3D RGB LED Cube

Luke Ausley, Joshua Moyerman, Andrew Smith

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

***Abstract* — The 3D RGB LED Cube is an exploration into precise and efficient hardware and software methods of simultaneously controlling individual numerous LED's with several high-performance requirements. The 10x10x10 voxel LED cube operates under strict timing conditions, providing complex animations to the viewer in bright full-color at high refresh rates. The conclusion of the project demonstrates the effectiveness of a joint FPGA/Processor control structure, a MOSFET/LED Driver current modulation structure, as well as a custom software solution for applications requiring fast, high-current, as well as color and time accurate LED control.**

***Index Terms* — LED, RGB, FPGA, Microprocessor, LED Driver, MOSFET, LED lattice animation, LED animation simulation.**

I. Introduction

The 3D RGB LED cube functioning prototype is built to a specific set of standards. Operating at a resolution of 10x10x10 voxels (three-dimensional pixels), the cube utilizes 1000 LED's and operates based upon the concept of persistence of vision (rapid blinking to give the appearance of a solid light source). By maintaining a minimum LED refresh rate of 100Hz, the cube is able to display seemingly continuous graphics to the observer at a minimum animation refresh rate of 25fps, while operating in full 24bit color space, each LED capable of producing 16.8 million individual colors. The cube size is outer casing which is slightly larger. The casing consists of acrylic sheets to protect the LED cube without diminishing the visibility of the cube. The entire system is powered entirely from a common AC wall outlet through the use of a commercial grade high current power supply. While this type of high-performing LED cube has been demonstrated before – this version is impressive due to the control and software design. The implementation of this project achieves efficient high standards of accuracy and performance through a previously unutilized methodology of combining an FPGA/Microprocessor to control the LED's. Further, the current modulation control structure operates more effectively through our implementation than in commonly-seen projects. Combine the innovative hardware design with a custom software suite with impressive features such as full animation simulation, real-time dynamic animation display, as well as pre-programmed animation display, and the result is a an LED cube that outperforms competition while providing a valuable proof-of-concept for less-common methods of software and hardware LED control.

II. System Concept

The basic idea for the 3D RGB LED cube operation will be explained here, in addition, several concepts key to the efficient and effective functionality of the LED cube will be discussed in detail. Fig. 1 details the system-level block diagram.

A. Operation Overview

The instruction flow begins with the user providing input to the computer on the software side. The user inputs animation commands via a GUI (either real-time to the cube, or by creating a sequence of animations ahead of time to display). This information is communicated to the embedded processor which communicates with the FPGA to provide instructions to the row and column circuitry to light up the LED cube. A method of wiring the cube, called multiplexing (discussed later) allows us to power only 1 horizontal plane at once. By cycling through each 10 planes very quickly, where the FPGA excels, what appears to be a solid animation is shown to the viewer.

