps
module mealy 1010 seq det non over(Clk, Rst, In, OP, sta);
input Clk,Rst,In; // Input signals
output OP;  // Output signal
output [1:0]sta;
reg op;  // register to store output
reg [1:0]state; // state register
parameter s0 = 0, // state registers
           s1 = 1,
           s10 = 2,
           s101 = 3;
reg [1:0]State;
always @(posedge Clk,negedge Rst) begin
     if(!Rst)
    begin
         State <= s0;
         op <= 0;
     end
    else begin
         case (State)
              s0 : State <= In ? s1:s0 ;
              s1 : State <= In ? s1:s10 ;
```

```
FSM
```

```
s10 : State <= In ? s101:s0 ;
             s101 : State <= In ? s1:s0 ;
             default : State <= s0;</pre>
         endcase
    end
end
always @(State,In) begin
    case (State)
         s0 : op \leq 0;
         s10 : op <= 0;
         s101: op <= In ? 0:1;
         default : op <= 0;</pre>
    endcase
end
assign OP = op;
assign sta = State;
endmodule
```

TESTBENCH MODULE

```
`timescale 1ns / 1ps
module mealy 1010 seq det non over tb ();
// Signal Declarations
reg Clk,Rst,In;
wire OP;
wire [1:0]State;
//instantiating design to test
mealy 1010 seq det non over UUT(Clk,Rst,In,OP,State);
// Monitoring the simulation results
initial begin
     $monitor("Input = %b || State = %b || Output = %b", In, State, OP);
end
// clock signal generation
initial begin
     Clk = 1'b0;
     forever #5 Clk = ~Clk;
end
// input signals
initial begin
          Rst = 1'b0;
     #15 Rst = 1'b1; end
initial begin
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
```

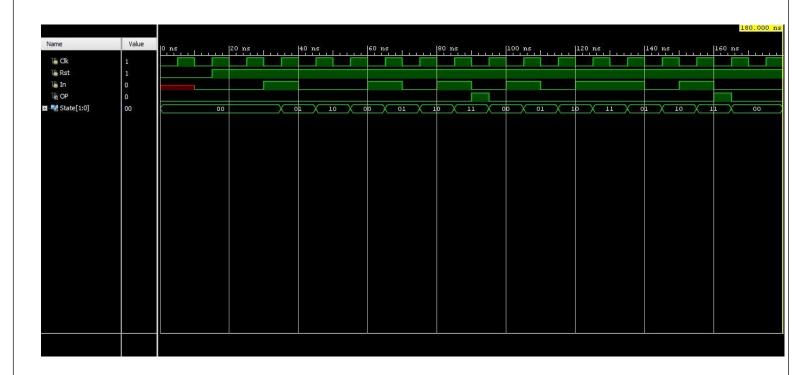
```
SEQUENCE DETECTORS
```

```
FSM
    #10 In = 1'b1;
    #10 In = 1'b0;
    #10 In
            = 1'b1;
    #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 $finish();
```

end

endmodule

SIMULATION WAVEFORMS



FSM

SIMULATION OUTPUTS

```
Input = x || State = 00 || Output = 0
Input = 0 || State = 00 || Output = 0
Input = 1 || State = 00 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 0 || State = 00 || Output = 0
Input = 1 || State = 00 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 1 || State = 10 || Output = 0
Input = 1 || State = 11 || Output = 0
Input = 0 || State = 11 || Output = 1
Input = 0 || State = 00 || Output = 0
Input = 1 || State = 00 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 1 || State = 10 || Output = 0
Input = 1 || State = 11 || Output = 0
Input = 1 || State = 01 || Output = 0
Input = 0 || State = 01 || Output = 0
Input = 0 || State = 10 || Output = 0
Input = 1 || State = 10 || Output = 0
Input = 1 || State = 11 || Output = 0
Input = 0 || State = 11 || Output = 1
Input = 0 || State = 00 || Output = 0
```

THREE PROCEDURAL BLOCK

DESIGN MODULE

```
`timescale 1ns / 1ps
module mealy 1010 seq det non over(Clk,Rst,In,OP,cs,ns);
input Clk,Rst,In; // Input signals
output OP;  // Output signal
output [1:0]cs,ns;
reg op; // register to store output
reg state; // state register
parameter s0 = 0, // state registers
            s1
               = 1,
            s10 = 2,
            s101 = 3;
reg [1:0]C_State, N_State; // state declarations
// State register
always @(posedge Clk, negedge Rst) begin
     if(!Rst)
     begin
          C State <= s0;
          op <= 0;
     end
     else begin
          C State <= N State;</pre>
     end
end
// Next state logic
```

```
always @(C State,In) begin
     case (C_State)
          s0 : N State <= In ? s1:s0 ;
          s1 : N State <= In ? s1:s10 ;
          s10 : N_State <= In ? s101:s0 ;</pre>
          s101: N State <= In ? s1:s0 ;
          default : N_State <= s0;</pre>
     endcase
end
// Output logic
always @(C_State,In) begin
     case(C State)
          s0 : op <= 0;
          s1 : op <= 0;
          s10 : op <= 0;
          s101: op <= In ? 0:1 ;
          default : op <= 0;</pre>
     endcase
end
```

