### EE 371 Lab 3 Report

Jiarong Qian

1. Introduction

In this lab, I implemented a line drawing module based on Bersenham’s algorithm. I first draw a fixed straight line on the screen, then I used another module to move the line around the screen.

1. Lab Procedures

In order to draw a line on the screen, I wrote a module that takes in the coordinates of the start and end point and walks through every point on the line. The coordinates of all points in between will be inputted to the VGA buffer writing module and draws the line.

The module first determines whether the line is steep or not by comparing the difference between the start and the end for x and y coordinates. If the line slope is steep, it switches the x and y coordinate so that we can always have a line function with slope less than or equal to 1. Then the module compares the start and the end of x to determine if we need to switch these two so that x is always increasing. Next, the module determines if y is increasing or decreasing by comparing the start and the end of y.

For each clock cycle, the x increments by one. The error value, which is set to - [(x\_end - x\_start) / 2] initially, increments by |y\_end - y\_start| as well. Once the error is non-negative, y would increase by 1 and error would decrease by (x\_end - x\_start). If the line is steep, the x is outputted as y and the y is outputted as x due to the previous switch, otherwise they are just outputted correspondingly. In this way we can get a line similar to the slope of the original line.

After drawing a fixed line successfully, I made the line move around the screen by periodically updating the start and end point. I implemented a module of a three-state FSM. The state diagram is given below.

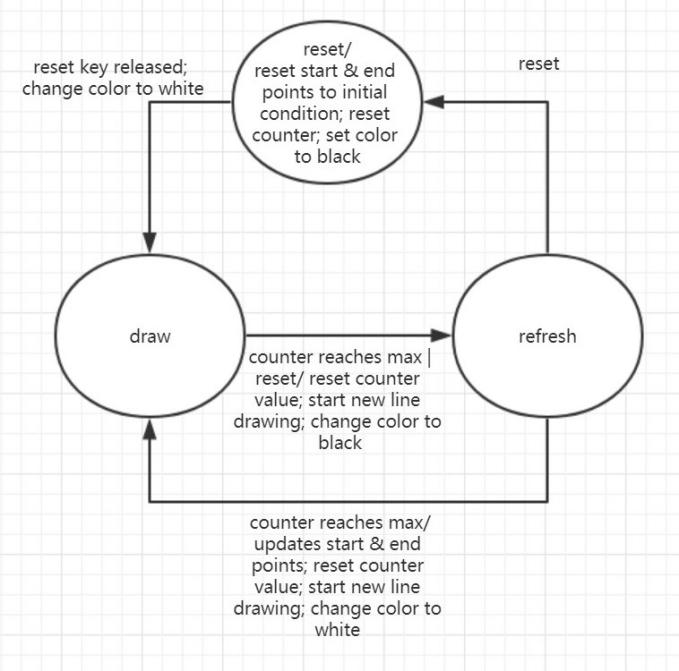


Figure 1. State Diagram of line\_animation

I introduced a counter because my line\_drawer and line\_animation module runs at the same clock. Since line\_drawer only draws one point at each clock cycle, I need to give enough delay so that the line\_drawer can finish drawing.

After the counter reaches the maximum, the drawing would be finished, the system goes to the state of refresh and redraws the same path but with color black. Before moving to the next drawing state, the system updates the new coordinate and also makes sure that the coordinate stays within the bound. If the reset switch is enabled, the module goes immediately from drawing to refreshing. After clearing the screen, the system stays at reset until the reset switch is off.

1. Results

By putting the modules into modelsim, I tests the modules and made sure that they were working correctly. After loading the system onto the FPGA board and connect the board to a computer screen, there was a line moving around the screen, though there are boundary conditions that there are some white lines at the edge of the screen.

1. Problem Faced & Feedback

I spent a long time working on the lab, probably 30 hours. I initially did not use Bersenham’s algorithm. Instead I used pure math to get the coordinates and the problem with that was there was overflow happening all the time and the coordinate values suddenly shift. After I used Bersenham’s algorithm, this problem was solved.

I spent even more time on getting the line moving. The module was hard to simulate when I used different clocks for drawing and animation so I chose to let them run on the same clock and use a counter to introduce a delay. Additionally, when the counter maximum is small, the drawing happens really fast and the lines look like curves. They are also discrete because the screen clearing happens fast as well.

