Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 6154495, 15 pages
https://doi.org/10.1155/2022/6154495

Research Article

A Real-Time Baseband Processor for Li-Fi Internet Access

Erwin Setiawan,[1] Trio Adiono,[1,2] Rahmat Mulyawan,[1,2] Nana Sutisna,[1,2] Infall Syafalni,[1,2] and Wasiu O. Popoola[3]

[1]*University Center of Excellence on Microelectronics, Bandung Institute of Technology, Indonesia*
[2]*School of Electrical Engineering and Informatics, Bandung Institute of Technology, Indonesia*
[3]*School of Engineering, The University of Edinburgh, UK*

Correspondence should be addressed to Rahmat Mulyawan; rahmat.mulyawan@itb.ac.id

Received 16 November 2021; Revised 15 August 2022; Accepted 21 September 2022; Published 8 November 2022

Academic Editor: Parameshachari B D

In the past years, light fidelity (Li-Fi) has been gaining popularity in the research. However, many researches have done only offline transmission with laboratory instruments. As a result, it is not practical to be used in commercial product, due to its cost and size. Therefore, the objective of this paper is to address problems that arise from the real-time implementation of Li-Fi system using the commercial off-the-shelf components. The implementation was developed on a low-cost system-on-chip (SoC) field-programmable gate array (FPGA). The implementation supports orthogonal frequency division multiplexing (OFDM) modulation including time synchronization. In order to build a system that is more practical to be used in commercial product, the network stack that supports TCP and ICMP was also developed. As a result, the user client can easily access the Internet using the available web browsers. The results showed that the system is functionally verified enabling it for real-time transmission to be used to access the Internet. According to the result, our baseband processor can transfer data with a maximum throughput of 1 Mbps at 125 Mhz of FPGA.

1. Introduction

Light fidelity (Li-Fi) is a new wireless communication technology that uses light as its medium. Instead of RF spectrum, Li-Fi uses the infrared and visible light spectrum as an attempt to address the RF spectrum congestion. Therefore, Li-Fi is often referred as visible light communication (VLC). Li-Fi is proposed as a 5th generation (5G) technology. It is a complementary technology to the RF technology [1].

Over the past years, there are a lot of researches demonstrating data transmission using visible light. However, the demonstrators are mostly based on offline signal processing [2, 3], i.e., they use an arbitrary waveform generator (AWG) at the transmit side and an oscilloscope at the receive side. Even though they achieved Gbps of data rate, there are still problems on how to implement that as consumer product.

In the context of developing Li-Fi device for consumer products, the Li-Fi should be able to do real-time transmission, as an example for high speed Internet access. There are several challenges that arise from real-time implementation of Li-Fi. One of the examples is the cost of the component. Expensive laboratory instruments or components will certainly not work. For example, laboratory instruments rely on GSPS ADC that may be too expensive for commercial products.

In this work, we propose a Li-Fi baseband processor. The processor was implemented on low-cost FPGA. The design can be easily retargeted to ASIC technology. We propose Li-Fi system architecture that uses HW/SW codesign methodology, in order to realize a complete network stack. Our proposed architecture has been tested for Internet access using the standard available web browsers. We also discuss several challenges that arise from real-time implementation of Li-Fi.

2. Related Works

There are several VLC demonstrators found in the literature. As the works in [2, 3], in which it was possible to achieve a

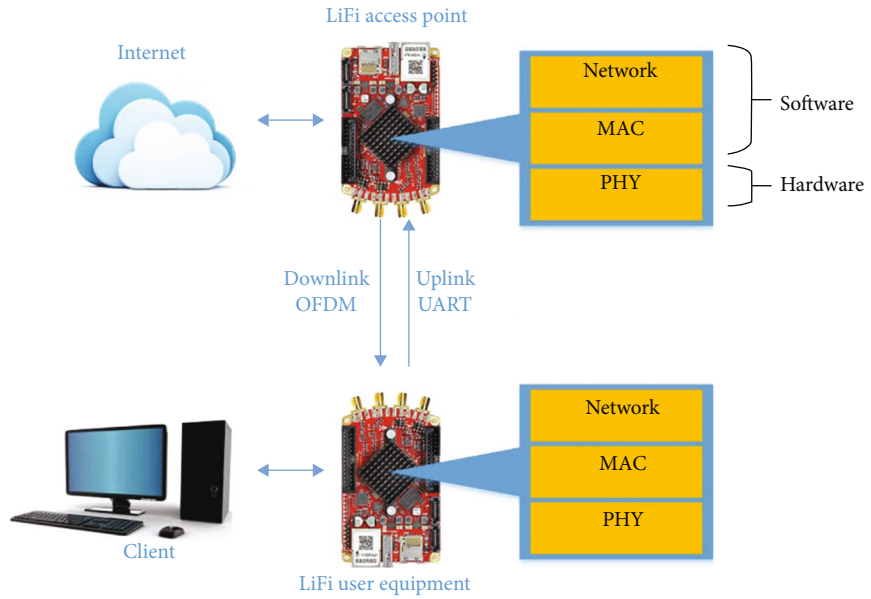

Figure 1: Proposed Li-Fi system for Internet access.

data rate of Gbps. However, an AWG and oscilloscope was used in the demonstrator.

In [2], a VLC demonstrator was developed and it achieved 11.28 Gbps of data rate using wavelength division multiplexing orthogonal frequency division multiplexing (WDM-OFDM) modulation, but it is based on offline signal processing. In [3], a VLC demonstrator that achieved 2 Gbps using OFDM was demonstrated, but it is also based on offline signal processing.

There are several VLC demonstrators that are based on real-time signal processing, i.e., they use FPGA or digital signal processor (DSP) at both transmit and receive side. In [4], a VLC demonstrator was developed and achieved 150 Mbps. The system relies on the Xilinx Virtex-6 FPGA. They use system generator to convert the design from high level programming language to the hardware description language (HDL). However, the network layer was not implemented yet. In [5], a real-time VLC demonstrator was developed. It achieved a data rate of 2.5 Gbps. However, it relies on the high-end and high cost instrument, and also the network layer was not implemented yet.

Even though the works in [4, 5] have demonstrated real-time transmission, their focus is on the PHY layer. There is still problem that needs to be addressed on the network layer. In work [6], a real-time VLC demonstrator was developed, and it can send real-time data through the channel, but it is limited to text data. In works [7, 8], they have demonstrated real-time VLC transmission that employed SoC FPGA, but the modulation is different. Both works use OOK modulation, but our works use OFDM.

3. System Design

3.1. Overview. The system block diagram of the Li-Fi system is shown in Figure 1. It consists of two devices for access point (AP) and user equipment (UE). The AP device is connected to the Internet, and the UE device is connected to the client. The downlink channel was designed using the OFDM modulation, and the uplink channel was designed using UART protocol. The UART protocol, which is not designed for Li-Fi, was used in this work for uplink because the main focus of this work is on the downlink using OFDM. Moreover, due to the limited resource element of the FPGA, it is not possible to incorporate both OFDM TX and RX designs into one FPGA. The problem regarding the modulation that is suitable for uplink still needs to be addressed in the future work.

3.2. System Architecture. Figure 2 shows the proposed SoC architecture for Li-Fi baseband processor. The architecture was implemented on the Red Pitaya board that uses Xilinx Zynq-7000 programmable SoC. It consists of processing system (PS) part and programmable logic (PL) or FPGA part. On the PS, the important component is the dual core ARM Cortex-A9 processor, in which the network layer was implemented as software. The Ethernet peripheral is also important because it connects the system to the Internet, in case of AP, and to the client, in case of UE. Ethernet was chosen in this architecture because it is widely used as an interface in computer network. The rest of the peripherals, such as USB, UART, and GPIO were used for debugging purposes.

On the PL, the PHY layer was implemented. The PHY layer is the baseband processor for OFDM and UART processing. The baseband processor was integrated to the main processor by using AXI4 bus. The AXI4 bus was chosen because it is an industry standard. It is a widely used bus for on-chip communication because it is based on the popular processor architecture, ARM.

