Received: 18 October 2019 | Revised: 18 December 2019 | Accepted: 5 January 2020

DOI: 10.1002/spe.2796

SPECIAL ISSUE PAPER

WILEY

An energy efficient and low overhead fault mitigation technique for internet of thing edge devices reliable on-chip communication

Muhammad Ibrahim[1] | **Naveed Khan Baloch**[1] | **Sheraz Anjum**[2] | **Yousaf Bin Zikria**[3] | **Sung Won Kim**[3]

[1]Computer Engineering, University of Engineering and Technology Taxila, Taxila, Pakistan

[2]Computer Science, COMSATS University Islamabad, Wah Campus, Wah Cantt, Pakistan

[3]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan, Korea

Correspondence
Yousaf Bin Zikria, Department of Information and Communication Engineering, Yeungnam University, 280 Daehak-Ro, Gyeongsan, Gyeongbuk 38541, Korea.
Email: yousafbinzikria@gmail.com

Funding information
ITRC, IITP-2019-2016-0-00313; National Research Foundation of Korea, 22A20130012814

Summary

Soft errors in network-on-chip (NoC) such as single bit upsets and multibit upsets cause hazardous effects such as congestion, deadlock, livelock, and corruption of data. Error-correcting codes (ECCs) are the best choices to handle these soft errors in links and memory buffers of NoC, which is the need of all modern systems, including internet of thing (IoT) edge devices. Many of these ECCs cannot correct both random and burst errors. Specific codes possess the correction and detection capability at the cost of an increase in area, latency, and energy. In this article, a coding technique is proposed by using a single error correction double error detection-triple adjacent error correction-six adjacent error detection (SEC-DED-TAEC-6AED) (24,16) I5, that provides both random and burst error fault tolerance for NoC. The proposed technique decreases the area, energy, and latency cost of the whole NoC. It also reduces the area overhead to 173.41% and 117.91% compare to joint crosstalk avoidance multiple error correction (JCAMEC) and joint crosstalk multiple error correction (JMEC), respectively. Besides, the delay overhead of the proposed technique reduces to 4.2% and 91.97% compared with JCAMEC and JMEC, respectively. The simulation results show that the proposed code possesses an enhanced ability of error correction and detection with 3.5 times less redundant bits and a 30% fast code rate compared with JMEC and JCAMEC. Hence, the proposed scheme can effectively be used for detecting and correcting single and multiple bit errors for on-chip communication.

KEYWORDS

burst error correction, ECC, edge devices, fault tolerance, internet of things, links crosstalk, memory buffer, network-on-chip, reliability

Equally contributing First authors.

1 | INTRODUCTION

The internet of things (IoT) is the mechanism of getting the data from the environment using sensors and sending it to the remote server (cloud).[1,2] All the intelligence on the data is produced on the cloud with the help of machine learning algorithms and artificial intelligence. Recently researchers move the trend of processing the data from cloud to Fog computing and then to Edge computing.[3] So, edge computing is the process of sensing the data on the node (end) device and processing it on the same device. There is many difference in computation power for cloud and edge devices; cloud servers can process the information so fast compared with the edge devices. So there is a need for computationally intensive edge devices that can perform the computation.[4] Network-on-chip (NoC) can play a major role in the construction of such devices by providing the communication backbone to the system on chips (SoC) designs and by connecting many computation devices on the same chip.[5] These edge devices must be error-free or fault-tolerant because they now develop wisdom from the raw data.

NoC provides the communication backbone[6] to overcome the challenges of modularity, scalability, and efficiency faced by the traditional bus-based SoC.[7] Because of the latest advancement in a very large-scale integration technology and the shrinking size of transistors, we can put more logic in a single chip. We can also restate Moor's law as "The number of cores on a chip doubles every 18 months."[8] NoC design method improves the scalability issue of new chips by empowering them to integrate an increasing number of intellectual properties (IPs) on the same silicon die. It also improves the power efficiency of today's complex SoC's by using packet switching for on-chip communication. With the shrinking size of the transistors, integrated circuits are becoming more vulnerable to both permanent and soft errors.[7] Permanent errors are caused by physical damage in the circuit during the manufacturing process. It also occurs because of the aging effect and can never be healed. Soft errors are mainly triggered by glitches, process variation, crosstalk coupling,[9] an alpha particle produced single event upsets (SEU),[10] and/or single event transients.[11] When a high energy neutron and alpha particles originated from the impurities of the packaging material hit a semiconductor vicinity of a circuit, they induce a current pulse term as a SEU in the signal value. According to the author,[12] neutrons generate more error than impurities present in the packaging material. The probability of soft error[13] in the aerospace applications and on the top of mountains is ten times greater than sea level.[14] The lengthy link wires inside the NoC are affected by electromagnetic interference. Moreover, the network on the chip also interferes with the power fluctuations and crosstalk problem. Crosstalk is becoming one of the major noise sources because it indirectly induces peak noise voltage.[15] The fundamental components of NoC architecture are the router, network interface (NI) link, and processing element (PE). Routers are used to connect IP cores via NIs and links. With the aggression in technology scaling, the wires used to connect the routers are becoming narrower to each other. Equation (1) gives the fault probability η for the link wires that increases with the increase in the link width ω,[16] where ε represent the bit error rate. Multiple bit errors induced in links due to the causes mentioned above are becoming one of the challenging problems in on-chip networks.[16,17]

$$\eta = 1 - (1 - \varepsilon)^{\omega}. \tag{1}$$

These faults may cause a severe effect on the network that includes data corruption, congestion, and deadlocks. The packet may be redirected to the wrong destinations due to a single bit of fault in the header flit. This problem may eventually become the main cause of increasing latency and deadlock in the network. Moreover, retransmission is not possible if the source router information is corrupted. Therefore, the information in the header flit must be protected from soft faults. Besides, multiple bits may be corrupted inside the data flit while traveling from one router to the other due to crosstalk. Thus, there must be a mechanism to protect the header, and the data flits to get full advantage of the on-chip communication architecture. In this article, we have proposed an encoding-decoding technique using single error correction double error detection-triple adjacent error correction-six adjacent error detection (SEC-DED-TAEC-6AED) (24,16) I5[18] coding scheme. First, we divide a 64-bit flit into four rows with each row having a 16-bit length. Flit is the message bits that are generated by any IP core. The error correction capability of the used (SEC-DED-TAEC-6AED) (24,16) I5 coding scheme is three bit for a 16-bit message vector. Therefore, the error correction capability of the proposed code becomes 12-bits for a 64-bit flit. The proposed technique can correct single, double, as well as triple errors simultaneously compared with the functionality of Hsiao[19] and Hamming[20] codes. The full form of all the acronyms used in this is shown in Table 1. In short, the main contributions of this article are as follows.

TABLE 1 Acronyms used in the article

Short Form	Full Form
NoC	Network-on-chip
SEC-DED-3AEC-6AED	Single error correction-double error detection-three adjacent error correction-six adjacent error detection
IoT	Internet of things
JCAMEC	Joint crosstalk avoidance multiple error correction
JMEC	Joint multiple error correction
JTEC	Joint triple error correction
JCAMEC	Joint crosstalk avoidance multiple error correction
SEC-DED	Single error correction-double error detection
SoC	System on chip
VLSI	Very large-scale integration
FEC	Forwarding error correction
ARQ	Automatic repeat request
HARQ	Hybrid ARQ

1. We proposed a new coding technique, which is capable of handling both single and multibit adjacent bit errors simultaneously at the cost of less number of parity bits.
2. We also proposed a new encoder-decoder placement methodology for NoC architecture.

The rest of the article is organized as follows. Section 2 presents related work. Section 3 is about the proposed work. Section 4 reveals the encoder and decoder placement. Section 5 summarizes the performance evaluation. Section 6 concludes the article.

