IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 13, 2021, accepted July 5, 2021, date of publication July 8, 2021, date of current version July 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3095514

Resource-Optimized Design of Bit-Shuffle Block Coding for MIMO-VLC

YOUNGSUN HAN[1], (Member, IEEE), SANGHYEON LEE[1], BYUNG WOOK KIM[2], (Member, IEEE), AND YONGTAE KIM[3], (Member, IEEE)

[1]Department of Computer Engineering, Pukyong National University, Nam-gu, Busan 48513, South Korea
[2]Department of Information and Communication Engineering, Changwon National University, Uichang-gu, Changwon, Gyeongnam 51140, South Korea
[3]School of Computer Science and Engineering, Kyungpook National University, Buk-gu, Daegu, 41566, South Korea

Corresponding authors: Byung Wook Kim (bwkim@changwon.ac.kr) and Yongtae Kim (yongtae@knu.ac.kr)

This work was supported in part by the Pukyong National University Research Fund (2019) under Grant CD20191535, and in part by the National Research Foundation of Korea (NRF) Grant through the Korean Government (MSIT) under Grant NRF-2020R1A4A1019628.

ABSTRACT Designing a run-length-limited (RLL) code for a visible light communication (VLC) system requires consideration of several performance factors including hardware overhead, transmission efficiency, and direct current (DC) balance. This paper proposes a novel RLL code called bit-shuffle block codes for high-rate multiple-input multiple-output (MIMO)-VLC systems that can offer excellent constant illumination and transmission efficiency with extensively-optimized hardware resources. Due to the structure of the Omega network, the proposed bit-shuffle block coding method can increase transmission efficiency and significantly reduce hardware overhead. In addition, it can generate codewords dynamically, which guarantees DC balance with multiple LEDs to serve the dual purpose of illumination and communication. Experimental results confirm that the proposed bit-shuffle block coding demonstrate excellent performance in resource optimization, and can be a viable solution for MIMO-VLC applications that can send large volumes data with limited hardware resources.

INDEX TERMS Multiple-input multiple-output (MIMO), visible light communication (VLC), run-length-limited (RLL) codes, hardware resource optimization, bit-shuffling.

I. INTRODUCTION

Recently, visible light communication (VLC) has emerged as a valuable communication technology because it supports the use of light-emitting diodes (LEDs) for simultaneous services of illumination and information transmission [1]–[4]. The fast-switching capability of LEDs enables data transmission by modulating the light signal via an optical wireless channel. While conventional wireless frequency bands are constrained by various regulations, VLC provides many advantages, e.g., huge and unregulated spectrum, potentially high data rate, high security, low-cost implementation and no electromagnetic interference. Therefore, VLC has the potential to realize massive human-to-machine and machine-to-machine communication (M2M) in 6G applications [5], [6]. Typically, M2M devices are required to be small and low cost; thus, implementing a VLC device with minimum hardware is an important issue in providing the expansion and flexibility of M2M communication networks.

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

VLC modulates the intensity of LED signals to transmit data, and the use of multiple LEDs can increase communication capacity by exploiting the advantages of multiple-input multiple-output (MIMO) technology. As shown in Fig. 1, multiple luminaires provide lighting and act as a transmitter for communications. The receiver consists of optical lens, filters, and the detector array to uncorrelate optical MIMO channel matrix coefficients to achieve parallel data streams. Given that LEDs are used simultaneously for illumination and communication in MIMO-VLC, direct current (DC) level balancing is important in terms of maintaining consistency in brightness. In previous VLC studies, run-length-limited (RLL) codes were widely adopted because they makes the brightness maintained 50% constantly by eliminating sequences of the continuous 0's or 1's. According to the VLC standard [7], PHY I and PHY II leverage Manchester, 4B6B, and 8B10B codes to support consistent LED brightness. These DC balanced codes have an equal number of 0's and 1's in all of their codewords. In addition, several RLL codes have been applied for single-input single-output (SISO) VLC systems to enhance performance in terms of transmission

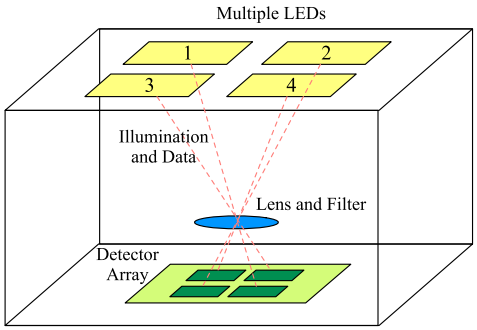

FIGURE 1. MIMO-VLC system model.

efficiency, flicker mitigation, and dimming support [8]–[13]. Several studies on MIMO-VLC systems have investigated methods to increase the capacity of VLC using multiple LEDs [14]–[21]. Among the various MIMO transmission techniques, spatial multiplexing is the most widely adopted in band-limited MIMO-VLC systems where independent data streams are transmitted from each transmitter, thereby offering multiplexing gains and greatly enhanced communication capacities. In [14], a superposed odd-order 32 QAM constellation scheme for 2 × 2 MIMO-VLC systems was proposed to improve BER performance and dynamic range of the driving voltage. An optimal constellation design for the 2 × 2 MIMO-VLC system has been presented in [15], and it improved error performance compared to existing methods. The singular value decomposition (SVD)-based low-complexity scheme for point-to-point MIMO-VLC was proposed to enhance achievable rates [16]. Angle diversity transmitters and receivers and an optimal signal vector design [17] have been introduced to MIMO-VLC systems. In [18], an space-division multiple access (SDMA) technique for multiuser MIMO-VLC systems was proposed to improve the available bandwidth of each user and enhance the achievable rate. Mathematical modelling of positioning in MIMO-VLC systems has been developed [19], and a secure MIMO-VLC system based on the user's location and encryption without affecting efficiency has also been designed. Spatial modulation schemes with dimming control have been proposed [20], and additional compensation symbols in the RLL codes have been proposed for VLC systems [21]. The abovementioned methods increase the data rate using multiple LEDs; however, these studies did not consider situations where both hardware resource consumption and transmission efficiency are strongly constrained. Considering the trend of requiring compact hardware and portable communications devices for massive M2M applications, a new type of RLL code for MIMO-VLC should be studied.

Therefore, in this paper, we propose a novel bit-shuffle block coding scheme to provide a resource-optimized solution for MIMO-VLC systems. To support multi-stream transmission using multiple LEDs, a block coding architecture comprising an Omega network-based encoder and decoder was designed. Using the weighted Hamming distance, it is possible to dynamically generate codewords that can significantly reduce hardware overhead while balancing the dimming levels of multiple LEDs. Our experimental results demonstrate that the proposed bit-shuffle block codes achieve superior resource optimization performance compared to existing RLL codes. Our primary contributions of this paper are summarized as follows:

- We propose bit-shuffle block codes for MIMO-VLC that reduces hardware resource consumption and maintains a dimming level of approximately 50% while minimizing transmission overhead.
- We confirm that the proposed bit-shuffle block coding scheme reduces transmission overhead to 11.1% in 64-bit MIMO-VLC, i.e., transmission efficiency can be improved by 4.5×, 3.0×, and 1.8× over Manchester, 4B6B, and 8B10B, respectively.
- We present the detailed structure of the bit-shuffle block coding scheme for compact hardware implementation and demonstrate that it requires much fewer hardware resources and power consumption than the 4B6B and 8B10B codes.

The remainder of this paper is organized as follows. In Section II, we introduce the encoder and decoder structure of the proposed bit-shuffle block coding scheme. In Section III, we present a hardware implementation of the proposed scheme. In Section IV, we present the results of performance comparisons of the proposed scheme to conventional RLL schemes in terms of transmission efficiency, hardware overhead, and DC balance. Finally, we conclude the work in Section V.

