FPGA Air Brush

6.111 Project Proposal

Oscar Guevara
Junior Neeranartvong

1 Overview

This project implements an RGB color tracking and recognition system developed for human-computer interaction. Our design allows the user to make use of predetermined color schemes to control various parameters on a drawing application similar to MS Paint. This is accomplished by the use of a glove with various colored areas which the user can either show or hide by the movement of their hand. The visible color scheme is then captured by a VGA camera to be identified and processed by the FPGA. The resulting drawing is then displayed onto a VGA monitor. The user, at any point, has the option to load a previous drawing or store the current one onto a microSD card. This system will be modeled using Verilog HDL and implemented using a VGA camera, the Nexys 4 FPGA Board and a VGA monitor.

2 Design

2.1 Complexity/Challenges

I. The overall project is a combination of various modules working together. In order for it to be successful, each module must be implemented independently of each other but be able to integrate into the overall goal. The main modules include; color detection, gesture mapping, paint gui, save/load, and VGA output.

II. Specifically with gesture and color recognition, unwanted feedback might produce glitches and incorrect inputs being fed to the FPGA. To avoid this, we are required to implement tight constraints on what input we can consider stable. This increases our pre-condition requirements including an increase in resolution or distinction of possible inputs or a more robust set of filters. In turn, this requires either extra processing power and or more memory storage.

III. Scaling of our paint implementation in the future will be in great part determined by the way we define and store collected data. Common features such as undo and redo require sequential storage of data. In order to meet default industry standards such as these, our design must be modular, sequential and fast all at the same time.

2.2 Block Diagram

Below is the high-level block diagram. First, the RGB image data from the VGA camera is fed into the serial port of Nexys 4. Then, the Hand Recognition module converts the image data into hand/finger positions based on the visible color scheme, which will then be processed to determine commands controlling the Paint Logic. Third, the visualization of the current drawing will be handled by the graphics module which will transmit data via VGA port to the monitor. Fourth, the BRAM module is used to stored the current drawing data, this allows quick access to it by Paint Logic. Finally, the Storage module allows saving the drawing into the SD card and loading the drawing back to Paint Logic.