3.3. Hardware Software Stack. Figure 3 shows the stack that was used to implement the complete network stack. On the

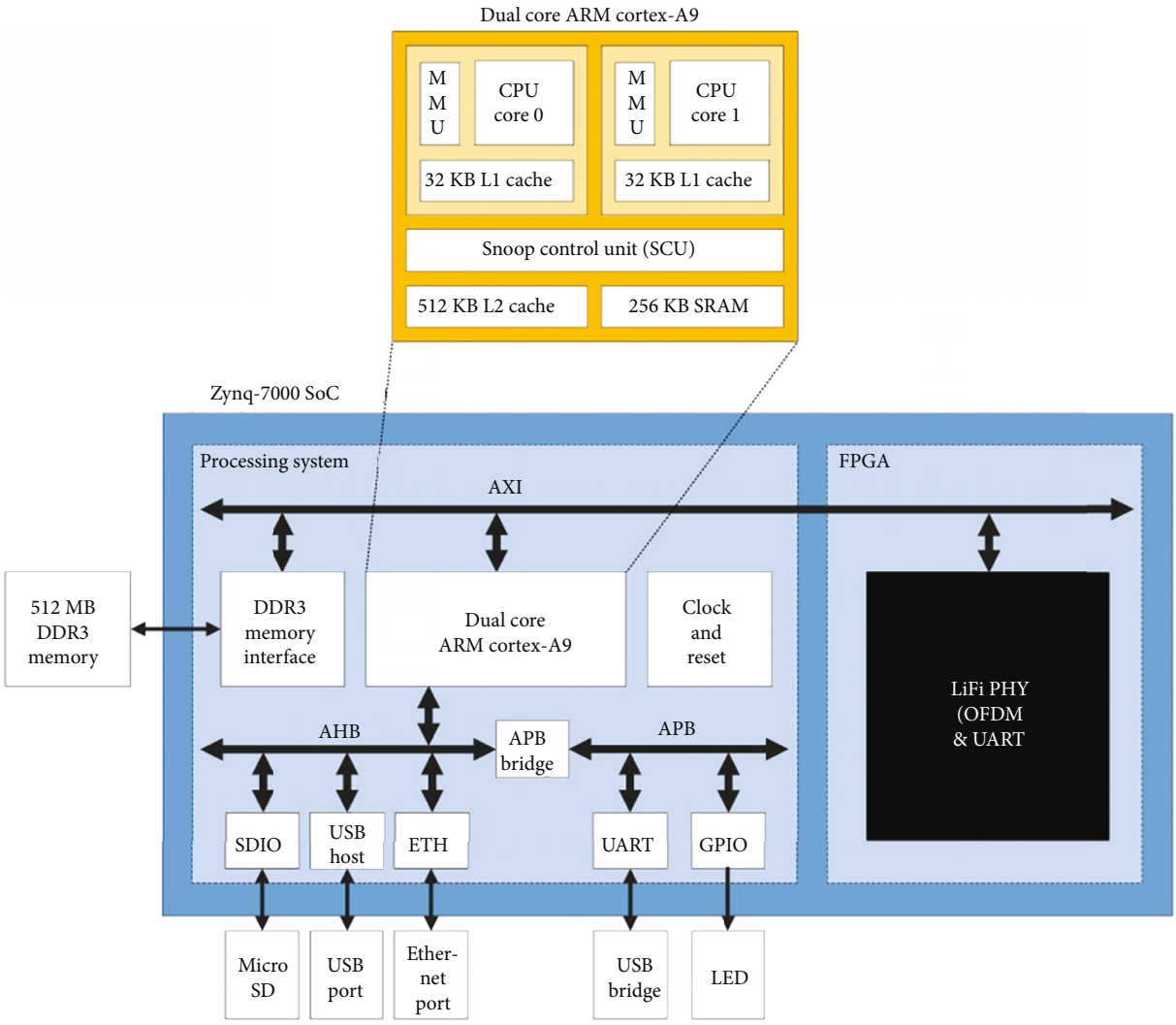

Figure 2: Proposed SoC architecture of the Li-Fi baseband processor.

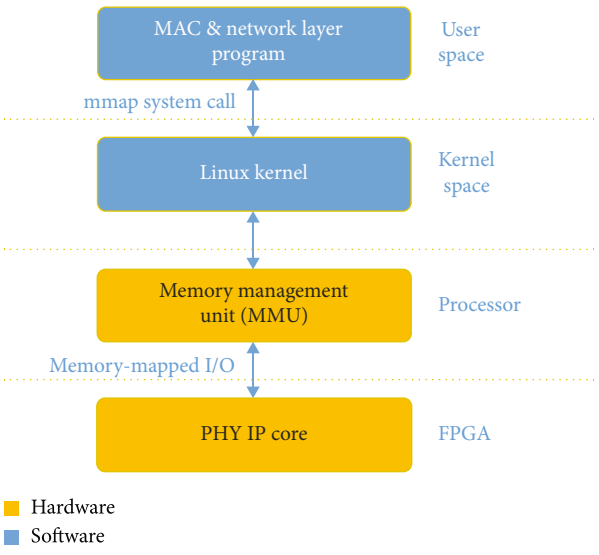

Figure 3: Hardware and software stack of the Li-Fi system.

Table 1: Red Pitaya Specifications.

Parameters	Value
Processor	Dual Core ARM cortex-A9
FPGA	Xilinx 28 nm Artix-7
Connectivity	Ethernet 1 Gbit, USB 2.0
DAC	14 bit, 125 MS/s
ADC	14 bit, 125 MS/s
Analog bandwidth	50 MHz

hardware part, the baseband processor is a memory-mapped component, so it has physical address space that can be accessed from the main processor. The memory management unit (MMU) was used because a Linux operating system (OS) was used in this system. The MMU maps the physical address of the baseband processor to the virtual address of the network layer program. Therefore, the baseband processor can be accessed from the network layer program.

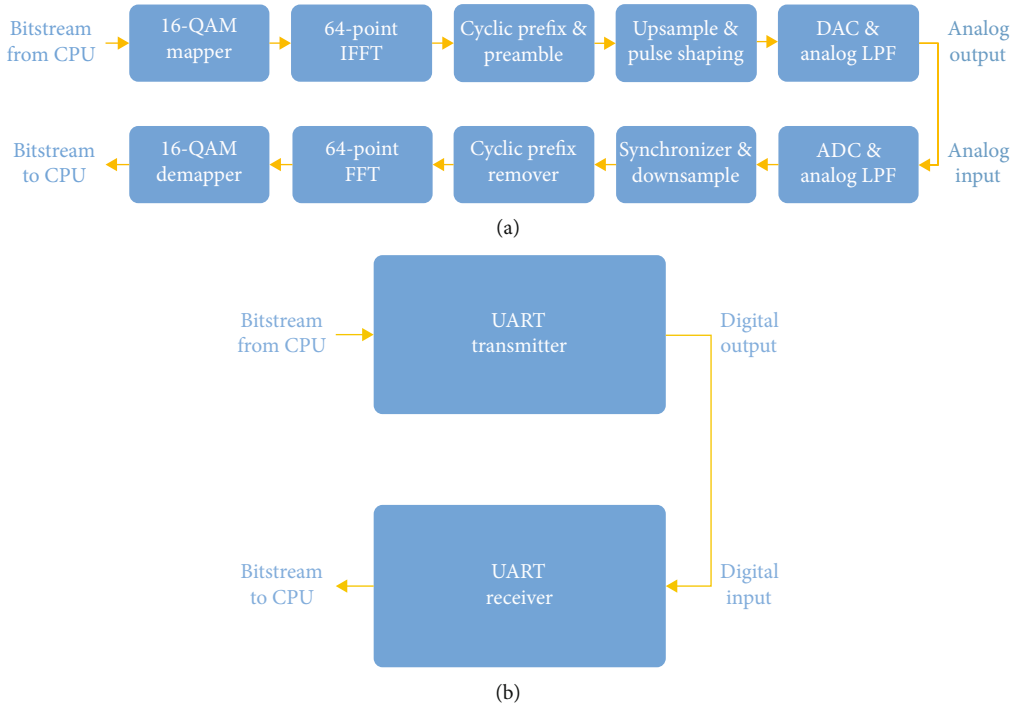

Figure 4: Proposed baseband processor block diagram: (a) downlink block diagram, (b) uplink block diagram.

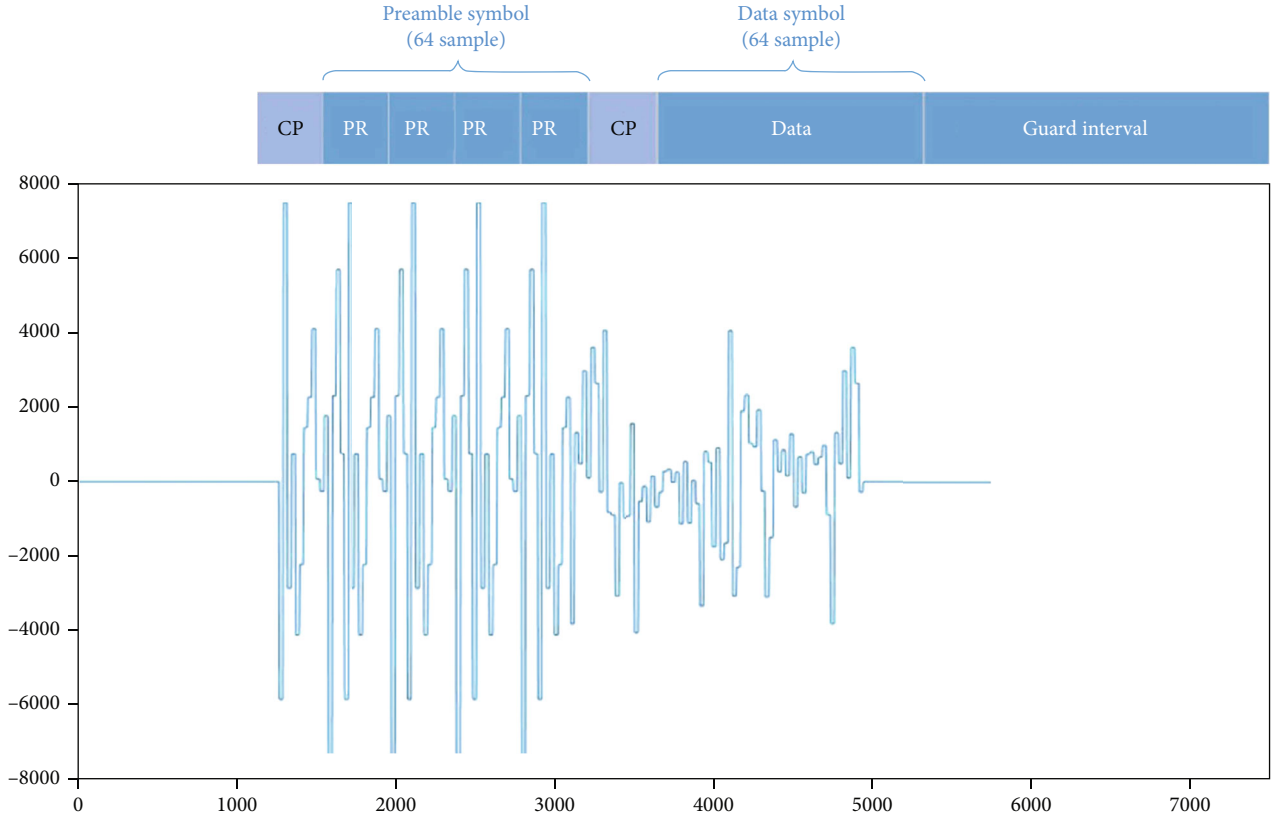

Figure 5: One OFDM frame.

