IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 June 2024, accepted 30 June 2024, date of publication 9 July 2024, date of current version 17 July 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3425720*

RESEARCH ARTICLE

# A Scalable Design of a Full-Stack Real-Time OFDM Baseband Processor for Network-Enabled VLC Systems

**TRIO ADIONO[1,2], (Senior Member, IEEE), ERWIN SETIAWAN[2], MICHAEL JONATHAN[2], RAHMAT MULYAWAN[1,2], (Member, IEEE), NANA SUTISNA[1,2], (Member, IEEE), INFALL SYAFALNI[1,2], (Member, IEEE), AND WASIU O. POPOOLA[3], (Senior Member, IEEE)**

[1]School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, West Java 40132, Indonesia
[2]University Center of Excellence on Microelectronics, Institut Teknologi Bandung, Bandung, West Java 40132, Indonesia
[3]School of Engineering, Institute for Digital Communications, The University of Edinburgh, EH9 3JL Edinburgh, U.K.

Corresponding author: Nana Sutisna (nana.sutisna@staff.stei.itb.ac.id)

This work was supported in part by the Institut Teknologi Bandung (ITB) World Class Professor Program, International Research Program; and in part by Indonesia Collaboration Research Program.

**ABSTRACT** Visible Light Communication (VLC) system, a wireless system that utilizing the visible light spectrum to transmit data has been considered to become a complementary technology to radio frequency (RF) wireless technology in overcoming spectrum limitations. Recently, numerous VLC research projects have been conducted. However, many of these experiments were performed under ideal conditions, and only a few demonstrated real-time results with networking capabilities. To address this limitation, this paper proposes and details the design of a real-time VLC prototype that supports TCP/IP networking. First, we propose a system-on-chip (SoC) architecture for a VLC system. We design Register-Transfer-Level (RTL) modules using orthogonal frequency-division multiplexing (OFDM) for time synchronization, channel estimation, and equalization. Additionally, we optimize the Sampling Frequency Offset (SFO) estimation and compensation in the receiver to address the mismatch between TX and RX clock oscillators. The optimization of the SFO module is done by eliminating division operations. To realize full-stack design we integrate our design, consisting baseband processor in an FPGA as physical layer device, Linux's kernel TCP/IP stack and the analog output to the VLC analog front-end. This full system design are evaluated over real channel at 1 m distance of transmitter and receiver. Experimental evaluations show that our VLC system can do network tests in an end-to-end system with commercial-off-the-shelf (COTS) networking devices and real applications such as internet browsers. Our VLC system offers a maximum real-time net TCP data rate of 6.65 Mb/s as measured using the `iperf` tool. Theses experimental results show that our VLC system achieves a 6× data rate improvement as compared to the state-of-the-art work. Specifically, our SFO compensator results demonstrate an improvement in TCP data rate of 8.25× and a reduction in frame loss of 22.42%. Moreover, our system also provides design features in terms of flexibility and scalability for future development.

**INDEX TERMS** Baseband processor, channel equalization, channel estimation, FPGA, Li-Fi, OFDM, RTL design, system-on-chip, sampling offset, time synchronization, visible light communication.

## I. INTRODUCTION

The visible light communication (VLC) systems are rapidly gaining research interest amidst the scarcity of the RF spectrum. This technology complements the RF-based

The associate editor coordinating the review of this manuscript and approving it for publication was Yunlong Cai.

wireless communication system [1]. VLC can be done with a light-emitting diode (LED) or laser diode (LD) at the transmitter (TX) and a photo-detector at the receiver (RX). The advantage of using visible light is that it can be used simultaneously for illumination and wireless data transmission [2]. VLC is also a communication method for use in underwater communication because light has less attenuation than RF signals. Therefore, underwater visible light communication (UVLC) can complement traditional acoustic communication for high data rate communication [3]. Furthermore, VLC can be combined with RF to form a hybrid RF/VLC system [4] or a hybrid RF/UOWC system [5].

### A. RESEARCH PROBLEM

Research in Visible Light Communication (VLC) encompasses various technical challenges. One significant challenge involves developing LED devices, such as micro-LEDs [6], [7], [8], with bandwidths exceeding 1 GHz. Another critical area of investigation is achieving data rates of 10-20 Gbps [9], [10], [11] through advanced optical wireless communication models, algorithms, and experimental setups. However, these performance metrics are typically obtained under controlled laboratory conditions using high-end equipment, such as waveform generators and oscilloscopes, with multi-giga sample per second (GSPS) capabilities, rendering them impractical for commercial applications.

To bridge this gap, our research focuses on developing a VLC prototype device capable of real-time communication, aligning closer to practical deployment. A fundamental requirement for such a prototype is the integration of TCP/IP support to ensure seamless networking capabilities.

Real-time VLC prototypes can be divided into two categories: low-speed and high-speed. Low-speed devices can be used for applications that require small amounts of data, such as the Internet of Things (IoT). High-speed devices can be used for applications that require large amounts of data, such as video streaming. The state-of-the-art for the VLC platform is described in [12]. In this paper, we modified and added some references to the state-of-the-art summary of the VLC platform as shown in Table 1. This table summarizes the state-of-the-art VLC platform that supports networking with TCP/IP only. This is because TCP/IP support is an important capability for a final product.

Existing VLC platforms can be classified based on the processor used:

- *General purpose processor (GPP)-based platforms*: OpenVLC 1.4 [12] and DenseVLC [16] use a single-board computer (SBC), while EnLighting [18] uses a micro-controller.
- *Field programmable gate arrays (FPGA)-based platforms*: References [15] and [17] use the Xilinx Zynq 7000 programmable system-on-chip (SoC), i.e., has an FPGA in it. There are many references that use FPGAs, but we don't include them because the majority don't support TCP/IP.
- *COTS Wi-Fi module-based platforms*: WiFi-over-VLC (WoV) [13] and MIMO WoV [14] use Intel's commercial-off-the-shelf (COTS) Wi-Fi application specific integrated circuit (ASIC). We classify these references as semi-commercial products because they use a COTS chip dedicated to wireless communication.
- *Commercial platforms*: Some commercial products, such as Trulifi 6002 [19], LiFiMax [20], and LiFi-XC [21] achieve a data rate of tens to hundreds of Mb/s. However, these products are proprietary, so no implementation details are available.

Based on references [12], [16], and [18], the main limitation is that the processing power and sampling rate of GPP are very limited because they are not designed for signal processing purposes. Consequently, the modulation methods available are restricted to digital modulation, such as on-off keying (OOK). Although FPGA offers greater processing power than GPP, references [15] and [17] still utilize OOK modulation. Additionally, packet processing between PHY and TCP/IP is still performed in the Linux user space, which can introduce significant overhead. References [13] and [14] use an interesting method by employing the COTS Wi-Fi module for VLC transmission. This system uses modulation for high-speed data transfer, namely orthogonal frequency division multiplexing (OFDM). Then, because this is a COTS device, there is already an automatic gain control that allows the VLC system to be used at various distances. However, this system is not flexible in terms of research and development because it uses a proprietary Wi-Fi ASIC chip that cannot be modified in its baseband processing system.

As outlined above, a gap exists in developing networked VLC prototypes that are high-speed, flexible, and highly integrated, particularly with TCP/IP networks. To address this, we propose a system-on-chip (SoC) architecture and register-transfer level (RTL) design for an OFDM baseband processor in a network-enabled VLC prototype. This prototype supports high data rates in the tens of Mb/s range and offers significant flexibility. Additionally, we propose integrating TCP/IP processing for comprehensive system functionality.

### B. RESEARCH CHALLENGES

Wireless communication device design requires cross-disciplinary knowledge and know-how. Knowledge of the OSI model [22] and how to implement it at the OS kernel level, embedded systems, FPGA, digital signal processing, and analog circuit design is needed to perform research in this field. To tackle this challenge, we propose and use a highly integrated SoC FPGA platform. We implemented our architecture on the Ecylpse Z7 [23] board that uses a Xilinx Zynq 7000 programmable SoC FPGA [24]. This board already has an integrated DAC board (Zmod AWG [25]) and an integrated ADC board (Zmod Scope [26]), making development easier. Then, we also used open source references from openwifi [27] to design the Linux kernel

**TABLE 1. A summary of state-of-the-art VLC platforms.**

| Year | Name | Processor | Data rate | TCP/IP support | Commercial product |
|---|---|---|---|---|---|
| **2024** | **Our work** | **Eclypse Z7 Xilinx Zynq 7000** | **6.65 Mb/s** | **Yes** | **No** |
| 2023 | OpenVLC 1.4 [12] | BeagleBone Black | 1 Mb/s | Yes | No |
| 2020 | WiFi-over-VLC (WoV) [13] | Intel WiFi Link 5300 | 150 Mb/s | Yes | Half (Intel) |
| 2020 | MIMO WoV [14] | Intel WiFi Link 5300 | 300 Mb/s | Yes | Half (Intel) |
| 2020 | [15] | ZYBO Xilinx Zynq 7000 | 500 kb/s | Yes | No |
| 2020 | DenseVLC [16] | BeagleBone Black | 33.9 kb/s | Yes | No |
| 2017 | [17] | ZYBO Xilinx Zynq 7000 | 11 kb/s | Yes | No |
| 2016 | EnLighting [18] | Atheros AR9331 & ATmega328p | 400 b/s | Yes | No |
| 2019 | Trulifi 6002 [19] | unknown | 220 Mb/s | Yes | Yes (Signify) |
| 2019 | LiFiMax [20] | unknown | 150 Mb/s | Yes | Yes (Oledcomm) |
| 2017 | LiFi-XC [21] | unknown | 43 Mb/s | Yes | Yes (PureLiFi) |

module for interfacing between baseband processor and Linux TCP/IP processing.

Another challenge in implementing a real-time OFDM system is the sampling offset problem. In many VLC experiments, this sampling offset problem is usually assumed to be ideal [28]. The system in [29] and [30] achieves more than 1 Gb/s data rate using FPGA. The systems have not yet integrated with TCP/IP, and because of that, we did not include it in the state-of-the-art table. The most important issue is the problem with the system clock used. The system still uses the common clock for both TX and RX, where in real conditions, this is not possible. The problem with sampling offset in an OFDM system consists of two problems, namely sampling frequency offset (SFO) and sampling phase offset (SPO).

### C. OUR CONTRIBUTIONS

To address the above gap and challenges, we propose an OFDM baseband processor for a network-enabled VLC system platform that employs the following key techniques:

1) **Architecture design of the OFDM system based on SoC.** We propose an SoC architecture for building the VLC prototype. This architecture is based on a highly integrated SoC FPGA platform. We elaborate and compare our architecture with other state-of-the-arts.
2) **RTL design of the essential OFDM modules.** The modules include time synchronization, sampling offset estimation and compensation, and channel estimation and equalization. We elaborate on the RTL design method for time synchronization, channel estimation and equalization. We simplify the method from [39] and [40] to produce an RTL design for sampling offset estimation and compensation that requires minimal computational processing by eliminating the division process.
3) **System integration of the OFDM baseband processor** both to the TCP/IP stack via a loadable Linux kernel module/driver design and to the VLC analog front-end (AFE). We use the indoor VLC channel at a distance of 1 m.
4) **Evaluation of the network performance** by testing the VLC system with COTS network device to do various network tests using real applications (like `iperf` [31], web browser, etc.).

