ECE241 Digital Systems Project 2014

1. Introduction

Our main criteria for settling on a project idea were that it be cool, and different from the projects of other groups. This led us away from coming up with a game to animate, and towards integrating some kind of sensor into our project. Our first idea was to control a toy car with eye movement. Due to expenses and time constraints we settled on the idea of interpreting video data frame by frame using a colour detection method that would deduce the direction of movement of an object/image. We planned to animate a maze game and use the directions to steer an image of a car through the maze. A few days before the project was due, we realized the maze game was too ambitious, so we decided the program would simply note the directions for a theoretical object/image by outputting the information using LED’s. We created this project with what we believe is logically sound code, but could not make it function as desired. This caused us to create a new project that used code from the last project to store and handle pixel data. We figured out different ways of altering the pixel data to create different filters for our displayed video through the VGA output. Both the codes for this project and the project we could not figure out are in the appendices.

2. The Design

The basic job of this project is to take the video input from a camera, add a filter to it and then display it on the screen using a VGA cable. This basic job is divided into multiple small tasks. In the description of the design of this project we will first explain the tasks which will be divided into subtasks which the explained in the later paragraph.

The video input from the camera cannot be directly processed or used, it has to be converted to the right format. The camera is attached to the composite NTSC port which sends the video in analog form to the analog-to-digital converter, which converts it to digital format. This data is then sent to the Video-In decoder which converts the data to small packets of data which are called Avalon streaming packets. The Avalon streaming packets send data in a format such that each frame is divided into 720 columns and 244 rows which has 16 bits per pixel in 4:2:2 YCrCb color space. This data is then converted to data in a format where each frame is divided into 320 columns and 240 rows which has 16 bits per pixel in the RGB color space. This data is then sent to the pixel buffer direct-memory-access (DMA) core which writes the video stream into the SRAM/SSRAM. The data is read from the SRAM/SSRAM by the pixel buffer DMA core which sends the frames to the filter system where a filter is added to the video. This processed data is then sent to the VGA controller which sends timing and data signals to the VGA port to display the video on the screen.

We will now explain the data conversion processes mentioned in the previous paragraph. The data sent by the Video-In decoder has 720 columns and 244 rows for every frame with 16 bits per pixel in 4:2:2 YCrCb color space, but the pixel buffer DMA requires the data to have 720 columns and 244 rows for every frame with 16 bits per pixel RGB color space. This conversion is done in multiple steps. The Chroma Resampler converts the pixel from the 4:2:2 YCrCb to the 4:4:4 YCrCb formats, while maintaining the frame resolution of 720 x 244. This is then converted by the Color Space Converter into 24 bit RGB pixels with the frame resolution of 720 x 244. The RGB Resampler then converts this data to 16 bit RGB pixels with frame resolution of 720 x 244. The Clipper trims the stream from the 720 x 244 resolution to a 640 x 240 resolution by dropping the columns and rows around the exterior of the frame. Lastly, the Scaler reduces the stream to 320 x 240 by dropping every other pixel. This data is then saved in the SRAM/SSRAM. The Resampler core then converts this data from 16 bit RGB with frame resolution of 320 x 240 to 30 bit RGB with frame resolution of 320 x 240. Then finally the Scaler core replicates each pixel 4 times to get data in the format of 30 bit RGB pixels with frame resolution of 640 x 480 which is the only format accepted by the VGA controller.

3. Report on Success

We experienced some successes and failures while creating our project. We were forced to change the function of the project because of our failure to fix the bugs in our colour detection algorithm. It is possible that the algorithm doesn’t work because the amount of logic involved in analyzing pixel data and understanding what it means for the direction of movement of a theoretical object cannot be done within the 25 (?) MHz clock cycle allotted for it, using the Verilog code that we wrote. Nonetheless, we changed the function of our project to displaying and filtering video. We were successful at displaying video and altering pixel data to create different cool filters, such as a negative filter. However, we failed to create a way of using all our filters from one quartus II project. We tried to do this using switches and if statements, but we could not figure out why the switches did not work within an always block controlled by the positive edge of a 25 MHz clock. In summary, we did not create the cool project we envisioned, but something interesting and still difficult instead.

4. What would you do differently

If we were given another chance to do this project, we would change our method and timeline for creating the project, and our project choice. To create our project we spent a lot of time figuring out how the video input worked and thinking through the logic we used in our colour detection algorithm. It would have been smarter to write and test code as early as possible, and build upon that code, as this would have helped us realize what was feasible from within our constraints. It would also have been smarter to choose a project that used animation, math and coding/decoding of information. These are tools we have learned more about, and are better prepared to use. These 2 changes ensure that we are more comfortable with our project and it will be subject to less idea changes.

Appendix