System Design Document for Pong Game

University at Buffalo, The State University at New York
EE478 Fall 2019

			Revision History			
Written By	Daniel Giovino					
Engineer	Daniel Giovino		**Rev**	**Date**	**Session**	**Approved/Grade**
Engineer	Peter M. VanNostrand		Orig	12/02/19	Mon. 9AM	
Teaching Assistant						
University at Buffalo The State University of New York						**EE478F19 FINAL LAB**

Table of Contents

List of Figures

List of Tables

No table of figures entries found.

1. Introduction

1.1 Overview

This device implements a version of the classic video game pong. Pong is an early two player arcade game based on table tennis and was first released in 1972. In the game each player controls a rectangle placed on the left or right edge of the screen, this represents the player’s ping pong paddle. Then a ball, represented by a square, moves across the screen and the players must move their paddles to intercept the ball and bounce it back towards the other player. On the Zybo board players can move their paddles up and down by using the on board push buttons. Depending on the vertical speed of the paddle when it intercepts the ball, the ball may gain or loose some vertical speed allowing the ball to move in diagonals. When the ball hits the top or bottom edge of the screen it bounces and continues in the same horizontal direction, when the ball hits the left or right edge of the screen, the opposing player scores a point. In our implementation the score is displayed on the LEDs as two 2-bit numbers indicating the score for the left and right players respectively.

In our implementation of pong we have added a few new features to the original game. When the first slider switch is set to on the ball increases in speed to make the game more challenging. When the second slider switch is turned on the player’s paddle becomes smaller each time they bounce the ball, and when the third slider switch is on the ball becomes a circle rather than a square. These extra features are individually selectable and can be used independently, in any combination, or all at once.

To create the game logic for this device we first designed the simple operations that would need to be performed on paper, then implemented functionality using VHDL before testing on a Xilinx FPGA. Design for this project was done iteratively with each feature being fully implemented and tested before further functionality was added. Testing was performed on a Xilinx FPGA using the Xilinx Vivado CAD tool.

For this lab the game board, including paddles and ball, were rendered by the Xlinix FPGA for display to a monitor over the HDMI out port.

1.2 Document Scope

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

1.3 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.

2. System Design Overview

2.1 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.

