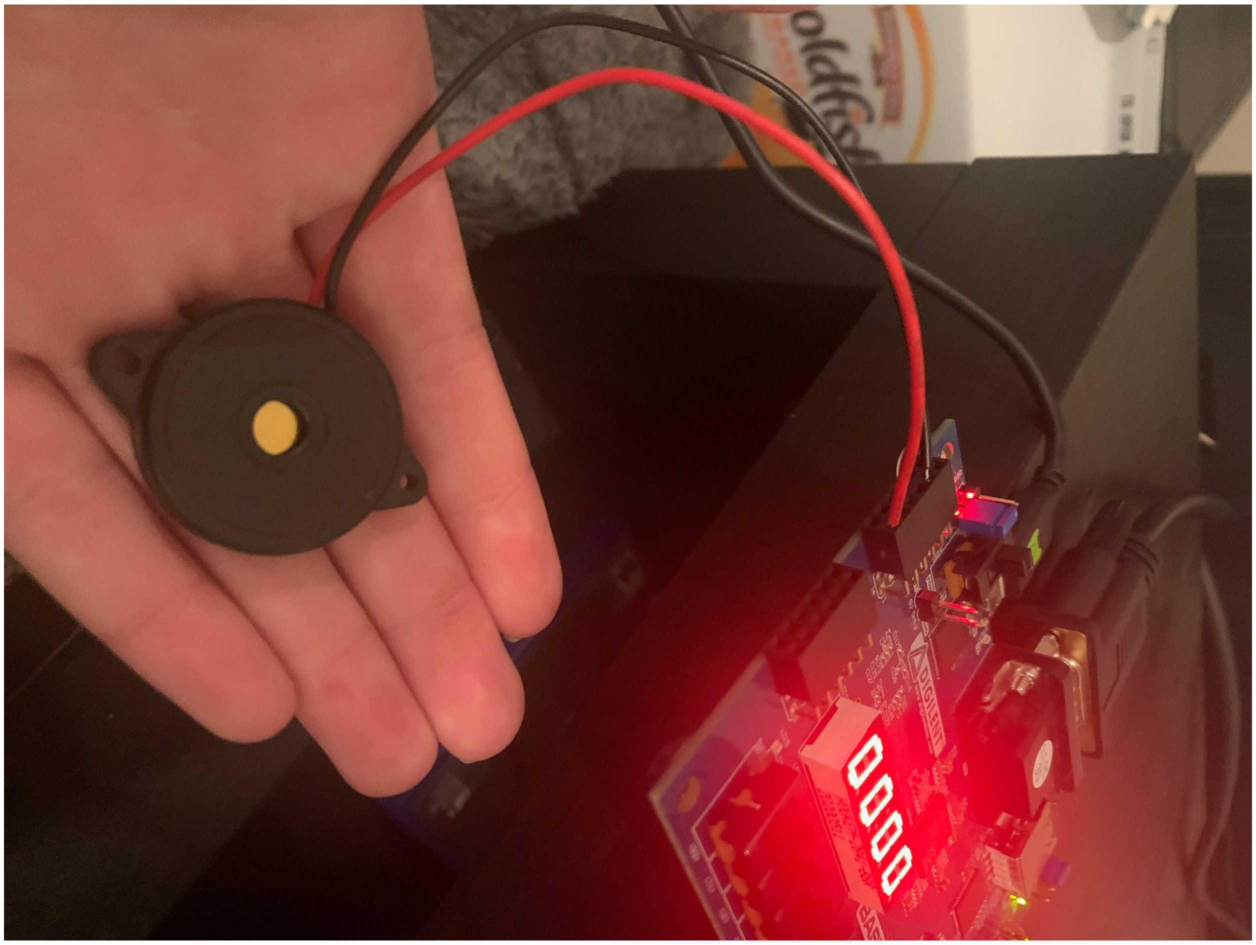
Pinball on Basys3 using VGA by Hannah Galocy

Introduction / Summary:

For my final project in CompE470L, I've created a pinball game using a Basys3, Verilog, a speaker, and of course a VGA cable. My father owns many pinball games, and I grew up with them so this project was something I really enjoyed completing. In the beginning, I didn't know where to start. I experimented with "Pong" games people created online and eventually morphed it into my pinball game complete with scoring, flippers, a sound effect, tilting, and targets. As I was creating the game, I found that there were situations in which the game could become slow or I felt that, as a user, I couldn't manipulate the ball as much as I liked. Because of this, I added a moving target and the tilting capability. If you are familiar with physical pinball games, you're aware that the ball can sink down the middle resulting in a lost ball regardless of how you use your flippers. To remedy this, pinball players will bump or "tilt" the machine to try to get the ball on a different trajectory. In the case of my game, a user can "tilt" causing the ball to switch it's horizontal trajectory to the other direction. After having a few beta testers, I've found the tilting capability can be used to get the ball to hit more targets rather than solely act as a saving grace.

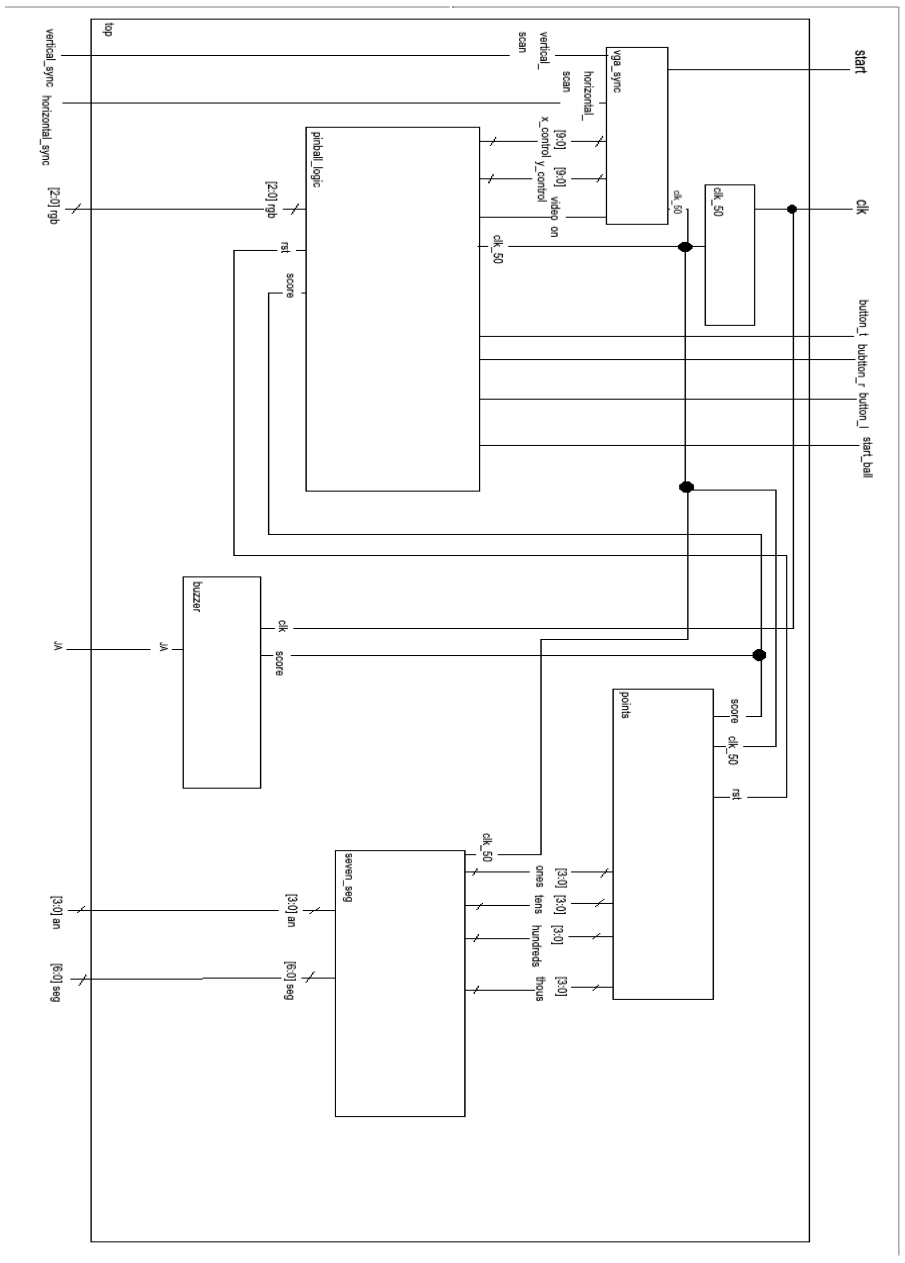
User Instructions

Starting up: To set up the game, make sure you have the Basys3 board with the bit file programmed. Also make sure that it is connected to a power source via micro USB. To get it working with a display, connect a VGA cord. Lastly connect the speaker as shown below.



