Z-Goggles

Cody Baxley, Geoffrey Jean, William Petty, Christopher J. Silver

School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, Florida, 32816-2450

***Abstract* — Z-Goggles is a vision modification system meant to alter the user's view of the world. It accomplishes this by performing several image processing functions on a digital video signal and presenting the results to the user via a helmet mounted display unit. The entire system is worn by the user and powered by a battery-driven power supply. In addition to its main image processing functions, the system has the capability of being expanded to see in the Infrared and Ultraviolet spectrums through the use of filtering lenses.**

***Index Terms* — Field programmable gate arrays, digital-analog conversion, parallel programming, digital cameras, video signal processing.**

I. Introduction

Z-Goggles doesn't really stand for anything, it is just a name that makes you wonder what it could possibly be. The Z-Goggles system is a vision enhancement system, though not in the normal sense. It allows the user to modify what they do see in a number of ways. Mainly, this is done for the enjoyment of the user. Who doesn't want to see the world upside down? What does the world look like when the colors are inverted? What about a blurry image?

Essentially, this is a video/image processing capability, allowing the user to modify the display with 3 functions. Z-Goggles can flip the display, create a photo-negative effect, and simulate blurry vision. Another twist is that the Z-Goggles are portable and capable of being worn outside. The Z-Goggles are designed to go where the user goes. It is battery-powered, and the display is integrated with a helmet for ease of use. The required power supply is located in a backpack that is worn with the helmet, so the complete system is wearable in two sections. Overall, the system isn't heavy, so the user should be capable of wearing it for extended periods. The way the system is currently designed is as a single camera to FPGA unit. Conceivably, the unit could be paired with other like units with differently filtered lenses, to allow viewing in other spectrums. This allows for future growth of the system for other applications.

Regarding the technical implementation, the entire system consists of relatively few parts. A digital camera connects to an FPGA for processing and display control. A VGA output card interfaces with the FPGA processor to output to a helmet-mounted display unit. The user can select functions via a controller, which is built upon a microcontroller which interfaces with the processing FPGA. All power is provided by the backpack power supply. A concept drawing of the Z-Goggles is shown in Fig. 1.

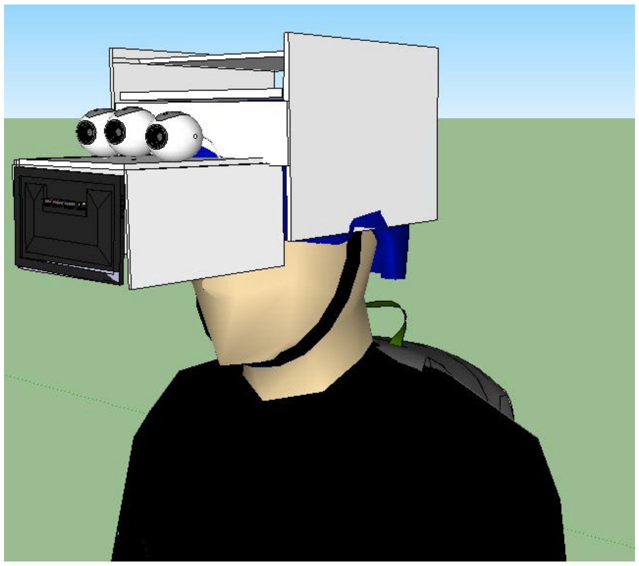


Fig. 1. Concept drawing of the Z-Goggles system. The Display unit is in front of the user's face, and multiple cameras are shown to indicate the possibility of using multiple cameras for multi-spectral viewing.

II. Digital Camera

The digital camera used in the Z-Goggles design was chosen for its digital output and ease of interface with the FPGA Video Processor. Our group chose the C3188A Color Camera module based on the OV7620 image sensor from OmniVision. This camera has a pinout header that mates well with a modified cable to the FPGA, and also provides digital 8 or 16-bit RGB progressive video data output. This is the type of video we wished to use for processing and VGA output, so it was the ideal choice. Another reason this camera was chosen was its proven capability in microsatellite projects. This allowed some confidence in its ability to perform when compared to other possible camera options. The Camera/FPGA interface setup is shown in Fig. 2.