*Key results:* We evaluated our design and found it achieves data rate improvements of 13× and 6× compared to state-of-the-art systems [12], [15]. Additionally, our system offers greater flexibility and configurability in baseband processing than existing solutions [13], [14]. Our SFO compensator enhances the TCP data rate by 8.25× and reduces frame loss by 22.42%.

## II. PROPOSED SYSTEM MODEL

The most widely used modulation in LED-based VLC is intensity modulation/direct detection (IM/DD) [32]. No carrier signal is used. Modulation is done by changing the intensity of the LED. However, the OFDM characteristics used in RF wireless systems are complex and bipolar, but in IM systems, they must be real and unipolar. So, OFDM for RF systems needs to be modified so that it can be used for LED-based VLC [33]. In our proposed model, we also add SFO and SPO model and simulation.

### A. OFDM TRANSMITTER

The TX data bit stream is encoded using convolutional code [35]. We use two code rate values, $n/k$, which are $1/2$ and $3/4$. After that, an interleaver is used to change the order of the encoded data bits to be resistant to the burst of bit errors. We use a rectangular block interleaver from the Xilinx IP core [36]. Then, these bits are grouped per $M$-bit to form a complex value in the constellation IQ, where $M$ is 1 for BPSK, 2 for QPSK, and 4 for 16-QAM.

One OFDM symbol is composed of 256 subcarriers with an index from $-128$ to 127, as shown in Fig. 1. This subcarrier mapping is based on the IEEE 802.16d standard. In OFDM for VLC, this mapping must be modified from index $-127$ to index $-1$. This location contains the complex conjugate from index locations 1 to 127. So, we can define one OFDM symbol in the frequency domain as $X(k) = [0, X^*_{127}, \ldots, X^*_1, 0, X_1, \ldots, X_{127}]$. The Hermitian symmetry method is used so that the output from the IFFT process is only a real value. There are four types of subcarriers, namely

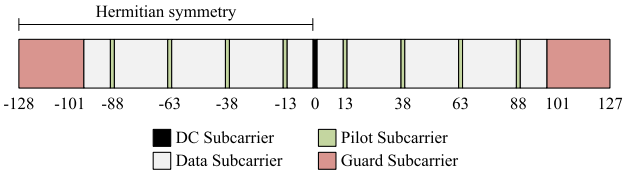


**FIGURE 1.** OFDM symbol allocation and Hermitian symmetry diagram.

DC, data, pilot, and guard subcarriers. The value of the DC subcarrier is 0. The subcarrier data contains complex IQ data that comes from the modulator. The pilot subcarrier contains the value $1 + 0i$, which is used for the SFO estimation and channel estimation at the RX. The subcarrier guard contains a value of 0.

Every OFDM symbol $X(k)$ that contains data subcarriers is then modulated by inverse fast Fourier transform (IFFT) to produce the time domain OFDM symbol $x(n)$ as defined in (1). Where $N$ is the IFFT size, which is 256.

$$x(n) = \sum_{k=0}^{N-1} X(k).e^{\frac{j2\pi kn}{N}} = IFFT\{X(k)\} \tag{1}$$

The IFFT is applied to data subcarriers and also preamble subcarriers. Preambles consist of a short preamble and a long preamble. Let $X_{SP}(k)$ and $X_{LP}(k)$ be the frequency domain of the preamble sequence obtained from the IEEE 802.16d standard [37]. Then, the time domain symbols of the short preamble and the long preamble are defined as $x_{SP}(n) = IFFT\{X_{SP}(k)\}$ and $x_{LP}(n) = IFFT\{X_{LP}(k)\}$, respectively.

One OFDM frame is composed of all OFDM symbols after cyclic prefic addition $X_{CP}(n)$ that consist of a short preamble, a long preamble, and data. This OFDM frame is then upsampled with an upsampling factor of 5. After that, it is filtered with a square-root raised cosine (SRRC) finite impulse response (FIR) filter of order 40.

### B. CHANNEL MODEL

The simulated channel model in this system consists of sampling offset and AWGN noise effects. Let $x(t)$ be the continuous OFDM signal at TX, where $t$ is the time, which further will be converted to discrete signal, and $y[n]$ be the OFDM signal at RX, then $y[n]$ can be defined as

$$y[n] = x((n + \phi).T_{dac}.(1 + \Delta) + n_{AWGN}(n) \tag{2}$$

where $n$ is the sampling index, $\phi$ is the initial sampling position misalignment between the DAC and ADC that causes SPO [38], $T_{dac}$ is the DAC's sampling period, $\Delta$ is the oscillator tolerance that causes SFO, and $n_{AWGN}(n)$ is the AWGN noise. In the ideal condition, $\phi$ and $\Delta$ are zero. The SPO and SFO effects are illustrated in Fig. 2. The SFO is caused by $\phi$. The SFO occurs when $T_{adc} = T_{dac}(1 + \Delta)$, where $\Delta \neq 0$.

In addition to this channel model, we also use the ITU channel model for indoor office (channel A and B) [34] to test the OFDM system.

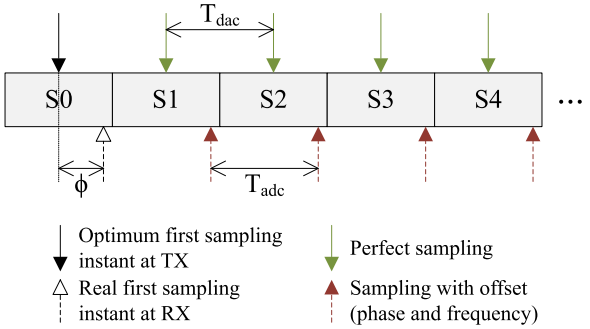


**FIGURE 2.** Illustration of sampling offset.

### C. OFDM RECEIVER

On the RX side, a match filter is used to maximize the signal-to-noise ratio (SNR) of the received OFDM signals. The match filter employs the same structure and coefficients as the pulse shaping filter. After filtering, the signal is downsampled by a factor of 5.

Let $y(n)$ be the received OFDM signal after the match filter and downsampling process. The timing metric to determine the start symbol is obtained by calculating the cross-correlation between the received signal $y(n)$ and the noise-free preamble that is stored in the RX's memory, defined as $p_q, q = 0, \ldots, Q-1$. The cross-correlation function is defined as

$$\Phi(n) = \left| \sum_{q=0}^{Q-1} y(n+q).p(q) \right| \tag{3}$$

where $Q$ is the length of the preamble. In the IEEE 802.16d standard, $Q$ is $N/4 = 64$ samples. Then, find the sample location $n$ where the correlation value is greater than a certain threshold $th$. i.e., it is defined as

$$\Phi(n) > th \tag{4}$$

There are 5 sample locations of correlation peaks, i.e., where the correlation value is greater than the threshold. The start of the symbol is defined at the fifth location.

Every OFDM symbol $y(n)$ is then demodulated by the FFT process to produce the frequency domain OFDM symbol $Y(k)$ as defined in (5). Where $N$ is the FFT size, which is 256.

$$Y(k) = \sum_{k=0}^{N-1} y(n).e^{\frac{-j2\pi kn}{N}} = FFT\{y(n)\} \tag{5}$$

The FFT process is applied to the data symbol and also to the long preamble symbol. Let $y_{LP}(n)$ be the time domain of the long preamble. Then, the frequency domain symbols of the long preamble are defined as $Y_{LP}(k) = FFT\{y_{LP}(n)\}$.

SFO occurs because the oscillators used for the DAC and ADC are independent, so they have a non-zero tolerance. As a result, the frequency sampling is not the same on the DAC and ADC [39]. We perform SFO compensation using the pilot method [40]. The SFO estimation process begins by

calculating the angle between the RX pilots and TX pilots, which is defined as

$$\hat{\phi}_{k_i} = angle(\frac{Y_{k_i}}{X_{k_i}}),\ k_i \in Pilot,\ i = [1, 2, \ldots, 8] \quad (6)$$

This process is carried out for the long preamble symbol and all data symbols. Then, the next step is to calculate the slope, which is defined as

$$s_n = \frac{\sum_{i=1}^{8} k_i.\hat{\phi}_{k_i}}{\sum_{i=1}^{8} k_i^2} \quad (7)$$

After the slope is obtained, the next step is to compensate each subcarrier in the OFDM symbol, which is defined as

$$Y_i' = Y_i.e^{-jis_n},\ i = [0, 1, \ldots, 127, -128, -127, \ldots, -1] \quad (8)$$

This compensation is applied to the long preamble and all data symbols.

Channel estimation is carried out to figure out the VLC channel characteristics. The channel estimation can also be used to estimate SPO and use equalization to compensate it [38]. In our OFDM system, the channel estimation process is carried out by using the long preamble and pilots located in each data symbol. We use the long preamble pattern from the IEEE 802.16d standard as a basis. Then, we modify that long preamble by adding Hermitian symmetry and pilots for SFO estimation purposes.

We use least squares estimation techniques that ignore the effect of AWGN noise to estimate channel response at the long preamble, which is defined as

$$H_{LP}(k) = \frac{Y_{LP}(k)}{X_{LP}(k)} \quad (9)$$

where $Y_{LP}(k)$ is the received long preamble in the $k$-th subcarrier, and $X_{LP}(k)$ is the transmitted long preamble in the $k$-th subcarrier. The calculation is done only for $k = [-100, -98, \ldots, -6, -4, -2, 2, 4, 6, \ldots, 98, 100]$. Then, we perform linear interpolation to get the channel response for $k = [-99, -97, \ldots, -5, -3, -1, 1, 3, 5, \ldots, 97, 99]$. The next step is to perform least squares estimation for pilot subcarriers in data symbols, which is defined as

$$H_{pilot}(l) = \frac{Y_{pilot}(l)}{X_{pilot}(l)} \quad (10)$$

where $Y_{pilot}(l)$ is the received pilots in data symbols, $X_{pilot}(l)$ is the transmitted pilots in data symbols, and $l = [-88, -63, -38, -13, 13, 38, 63, 88]$ is the pilot index. Then, we calculate the channel response variation between $H_{pilot}(l)$ and $H_{LP}(l)$, which is defined as

$$d(l) = \frac{H_{pilot}(l)}{H_{LP}(l)} \quad (11)$$

The value of $d(l)$ will be averaged over 8 pilots to get the channel response update constant value $c_{update}$. Finally, the estimated channel response for each data symbol is defined as

$$H_{data}(k) = H_{LP}(k).c_{update} \quad (12)$$

The equalization method used in our OFDM system is minimum mean squared error equalization (MMSE), defined in (13). This equalization method takes AWGN noise into account. The noise variance $\sigma_n^2$ is calculated from the guard subcarrier, which is filled with a zero value at the TX.

$$Y_{equ}(k) = \frac{Y_{data}(k)}{|H_{data}(k)|^2 + \sigma_n^2}.(H_{data}(k))^* \quad (13)$$

The demodulation process is carried out using the soft decision method, which is not exactly the opposite of the modulation process (hard decision). The soft decision method gives quantization for each data bit and leaves the IQ to data bit conversion process to the Viterbi decoder. In this design, we use 5-bit soft decision quantization. Deinterleaver is the reverse process of interleaver. We also use the rectangular block deinterleaver from the Xilinx IP core [36]. Viterbi decoding [41], [42] is used to decode the soft bits from the deinterleaver output.