```
//
assign OP = op;
assign cs = C State;
assign ns = N State;
endmodule
```

TESTBENCH MODULE

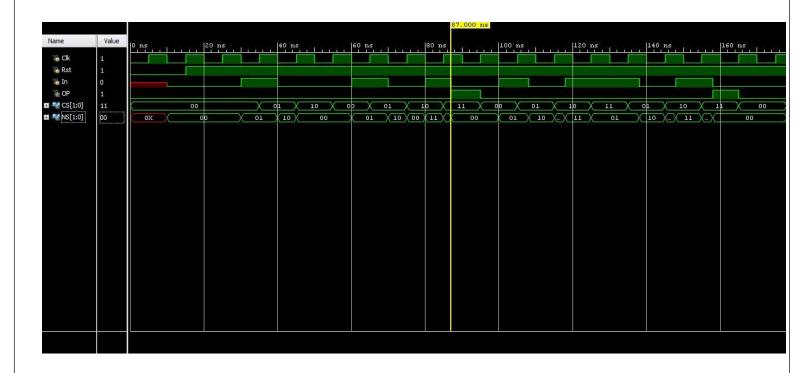
```
`timescale 1ns / 1ps
module mealy 1010 seq det non over tb ();
// Signal Declarations
reg Clk,Rst,In;
wire OP;
wire [1:0]CS,NS;
//instantiating design to test
mealy 1010 seq det non over UUT(Clk,Rst,In,OP,CS,NS);
// Monitoring the simulation results
initial begin
     $monitor("Current State = %b || Input = %b || Next State = %b ||
Output = %b",UUT.C State,In,UUT.N State,OP);
end
// clock signal generation
initial begin
     Clk = 1'b0;
     forever #5 Clk = ~Clk;
end
// input signals
initial begin
          Rst = 1'b0;
PABBULETI VENU
```

```
FSM
    #15 Rst = 1'b1; end
initial begin
    #10 In = 1'b0;
    #10 In = 1'b0;
    #10 In = 1'b1;
    #10 In = 1'b0;
    #10 In = 1'b0;
    #10 In = 1'b1;
    #10 In = 1'b0;
    #10 In = 1'b1;
    #7
        In = 1'b0;
    #13 In = 1'b1;
    #8
        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 $finish();
```

end

endmodule

SIMULATION WAVEFORMS



SIMULATION OUTPUTS

| Current | State | = | 00 | H | Input | = | х | 11 | Next | State | = | 0x | H | Output | = | 0 |
|---------|-------|---|----|----|-------|---|---|----|------|-------|---|----|----|--------|---|---|
| Current | State | = | 00 | 11 | Input | = | 0 | JJ | Next | State | = | 00 | 11 | Output | = | 0 |
| Current | State | = | 00 | 11 | Input | = | 1 | 11 | Next | State | = | 01 | 11 | Output | = | 0 |
| Current | State | = | 01 | 11 | Input | = | 1 | 11 | Next | State | = | 01 | 11 | Output | = | 0 |
| Current | State | = | 01 | 11 | Input | = | 0 | 11 | Next | State | = | 10 | 11 | Output | = | 0 |
| Current | State | = | 10 | 11 | Input | = | 0 | 11 | Next | State | = | 00 | 11 | Output | = | 0 |
| Current | State | = | 00 | 11 | Input | = | 0 | 11 | Next | State | = | 00 | 11 | Output | = | 0 |
| Current | State | = | 00 | 11 | Input | = | 1 | 11 | Next | State | = | 01 | 11 | Output | = | 0 |
| Current | State | = | 01 | П | Input | = | 1 | 11 | Next | State | = | 01 | ΪĨ | Output | = | 0 |
| Current | State | = | 01 | 11 | Input | = | 0 | IJ | Next | State | = | 10 | 11 | Output | = | 0 |
| Current | State | = | 10 | 11 | Input | = | 0 | 11 | Next | State | = | 00 | 11 | Output | = | 0 |
| Current | State | = | 10 | 11 | Input | = | 1 | 11 | Next | State | = | 11 | 11 | Output | = | 0 |
| Current | State | = | 11 | 11 | Input | = | 1 | 11 | Next | State | = | 01 | 11 | Output | = | 0 |
| Current | State | = | 11 | 11 | Input | = | 0 | 11 | Next | State | = | 00 | 11 | Output | = | 1 |
| Current | State | = | 00 | 11 | Input | = | 0 | 11 | Next | State | = | 00 | 11 | Output | = | 0 |
| Current | State | = | 00 | 11 | Input | = | 1 | 11 | Next | State | = | 01 | 11 | Output | = | 0 |
| Current | State | = | 01 | П | Input | = | 1 | 11 | Next | State | = | 01 | H | Output | = | 0 |
| Current | State | = | 01 | 11 | Input | = | 0 | Ш | Next | State | = | 10 | 11 | Output | = | 0 |
| Current | State | = | 10 | 11 | Input | = | 0 | 11 | Next | State | = | 00 | 11 | Output | = | 0 |
| Current | State | = | 10 | 11 | Input | = | 1 | 11 | Next | State | = | 11 | 11 | Output | = | 0 |
| Current | State | = | 11 | 11 | Input | = | 1 | 11 | Next | State | = | 01 | 11 | Output | = | 0 |
| Current | State | = | 01 | 11 | Input | = | 1 | 11 | Next | State | = | 01 | 11 | Output | = | 0 |
| Current | State | = | 01 | 11 | Input | = | 0 | 11 | Next | State | = | 10 | 11 | Output | = | 0 |
| Current | State | = | 10 | 11 | Input | = | 0 | 11 | Next | State | = | 00 | 11 | Output | = | 0 |
| Current | State | = | 10 | TT | Input | = | 1 | 11 | Next | State | = | 11 | II | Output | = | 0 |
| Current | State | = | 11 | П | Input | = | 1 | 11 | Next | State | = | 01 | П | Output | = | 0 |
| Current | State | = | 11 | 11 | Input | = | 0 | 11 | Next | State | = | 00 | 11 | Output | = | 1 |
| Current | State | = | 00 | 11 | Input | = | 0 | 11 | Next | State | = | 00 | 11 | Output | = | 0 |
| | | | | | | | | | | | | | | | | |

SEQUENCE DETECTOR 1010 USING MEALY MACHINE (NON OVERLAPPING)

DESIGN MODULE

