Connor Aksama
EE 371
January 27, 2023
Lab 3 Report

Procedure

This lab is comprised of two tasks. The first task involved implementing Bersenham's algorithm in Verilog and demonstrating its functionality via a VGA display; the second task involved utilizing this line drawing algorithm to animate a line on the video display.

Task 1

I approached this task by first planning out the implementation of the C-style algorithm in a Verilog module. I determined that the initial part of the algorithm before the main loop should be contained with the combinational block of the module since these calculations were entirely based on the input points. Thus, the portion of the algorithm contained in the main loop would belong in the sequential block of the module since the output values were dependent on the previous sequence of values.

A testbench was then created for this module, simulated in ModelSIm, and was compared against the output from a procedural implementation of the algorithm for correctness.

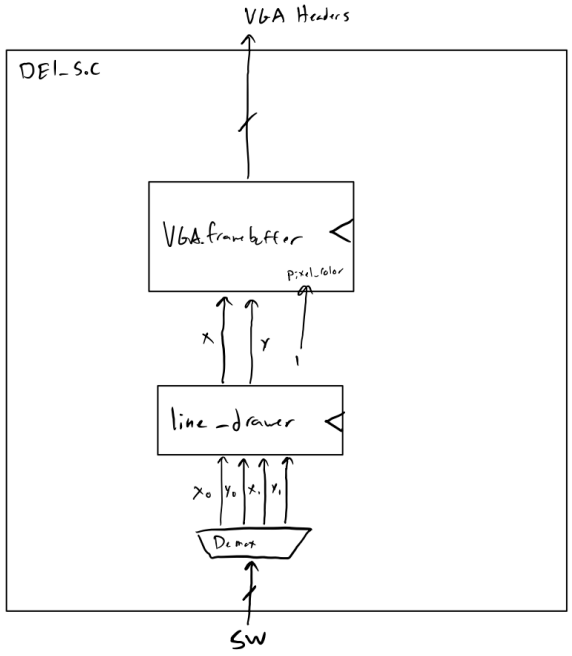


Figure 1. Top-Level Block Diagram for Lab 3 Task 1.

Task 2

I approached this task by first planning out the animation that was going to be displayed. I decided that I would implement this animation as a sequence of timed “frames” (pairs of points) interlaced by screen clears. To achieve this, I created a timed module that would cycle through a series of pre-programmed points once an internal timer reached a certain value. The module would output the next pair of points to be drawn on the screen, which was fed into the line drawing module from Task 1.

Additionally, a module that would clear the screen when enabled was created. The function of this module is simply to cycle through and output each coordinate on the screen in sequence. This point is directly fed into the VGA buffer module along with a black pixel color. Given enough cycles, this would clear the entire video display.

These modules were then tested and simulated in ModelSim before uploaded and tested on hardware.

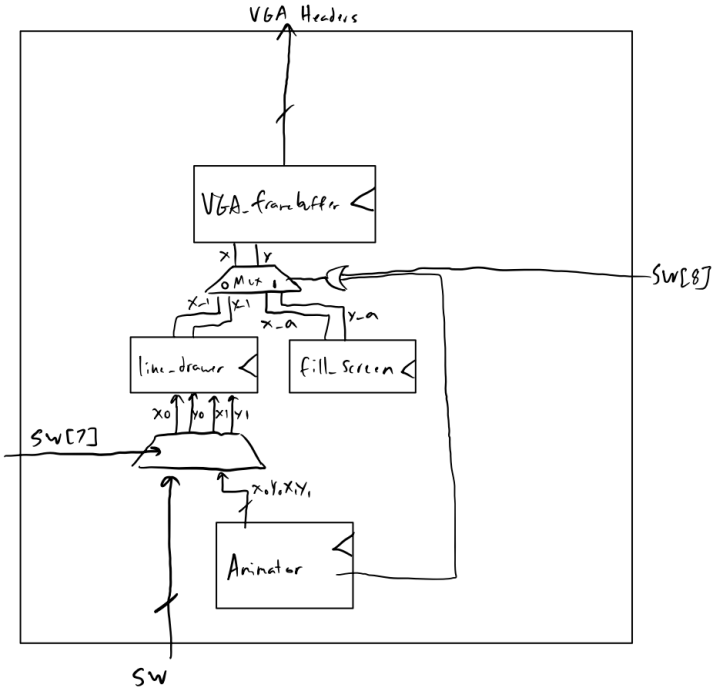


Figure 2. Top-Level Block Diagram for Lab 3 Task 2.

