LAB 6 – Raja Aadhithan

Design – Sequence Detector:

Code:

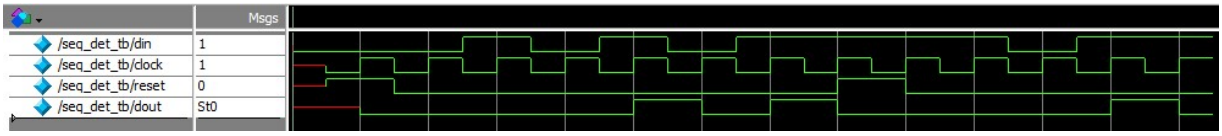
```
module seq_det(input seq_in,clock,reset,output det_o);
reg [1:0]state,next_state;
parameter IDLE=2'b00, STATE1 = 2'b01, STATE2 = 2'b10, STATE3=2'b11;

always@(posedge clock)begin
    if(reset)begin
        state<=IDLE;
    end
    else begin
        state<=next_state;
    end
end
always@(*)begin
     case(state)
        IDLE   : next_state = seq_in ? STATE1 : IDLE ;
        STATE1 : next_state = seq_in ? STATE1 : STATE2 ;
        STATE2 : next_state = seq_in ? STATE3 : IDLE ;
        STATE3 : next_state = seq_in ? STATE1 : STATE2 ;
        default: next_state=IDLE;
     endcase
   end
assign det_o = (state==STATE3);
endmodule
```

Testbench:

```
module seq_det_tb();

   //Testbench global variables
   reg  din,clock,reset;
   wire dout;

   //Parameter constant for CYCLE
   parameter CYCLE = 10;

   //DUT Instantiation
   seq_det SQD(.seq_in(din),
          .clock(clock),
               .reset(reset),
          .det_o(dout));

   //Step1 : Generate clock, using parameter "CYCLE"
   always
      begin
    #(CYCLE/2);
    clock = 1'b0;
    #(CYCLE/2);
    clock=~clock;
      end

   /*Step2 : Write a task named "initialize" to initialize
   the input din of sequence detector*/
   task initialize( );
      begin
         din = 0;
      end
   endtask
```

```
  //Delay task
  task delay(input integer i);
     begin
   #i;
     end
  endtask

  /*Step3 : Write a task named "RESET" to reset the design,
  use delay task for adding delays*/
  //Reset task
  task RESET();
     begin
   delay(5);
   reset=1'b1;
   delay(10);
   reset=1'b0;
     end
  endtask

  /*Step4 : Write a task named "stimulus" which provides input to
  design on negedge of clock*/
  task stimulus(input data);
     begin
        @(negedge clock);
        din = data;
     end
  endtask
  //Process to monitor the changes in the variables
  initial
     $monitor("Reset=%b, state=%b, Din=%b, Output Dout=%b",
     reset,SQD.state,din,dout);

  /*Process to display a string after the sequence is detected and dout is asserted.
  SQD.state is used here as a path hierarchy where SQD is the instance name acting
  like a handle to access the internal register "state" */
  always@(SQD.state or dout)
     begin
   if(SQD.state==2'b11 && dout==1)
      $display("Correct output at state %b", SQD.state);
     end

  /*Process to generate stimulus by calling the tasks and
  passing the sequence in an overlapping mode*/
  initial
     begin
   initialize;
   RESET;
   stimulus(0);
   stimulus(1);
   stimulus(0);
   stimulus(1);
   stimulus(0);
   stimulus(1);
   stimulus(1);
   RESET;
   stimulus(1);
   stimulus(0);
   stimulus(1);
   stimulus(1);
   delay(10);
   $finish;
     end


endmodule
```

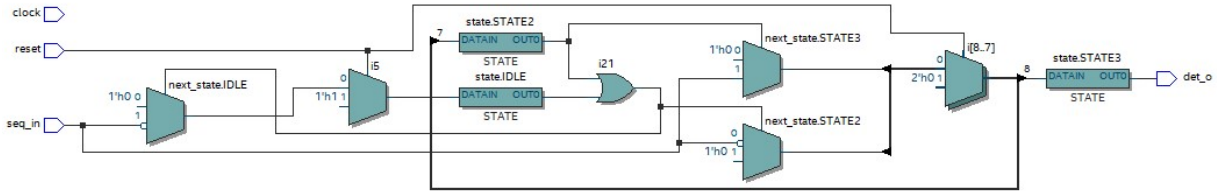
Wave:



Output:

```
VSIM 41> run -all
# Reset=x, state=xx, Din=0, Output Dout=x
# Reset=1, state=xx, Din=0, Output Dout=x
# Reset=1, state=00, Din=0, Output Dout=0
# Reset=0, state=00, Din=0, Output Dout=0
# Reset=0, state=00, Din=1, Output Dout=0
# Reset=0, state=01, Din=1, Output Dout=0
# Reset=0, state=01, Din=0, Output Dout=0
# Reset=0, state=10, Din=0, Output Dout=0
# Reset=0, state=10, Din=1, Output Dout=0
# Correct output at state 11
# Reset=0, state=11, Din=1, Output Dout=1
# Reset=0, state=11, Din=0, Output Dout=1
# Reset=0, state=10, Din=0, Output Dout=0
# Reset=0, state=10, Din=1, Output Dout=0
# Correct output at state 11
# Reset=0, state=11, Din=1, Output Dout=1
# Reset=1, state=00, Din=1, Output Dout=0
# Reset=0, state=01, Din=1, Output Dout=0
# Reset=0, state=01, Din=0, Output Dout=0
# Reset=0, state=10, Din=0, Output Dout=0
# Reset=0, state=10, Din=1, Output Dout=0
# Correct output at state 11
# Reset=0, state=11, Din=1, Output Dout=1
# Reset=0, state=01, Din=1, Output Dout=0
# ** Note: $finish    : C:/Users/Aadhithan/Documents/Ve
6/tb/seq_det_tb.v(117)
#    Time: 135 ps  Iteration: 0  Instance: /seq_det_tb
```