```
`timescale 1ns / 1ps
module mealy 1010 seq det non over(Clk,Rst,In,OP,cs,ns);
input Clk,Rst,In; // Input signals
output OP;
                  // Output signal
output [1:0]cs,ns; // To represent the present and next states
parameter s0 = 0, // state registers
            s1 = 1,
            s10 = 2,
            s101 = 3;
reg [1:0]C State, N State; // state declarations
// State register
always @(posedge Clk, negedge Rst) begin
     if(!Rst)
     begin
          C State <= s0;
     end
     else begin
          C State <= N State;</pre>
     end
end
// Next state logic
```

```
FSM
```

```
always @(C_State,In) begin
     case (C_State)
           s0 : N State <= In ? s1:s0 ;
           s1 : N_State <= In ? s1:s10 ;
           s10 : N State <= In ? s101:s0 ;</pre>
          s101: N_State <= In ? s1:s0 ;</pre>
          default : N_State <= s0;</pre>
     endcase
end
// Output logic
assign OP = ((C_State == s101 ) && (!In) );
assign cs = C_State;
assign ns = N_State;
endmodule
```

TESTBENCH MODULE

```
`timescale 1ns / 1ps
module mealy 1010 seq det non_over_tb ();
// Signal Declarations
reg Clk,Rst,In;
wire OP;
wire [1:0]CS,NS;
//instantiating design to test
mealy 1010 seq det non over UUT(Clk,Rst,In,OP,CS,NS);
// Monitoring the simulation results
initial begin
     $monitor("Current State = %b || Input = %b || Next State = %b ||
Output = %b",UUT.C State,In,UUT.N State,OP);
end
// clock signal generation
initial begin
     Clk = 1'b0;
     forever #5 Clk = ~Clk;
end
// input signals
initial begin
          Rst = 1'b0;
     #15 Rst = 1'b1; end
initial begin
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
```

```
FSM #10 In = 1'b1; #7 In = 1'b0;
```

```
#7 In = 1'b0;

#13 In = 1'b1;

#8 In = 1'b0;

#10 In = 1'b1;

#10 In = 1'b1;

#10 In = 1'b0;

#10 In = 1'b0;

#10 In = 1'b0;

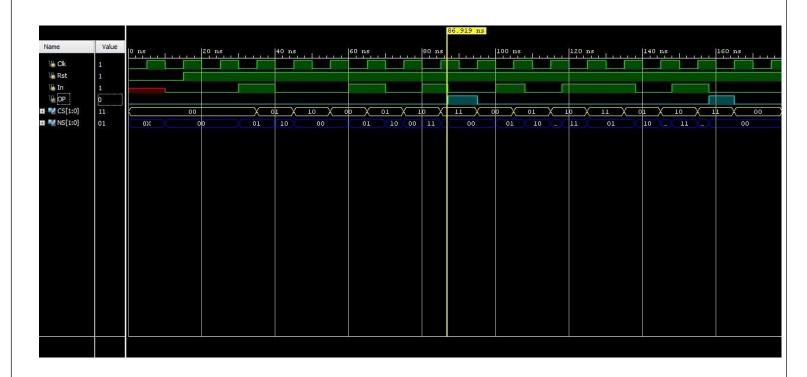
#10 In = 1'b0;

#10 In = 1'b0;
```

end

endmodule

SIMULATION WAVEFORMS



SIMULATION OUTPUTS

```
Current State = 00 || Input = x || Next State = 0x || Output = 0
Current State = 00 || Input = 0 || Next State = 00 || Output = 0
Current State = 00 || Input = 1 || Next State = 01 || Output = 0
Current State = 01 || Input = 1 || Next State = 01 || Output = 0
Current State = 01 || Input = 0 || Next State = 10 || Output = 0
Current State = 10 || Input = 0 || Next State = 00 || Output = 0
Current State = 00 || Input = 0 || Next State = 00 || Output = 0
Current State = 00 || Input = 1 || Next State = 01 || Output = 0
Current State = 01 || Input = 1 || Next State = 01 || Output = 0
Current State = 01 || Input = 0 || Next State = 10 || Output = 0
Current State = 10 || Input = 0 || Next State = 00 || Output = 0
Current State = 10 || Input = 1 || Next State = 11 || Output = 0
Current State = 11 || Input = 1 || Next State = 01 || Output = 0
Current State = 11 || Input = 0 || Next State = 00 || Output = 1
Current State = 00 || Input = 0 || Next State = 00 || Output = 0
Current State = 00 || Input = 1 || Next State = 01 || Output = 0
Current State = 01 || Input = 1 || Next State = 01 || Output = 0
Current State = 01 || Input = 0 || Next State = 10 || Output = 0
Current State = 10 || Input = 0 || Next State = 00 || Output = 0
Current State = 10 || Input = 1 || Next State = 11 || Output = 0
Current State = 11 || Input = 1 || Next State = 01 || Output = 0
Current State = 01 || Input = 1 || Next State = 01 || Output = 0
Current State = 01 || Input = 0 || Next State = 10 || Output = 0
Current State = 10 || Input = 0 || Next State = 00 || Output = 0
Current State = 10 || Input = 1 || Next State = 11 || Output = 0
Current State = 11 || Input = 1 || Next State = 01 || Output = 0
Current State = 11 || Input = 0 || Next State = 00 || Output = 1
Current State = 00 || Input = 0 || Next State = 00 || Output = 0
```

SEQUENCE DETECTOR 1010 USING MEALYMACHINE (OVERLAPPING)

DESIGN MODULE