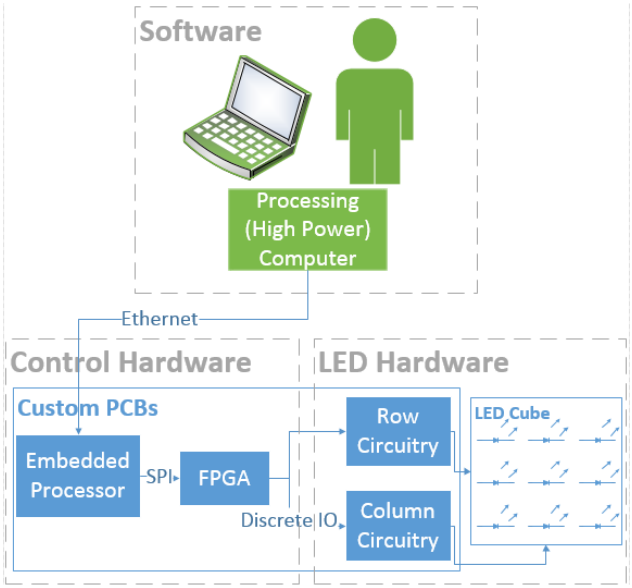


Fig. 1. High-level system block diagram

B. Multiplexing

One of the most common methods for driving large arrays of LEDs is to use a multiplexing scheme. This is the method that is used for most television and large format screens. Multiplexing controls large two dimension arrays of LEDs with less control lines than wiring each of the LEDs individually. With multiplexing, all of the anodes of a row are connected together and the cathodes of a column are connected together. By applying power to only one row, and grounding only one column, an individual LED can be turned on. By applying power to multiple rows, and grounding only one column, all of the LEDs in one column can be controlled. After this column has been controlled, the next column can be grounded, and its appropriate rows be powered. This cycle is repeated for each of the columns in the display, rapidly enough such that the persistence of human vision perceives that each of the LEDs that was turned on at one point, is constantly on. At a rate of approximately 50Hz, most people see the LEDs as constantly on, however some detect flicker. After approximately 100Hz, the scanning appears to be smooth. Using multiplexing, the number of control lines is drastically reduced. For an array of ten by ten LEDs only twenty control lines are required, as opposed to the 100 control lines that would be required by wiring them individually. Wiring the cube as an array of ten LEDs by three hundred (one hundred by three colors), 310 control lines are required, a drastic reduction from the 1,000 required by using the individual wiring method. Multiplexing also allows for easier wiring of the cube, anodes can be connected by plane, and cathodes by column or row, helping to provide structural support.

C. Pulse-Width Modulation

Pulse Width Modulation is used to control the color of each diode. This is done by varying the duty cycle of each individual red, green, and blue diode. A higher duty cycle (diode is on more often) corresponds to a brighter LED of that specific color. By varying the duty cycles, and correspondingly the brightness of each LED color, the user is able to display any 24 bit color from any LED through the current modulation structure controlled by the hardware control structure which receives user input via the software.

III. HARDWARE SYSTEM COMPONENTS

Each of the primary hardware components that have been noted in the previous sections will be mentioned in detail here with their specifications listed and discussed. Critical design concerns related to the selection of each component, as well as potential concerns will also be mentioned. Specifics about the integration of these hardware components to achieve the final result will be discussed in Section V.

A. LEDs

The decision for which LEDs to utilize in the 3D cube is absolutely the most critical choice to make. As the LEDs are really the principal component of the LED cube, the group chose a more difficult yet more rewarding path. Rather than utilizing commercial-off-the-shelf (COTS) diodes, custom LEDs were specified and ordered. This choice allowed a much higher degree of control over the rest of the design components. As the majority of design hinges upon the specifications of the LEDs, an initial custom design allowed the remainder of the design choices to be made with significantly more ease than if the LEDs presented more extensive design challenges and downsides. Our custom LED package has a 20mA forward drive current, and a max forward drive current of 100mA at 1/10 duty cycle. We plan to drive the LEDs at half of this maximum value, providing brightness with little risk of damaging the LEDs. The operating temperature is -40°C to 100°C with a 5V reverse voltage rating and a 1000V electrostatic discharge (ESD) threshold.

B. LED Drivers

The TLC5948A is one of the newer, more robust LED drivers that TI offers. Similar to its predecessors, it operates 16 channels at constant current, supplying either 2mA-45mA (Vin < 3.6V) or 2mA-60mA (Vin > 3.6V). Where it outshines the competition is with 16 bit brightness control (BC), stepping in 65536 increments. Further, it has 7 bits of dot correction (DC), allowing for a more precise constant-current correction between channels and separate devices. Another interesting feature of the TLC5948A is its 7 bits of global brightness control. TI cites the constant current accuracy of this driver at .6% (typ), 2% (max) channel-to-channel and 1% (typ), 4% (max) device-to-device. The TLC5948A has a data refresh date at 33MHz, a power saving mode to minimize Vin current, and expanded error notifications. It's six error flags are LED open detection (LOD), LED short detection (LSD), output leakage detection (OLD) reference current terminal short flag detection (ISF), pre-thermal warning (PTW), and thermal error flag (TEF). The error notifications are retrieved via a serial interface port. The operating temperature of the TLC5948A is -40 C to +85 C. Its maximum LED voltage is 10V. It is available in 24HTSSOP and 24SSOP packages. The Texas

Instruments TLC5948A is the best option for this design, it's a top of the line LED driver with all the extra functions. The primary selling point for this driver is its 16 bit brightness control. This will provide many more color variations than a smaller-resolution dimmer and greatly enhance the visual performance of the LED cube. The shortcomings of this driver compared to the others considered for this project ended up being moot. While it only has a 10V maximum LED voltage, this exceeds our requirements. Although it cannot provide more than 60mA of current (at 5.5V), the LEDs chosen for this project only require 50mA of current. The expanded error detection and 33MHz data refresh rate are some other less important reasons that contributes to the fact that the TLC5948A rises above its competitors for this application.