3.4. Development Platform. The selection of development platform has an important factor. In this work, Red Pitaya [9] development board was used. This board was chosen because it has small form factor and relatively low cost. It also has main components required to implement the system, which are the main processor, FPGA, Ethernet, digital-to-analog converter

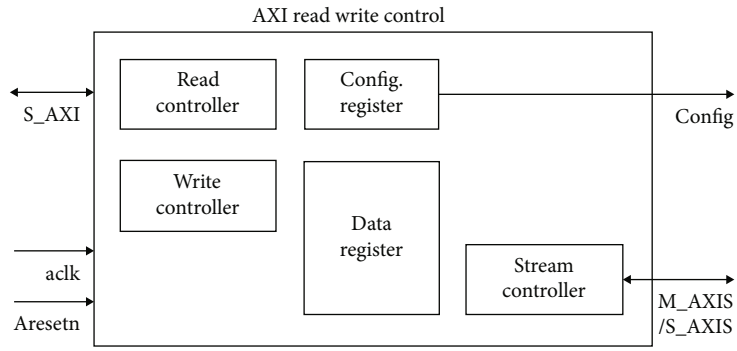

Figure 6: AXI read and write controller block.

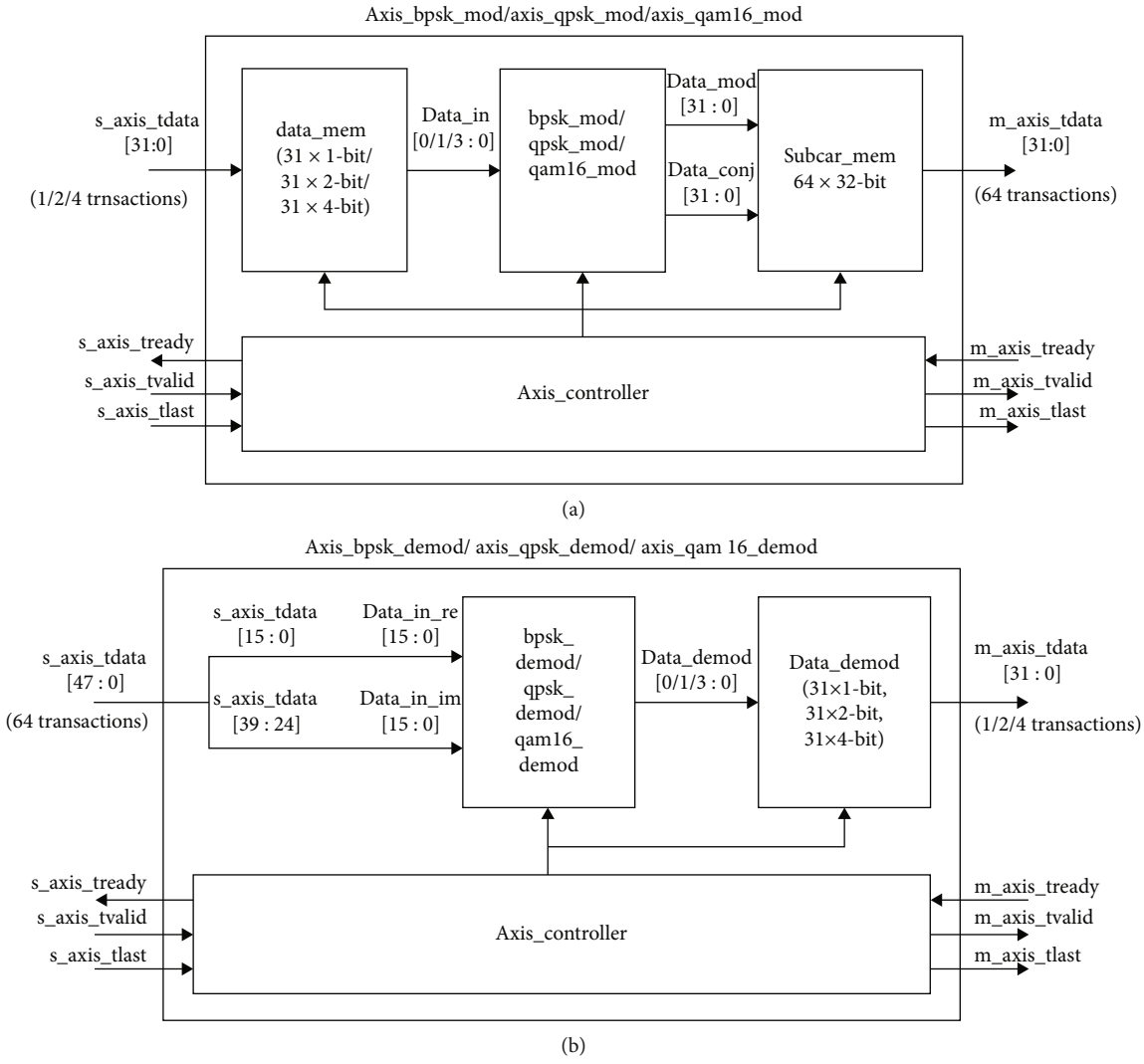

Figure 7: (a) QAM mapper block and (b) QAM demapper block.

(DAC), and analog-to-digital converter (ADC). The specifications of the board are shown in Table 1.

4. Baseband Processor Design

4.1. OFDM Model. In this work, the OFDM modulation was proposed for the downlink channel. Figure 4(a) shows the proposed OFDM block diagram. At the transmit side, the OFDM baseband model consists of several blocks: QAM mapper, IFFT, cyclic prefix and preamble, and upsample and pulse shaping. On the receive side, the OFDM baseband model consists of several blocks: synchronizer and downsample, cyclic prefix remover, FFT, and QAM demapper. The DAC and ADC convert the baseband signal from digital

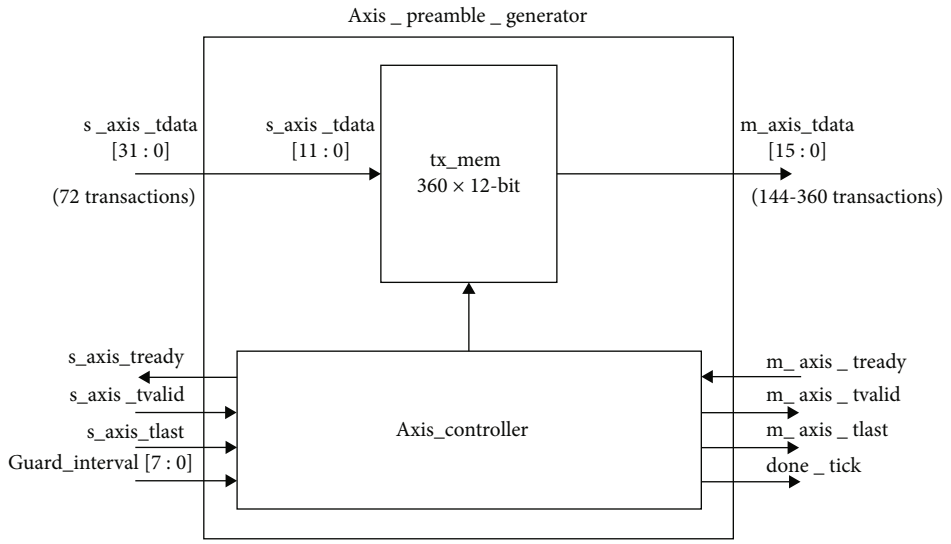

Figure 8: Preamble insertion block.

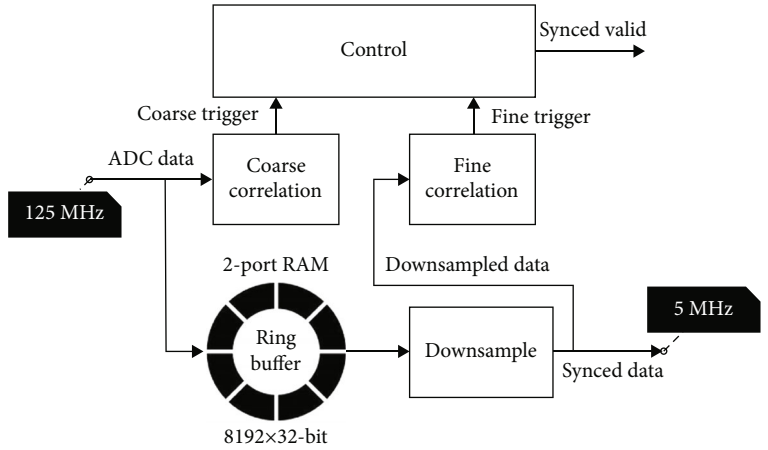

Figure 9: Synchronizer block.

