System Design Document

for Pong Game

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 | 12/02/19 | Mon. 9AM |  |
| **Teaching Assistant** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  | **EE478F19**  **FINAL LAB** |

# 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

## Pong System Block Diagram and Description

The system takes four button inputs labeled as BTN0, BTN1, BTN2, and BTN3 in Figure 1, the system clock labeled as CLK in Figure 1, and 3 slider switches labeled as SW0, SW1, and SW2 in Figure 1. The button inputs are connected to the paddle controllers. The system clock is an input to the PLL, which turns the clock from a frequency of 125 MHz to a frequency of 74.25 MHz named Pclk in Figure 1. The first switch (SW0) is connected to the max count changer to change the speed the ball travels at. The second switch (SW1) toggles if the paddles shrink when the ball bounces off them or not. The last switch used (SW2) controls if the ball is displayed as a circle or a square, but does not change the way the ball physics work.

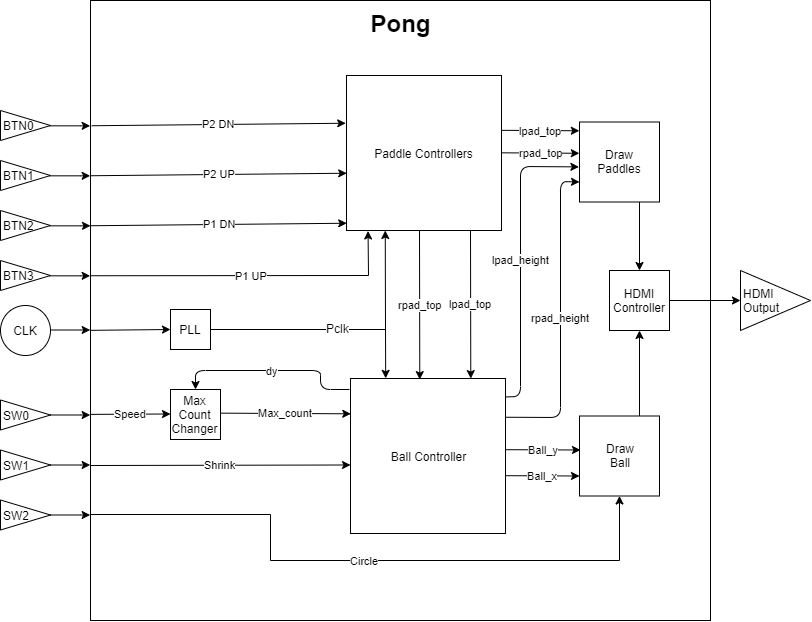


Figure 1: Pong System Diagram

Ultimately, the dimensions and locations of the paddles and balls are sent to the modules responsible for drawing the ball and paddles with the HDMI display.

## PLL Module

This module turns the system clock of frequency 125 MHZ into a clock signal named Pclk of frequency 74.25 MHZ. No schematic is given for this module since Vivado makes this circuit and we haven’t yet learned how it does this.

## Max Counter Changer Module

This module changes the max count value for a counter used in the ball controlled module. This max counter value changes the speed of the ball since it changes the frequency at which the ball moves. Since the ball changes its angle when coming in contact with a moving paddle, the frequency of the ball moving between pixels must change as well to keep the ball moving at the same speed across the screen.

For example, suppose the ball has a speed of k when moving straight across with dx = 1 and dy = 0 (1 pixel change each counter period in x direction, 0 pixel change in y direction) which is then changed to dx = 1 and dy = 1 (note dx always is 1 or -1, only dy changes). In this scenario, the ball is moving sqrt(2) times further in each counter period (Pythagorean Theorem). Using v = d/t with a constant v and an increase of d by a factor of sqrt(2) means the period must also increase by a factor of sqrt(2), hence the need for this module. Since dy can range from -2 to 2 and is an integer, a 5 input MUX is used. Note these speeds are denoted as a product of fast\_speed or slow\_speed and one of these factors in Figure 2 below, but this math was done with a calculator to produce a ball with a speed like in the real game of pong.

