A Novel and Fast Hardware Implementation for Golay Code Encoder

Morteza NAZERI, Abdalhossein REZAI

Academic Center for Education, Culture and Research Institute of Higher Education,
Isfahan Branch, 84175-443 Isfahan, Iran

mortaza_nazeri@yahoo.com, rezaie@acecr.ac.ir

DOI: 10.15598/aeee.v16i4.2735

Abstract. *The Error Correction Code (ECC) is utilized to reduce the probability of error in digital systems. The binary Golay code is an ECC that can correct any combination of three or fewer random errors over a block of 23 digits. This code can be extended by appending a parity check bit to each codeword. There are several algorithms for constructing of Golay code, but more of them are not comfortable for hardware implementation. In this paper, an efficient hardware architecture is presented for the encoder of both binary Golay code and extended Golay code based on CRC. The proposed Golay code encoder is constructed of three units, which are designed carefully: data path, control unit and conversion unit. The proposed architecture is implemented on FPGA using Xilinx ISE 14.2. The implementation results demonstrate that low latency, high throughput, low area and less complexity are the advantages of this architecture compared to previous architectures. Thus, this hardware module can be used for high-speed digital systems.*

Keywords

Binary Golay code, Error Correction Code (ECC), extended binary Golay code, FPGA, hardware implementation.

1. Introduction

A major concern of digital system designers is the control of error so that the data can be reliably reproduced [1]. Thus, a reliable digital system must be able to detect and correct such errors. To reach this goal, error detecting and correcting codes are used. In fact, the Error Correction Codes (ECCs) are utilized for recognizing and correcting errors in digital systems [1], [2], [3] and [4].

The ECC theory started in the 1940s with the works of Hamming [5], and Golay and Shannon [6] to detect errors. The ECCs can be divided into several types. Some of these codes only detect a single bit error and other codes can detect multi-bit errors. In addition, there are some codes that can identify the exact location of the errors, which are called error detection and correction codes [1].

The binary Golay code (G_{23}) is a perfect linear error correcting code that can correct any combination of three or fewer random errors over 23 digits block. The Golay code can be extended by appending a parity check bit to each codeword. Binary Golay code (G_{23}) is shown as $(23, 12, 7)$, which means the codeworde length is 23, the message length is 12 and Hamming distance is 7. On the other hand, the extended binary Golay code (G_{24}) is half-rate code as $(24, 12, 8)$. It is one type of block codes in which any 3 bits error can be corrected or any 7 bits error can be detected [7]. The encoder for Golay code provides 12 data bits and 11 check bits that are generated according to the polynomial for the $(23, 12)$ Golay code [8].

A review of the various works to design and implement of Golay code introduces several algorithms for Golay code, but there are a few algorithms that can be implemented in hardware [8], [9] and [10]. Weng and Lee [11] developed a hardware architecture for Golay code encoder based on Cyclic Redundancy Check (CRC) and Linear Feedback Shift Register (LFSR), but due to less throughput and high latency, it is not suitable for high-speed data communication links. Sarangi and Banerjee [7] have proposed an algorithm and architecture for hardware implementation of Golay code encoder in FPGA. Authors of [7] utilized the encoding architecture that processes the codeword instead of the bit per clock cycle. The synthesis frequency in this architecture is 238.575 MHz, which will be improved in this paper.

This paper presents and evaluates an efficient encoder architecture for both binary Golay code and extended binary Golay code. The proposed architecture is implemented on various FPGA devices. The implementation results show that the proposed encoder architecture has advantages compared to encoder architectures in recent related works.

The rest of this paper is organized as follows. Section 2. briefly describes the required background. The proposed binary and extended Golay code architectures are presented in Sec. 3. The hardware implementation of the proposed encoder architecture is presented and compared in Sec. 4. Finally, Sec. 5. concludes this paper.

2. Background

2.1. Cyclic Redundancy Check

Cyclic redundancy check or simply CRC is one of the powerful method, which is used for detecting and correcting errors on serial data communication link, digital network and in mass storage devices [12]. To check for errors, CRC uses a math calculation to generate an integer based on the transmitted message. The transmitter performs the calculation before sending its results to the receiver. The receiver repeats the same calculation after transmission. If both transmitter and receiver obtain the same result, it is assumed that the transmission is error-free [12].