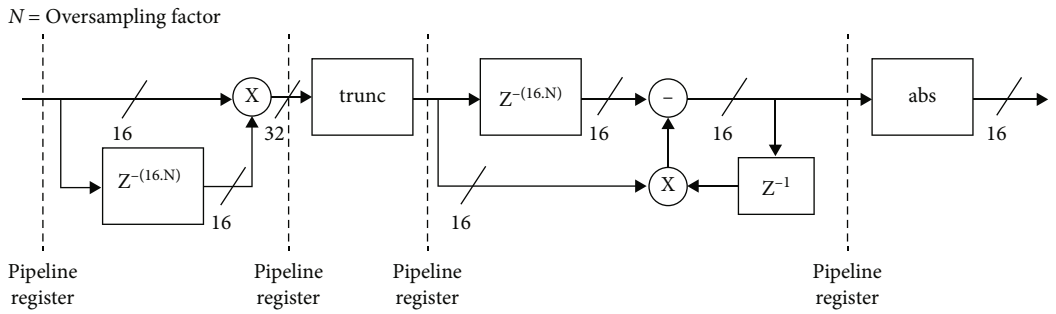

Figure 10: Autocorrelation block.

to analog and vice versa. For the uplink channel, the UART protocol block diagram is shown in Figure 4(b). It consists of UART transmitter and receiver.

4.1.1. QAM Mapper and Demapper. The function of mapper block is for mapping the source bit stream from the main processor to the M-QAM ($M = 2, 4, 16$) complex symbols. At the receiver, the demapper block reverses the mapper operation. The received complex symbols from FFT are demapped back into the bit stream and are sent to the main processor of the receiver. Let $X[n]$ be the complex symbols obtained after mapping process. It is an n-dimensional

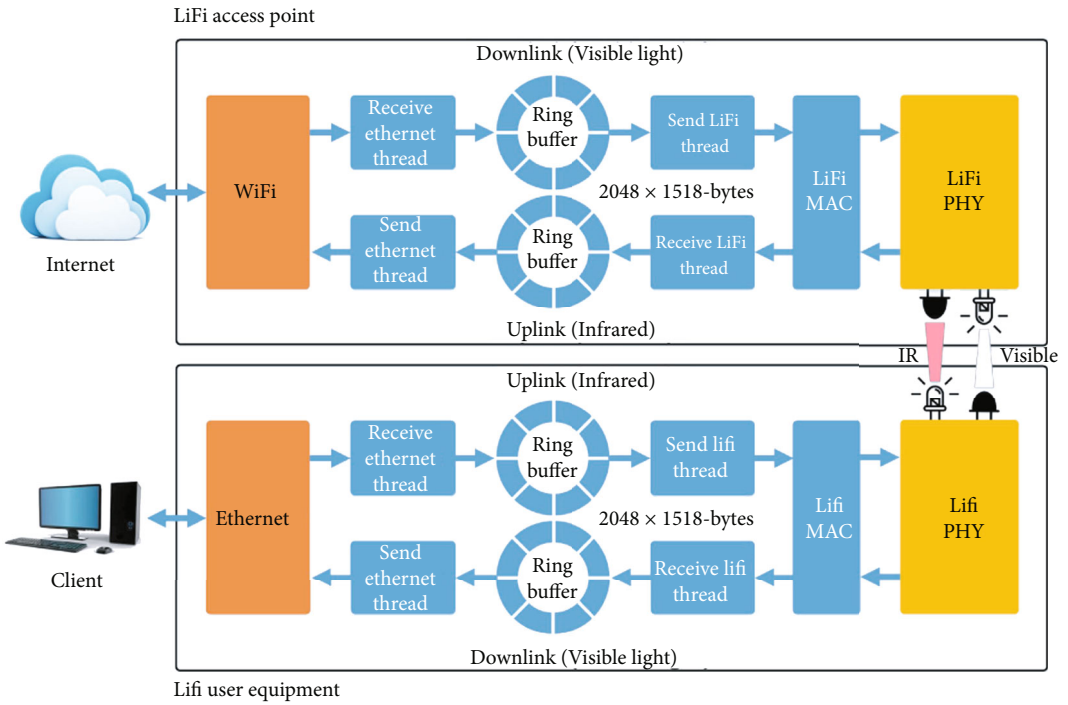

Figure 11: Software architecture of network layer program.

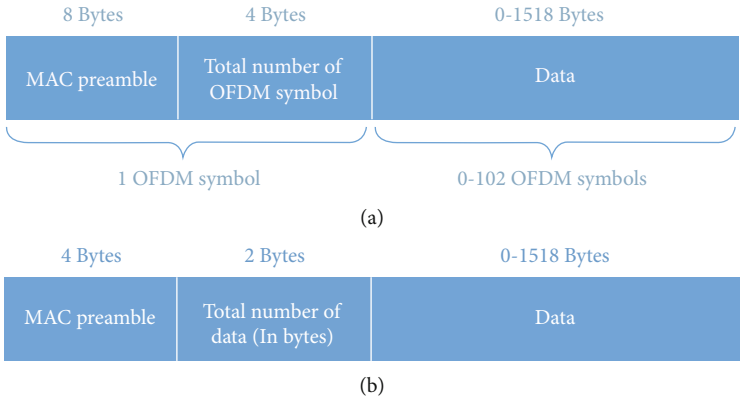

Figure 12: Li-Fi packets: (a) downlink packet and (b) uplink packet.

vector, where $n = 24,28,30$ corresponds to $M = 2, 4, 16$, respectively.

$$\boldsymbol{x} = [x_0, x_1, \cdots, x_{n-2}, x_{n-1}]. \quad (1)$$

4.1.2. IFFT and FFT. The complex symbols from QAM mapper are grouped into one OFDM symbol in frequency domain. Then, it is sent to the IFFT block. This block transforms one OFDM from frequency domain to time domain. A technique called Hermitian symmetry was employed to obtain a real valued signal in time domain [10]. Let X_{HM} be the complex symbols after Hermitian symmetry insertion [11]. Where N is defined as the IFFT/FFT size. This Hermitian symmetry forces the output of IFFT to be real valued.

$$X_{\mathrm{HM}}[k] = \left(X[0], \cdots, X\left[\frac{N}{2} - 1\right], X^*\left[\frac{N}{2} - 1\right], \cdots, X^*[0]\right). \quad (2)$$

The real valued time domain signal is given by [11]

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k] e^{j2\pi\, \mathrm{kn}/N}. \quad (3)$$

At the receiver, the FFT block reverses the IFFT operation. It transforms the received time domain signal back to

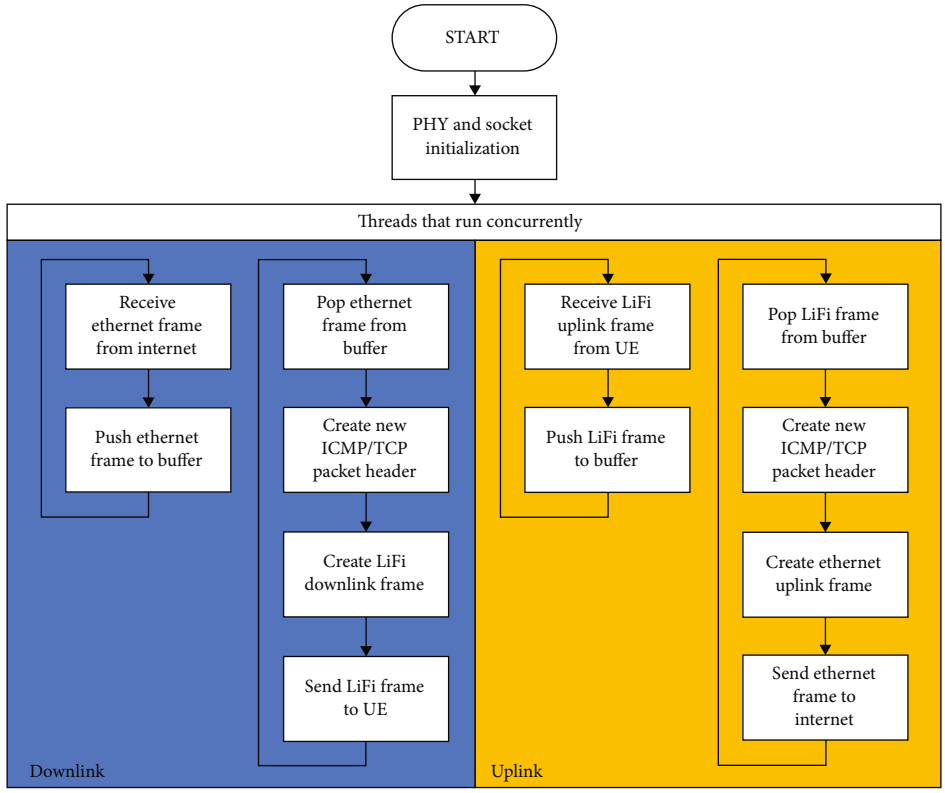

Figure 13: Flowchart of the network program for AP.