C. MOSFETs

The MOSFET chosen for this project is the Vishay/Siliconix Si4101DY. The most attractive quality of this MOSFET is it's low Rds(on) resistance, at 6.6mΩ (typ) and 8mΩ (max). This is vital, as at the max current of 15A, the max power dissipated in the MOSFET will be $P = 15^2 * .008 = 1.8W$. However, if the driver board is split into two separate boards, the power dissipated reduces to $P = 7.5^2 * .008 = .45W$, a very reasonable value. This low value of Rds(on) is difficult to achieve with such a high current - but clearly it is an important concerns for this design. If it were not this low, we would have serious concerns related to power dissipation in the PCBs. Additionally, the turn off delay time of 60ns (typ) is reasonable, whereas similar competitors are much higher. A correct MOSFET choice was vital, as both the problems of power dissipation and high turn off delay time had significant consequences. However, planning ahead and appropriately anticipating and accounting for these concerns with this MOSFET choice avoided any issue.

D. Microprocessor

The PIC24 by Microchip is a relatively powerful, 16 bit Harvard architecture processor. The processor is capable of running at 40 MIPS, more than double that of the Atmega chip listed above. This instruction rate is more than enough to be able to perform all of the processes required. The Microchip line of processors are compatible with their line of ethernet interfaces, via SPI and parallel interfaces. When using the Microchip line of ethernet interfaces, Microchip's entire TCP/IP stack can be used, which would greatly reduce the level of effort the group would have to put in to get a working solution up and running. The group chose to use the PIC24HJ256GP206A as the embedded processor for the project. This processor encompassed all of the features that we needed in order to provide a pleasing display, while being easy enough to work with, as well as cost effective. The PIC24 is not the fastest of the processors that the group considered, but is more than powerful enough to be able to communicate with all of the peripheral devices, and provide a low latency refresh rate. The PIC24 also is available in a much easier to work with package, TQFP, as opposed to the BGA package of the AM335X. The PIC was chosen over Atmel's Atmega device and other close competitors due to a higher overall clock rate, and a greater wealth of manufacturer supported development libraries and example code.

E. FPGA

Due to the strict timing requirements of the LED Drivers chosen, a microcontroller is not ideal for communicating with the drivers used. The group has decided to use an FPGA to handle the strict timing requirements of the LED drivers, and to lighten the total load on the CPU, to allow it to focus on data generation and transformation. The FPGA will be used to generate all of the necessary control signals for driving the LED Drivers, and the MOSFETs in order to provide the highest refresh rate possible. FPGA devices by their very nature are ideal for strict timing due to the fact that they are hardware devices, there are no interrupts to slow execution. Given the needs of the project, the group considers the best choice FPGA to be from the Xilinx Spartan 3 line. Specifically, the Xilinx XC3S200AVQ100 Spartan 3A. The group is limited to using an FPGA available in a TQFP package due to the ease of its use compared to a BGA or QFN package. This package will also simplify the design of the printed circuit board, allowing for the group to be able to route all circuit board traces with ease on a minimally complex board. The Xilinx Spartan 3A meets or exceeds all of the desired features for the system FPGA.

F. Power Supply

The Mean Well SP-200-5 provides an output current of 40A at 5V (200W). It accepts 90-264V AC or 254-370V DC as its input voltage. It is compactly enclosed at 200mm L x 100mm W x 50mm H. Although higher-end power supplies like this one will operate near 100% capacity, using a 200W power supply ensures that we will operate at an estimated 50% (5V * 20A = 100W) capacity with room to add any number of substantial additional features or functions if necessary. The Mean Well SP-200-5 is a clear power supply choice for this project.

