SkyFi

The Wireless Networking Paradigm

Dhruva Kaushal, *Electrical and Computer Engineering, Carnegie Mellon University*
Syed Raziq Mohideen, *Electrical and Computer Engineering, Carnegie Mellon University*
Sribhuvan Sajja, *Electrical and Computer Engineering, Carnegie Mellon University*

***Abstract*—SkyFi is a near-range full-duplex communication system which uses infrared or visible light (VLC) at the physical layer to transfer data. Precisely-timed pulses of light are transmitted via Overlapping Pulse Position Modulation. Using simple components (LEDs, photodiodes) and FPGA-based encoding/decoding, we were able to achieve speeds of at least 1.1 Mbps at a distance of a few centimeters to a meter. SkyFi demonstrates the reliability, performance, and robustness of VLC as a viable communication medium for the future.**

***Index Terms*—FPGA, Infrared, Modulation, Networking, Visible Light Communication, Wireless Communication**

I. Introduction

Wireless data transfer is a key component of communication between the numerous devices that surround us today. With the advent of IoT and smart-devices, the wireless networks that support them also need to scale up proportionally. Many common wireless networks such as Wi-Fi, Bluetooth, and 4G are based on data transmission using radio waves. When the number of devices increases, the fixed bandwidth makes data transmission slower as radio waves are a small part of the electromagnetic spectrum. It is becoming increasingly difficult to develop new radio-based standards due to the decreasing available spectrum. These networks are also susceptible to security threats since radio waves can pass through walls. In places like hospitals, airports, and military bases, radio waves are intentionally interfered with so that data transfer is restricted. With our capstone project, we aim to address some of the above key technical challenges using visible light since recent research predicts visible light communication to have comparable transmission speeds, orders of magnitude more bandwidth than radio waves, suitability in areas sensitive to electromagnetic interference, and increased security and directionality.

SkyFi is a system that demonstrates the benefits of visible light communication. It supports full-duplex communications between two nodes. As VLC is still a nascent technology, there is ongoing research on how to most effectively exploit the available spectrum. Based on current understanding, we aimed for a throughput of at least 1 Mbps with greater than 99% accuracy. As a secondary goal, we sought to maximize the distance between nodes to at least 1 meter.

II. Design & System Description

Refer to Figure 13 for the complete system level block diagram. The overall design can be broken down into three sections.

A. The Physical Layer

In the data-transfer and communication system, the physical layer is the first and lowest layer in a network. It is the fundamental layer which underlies the higher level functions in the network. It consists of the electronic circuit transmission technology of a network. For our project, this represents the circuitry which would transmit the signal, in the form of bits, from the transmitting FPGA to the receiving FPGA. We created two circuits: a transmitter and a receiver. The transmitter takes a digital signal from the GPIO pin of the FPGA and broadcasts it to any listening receiver circuits by modulating an LED. This idea of modulating an LED forms the backbone of the physical layer in visible light communication. The receiver circuit is responsible for picking up any broadcasted signal from a transmitter circuit and reconstructing the transmitted signal successfully. This receiver would be connected to the GPIO pin of the receiver circuit which would interpret the received signal.

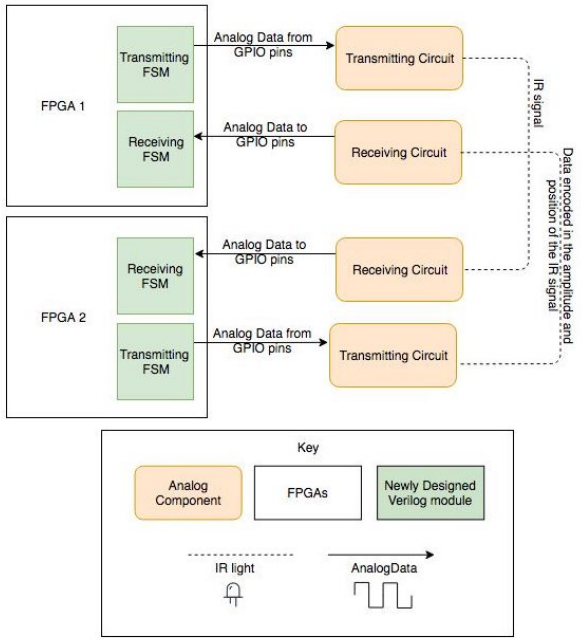

Fig. 1. Block Diagram depicting interaction and signals between components

Figure 1 depicts the interactions between the various components of the physical layer which enable a full duplex communication between two distinct nodes. The transmitting

FSM of the transmitter FPGA drives an analog voltage signal which represents the data we want to broadcast to receivers. The signal flickers the LED in the transmitter circuit, thus broadcasting to all receivers within the viewing angle of the LED. The photodiode in the receiver circuit which is in the field of view will generate a current corresponding to the wavelength and intensity of the incident light. This is amplified and then reconstructed and sent to the receiver FPGA for interpretation.

B. The Data Link Layer

The data link layer is the bridge between the physical and application layers of the networked system. It packages bits from the application into frames which are sent out as pulses to be emitted by the LEDs, and it receives other nodes' pulses through the photodiode and interprets them as packets of data. This is referred to as modulation and demodulation and it is where a great deal of research is being done regarding visible light communication.

Our modulation technique is inspired by a Dartmouth project named DarkLight [1]. It is called Overlapping Pulse Position Modulation (OPPM) and it is an advanced version of Pulse Position Modulation. Bits are transmitted as n-bit symbols, and each symbol is represented as a single pulse of length t_{pulse} within a period of time t_{symbol}. Each symbol period is divided into 2^n time slots of length L. Data is encoded by the time slot in which the pulse is located (hence, pulse position). Unlike regular PPM, OPPM is only concerned with the rising edge of the pulse, allowing for $t_{\text{pulse}} > L$ (hence, overlapping). To achieve our speed goal of at least 1 Mbps, we used 2-bit symbols and a time-slot width of 400 ns. Refer to Section III-B for information regarding how these values were determined.