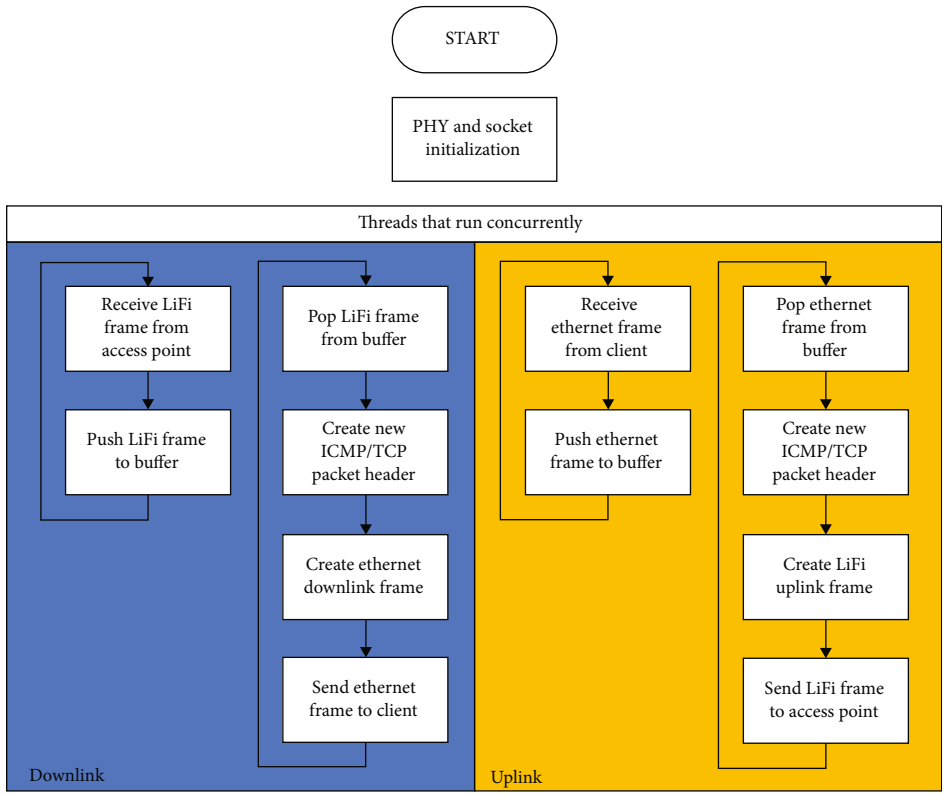

Figure 14: Flowchart of the network program for UE.

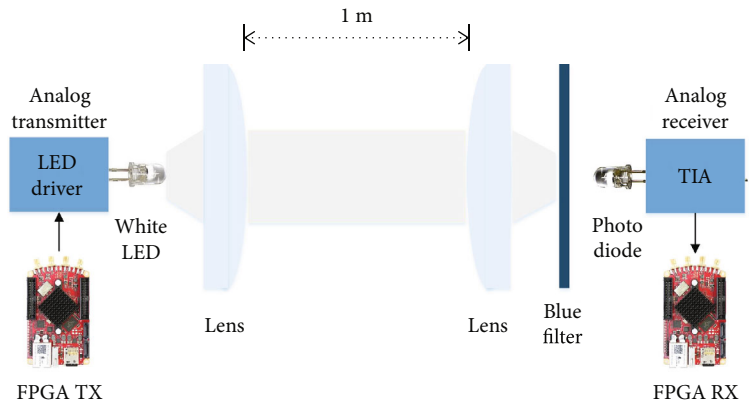

Figure 15: Analog and optical block diagram.

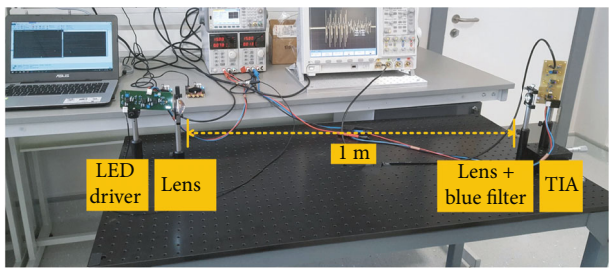

Figure 16: Photograph of analog and optical setup.

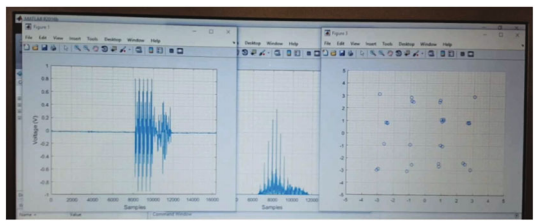

Figure 17: Photograph of real-time OFDM reception.

the frequency domain. Then, the obtained complex symbols are sent to the demapper block.

The transformation of the time domain signal to the frequency domain signal is given by [11]

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi \mathrm{kn}/N}. \quad (4)$$

4.1.3. Cyclic Prefix and Preamble Insertion. Cyclic prefix is inserted in front of every OFDM data symbol. It is a copy of several samples of the OFDM data symbol. Let x_{CP} be the time domain signal after the cyclic prefix is applied. Where L is defined as the length of cyclic prefix.

$$x_{\mathrm{CP}}[n] = (x[N-L+1], \cdots, x[N-1], x[0], \cdots, x[N-1]). \quad (5)$$

Table 2: AP and UE baseband processor's FPGA utilization.

Resource	Utilization (%)
AP baseband processor	
(i) LUT	32.57
(ii) LUTRAM	5.90
(iii) FF	32.63
(iv) BRAM	21.67
(v) DSP	18.75
UE baseband processor	
(i) LUT	26.77
(ii) LUTRAM	21.03
(iii) FF	16.85
(iv) BRAM	60.83
(v) DSP	36.25

Preamble is inserted in front of the data symbol after cyclic prefix x_{CP}. The time domain signal after preamble insertion is defined as

$$x_{\mathrm{PR+CP}} = (x_{\mathrm{PR}}, x_{\mathrm{CP}}). \quad (6)$$

One OFDM frame consists of preamble, data, and guard interval as shown in Figure 5. The preamble has 64 samples that are constructed from four-repeated sequence, in which each of the sequence has 16 samples.

4.1.4. Upsample and Pulse Shaping. One OFDM frame is sent to the upsample block. This block upsamples the OFDM samples with an oversampling factor of 25. Pulse shaping filter is employed to filter the upsampled OFDM frame before sending it through the DAC.

4.1.5. Synchronizer and Downsample. The main function of synchronizer is for detecting the start sample of the OFDM frame. The synchronizer module consists of coarse and fine correlation. The coarse correlation is applied using

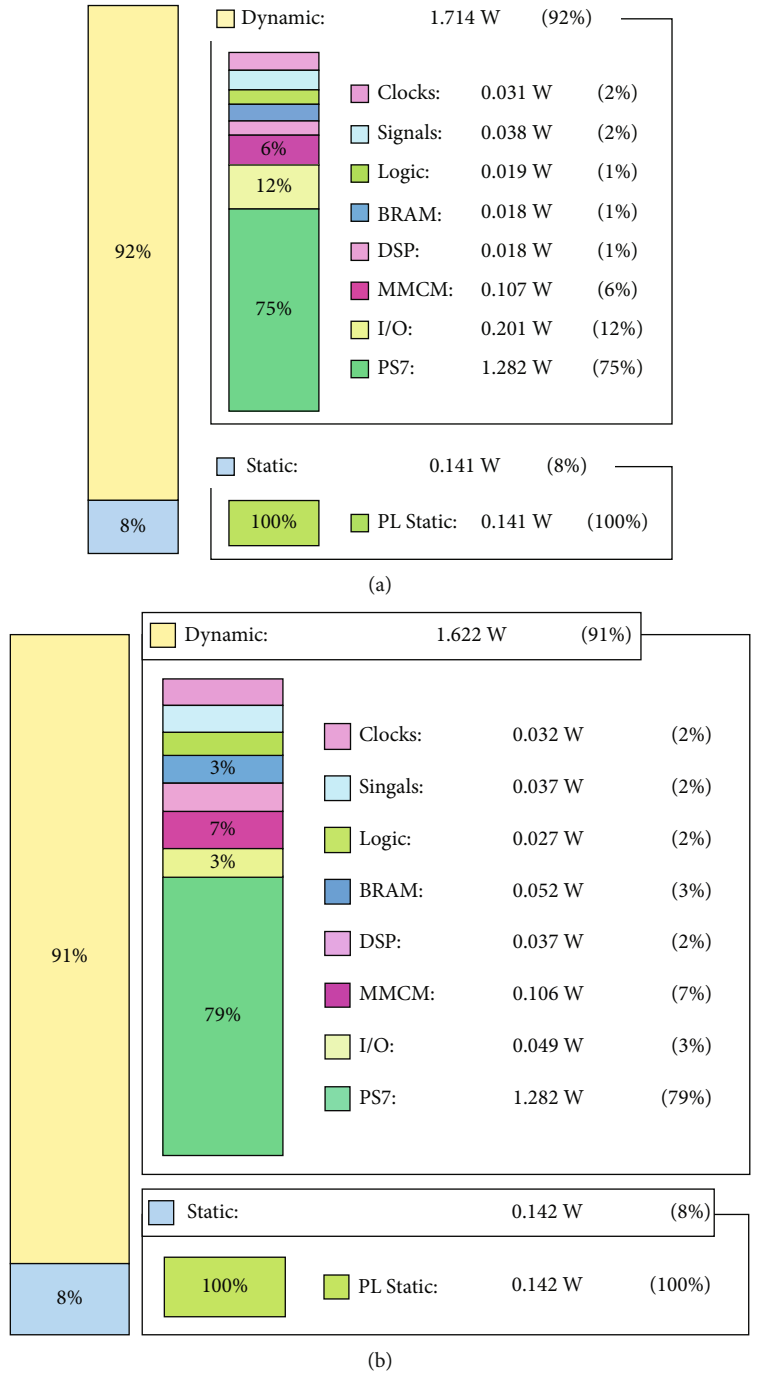

Figure 18: (a) AP and (b) UE on-chip power consumption.