2 | RELATED WORK

In the NoC router and IoT devices, the internal links and buffers are the most affected components due to soft errors. Different soft error avoidance techniques are proposed for NoC, including information redundancy,[13,24] spatial redundancy,[20,25] and temporal redundancy.[13,26] In general, spatial redundancy is used for handling permanent faults, and temporal redundancy is used for mitigating transient and intermittent faults. Information redundancy can help us with tolerating all types of faults. Multiple faults can occur simultaneously in NoC. Therefore, a combination of redundancy techniques is required to address them all. Fault diagnosis tells us about the fault location and classifies fault as a temporal, information, or spatial fault. Due to easy implementation and simplicity, forward error detection and error-correcting code (ECC) are the most used techniques to handle the buffer and link faults in the NoC architecture. Most researchers have used the Hamming code for error correction.[20] The Hamming code can detect two-bit errors, but can only correct a single bit error. Hsiao SEC-DED has been proposed.[19] This code can handle both permanent and transient faults. However, a limited number of errors can be tolerated, but one cannot agree on this when permanent faults accumulate with time. To detect burst error, a cyclic redundancy code was introduced in Reference 27. The disadvantages of this scheme include complex implementation, and it can only handle transient faults. It also cannot locate the position of the fault. More complex code such as binary Bose-Chaudhuri-Hocquenghem has been used in Reference 28, this code can correct more than one bit at the cost of a large area and power consumption. More powerful code such as joint triple error correction-quadruple error detection (JTEC-QED) was introduced.[23] The advantages of this code are simultaneous triple error correction and quadruple error detection, energy efficiency, and lower latency. The disadvantages of this scheme are complex encoder and decoder design, area overhead, and more power consumption. Joint triple error correction-simultaneous quadruple error detection (JTEC-SQED) was enhanced by introducing lower power consumption.[29] The proposed adaptive crosstalk-aware multibit error control code to further decrease the bus power

TABLE 2 Related coding techniques with their features and differences

S.No	Coding Tech	Flit Size	Features	Differences
1	Hamming	(12, 8)	Single error correction (only transient faults)	Unable to correct more than one-bit error
2	Hsiao SEC-DED	(13, 8)	Single error correction double error detection (transient permanent faults)	Limited number of error correction and detection
3	CRC	Depend	Can handle only transient faults	Complex implementation
4	JMEC[21]	(176,64)	Multiple error correction and detection	Complex encoder/decoder design, a usage of more parity bits than data bits
5	JCAMEC[22]	(176,64)	Multiple error correction and detection, adjacent error correction	Complex encoder/decoder design, a usage of more parity bits than data bits
6	JTEC-QED[23]	(145,64))	Three-bit error correction, four adjacent bit error detection, power efficient	Complex encoder/decoder design, a usage of more parity bits than data bits

consumption by providing suitable communication resiliency based on the runtime noise level. Because of its multilevel architecture, the encoder and decoder implementation is very complex. The authors[30] proposed a new joint scheme providing seven error detection at the cost of no error correction. In Reference 31, the authors proposed an encoding scheme to reduce the link power consumption of NoC. But the authors have not cleared that if the number of error bits increases, then how the proposed method will handle it. The authors,[24,32] combined the error detection techniques with forwarding error correction, automatic repeat request (ARQ), and hybrid ARQ for error correction. But the overhead of this scheme is increased latency and encoder-decoder complexity Recently the authors[22] proposed a technique called joint crosstalk avoidance multiple error correction (JCAMEC). They have divided the flit into three rows. Each row is called a group, and each column a vector. Parity check bits are computed for each column, and three groups encoded with extended hamming coding techniques can correct multibit errors. They added 72 extra bits for 32-bit flit size. The author[21] proposed a technique called joint crosstalk multiple error correction (JMEC), which is a combination of odd-weight column codes and duplication of the encoded bits. Like JCAMEC they also divided the flit into four parallel rows where each row is encoded with Hsiao (13, 8), and then duplication takes place. The problem with these techniques is the use of 72 additional bits for 32-bit flit size.

The authors show[33] that besides standard effects of soft errors such as packet loss and rerouting, and there exist static effects as well. These static effects may lead to the continuous corruption of bits and blocking of the whole system during the operational time. A single fault may paralyze the whole chip, and it is absolutely necessary to find the mechanism to tolerate (transient) errors. Different authors have proposed different techniques that can correct more than triple errors, but are highly complex, require more area overhead, more power consumption, and increase the average latency of the system. Therefore, there is a need to have a coding technique, which can correct and detect all type of multiple and single-bit errors. the coding technique must have a minimum number of redundant bits and have less effect on the average latency of the system. The proposed technique in Section 3 is less complicated, takes less area, and also comprise less delay. This technique can correct burst error and next bit error at the cost of just 32 extra bits for 64-bit flit size. Table 2 shows a summary of related coding techniques.

3 | PROPOSED TECHNIQUES

The proposed technique is based on the SEC-DED error correction code subclass for adjacent MBU tolerance.[34] The proposed coding scheme has the ability to increase the reliability of on-chip networks at a low cost.

3.1 | Introduction to SEC-DED-xAEC-yAED

The SEC-DED-xAEC-yAED codes presented[18] can detect and correct soft errors in the static random access memory. These codes offer the traditional SEC-DED functionality, as provided by Hamming and Hsiao codes. These codes offer

FIGURE 1 Generalized parity generator matrix

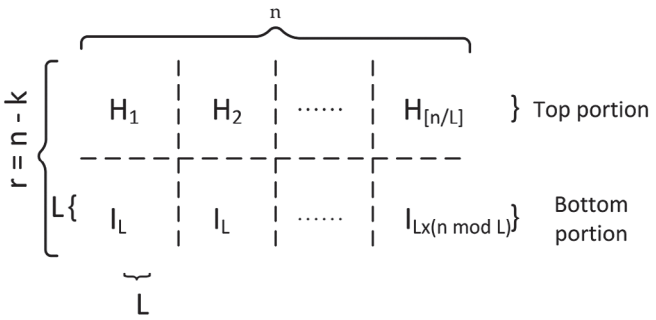
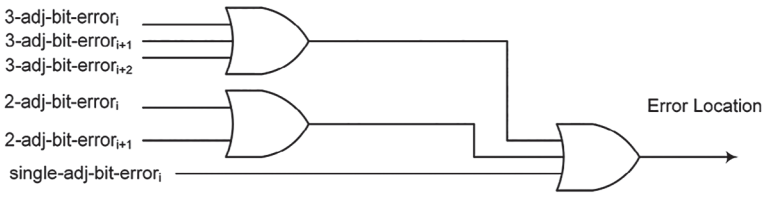


FIGURE 2 Syndrome decoder circuit for SEC-DED-TAEC-6AED. SEC-DED-TAEC-6AED, single error correction double error detection-triple adjacent error correction-six adjacent error detection



scalable adjacent and burst error detection (yAED, zBED), the option for TAEC, and reduction in misscorrection probabilities compared with other codes. Besides a SEC-DED, these codes emphasized on adjacent error correction and adjacent error detection because these type of MBUs form the primary error pattern that limits the efficiency of SEC-DED ECCs in scaled links and memory technologies. The generalized parity check matrix structure for the class of code is shown in Figure 1. It follows the conventional (n; k) nomenclature appended with an identity matrix size, IL. H-matrix structure uses the repeated identity matrix configuration as used in the Reed-Solomon matrix, along with a series of Hi submatrices that has each been designed using a selective bit-placement strategy as did in Dutta codes.[35] The matrix is divided into two separate portions (top and bottom), where each one is responsible for providing different behavior. The bottom portion consists of a series of horizontally L × L concatenated identity matrices, IL, abridged to fit the n-bit codeword size. This generalized matrix is defined in Reference 34. The encoding/decoding processes for the proposed codes use standard XOR logic for the check and syndrome bit generation in a manner identical to the Hamming and Hsiao SEC-DED schemes. The syndrome decoder logic has been modified to include correctable error matching signals for each of the adjacent bit upset syndrome patterns. By ORing the error matching signals together for each code-bit error location, the syndrome decoder indicates, which particular bits have been involved in an upset. For the SEC-DED-TAEC-yAED codes, the syndrome decoder circuit is modified, as shown in Figure 2. Since these codes can correct triple adjacent bit errors, there are $n - 2$ additional error-correcting syndrome patterns to match. The ith bit of the n-bit codeword will be corrected under one of any six conditions. These are if: (i) a single-bit error occurred at the ith bit, (ii) a double adjacent bit error occurred at either the $(i - 1, i)$, or (iii) $(i, i + 1)$ bits, or (iv) a triple adjacent bit error occurred at the $(i - 2, i - 1, i)$, (v) $(i - 1, i, i + 1)$, or (vi) $(i, i + 1, i + 2)$ bits. By ORING the error matching signals corresponding to each of these syndrome patterns, the upset of a particular code-bit can be determined by the syndrome decoder.

3.2 | SEC-DED-TAEC-6AED (24,16) I5

SEC-DED-TAEC-6AED (24,16) I5 Code is one of the above-mentioned classes of codes. In addition to SEC-DED capability, this code can correct TAEC and six adjacent error detection. First, the required number of check-bits, "r" must satisfy the relation of Equation (2).

$$2^{(r-L)} \geq [n/L], \tag{2}$$

where r is the number of redundant bits, n is the length of the codeword, and L is the size of the repeated identity matrix ensures that there are a sufficient number of check-bits such that each complete column is unique.