Figure 2 summarizes the high-level design of the Data Link Layer. The Encoder buffers the data to be sent and divides the packet into a series of data pulses. Each packet is preceded by two preamble pulses for synchronization. Each data pulse is transmitted using the Modulator, which uses counters to emit pulses in the correct slots. On the receiving side, a low-pass filter is used for debouncing purposes. The Decoder looks for rise edges of each packet. The preamble pulses are used to synchronize the Decoder's counters so it properly demodulates the data pulses. The packet is then reassembled and presented to the application layer.

C. The Application Layer

The application layer is the layer closest to the end user. We wanted an application which demonstrates the working of our network while also showcasing its real-world usefulness. The scenario we thought would best show this was a file transfer over the SkyFi network. We simulate a duplex file transfer by allowing two nodes to send packets of data to each other. The data is generated psuedo randomly by a linear feedback shift register (LFSR). The LFSR of both nodes is initialized with the same seed. This allows both nodes to know exactly what data is being sent and what data it should be receiving. Once a node receives its expected data, it increments a counter and goes onto generate and send the next random number. This will work as a signal to the other node that the sent data has been received and will increment its own counter and generate the next random number to send. We can use the counters to practically measure the number of packets of data correctly received by each node.

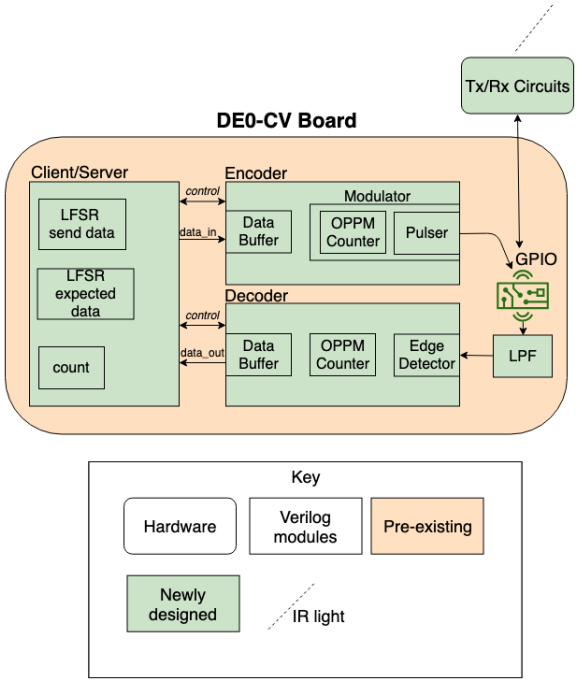

Fig. 2. Data Link Block Diagram

D. Changes From Design Report

Our final project is different than the one described in the design report, and therefore changes were made. Briefly, the originally-proposed project was a network of nodes connected to a central hub using VLC. Users would be able to play multiplayer Pong, with the hub acting as a game server. This project had a much lower speed requirement of about 1.6 kbps but also directly demonstrated the concurrency support of OPPM.

1) Data Link Layer Changes: Different OPPM parameters were needed in order to achieve the combination of speed, distance, and concurrency. In our design, we used 8-bit symbols and a time slot width of 16.22 μs for a throughput of 1.926 Kbps. This speed was sufficient to support the Application Layer. Because the server had to handle concurrent transmissions, we were especially concerned with goodput. Through probabilistic analysis, we chose a speed that balanced raw throughput and collision rates.

Relatedly, there was more of a focus on error handling. We had selected Reed Solomon codes, as they support short packets, have hard decision decoding support, are low in complexity, and are suited for transmission purposes. However, from our testing, we found that error correction was not necessary for our system, though this would definitely be used for any networking built on top of SkyFi.

III. Design Trade Offs

A. Application Constraints

Our application drives our design choices. With our new application of data transfer, the main goal is speed and accuracy. In the real world, transferring a file as quickly and reliably as possible is of utmost importance. We start with a base speed of 1.6 kbps. This speed ensures an extremely low error rate due to the high probability of correctly receiving pulses. We then incrementally modify speed and accuracy to get the best parameters for our project, as described below.

B. Modulation Parameters

In Overlapping Pulse Position Modulation, the bits per symbol n, symbol width t_{symbol}, and time slot width L are all variables that can determine the data rate or throughput T. They are related by Equation 1:

$$T = \frac{n}{t_{\text{symbol}}} = \frac{n}{2^n L} \tag{1}$$

In our design space exploration, we set n and L as independent variables and plotted the resulting throughput in Figure 3. It is clear that reducing the bits per symbol and time slot width both increase data rate. Ideally, to reach 1 Mbps, we would make $n = 2$ and $L \leq 500$ ns.