autocorrelation [12]. The received signal and the delayed version of it is auto correlated which is defined as

$$Y_{\text{coarse}}[n] = \left| \sum_{r=0}^{R-1} y[n-r]y[n-r-L] \right|, \tag{7}$$

where R is the repetition interval in preamble and L is the delay length in samples. Both R and L in our system is 16. After the signal is downsampled, the fine correlation is applied using cross correlation. The received signal and the preamble that is stored in memory is cross correlated which is defined as

$$Y_{\text{fine}}[n] = \sum_{r=0}^{R-1} y_{\text{down}}[n-r]y_{\text{rom}}[r]. \tag{8}$$

4.2. RTL Implementation. The register transfer level (RTL) of the baseband processor was implemented using Verilog HDL.

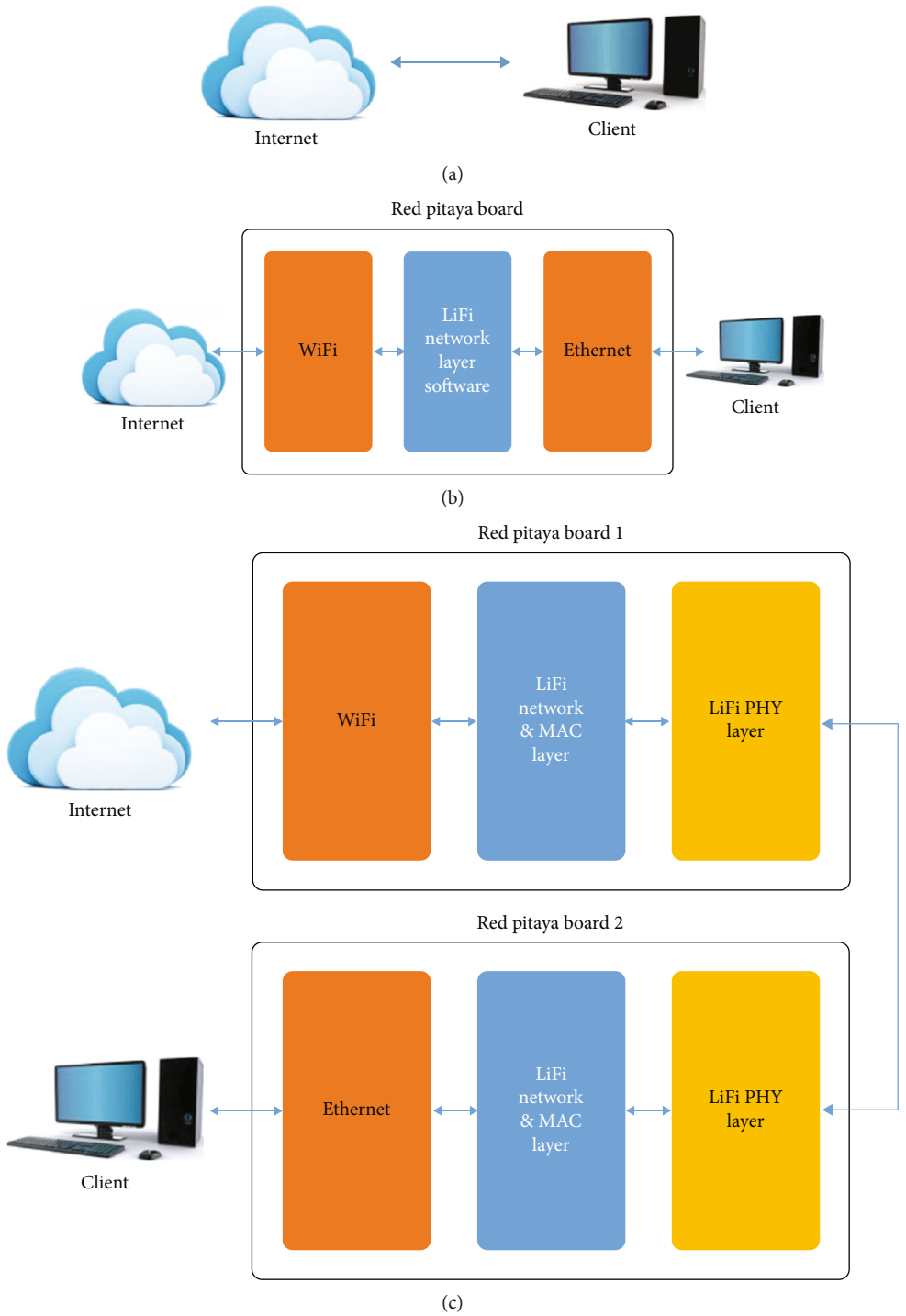

Figure 19: Experiment setup 1 (a), 2 (b), and 3 (c).

4.2.1. AXI Read and Write. This RTL block was designed as interfaces between the main processor and PHY block. The block diagram of this block is shown in Figure 6. It implements the AXI4-lite and AXI4-stream protocol. It converts the memory-mapped data to the stream data. The slave port S_AXI is connected to the main processor through the master port of the AXI interconnect. Then, the M_AXIS or S_AXIS is connected to PHY block. The data are temporarily stored in the data register.

4.2.2. Mapper and Demapper. Figure 7(a) shows the block diagram of the mapper. It receives stream data from the AXI read and write controller. Firstly, the data are stored in the data_mem buffer. Secondly, the data are mapped into complex symbols. It is implemented using a look-up table (LUT). Finally, the mapped data are stored in the subcar_mem buffer. The controller arranges the mapped data into one OFDM subcarrier, in which the Hermitian data are included.

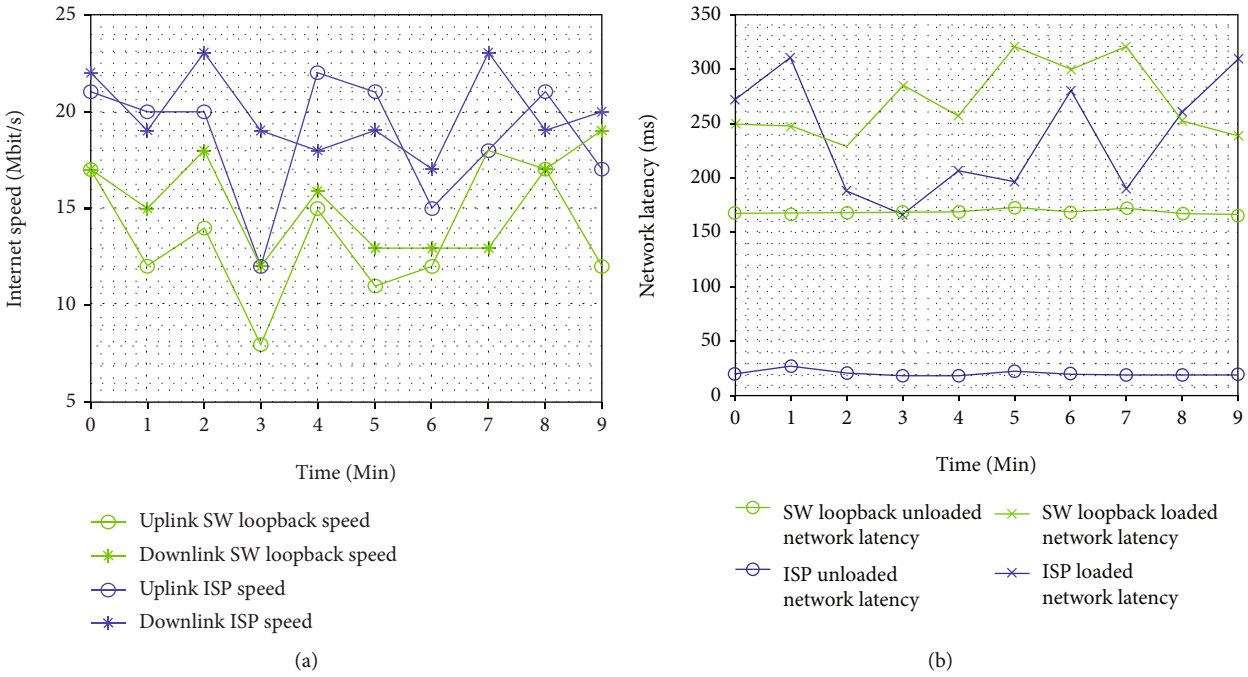

Figure 20: Internet speed (a) and latency (b) comparison between experiment setup 1 and 2.

