EXPERIMENT - 8

■ AIM - Design and simulation of a final state machine in Verilog to detect a given sequence of bits.

■ COMPONENT / SOFTWARE USED

Component / Software	Specification
• ICs	-
• Breadboard, power supply, LEDs Resistors, Switches, Connecting wires	-
• Softwares used	Vivado 2016.1

■ THEORY

A finite state machine (FSM) is a sequential logic circuit with a finite set of defined states, utilizing memory to store its current state. Combinational logic processess the present state and inputs to determine the subsequent state. For instance, a simple FSM could be a counter that increments with each clock cycle.

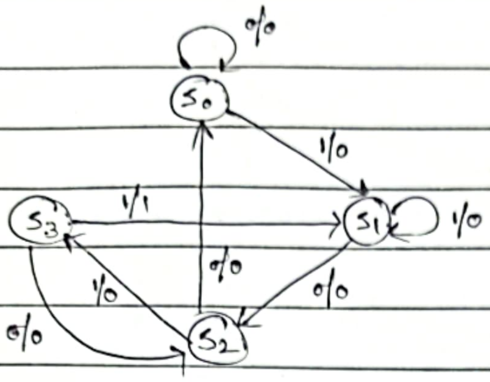
In more complex scenarios, like computers, FSMs handle diverse inputs (keyboard, network, mouse, memory) and numerous states associated with program addresses in memory. However for simplicity in this context, we'll focus on state machines resembling the described counter rather than intricate computer systems.

These state machines extend beyond computers and can represent various processes with predictable algorithms. Two prevalent design approaches are mealy and moore. In Mealy machines, the output depends on both the present state and input, while in Moore machines, the output relies solely on the present state

In a Moore model, outputs synchronize with the clock through flip flops. In Mealy models, outputs can change during the clock cycle based on input changes, potentially causing momentary false values. To synchronise a Mealy circuit, inputs are synchronised with the clock and outputs are sampled just before the clock just before by changing inputs at the inactive clock edge to ensure stability before the active edge.

To design a state machine, one must identify system inputs, states and transitions, typically depicted in a state transition table. The diagram serves as a basis for creating a truth table, incorporating system inputs and current state values. The truth table output corresponding to the next state. Combinational logic is then employed to implement the functions necessary for determining next state values. Boolean logic minimization techniques are applied to optimize these these functions.

Sequence detector is a sequential machine that generates an output 1 every time the desired sequence is detected and output 0 at rest all other times. In this experiment a sequence detector for bit sequence '1011' is designed. The input is represented with Din and output with Dout. The design of the sequence detector is illustrated below using both the methods.



Mealy State Machine for detecting a sequence of 1011

- When in initial state (S0) the machine gets the input of '1' it jumps to the next state with the output equal to '0'. If the input is '0' it stays in the same state.

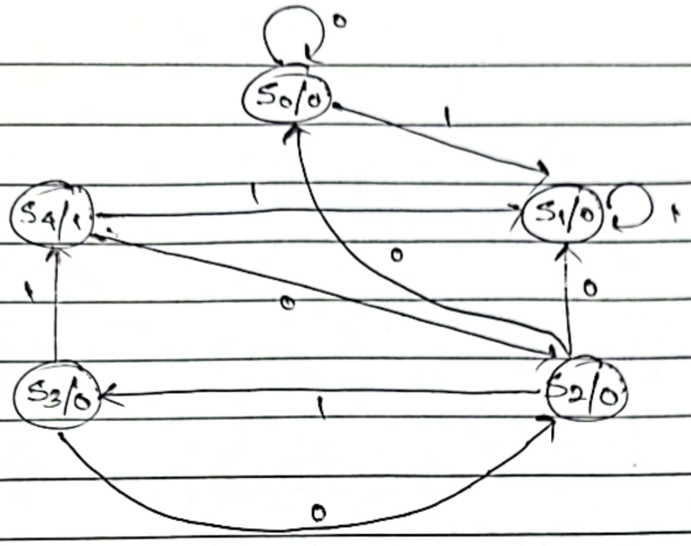
- When in 2nd state (S1), the machine gets an output of '0' it inputs to the 3rd state with the output equal to 0. If it gets an output of 1 it stays in the same state.

- When in 3rd state (S2), the machine gets an input of '1' it jumps of the 4th state with the output equal to 0. If the input received is '0' it goes back to the initial state

- When in 4th state (S3), the machine gets an input of '1' it jumps back to the 2nd state, with the output equal to 1. If the input received is '0', it the input received is 0 it goes back to the 3rd state.

As there are no relevant states thus the final states are S0, S1, S2 and S3. There four states therefore two state variables are required. Binary values are assigned to the states arbitarily.

Teacher's Signature :



Moore State Machine for detecting a sequence of 1011

- In initial state (S0) the output of the detector is 0. When output machine gets the input of 1 it jumps to the next state. If the input is 0 it stays in the same state.

- In 2nd state (S1) the output of the detector is 0. When machine gets an input of 0 it jumps to the 3rd state. If it gets an input of 1, it stays in the same state.

