6.111 PROJECT PROPOSAL

CHROMA KEY COMPOSITING WITH FPGA

Daniel Moon and Thipok Ben Rak-amnouykit *6.111 Fall 2014*

1 Overview

Chroma Key Compositing is a special effects technique commonplace in fields such as news casting, video production, and movies. Chroma Key Compositing identifies a color, referred to as the Chroma key, in a video feed, then replaces the color and layers an image or another video feed in lieu of the color. In most modern-day Chroma Key Compositing, the video is taken and post-processed with editing software, such as Adobe After Effects. In this project, we will replicate the technique on an FPGA, which allows us to process the video feed and stream the output to a VGA monitor in real time. Our project therefore offers a solution to removing an intermediary step between video recording and final production.

To build a Chroma Key Compositing system, we will use one NTSC camera to capture the video feed of a Chroma key background with obstructions in the foreground. The second NTSC camera will capture another video feed that will replace the Chroma key. We will store the video feeds in a DDR ram use an FPGA module to choose to control video overlaying. The module replace a pixel in the first video feed with a corresponding pixel from the second feed when the Chroma key of this pixel is within the selected Chroma key range. The final product will have two video feeds composited correctly and streamed to a VGA monitor in real time.

For the minimal design, we will start by overlaying the Chroma key of a video feed with a static image. Once we have accomplished this task, we would like to composite two video feeds. This step requires correct synchronization of the video feeds, such that the VGA monitor output a new frame with no delay from either camera, despite their asynchronous refreshing times. Finally, if we successfully composite two video feeds, we would like to implement a module for morphological processing. Morphological processing improves Chroma Key Compositing's robustness by correcting any error of having pixels in the foreground's obstructions with similar color as the Chroma Key.

1 Design Decisions

This section describes the design of our Chroma Key Compositing system. The project is partitioned into three sections: the video system module, the memory module, and the selector module. Figure 1 displays a detailed block diagram of the design proposal.

i. Video System

Our video system will consist primarily of two NTSC cameras. We chose NTSC cameras because of their relatively low resolution, which lowers the size of storage memory needed. NTSC camera also has existing data-processing modules that reduce the complexity of data-format transformation between the camera and the memory module. Specifically, the memory module will receive input in YCbCr color. Also note that the video feeds from both cameras will be on similar scale, allowing the selector module to perform 1-to-1 pixel replacements without further scaling.

ii. Memory

Our memory module will consist of a memory arbiter and an external DDR RAM. An arbiter takes in video feeds from both cameras, selects one video feed, and stores the data at a specific location in the memory bus, then switches to the other feed and stores its data at another specific location. An external DDR RAM is necessary because the built-in memory capacity of the FPGA is severely limited.

iii. Selector

Our selector module will initially be a multiplexer that compare pixels from the first video feed with the given Chroma key range and replace selected pixels with corresponding pixels from the second video feed. In the more complex level, morphological image processing will be implemented into this module to correct overlaid pixels in the foreground. After completing pixel replacements, the selector module also converts pixels from YCbCr format to RGB format and output this signal to the VGA monitor.

3 Implementation

Fig. 1 is our block diagram for our Chroma Key Compositing system.