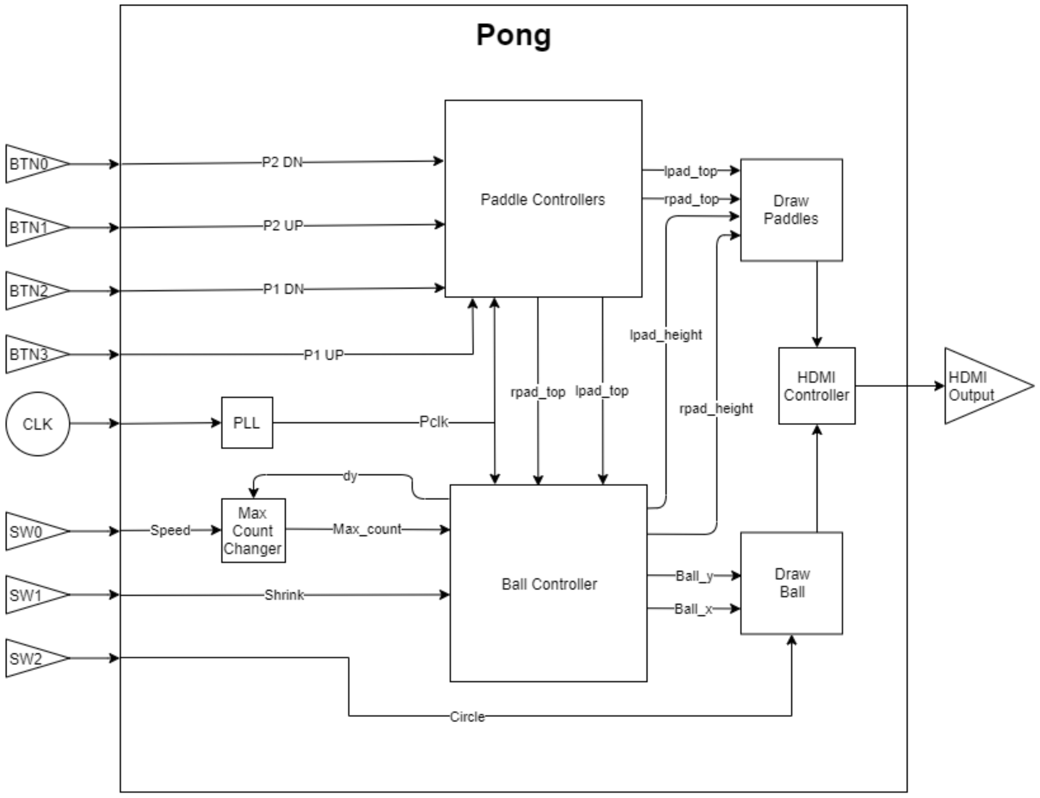

Figure 1: Pong System Diagram

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.

2.2 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.

2.3 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 magnitude). 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 (which are both constants) 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.

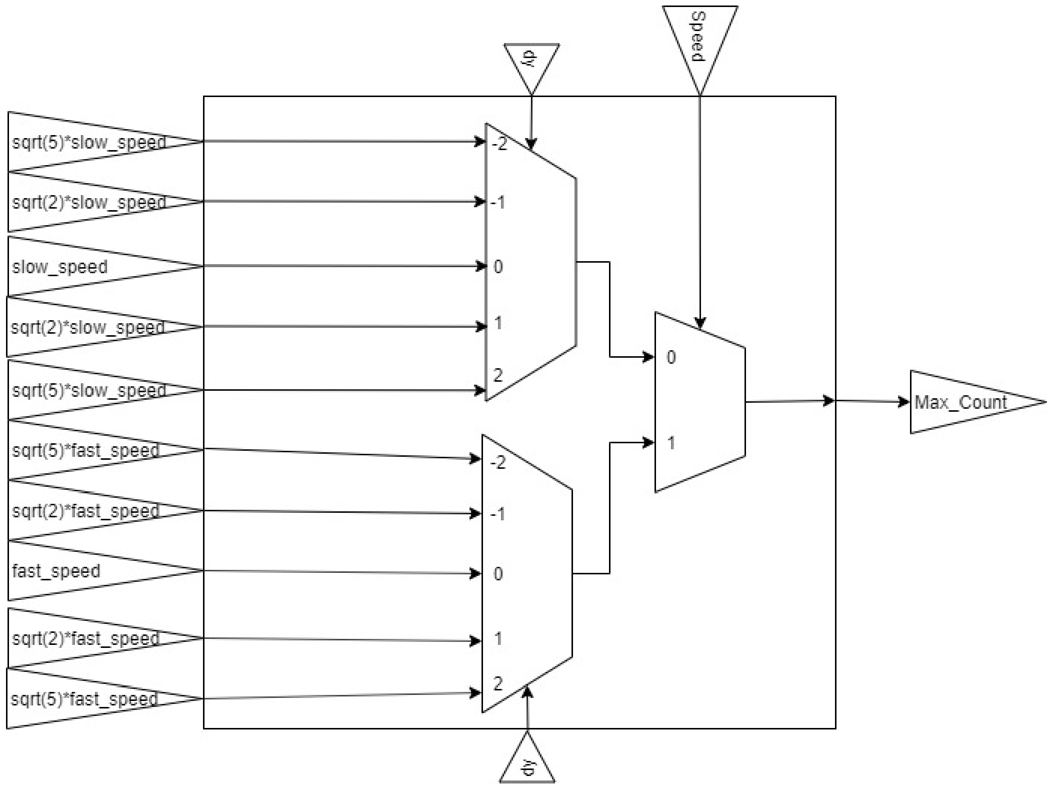

Figure 2: Max Counter Changer Circuit

2.4 Counter Module

This module implements the counter for the system. The terminal count of this counter is determined in the previous module and is input to this one as “Max_Count”. This counter simply adds one to the current count value until the terminal count is reached (or exceeded if the value of Max_Count is decreased). When this happens, the counter is reset to zero. For one Pclk period, the signal “A” is output as a one-shot high signal. Signal “A” is used to drive all of the flip-flops in the rest of the modules.

