

ECE371 Design of Digital Circuits and Systems  
  
Winter 2019

**Lab 3 - VGA Display Interface on FPGA**

Student: Jeff Josephsen  
Professor: Dr R. Hussein

February 2nd, 2019

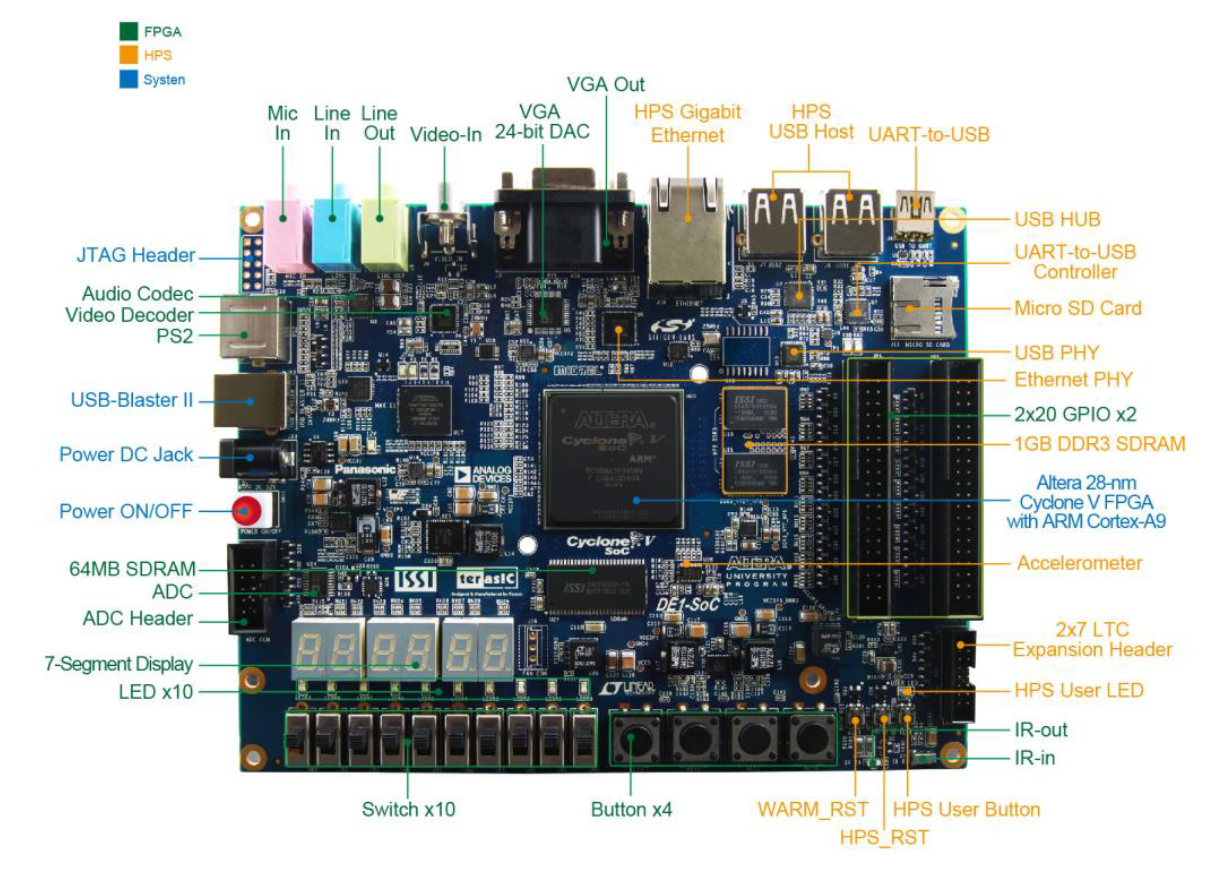
**Abstract:**

This lab was an investigation into the use of SystemVerilog software to utilize and update VGA controller firmware implemented on FPGA hardware. The VGA controller firmware comprising 500 lines of code was segmented into three modules. These modules included a top level module responsible for connecting to the hardware and both instantiating and directing the other modules, a line drawer module responsible for drawing individual lines in either black or white, and a VGA frame buffer module responsible for instantiating sufficient memory allocation and writing to or reading from the appropriative VGA ports with the correct timing.