It should be noted that the high-throughput rate computation is hard to achieve without using hardware acceleration [13], [14] and [15]. One of the more established hardware solutions for CRC is LFSR. In this method, every n-bit data word needs n clock cycles to calculate the checksum. So, this property causes high-latency and low-throughput.

2.2. The Golay Code

The binary Golay code is defined in a Galois field $GF(2)$ by its length 23, number of data 12, and minimum distance 7. We use $(23, 12, 7)$ to denote binary Golay code in the binary field $GF(2)$. In other words, the covering radius is the maximum number of hard decision errors, which are correctable by this code. In the perfect codes such as the Golay code, the covering radius is calculated as $\frac{d_{\min}-1}{2}$. For the Hamming code $[2m-1, 2m-m-1, 3]$, the covering radius is 1 and for the binary Golay code, the covering radius is 3 [16]. This code can be put in the cyclic form. Hence, it can be encoded and decoded based on its cyclic structure. It is generated by one of the flowing polynomials [1] and [9]:

$$g_1(X) = 1 + X^2 + X^4 + X^5 + X^6 + X^{10} + X^{11}, \quad (1)$$

$$g_2(X) = 1 + X + X^5 + X^6 + X^7 + X^9 + X^{11}. \quad (2)$$

Both $g_1(X)$ and $g_2(X)$ are factors of $X^{23} + 1$ and $X^{23} + 1 = (1 + X)g_1(X)g_2(X)$, which are polynomial generators [1] and [7].

The encoding can be done using an 11-stage shift register by feedback connections according to $g_1(X)$ or $g_2(X)$ [1]. The remainder of the division gives the required check bits with the data giving us the Golay codeword. The minimum Hamming distance of binary Golay code $(23, 12)$ is 7. The number of corrected error bits is 3. All the 3 bits error can be corrected [17], [18] and [19]. As a result, it is called perfect code. It meets the following equation [3], [17] and [19]:

$$2^{n-k} = \sum_{i=0}^{t} \binom{n}{i}. \quad (3)$$

In this equation, n denotes the number of codeword bits and k denotes the number of message bits. For the binary Golay code $(23, 12, 7)$, the above equation is as follows:

$$2^{23-12} = 2048 = 1 + \binom{23}{1} + \binom{23}{2} + \binom{23}{3}. \quad (4)$$

This satisfies the Hamming bound. It is said that Golay discovered this code by observing this relationship in Pascal's triangle [20]. For the extended Golay code $(24, 12)$, $n = 24$, and $k = 12$. The distance of this extended Golay code is increased to 8. In addition, all 3 bits error can be corrected [17], [21] and [22].

2.3. The Binary Golay Code Algorithm Based on CRC

The encoding in Golay code consists of followings steps [1], [7] and [23]:

- Step 1: The generator polynomial $g(X)$ is selected for long division process. It is equivalent to 110001110101 or 101011100011 in the binary digit form [1], [7] and [23].
- Step 2: Multiply data $u(x)$ by X^{23-12}. It is equivalent to appending 11 zeros to the right of message [1], [7] and [23].
- Step 3: The remainder $b(x)$ is obtained (the parity check bit) from dividing $X^{n-k}u(x)$ by generator polynomial $g(X)$ [1], [7] and [23]. In digital form,

it is equivalent to the remaining bits except for the most significant bit, which is resulted at the end of the long division process [1], [7] and [23]. The message and check bits provide the codeword of the encoded Golay code $(23, 12, 7)$.

- Step 4: $b(x)$ and $X^{n-k}u(x)$ are combined to obtain the code polynomial $b(x) + X^{n-k}u(x)$. It is equivalent to appending parity check bit resulted in step 3 to end of message [1], [7] and [23].

2.4. The Extended Binary Golay Code

The extended binary Golay code $(24, 12, 8)$ is generated by appending the parity bit to the binary Golay code. When the binary Golay code weight is even, the parity bit 0 is concatenated to binary Golay code. Otherwise, parity bit 1 is concatenated [7]. The extended binary Golay code weight must be a multiple of four and equal or greater than 8 [7] and [24]. Figure 1 shows an example to verify the algorithm for long division [7].

