An

Project report on

Sequence Detector(111)

Using Verilog

1. Introduction

Finite State Machine (FSM) is widely used in digital design to detect specific bit patterns. In this project, a Moore FSM is implemented using Verilog to detect the sequence "111" in a serial input stream. The design is simulated and verified using a testbench, and RTL schematic is generated to confirm correct synthesis.

2. Objective

- To design a **non-overlapping sequence detector** for detecting the bit sequence `111`.
- To implement the FSM using **Verilog HDL**.
- To simulate the design using a testbench.
- To verify the results with **waveform output** and **RTL schematic**.

3.Theory:

A sequence detector can be designed using **FSM (Finite State Machine)**. Two common types of FSM are **Moore** and **Mealy**. In a **Moore machine**, the output depends only on the current state.

For detecting `111`, four states are used:

S0 → No `1` detected

S1 → One `1` detected

S2 → Two consecutive `1`s detected

S3 → Three consecutive `1`s detected → output `z=1`

Since it is a **non-overlapping detector**, once `111` is detected, the FSM resets back to `S0`.

4. Design Methodology

- he FSM is described using **Verilog HDL**.
- `always` block with `posedge clk` is used for **state transition**.
- `case` statement is used to describe **next state logic**.
- Output `z` is asserted `1` only in **S3** state

3.2 Verilog Code (RTL):

Project Summary | Schematic | Sequence_detector.v | sequence_detector_tb.v

K:/sequence_detector_2/sequence_detector_2.srcs/sources_1/new/Sequence_detector.v

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 20.08.2025 22:42:41
// Design Name:
// Module Name: Sequence_detector
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Sequence_detector(
    input clk,x,reset,
    output reg z
    );
     parameter S0=2'b00,  // No '1'
               S1=2'b01,  // 1 consecutive '1'
               S2=2'b10,  // 2 consecutive '1's
               S3=2'b11;  // 3 consecutive '1's detected

```

Project Summary | Schematic | Sequence_detector.v | sequence_detector_tb.v

K:/sequence_detector_2/sequence_detector_2.srcs/sources_1/new/Sequence_detector.v

```verilog

    reg [1:0] state, next_state;

    // State transition
    always @(posedge clk or posedge reset) begin
        if (reset)
            state <= S0;
        else
            state <= next_state;
    end

    // Next state logic
    always @(*) begin
        case(state)
            S0: next_state = (x) ? S1 : S0;
            S1: next_state = (x) ? S2 : S0;
            S2: next_state = (x) ? S3 : S0;
            S3: next_state = (x) ? S0 : S0;  // non-overlap -> reset
            default: next_state = S0;
        endcase
    end

