FPGA Tic Tac Toe Game with VGA Display

List of Authors (John Akroush, Nicholas Wheelis, Tao Wang)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: johnakroush@oakland.edu, nwheelis@oakland.edu, taowang@oakland.edu

Abstract—

I. INTRODUCTION

Tic Tac Toe is a staple pastime for people to play with friends when they are bored. The project covers registers, decoders, multiplexers, addresses, and time with clocks. We will learn how to display code to a VGA display during this project. We use sw0-1 as the player/color selection, sw2-sw5 as the address, and register 0-8 translates to squares 1-9 respectively.

1	2	3
4	5	6
7	8	9

Fig 1: Tic Tac Toe board

II. METHODOLOGY

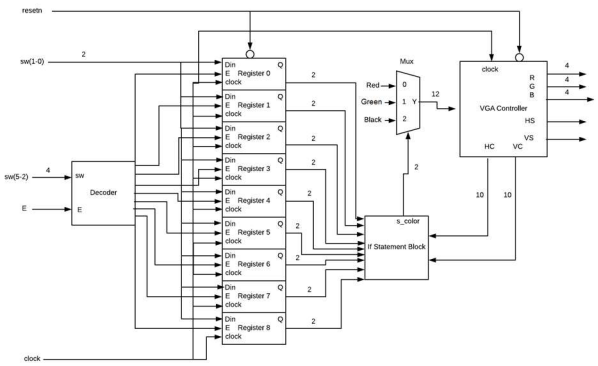

Fig 2: Circuit Diagram

The general idea was for our group to recreate the tic tac toe board game using a FPGA board and display it using a VGA display. A decoder with enable is used to decide which register to access. Each register corresponds to one spot on the tic tac toe. The register passes the player value through to a block of if statements, if the specific register is chosen by the decoder output. This if statement block handles the color assignments for each rectangle; red for player one and green for player two. If the register has not been accessed previously, or if it is currently not being accessed, then the if statement block tells the circuit to color the corresponding rectangle black. The color value output from the if statement block is sent to a 3 to 1 multiplexer that associates each value to one 12-bit binary value for its corresponding color. For example, player one, which has a value of 01, is assigned to the color red (F00). The output color of the multiplexer is sent to the code for the VGA display. The VGA code scans through each pixel from left to right and top to bottom starting with pixel 0 of the display region. It sends the location value of each pixel, HC and VC, to the if statement block which uses those to determine the color for each rectangle. The VGA code also colors any non-display region black, otherwise it would not work.

A. Decoder

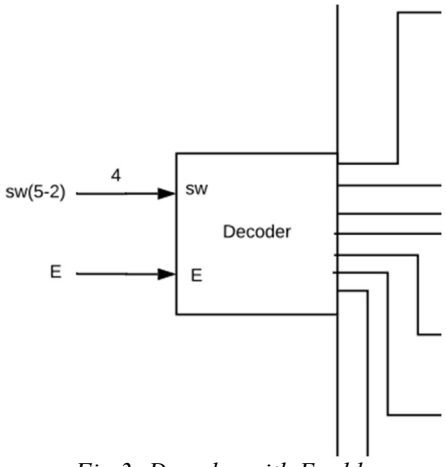

Fig 3: Decoder with Enable

The decoder is the first part in the design. It takes user inputs for the address and enable. It uses the inputted address from sw2-sw5 to decide what register that the player stores their color into. The player will use the switch to enable the system to store the color into the selected register. For example, if player 1 puts in the address “0011”, and presses enable, then register 3 will have the color red stored.

B. Registers

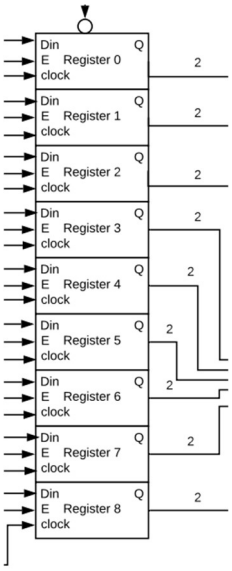

Fig 4: Registers with Enable

The registers are the second part of this design. There are nine registers that are used. They are used to store the color which is selected by sw0-1. They are enabled by the decoder. They are an integral part of the design because they store the color for the player for the rest of the game until the resetn button is pressed which clears all the registers and restarts the game. For example, the decoder uses its inputted address, say 0111, to select a register, register 7, and enables it which saves din, 01, which is red, to the register.

C. Multiplexer

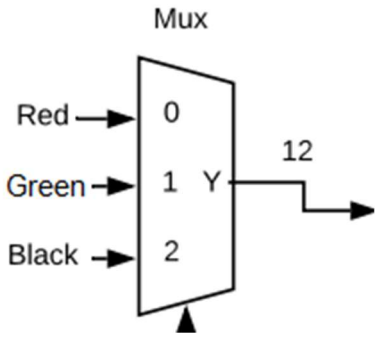

Fig 5: Color Multiplexer

The multiplexer in this design is used to decide the color the VGA controller will display. After the if statement decides the color, it sends the 2-bit selection to the mux to have it output whatever color has been decided. For example, if s_color is 01 then the mux will output the 12-bit code for the color red.