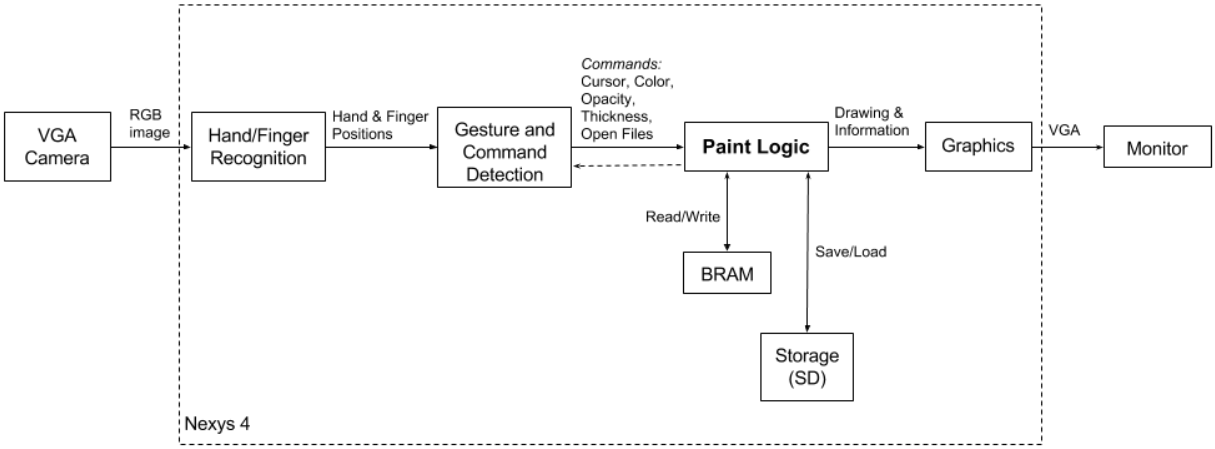

Figure 1. Project Overview Block Diagram

3 Implementations

3.1 Camera and Data Feeding

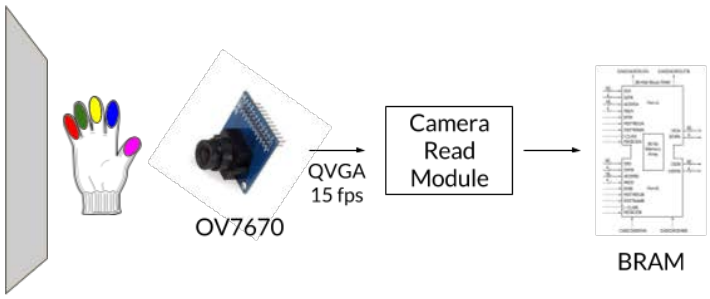

Figure 2. Camera and Data Feeding Diagram

The module will read an image frame from the VGA camera (OV7670) through SCCB interface. It will store the image frame data into BRAM. We will instruct OV7670 to transmit the RGB565 (16-bit) image with resolution 320 x 240 pixels at 15 frames per second to the Nexys4. Due to BRAM size constraints, we will only sample and store RGB332 (8-bit) image frame into the BRAM. Additional preprocessing to decrease data complexity might possibly be necessary.

3.2 Hand/Finger Recognition

We will extract hand and finger positions from the post-processed RGB image. Initially, we will focus on only two colors, later we will implement all other fingers. This module is responsible for detecting the position of each color section. The challenging part of this module is how well we could develop an algorithm with high accuracy, that is, which filters we should implement. To start off, we will use boundary detection of the colored area and find the mean/center point of each finger, and send that position data into the Gesture and Command Detection Module.

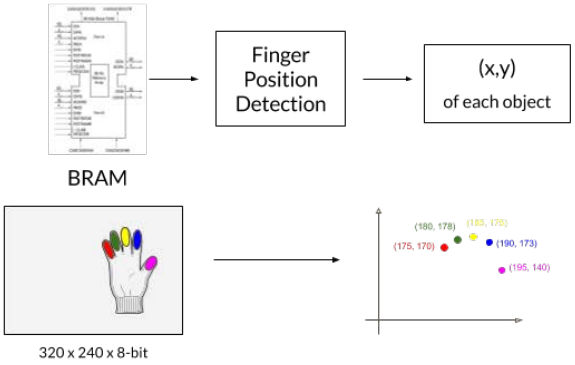

Figure 3. Finger Position Detection Module

More concisely, will use Gaussian filtering technique to reduce pixel noises. Then, we will use thresholding parameters to detect specific colors in the range of our interest. This will provide us an object to which we define its boundaries. We will finally then be able to find the centroid of each finger and send it to the gesture control module.

3.3 Gesture Control and Command Classification

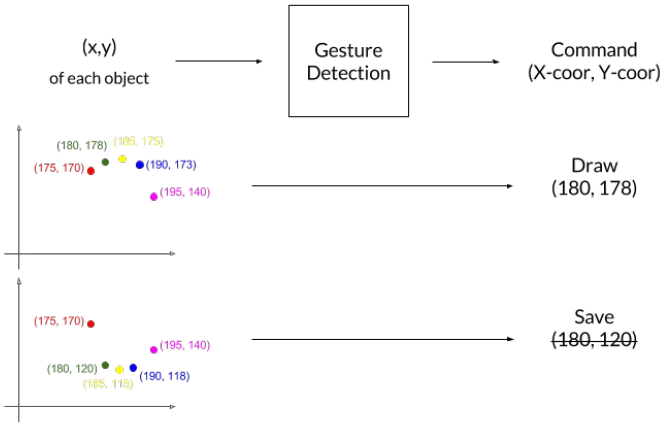

Figure 4. Examples of the commands used in Gesture Detection module

The Command Classification module will distinguish the combination of finger locations/color schemes to determine the command. Apart from determining the current position of the cursor, different finger configurations will be used to control the brush color, brush thickness, brush opacity, and loading/saving files.

3.4 Paint Logic

Paint Logic is the main part of the project. It will retrieve drawing data from Memory module and update the drawing via the information from gesture commands. More specifically, the Paint Logic will retrieve/send the information from Gesture Control module in order to draw out an image on screen. Regarding communication, we will be able to read and write using the paint module at the same time allowing us to edit and display the image in almost real time.