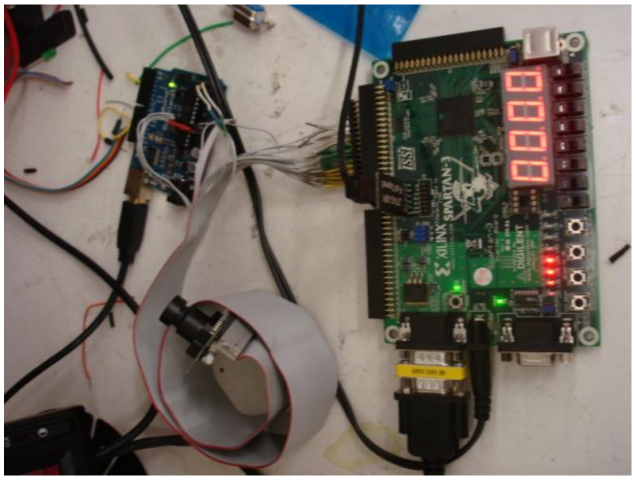


Fig. 2. Camera connected to the FPGA Video Processor.

Communication with the camera consists of I2C control of the state of its data output and other commands. Due to difficulties with the camera's I2C function, our group decided to deal with the camera data as it is given to the FPGA. Essentially, the Video Processor takes in YUV digital data from the C3188A, and converts it into RGB. The rest of the information, including clock, horizontal and vertical sync signals, and data, are simply output from the camera into the FPGA.

III. FPGA Video Processor

The purpose of the FPGA is to act as a Video Processor by taking the data output from the camera and reformatting it into a properly synced series of images. These images can then be modified with the Z-Goggles image modification functions, and run into the VGA card. This may sound simple, but the main processes of image framing and memory-mapping the data output from the C3188A digital camera is a large portion of the design requirement for the entire Z-Goggles system. Also, each image modification function requires a module be designed to enact it.

A Digilent Spartan-3 FPGA board was chosen as the Video Processor. The main reason this board was chosen was its low cost and its available SRAM. This allows the memory-mapped video setup necessary for our processing engine. Doing without this capability turned out to be unrealistic, especially considering the image modifications we wish to accomplish that would flip data relative to a constant position. This is impossible without memory to hold a single frame.

To begin with, we must frame the image data coming from the C3188A digital camera. Once properly framed, the data is stored in memory. If necessary to output properly using the VGA card, we then convert the YUV data into RGB format. Next, the data is modified according to the user's preference made on the remote. Finally, the data is output to the VGA card using the VGA output controller.

A. Image Framing

The image from the camera comes in along two clock pulses in YU and YV 16-bit words on a 27 MHz bus. The onboard system uses a 25 MHz clock for video output synchronization. This means that the system cannot operate synchronously and needs some form of buffering to support error-free video throughput.

Because of the mismatched clocks, ideally some form of internal dual-ported RAM would be the most simple and reliable method of buffering. The board however contains 2 SRAM chips both holding 256k 16-bit words, with a shared address bus. Most boards with dual-ported RAM were too expensive for us to consider. The memory controller to manage these constrained resources is discussed in the next section.

Image input data is only latched in within the camera's active region, denoted by a low signal on VSYNC and a high signal on HSYNC, with clock pulses marking valid pixel input within each line. Internal state variables track these 3 signals to generate the input pixel coordinates. Each VSYNC high signal resets all the values, each HSYNC transition from high to low marks the next input line, and each clock pulse during HSYNC high denotes a half pixel of input. With these internal states, tracking can be accomplished to frame video input into memory in an asynchronous manner. Fig. 3 shows the timing signals and system states/behavior.

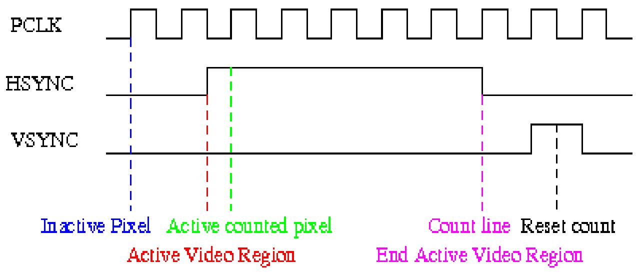


Fig. 3. Timing signals and system behavior during the image framing process.

