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

UE23CS251A – DIGITAL DESIGN & COMPUTER ORGANIZATION LABORATORY

PROJECT REPORT

ON

"SEQUENCE GENERATOR"
Team No. 11

SUBMITTED BY

NAME	SRN
1) Navyashree SP	PES2UG23CS374
2) Preksha Kamalesh	PES2UG23CS902
3) Monisha Sharma	PES2UG23CS906
4) Nandana Mathew	PES2UG23CS913

AUGUST – DECEMBER 2024

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	3
2.	CIRCUIT DIAGRAM	4
3.	MAIN VERILOG CODE	5
4.	TEST BENCH FILE	5
5.	SCREEN SHOTS OF THE OUTPUT	6-7

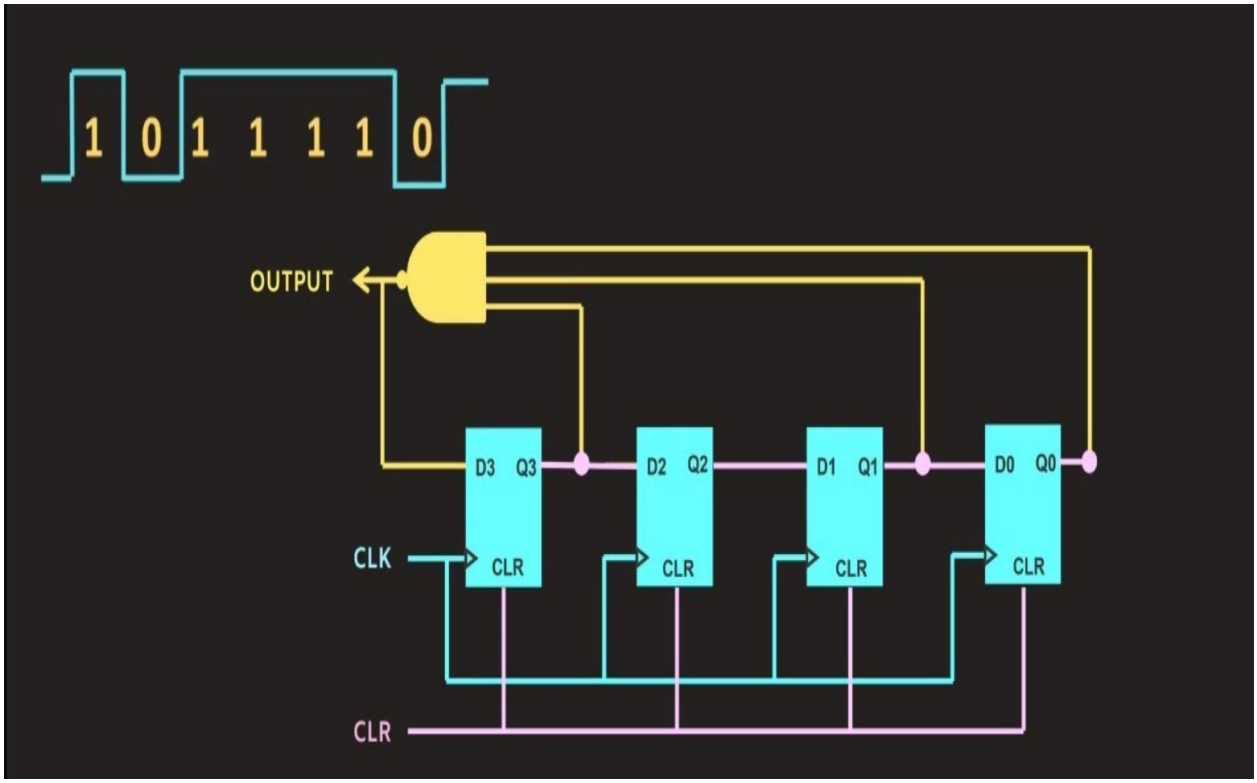
ABSTRACT OF THE PROJECT:

A 4-bit sequence generator using D flip-flops is a sequential digital circuit designed to generate a binary counting sequence from 0000 to 1111 (0 to 15 in decimal) in synchronization with a clock signal. The circuit uses four D flip-flops, where each flip-flop represents one bit of the binary sequence. The D flip-flops are arranged in a chain to form a binary counter, with the output of one flip-flop connected to the input of the next.