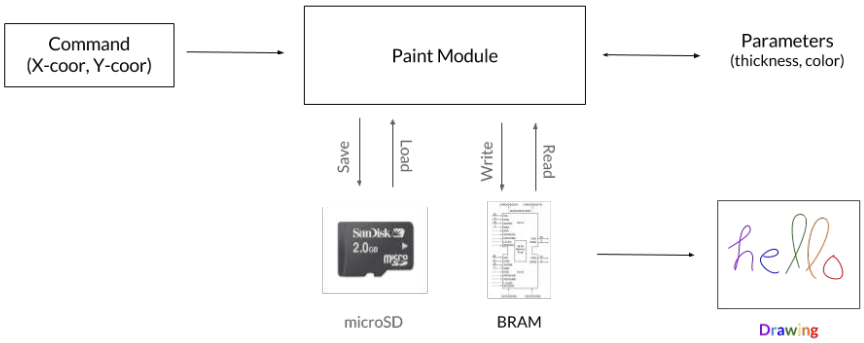

Figure 5. Paint Module Diagram

Furthermore, the Paint Logic will connect to the storage module so that it will be able to save/load data from Graphical Interface controlled by finger gesture commands or the diplayed save/load button.

3.5 BRAM Memory Module

The memory module is responsible for encoding data in a format that can be used to print to the monitor screen, and which can be dumped to a file for long-term storage. It is also responsible for storing the input image from the camera to be used by Hand/Finger Recognition module. Note that we could not store the whole image with resolution 640 x 480 pixels to BRAM due to its space limitation to 4860 Kbits. The limited space of BRAM will be one of the biggest challenges of this project if we avoid DDR2, which would require more sophisticated technique on data access.

One possibility to reduce space required to be stored in BRAM from the camera is based on the assumption that consecutive image frame from the camera does not vary significantly. The preprocessing data we are going to store will only consist of the areas around the past positions of colored fingers/hands. Suppose that the hand is around 1 m away from the camera which takes only small part of the image, this will reduce the number of pixels in consideration. However, the challenge of this technique will be that the number of pixels are varying, which might be hard to control in real implementation.

Another possibility is to sample a frame of 320 x 240 pixels from 640 x 480 pixels. This will decrease a significant memory space required to be stored. But the trade off is the less accuracy we could track the hands/fingers.

3.6 Storage Module (SD Card)

The storage module is in charge of taking the data currently displayed on the screen and saving it in sequential order to a file in SD memory. Further, it is also responsible for loading a file from SD memory into the current buffer to be displayed and edited on screen. One possibility of file reading is that the storage module will read data from SD memory and store them in BRAM Memory so that the painting data is easily accessible. The significant issue is the size of BRAM, implying that we could not store the current image with high resolution and color depth.

3.7 Graphical Interface

Figure 6. Design of the Graphical Interface

Initially, the entire monitor screen will act as a single sketch panel with no other visual parameters. Once that initial part is working, we will overlay on the right of the screen the parameters for the application which the user is able to change. Following that, if possible, we will rescale the screen area in which the user is allowed to write and develop a more friendly and aesthetically pleasant user interface.

4 Timeline

Task	Oct 31	Nov 7	Nov 14	Nov 21	Nov 28	Dec 5	Dec 12
Final Proposal Draft							
Block Diagram Meeting							
Project Design Presentation							
Figure out the solid idea/plan on how to import data from camera, image processing technique to detect fingers, how to distinguish finger positions for control, how to read/write memory, how to save/load file, and drawing data structure							
Checklist Meeting							
Finish camera connection							
Finish memory (BRAM/SD) connection							
Finish gesture control from finger positions							
Finish finger detection							
Finish GUI							
Finish reading/writing data to memory							
Optimization/Testing							
Polishing/Add a game							
Checkoff/VDO Recording							

5 Resources

All devices/tools are listed below.

- VGA Camera (OV7670)
- Gloves with different (main) colors at each finger
- Monitor with resolution at least 1024 x 768 (XVGA)
- Nexys 4 DDR Board

6 Conclusion

FPGA Air Brush project superficially does not look complex. However, the numerous modules of the project take time to implement and optimize. The project will give us an opportunity to learn how to apply image processing techniques to detect human gestures and find an effective way to store and read data from both BRAM and SD storage. Due to the limitation of Nexys 4 FPGA described in the implementation part, we will constantly need to compromise between different trade-offs, particularly between memory usage and the accuracy of the system. Once complete, our Air Brush control system will be able to be extended to various applications such as games. We hope that the project will be a good opportunity for us to learn how to make a good system and inspire others who might be interested in human-computer based applications.