B. Memory-mapping Image Data

To map the input and output data into RAM with the lowest possible latency, no conversion is done and the x and y coordinates are used directly for the address space. The x coordinate uses the lowest 9 bits to fit all the pixels of a given line into memory. This is enough bits to cover the entire 480 possible locations, since 2^9 equals 512. The y coordinate uses the lowest 9 bits as the upper 9 bits in the memory address. The MSB of the y coordinate controls the chip enable (CE) lines of the two memory chips to alternate them for the higher and lower parts of the frame. This is the solution to the constrained memory resources we mentioned earlier.

Because the input and output clocks are not synchronized in any way, including phase adjustment, characteristic diagonal banding will occur along an error region. In this region, contention for the address and memory busses will produce incorrect data on the output side of the system. To reduce this contention, a memory controller must be used which accepts input and output requests and buffers them accordingly.

To reduce the logic involved in the design phase, we favored reads over writes. Although writes occur more frequently, the time taken to execute the write is less important than time taken to execute a read. A delayed read operation will result in some errant pixel data, and given that the output DAC latches along the positive edge of the pixel clock, either the video clock must be delayed to compensate, or the delayed read will be as bad as a failed read. This type of failure will happen at regular intervals forming observable patterns in the output video even if the distortions themselves are minimal. A delayed write operation can only fail in the event that the delay takes longer than the frame takes to write. In most cases it is likely that the write operation will happen successfully before the next write operation. As the phase of the clocks grow apart, it is possible that a write may not be successful because of a neighboring read request very nearly follows the end of a failed write request. In our current design, it is unlikely that more than 1 error will accumulate, and if consecutive errors occur they will be more dispersed and thus less noticeable within the video window.

The current design is composed of the memory interface, internal buffers, and controller logic. The logic accepts read and write requests and on their positive edge. If the current request is a write request and there is a read or write in progress it will be buffered internally until those requests have been processed. The memory device I/O latency is 10ns, and the clocks driving the read and write requests are spaced by about 40ns, so the chance of having an error persist for more than 1 read/write cycle is low. When there is no active request, the buffered write request will be pushed as a normal write request in the system. If there is a read request during an existing write request, the current write will be aborted and buffered. If there is a write request during a read request it will be buffered until the read request finishes. If a write request occurs during a write, it will be buffered. Reads can only take place every 40ns; and, they take precedence over everything. However, if a read request occurs during a read, something is severely broken. This case is not accounted for because the rest of the system will very obviously be malfunctioning. This memory controller is sufficient to remove contention for the address and data lines and remove the entire class of errors related to those shared busses.

C. YUV to RGB Conversion

There are many different color standards that may be considered and it's not precisely clear which color map the camera uses. The documentation doesn't lead to a direct answer, other than the fact that it is known to be a form of YUV data. The standard we were using specified three equations for converting YUV format data to RGB format data. They are

$$\mathbf{R = Y + 1.398*V} \tag{1}$$

and

$$\mathbf{B = Y + 2.032*U} \tag{2}$$

and

$$\mathbf{G = Y - 0.395*U - 0.561*V.} \tag{3}$$

Without floating point availability or the latency to even do large fixed-point multiplication, near-approximations are used. Our actual system implements these three equations instead:

$$\mathbf{R = Y + 1.5*V} \tag{4}$$

and

$$\mathbf{B = Y + 2*U} \tag{5}$$

and

$$\mathbf{G = Y - 0.5*U - 0.5*V.} \tag{6}$$

This change is made for the sake of time and the aforementioned accuracy issues. While this change in order of accuracy is dramatic, the color bias remaining the same across the image is more important than the color bias being very true to life. This removes implementation complexity and reduces latency. Also, quality of color is not one of the specifications we are required to meet at this time.

D. Image Modification Functions

The simpler image modification functions directly operate on a single input pixel supplied from RAM. These functions include color inversion and palette adjustment. In the case of color inversion, the current red, green, and blue components of each pixel are subtracted from the maximum color value of that component (64 for green, 32 for blue and red, given a 5-6-5 16 bit RGB format). This results in an image with inverted color components, with white becoming black, etc. An example of this type of operation is shown in Fig. 4. A palette shift is not a user modification, but it is used by the designers to shift colors to correct any perceived color glitches. Essentially, it is simply a quick color change to make the output video look better.

