Projectg 1 – Sequence Detector

1. Design

“a. Melay overloading and non-over loading ”

```
`timescale 1ns / 1ps
 module seq_1010_melay(
    input i_clock,
    input i_reset,
   // btn
    input i_btn,
   //led
    output reg o_led
     );
   //locsal parameters
   localparam [2:0] s0 = 0 ,s1 = 1 ,s2 = 2 ,s3 = 3 ;
   //internal regesters or wire declarstions
   reg [2:0] state,next_state;
   //reset condition
 always@ (posedge i_clock) begin
   if (i_reset)
      state <= 3'b000        ;
   else
      state <= next_state     ;
 end

 always@(*)begin

 //store the state

   next_state = state;

 //state machine

   case(state)
      s0 : next_state <= i_btn ? s1:s0 ;
      s1 : next_state <= i_btn ? s1:s2 ;
      s2 : next_state <= i_btn ? s3:s0 ;
      s3 : next_state <= i_btn ? s1:s0 ; //non overloadding
      //s3 : next_state <= i_btn ? s1:s2 ; //overloading
   endcase
end
```

```
always@(posedge i_clock) begin
   if (i_reset)
      o_led <= 0;
   else begin
      if(~i_btn & (state==s3))
         o_led <=1'b1;
      else
         o_led <= 1'b0;
      end
end
endmodule
```

“b.moore overloading and non-overloading”

```
`timescale 1ns / 1ps
 module seq_1010_moore(
   input i_clock,
   input i_reset,
   // btn
   input i_btn,
   //led
   output o_led
    );

 //locsal parameters
   localparam [2:0] s0 = 0 ,s1 = 1 ,s2 = 2 ,s3 = 3 ,s4 = 4 ;

   //internal regesters or wire declarstions
   reg [2:0] state,next_state;

   //reset condition
always@ (posedge i_clock) begin
   if (i_reset)
      state <= 3'b000        ;
   else
      state <= next_state     ;
end
always@(*)begin




//store the state

   next_state = state;
//state machine

   case(state)
      s0 : next_state <= i_btn ? s1:s0 ;
      s1 : next_state <= i_btn ? s1:s2 ;
      s2 : next_state <= i_btn ? s3:s0 ;
```

```
      s3 : next_state <= i_btn ? s1:s4 ;
      //s4 : next_state <= i_btn ? s1:s0 ; //non overloadding
      s4 : next_state <= i_btn ? s3:s0 ; //overloading
   endcase
end
assign o_led = (state == s4) ? 1:0  ;
endmodule
```

2.Test bench

"(Melay test bench)"

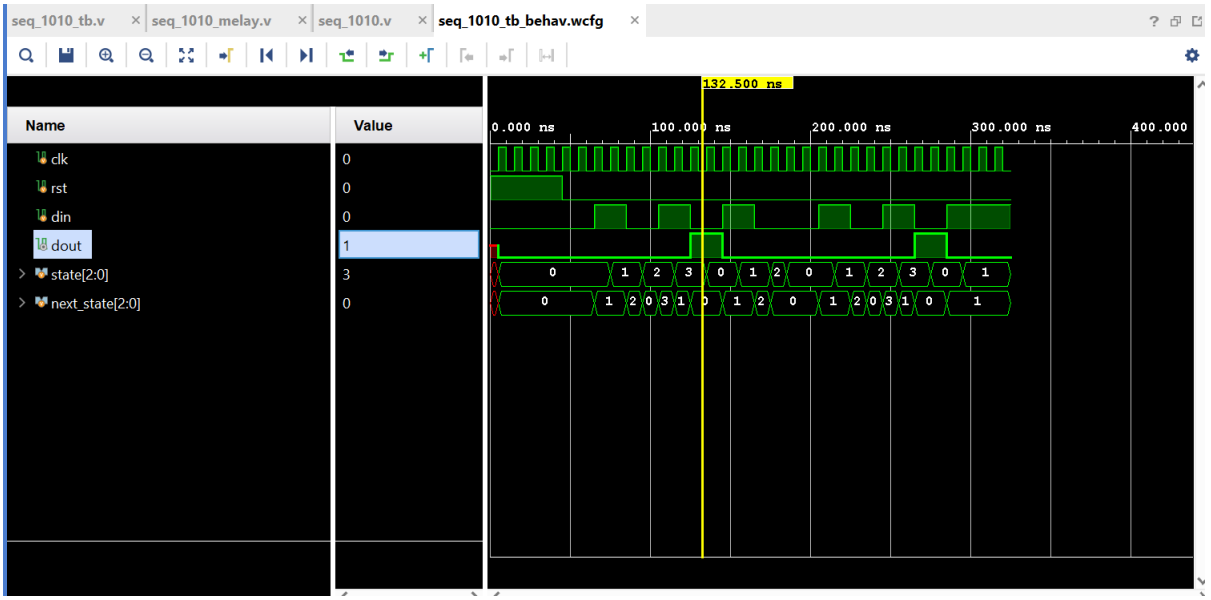
```
`timescale 1ns / 1ps
module seq_1010_tb;

//internal regs/wires
 reg clk,rst,din;
 wire dout     ;
always
   #5 clk = ~clk;
initial begin
   clk=0;
   rst=1;
   din=0;
   #45 rst = 0;
   #20 din = 1;
   #20 din = 0;
   #20 din = 1;
   #20 din = 0;
   #20 din = 1;
   #20 din = 0;
   #20 din = 0;
   #20 din = 1;
   #20 din = 0;
   #20 din = 1;
   #20 din = 0;
   #20 din = 1;
   #40 $stop;
end

//initializaton
seq_1010_melay uut(
.i_clock(clk),
.i_reset(rst),
.i_btn(din),
.o_led(dout)
);
endmodule
```

