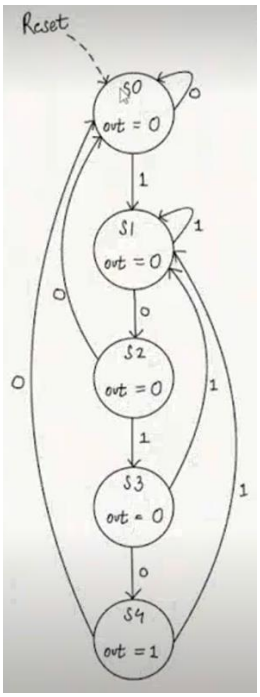
# Moore State Machine

In Moore machine, output only depends on the present state. It is independent of current input.

## 1. Non-Overlapping Moore State Machine




In this type of sequence detector does not allow overlap, but resets itself to the start state when the sequence has been detected.

***For example,*** after the initial sequence 1101 has been detected, the detector with no overlap resets and starts searching for the initial 1 of the next sequence.

# Design Code

```
`timescale 1ns / 1ps
module seq_1010(
//Global Signals
    input   i_clock,
    input   i_reset,
//Button
    input   i_btn,
//LED
    output o_led
) ;
//Local Parameters
localparam [2:0] S0=0, S1=1, S2=2, S3=3, S4=4;

//Internal Registers or wire declarations
reg [2:0] state, next_state;

//Reset Condition
always@(posedge i_clock) begin
    if(i_reset)
        state<=3'b000;
    else
```

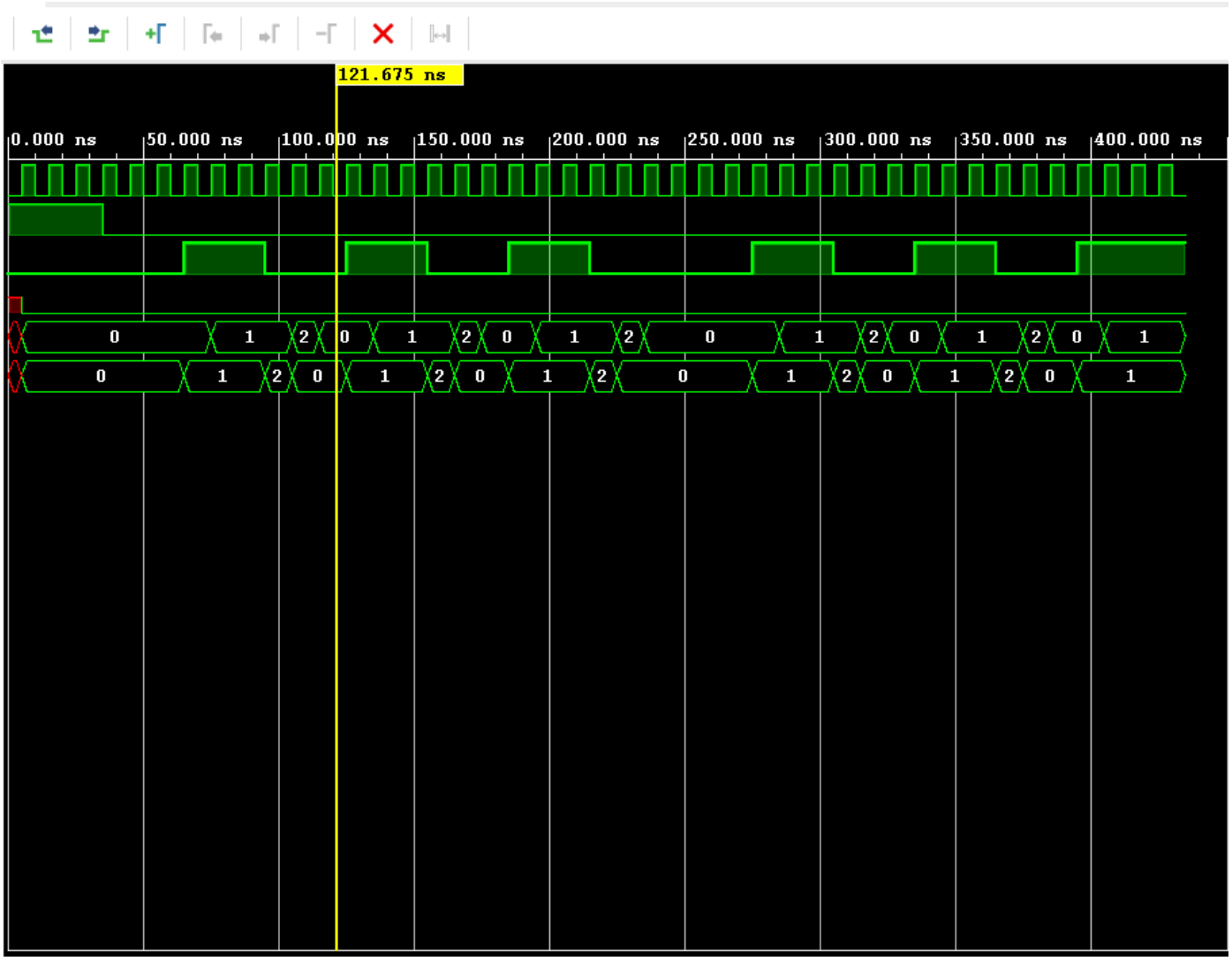
```
always@(*) begin
    //Store the state
    next_state = state;

    //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 overlapping
//      S4: next_state <= i_btn ? S3:S0;  //Overlapping
    endcase

end

assign o_led = (state == S4) ? 1 : 0;
endmodule
```

