Efficient Hardware Implementation of Encoder and Decoder for Golay Code

Satyabrata Sarangi and Swapna Banerjee

***Abstract*— This brief lays out cyclic redundancy check-based encoding scheme and presents an efficient implementation of the encoding algorithm in field programmable gate array (FPGA) prototype for both the binary Golay code (G_{23}) and extended binary Golay code (G_{24}). High speed with low-latency architecture has been designed and implemented in Virtex-4 FPGA for Golay encoder without incorporating linear feedback shift register. This brief also presents an optimized and low-complexity decoding architecture for extended binary Golay code (24, 12, 8) based on an incomplete maximum likelihood decoding scheme. The proposed architecture for decoder occupies less area and has lower latency than some of the recent work published in this area. The encoder module runs at 238.575 MHz, while the proposed architecture for decoder has an operating clock frequency of 195.028 MHz. The proposed hardware modules may be a good candidate for forward error correction in communication link, which demands a high-speed system.**

***Index Terms*— Architecture, decoder, encoder, field programmable gate array (FPGA), Golay code.**

I. Introduction

The Golay code was presented in [1] to address error correcting phenomena. The binary Golay code (G_{23}) is represented as (23, 12, 7), while the extended binary Golay code (G_{24}) is as (24, 12, 8). The extended Golay code has been used extensively in deep space network of JPL-NASA as well as in the Voyager imaging system [5]. In addition, Golay code plays a vital role in different applications like coded excitation for a laser [6] and ultrasound imaging due to the complete sidelobe nullification property of complementary Golay pair. All these applications need generation of Golay sequence, which is fed as trigger to the laser modules. However, for generating Golay code an automatic pattern generator is used, which is of very high cost. To combat this problem, a hardware module programmed to yield a Golay encoded codeword may be used. Golay decoder is used extensively in communication links for forward error correction. Therefore, a high speed and high throughput hardware for decoder could be useful in communication links for forward error correction.

Literature surveys were conducted in [2]–[4], which deal with encoding methods for Golay code, but these are not suitable for hardware implementation due to complexity of the algorithms. Weng and Lee [5] invented an encoding method for Golay code comprising of a linear feedback shift register (LFSR), an overall parity bit generator, a clock doubler, a five bit counter, and some switching logic. Conventionally, LFSR-based cyclic redundancy check (CRC) generation scheme [7]–[9] is preferred for hardware implementation of encoding process, but it has several drawbacks like high latency and less throughput, hence making it unsuitable for high-speed applications.

Over the years, many decoding techniques [12]–[14], [17]–[19], [21], [22] for Golay code have been presented. Based on these algorithms several hardware architectures [10], [15], [16], [20], [23], [24] have been proposed for decoder. Most recently, Alimohammad and Fard [24] implemented a novel architecture for Golay decoder based on imperfect maximum likelihood decoding (IMLD) scheme, which is also considered in this brief for designing hardware.

Manuscript received October 21, 2013; revised May 10, 2014; accepted July 16, 2014. Date of publication August 22, 2014; date of current version August 21, 2015.

The authors are with the Department of Electronics and Electrical Communication Engineering, IIT Kharagpur, Kharagpur 721302, India (e-mail: satyabratasarangi.01@gmail.com; swapna@ece.iitkgp.ernet.in).

Digital Object Identifier 10.1109/TVLSI.2014.2346712

The rest of this brief is organized as follows. Section II gives an introduction on Golay code, encoding, and decoding schemes. The proposed algorithm, architecture for encoding Golay code and the hardware implementation results have been illustrated in Section III, while Section IV depicts the proposed hardware architecture for implementing decoder using IMLD algorithm and the results of hardware implementation of proposed decoder module. Finally, conclusions are drawn in Section V.

II. Golay Code

The binary Golay code is represented as (23, 12, 7) that depicts that length of codeword is 23 bits, while message is of 12 bits and the minimum distance between two binary Golay codes is 7. A Galois field (GF) is necessary to construct binary codes. In general, binary field is denoted by GF (2), which supports different binary arithmetic operations. The generation of coding sequence needs a generator polynomial. The possible generator polynomials [13] over GF (2) for Golay (23, 12, 7) code are $x^{11}+x^{10}+x^6+x^5+x^4+x^2+x^1$ and $x^{11}+x^9+x^7+x^6+x^5+x^1+1$. In this brief, AE3h is considered as the characteristic polynomial. The remainder of the long division gives the required check bits. Finally, appending the generated check bits with the message gives us the extended Golay codeword.