    // Output logic (Moore FSM)
    always @(*) begin
        case(state)
            S3: z = 1;
            default: z = 0;
        endcase
    end
endmodule
```

4. Testbench Code:

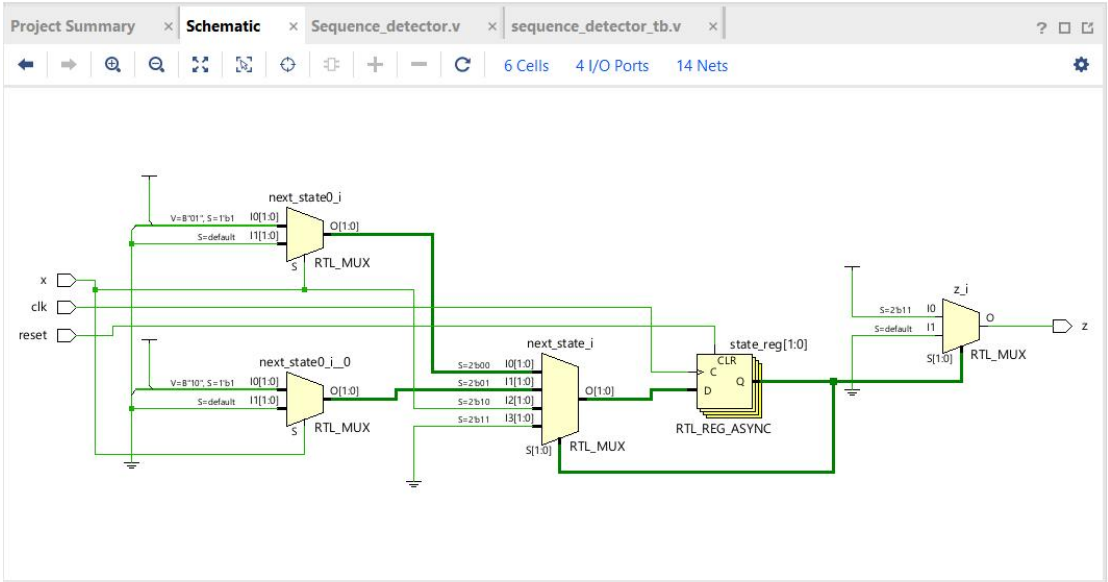
Project Summary | Schematic | Sequence_detector.v | sequence_detector_tb.v

K:/sequence_detector_2/sequence_detector_2.srcs/sim_1/new/sequence_detector_tb.v

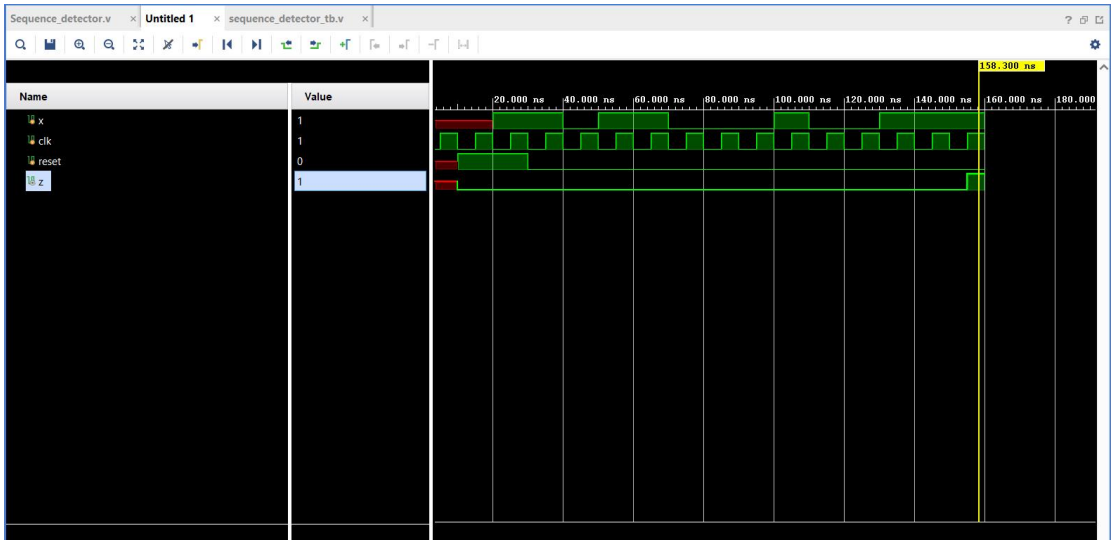
```
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module sequence_detector_tb;
reg x,clk,reset;
wire z;
initial
clk=1'b0;
always #5 clk=~clk;
Sequence_detector dut(.x(x), .clk(clk), .reset(reset), .z(z));
initial begin
#10 reset=1;
#10 x=1;
#10reset=0;
#10 x=0; #10 x=1 ;#10 x=1; #10 x=0;
#10 x=0; #10 x=0 ;#10 x=1; #10 x=0;
#10 x=0; #10 x=1 ;#10 x=1; #10 x=1;
#10 $finish;
end

endmodule
```

Tcl Console | Messages | Log | Reports | Design Runs

5.2 RTL Schematic:



5.3 Simulation Result:



6. Conclusion

The design of a non-overlapping sequence detector for 111 was successfully implemented in Verilog. Simulation verified that the detector works correctly and generates output z=1 whenever the sequence 111 is encountered. This project demonstrates the application of FSM in digital circuit design using Verilog HDL.