II. BIT-SHUFFLE BLOCK CODING FOR MIMO-VLC

In this section, we present the encoding and decoding algorithms of the proposed bit-shuffle block coding, which is optimized in terms of both hardware resource consumption and DC balancing for MIMO-VLC. In addition, the codeword generation and selection methods based on the weighted Hamming distance are described.

A. OVERALL ARCHITECTURE

Figure 2 shows the structure of the encoder and decoder in the proposed resource optimized bit-shuffle block coding for MIMO-VLC. The encoder transforms the input data D^{in} to the encoded data and metadata $\{D|M\}^{en}$ through the following steps: XORing, bit shuffling, weighted Hamming distance, and codeword selection. Note that XORing means to perform an exclusive OR (XOR) operation on two inputs. Also, metadata usually indicates additional data for data, and in our proposed method, it means a combination of hash and XOR bits excluding shuffled data bits from the encoded output. To reduce the hardware resource consumption, the encoder is designed to repeatedly perform these steps for producing the encoded output. First of all, we transform the input data to a different bit sequence by XORing with an XOR pattern selected by the XOR index X^{can}. This step enables to change the numbers of 0's and 1's of the input

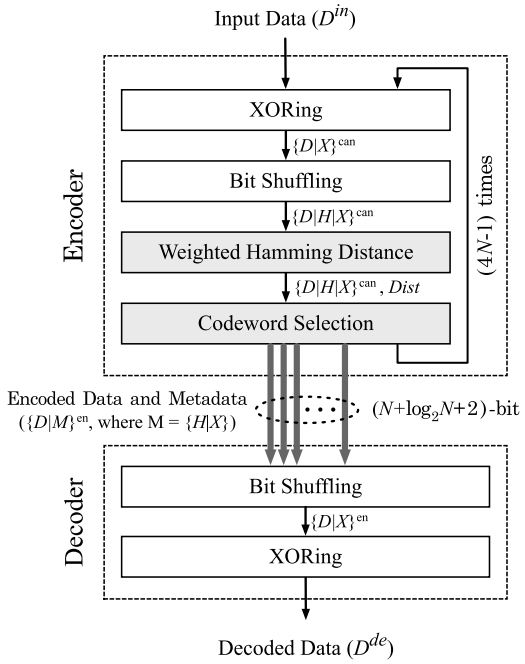

FIGURE 2. Overall architecture of the resource-optimized bit-shuffle coding for MIMO-VLC. *N* represents the input data width.

data so that it contributes to producing more diverse bit sequences in the subsequent bit shuffling step. Note that the diverse bit sequences are required to maintain constant illumination while multiple LEDs are used for communications.

Second, bit shuffling is the core step of the proposed encoding algorithm that generates several candidates by shuffling the XORed bit sequence using an Omega network [13] routed by the hash index H^{can}. The term XORed is used to mean that an XOR operation has already been applied or to indicate its result. The Omega network is a multistage interconnection network composed of multiple stages of switches, supporting a perfect shuffle connection where the output of each stage is connected to the input of the next stage. The equation for bit shuffling is presented as follows:

$$b'_n = b_{n\oplus h} \quad \text{for all } n \text{ where } 0 \le n < N. \tag{1}$$

Here, each bit position b_n of the N-bit input data is shuffled to a new bit position b'_n by XORing with a specific hash h, where $\oplus$ indicates an XOR operation between its two operands. The binary signal of the newly obtained bit position is expressed by the light intensity of each LED at the transmitter, and the multi-stream data is transmitted through a wireless optical channel. Figure 3 shows an example in which 8-bit input b is shuffled to output b' by a hash index 101 in an 8×8 Omega network. Next, we calculate the weighted Hamming distance of the current bit-shuffled candidate $\{D|M\}^{can}$ against the previous encoded output. The distance is computed by applying different weights to compensate for the dimming level difference between the encoded data and the metadata. Finally, in the codeword selection step, we select the one candidate with the maximum weighted Hamming distance among the candidates so that we can find the bit

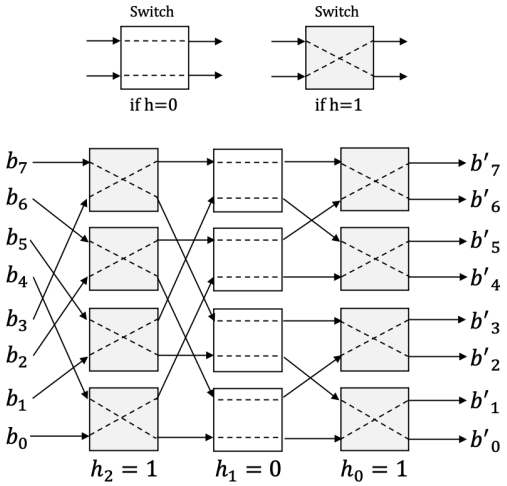

FIGURE 3. An example of an 8 × 8 Omega network. The white box indicates that the switch passes two inputs to the output without swapping if *h* is 0, and the gray box does that the switch exchanges two inputs and outputs them if *h* is 1.

sequence most opposite to the previous encoded output. Thus, we can maintain the dimming level of approximately 50% at each bit position. Note that a single encoding repeats these four steps iteratively for $4 \times N$ shuffled candidates when the input data is N-bit. After the iterations, the encoded output $\{D|M\}^{en}$ is selected and transmitted wirelessly in visible light wavelengths from multiple LEDs.

On the receiver side, multiple photo-detector (PD) arrays are employed to detect the light signal and convert the light photons into digital signals. As shown in Figure 1, one of the significant advantages of bit-shuffle block coding is that we can implement the decoder in a very simply design. It is sufficient for the decoding algorithm to only perform the following two steps sequentially: bit shuffling and XORing. To obtain the decoded output D^{de}, we first apply bit shuffling to the encoded data using the hash index H^{en} from the metadata. We can then output the decoded data by XORing with a bit pattern selected by the XOR index X^{en}.

B. ENCODING AND DECODING ALGORITHMS

Algorithm 1 presents the block encoding algorithm of our resource-optimized bit-shuffle coding for MIMO-VLC. When the input data width is N-bit, the inputs of the procedure are the previously encoded data and metadata $\{D|M\}^{old}$, which is the encoded output produced from the previous input data, and the new input data D^{in}. It also returns a concatenation of the encoded data and metadata $\{D|M\}^{en}$ as output.

To reduce the use of redundant hardware resources, the encoding algorithm is designed to repeatedly perform the four steps of the bit-shuffle coding for different XOR and hash indexes, i.e., p and h, with a nested iteration. We name the four steps as XORing, bit shuffling, weighted Hamming distance, and codeword selection, respectively. First of all, the input data D^{in} is XORed into a new $D^{in'}$ by the Bit-pattern$_p$ selected by the XOR index p in the XORing

step. Thus, we can obtain at most K different $D^{in'}$ for a single D^{in} when we employ K bit-patterns through this procedure. After then, we iterate the remaining three steps N times for each $D^{in'}$ to find the bit-shuffled candidate with the largest weighted Hamming distance against the previously encoded data and metadata $\{D|M\}^{old}$. For each iteration, we produce a bit-shuffled data candidate D^{can} by shuffling $D^{in'}$ with the hash index H^{can} in the bit shuffling step. We also calculate the weighted Hamming distance of the shuffled candidate $\{D|M\}^{can}$ against $\{D|M\}^{old}$, and select one candidate of the maximum Hamming distance by comparing with the other candidates. Finally, we return the encoded data and metadata $\{D|M\}^{en}$ by repeating the previous steps for K different $D^{in'}$.