There is also a speed select switch which selects a 2-input MUX to be from the fast set of speeds or slow set.

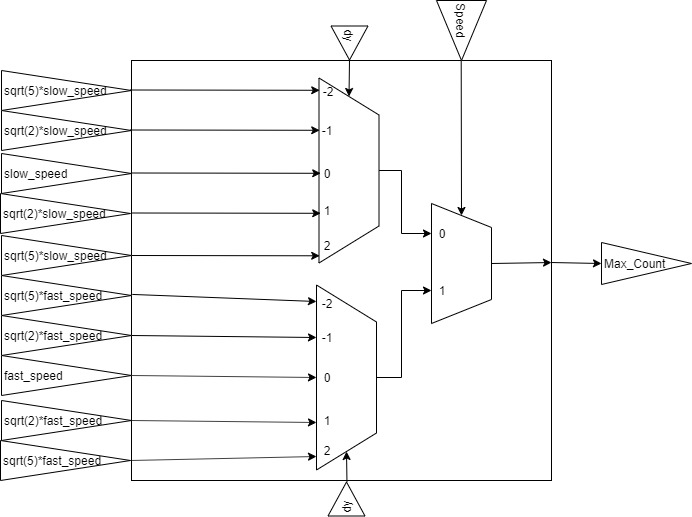


Figure 2: Max Counter Changer Circuit

## Paddle Controller

The paddle controller moves each paddle when the player hits an appropriate button. See figure 3 for the controls for the right paddle. Please note that the other paddle is controlled by an identical circuit except with inputs from player 1 and all rpad names as lpad names.

The first part of this circuit uses which direction the player is trying to move the paddle and sends either a “+1” , “-1” , or “0” to the adder which adds this to the current location of the top of the paddle. This value is normally used to update the value for the top of the paddle, unless the paddle is already at the top or bottom of the screen, in which case the paddle position remains the same.

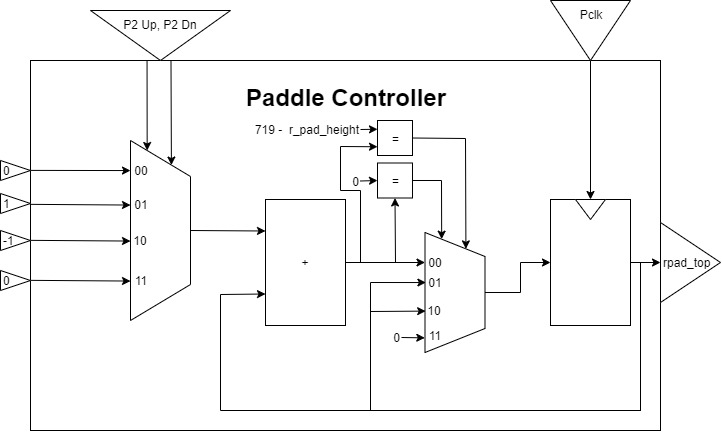


Figure 3: Paddle Controller

## Ball Controller Module

This module controls when the ball should bounce, receive a change in direction, or score. This module is split into multiple submodules for the purposes of being able to clearly show how it works. The first submodule can be seen in Figure 4 below. This submodule controls when the ball bounces off the top or bottom of the screen. When the ball becomes a radius away from the top or bottom of the screen, the value of dy is negated.

