

An efficient high throughput BCH module for multi-bits error correction mechanism on hardware platform

Rohith Puttaraju, Ramesha Muniyappa

Department of Electrical, Electronics and Communication Engineering, GITAM (Deemed to be University), Bengaluru, India

Article Info

Article history:

Received Jan 30, 2024

Revised Feb 24, 2024

Accepted Mar 16, 2024

Keywords:

Bose-chaudhuri-hocquenghem

Berlekamp-massey algorithm

Chien search

Encoder

Error correction codes

ABSTRACT

The bose-chaudhuri-hocquenghem (BCH) codes are a cyclic error correction codes (ECC) class. The BCH is constructed by using a polynomial over the Galois field. The BCH codes can detect and correct the multi-bits with an easy decoding mechanism. The BCH codes are used in most of the storage device's cryptography, disk drives, and satellite applications. This manuscript presents an efficient high-throughput BCH module with an encoding and decoding mechanism for multi-bit corrections. The BCH code of (15, k) is used to construct the encoder and decoder architectures. The BCH encoder decoder (ED) module with single error correction (SEC), double error correction (DEC), and triple-error correction (TEC) are discussed in detail. The BCH encoder module uses a linear feedback shift register (LFSR). The BCH decoder with SEC and DEC is constructed using the syndrome generator module (SGM) and chien search module (CSM). The BCH decoder with TEC is designed using SGM, inversion-based berlekamp-massey-algorithm (BMA), and CSMs. The BCH-ED module with SEC, DEC, and TEC utilizes <1 % chip area on Artix-7 FPGA. The BCH-ED with SEC, DEC, and TEC achieves a throughput of 7.13 Gbps, 1.2 Gbps, and 0.803 Gbps, respectively. Lastly, the BCH module is compared with existing BCH approaches with better improvement in chip area, frequency, and throughput parameters.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Rohith Puttaraju

Department of Electrical, Electronics and Communication Engineering

GITAM (Deemed to be University) Bengaluru Campus

Doddaballapura, India

Email: rohithgowda1985@gmail.com

1. INTRODUCTION

Internet traffic significantly increased in most advanced applications like social networks, video streaming, and cloud computing in the past few years. Many telecom operators consider optical networks with a high data rate to compensate for this issue. However, the optical networks are limited to disturbances or noises. The forward error correction (FEC) approaches are suitable to solve the above issues and also improve the data rates. The FEC is most suitable for advanced communications and networks for end-end data transmission. Many metrics must be considered while designing the FEC approaches, including bit error rate (BER) performance, coding gain, complexity in implementation, ability to correct the burst errors, and transmission overhead. The FEC codes are also called error correction codes (ECC) and are classified into traditional FEC and streaming codes. Examples of the traditional FECs are rateless codes, random-linear convolutional codes (RLC), and reed-solomon (RS) codes. Examples of the streaming codes are shifted repetition (Maximally shifted codes), shifted RLC, and concatenated codes [1], [2]. The linear codes transmit the symbols or bits on the communication channel. In contrast, they communicated if any errors occurred,

which will be detected and corrected on the receiver side. The linear codes are partitioned into convolutional and block codes based on codewords as per the concept of coding theory. The most popular linear codes are cyclic codes, hamming codes, RS codes, low-density-parity-check (LDPC) codes, turbo codes, polynomial codes, and parity codes.

The bose-chaudhuri-hocquenghem (BCH) codes are examples of the polynomial codes and class of the cyclic ECCs [3]. The BCH codes are framed using polynomials over the Galois field. The BCH codes can detect and correct the number of symbol errors. The decoding mechanism of the BCH codes is easily designed using the syndrome decoding approach. The BCH codes used many applications, including solid-state drives, cryptography, disk drives, compact disc players, flash drives, and bar codes [4], [5]. The subclass of the BCH code is narrow-sense primitive-based BCH codes and is known to improve performance metrics using features like minimum distance and dimensions [6]. The BCH codes' encoding and decoding mechanisms are implemented in software or hardware-based approaches. The execution time and BER performance are better in software-based than hardware approaches. However, BCH codes using application-specific integrated circuits (ASICs) or Field programmable gate arrays (FPGAs) as hardware approaches provide high Throughput, low-latency, and low-complexity features than software-based approaches [7]-[10].

The existing works of the BCH codes are discussed in this section. Mandal *et al.* [11] present the BCH codes with single-bit correction for Redox-based Random access memory (ReRAM) computation. The BCH codes of (15, 11) are incorporated in ReRAM for decoding operation. The BCH module uses 251 and 230 clock cycles to complete encoding and decoding operations. Choi *et al.* [12] describe the low-power and high-efficiency-based short BCH codes for memory applications. The Double error correction and triple error detection (DEC-TED) based BCH is designed for (79, 64) codes. This BCH decoder saves around 70% of Power and up to 48% latency compared to conventional BCH decoders. Chand *et al.* [13] explain the implementation of the forward error correction (FEC) and BCH using Matlab and VHDL. The BCH code (63, 36, and 5) is considered the FEC mechanism. Gunasekaran *et al.* [14] discuss the BCH decoder mechanism with pipeline architecture for M-Ary recording channels. The work improves the chip area by 46 % more than the existing 2-stage pipelined BCH decoder. The BCH decoder works with a throughput of 2.11 Gbps and 4.63 Gbps on Kintex-7 FPGA and ASIC 45-nm Technology, respectively. Cai *et al.* [15] describe the Concatenated LDPC with BCH codes for optical fiber communications. The LDPC-based convolutional codes (CC) offer a throughput of up to 150 Gbps. The concatenated design provides the BER up to 10-15 on hardware.