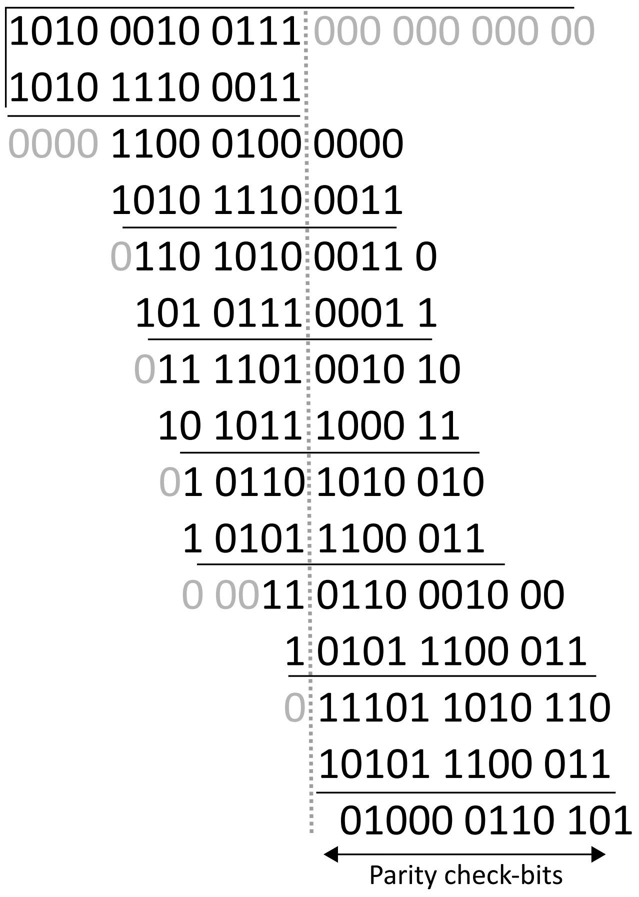

Fig. 1: An example for the generation of check bits [7].

In this example, the message is A27h, $M(x) = \text{A27h}$, the polynomial is $g(X) = (1010\ 0010\ 0111\ 0000\ 000)_2$ and the generated check bits are 435h. As a result, the encoded codeword is A27435h. So, the generated G_{24} codeword is $(1010\ 0010\ 0111\ 1000\ 0110\ 1011)_2$ [1] and [7]. The weight of generated codeword in this example is 12 that is greater than eight and multiple of four. Thus, the generated codeword is valid.

3. The Proposed Architectures

The proposed architecture for the binary and extended binary Golay code is constructed in three units:

- data unit or data path,
- control unit,
- conversion unit,

which will be described in this section as follows.

3.1. The Proposed Binary Data Unit Architecture

Figure 2 shows the proposed architecture for the binary Golay code generator.

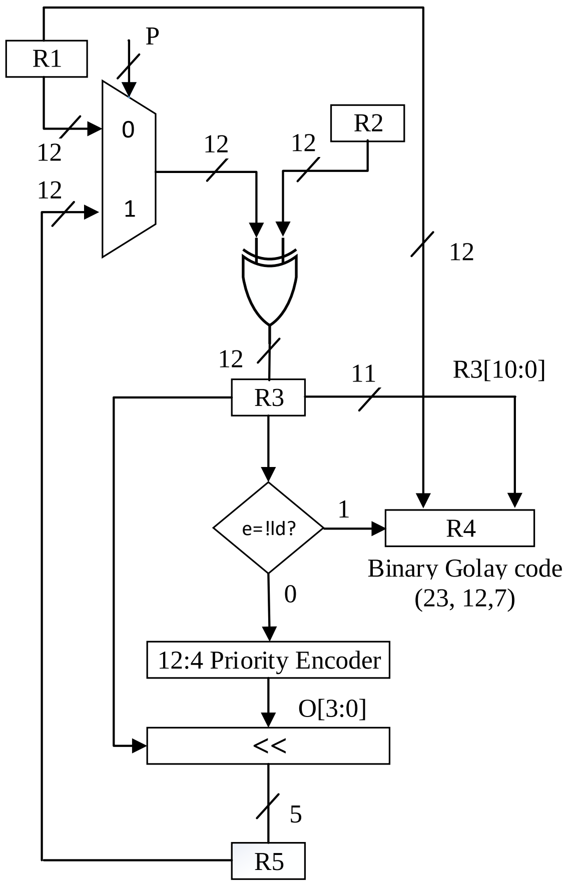

Fig. 2: The data path of the proposed architecture for the binary Golay code generator.