Algorithm 1 Block Encoding Algorithm

INPUT $\{D|M\}^{old}$: previously encoded data and metadata, D^{in}: input data
OUTPUT $\{D|M\}^{en}$: encoded data and metadata.

procedure Encode($\{D|M\}^{old}$, D^{in})
  $Dist^{max} = 0$
  // Repeat for K bit patterns
  for $p \leftarrow 0$ to K-1 **do**
    // Diverge D^{in} using XORing
    $D^{in'} = D^{in} \bigoplus$ Bit-pattern$_p$
    // Repeat for N candidates
    for $h \leftarrow 0$ to N-1 **do**
      $H^{can} = h$
      $X^{can} = p$
      // Shuffle D^{in} bits by H^{can}
      D^{can} = ShuffleBit($D^{in'}$, H^{can})
      $M^{can} = \{H|X\}^{can}$
      // Calculate weighted Hamming distance
      $Dist^{can}$ = WHDist($\{D|M\}^{can}$, $\{D|M\}^{old}$)
      // Find a candidate of max Hamming distance
      if $Dist^{max} < Dist^{can}$ **then**
        $\{D|M\}^{en} = \{D|M\}^{can}$
        $Dist^{max} = Dist^{can}$
      end if
    end for
  end for
end procedure

Algorithm 2 Block Decoding Algorithm

INPUT $\{D|M\}^{en}$: encoded data and metadata
OUTPUT D^{de}: decoded data.

procedure Decode($\{D|M\}^{en}$)
  // Shuffle D^{en} bits by H^{en}
  $D^{en'}$ = ShuffleBit(D^{en}, H^{en})
  // Perform XORing
  $p = X^{en}$
  $D^{de} = D^{en'} \bigoplus$ Bit-pattern$_p$
end procedure

As shown in Algorithm 2, the block decoding algorithm of the proposed bit-shuffle coding is more straightforward compared to the encoding algorithm. It consists of only the two sequential steps of bit shuffling and XORing. The algorithm's input is an encoded data and metadata $\{D|M\}^{en}$ and the output is a decoded data bit sequence D^{de}. We obtain both the hash and XOR indexes, i.e., H^{en} and X^{en}, from the encoded metadata M^{en}. First, we produce $D^{en'}$ from the encoded data D^{en} by shuffling with hash index H^{en} in the bit shuffling step. Second, the decoded output D^{de} is transformed from $D^{en'}$ by XORing with the bit-pattern of the given XOR index X^{en}. As a result, the hardware complexity of the proposed decoder can be significantly lower than that of the encoder since the decoding algorithm requires bit-shuffling and XORing only once in sequence.

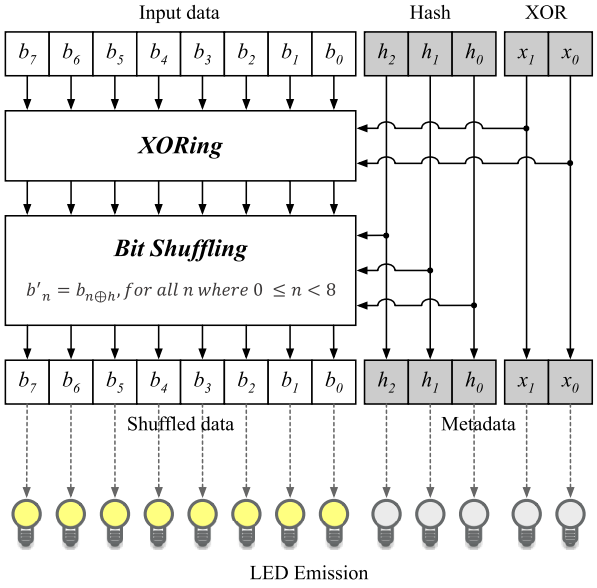

FIGURE 4. LED emission of an encoded output, i.e., shuffled data and metadata bits, in MIMO-VLC when the input data is 8-bit.

C. WEIGHTED HAMMING DISTANCE

Figure 4 illustrates how the input data is transformed into the shuffled data and metadata by successive XORing and bit-shuffling for data transmission from multiple LEDs. In detail, the input data bits b are XORed and bit-shuffled by the given XOR and hash indexes, i.e., x and h, respectively. and the encoded output comprises the shuffled data bits, hash bits, and XOR bits. The bit-flip frequency of the metadata bits is thus constrained relative to that of the shuffled data bits because the hash and XOR bits are returned without any manipulation. If we do not handle the metadata bits separately from the shuffled data bits when calculating the Hamming distance, the dimming level will become unstable at the bit positions of the metadata as shown in Figure 5. Note that the generated codeword of SISO-VLC is the time-domain signal and expressed as light intensity in a single LED. On the other hand, in MIMO-VLC, the codeword is expressed in the space-domain where multiple LEDs are located. Here, if the bit distribution of the upper and lower regions of the codeword generated for MIMO-VLC is not similar, unbalanced illumination for multiple LEDs is perceived to the human eye.

To mitigate this issue, we employ the weighted Hamming distance of the shuffled candidate from the previously

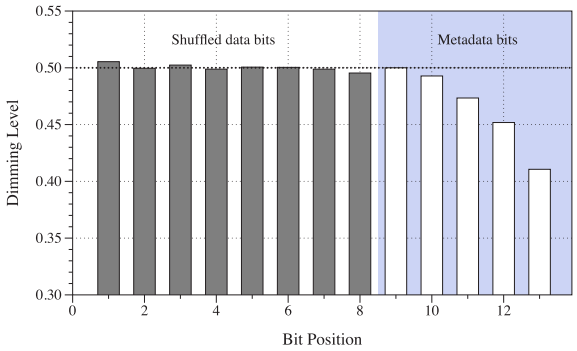

FIGURE 5. Dimming level difference between shuffled data and metadata bits when the weighted Hamming distance is not applied. The input data width is 8-bit.

encoded output in the codeword selection. The weighted Hamming distance is calculated by multiplying three different weights to the number of bit-flips of the shuffled data, hash, and XOR bits, respectively, and accumulating them. Since the codeword selection takes a shuffled candidate of the maximum Hamming distance as an encoded output, we can increase the bit-flip frequency of the metadata bits by applying a higher weight to them. For example, if we apply the weights of 1, 2, and 4 to the bit-flips on the data, hash, and XOR bits, respectively, a bit-flip on XOR bits becomes four times more dominant than that on the data bits in calculating the Hamming distance; thus, it promotes more bit-flips on the XOR bits. Therefore, we can obtain the stable DC balance along the multiple LEDs by setting the dimming ratio of approximately 0.5 at the bit positions of the metadata by adopting the weighted Hamming distance. In order to maintain constant illumination when LED is used for communication, most of the VLC-based DC balance coding schemes aim to set the dimming ratio to 0.5. The reason is that dimming can be controlled using compensation symbols provided by the IEEE 802.15.7 VLC standard [7]. By attaching compensation symbols to the end of the bit shuffle block code, the user can adjust the desired lighting level.

III. HARDWARE DESIGN

In this section, we describe the hardware designs of the encoder and decoder for our proposed bit-shuffle block coding in MIMO-VLC. We present a detailed implementation of their hardware components to reduce hardware resource consumption.