The extended Golay code (24, 12, 8) can be generated by appending a parity bit with the binary Golay code or using a generator matrix G, which is defined as $[I, B]$, where I denotes an identity matrix of order 12

$$B = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}. \quad (1)$$

III. Algorithm, Proposed Architecture, and Comparison of Results for Golay Encoder

A. Proposed Algorithm for Encoder

The steps required to accomplish the encoding procedure are enlisted as follows.

1) A characteristic polynomial $G(x)$ is chosen for check bits generation.
2) 11 zeros are appended to the right of message $M(x)$, such that resultant polynomial $P(x)$ participates in long division process with $G(x)$.
3) The remainder bits except the most significant bit (MSB) resulted at the end of the division operation are the check bits for G_{23}. Appending check bits with the message gives us the encoded Golay (23, 12, 7) codeword.
4) A parity bit is added to convert the binary Golay code into extended binary Golay code (24, 12, 8). If the weight of binary Golay code is even, then parity bit 0 is appended, otherwise 1 is appended.

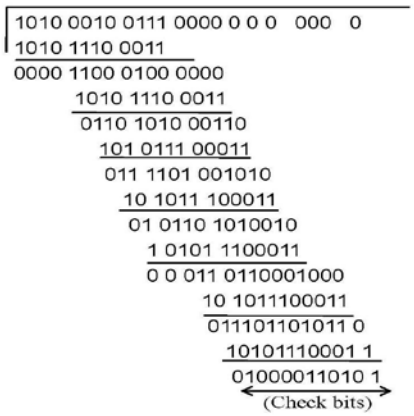

Fig. 1. Example of check bits generation.

The proposed encoder algorithm clearly follows the basic CRC generation process and includes a method for converting binary Golay code to extended Golay code before proceeding for designing architecture.

An example of Golay code word generation based on the above mentioned algorithm is shown in Fig. 1. Let us say, the message to be encoded is A27h. Hence, $M(x)$ = A27h and $P(x)$ in binary format is represented as 1010 0010 0111 0000 0000 000. Finally, the generated check bits in hexadecimal format are 435h. Hence, the encoded codeword for the message bits (A27h) is A27435h. This is a binary Golay code word. To convert it into an extended Golay code, a parity bit 1 is appended, as weight of A27435h is 11 (odd). Finally, the generated Golay (24, 12, 8) codeword is (1010 0010 0111 1000 0110 101 1). The validity of the generated Golay code can be tested by measuring the weight of the code.

The weight of every G_{24} code should be a multiple of four and greater than equal to eight. The generated code word shown in the example has a weight of 12, which is a multiple of four and greater than eight. Hence, the generated code is valid and thus the algorithm.

B. Proposed Architecture for Binary Golay Code and Extended Golay Code

In this section, optimized architectures for encoder are shown in Figs. 2 and 3. The whole architecture is partitioned into two parts, one for G_{23} generation and the other showing conversion of G_{23} to G_{24}.

In each step during polynomial division, simply binary XOR operation occurs for modulo-2 subtraction. The residual result obtained at each step during the division process is circularly left shifted by number of leading zeros present in the result. A 12:4 priority encoder is used to detect efficiently the number of leading zeros before first 1 bit in the residual result in each step. A circular shift register is used to shift the intermediate result by the output of priority encoder. A 2:1 multiplexer is used to select the initial message or the circularly shifted intermediate result. The control signal used for the multiplexer and the controlled subtractor is denoted as p, which is bit wise OR operation of priority encoder output.

