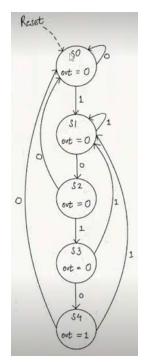
# Moore State Machine

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.

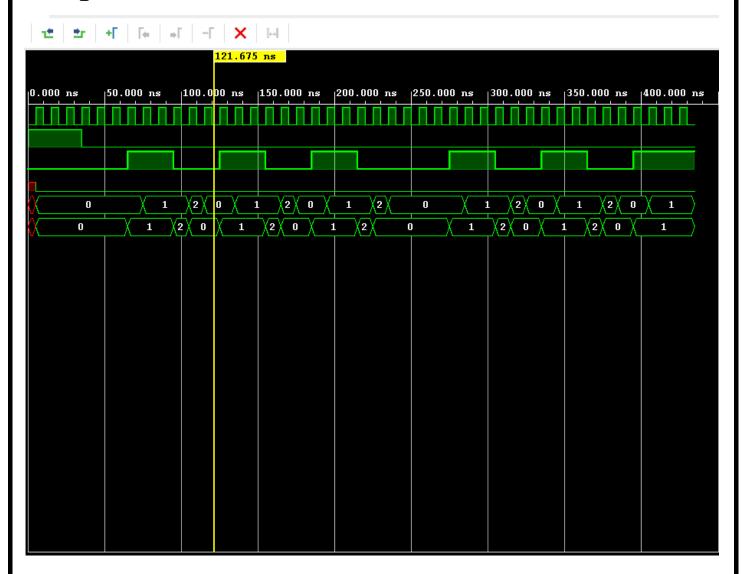
## Design Code

```
timescale 1ns / 1ps
module seq 1010(
//Global Signals
    input i_clock,
    input i reset,
//Button
    input i btn,
    output o_led
//Local Parameters
localparam [2:0] S0=0, S1=1, S2=2, S3=3, S4=4;
//Internal Registers or wire declarations
req [2:0] state, next_state;
//Reset Condition
always@(posedge i clock) begin
    if(i reset)
        state<=3'b000;
    else
```

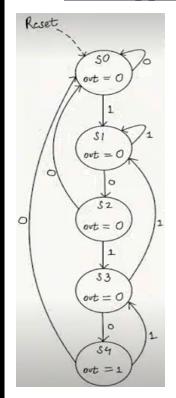
### Testbench Code

```
`timescale 1ns / 1ps
                                 //initial begin
module seq_1010_tb;
                                 // $dumpfile("dump.vod");
                                 // $dumpvars(0);
//Internal Registers/Wires
                                 // end
reg clk, rst, din;
wire dout;
                                  //Instantiation
always
                                 seq 1010 uut(
   #5 clk = ~clk;
                                  .i clock(clk),
initial begin
                                  .i reset(rst),
                                  .i btn(din),
    clk = 0;
                                  .o led(dout)
    rst=1;
                                  );
    din=0;
    #35 \text{ rst} = 0;
                                 endmodule
    #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
```

## **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

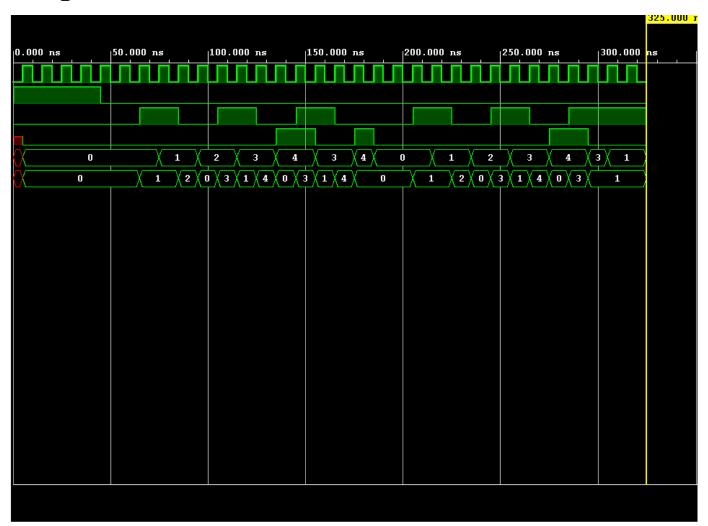
```
module seq_1010(
                                                       always@(*) begin
//Global Signals
    input i clock,
                                                           //Store the state
    input i_reset,
                                                           next state = state;
//Button
   input i btn,
                                                           //State machine
//LED
                                                           case(state)
    output o_led
                                                               S0: next state <= i btn ? S1:S0;
                                                               S1: next_state <= i_btn ? S1:S2;</pre>
//Local Parameters
localparam [2:0] S0=0, S1=1, S2=2, S3=3, S4=4;
                                                             S2: next_state <= i_btn ? S3:S0;
                                                               S3: next state <= i btn ? S1:S4;
//Internal Registers or wire declarations
                                                              S4: next state <= i btn ? S1:S0;
                                                                                               //Non overlapping
reg [2:0] state, next_state;
                                                             S4: next state <= i btn ? S3:S0; //Overlapping
                                                           endcase
//Reset Condition
always@(posedge i_clock) begin
   if(i_reset)
                                                       end
        state<=3'b000;
    else
                                                       assign o_led = (state == S4) ? 1 : 0;
        state<=next state;
                                                       endmodule
end
```