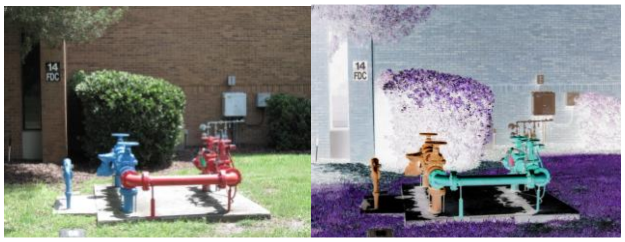


Fig. 4. Color Inversion example.

The second capability of the Z-Goggles is to flip the video upside down, or do a Vertical Flip. This capability is accomplished by saving the pixels in memory in a reverse manner. Instead of the 'top left' pixel being saved in the 'top left' spot in memory, it is saved in the 'bottom right'. This is only a little more complicated than the single pixel operations. An example of this type of operation is shown in Fig. 5.

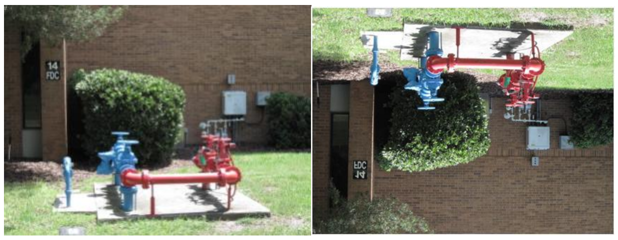


Fig. 5. Vertical Flip example.

More advanced functions like Blur and Edge Detection have their own input cache. Each input frame, 3 pixels in the lines of the next pixel are read. Each of these pixels comes from a different line of input and is stored in a 3x3 matrix of pixels for that particular function to use. On read, each of the lines of the matrix is shifted over into the next cell, reducing the memory input latency requirement to meet the throughput.

Blur is accomplished using an adder with parallel shifters for each color channel to create an even-weighted blur across each pixel. The output is otherwise exactly the same. An example of Blur is shown in Fig. 6.

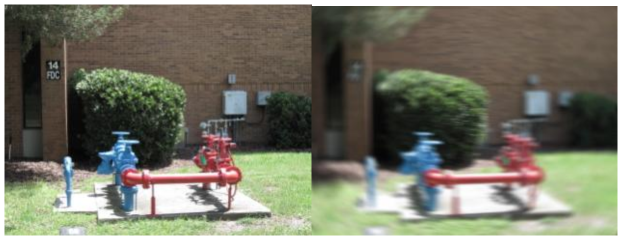


Fig. 6. Blur example.

Edge detection is more complex. It is implemented using a spatial mask derived from the Sobel Operator. With our design, a series of addition and subtraction circuits and shifters can derive horizontal and vertical gradient information, which is then summed and measured against a threshold value to determine if there is in fact an edge present. A diagram of the Edge Detection system can be seen in Fig. 7. Edge Detection is not a specified user function, but the design has been created and may be included if time allows.

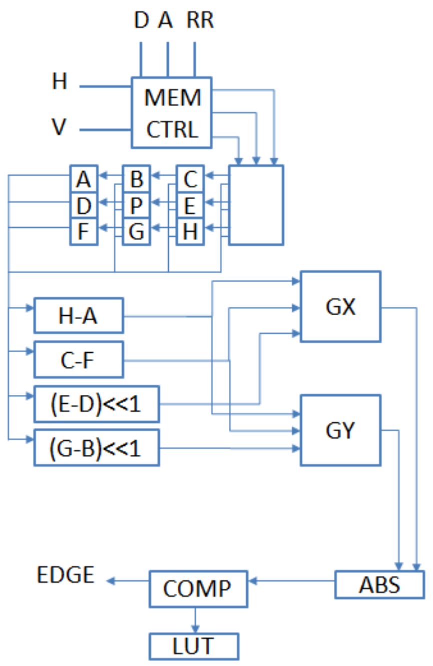


Fig. 7. Edge Detection example.

E. VGA Output

The VGA output section of the Video Processor tracks the current and future pixels to be used and has a pixel buffer for the next processed pixel. It generates its own horizontal and vertical synchronization signals from timing offsets defined in the system through experimentation.