A. ENCODER AND DECODER

Figure 6 exhibits the overall architecture of the proposed bit-shuffle block encoder that performs the encoding algorithm shown in Algorithm 1. The encoder mainly consists of the following five modules: XorBit, ShuffleBit, WeightedHDist, SelectMax, and a control unit. The control unit is designed to operate synchronously by a system clock and outputs three different control signals, i.e., X, H, and S, for every clock cycle according to the control flow of Algorithm 1. The required system clock frequency varies according to the input

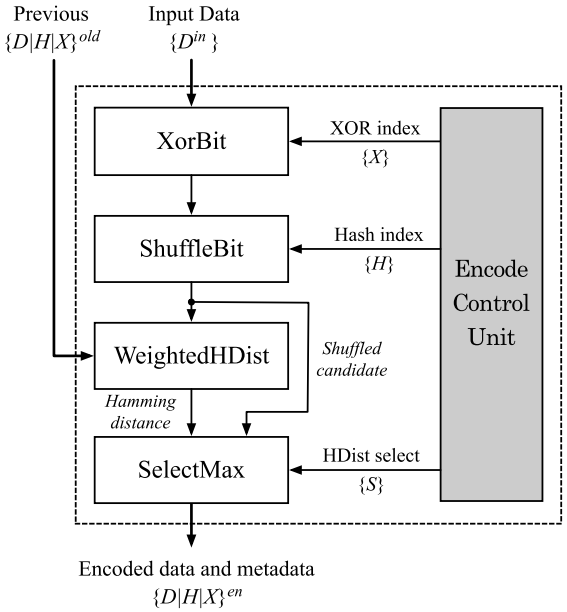

FIGURE 6. Block diagram of the proposed bit-shuffle encoder in MIMO-VLC. The white boxes indicate the modules of the encoder, and the gray box represents the control unit.

bit width and transmission rate. As a result, we can obtain the encoded data and metadata from the input data and a previously encoded output by repeatedly executing the four modules, excluding the controller.

First of all, the input data D^{in} is XORed with the XOR pattern selected by the XOR index X in the XorBit module. Second, the XORed input data is bit-shuffled with the hash index H in the ShuffleBit module. Then, the weighted Hamming distance of the shuffled candidate against the previously encoded output $\{D|H|X\}^{old}$ is calculated in the WeightedHDist module. Finally, the SelectMax module receives the shuffled candidate and its Hamming distance as inputs and selects one candidate with the maximum Hamming distance by comparing it to the previous candidates. The encoding process begins as the X, H, and S signals are initialized to all 0 in the first clock cycle. We also repeat these four steps N times when the input data is N-bit and finally return the shuffled candidate of the max Hamming distance as the encoded output. It means that we need N clock cycles to obtain the output for the N-bit input data.

Figure 7 illustrates the process that the encoder produces four different shuffled candidates by changing the hash index from 0 to 3 when encoding the 4-bit input data 1010. The XOR index is fixed at 0, and the previously encoded output is assumed to be 10101011 in this example. In other words, Figure 7 presents how the encoder repeatedly performs the inner loop body as h increases from 0 to 3 when p is 0, $\{D|M\}^{old}$ is 10101011, and D^{in} is 1010 in Algorithm 1. The input data is first transformed to 1010 after XORing with the first XOR pattern 0000. Then, the code selections are performed iteratively four times, i.e., #1, #2, #3, and #4, by employing the ShuffleBit, WeightedHDist, and SelectMax modules for the hash index of 0 to 3 in sequence.

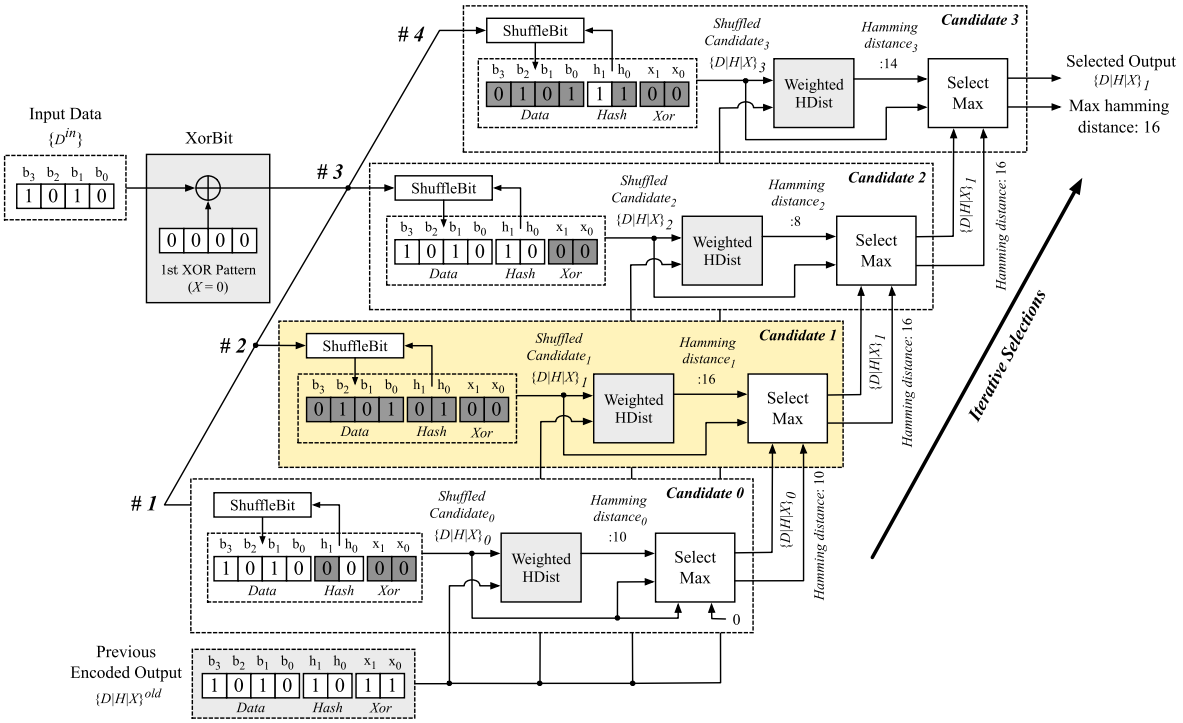

FIGURE 7. Example of iterative selection in our proposed bit-shuffle encoding for MIMO-VLC when the 4-bit input data is 1010, the previously encoded output is 10101011, and the XOR index is 0. The dark gray boxes indicate which bits are flipped against the previously encoded output. The yellow box also does the shuffled candidate 1 with the maximum weighted Hamming distance of 16 when the weights for the data, hash, and XOR bits are 1, 2, and 4, respectively.

In the first iteration, we shuffle the XORed input data into 1010 by applying the hash index 0 and get the first shuffled candidate 10100000, i.e., $\{D|H|X\}_0$, by concatenating the metadata bits 0000. We can find that the h_1, x_1, and x_0 bits of the shuffled candidate differ from the corresponding bits of the previously encoded output $\{D|H|X\}^{old}$. Thus, in this example, we obtain the weighted Hamming distance of 10 by adopting different weights of 1, 2, and 4 for the data, hash, and XOR bits, respectively. The first candidate is then selected immediately by comparing to itself in the SelectMax module and transferred to the second iteration. In the second iteration, we obtain the shuffled candidate of 01010100 using the hash index of 1. Its weighted Hamming distance is 16, i.e., $4 \times 1 + 2 \times 2 + 2 \times 4$, because all the bits are flipped compared with the previously encoded output. 16 is greater than the Hamming distance of 10 from the previous iteration so that the current shuffled candidate, i.e., $\{D|H|X\}_1$, and its Hamming distance of 16 are passed to the next iteration. Next, the third iteration produces the shuffled candidate of 10101000, and its hamming distance is eight since only the two XOR bits are different from $\{D|H|X\}^{old}$. Therefore, $\{D|H|X\}_1$ and 16 are selected and returned to the fourth iteration since eight is less than the maximum Hamming distance of 16. Finally, we get the fourth shuffled candidate of 01011100, i.e., $\{D|H|X\}_3$, by combining the shuffled data of 0101 for the hash index of 3 with its metadata. The seven bits of this candidate are different from those of the previously encoded output, so the weighted Hamming distance is calculated as 14. Since 14 is less than the maximum Hamming distance of 16 from the previous iteration, the output candidate and Hamming distance are selected as $\{D|H|X\}_1$ and 16, respectively. Note that we can find a shuffled candidate of the maximum Hamming distance by also performing such iterative selections for the XOR indexes of 1, 2, and 3. As a result, $\{D|H|X\}_1$ becomes the encoded output in this example. It is entirely inverted for every bit of the previously encoded output, although the input data bits are the same as those of the previous output. It means that we can maintain a constant dimming level on each bit position using the proposed bit-shuffle coding in MIMO-VLC.