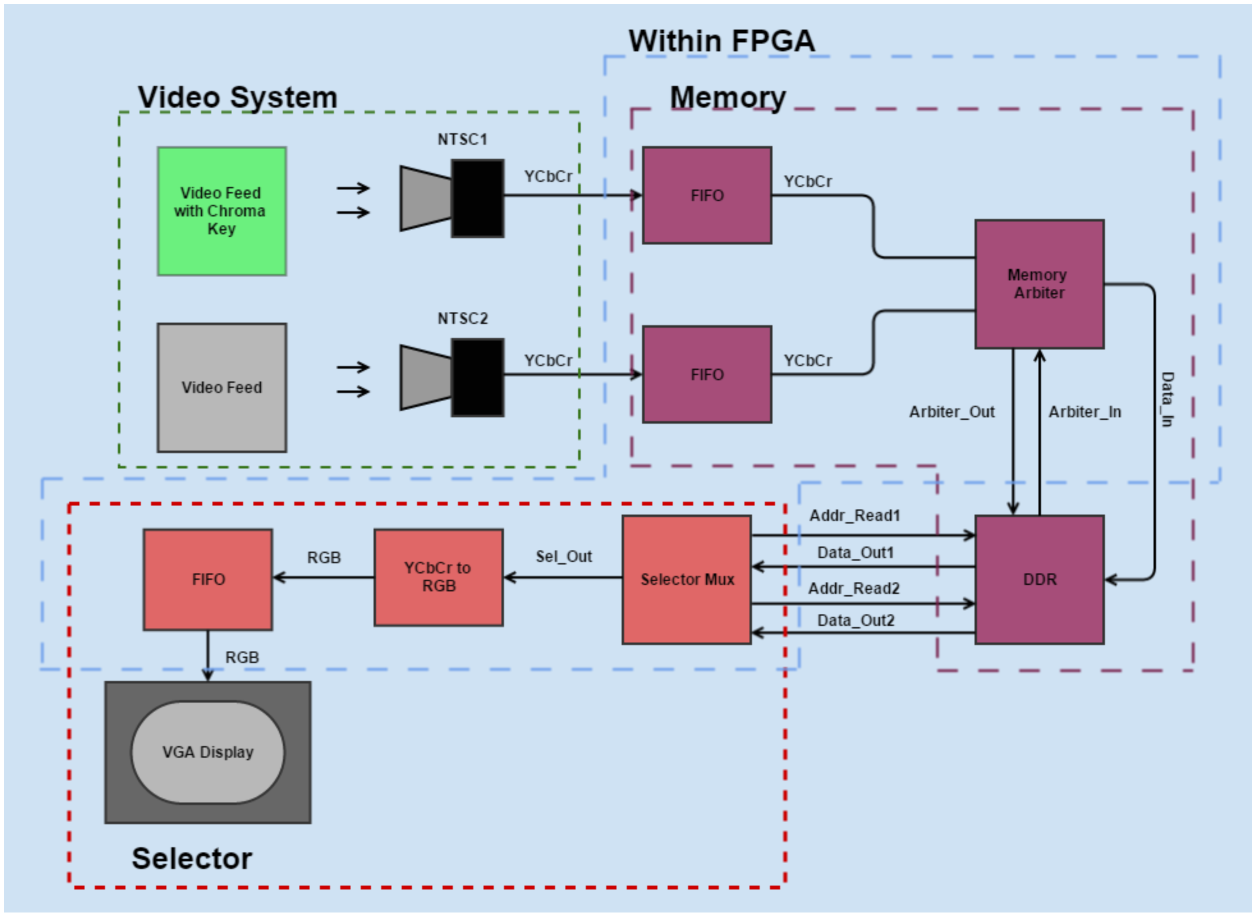

Figure 1: The Overall Block Diagram for the chroma key compositing. Above is a block diagram of the various modules that comprise the chroma key compositing. Chroma key compositing is composed of three blocks, the Video System (two NTSC cameras, the chroma feed and a video feed), the Memory (which consists of two FIFOs, a Memory Arbiter and a DDR) and the Selector (which consists of a Selector Mux, a YCbCr2RGB, a FIFO and a VGA display. The arrows represent the inputs and outputs of the modules. Additionally, a specified clock will connect to the modules within the FPGA while the NTSC cameras and VGA display will each have their own internal clock.

We decided to approach our design project in three modules. Our idea is that we would segment the system into sensors, memory, and the algorithm that does the proper chroma keying.

i. **Video System** – by Daniel Moon

Our video system first consists of two NTSC cameras that will be in progressive display which takes a view and outputs color in the form of YCbCr and location of the bits. The NTSC camera has two input settings for the video feed being wired into the memory module. It can either send information interlaced or progressive. We will be using progressive mode in our video as it sends bits in from the top left reads left to right, increments to the next row all the way to the bottom right. This makes memory storage and memory retrieval very easy, since we will only have to store start and end address value for the pixel information. These cameras also have an auto-gain feature that combats any issues we may have with the video having an extraneous noise due to underexposure.

Also, YCbCr is our color display design choice since it has a luminance component (Y), a color difference blue (Cb) and a color difference red (Cr). This means that if we have the correct Cb and Cr to output the chroma key we are compositing on, we can use Y to take care of a range of luminance of that color which may occur due to uneven lighting and shadows casted by obstructions in the foreground.

