ECE342 - Computer Hardware

University of Toronto

Lab 4: Implementing Circuits with Algorithmic State Machine (ASM) Charts

Introduction

In class you have been familiarized with Algorithmic State Machine (ASM) charts and how to use them to build logic circuit. In this laboratory exercise we apply ASM charts to create a circuit to draw line segments on a monitor, using a line drawing algorithm (LDA) known as Bresenham's line algorithm.

Description

Bresenham's line algorithm is a straight forward approach to place pixels on a monitor screen in such a way as to approximate a line. The idea is to draw a line between points (x_0, y_0) and (x_1, y_1) by computing the slope of the line. As each pixel is drawn it is important to decide where to place it to approximate a straight line as closely as possible. This idea is shown in Figure 1.

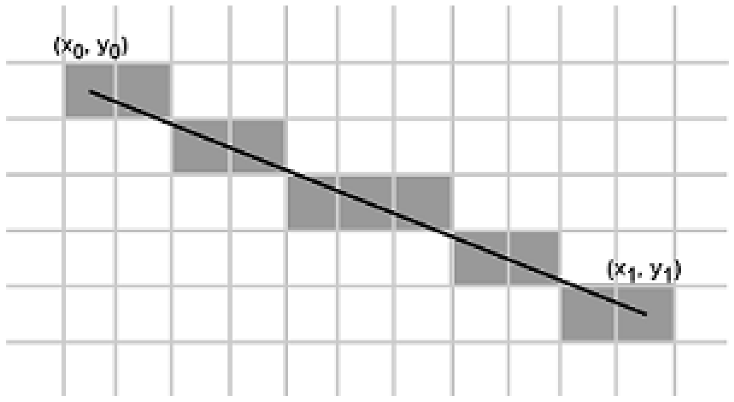

Figure 1: Approximating a line using pixels

The algorithm uses floating point values to compute the slope, which for us is not very desirable. This is mostly because to use floating point mathematics in a digital circuit requires a lot of work to create a floating point unit. It is also beyond the scope of this lab. Fortunately, a similar obstacle has been encountered early in the history of computer graphics and the algorithm has been generalized and optimized to use integer mathematics, with addition, subtraction and shifting operations only. The generalized and optimized Bresenhams line algorithm is shown in Figure 2.

In the above algorithm the function called *plot(x, y)* draws a pixel at (x, y) on the screen. This particular function will be facilitated by a VGA Adapter provided to you in the starter kit.

The VGA Adapter provided has inputs **X**, **Y**, **colour**, **CLOCK_50** and **plot**. When the input plot toggles high and the positive edge of the CLOCK_50 occurs, the VGA Adapter draws a pixel with a 3-bit colour at location (X, Y) on the screen. The valid (X, Y) coordinates are between $(0, 0)$ and $(319, 239)$. There are also a number of outputs that are meant to directly drive particular pins on the FPGA. These pins are connected to a Digital-To-Analogue (VGA DAC) converters that control a monitor. A shell file with an instantiated VGA Adapter is already

```
function Bresenhams_line_algorithm($x_0, y_0, x_1, y_1$)
{
    bool steep = $abs(y_1 - y_0) > abs(x_1 - x_0)$;
    int deltax, deltay, error, ystep, x, y;

    /* Preprocessing inputs */
    if (steep) {
        swap($x_0$, $y_0$)
        swap($x_1$, $y_1$)
    }
    if $(x_0 > x_1)$ {
        swap($x_0$, $x_1$)
        swap($y_0$, $y_1$)
    }

    /* Setup local variables */
    deltax = $x_1 - x_0$;
    deltay = $abs(y_1 - y_0)$;
    error = $-(deltax/2)$; /* This is a right shift by 1 */
    y = $y_0$;
    if $(y_0 < y_1)$
        ystep = 1;
    else
        ystep = -1;

    /* Draw a line */
    for $(x = x_0; x <= x_1; x++)$ {
        if (steep)
            plot(y,x);
        else
            plot(x,y);
        error = error + deltay;
        if $(error > 0)$ {
            y = y + ystep;
            error = error - deltax;
        }
    }
}
```

Figure 2: Generalized and Optimized Bresenham's line algorithm.

provided for you in the starter kit. Outputs of the VGA Adapter are already connected correctly, so you need not worry about them.