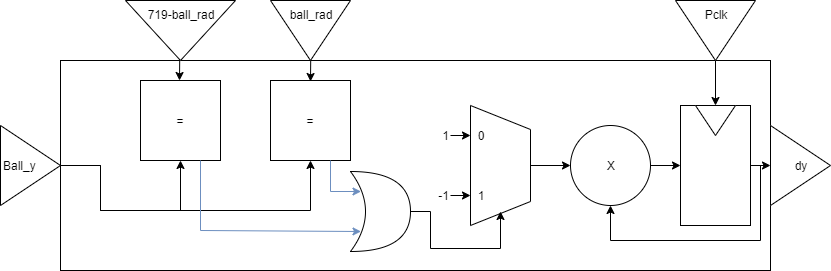


Figure 4: Top/Bottom Bounce Circuit

## Ball Bounce off Paddle Submodule

The next submodule can be seen in Figure 5 below, which shows the logic involved with bouncing the ball on a paddle. When the ball becomes a radius away from the edge of the paddle and is at a proper y value as to hit the paddle, dx is negated. If the paddle is moving at this time, the value of dy is adjusted in the direction the paddle is moving (bounded at +/- 2). Note for simplicity, the drawing below only covers the ball bouncing off the right paddle, but it is the same process for bouncing off the left paddle. Note this submodule is combined in a way with the previous submodule as to not give a multiple driver error for dy.

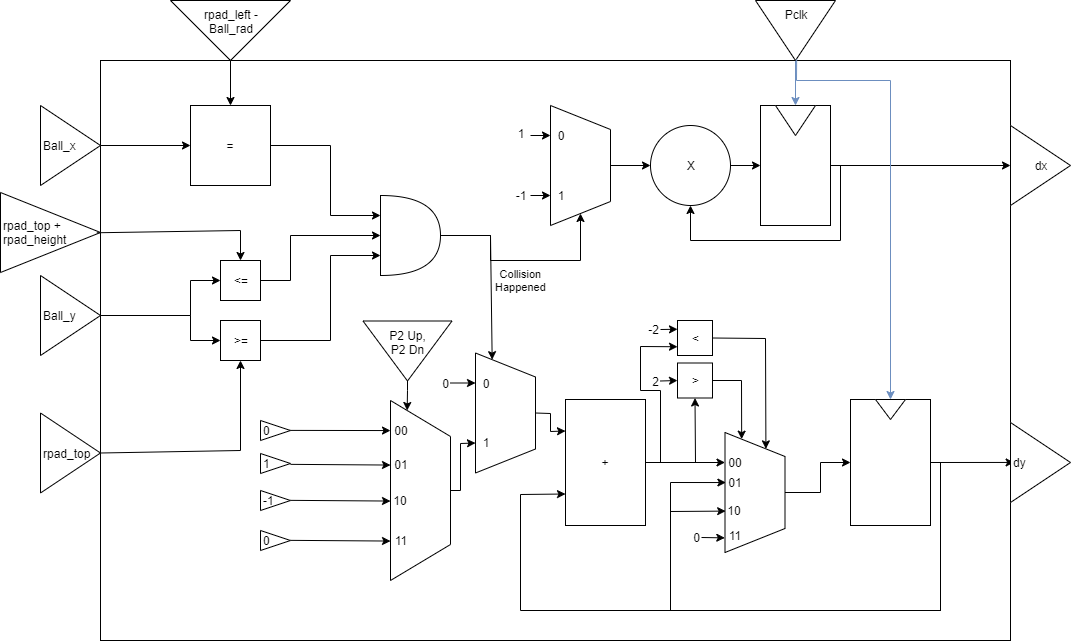


Figure 5: Paddle Bounce Circuit

## Ball Move Submodule

Now that the values of dx and dy are set to the proper signed values, they can simply be added to the current location of the ball (x and y) in order to obtain where the ball should be next. See Figure 6 below.

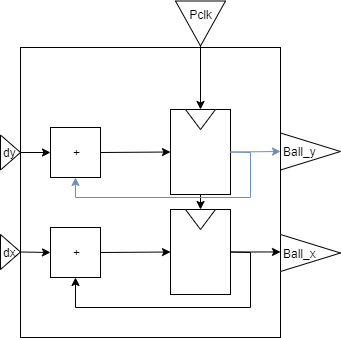


Figure 6: Ball Move Circuit

## Ball Score Submodule

When the value of ball\_x is such that the ball is at the left or right edge of the screen, the ball is reset to the middle of the screen. The value of dx is negated so the player who was scored on does not receive the ball next. The value of dy is pseudo-randomly set to either -1 or 1 based on the value of hcount, which is a signal who is independent of this module. See figure 7 below. Since the paddles are off the edge of the screen by more than a ball diameter, there can never be a situation where the ball should bounce back from the paddle at this point.