In this architecture, R1, R2, R3 and R5 in data path are 12-bit registers and R4 is 23-bit register. The value of the polynomial generator is stored in the register R2. The 2:1 multiplexer is utilized to choose a primary message (the contents of the register R1) or shifted the intermediate results (the contents of the register R5). In each stage of the data unit, a simple binary XOR gate is used for modulo-2 subtraction. The result of bitwise XOR operation is saved in register R3. A 12:4 priority encoder has been utilized for the detecting of the number of zero bits before the first bit 1 remaining at any stage.

A circular shift register is utilized to shift the intermediate result to left within the output of the priority encoder. Then, the results are saved in the register R5 and then loop back to cycle until e become 1 and Golay code is constructed and loaded in R4 register.

3.2. The Proposed Binary Control Unit Architecture

Figure 3 shows the control unit, which is utilized for loop control mechanism and comprised of two multiplexers, one subtractor and two registers, R6 and R7.

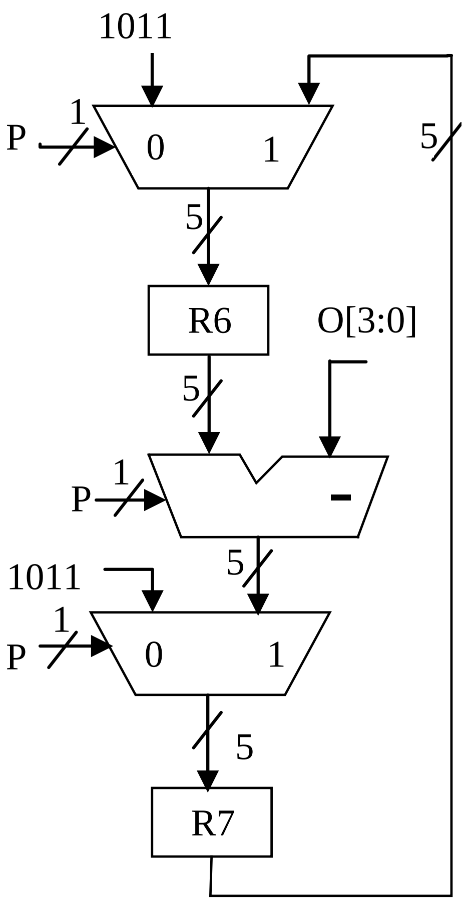

Fig. 3: The control unit of the proposed architecture for the binary Golay code generation.

All multiplexers, which are used in the data path, and subtractor in control unit are activated by signal P, and all multiplexers in control unit are activated by signal S. It should be noted that signal P is the bit by bit OR operation of priority encoder output and S is the bit by bit OR operation of R5 register. If the priority encoder output is zero, then signal P will be zero, and otherwise, it will be one.

As it is shown in Fig. 3, in the first step of the control unit, registers R6 and R7 are loaded with $(11)_{10} = (001011)_2$. So, one input of subtractor is initialized with 11 that shows the number of zero to be added in step 2 of the algorithm. It also gets updated with the contents of R7 when signal P becomes one. Another input of the subtractor is the output of priority encoder.

The result of the subtraction is either zero or a negative value after the final iteration that is saved in register R7. Where the value of the register R7 becomes zero, the register R4 is loaded that denotes the end of the division process. As a result, the check bits generation process is ended.

The signal 'Ld' in Fig. 2 is the control signal for controlling the load of the parity check bit value or R3 [10:0] contents to register R4 at final iteration. Note that Ld =! (R7 [4]) & (|(R7)) where (|(R7) means bit by bit OR operation of R7 bits.

The above expression is developed based on the fact that signal 'Ld' must become zero where the value of register R7 is either zero (|(R7) = 0) or negative ((R7 [4]) = 1). As signal 'Ld' becomes zero, signal 'e' becomes one and eleven bits of register R3 [10:0] are sent to R4 [10:0]. In addition, data contents are sent to R4 [22:11]. As a result, iteration loop is broken and the content of the register R4 gives the encoded codeword.

3.3. The Proposed Conversion Binary Golay Code to Extended Binary Golay Code Architecture

The proposed hardware architecture for the converting binary Golay code to extended binary Golay code is shown in Fig. 4.

