END SEMESTER ASSESSMENT (ESA)
B.TECH. (CSE)
III SEMESTER

UE22CS251A – DIGITAL DESIGN & COMPUTER ORGANIZATION LABORATORY

PROJECT REPORT

ON

3 BIT UP/DOWN COUNTER

SUBMITTED BY

NAME	SRN
1) Shreya S	PES2UG22CS534
2) Shashank Bakshi	PES2UG22CS518
3) Shreya Madhusudan	PES2UG22CS533

AUGUST – DECEMBER 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

ELECTRONIC CITY CAMPUS,

BENGALURU – 560100, KARNATAKA, INDIA

TABLE OF CONTENTS		
Sl.No	TOPIC	PAGE No
1.	ABSTRACT OF THE PROJECT	
2.	CIRCUIT DIAGRAM	
3.	MAIN VERILOG CODE	
4.	TEST BENCH FILE	
5.	SCREEN SHOTS OF THE OUTPUT	

ABSTRACT OF THE PROJECT:

Abstract:

This project focuses on the design and implementation of a 3-bit up/down ring counter using the Verilog hardware description language. A ring counter is a type of digital counter that cycles through a sequence of states in a circular manner. The counter developed in this project is capable of counting both upwards and downwards, providing versatility in its applications.

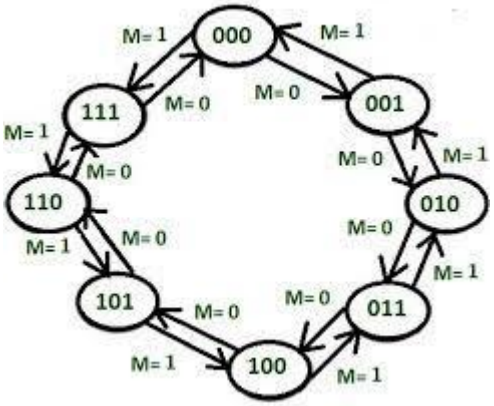
The primary objective of this project is to create a 3-bit ring counter that counts upwards and downwards based on user-defined conditions. The implementation involves the use of Verilog to describe the counter's behavior, including the definition of finite states through a Finite State Machine (FSM).

The project employs a systematic approach to design, beginning with the definition of the counter's states and transitions using an FSM. The Verilog hardware description language is then utilized to translate the FSM into synthesizable code. The counter's functionality includes counting upwards in a sequential manner and counting downwards by decrementing from the highest state.

