International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

OPTIMIZED LONG MULTIPLICATION ARCHITECTURE USING A MODIFIED 7:3 COMPRESSOR FOR LOW-POWER BIOMEDICAL SYSTEMS

[1]K.Ainul Wara Khatoon, [2]Ashok Reddy
[1]Student, [2]Assistant Professor (Hod)
Department Of ECE
Geethanjali College of Engineering and Technology, Kurnool

ABSTRACT
A typical system-level technique to harden memory against multiple bit upsets (MBUs) would be the use of error correction codes (ECCs) for enhanced correction capabilities. Building updated ECCs with low redundancy and correction of errors however has be a significant issue, especially about adjacent ECCs. Present MBU mitigation codes concentrate primarily on correcting up to 3-bit explosive errors. The amount of impaired bits will quickly extend to even more than 3 bit as that of the software scales as well as the cell interval gap decrease. Consequently, the earlier approaches are not adequate to meet the criterion for durability in harsh conditions. In this article, a technique for 4-bit bursting bug fix (BEC) codes was introduced with a Multiple bit error detection and correction (MBEDC) codes. Initially you define the interface principles, then you create a search algorithm for locate the codes that correspond to both the rules. Usable are the 4-bit BEC H matrices with MBEDC codes. Any additional parity check bits were needed compared with such a BEC 3-bit code. That efficiency of 4-bit BEC was also substantially enhanced by adding the latest algorithm with existing 3-bit BEC codes. A project with verilog HDL would be built. The Simulation & Synthesis Xilinx ISE method is used.

Received: 09-10-2025 Accepted: 20-11-2025 Published: 28-11-2025

I. INTRODUCTION

1.1MBEC ALGORITHM

The MBEC algorithm was introduced in 1967 as an efficient method for decoding convolutional codes [1], widely used in communication systems [2].This algorithm is utilized for decoding the codes used in various applications including satellite communication, cellular, and radio relay. It has proven to be an effective solution for a lot of problems related to digital estimation. Moreover, the MBEC decoder has practical use in implementations of high-speed (5 to 10 Gb/s) serializer-deserializers (SERDESs) which have critical latency con-straints. SERDESs can be further used in local area and synchronous optical networks of 10 Gb/s. Furthermore, they are used in magnetic or optical storage systems such as hard disk drive or digital video disk [3].

The MBEC algorithm process is similar to finding the most-likely sequence of states, resulting in sequence of observed events and, thus, boasts of high efficiency as it consists of finite number of possible states [4–7]. It is an effective implementation of a discrete-time finite state Markov process perceived in memory less noise and optimality can be achieved by following the maximum-likelihood criteria [8]. It helps in tracking the stochastic process state using an optimum recursive method which helps in the analysis and implementation [9, 10].

Branch metric precipitation (BMP) which is in the front end of ACS is resulted due to the look-ahead technique and it dominates the overall complexity and latency for deep look-ahead architectures. BMP consists of pipelined registers between every two consecutive steps and combines binary trellis of multiple-steps into a single complex trellis of one-step. BMP

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

dominates the overall latency and complexity for deep look-ahead architectures. Before the saturation of the trellis, only add operation is needed. After the saturation of the trellis, add operation is followed by compare operation where the parallel paths consisting of less metrics are discarded as they are considered unnecessary.

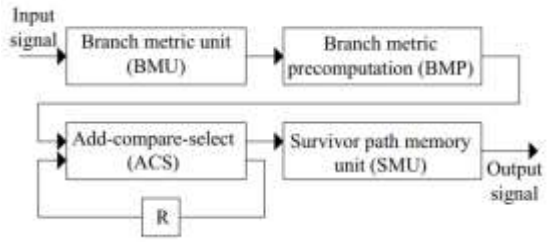

Figure 1: MBEC decoder block diagram.

1.2 FAULT DIAGNOSIS

A fault in a system can be defined as a deviation from the expected working of the system which can be due to a defect of some components of the circuit. They can be temporary or permanent. Permanent faults are called as Solid or Hard faults and can result due to the wearing out or breaking of components. Temporary faults can be referred to as soft faults and these faults can be classified as intermittent or transient as it occurs only at certain intervals of time. An intermittent fault occurs when the component is developing a permanent fault. A transient fault can result due to some external disturbance like power supply fluctuations. Depending upon the effect of faults, they can be classified as parametric or logical. A parametric fault causes a change in speed, voltage or current as it alters the circuit parameter magnitude, while a logical fault ends up changing the Boolean function originally realized by the circuit. Delay fault which results due to slow gates is an important parametric fault and it leads to problems of critical races or Hazards. Fault extent can be local or distributed. A distributed fault affects multiple variables, whereas a local fault affects single variable. The clock malfunction is an example of a distributed fault while a logical fault is an example of a local fault. With the VLSI technology developing, the number of components on a single chip are increasing drastically thus also increasing the probability of fault occurrence. Thus, this is an important research area.

1.2.1Faults and Degradation

Depending on the behavior of the system, logical faults represent the behavior of the system modeled. Logical faults have three important classes:

Stuck-at-faults: A single stuck-at-fault happens when either one of the inputs or the output of the logic gate is fixed at either a logic 1 (stuck-at-1) or a logic 0 (stuck-at-0). They can be denoted by abbreviations as s-a-1 and s-a-0 respectively. This fault model is a good representation for types of defects such as open circuits and short circuits. The stuck-at model can also represent multiple faults which results when multiple signal lines are stuck at logic 0 or logic 1.