## III. PROPOSED SYSTEM ARCHITECTURE

In this section, we elaborate on our proposed SoC architecture for the VLC prototype. At the end of this section, we also compare the advantages of our architecture to the state-of-the-arts.

### A. SYSTEM OVERVIEW

Fig. 3 shows the system overview block diagram of our proposed VLC system. It consists of a COTS Wi-Fi router that is connected to the internet. This router is configured in client mode and connected to another Wi-Fi AP that has an internet connection. The connection to the host FPGA board is done using Ethernet. A point-to-point connection occurs between the host FPGA board and the client FPGA board. The connection on the downlink line uses the VLC channel, while the uplink line uses a cable. The connection from the client FPGA board to the client laptop is done using Ethernet.

### B. OFDM FRAME FORMAT

The frame format used in our OFDM system is based on the IEEE 802.16d and IEEE 802.11a standards. Fig. 4 shows the proposed OFDM frame format. It consists of a short preamble symbol, a long preamble symbol, a header symbol, and a variable number of data symbols. The short preamble is used for time synchronization, and the long preamble is used for channel estimation. The header symbol contains information for processing the data symbols. The contents of the header symbol are modulation type (MCS), data length, and CRC16 to detect errors. We chose a maximum data length of 1600 bytes to accommodate the data size of the Ethernet frame (1518 bytes).

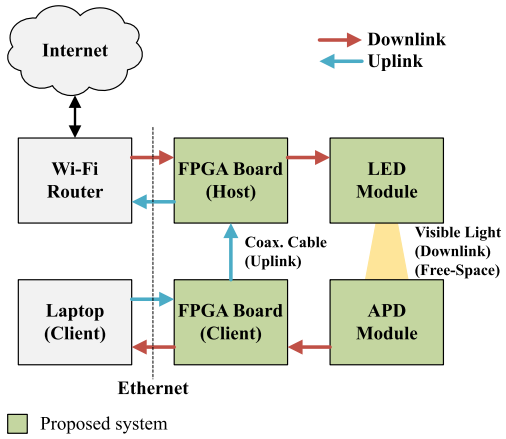


**FIGURE 3.** Overall system block diagram.

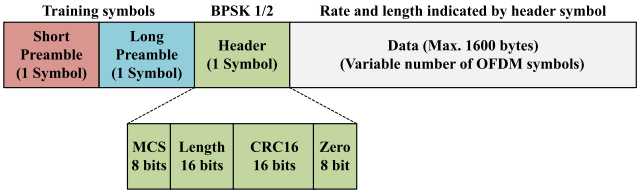


**FIGURE 4.** OFDM frame format.

### C. HARDWARE SOFTWARE PARTITIONING

Fig. 5 shows the partitioning of hardware and software in our proposed VLC system. This system consists of three layers, namely FPGA hardware, Linux kernel space, and user space. The FPGA hardware layer includes our baseband processor implementation, responsible for sending and receiving OFDM signals to and from the DAC and ADC. The Linux kernel space consists of a system call and network socket as an interface to the user application (`iperf`, web browser), the Linux network protocol stack (TCP, UDP, IP, etc.), the net device data structure, the Ethernet driver, and our proposed VLC driver. The VLC driver exchanges data frames with network protocols via the net device data structure. Standard Linux management tools such as `ifconfig` and `iptables` can be used to configure the network.

### D. SOC ARCHITECTURE

Fig. 6 shows our proposed SoC block diagram that is implemented on the Xilinx Zynq Programmable SoC. The system consists of a processing system and programmable logic. On the processing system side, we use an SD card for Linux OS, UART for debug terminal, Ethernet for communication, and GPIO for LED indicator. These peripherals are connected to the ARM Cortex-A8 processor via the AHB/APB bus. On the programmable logic side, there is our proposed OFDM baseband processor design. Data transfer to and from DDR memory is carried out using TX and RX DMA via the AXI bus. TX and RX ring control are used to manage data frames from/to DMA. The data frames themselves are stored in FIFO. The TX frame control

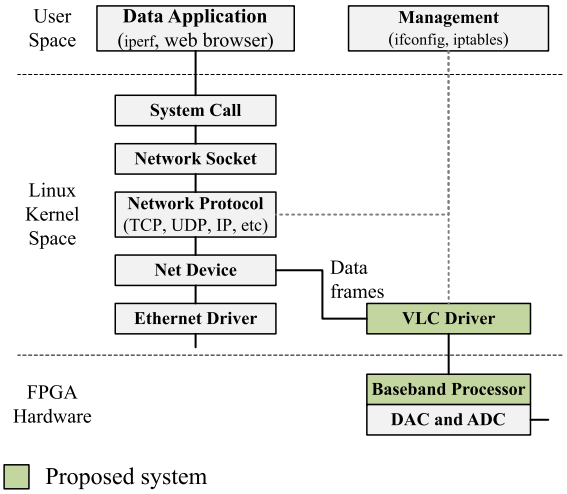


**FIGURE 5.** Hardware software partitioning.

reads one frame from the FIFO and then writes it to the on-chip RAM. After that, the data bytes are read by the TX OFDM control and sent to the OFDM TX datapath for OFDM modulation. On the RX side, the digitized OFDM signal is demodulated by the RX OFDM datapath. Then, the RX frame control reads one frame from the on-chip RAM and then writes it to the FIFO.

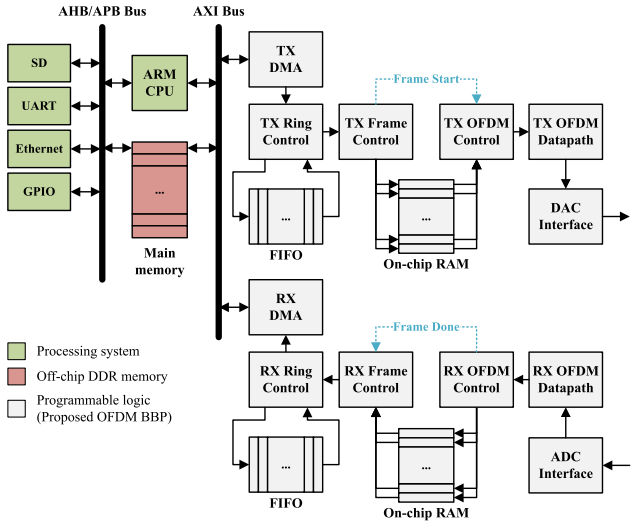


**FIGURE 6.** SoC architecture block diagram.

The main TX and RX frame control modules arrange all blocks in the OFDM datapath by providing control signals to the respected modules based on Finite State Machine (FSM), as shown in Fig. 7. In the TX FSM, the system will enter the state `IDLE` when it is reset. An asserted start signal from the CPU (TX driver) will change the state to `MCS`. In `MCS` state, the system will read the MCS and length from on-chip RAM, and it will configure the datapath, respectively, for the TX data process. In the `PREAMBLE` and `HEADER` states, the system processes and sends preamble

and header symbols, respectively. In the DATA state, the system processes and sends data symbols until the number of required data payloads is reached. After finishing the transmit process, the system will go back to IDLE state.

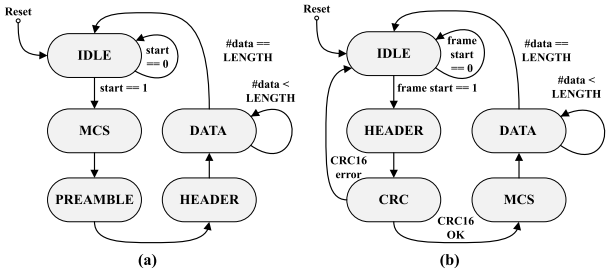


**FIGURE 7.** (a) TX main control FSM; (b) RX main control FSM.

In the RX FSM, the system will enter the state IDLE when it is reset. An asserted start signal from the time synchronization module will change the state to HEADER. In the HEADER state, the system processes the header symbol. In CRC state, the system will check the CRC16. If the result is an error, then it will go back to state IDLE. If the result is okay, then it will go to MCS state. In MCS state, the system will read the MCS and length, and it will configure the datapath, respectively, for the RX data process. In the DATA state, the system receives and processes data symbols until the number of required data payloads is reached. After finishing the receive process, the system will go back to IDLE state.

### E. DIGITAL-TO-ANALOG INTERFACE

Fig. 8 shows the block diagram of the digital-to-analog interface. On the FPGA chip part, there are upsampling and downsampling blocks with a rate of 5×. The baseband clock frequency is set to 50 MHz. Therefore, the baseband sampling rate is 10 MSPS, and the sampling rates of the filter and DAC/ADC are 50 MSPS. There is also digital gain on the FPGA chip, both for TX and RX. The maximum gain of TX is 2×, and RX is 4×.

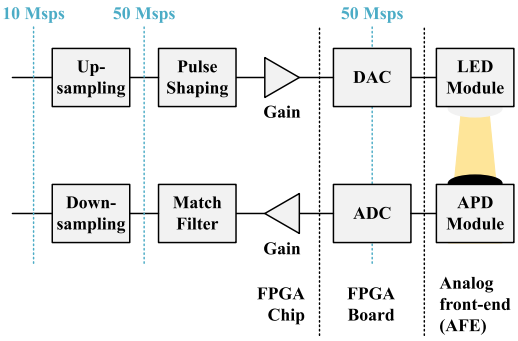


**FIGURE 8.** Digital-to-analog interface diagram.

The TX OFDM frame from baseband is converted from digital to analog by using a DAC that runs at 50 MHz. After that, it goes to the LED module. On the RX side, the light intensity detected by the avalanche photodiode (APD) module is converted from analog to digital by using an ADC that runs at 50 MHz. The LED module consists of a bias-tee ZFBT-4R2GW+ [43] from Mini-Circuits, a resistor of 15Ω, and a standard 5 mm white LED. The bias-tee adds a DC signal to the OFDM signal so that the output is non-negative and is in the linear region of the LED. On the RX side, we use APD module C12702-11 [44] from Hammamatsu. It has a −3 dB frequency bandwidth from 4 kHz to 100 MHz.

### F. ARCHITECTURE COMPARISON WITH STATE-OF-THE-ART

For the sake of clarity, we summarize our proposed architecture and provide a comparison to the state-of-the-art system. In [12] and [16], the PHY layer is limited by the maximum BBB sampling rate (2.1 MHz). In [18], the PHY layer is limited by the maximum ATmega328p sampling rate, and it is also not an integrated chip with the Qualcomm SoC. These architectures are based on GPP, which is not specialized for high-speed signal processing. The architecture of our proposed design is essentially different from these designs because we proposed our system with an SoC architecture and implemented our signal processing on an FPGA, which is specialized for high-speed signal processing due to parallelism.