3.3 | How SEC-DED-TAEC-6AED (24,16) I5 works

This section provides an encoding and decoding example for the SEC-DED-TAEC-6AED (24,16) I5 code in the event of a three-adjacent bit error. In this example, we assume that data word $d = (1010\ 1010\ 1010\ 1010)$ is to be transmitted, and the three-adjacent errors $e = (00011\ 10000\ 00000\ 00000\ 0000)$ are injected into the code word V during transmission. The following Equation (3) is used for the calculation of check bits.

$$C = V * G^T, \tag{3}$$

where V is the codeword and G^T is the transpose of the generator matrix. Check bits can be calculated using the following equations.

$$\begin{aligned}
c_1 &= d_1 + d_2 + d_2 + d_3 + d_4 + d_5 + d_6 + d_{14} + d_{15} \\
c_2 &= d_1 + d_3 + d_9 + d_{11} + d_{13} + d_{16} \\
c_3 &= d_7 + d_8 + d_{10} + d_{12} + d_{14} + d_{15} \\
c_4 &= c_1 + d_7 + d_9 + d_{14} \\
c_5 &= d_1 + d_5 + d_5 + d_{10} + c_2 \\
c_6 &= d_2 + c_3 + d_{11} + d_{15} \\
c_7 &= d_3 + d_6 + d_{12} + d_{16} \\
c_8 &= d_4 + d_8 + d_{13}
\end{aligned} \tag{4}$$

Using equation the computed check bits for the given data word comes out to be $C = (010\ 01011)$. The arrangement of the SEC-DED-TAEC-6AED (24, 16) I5 codeword is given as $V =$ (c1d1d2d3d4c4d5c6d6c8 d7c5c3c7d8 d9d10d11d12d13 d14c2d15d16). By combining the computed check bits with the given data word d the obtain codeword V becomes $V =$ (01010 01001 11010 10101 0110). The three-adjacent errors injected into the transmitted codeword V can be represented by XORing the error vector e with the transmitted codeword V. The received codeword (rc) given in Equation (5) then mathematically describes the data corruption process.

$$\text{rc} = V \oplus e. \tag{5}$$

After XORing e with V the received code becomes rc = (01001 11001 11010 10101 0110). Decoding takes place at the receiver-side, where the syndrome is calculated by the following Equation (6).

$$S = rc \oplus H^T. \tag{6}$$

The calculated syndrome value becomes $S = (0101\ 0011)$. As S nonzero, it means the rc is faulty. The syndrome value matches the XOR combination of the fourth, fifth, and sixth columns in the H-matrix shown in Figure 3. The syndrome decoder translates the eight-bit syndrome pattern into the 24-bit error location vector as $E_{\text{LOC}} =$ (000111000000000000000000).The syndrome decoder is functionally correct, if the error location vector, E_{LOC}, is equal to the injected error vector e. Finally, bitwise XORing the error location vectorE_{LOC} with the rc, gives the corrected codeword, u, as defined below in Equation (7). Hence the value of u becomes (01010 01001 11010 10101 0110).

$$u = rc \oplus E_{\text{LOC}}. \tag{7}$$

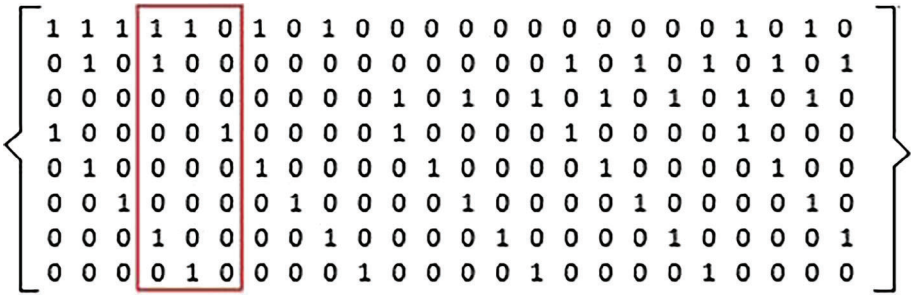


FIGURE 3 Parity check matrix for SEC-DED-TAEC-6AED (24, 16) I5. SEC-DED-TAEC-6AED, single error correction double error detection-triple adjacent error correction-six adjacent error detection

3.4 | Proposed ECC encoder and decoder design

The proposed coding scheme is a combination of small SEC-DED-TAEC-6AED encoder and SEC-DED-TAEC-6AED decoder, respectively. First the "n" bit data word is divided by "x," where "x" is taken equal to 4. As a result, the data word becomes in small pieces of data bits "k." As shown in Figure 5, after that, these "x" pieces of "k" bits are separately encoded by "x" no of SEC-DED-TAEC-6AED encoders. Each encoder will add "r" number of redundancy bits to each "k" bits to generate encoded data bits "i" for each chunk as given by Equation (8).

$$i = k + r. \tag{8}$$

The smaller encoded data "i" are then combined to get the total number of bits in ca codeword "N," As given in Equation (9).

$$N = x + i. \tag{9}$$

For a clear understanding, the complete flowchart of the encoding process is shown in Figure 4. At the receiver side, the rc is first separated in "x" number of chunks and then forwarded to the SEC-DED-TAEC-6AED decoders separately, as shown in Figure 6. The decoder decodes the received data and generates the syndrome bits to check whether it equal to zero or not. The proposed technique can correct a maximum of three times × errors and a minimum of three-adjacent errors. The data word N is taken to be 64 bits. Therefore, four numbers of encoders and decoders are required for the hardware design, and the smaller pieces of data k become 16 bits. Similarly, as in Figure 6, four chunks of data were feed to four different SEC-DED-TAEC-6AED encoders, as shown in Figure 5. Each encoder adds eight check bits to the 16-bit