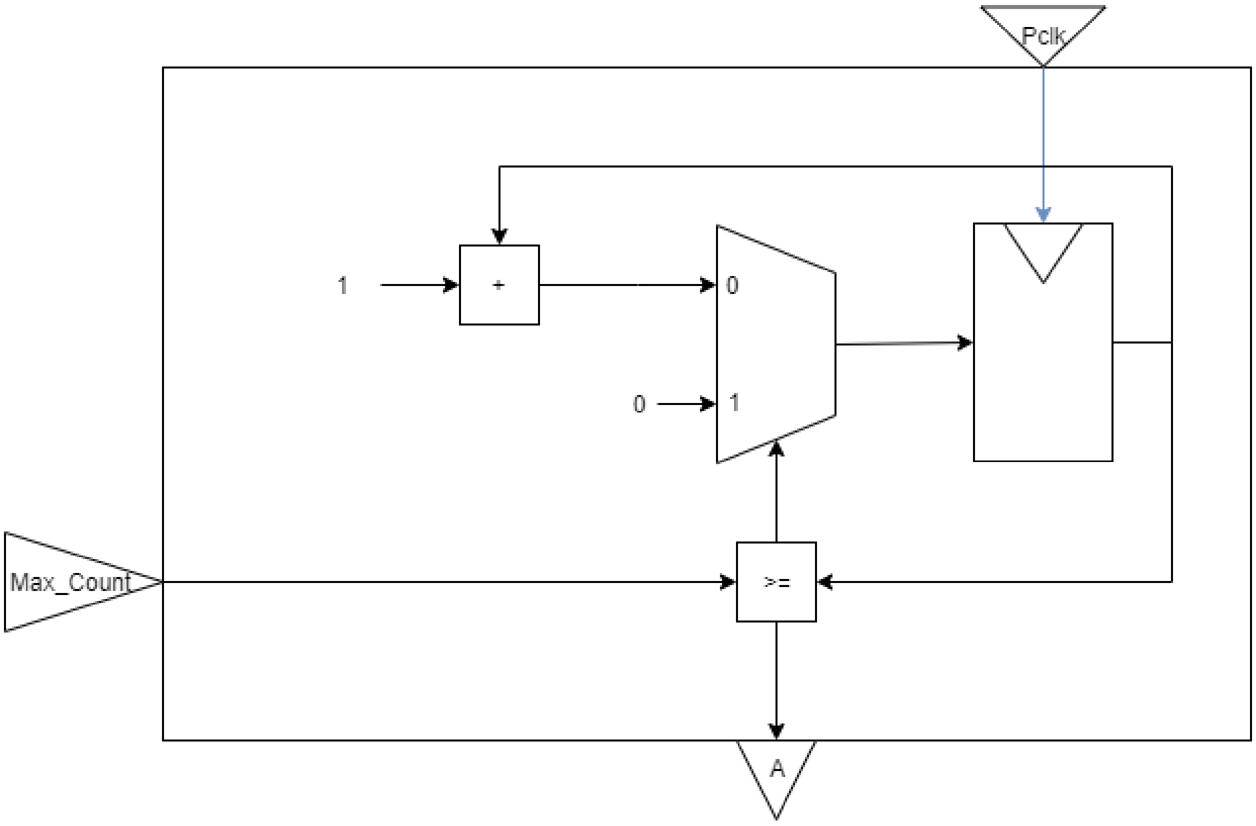

Figure 3: Paddle Controller

2.5 Paddle Controller Module

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.