A controlled subtractor is used for loop control mechanism. Initially, one input of subtractor is initialized with 11, which is the number of zeros appended in the first step of the long division process and it gets updated with the content of $R7$ register due to multiplexer selection after each iteration. The output of the priority encoder is the other input to the subtractor. After the final iteration, the result of subtractor is zero, which is stored in register $R7$. The register $R6$ is loaded when the content of register $R7$ becomes zero, which depicts the end of the division process and hence the check bits generation process. The Ld control signal controls the loading of the $R3$ [10:0]

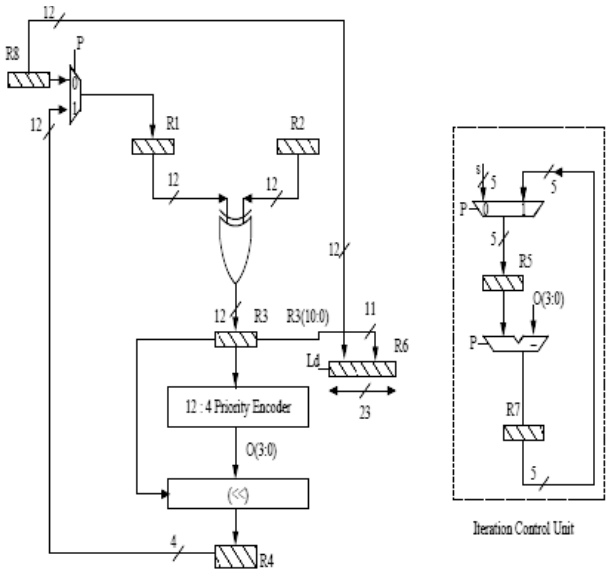

Fig. 2. Architecture for generating binary Golay code.

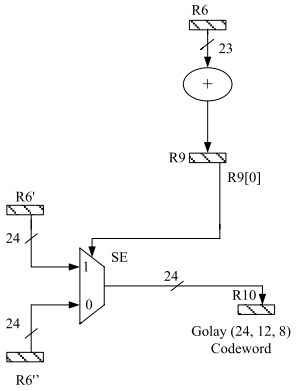

Fig. 3. Architecture for conversion of binary Golay code into extended Golay code.

contents at final iteration into $R6$. Logically, Ld = (!($R7$ [4]) & (|($R7$))), where |($R7$) denotes bit-wise OR of $R7$ register. When Ld becomes zero, $R3$ [10:0] is mapped to $R6$ [10:0] and $R8$ contents are mapped into $R6$ [22:11]. In this way, the content of register $R6$ gives the encoded codeword.

Hardware architecture for converting binary Golay code to extended Golay code has been shown in Fig. 3. Weight of the binary Goly code is stored in $R9$. Register $R6'$ contains content of $R6$ appended with 0 and $R6''$ contains content of $R6$ appended with 1. A 2:1 multiplexer is used to select either $R6'$ or $R6''$ depending on $R9$ [0], which acts as the select line for the multiplexer. Finally, $R10$ contains Golay (24, 12, 8) encoded codeword.

C. Hardware Implementation Results of Encoder Module

The proposed algorithm for encoder has been verified using MATLAB R2009b tool and the hardware architecture shown in Figs. 2 and 3 have been implemented in XC4vlx160-12-ff1148 field programmable gate array (FPGA) using Xilinx ISE tool. The synthesized frequency for encoder module is 238.575 MHz. The proposed architecture results in a codeword per clock cycle instead of single check bit per clock cycle, hence making a high data rate enabled system. In addition, this architecture utilizes 187 look up tables (LUTs) and 103 slices out of 135 168 each. The synthesis results obtained from Synopsys design compiler using

TABLE I
COMPARISON OF THE PROPOSED ENCODER ARCHITECTURE CONSIDERING LATENCY AND CLOCKING MECHANISM

Attribute	[5]	Proposed
Latency (Clock Cycles)	23	12(Maximum)
Clocking Mechanism	System Clock + Clock Doubler	System Clock

TABLE II
COMPARISON OF THE PROPOSED ENCODER ARCHITECTURE CONSIDERING LATENCY AND LUT UTILIZATION

Reference	LUT Utilization (%)	Latency(Clock cycles)
[7]	1.33	12
[8]	1.72	12
Proposed	0.14	12

0.18-μm UMC technology depict that total dynamic power consumed by the architecture is 963.52 μW, while total cell area is 12389.83 μm^2.

Table I depicts that the proposed architecture avoids any extra clocking mechanism unlike [5], which in turn reduces area and power. Another crucial point is that the proposed architecture has a latency of maximum 12 clock cycles instead of 23 as shown in [5].

Table II represents that the proposed architecture outperforms parallel CRC-11 circuits [7], [8] in terms of percentage of LUT utilization maintaining same latency. In addition, parallel implementation incurs significant increase in area (5%–10% per bit as shown in [7] and [8]), which should be avoided.