In this architecture, the weight of the binary Golay code can be found by adding all 23 bits of the register R4 and the resulting sum is stored in the register R8. Register R9 contains content of R4 concatenated with R8 [0]. If the contents of R8, which is the number of one's in the binary Golay code becomes even, then R8 [0] will be 0 and vice versa, which acts as the parity bit. Finally, R9 contains extended binary Golay codeword.

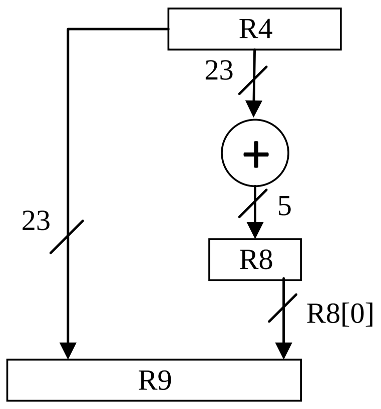

Fig. 4: The proposed architecture for converting the binary Golay code into the extended binary Golay code.

4. Hardware Implementation and Comparison of The Proposed Architectures

The Golay encoder algorithm based on CRC is verified by using MATLAB R2014a tool. In addition, the proposed hardware architectures, which are shown in Figs 2-4, have been implemented on Virtex 5 (XC5vfx70t), Virtex 6 (XC6vlx760l-1Lff1760), Virtex 7 (XC7vx330tl-2Lffg1157) and Spartan 6 (XC6lx150l-1Lfgg767) FPGA devices using Verilog description language in Xilinx ISE 14.2.

Table 1 shows the implementation results of the proposed architecture for the Golay code encoder on various FPGA devices.

Tab. 1: The implementation results for the proposed architecture on varies FPGAs.

FPGA device	Synthesized frequency (MHz)	Number of LUT	Number of Slices (Area)
Virtex 5	262.469	138	76
Virtex 6	356.506	87	88
Virtex 7	334.874	159	88
Spartan 6	121.407	88	90

To perform a fair comparison with previous hardware implementations, the proposed Golay code encoder architecture is re-implemented on Virtex 4 FPGA device. The post place and route values of the proposed architecture are shown in Tab. 2, Tab. 3 and Tab. 4 contrasted with [7], [11], [12], [17] and [25].

Table 2 compares the properties of the hardware implementation of Golay code encoder architectures.

Based on our implementation results that are shown in Tab. 2 and Fig. 2, Fig. 3 and Fig. 4 the proposed architecture for the Golay code encoder occupies less area in comparison with encoder architecture in [7]. It is because the proposed architecture requires fewer registers. Based on these implementation results, the number of required 4-input LUT, and the number of the occupied slices in the proposed encoder architecture are reduced compared to architecture in [7] by about 36.9 % and 14.6 %, respectively. In addition, the synthesized frequency is improved from 238.575 to 251.247 MHz in comparison with encoder architecture in [7] and [17].

Tab. 2: Comparison of Golay code encoder.

Reference	Number of 4 input LUT	Number of occupied Slices (Area)	Synthesized frequency (MHz)
[7]	187	103	238.575
[17]	13	Not mentioned	238.575
This paper	118	88	251.247

Table 3 represents a comparison of latency and clocking mechanism of the proposed architecture with other architectures in [7] and [11].

Tab. 3: The compassion of latency and clocking mechanism of the Golay code architectures.

Reference	Latency (clock cycle)	Clocking mechanism
[11]	23	System clock + clock doubler
[7]	12	System clock
This paper	12	System clock

Based on our implementation results that are shown in Tab. 3, the proposed Golay code architecture, similar to [7], avoid any extra clocking mechanism compared to [11].

Table 4 compares the proposed architecture with parallel CRC-11 circuits.

Tab. 4: The comparison of the proposed architecture with parallel CRC-11 circuits.

Reference	LUT utilization (%)	Latency (clock cycle)
[12]	1.33	12
[25]	1.72	12
[7]	0.14	12
This paper	0.076	12

Table 4 depicts that the percent of the utilized LUT is reduced by about 45.8 %, 94.3 %, and 95.6 % compared to parallel CRC-11 circuits in [7], [12] and [25], respectively. In addition, Tab. 5 compares the properties of the hardware implementation of Golay code encoder architectures on Virtex 5 FPGA device.

Tab. 5: Comparison of Golay code encoder.