```
`timescale 1ns / 1ps
module mealy 1010 seq det over(Clk,Rst,In,OP,CS,NS);
input Clk,Rst,In; // Input signals
output OP;  // Output signal
output [1:0]CS,NS;
parameter s0 = 0, // state registers
            s1 = 1,
            s10 = 2,
            s101 = 3;
reg [1:0]C State, N State; // state declarations
// State register
always @(posedge Clk, negedge Rst) begin
     if(!Rst)
     begin
          C State <= s0;
     end
     else begin
          C State <= N State;</pre>
     end
end
```

```
FSM
// Next state logic
always @(C_State,In) begin
     case(C State)
           s0 : N_State <= In ? s1:s0 ;
           s1 : N State <= In ? s1:s10 ;
           s10 : N_State <= In ? s101:s0 ;</pre>
           s101: N_State \leftarrow In ? s1:s10 ;
          default : N State <= s0;</pre>
     endcase
end
// Output logic
assign OP = ((C State == s101) && (!In));
//states
assign CS = C State ;
assign NS = N State ;
endmodule
```

TESTBENCH MODULE

```
`timescale 1ns / 1ps
module mealy 1010 seq det over tb ();
// Signal Declarations
reg Clk,Rst,In;
wire OP;
wire CS, NS;
//instantiating design to test
mealy 1010 seq det over UUT(Clk,Rst,In,OP,CS,NS);
// Monitoring the simulation results
initial begin
     $monitor("Time = %0t ||Current State = %b || Input = %b || Next State
= %b || Output = %b",$time,CS,In,NS,OP);
end
// clock signal generation
initial begin
     Clk = 1'b0;
     forever #5 Clk = ~Clk;
end
// input signals
initial begin
         Rst = 1'b0;
     #15 Rst = 1'b1; end
initial begin
         In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
```

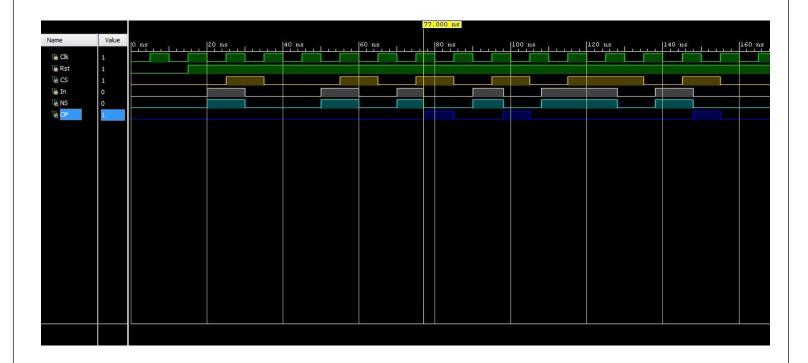
```
FSM #10 In = 1'b0;
```

```
#10 In
       = 1'b1;
#7
   Ιn
       = 1'b0;
#13 In
       = 1'b1;
#8
    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 $finish();
```

end

endmodule

SIMULATION WAVEFORMS



SIMULATION OUTPUTS

```
Time = 0 ||Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Time = 20000 ||Current State = 0 || Input = 1 || Next State = 1 || Output = 0
Time = 25000 ||Current State = 1 || Input = 1 || Next State = 1 || Output = 0
Time = 30000 ||Current State = 1 || Input = 0 || Next State = 0 || Output = 0
Time = 35000 ||Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Time = 50000 ||Current State = 0 || Input = 1 || Next State = 1 || Output = 0
Time = 55000 ||Current State = 1 || Input = 1 || Next State = 1 || Output = 0
Time = 60000 ||Current State = 1 || Input = 0 || Next State = 0 || Output = 0
Time = 65000 ||Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Time = 70000 ||Current State = 0 || Input = 1 || Next State = 1 || Output = 0
Time = 75000 ||Current State = 1 || Input = 1 || Next State = 1 || Output = 0
Time = 77000 ||Current State = 1 || Input = 0 || Next State = 0 || Output = 1
Time = 85000 ||Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Time = 90000 ||Current State = 0 || Input = 1 || Next State = 1 || Output = 0
Time = 95000 ||Current State = 1 || Input = 1 || Next State = 1 || Output = 0
Time = 98000 ||Current State = 1 || Input = 0 || Next State = 0 || Output = 1
Time = 105000 ||Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Time = 108000 | Current State = 0 | | Input = 1 | | Next State = 1 | | Output = 0
Time = 115000 | Current State = 1 | Input = 1 | Next State = 1 | Output = 0
Time = 128000 ||Current State = 1 || Input = 0 || Next State = 0 || Output = 0
Time = 135000 ||Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Time = 138000 | | Current State = 0 | | Input = 1 | | Next State = 1 | | Output = 0
Time = 145000 | | Current State = 1 | | Input = 1 | | Next State = 1 | | Output = 0
Time = 148000 | Current State = 1 | | Input = 0 | | Next State = 0 | | Output = 1
Time = 155000 | Current State = 0 | | Input = 0 | | Next State = 0 | | Output = 0
```

SEQUENCE DETECTOR 1010 USING MOORE MACHINE (NON OVERLAPPING)

DESIGN MOUDLE