Subbiah and Ogunfunmi [16] explain the Hybrid BCH decoder module for advanced flash memories. The syndrome generator offers area-efficient features in the hybrid BCH decoder. The work analyzes the computation time, area, BER, and Throughput at different block sizes. Hiller *et al.* [17] describe a detailed review of the error correction mechanism for Physical unclonable functions (PUFs). The work analyzes the linear-based, pointer-based, and error-correcting code (ECC) based approaches concerning the PUFs. The work also evaluates the implementation complexity, error probability, and symbol recovery. Kwon and Shin [18] present the blind reconstruction of BCH codes. The work improves the correct reconstruction probability using lower-bound BCH codes than RS codes. Adalid *et al.* [19] explain the DEC-based BCH codes with overhead reduction. The work reduces the overhead of 2.8 % in propagation delay, 15.7 % in Power, and 1.3 % in chip area than conventional DEC-based BCH codes. Rao *et al.* [20] describe the BCH-based encoder and adaptive decoder for the (15, 6, 2) mechanism. The BCH codes perform DEC-TED and reduce around 65 % of Power and up to 35 % latency compared to conventional BCH decoders.

Mahdy *et al.* [21] present the parallel BCH with LDPC codes for flash memory applications in a Matlab environment. The work analyses the LDPC and concatenated coding (BCH +LDPC) performance at different code lengths by concerning the BER and SNR. Nageen *et al.* [22] describe the turbo product decoder for single error correction (SEC) and DEC on FPGA. The decoder uses a pipeline mechanism to improve resource (area, latency, and frequency) optimization. The researcher [23], [24] discuss the BCH product codes using the frequency domain on Kintex-7 FPGA. The work reduces the hardware complexity and offers low Latency. The BCH decoder offers a high Throughput of 5.6 Gbps at 100 MHz. Freudenberger *et al.* [25] explain the BCH with complex and soft-input decoding mechanisms to reduce the complexity. The work uses Peterson's algorithm for error correction concerning the SEC, DEC, and TED rather than the Berlekamp–Massey algorithm. Matsenko *et al.* [26] present the FEC-based fractal decoder for short-reach optical interconnects on the FPGA platform. The Wavelength division multiplexed (WDM) based short-reach optical interconnects module is interleaved with a BCH+LDPC-based fractal decoder to analyze the ECCs.

The efficient high throughput BCH encoder-decoder module with multi-bit error correction is presented in this manuscript. The contribution of the proposed work is highlighted as follows: The coefficients of error location polynomials are calculated directly without using BMA for SEC and DEC-based BCH codes in the decoder, which optimizes the chip area and power. The key equation solver is

designed using inversion-based BMA architecture for triple error correction (TEC). The synchome register is updated based on circular rotation as per iterations (r) in the BMA architecture, which optimizes the resources in chip and improves the throughput. The proposed BCH module provides low-chip area, high operating frequency, minimal power consumption, and high Throughput on the FPGA platform. The proposed BCH codes are compared with existing BCH codes with improvements in performance metrics (area, frequency and throughput).

The manuscript's organization is as follows: the BCH encoder and decoder architectures with mathematical equations are explained in detail in section 2. The BCH module simulation, synthesis, and performance comparison results are discussed in section 3. Lastly, it concludes the overall work with futhersitc suggestions in section 4.

2. HARDWARE ARCHITECTURE OF BCH MODULE

The BCH codes provide error correction capabilities and are used in most communication systems as error correcting codes (ECC). The BCH encoder-decoder architecture is illustrated in Figure 1. The BCH code (n, k) architecture contains encoder and decoder units and three registers for the encoder, decoder, and error modules. The encoder register receives the k-bit data information in parallel and serially generates the 1-bit register output. The 1-bit serial data is passed to the BCH encoder and performs the encoding operation using LFSR. The n-bit error data is stored in the error register and serially generates the 1-bit error output. The error bit and encoded data are XOR'ed to generate the corrupted data bit. The BCH decoder receives the corrupted bit and generates the decoded data serially. The decoder register receives the decoded data and generates the parallel k-bit output. The Encoder and error registers work similarly to parallel in serial out (PISO). Similarly, the decoder register works in a serial-in-parallel-out (SIPO) manner. The detailed encoding and decoding operations for SEC, DEC, and TEC are discussed below.

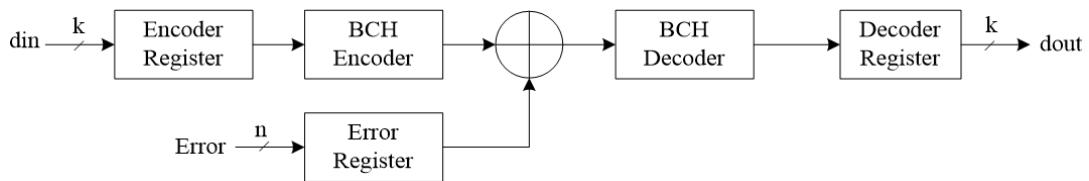


Figure 1. BCH encoder-decoder architecture

2.1. BCH encoder

The binary BCH codes (n, k) are defined with few parameters, namely: Length of the codeword in bits $n = 2^m - 1$, data bits in codeword $k \geq n - m * t$; e number of error bits to be corrected (t), and minimum distance $d_m \geq 2 * t + 1$. The codeword of the BCH (n, k, t) codes is generated using data bits (d) and generator polynomial (g). Then the polynomial of the codeword $c(x) = c_0 + c_1x + c_2x^2 + \dots + c_{n-1}x^{n-1}$ is defined as $c(x) = d(x) + g(x)$. The data bits are framed into data polynomial: $d(x) = d_0 + d_1x + d_2x^2 + \dots + d_{k-1}x^{k-1}$; where $d_i \in GF(2)$. The number of parity bits (n-k) is equal to the degree of g(x) or product of (m * t). The error correcting codes (ECC) are defined in terms of the Generator Polynomial g(x). The generator polynomial for t error bits in the BCH encoding process is represented in (1). The codewords of the BCH are encoded using data bits with a linear feedback shift register (LFSR) and are represented in (2). The parity bits are calculated using the remainder polynomial rp(x) and are represented using (3) as follows [27].

