FPGA iPad

6.111 Final Project

Alex Leffell and Sienna Ramos

Overview

We will recreate the iPad experience of manipulating images using finger gestures using an FPGA, NTSC camera and VGA display. The NTSC camera will capture the user's gestural input. This will include using a black glove with white fingertips to make it easier for the camera to see motion. The FPGA will convert the camera data into image processing commands as well as process the images accordingly. At a minimum, our design will allow a user to zoom in on an image by increasing the distance between their fingers and translate the image on the screen by moving their fingers together. The system will only manipulate a single image stored in flash memory. Once achieving the baseline goal, the system will be modified to be able to shrink, rotate and change the image being displayed.

The system contains two high level modules, gesture recognition and image processing. The gesture recognition module takes in images from the camera, detects gestures from the camera data and outputs commands to the image processor based on those gestures. The image processing module scales and rotates an image stored in flash based on the given commands. This module saves the modified image in an output buffer, from which the VGA driver pulls pixel data to display. Every module will operate independently on the same clock domain. The gesture recognition module will continuously output commands and the image processor will read those commands whenever it is done processing an image.

Goals

Baseline

- Detect features from NTSC camera
- Generate translation commands from features
- Read image from SD card and display in monochrome on VGA screen
- Translate image on screen based on command

Expected

- Detect zoom and rotate gestures from camera
- Generate commands for zoom and rotate
- Zoom image from 100% to 200%

