System Design Document

for the State Machine

with Audio Output

University at Buffalo, The State University at New York

EE478 Fall 2019

The information enclosed in this document and the source code generated for this laboratory assignment were generated by the named students on this cover page. Detection of plagiarism in a submitted lab or project design document will result in an automatic failing grade in the course by all parties involved.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Written By** | Daniel Giovino |  | **Revision History** | | | |
| **Engineer** | Daniel Giovino |  | **Rev** | **Date** | **Session** | **Approved/Grade** |
| **Engineer** | Peter M. VanNostrand |  | Orig | 10/21/19 | Mon. 9AM |  |
| **Teaching Assistant** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | **EE478F19**  **LAB 4** |

# Table of Contents

Table of Contents 2

List of Figures 2

List of Tables 2

1. Introduction 3

1.1 Overview 3

1.2 Document Scope 3

1.3 Intended Audience 3

2. System Design Overview 4

2.1 System Block Diagram and Description 4

2.2 PLL Module 5

2.3 Clock Divider Module 6

2.4 State Machine Module 7

2.5 Square Wave Generator Module 8

3. Testing and Verification 9

3.1 Testbench and Simulation 9

3.2 Objective Verification 9

Glossary 10

List of Abbreviations 10

Hardware References 10

# List of Figures

Figure 1: State Machine with Audio Output System Diagram 4

Figure 2: Clock Divider Circuit 6

Figure 3: Square Wave Generator 8

# List of Tables

Table 1: State Machine Truth Table 7

# Introduction

## Overview

This device uses a simple state machine to control the output notes to an audio codec chip. The output notes cycle through 4 different notes, always in the same order, and only starts the sequence when a pushbutton is pressed. This state machine was first designed on paper and then implemented and tested on a Xilinx FPGA using the Xilinx Vivado CAD tool.

## Document Scope

This document has been written to provide information regarding the state machine with audio output circuit developed in Lab 4 of EE478. The inputs, outputs, and operation of this device are described below as well as details regarding implementation on an FPGA.

## Intended Audience

This document is intended for use by the EE478 TAs and professors for use in grading as well as by the members of group 1L in the Monday 9am lab session.

# System Design Overview

## System Block Diagram and Description

The state machine takes a button input labeled as BTN0 in Figure 1 and the system clock labeled as CLK in Figure 1. The button input is connected to the state machine. The system clock is an input to the PLL, which turns the clock from a frequency of 125 MHz to a frequency of 12.288 MHz named Mclock in Figure 1.

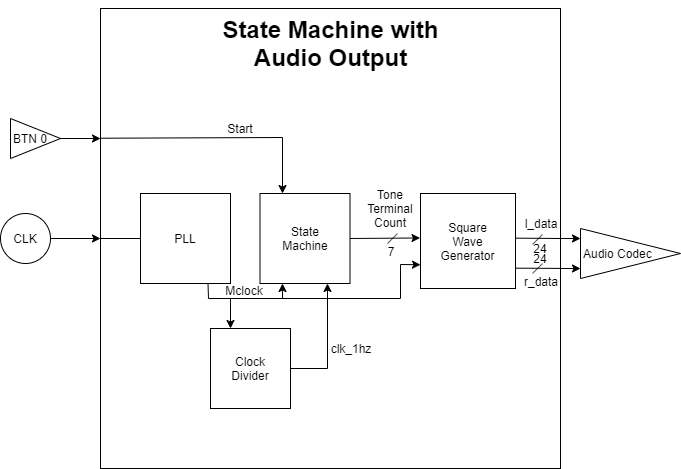


Figure 1: State Machine with Audio Output System Diagram

The clock divider circuit further divides the Mclock signal to a clock with a frequency of 1 Hz with a one-shot output. This 1 Hz clock is used by the state machine in addition to Mclock to produce a varying tone terminal count when the button is pressed. The tone terminal count and Mclock are sent to the square wave generator, which generates two identical square wave outputs l\_data and r\_data with a frequency of tone terminal count. These two square waves are sent to the audio codec which outputs notes with these frequencies.