$$\begin{aligned}
 g(x) &= 1 + x + x^4 \text{ for } t = 1; \\
 g(x) &= 1 + x^4 + x^6 + x^7 + x^8 \text{ for } t = 2; \\
 g(x) &= 1 + x + x^2 + x^4 + x^5 + x^8 + x^{10} \text{ for } t = 3;
 \end{aligned}
 \tag{1}$$

$$c(x) = rp(x) + d(x) * x^{n-k}
 \tag{2}$$

$$rp(x) = \sum_{i=0}^{n-k} rp_i x^i = x^{n-k} d(x) \text{ mod } g(x)
 \tag{3}$$

Where $rp_i=0$ to (n-k) and data bits $d(x)=0$ to (k-1) are elements of the GF (2⁴). The LFSR is initialized with zero. The data bits d(x) are transmitted during l to k clock cycles, and LFSR generates the parity bits. The generated parity bits in the LFSR are transmitted in parallel during k+1 to n clock cycles. The (n-k) parity bits are appended with data bits (k) to form the encoded information. The 4, 8, and 10 parity bits

are generated for SEC, DEC, and TEC during the encoding mechanism. The LFSR generation for BCH (15, 7, 2) encoder for DEC is represented in Figure 2.

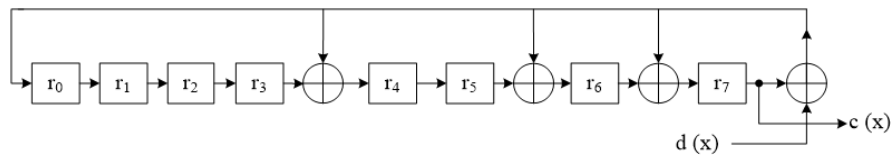


Figure 2. LFSR generation for BCH (15, 7, 2) Encoder

2.2. BCH decoder

The BCH decoding operation is divided into three steps, namely: i) syndrome calculation, ii) key equation solving, and iii) error location finding. The BEC codes for SEC and DEC use only the syndrome calculation step for decoding, which provides the error location polynomial coefficient. So key Equation solving (step 2) is not considered in both SEC and DEC processes. At the same time, TEC uses all three steps to perform the BCH decoding operations. In that, the key Equation solving is a more complex step in the decoding process and consumes more resources than the other two steps. The hardware architecture of the BCH decoder for SEC and DEC is illustrated in Figure 3. It mainly contains the syndrome generator module (SGM), a Chien search module (CSM), an error detection module, a serial-in-serial-out (SISO) register, and a control module. The SGM produced the syndromes and was used further to calculate the error location polynomial coefficients. The CSM finds the error locations, and the detection module detects the error bits. The SISO register shifts the input data bits based on control signal output, and results are XOR'ed with error bits to produce the corrected output bits. The control module acts counter with decoding logic. Similarly, the hardware architecture of the BCH decoder for TEC is illustrated in Figure 4. The architecture is similar to SEC/DEC with the addition of the Berlekamp-Massey algorithm (BMA). The inversion-based BMA is used to find the error location polynomial coefficients.

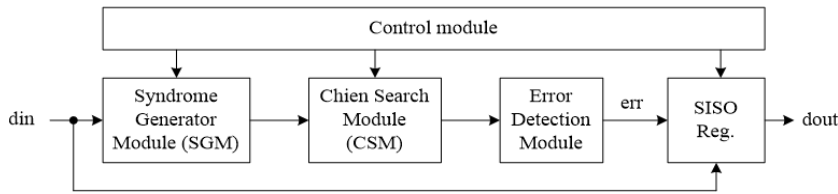


Figure 3. Hardware architecture of the BCH decoder for SEC and DEC

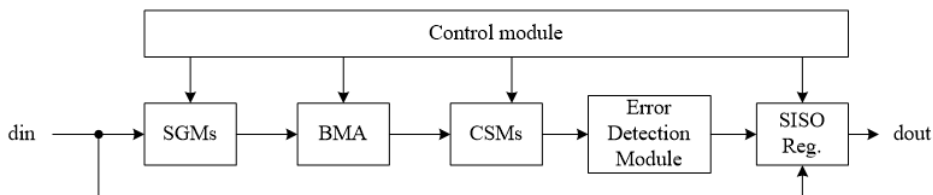


Figure 4. Hardware architecture of the BCH decoder for TEC

2.2.1. Syndrome calculation

The BCH decoder received the codeword $c(x)$ and was used further for syndrome calculation. The received codeword $c(x)$ is corrupted with error inputs $e(x)$ and acts as an input to the syndrome generator module (SGM). The received polynomial is represented in (4). Where the polynomial of the received codeword is $r(x) = r_0 + r_1x + r_2x^2 + \dots + r_{n-1}x^{n-1}$ and error input is $e(x) = e_0 + e_1x + e_2x^2 + \dots + e_{n-1}x^{n-1}$. The syndrome (S_i) is calculated [28] using (5). The syndrome is calculated for SEC operation is represented using (6).

