2024

Sequence Generator

MINI PROJECT

DIGITAL DESIGN AND CIRCUIT OPTIMIZATION

TABLE OF CONTENTS

PROJECT TEAM OVERVIEW

TEAM MEMBERS

NAME	SRN
Paddhariya Mohit Kaushik	PES1UG24CS816
C S Dipak	PES1UG23CS907
Chennupati Gundeep	PES1UG23CS160
Bysani Niravann	PES1UG23CS152

TEAM MEMBERS & ROLES

The project was led by **Mohit Paddhariya**, who served as the **Team Leader** and was responsible for most of the development, planning, and execution. Mohit took charge of overseeing the entire project from start to finish, ensuring that all aspects were completed efficiently and effectively.

The remaining team members contributed to supporting roles, assisting with specific tasks as needed:

- **Chennupati Gundeep**, assisting with initial research and basic setup tasks.
- **C S Dipak**, contributing to data collection and minor bug fixing.
- **Bysani Niravann**, helping with documentation and some testing efforts.

PROJECT REPOSITORY

You can explore the project's source code on the official repository:

GitHub Repository Link

INTRODUCTION

In digital design, a sequence generator is a circuit that produces a specific sequence of binary numbers over time. It's useful for creating patterns or signals that are needed in various digital systems, like clocks, counters, or communication devices. This document provides a detailed description and implementation of a 3-bit sequence generator using a simple state machine.

PROBLEM STATEMENT

Design and implement a sequence generator. A sequence generator produces a series of binary outputs based on a defined set of states. In this case, the generator is designed to output a 3-bit sequence that cycles through four predefined states.

MODULE DESCRIPTION

SEQUENCE GENERATOR

This module generates a 3-bit sequence output based on a simple state machine. The state transitions are controlled by a clock signal and can be reset asynchronously. The sequence generated cycles through four predefined states, outputting specific 3-bit values for each state.

CODE

```
module sequence_generator(
   input clk,
   input reset,
   output reg [2:0] seq_out
);
   reg [1:0] state;
   parameter S0 = 3'b101, S1 = 3'b110, S2 = 3'b011, S3 = 3'b001;

   always @(posedge clk or posedge reset) begin
      if (reset)
         state <= 2'b00;
      else
         state <= state + 1;
   end

   always @(state) begin
      case(state)
```

```
        2'b00: seq_out = S0;
        2'b01: seq_out = S1;
        2'b10: seq_out = S2;
        2'b11: seq_out = S3;
        default: seq_out = S0;
      endcase
    end
endmodule
```

TESTBENCH

The testbench module, tb_sequence_generator, simulates the clock and reset signals to verify the functionality of the sequence generator. It includes a clock generation process and initializes the reset signal to check the output sequence.

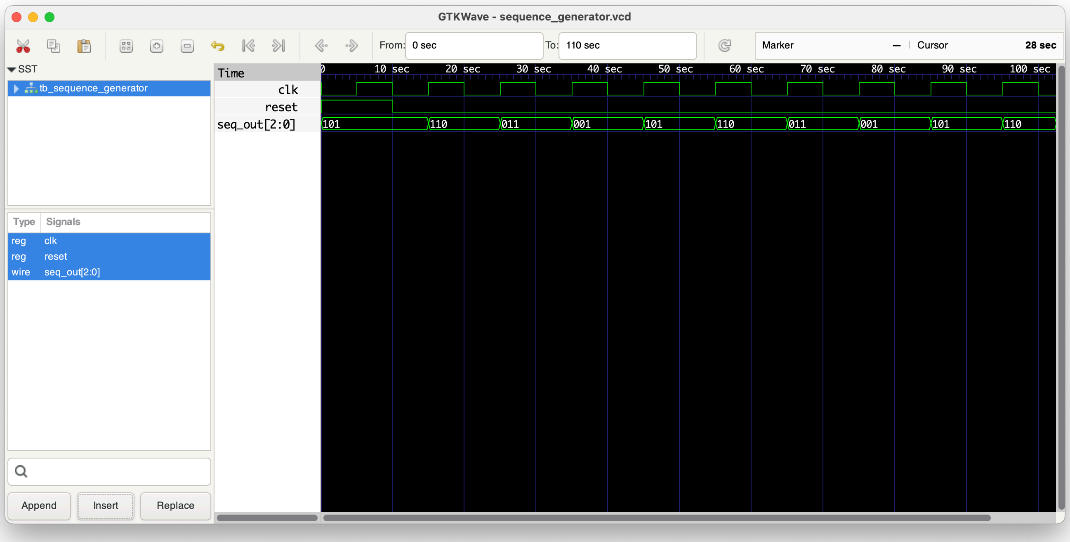
CODE

```
module tb_sequence_generator;
    reg clk;
    reg reset;
    wire [2:0] seq_out;

    sequence_generator uut (.clk(clk), .reset(reset), .seq_out(seq_out));

    initial begin
      clk = 0;
      forever #5 clk = ~clk;
    end

    initial begin
      $dumpfile("sequence_generator.vcd");
      $dumpvars(0, tb_sequence_generator);
      reset = 1; #10; reset = 0;
      #100;
      $finish;
    end
endmodule
```

EXPLANATION

A sequence generator produces a repeating or predictable series of bits, often used in electronics for tasks like timing or data control. It starts from an initial value and transitions between states with a clock signal. The testbench checks the functionality by generating signals and capturing the output.

GTKWAVE OUTPUT


GTKWave - sequence_generator.vcd
From: 0 sec
To: 110 sec
Marker
Cursor 28 sec
SST
tb_sequence_generator
Type Signals
reg clk
reg reset
wire seq_out[2:0]
Append
Insert
Replace
Time
clk
reset
seq_out[2:0]
0
10 sec
20 sec
30 sec
40 sec
50 sec
60 sec
70 sec
80 sec
90 sec
100 sec
101
110
011
001
101
110
011
001
101
110