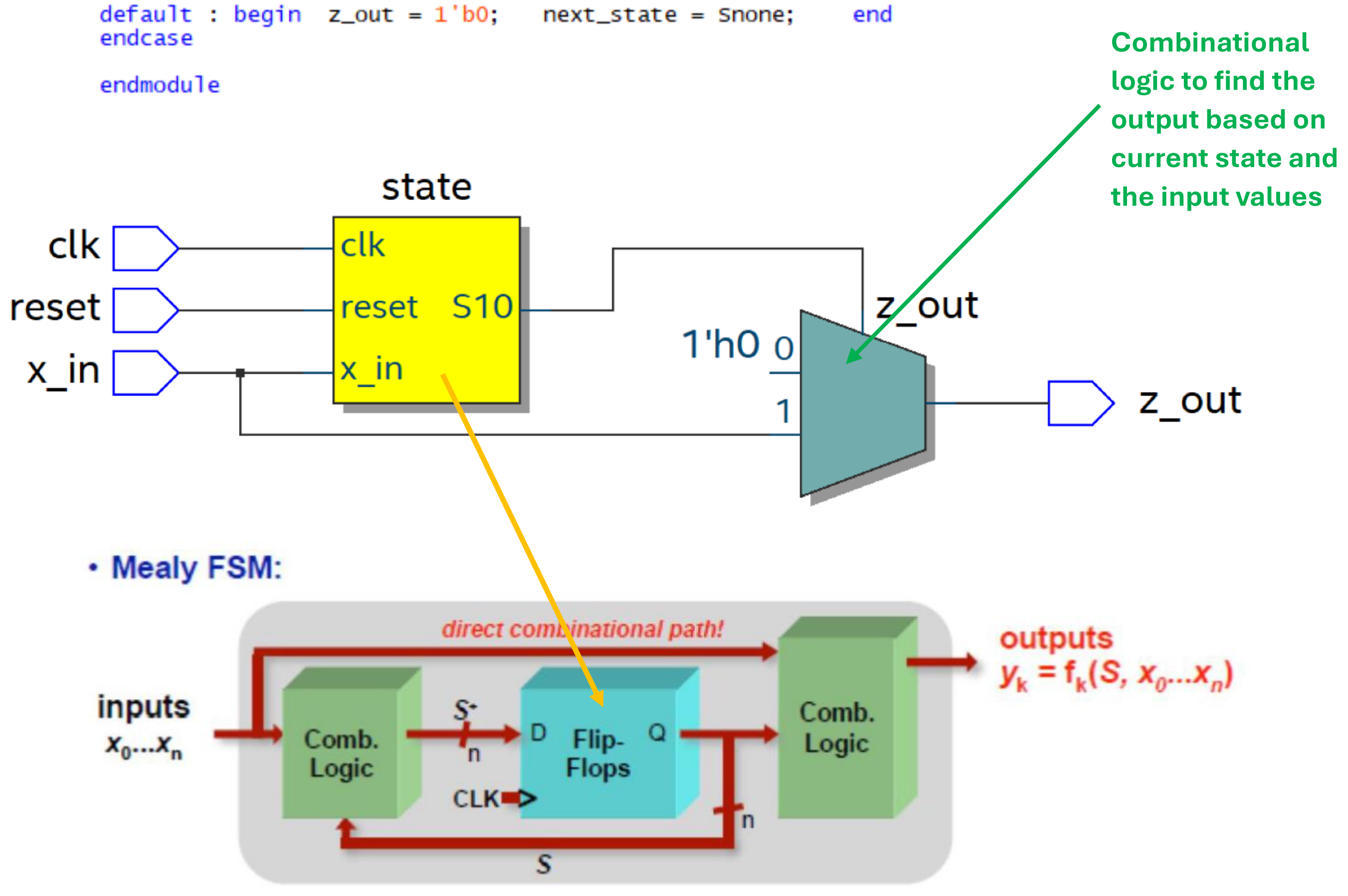
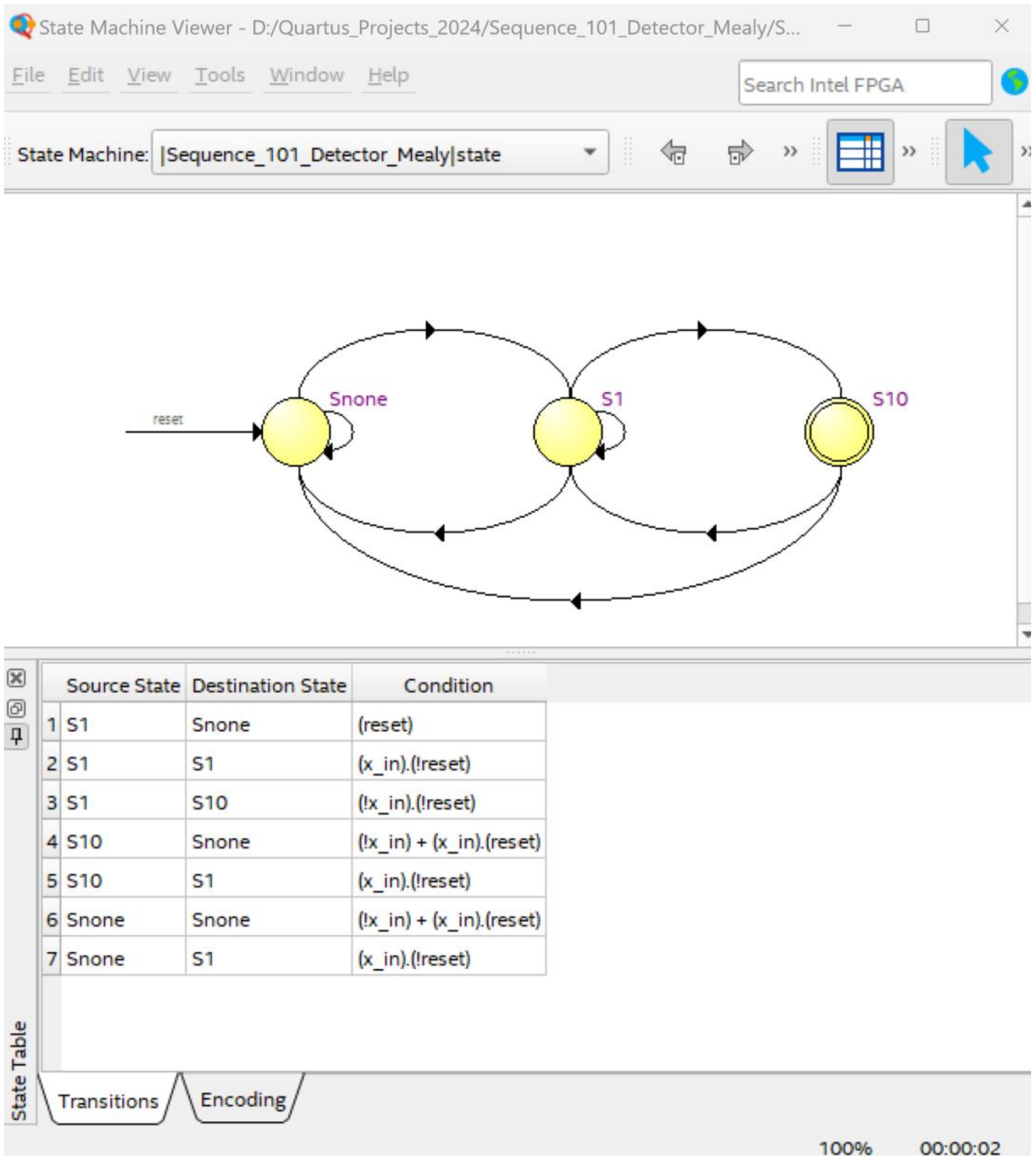
Example Design: Sequence Detector – Mealy FSM – Video 2

```
/*-----------------------------------------------------------------------------------------
  The module receives a single bit input x_in. Whenever x_in performs a sequence 101
  the z_out flag is raised high indicating that 101 was detected. The sequences can be
  overlapping. The design is performed using a Mealy FSM.
  The explanation of the FSM state diagram is given in video 2.
-----------------------------------------------------------------------------------------  */

  module Sequence_101_Detector_Mealy (z_out, clk, reset, x_in);

  output reg z_out;
  input      clk, reset;
  input      x_in;

  reg [1:0] state, next_state;

  parameter Snone = 2'b00;    // None of the correct sequence bits received
  parameter S1    = 2'b01;    // A one was received
  parameter S10   = 2'b10;    // A 10 sequence was received

  // sequential logic (flip flop) updating the state register
  always @ (posedge clk)
    if (reset)  state <= Snone;
    else        state <= next_state;


 // combinational logic determining the next_state and the output
 always @ *
   case (state)
   Snone    : begin z_out = 1'b0;
                if  (x_in)  next_state = S1;
                else        next_state = Snone;
              end
   S1       : begin z_out = 1'b0;
                if (x_in)   next_state = S1;
                else        next_state = S10;
              end
   S10      : begin
                if (x_in) begin   z_out = 1'b1; next_state = S1;    end
                else      begin   z_out = 1'b0; next_state = Snone; end
              end
   default  : begin  z_out = 1'b0;   next_state = Snone;    end
   endcase

   endmodule
```