# Testbench Code

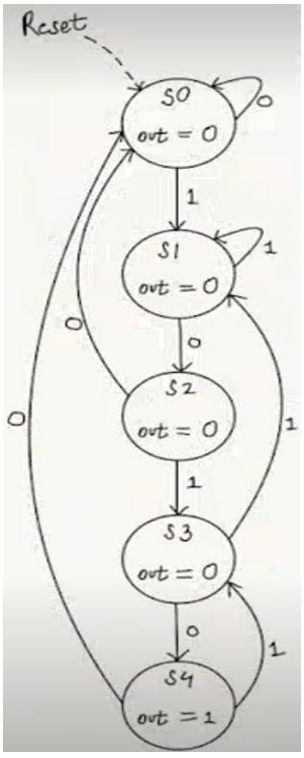
```
`timescale 1ns / 1ps

module seq_1010_tb;

//Internal Registers/Wires
reg clk, rst, din;
wire dout;

always
    #5 clk = ~clk;

initial begin

    clk = 0;
    rst=1;
    din=0;

    #35 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
```

```
//initial begin
//  $dumpfile("dump.vod");
//  $dumpvars(0);
// end

//Instantiation
seq_1010 uut(
.i_clock(clk),
.i_reset(rst),
.i_btn(din),
.o_led(dout)
);

endmodule
```

# Output Waveform:




## 2. Overlapping Moore State Machine




In this type of sequence detector allows overlap, the final bits of one sequence can be the start of another sequence.

***For example,*** will be an 1101 sequence detector. It raises an output of 1 when the last 4 binary bits received are 1101.

# Design Code

```
module seq_1010(
//Global Signals
    input   i_clock,
    input   i_reset,
//Button
    input   i_btn,
//LED
    output o_led
) ;
//Local Parameters
localparam [2:0] S0=0, S1=1, S2=2, S3=3, S4=4;

//Internal Registers or wire declarations
reg [2:0] state, next_state;

//Reset Condition
always@(posedge i_clock) begin
    if(i_reset)
        state<=3'b000;
    else
        state<=next_state;
end
```