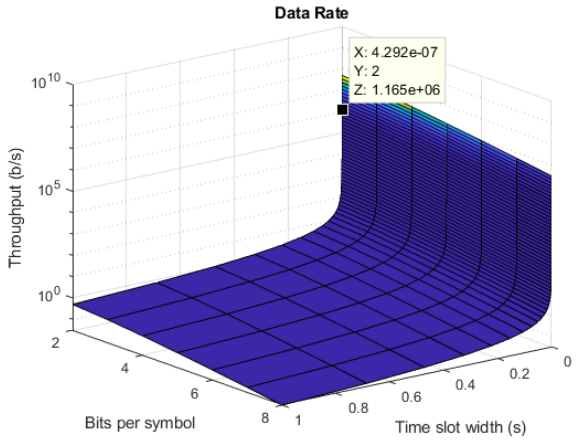

Fig. 3. Data Rate

Our system demonstration does not require the receiver to disambiguate between multiple concurrent transmissions, so a few design choices were simplified. In order to distinguish between multiple transmitters, the receiver would need to be able to accurately determine when the rising edge occurs. Depending on the sampling rate, there is an offset in the measured rising edge from the start of the intended time slot. In [1] the distribution of offsets tends to follow a Gaussian distribution. For a 1 MS/s sampling rate, the offset range $\theta = 1\mu$s covers at least 90% of this distribution. However, since we only have two nodes, θ is allowed to be much wider and cover more of the slot.

Another simplification is that we did not have to worry about collisions between multiple transmissions. It is *likely* but not **guaranteed** that different transmitters' clocks (and thus pulses) are unaligned. The probability of two transmitters sending a pulse within the same time period (which is the offset range θ) is given by Equation 2:

$$p = 1 - (\frac{L - N\theta}{L})^{N-1} \tag{2}$$

This equation, proven in [1], demonstrates the effects of slot width, number of devices, and offset range have on the probability. In the original plan for SkyFi, $N = 4$ and $\theta = 1\mu$s so Figure 4 demonstrates the relationship between L and p.

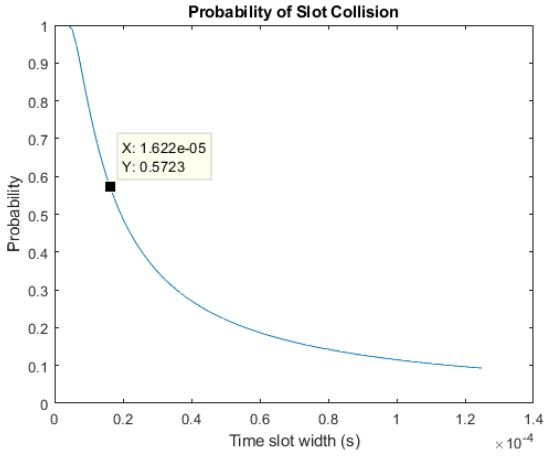

Fig. 4. Collision Probability

As is evident, shorter time slots lead to more frequent collisions. Thus, we had to balance between speed and reliability in the multi-node SkyFi system. Here, we do not have to worry about collisions so we could focus solely on accuracy and distance.

C. Equipment for the Physical Circuit

In order to support the above mentioned data-rate, the physical circuit needs to have parts which are able to operate at that frequency. Specifically, the IR LED, photodiode and the op-amp should be able to operate as expected.

As our IR LED, we chose the VSLY3943. The key rationale for choosing the VSLY3943 was that it has a cut off frequency of 12MHz which is more than what we need to transmit at at least 1 Mbps. In addition, the rise and fall times are both less than 6 ns, meaning the LED can support pulses as short as we need to.

Initially, we planned to order an off-the-shelf trans-impedance amplifier to use in the receiver circuit. However, we found that we get more control over the voltage levels by assembling this ourselves. Most of the equipment we need is available in the CMU labs. The only component we will order is the operational amplifier. We chose the LM 7171 op-amp. The choice of the this op-amp was dictated by the gain-bandwidth product value. We know that op-amps have a finite bandwidth. A fairly practical approximation of the frequency response is the integral of the gain. The gain of an op-amp is inversely proportional to its frequency. As an example, an op-amp with a gain-bandwidth product of 600MHz has a gain

of 6 at a frequency of 100MHz. Another factor that we needed to consider was the slew rate of the op-amp. The slew rate is defined as the maximum rate of change of the output voltage in response to the input voltage. Since we are operating at high frequencies, we need the slew rate to be as high as possible. In an ideal op-amp, the bandwidth and slew rates are both infinite. For our purposes, the LM7171 which has a gain-bandwidth product of 20MHz as well as a slew rate of 4100V/s, works perfectly and is strictly higher than needed.

The QSD2030F photodiode in the circuit acts as a current source that peaks for the 880 nm wavelength which corresponds to the IR part of the electromagnetic spectrum. This was another factor that determined our choice of the IR LED as the transmitter, which peaks at 940 nm. The QSD2030F has a rise and fall time of 5 ns which means it can theoretically operate at frequencies as high as 50 MHz which is much higher than the frequency we intend to operate at. It also has a forward voltage of 1.3V which is almost equal to the forward voltage of our IR LED.

Note that having the physical circuit support frequencies higher than what we expect to be operating at allows us to make further speed optimizations later on without having to change the design of the physical circuit.

D. Modulation Parameter Testing

Although we had identified parameters that should theoretically work, we still didn't know how reliable the communication would be once integrated with the physical layer. We performed a series of experiments to test how far we could push our system on speed, accuracy, and distance and so we could make implementation changes far enough in advance should they be needed.

The first test we performed was a test on how receptive the edge detection was given varying pulse widths. We used a function generator to transmit pulses of varying widths. Due to the nature of our receiving circuitry, the pulse widths did constrain the slot width; if pulses overlapped, then their edges would be lost. Finally, our filters used a shift register to buffer the most recent samples. The "history size" turned out to be another important variable. We fixed the distance at about 3 cm when performing these tests. Our results are presented in Figure 5 and Table I.