Many monitors will support various timing adjustments, and modern LCDs will attempt to do active region determination from color output sections, but seem to get it wrong more than right. Without being able to use any form of EDID queries for the specific monitors, each test monitor has its own hand-tuned set of constants. To assist in this tuning a special test pattern generator was developed which supplies 60x60 pixel blocks at each corner further divided into 20x20 pixel blocks. This gives us information about framing, edge behavior, color skew, and sub-pixel appearance. The next part looks at 8 colors in different sized bands to see other effects of color bleed and provide large vertical sections in which any skew issues will be more apparent and measurable. Below this are independent color gradients, and full color range gradients to look at the dynamic range of the monitor and DAC in use. A small set of gray colors was selected in various sized bars for more timing and brightness analysis. This pattern can be seen in Fig. 8.

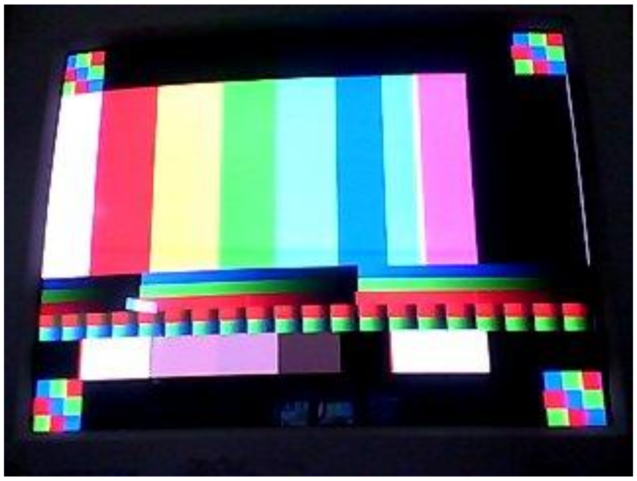


Fig. 8. Test pattern for output timing estimation.

The pixel data is taken from the buffer and supplied to the VGA card's Digital to Analog Converter. The DAC's sync generation system is bypassed to supply HSYNC and VSYNC directly to the monitor, with the DAC's blanking input driven high to tell the DAC that it is always in a non-blanking state. This avoids the superposition of sync signals in any of the color channels because they are not needed and lets the DAC operate on a pixel-by-pixel basis at every clock edge from the input clock. Essentially, this creates a simple VGA output pipeline for the data sent to the VGA card from the Video Processor.

IV. User Interface

The User Interface for the system is necessary to allow the user to control the image modification functions that the Video Processor provides. In this way, single or multiple functions may be selected to modify the video output shown on the Display screen in real time, as the user wears the Z-Goggles system. Shown in Fig. 9 is a simple flow diagram of the User Interface.

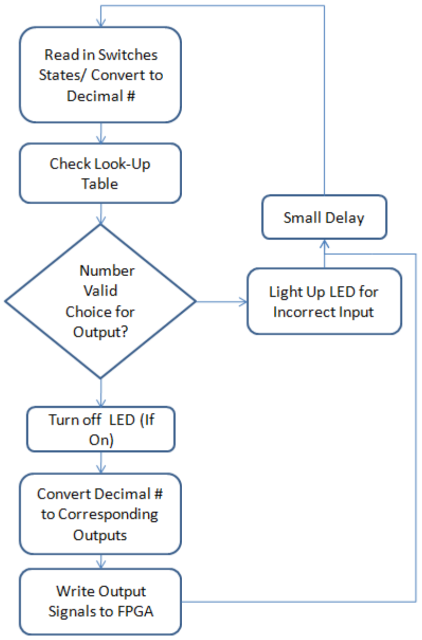

Read in Switches States/ Convert to Decimal #
Check Look-Up Table
Number Valid Choice for Output?
Light Up LED for Incorrect Input
Small Delay
Turn off LED (If On)
Convert Decimal # to Corresponding Outputs
Write Output Signals to FPGA

Fig. 9. UI Flow diagram

To accomplish this, we decided to use a simple set of switches and the Atmel ATMEGA328 microcontroller. The Arduino Duemilanove board was used to develop functional code for the microcontroller. In this case, the microcontroller simply checks for the signal change from each switch, checks to see if the signals are a valid choice, and informs the FPGA of the change. There is no set switch position, so it is a direct on or off command, depending on what state the function is currently under.