The circuit is driven by a clock signal that triggers state changes on each rising edge, making the output increment by 1 with each clock pulse. A reset signal is used to initialize the flip-flops and set the output to 0000, ensuring the counter starts from the beginning of the sequence. As the clock progresses, the circuit produces a continuous binary count, which can be used in applications like timers, counters, or sequence detection.

This design is fundamental in digital electronics, providing a basic yet versatile method for generating binary sequences. The simplicity and efficiency of D flip-flops make this approach ideal for implementing small-scale binary counters, often used in microcontrollers, digital clocks, and other timing-related circuits.

CIRCUIT DIAGRAM:



MAIN VERILOG CODE:

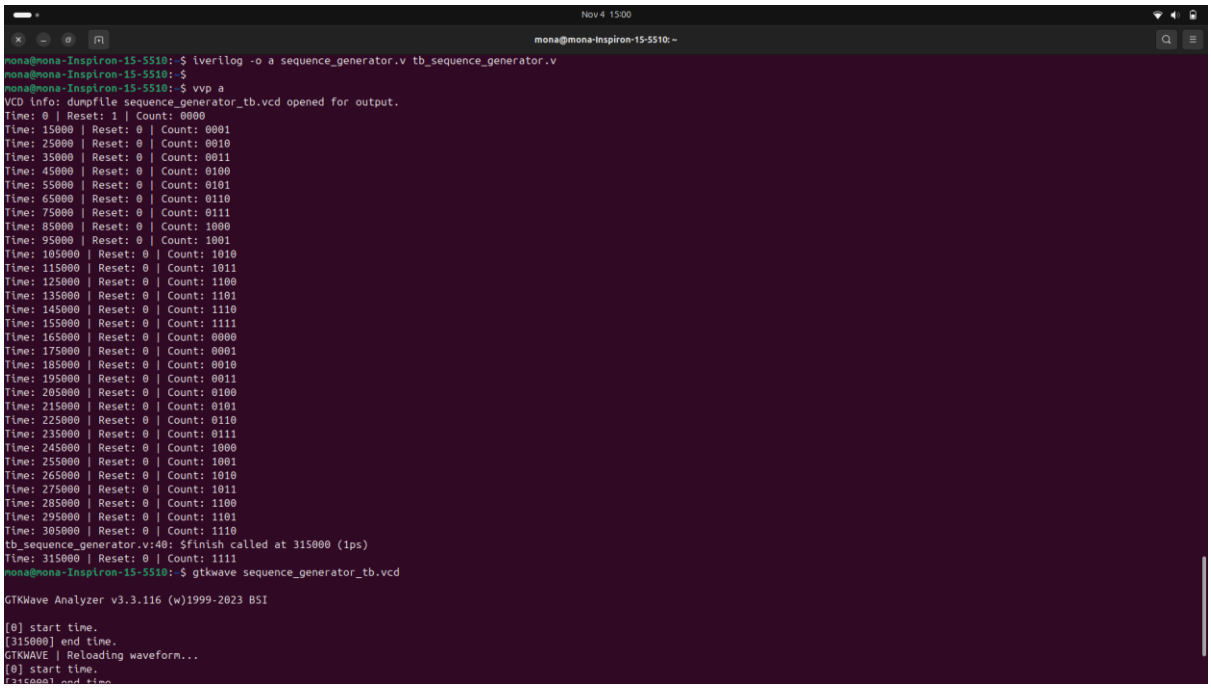
sequence_generator.v

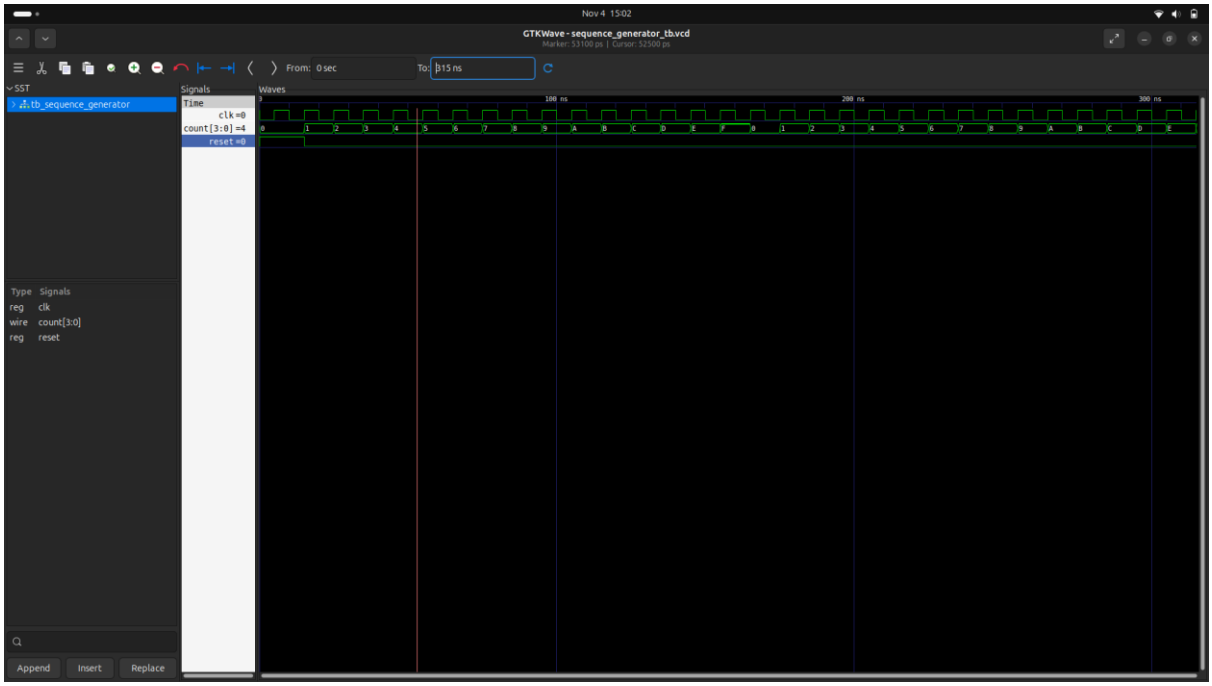
```
module sequence_generator (
                input wire clk,
                input wire reset,
                output reg [3:0] count
            );
                always @(posedge clk or posedge reset) begin
                    if (reset) begin
                        count <= 4'b0000;
                    end else begin
                        count <= count + 1;
                    end
                end
endmodule
```

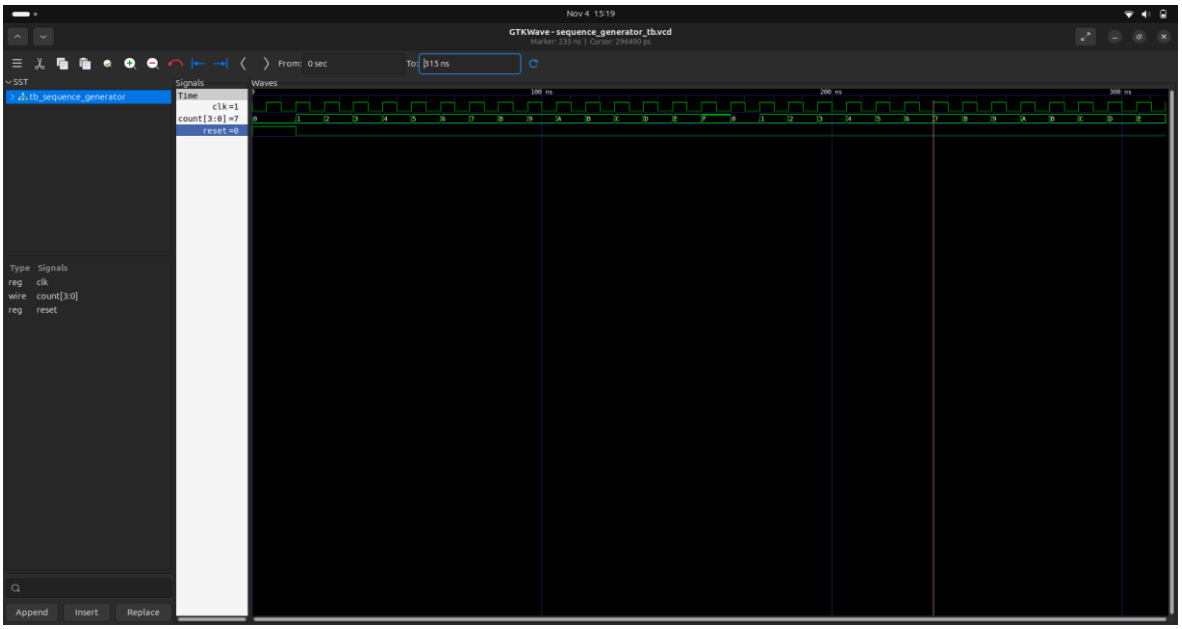
TEST BENCH FILE:

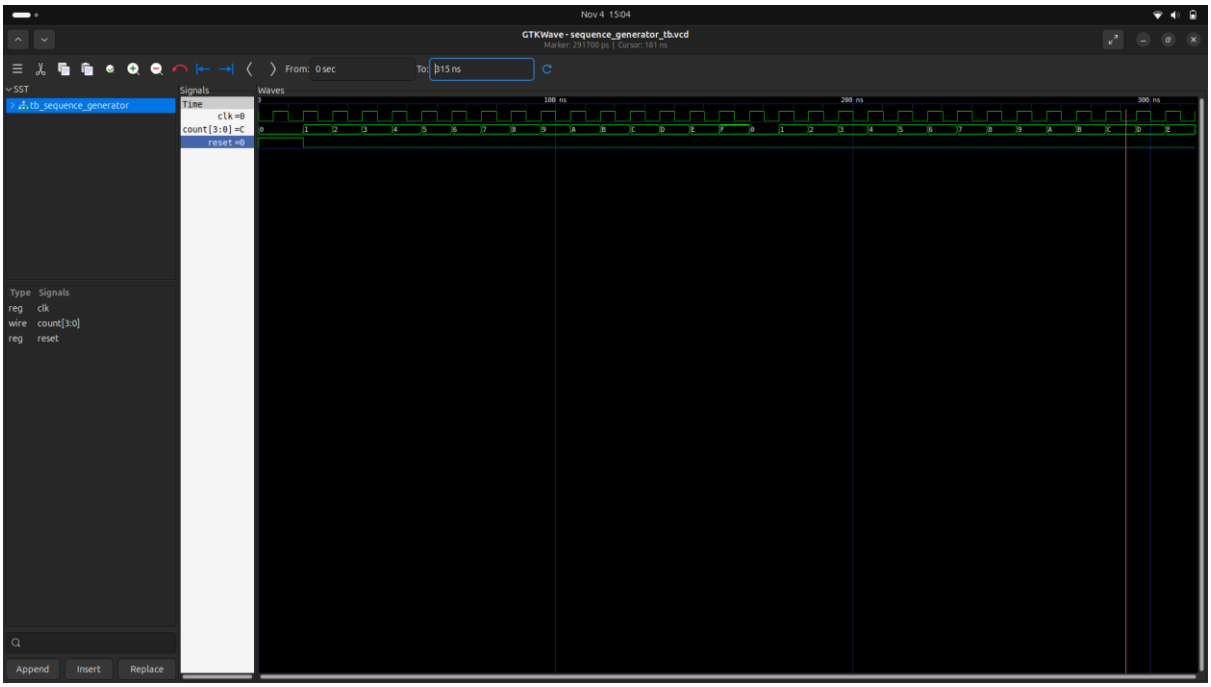
```
`timescale 1ns / 1ps
module tb_sequence_generator;

    reg clk;
    reg reset;
    wire [3:0] count;
    sequence_generator uut (
        .clk(clk),
        .reset(reset),
        .count(count)
    );
    initial begin
        clk = 0;
        forever #5 clk = ~clk;
    end
    initial begin
        $monitor("Time: %0t | Reset: %b | Count: %b", $time, reset, count);
        $dumpfile("sequence_generator_tb.vcd"); // Set output VCD file name
        $dumpvars(0, tb_sequence_generator);    // Dump variables in the current
module
        reset = 1;
        #15;
        reset = 0;
        #300;
        $finish;
    end
endmodule
```

SCREEN SHOT OF THE OUTPUT:






Nov 4 15:19
GTKWave - sequence_generator_tb.vcd
From: 0 sec
To: 315 ns
SST
tb_sequence_generator
Signals
Waves
Time
clk =1
count[3:0] =7
reset =0
100 ns
200 ns
300 ns
Type Signals
reg clk
wire count[3:0]
reg reset
Append
Insert
Replace


Nov 4 15:04
GTKWave - sequence_generator_tb.vcd
Marker: 291700 ps | Cursor: 181 ns
From: 0 sec
To: 315 ns
SST
tb_sequence_generator
Signals
Waves
Time
clk =0
count[3:0] =C
reset =0
100 ns
200 ns
300 ns
Type Signals
reg clk
wire count[3:0]
reg reset
Append
Insert
Replace