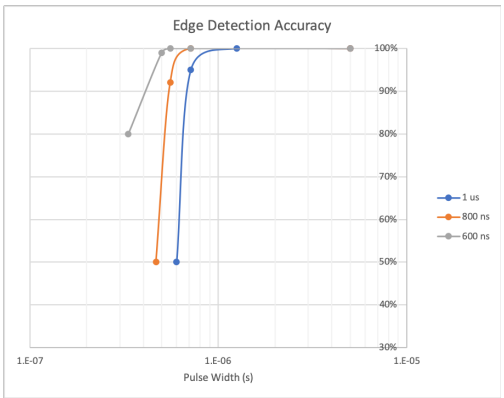

Fig. 5. Experimental Results

TABLE I
EXPERIMENTAL RESULTS

History Size (s)	Pulse Width (s)	Accuracy	Theta (s)
1.00E-06	5.00E-04	100%	1.23E-06
1.00E-06	5.00E-05	100%	1.23E-06
1.00E-06	5.00E-06	100%	1.23E-06
1.00E-06	1.25E-06	100%	1.23E-06
1.00E-06	7.14E-07	95%	1.23E-06
1.00E-06	6.00E-07	50%	1.23E-06
8.00E-07	5.00E-06	100%	1.23E-06
8.00E-07	7.14E-07	100%	1.23E-06
8.00E-07	5.56E-07	92%	1.23E-06
8.00E-07	4.67E-07	50%	1.23E-06
5.00E-07	5.00E-06	100%	6.60E-07
5.00E-07	7.14E-07	100%	6.60E-07
5.00E-07	5.56E-07	100%	6.60E-07
5.00E-07	5.00E-07	99%	6.60E-07
5.00E-07	3.33E-07	80%	6.60E-07

We found that the filter size played a bigger role in the edge detection accuracy than we expected, and this inspired us to change our implementation of our filter to be more reliable. Once we did, we were able to use pulse widths even lower than 500 ns, down to just 100 ns. This let us set the slot width to 500 ns and achieve at least 1 Mbps speeds. As described in Section V we performed further validation tests to ensure reliability.

IV. IMPLEMENTATION PLAN

A. The Physical Layer

The way we implemented this was by constructing the physical circuits onto a breadboard and wiring them to the FPGA. Each node had a transmitter and a receiver circuit.

The transmitting circuit is a simple IR modulation circuit which flickers an IR LED. The on voltage of the selected IR LED is 1.4 V. This means that for the bit 0 for which the FPGA will supply a 0 V, the LED will be off. Conversely, for the bit 1 for which the FPGA supplies 3.3 V and 5 W of power, the LED will be on. The circuit is thus able to broadcast the data as a stream to any listening receiver circuits inside its field of transmission. The intensity of the transmitted light is proportional to the voltage applied across the circuit. The only device that requires a power source in this circuit is the FPGA. The rest of the circuit is driven by the power supplied by the FPGA. We considered constructing alternative circuitry involving transistors to drive higher current through the LED, but this was deemed unnecessary to achieve the speed and reliability goals. If we were to expand on SkyFi and increase distance, this would be an area to look into.

The receiving circuit is responsible for picking up any signal being broadcast in the field of the photodiode and successfully reconstruct the originally sent signal. The way the above circuit does this is by using a trans-impedance amplifying circuit with the photodiode acting as a current source. The current source produces a potential difference which gets amplified at the output of the op-amp. The GPIO pin of the receiving FPGA is connected in parallel with a small resistor at the output node of the op-amp. Hence, it is able to read the amplified signal. The photodiode produces a current proportional to the intensity of light incident on it. Since the

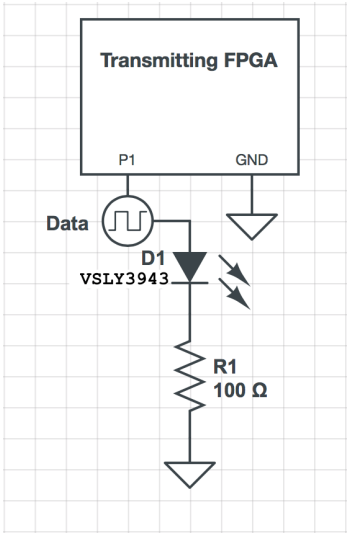

Fig. 6. The Transmitting Circuit Diagram

photodiode is largely in phase with the transmitting IR LED, as long as we choose components with the correct bandwidth, we know for certain that when the IR LED is transmitting a 0 bit, the photodiode will not produce any current and thus the voltage will be 0 at the output of the op-amp. Similarly, when the IR LED is transmitting a 1 bit, it has a voltage of 3.3 V across the circuit. The photodiode will produce a proportional current and according to our calculations, the output node of the op-amp will have a value of 3.3 V which will be interpreted as a 1 bit by the FPGA.

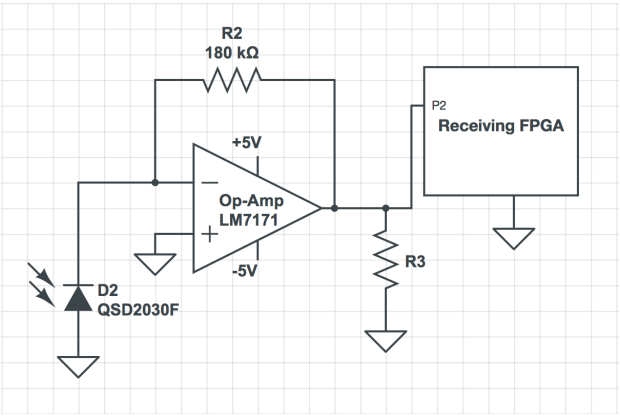

Fig. 7. The Receiving Circuit Diagram

B. The Data Link Layer

The data link layer runs on the FPGA. All FPGA development (including at the Application Layer) was done using Synopsys VCS to develop and Intel Quartus to synthesize our hardware designs. Each module (modulation/demodulation, encoding/decoding, etc.) was developed in isolation and tested before being integrated as part of higher-level constructs.