In addition to the switches, there is also an LED that is included to inform the user if an invalid combination of image modification functions is chosen. This UI circuit will be completed on a PCB; Fig. 10 is the schematic that the PCB design is based upon.

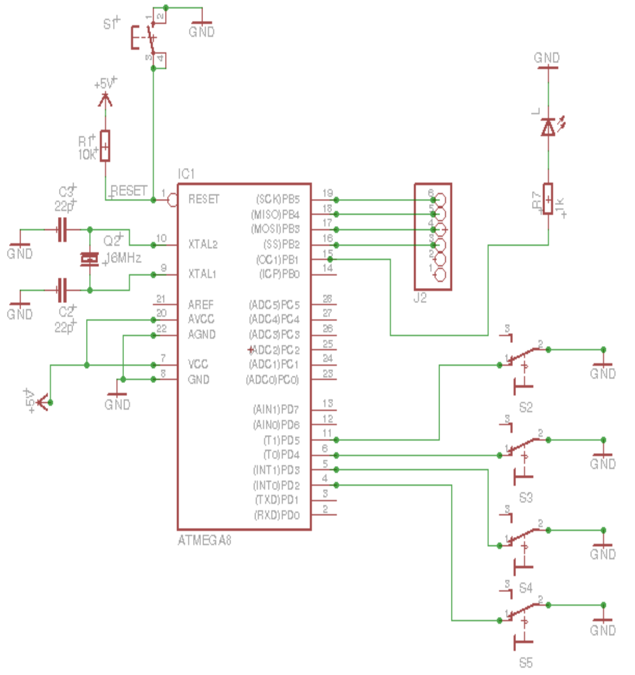


Fig. 10. User Interface Schematic.

V. VGA Board

One disadvantage to the chosen FPGA platform is its 8-color VGA output, which is insufficient to output 8 or 16-bit video. This is the reason a separate VGA card is required; the C3188A camera data output has much higher color depth than the Spartan-3 board can inherently handle. To solve this problem our group resolved to create a VGA output card that would handle the 8 or 16 bit color output data. This would require using a Digital to Analog Converter chip to convert the digital video signal into an analog output for our VGA display. Current VGA conversion circuits use a triple conversion process that uses one chip to convert the RGB/YUV digital signals into component red, green and blue signals. Our group chose the THS8133b digital to analog converter to accomplish this because of its ability to handle our maximum bit width for each color, as well as TI's generous distribution of samples for student projects.

Instead of designing a PCB to mount the DAC, our group decided to surface mount it to a premade 40 pin breakout board and construct connections to a DB-15 connector to connect to the Display unit. This solution was much quicker and cheaper than designing a PCB. To ensure a consistent power supply to the DAC, we soldered a 0.1μF capacitor between the analog power supply and the compensation terminal. To achieve an internal voltage reference of 1.35V we soldered a 0.1μF capacitor between the reference voltage pin and the analog ground. We wanted to make the full scale current 26.67 mA, so we soldered a 430Ω resistor between the full scale adjust control and the analog ground. This is necessary because we are running with sync on all RGB components instead of from one color (usually green). The other input pins are not used, so they are linked to ground to prevent any noise from interfering with the signal due to open pins [1]. Shown in Fig. 11 is the DAC prior to capacitor and resistor components being added.

Fig. 11. DAC soldered to the breakout board

VI. Display Unit

For the display unit we decided to go with a commercial off the shelf (COTS) LCD screen, the Accelevision LCDP5VGA. The screen has a pixel resolution of 640x480 which matches the ideal pixel resolution output that was specified during the design phase. The decision to use this display came after we realized that output to a computer monitor was more directly achievable with our FPGA than output in a composite format. In essence, this screen is a small computer monitor. The screen came with a DB-15 VGA connection so nothing more was needed to get it to function properly other than give it proper VGA signals.

Also, this display unit was chosen because it could be run directly from the 12 Volt battery power source without the use of DC-to-DC converters. As shown in Fig. 12, the backlight on the screen is acceptably bright when it is being run off directly from the battery at roughly 12V. It has a current draw of 300 mA, which is quite a bit less than we had originally assumed a display would draw. This should allow much longer system use-time before recharging the battery.