Appendix

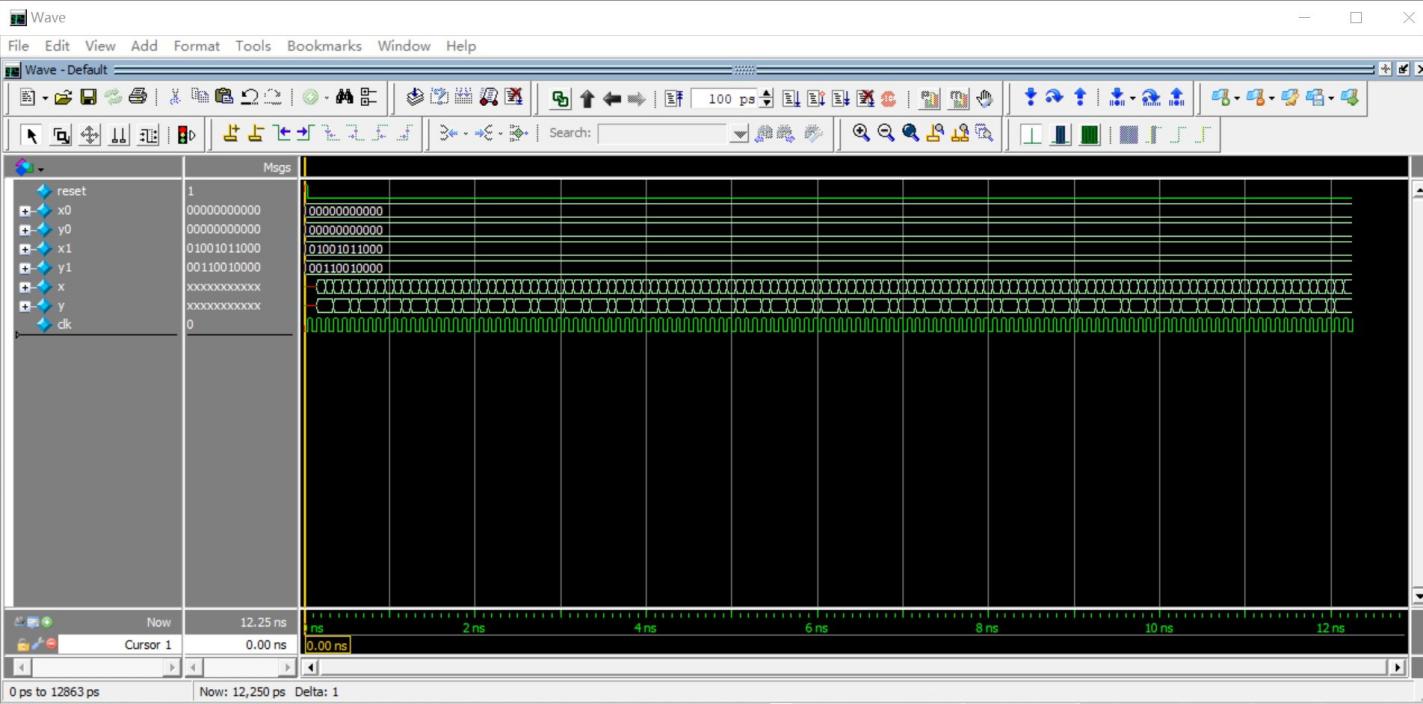


Figure 2. line\_drawer Simulation

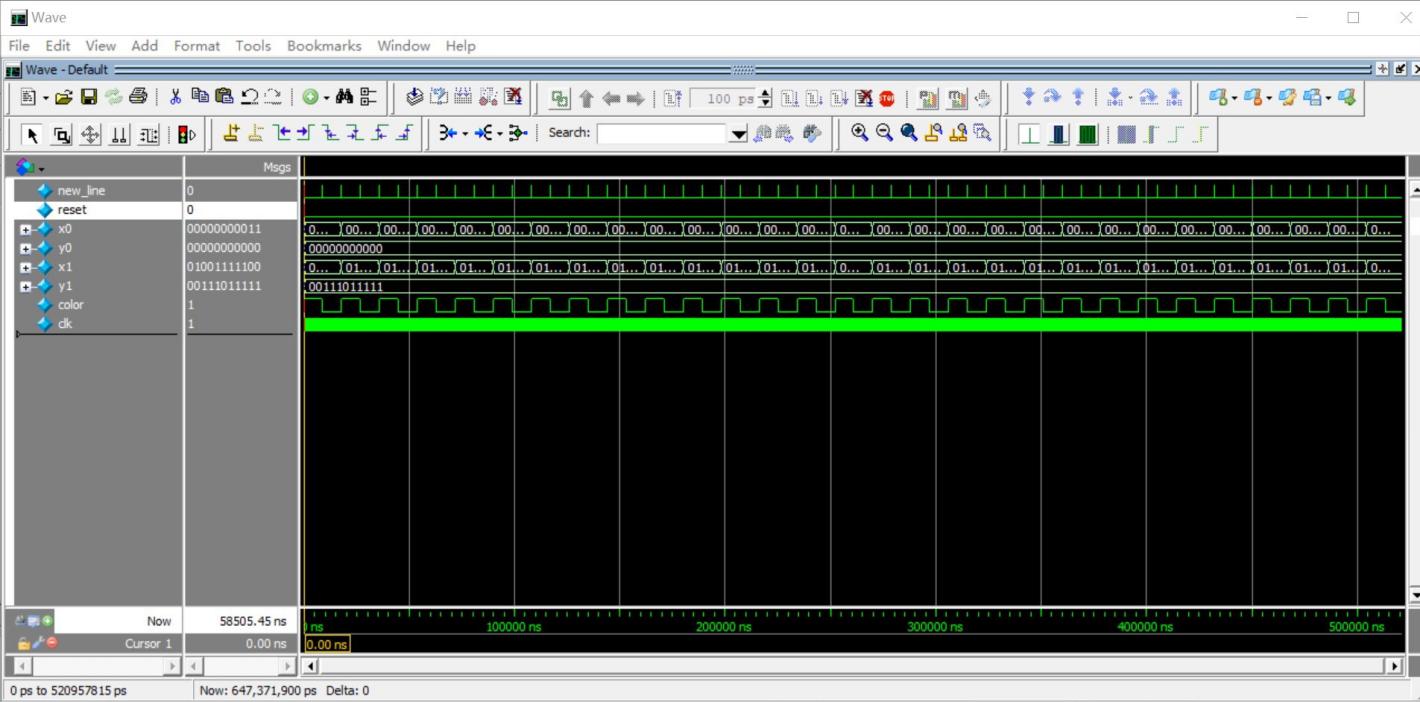