Playing the game: Display your game by using the right most switch on your Basys3 board. You will now see the game field complete with a red moving target up top, 3 cyan circles, and 1 pink circle. These are your targets to accumulate points. At the bottom of the screen, there are two flippers that you can control with the left and right buttons on your Basys3 board. To start the game(or reset the game and score) press the center button. This will cause the ball to start moving. The goal is to not let the ball sink between the two bottom flippers. If you're in a rough spot and want to switch the ball's horizontal trajectory, press the top "tilt" button. To keep the game going and to accumulate points, use the flippers and the tilt button to hit the targets. BE CAREFUL: The pink circle makes the ball go really quick. Your score will be displayed on the Basys 3's seven segment displays. You wipe your score by pressing the center button.

Block Diagram



Module Descriptions

My project consists of a total of six modules. They are top, pinball_logic, vga_sync, buzzer, points, and seven_seg.

top: My top module acts as a driver program calling on all the other modules and taking in the Basys3's inputs and sending the Basys3's outputs. These inputs are clk(100MHz), start(rightmost switch), the left button, right button, center button, and top button. These outputs are the 3 bits of RGB for the VGA, horizontal_sync and vertical sync for the VGA, the 7 bits for the seven segment displays, the 4 bits for the anodes of the seven segment displays, and JA to send pulses to the speaker. The module also manipulates the clk to a 50MHz clock called clk_50 to pass and sync up the other modules.

vga_sync: This module's inputs are start(rightmost switch) and the clk_50. The outputs are horizontal_sync, vertical_sync, [9:0]x_control, [9:0]y_control, and display_on. This module outputs display_on to tell the screen to display anything or not. This module also "syncs" up vertical_sync and horizontal_sync with y_control and x_control. Because of this, x_control will work as the x component of the horizontal scan of the VGA and make it possible for pinball_logic to draw to the screen and manipulate the ball's behavior. The same is done for the vertical scanning of the VGA and y component. **This module is not my code.** This module was translated from VHDL by CynicalApe and his original code can be found at this link https://github.com/CynicalApe/BASYS3-PONG/blob/master/sources/new/vga.sv. The original code Cynical Ape used can be found at this website in VHDL (not in English) by Jan Bukowski https://mikrokontroler.pl/.

pinball_logic: This module is what makes the project pinball. All of the logic for the actual game of pinball is written in this module as well as the means to display its flippers, targets, and ball. Its inputs are the x_control and y_control from the vga_sync module, clk_50, the four buttons to be used, and the video_on flag from vga_sync. Its outputs are the 3 bits for RGB, a score flag, and a rst flag. The first chunk of the module is declaring the different variables and parameters to be used for the logic and display of the different objects in the game. Further down, I have an initial block to set all the registers to their default values. I then create a clock divider to produce about 30Hz to be used for the combinational logic of the ball and other entities. Finally I have an always block triggered by the posedge of the clk to check for whether the star_ball button or tilt button was pushed and sets variables appropriately. This is where rst will trigger the points module to reset the score values. After this I have the animation logic for the flippers, top bar, and ball. Finally, I have a mux to display the values based on the x_control and y_control values.

