ECE532 – Digital Systems Design Winter

2011

Wild West

Final Report

Linghan Li [redacted]
Mengting Xie
Srinivasan Kidambi

Table of contents

1. Project Overview

1.1 Project Background

The inspiration for our project is from the classic video game Outlaw on Atari 2600. In the game, players shoot as cowboys against each other. Players fight on city streets crowded with wagons and score hits to win.

1.2 Project Goal

Our team decided to implement the classic shooting video game on a Xilinx XUP Virtex-II Pro Development System. In the game, players shoot at each other and score hits to win. Our vision is to make this game very interactive and intuitive by allowing players to use their hand gestures as weapons and hand motions to fire and move the guns on screen (Figure 1). Players hold their hands, in front of a camera, in specially coloured gloves and perform a set of predefined hand gestures intended for three types of guns (Figure 2). These hand gestures will be mapped, by our system, to one of three guns supported in our game, and the image will be projected onto a screen. Also, movements will be identified to allow players to move guns up and down, rotate the guns and fire the guns on the screen. The coloured gloves facilitate the detection of location of gun as well as special hand motions. Players should dodge each other's bullets by moving their guns up and down and try to score hits on the opponent. A Cannon digital camera was used to track positions of hands and feed video stream into Diligent Video Decoder Board (VDEC1).

Block diagrams of the control flow and all the components in our system can be found in the next section.

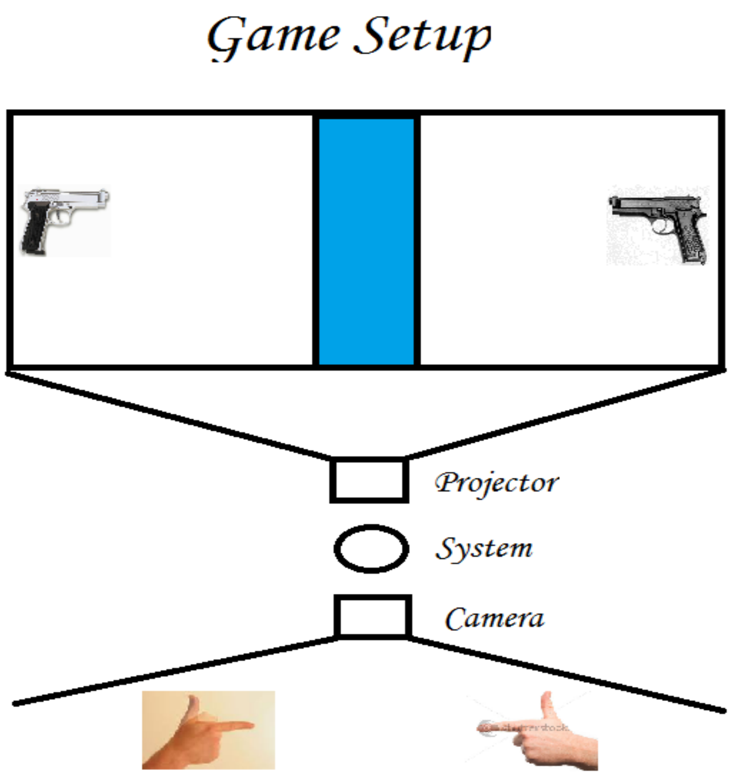


Figure 1 System Setup

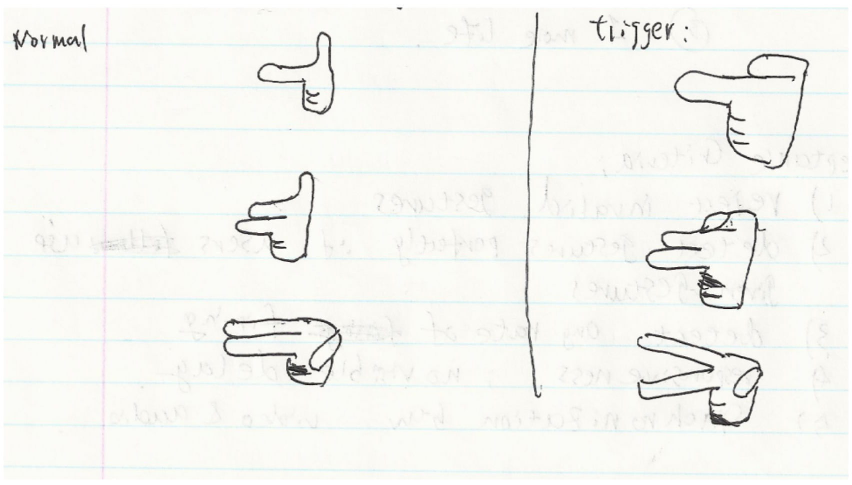
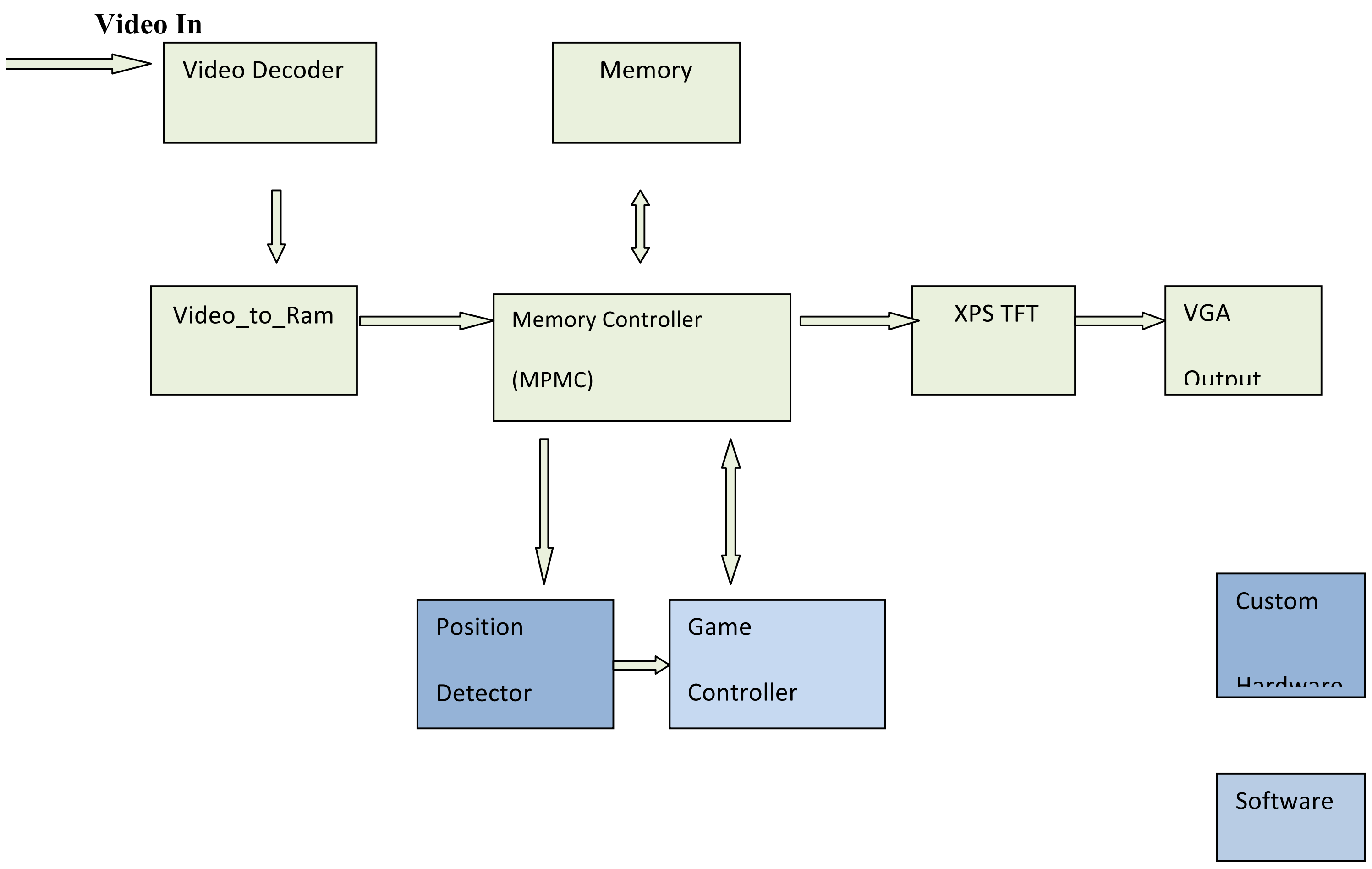


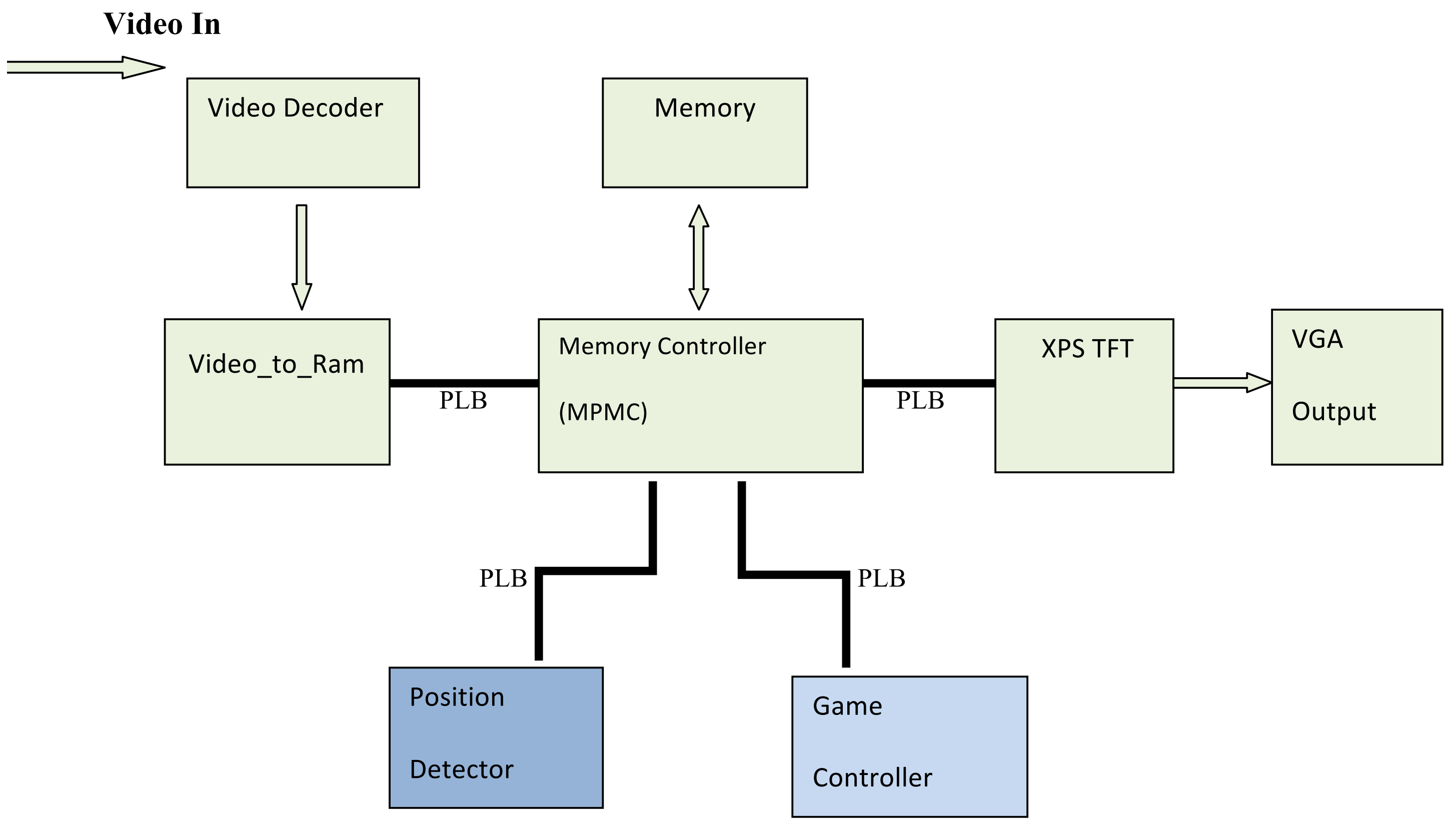
Figure 2 Hand Gestures to Detect

1.3 Block Diagram

1.3.1 Control Flow Diagram



1.3.2 System Diagram



1.4 Description of IPs

This section provides a brief description of the components as appearing in the system block diagram. Further details on the blocks in the System Diagram can be found in section 3.

Name	Function	Origin
Video Decoder	Digitizes video signal from camera and stores frames into memory	Xilinx IP
Video_to_RAM	Convert the YCbCr video stream from VDEC1 to RGB then write every frame into MPMC	Borrowed IP
Memory Controller	Provides multiple interfaces to memory from multiple sources. Needs 2 write ports for the Video Decoder and Game Controller, 2 read ports for the Position Locator and VGA output	Xilinx IP
Position Detector	Locate coordinates with specified colour based on digitized video frames in memory. Custom module implemented in hardware	Custom IP
Game Controller (Microblaze)	FSM for controlling overall game operations. Analyse the inputs from position detector and update the game states accordingly	Custom IP
VGA Output	Reads modified frames stored in memory by Game Controller module, outputs to screen. This is handled by the XPS TFT core	Xilinx IP
TFT	Convert video frames from MPMC into video signals for VGA to output	Xilinx IP
PLB(plb_v46)	Processor Local Bus used to interfaces to IP cores	Xilinx
Debug Module	Debug Module used to enable XMD	Xilinx
XGpio (xps_gpio)	Used to write and read signals	Xilinx
Video Daugher Card	Take video stream from camera	Analog Device
Video Camera	Capture video of player hand movement	
VGA monitor	Display the game for players	
IIC		Xlinx
dlmb	Data memory controller interfaced through Local Memory Bus	Xilinx
ilmb	Instruction memory controller interfaced through Local Memory Bus	Xilinx

2 Outcome

The final result of our project is successful prototype. The project takes video steam from digital camera and draw graphics to VGA output. The game starts with two guns appearing on the left and right sides of screen respectively and 5 points for each player. The project is able to detect a red point on the hand of one player and a blue point on the other player's hand. The detected points are analysed and used to update the locations of guns on the screen. Therefore the players are able to control the movement of the guns. The detection of triggering are not accomplished due to lack of time. The player moves the gun to avoid being shot. Failure to avoid collision between gun and bullets leads to losing points. The first player who loses all 5 points loses the game.

Considering the ease of calculation and implementation, graphic controller was done in software as part of game controller. However the reading from position detector is slower than the processing rate of position detector therefore the coordinates of detected area have to be processed to average value in position detector. A divider module is implemented to calculate the averaging in position detector.

3 Future Work

In the future, the Video_to_Ram would be modified to convert YCbCr video stream to HSV which is less sensitive to the brightness difference because in HSV colourspace saturation is ignored. Therefore the colour values are not affected heavily by the environment. The position detector would be able to identify more accurate coordinates instead of average coordinates. The accurate coordinates could be used to detect the triggering hand gesture by checking if the two detected points are close within certain threshold value. The detected triggering hand gesture enables players to have more control of the gun and choose when to shoot the bullets instead of shooting them automatically and continuously.

To make the game more interesting, one more point for each hand could be used to calculate the angle of the gun. The angular orientation of the hand changes the bullet shooting direction. Also obstacles could be added to the middle of two players to increase the difficulty of the game.

4 Block Descriptions

4.1 MicroBlaze Processor

MicroBlaze is a 32-bit soft-processor provided by Xilinx. It runs as the Game Controller of the entire shooting game.

4.1.1 Main functions

The main functions of Game Controller are listed below.

- Setup and configure the VDEC by reading configuration constants and sending them to VDEC
- Configure detection point colours which are read in position detection
- Initialize the UART and flush all the data in FIFO
- Initialize VGA output variables
- Draw the background of the game
- Draw and update gun positions
- Draw and update bullet locations
- Detect collision
- Draw and update scores

4.1.2 Graphic Drawer Algorithm

The main role of Game Controller is to analyse data from Position Detector and draw corresponding output on the screen. Initially a graphic controller was designed in hardware to draw images to VGA output. However, it was implemented in software due to consideration of math calculations. It is difficult to implement math calculations in hardware due to unavailability of IP's such as the "Cordic" Math IP, Divider IP. Our project also requires similar math calculations, such as trigonometry, to determine bullets' paths.

Video output is stored in a separate memory location from the video input. At the start of the game, the game controller draws a fixed background for the game and it's maintained throughout the game. For drawing shapes in game screen like the guns, bullets and alphabets, the Bresenham's Line-Drawing algorithm is used. During the game, game controller uses the inputs from position detector to update the positions of guns.

A collision detection algorithm is developed to detect collision between gun and bullet by looking for intersection between areas within gun and bullet. Then the game controller calculates and updates the scores for two players on screen. When the game ends, the game controller draws the end game screen.

4.2 Position Detector:

This custom hardware module processes video frames from memory and determines the coordinates of special identification points on the glove. These coordinates are used by Game Controller to determine the position. Figure 3 shows the interface of the Position Detector module.

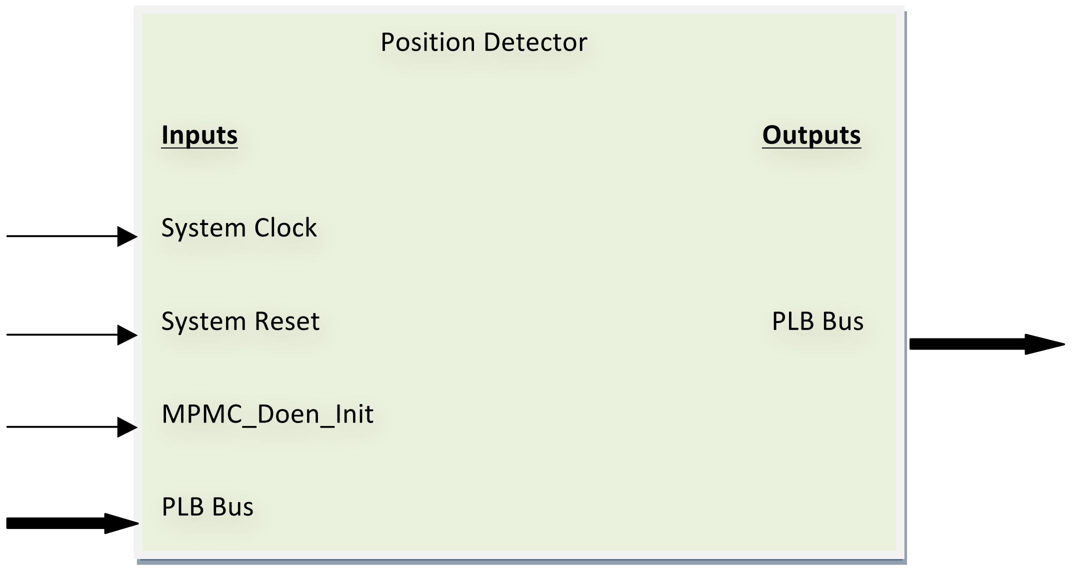


Figure 3

The Position Detector module uses the standard 100MHz system clock and system reset. It waits for the MPMC memory controller's initialization before the MPMC_Done_Init signal comes. Then the video frame data can be read from memory through a PLB bus to MPMC. To reduce the bandwidth of PLB bus to MPMC, a PLB bus is used to connect Position Detector to MicroBlaze and software accessible slave registers are used:

- Frame base address
- Run signal
- Minimum and Maximum values of point 1 colour
- Minimum and Maximum values of point 2 colour
- Registers for debug purpose

Figure 4 shows the flow of Position Detector.