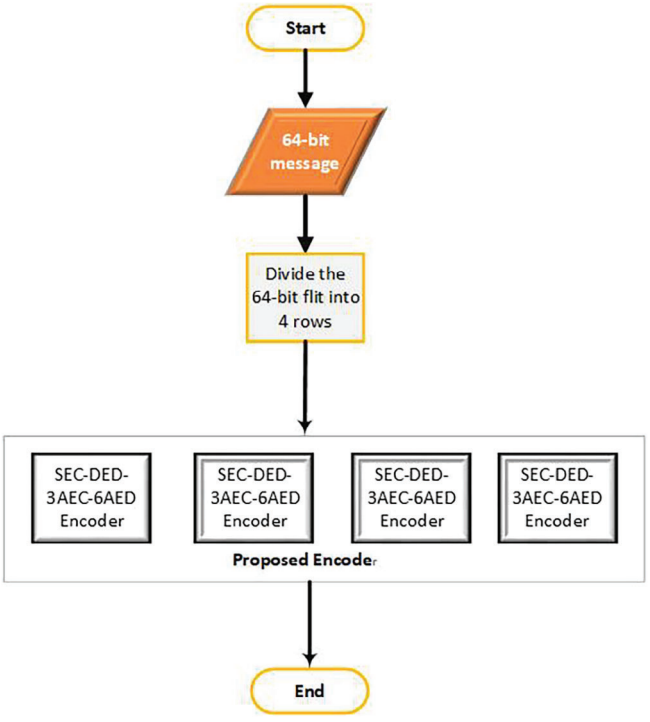


FIGURE 4 Flowchart of encoding process

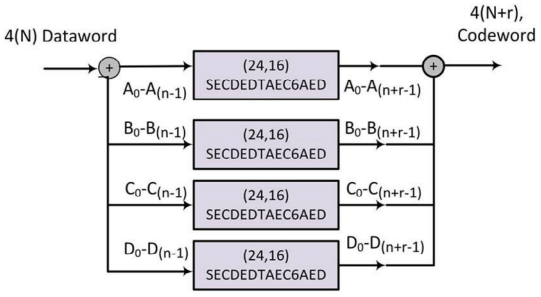


FIGURE 5 Proposed encoder for 64-bit flit

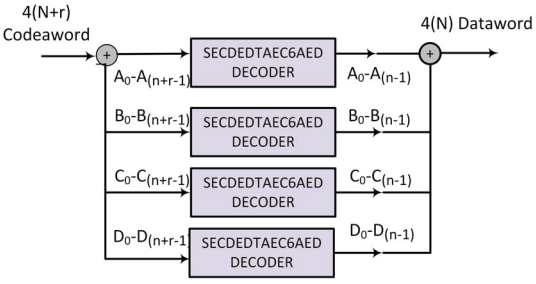


FIGURE 6 Proposed decoder

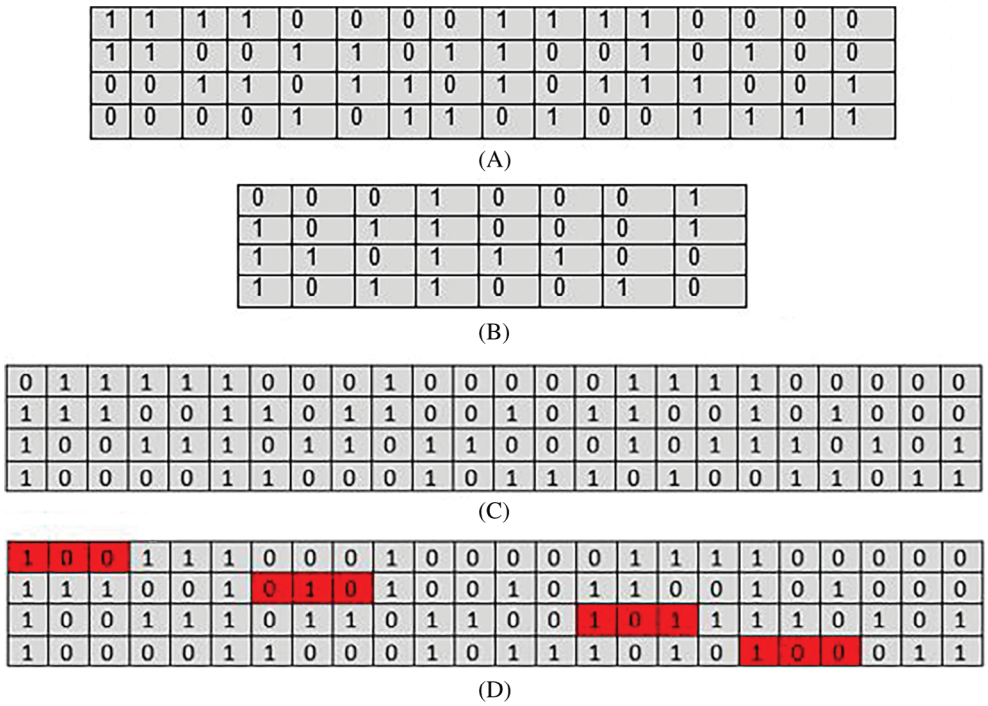


FIGURE 7 (A) 64-bit flit for NoC, (B) check bits, (C) a generated codeword of 96-bit size, and (D) the corrupted codeword after error injection. NoC, network-on-chip

size of data bits and generates 24-bit of the codeword. Therefore, the proposed encoder takes 64-bit of flit and produces a 96-bit of the codeword. At the receiver end, the 96-bit size of flit is received at the decoder, as shown in Figure 6, below. After passing through the decoder, the 64 bit corrected data word is generated.

3.5 | Encoding and decoding example for proposed method

In Section 3.4, the encoding and decoding process is explained for SEC-DED-TAEC-6AED (24, 16) I5. This example is included to show the error correction capability of the proposed technique. We have utilized the five-input and five-output virtual channel based router for 2D NoC. Data is communicated in the form of flits, which is a smaller unit of the packet. Flits are injected into the router from the local port and traverse the network to reach their destination router. Each flit consists of 64-bits. At the sender side, the encoder takes the 64-bit flit, data bits, as shown in Figure 7A below and encoded according to the proposed encoder. The generated check bits for each row of the given 64-bit data word is shown in Figure 7B. The input of the encoder is 64-bits flit, and the output is 96-bits flit that includes the 32 check bits, as shown in Figure 7C. Assume that during the transmission errors occurred at different positions and corrupt the encoded data, as shown in Figure 7D. The three adjacent bits errors for each row are shown in red color.

At the destination NI, this corrupted codeword is received and fed into the decoder, as shown in Figure 6. The decoder generates syndrome bits for the rc. If the syndrome matrix is equal to zero, then there is no error in the rc and extracts the original data word from the rc, and if the syndrome matrix is nonzero, then there can be three scenarios as given under; Checking the syndrome matrix for single error detection and correction.

1. Checking the syndrome matrix for two adjacent error detection and correction.
2. Checking the syndrome matrix for three adjacent error detection and correction.

FIGURE 8 Syndrome matrix

1	1	0	1	1	1	0	0
0	0	0	0	1	1	1	0
0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	1

In case of a single error, the techniques will work in a similar manner as in Hsiao and Hamming codes, but for the double and triple adjacent error, the scenarios will be different. For double adjacent bit error detection, the syndrome matrix will be compared with the XOR value of every two adjacent columns in the parity check matrix H. If syndrome matrix matches the XOR value of any two adjacent columns, then it means that there are two adjacent bit errors in the rc. For triple adjacent error detection and correction, the syndrome matrix is matched with XOR value of every three adjacent columns value in the parity check matrix. If the syndrome value matches the XOR value of any three adjacent columns, then it means that there are three bits of error at corresponding positions in the rc. The generated syndrome matrix for the above-rc is shown in Figure 8. From the above syndrome matrix shown in Figure 8, it is clear that syndrome matrix S is nonzero for each row, it means that there is an error in each row of the rc. To c4orrect the code word, the decoder will match every row of the syndrome matrix with the XOR value of every three adjacent columns in the parity check matrix. By matching every row of the syndrome matrix with parity check matrix H, the following results are produced.

1. Row no.1 of the syndrome matrix is the same as the XOR value of the first, second, and third columns of the parity check matrix.
2. Row no.2 of the syndrome matrix is the same as the XOR value of the seventh, eighth, and ninth columns of the parity check matrix.
3. Row no.3 of the syndrome matrix is the same as the XOR value of the 15th, 16th, and 17th columns of the parity check matrix.
4. Row no.4 of the syndrome matrix is the same as the XOR value of the 19th, 20th, and 21st columns of the parity check matrix.

These results shows that there is three adjacent bit error at position first, second, and third in row no.1, three adjacent bit error at position seventh, eighth, and ninth in row no.2, three adjacent bit error at position 15th, 16th, and 17th in row no.3, and three adjacent bit error at position 19th, 20th, and 21st in row no. 4 of received code word.

After error detection the decoder will generate error location vectors E_{LOC}, for each row, and will XOR with received code word, after Xoring the correct code word is produced. For each row E_{LOC}, are shown below.

$$E_{\mathrm{LOC1}} = 111000000000000000000000$$
$$E_{\mathrm{LOC2}} = 000000111000000000000000$$
$$E_{\mathrm{LOC3}} = 000000000000001110000000$$
$$E_{\mathrm{LOC4}} = 000000000000000000111000.$$

One can see here that we have injected 12 errors in the 64-bit size of flit and the decoder corrected back all the 12 errors and received the original code word. In addition to the single and double adjacent bit errors, the code can correct adjacent triple errors, which in turn increased the error correction capability up to 12-bit errors.

4 | ENCODER-DECODER PLACEMENT

4.1 | Introduction

The placement of encoder/decoder in the NoC architecture has a huge impact over the latency, throughput, area, energy consumption, and resilience against system errors. The author of Reference[30] has shown how encoder-decoder placement affects the overall NoC area, router area, and delay. They have shown that hop-2-hop (H2H) encoder-decoder placement has increased the router area by approximately 20%, and an increase in energy is 3.2% compared with E2E

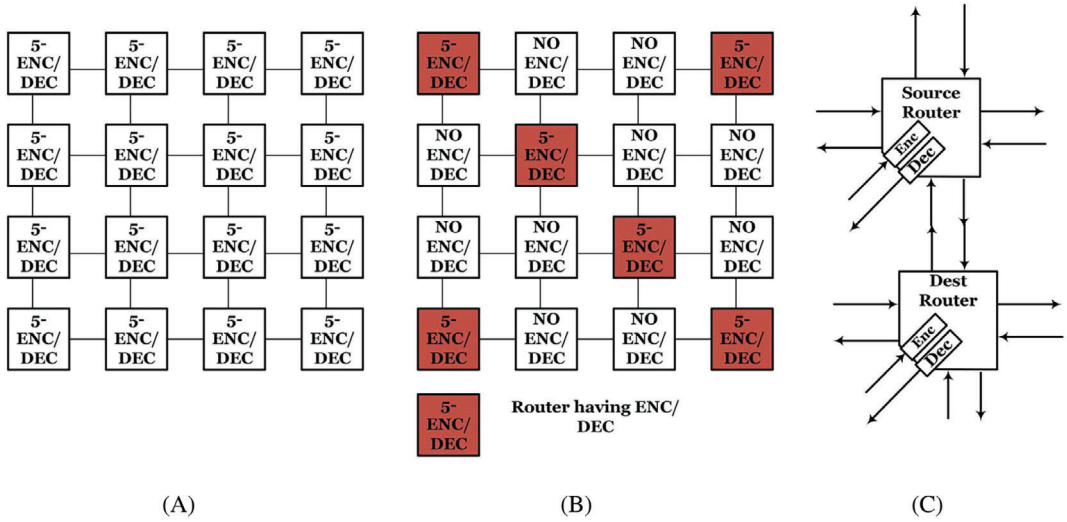


FIGURE 9 Encoder-decoder placement methodologies inside NoC architecture. (A) H2H, (B) selective H2H, and (C) E2E. E2E, end-to-end; H2H, hop-2-hop; NoC, network-on-chip