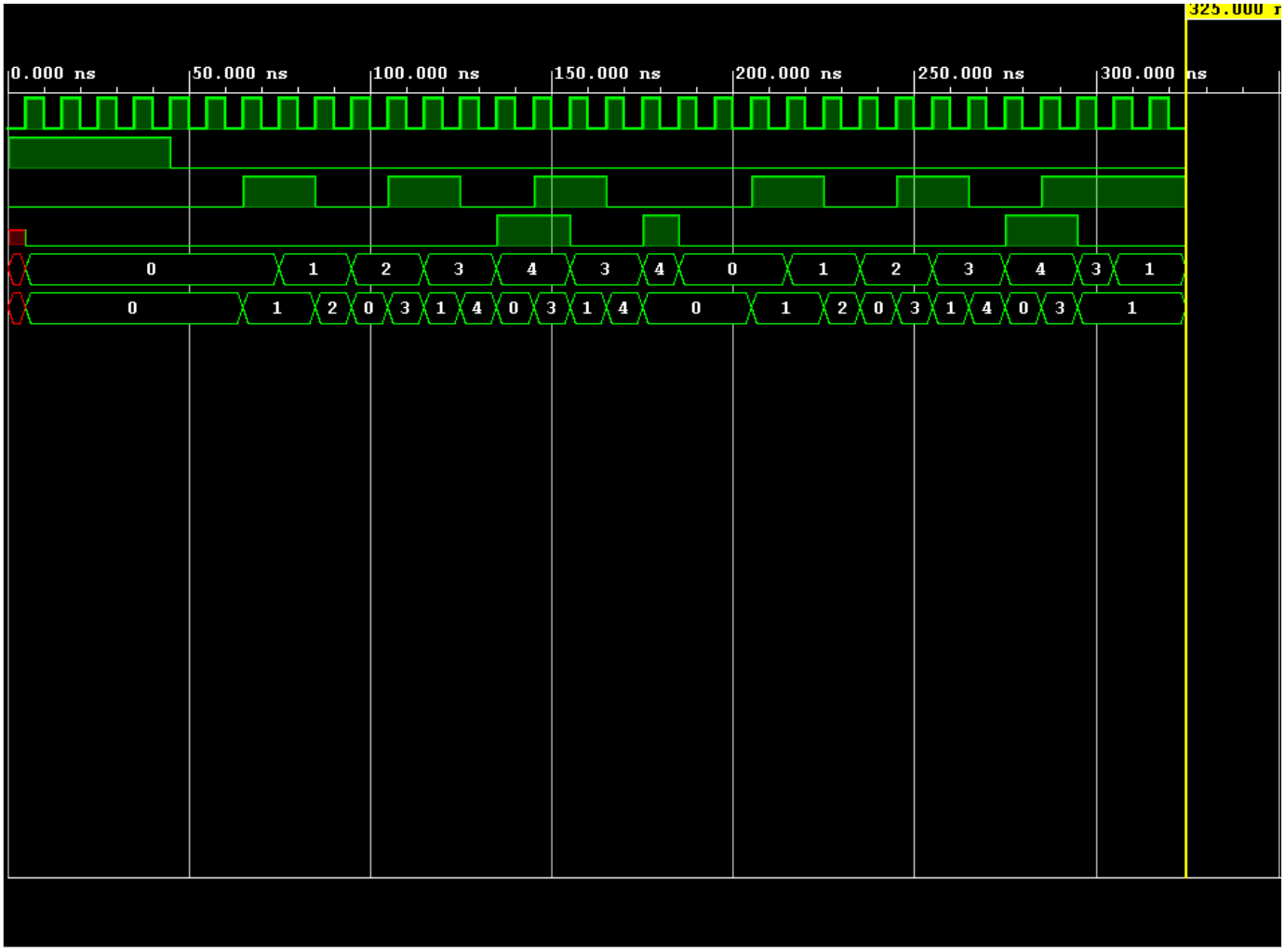
```
always@(*) begin
    //Store the state
    next_state = state;

    //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 overlapping
      S4: next_state <= i_btn ? S3:S0;  //Overlapping
    endcase

end

assign o_led = (state == S4) ? 1 : 0;
endmodule
```

# Testbench Code

```
 `timescale 1ns / 1ps

module seq_1010_tb;

 //Internal Registers/Wires
 reg clk, rst, din;
 wire dout;

always
    #5 clk = ~clk;

initial begin

    clk = 0;
    rst=1;
    din=0;

    #35 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
```

```
//initial begin
//  $dumpfile("dump.vcd");
//  $dumpvars(0);
// end

//Instantiation
seq_1010 uut(
.i_clock(clk),
.i_reset(rst),
.i_btn(din),
.o_led(dout)
);

endmodule
```