The Encoder stores a packet of data at a time in its internal buffer, which is a shift register that allows n bits to be transmitted at a time. The OPPM Counter counts off slots, and the Pulser transmits each pulse, of length t_{pulse}, in the correct slot. The Pulser's output goes through one of the GPIO pins to the physical layer. Refer to Figure 8 for an overview of the transmitter FSM.

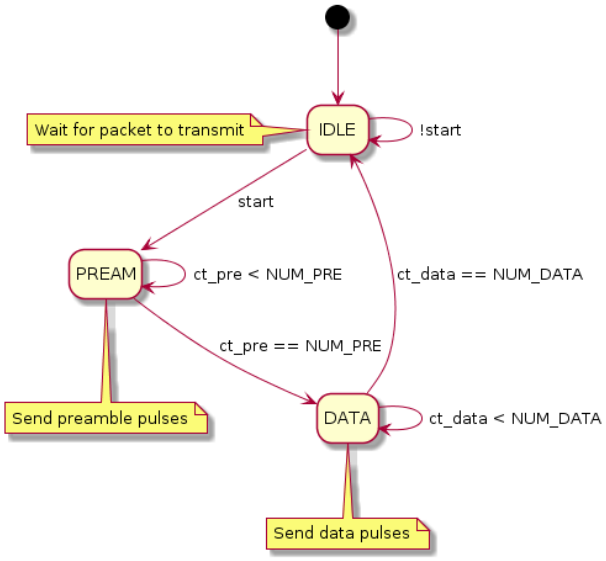

Fig. 8. Encoder FSM

On the receiving side, we use a simple filter that resembles a switch debouncer. It samples the GPIO input at the FPGA's clock frequency (50 MHz) and buffers the last few samples. We only needed a history size of 100 ns to reliably filter out noise. Then, the Decoder uses a simple Edge Detector to find rise edges. When the preamble pulses are received, the Decoder's counters are synchronized, and it accurately demodulates the incoming data pulses. Refer to Figure 9 for an overview of the receiver FSM.

To achieve at least 1 Mbps, we used $t_{\text{pulse}} = 100$ ns, $n = 2$, $L = 400$ ns, $\theta = 160$ ns, and a filter history size of 100 ns. Distance was not a primary goal, but we were able to transmit at about 5-10 centimeters. With slower speeds (1.6 Kbps), we could get up to 1 meter, showing that our modulation technique has potential to be developed with better components for higher performance in all areas.

C. The Application Layer

The main FSM of the Application Layer is straightforward; each node has an expected data and data send inputs. When the data packet received from the other node matches the expected data input, the FSM generates a signal that increments a counter and causes the LFSRs to generate the next random values. The node will then start to send the new random number while expecting a different value from the communicating external node. Figure 10 shows this process in a visual format.

Every packet transmitted was 8 bits long in our implementation. Thus, 2 preamble pulses and 4 data pulses are transmitted for every packet of data transmitted between the two nodes.