$$r(x) = e(x) + c(x) \tag{4}$$

$$S_i = r(\alpha^i) = r_0 + r_1\alpha^i + r_2\alpha^{2i} + \dots + r_{n-1}\alpha^{(n-1)i}; \text{ where } (1 \leq i \leq 2t - 1) \tag{5}$$

$$S_1 = r(\alpha^1) = r_0 + r_1\alpha^1 + r_2\alpha^2 + \dots + r_{14}\alpha^{14} \tag{6}$$

Similarly, the syndrome is calculated for DEC operation is represented using in (7). Where the primitive element of GF (2⁴) is 'α.' similarly, three syndromes are calculated for TEC. If no error occurred in received codeword, then there is no need to calculate the syndromes. So, the calculation of the syndrome is influenced more by error polynomial. When syndromes are non-zero, a further step is to calculate the error location polynomial's coefficients in the key Equation solving step.

$$\begin{aligned} S_1 &= r(\alpha^1) = r_0 + r_1\alpha^1 + r_2\alpha^2 + \dots + r_{14}\alpha^{14} \\ S_3 &= r(\alpha^3) = r_0 + r_1\alpha^3 + r_2\alpha^4 + \dots + r_{14}\alpha^{28} \end{aligned} \tag{7}$$

2.2.2. Key equation solver

The second step of the BCH decoding is the coefficient finding of the error location polynomial $\lambda(x) = \lambda_0 + \lambda_1x + \dots + \lambda_t x^t$ concerning the syndromes (S_i). The relationship between λ_i and syndromes are represented using in (8). The error position is calculated using roots of λ(x); The coefficients of the λ(x) and It is calculated directly without using BMA for SEC and DEC-based BCH codes. The coefficients of the λ(x) SEC and DEC are represented using in (9-10) as follows [28].

$$\sum_{i=0}^t S_{t+j-i} \lambda_i = 0 \quad (j = 1, 2 \dots t) \tag{8}$$

$$\lambda(x) = 1 + S_1x \tag{9}$$

$$\lambda(x) = 1 + \lambda_1x + \lambda_2x^2 = 1 + S_1x + (S_1^2 + S_3 * S_1^{-1})x^2 \tag{10}$$

The coefficients of the λ(x) for TEC-based BCH codes are calculated using inversion-based berlekamp-massey algorithm (BMA) and is represented using in (11) [29]. The number of iterations (r) is the same as the number of errors (t). Where discrepancy d_r = $\sum_{i=0}^t S_{2r-i+1} \lambda_i^r$, Updated discrepancy d_p = d_r (when sel = 0 or 1). The selection line (sel) is considered based on the discrepancy d_r. The updated BMA coefficients are represented using in (12).

$$\lambda^r(x) = \lambda^{r-1}(x) - \beta^r(x) \cdot d_r d_p^{-1} \quad (r = 1, 2 \dots t - 1) \tag{11}$$

$$\beta^{r+1}(x) = \begin{cases} \beta^r(x) \cdot x^2 & \text{if } (sel = 0) \\ \lambda^{r-1}(x) \cdot x^2 & \text{if } (sel \neq 0) \end{cases} \tag{12}$$

The BMA architecture for Key Equation solver in BCH decoder is modelled as (11) and (12) in Figure 5. The architecture uses λ^r and β^r registers which stores the λ^r(x) and β^r(x) coefficients. The syndrome register acts a shift register which is multiplied with λ^r register index-wise and later summed to generate the discrepancy d_r. The β^r register is multiplied with d_rd_p⁻¹ to generate the correction factor (C_F). The coefficients of λ^r(x) are updated based on C_F and λ^r registers.

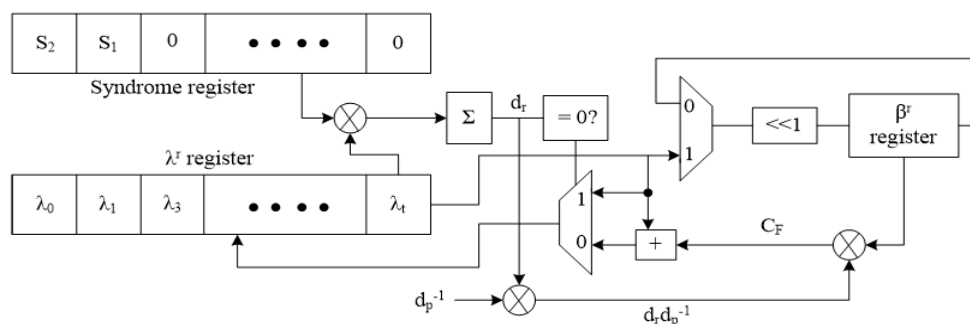


Figure 5. BMA architecture for key equation solver in BCH decoder

2.2.3. Error location finding approach

The last step of the BCH decoding is to find the error location values, and are reciprocals of the roots of the $\lambda(x)$. The error location values are calculated by substituting the $1, \alpha, \alpha^2 \dots \alpha^{n-1}$ into the $\lambda(x)$ using the Chien search approach. The Chien search sum is calculated at every clock cycle and represented using in (13) as follows:

$$\lambda_0 + \lambda_1\alpha^i + \lambda_2\alpha^{2i} + \dots + \lambda_t\alpha^{ti} \quad (i = 0,1, \dots k - 1) \tag{13}$$

The received bit (r_{n-1-i}) is corrupted only if $\lambda(\alpha^i) = 0$; the received bit (r_{n-1-i}) is corrected only when the Chien search sum at clock cycle 'i' equals zero. The hardware architecture of the Chien search module (CSM) is represented in Figure 6. The error location polynomials ($\lambda_0, \lambda_1, \lambda_2 \dots \lambda_t$) and primitive elements ($1, \alpha, \alpha^2 \dots \alpha^t$) are multiplied using a constant multiplier. The individual multiplied outputs are added on each clock cycle till 't' iterations. The CSM output was used further to detect the error.