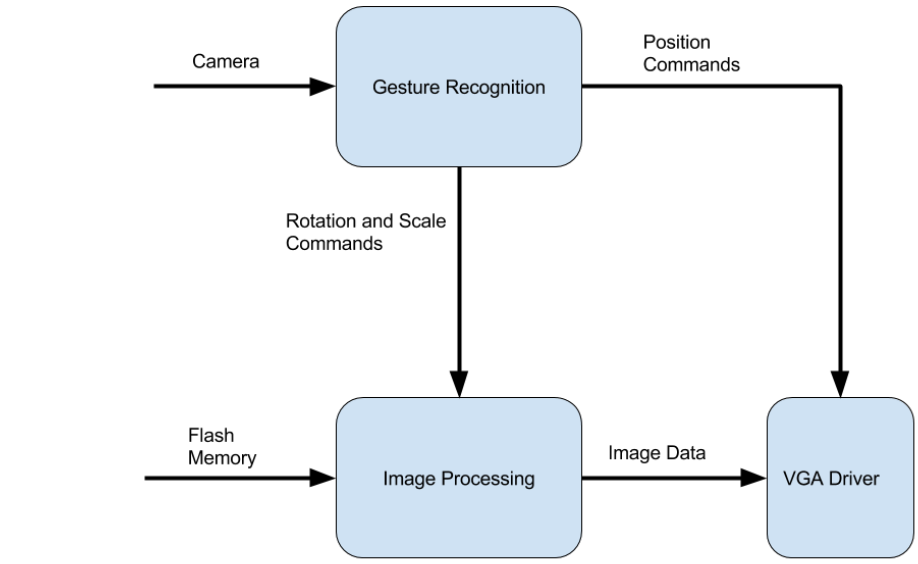
Stretch

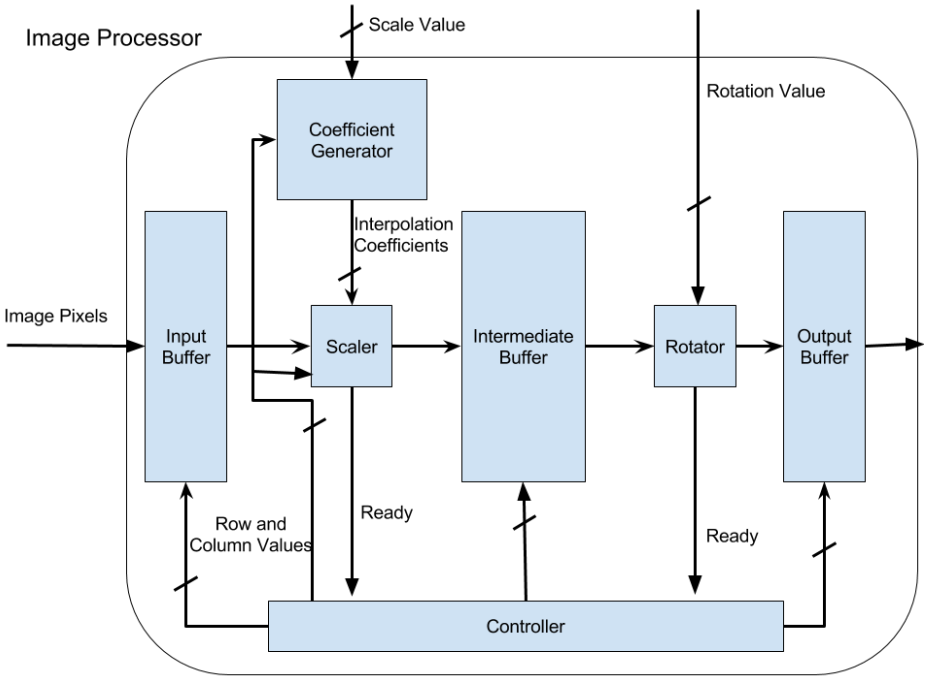
- Shrink image from 100% - 50%
- Rotate image
- Add gesture for changing image
- Be able to change image being manipulated

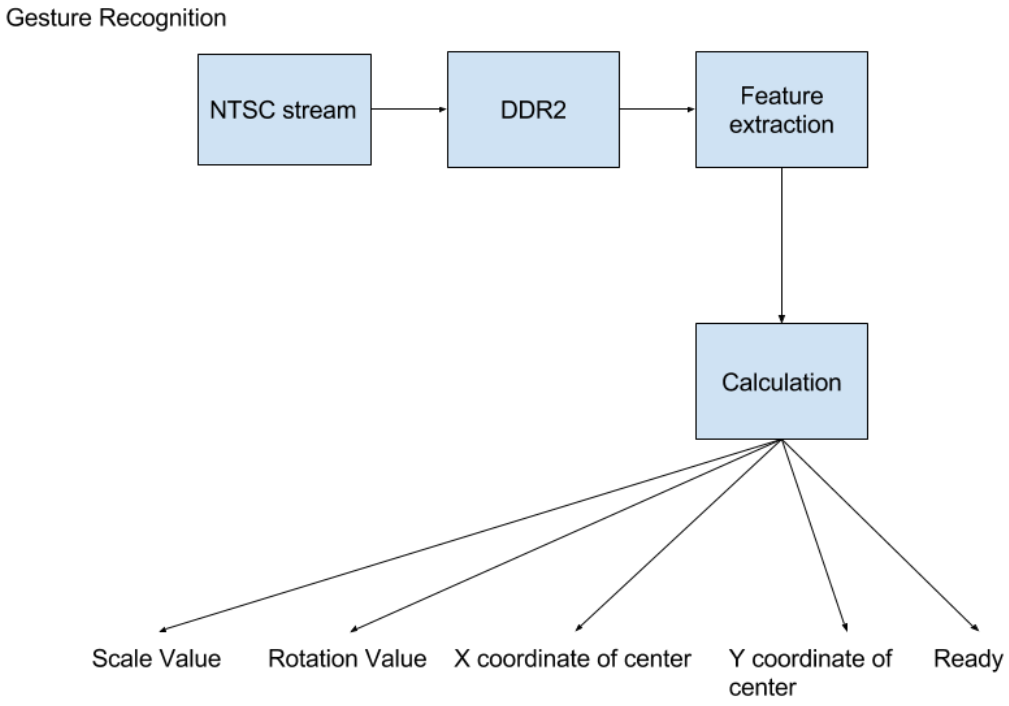
Design

Block Diagrams

Top Level Block Diagram







Modules

Support

VGA controller:

Inputs: image data, image center x and y values
Outputs: VGA signals to screen
This module generates syncing signals for VGA and outputs the image. It will mostly be developed from the existing code from the labs. The main difference is that our module will also take in x and y values and display the image around that coordinate. This module will be tested by instantiating a simple blob and moving it around on the screen using the Nexsys 4 buttons.