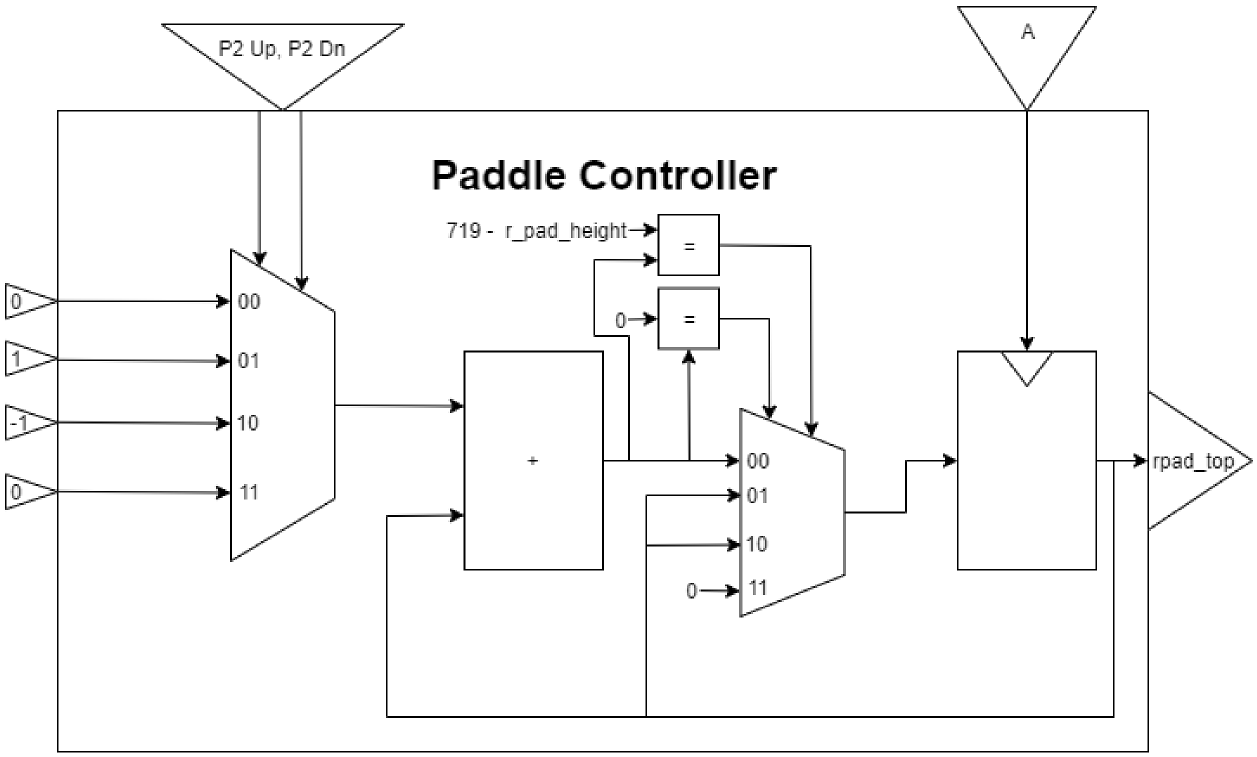

Figure 4: Paddle Controller

2.6 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.

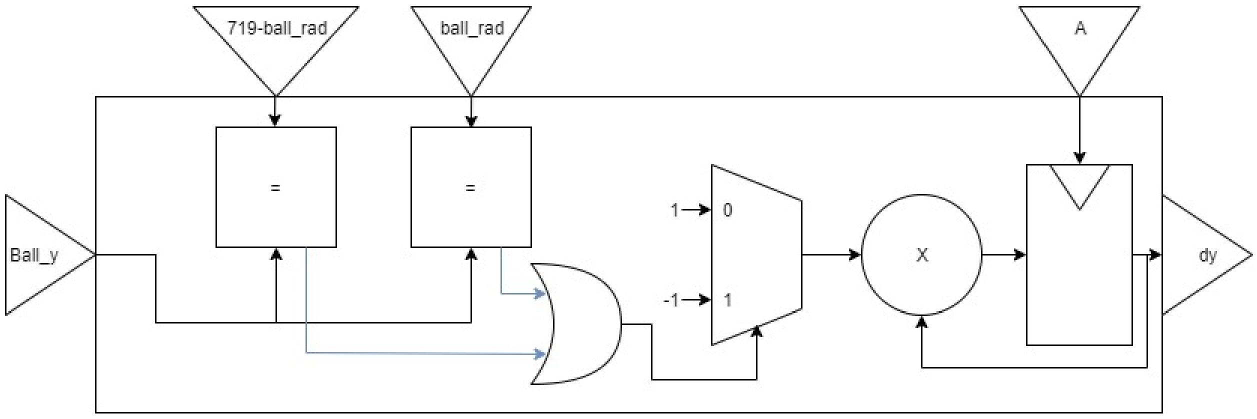

Figure 5: Top/Bottom Bounce Circuit

2.7 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.