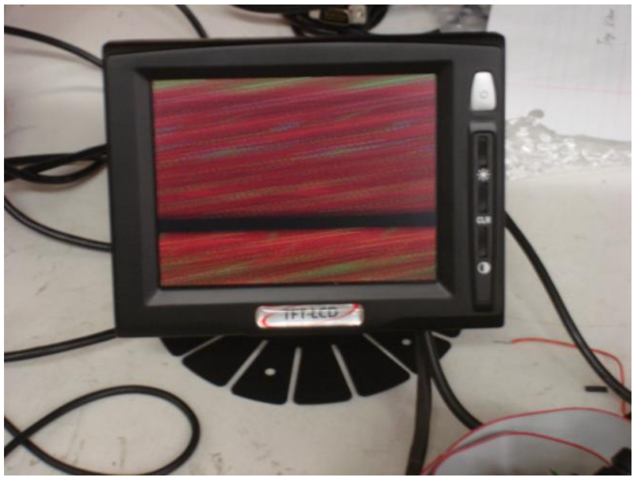


Fig. 12. Accelevision LCDP5VGA Display Unit. In this picture, it is displaying luminance values output from the camera, but without memory mapping the data.

VII. Power Supply

To power the entire system, as well as create a portable system, we must use a battery-powered power supply. Our chosen design consists of one voltage regulator that converts the 12VDC power source into a 5V source. The 5V source is for the FPGA Video Processor and the UI microcontroller. The FPGA has an onboard 5V supply that will run the C3188A digital camera. The Display unit runs from the 12V battery directly. The general power supply design is shown in Fig. 13.

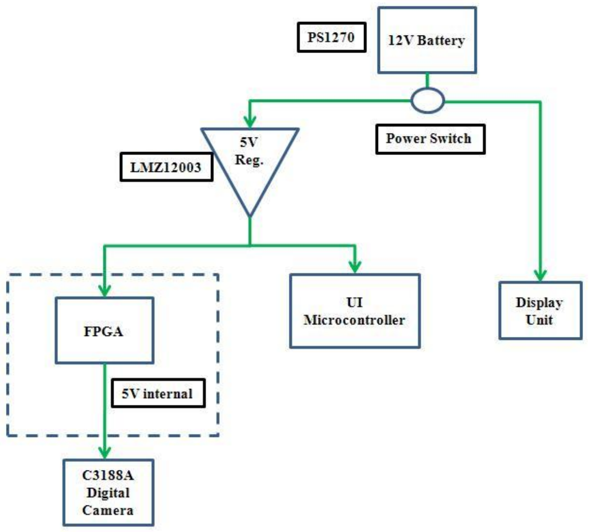


Fig. 13. Power Supply Design

The 5V regulator we have chosen is the LMZ12003. This is a micromodule from National Semiconductor, which has low loss due to heat, and high efficiency over its entire range of current output. In addition, it features protection against overvoltage and inrush current situations. The ENABLE pin on the device allows us to set a low voltage threshold beyond which the system automatically turns off. Using this feature, when the power switch is flipped to the off position, the regulator will turn itself and the rest of the system off [2]. Shown in Fig. 14. is the LMZ12003 regulator in the evaluation board. This board need only have two resistors modified to adjust the voltage output to 5V. R_{FBT} is removed and replaced with a 5.62 kOhm resistor, and R_{FBB} is removed and replaced with a 1.07 kOhm resistor [2]-[3].

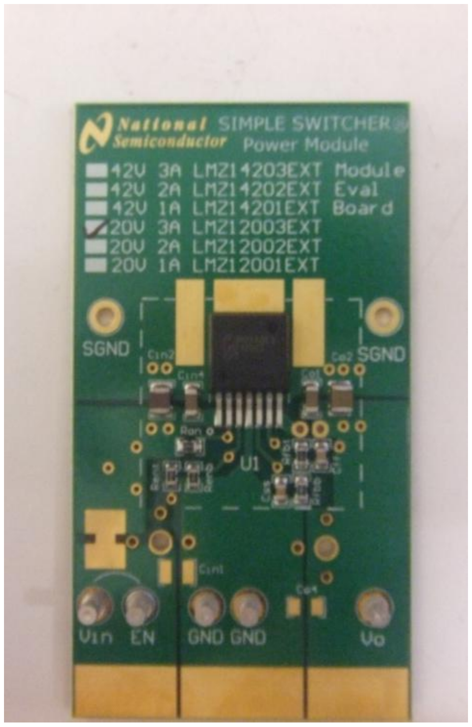


Fig. 14. LMZ12003 Evaluation Board.