### Testbench Code

end

```
`timescale 1ns / 1ps
                                     //initial begin
                                     // $dumpfile("dump.vod");
module seq_1010_tb;
                                     // $dumpvars(0);
                                     // end
 //Internal Registers/Wires
reg clk, rst, din;
wire dout;
                                     //Instantiation
                                     seq 1010 uut(
always
#5 clk = ~clk;
                                     .i clock(clk),
                                     .i reset(rst),
initial begin
                                     .i_btn(din),
     clk = 0;
                                      .o led(dout)
     rst=1;
                                      );
     din=0;
                                     endmodule
     #35 \text{ 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;
```

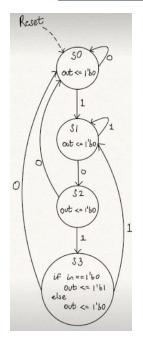
## **Output Waveform**



# 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
                                               always@(*) begin
                                                  //Store the state
                                                  next state = state;
module seq 1010 mealy(
   //Global Signals
                                                  //State machine
    input i_clock,
                                                  case (state)
    input i_reset,
                                                     S0: next_state <= i_btn ? S1:S0;
    //Button
                                                     S1: next state <= i btn ? S1:S2;
    input
           i btn,
                                                     S2: next_state <= i_btn ? S3:S0;
    //LED
                                                     S3: next state <= i btn ? S1:S0;
    output o_led
                                                    S3: next state <= i btn ? S3:S0; //Overlapping
                                                  endcase
//Local Parameters
localparam [2:0] S0=0, S1=1, S2=2, S3=3;
                                               always@(posedge i_clock) begin
//Internal Registers or wire declarations
                                                  if (i_reset)
reg [2:0] state, next_state;
                                                     o_led <= 0;
                                                  else begin
//Reset Condition
                                                     if (~i_btn & (state == S3))
always@(posedge i clock) begin
                                                         o_led <= 1'b1;
    if(i_reset)
        state<=3'b000;
                                                         o_led <= 1'b0;
                                                  end
        state<=next_state;
                                               end
end
                                               endmodule
```

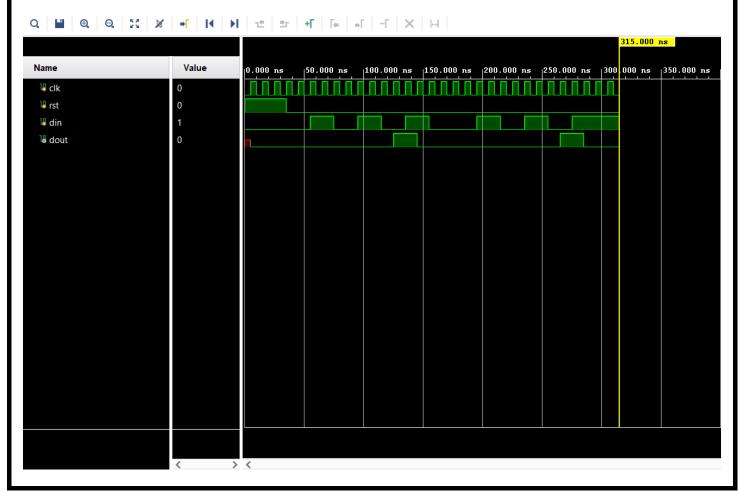
### Testbench Code

```
`timescale 1ns / 1ps
module seq_1010_tb;
 //Internal Registers/Wires
 reg clk, rst, din;
 wire dout;
   #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
```

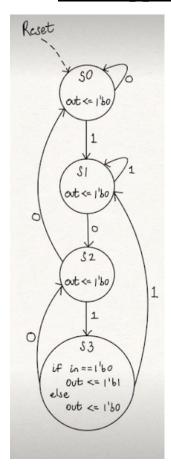
```
//initial begin
// $dumpfile("dump.vod");
// $dumpvars(0);
// end

//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
                                             always@(*) begin
                                                 //Store the state
module seq 1010 mealy(
                                                 next_state = state;
   //Global Signals
   input i_clock,
                                                 //State machine
    input i_reset,
                                                 case (state)
                                                     S0: next_state <= i_btn ? S1:S0;
    //Button
                                                    S1: next state <= i btn ? S1:S2;
    input i_btn,
                                                    S2: next_state <= i_btn ? S3:S0;
    //LED
                                                    S3: next state <= i btn ? S1:S0; //Non overlapping
    output reg o_led
                                                     S3: next_state <= i_btn ? S1:S2; //Overlapping
                                                 endcase
//Local Parameters
localparam [2:0] S0=0, S1=1, S2=2, S3=3;
                                             always@(posedge i_clock) begin
//Internal Registers or wire declarations
                                                 if (i reset)
reg [2:0] state, next state;
                                                     o_led <= 0;
                                                 else begin
//Reset Condition
                                                     if (~i_btn & (state == S3))
always@(posedge i_clock) begin
                                                        o_led <= 1'b1;
                                                     else
    if(i reset)
                                                        o led <= 1'b0;
        state<=3'b000;
                                                 end
    else
                                             end
        state<=next_state;
                                             endmodule
end
```

### 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