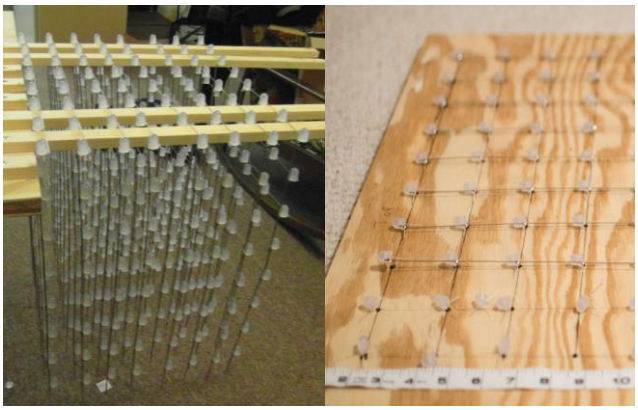


Fig. 2. Vertical and horizontal plane construction jigs [1]

IV. Structure Detail

A. Lattice

The structure of the cube is comprised of 20 gauge pre-tinned copper wire. This will provide both the electrical connections for the anodes and cathodes of the LEDs as well as the structural support for the cube. The wire is sold in spools of 100s of feet, with approximately 1000ft of wire needed for this construction application. Each piece of wire for the columns will be cut into lengths of approximately 26in each, and each piece of wire for rows will be in lengths of 22in each. The wire will be stretched to straighten the wire and increase its strength, preventing the wire frame from becoming bent. In the 3-dimensional coordinate system of the cube, each x-y plane will be soldered to the anodes of the LEDs, while the wires in z plane will reference the cathodes. The pitch, or how far each LED will be spaced from each other, will be 2in.

C. Base/Case

The cube is placed on a wooden base with holes for the columns (cathodes). Hidden in the base will be the control hardware: control board and driver boards as well as the ribbon cables that are soldered to connect the driver boards to the LED wiring. The outside of the LED cube will have acrylic paneling that attached to the base, protecting the delicate LEDs and semi-delicate copper wire structure without minimizing visibility.

V. Hardware Detail

A. Hardware System Overview

The hardware is comprised of two primary components: the control hardware and the LED hardware. Fig. 3 outlines a block diagram of these two components. The control hardware is simply the embedded processor and the FPGA. The microprocessor handles the interfacing with the user – as its increased processing speed is well-suited to this task. The FPGA however, gains a clear advantage in handling the interfacing with the LED drivers and MOSFETs, due to the strict timing constraints necessary to operate the cube at a refresh rate of 100Hz. Together, these two components select which LEDs are on at any moment. The task of actually turning on the LEDs is completed by 20 LED drivers and 20 MOSFETs split over 2 PCBs. Splitting the driver components over 2 PCBs achieves several important things: it significantly reduces the power consumption by cutting the current through any component in half, and it also decreases the risk that any one error or malfunction will have a cascading effect. Driver PCB 1 controls half of the vertical planes of the cube, and Driver PCB 2 control the other half of the vertical planes of the cube. The 10 MOSFETs housed on each board, when turned on, select which of the 10 horizontal planes are set to HIGH (common anode). The 10 LED drivers on each board have a maximum of 160 channels, which are able to control which of the 50 packaged LEDs * RGB = 150 LED's in that selected horizontal plane are on, and vary the duty cycle (select the individual color of each of the 50 packaged LED's).

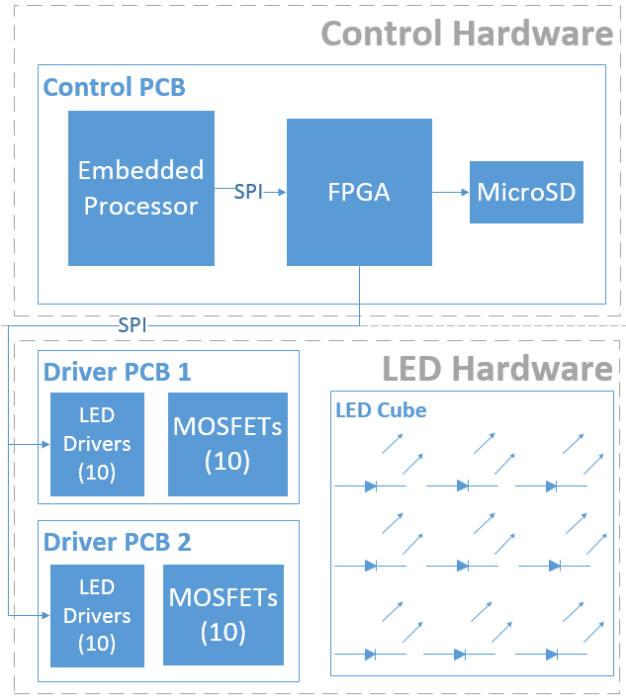


Fig. 3. Hardware system block diagram

Provide accurate instructions to the LED drivers in both driver PCBs as to which LED's in a horizontal plane are on, and what color they are, and cycle through each of the 10 MOSFETs on both PCBs at 100Hz, changing the LED driver instructions with each MOSFET, and an colorful, accurate, seemingly continuously lit animation is shown to the viewer.

