California State University, Northridge
ECE 524L - Advanced FPGA Design
Fall 2022

Lab 5: VGA Continued

October 16, 2022
Instructor: Saba Janamian

Troy Israel

Introduction:

The purpose of this lab was to continue our experiments with the VGA Pmod that we used in the previous lab. We will be expanding and adding more features to the pre designed vhd file that we used. We will be adding features like showing various colors on the screen, different patterns and resizing the various patterns from small to large. In order for us to utilize the ten different features in our lab we will be implementing both buttons and switches so that by a specific switch combination that would basically select a different feature. The buttons will be used to adjust the size of various patterns.

Procedure:

To start this lab experiment we first will create a new project and incorporate the VGA source file that we used in the previous lab. Obviously we should use the source files from github so it is a fresh download. We will first need to make sure the submodules have been added from the github link and that all the files are updated before we get into Vivado. Once we have everything updated we can launch Vivado. In Vivado we will first need to create a VGA controller module or add the VGA controller that came from the github link. Either way we will be adding the same code, either the file is brought into through settings or copied into the file we made ourselves.

Once the VGA controller code has been added to the project, we will see a question mark icon underneath our top module. This is informing us that we do not have a clock wizard. While the code we download instiants a clock wizard we need to add it to our design, it doesn't get added automatically. We simply go to the IP catalog and look for the clock wizard IP. We will modify the clock wizard so that it can support a screen resolution of 1920x1080. We change the clock frequency to 148.5 which the VGA controller code that we download tells us that this frequency we need to set it to. We also need to make sure the clock wizard does not have reset or is locked, so we remove those two things.

The VGA controller module is where we will be doing most of the work. Firstly, we add two input ports into the entity section. One is an input port defined for the buttons and the other will be for the switches. These two input ports will be 4-bits because the Zybo Z7 has both 4 switches and buttons.

For adjusting the box constants we have to modify the X max and Y max. This is because we need to make sure that for the ball bouncing around the screen, the ball won’t get stuck in a

specific region. We also change the box width which will make the box much larger or smaller if we choose. The X max will be set to the frame width while the y max will be set to 0. This ensures that the full screen will be used.

The next step is to add the necessary code to generate the two images before we get into the main architecture of the design. We will need to create two ROM definitions. The first one will generate the ball that will be moving around the screen and the other will create a pixelated image that will be resized later on in the lab. We will also add any necessary signals for the two images.

Inside the test pattern logic of the control file, we first need to comment out the vga reg, green and blue assignments. The reason is because we will not be using it for this lab. To implement the various features we want for our design we first need to create a process block. Everything inside the process block will be housed inside an if block when active is 1. If active is 1 then a rather large case block sensitive to our switch inputs will enable.

Switch States 0000 to 0011 (0 to 3):

These 4 switch combinations will display a single color. 0000 the screen will be black and 0001 to 0011 will be red, green and blue individually. To implement this feature, we will be setting the vga_red, green and blue to either 0 or 1 depending on the color. For example, all three signals will be 0 when the screen is black. But for red, let's say, only the vga_red signal will be set to 1 while the blue and green signal will be set to 0.

Switch States 0100 (4):

For this case, we are going to display red, green and blue horizontally and we want each color to take up a third of the screen. In order to implement this we first need to create a signal called ONE_THIRD and this will be a natural value that will be the frame width divided by 3. We will set a range for where we want each color to appear. We will reuse code from switch states 0000 to 0011 but will need to use if/elsif statements to implement the range. For example we want green in the middle and in order to do that we need it to get the max value of the ONE_THIRD but also 2*ONE_THIRD value. This is how we can get the green color to appear as a strip in the middle of the screen while red and blue will be on either side of it.

Switch State 0101 (5):

This feature will be very similar to the previous feature but instead of 3 colors we will do 8 colors (White, yellow, cyan, green, magenta, red, blue, black). Which means we create another signal called ONE_EIGTH and we will also be reusing the same portions of the if/else block but with more of them. The main difference is that the 3 vga signals will have a specific combination of 0's and 1's that will make a specific color.

Switch State 0110 (6):

We will be implementing 8 different shades of gray. This will be very similar to the previous switch state. However the only main difference is for each of the three vga signals instead of using single bits: 0 and 1, we will be using 4 bit values.

Switch State 0111 to 1010 (7-10):

For these three switch combinations they will all involve the use of the buttons. 0111 will display a horizontal pattern and when you push buttons 0,1, 2 or 3 it will change the size of each strip. A very similar thing will happen for the 1000 switch combination but this time it will be vertically. For combination 1010, it will display a checkerboard pattern and again pressing the different buttons will increase or decrease the size of the squares inside the checkerboard.

Switch State 1011 (11):

This case will be the red ball moving around the screen very close to what we did in the previous lab.

Switch State 1100 (12):

For the final case, we will use the other ROM data that we added to our code and have it display that pixelated image. We will have to resize the image to make it larger and change its position.