placement. Similarly, the NoC total area is increases by 13.3%. Researchers have proposed many encoder/decoder placement methods up till now. These methods include end-to-end (E2E),[36] H2H[23,37] or selective hop-to-hop placements. E2E is a method where encoding and decoding take place at a NI or source and destination nodes that comprise of minimum average network latency. In the case of H2H shown in Figure 12A, packets are encoded and decoded at every router. It is clear from Figure 9A that each switch has got five pairs of encoder-decoder, one encoder-decoder for each input-output port. Due to encoder-decoder pair at each input-output port of H2H placement, the maximum resistance against error can be achieved at the cost of maximum latency overhead. The second drawback of H2H is the huge increase in area overhead and delay of the network. To reduce the area and delay overhead, the researcher proposed a selective H2H placement method shown in Figure 9B. Instead of placing encoder/decoder at every router, they placed encoder/decoder at those routers, which are supposed to be more prone to faults. The ones with red color are the routers having encoder-decoder, and ones with light gray color are the routers having no encoder-decoder. There are many techniques for how these routers are selected for the inclusion of error correction circuitry. Figure 9C shows the E2E placement methodology. It is clear from the figure that in this methodology, the encoder and decoder are placed in every router, but only source and destination routers encoder-decoder are active to provide fault tolerance. The E2E codec mechanism has very less effect over baseline average latency because encoding-decoding takes place only at the source and destination routers.

4.2 | Proposed encoder-decoder placement methodology

In Reference 6, the author has placed the encoder-decoder in NI of each router and placing the detector at each input port of every router. We have modified the mentioned E2E methodology by removing the detectors and only placed the encoder-decoder in the NI of each router. In the modified architecture, we have removed the mux, sending/receiving controllers, and asynchronous FIFO, which can be seen in Figure 10. Due to our focus on SEU and crosstalk type of errors, the proposed ECC technique can correct up to 12 errors. Packets are only encoded at a NI of the source router and decoded at the destination routers NI to correct the errors. Figure 10A shows the big picture of a proposed placement methodology, and Figure 10B shows the modified microarchitecture of used[6] placement methodology. Figure 10A shows three parts, namely, PE, NI, and router with input-output buffers. The NI consists of the pack and unpacks units with the proposed encoder-decoder. At the source PE, NI takes 64-bit of data from the PE and passes it to encoder through the packing unit. The proposed encoder converts the 64-bit data into a 96-bit codeword and passes it to the router input buffer. At a destination router NI, the 96-bit flit is first passed through the proposed decoder. The decoder checks the flit whether the flit is effected by the fault or not. The error is corrected in case of received effected flit. Otherwise, the original flit is passed to the destination PE.

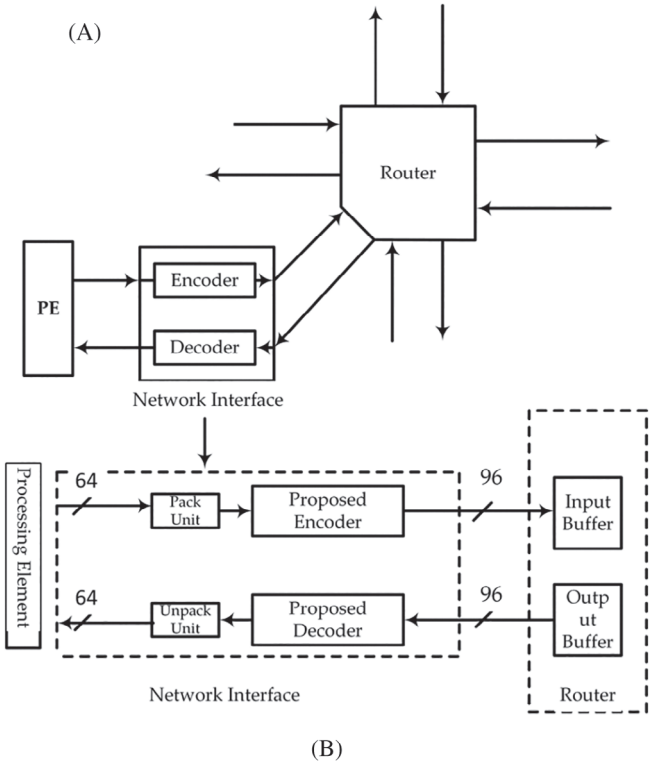


FIGURE 10 End-to-end placement methodology. (A) Big picture of E2E and (B) proposed NI microarchitecture. E2E, end-to-end; NI, network interface

5 | SIMULATION RESULTS AND HARDWARE OVERHEAD

5.1 | Simulation setup

This section discusses the network level performance, area overhead, error correction/detection capability, delay overhead, and miscorrection probability of the proposed code. The list of all evaluated parameters is shown in Table 3. The proposed codec and its competitors, namely, JMEC, JCAMEC, and JTEC are implemented in Verilog HDL and simulated using Gem5 (Garnet2.0) to evaluate the network performance parameters. The 8×8 NoC mesh topology having 64 nodes is considered. Every router has five ports and one cycle pipeline stage. By one cycle pipeline, we mean that all the stages are done in one cycle of the clock. Each NI injects 10 000 packets at an injection rate from 0.1 to 1 flit/NI/Cycle. Three different traffic patterns are applied to evaluate the performance of all the coding techniques. The list of network configuration parameters is also shown in Table 4. Following experiments and analysis, the flit size is 64-bit with 16-bit of chunk size of each row is assumed. The simulations for three adjacent bit error correction, six adjacent error detection, and SEC in each chunk of 16-bit is performed. All together, its adjacent error correction capability becomes 12 bits in addition to a SEC and error detection capability.

TABLE 3 List of simulation parameters

S.No	Parameter Name
1	Code rate
2	Bit overhead
3	Error correction and detection capability
4	Average packet latency
5	Area overhead
6	Delay overhead
7	Energy dissipation

TABLE 4 Network configuration parameters

Number of Nodes	64
Topology	8 × 8 NoC mesh topology
Number of input ports	5
Pipeline stages	One cycle pipeline stage
Injection rate	From 0.1 to 1 flit/NI/Cycle
Flit size	64-bit

5.2 | Fault model

For a single-bit error, the Gaussian pulse function, which has been used broadly, is taken as a model for bit error rate ε.

$$\varepsilon = Q\left(\frac{Vdd}{\sigma_N}\right) = \int_{\frac{Vdd}{2\sigma_N}}^{\infty} \frac{1}{\sqrt{2\pi}}^{e-y2}/2dy. \tag{10}$$

In the above equation, σN is the standard deviation from noise voltage, and Vdd is the supply voltage. Error is typically assumed to be statistically independent. As a result, the flit error rate for a K-bit flit can be modeled by the below equation.

$$\text{Flit error rate} = 1 - (1-\varepsilon)^K \approx K\varepsilon. \tag{11}$$

Due to the scaling of technology, the interconnect links become more susceptible to coupling noise, which is expected to increase in the future. According to Reference 18, fault event on a wire has probability β termed as the adjacent coupling coefficient of affecting neighboring wires. For ease of analysis of multibit errors, the probability that a flit contains three-bit adjacent errors is modeled by Equation (12), as given below.

$$P(3) = K \times \varepsilon \times (1-\varepsilon)^{(K-1)} \times \beta^2. \tag{12}$$

5.3 | Reliability of SEC-DED-TAEC-6AED (24, 16) I5

The reliability of coding technique can be found through miscorrection probability $P(e-\text{Random error})$, which is defined as the number of error patterns that are shared with error-correcting syndromes divided by the total number of error patterns producible for the given error type minus the number of error-correcting syndromes. The miscorrection probability of an $e-$ random error for a particular $(n; k)$ code is given by Equation (13).

$$P(e-\text{Random error}) = \frac{\text{Sharable syndromes}}{\binom{n}{e} e.\text{Sharable syndromes}}, \tag{13}$$

where sharable syndromes is the number of linear combinations of e columns vectors in the H-matrix that falsely produce either an error-correcting syndrome or the zero vector. This value is then divided by the total number of e-bit combinations for the given codeword size minus the number of e-column error-correcting syndrome patterns. Equation (14) gives the miscorrection probability of a nonadjacent triple error being masked as an adjacent triple error.

$$P(\text{Miscorrection probability}) = \frac{\text{Sharable syndromes}}{\binom{n}{3} e.\text{Sharable syndromes}}. \tag{14}$$

The miscorrection probability of the used coding scheme is compared with Dutta, and SEC-DED, and shown in Table 5.