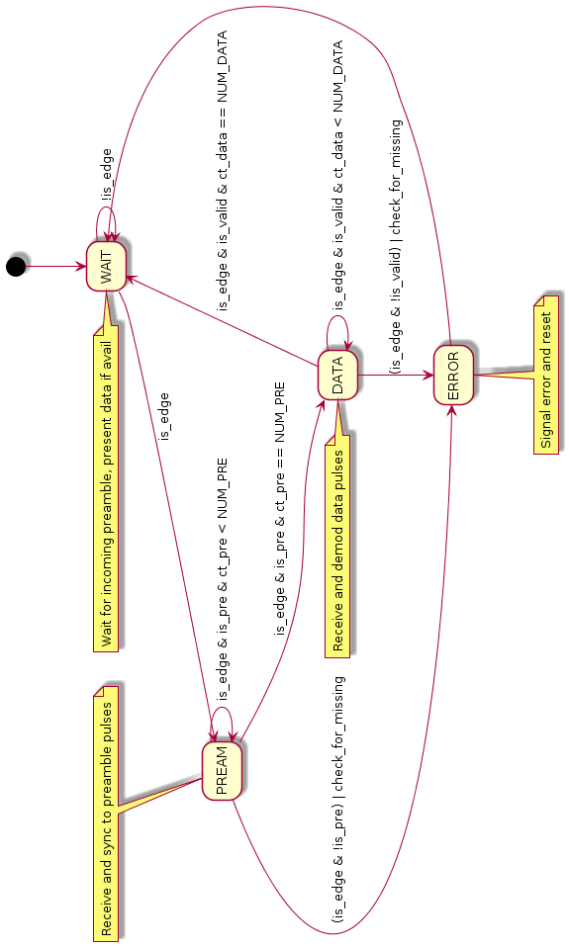

Fig. 9. Decoder FSM

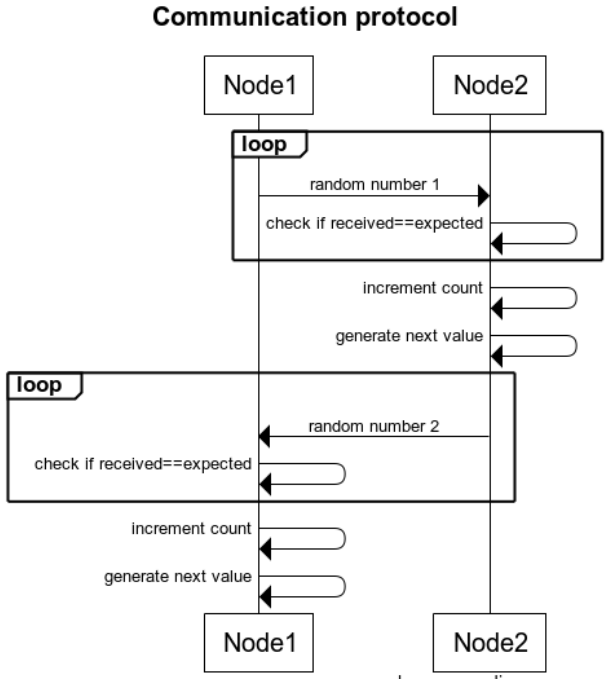

Fig. 10. The application layer communication protocol

V. Validation Plan

A. The Physical Layer

In the physical transmission of our data, our goal is to accurately transmit a sequence of bits as an analog signal between the two FPGAs. Thus it would be intuitive to perform a quantitative and qualitative comparison of the transmitted wave and the received wave. This includes factors such as the difference in amplitude, phase and waveform of the two signals. In addition, we also need to pay close attention to the rise time as a very high rise time can limit the speed of communication. The transmitter and receiver circuits can be individually tested by sending a simple square wave on one end and validating whether an FPGA can reproduce the square wave on the input GPIO pins using Signal Tap or visually verifying the square wave using an oscilloscope.

Our application is also a form of quantitative validation. We planned to code a short pseudo-random number generator on the source and destination FPGAs and initialize it with the same seed. This resulted in the sequence of numbers produced on both FPGAs being identical. We then send alternate numbers between the FPGAs and validate whether the number received from the circuit is the number expected from the sequence. The percentage of number of correctly sent bits from a source node to a destination node through the hub will give a quantitative measure of the correctness of our entire system.

In the following plots, the yellow graph depicts a signal we sent from one FPGAs GPIO pins and the green graph depicts the signal we received at the receiver FPGAs GPIO pins.

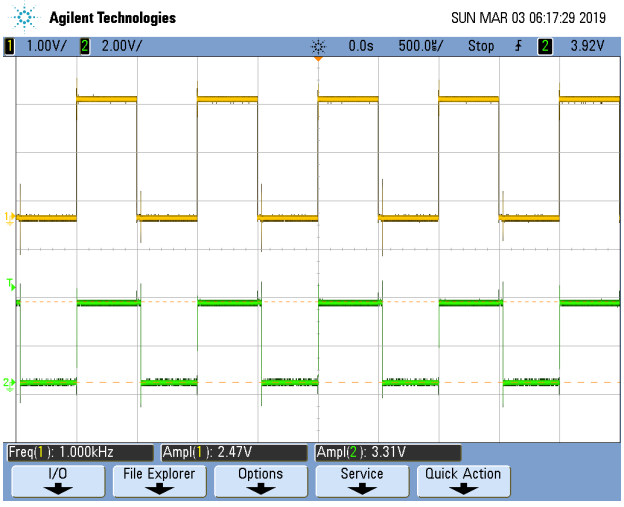

Fig. 11. The Transmitter FPGA's signal (yellow) and the Receiver FPGA's signal (green)

Additionally, we checked the rise time and phase difference in the two graphs at a 1kHz and measured it to be of the order of 3μs. This measurement was taken by recording the difference in time between the 0 voltage and the 90% amplitude voltage. Although we expect it to change as we operate at higher frequencies this proves that our validation procedure are sound.

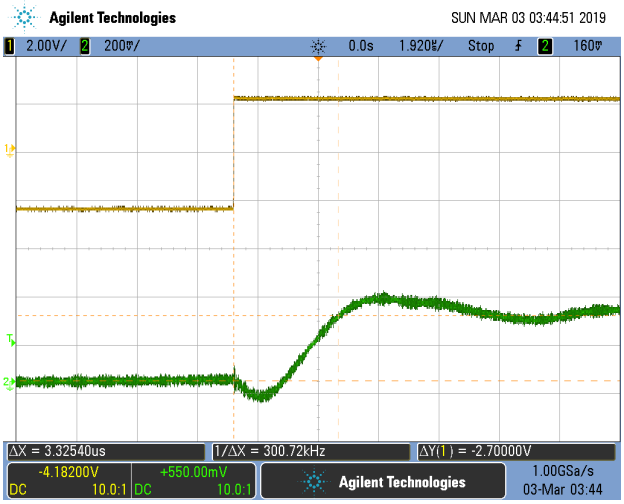

Fig. 12. Rise Time Measurement with a square wave input

B. The Data Link Layer

Using Synopsys VCS, we implemented testbenches for the modules on the data link layer. This include both manual inspection of waveforms and automated testing via SystemVerilog testbench constructs like assertions and properties. All modules were unit-tested before being composed together into larger integration tests.

The component modules of the Encoder and Decoder were tested in simulation without any dependence on the physical or application layers. These modules were then put together for integration testing. Absent any errors, we found encoded bitstreams to be perfectly decoded, and modulated signals to be perfectly demodulated. In other words, in an environment of simulated perfect transmissions, the data link layer should never signal errant packets or misinterpret pulses. Once correctness was established, resilience was tested.

For OPPM, we know that two rise edges with an offset greater than θ must be classified as coming from two different sources. Tests can be written to check for this. Although our project does not involve multiple transmitters to one receiver, we still had to ensure the offset handling worked as expected. We obtained perfect classification of rise edges separated by sufficiently-supported offsets.