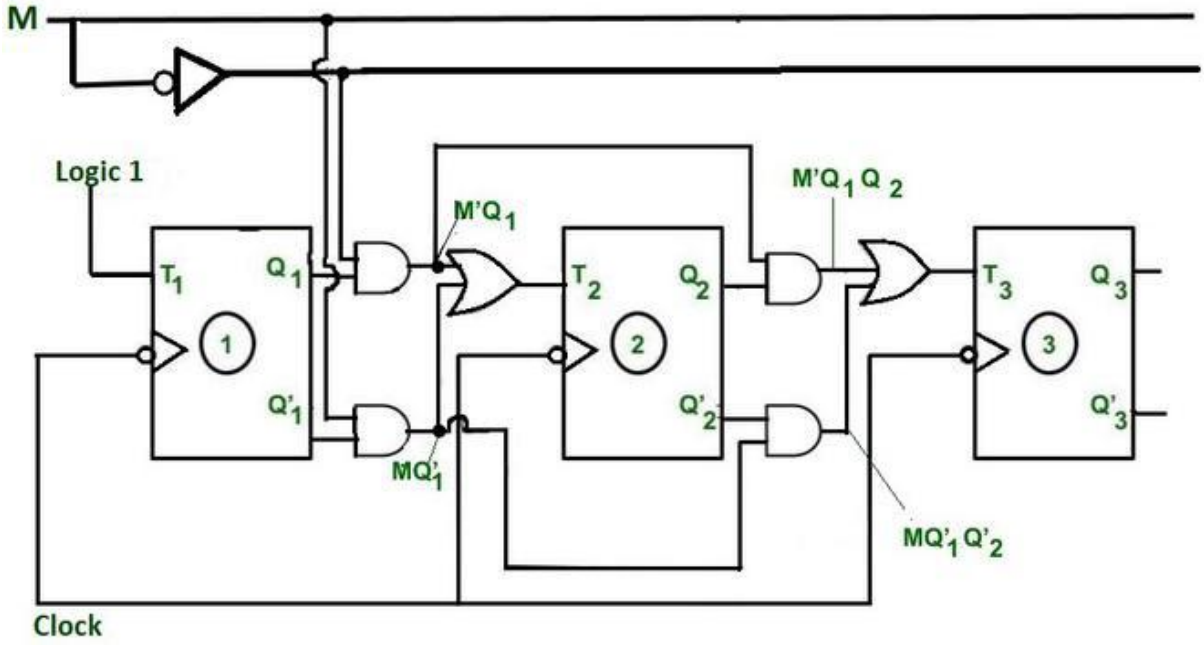
Finite State Machine (FSM): The states of the FSM represent the different values of the counter, and transitions between states are triggered by clock edges and user-defined conditions. By incorporating an FSM, the design ensures a clear, organized, and systematic approach to controlling the counter's operation.

1. **States:**
    - In a 3-bit up/down counter, the states represent the different values the counter can hold. Since it's a 3-bit counter, it can represent values from 0 to 7 (binary 000 to 111).
    - So, there are eight possible states: 000, 001, 010, 011, 100, 101, 110, 111.
2. **Inputs:**
    - The inputs to the FSM are the signals that control the counter's behaviour. In this case, there are typically two inputs:
        - **Count Up (CU):** This input signal increments the counter by 1.
        - **Count Down (CD):** This input signal decrements the counter by 1.
3. **Transitions:**
    - The transitions between states are determined by the inputs. For example:
        - If the current state is 010 and CU is asserted, the next state will be 011.
        - If the current state is 101 and CD is asserted, the next state will be 100.

FSM DIAGRAM



CIRCUIT DIAGRAM:



MAIN VERILOG CODE:

```
module up_counter(input clk, reset, output[3:0] counter);

    reg [3:0] counter_up;                           // Initialise register counter_up to find up
counter states

    always @(posedge clk or posedge reset)          // Postive edge triggered circuit

    begin

        if(reset)

            begin

                counter_up = 4'd0;                  // If reset==1, set counter to 0000

            end

        else

            begin

                counter_up = counter_up + 4'd1;     // Else, add 0001 to counter

            end

    end

    assign counter = counter_up;                    // Return new value of counter

endmodule


module down_counter(input clk, reset, output [3:0] counter);

    reg [3:0] counter_down;                         // Initialise register counter_down to find
down counter states

    always @(posedge clk or posedge reset)          // Postive edge triggered circuit

    begin

        if(reset)

            begin

                counter_down = 4'hf;                // If reset==1, set counter to 1111

            end

        else

            begin

                counter_down = counter_down - 4'd1; // Else, subtract 0001 from counter

            end

    end

    assign counter = counter_down;                  // Return new value of counter

endmodule
```

TEST BENCH FILE:

```
 module counter_testbench();
    reg clk, reset;
    wire [3:0] counter1,counter2;

    up_counter up(clk, reset, counter1);
    down_counter down(clk, reset, counter2);

    initial begin
        $dumpfile("Counter.vcd");
        $dumpvars(0,counter_testbench);
    end
    initial begin
        clk=0;
        repeat(96)
            #5 clk=~clk;
    end
    initial begin
        reset = 1;
        #10
        reset = 0;
    end
endmodule
```

SCREEN SHOT OF THE OUTPUT: