RUN, STEPHEN, RUN:
Shoot First, Ask Questions Later

Andrew Bui*, Cathy Chen†, Johnny Chin*, Andrew Sabatino*, Michael Scott†

*Department of Electrical Engineering
†Department of Computer Engineering
School of Engineering and Applied Science,
Columbia University in the City of New York

Abstract:

The design details of our target acquisition and missile launching project is presented in this paper. The main goal of this project is to create a platform that locates and shoots a non-lethal dart at a pre-determined target and tracking device. This project utilizes both the hardware and software capabilities of the Altera DE2 Board, while also using our knowledge of ballistics, image recognition and robotics. The hardware design is first presented along with a high level block diagram of the project. Each hardware component and peripheral is described in detail. The software design is then presented and its modules are described in details. References used in the design construction are presented at the end of the paper.

Introduction:

This project aims to create an automatic target location and missile launching robot on the FPGA. Through a camera interface, the device will be able to locate the position of a target, calculate its location in relation to the missile launcher, acquire a "lock" on the target and eliminate the threat. We have purchased a Dream Cheeky missile launcher that can rotate in both pitch and pan that we plan to use for this project, as well as a digital camcorder for sight.

System Overview:

Hardware:

Video Decoder:

The video decoder will be used to convert the raw data into a digital format that is used to obtain target position after image processing through software. Since the Altera DE2 board comes with ADV7181 TV Decoder chip, we will utilize it for converting the standard NTSC input from the camera to an output of the CCIR656 YCbCr 4:2:2 standard. Given the requirement of the standard, a clock of 27MHz will be used to clock the decoder. In addition to the decoder chip, we need to implement a module that does 4:2:2 to 4:4:4 format conversion as well as the YUV to RGB color space conversion. The 16 bit data generated is then passed on to the Avalon Bus for image processing to obtain target location.

YcrCb to RGB Converter:

Because the signals from the video decoder are encoded in YCrCb, they must be converted to RGB by using the following set of equations:

$$\begin{aligned} Y &= 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \\ Cr &= 0.5 \cdot R - 0.4148 \cdot G - 0.0813 \cdot B + 128 \\ Cb &= -0.1687 \cdot R - 0.3313 \cdot G + 0.114 \cdot B + 128 \end{aligned}$$

This function will be performed in a separate peripheral because of its possible computational intensity. Various optimizations can, and will, be explored at implementation, to speed up the floating-

point computations in hardware. One possible exploit is making use of a look-up table for the multiplication, which has been explored in a previous project.

X-Y coordinate generator:
One of the significant goals of the missile targeting is the creation and implementation of hardware that generates X and Y coordinates of the desired target, taking the live video feed as its input. This offers many advantages over software implementation, mainly in speed and modularity. As the project is in its early stages, the target does not yet have an exact definition. Items under consideration are:

1. Physical normalization of negative space in the video frame – ie. Using a solid-colored backdrop, etc.
2. Using a single chromatic indicator for the target – i.e. The target is red, etc.
3. Geometric target configurations that would add ease to X-Y generation – i.e. Two concentric circles, etc.

There is some precedent for vision hardware specializing in X-Y generation. The Wiimote camera is the most visible and accessible implementation. The camera itself is attached to built-in image processing unit which outputs x-y coordinates for up to 4 points of IR light. The camera itself is monochromatic and has a low resolution of 128x96. The built-in processor uses 8x sub pixel analysis to provide 1024x768 resolution. IR filtering is achieved using a physical filter, and raw image data is not available to the host.

This image processing paradigm will be emulated; however one of the goals is to avoid the need for targets in the IR spectrum. IR targeting generally requires some form of active target, namely IR LEDs, limiting the targeting capabilities by design. Because a physical filter offers considerable simplification, however, the use of one will be explored in whatever visible spectrum is ultimately designated for the target. Attempts will be made to find the established hardware design of the built-in image processing chip, however this design does not appear to be readily available.

Open source computer vision software is readily available and is a likely starting point. One notable example is FreeTrack, an open source head tracking software. This software can be used in conjunction with hardware such as TrackIR (an IR camera, similar to the Wiimote) to implement head-tracking controls in video games. The sophistication of head tracking exceeds the scope of this project, and the implication of head tracking escalates the threat of a USB missile launcher; however it may serve as a useful reference.

SRAM:
The *ISSI* IS61LV25616, on the Altera DE2 FPGA board, is a high-speed, 4,194,304-bit static RAM organized as 262,144 words by 16 bits. The pin description for the SRAM is listed below:

SRAM PIN DESCRIPTIONS

A0-A17	Address Inputs
I/O0-I/O15	Data Inputs/Outputs
CE	Chip Enable Input
OE	Output Enable Input
WE	Write Enable Input
LB	Lower-byte Control (I/O0-I/O7)
UB	Upper-byte Control (I/O8-I/O15)
NC	No Connection
Vcc	Power
GND	Ground

The SRAM is used to store pixel data from the YcrCb to RGB Converter. The SRAM acts a frame buffer between the Video Decoder (via the YcrCb to RGB Converter) and the VGA

Controller to facilitate the output of the image from the camera to the VGA monitor.

There are a few challenges, regarding the resolution of the frame buffer, in using the SRAM. A 640X480 buffer at 16 bits / pixel would take up 614 kB which is more than the SRAM size of 512 kB. To solve this challenge the resolution that is stored in the buffer can be scaled down to 320X240 which will result in a 1 to 4 mapping from buffer to VGA pixel. This resolution will suit our application since the frames shown on the VGA will only be used to monitor (show the user) the output from the camera.

The second challenge presented by the SRAM is the fact that it cannot be written to and read from concurrently. The timing diagrams below displays this fact.