B. PCB Design

Implementing the design explained in the previous section is not trivial. The PCB stage of the design is an involved and complicated process. Without discussing each component of the PCB design in detail, which would be outside the scope of this paper, Fig. 4 and Fig. 5 will show the PCB layouts of the control and driver boards, respectively, describing them in a very general sense. One important design decision to note is that on the driver PCB, each horizontal plane (MOSFET) on both boards are individually fused. This limits the consequence of a local malfunction to only 5% of the LED's on the cube, rather than a significant portion of them.

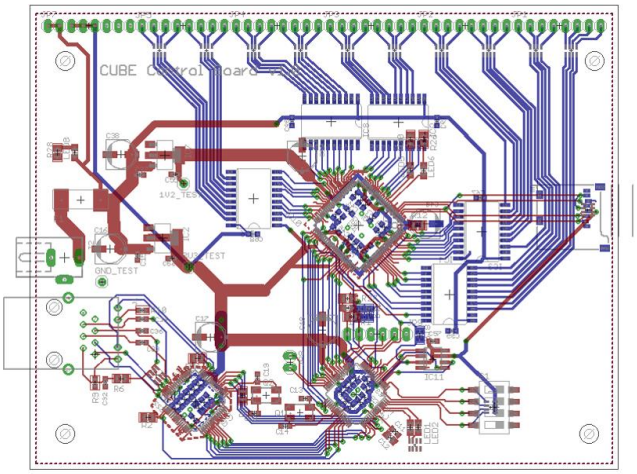


Fig. 4. Control PCB Layout

The largest element in Fig. 4 is the microprocessor, and the smaller element is the FPGA. The thin blue lines are the input/output lines for serial data, whereas the thicker, red lines provide the power to the components. Thicker lines are used when higher current is flowing. Note how no 90 degree angles are utilizes, as this can cause RF interference, an unwanted effect. Additionally, traces are made as short as possible, as with an increased trace length comes increased resistance and capacitance, also unwanted effects.

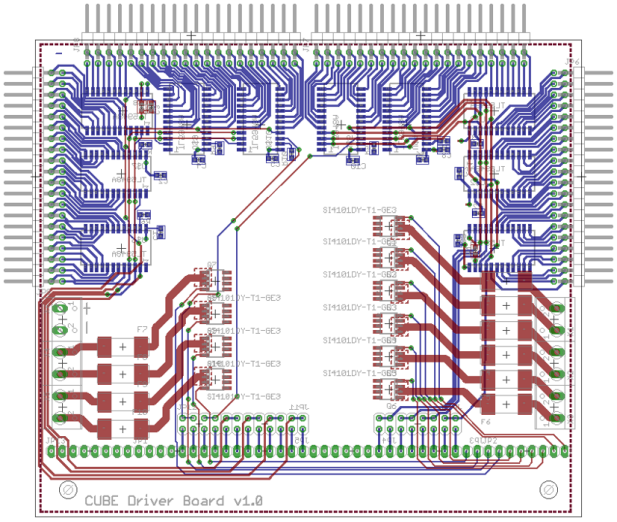


Fig. 5. Driver PCB Layout

In Fig. 5, the thicker red lines attach to the fuses and the MOSFETs. The MOSFETs are the components which have the most potential current through them. The smaller blue lines connect to the LED drivers, which output to the pins on the top and sides which connect to the LED columns via soldered ribbon cables. The same design principles that were mentioned for the control PCB are utilized here.