IV. ALGORITHM, PROPOSED ARCHITECTURE, AND COMPARISON OF RESULTS FOR GOLAY DECODER

A. Algorithm for Decoding Extended Golay Code

To decode the extended binary Golay code, IMLD algorithm is considered Algorithm 1 are [19], [24]. Assume that **W** is the received codeword, $\mathbf{b}_i$ denotes the ith row of **B** and **u** is the error pattern. Two syndromes (S and SB) are possible due to possibility of two parity check matrices (either $[I|B]$ or $[B|I]$). The pseudocode for the algorithm is stated in Algorithm 1.

B. Proposed Architecture for Decoding Extended Golay Code

1) Syndrome Measurement Unit: Syndrome **S** is evaluated performing multiplication of received codeword **cw** with parity check matrix **H**. Therefore, the logical expression for calculating MSB bit of syndrome vector is given as follows:

$$S[11] = cw\ [23]\ \text{XOR}\ cw\ [11]\ \text{XOR}\ cw\ [10]\ \text{XOR}\ cw\ [8]\ \text{XOR}\ cw\ [7]\ \text{XOR}\ cw\ [6]\ \text{XOR}\ cw\ [2]\ \text{XOR}\ cw\ [0].$$

Likewise, other bits of syndrome can be calculated following the above logic. The delay optimized structure for calculating S [11] bit of syndrome vector is shown in Fig. 4.

2) Weight Measurement Unit: The weight measurement unit primarily counts the number of binary 1 in the sequence, which can be efficiently done by the circuit shown in Fig. 5, which results in less critical path delay.

3) Calculation of S + b_i and SB + b_i: This unit calculates S + b_i and SB + b_i ($1 \le i \le 12$) by inverting some of the bits of registers S and SB or keeping those unchanged. The logical contents of S + b_1 is given by

$$S + b_1 = \{\sim S[11], \sim S[10], S[9], \sim S[8], \sim S[7], \sim S[6], S[5], S[4], S[3], \sim S[2], S[1], \sim S[0]\}$$

where, symbol $\sim$ denotes logical NOT.

Algorithm 1 IMLD Algorithm for Decoding Extended Golay Code

Syndrome $S = W * H$ is computed first.
If (wt (S)$\le$ 3)
  u= [S, 0];
Else
{
If (wt ($S + b_i$) $\le$ 2) for some row bi of B
  u= [$S+b_i$, e_i];
Else
{
Second Syndrome SB is computed.
If (wt (SB) $\le$ 3)
  u= [0, SB];
Else
  {
  If (wt ($SB + b_i$) $\le$2) for some row bi of B
    u= [e_i , $SB + b_i$] ;
  Else Retransmission is requested.
  }}}
Where, wt (x) represents weight of a variable 'x'

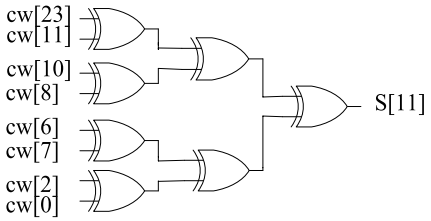

Fig. 4. Delay optimized structure for syndrome measurement.

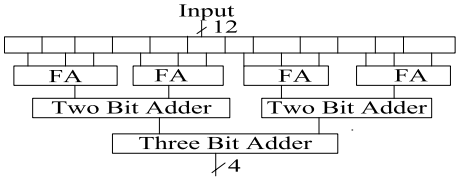

Fig. 5. Structure for weight measurement.

In the similar fashion, other S + b_i are calculated and for calculating SB+b_i, S is replaced with SB in each expression.

4) Selection of S+b_i and SB+b_i for Which Weight Is Less Than Two: This unit selects the S + b_i or SB + b_i for which weight is less than equal to two. To obtain a weight of less than equal to two with the exception of zero, as it never occurs, we have incorporated a logic shown in Fig. 6(a). In this way, after performing for all the 12 registers, we get 12 bits as output. These bits are then fed to a 12:1 priority encoder as inputs and subsequently a 13:1 multiplexer is used to select the desired register content, which has satisfied the weight constraint. The four output of the priority encoder serves as the select signal for the multiplexer. In Fig. 6(c), S+b_i are designated by S(i).