**Introduction:**

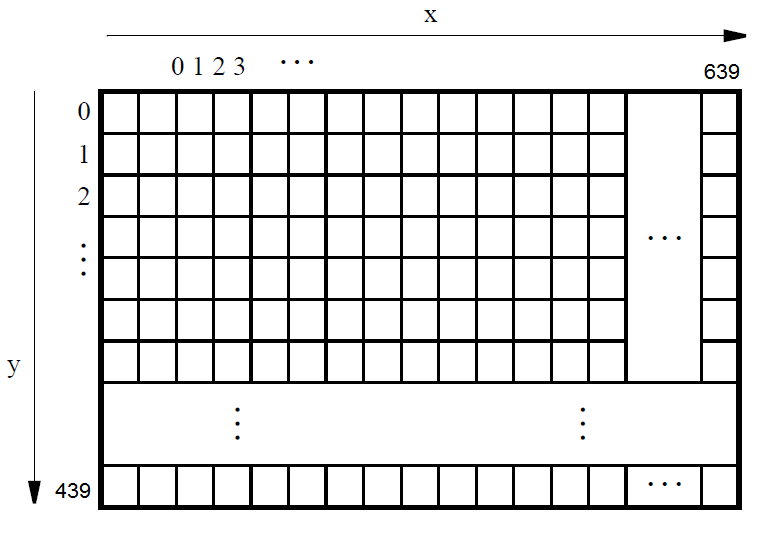
Relevant Background

The DE1-SoC Development kit as can be seen in the following image encompasses the Altera Cyclone V Soc 5CSEMA5F31 chip and peripherals. This development board has, among other things, 85K of Programmable Logic elements, a VGA 24-bit DAC, and a VGA output port.

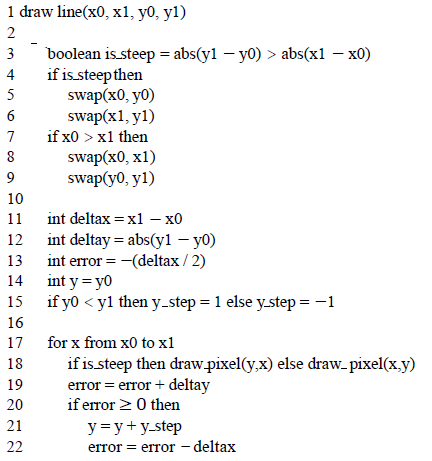


Those Programmable Logic Elements, VGA DAC, and VGA output port are used in this lab.

Also, the video output port of the board has a buffer for each pixel output to a 640 x 480 pixel VGA display. The layout of these pixels can be seen to begin in the upper left corner as can be seen in the following image.



In drawing a single line across the screen the software needed to find the appropriate Y value of each coordinate given the X value. Calculating these values was done by a firmware implementation of the Bresenham's Algorithm from scratch. As Bresenham's Algorithm is done in steps and FPGA Hardware Definition Language "HDL" programming is programmed in gates the real trick was to think through the algorithm to produce new logical assignments such that the appropriate parts of the algorithm would happen in sequential order. The following pseudo c-code was given to the students for use in the lab.