- In 3rd state (S2) the output of the detector is 0. When machine gets an input of 1, it jumps to the 4th state. If the input received is 0, it goes back to the initial state.

- In 4th state (S3) the output of the detector is 0. When machine gets an input of 1, it jumps to the 5th state. If the input received is 0, it goes back to the 3rd state.

Teacher's Signature :

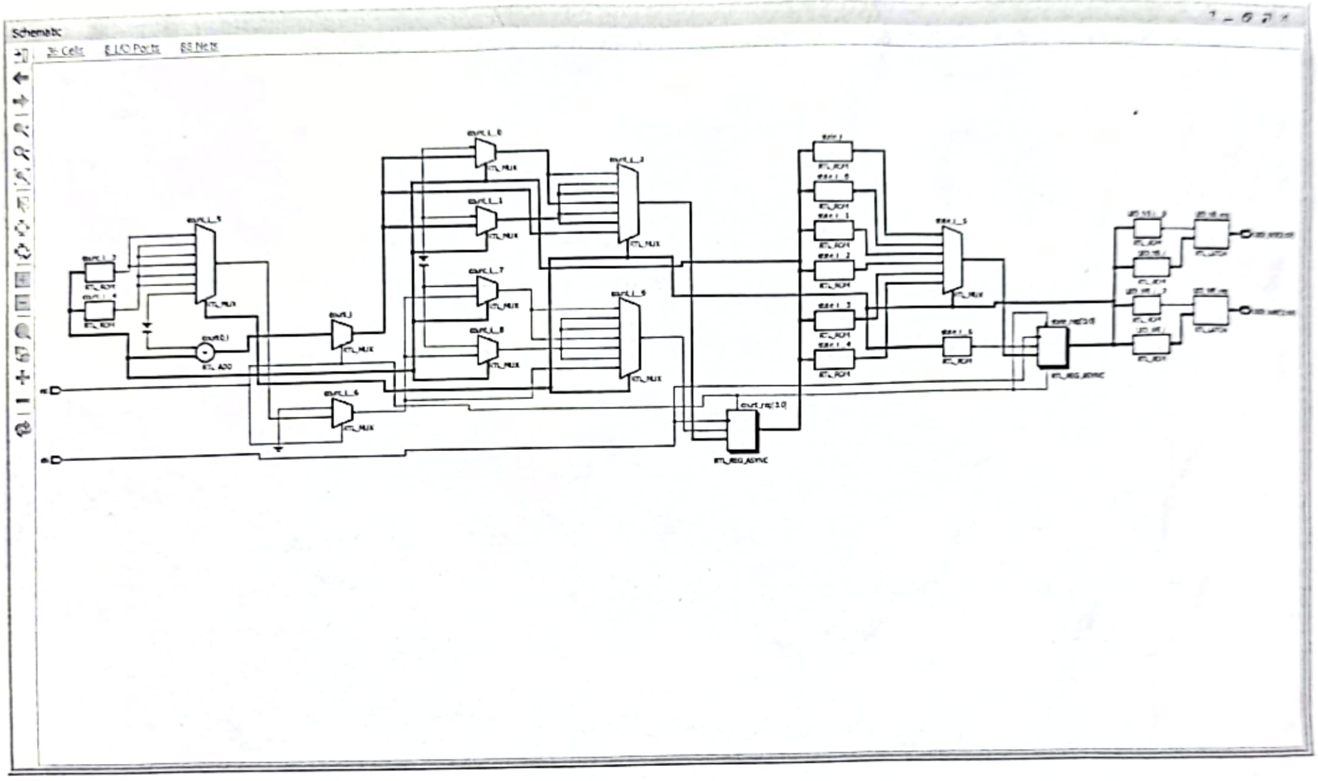
- In the 5th state, the output of the detector is 1, when machine gets an input of 0 it jumps to the 3rd state, otherwise it jumps to the 2nd state.

Present State	Next State		Output
	Din=0	Din=1	Dout
S_0	S_0	S_1	0
S_1	S_2	S_1	0
S_2	S_0	S_3	0
S_3	S_2	S_4	0
S_4	S_2	S_1	1

Moore State table for detecting a sequence of 1011

Design Source Code For Sequence Detector 1011

```
module Seq_Det (input clk, input rst, input Din, output Dout);
reg [1:0] state;
initial
begin
state <= 2'b00;
end
always @ (posedge clk)
begin
if (rst)
state <= 2'b00;
else
begin
case ({state,Din})
3'b000: begin
```