Result:

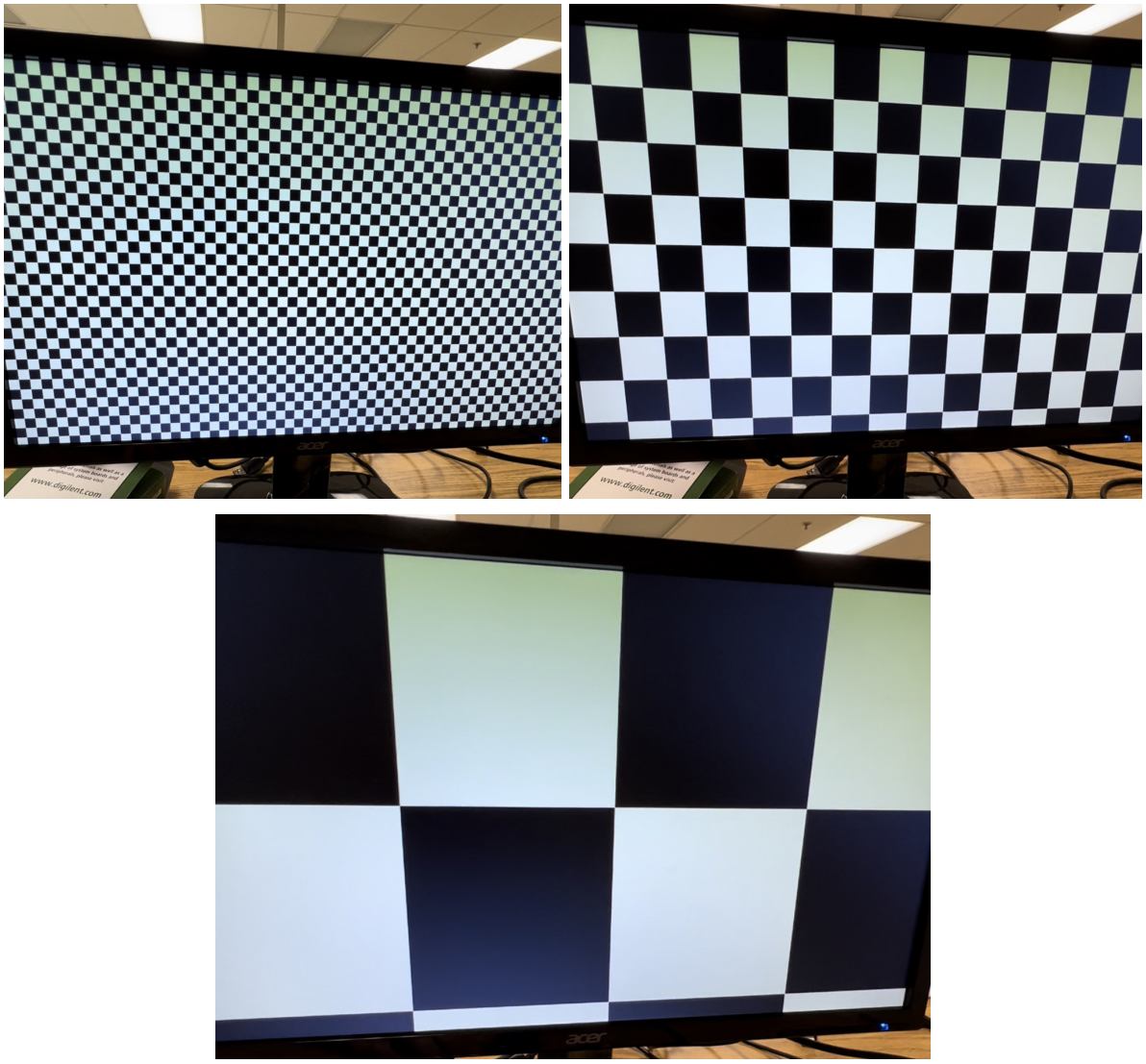

Fig 5.1 Various sizing that can be done when pressing the buttons

The three screenshots above in Fig 5.1 showcase how the VGA will display various sizes for specific switch and button combinations. However, there is one thing to note that is we have to hold down the button on the FPGA in order for the VGA to display a specific image size pattern. This is because of the fact that we did not include a debouncer logic and other button logic that would basically allow us to just simply press the button one time instead of having to hold it down.

Fig 5.2 Resized image of the 2nd ROM

For the 2nd image we needed to resize the image and move its placement. I was able to resize the image by adjusting the img_pxl. In order to do this I first needed to adjust the img_pxl signal assignment. Inside there were parameters such as: 250+27. What I did was remove the 250 and doubled 27 to 54 and doing so made the image appear much larger. The part that I struggled with was figuring out how to get the image in a better spot.

Conclusion:

In conclusion this lab showed us how we can further improve our VGA design from the previous lab by introducing more features to it. I really enjoyed this lab a lot because of how visual it was. It was really cool to see how we can code a variety of different things and have our code make stuff appear on the screen. I felt that this lab definitely gave more insight on the VGA's functionality and its capabilities. This lab also showed us how we can manipulate a set of bits to get the output on the screen differently. For example, we made signals that modified the frame width and by modifying that value we could adjust the color sizing and how much they take up the screen. It was also interesting to see how we could set the 3 vga signal colors to different combinations of 0 and 1 and that would show a mixed color.

Appendix:

Youtube: https://www.youtube.com/watch?v=lt1yGDbkMOs
Github: https://github.com/csun-ece/fa22-e524-lab5-level0-troykerim