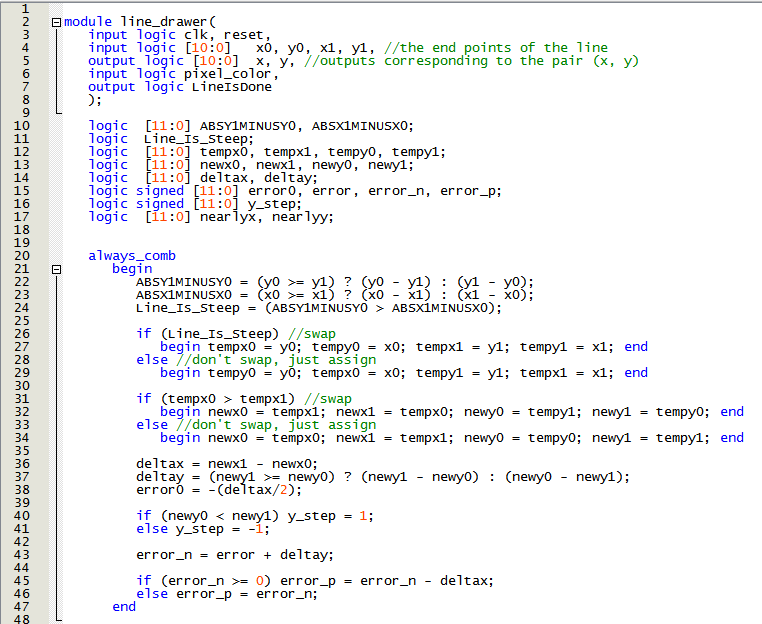


Purpose of the Lab

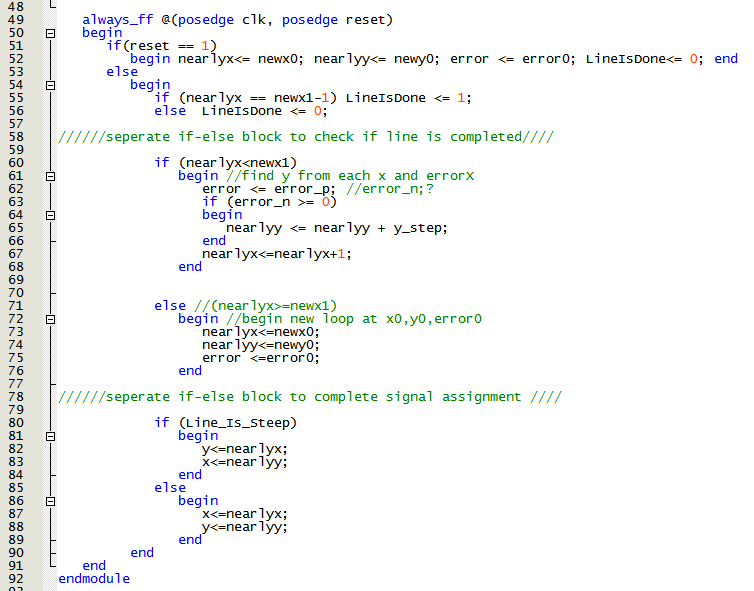
The purpose of the lab was to introduce the students to VGA out and use of memory buffers. Another focus of the lab was human level interpreting from a sequential algorithm to HDL code such that sequential behavior could be created appropriately. This focus was also seen later in the lab where the students was asked to create a moving animation of lines across the screen. Other skills were also exercised in the lab including instantiating modules, register, and wires. wiring module instantiations together and the appropriate inputs and outputs of the board. Also, the continued exercise in writing test bench simulations proved to be helpful.

**Procedure:**

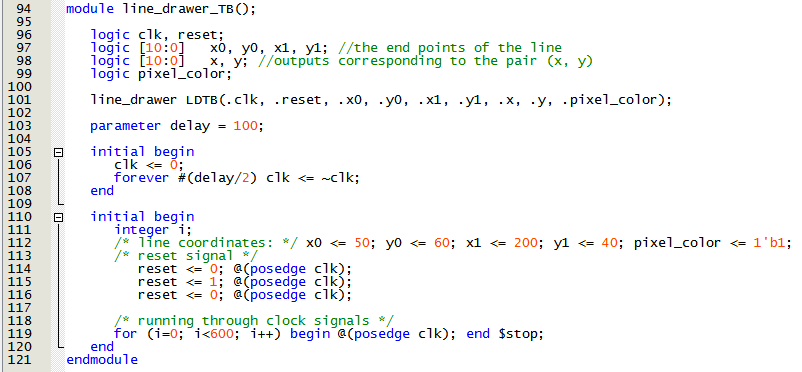
The lab was broken down in to three tasks.   
  