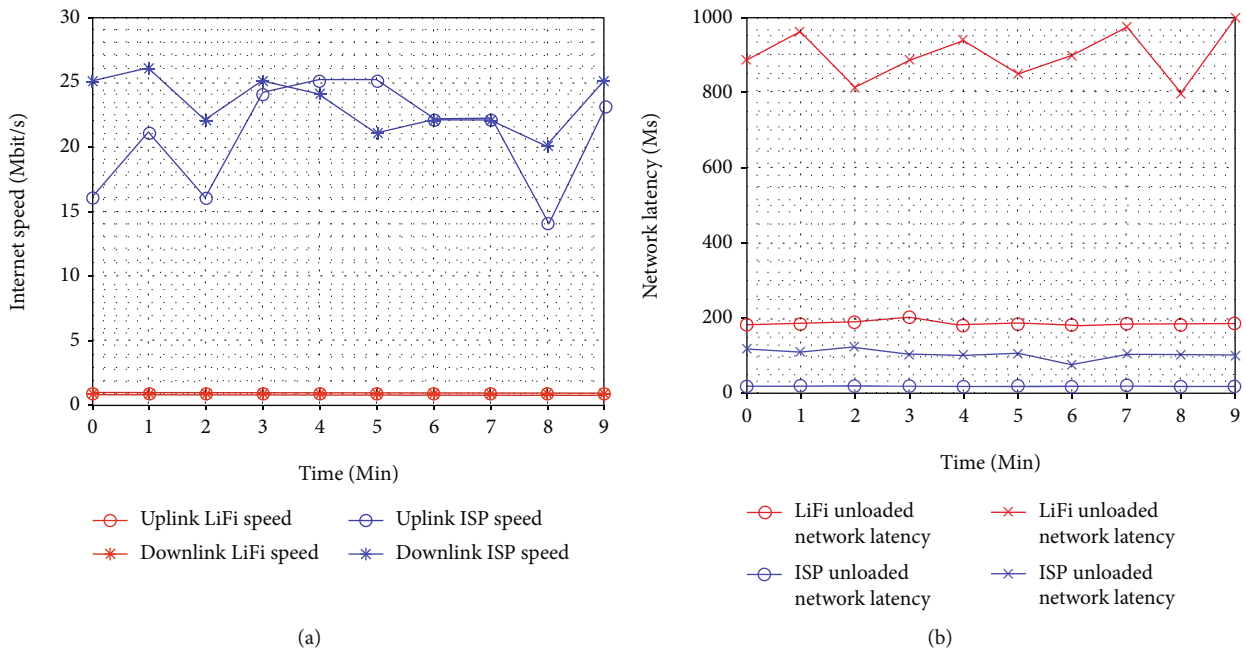

Figure 21: Internet speed (a) and latency (b) comparison between experiment setup 1 and 3.

Figure 7(b) shows the block diagram of the demapper. Firstly, it receives data from FFT. Secondly, the received data are demapped. Finally, the demapped data are stored in data_demod buffer.

4.2.3. Preamble Insertion. Figure 8 shows the block diagram of preamble insertion. It receives the time domain data from IFFT. Then, the received data are stored in tx_mem buffer. The received data are concatenated with the preamble samples.

4.2.4. Synchronizer. Figure 9 shows the block diagram of the synchronizer. The coarse correlation block is the implementation of Equation (7) and the fine correlation block is the implementation of Equation (8). Firstly, the data input comes from the ADC at rate of 125 MHz. Then, the data are stored inside the 2-port RAM, and at the same time, coarse correlation is calculated using coarse correlation block. At the end peak of coarse correlation output, a trigger signal is generated to start the the controller. After coarse trigger is detected, the data are read from the 2-port RAM. Then, the data enter the

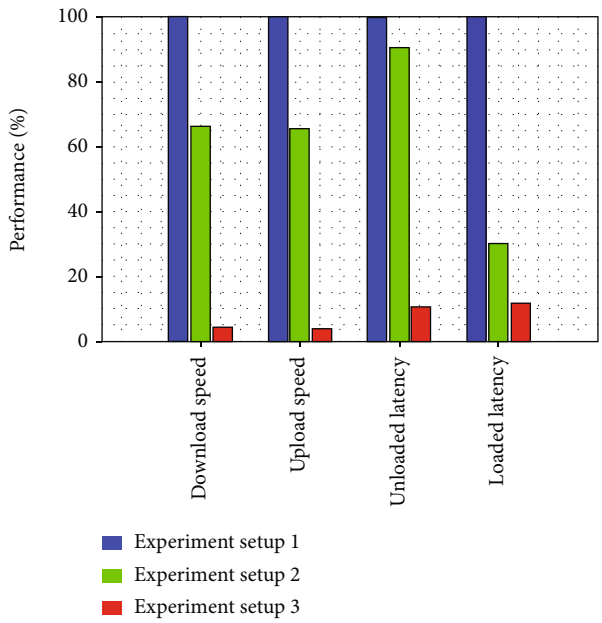

Figure 22: Normalized performance of all experiment setups.

downsample block with a factor of 25. Finally, fine correlation is calculated on this downsampled data in order to get the fine trigger. Then, the controller sends valid signal indicating the start of OFDM data symbol.

Figure 10 shows the block diagram of the autocorrelation datapath. This is the implementation of autocorrelation in recursive form [12]. To reduce size of the circuit, we use 16-bit fixed-point operation. We have improved the circuit by applying 4 stage pipeline register in order to increase the throughput. The throughput is improved by a factor of three [13].

5. Network Layer Design

5.1. Software Architecture. The software architecture of the network layer program is shown in Figure 11. Network layer was implemented as a program that runs on the ARM processor of Zynq SoC. Specifically, it is a user space program that forwards data from baseband processor (LiFi PHY) to the Ethernet/WLAN interface, and vice versa. The program was designed to be a multithreaded program. So, the program is capable to process more than one task in parallel. The ring buffer was used to hold the Ethernet packet and also it was used as a transfer mechanism between threads.

On the AP side, there are four threads and two ring buffers. The first two threads and one ring buffer was used to read the downlink Ethernet packets from the Internet, convert them to the Li-Fi packets, and send them to the Li-Fi PHY layer. The last two threads and the last ring buffer was used to read the uplink packets from the Li-Fi PHY layer, convert them to Ethernet packets, and send them to the Internet. The Li-Fi MAC layer was designed for packet conversion from the Ethernet packets to the Li-Fi packets.

On the UE side, there are also four threads and two ring buffers. The first two threads and one ring buffer was used to receive the uplink Ethernet packets from client, convert them to Li-Fi packets, and send them to the Li-Fi PHY layer. The last two threads and the last ring buffer was used to read the downlink packets from the Li-Fi PHY layer, convert them to Ethernet packets, and send them to the client.

5.2. Packets Format. Figure 12(a) shows the packet format for the Li-Fi downlink MAC. It starts with 8 bytes of MAC preamble. It is used to identify the start of each MAC packet. Then, the 4 bytes after preamble is total number of OFDM symbol (0-102) corresponds to the data size (0-1518 bytes). Figure 12(b) shows the packet format for the Li-Fi uplink MAC. It starts with 4 bytes of MAC preamble and 2 bytes contain total number of data size (0-1518 bytes).

5.3. Network Layer Flowcharts. Figure 13 shows the flowchart of the AP. The PHY layer and the socket for Ethernet are initialized. There are four tasks that run in parallel to process the downlink and uplink frames. The first thread receives an Ethernet frame from the Internet and push it to the first ring buffer. The second thread pops an Ethernet frame from the same ring buffer. Then, the Ethernet frame is converted to the Li-Fi downlink frame. Finally, the Li-Fi downlink frame is sent to the Li-Fi PHY layer. The third thread receives a Li-Fi uplink frame from the UE and pushes it to the second ring buffer. The last thread pops a Li-Fi uplink frame from the same ring buffer. Then, it is converted to the Ethernet frame. Finally, it is sent to the Internet as an Ethernet uplink frame.

Figure 14 shows the flowchart of the UE. The flow is similar to the flow of AP. The PHY layer and the socket for Ethernet are initialized. There are four tasks that run in parallel to process the downlink and uplink frames. The first thread receives a Li-Fi frame from the AP and pushes it to the first ring buffer. The second thread pops a Li-Fi frame from the same ring buffer. Then, the Li-Fi frame is converted to the Ethernet downlink frame. Finally, the Ethernet downlink frame is sent to the client. The third thread receives an Ethernet uplink frame from the client and pushes it to the second ring buffer. The last thread pops an Ethernet uplink frame from the same ring buffer. Then, it is converted to the Li-Fi frame. Finally, it is sent to the AP as a Li-Fi uplink frame.

6. Integration with Analog and Optical Front-End

Figure 15 shows the analog and optical block diagram. On the transmitter side, the OFDM baseband signal from FPGA is sent to the LED driver circuit. The circuit adds DC bias to the signal, so that the LED operates in the linear region. On the receiver side, the light passes through the lens in order to focus the incoming light. Then, the light passes through the blue filter to filter out the yellow component of the white light. After that, the received signal is sent to the transimpedance amplifier (TIA) circuit. The circuit removes the DC bias and amplifies the signal. Finally, the signal is sent to the FPGA.