Reference	Number of LUT	Number of occupied Slices (Area)
[26]	211	109
This paper	138	76

Based on our implementation results that are shown in Tab. 5, the proposed architecture for the Golay code encoder occupies less area and LUT in comparison with encoder architecture in [26]. Based on these implementation results, the number of required LUT, and the number of the occupied slices in the proposed encoder architecture are reduced compared to architecture in [26] by about 34.6 % and 30.3 %, respectively.

5. Conclusion

In this paper, an efficient encoder architecture was designed based on CRC for both binary Golay code and extended binary Golay code. The proposed Golay code architecture was implemented on Virtex 4, Virtex 5, Virtex 6, Virtex 7 and Spartan 6 FPGA devices using Xilinx ISE 14.2. The implementation results showed that the developed encoder architecture for both binary Golay code and extended binary Golay code has advantages compared to Golay code encoder architectures in [7], [11], [12], [17], [25] and [26]. Therefore, the developed encoder modules for Golay code can be used for various applications in high-speed communication links and mass storage devices.

References

[1] LIN, S. and D. COSTELLO. *Error Control Coding*. 2nd ed. Englewood Cliffs: Prentice Hall, 2010. ISBN 978-0130426727.

[2] REVIRIEGO, P., S. LIU, L. XIAO and J. A. MAESTRO. An Efficient Single and Double-Adjacent Error Correcting Parallel Decoder for the (24,12) Extended Golay Code. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2016, vol. 24, iss. 4, pp. 1603–1606. ISSN 1063-8210. DOI: 10.1109/TVLSI.2015.2465846.

[3] MCWILLIAMS, F. and N. SLOANE. *The Theory of Error-Correcting Codes*. Amsterdam: Elsevier, 1983. ISBN 978-0444851932.

[4] MITRA, J. and T. NAYAK. Reconfigurable very high throughput low latency VLSI (FPGA) design architecture of CRC 32. *Integration*. 2017, vol. 56, iss. 1, pp. 1–14. ISSN 0167-9260. DOI: 10.1016/j.vlsi.2016.09.005.

[5] HAMMING, R. W. Error detecting and error correcting codes. *The Bell System Technical Journal*. 1950, vol. 29, iss. 2, pp. 147–160. ISSN 0005-8580. DOI: 10.1002/j.1538-7305.1950.tb00463.x.

[6] SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*. 1948, vol. 27, iss. 3, pp. 379–423. ISSN 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[7] SARANGI, S. and S. BANERJEE. Efficient Hardware Implementation of Encoder and Decoder for Golay Code. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2015, vol. 23, iss. 9, pp. 1965–1968. ISSN 1063-8210. DOI: 10.1109/TVLSI.2014.2346712.

[8] PENG, X.-H. and P. G. FARRELL. On Construction of the $(24, 12, 8)$ Golay Codes. *IEEE transactions on information theory*. 2006, vol. 52, iss. 8, pp. 3669–3675. ISSN 0018-9448. DOI: 10.1109/TIT.2006.876247.

[9] HONARY, B. and G. MARKARIAN. New simple encoder and trellis decoder for Golay codes. *Electronics Letters*. 1993, vol. 29, no. 25, pp. 2170–2171. ISSN 0013-5194. DOI: 10.1049/el:19931456.

[10] CLASSON, B. K. *Method, system, apparatus, and phone for error control of Golay encoded data signals*. U.S. Patent 6199189, 2001.

[11] WENG, M.-I. and L.-N LEE. *Weighted erasure codec for the (24, 12) extended Golay code*. U.S. Patent 4397022, 1983.

[12] SPRACHMANN, M. Automatic generation of parallel CRC circuits. *IEEE Design & Test of Computers*. 2001, vol. 18, iss. 3, pp. 108–114. ISSN 0740-7475. DOI: 10.1109/54.922807.

[13] REPKA, M. Note on Modular Reduction in Extended Finite Fields and Polynomial Rings for Simple Hardware. *Journal of Electrical Engineering*. 2016, vol. 67, iss. 1, pp. 56–60. ISSN 1339-309X. DOI: 10.1515/jee-2016-0008.

[14] REZAI, A. and P. KESHAVARZI. High-performance scalable architecture for modular multiplication using a new digit-serial computation. *Microelectronics Journal*. 2016, vol. 55, iss. 1, pp. 169–178. ISSN 0026-2692. DOI: 10.1016/j.mejo.2016.07.012.