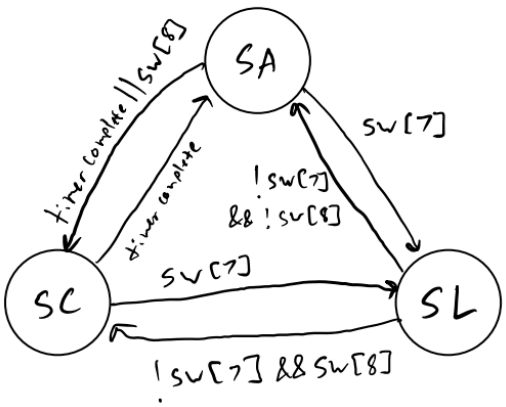


Figure 3. FSM for the Lab 3 Task 2 Design.

This FSM transitions between types of output to the VGA display. In state SA, a line is drawn to the screen whose endpoints are determined by a sequence stored in the animator module. After an internal timer in the animator module expires, it outputs a clear signal (state SC), which triggers the fill screen module to write to each pixel on the screen. This state is also enterable when SW8 is raised and SW7 is low. If SW7 is raised (state SL), the endpoints of the line are determined by the status of the input switches.

Results

Task 1

The following are screenshots from ModelSim simulations for each module used in the simulated RAM module.

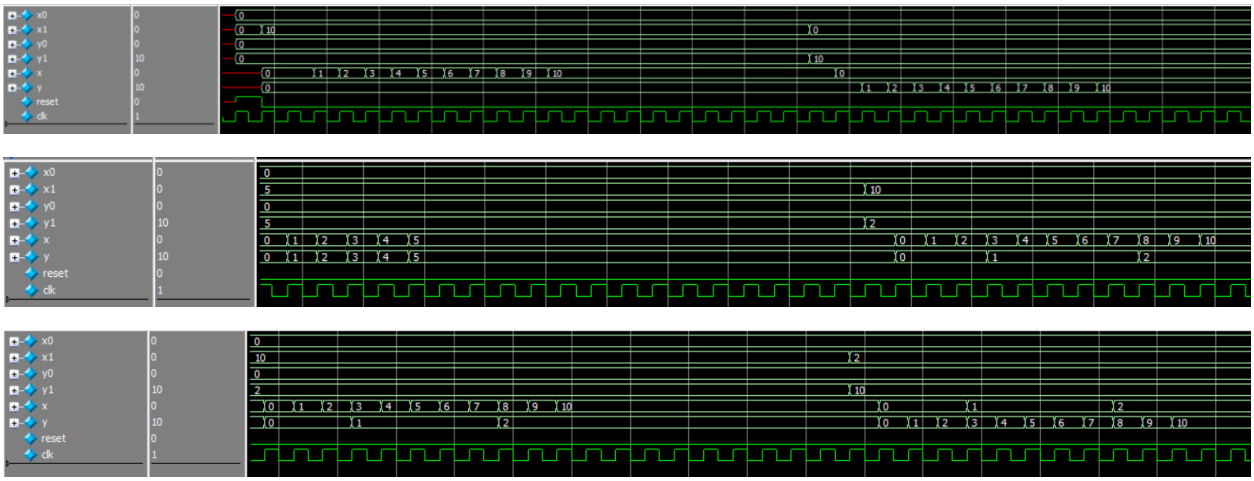


Figure 4. ModelSim Waveform for the Line Drawer testbench.

This testbench runs through different types of line slopes: horizontal, vertical, diagonal, shallow and steep. The output x and y were compared against the output of a C implementation of Bersenham's algorithm, which verified that the module was outputting the correct values for x and y each clock cycle.

The testbench for the top level module can be found in the following section.