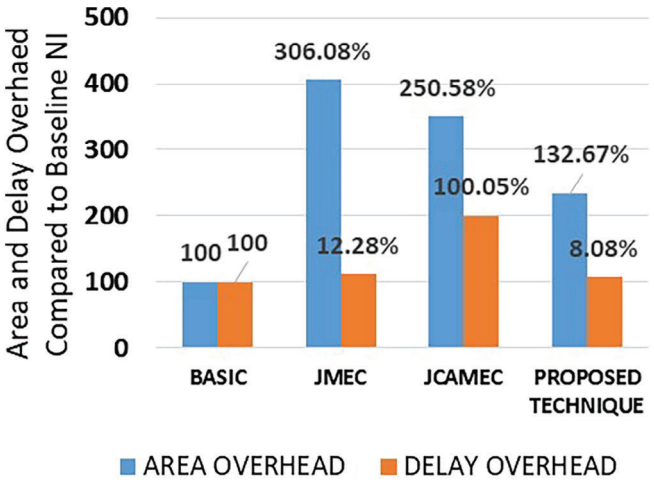
From the above table, it is clear that the misscorrection probability of the Dutta code is 65.7%, and that of Hsiao SEC-DED code is 65.2% while the proposed code has shown to reduce the probability to 39.4%. The SEC-DED code can correct the only one-bit error and is unable to correct multibit errors. Therefore, its misscorrection probability will increase

TABLE 5 Misscorrection probability of different ECCs

(n, k)	Misscorrection Probability P(3-Random)%
Topology	8 × 8 NoC mesh topology
(22,16) SEC-DED[30]	65.2
(22,16) DAEC	65.7
(24,16) SEC-DED-TAEC-6AED, Proposed	39.4

Abbreviations: ECCs, error-correcting codes; NoC, network-on-chip; SEC-DED-TAEC-6AED, single error correction double error detection-triple adjacent error correction-six adjacent error detection.

FIGURE 11 Area and delay overhead comparison of different error-correcting codes



with the increase in several error bits. Similarly, the Dutta code can correct only up to the two-bit error. Its misscorrection probability remains stable if the number of error bit is two or less. Still, if it increases more than two, then its miss correction probability will also increase.

5.4 | NI area and delay overhead

To evaluate the impact of ECC encoder and decoder on area and delay of NI, the baseline and the proposed version of NI was developed using Verilog HDL. The Cadence RTL compiler was used for synthesizing the codes by setting the frequency to 1 GHz and selecting 45 nm technology. The NI of the NoC router with different ECC's was implemented, and the results were compared with the baseline NI. The area and delay overhead of using different ECC with baseline NI are shown in Figure 11 below. The area overhead of JMEC and JCAMEC is 308.08% and 250.58%, respectively. The proposed technique comprised reduced area overhead, that is, 132.67%. If we look at the detailed view of JMEC and JCAMEC, they first use the hamming technique, after that duplication takes place and finally parity bit is added, which is the main reason of complex encoder and decoder design that in turn makes the area overhead too high, which is not acceptable for real-time systems. Similarly, the delay overhead of the JTEC[20] and JCAMEC are 12.28% and 100.05%, respectively, while the proposed technique has also reduced the delay to 8.08%, which is just almost 8% greater than the baseline NI.

5.5 | Code rate and bit overhead of different ECC

In this section, the proposed technique is compared with hamming, JMEC, and JCAMEC in terms of code rate and bit overhead. The code rate of any code can be calculated by k/n where k is useful bits and n the total number of bits in a codeword. Hamming code possesses the most code rate, but can correct a single error, which is likely not the best choice for the worst scenario. The proposed technique has a code rate of 66.66%, which is 30.3% greater than JMEC and JCAMEC, and 22.53% greater than JTEC-QED[20] as shown in Table 6, and Figure 12. Due to its higher code rate, it will affect the

Coding Tech	Code Rate (%)	Bit Overhead (%)	Flit Size
JMEC[21]	36.36	175	(176,64)
JCAMEC[22]	36.36	175	(176,64)
JTEC-QED[23]	44.13	126.56	(145,64)
Proposed scheme	66.66	50	(96,64)

TABLE 6 Code rate and bit comparison of different ECC

Abbreviations: ECC, error-correcting code; JCAMEC, joint crosstalk avoidance multiple error correction; JMEC, joint crosstalk multiple error correction; JTEC-QED, joint triple error correction-quadruple error detection.

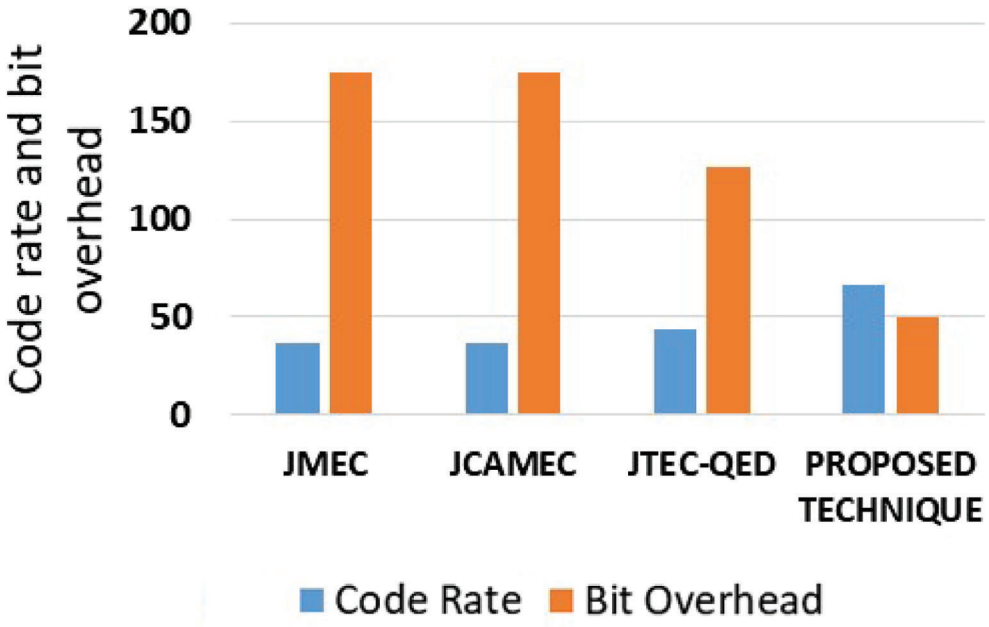


FIGURE 12 Code rate and bit overhead comparison of different error-correcting code

latency, not the throughput of the NoC router. The bit overhead of the proposed code is also compared with JMEC and JCAMEC. Bit overhead of any code can be calculated by r/k where r, is the number of redundant bits, and k is the number of useful bits. Hamming code has the smallest bit overhead compared with other codes, but the hamming code is unable to correct more than a single bit error. The proposed code has 3.5 times less bit overhead than JMEC and JCAMEC, as shown in Table 6 and Figure 12.

5.6 | Error correction and detection capability of different techniques

In this section, error correction and detection of the proposed code are compared with JMEC, JTEC-QED, and hamming SEC-DEC. The simulation shows that the proposed code can correct 12 burst errors in addition to the single-bit error in each row of 64-bit flit size. The proposed code is three-bit greater than JMEC for just 32 bit overhead instead of 72 bit overhead in JMEC and nine bits greater than JTEC-QED, which can correct just three-bit error at the cost of 81 extra bits. Proposed interleaving code can correct 12 random bits error having the same number of extra bits as in burst error correction. The proposed code also provides 24 bits of error detection capability, which is higher than JMEC and JTEC-QED. The results are tabulated in Table 7. Therefore, it can be concluded that for worst-case scenarios, the proposed code is the best option because it can correct and detect all types of error simultaneously having less redundant bits than JMEC, JTEC-QED, and JCAMEC. Table 8 shows the simultaneous error detection and correction capability of different ECC codes. Simulation results show that in addition to SEC-DED properties, the proposed code can correct the triple adjacent error, and detect the six adjacent errors. Table 6 clearly reveals that compared with other ECC SEC-DED, double-adjacent error correction, JMEC, and JCAMEC, only the proposed code can correct all types of errors simultaneously.