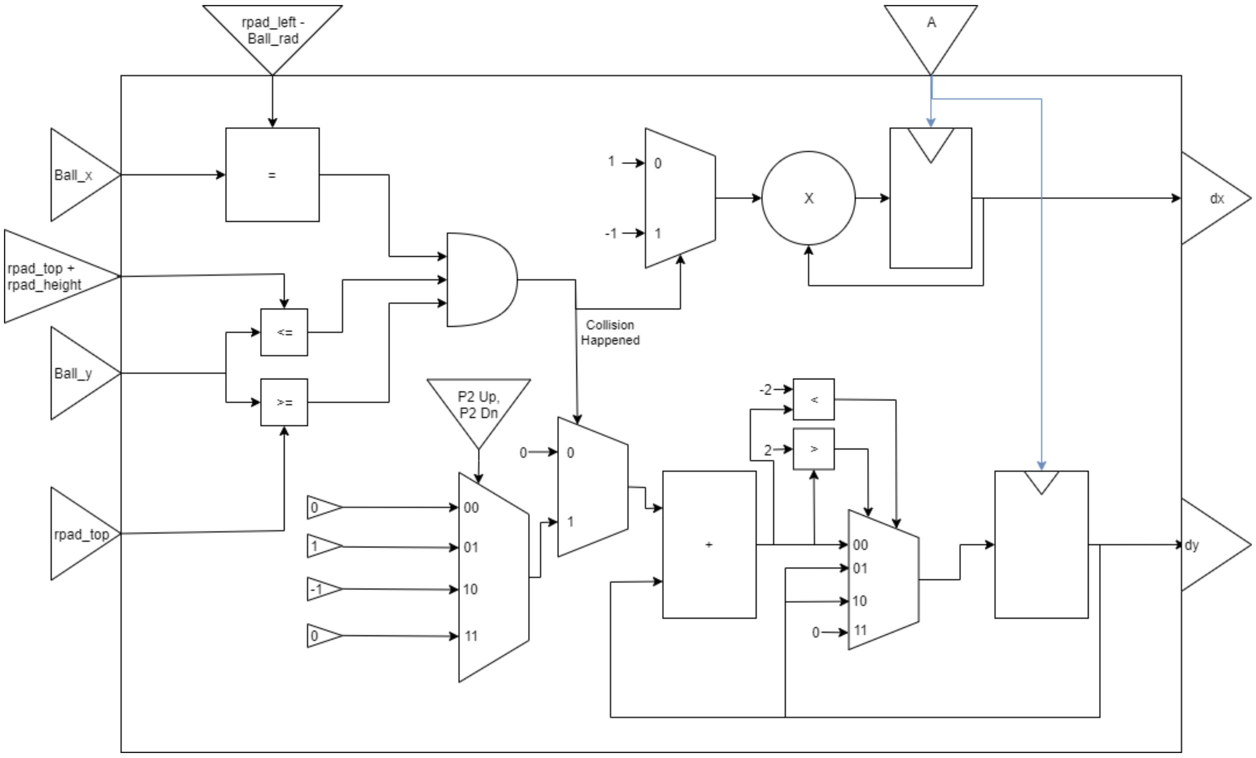

Figure 6: Paddle Bounce Circuit

2.8 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.