C. Results of Hardware Implementation of Decoder Module

The IMLD algorithm [19], [24] for decoding extended binary Golay code has been verified using MATLAB R2009b tool and the proposed hardware architecture has been implemented in XC4vlx160-12-ff1148FPGA using Xilinx ISE tool. The synthesized frequency for the decoder module is 195.082 MHz. In addition, this architecture utilizes 785 LUTs and 230 slices out of 135 168 each. The synthesis results obtained from Synopsys design compiler using

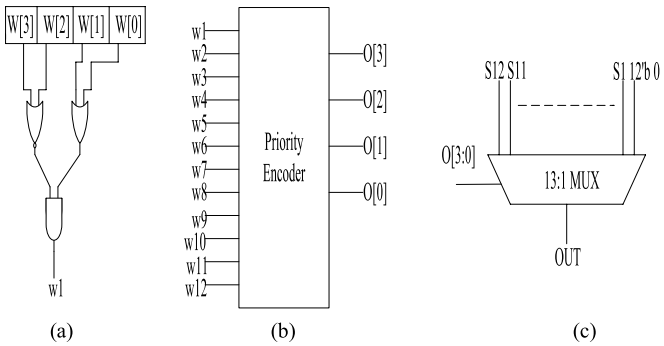

Fig. 6. (a) Structure to test weight constraint. (b) Priority encoder for generating selection signal for multiplexer. (c) Multiplexer to select the register satisfying weight constraint.

TABLE III
COMPARISON OF THE PROPOSED DECODER ARCHITECTURE CONSIDERING RESOURCE UTILIZATION AND SPEED

Reference	Slices (% of utilization)	Frequency (MHz)
Proposed	1.9	80
[24]	2.3	87.3
Device	Virtex-E	

TABLE IV
COMPARISON OF THE PROPOSED DECODER ARCHITECTURE CONSIDERING THROUGHPUT, LATENCY, AND AREA

Reference	Throughput	Latency	Area (Number of Gates)
[10]	1 output/144 clock cycles	576 clock cycles	-
[15]	-	23 gate level delay	6000
[16]	-	-	3500
[23]	1 output/24 clock cycles	48 clock cycles	4000
Proposed	1 output/clock cycle	27 clock cycles	3013

0.18-μm UMC technology depict that total dynamic power consumed by the architecture is 14.65 mW and total cell area is 30129.32 μm^2.

Table III clearly shows that the proposed decoder architecture has better resource utilization and speed than the most recent work [24] incorporating same algorithm for decoder. Table IV presents the comparison in terms of latency, throughput, and area. The proposed architecture has a latency of 27 clock cycles, which is the least among all and occupies an area of 3013 equivalent gates, which is also the least among the available literature. Another crucial factor is throughput, which depicts the number of outputs per clock cycle. The proposed architecture can yield an output (24 bits) per clock cycle, which is lacking in [10] and [23]. Therefore, the proposed architecture for decoder looks promising for high data rate system.

V. CONCLUSION

Efficient hardware architecture for both binary Golay encoder and extended binary Golay encoder have been designed and implemented after verifying the proposed algorithm. The results obtained from simulation state that the proposed hardware architecture for encoder supersedes the conventional LFSR-based CRC generation schemes. Similarly, the proposed hardware module for decoder shows better performance to some of the recent publications considering various performance metrics. These hardware modules for encoder and decoder can be a good candidate for various applications in high-speed communication links, photo spectroscopy, and ultrasonography.