## PLL Module

This module turns the system clock of frequency 125 MHZ into a clock signal named Mclock of frequency 12.288 MHZ. This module was made using the built-in clocking wizard in Vivado. The wizard multiplies the system clock signal by 23 and divides it by 234 to get the output frequency to almost exactly 12.288 MHz on Mclock. No schematic is given for this module since Vivado makes this circuit and we haven’t yet learned how it does this.

## Clock Divider Module

This module implements a clock divider circuit. It outputs a 1 Hz clock by dividing Mclock by 12,288,000. This is done by counting 12,288,000 cycles of Mclock. Clk\_1hz is a one-shot signal, so it is set to 1 only when the count = 12,288,000 cycles. When clk\_1hz is 1, the count is reset to 0 on the next rising edge of Mclock. This is shown in Figure 2 below.

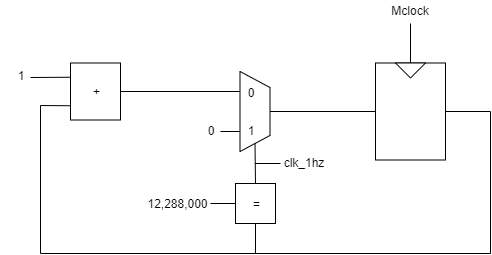


Figure 2: Clock Divider Circuit

## State Machine Module

This module has inputs of the one-shot 1hz clock signal generated in the previous model and the start bit which is ‘1’ when BTN0 is pressed. The tone terminal count is set to 4 different constants shown in Table 1 in states S1-S4. In state S0, the state machine is in standby and sends 0 as the output constant to tell the future modules to standby. When in standby, the state machine moves to state S1 as soon as start = ‘1’. All other state transitions from S1 to S2, S2 to S3, S3 to S4, and S4 to S0 occur when clock\_1hz = ‘1’. This is shown in the table below (X means don’t care).

Table 1: State Machine Truth Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| State | clk\_1hz | Start | Next State | Tone Terminal Count |
| S0 | X | 0 | S0 | 0 |
| S0 | X | 1 | S1 | 0 |
| S1 | 1 | X | S2 | Count\_C5 |
| S2 | 1 | X | S3 | Count\_E5 |
| S3 | 1 | X | S4 | Count\_G5 |
| S4 | 1 | X | S0 | Count\_C6 |

This module was implemented using a case state statement inside a process statement, which means this module uses sequential logic to implement.

## Square Wave Generator Module

This module generates a square wave with a variable period (and thus frequency). The period is controlled by the signal tone terminal count (generated in the state machine module). The counter in this module increments by one every rising edge of Mclock, and resets to 0 whenever the counter = tone terminal count. The outputs of this module, l\_data and r\_data, are both set to all ‘0’s when the counter < (tone terminal count / 2) and set to X"0FFFFF" otherwise. This data is sent to the Audio Codec. All of this is shown below in Figure 3.



Figure 3: Square Wave Generator

# Testing and Verification

To ensure that our circuit functioned as intended we performed testing. The way we tested this code was by synthesized our design and loading it onto an FPGA where we performed testing of the physical circuit. A testbench, which is a good programming practice, was not used in this lab.

## Testbench and Simulation

A testbench was not used in this lab, even though it is a good programming practice. A testbench could have been used to test each module of this design individually.

## Objective Verification

Once we synthesized our design and loaded it onto the FPGA we tested the circuit by plugging earbuds into the black HPH OUT jack on the board. We first listened through the earbuds without pressing BTN 0 to make sure no notes were being played. Then, we pressed the button and listened for 4 unique notes to be played one after another with a one second delay before changing notes. After this, no notes were played until the next button press, as expected, thus verifying the circuit.

# Glossary

This section includes helpful supplemental information to aid in understanding the content of this report.

## List of Abbreviations

Below are selected abbreviations used in this document

* VHDL: VHSIC Hardware Description Language
* FPGA: Field Programmable Gate Array
* CAD: Computer Aided Design
* PLL: Phase-Locked Loop

## Hardware References

* Zybo Z7 Manual ([DigitalInc](https://reference.digilentinc.com/_media/reference/programmable-logic/zybo-z7/zybo-z7_rm.pdf))
* Xilinx Vivado CAD tool ([Xilinx](https://www.xilinx.com/products/design-tools/vivado.html))