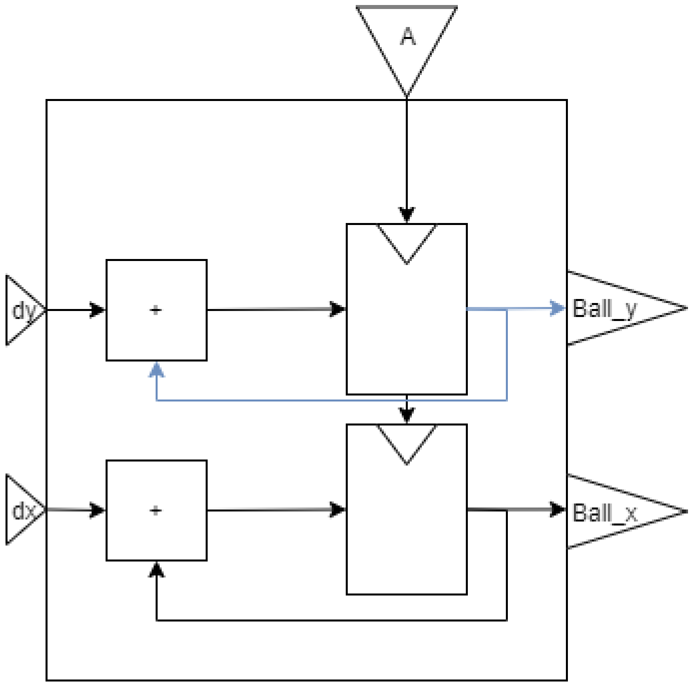

Figure 7: Ball Move Circuit

2.9 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.

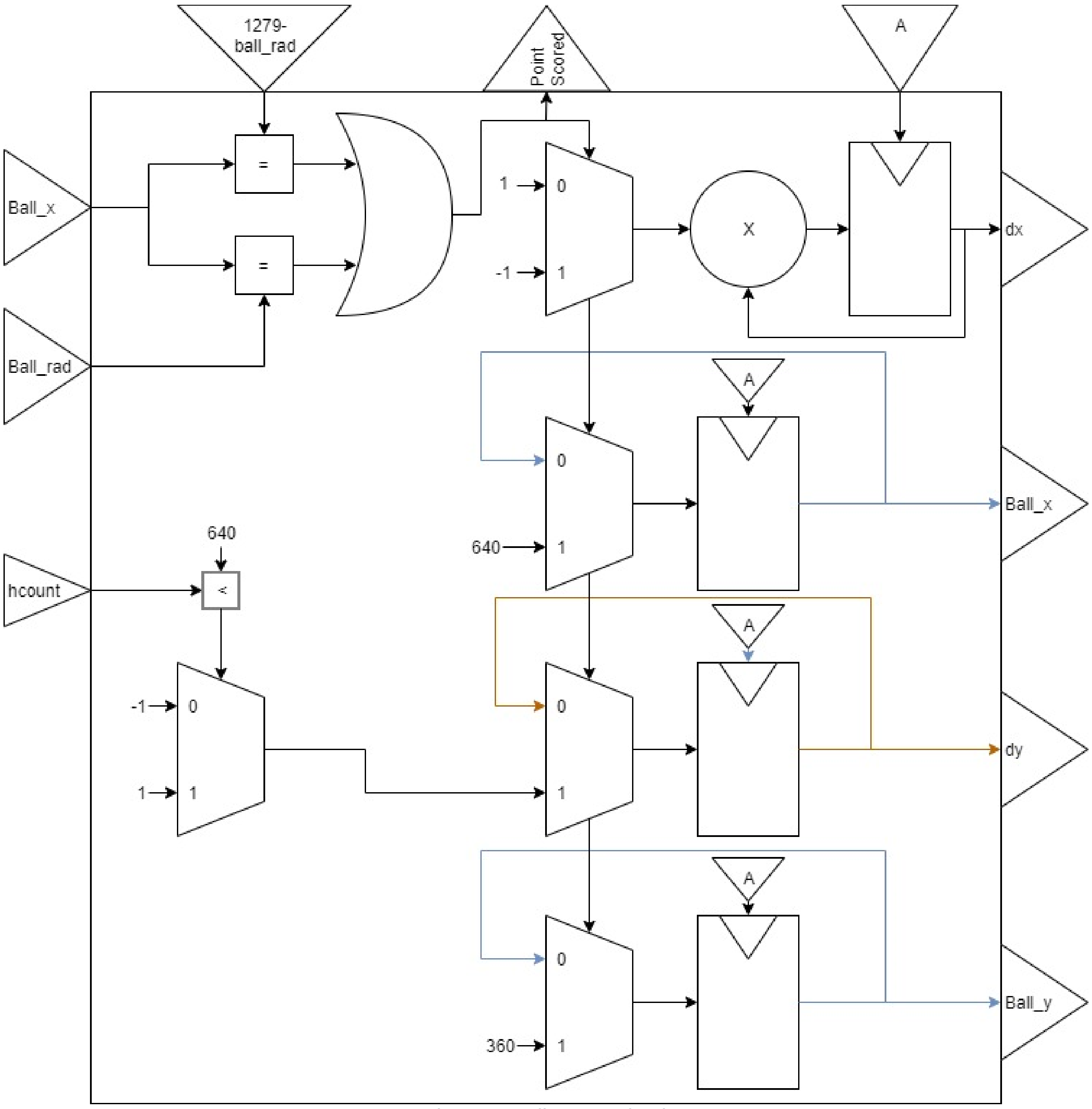

Figure 8: Ball Score Circuit

2.10 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.