```
`timescale 1ns / 1ps
module moore 1010 seq det non over(Clk,Rst,In,OP,CS,NS);
input Clk,Rst,In; // Input signals
output OP;  // Output signal
output [2:0]CS,NS; // states
          s0 = 0 , // state registers
parameter
              = 1 ,
           s1
           s10 = 2,
           s101 = 3,
           s1010 = 4;
reg [2:0]C State, N State; // state declarations
// State register
always @(posedge Clk,negedge Rst) begin
    if(!Rst)
    begin
         C State <= s0;
     end
    else begin
         C State <= N State;
     end
end
```

```
FSM
// Next state logic
always @(C_State,In) begin
     case(C State)
          s0
              : N State <= In ? s1:s0 ;
               : N State <= In ? s1:s10 ;
          s1
          s10 : N State <= In ? s101:s0 ;</pre>
          s101 : N_State <= In ? s1:s1010 ;</pre>
          s1010 : N_State <= In ? s1 : s0 ;
          default : N State <= s0;</pre>
     endcase
end
// Output logic
assign OP = (C_State == s1010) ;
//
assign CS = C State ;
assign NS = N_State ;
endmodule
```

TESTBENCH MODULE

```
`timescale 1ns / 1ps
module moore 1010 seq det non over tb ();
// Signal Declarations
reg Clk,Rst,In;
wire OP;
wire [2:0] CS,NS;
//instantiating design to test
moore 1010 seq det non over UUT(Clk,Rst,In,OP,CS,NS);
// Monitoring the simulation results
initial begin
     $monitor("Time = %0t || Current State = %b || Input = %b || Next
State = %b || Output = %b", $time, CS, In, NS, OP);
end
// clock signal generation
initial begin
     Clk = 1'b0;
     forever #5 Clk = ~Clk;
end
// input signals
initial begin
         Rst = 1'b0;
     #15 Rst = 1'b1; end
initial begin
         In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
```

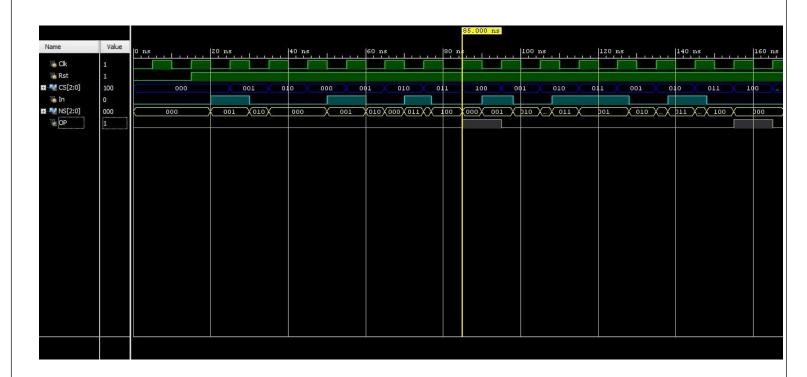
```
FSM
                                                              SEQUENCE DETECTORS
    #10 In = 1'b0;
    #10 In
             = 1'b1;
    #7
        Ιn
             = 1'b0;
    #13 In
             = 1'b1;
    #8
        In
             = 1'b0;
    #10 In
             = 1'b1;
    #10 In
             = 1'b1;
    #10 In = 1'b0;
    #10 In = 1'b1;
    #10 In
             = 1'b0;
    #10 In = 1'b0;
```

end

endmodule

SIMULATION WAVEFORMS

#10 \$finish();



FSM

SIMULATION OUTPUTS

```
Time = 0 || Current State = 000 || Input = 0 || Next State = 000 || Output = 0
Time = 20000 || Current State = 000 || Input = 1 || Next State = 001 || Output = 0
Time = 25000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
Time = 30000 || Current State = 001 || Input = 0 || Next State = 010 || Output = 0
Time = 35000 || Current State = 010 || Input = 0 || Next State = 000 || Output = 0
Time = 45000 || Current State = 000 || Input = 0 || Next State = 000 || Output = 0
Time = 50000 || Current State = 000 || Input = 1 || Next State = 001 || Output = 0
Time = 55000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
Time = 60000 || Current State = 001 || Input = 0 || Next State = 010 || Cutput = 0
Time = 65000 || Current State = 010 || Input = 0 || Next State = 000 || Output = 0
Time = 70000 || Current State = 010 || Input = 1 || Next State = 011 || Output = 0
Time = 75000 || Current State = 011 || Input = 1 || Next State = 001 || Output = 0
Time = 77000 || Current State = 011 || Input = 0 || Next State = 100 || Output = 0
Time = 85000 || Current State = 100 || Input = 0 || Next State = 000 || Output = 1
Time = 90000 || Current State = 100 || Input = 1 || Next State = 001 || Output = 1
Time = 95000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
Time = 98000 || Current State = 001 || Input = 0 || Next State = 010 || Output = 0
Time = 105000 || Current State = 010 || Input = 0 || Next State = 000 || Output = 0
Time = 108000 || Current State = 010 || Input = 1 || Next State = 011 || Output = 0
Time = 115000 || Current State = 011 || Input = 1 || Next State = 001 || Output = 0
Time = 125000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
Time = 128000 || Current State = 001 || Input = 0 || Next State = 010 || Output = 0
Time = 135000 || Current State = 010 || Input = 0 || Next State = 000 || Output = 0
Time = 138000 || Current State = 010 || Input = 1 || Next State = 011 || Output = 0
Time = 145000 || Current State = 011 || Input = 1 || Next State = 001 || Output = 0
Time = 148000 || Current State = 011 || Input = 0 || Next State = 100 || Output = 0
Time = 155000 || Current State = 100 || Input = 0 || Next State = 000 || Output = 1
Time = 165000 || Current State = 000 || Input = 0 || Next State = 000 || Output = 0
```

SEQUENCE DETECTOR 1010 USING MOORE MACHINE (OVERLAPPING)

DESIGN MODULE