In the first task we were asked download a pre-coded Quartus project containing SystemVerilog Files. Then to compile and synthesize the code using system tools and to upload the synthesized code to the development board. Next, the board was to be connected to the video monitor by a VGA to HDMI cable.   
  
In the second task we were asked to update the line drawer module to include Bresenham's Algorithm. Parts of the algorithm are best handled in an always\_comb block and other parts of the algorithm are best handled in ah always\_ff block. The following image shows signal declarations and always\_comb block of that algorithm as written in SystemVerilog Code.



The next image shows the always\_ff sequential part of the algorithm.

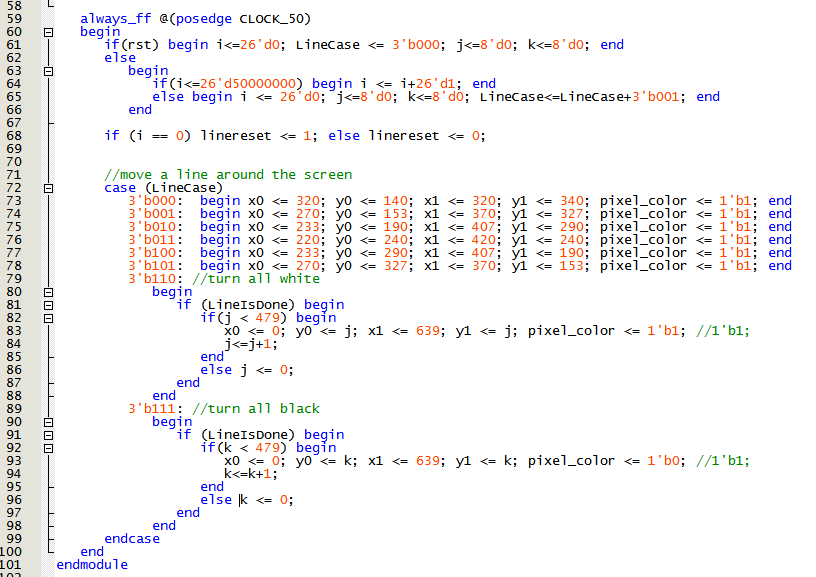


Lastly, a testbench was written to debug any signals that were not instantiating correctly or giving the correct result. The code for the testbench can be seen in the following image.



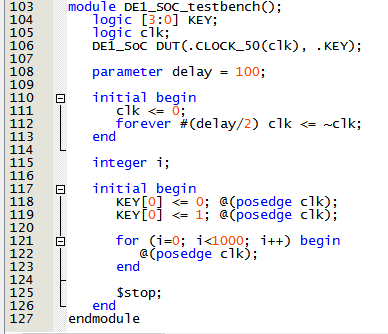
The results of the ModelSim waveform results of the test bench simulation can be found in the results section of this report.

In the last task of the lab we were asked to create a moving animation of lines and also a clearing of the screen. The implementation of this section of the lab was done in the top level module in eight cases. Between each case was a 50 million clock cycle delay which is approximately one second. Of the cases six cases were given to drawing six lines creating an image. One case was given to drawing 480 consecutive horizontal white lines from the top of the screen to its bottom to turn on the whole screen. The last case was given to drawing 480 consecutive horizontal black lines to clear the whole screen. The majority of this code can be seen in the following image.



The resulting image as produced on the screen can be seen in the results section of the lab.