REFERENCES

[1] M. J. E. Golay, "Notes on digital coding," *Proc. IRE*, vol. 37, p. 657, Jun. 1949.
[2] X.-H. Peng and P. G. Farrell, "On construction of the (24, 12, 8) Golay codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 8, pp. 3669–3675, Aug. 2006.
[3] B. Honary and G. Markarian, "New simple encoder and trellis decoder for Golay codes," *Electron. Lett.*, vol. 29, no. 25, pp. 2170–2171, Dec. 1993.
[4] B. K. Classon, "Method, system, apparatus, and phone for error control of Golay encoded data signals," U.S. Patent 6 199 189, Mar. 6, 2001.
[5] M.-I. Weng and L.-N. Lee, "Weighted erasure codec for the (24, 12) extended Golay code," U.S. Patent 4 397 022, Aug. 2, 1983.
[6] S.-Y. Su and P.-C. Li, "Photoacoustic signal generation with Golay coded excitation," in *Proc. IEEE Ultrason. Symp. (IUS)*, Oct. 2010, pp. 2151–2154.
[7] M. Spachmann, "Automatic generation of parallel CRC circuits," *IEEE Des. Test. Comput.*, vol. 18, no. 3, pp. 108–114, May/Jun. 2001.
[8] G. Campobello, G. Patane, and M. Russo, "Parallel CRC realization," *IEEE Trans. Comput.*, vol. 52, no. 10, pp. 1312–1319, Oct. 2003.
[9] R. Nair, G. Ryan, and F. Farzaneh, "A symbol based algorithm for hardware implementation of cyclic redundancy check (CRC)," in *Proc. VHDL Int. Users' Forum*, Oct. 1997, pp. 82–87.
[10] A. D. Abbaszadeh and C. K. Rushforth, "VLSI implementation of a maximum-likelihood decoder for the Golay (24, 12) code," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 3, pp. 558–565, Apr. 1988.
[11] J. Snyders and Y. Be'ery, "Maximum likelihood soft decoding of binary block codes and decoders for the Golay codes," *IEEE Trans. Inf. Theory*, vol. 35, no. 5, pp. 963–975, Sep. 1989.
[12] I. S. Reed, X. Yin, T. K. Truong, and J. K. Holmes, "Decoding the (24, 12, 8) Golay code," *IEE Proc. E Comput. Digit. Techn.*, vol. 137, no. 3, pp. 202–206, May 1990.
[13] S.-W. Wei and C.-H. Wei, "On high-speed decoding of the (23, 12, 7) Golay code," *IEEE Trans. Inf. Theory*, vol. 36, no. 3, pp. 692–695, May 1990.
[14] A. Vardy and Y. Be'ery, "More efficient soft decoding of the Golay codes," *IEEE Trans. Inf. Theory*, vol. 37, no. 3, pp. 667–672, May 1991.
[15] W. Cao, "High-speed parallel VLSI-architecture for the (24, 12) Golay decoder with optimized permutation decoding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), Connecting World*, vol. 4. May 1996, pp. 61–64.
[16] W. Cao, "High-speed parallel hard and soft-decision Golay decoder: Algorithm and VLSI-architecture," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP).*, vol. 6. May 1996, pp. 3295–3297.
[17] G. Solomon, "Golay encoding/decoding via BCH-hamming," *Comput. Math. Appl.*, vol. 39, no. 11, pp. 103–108, Jun. 2000.
[18] I. Boyarinov, I. Martin, and B. Honary, "High-speed decoding of extended Golay code," *IEE Proc. Commun.*, vol. 147, no. 6, pp. 333–336, Dec. 2000.
[19] D. C. Hankerson *et al.*, *Coding Theory and Cryptography: The Essentials*, 2nd ed. New York, NY, USA: Marcel Dekker, 2000.
[20] M.-H. Jing, Y.-C. Su, J.-H. Chen, Z.-H. Chen, and Y. Chang, "High-speed low-complexity Golay decoder based on syndrome-weight determination," in *Proc. 7th Int. Conf. Inf., Commun., Signal Process. (ICICS)*, Dec. 2009, pp. 1–4.
[21] T.-C. Lin, H.-C. Chang, H.-P. Lee, and T.-K. Truong, "On the decoding of the (24, 12, 8) Golay code," *Inf. Sci.*, vol. 180, no. 23, pp. 4729–4736, Dec. 2010.
[22] P. Adde, D. G. Toro, and C. Jego, "Design of an efficient maximum likelihood soft decoder for systematic short block codes," *IEEE Trans. Signal Process.*, vol. 60, no. 7, pp. 3914–3919, Jul. 2012.
[23] P. Adde and R. Le Bidan, "A low-complexity soft-decision decoding architecture for the binary extended Golay code," in *Proc. 19th IEEE Int. Conf. Electron., Circuits, Syst. (ICECS)*, Dec. 2012, pp. 705–708.
[24] A. Alimohammad and S. F. Fard, "FPGA-based bit error rate performance measurement of wireless systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 7, pp. 1583–1592, Jul. 2014.