[15] REZAI, A. and P. KESHAVARZI. High-Throughput Modular Multiplication and Exponentiation Algorithms Using Multibit-Scan–Multibit-Shift Technique. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2015, vol. 23, iss. 9, pp. 1710–1719. ISSN 1063-8210. DOI: 10.1109/TVLSI.2014.2355854.

[16] TOMLINSON, M., C. J. TJHAI, M. A. AMBROZE, M. AHMED and M. JIBRIL. *Error Correction Coding and Decoding Bounds, Codes, Decoders, Analysis and Applications*. Plymouth: Springer, 2015. ISBN 978-3-319-51103-0.

[17] GOWRI, S. and K. RIBANA. High-Speed Encoder and Decoder of the Binary Golay Code. *Journal of Chemical and Pharmaceutical Sciences*. 2016, vol. 2016, iss. 8, pp. 125–128. ISSN 0974-2115.

[18] BIOGLIO, V. and I. LAND. Polar-Code Construction of Golay Codes. *IEEE Communications Letters*. 2018, vol. 22, iss. 3, pp. 466–469. ISSN 1089-7798. DOI: 10.1109/LCOMM.2018.2793273.

[19] PETERSON, W. W. and E. J. WELDON. *Error-correcting codes*. Cambridge: MIT Press, 1972. ISBN 978-0262160390.

[20] JUSTESEN, J. and T. HOHOLDT. *A Course in Error-Correcting Codes*. Zurich: European Mathematical Society, 2004. ISBN 978-3-03719-001-2.

[21] SATISH BABU, P., S. ANUSHA, V. N. JYOTHI, T. V. KUMAR and M. H. M. BASHA. An Efficient Single and Double-Adjacent Error Correcting Parallel Decoder for the (24,12) Extended Golay Code. *International Journal of Research*. 2018, vol. 5, iss. 7, pp. 1914–1917. ISSN 2348-6848.

[22] ZHANG, P., F. C. M. LAU and C.-W. SHAM. Design of a High-Throughput Low-Latency Extended Golay Decoder. In: *23rd Asia-Pacific Conference on Communications*. Perth: IEEE, 2017, pp. 1–4. ISBN 978-1-7405-2390-5. DOI: 10.23919/APCC.2017.8304002.

[23] RANGARE, U. and R. THAKUR. A Review on Design and Simulation of Extended Golay Decoder. *International Research Journal of Engineering and Technology*. 2016, vol. 3, iss. 6, pp. 1051–1054. ISSN 2395-0072.

[24] MOON, T. K. *Error Correction Coding, Mathematical Methods and Algorithms*. New York: John Wiley & Sons, 2005. ISBN 978-0471648000.

[25] CAMPOBELLO, G., G. PATANE and M. RUSSO. Parallel CRC realization. *IEEE Transactions on Computers*. 2003, vol. 52, iss. 10, pp. 1312–1319. ISSN 0018-9340. DOI: 10.1109/TC.2003.1234528.

[26] MANIKANDAN, J., S. SHRUTHI, S. J. MANGALA and V. K. AGRAWAL. Design and Implementation of Reconfigurable Coders for Communication Systems. In: *International Conference on VLSI Systems, Architectures, Technology and Applications*. Bangalore: IEEE, 2016, pp. 1–5. ISBN 978-1-5090-0033-3. DOI: 10.1109/VLSI-SATA.2016.7593063.

About Authors

Morteza NAZERI received the B.Sc. degree from the Mohajer technical University, Isfahan, Iran, in 2014 and the M.Sc. degree from Academic Center for Education, Culture, and Research (ACECR) institute of higher education, Isfahan branch, Isfahan, Iran, in 2017, all in electrical engineering. His current research interests include VLSI Design.

Abdalhossein REZAI (corresponding author) received the B.Sc. and M.Sc. degrees from the Isfahan University of Technology (IUT), Isfahan, Iran, in 1999, and 2003, respectively, and the Ph.D. degree from Semnan University, Semnan, Iran in 2013, all in electrical engineering. He is currently an Assistant Professor with the Academic Center for Education, Culture, and Research (ACECR), IUT branch. His current research interests include network security, cryptography algorithm and its application, and neural network implementation in nanoelectronics.