```
`timescale 1ns / 1ps
module moore 1010_seq_det_over(Clk,Rst,In,OP,CS,NS);
input Clk,Rst,In; // Input signals
output OP;  // Output signal
output [2:0]CS,NS ;
parameter s0 = 0, // state registers
           s1 = 1,
           s10 = 2,
           s101 = 3,
           s1010 = 4;
reg [2:0]C State, N State; // state declarations
// State register
always @(posedge Clk, negedge Rst) begin
     if(!Rst)
    begin
         C State <= s0;
     end
     else begin
         C State <= N State;
     end
end
```

```
FSM
// Next state logic
always @(C_State,In) begin
    case(C State)
         s0
            : N State <= In ? s1 : s0 ;
              : N State <= In ? s1 : s10 ;
         s1
         s10 : N State <= In ? s101 : s0 ;
         s101 : N_State \leftarrow In ? s1 : s1010 ;
         default : N State <= s0;</pre>
    endcase
end
// Output logic
assign OP = (C_State == s1010);
//
assign CS = C State ;
assign NS = N State ;
endmodule
```

TESTBENCH MODULE

```
`timescale 1ns / 1ps
module moore 1010 seq det over tb ();
// Signal Declarations
reg Clk,Rst,In;
wire OP;
wire [2:0]CS,NS;
//instantiating design to test
moore 1010 seq det over UUT(Clk,Rst,In,OP,CS,NS);
// Monitoring the simulation results
initial begin
     $monitor("Time = %0t || Current State = %b || Input = %b || Next
State = %b || Output = %b", $time, CS, In, NS, OP);
end
// clock signal generation
initial begin
     Clk = 1'b0;
     forever #5 Clk = ~Clk;
end
// input signals
initial begin
         Rst = 1'b0;
     #15 Rst = 1'b1; end
initial begin
         In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
```

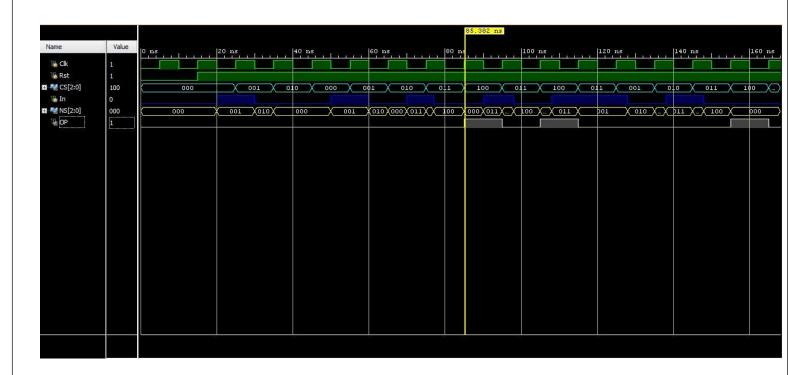
```
FSM
                                                              SEQUENCE DETECTORS
    #10 In = 1'b0;
    #10 In
             = 1'b1;
    #7
        In
             = 1'b0;
    #13 In
             = 1'b1;
    #8
        In
            = 1'b0;
    #10 In
            = 1'b1;
    #10 In = 1'b1;
    #10 In = 1'b0;
    #10 In = 1'b1;
    #10 In = 1'b0;
    #10 In = 1'b0;
```

end

endmodule

SIMULATION WAVEFORMS

#10 \$finish();



SIMULATION OUTPUTS

```
Time = 0 || Current State = 000 || Input = 0 || Next State = 000 || Output = 0
Time = 20000 || Current State = 000 || Input = 1 || Next State = 001 || Output = 0
Time = 25000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
Time = 30000 || Current State = 001 || Input = 0 || Next State = 010 || Output = 0
Time = 35000 || Current State = 010 || Input = 0 || Next State = 000 || Output = 0
Time = 45000 || Current State = 000 || Input = 0 || Next State = 000 || Output = 0
Time = 50000 || Current State = 000 || Input = 1 || Next State = 001 || Output = 0
Time = 55000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
Time = 60000 || Current State = 001 || Input = 0 || Next State = 010 || Output = 0
Time = 65000 || Current State = 010 || Input = 0 || Next State = 000 || Output = 0
Time = 70000 || Current State = 010 || Input = 1 || Next State = 011 || Output = 0
Time = 75000 || Current State = 011 || Input = 1 || Next State = 001 || Output = 0
Time = 77000 || Current State = 011 || Input = 0 || Next State = 100 || Output = 0
Time = 85000 || Current State = 100 || Input = 0 || Next State = 000 || Output = 1
Time = 90000 || Current State = 100 || Input = 1 || Next State = 011 || Output = 1
Time = 95000 || Current State = 011 || Input = 1 || Next State = 001 || Output = 0
Time = 98000 || Current State = 011 || Input = 0 || Next State = 100 || Output = 0
Time = 105000 || Current State = 100 || Input = 0 || Next State = 000 || Output = 1
Time = 108000 || Current State = 100 || Input = 1 || Next State = 011 || Output = 1
Time = 115000 || Current State = 011 || Input = 1 || Next State = 001 || Output = 0
Time = 125000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
Time = 128000 || Current State = 001 || Input = 0 || Next State = 010 || Output = 0
Time = 135000 || Current State = 010 || Input = 0 || Next State = 000 || Output = 0
Time = 138000 || Current State = 010 || Input = 1 || Next State = 011 || Output = 0
Time = 145000 || Current State = 011 || Input = 1 || Next State = 001 || Output = 0
Time = 148000 || Current State = 011 || Input = 0 || Next State = 100 || Output = 0
Time = 155000 || Current State = 100 || Input = 0 || Next State = 000 || Output = 1
Time = 165000 || Current State = 000 || Input = 0 || Next State = 000 || Output = 0
```

SEQUENCE DETECTOR 101X USING MEALY MACHINE (OVERLAPPING)

DESIGN MODULE