Figure 8 presents a hardware design of our proposed bit-shuffle decoder for MIMO-VLC. The decoder architecture is very simple and consists of only two modules: ShuffleBit and XorBit. It is designed to perform bit-shuffling and XORing in order for returning the decoded data D^{de} from the encoded data and metadata $\{D|H|X\}^{en}$. The decoder design is significantly lightweight; thus, we can easily apply the bit-shuffle coding technique to low-power MIMO-VLC receivers in Internet of Thing environments.

B. HARDWARE MODULES

Figure 9 presents the detailed circuit design of the four hardware modules employed in the encoder and decoder.

First of all, Figure 9(a) shows the XorBit module that uses a bit-pattern table containing four different N-bit bit sequences. The module returns the XORed input data by XORing, the N-bit input data with the bit-pattern selected by a 4-to-1 multiplexer (MUX) using an XOR index.

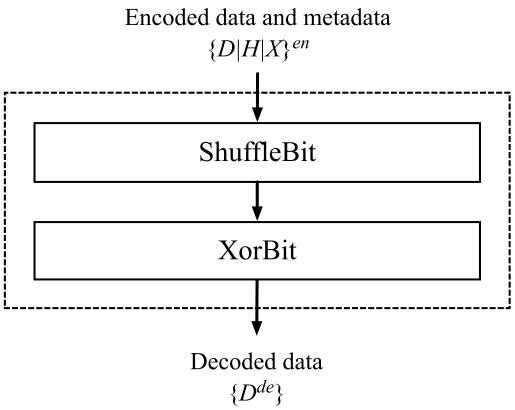

FIGURE 8. Decoder design in the proposed bit-shuffle coding for MIMO-VLC.

The more the module employs different bit-patterns, we can further diversify the XORed input data to make the dimming level constant. However, to reduce the hardware resource consumption while maintaining the constant dimming level, the current hardware implementation only adopts the four bit-patterns selectable with a 2-bit XOR index.

Second, the ShuffleBit module produces the N-bit shuffled data by shuffling all bits of the N-bit input data using a hash index as shown in Figure 9(b). The ShuffledBit module uses N instances of the shuffling bit position (SBP) module, which comprises an N-to-1 MUX and a $\log_2$N-bit XOR gate. The XOR gate is used to calculate the shuffled bit position from the original bit position i using the hash index. Thus, the i-th SBP outputs the i-th bit of the shuffled data by selecting the input data bit at the shuffled bit position.

Third, Figure 9(c) shows that the WeightedHDist module returns the weighted Hamming distance of a shuffled candidate against the previously encoded output. The module employs three hamming distance (HDist) modules that perform XORing and counting the number of 1s for the data, hash, and XOR bits. The weighted Hamming distance is calculated by summing the results of multiplying three different weights, i.e., α_1, α_2, and α_3, to the outputs of the HDist modules, respectively.

Finally, Figure 9(d) shows the SelectMax module, which compares the input shuffled candidate to the shuffled candidate of the max Hamming distance stored in the registers, i.e., Reg, at the previous clock cycle and returns one of a greater Hamming distance. By assigning the HDist select signal generated from the encoder's control unit to the two left MUXs, we can choose to use either the initial values or the registered shuffled candidate and Hamming distance for the comparison. In other words, if the select signal to the left above MUX is 0, an initial value of 0 is applied to the "Greater than" comparator so that it selects the new input Hamming distance but not the registered one. Otherwise, the registered max Hamming distance is selected for comparison with the new input Hamming distance. The input shuffled candidate is also selected by the left below MUX when the select signal is 0; thus, it is determined as the greater one regardless of the comparator's output. On the other hand, the MUX outputs the registered shuffled candidate if the signal is set to 1 so that the comparison result determines the max shuffled candidate to return. In this way, one "Greater than" comparison is performed every clock cycle, and the output max Hamming distance and shuffled candidate are stored in the registers synchronously to the clock and used for the comparison of the next clock cycle. With the overall process of hardware resource optimization, it is possible to implement a compact MIMO-VLC system that can expand the area of massive M2M communication services.

IV. PERFORMANCE ANALYSIS

In this section, we evaluate the performance of our proposed bit-shuffle block coding in MIMO-VLC compared to existing RLL codings, i.e., Manchester, 4B6B, and 8B10B, which are included in the IEEE 802.15.7 standard [7].

A. EXPERIMENTAL SETUP

As performance metrics, we evaluated the transmission overhead and dimming level by producing codeword sequences via a Monte-Carlo simulation with 50,000 runs. To analyze the inherent characteristics and performance presented by bit-shuffle block coding, channel coding was not considered. In addition, we investigated the hardware resource consumption, i.e., area and power, of the proposed bit-shuffle block coding and made a comparison with the existing RLL coding schemes for MIMO-VLC. To do this, we implemented the encoders and decoders of the 8-, 16-, 32-, and 64-bit input data for all the coding techniques. Here, the encoders and decoders were designed in Verilog Hardware Description Language (HDL). Synopsys Design Compiler was used to synthesize the encoders and decoders in a 32-*nm* CMOS technology to obtain the area and power consumption [22]–[24].

B. TRANSMISSION OVERHEAD

Transmission efficiency is defined as the number of information bits divided by the number of transmitted bits. A higher transmission efficiency means faster transmission. To verify the transmission efficiency, we compared the codeword lengths of the proposed bit-shuffle block coding to those of the conventional Manchester, 4B6B, and 8B10B codes. Table 1 shows the transmission overhead of the RLL codes for input bit widths of 8, 16, 32, and 64 bits. The transmission overhead was calculated by dividing the required redundant bits by the transmitted bits. As can be seen, the proposed scheme produces considerably shorter codewords than Manchester coding. When N bit input data are encoded, Manchester coding causes a severe reduction in the data transmission rate, because it produces a $2N$-bit encoded output, whereas the proposed scheme returns $(N + log_2\ N + 2)$-bit encoded output. This length comprises N bit encoded data, a $log_2\ N$-bit hash, and two XOR bit. In addition, the transmission efficiency of the bit-shuffle block code increases as the bit width increases. For bit widths of 8 and 16 bits, the bit-shuffle block code exhibits a codeword length that