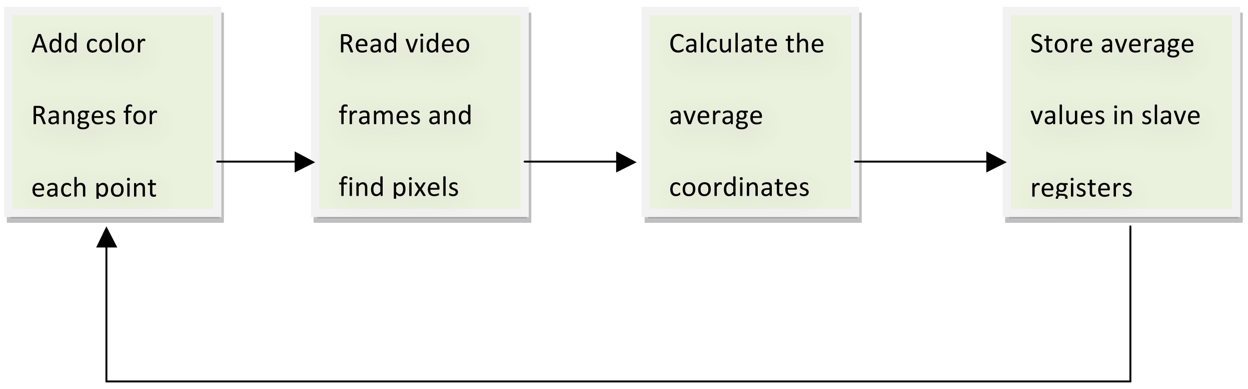

Add color Ranges for each point
Read video frames and find pixels
Calculate the average coordinates
Store average values in slave registers

Figure 4: Flow Diagram of Position Detector

The algorithm used for Position Detector is to find all the pixels with colours within the specified colour ranges in a frame and store the average coordinates of the pixels. Using the RGB colour scheme, every colour detected has Red, Blue, Green components respectively. The specified colour ranges and frame base address are repeatedly read from Microblaze through PLB bus. The pixel is read from memory one by one via MPMC. Position Detector keeps track of the pixel position within a frame which is needed for find out the coordinate of it. For each pixel its Red, Green and Blue values are compared to corresponding colour ranges to determine if the pixel has the specified colour. To increase the accuracy, the position detector accumulates the x or y coordinate values of every occurrence of pixel with colour that satisfies the specified colour ranges. At the end of each frame, the average of accumulated values is calculated and stored as the detected point coordinate. Finally the Game Controller can use the coordinates in the corresponding slave registers to update gun positions on the screen.

4.3 Video to RAM:

This is a module that takes input from video decoder and stores the frame into memory; this module is an existing IP because previous years' projects such as [2, 3] were able to incorporate it from Jeff Goeders' video_to_ram project. According to the group who worked on the "Human Pong" project, HSV colour space is more robust than either YUV or RGB colour space. So this module might need to be modified to support HSV colour space representation [2].

4.4 MPMC:

The MPMC module allows multiple devices to use the memory. Specifically, it allows video frames to be written into the memory, our Position Detector to read frames from the memory for processing, our Graphics Controller to write a game frame into memory, and the drawn game frame for our game to be sent through the VGA output.

4.5 XPS TFT:

The XPS TFT module is an IP core provided by the Xilinx core library. It reads video frames from MPMC and generates signals to VGA port[2, 3]. This module will be used to output frames generated by our game controller to either a projector or monitor connected to the VAG port.

4.6 VGA Output:

Video output to display the game screen consisting of the guns, bullets and the background.

4.7 Debug Module

The MicroBlaze Debug Module (MDM) was used to debug Microblaze processors through XMD and JTAG. It allows us to debug other memory mapped peripherals by reading from and writing to different memory addresses of the peripherals.

5 Appendixe

5.1 Zip File Directory

This section provides the directory structure.

```
/__XPS/ – Xilinx generated directory
/blkdiagram/ – Xilinx generated directory
/data/ – Xilinx generated directory
/drivers/ -drivers directory
        /position_detector_v1_00_a / - driver files for position detector module
/doc/ – Contains group report
         /GroupReport.doc – Group report
/etc/ – Xilinx generated directory
/lib/ – Xilinx generated directory
/pcores/ – Cores directory
         /led_debug_mux_v1_00_a/ – core used for debugging
         /position_detector_v1_00_a/ – Position detector core
         /video_to_ram_v1_00_a/ – core for writing video input to memory, modified from Jeffrey
                                  Goeders' core.
/report/ – Xilinx generated directory
/sw/ – Software directory
/_impactbatch.log – Xilinx generated file
/bitinit.log – Xilinx generated file
/clock_generator_0.log – Xilinx generated file
/platgetn.opt – Xilinx generated file
/system.log – Xilinx generated file
/system.log.bak – Xilinx generated file
/system.make – Xilinx generated file
/system.mhs – Xilinx generated file
/system.mss – Xilinx generated file
/system.xmp – Xilinx generated file
/system_incl.make – Xilinx generated file
```

5.2 References

video_to_ram core from Jeffrey Goeders, retrieved from course discussion board.

http://en.wikipedia.org/wiki/Bresenham's_line_algorithm