C. Firmware

The firmware component of the LED Cube is responsible for taking the lighting data from the ethernet interface of the processing computer and transforming that data to usable signals to the LED Drivers and MOSFETs contained within the system. The firmware is broken up into two sections, the set of instructions that will run on the PIC24 processor, and the hardware design of the FPGA. As seen in the high level block diagram, the PIC24 is responsible for receiving the lighting data from the host computer, and will buffer the data before passing it along to the FPGA. The processor will also manage the storage of any lighting transmissions received for future playback without the host computer. It will accomplish this with the usage of a common micro Secure Digital card. Animations stored will be played back at a later time, reading from this card. The FPGA will be responsible for receiving data from the embedded processor and will push the data out to the LED cube itself. The actual multiplex timing of the LED cube will be handled by the FPGA. The FPGA will read the data from its internal buffer and will pull only the data for each specific layer and send it out to each of the twenty LED drivers, while engaging the MOSFETs for that specific layer. This process will be repeated at a rapid

rate, well above the human persistence of vision in order to achieve a seamless, flicker free portrayal of the animation received from the host computer.

D. Communication

The hardware aspect of the system will be communicating with the host software, and other commodity peripherals using the Streaming ACN (sACN E1.31). The group chose to use this protocol due to two primary factors: its industry adoption, and its suitability for the project. The data is sent via the TCP/IP network stack from the host computer to the embedded processor. The data is then buffered and transformed before being passed into the FPGA via a Serial Peripheral Interface (SPI) on the Controller PCB. The SPI lines will enable very rapid communication between the two elements on the PCB. The FPGA will then send the lighting data out to the TLC5948A LED Drivers via four independent SPI lines, each controlling five individual LED drivers, to reduce latency, and increase the throughput and refresh rate. The processor will communicate via the SDCA's published subset of the Secure Digital specification. This subset is more than what will be required to be implemented by the group in order to record and playback the sequences from the SD card. This portion of the standard is available freely to all people, and does not require an implementer to join the SDCA group.

VII. Software Detail

The LED Cube software is comprised of three major components: animation, communication, and simulation. With the effective implementation of each of these components, the software can achieve two things: complete control of each aspects of functionality that the physical 3D RGB LED cube is capable of, as well as independent functionality testing (software operation can be tested while quarantined from all hardware interaction).

A. Animation

The animation component of the software represents the RGB LED Cube as an array of color values representing each voxel that is addressable in both 1D and 3D coordinate systems. Each RGB voxel is represented as a 32-bit value with four 8-bit channels for red, green, blue, and an alpha channel that is available for additional features. One feature we have applied to this alpha channel is the visible transparency of the hardware simulator, but it could also be utilized for anything else we may decide.

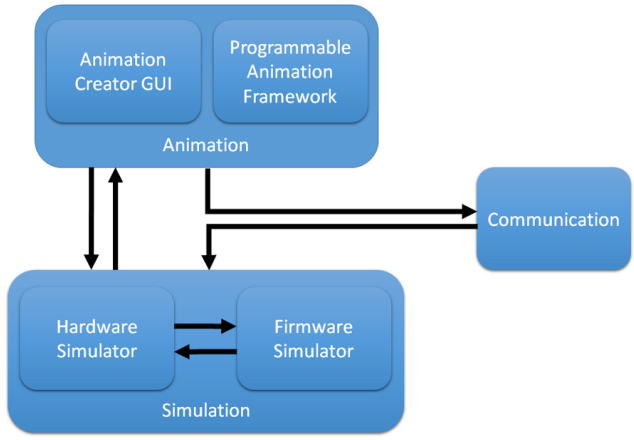


Fig. 6. Software block diagram

The software representation of the LED cube provides a functional framework for simple manipulation of the color data. The completion of this component allows us to create a new animation very quickly by manipulating the existing data on the cube in place and adding new data as necessary. This feature is beneficial because adjacent time steps will typical have similar lighting control data. The cube is divided further into layers in all three primary axes, which allows for an improved framework for animations such as scrolling. Translation of voxels within the cube is accomplished by treating the cube as a double-ended queue in each axis. Animations are built by creating a list of lighting control data and time duration pairs known as frames. A similar framework exists for animations to allow for rapid creation of new animation effects. Each effect contains the color data for the current cube and a value for the current iteration. Further complexity can be added to an effect by extending the base class and creating specific datatypes and functionality while maintaining