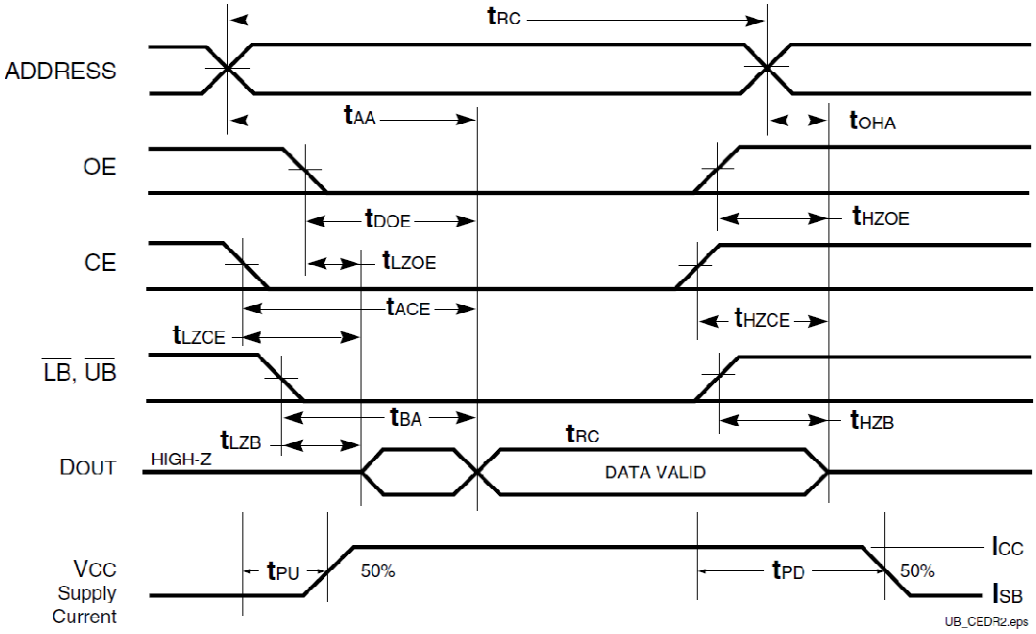

Figure 1 – Timing Diagram for SRAM read

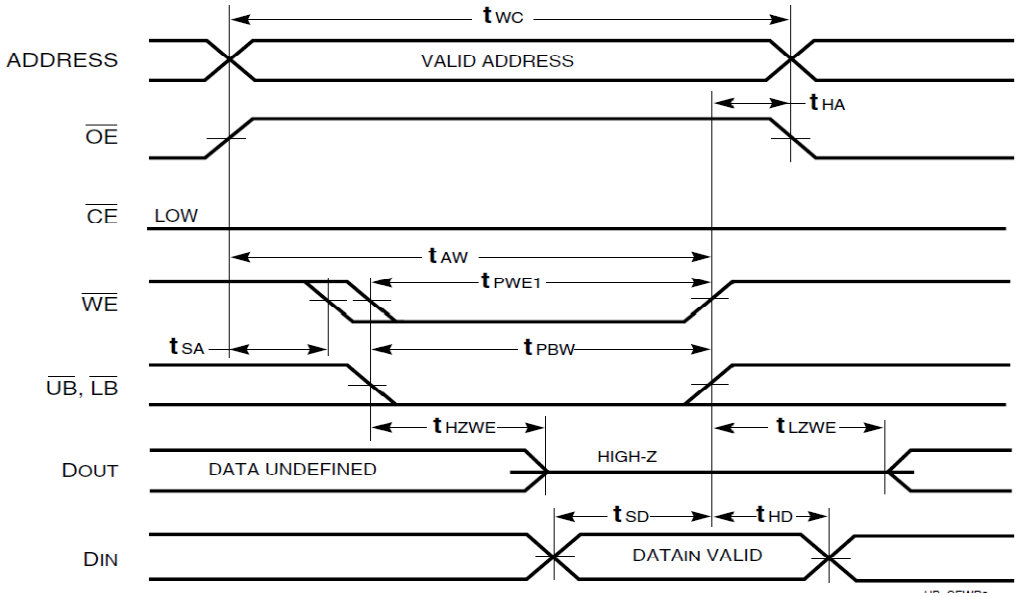

Figure 2 – Timing Diagram for SRAM write which is WE controlled

The write to the SRAM and the VGA controller drawing pixels cannot happen during the same period of time to ensure that the correct image is displayed. This challenge can be resolved by having the VGA controller control the read/write operations of the SRAM.

VGA controller:
The VGA display will be used to debug the system, allowing the user to see what the system sees during target acquisition and firing. The VGA controller will be implemented as a peripheral hung off the Avalon bus, clocked off the halved 50 MHz global clock, as this corresponds to the necessary 25 MHz pixel clock necessary for the VGA protocol. Redrawing will be triggered off the VGA Sync signal to limit the redraw rate to 60Hz, the native rate of the lab monitors. The display will resolve at 320x240 to match the resolution of the CCD sensor. The framebuffer will be located in the SRAM, which is a copy of the SDRAM video data after the post-processing for intensity and RGB filtering. The truncation of every other row and column in the video space will ensure that the data will fit into the SRAM, which is limited at 512 kB. A system flag will let the system know when the framebuffer has been completely written to the display, to ensure stable operation.

USB controller for Rocket Launcher:
The USB rocket launcher from Dream Cheeky ™ comes with software that controls the movement of the launcher. The commands for the launcher are as follows:

0 = stop
1 = up
2 = down
4 = left
8 = right
16 = fire
32 = reset

Using this information and the Host USB port on the Altera DE2, a USB controller will be built that outputs the 3 packets needed, with a blank heading and trailing packet and a middle packet with information.

USB protocol on the DE2 board is provided by the Philips ISP1362 single chip USB controller. The chip provides data transfer for both host and device interfaces and is compliant with USB Specification Rv. 2.0. A device driver to control the rocket launcher will need to be written on the NIOSII.

Software:

NIOS PROCESSOR:

Video processing of X-Y coordinates/ Calibration:
The X-Y coordinates will be extracted from the video signal after hardware-implemented filtering of the RGB and intensity characteristics. Under ideal conditions, this should result in a set of two points of high intensity green and blue. The software should be able to calculate the intersection of the lines drawn between both the green and blue points, thus finding a 2-dimension projection of the target in the pixel space. An initial calibration of the distances that will be used for the green and blue

markers will anchor a physical distance in the system, allowing the use of similarity transforms to perceive an apparent sense of depth. This will allow the system to find the target's location in a 3 dimensional space. A lookup table will implement the necessary inclination angle for a given distance using basic ballistics equations. This should ensure a "hit" in a roughly spherical area around the point of intersection of the green and blue markers, when taking into account variance of the system actuator.

Aiming Launcher and firing:

Once we obtain the target location, we can control the rocket launcher using USB, following the protocol for controlling the rocket launcher. We will need to conduct many trials to observe the change in where the darts land with respect to how many times each command is sent. These statistics will be used in C program in which we process the coordinates and translate them into a series of mechanical commands for aiming and firing.

REFERENCES:

http://wiibrew.org/wiki/Wiimote#IR_Camera
http://www.instructables.com/id/Wii-Remote-IR-Camera-Hack/
http://wiki.wiimoteproject.com/IR_Sensor
http://www.sparkfun.com/commerce/pre...=Wii-Internals
http://www.free-track.net/english/
http://en.wikipedia.org/wiki/TrackIR
http://dgwilson.wordpress.com/
http://www.amctrl.com/rocketlauncher.html
http://www.planetanalog.com/showArticle.jhtml?articleID=219000119
http://www.pcs-ip.eu/index.php/main/edu/3
http://www.cs.columbia.edu/~sedwards/classes/2010/4840/lab2.pdf
http://www.roborealm.com/help/DC_Missile.php
http://dreamcheeky.com/index.php?pagename=product#
http://www.cs.columbia.edu/~sedwards/classes/2010/4840/ISSI-IS61LV25616-SRAM.pdf
http://www.cs.columbia.edu/~sedwards/classes/2009/4840/designs/RVD.pdf