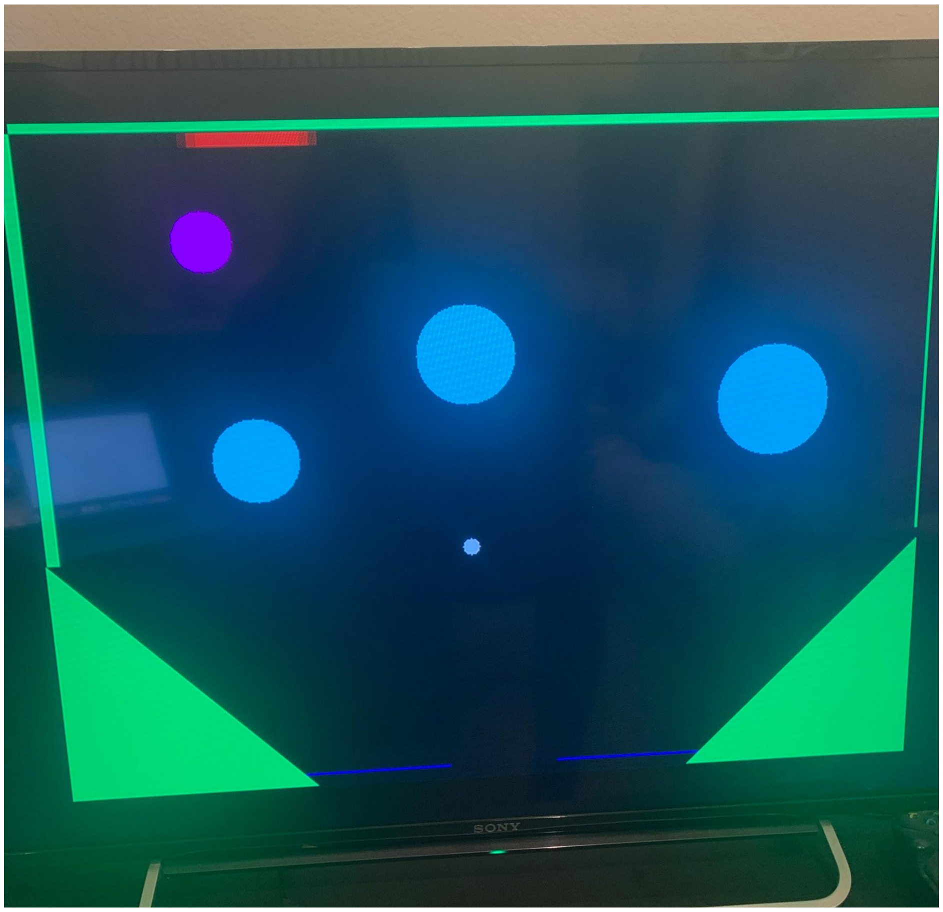
buzzer: This is a short module designed to trigger the speaker when an object is hit. It's inputs are clk and the score flag from my pinball_logic module. It's output is JA for the speaker. The speaker plays at about 765 Hz and has a duration of about a 6th of a second.

points: This module is for accumulating points when an object is hit or resetting them to 0 when the game restarts. Its inputs are clk_50, score for pinball_logic, and rst from pinball_logic. Its outputs are ones, tens, hundreds, and thous that are 4 bits each. When the score flag is triggered, the values of ones, tens, hundreds, and thousands are incremented as if they are all digits of one number. If rst is triggered, they are all reset to 0.

Seven_segment: This module controls how the score of the game is displayed. Its inputs are clk_50, ones, tens, hundreds, and thous from the points module. Its outputs are an and seg to control the seven segment display's anodes and cathodes. The logic is heavily based on the logic used for the clock project earlier in the semester.

Validation Results / Photo

Once the ball appears to have "hit" where one of the circles is drawn to the screen, the speaker makes its sound effect and a point is added to the user's score which is displayed on the seven segment display directly on the Basys3 board. Once the ball passes between the two flippers, the game resets with the ball going to its default position and the score going back to 0. If the user presses the reset button in the middle of the game, the game behaves as if the ball has passed between the two flippers and resets. The game behaves as intended.



Sources and Inspiration

Pong Game:

https://www.fpga4fun.com/PongGame.html

https://www.instructables.com/CPE-133/

https://github.com/CynicalApe/BASYS3-PONG

Using a Buzzer:

https://www.instructables.com/TimerBuzzer-for-Basys-3-in-VHDL/

Translating from VHDL:

https://blog.digilentinc.com/battle-over-the-fpga-vhdl-vs-verilog-who-is-the-true-champ/

Using VGA:

https://embeddedthoughts.com/2016/07/29/driving-a-vga-monitor-using-an-fpga/