Reference [15] uses digital modulation based on IEEE 802.15.7 PHY.II.1 Standard, while reference [17] uses UART protocol. PHY driver and packet processing are done in Linux user space software instead of kernel space, which causes data rate limitations. The modulation of our proposed design is essentially different from these designs because we proposed our system with OFDM modulation. OFDM modulation is used in many wireless systems today. In terms of the PHY driver and packet processing, we implemented ours as a loadable Linux kernel module that runs in kernel space. With this method, our driver is closer to the TCP/IP processing module of the Linux system. As a result, it will reduce the bottleneck of data transfer [45].

In [13] and [14], the RF signal from this unmodified Wi-Fi ASIC chip is down-converted to match the VLC analog front-end (AFE) specifications. For the RX part, the signal from the VLC RX is upconverted so that it can be processed by the unmodified Wi-Fi chip. For the MAC and TCP/IP processing sections, they use the standard Linux operating system (OS) run on an Intel NUC mini PC. The architecture of our proposed design is essentially different from these designs because we proposed our system with an SoC architecture, whereas the previous designs used a COTS Wi-Fi ASIC that is specialized for RF.

## IV. PROPOSED RTL DESIGN

The proposed RTL design includes essential OFDM modules: time synchronization, SFO estimation and compensation, channel estimation and equalization, as well as the DMA interface and Linux driver. These modules critically impact the receiver's performance and the overall system. However, the state-of-the-art works listed in Table 1 have not explored

the RTL design of these modules. Additionally, the SFO estimation design in [40] was tested using an AWGN channel and an oscilloscope, rather than an FPGA. This motivates our investigation into the RTL design of these crucial OFDM modules. Detailed descriptions of these modules are provided below.

### A. TIME SYNCHRONIZATION MODULE

Fig. 9 shows the block diagram of the time synchronization module. The input of this module is the free-running signal from the match filter. The objective of time synchronization is to find the start of the OFDM frame by detecting peaks from the cross-correlation process defined in (3). The cross-correlation process is carried out by quantifying input data from 16-bit to 2-bit to save FPGA resources. Then, the input data goes into the shift register, which stores 64 IQ samples. For each IQ sample that arrives, the contents of the shift register will be multiplied by the contents of the ROM and then added up. These results will be absolute and summed to get the final cross-correlation results.

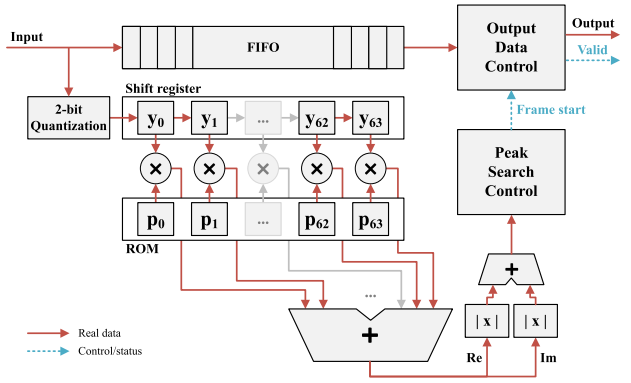


**FIGURE 9.** Time synchronization module.

The peak search control module will compare the cross-correlation results with a threshold value. If it is above the threshold, then it is identified as a peak. The module will count the number of peaks that occur until five peaks are detected. The fifth peak is the start of the OFDM frame. This process is illustrated in Fig. 10.

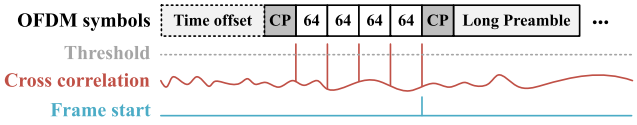


**FIGURE 10.** Timing diagram of the time synchronization module.

### B. SFO ESTIMATION AND COMPENSATION MODULE

SFO estimation is carried out using the pilot method [39], [40]. In this method, the pilots are located on each OFDM symbol. Every OFDM symbol except the short preamble has 8 pilots, which are used for SFO estimation and also for channel estimation. The SFO estimation process begins by calculating the angle of the pilot, which is defined in (6). In our OFDM system, TX pilot $X_{k_i}$ has a real value ($1.4142 + 0i$ for the long preamble symbol and $1 + 0i$ for the data symbol) so that the division calculation in polar form is only subtracted by 0. Then, (6) can be simplified to

$$\hat{\phi}_{k_i} = angle(Y_{k_i}),\ k_i \in Pilot,\ i = [1, 2, \ldots, 8] \tag{14}$$

This simplification eliminates the division operation, thus simplifying the hardware design. The next step for estimating SFO is to calculate the slope, which is defined in (7). In our OFDM system, the pilot location is fixed, therefore the index $k_i$ is always the same and never changes, so that the (7) can be simplified to

$$s_n = \sum_{i=1}^{8} k_i \hat{\phi}_{k_i}.c \tag{15}$$

where $c$ is a constant that is pre-calculated and stored in the register. $c$ is defined as

$$c = \frac{1}{\sum_{i=1}^{8} k_i^2} = 3.7509.10^{-5} \tag{16}$$

This simplification replaces the division operation with the multiplication operation. Therefore, the hardware design only requires multiplication and addition operations. After the slope is obtained, the next step is to compensate each subcarrier in the OFDM symbol, which is defined in (8). In hardware, this calculation is done with CORDIC rotation mode, so that each subcarrier from the FFT is rotated by the value $-is_n$.

Fig. 11 shows the block diagram of hardware design. The process is carried out for every single OFDM symbol, which consists of 256 subcarriers. The 8 pilots are separated from the rest of the OFDM symbol by a pilot extractor block. After that, they are stored in FIFO. CORDIC 0 (translation mode) is used to convert from rectangular form to polar form to get the calculation of (14). Then, the result will be multiplied by the

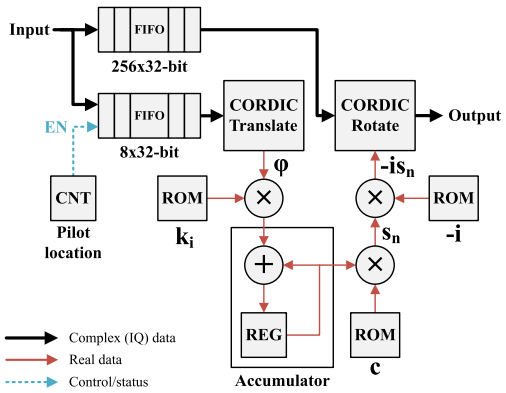


**FIGURE 11.** SFO estimation and compensation module.

pilot subcarrier index, which is then accumulated in a register. After multiplying by $c$, the slope value is obtained, which is the output of Eq. 15. Finally, the compensation process

is carried out by multiplying the index subcarrier data with the slope. Then, this value becomes the rotation angle of CORDIC 1 (rotation mode), which will rotate each subcarrier of the OFDM symbol.

As a remark, in this SFO module implementation, we have optimized the circuit by eliminating two division operations. One division operation is removed, and the other one is replaced with a multiplication operation.

### C. CHANNEL ESTIMATION AND EQUALIZATION MODULE

Channel estimation and equalization is a mandatory module for the OFDM system. However, in the state-of-the-art works in Table 1, the design of this module in RTL has not been explored. Therefore, this motivates us to propose this module.

Fig. 12 shows our proposed channel estimator RTL design for estimating the channel response at long preamble $H_{LP}(k)$ defined in (9). The method used is least squares and interpolation. The design uses a complex divider to get channel responses for even indexed subcarriers. Then, the channel response for odd-indexed subcarriers is obtained using linear interpolation defined in (17), except for subcarriers with indexes of $-1$, 0, and 1. The linear interpolation is implemented using an adder, subtractor, and shifter. Then, for the exceptions, the channel response is calculated by (18), (19), and (20), respectively. This is done because the subcarrier at index 0 is the DC subcarrier that contains 0. At the output, there is a multiplexer that will select the output sequentially based on the subcarrier index.

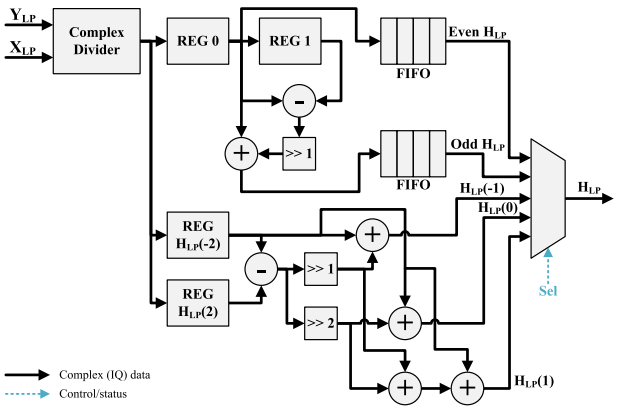


**FIGURE 12.** Long preamble channel estimation and interpolation module.

$$H(n) = H(n-1) + \frac{1}{2}(H(n+1) - H(n-1)) \quad (17)$$

$$H(-1) = H(-2) + \frac{1}{4}(H(2) - H(-2)) \quad (18)$$

$$H(0) = H(-2) + \frac{2}{4}(H(2) - H(-2)) \quad (19)$$

$$H(1) = H(-2) + \frac{3}{4}(H(2) - H(-2)) \quad (20)$$

Fig. 13 shows our proposed pilot channel estimator and equalizer. The first part of the circuit is to calculate the difference between long preamble channel response and pilot channel response $H_{pilot}/H_{LP}$ defined in (11). This process is done only for 8 pilot subcarriers. After that, the results are averaged. The averaging circuit is implemented using an adder, register, and shifter. Then, the result $c_{update}$ is used to update the long preamble channel response to obtain $H_{data}$, i.e., channel response for every data symbol. The $H_{data}$ and noise variance $\sigma_n^2$ are then processed by weight block to obtain the MMSE weight. Finally, the data subcarrier $Y_{data}$ is multiplied by the MMSE weight to obtain equalized data $Y_{equ}$.