“(moore test bench)”

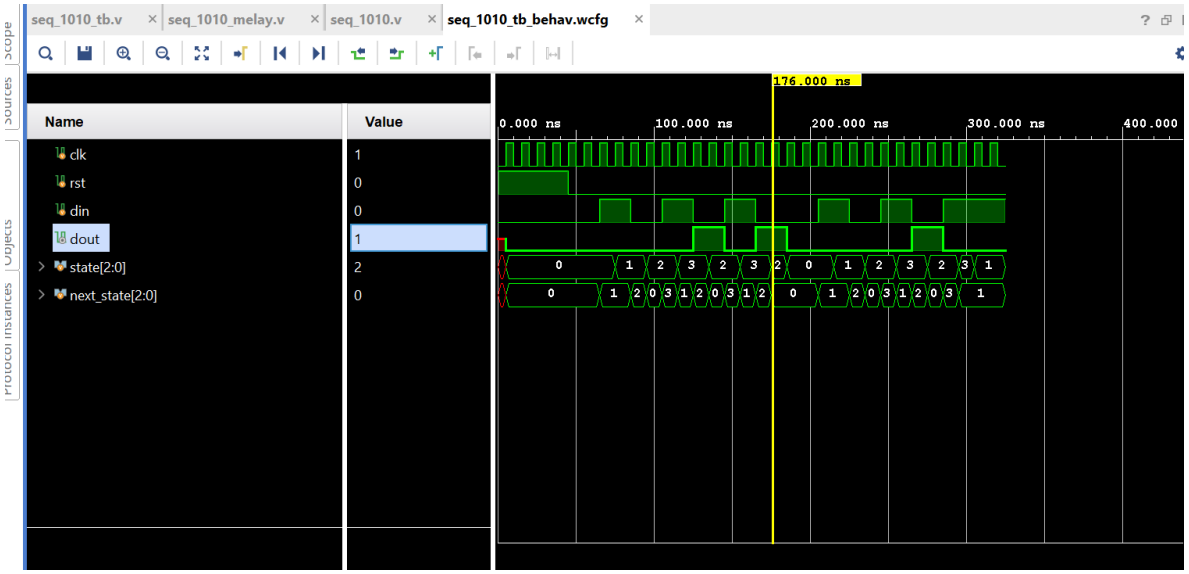
```
`timescale 1ns / 1ps
 module seq_1010_tb;
//internal regs/wires
reg clk,rst,din;
wire dout     ;
always
   #5 clk = ~clk;
initial begin
   clk=0;
   rst=1;
   din=0;

   #45 rst = 0;
   #20 din = 1;
   #20 din = 0;
   #20 din = 1;
   #20 din = 0;
   #20 din = 1;
   #20 din = 0;
   #20 din = 0;
   #20 din = 1;
   #20 din = 0;
   #20 din = 1;
   #20 din = 0;
   #20 din = 1;

   #40 $stop;
end

//initializaton
seq_1010_moore uut(
.i_clock(clk),
.i_reset(rst),
.i_btn(din),
.o_led(dout)
);

endmodule
```

3. Simulation Results

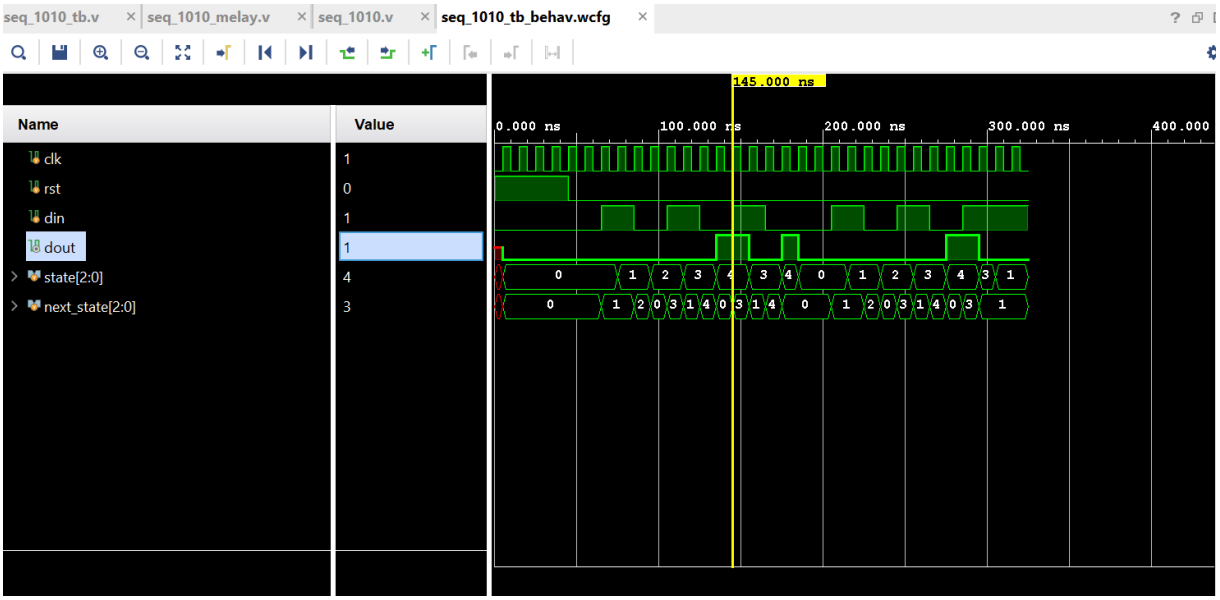
a.melay non-overloading



b. melay overloading:



c. moore overloading



d. moore non-overloading