As described in Section III-D, we performed integration testing with the physical and data link layers to determine the fastest throughput we could obtain. We performed subsequent tests with the application layer to ensure that all components of SkyFi were operational. We slowly incremented the speed until it was no longer reliable (below 99% accuracy), as determined via SignalTap probing. We instantiated internal counters to keep track of correct and incorrect packets received. With this testing methodology we were able to push beyond 1 Mbps.

C. The Application Layer

The first aspect to test involves making sure the application works as expected in simulation. On hardware, the first aspect to be tested is to make sure that the application works correctly with the encoder and decoder modules with the physical layer replaced by a wire. This allows us to make sure that the application is properly integrated with the data link layer. Only when this is done should the developer move on to integration with the physical layer. There are many sources of random errors that can be caused by the physical layer and debugging these without making sure the application is reliable is a difficult task.

A linear feedback shift register is used to generate psuedo random numbers which we use as data to send across the network. An LFSR is a shift register whose input bit is a linear function of the current value. The LFSR module we use generates the next value using the polynomial $x^{32} + x^{28} + 1$.

Using the LFSR allows us to get a good mix of psuedo random numbers which allows for better validation and testing on the number of packets being correctly sent and received. This module will need to be separately tested to be certain that both nodes indeed have the same RNG and are sending and receiving data in a synchronous manner.

VI. Project Management

A. Schedule

A detailed version of the schedule is attached as Figure 14.

B. Work Distribution

In general, we distributed the work by assigning each of us to a layer. Dhruva is primarily responsible for the Physical Layer, Raziq is responsible for the Data Link Layer, and Bhuvan is responsible for the Application Layer. As needed, we assisted each other with tasks.

C. Risk Management

We provisioned plenty of time in our project for unexpected hurdles in implementation and in the arrival of parts. We worked concurrently on our independent tasks so that no time was wasted and so our efforts could be parallelized. When we had to shift gears in our project, we were able to achieve our new goals, because many of our existing work could be used for the new objective. This allowed us to complete our project in time for the final presentation and public demo.

D. Bill Of Materials

Table II lists all the components the project utilized for the final demo:

TABLE II
Details of all components required

Component	Price	Quantity	Source
Terasic DE0-CV FPGAs	0	5	CMU Labs
IR LED: VSLY3943	$28.11	10	Mouser
Photodiode: QSD2030F	0	10	CMU Labs
Op-Amp: LM7171	$28.87	10	Digi-Key
Resistors	0	-	CMU Labs
Jumper Wires	0	-	CMU Labs
Breadboards	0	2	CMU Labs

VII. Related Work

As mentioned before, we were inspired by the work of the DarkLight team. We used their OPPM technique in a slightly different domain, optimizing for speed and accuracy instead of concurrency. There are also many other research papers describing various VLC techniques, which are summarized in [2].

VIII. Summary

We developed SkyFi as a near-range full-duplex VLC system that demonstrates many of the benefits of infrared and visible light as a fast, reliable, and robust transmission medium. We achieved a throughput of at least 1 Mbps, comparing favorably with standards like Bluetooth, NFC, and Zigbee. We also achieved high accuracy of at least 99%. While our distance was less than Bluetooth and Zigbee, we believe that we could get closer with more performant components. In addition, the modulation technique we used, OPPM, allows for concurrent reception of transmissions, something the aforementioned standards cannot do. This is another advantage that we would hope to exploit in future versions of SkyFi.

As a team, we learned a lot about practical engineering design with real scenarios in mind. We had to both come up with a comprehensive design and implement our vision. We also gained experience working with nascent technologies and with interfacing between the analog and digital world.

IX. Acknowledgements

We would like to thank our Professor, Dr. Tamal Mukherjee, our teaching assistant Shraddha Navalgund and the rest of the 18-500 course staff for their constant guidance and dedication to making this an interesting course and providing us with plenty of learning opportunities.

References

[1] Z. Tian, K. Wright, and X. Zhou, "The DarkLight Rises: Visible Light Communication in the Dark," in MobiCom '16 Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, pp. 2–15, October 2016.

[2] S. Rajagopal, R. D. Roberts and S. Lim, "IEEE 802.15.7 visible light communication: modulation schemes and dimming support," IEEE Communications Magazine, vol. 50, no. 3, pp. 72–82, March 2012.