RTL:



Design : Vending machine:

Code:

```
module coin(input clk,rst,i,j, output x,y);
parameter idle = 3'b000, st1 = 3'b001, st2 = 3'b010, st3 = 3'b011, st4 = 3'b100;
reg[2:0]next_state,state;
always@(*)begin
    case(state)
    idle : next_state = i ? ( j ? st2 : st1) : idle ;
    st1  : next_state = i ? ( j ? st3 : st2) : st1 ;
    st2  : next_state = i ? ( j ? st4 : st3) : st2 ;
    st3  : next_state = idle;
    st4  : next_state = idle;
    default: next_state = idle;
    endcase
end
always@(posedge clk or posedge rst)begin
    if(rst) state<=idle;
    else state<=next_state;
end
assign x = (state==st3)||(state==st4);
assign y = (state==st4);
endmodule
```

Testbench:

```
module coin_tb();
reg clk,reset,i,j;
wire x,y;
coin dut(clk,reset,i,j,x,y);
initial begin
    clk = 1;
    forever #5 clk = ~clk;
end
initial begin
    $monitor("@time:%3d,input (ij) is %b,%b and output (xy) is %b,%b",$time,i,j,x,y);
    reset = 1 ;
    #15;
    reset = 0;
    i = 0;
    j = 1;
    #10;
    i = 0;
    j = 0;
    #10;
    i = 1;
    j = 1;
    #10;
    i = 1;
    j = 0;
    #10;
    i = 0;
    j = 1;
    #10;
    i = 1;
    j = 1;
    #10;
    i = 1;
    j = 1;
    #10;
    i = 1;
    j = 0;
    #10;
```

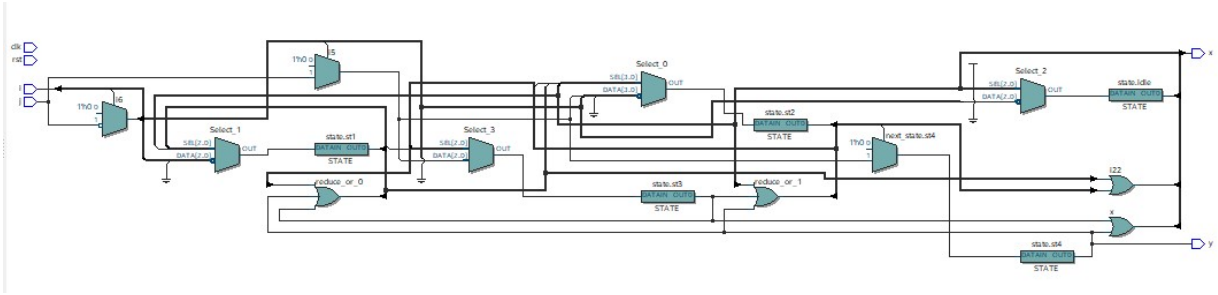
```
    i = 1;
    j = 1;
    #10;
    i = 1;
    j = 0;
    #10;
    i = 0;
    j = 1;
    #10;
    i = 1;
    j = 1;
    #10;
    i = 1;
    j = 1;
    #10;
    i = 1;
    j = 0;
    #10;
    $finish;
end
endmodule
```

Wave:

Output:

```
VSIM 45> run -all
# @time:  0,input (ij) is x,x and output (xy) is 0,0
# @time: 15,input (ij) is 0,1 and output (xy) is 0,0
# @time: 25,input (ij) is 0,0 and output (xy) is 0,0
# @time: 35,input (ij) is 1,1 and output (xy) is 0,0
# @time: 45,input (ij) is 1,0 and output (xy) is 0,0
# @time: 50,input (ij) is 1,0 and output (xy) is 1,0
# @time: 55,input (ij) is 0,1 and output (xy) is 1,0
# @time: 60,input (ij) is 0,1 and output (xy) is 0,0
# @time: 65,input (ij) is 1,1 and output (xy) is 0,0
# @time: 80,input (ij) is 1,1 and output (xy) is 1,1
# @time: 85,input (ij) is 1,0 and output (xy) is 1,1
# @time: 90,input (ij) is 1,0 and output (xy) is 0,0
# @time: 95,input (ij) is 1,1 and output (xy) is 0,0
# @time:105,input (ij) is 1,0 and output (xy) is 0,0
# @time:110,input (ij) is 1,0 and output (xy) is 1,0
# @time:115,input (ij) is 0,1 and output (xy) is 1,0
# @time:120,input (ij) is 0,1 and output (xy) is 0,0
# @time:125,input (ij) is 1,1 and output (xy) is 0,0
# @time:140,input (ij) is 1,1 and output (xy) is 1,1
# @time:145,input (ij) is 1,0 and output (xy) is 1,1
# @time:150,input (ij) is 1,0 and output (xy) is 0,0
# ** Note: $finish    : C:/Users/Aadhithan/Documents/Ve
#    Time: 155 ps  Iteration: 0  Instance: /coin_tb
# 1
```

RTL:


clk
rst
I
I
I5
I6
Select_1
Select_3
Select_0
Select_2
state.st1
state.st2
state.st3
state.st4
state.idle
STATE
reduce_or_0
reduce_or_1
next_state.st4
I22
x
y