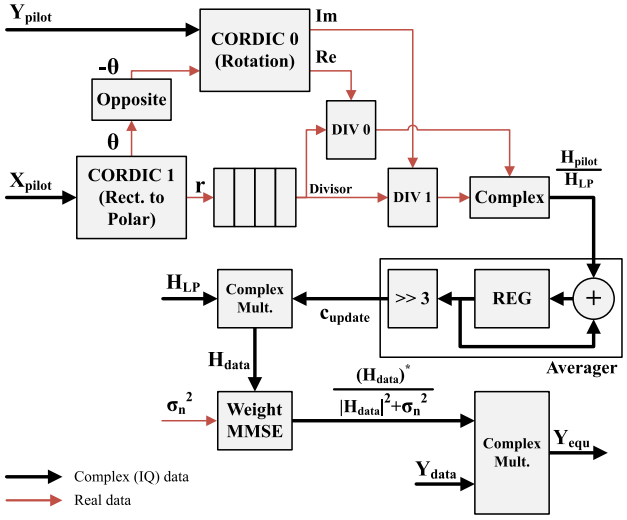


**FIGURE 13.** Pilot channel estimation, channel response updater, and equalization module.

Fig. 14 shows our proposed noise power estimator module calculated by (21). The noise power is estimated from guard subcarrier $g_i$. There is a total of 110 guard values for both I and Q combined.

$$\sigma_n^2 = \frac{\sum_{i=1}^{55}(Re(g_i)^2 + Im(g_i)^2)}{110} \quad (21)$$

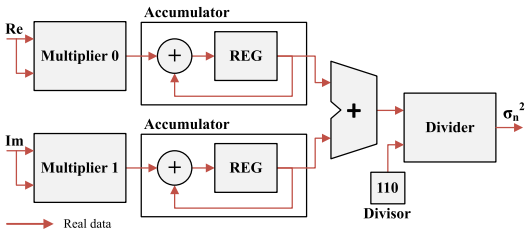


**FIGURE 14.** Noise power estimation module.

### D. DMA INTERFACE AND LINUX DRIVER

Data transfer between hardware and software becomes important to achieve minimum overhead. This can be achieved using DMA and interrupt methods, which are programmed as a loadable Linux kernel module/driver. The method used is adapted from openwifi [27]. Fig. 15 shows the

data flow diagram for the DMA interface and Linux driver. This is the procedure for how data flows:

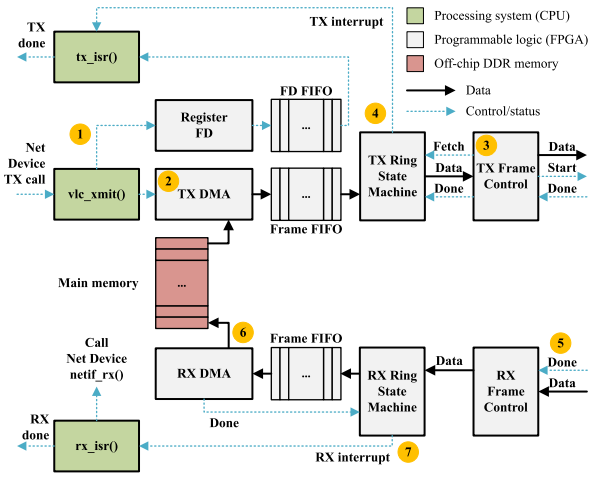


**FIGURE 15.** DMA interface and Linux driver data flow diagram.

1) When there is a data frame, i.e., Linux's socket buffer (SKB), that must be sent from the net device, the net device calls the `vlc_xmit()` function. This function creates a frame descriptor (FD) that contains frame information such as frame size and pointer data to be sent to the FD register and stored in the FD FIFO.
2) The `vlc_xmit()` function performs TX DMA setup by setting the data pointer where the DMA should read data. Then, the TX DMA transfer is executed. The DMA reads data from DDR memory and then sends it to the frame FIFO.
3) The TX frame control requests data from the TX ring control state machine. If a frame is available, then it is sent to the TX OFDM datapath, and the transmission is started. After the process is complete, the done signal is received.
4) The TX ring state machine waits for the done signal from TX frame control. When the done signal occurs, a TX interrupt signal will be sent. In the `tx_isr()` interrupt handler function, the FD that contains frame information will be read, and then the memory location that stores the SKB data will be freed. Finally, the TX done signal toggles an on-board LED to indicate TX activity.
5) In the RX section, the demodulated OFDM bytes are written to the on-chip RAM. A done signal is received when the process is completed. Then, RX frame control sends the frame to the RX ring state machine and to the frame FIFO. The RX ring state machine adds a timestamp to every RX frame.
6) RX DMA reads the frame from frame FIFO and writes it to DDR memory. After the transfer process is completed, there is a done signal (DMA interrupt complete) to the RX ring state machine. The RX ring state machine sends an RX interrupt to the CPU.
7) In the `rx_isr()` interrupt handler function, the data pointer to the received frame is read and sent to the `netif_rx()` function in the form of SKB data structure. The `netif_rx()` is a function from Linux's net device structure that processes the frame to the network layer. Finally, the RX done signal toggles an on-board LED to indicate RX activity.

The Linux driver is implemented as a loadable kernel module on the Linux OS. We use PetaLinux OS version 2019.1 [46]. It is an embedded Linux Software Development Kit (SDK) targeting the Xilinx SoC FPGA.

## V. PERFORMANCE RESULTS AND DISCUSSIONS

This section reports and discusses the simulation results of the model, the experimental setup, real-time performance, and comparison with other works.

### A. SIMULATION RESULTS

#### 1) SFO COMPENSATION

Our SFO compensation model is evaluated using a MATLAB simulation. The SFO compensator can reduce the error vector magnitude (EVM) to below 10% from SFO −100 to 100 ppm. The details of SFO performance are published in [47]. Fig. 16 shows the result of bit error rate (BER) simulation under various SFO values with 16-QAM modulation. At SFO values of 20 and 80 ppm without compensation (red and magenta curves), the achieved BER curve reaches the error floor. After compensation, the achieved BER curves (blue and black curves) are approach the ideal condition, which has an SFO of 0 ppm (green curve).

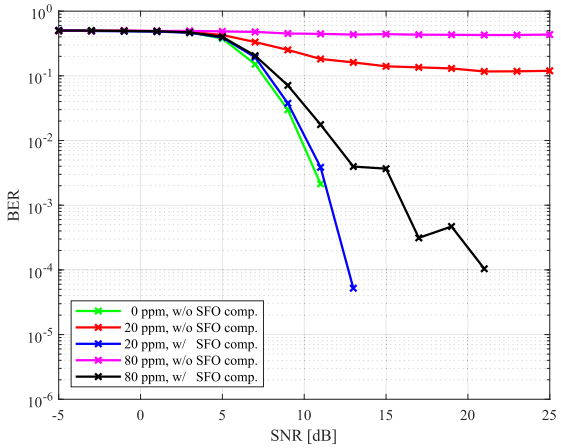


**FIGURE 16.** BER vs. SNR performance under the SFO effect.

#### 2) SPO COMPENSATION

Our MMSE equalizer can reduce the EVM to below 10% from SPO −0.5 to 0.5. The details of SFO performance are published in [47]. Fig. 17 shows the result of bit error rate (BER) simulation under various SPO values with 16-QAM modulation. At normalized SPO values of −0.3 and 0.12 without equalization (red and magenta curves), the

achieved BER curve is at the error floor. After equalization, the achieved BER curves (blue and black curves) are close to the ideal condition, which has an SPO of 0 (green curve).

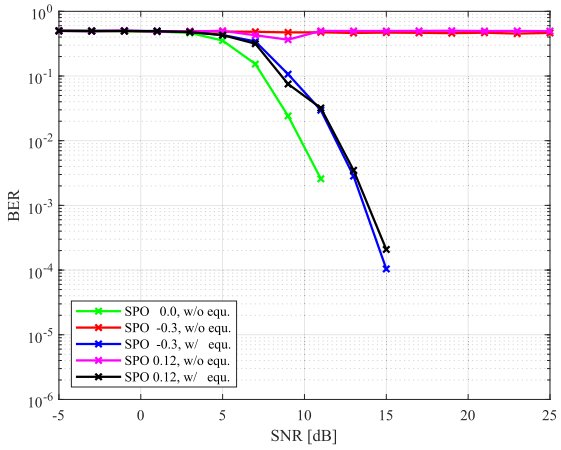


**FIGURE 17.** BER vs. SNR performance under the SPO effect.

### 3) BIT ERROR RATE

The BER simulation for the whole OFDM system is carried out on the AWGN channel. The normalized SPO value is 0.12 because this is the value at which the EVM result is the worst. The SFO value used is 10 ppm because it is the tolerance of the oscillator on our target FPGA board. Based on the simulation result, the minimum SNR required to reach zero BER is 1 dB for BPSK and 17 dB for 16-QAM.

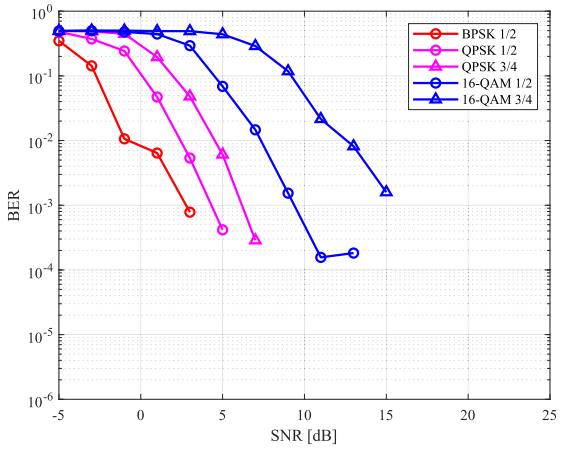


**FIGURE 18.** BER vs. SNR for the overall OFDM system under the following conditions: AWGN, SPO=0.12, SFO=10 ppm, MTU=1600 bytes [47].

## B. FPGA RESOURCE

### 1) OVERALL SYSTEM

Our RTL design is synthesized and implemented for Eclypse Z7 boards that use the Xilinx Zynq 7000 XC7Z020-1CLG484C chip with the Xilinx Vivado version 2019.1. Our design can run at a maximum clock frequency of 50 MHz. Total on-chip power consumption obtained from the Xilinx Vivado tool is 2.421 W. Table 2 shows the FPGA available resource and its utilization.

**TABLE 2.** FPGA resource utilization.

| Resource | Utilization | Available | Utilization% |
|---|---|---|---|
| LUT | 43046 | 53200 | 81 |
| LUTRAM | 3392 | 17400 | 19 |
| FF | 46786 | 106400 | 44 |
| BRAM | 89 | 140 | 64 |
| DSP | 48 | 220 | 22 |

For the sake of clarity, we cannot compare our FPGA resource utilization to the state-of-the-art because such data is not available. In work [12], [16], and [18], they use a GPP-based platform, not an FPGA. In work [13] and [14], they use COTS Wi-Fi chip, so the resource utilization is not available. Although in work [15] and [17] they use FPGA but do not provide the data.

### 2) SFO OPTIMIZATION

In our SFO module implementation, we have optimized the circuit by eliminating two division operations. One division operation is removed, and the other one is replaced with a multiplication operation. Table 3 shows a comparison of FPGA resource utilization of the required operations before and after optimization. The resource comparison is done between two 16-bit fixed-point dividers compared to one 16-bit fixed-point multiplier.