```
`timescale 1ns / 1ps
module mealy 101X seq det over(Clk,Rst,In,OP,CS,NS);
input Clk,Rst,In; // Input signals
output OP;  // Output signal
output [1:0]CS,NS; // states
parameter s0 = 0, // state registers
            s1 = 1,
            s10 = 2,
            s101 = 3;
reg [1:0]C State, N State; // state declarations
// State register
always @(posedge Clk, negedge Rst) begin
     if(!Rst)
    begin
          C State <= s0;
     end
     else begin
          C State <= N State;</pre>
     end
end
```

endcase

end

FSM

// Next state logic

always @(C_State,In) begin

s0 : N State <= In ? s1:s0 ;

s1 : N State <= In ? s1:s10 ;

s10 : N State <= In ? s101:s0 ;</pre>

s101: N State <= In ? s1:s10 ;

default : N State <= s0;</pre>

case(C State)

```
// Output logic
assign OP = ((C_State == s101 ) ?(In ? 1:1 ):(In ? 0 : 0)) ;

//assing OP = (C_State == s101) --> we can use it also for this sequence
Detector

//
assign CS = C_State ;
assign NS = N_State ;
endmodule
```

TESTBENCH MODULE

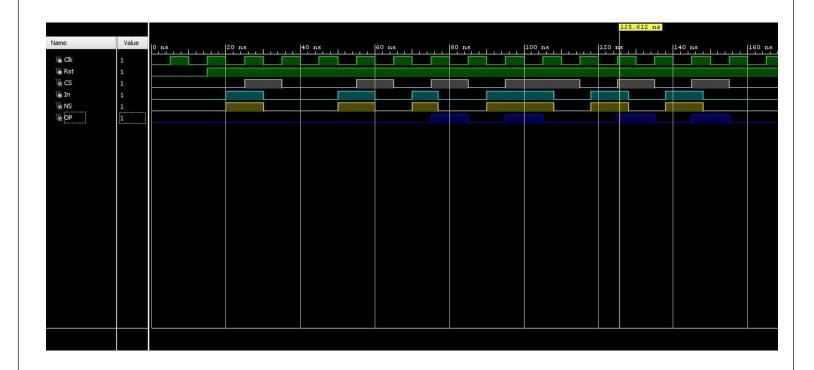
```
`timescale 1ns / 1ps
module mealy 101X seq det over tb ();
// Signal Declarations
reg Clk,Rst,In;
wire OP;
wire CS, NS;
//instantiating design to test
mealy 101X seq det over UUT(Clk,Rst,In,OP,CS,NS);
// Monitoring the simulation results
initial begin
     $monitor("Current State = %b || Input = %b || Next State = %b ||
Output = %b", CS, In, NS, OP);
end
// clock signal generation
initial begin
     Clk = 1'b0;
     forever #5 Clk = ~Clk;
end
// input signals
initial begin
         Rst = 1'b0;
     #15 Rst = 1'b1; end
initial begin
         In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
```

```
FSM
                                                             SEQUENCE DETECTORS
    #10 In = 1'b0;
    #10 In
            = 1'b1;
    #7
        In
            = 1'b0;
    #13 In
            = 1'b1;
    # 8 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 $finish();
```

endmodule

end

SIMULATION WAVEFORMS



FSM

SIMULATION OUTPUTS