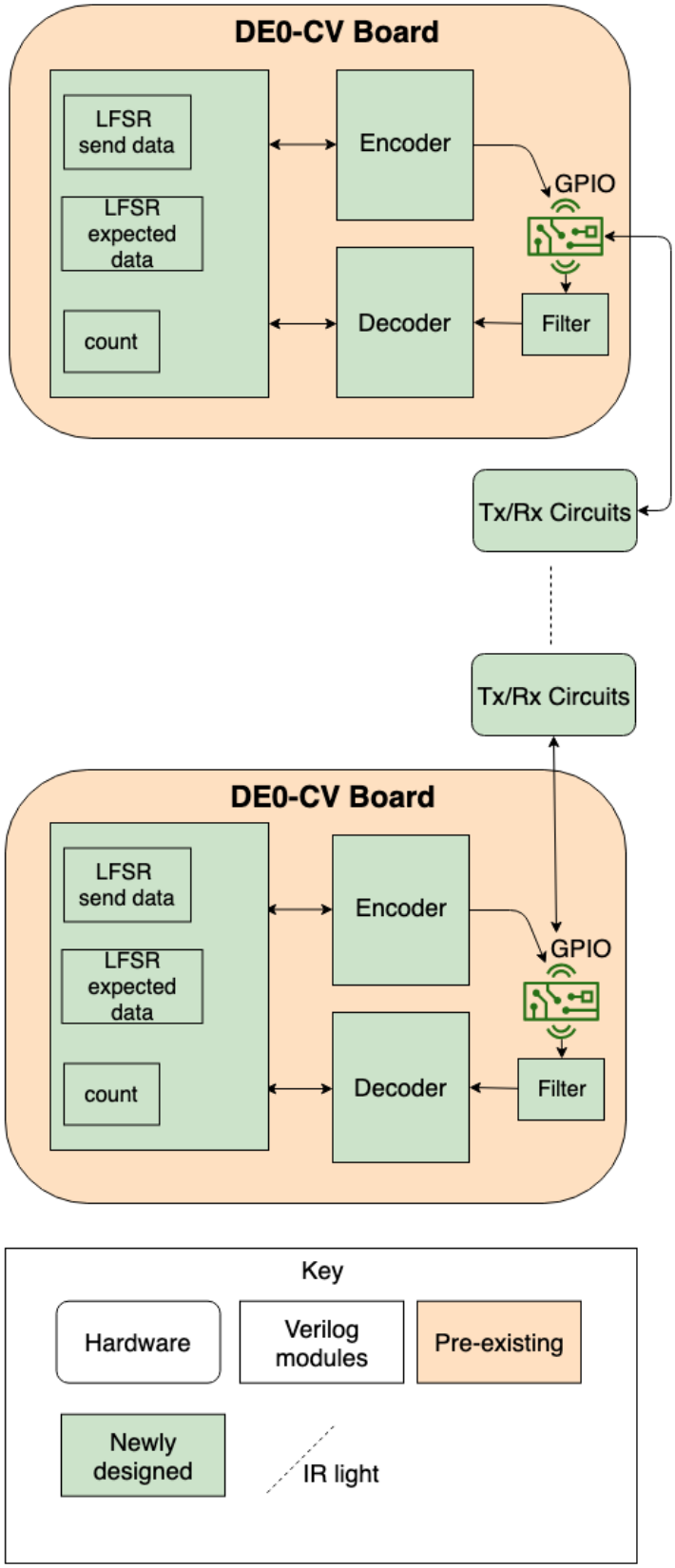

Fig. 13. Overall System Level Block Diagram

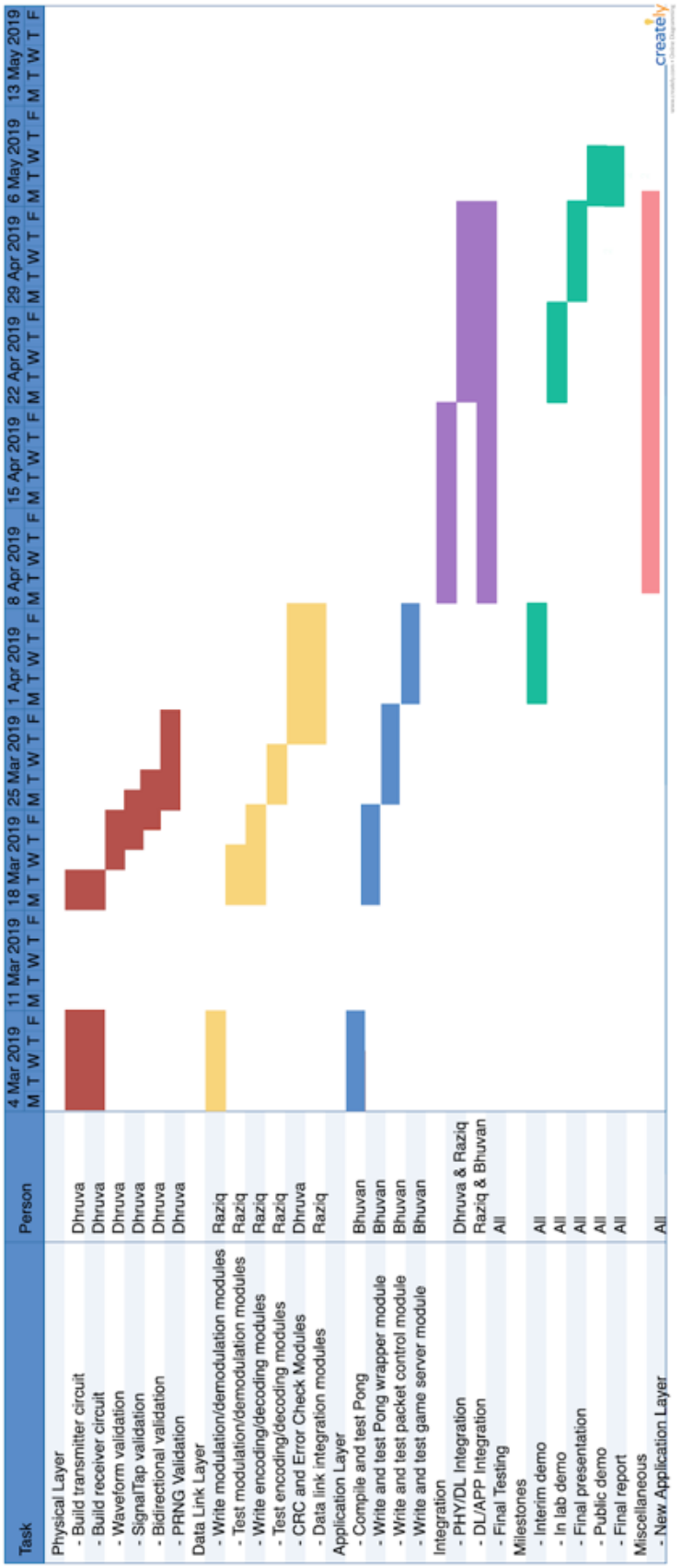

Fig. 14. Final Execution Plan