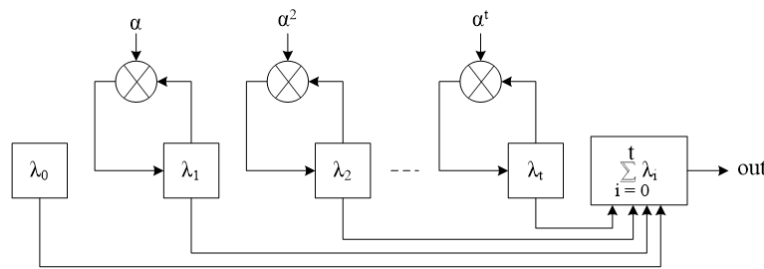


Figure 6. Hardware architecture of the Chien search module

3. RESULTS AND DISCUSSION

The results of the BCH-Encoder Decoders (ED) with multi-bit error corrections (SEC, DEC, and TEC) are discussed in this section. The BCH codes are designed and implemented using Verilog-HDL on the Xilinx ISE environment. The Artix-7 FPGA with XC7A100-TCSG324-3 device is considered for synthesis and implementation. The simulation results are analyzed and verified using the Modelsim simulator. The Xilinx-Xpower analyzer tool is used for total power generation for BCH designs. The simulation results of the BCH ED (15, 7, 2) for DEC are illustrated in Figure 7. The global clock (clk) is activated with an asynchronous reset (rst) signal to start the BCH ED operations. Define the 7-bit data input-7f, 79, 4f, and 4C in a sequence with delay. The 15-bit error signal is set by corrupting two random bits (Number of 1's) for DEC in the design process. The 7-bit BCH decoder output is obtained after performing the BCH encoding-decoding operations with a Latency of 40.5 clock cycles. The received and transmitted bits are matched by correcting the 2-bit errors.

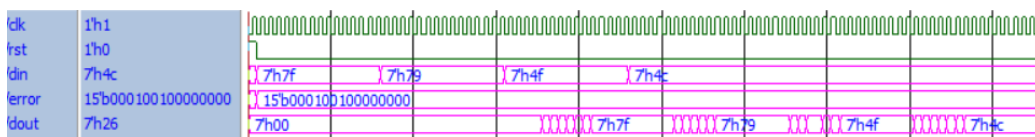


Figure 7. Simulation results of BCH-ED (15, 7, 2) module for DEC

The resource utilization parameters are considered after place and route operation in the Xilinx environment. The resource utilization of BCH-ED designs on Artix-7 FPGA is tabulated in Table 1 concerning the different data bits (k) and error corrections (t). Table 1 contains the realization of area, frequency, power and performance metrics for different BCH-ED designs.

The BCH-ED (15, 11, 1) design utilizes slices of 82, LUTs of 53, operates at 648.315 MHz, and consumes 85 mW of total Power. Similarly, The BCH-ED (15, 7, 2) design utilizes slices of 88, LUTs of 73, operates at 343.03 MHz, and consumes 86 mW of total Power. The BCH-ED (15, 5, 3) design utilizes slices of 96, LUTs of 91, operates at 482.346 MHz, and consumes 87 mW of total Power on Artix-7 FPGA.

The performance metrics are realized using Latency in terms of clock cycles (CC), Throughput (Gbps), and hardware efficiency (Mbps/Slice). The Throughput is calculated using data bits (k), maximum

obtained frequency (MHz), and a number of error bits to be corrected (t). So, Throughput = (data bits * frequency)/ number of error bits to be corrected. The system or hardware efficiency is calculated using obtained Throughput and utilized area (slices). The BCH-ED (15, 11, 1) design obtains the Throughput of 7.131 Gbps with a Latency of 44.5 CC and an efficiency of 86.96 Mbps/Slice. Similarly, The BCH-ED (15, 7, 2) design obtains the Throughput of 1.2 Gbps with a Latency of 40.5 CC and an efficiency of 13.63 Mbps/Slice. The BCH-ED (15, 5, 3) design obtains a Throughput of 0.803 Gbps with a latency of 67.5 CC and an efficiency of 8.364 Mbps/Slice. The performance metrics realization of BCH ED on Artix-7 FPGA is represented in Figure 8. As the ‘t’ bit increases, the chip area and Power of the BCH designs will also increase. The BCH-ED (15, 5, 3) for TEC utilizes more clock cycles and Power due to the advanced decoding mechanism. As the ‘t’ bit increases, the Throughput and hardware efficiency will be decreased drastically.

The resource utilization of individual BCH encoders and decoders for SEC, DEC, and TEC are tabulated in Table 2, and the graphical representation is in Figure 9. The BCH encoder (15, 11, 1) utilizes slices of 10 and operates at 605.18 MHz with a combinational delay of 0.729 ns. The BCH encoder (15, 7, 2) utilizes slices of 14 and operates at 866.927 MHz with a delay of 0.756 ns. The BCH encoder (15, 5, 3) utilizes slices of 16 and operates at 866.927 MHz with a delay of 0.771 ns. In contrast, the BCH decoder (15, 11, 1) utilizes slices of 35, LUTs of 18, and operates at 642.756 MHz. Similarly, the BCH decoder (15, 7, 2) utilizes slices of 43, LUTs of 41, and operates at 342.454 MHz. The BCH decoder (15, 5, 3) utilizes slices of 47, LUTs of 52, and operates at 482.346 MHz.