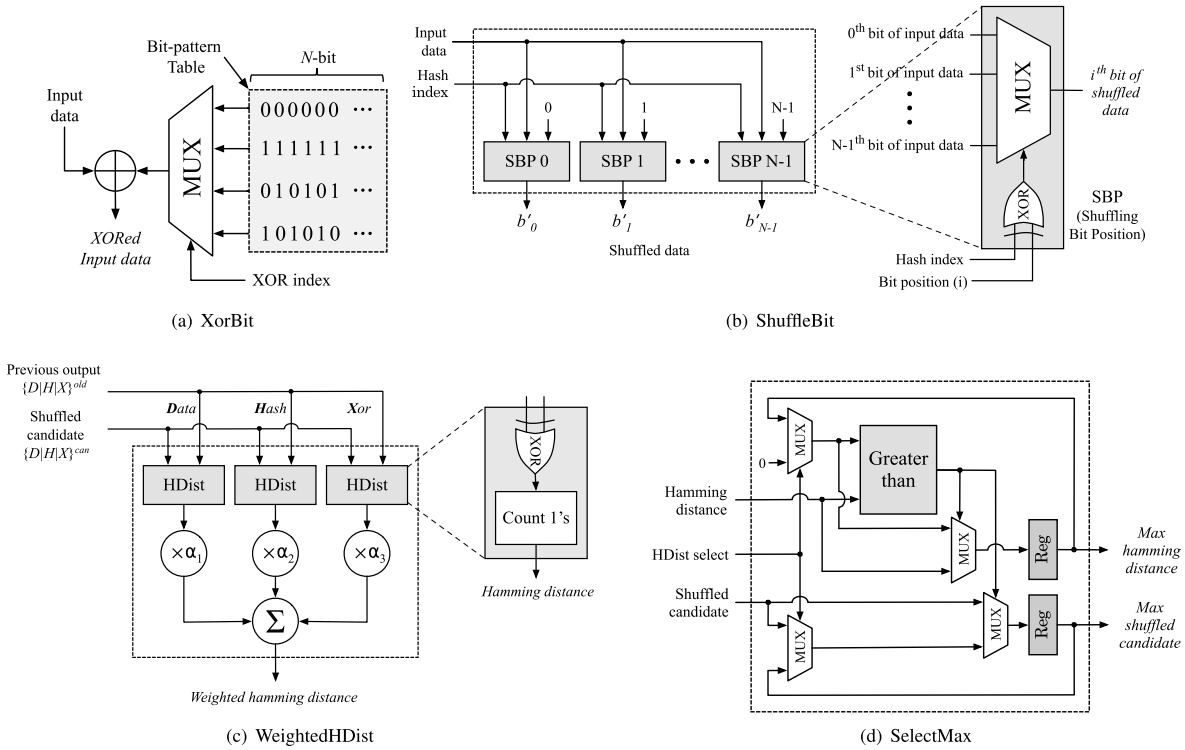

FIGURE 9. Hardware modules of the bit-shuffle encoder and decoder for MIMO-VLC, where the input data width is N-bit.

is comparable to the 4B6B and 8B10B schemes. However, when the bit width is greater than 32 bits, the proposed scheme provides a significantly smaller codeword length than the compared methods. In particular, for a bit width of 64 bits, the transmission overhead of the proposed method is only 11.1%, which is approximately the one-half and one-third of the 8B10B and 4B6B codes, respectively. This is due to the fact that only $log_2 N$-wise independent hash bits are added to data length N due to the nature of the Omega network; thus, high transmission efficiency can be guaranteed for long data bits. Generally, in a transmission service that requires a high data rate, a large amount of data is contained in a single packet by increasing the payload length. Increasing the bit width can be equated to increasing the payload length in terms of data transmission, and this fits the purpose of the MIMO system, which is useful when transmitting large-sized data.

To observe the effect of bit shuffle block coding on the probability of bit error, the BER graph is shown in Figure 10. As shown in the figure below, the BER performance of the proposed scheme is similar to that of other RLL codes (4B6B, 8B10B, and Manchester codes). Despite the excellent resource-optimized hardware and DC balancing performance of the bit shuffle block coding scheme, degradation of BER performance does not occur either the low or high signal-to-noise ratio (SNR) region.

C. HARDWARE RESOURCE CONSUMPTION

Table 1 summarizes the hardware performance of the encoders and decoders of the proposed and existing coding schemes. Since we adopt the amount of hardware resource

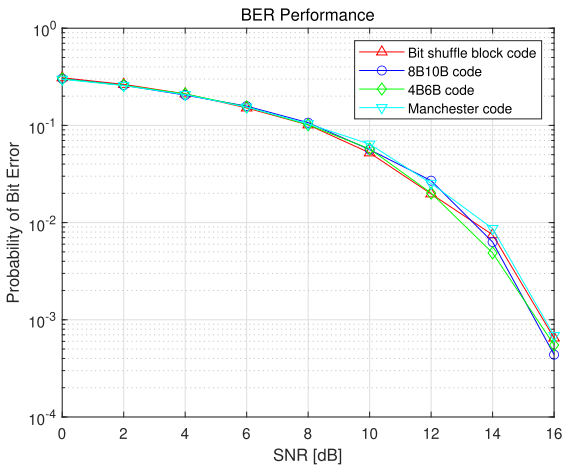

FIGURE 10. Performance comparison of the bit error rate of various RLL codes.

usage against the transmission efficiency as the cost of a code design for MIMO-VLC, here we discuss that the bit-shuffle coding is superior to other schemes in terms of the cost.

The system clock frequency needs to change according to the input bit size and transmission data rate to perform our encoding algorithm in MIMO-VLC, whereas the frequency for decoding is consistent. Our encoder design requires a relatively higher system clock frequency than the existing schemes since it adopts iterative execution to reduce its hardware resource consumption as mentioned in Section III-A. Thus, we could achieve significant performance improvements in terms of area and power, compared to 4B6B and 8B10B. The clock frequencies and target data rates of the

TABLE 1. Design parameters and performance results.

Design	Input bit width	Coding	System clock (MHz)	Data rate (Mbps)	Transmission overhead (%)	Area (μm^2)	Power (μW)
Encoder	8 bits	Manchester	1	8	50.0	104.36	12.18
		4B6B	2	8	33.3	1632.35	144.78
		8B10B	4	8	20.0	3843.36	290.97
		Proposed	32	8	38.5	784.60	78.53
	16 bits	Manchester	1	16	50.0	204.44	22.89
		4B6B	2	16	33.3	3157.81	281.42
		8B10B	4	16	20.0	7503.47	563.36
		Proposed	64	16	27.3	1489.28	157.80
	32 bits	Manchester	1	32	50.0	442.68	45.22
		4B6B	2	32	33.3	6193.13	555.29
		8B10B	4	32	20.0	15165.22	1115.20
		Proposed	128	32	17.9	2752.85	368.06
	64 bits	Manchester	1	64	50.0	896.36	89.94
		4B6B	2	64	33.3	12617.63	1107.50
		8B10B	4	64	20.0	31404.51	2232.10
		Proposed	256	64	11.1	5004.97	911.34
Decoder	8 bits	Manchester	1	8	50.0	91.47	10.57
		4B6B	1	8	33.3	2240.84	169.63
		8B10B	1	8	20.0	4729.93	310.08
		Proposed	1	8	38.5	201.41	17.07
	16 bits	Manchester	1	16	50.0	178.70	20.36
		4B6B	1	16	33.3	4390.91	330.78
		8B10B	1	16	20.0	9281.76	602.38
		Proposed	1	16	27.3	533.75	38.09
	32 bits	Manchester	1	32	50.0	405.02	39.21
		4B6B	1	32	33.3	8649.76	645.72
		8B10B	1	32	20.0	18671.63	1181.70
		Proposed	1	32	17.9	1186.81	87.89
	64 bits	Manchester	1	64	50.0	822.39	79.85
		4B6B	1	64	33.3	17571.81	1301.30
		8B10B	1	64	20.0	38360.87	2366.90
		Proposed	1	64	11.1	2683.35	202.49

bit-shuffle encoder as well as the frequencies of the compared coding schemes to satisfy the same data rate are also shown in Table 1. Note that we synthesized each encoder and decoder to operate at the corresponding clock frequency in Table 1, and then we obtained the area and power values.