The power source we have chosen is a PS1270 battery, which is a 12V output, 7Ah battery. The maximum current that the LMZ120003 regulator can output is 3A. Our system is well under this mark, with expected draw under 1.5A. At this level, the battery will operate for over 4 hours without recharge. Our goal for system battery life is 2 hours, so the battery capacity is double what we need. The battery is easily left in a charger until use, due to the fact that it is a Lead Acid battery which prefers to be left under a maintenance charge.

VIII. Physical Distribution

The overall system is held in two physical places on the user's body. The most complex aspect of distribution is the helmet, where the FPGA Video Processor, camera, display, UI controller, and most of the power supply is located. A backpack is included, which holds only the battery. This is done to isolate the battery from other aspects of the system for safety reasons. The connections between these two parts of the system are the power lines that connect the battery to the rest of the power supply.

The helmet setup consists of a thin sheet of plexiglass attached to an easily adjustable construction helmet. On this sheet is the FPGA Video Processor and the evaluation board portion of the power supply. The system On/Off switch is located here with the power supply. The UI controller is attached to the side of the helmet in such a way as to allow access to the switches for user input. The display LCD is hung out in front of the user's face with metal bars attached to the side of the helmet. On top of the display fixture, the camera is mounted to allow clear view in front of the user. The helmet is weighted to balance the torque these components exert on the helmet. The backpack setup is simply the battery placed in a backpack. It is covered for safety and electrical tape is used to cover the power wire connection to the battery.

IX. Conclusion

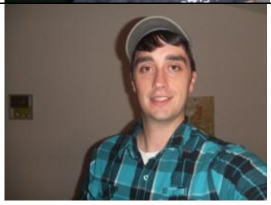
The Z-goggles system started as a fun way to apply what our group has learned; and, it ends as an exercise in problem-solving with low cost components. The difficulty of the project lies in utilizing these components to create a portable video processing system from the ground up. Every component, up to and including such mundane things as a VGA port, had to be designed and accounted for within the system. The FPGA Video Processor acts in the stead of many separate hardware components, and is built under many constraints on a low cost development platform. Our group has learned the art of design by dealing with many design changes throughout the creation of the system. Some of these changes were forced upon us by necessity, while some were added to better the system during the prototyping process. The result of these changes is a more compact system that has less functionality; but, it functions much more efficiently than the original design.

In the end, our group takes away a great deal of experience in the issues and triumphs that go along with a difficult design endeavor. There is no doubt that the Z-Goggles project embodies thinking outside the box to create something interesting. That was always the essence of our group's philosophy. Achieving that goal makes the Z-Goggles a successful Senior Design project.

Acknowledgements

The authors wish to acknowledge the advice given by Dr. Arthur Weeks and Dr. Samuel Richie regarding their overall project. Also, the authors would like to thank Tina McArthur of Lockheed Martin for advice regarding the power supply.

Group Biographies

	Cody Baxley will be graduating with a Bachelor's degree in Electrical Engineering from UCF in December, 2010. He plans to begin work and soon after pursue a Master's degree in Optics or EE with a focus in Opto-Electronics.
	Christopher Silver will be graduating with a Bachelor's degree in Electrical Engineering from UCF in December, 2010. He plans to begin work soon focusing on embedded system design and programming with a special interest in FPGA development for time-constrained applications.
	Geoffrey Jean is an electrical engineering student at the University of Central Florida. His current interests lie in microwave engineering, optical engineering and hardware/software programming.
	Will Petty is a Computer Engineering major at the University of Central Florida. His current interests lie primarily with software development in C, C++, Java, and Verilog. In the future, Will wants to work for a company doing a wide range of software development.

References

[1] THS8133b Triple 10-bit D/A Converter data sheet (http://focus.ti.com/docs/prod/folders/print/ths8133b.html)

[2] LMZ12003 3A SIMPLE SWITCHER® Power Module (http://www.national.com/ds/LM/LMZ12003.pdf)

[3] App. Note 2024: LMZ1200x Evaluation Board (http://www.national.com/an/AN/AN-2024.pdf)