Table 1. Resource utilization of BCH-ED on Artix-7 FPGA

Resources utilized	BCH (15,11,1)	BCH (15,7,2)	BCH (15,5,3)
	Area		
Slice registers	82	88	96
Slice-LUTs	53	73	91
LUT FF-Pairs	51	59	70
Max. frequency (MHz)	648.315	343.03	482.346
Total power (mW)	85	86	87
Latency (CC)	44.5	40.5	67.5
Throughput (Gbps)	7.131	1.2	0.803
Hardware efficiency (Mbps/Slice)	86.96	13.63	8.364

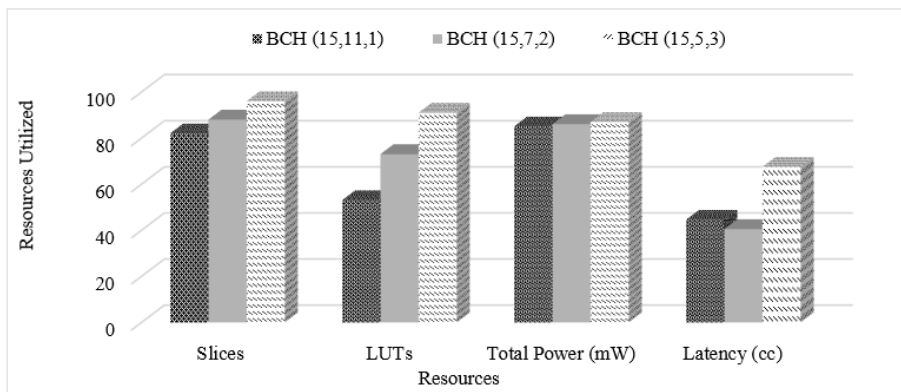


Figure 8. Performance metrics realization of BCH ED on Artix-7 FPGA

Table 2. Resource utilization of BCH encoder and decoders

Resources	(15,11,1)	(15,7,2)	(15,5,3)
BCH Encoders			
Slices	10	14	16
LUTs	10	11	13
LUT-FFs	9	10	12
Max. Frequency (MHz)	605.18	866.927	866.927
Comb. Delay (ns)	0.729	0.756	0.771
BCH Decoders			
Slices	33	43	47
LUTs	18	41	52
LUT-FFs	16	26	33
Max. Frequency (MHz)	642.756	342.454	482.346

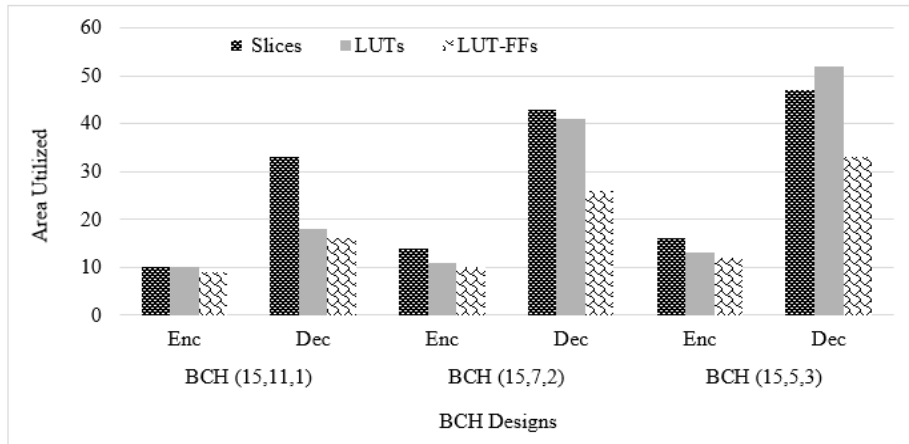


Figure 9. Chip area utilization of BCH encoder and decoders on artix-7 FPGA

The performance comparison of the proposed BCH ED with existing BCH approaches is tabulated in Table 3. The Parameters like the selection of the FPGAs, Chip area (Slices, LUTs and LUT-FFs), and Frequency and Throughput are considered for performance comparison of BCH codes. The BCH (255, 131, 1) with SEC is designed with an inversion-free BMA algorithm [7] on Artix-7 FPGA. The BCH module works at 200 MHz and obtains a Throughput of 5.1 Gbps. The proposed BCH –ED with SEC reduces the area overhead by 69.4 % concerning the slices, improves the frequency by 69.13 %, and has a Throughput of 28.48 % than the existing BCH design [7]. The BCH with SEC [8] is implemented on Virtex-5 FPGA. The proposed BCH –ED with SEC reduces the area overhead by 19.6 % concerning the slices, improving the frequency by 76 % and Throughput by 67.7 % than the existing BCH design [8]. The BCH codes [11], [14] [30] are implemented on Kintex-7 FPGA. The proposed BCH –ED with SEC improving the Throughput by 98 % , 47.44 % and 75 % than the existing BCH [11], BCH [14] and BCH design [30] respectively on Kintex-7 FPGA. The BCH with SEC [25] is implemented on Virtex-7 FPGA. The proposed BCH –ED with SEC reduces the area overhead by 87.5 % concerning the LUTs, improving the frequency by 58.3 % and Throughput by 98% than the existing BCH design [25] on Virtex-7 FPGA. The proposed BCH-ED (15, 11, 1) uses a simple decoding mechanism. The error location polynomial coefficients are assigned without algorithms like BMA to solve the key equations in the BCH decoder with SEC.