TABLE 7 Error detection and correction capability of different ECC

Coding Techniques	Adjacent Error Detection	Burst Error Correction	Random Error Detection	Single Error Correction
Hamming (72,64)	0	0	2	Yes
JMEC[21]	9	9	12	Yes
JTEC-QED[23]	0	0	3	Yes
Proposed scheme	24	12	12	Yes

Abbreviations: ECC, error-correcting code; JMEC, joint crosstalk multiple error correction; JTEC-QED, joint triple error correction-quadruple error detection.

TABLE 8 Simultaneous errors correction and detection capability of different ECC

Uncorrectable Error	Correction Type					
Error Type	NO ECC	SEC-DED	DAEC	JMEC	JCAMEC	Proposed Technique
Single	No	Yes	Yes	Yes	Yes	Yes
Double adjacent	No	No	Yes	Yes	Yes	Yes
Triple adjacent	No	No	No	Yes	Yes	Yes
Single+2-adj+3-adj	No	No	No	No	No	Yes

Abbreviations: DAEC, double-adjacent error correction; ECC, error-correcting code; JCAMEC, joint crosstalk avoidance multiple error correction; JMEC, joint crosstalk multiple error correction; SEC-DED, single error correction double error detection.

5.7 | Network level evaluation of proposed codec

For a fair comparison, all the codecs were, namely, JMEC, JCAMEC, and JTEC were implemented in Verilog HDL and evaluated for performance under three different traffic patterns, that is; uniform random, shuffle, and transpose. The latency results of all competitors and proposed codecs for injection rate varying from 0.1 to 1 flit/NI/cycle under different traffic patterns are shown in Figure 13. The mesh XY routing algorithm is considered. The switch architecture adopted has four functional stages, namely, input arbitration, rout computation, virtual channel allocation, and switch allocation. Each input port has four virtual channels, in which every virtual channel is four flits deep. Figure 13 shows the penalties of average packet latency for different coding schemes. From the figure, it is clear that JMEC comprised of the maximum latency overhead due to its complex encoder-decoder design and H2H encoder-decoder placement. JCAMEC and JTEC have almost the same latency for uniform random traffic. JCAMEC uses the E2E placement, but there overhead in average latency is due to the complex encoder/decoder design and usage of maximum parity bits for detection and correction. So we can say that encoder-decoder design and placement play a key role in average network latency. The simulation results show that the JMEC, JTEC, and JCAMEC have the same latency for shuffle and transpose traffic pattern approximately. These all schemes have considered the same pipeline stages, that is, Input arbitration, RC, VCA, and SA. JMEC and JTEC have used the H2H encoder-decoder placement methodology, and JCAMEC has used the E2E placement technique. Despite having the same pipeline stages and the same architecture, their latency is more than the proposed technique. This is because that the proposed technique uses the NI encoder-decoder placement. Compared with JMEC in all three traffic pattern proposed codec has reduced the latency by 10% and 5% reduction compared with JTEC and JCAMEC. The energy dissipations by all codecs under consideration are shown per cycle in Figure 14 for Mesh NoC against injection load. Injection rate or injection load can be defined as the number of flits injected by each PE into the network in each clock cycle. Figure 14 reveals the energy dissipated by all packets in the NoC per simulation cycle. The channel BER is assumed to be 10 to 20 in the simulation. As evident from Figure 14, the energy consumption of the uncoded is higher of all the codes. This is because of the retransmission of failure packets repeatedly. Due to H2H encoder/decoder placement and area overhead JMEC has the second higher energy consumption. JCAMEC has reduced the energy consumption compared with uncoded and JMEC as they use the E2E encoder/decoder placement. The proposed codec consumes the lowest energy per cycle in the network due to its less area overhead and NI encoder/decoder placement. From the figure, it is clear that the graph saturates beyond the injection rate of 0.3.

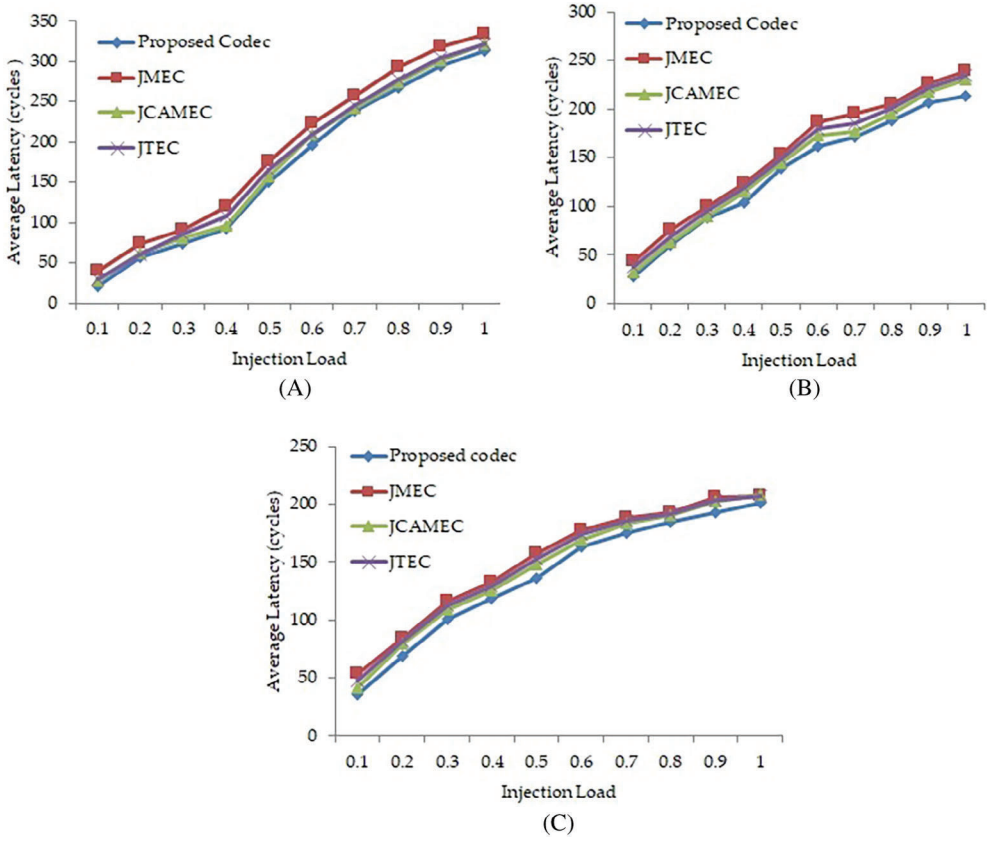


FIGURE 13 Average packet latency evaluation against different injection load for different traffic patterns. (A) Uniform random, (B) shuffle, and (C) transpose

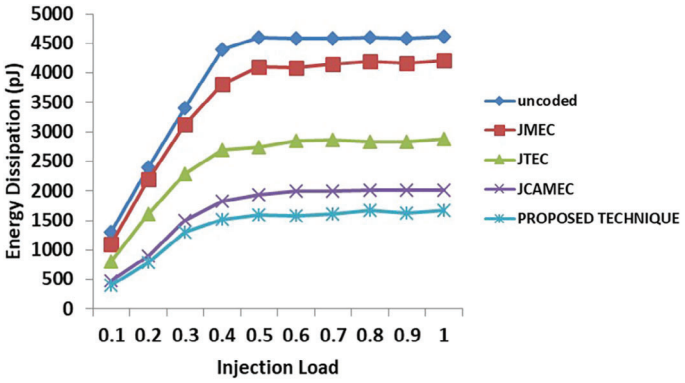


FIGURE 14 Energy dissipation evaluation of the proposed codec compared with different code

6 | CONCLUSION

With the shrinking size of the transistor, a massive increase in soft on-chip error rate has been observed. Different researchers have proposed different fault avoidance and fault mitigation techniques. These techniques having different pros and cons are applied according to the type of faults or errors in the network. This article has proposed an energy-efficient single, random, and burst error coding technique, which proved to be a better code as compared with JTEC-QED, JCAMEC, and JMEC. The proposed code possesses more code rate with less bit overhead compared with its competitors. The proposed code proved to be a better choice as it has got the same detection and correction capability with less energy consumption, less delay, and a small area overhead. Future work would include the analysis of the proposed code for different encoder-decoder placement methodologies in NoC architecture. Furthermore, by dividing the flit into six or eight groups, more analytical results could be obtained in terms of area, latency, and energy.