Task 2

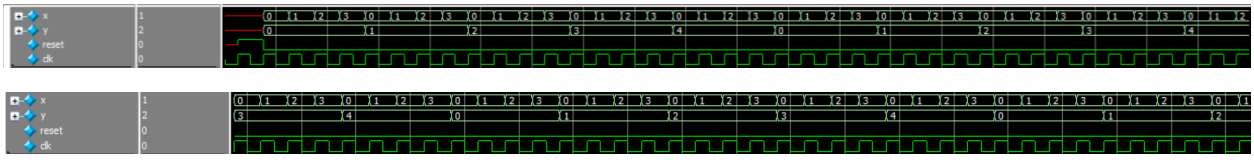
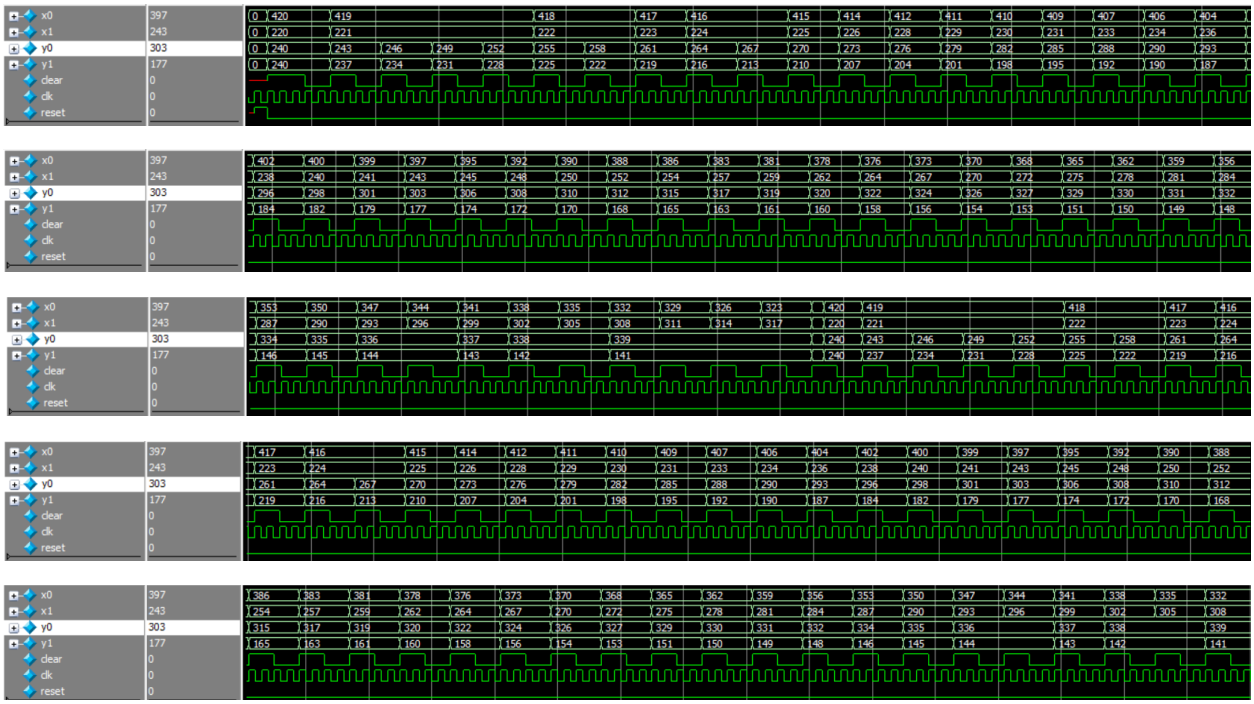


Figure 5. ModelSim Waveform for the Fill Screen testbench.

This testbench instantiates the fill screen module for a test screen with width 4 and height 5. The testbench then monitors the output of the module, which is expected to cycle through each pixel on the screen. The waveform shows that every pixel on this screen gets output and cycles back to (0,0).

Figure 6. ModelSim Waveform for the 32-bit Counter testbench.

This testbench shows the counter being incremented, decremented, underflowed, and overflowed correctly.



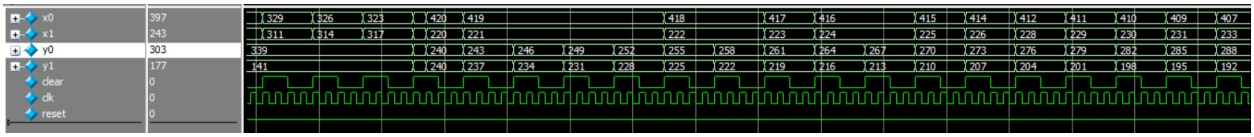


Figure 7. ModelSim Waveform for the Animator testbench.

This testbench shows the animator cycling through each “frame” (set of line endpoints) being sequentially output after the specified frame duration of 2 clock cycles (for testing purposes). After two cycles of the current frame being output, the clear signal is raised for 2 cycles, then the next frame is output. This process is continued for every frame in the sequence, which then cycles back to the beginning of the sequence once the last frame is output.

Figure 8. ModelSim Waveform for the top level DE1_SoC testbench.

This testbench first shows a portion of each of the arbitrary lines drawable in the SL state (i.e. when SW7 is raised). Next, the simulation demonstrates that when SW7 is lowered and SW8 is raised,

Final Product

The overarching goal of this lab was to gain experience working with a VGA controller, a VGA buffer, and implementing a procedural algorithm in Verilog. The main result of Task 1 was a module that implemented Bersenham's algorithm by taking in two cartesian coordinates as input, and outputting the coordinates of the line between the two points. As requested by the spec, the functionality of this module was then demonstrated by drawing different arbitrary lines on the display, which were toggled using the onboard switches.

In Task 2, I created additional modules to display an animation on the display. When the animation was enabled, a line would rotate and jump around the screen in a cyclic manner. As stated in the spec, this display also had the ability to be cleared to black, which is achieved by turning on an onboard switch.

Appendix: SystemVerilog Code

(See next page)