```
Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Current State = 0 || Input = 1 || Next State = 1 || Output = 0
Current State = 1 || Input = 1 || Next State = 1 || Output = 0
Current State = 1 || Input = 0 || Next State = 0 || Output = 0
Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Current State = 0 || Input = 1 || Next State = 1 || Output = 0
Current State = 1 || Input = 1 || Next State = 1 || Output = 0
Current State = 1 || Input = 0 || Next State = 0 || Output = 0
Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Current State = 0 || Input = 1 || Next State = 1 || Output = 0
Current State = 1 || Input = 1 || Next State = 1 || Output = 1
Current State = 1 || Input = 0 || Next State = 0 || Output = 1
Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Current State = 0 || Input = 1 || Next State = 1 || Output = 0
Current State = 1 || Input = 1 || Next State = 1 || Output = 1
Current State = 1 || Input = 1 || Next State = 1 || Output = 0
Current State = 1 || Input = 0 || Next State = 0 || Output = 0
Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Current State = 0 || Input = 1 || Next State = 1 || Output = 0
Current State = 1 || Input = 1 || Next State = 1 || Output = 1
Current State = 1 || Input = 0 || Next State = 0 || Output = 1
Current State = 0 || Input = 0 || Next State = 0 || Output = 0
Current State = 0 || Input = 1 || Next State = 1 || Output = 0
Current State = 1 || Input = 1 || Next State = 1 || Output = 1
Current State = 1 || Input = 0 || Next State = 0 || Output = 1
Current State = 0 || Input = 0 || Next State = 0 || Output = 0
```

SEQUENCE DETECTOR 10X0 USING MOORE MACHINE (OVERLAPPING)

DESIGN MODULE

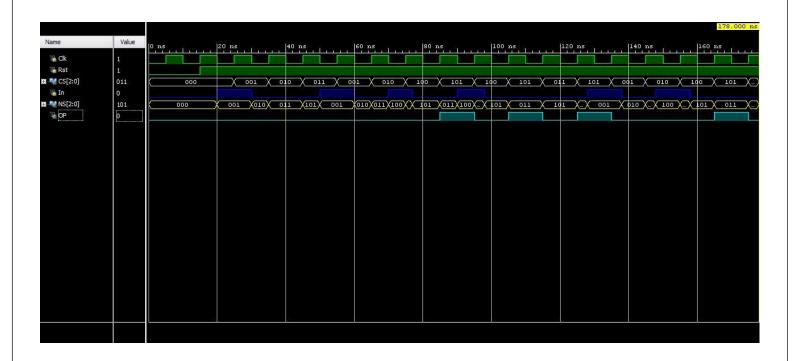
```
`timescale 1ns / 1ps
module moore 10X0 seq det over(Clk,Rst,In,OP,CS,NS);
input Clk,Rst,In; // Input signals
output OP;  // Output signal
output [2:0]CS,NS; // states
          s0 = 0 , // state registers
parameter
               = 1 ,
            s1
            s10 = 2,
            s100 = 3,
            s101 = 4,
            s10X0 = 5;
reg x; // to know the 10X0--> X direction(1010/1000)
reg [2:0]C State, N State; // state declarations
// State register
always @(posedge Clk, negedge Rst) begin
     if(!Rst)
    begin
          C State <= s0;
     end
     else begin
          C State <= N State;</pre>
     end
end
```

```
FSM
// Next state logic
always @(C_State,In) begin
     case(C State)
          s0
                : N State <= In ? s1 : s0 ;
                : N State <= In ? s1 : s10 ;
          s1
          s10
                : N State <= In ? s101 : s100 ;
                : begin N State <= In ? s1 : s10X0 ; x <= 0 ;end
          s100
                : begin N State <= In ? s1 : s10X0 ; x <= 1 ; end
          s101
          s10X0 : N State \leftarrow x ? (In ? s101 : s100 ) : (In ? s1 : s0 ) ;
          default : N State <= s0;</pre>
     endcase
end
// Output logic
assign OP = (C State == s10X0);
//
assign CS = C_State ;
assign NS = N State ;
endmodule
```

TESTBENCH MODULE

```
`timescale 1ns / 1ps
module moore 10X0 seq_det_over_tb ();
// Signal Declarations
reg Clk,Rst,In;
wire OP;
wire [2:0]CS,NS;
//instantiating design to test
moore 10X0 seq det over UUT(Clk,Rst,In,OP,CS,NS);
// Monitoring the simulation results
initial begin
     $monitor("time = %0t || Current State = %b || Input = %b || Next
State = %b || Output = %b", $time, CS, In, NS, OP);
end
// clock signal generation
initial begin
     Clk = 1'b0;
     forever #5 Clk = ~Clk;
end
// input signals
initial begin
         Rst = 1'b0;
     #15 Rst = 1'b1; end
initial begin
         In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #10 In = 1'b0;
     #10 In = 1'b1;
     #7 In = 1'b0;
     #13 In = 1'b1;
     #8 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 $finish();
end
endmodule
```

SIMULATION WAVEFORMS



SIMULATION OUTPUTS

```
time = 0 || Current State = 000 || Input = 0 || Next State = 000 || Output = 0
time = 20000 || Current State = 000 || Input = 1 || Next State = 001 || Output = 0
time = 25000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
time = 30000 || Current State = 001 || Input = 0 || Next State = 010 || Output = 0
time = 35000 || Current State = 010 || Input = 0 || Next State = 011 || Output = 0
time = 45000 || Current State = 011 || Input = 0 || Next State = 100 || Output = 0
time = 50000 || Current State = 011 || Input = 1 || Next State = 001 || Output = 0
time = 55000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
time = 60000 || Current State = 001 || Input = 0 || Next State = 010 || Output = 0
time = 65000 || Current State = 010 || Input = 0 || Next State = 011 || Output = 0
time = 70000 || Current State = 010 || Input = 1 || Next State = 100 || Output = 0
time = 75000 || Current State = 100 || Input = 1 || Next State = 001 || Output = 1
time = 77000 || Current State = 100 || Input = 0 || Next State = 100 || Output = 1
time = 90000 || Current State = 100 || Input = 1 || Next State = 001 || Output = 1
time = 95000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
time = 98000 || Current State = 001 || Input = 0 || Next State = 010 || Output = 0
time = 105000 || Current State = 010 || Input = 0 || Next State = 011 || Output = 0
time = 108000 || Current State = 010 || Input = 1 || Next State = 100 || Output = 0
time = 115000 || Current State = 100 || Input = 1 || Next State = 001 || Output = 1
time = 125000 || Current State = 001 || Input = 1 || Next State = 001 || Output = 0
time = 128000 || Current State = 001 || Input = 0 || Next State = 010 || Output = 0
time = 135000 || Current State = 010 || Input = 0 || Next State = 011 || Output = 0
time = 138000 || Current State = 010 || Input = 1 || Next State = 100 || Output = 0
time = 145000 || Current State = 100 || Input = 1 || Next State = 001 || Output = 1
time = 148000 || Current State = 100 || Input = 0 || Next State = 100 || Output = 1
```