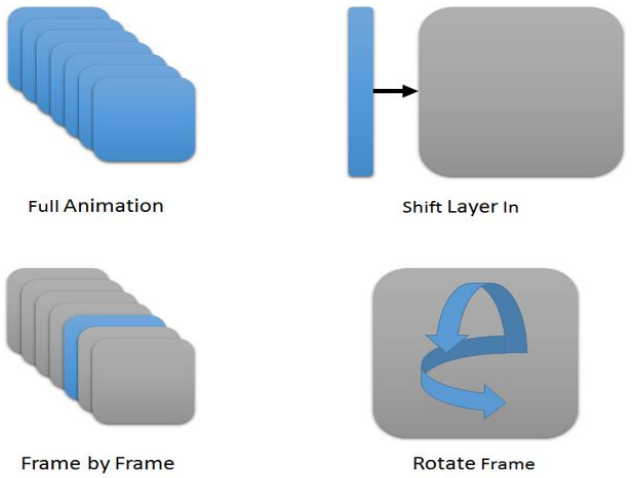


Fig. 7. Animation display methods

compatibility with all other effects. Once effects are created, they can be sent to the LED Cube in real-time or saved to disk for later transmission. Real-time animations allow for complex animations to be generated on a host machine and sent to the LED Cube in a dynamic manner. This allows for the incorporation of the additional computing power of the host machine for advanced animations that the FPGA cannot generate, this could include waveform generation based upon an audio input, utilizing a fast-fourier-transform, or any other computation-intensive animation generation. Generating animations and saving them to disk allows for complex animations to be computed once and replayed with a relatively small tradeoff in storage. A one-minute animation at 25 frames per second will only take up 4.3 MB of disk space.

B. Communication

The software transmits lighting control data to the control firmware by utilizing the Streaming Architecture for Control Networks (sACN) protocol. The software fully complies with the E1.31 standard for transmitting a common serial lighting control protocol over an Ethernet interface. The sACN protocol groups collections of up to 512 lighting control channels together in what is called a universe. A producer of sACN content will transmit each universe on a unique multicast address corresponding and consumers subscribe to the multicast addresses for their respective universes. The control data for the full LED Cube at 3000 channels is too large to store in a single packet due to the data limitations of the communication protocol. For this reason, the cube is divided into approximately even segments across six universes so that each cube section can fit into the 512 channel universe limit. Each voxel contains a red, green, and a blue color channel for a total of 3 channels per voxel. The first five universes contain 167 voxels each for a total of 501 channels in each of the universes, while the final packet (universe) contains the remaining 165 voxels for a total of 495 channels in that universe. Together, these 6 universes contain the 1000 voxels (3000 channels) for the full control of the LED cube. The sACN protocol is extended beyond the tradition real-time transmission usage to include additional transmission modes such as offline transmission of full animations for storage and replay at a later time. The first byte of DMX512 data in a sACN corresponds to the START CODE of each universe, which is typically 0x00 for real-time transmissions. The E1.31 specification does not mandate a required value, therefore we define transmission modes using this value. The START CODE byte is further divided into the upper and lower nibble which correspond to major and minor modes of transmission respectively. This allows the capability for up to 16 transmission techniques with up to 16 unique message types within each technique.