Figure 3. line\_animation Simulation

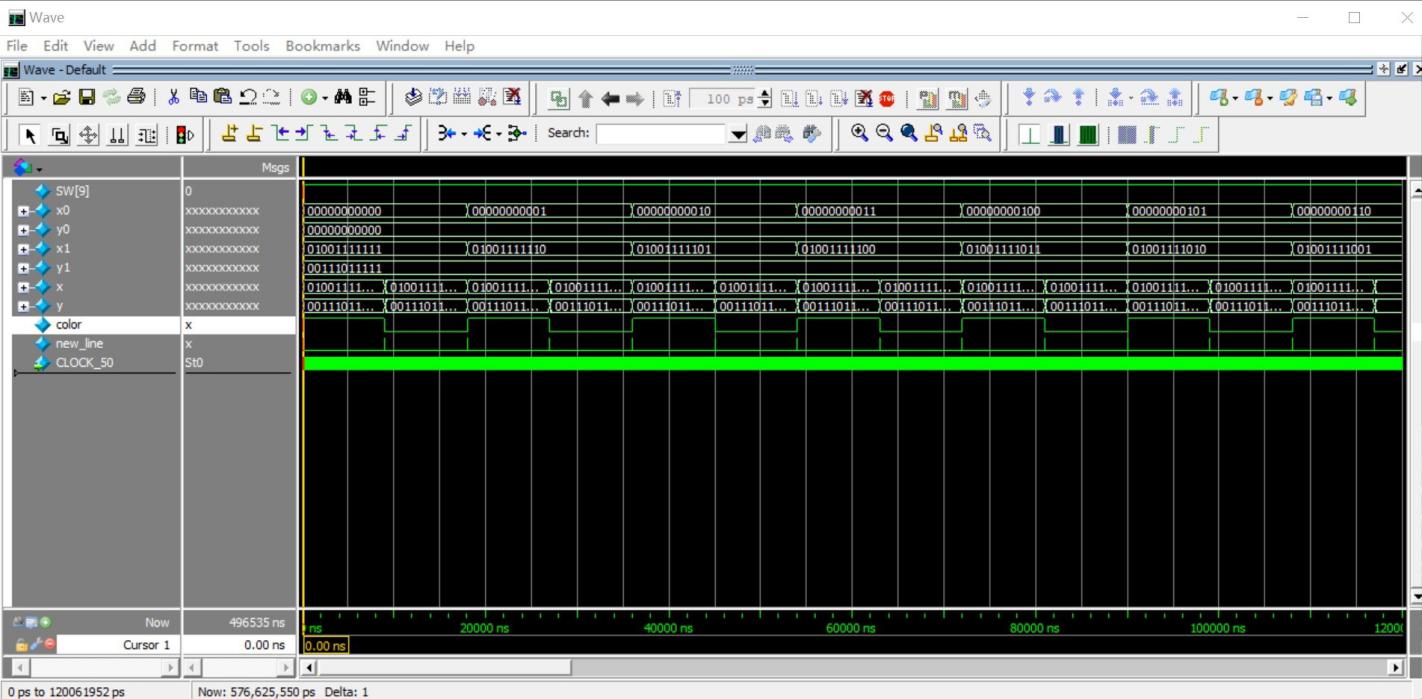


Figure 4. DE1\_SoC Simulation