Table 3. Performance comparison of proposed BCH Code with existing BCH approaches

Parameters	(n, k, t)	FPGA	Slice	LUTs	LUT-FFs	Frequency (MHz)	Throughput (Gbps)
BCH [7]	(255,131,1)	Artix-7	268	171	158	200	5.1
This work	(15,11,1)	Airtex-7	82	53	51	648.315	7.131
BCH [8]	(15,11,1)	Virtex-5	102	77	68	114.771	1.7
This work	(15,11,1)	Virtex-5	83	52	48	479.272	5.27
BCH [11]	(15,11,1)	Kintex-7	379	316	NA	100	0.115
BCH [14]	(255,191,1)	Kintex-7	NA	726	374	434.93	4.63
BCH [30]	NA	Kintex-7	NA	1313	394	275	2.2
This work	(15,11,1)	Kintex-7	83	53	51	800.288	8.81
BCH [25]	(15,11,1)	Virtex -7	NA	426	84	333	0.111
This work	(15,11,1)	Virtex -7	83	53	51	800.288	8.81

4. CONCLUSION

This manuscript describes the efficient BCH architecture with an encoding-decoding mechanism for multi-bit error correction on the FPGA platform. The BCH module can detect and correct single, double, and triple errors in this work. The BCH encoder is designed using LFSR. The BCH decoder with SEC and DEC is designed without using the BMA approach to improve the chip area. The BCH decoder with TEC is designed using an inversion-based BMA approach. The BMA with an inversion approach reduces the BMA calculation time. The resource utilization of individual BCH-encoder and decoder modules with SEC, DEC, and TEC are realized in detail. The BCH module with SEC, DEC, and TEC utilizes < 1 % chip area and low-power consumption on the chip. The Throughput of 7.14 Gbps, 1.2 Gbps, and 0.803 Gbps is obtained for the BCH module with SEC, DEC, and TEC, respectively. The BCH module is compared with an existing similar

approach with improved chip area (slices and LUTs), frequency, and Throughput. This module will be integrated with LDPC codes to construct the forward error correction (FEC) transceiver for suitable, efficient data communication.




REFERENCES

- [1] G. Tzimpragos, C. Kachris, I. B. Djordjevic, M. Cvijetic, D. Soudris, and I. Tomkos, "A survey on FEC codes for 100 G and beyond optical networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 209–221, 2016, doi: 10.1109/COMST.2014.2361754.
- [2] A. Badr, A. Khisti, W.-T. Tan, and J. Apostolopoulos, "Perfecting protection for interactive multimedia: a survey of forwarding error correction for low-delay interactive applications," *IEEE Signal Processing Magazine*, vol. 34, no. 2, pp. 95–113, Mar. 2017, doi: 10.1109/MSP.2016.2639062.
- [3] J. Rosenthal and F. V. York, "BCH convolutional codes," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1833–1844, 1999, doi: 10.1109/18.782104.
- [4] S. A. Aly, A. Klappenecker, and P. K. Sarvepalli, "On quantum and classical BCH codes," *IEEE Transactions on Information Theory*, vol. 53, no. 3, pp. 1183–1188, Mar. 2007, doi: 10.1109/TIT.2006.890730.
- [5] O. Gazi, *Forward Error Correction via Channel Coding*. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-33380-5.
- [6] C. Ding, "Parameters of several classes of BCH codes," *IEEE Transactions on Information Theory*, vol. 61, no. 10, pp. 5322–5330, Oct. 2015, doi: 10.1109/TIT.2015.2470251.
- [7] D. Azinovic, K. Tittelbach-Helmrich, and Z. Stamenkovic, "Performance investigation on BCH codec implementations," in *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, IEEE, Dec. 2016, pp. 280–285. doi: 10.1109/ISSPIT.2016.7886049.
- [8] H. Setiawan, "A low-complexity and high-throughput RTL design of a BCH (15,7) decoder," *ITB Journal of Information and Communication Technology*, vol. 6, no. 2, pp. 112–130, 2012, doi: 10.5614/itbj.ict.2012.6.2.2.
- [9] B. Jarvis and K. Gaj, "Selection of an error-correcting code for FPGA-based physical unclonable functions," in *2017 International Conference on Field Programmable Technology (ICFPT)*, IEEE, Dec. 2017, pp. 243–246. doi: 10.1109/FPT.2017.8280151.
- [10] S. Ning, "Advanced bit flip concatenates BCH code demonstrates 0.93% correctable BER and faster decoding on (36 864, 32 768) emerging memories," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4404–4412, Dec. 2018, doi: 10.1109/TCSI.2018.2840350.
- [11] S. Mandal, Y. Tavva, D. Bhattacharjee, and A. Chattopadhyay, "ReRAM based in-memory computation of single bit error correcting BCH code," in *IFIP/IEEE International Conference on Very Large Scale Integration - System on a Chip*, 2019, pp. 128–146. doi: 10.1007/978-3-030-23425-6_7.
- [12] S. Choi, H. K. Ahn, B. K. Song, J. P. Kim, S. H. Kang, and S.-O. Jung, "A decoder for short BCH codes with high decoding efficiency and low power for emerging memories," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 2, pp. 387–397, Feb. 2019, doi: 10.1109/TVLSI.2018.2877147.
- [13] E. Chand, C. A. Reddy, and M. Aswathakumara, "FEC & BCH: study and implementation on VHDL," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, Mar. 2019, pp. 1099–1101. doi: 10.1109/ICCMC.2019.8819823.
- [14] A. Gunasekaran, N. C. Jose, and S. Srinivasa Garani, "An optimized BCH decoder design architecture for adaptive M-ary recording channels," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, IEEE, Aug. 2019, pp. 710–713. doi: 10.1109/MWSCAS.2019.8884872.
- [15] Yi Cai *et al.*, "FPGA assisted design of concatenated LDPC convolutional and BCH codes for optical fiber communications," in *45th European Conference on Optical Communication (ECOC 2019)*, Institution of Engineering and Technology, 2019, pp. 48 (4 pp.)-48 (4 pp.). doi: 10.1049/cp.2019.0782.
- [16] A. Subbiah and T. Ogunfunmi, "A flexible hybrid BCH decoder for modern NAND flash memories using general purpose graphical processing units (GPGPUs)," *Micromachines*, vol. 10, no. 6, p. 365, May 2019, doi: 10.3390/mi10060365.
- [17] M. Hiller, L. Kürzinger, and G. Sigl, "Review of error correction for PUFs and evaluation on state-of-the-art FPGAs," *Journal of Cryptographic Engineering*, vol. 10, no. 3, pp. 229–247, Sep. 2020, doi: 10.1007/s13389-020-00223-w.
- [18] S. Kwon and D.-J. Shin, "Analysis of blind reconstruction of BCH codes," *Entropy*, vol. 22, no. 11, p. 1256, Nov. 2020, doi: 10.3390/e22111256.
- [19] L.-J. Saiz-Adalid, J. Gracia-Morán, D. Gil-Tomás, J.-C. Baraza-Calvo, and P.-J. Gil-Vicente, "Reducing the overhead of BCH codes: new double error correction codes," *Electronics*, vol. 9, no. 11, p. 1897, Nov. 2020, doi: 10.3390/electronics9111897.
- [20] P. Rao, P. Babshet, R. Arun Babu, and M. S. Sunita, "Encoder and adaptive decoder for a (15,6,2) DEC-TED BCH code," in *2020 IEEE 17th India Council International Conference (INDICON)*, IEEE, Dec. 2020, pp. 1–6. doi: 10.1109/INDICON49873.2020.9342357.
- [21] A. M. Mahdy, M. Abdelaziz, and M. H. A. El-Azeem, "Design and simulation of parallel BCH code with LDPC code for flash memories," in *2020 12th International Conference on Electrical Engineering (ICEENG)*, IEEE, Jul. 2020, pp. 196–199. doi: 10.1109/ICEENG45378.2020.9171743.
- [22] N. Nageen, Subhashini, and V. Bhatia, "An efficient FPGA implementation of turbo product code decoder with single and double error correction," in *2020 National Conference on Communications (NCC)*, IEEE, Feb. 2020, pp. 1–6. doi: 10.1109/NCC48643.2020.9055995.
- [23] A. Mondal and S. S. Garani, "Efficient hardware design architectures for BCH product codes in the frequency domain," in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, IEEE, Aug. 2020, pp. 703–706. doi: 10.1109/MWSCAS48704.2020.9184633.
- [24] A. Mondal and S. S. Garani, "Efficient hardware architectures for 2-D BCH codes in the frequency domain for two-dimensional data storage applications," *IEEE Transactions on Magnetics*, vol. 57, no. 5, pp. 1–14, May 2021, doi: 10.1109/TMAG.2021.3060807.
- [25] J. Freudenberger, D. Nicolas Bailon, and M. Safieh, "Reduced complexity hard- and soft-input BCH decoding with applications in concatenated codes," *IET Circuits, Devices & Systems*, vol. 15, no. 3, pp. 284–296, May 2021, doi: 10.1049/cds2.12026.
- [26] S. Matsenko *et al.*, "FPGA-implemented fractal decoder with forward error correction in short-reach optical interconnects," *Entropy*, vol. 24, no. 1, p. 122, Jan. 2022, doi: 10.3390/e24010122.




