Paint Tool Program

Duncan Taylor, Cabir Yavuz, Daniel Wolfe
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
dttaylo2@oakland.edu, cyavuz@oakland.edu, DaWolfe2@oakland.edu

I. Introduction

The purpose of this project is to create a program that will allow a user to draw images to a display monitor with a mouse. The user will control the focus of mouse's position with a ps/2 mouse. The user will make color selections by using the left click button on the mouse. Then, using that same button will be able to draw by holding that button down and simply dragging the mouse in any desired direction. When a new drawing is desired, a clear switch, located on the FPGA, is switched into active mode (high). The drawing area is set back to its initial color value. Switching the switch back into non-active mode returns the program into a condition where the user can begin drawing once again.

II. Methodology

A. Program Overview

The top level of this program contains 2 modules. Shown in figure 1, it consists of the "VGA_Control" and the "mouse_top" modules. The "mouse_top" module is what gets the signals from the mouse, makes calculations based on that information, and then outputs information to the "VGA_Control." It will be in control of mouse "pointer's" location, the "clicker" signal, which controls the writing function, and the color it needs to write. The "VGA_Control" contains the modules that control the display and the RAM. We switched the RAM from the referenced "VGA_Control" material with a dual port RAM. This allows the RAM to be constantly reading while also writing when invoked by the "mouse_top."

B. Mouse

The mouse component uses a ps/2 configuration to input and receive data from the mouse. Mouse button signals and 9 bit x and y directional signals are received from the mouse. The left mouse button signal was a single bit that is sent straight to the VGA control module. The 9 bit directional signals first needed to be configured for movement in the mouse module. The most significant bit from this 9 bit signal is the parity bit; this determines the direction the mouse has moved. The rest of the signal needed to be converted into positional coordinates to represent a mouse cursor on the VGA. To represent a mouse cursor, two 10 bit signals corresponding to x and y values were made that would represent a specific pixel. Figure 2 shows a display of the data received from the mouse. To simulate mouse movement, the 9 bit signals were taken and converted into left, right, up, or down movement. If the mouse was determined to move left, then the 10 bit x-direction signal would decrease, moving left on the VGA. With this functionality in place, the x and y mouse cursor coordinates were output from the mouse module along with the left mouse button signal and sent to the VGA control module.

C. VGA Control

The primary output of the Paint Tool project is the VGA. The VGA outputs RGB color values, a horizontal sync, and a vertical sync which determines the colors and shapes that are being displayed. The module sets a 3 bit color to every pixel of a 640 by 480 resolution. In order to control what color is written to each pixel, two signals called hcount and vcount are used. These 10 bit signals cycle and correspond to every pixel that is being displayed on the VGA. Using these two signals with the color input, a multiplexor can be used to display different colors in different areas on the monitor. The multiplexer has an RGB input for every object that is to be displayed and outputs each color during a specified interval of hcount and vcount.

In our project, there are many different objects displayed from the multiplexer but two of them are constantly changing, important values. These values are the mouse cursor and the writing area. The mouse cursor can move to any pixel on the VGA and always need to be displayed. In VHDL, *if* and *elsif* statements are used to determine which multiplexer input is sent to be output. So this object is the first *if* statement for the multiplexer as it takes priority over all the other statements. The other important object is the writing space. The writing space is displayed from the RAM component which constantly reads every address. The only part of the VGA we want to be able to write and read from is the writing space. So to make the process simpler, we coded the project to write and read from every address but set the multiplexer to display these values with the least priority. The writing space is the *else* statement for the multiplexer so that all the addresses of RAM are read but only displayed in the writing space.

D. Dual Port Ram

One of the most important components in the paint tool project was the dual-port RAM. Since a large number of bits needed to be stored and displayed on the VGA; a

memory component was needed. Due to how this project worked, a single address with a read or write bit would not work. The paint tool needed to constantly read from every address to keep displaying the drawing on the VGA while being able to write whenever desired. Dual-port RAM uses two addresses; one address is read from at all times. This address is set to the same values as hcount and vcount so that the address is cycled through all pixel values and displays the color corresponding to what is written. The other address has a write bit that is set on or off. This address is set to the mouse cursor so that whatever pixel the mouse cursor is on, the address that writes is set to the same pixel. The write bit for this address is set high when the left mouse button is pressed down. With one address following the mouse cursor and able to write with the mouse button, and the other address cycling through all the pixel values and reading from them, we are able to write to the RAM and read from it separately.