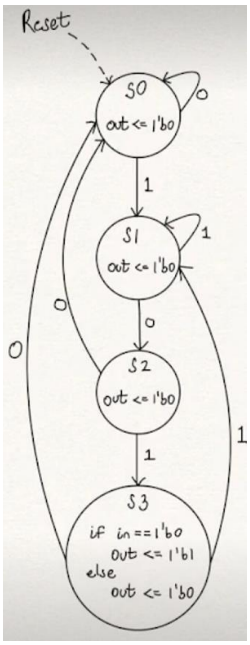
# Output Waveform



# Mealy State Machine

In mealy machine, output depends on the present state and current input.

## 1. Non-Overlapping Mealy State Machine




In a non-overlapping sequence detector, the last bit of one sequence does not become the first bit of the next sequence. For example:

Input :0110101011001

Output:0000100010000

# Design code

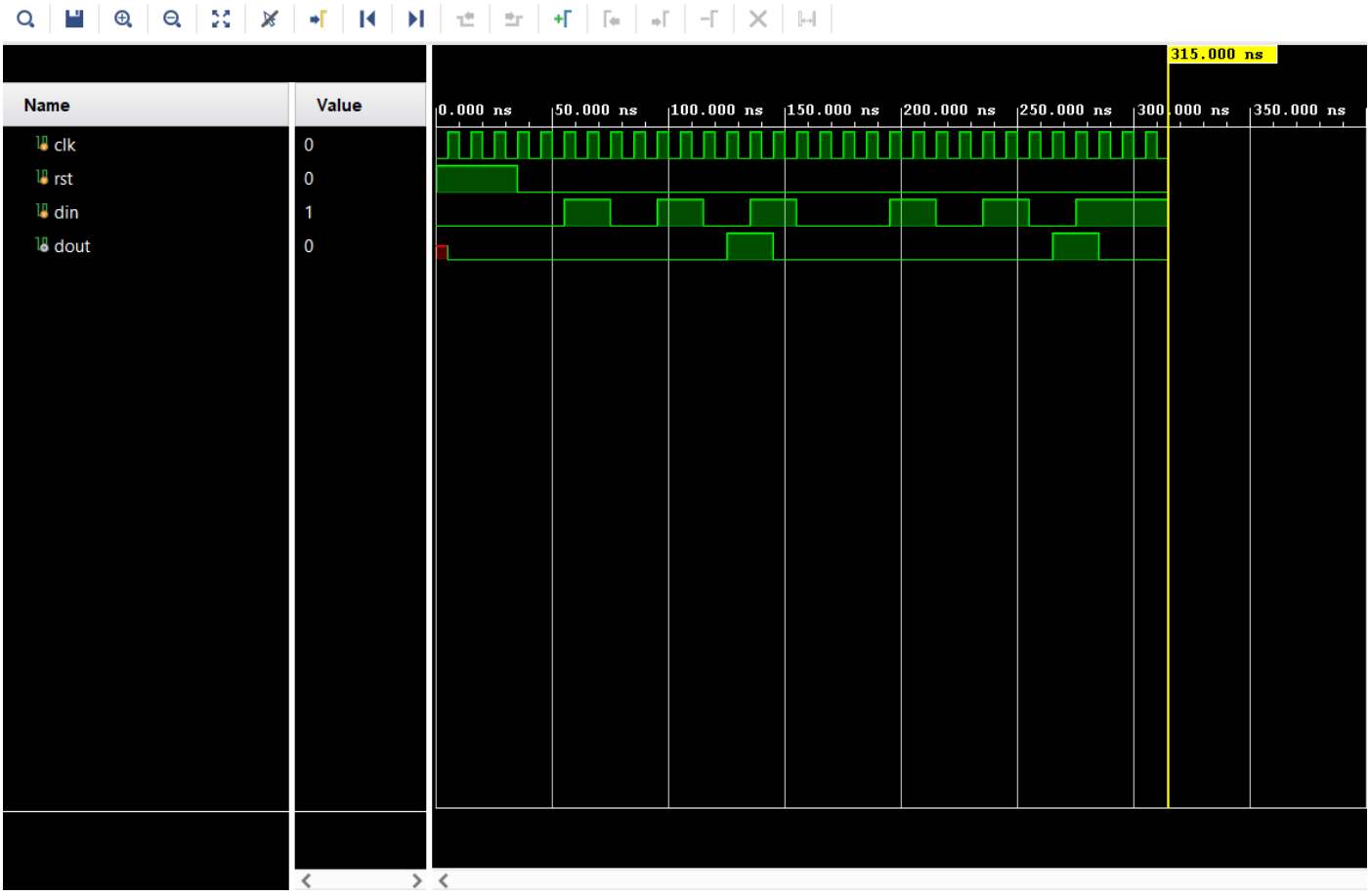
```
`timescale 1ns / 1ps

module seq_1010_mealy(
   //Global Signals
    input   i_clock,
    input   i_reset,
    //Button
    input   i_btn,
    //LED
    output  o_led
    );
//Local Parameters
localparam [2:0] S0=0, S1=1, S2=2, S3=3;

//Internal Registers or wire declarations
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 overlapping
//      S3: next_state <= i_btn ? S3:S0;  //Overlapping
    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
```