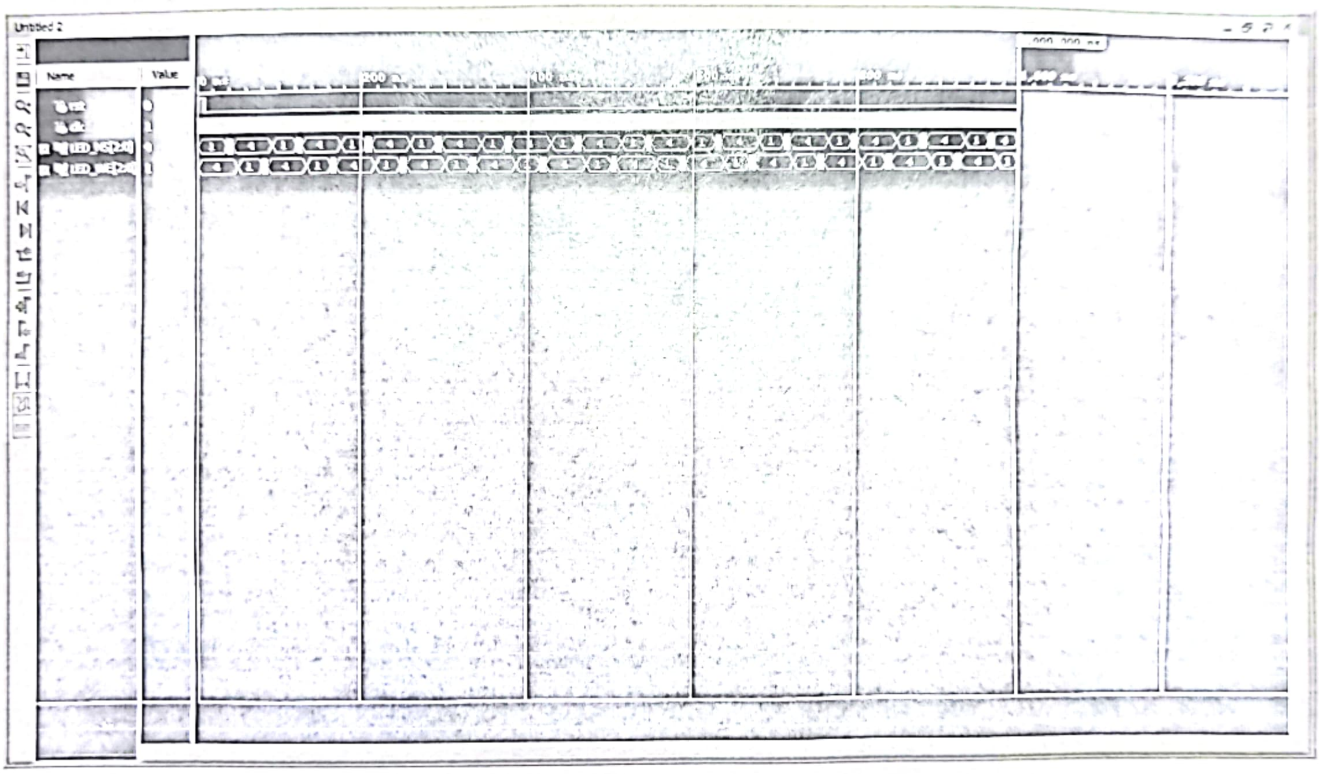
Schematic Representation of Sequence Detector 1011

Date ....

Expt. No. ....

```
        state <= 2'b00;
        end
        3'b001: begin
        state <= 2'b01;
        end
        3'b010: begin
         state <= 2'b10;
        end
       3'b011: begin
       state <= 2'b01;
        end
       3'b100: begin
        state <= 2'b10;
        end
       3'b101: begin
         state <= 2'b11;
         end
       3'b110: begin
          state <= 2'b10;
          end
       3'b111: begin
           state <= 2'b01;
           end
         endcase
       end
       end
  assign Dout = (({state, Din}) == 3'b111) ? 1'b1 : 1'b0;
endmodule
```

Teacher's Signature : ____________



Waveform of Sequence Detector 1011

Testbench code for Sequence Detector "1011"

```
module test_Seq_Det();
 reg clk, rst, Din;
 wire Dout;

 Seq_Det det1 (clk, rst, Din, Dout);

 initial
   begin
   clk = 0;
   forever #10 clk = ~clk;
  end

   initial
  begin
    rst = 0;
   Din = 0; #20
   Din = 1; #20
   Din = 0; #20
   Din = 1 ; #20
   Din = 1 ; #20
   Din = 0 ; #20
   Din = 1 ; #20
   Din = 1 ; #20
   Din = 0; #20
   Din = 0; #20
   Din = 0 ; #20
   Din = 1; #20
```

Teacher's Signature : ________________

Date ....

Expt. No. ....   Page No. 60

```
$finish
end
endmodule
```

■ PROCEDURE

(a) Create a module with required number of variables and mention its input/output.
(b) Write the behavioural description of sequence detector circuit
(c) Synthesize to create RTL Schematic
(d) Create another module referred as test bench to verify the functionality and to obtain the waveform of input and output.
(e) Follow the steps required to simulate the design and compare the obtained output with the corresponding truth table.
(f) Take the screenshots of the RTL schematic and simulated waveforms.

■ CONCLUSION

Design and simulation of a finite state machine in Verilog to detect a given sequence of bits.

Teacher's Signature : ____________