Also, a testbench was written to test the top level module and the animation. The code of the test bench can be seen in the following image.



This test bench was helpful in debugging signals that were not instantiating as expected or that were not increment as expected.

**Results:**

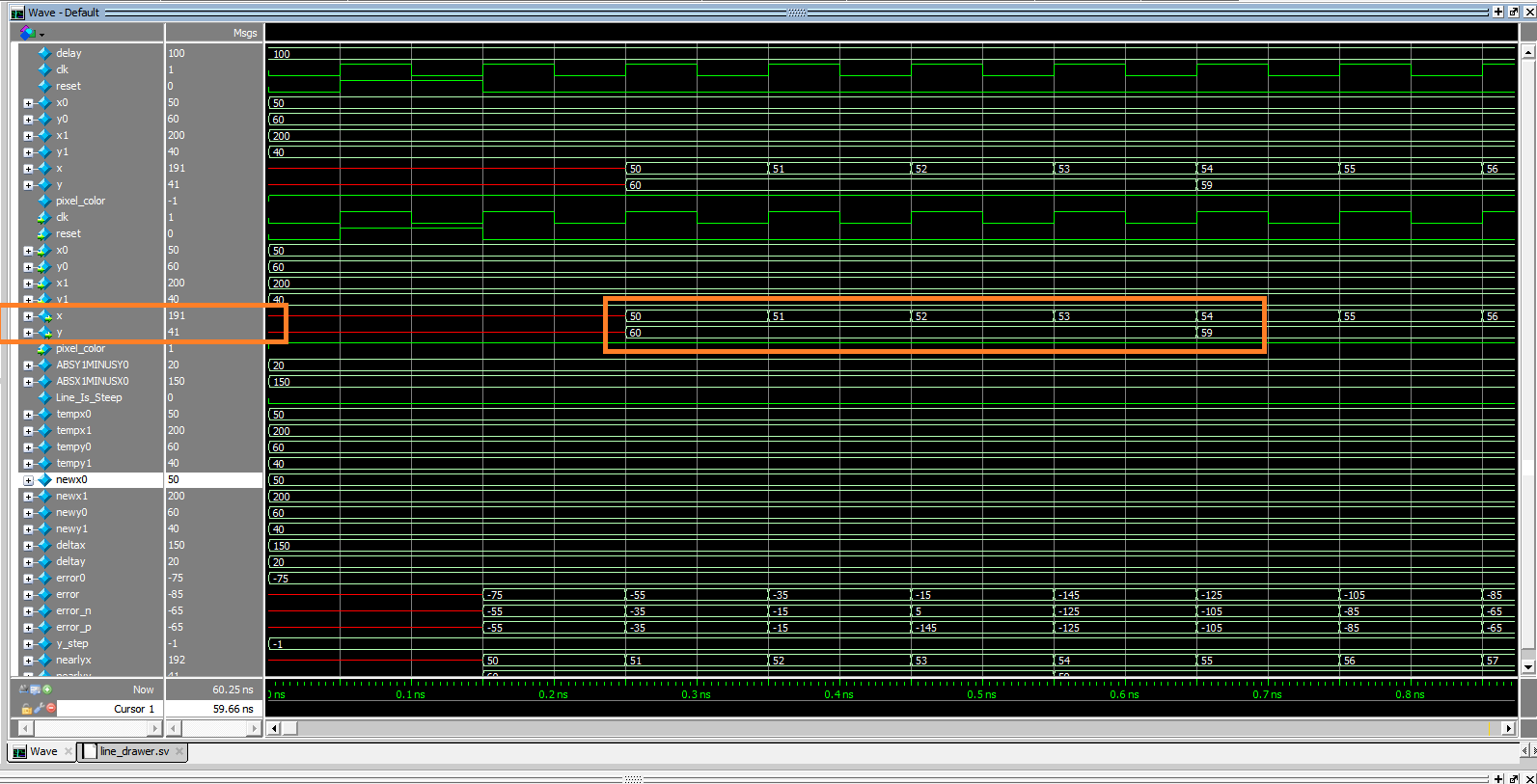
*Here, present and discuss your findings. Explain them, and how (or if) they differed from the*