Secondly, we also need to address the chroma key that will be used in our experiment. We will be using a green screen as a chroma key and will use a YCbCr value close to the green screen's color for our selector module. Preferably, we will place the chroma key in a bright location in lab which would provide a bright and saturated image of the chroma key. We will zoom in on the chroma key such that the entire chroma key video feed is encompassed by the green screen. This will make our compositing a one to one scaling and will get rid of any issues with rescaling another video feed onto the chroma key.

ii. **Memory** – by Ben Rak-amnouykit

Memory becomes a large issue when we are in the final steps of storing the feeds of two cameras. Our FPGA will not be able to write, read and store megabytes of memory which will require us to use a DDR ram and a memory arbiter to handle any data coming in from the NTSC camera. Our DDR will be around 1GB in size and offer high transfer rates on the order of 1-2 GB/s. But we need to be aware of the bandwidth limitations of our DDR with regards to total information we receive from the cameras. From a top level perspective, we see that our bandwidth has two constraints:

$$BW \geq f_r * N_{f\{a\}} * p_f * b_p$$

and

$$BW \leq \max(DDR_{BW})$$

Where BW is bandwidth, f_r is frame rate, $N_{f\{a\}}$ is number of frames accessed, p_f is pixels per frame, b_p is bits per pixel, and $\max(DDR_{BW})$ is maximum bandwidth DDR. We will further revisit this issue in the Testing section of our proposal. So we will make design decisions to sacrifice certain information such as a reduction of frame rate (i.e. from 60 fps to 30fps) in order to fit our constraints.

The Memory arbiter serves an important role for correctly placing the output of the chroma key camera feed to the one set of address lines and the second camera feed to another disjoint set of address lines. We plan to store one frame of a video feed into memory after the other and then wait the correct frame rate to store the next two frames from the two video feeds.

We also cannot forget to include a FIFO buffer for each of the cameras to order the data being fed into the arbiter. There is also an issue with synchronizing the buffer such that it readies the data at the correct frame rate to be sent to the arbiter.

iii. **Selector** – Daniel Moon and Ben Rak-amnouykit

The selector module checks the chroma key video feed with the specified color and luminance. If the pixel being checked is in the color and luminence range, the selector will increment the address value correctly such that it accesses the correct pixel in the second video feed and outputs that pixel instead of the pixel from the chroma key video feed. This is a standard multiplexer and is not the end-all goal for our design project.

Mophological Image Processing is our final goal for the selector module. It involves manipulating an image by expressing it in terms of dilation, erosion and boundary. This allows us to have a more powerful Chroma Key Compositing algorithm which not only checks for pixel color but also guarantees boundary detection such that any material, such as clothing, which may have the chroma key color not be composited by the second video feed.

Finally, we will send the correct pixels through a YCbCr to RGB converter. This conversion is simply a linear transformation which can be done with matrix mathematics. RGB is the color scheme used for VGA display. We also need other information such as hsync and vsync which organizes the pixels in a coordinate plane to be displayed on the VGA. ADV7125 would be the converter module that would take care of this YCbCr issue. Lastly, we will also need another FIFO buffer to correctly feed the frame information in order for the VGA to express it accurately in progressive scan. Figure 3 articulates the display that would be shown on the VGA monitor.