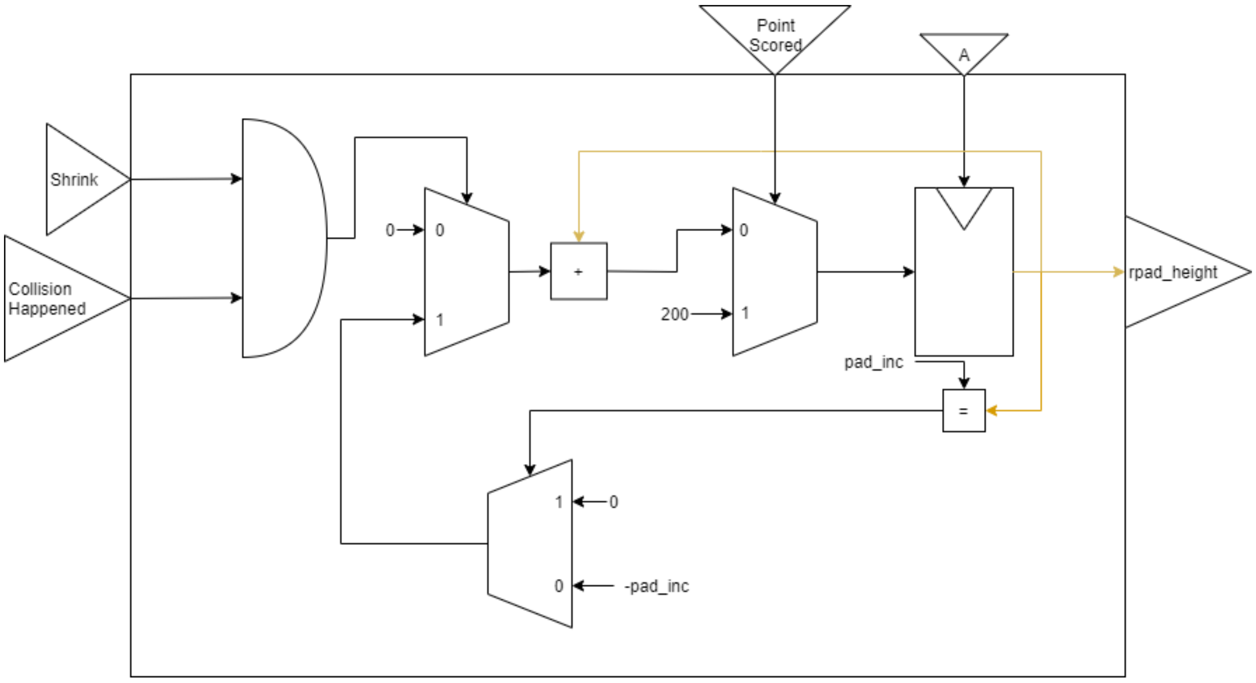

Figure 9: Shrink Circuit

2.11 Draw Module

This module draws all of the data we have using HDMI. The majority of this functionality is implemented using the code provided by the instructors for lab 5. See the lab 5 design document for further information on how the HDMI module is instantiated and driven.

3. Testing and Verification

To ensure that our circuit functioned as intended we performed testing. The way we tested this code was by synthesizing 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 due to time limitations and complexity of the design.

3.1 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.

3.2 Objective Verification

Once we synthesized our design and loaded it onto the FPGA we tested the circuit by connecting a monitor the HDMI output of the Xilinx board. We then observed the paddles and ball being displayed and used the push buttons to manipulate the paddles to ensure that they moved as expected. After this we performed copious play testing to ensure that the ball collision and scoring were working as expected.

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)
- Xilinx Vivado CAD tool (Xilinx)