The circuit you are to design is supposed to take an (X, Y) coordinate from a user and draw a line from point (X_0, Y_0) to point (X, Y). Once the line is drawn the circuit is to update (X_0, Y_0) to be (X, Y), and allow the user to input another point (X, Y). As a result a user should be able to draw a set of connected lines by providing the coordinates of subsequent line endpoints. Assume that initially $(X_0, Y_0) = (0, 0)$.

To help you complete the lab, a system level diagram is provided in Figure 3. Note that not all signals are shown. You will have to complete the diagram as part of your preparation.

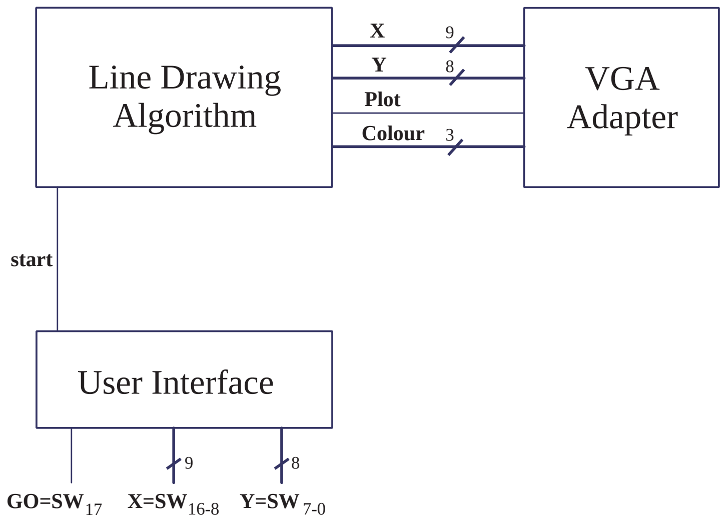

Figure 3: Incomplete system diagram.

Preparation

To prepare for this laboratory you will need to design this circuit by creating a set of ASM charts to represent the operation of the circuit. The first ASM chart should describe a user interface (UI). The user interface must allow a user to specify a point (X, Y) and wait for a GO signal to be asserted. When the GO signal is asserted your UI needs to start up the line algorithm state machine and wait for it to finish. Once the line is drawn the UI will update the (X_0, Y_0) coordinates and then allow the user to enter a new point (X, Y).

The second ASM chart must implement Bresenham's line algorithm as shown in Figure 2. The FSM must not process data until its start signals is triggered. Also the FSM must produce a done signal when the algorithm is finished. The ASM chart must correctly address the timing (i.e. scheduling of operations to specific clock cycles) of the circuit to ensure proper operation of the circuit.

Your preparation to be presented to the TA at the beginning of the lab is as follows:

Part 1
- An ASM chart for the LDA.
- A diagram of the circuit elements needed to implement the LDA datapath.
- A second ASM chart that represent only the FSM needed for the LDA control circuit.

Part 2
- An ASM chart for the UI.
- A diagram of the circuit elements needed to implements the UI datapath.
- A second ASM chart that represent only the FSM needed for the UI control circuit.

Part 3
- Complete the system schematic diagram in Figure 3. The three main blocks are shown, but you must specify the remaining signals and their direction.

Part 4
- Verilog source code and simulation for the LDA
- Verilog source code and simulation for the UI
- Verilog source code and simulation for the complete system, i.e. circuit consisting of the UI, the LDA and the VGA Adapter

IMPORTANT: Your simulation must be printed and commented prior to the lab. The comments you place on the simulation must clearly indicate that you understand how the circuit operates as well as show that your circuit is designed correctly.

In the lab

In the lab you will be asked to test and demonstrate your circuits on the Altera DE2 board and the VGA screen.

- Demonstrate your implementation of the LDA (without the UI). Draw a line between a pair of points on the screen. You can set the pair of points inside your code, i.e. the start and end points should be constants in your Verilog code. Any change of the start/end points should be reflected accordingly on the monitor display after recompilation of your circuit.
- Show the complete system running with both the LDA and the UI.

Notes

To simplify some of the steps a starter kit has been provided on the course website. The starter kit is a ZIP archive containing a Quartus II project with a shell Verilog file.

In the labs, copy your files to the local hard drive (`C:`). The tools run much faster when they access local disk. The network drive and USB stick are much slower devices. Make sure to backup files once you are done with the lab.

Marking scheme

Preparation

- Part 1 (2 marks)
- Part 2 (2 marks)
- Part 3 (1 marks)
- Part 4 (3 marks)

In-lab

- LDA operation (4 marks)
- Complete system operation, LDA and UI (4)