**TABLE 3.** A comparison of FPGA resource utilization for SFO module operations.

| Module | Before optimization | After optimization |
|---|---|---|
| Divider | 2 | 0 |
| Multiplier | 0 | 1 |
| **Resource** | **Before optimization** | **After optimization** |
| Slice LUTs | 820 | 300 |
| Slice register | 1118 | 155 |
| Slice | 280 | 84 |
| LUT as logic | 794 | 300 |
| LUT as memory | 26 | 0 |

## C. EXPERIMENTAL SETUP

Fig. 19 shows a photograph of our experimental setup. We use a collimating lens and a focusing lens from Thorlabs Optics [48]. We also use optical posts and mounting from Thorlabs Optomechanical Components [49]. The LED module is placed at the focal length of the collimating lens so that the light is aligned in a parallel fashion in order to increase the distance. The APD module is placed at the focal length of the focusing lens in order to focus the light onto a 1 mm photodiode. The distance between lenses is 1 m. In this experimental setup, the analog subsystem is currently under development. We are utilizing the digital amplifier within the FPGA, without additional analog amplifiers on the transmission (TX) or reception (RX) sides. Future

improvements, including the use of a high-power LED and supplementary analog amplifiers on both the TX and RX sides, are anticipated to enhance the transmission distance. The OFDM bandwidth measured by the spectrum analyzer is 4 MHz.

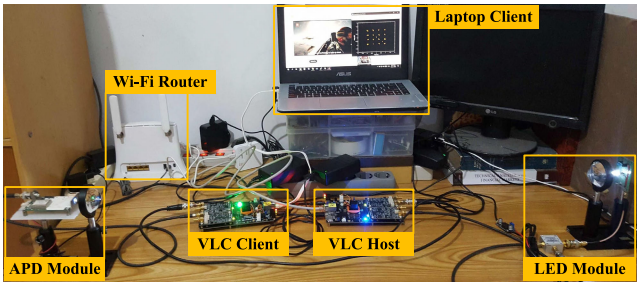


**FIGURE 19.** A photograph of an experimental setup illustrating the proposed VLC system.

Fig. 20 shows our network diagram. Our VLC system has a Linux driver, so it is detected as a standard interface in the Linux system, named `vlc0`. We use a wireless router from TP-Link (TL-MR3020) as an internet source. This router is connected to our VLC host board using Ethernet. The gateway address of the VLC host board is set to the router's IP address. On the VLC client board, the gateway address is set to the VLC host IP address. This VLC client board is connected to the client laptop. The client laptop gateway is configured to the VLC client IP address. We use the `iptables` program in order to setup interface forwarding between interfaces `eth0` and `vlc0`.

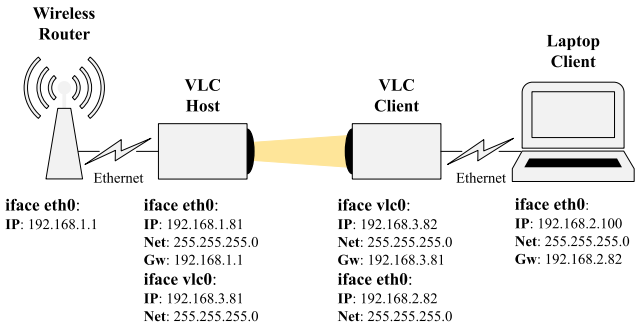


**FIGURE 20.** Network diagram of the system. The IP address settings for each interface are shown.

### D. REAL-TIME PERFORMANCE

#### 1) BIT ERROR RATE

Fig. 21 shows our real-time IQ constellation plot from our data logging program at a distance of 1 m for QPSK and 16-QAM modulation. BER measurements were carried out on the system. We use test data as many as 1000 frames. Each frame contains 1500 bytes of data, so the total number of test data is 12000000 bits. All types of modulation get a BER value of 0 at a distance of 1 m. This condition is in accordance with the BER simulation results at SNR > 20 dB.

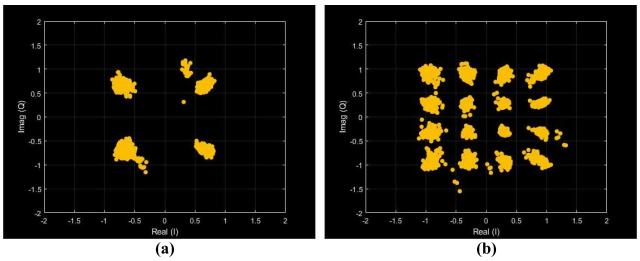


**FIGURE 21.** Real-time IQ constellation plot for (a) QPSK and (b) 16-QAM captured from our data logging program.

**TABLE 4.** Theoretical data rate, measured data rate, and its comparison.

| Modulation & rate | Theoretical date rate [Mb/s] | Measured date rate (UDP) [Mb/s] | Measured date rate (TCP) [Mb/s] | Difference (theor. vs. TCP) [%] |
|---|---|---|---|---|
| QPSK 1/2 | 2.60 | 2.49 | 2.44 | -6.2 |
| QPSK 3/4 | 3.90 | 3.68 | 3.61 | -7.4 |
| 16-QAM 1/2 | 5.15 | 4.79 | 4.67 | -9.4 |
| 16-QAM 3/4 | 7.45 | 6.82 | 6.65 | -10.8 |

#### 2) DATA RATE

We can calculate the maximum theoretical data rate of the OFDM baseband processor by using (22). Where $T_{pre}$ is the duration of short and long preamble symbols, $T_{head}$ is the duration of header symbol, $T_{data}$ is the duration of data symbols, and $T_{space}$ is the duration of spacing between frames.

$$data\ rate = \frac{payload\ size}{(T_{pre} + T_{head} + T_{data} + T_{space})} \tag{22}$$

In this measurement, we choose a payload size of 1470 bytes according to the value of the Ethernet MTU and the default value of the `iperf` payload. In a 50 MHz baseband clock system, the durations of $T_{pre}$, $T_{head}$, and $T_{space}$ are $64\mu s$, $32\mu s$, and $137.5\mu s$, respectively. The duration of $T_{data}$ is variable according to the modulation and encoding rate. The value of $T_{data}$ can be calculated as $32\mu s * N_{sym}$, where $N_{sym}$ is the number of OFDM data symbols defined as

$$N_{sym} = \left\lceil \frac{N_{payload}}{(96 * N_{bpsc} * rate) - 8} \right\rceil \tag{23}$$

where $N_{payload}$ is the number of data payloads in bits, $N_{bpsc}$ is the number of bits in one subcarrier, and *rate* is the encoding rate.

Table 4 shows the theoretical data rate and measured data rate. The data rate is measured between the VLC host and VLC client at a distance of 1 m. There are two real data rates that are measured using `iperf`, namely for the UDP and TCP protocols. For all modulations and rates, there is about a 6-10% drop between the theoretical data rate and the real TCP data rate. This is justified due to processing in the Linux network protocol and driver. Compared to each other, UDP achieves a higher data rate than TCP. This is justified due to the overhead from the handshake protocol of the TCP connection.

We evaluate the effect of SFO on data rate in real-time condition. The real-time TCP data rate on the system without SFO compensation is much lower than the system with SFO compensation, as shown in Fig. 22. The system with SFO compensation has an average data rate of 4.35 Mb/s, whereas the system without SFO compensation has an average data rate of 0.47 Mb/s. Our SFO compensation module boosts the TCP data rate by 8.25x on average.

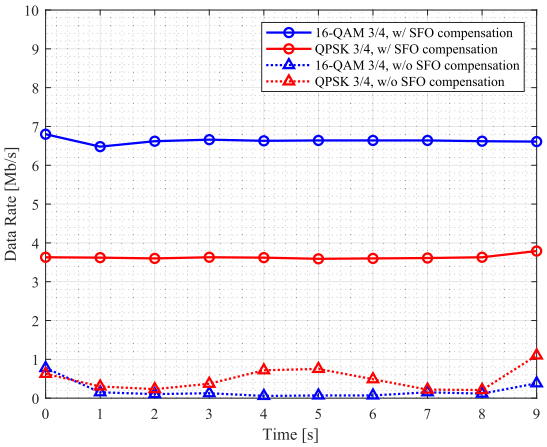


**FIGURE 22.** Effect of SFO on data rate [50].

#### 3) LATENCY, JITTER, AND PACKET LOSS

Table 5 shows measured latency, jitter, and packet loss. These parameters are measured between the VLC host and VLC client at a distance of 1 m. The average latency, jitter, and packet loss for all modulations and rates are 0.905 ms, 0.003 ms, and 0.155%, respectively. The result shows that our system is at an acceptable latency, jitter, and packet loss level.

**TABLE 5.** Latency, jitter, and packet loss measurements.

| Modulation & rate | Latency [ms] | Jitter [ms] | Packet loss [%] |
|---|---|---|---|
| QPSK 1/2 | 1.06 | 0.004 | 0.05 |
| QPSK 3/4 | 0.95 | 0.003 | 0.16 |
| 16-QAM 1/2 | 0.81 | 0.003 | 0.17 |
| 16-QAM 3/4 | 0.80 | 0.002 | 0.24 |

The packet loss comparison between the system with SFO compensation and the system without SFO compensation is displayed in Fig. 23. This data is obtained from real-time measurements using `iperf` UDP. In the system without SFO compensation, the average packet loss is 22.92%, whereas in the system with SFO compensation, the average packet loss is 0.50%. Our SFO compensating module reduces the packet loss by 22.42% on average.

### E. COMPARISON OF TRANSMISSION QUALITY WITH OTHER WORKS

Our work addresses the limitation of sampling rate by still using a low-cost FPGA, but paired with a DAC/ADC that has a maximum 100 MHz sampling rate. The modulation that we use is OFDM. Table 6 shows a comparison of our prototype data rates and average packet loss with other works. Compared to the state-of-the-art [12] and [15], our system achieves 13× and 6× data rate improvements, respectively. Compared to [13] and [14], in terms of data rate comparison, our work is much lower because they use COTS Wi-Fi ASIC, which can be categorized as proprietary. However, our system also provides more flexibility and configurability in the baseband processing because we use FPGAs. It is important to note that our design is scalable, providing flexibility through its implementation in FPGA, which offers advantages over other prototypes such as [13] and [14]. This scalability allows for potential data rate enhancements with larger FPGA resources. In terms of average packet loss, our work demonstrates a better rate compared to [16], while other references do not provide quantitative results for average packet loss.