- [27] S. Lin and J. Daniel J. Costello, *Error control coding, Fundamentals and applications*. 1983. [Online]. Available: <https://pg024ec.files.wordpress.com/2013/09/error-control-coding-by-shu-lin.pdf>
- [28] R. E. Blahut, *Theory and practice of error control codes*. Addison-Wesley Publishing Company, 1983. [Online]. Available: https://books.google.co.id/books/about/Theory_and_Practice_of_Error_Control_Cod.html?id=vuVQAAAAMAAJ&redir_esc=y
- [29] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969, doi: 10.1109/TIT.1969.1054260.
- [30] G. Quintarelli, M. Bertolucci, and P. Nannipieri, "Design and implementation of a DVB-S2 reconfigurable datapath BCH Encoder for high data-rate payload data telemetry," *IEEE Access*, vol. 11, pp. 120281–120291, 2023, doi: 10.1109/ACCESS.2023.3327786.

BIOGRAPHIES OF AUTHORS



Rohith Puttaraju    is a Research Scholar at GITAM University, Bengaluru campus. He completed his BE from Sri Jagadguru Chandrashekaranaatha Swamiji Institute of Technology, (SJCIT) Chikkaballapura and his M. Tech from the Malnad College of Engineering, Hassan. His research interests are in the area of Forward Error Correcting (FEC) Codes, Hardware Architecture of FPGAs, Embedded Systems and 5G and beyond. He has contributed to 3 research journals and conference and an Indian patent. His research papers were published in leading journals and conferences. He can be contacted at email: rohithgowda1985@gmail.com.



Ramesha Muniyappa    is the Research Supervisor at GITAM University, Bengaluru campus. He completed his BE from the VTU, Belagavi, and M. Tech from Dayananda Sagar College of Engineering, Bangalore, and Ph.D. degrees from the GITAM University, Visakhapatnam. His research interests are in the area of the next-generation wireless communication system with special emphasis on various 5G technologies such as massive MIMO, mm-Wave, OFDM, FBMC, NOMA, and others. He has contributed to more than 15 research journals and Two Indian Patents. His research papers were published extensively in leading international journals and conferences. He can be contacted at email: rameshmalur037@gmail.com.