# Testbench Code

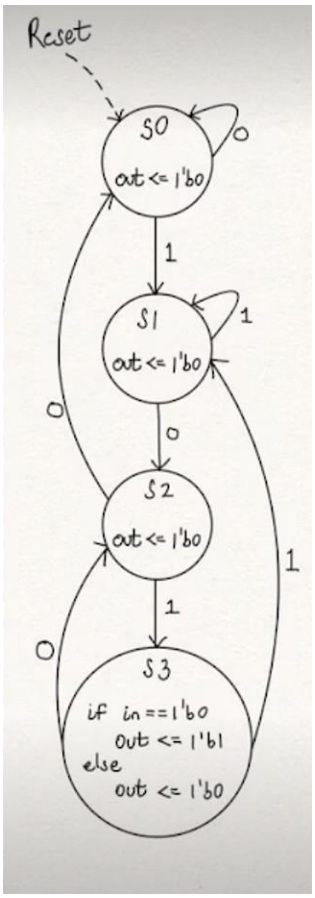
```
`timescale 1ns / 1ps

module seq_1010_tb;

//Internal Registers/Wires
reg clk, rst, din;
wire dout;

always
    #5 clk = ~clk;

initial begin

    clk = 0;
    rst=1;
    din=0;

    #35 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
```

```
//initial begin
//  $dumpfile("dump.vod");
//  $dumpvars(0);
// end

//Instantiation
seq_1010 uut(
.i_clock(clk),
.i_reset(rst),
.i_btn(din),
.o_led(dout)
);

endmodule
```

# Output Waveform



## 2. Overlapping Mealy State Machine




In an overlapping sequence detector, the last bit of one sequence becomes the first bit of the next sequence. For example:

Input :0110101011001

Output:0000101010000

# Design Code

```
`timescale 1ns / 1ps

module seq_1010_mealy(
    //Global Signals
    input   i_clock,
    input   i_reset,
    //Button
    input   i_btn,
    //LED
    output reg o_led
    );
//Local Parameters
localparam [2:0] S0=0, S1=1, S2=2, S3=3;

//Internal Registers or wire declarations
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 overlapping
        S3: next_state <= i_btn ? S1:S2;  //Overlapping
    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
```

# Testbench Code

```
`timescale 1ns / 1ps

module seq_1010_tb;

//Internal Registers/Wires
reg clk, rst, din;
wire dout;

always
    #5 clk = ~clk;

initial begin

    clk = 0;
    rst=1;
    din=0;

    #35 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
```

```
//initial begin
//  $dumpfile("dump.vod");
//  $dumpvars(0);
// end

//Instantiation
seq_1010 uut(
.i_clock(clk),
.i_reset(rst),
.i_btn(din),
.o_led(dout)
);

endmodule
```

# Output Waveform