The results clearly demonstrate that the proposed encoder and decoder outperform the 4B6B and 8B10B encoders and decoders in terms of both area and power for all input data sizes. Specifically, the proposed bit-shuffle encoder reduces the area by 79.59%, 80.15%, 81.85%, and 84.06%, respectively, compared to the 8B10B encoders with input data sizes of 8, 16, 32, and 64 bits. In addition, the proposed encoder consumes 73.01%, 71.99%, 67.00%, and 59.17% less power for 8-, 16-, 32-, and 64-bit inputs, respectively, compared to the 8B10B encoder. When compared against the 4B6B encoder, the proposed design achieves area reductions of 51.93% $\sim$ 60.33% and power reductions of 17.71% $\sim$ 45.76%, respectively, at the given input bit widths.

A comparison of the proposed decoder to the 8B10B decoder shows that the proposed decoder has up to 23.48, 17.39, 15.73, and 14.30 times greater area efficiency at 8-, 16-, 32-, and 64-bit inputs, respectively. In addition, our decoder achieves greater than 90% power reduction compared to the 8B10B decoder for all input bit widths. Importantly, the hardware performance comparison reveals that the proposed bit-shuffle coding scheme allows more area and power improvement in decoding than encoding for MIMO-VLC. This makes the bit-shuffle coding more attractive and beneficial for unidirectional VLC applications because the decoder significantly impacts overall system performance. Generally, resource-constrained receivers are widely adopted in many light-weight VLC applications and those receivers only include the decoder. Unfortunately, the proposed encoder and decoder consume more area and power than the Manchester counterparts. However, we have proved the proposed encoder and decoder are more applicable to MIMO-VLC because Manchester coding incurs significantly more transmission overheads than our coding scheme; thus, Manchester coding demonstrates a poor transmission efficiency. In addition, the transmission efficiency improvement of the proposed coding compared to Manchester coding increases at the higher input data sizes. This may hinder Manchester coding from being adopted in MIMO-VLC applications that require fast data transmission although Manchester coding requires less area and power than the proposed scheme.

D. DIMMING LEVEL

Figure 11 exhibits the variance of dimming level for each bit position in the output bit sequence generated by applying the proposed bit-shuffle block encoding to 8, 16, 32, and 64-bit input data, respectively. The binary signal of each bit position is expressed by the light intensity of each LED at the transmitter. The dimming level was calculated using the

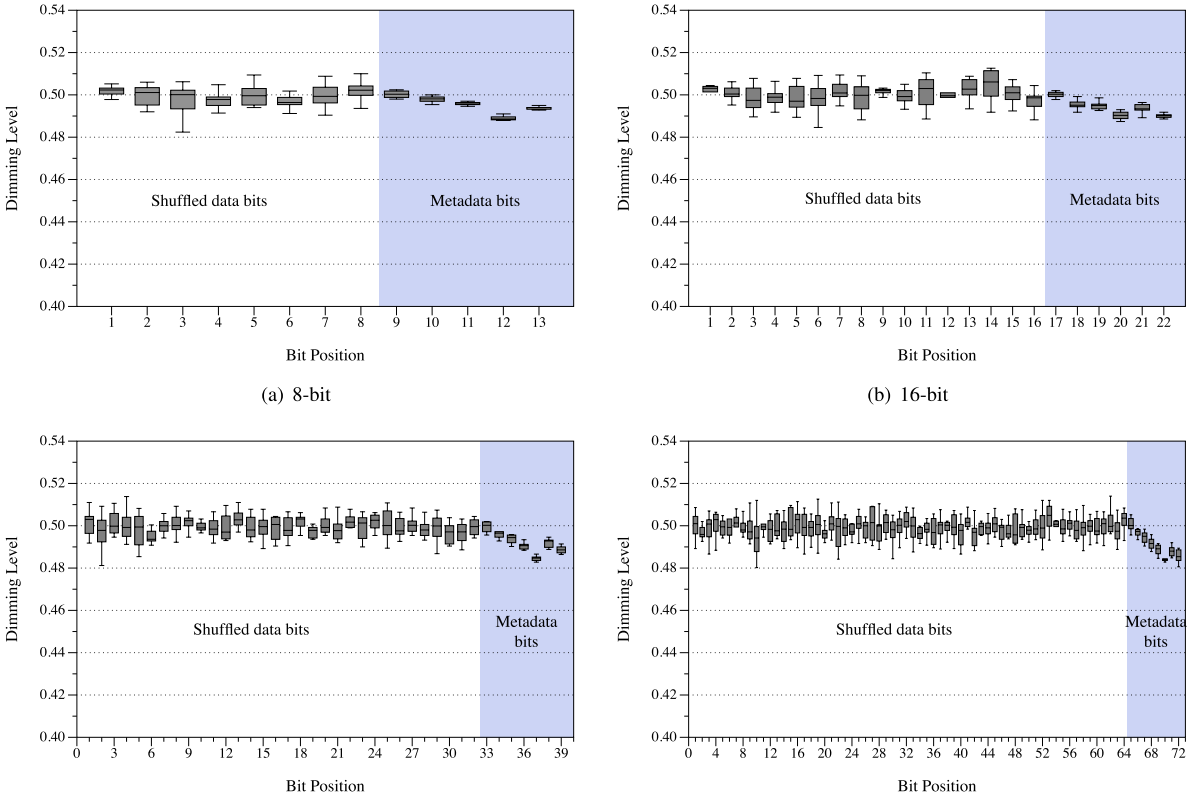

FIGURE 11. Dimming level variance at each bit position in the encoded output for different input bit widths, i.e., 8, 16, 32, and 64. The left white part of each graph shows the dimming level of the shuffled data bits, and the right colored part does that of the metadata bits.

cumulative number of 0's and 1's for each bit position by encoding Monte-Carlo random input data 5,000 times. We also obtained the variance by repeating the measurement of the dimming level 10 times. As a result, we found that the dimming level at each bit position for all input bit widths was stably distributed between 0.48 and 0.52. Note that the slight light changes around the 50% dimming level can be measured by instruments but are difficult to perceive by the human eye. In addition, the proposed scheme significantly improves the transmission overhead and hardware resource consumption, so this minor dimming level change can be acceptable for the practical use. Although the weighted Hamming distance is employed for the encoding, there is a large difference in the dimming level between the shuffled data and metadata bits for the 32 and 64-bit input data. The dimming level of shuffled data bits is distributed from 0.5 with an error range of ± 0.01, whereas that of the metadata bits is biased between 0.48 and 0.5 rather than a relatively small variance. In this experiment, we assigned values of 1, 3, and 4 to the weights of α_1, α_2, and α_3, respectively, for all input bit widths to minimize hardware resource consumption for the weighted Hamming distance calculation. Thus, we assure that it can be relieved by adopting different weight values for different input widths. Alternatively, it can be resolved by differential biasing in LED emission. These solutions will be considered in future research.

V. CONCLUSION

Due to the increasing demand for VLC devices with small hardware and the potential to increase the data rate using multiple LEDs, the implementation of resource-constrained MIMO-VLC systems is of significant interest. To develop an RLL code technique that satisfies these requirements, we have proposed a bit-shuffle block coding scheme that comprises a resource-optimized encoder and decoder structure. The use of the Omega network in the encoder reduces transmission overhead and significantly reduces hardware overhead for implementation of compact VLC devices. The dynamic code generation based on the weighted Hamming distance provides DC balance on multiple LEDs, which retains the original purpose of illumination. The experimental results have proved that the proposed scheme can realize resource-optimized RLL coding for MIMO-VLC systems and that the proposed scheme outperforms conventional schemes in terms of hardware overhead and transmission efficiency.

