https://www.pes.edu/wp-content/uploads/2019/09/pes_logo.png

**FINAL SEMESTER ASSESSMENT (FSA) B.TECH. (CSE)**

**III SEMESTER**

**UE18CS206 – DIGITAL DESIGN & COMPUTER ORGANIZATION LABORATORY**

**PROJECT REPORT**

**ON**

“DESIGN AND IMPLEMENT A SEQUENCE GENERATOR”

SUBMITTED BY

**NAME SRN**

1. **Divyani Das PES2201800059**
2. **Meenal Shah PES2201800018**
3. **Garima Gaur PES2201800295**
4. **Mukthi Prada PES2201800310**

**AUGUST – DECEMBER 2019**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**ELECTRONIC CITY CAMPUS,**

**BENGALURU – 560100, KARNATAKA, INDIA**

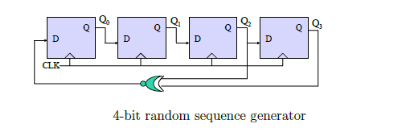
|  |  |  |
| --- | --- | --- |
| TABLE OF CONTENTS | | |
| SL.No | TOPIC | PAGE No |
|  | ABSTRACT OF THE PROJECT | 1 |
|  | CIRCUIT DIAGRAM |  |
|  | MAIN VERILOG CODE |  |
|  | TEST BENCH FILE |  |
|  | SCREEN SHOTS OF THE OUTPUT |  |

ABSTRACT OF THE PROJECT

A sequence generator is a digital logic circuit whose purpose is to produce a prescribed sequence of outputs. Each output will be one of a number of symbols or of binary logic levels. The sequence may be of indefinite length or of predetermined fixed length. A binary counter is a special type of sequence generator. Sequence generators are useful in a wide variety of coding and control applications. Sequence generators are nothing but a set of digital circuits which are designed to result in a specific bit sequence at their output. There are several ways in which these circuits can be designed including those which are based on multiplexers and flip-flops.

In our project, we constructed a four bit sequence generator using four registers s0, s1, s2 and s3. S0 stores the XOR of s1 and s3, while the output which gets printed on the console is s3. The output is fed back to the circuit to create a sequence.

CIRCUIT DIAGRAM



MAIN VERILOG CODE

module sequencegenerator(input clk, reset,output s3);

reg s1, s2, s3;

wire s0;

assign s0 = s1 ^ s3;

//using bitwise exclusive exclusive or to finding the value of s0

// STATE MACHINE

always @ (posedge clk or reset)

begin

// INITIAL STATE SHOULDN'T BE 000 => 100

//here 000 initial input leads to sequence of 000...

//so depending on the input here we can generate the

//required sequence that we wanted

//here input is 100

if(reset) begin//@beginning the input should be 100

s1 <= 1;

s2 <= 0;

s3 <= 0;

end

else begin //else moving the sequence

s1 <= s0;

s2 <= s1;

s3 <= s2;

end

end

//returing the output s3 every time

endmodule

TESTBENCH FILE

`timescale 1ns / 1ps

module Test\_sequencegenerator;

// Inputs

reg clk;

reg reset;

// Outputs

wire s3;

// Instantiate the Sequence Generator Test (SGT)

sequencegenerator SGT (.clk(clk), .reset(reset), .s3(s3));

initial begin

$dumpfile("Sequence.vcd");

$dumpvars;

// Initialize Inputs

clk = 0;

reset = 0;

// Wait 100 ns for global reset to finish

//#100;

// Add stimulus here

#10 reset = 1;

#10 reset = 0;

#200 $finish;

end

always begin

#5 clk = ~clk;

end

//PRINT SEQUENCE

always @ (posedge clk)

$write("%b ",s3);

endmodule

SCREENSHOTS OF THE OUTPUT