The details of the integration and experiment have been published in [14]. Figure 16 shows the photograph of experiment. The distance between TX and RX is 1 m. The 3 dB

Table 3: Comparison with other works.

Reference	Modulation	Data rate	Baseband processing	Network access	Estimated cost
[2]	WDM-OFDM	11.28 Gbps	Offline	No	—
[3]	OFDM	2 Gbps	Offline	No	—
[4]	OFDM	150 mbps	Real-time	No	Us$12919
[5]	OFDM	2.5 Gbps	Real-time	No	—
[6]	OFDM	26.8 kbps	Real-time	No	Us$60
[7]	OOK	0.87 mbps	Real-time	Yes	Us$498
[8]	OOK	0.5 mbps	Real-time	Yes	Us$498
Our work	OFDM	1 mbps	Real-time	Yes	Us$762

bandwidth of this VLC link is 4 MHz. Figure 17 shows the photograph of real-time OFDM reception using MATLAB. It shows the time domain OFDM signal, correlation, and QAM16 constellation.

7. Result

7.1. FPGA Synthesis. The baseband processor design was synthesized with the Xilinx Vivado [15] tools. The Zynq-7000 xc7z010clg400-1 was chosen as the target FPGA. Table 2 shows the utilization of the AP and UE baseband processor. From the synthesis result, we obtain the maximum working frequency of the baseband processor. For AP baseband processor, the maximum working frequency is 137 MHz, and for UE is 134 MHz. Therefore, both of the AP and UE's working frequency meets our requirement, which is 125 MHz. Figures 18(a) and 18(b) show the on-chip power consumption of the AP and UE baseband processor.

7.2. Internet Access Performance. In this work, three experiment setups were evaluated. The first experiment setup was carried out to measure the speed of the Internet connection without our Li-Fi device. This setup is shown in Figure 19(a). The second experiment setup was carried out to measure the speed of the Internet connection with only the network layer of Li-Fi device. This setup is shown in Figure 19(b). The third experiment setup was carried out to measure the speed of the Internet connection with our network and PHY layer of Li-Fi device. This setup is shown in Figure 19(c).

Figure 20(a) shows the comparison of Internet speed between experiment setup 1 and 2. The Internet speed of experiment 2 was lower than experiment 1. This is due to the fact that network layer program was implemented in the user space instead of kernel space. For future improvement, the network layer program should be implemented on the kernel, so it will get highest scheduling priority.

Figure 20(b) shows the comparison of the network latency between experiment setup 1 and 2. The unloaded latency and loaded latency was measured. The unloaded latency was measured as the time required for the round trip of requests to the server of the speed test website when there is no other traffics on the device's network. The loaded latency was measured as the time required for the round trip of requests to the server of the speed test website when there is heavy traffics on the device's network.

Figure 21(a) shows the comparison of Internet speed between experiment setup 1 and 3. The measured Internet speed of experiment 3 was around 1 Mbit/s. This is due to the processing time of the PHY layer and bandwidth limitation of analog and optical front-end. Figure 21(b) shows the comparison of the network latency between experiment setup 1 and 3.

Figure 22 shows the normalized performance from the experiments. The download speed of the experiment setup 2 was 30% slower than the experiment setup 1. This is due to the network stack program was implemented in user space instead of the kernel space. Other than that, we have not employed DMA transfer and interrupt as the typical network interface card (NIC) uses it.

7.3. Comparison with Other Works. Table 3 compares this work with other works. The related works presented in [2, 3] propose offline VLC transmission using OFDM modulation. Compared to [2, 3], in terms of data rate, our work is much slower, but our proposed architecture works real-time. The related works presented in [4, 5] propose real-time VLC transmission. However, they have not implemented the network stack yet. Compared to [4, 5], which are also real-time processing, in our proposed architecture, the network stack has been implemented. As a result, our proposed architecture can access the network/Internet.

Compared to [6], our work has faster data rate and can access Internet. This is because our work use FPGA, while [6] use microcontroller. Compared to [7, 8], which are also real-time processing and have Internet access, our work use the OFDM modulation and has a better data rate.

We also compare our work with others in terms of cost of the TX and RX board. The related works presented in [2, 3], and [5] use AWG and oscilloscope, therefore we cannot compare them with our work because the cost difference would be huge and unfair. Only works [4, 6, 7], and [8] use development board as its TX and RX.

Work [4] uses a high-end FPGA board Virtex-6 FPGA ML605 evaluation kit, so the cost is high. Work [6] uses a microcontroller board STM32F4 discovery, which is not suitable for high-speed signal processing, so the cost is very low.

Work [7, 8] use the same FPGA board Zynq-7000 ZYBO development board. This board is similar to the FPGA board

we use (Red Pitaya). They use the same FPGA SoC, the Xilinx Z-7010 chip, so they have same performance. The main difference is that on the ZYBO board there is no high speed DAC and ADC. Our work requires DAC and ADC because the modulation we use is OFDM, so we choose the Red Pitaya board, which has a cost greater than the ZYBO board.

8. Conclusion

In this paper, we discuss challenges that arise from real-time implementation of baseband processor for Li-Fi system. We apply our proposed implementation methodologies in order to build a prototype for real-time Internet access using low-cost FPGA. The system works at 125 MHz of clock frequency. The experiment was done using analog and optical front-end. All layers are successfully implemented using SoC FPGA. The achieved throughput is still limited to 1 Mbps because of optical bandwidth limitation and the implementation of network stack that not yet at kernel level.

For future improvement, we can apply preequalization circuit to the LED driver to increase the optical bandwidth. Also, we can implement the driver in kernel module with DMA and interrupt, so that it can maximize the data transfer between hardware and software. Furthermore, we can reuse the Linux TCP/IP stack instead of using our own stack.

Data Availability

Data is available on request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is funded by the Ministry of Education, Culture, Research, and Technology (Kemendikbudristek) of the Republic of Indonesia.

References

[1] H. Haas, "LiFi is a paradigm-shifting 5G technology," *Reviews in Physics*, vol. 3, pp. 26–31, 2018.

[2] G. Faulkner, D. Tsonev, E. Xie et al., "Led based wavelength division multiplexed 10 gb/s visible light communications," *Journal of Lightwave Technology*, vol. 34, pp. 3047–3052, 2016.

[3] Z. Wang, J. Shi, Y. Wang et al., "2.0-gb/s visible light link based on adaptive bit allocation OFDM of a single phosphorescent white led," *IEEE Photonics Journal*, vol. 7, no. 5, pp. 1–8, 2015.

[4] C. Ribeiro, M. Figueiredo, and L. N. Alves, "Live demonstration: 150mbps+ DCO-OFDM VLC," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 457, Montreal, QC, Canada, May 2016.

[5] Y. Ha, S. Han, C. Wang, G. Li, and N. Chi, "A 2.5 gb/s real-time visible-light communication system based on phosphorescent white led," in *2019 7th International Conference on Information, Communication and Networks (ICICN)*, pp. 140–145, Macao, 2019.

[6] R. V. W. Putra, W. A. Cahyadi, T. Adiono, A. Pradana, and Y. H. Chung, "Physical layer design with analog front end for bidirectional DCO-OFDM visible light communications," *Optik*, vol. 138, pp. 103–118, 2017.

[7] S. Fuada, T. Adiono, and R. A. Saputro, "Rapid development of system-on-chip (soc) for network-enabled visible light communications," *International Journal of Recent Contributions from Engineering, Science & IT (iJES)*, vol. 6, no. 1, pp. 107–119, 2018.

[8] F. Ismail, S. Fuada, T. Adiono, and E. Setiawan, "Prototyping the Li-Fi system based on IEEE 802.15.7 PHY.II.1 standard compliance," *Journal of Communications*, vol. 15, pp. 519–527, 2020.

[9] R. Pitaya, "Red pitaya products," 2021, https://redpitaya.com.

[10] M. Elamassie, M. Uysal, B. Aly, F. Otosan, and E. Kınav, "Experimental evaluation of unipolar OFDM VLC system on software defined platform," in *2019 15th International Conference on Telecommunications (ConTEL)*, Graz, Austria, 2019.

[11] E. Setiawan, T. Adiono, and S. Fuada, "Modelling the OFDM-based phy layer in soc for visible light communication," *International Journal of Recent Contributions from Engineering, Science & IT (iJES)*, vol. 7, no. 3, pp. 79–89, 2019.

[12] P. Y. Tsai, T. D. Chiueh, and I. W. Lai, *Baseband Receiver Design for Wireless MIMO-OFDM Communications*, Wiley, Singapore, 2nd edition, 2012.

[13] E. Setiawan and T. Adiono, "Throughput improvement of an autocorrelation block for time synchronization in OFDM-based LiFi," in *2019 International SoC Design Conference (ISOCC)*, Jeju, Korea (South), 2019.

[14] I. N. O. Osahon, E. Setiawan, T. Adiono, and W. O. Popoola, "Experimental demonstration of visible light communication using white led, blue filter and soc based test-bed," in *2019 International Symposium on Electronics and Smart Devices (ISESD)*, Indonesia, October 2019.

[15] Xilinx, "Xilinx vivado," 2021, https://www.xilinx.com/products/designtools/vivado.html.