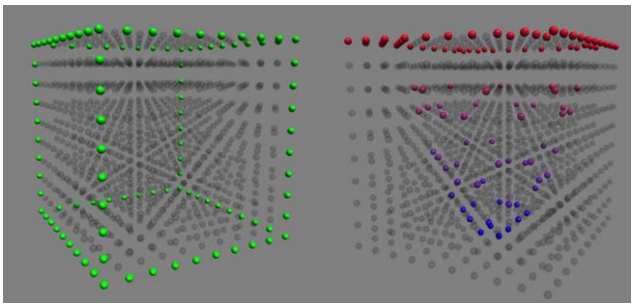


Fig. 8. View of animation simulator

C. Simulation

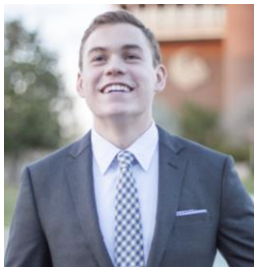
Through the use of software simulators, we are able to develop and test software features independently of the hardware and firmware development process. The hardware simulator simulates the physical LED cube in software using the OpenGL framework to create a 3D rendering of the LED Cube in space. Each LED voxel is individually addressable and can obtain any of the 24-bit RGB values the physical LED cube is capable of displaying. The OpenGL rendering utilizes the alpha channel of each voxel as a transparency value based on the intensity of the other color channels. The rendering of semi-transparent voxels in simulation mirrors the physical phenomena exhibited by LEDs that a fully lit LED remains partially visible when placed directly behind a less intense LED. In simulation, users have the ability to rotate the cube along any of the three primary axes in order to view animations from all angles similar to walking around the physical cube structure. The hardware simulator allows for the development and debugging of any type of unique animation that the LED cube is able to display without the need to connect to our physical cube itself. The firmware simulator recreates our control firmware in software. The firmware simulator interfaces directly between the ethernet interface and the hardware simulator and behaves as nearly as possible to the actual control firmware. All lighting control data processed by the firmware simulator is passed over a network interface and can be placed on separate computers recreate the network environment we will utilize or on the same computer for convenient debugging of transmission methods. The simulation component is key to the success of the software portion of the project as a whole due to the flexibility and power it gives us as developers. By creating accurate simulators using well-

defined input and output parameters of both the firmware and hardware components, the software is able to be developed and debugged entirely separate from the other portions of the project with minimal integration testing at the culmination of the project. This is such as significant component to the design process as it allows the two separate components of the LED cube design: hardware and software development to proceed parallel to each other, both meeting at final completion simultaneously, fully prepared for seamless integration.

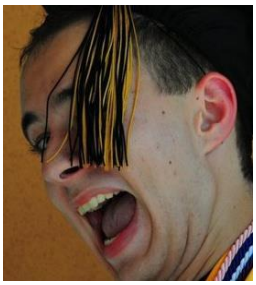
VIII. Conclusion

The completion of this project took its group members past their formal classroom electrical and computer engineering experience. By covering a broad range of topics, the 3D RGB LED cube challenged each group member to grow their expertise in their respective areas of interest. By completing the senior design process of planning, designing, and executing that design, our group has acquired a sense of competence in the real-world engineering process. Beyond personal development, the result of this project – a functioning 3D RGB LED cube prototype – was successful. Although numerous design and execution challenges presented themselves, they were well anticipated by our group and handled appropriately within a reasonable time frame. This commitment to planning, effective management of time, resources, and effort, as well as individual technical competency of each member contributed to the final success of the project, providing a valuable conclusion. We have demonstrated that our method of software and hardware integration, our control structure, and each of the other significant design aspects are an efficient and effective way to create 3D animations displayed on a LED cube.

Engineers



Luke Ausley will graduate Summa Cum Laude with his Bachelor of Science in Electrical Engineering (BSEE) in May 2014 to pursue a career in his field of interest: Optics. In his free time, Luke enjoys several hobbies with notable interests related to cars, landscape/nature photography and high-fidelity audio equipment. Enjoying traveling, one his life goals is to set foot on each continent. A recipient of the Department of Defense SMART scholarship, Luke has worked full-time as an engineering intern for the past 4 summers with the Air Force Research Lab's (AFRL) Munitions Directorate at Eglin AFB, FL. He has accepted post-graduation employment with the AFRL as an electronics engineer.



Joshua Moyerman anticipates graduating with his Bachelor of Science in Computer Engineering (BSPE) in May 2014 to pursue a career in the field of embedded development. Josh has volunteered and worked throughout his college career to expand his engineering experience. Joshua is currently working for Stellascapes, a company largely responsible for the sponsorship of this project, and hopes to continue working there after he graduates. He hopes to also spend more time enjoying his hobbies of photography, music, and reading.

Andrew Smith joined the Air Force ROTC his junior year of college and will commission into the United States Air Force (USAF) as a 2Lt when he completes his Bachelor of Science in Computer Engineering (BSPE) in May 2014. Andrew interned with the AFRL Information Directorate's Information Assurance Internship in the summer of 2013. He currently works as an intern in the College Work Experience Program with Lockheed Martin developing and supporting engineering tools for the Arrowhead fire control system. After graduation, Andrew will begin work with the USAF as a Cyberspace Operations Officer pursuing his passion for cyber security and defending the nation's network assets.

Acknowledgement

The authors wish to acknowledge the assistance and support of the Stellascapes.

References

[1] "RGB LED Cube," *How Not To Engineer* <http://www.hownottoengineer.com/projects/rgb-led-cube.html