ACKNOWLEDGEMENTS

This research was supported in part by the Brain Korea 21 Plus Program (No. 22A20130012814) funded by the National Research Foundation of Korea (NRF), in part by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2019-2016-0-00313) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation), and in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1A09082266) and in part by University of Engineering and Technology, Taxila, Pakistan.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

ORCID

Yousaf Bin Zikria https://orcid.org/0000-0002-6570-5306

REFERENCES

1. Musaddiq A, Zikria YB, Hahm O, Yu H, Bashir AK, Kim SW. A survey on resource management in IoT operating systems. *IEEE Access*. 2018;6:8459-8482.
2. Dar Z, Ahmad A, Khan FA, et al. A context-aware encryption protocol suite for edge computing-based IoT devices. *J Supercomput*. 2019;1-20. https://doi.org/10.1007/s11227-019-03021-2.
3. Siddiqui IF, Qureshi NMF, Shaikh MA, et al. Stuck-at fault analytics of IoT devices using knowledge-based data processing strategy in smart Grid. *Wirel Pers Commun*. 2019;106(4):1969-1983.
4. Felfernig A, Erdeniz SP, Jeran M, et al. Recommendation technologies for IoT edge devices. *Proc Comput Sci*. 2017;110:504-509.
5. Tariq UU, Ali H, Liu L, Panneerselvam J, Zhai X. Energy-efficient static task scheduling on VFI based NoC-HMPSoCs for intelligent edge devices in cyber-physical systems. *ACM Trans Intell Syst Technol*. 2019;10(6):1-22.
6. Benini L, De Micheli G. Networks on chips: a new SoC paradigm. *Computer*. 2002;35(1):70-78.
7. Chen X, Lu Z, Lei Y, Wang Y, Chen S. Multi-bit transient fault control for NoC links using 2D fault coding method. Paper presented at: Proceedings of the 2016 10th IEEE/ACM International Symposium on Networks-on-Chip (NOCS); 2016:1-8; IEEE.
8. Moore GE. Cramming more components onto integrated circuits. *Proc IEEE*. 1998;86(1):82-85.
9. Wang L, Xiong Z, Shi G, Wu F, Zeng W. Adaptive nonlocal sparse representation for dual-camera compressive hyperspectral imaging. *IEEE Trans Pattern Anal Mach Intell*. 2016;39(10):2104-2111.
10. Shanbhag N, Soumyanath K, Martin S. Reliable low-power design in the presence of deep submicron noise (embedded tutorial session). Paper presented at: Proceedings of the 2000 International Symposium on Low Power Electronics and Design; 2000:295-302; ACM.
11. Zhang M, Shanbhag NR. Soft-error-rate-analysis (SERA) methodology. *IEEE Trans Comput-Aided Des Integr Circuits Syst*. 2006;25(10):2140-2155.
12. Mohanram K, Touba NA. Cost-effective approach for reducing soft error failure rate in logic circuits. *ITC*. 2003;1:893-901.
13. Pande PP, Zhu H, Ganguly A, Grecu C. Energy reduction through crosstalk avoidance coding in NoC paradigm. Paper presented at: Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD'06); 2006:689-695; IEEE.
14. Gordon M, Goldhagen P, Rodbell K, et al. Measurement of the flux and energy spectrum of cosmic-ray induced neutrons on the ground. *IEEE Trans Nucl Sci*. 2004;51(6):3427-3434.
15. Agarwal K, Sylvester D, Blaauw D. Modeling and analysis of crosstalk noise in coupled RLC interconnects. *IEEE Trans Comput-Aided Des Integr Circuits Syst*. 2006;25(5):892-901.
16. Poluri P, Louri A. A soft error tolerant network-on-chip router pipeline for multi-core systems. *IEEE Comput Archit Lett*. 2014;14(2):107-110.
17. Xie L, Mei K, Li Y. Repair: a reliable partial-redundancy-based router in noc. Paper presented at: Proceedings of the 2013 IEEE 8th International Conference on Networking, Architecture and Storage; 2013:173-177; IEEE.
18. Neale A, Sachdev M. A new SEC-DED error correction code subclass for adjacent MBU tolerance in embedded memory. *IEEE Trans Device Mater Reliab*. 2012;13(1):223-230.
19. Elahi H, Butt Z, Eugnei M, Gaudenzi P, Israr A. Effects of variable resistance on smart structures of cubic reconnaissance satellites in various thermal and frequency shocking conditions. *J Mech Sci Technol*. 2017;31(9):4151-4157.
20. Sridhara SR, Shanbhag NR. Coding for system-on-chip networks: a unified framework. *IEEE Trans Very Large Scale Integr (VLSI) Syst*. 2005;13(6):655-667.
21. Gul M, Chouikha M, Wade M. Joint crosstalk aware burst error fault tolerance mechanism for reliable on-chip communication. In: *IEEE Transactions on Emerging Topics in Computing*. 2017. https://doi.org/10.1109/TETC.2017.2787549.
22. Teja TS, Kiran TS, Narayana TS, Vinodhini M, Murty N. Joint crosstalk avoidance with multiple bit error correction coding technique for NoC interconnect. Paper presented at: Proceedings of the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI); 2018:726-731; IEEE.
23. Ganguly A, Pande PP, Belzer B. Crosstalk-aware channel coding schemes for energy efficient and reliable NOC interconnects. *IEEE Trans Very Large Scale Integr (VLSI) Syst*. 2009;17(11):1626-1639.

24. Fu B, Ampadu P. *Error Control for Network-on-Chip Links*. Berlin, Germany: Springer Science & Business Media; 2011.
25. Kumar U, Umashankar B. Improved hamming code for error detection and correction. Paper presented at: Proceedings of the 2nd International Symposium on Wireless Pervasive Computing IEEE; 2007.
26. Pande PP, Ganguly A, Feero B, Belzer B, Grecu C. Design of low power & reliable networks on chip through joint crosstalk avoidance and forward error correction coding. Paper presented at: Proceedings of the 2006 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems IEEE; 2006:466-476.
27. Blake IF. Error control coding (S. Lin and DJ Costello; 2004)[book review]. *IEEE Trans Inf Theory*. 2005;51(4):1616-1617.
28. Morelos-Zaragoza RH. *The Art of Error Correcting Coding*. Hoboken, NJ: John Wiley & Sons; 2006.
29. Flayyih WN, Samsudin K, Hashim SJ, Ismail YI, Rokhani FZ. Adaptive multibit crosstalk-aware error control coding scheme for on-chip communication. *IEEE Trans Circuits Syst II: Exp Briefs*. 2015;63(2):166-170.
30. Flayyih WN, Samsudin K, Hashim SJ, Rokhani FZ, Ismail YI. Crosstalk-aware multiple error detection scheme based on two-dimensional parities for energy efficient network on chip. *IEEE Trans Circuits Syst I: Regul Pap*. 2014;61(7):2034-2047.
31. Palesi M, Ascia G, Fazzino F, Catania V. Data encoding schemes in networks on chip. *IEEE Trans Comput-Aided Des Integr Circuits Syst*. 2011;30(5):774-786.
32. Yu Q, Ampadu P. Transient and permanent error co-management method for reliable networks-on-chip. Paper presented at: Proceedings of the 2010 4th ACM/IEEE International Symposium on Networks-on-Chip; 2010:145-154; IEEE.
33. Rambo EA, Tschiene A, Diemer J, Ahrendts L, Ernst R. Fmea-based analysis of a network-on-chip for mixed-critical systems. Paper presented at: Proceedings of the 2014 8th IEEE/ACM International Symposium on Networks-on-Chip (NoCS); 2014:33-40; IEEE.
34. Neale A. Design and analysis of an adjacent multi-bit error correcting code for nanoscale SRAMs. 2014.
35. Dutta A, Touba NA. Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code. Paper presented at: Proceedings of the 25th IEEE VLSI Test Symposium (VTS'07); 2007:349-354; IEEE.
36. Shamshiri S, Ghofrani A, Cheng KT. End-to-end error correction and online diagnosis for on-chip networks. Paper presented at: Proceedings of the 2011 IEEE International Test Conference; 2011:1-10; IEEE.
37. Park D, Nicopoulos C, Kim J, Vijaykrishnan N, Das CR. Exploring fault-tolerant network-on-chip architectures. Paper presented at: Proceedings of the International Conference on Dependable Systems and Networks (DSN'06); 2006:93-104; IEEE.

How to cite this article: Ibrahim M, Baloch NK, Anjum S, Zikria YB, Kim SW. An energy efficient and low overhead fault mitigation technique for internet of thing edge devices reliable on-chip communication. *Softw Pract Exper*. 2021;51:2393–2410. https://doi.org/10.1002/spe.2796