III. Experimental Setup

This project utilizes a Nexus4 FPGA that interacts with an LCD monitor and a ps/2 mouse. The mouse will use 4 directional outputs and a left click signal. The right click and wheel signal will not be used. The mouse and monitor are connected directly into the FPGA.

IV. Results

This program successfully writes new color data to each individual pixel when requested by the mouse unit module. The user is able to draw pictures and select from any eight colors available to draw with. When the clear signal is sent by the FPGA's switch all the pixels in the drawing area are reset to their initial color value being white. The clear switch is return to the off position from which the user may begin drawing again. Figure 3 shows an example of a drawing being drawn successfully.

Conclusions

This program only just breaks the surface of what features can be added. Some functions that could be added are a way to save the current drawing, change the size of the cursor when drawing, add text, and draw saved images such as shape. A problem that was encountered dealt with memory space. Having the limited memory space caused us to limit the amount of colors available. Originally we wanted to use 12 bits to represent color but had to reduce it to 3 bits to get the program to compile.

References

[1] Chu, Pong. "FPGA PROTOTYPING BY VHDL EXAMPLES," John Wiley & Sons. 2008. Roy. pp. 199–212, 249-250.

[2] "VGA Control"-.http://dllamocca.org/Fall2013_WorkshopVHDL.htm

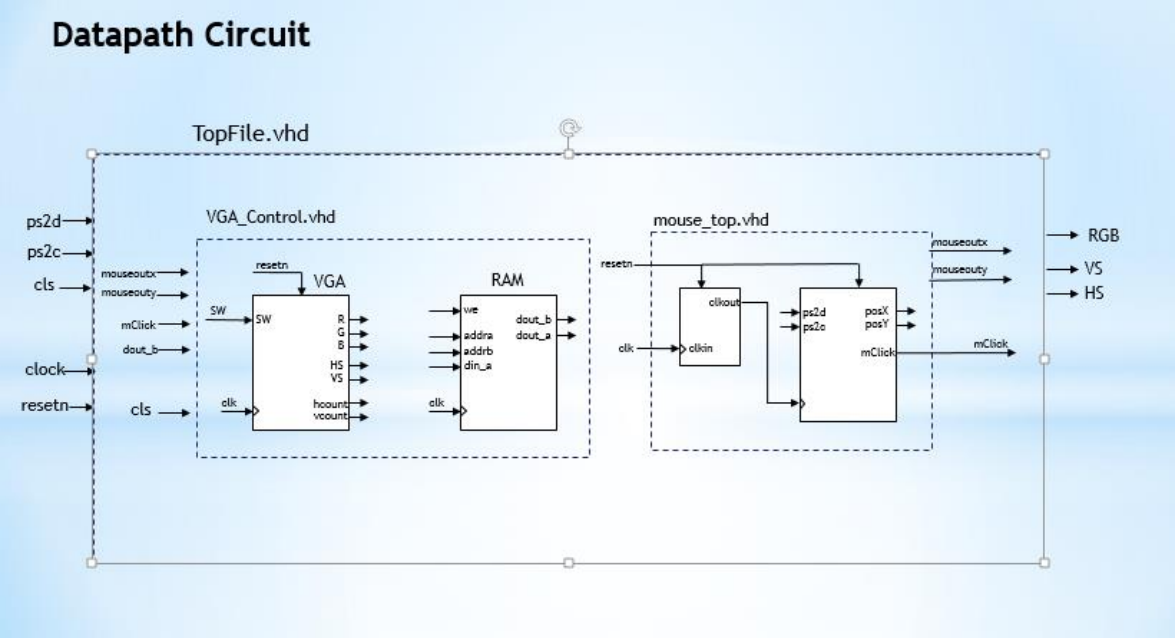

FIGURE 1

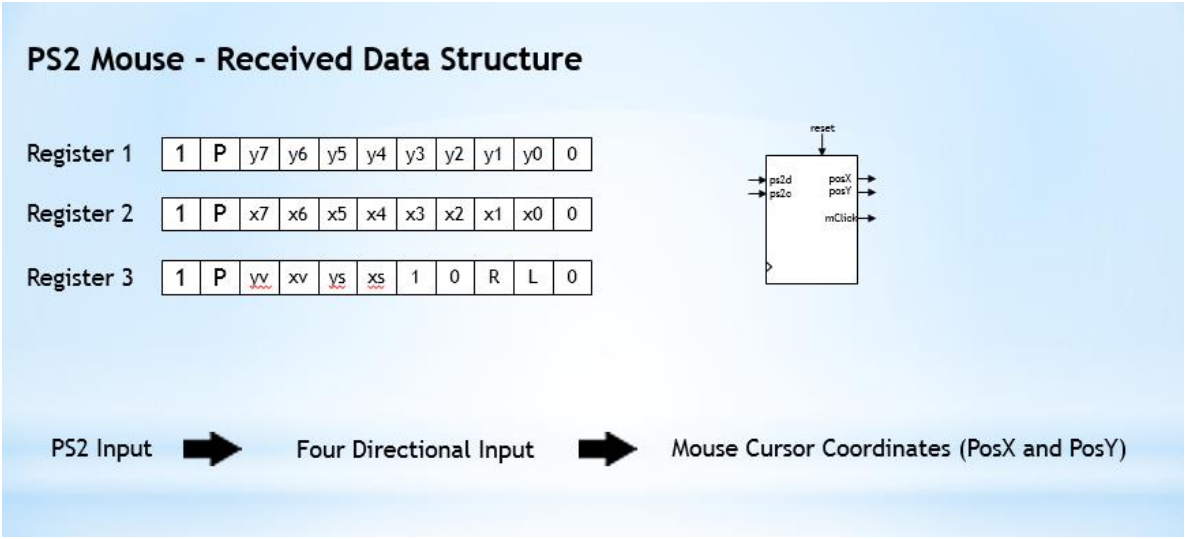

FIGURE 2

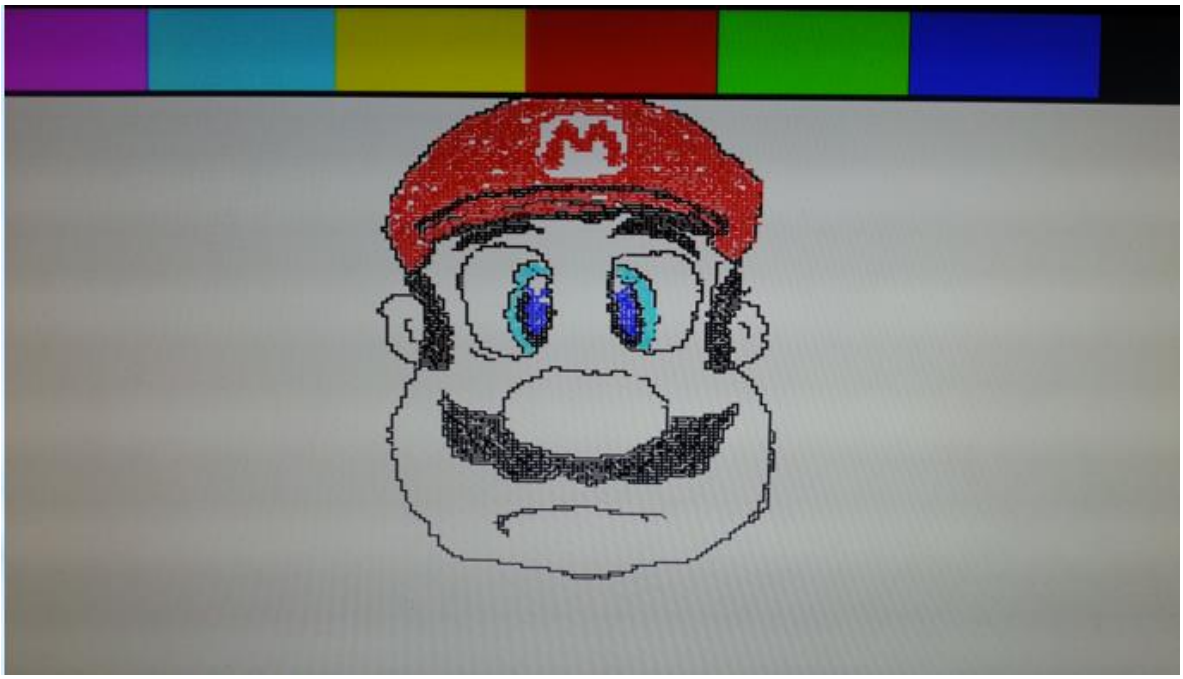

FIGURE 2