*expected. Use graphics to help elucidate your results. Answer any questions raised in the lab*

*report. Also use this section to talk about any known errors in your lab, or any challenges you*

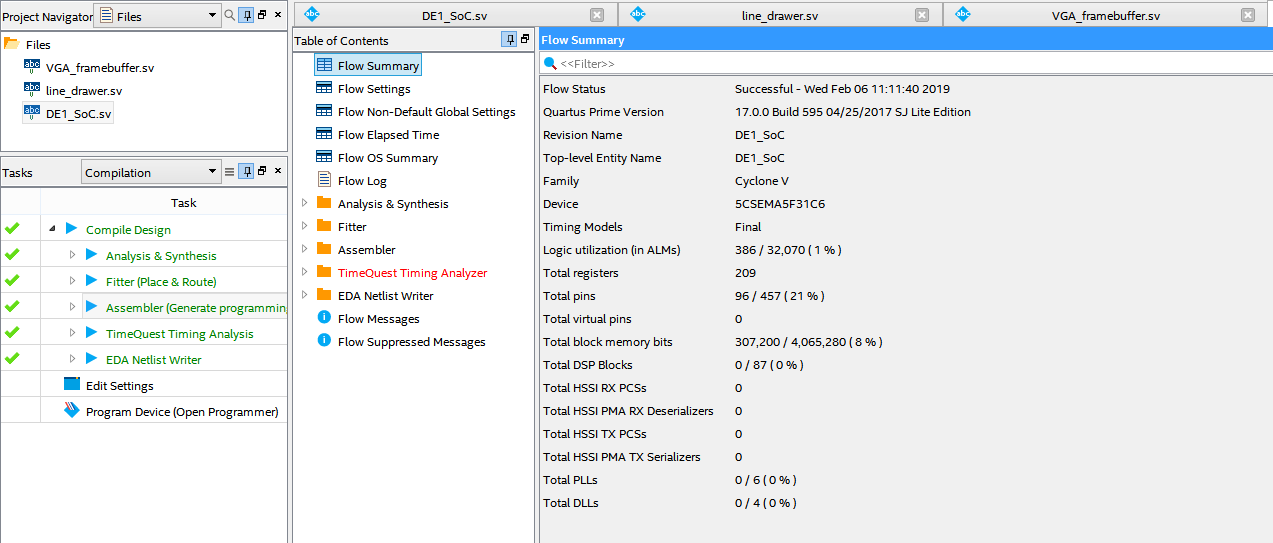
*faced.*

The following image shows the line drawer test bench simulation. In the orange boxes it can be seen that the output of the as-implemented Bresenham's Algorithm for line drawing produces the as-expect y-values while stepping through the x-values. The lines above the orange boxes show (x0=50, y0=60) and (x1=60, y1=200) which are the preset end points of the line.



The algorithm was shown to produce appropriate points along the line for both steep and non-steep lines. This can be seen in the following image of the VGA output from the last task of the lab. While a video cannot be shown in this paper it must suffice to say that the software write the lines once per second and both illuminated and cleared the screen as expects at seconds 7 and 8.

Lastly, I have include a flow summary of the complete compiled system. It can be seen in the following image.



**Conclusion:**  
I thought this lab did a great job of requiring me to use skills in interpreting from generic c-code to HDL SystemVerilog code. It required me to write test benches and simulate digital waveform results in ModelSim. I was required to think outside the box in creating a sequential animation and in updating modules such that new lines and their color could be updated within the animation. I thought this lab was very helpful in teaching the additional uses of FPGAs and becoming more familiar with SystemVerilog.

I believe that this lab took approximately 30 hours.