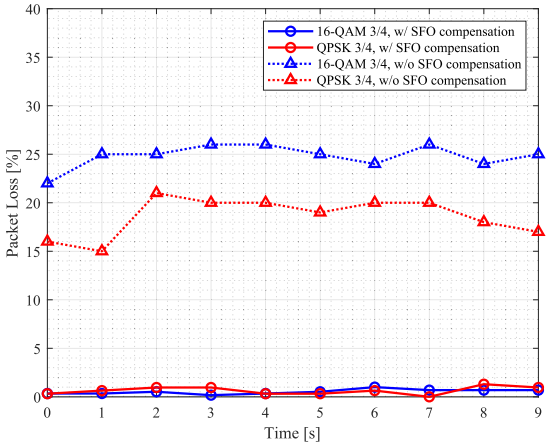


**FIGURE 23.** Effect of SFO on packet loss [50].

**TABLE 6.** Comparison of transmission quality with other works.

| Reference | Data rates | Average packet loss |
|---|---|---|
| EnLighting [18] | 400 b/s | - |
| [17] | 11 kb/s | - |
| DenseVLC [16] | 33.9 kb/s | 0.19% |
| [15] | 500 kb/s | - |
| OpenVLC 1.4 [12] | 1 Mb/s | - |
| **Our work** | **6.65 Mb/s** | 0.16% |

### F. SCALABILITY

To further increase the data rate, we estimate the potential data rate increase that can be achieved if we use a high-cost FPGA that has a higher sampling rate. Our current system uses a baseband clock of 50 MHz and oversampling in the baseband by 5×. Furthermore, the Hermitian symmetry causes, for the same baseband clock as RF (without Hermitian), the data rate to be reduced by half due to the loss of the imaginary data (Q) component. Based on these parameters, our real data rate, measured by `iperf` is

6.65 Mb/s. Therefore, we can calculate a conversion factor $c$ which can be used to estimate the estimated potential data rate increase. This conversion factor includes all the existing overhead from FPGA, Linux, and TCP/IP processing. The conversion factor $c$ is defined as

$$c = \frac{data\_rate}{(bb\_clock/over\_rate/herm\_fact)} \tag{24}$$

where *herm_fact* is 2 in the OFDM system that uses Hermitian and 1 in the OFDM system that doesn't use Hermitian. For our system, $c$ is $6.65/(50/5/2) = 1.33$. Then, we can use this c to estimate the potential data rate increase in Table 7.

**TABLE 7. The estimation of the potential real-time net data rate increase.**

| Baseband clock [MHz] | Oversampling rate | Hermitian factor | Estimated data rate [Mb/s] |
|---|---|---|---|
| 200 MHz | 5 | 2 | 26.6 |
| 200 MHz | 2 | 2 | 66.5 |
| 200 MHz | 2 | 1 | 133.0 |

Our system has limitations on the baseband clock that can be used because the FPGA resource usage is already close to 80% which makes routing more difficult, thereby increasing the critical path of our design. Further improvements can be made with high-end FPGAs, which have larger resources, higher speed grades, and optimization of the RTL design to reduce critical paths. With this FPGA, we hypothesize that the baseband clock can be increased up to 200 MHz. High-end FPGAs with GSPS DAC/ADC usually have an on-chip interpolation filter so that the baseband oversampling factor can be reduced to increase the baseband sample rate, as done on [27]. Finally, Hermitian symmetry can be eliminated, so that it will increase the data rate twice with the same baseband clock but requires a carrier signal, as done in [13] and [14]. This method has been standardized as light communication (LC) in the new IEEE 802.11bb standard [51] for VLC.

GSPS RFDAC/RFADC usually supports super-sample rate (SSR) data transfer, for example, the Xilinx RFSoC FPGA [52]. The SSR enables FPGAs with clock frequencies of only hundreds of MHz to process I/Q samples at GSPS rates [53]. This method requires RTL modules that process multiple samples in parallel in a single clock cycle. There is an FPGA implementation using this SSR method for free-space optical wireless communication [54]. The work proposes a low-latency system, but not the data rate yet. We can also propose this method for our design scalability.

Table 8 estimates our scalable design estimation for the SSR method. This estimation is done based on the RFSoC 4 × 2 board [55]. The baseband SSR of 400, 800, and 1600 MHz can be achieved using a 3200 MHz RFDAC/RFADC sample rate and FPGA clock frequencies of 50, 100, and 200 MHz, respectively. The RFDAC/RFADC uses an interpolation/decimation factor of 2 and an SSR factor of 8. With the SSR method, our scalable design is estimated to achieve 532.0 Mb/s of net data rate with 16-QAM 3/4 modulation, while the overall FPGA resource utilization is estimated to be increased by a factor of 2 to 4. This resource increase is a result of several RTL modules that need to be implemented as multi-core in order to be able to process two samples per clock cycle before the FPGA's upsample or downsample process.

**TABLE 8. The estimation of the potential real-time net data rate increase using the SSR method.**

| Baseband clock [MHz] | Baseband SSR [MHz] | Ovr. rate | Hermitian factor | Estimated data rate [Mb/s] |
|---|---|---|---|---|
| 50 MHz | 400 MHz | 4 | 1 | 133.0 |
| 100 MHz | 800 MHz | 4 | 1 | 266.0 |
| 200 MHz | 1600 MHz | 4 | 1 | 532.0 |

## VI. CONCLUSION

In this paper, we have proposed an SoC architecture and an RTL design of OFDM baseband processor for a network-enabled VLC system. We have proposed and optimized the SFO estimation and compensation module by eliminating division operations. We have also proposed the integration with TCP/IP of the Linux kernel and the AFE of VLC. As a result, our designed system is capable of real-time data transmission through the VLC channel. The real-time network performance is measured using a real application (`iperf`). Experimental results show that our VLC system has a maximum real-time TCP data rate of 6.65 Mb/s. Our VLC system achieves a 6x increase in data rate compared to state-of-the-art systems. Leveraging FPGA technology, our system offers flexibility and configurability, supporting future development. The SFO compensator results demonstrate an 8.25-fold improvement in TCP data rate and a 22.42% reduction in frame loss. Additionally, our design enhances flexibility and scalability, facilitating advancements in VLC technology.

## REFERENCES

[1] Z. Ghassemlooy, W. Popoola, and S. Rajbhandari, *Optical Wireless Communications: System and Channel Modelling With MATLAB*, 2nd ed., Boca Raton, FL, USA: CRC Press, 2019.

[2] H. Elgala, R. Mesleh, and H. Haas, "Indoor broadcasting via white LEDs and OFDM," *IEEE Trans. Consum. Electron.*, vol. 55, no. 3, pp. 1127–1134, Aug. 2009.

[3] M. F. Ali, D. N. K. Jayakody, and Y. Li, "Recent trends in underwater visible light communication (UVLC) systems," *IEEE Access*, vol. 10, pp. 22169–22225, 2022.

[4] H. Abuella, M. Elamassie, M. Uysal, Z. Xu, E. Serpedin, K. A. Qaraqe, and S. Ekin, "Hybrid RF/VLC systems: A comprehensive survey on network topologies, performance analyses, applications, and future directions," *IEEE Access*, vol. 9, pp. 160402–160436, 2021.

[5] L. Jan, G. Husnain, W. T. Sethi, I. U. Haq, Y. Y. Ghadi, and H. K. Alkahtani, "Empowering the future of hybrid MIMO-RF UOWC: Advanced statistical framework for channel modeling and optimization for the post-5G era and beyond," *IEEE Access*, vol. 11, pp. 106361–106373, 2023.

[6] R. Lin, Z. Jin, P. Qiu, Y. Liao, J. Hoo, S. Guo, X. Cui, and P. Tian, "High bandwidth series-biased green micro-LED array toward 6 Gbps visible light communication," *Opt. Lett.*, vol. 47, no. 13, p. 3343, 2022.

[7] F. Xu, Z. Jin, T. Tao, P. Tian, G. Wang, X. Liu, T. Zhi, Q.-A. Yan, D. Pan, Z. Xie, K. Xu, B. Liu, and R. Zhang, "C-plane blue micro-LED with 1.53 GHz bandwidth for high-speed visible light communication," *IEEE Electron Device Lett.*, vol. 43, no. 6, pp. 910–913, Jun. 2022.

[8] Z. Wei, L. Wang, Z. Liu, C. Zhang, C.-J. Chen, M.-C. Wu, Y. Yang, C. Yu, L. Wang, and H. Y. Fu, "Multigigabit visible light communication based on high-bandwidth InGaN quantum dot green micro-LED," *ACS Photon.*, vol. 9, no. 7, pp. 2354–2366, Jun. 2022.

[9] T. Z. Gutema, H. Haas, and W. O. Popoola, "WDM based 10.8 Gbps visible light communication with probabilistic shaping," *J. Lightw. Technol.*, vol. 40, no. 15, pp. 5062–5069, Aug. 2, 2022.

[10] R. Bian, I. Tavakkolnia, and H. Haas, "15.73 Gb/s visible light communication with off-the-shelf LEDs," *J. Lightw. Technol.*, vol. 37, no. 10, pp. 2418–2424, May 7, 2019.

[11] C. Lee, M. S. Islim, S. Das, A. Spark, S. Videv, P. Rudy, B. Shah, M. McLaurin, H. Haas, and J. Raring, "26 Gbit/S LiFi system with laser-based white light transmitter," *J. Lightw. Technol.*, vol. 40, no. 5, pp. 1432–1439, Mar. 4, 2022.

[12] B. G. Guzman, M. S. Mir, D. F. Fonseca, A. Galisteo, Q. Wang, and D. Giustiniano, "Prototyping visible light communication for the Internet of Things using OpenVLC," *IEEE Commun. Mag.*, vol. 61, no. 5, pp. 122–128, May 2023.

[13] P. Gawlowicz, E. A. Jarchlo, and A. Zubow, "WiFi over VLC using COTS devices," in *Proc. IEEE Conf. Comput. Commun. Workshops*, Jul. 2020, pp. 340–345.

[14] P. Gawlowicz, E. A. Jarchlo, and A. Zubow, "Bringing MIMO to VLC using COTS WiFi," in *Proc. IEEE Int. Conf. Commun. Workshops*, Jun. 2020, pp. 1–6.

[15] S. Fuada, T. Adiono, F. Ismail, and E. Setiawan, "Prototyping the Li-Fi system based on IEEE 802.15.7 PHY.II.1 standard compliance," *J. Commun.*, vol. 15, no. 6, pp. 519–527, 2020.

[16] J. Beysens, Q. Wang, A. Galisteo, D. Giustiniano, and S. Pollin, "A cell-free networking system with visible light," *IEEE/ACM Trans. Netw.*, vol. 28, no. 2, pp. 461–476, Apr. 2020.

[17] T. Adiono, S. Fuada, M. Luthfi, and R. Aji Saputro, "MAC layer design for network-enabled visible light communication systems compliant with IEEE 802.15.7," *EAI Endorsed Trans. Energy Web*, vol. 4, no. 14, Oct. 2017, Art. no. 153163.

[18] S. Schmid, T. Richner, S. Mangold, and T. R. Gross, "EnLighting: An indoor visible light communication system based on networked light bulbs," in *Proc. 13th Annu. IEEE Int. Conf. Sens., Commun., Netw. (SECON)*, Jun. 2016, pp. 1–9.

[19] (2024). *Trulifi 6002 System*. Accessed: Mar. 7, 2024. [Online]. Available: https://www.signify.com/global/innovation/trulifi/downloads

[20] (2024). *Oledcomm Products*. Accessed: Mar. 7, 2024. [Online]. Available: https://www.oledcomm.net/products/

[21] (2024). *LiFi-XC*. Accessed: Mar. 7, 2024. [Online]. Available: https://www.purelifi.com/products/lifi-xc/

[22] (2024). *OSI Model*. Accessed: Mar. 7, 2024. [Online]. Available: https://en.wikipedia.org/wiki/OSImodel

[23] (2024). *Eclypse Z7*. Accessed: Mar. 7, 2024. [Online]. Available: https://digilent.com/reference/programmable-logic/eclypse-z7/start

[24] (2024). *Zynq 7000 SoC*. Accessed: Mar. 7, 2024. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[25] (2024). *Zmod AWG*. Accessed: Mar. 7, 2024. [Online]. Available: https://digilent.com/reference/zmod/awg/start

[26] (2024). *Zmod Scope*. Accessed: Mar. 7, 2024. [Online]. Available: https://digilent.com/reference/zmod/scope/start

[27] X. Jiao, W. Liu, M. Mehari, M. Aslam, and I. Moerman, "Openwifi: A free and open-source IEEE802.11 SDR implementation on SoC," in *Proc. IEEE 91st Veh. Technol. Conf.*, May 2020, pp. 1–2.