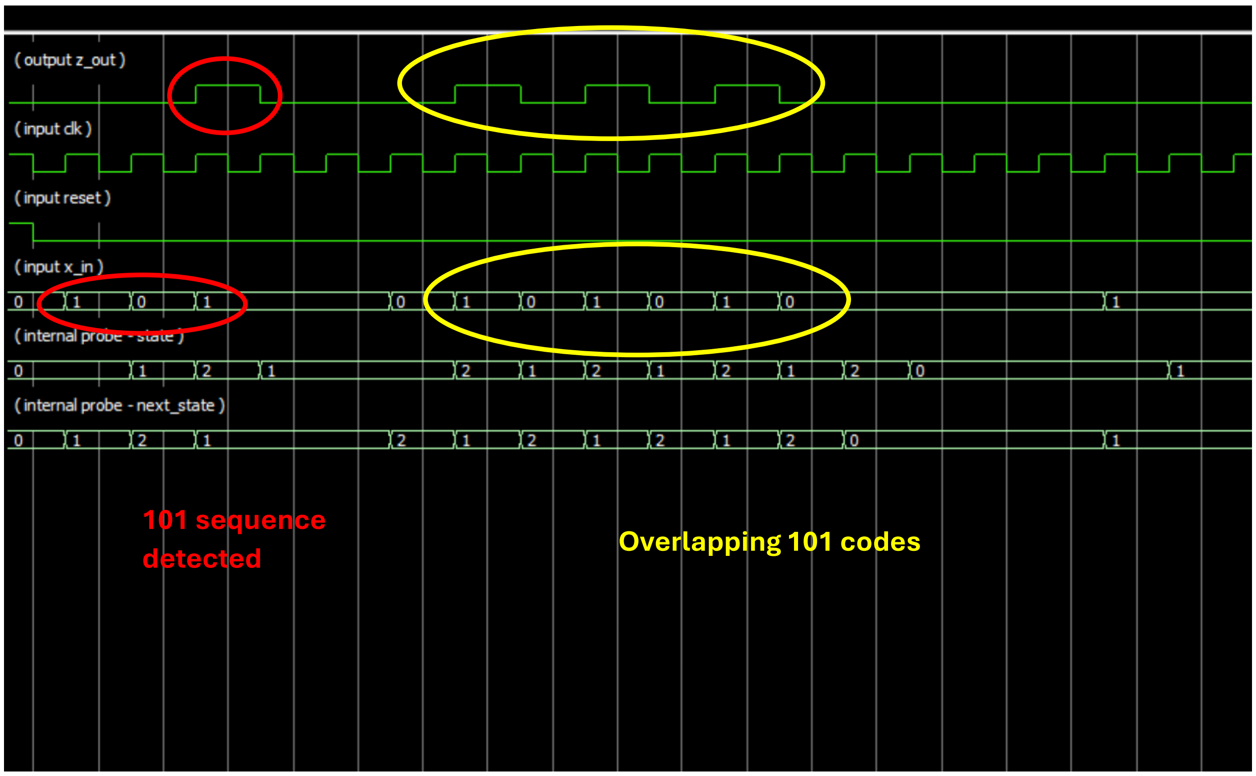

Combinational logic to find the output based on current state and the input values
state
clk
reset
x_in
clk
reset
S10
x_in
z_out
1'h0
0
1
z_out
• Mealy FSM:
direct combinational path!
inputs
$x_0 \ldots x_n$
Comb. Logic
S^+
n
D
Flip-Flops
Q
CLK
n
S
Comb. Logic
outputs
$y_k = f_k(S, x_0 \ldots x_n)$