Image Processing

Coefficient Generator (Alex)

Input: 8 bit signed scale value, row and column values
Outputs: interpolation coefficients

This module will be a lookup table that takes in a value for the desired scaling of the outputted image, row and column values of the pixel being interpolated and outputs the interpolation coefficients required by the Scaler for the interpolation window around that pixel. Several instances of this module can be generated to calculate coefficients in parallel, increasing throughput.

This module will be tested in a testbench that provides it with different scale, row and column values and the outputted coefficients can be confirmed on the logic analyser.

Buffers (Alex)

Inputs: image data, control signals
Outputs: image data

The input, intermediate and output buffers are memories that provide easy access to the image data for use by succeeding modules. Depending on the number of Scaler and Rotator instances, the input and intermediate memories will be organized in order to efficiently provide those modules with the pixels they need. Note that that for the expected goals, the Scaler outputs data directly to the output buffer, bypassing the Rotator and intermediate buffer.

This module will be tested by instantiating it with known pixel values and certain locations then addressing it with those locations and confirming the outputted pixel values on the logic analyzer

Controller (Alex)

Inputs: ready signals
Outputs: row and column values, memory addresses

The module keeps track of the locations of the current pixels being processed. It tells the buffers what data to output and the Coefficient Generator, Scaler, and Rotator the row and column of the pixel to process. It takes in ready signals from the Scaler and Rotator.

Scaler (Alex)

Inputs: interpolation coefficients,
Outputs: scaled image data

This module scales an image by interpolating pixels between known pixels. From the input buffer, it takes in pixels for a 4x4 window around the point to be interpolated and calculates the value for the interpolated pixel based on coefficients provided by the coefficient generator. Several instances of this module can be generated to calculate coefficients in parallel, increasing throughput.

The Scaler will first be tested by supplying it with a 4x4 window of pixels and a scale value and ensuring the interpolated point has the correct value.

Rotator (Stretch)

Inputs: rotation value, row and column values
Outputs: rotated image data

This module performs three shear operations, to perform rotation without requiring an anti-aliasing filter.

Gesture Recognition (Sienna)

NTSC to DDR2

Inputs: Stream from NTSC
Outputs: Data to be sent to DDR2
This module saves video image data onto the memory. We can test this by displaying the camera image onto the VGA screen.

Filter data
Inputs: NTSC stream
Outputs: Coordinates of the very bright pixels.

This module extracts feature data from the NTSC screen. We can graduate from saving all the camera data to only saving the location and luminosity of the pixels which have a brightness beyond a certain threshold. These bright pixels are the fingertips of the LED glove, and are the only information we care about for the rest of the calculations. We can test this by displaying only the LED fingertips on the VGA screen.

Command Generator

Inputs: feature data
Outputs: 8 bit signed scale value (50%-200% scaling), x and y coordinates of picture center, 8 bit rotate value (0-360 degrees), 1 bit command ready signal

This module uses the saved feature data to make calculations determining the output to be sent to the image processing module. For the scaling value, we need to track if the lights are further or closer together than previously. The rotation value is the angle between a vector parallel to the x-axis and the vector created by the two fingers. The x and y coordinates of the output are the x and y coordinates of the point exactly between the two fingers. When all the calculations have been performed, the module asserts “ready” as 1. We can test each of these calculations by displaying the values on the 7-segment LED displays on the Nexys 4.

Timeline, Responsibilities and Resources

Our projected timeline is given in the following table:

Date finished	**Task**	**Person**
November 10	Material Collection (camera, glove, and any accessories)	Sienna
	Experiment with memory architectures, Design Scaler and Coefficient Generator architectures	Alex

November 13	NTSC to DDR2 module created	Sienna
	Coefficient Generator and Buffers complete, Begin implementation of Controller and Scaler	Alex
November 17	Filter data module created	Sienna
	Controller and Scaler complete	Alex
November 22	Command generator module created	Sienna
	Integration and testing of image processing module complete	Alex
December 1	Integration	Alex and Sienna
December 8	Polishing	Alex and Sienna
December 13	Submit Project	Alex and Sienna

The resources we will need to complete this project are detailed below:

Item	**Quantity**	**Cost**
NTSC Camera	1	Free (borrowing from lab)
LEDs	2	$0.30
Gloves	1	$3.50
MicroSD Card	1	$8