[28] Y. Jiang, Y. Wang, P. Cao, M. Safari, J. Thompson, and H. Haas, "Robust and low-complexity timing synchronization for DCO-OFDM LiFi systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 1, pp. 53–65, Jan. 2018.

[29] C. Wang, G. Li, Y. Ha, S. Han, and N. Chi, "A 2.5Gb/s real-time visible-light communication system based on phosphorescent white LED," in *Proc. 7th Int. Conf. Inf., Commun. Netw. (ICICN)*, Apr. 2019, pp. 140–145.

[30] M. Shi, C. Wang, G. Li, Y. Liu, K. Wang, and N. Chi, "A 5Gb/s 2×2 MIMO real-time visible light communication system based on silicon substrate LEDs," in *Proc. Global LIFI Congr. (GLC)*, Jun. 2019, pp. 1–5.

[31] (2024). *What is IPerf/IPerf3?*. Accessed: Mar. 7, 2024. [Online]. Available: https://iperf.fr/

[32] J. M. Kahn and J. R. Barry, "Wireless infrared communications," *Proc. IEEE*, vol. 85, no. 2, pp. 265–298, Feb. 1997.

[33] R. Mesleh, H. Elgala, and H. Haas, "An overview of indoor OFDM/DMT optical wireless communication systems," in *Proc. 7th Int. Symp. Commun. Syst., Netw. Digit. Signal Process.*, Jul. 2010, pp. 566–570.

[34] *Guidelines for Evaluation of Radio Transmission Technologies for IMT-2000*, Standard ITU-R M.1225, 1997.

[35] A. Viterbi, "Convolutional codes and their performance in communication systems," *IEEE Trans. Commun. Technol.*, vols. COM-19, no. 5, pp. 751–772, Oct. 1971.

[36] (2024). *Interleaver/De-Interleaver V8.0 Product Guide (PG049)*. Accessed: Mar. 7, 2024. [Online]. Available: https://docs.xilinx.com/v/u/en-U.S./pg049-sid

[37] *Part 16: Air Interface for Fixed Broadband Wireless Access Systems*, IEEE Standard 802.16-2004, 2004.

[38] C. Chen, Y. Chen, N. Ding, Y. Wang, J.-C. Lin, X. Zeng, and D. D. Huang, "Accurate sampling timing acquisition for baseband OFDM power-line communication in non-Gaussian noise," *IEEE Trans. Commun.*, vol. 61, no. 4, pp. 1608–1620, Apr. 2013.

[39] H. Shafiee, B. Nourani, and M. Khoshgard, "Estimation and compensation of frequency offset in DAC/ADC clocks in OFDM systems," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2004, pp. 2397–2401.

[40] M. Chen, J. He, J. Tang, and L. Chen, "Pilot-aided sampling frequency offset estimation and compensation using DSP technique in DD-OOFDM systems," *Opt. Fiber Technol.*, vol. 20, no. 3, pp. 268–273, Jun. 2014.

[41] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[42] (2024). *LogiCORE IP Viterbi Decoder V9.0 Product Guide (AXI)*. Accessed: Mar. 7, 2024. [Online]. Available: https://docs.xilinx.com/v/u/9.0-English/pg027viterbidecoder

[43] (2024). *ZFBT-4R2GW+ Mini-Circuits*. Accessed: Mar. 7, 2024. [Online]. Available: https://www.minicircuits.com/pdfs/ZFBT-4R2GW+.pdf

[44] (2024). *C12702 Hamamatsu*. Accessed: Mar. 7, 2024. [Online]. Available: https://www.hamamatsu.com/content/dam/hamamatsu-photonics/sites/documents/99SALESLIBRARY/ssd/c12702serieskacc1214e.pdf

[45] K. Minghao, K. Y. Chyang, and E. K. Karuppiah, "Performance analysis and optimization of user space versus kernel space network application," in *Proc. 5th Student Conf. Res. Develop.*, 2007, pp. 1–18.

[46] (2019). *PetaLinux Tools Documentation UG1144 (v2019.1)*. Accessed: Mar. 7, 2024. [Online]. Available: https://docs.xilinx.com/v/u/2019.1-English/ug1144-petalinux-tools-reference-guide

[47] T. Adiono, E. Setiawan, M. Jonathan, R. Mulyawan, N. Sutisna, and I. Syafalni, "Performance evaluation of sampling clock offset compensation in OFDM VLC system," *Optics Express*, vol. 28, no. 2, pp. 2337–2348, Jun. 2023.

[48] *Optics*. Accessed: Mar. 7, 2024. [Online]. Available: https://www.thorlabs.com/navigation.cfm?guideid=2264

[49] *Optomechanics*. Accessed: Mar. 7, 2024. [Online]. Available: https://www.thorlabs.com/navigation.cfm?guideid=2262

[50] T. Adiono, E. Setiawan, M. Jonathan, R. Mulyawan, N. Sutisna, I. Syafalni, and W. O. Popoola, "FPGA implementation of SFO for OFDM-based network enabled Li-Fi system," in *Proc. IEEE Int. Symp. Circuits Syst.*, Oct. 2023, pp. 1–5.

[51] *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802.11bb-2023, 2023.

[52] (2024). *Zynq UltraScale+ RFSoC*. Accessed: Mar. 7, 2024. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/rfsoc.html

[53] J. O. Lacruz, R. R. Ortiz, and J. Widmer, "A real-time experimentation platform for sub-6 GHz and millimeter-wave MIMO systems," in *Proc. 2021 Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2021, pp. 427–439.

[54] C. McDonald, H. Claussen, R. Farrell, and J. Dooley, "FPGA implementation of a sub-400ns 6G free-space optical wireless communications transmitter," *Opt. Exp.*, vol. 31, no. 16, p. 25933, Jul. 2023.

[55] (2024). *RFSoC 4 × 2*. Accessed: Mar. 7, 2024. [Online]. Available: https://www.realdigital.org/hardware/rfsoc-4x2

**TRIO ADIONO** (Senior Member, IEEE) received the B.Eng. degree in electrical engineering and the M.Eng. degree in microelectronics from the Institut Teknologi Bandung, Indonesia, in 1994 and 1996, respectively, and the Ph.D. degree in VLSI design from Tokyo Institute of Technology, Japan, in 2002. He is currently a Professor with the School of Electrical Engineering and Informatics, and also the Head of the IC Design Laboratory, Microelectronics Center, Institut Teknologi Bandung. He holds a Japanese patent on a high-quality video compression systems. His research interests include VLSI design, signal and image processing, VLC, smart cards, and electronics solution design and integration.

**ERWIN SETIAWAN** received the B.Eng. degree in computer engineering from Maranatha Christian University, Indonesia, in 2014, and the M.Sc. degree in EE from Institut Teknologi Bandung (ITB), Indonesia, in 2018. He is currently a member of the Microelectronics Centre, ITB. His research interests include RTL design, FPGA prototyping, and HW/SW co-design.

**MICHAEL JONATHAN** received the B.Eng. degree in electrical engineering from the Institut Teknologi Bandung (ITB), in 2019. He is currently a member of the Microelectronics Center, ITB. His research interests include signal processing, RF amplifier design, and transceiver design for wireless communication.

**RAHMAT MULYAWAN** (Member, IEEE) received the B.Eng. degree in EE from ITB, Indonesia, in 2008, and the M.Sc. degree in EE from TU Delft, The Netherlands, in 2011. He is currently a member of the Microelectronics Centre, ITB. His research interests include intelligent signal processing, MIMO systems, and transceiver design for optical wireless communications.

**NANA SUTISNA** (Member, IEEE) received the B.S. degree in electrical engineering and the M.S. degree in microelectronics from the Institut Teknologi Bandung, Indonesia, in 2005 and 2011, respectively, and the Ph.D. degree in computer science and electronics from Kyushu Institute of Technology, in 2017. From 2017 to 2020, he was a Postdoctoral Fellow with the Department of Computer Science and System Engineering, Kyushu Institute of Technology. He is currently a Lecturer with Institut Teknologi Bandung. His research interests include VLSI design, baseband wireless system design, AI processor design, and HW/SW co-design and co-verification.

**INFALL SYAFALNI** (Member, IEEE) received the B.Eng. degree in electrical engineering from Institut Teknologi Bandung (ITB), Bandung, Indonesia, in 2008, the M.Sc. degree in electronics engineering from the University of Science Malaysia (USM), Penang, Malaysia, in 2011, and the Dr.Eng. degree in engineering from the Kyushu Institute of Technology (KIT), Iizuka, Japan, in 2014. From 2014 to 2015, he held a research position with KIT. From 2015 to 2018, he was an ASIC Engineer with the ASIC Development Group, Logic Research Company Ltd., Fukuoka, Japan. In 2019, he joined Institut Teknologi Bandung, Bandung, where he is currently an Assistant Professor with the School of Electrical Engineering and Informatics and a Researcher with the University Center of Excellence on Microelectronics, ITB. In 2023, he was a Visiting Scholar with the System Technology Co-optimization (STCO) program, Interuniversity Microelectronics Centre (IMEC), Leuven, Belgium. In 2024, he is also a Visiting Scholar at the System Design Laboratory, the University of Tokyo, Tokyo, Japan. His research interests include logic synthesis, logic design, VLSI design, and efficient circuits and algorithms.

**WASIU O. POPOOLA** (Senior Member, IEEE) is currently a University Reader and the Deputy Director of learning and teaching with the School of Engineering, The University of Edinburgh, Edinburgh, U.K. He has authored or co-authored more than 120 journal articles/conference papers/patent and over seven of those are invited articles. He also co-authored the book *Optical Wireless Communications: System and Channel Modeling with MATLAB* and many other book chapters. His publications have more 6200 citations and an H-index of 34 on Google Scholar. His primary research interests include digital and optical communications, including VLC, FSO, and fiber communications. One of his journal articles ranked No. 2 in terms of the number of full text downloads within IEEE Xplore, in 2008, from the hundreds of articles published by *IET Optoelectronics*, since 1980. He was an invited speaker at various events, including the 2016 IEEE Photonics Society Summer Topical. He is a fellow of the Institute of Engineering Technology and the Higher Education Academy. The article, he has co-authored with one of his Ph.D. students received the Best Poster Award at the 2016 IEEE ICSAE Conference. He is a Science Communicator appearing in science festivals and on ''BBC Radio 5live Science'' Program, in October 2017. He is an Associate Editor of IEEE Access.