REFERENCES

[1] H. Haas, L. Yin, Y. Wang, and C. Chen, "What is LiFi?" *J. Lightw. Technol.*, vol. 34, no. 6, pp. 1533–1544, Mar. 15, 2016.

[2] P. H. Pathak, X. Feng, P. Hu, and P. Mohapatra, "Visible light communication, networking, and sensing: A survey, potential and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2047–2077, 4th Quart., 2015.

[3] L. D. Tamang and B. W. Kim, "Optical camera communication for vehicular applications: A survey," *IEIE Trans. Smart Process. Comput.*, vol. 10, no. 2, pp. 136–145, Apr. 2021.
[4] D. Karunatilaka, F. Zafar, V. Kalavally, and R. Parthiban, "LED based indoor visible light communications: State of the art," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 3, pp. 1649–1678, Aug. 2015.
[5] H. Haas, "LiFi is a paradigm-shifting 5G technology," *Rev. Phys.*, vol. 3, pp. 26–31, Nov. 2017.
[6] L. I. Albraheem, L. H. Alhudaithy, A. A. Aljaser, M. R. Aldhafian, and G. M. Bahliwah, "Toward designing a Li-Fi-based hierarchical IoT architecture," *IEEE Access*, vol. 6, pp. 40811–40825, 2018.
[7] IEEE Standard for Local and Metropolitan Area Networks—Part 15.7, *Short-Range Wireless Optical Communication Using Visible Light*, IEEE Standard 802.15.7, 2011.
[8] K. Kim, K. Lee, and K. Lee, "Zero reduction codes for efficient transmission and enhanced brightness in visible light communication," *IET Optoelectron.*, vol. 11, no. 3, pp. 108–113, Jun. 2017.
[9] S. Rajagopal, R. D. Roberts, and S.-K. Lim, "IEEE 802.15.7 visible light communication: Modulation schemes and dimming support," *IEEE Commun. Mag.*, vol. 50, no. 3, pp. 72–82, Mar. 2012.
[10] C. E. Mejia, C. N. Georghiades, M. M. Abdallah, and Y. H. Al-Badarneh, "Code design for flicker mitigation in visible light communications using finite state machines," *IEEE Trans. Commun.*, vol. 65, no. 5, pp. 2091–2100, May 2017.
[11] K. Kim, K. Lee, and K. Lee, "Appropriate RLL coding scheme for effective dimming control in VLC," *Electron. Lett.*, vol. 52, no. 19, pp. 1622–1624, Sep. 2016.
[12] C. He, S. Cincotta, M. M. A. Mohammed, and J. Armstrong, "Angular diversity aperture (ADA) receivers for indoor multiple-input multiple-output (MIMO) visible light communications (VLC)," *IEEE Access*, vol. 7, pp. 145282–145301, 2019.
[13] Y. Han, Y. Kim, and B. W. Kim, "Bit-shuffle coding for flicker mitigation in visible light communication," *IEEE Access*, vol. 7, pp. 150271–150279, 2019.
[14] X. Guo and N. Chi, "Superposed 32QAM constellation design for 2×2 spatial multiplexing MIMO VLC systems," *J. Lightw. Technol.*, vol. 38, no. 7, pp. 1702–1711, Apr. 1, 2020.
[15] H. B. Cai, J. Zhang, Y. J. Zhu, J. K. Zhang, and X. Yang, "Optimal constellation design for indoor 2×2 MIMO visible light communications," *IEEE Commun. Lett.*, vol. 20, no. 2, pp. 264–267, Feb. 2016.
[16] Y. Zhai, H. Chi, J. Tong, and J. Xi, "Capacity maximized linear precoder design for spatial-multiplexing MIMO VLC systems," *IEEE Access*, vol. 8, pp. 63901–63909, 2020.
[17] D. Zheng, H. Zhang, and J. Song, "Spatial multiplexing MIMO visible light communications with densely distributed LEDs and PDs," *IEEE Photon. J.*, vol. 12, no. 5, Oct. 2020, Art. no. 7905807.
[18] C. Chen, Y. Yang, X. Deng, P. Du, and H. Yang, "Space division multiple access with distributed user grouping for multi-user MIMO-VLC systems," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 943–956, 2020.
[19] F. I. K. Mousa, N. Al Maadeed, K. Busawon, A. Bouridane, and R. Binns, "Secure MIMO visible light communication system based on user's location and encryption," *J. Lightw. Technol.*, vol. 35, no. 24, pp. 5324–5334, Dec. 15, 2017.
[20] Y. Yang, Z. Zeng, J. Cheng, and C. Guo, "Spatial dimming scheme for optical OFDM based visible light communication," *Opt. Exp.*, vol. 24, no. 26, pp. 30254–30263, Dec. 2016.
[21] H. Wang and S. Kim, "Optimal constellation design for indoor 2×2 MIMO visible light communications," *IEEE Photon. Technol. Lett.*, vol. 29, no. 19, pp. 1651–1654, Feb. 2017.
[22] S. Yu-yun, H. Qing-sheng, and H. Liming, "A 0.18μm pipelined 8B10B encoder for a high-speed SerDes," in *Proc. IEEE 12th Int. Conf. Commun. Technol.*, Nov. 2010, pp. 1039–1042.
[23] P. Nannipieri, D. Davalle, and L. Fanucci, "A novel parallel 8B/10B encoder: Architecture and comparison with classical solution," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 101, no. 7, pp. 1022–1120, 2018.
[24] H. Bhatnagar, *Advanced ASIC Chip Synthesis: Using Synopsys Design Compiler Physical Compiler and Prime Time*, 2nd ed. Norwell, MA, USA: Kluwer, 2002.

YOUNGSUN HAN (Member, IEEE) received the B.S. and Ph.D. degrees in electrical engineering from Korea University, Seoul, South Korea, in 2003 and 2009, respectively. He was a Senior Engineer with System LSI, Samsung Electronics, Suwon, South Korea, from 2009 to 2011. He was an Assistant/Associate Professor with the Department of Electronic Engineering, Kyungil University, Gyeongsan-si, South Korea, from 2011 to 2019. He is currently an Associate Professor with the Department of Computer Engineering, Pukyong National University, Busan, South Korea. His research interests include compiler construction, microarchitecture, high-performance computing, and SoC design.

SANGHYEON LEE is currently pursuing the integrated B.S. and M.S. degree from the Department of Computer Engineering, Pukyong National University, Busan, South Korea. His research interests include high-performance computing, memory systems, and SoC design.

BYUNG WOOK KIM (Member, IEEE) received the B.S. degree from the School of Electrical Engineering, Pusan National University, Pusan, South Korea, in 2005, and the M.S. and Ph.D. degrees from the Department of Electrical Engineering, KAIST, Daejeon, South Korea, in 2007 and 2012, respectively. He was a Senior Engineer with the Korea Electrotechnology Research Institute, Changwon-si, South Korea, from 2012 to 2013. He was an Assistant Professor with the School of Electrical and Railway Engineering, Kyungil University, Gyeongsan-si, South Korea, from 2013 to 2016, and the Department of ICT Automotive Engineering, Hoseo University, from 2016 to 2019. He is currently an Assistant Professor with the Department of Information and Communication Engineering, Changwon National University, Changwon-si. His research interests include visible light communications, machine learning, and image processing.

YONGTAE KIM (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, South Korea, in 2007 and 2009, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, USA, in 2013. From 2013 to 2018, he was a Software Engineer with Intel Corporation, Santa Clara, CA, USA. Since 2018, he has been with the School of Computer Science and Engineering, Kyungpook National University, Daegu, South Korea, where he is currently an Assistant Professor. His research interests include energy efficient integrated circuits and systems, particularly, neuromorphic computing and approximate computing, and new memory devices and architectures.