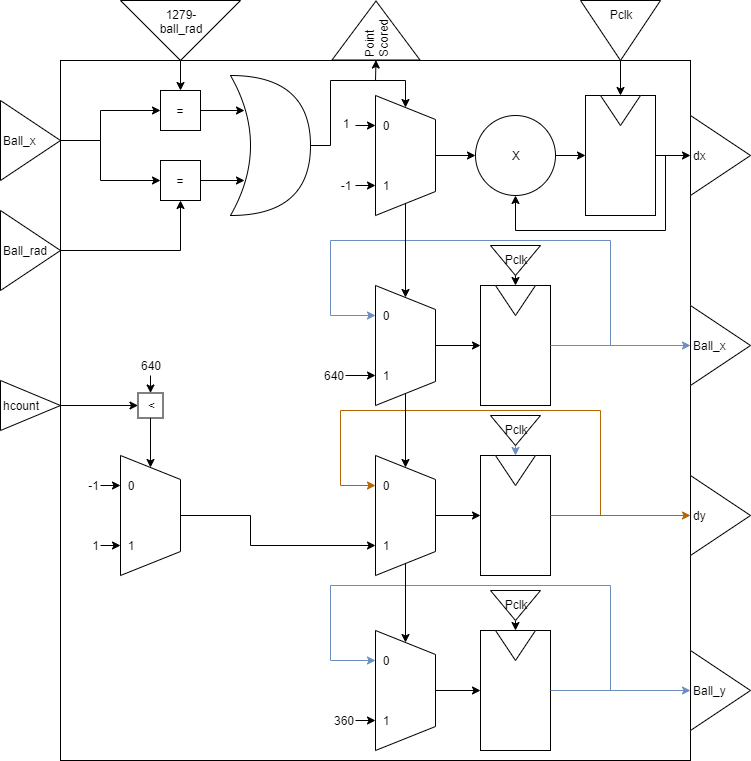


Figure 7: Ball Score Circuit

## Paddle Shrink Module

This module causes the paddle to shrink if the ball strikes it while the switch connected to shrink is high. The paddle will not shrink beyond a minimum threshold. The paddle height is reset each point. This module takes an input from the diagram in Figure 5 named Collision Happened, which is high when the ball strikes the paddle. See Figure 8 below. Note this diagram only shows the right paddle shrinking, since the Collision Happened signal from Figure 5 is for only the right paddle. There is an identical circuit for the left paddle very similar to this one.

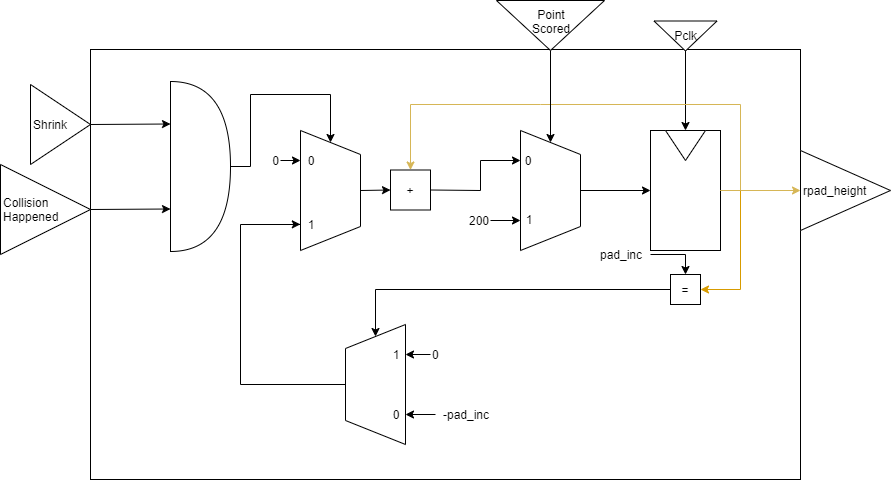


Figure 8: Shrink Circuit

## Draw Module

This module draws all of the data we have using HDMI. If the

# 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))