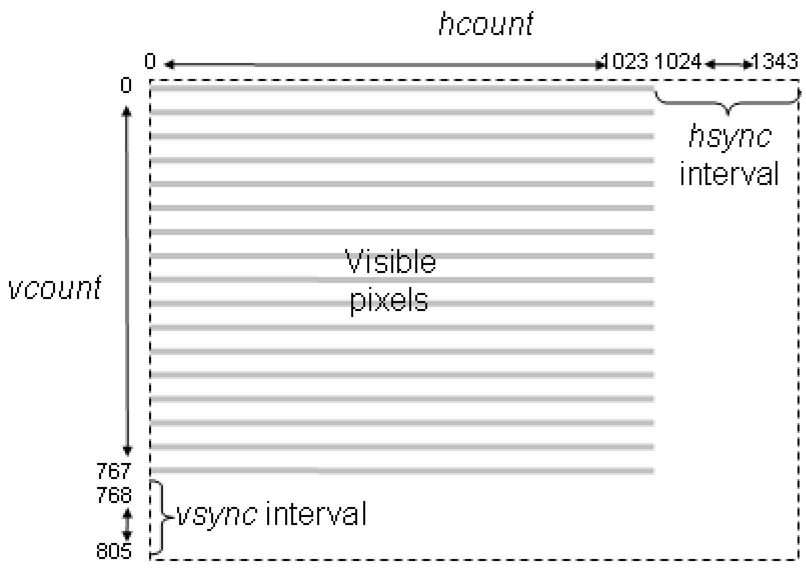

Fig. 3 Display of the VGA monitor. Hcount counts pixels in a scan line, vcount counts scan lines in a frame, and vsync/hsync interval stores the coordinates of the pixels.

4 Timeline

WEEK 1	10/26-10/31
Proposal Conference with Jose	
Draft Proposal Meeting with Michael	
Project Proposal Due	
WEEK 2	**11/2-11/7**
Block Diagram Conference	
Obtain Cameras, DDR, and Chroma Key	
Begin Video System Module with Static Image	
Begin Memory Module	
Project Design Presentation	
WEEK 3	**11/9-11/14**
Continue Video System Module with Static Image	
Complete Memory Module	
Begin Selector Module	
Submit Revised Project Proposal	
Project Checkoff Checklist Meeting	
WEEK 4	**11/16-11/21**
Complete Video System Module with Static Image	
Complete Selector Module	
Begin Video System Module with Video Feed	
Integrate Current Completed Modules	
WEEK 5	**11/23-11/28**
Project Status Update with Mentor	
Complete Video System Module with Video Feed (Video Feed Overlay)	
Integrate New Current Modules	
Debug Chroma Key Compositing System before leaving for Thanksgiving	
Ponder about Morphological Image Processing over Thanksgiving	
WEEK 6	**11/30-12/5**
Implement and Complete Morphological Processing	
Debug Final Project	
Test Final Project	
WEEK 7	**12/7-12/10**
Final Project Checkoff	
Project Demos and Videotaping	
Final Project Report Submission	

5 Testing

First of all, each module of the project will be tested and debugged separately. For the video system module, we will output YCbCr bits from the video feed and the static image to a logic analyzer. The memory module will be tested for correct DDR memory storage and retrieval of video data using a logic analyzer as well. To test the selector module, we inputs sample frames and a static image, and check whether the Chroma key pixels in the frames are replaced with correct pixels from the image. Then we test whether the processed frames are converting correctly to the RBG format by displaying the result on a monitor.

Once our initial modules are complete, we will integrate the system and test it with a real video feed. In the first step, the video feed will consist only of the background Chroma key pixel, and must be replaced completely by the given image. Afterward we move back the memory module and test if it can still store and retrieve two video feeds correctly using the logic analyzer. We will then test the Chroma Key Composition by seeing if overlays the video of the second feed on top of the chroma key in the first. If not, we will debug the system. It is here that we will see if there needs to be sacrifices in bandwidth to have our system working in real time. From a top level perspective, we mentioned from previously from our Implementation section that there two bandwidth constraints. We will have to make some design choices in order to satisfy these constraints in order to be able to output our Chroma Key Composite in real time. Another consideration at this point will be us having to check that we are doing all the computation fast enough so that we could output a video on the VGA at our designed frame rate. Finally, we will test if morphological processing works by wearing chroma key samples on ourselves and seeing if it does accurate border detection to see that these samples are not part of the chroma key screen. If this passes, we expect to do minor tweaking of the bandwidth to make the entire system work robustly in real time.

6 Conclusion

Our motivation for our final project is to replicate Chroma Key Compositing on our FPGA which eliminates an intermediary step between recording and video production to offer real time composited video stream. We decided to split the final project into a video system which captures and sends the chroma key and non-chroma key video feed, a memory system that stores and retrieves the feeds, and a selector that successfully composites the two feeds to a VGA output. Our final product should be a showcase where we have one camera facing in a direction in lab, another positioned in front of a chroma key background where students and staff could test and a VGA wired to the FPGA where others could see this special effects technique being executed.

7 Resources

APPENDIX A		RESOURCES
Item	Cost	Status
NTSC Camera (2)	Freely Provided	---
DDR RAM	Freely Provided	---
Chroma Key	$18.00	Purchased