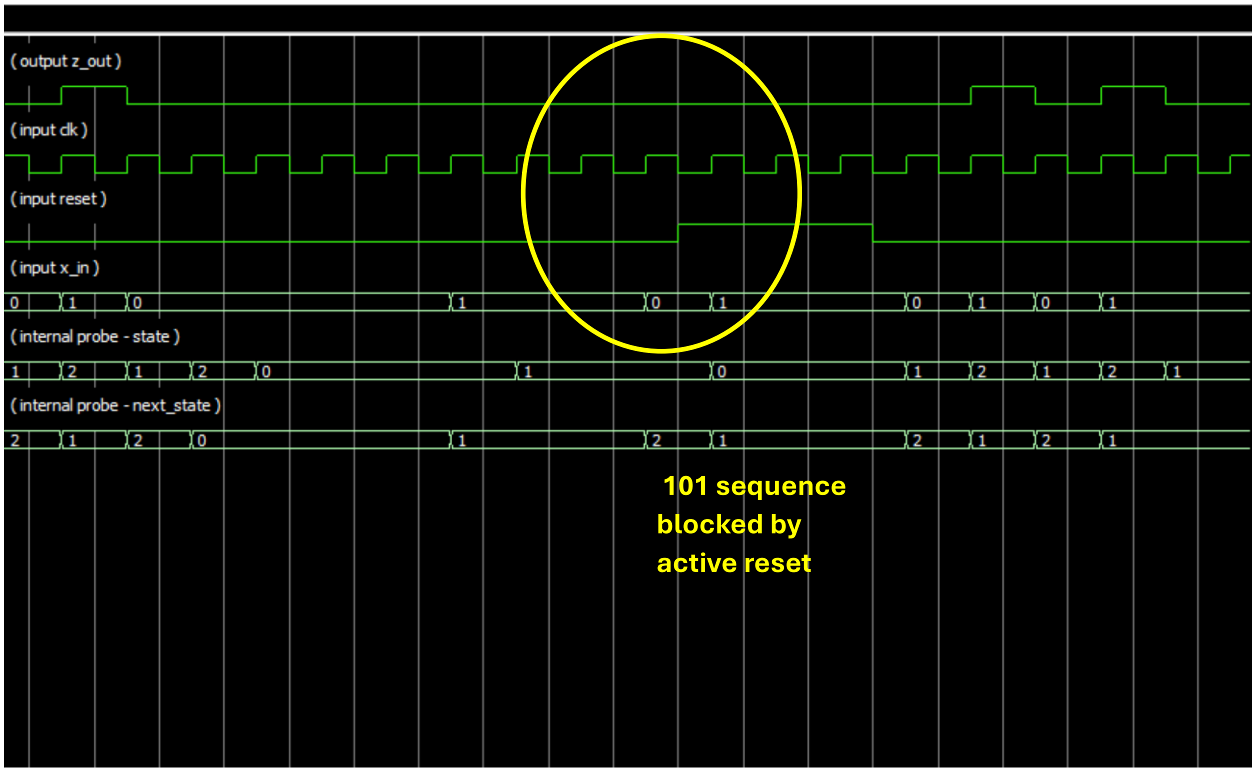
To obtain and verify the FSM state graph, go to **Tools**, **Netlist Viewer**, **State Machine Viewer**, or double click on the flip flop register in the RTL viewer.


State Machine Viewer - D:/Quartus_Projects_2024/Sequence_101_Detector_Mealy/S...
File Edit View Tools Window Help
Search Intel FPGA
State Machine: |Sequence_101_Detector_Mealy|state
reset
Snone
S1
S10
State Table
Transitions
Encoding
100%
00:00:02

	Source State	Destination State	Condition
1	S1	Snone	(reset)
2	S1	S1	(x_in).(!reset)
3	S1	S10	(!x_in).(!reset)
4	S10	Snone	(!x_in) + (x_in).(reset)
5	S10	S1	(x_in).(!reset)
6	Snone	Snone	(!x_in) + (x_in).(reset)
7	Snone	S1	(x_in).(!reset)

```
module Sequence_101_Detector_Mealy_TB ();

wire    z_out;
reg     clk, reset;
reg     x_in;

wire [1:0] state, next_state;   // internal probes

Sequence_101_Detector_Mealy UUT (z_out, clk, reset, x_in);

// internal probes to make it easy to track the logic and for troubleshooting
assign state = UUT.state;
assign next_state = UUT.next_state;

initial
begin
clk = 1'b0;
forever begin
    #5 clk = ~clk;  end
end

initial
begin
reset = 1'b1;
# 30 reset = 1'b0;
#200 reset = 1'b1;
# 30 reset = 1'b0;
end


 initial
 begin
     x_in = 1'b0;      #15

     forever begin

     x_in = 1'b1;  #10  x_in = 1'b0;  #10  x_in = 1'b1;  #10;  // 101 sequence
     x_in = 1'b0;  #10  x_in = 1'b1;  #10  x_in = 1'b1;  #10;  // 011
     x_in = 1'b1;  #10  x_in = 1'b0;  #10  x_in = 1'b1;  #10;  // 101 sequence
     x_in = 1'b0;  #10  x_in = 1'b1;  #10  x_in = 1'b0;  #10;  // overlapping 101
     x_in = 1'b1;  #10  x_in = 1'b0;  #10  x_in = 1'b0;  #10;  // overlapping
     x_in = 1'b0;  #10  x_in = 1'b0;  #10  x_in = 1'b0;  #10;  // 000
     x_in = 1'b1;  #10  x_in = 1'b1;  #10  x_in = 1'b1;  #10;  // 111
     x_in = 1'b0;  #10  x_in = 1'b1;  #10  x_in = 1'b1;  #10;  // 1..011
     end
 end
endmodule
```


(output z_out)
(input clk)
(input reset)
(input x_in)
(internal probe - state)
(internal probe - next_state)
101 sequence detected
Overlapping 101 codes


(output z_out)
(input clk)
(input reset)
(input x_in)
(internal probe - state)
(internal probe - next_state)
101 sequence blocked by active reset