D. If Statement Block

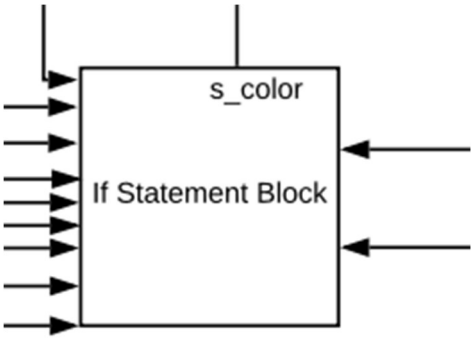

Fig 6: If Statement Block

The if statement block is a combination of if statements that depend on each of the registers output as well as the HC and VC outputs from the VGA controller. The if statements go through and pick what register to use and what its output is based on how each register is connected to a certain range of the display that is inputted from the HC and VC inputs. It outputs a 2-bit signal that is an input to the Multiplexer.

E. VGA Controller

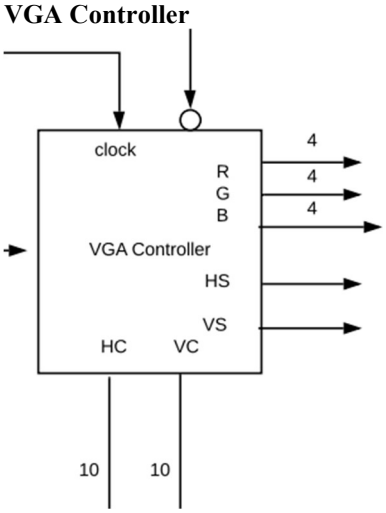

Fig 7: VGA Controller

The VGA controller is used to take the inputs from the rest of the design and display them onto the display being used. It does this in tandem with the multiplexer and if statement block. The controller sends two 10-bit signals called HC and VC outputs to the if statement block. It starts out with 0 and 0 which is the first pixel and would connect to a certain color based on the if statements and would go through every pixel doing the same. This is how displays work as they go row to row, left to right determining the color of each pixel until you have an image. This continually refreshed so it is constantly updated. For

example, if the HC was 10 and VC was 10, this would be a pixel in the top left rectangle, which, in the if statement block, is associated with register 0. This would make that pixel the corresponding color of the player playing.

III. Experimental Setup

Vivado 2018.3 software was used to program a Nexys A7 FPGA board with the tic tac toe program. The Nexys A7 was connected to a display via the VGA port using a VGA cable. It was expected that the display would work flawlessly. Once the code was complete and perfected, the display worked as we expected.

IV. Results

Video of the project working: https://www.youtube.com/watch?v=oqcOLH1gtnc&t=3s

Fig 8: Tic Tac Toe Game Display

Above is a video and picture showing the tic tac toe project in action. As seen in figure 8, colored rectangles, corresponding to each player (player one is red, while player two is green), are displayed on the screen in the player chosen locations. These locations are chosen by the if statement block that sets a certain horizontal and vertical pixel range (HC and VC from the VGA Controller) to each register. If a specific rectangle (aka register) is selected via the address input by the user, then the color for that player is applied to that register. The player number is sent to the color multiplexer which chooses the player's color and that color is sent to the VGA Controller which displays the color in the rectangle location.

We encountered several issues during the creation process that allowed us to gain a deeper understanding of the topics. One issue was figuring out exactly how the VGA Controller code works. We created multiple versions of the if statement block code that applied to the different misunderstandings of the VGA code. The current code is the correct application of the VGA Controller code. Another error we ran into was when we were attempting to test the project after it was 'completed.' We connected the FPGA to the display and we received an error on the display that said, "The current input timing is not supported by the monitor display. Please change your input timing." To troubleshoot the issue, we tested our project with multiple different monitors, but we got the same error. We eventually resolved the error in the code by changing the clock pixel ratio to the same value across all code files (100 MHz). After solving this issue, the tic tac toe project worked flawlessly.

The tic tac toe using an FPGA and VGA display project helped us get a deeper understanding of in class, as well as other, topics. We all gained experience using VHDL in Vivado, FPGA's, troubleshooting, and VGA displays.

Conclusions

The main take-away from creating this project was to apply the knowledge obtained from the course to a real life project. The topics used that were learned from the course include decoders, registers, enable, Vivado with VHDL, FPGA, if statements, and multiplexers. This project also taught us more topics than what were covered in the course, such as displaying to a VGA display. In terms of what can be improved with the project; given more time, we could add a feature that checks if a space is already taken to make sure the player can't choose the box that already has color. We can add something to determine which player wins and reset automatically. We could also add a scoreboard to keep track of each player's wins.

References

- Professor Daniel Llamocca's VGA Controller code.