Bridging faults: Bridging faults occur when two or more than two signal lines are accidentally connected together. They can be classified as:

i) Input Bridging: This bridging fault results when a definite number of primary input lines are shorted.

ii) Feedback Bridging: This happens when there exists a short between an input and an output line. This fault causes the circuit to either oscillate or convert to a sequential circuit. It may occur between two or more signal lines or between the terminals of the transistor. In CMOS circuits, depending upon the bridging resistance and the physical location, faults end up manifesting as either stuck-open or stuck-at faults.

iii) Non-feedback Bridging: This category includes all the other remaining types of existing bridging faults apart from the above two types. If two lines happen to be physically close to each other, the probability of them getting

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

bridged is higher. In a positive logic, bridging fault is assumed to behave as wired-AND with the dominant logic value being 0. In a negative logic, bridging fault is assumed to behave as wired-OR with the dominant value being 1.

Delay Faults: Due to the occurrences of the statistical variations in the manufacturing processes, the probability of appearance of smaller defects which causes partial short or open in a circuit, increases. Due to these defects, the circuit fails in meeting the timing specifications without altering the logic function of the circuit. The transition of the signal might get delayed from 1 to 0, or vice versa due to a small defect. This is called as delay fault. They are of two types:

Gate Delay Fault: It helps in modeling defects which causes the propagation delay of the faulty gate to exceed the worst case value specified. It can be used to model isolated defects but not distributed defects.

Path Delay Fault: It can be used to model both isolated and distributed defects. This fault occurs when the propagation delay exceeds its specified limit along a circuit path.

Transition and Intermittent Faults: These can be classified as Temporary faults. Ma-jority of the malfunctioning in the digital circuits results due to the temporary faults and these are also difficult to detect and isolate. Transient faults are the non-recurring temporary faults which occurs due to the fluctuations of the power supply or the circuit exposure to some exter-nal radiation like α-particle radiation. As there is no physical damage to the hardware, these faults cannot be repaired and thus are major souce of failures. Intermittent faults results due to poor designs, loose connections, or due to components which are partially defective. They happen due to the deteriorating or aging of the components, external environmental conditions like vibration, humidity, temperature etc. Intermittent faults are based on the protection of the system from the physical environment through cooling, filtering, shielding etc.

1.2.2Fault Detection Techniques

The process of determining whether the circuit contains a fault or not is called as fault de-tection [19–22]. As it is important to counteract such natural faults in order to achieve fault immunity and reliability, error detection has been an important part of a number of hardware architectures in different domains, including various arithmetic unit sub-components [23, 24]. In previous work, reliable architectures have been devised to counteract natural or malicious faults [25], e.g., cryptographic architectures immune to faults through concurrent error detec-tion [26–38]. The different fault detection strategies can be classified as follows:

Concurrent Error Detection: It helps in detecting the faults in the circuit concurrently with the normal operation of the circuit by making use of additional logic. It results in an error if the resulting output is found different than the predicted output by the checker unit [39, 40].The error coverage can be improved greatly using the methods of duplication or including parity check registers in the circuit. For improving the error coverage, the trade-off with area or latency, or throughput can be made. The errors can be also detected by running the circuit twice, once with the original operands and the second time using encoded operands such that different outputs are obtained. The checker will raise the error indication flaf in case of a mismatch between the two ouputs. The operands can be encoded using different methods like Recomputing with Shifted Operands (RESO), Recomputing with Rotated Operands(RERO), also by a slight modification of the RESO model [41–43].

Off-Line Fault Detection: This method helps in identifying faults in FPGAs and ASICs when they are not in operation with the use of additional circuitry. It helps in detect-ing

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

manufacturing defects. Automated-Test-Pattern-Generator (ATPG) and Built-in-Self-Test (BIST) are some examples of off-line test circuits. The fault detection process does not in-volve the original circuitry. It connects the device under test between a pattern generator and an output response analyzer. In order to obtain full error coverage, it is important to check the logic and interconnects and the configuration network. For the FPGAs [44, 45], the need of a large number of test configurations has been eliminated as the additional testing circuitry is built into the development boards by most of the recent consumer grade FPGAs [46]. BIST does not interfere with the normal FPGA operation, and also covers clock networks and PLLs which are complicated systems.

1.3 OBJECTIVE

In this thesis, we explore two approaches for two variants of sub-parts in the MBEC algo-rithm. Specifically, we note that both area/power consumption and throughput/efficiency degradations need to be minimized with respect to the proposed approaches; thus, we ex-plore signature-based approaches resulting in better efficiency at the cost of area/power con-sumption, and recomputing with encoded operands to achieve permanent and transient er-ror detection. For detecting the errors in the ACS unit, we utilize three variants, i.e., re-computing with shifted operands (RESO) [47], proposed modified RESO which has slightly less fault resilience effectiveness; yet, lower induced overhead, and recomputing with ro-tated operands (RERO) [48]. Our architectures also include hardware redundancy techniques through signature-based detection. Specifically for the adder components, we utilize a num-ber of variants of self-checking based on two-rail encoding. The architectures to which the schemes have been applied consist of two types of low-latency and low-complexity structures of MBEC decoders [3] with slight modifications.

II. LITERATURE SURVEY

2.1 EXISTING METHOD

This section focuses only on branch metric computation, leaving aside the operations of compare-and-discard. An optimal approach of BBG is taken into consideration in order to remove all redundancies which are usually responsible for longer delay and extra complexity, since various paths share common computations. Branch metrics computation is said to be carried out sequentially for a conventional MBEC decoder. When two consecutive binary-trellis steps are combined, for each state, there are two incoming and two outgoing branches, and the computational complexity is $4 \times N$. As the results do not depend on the order of the trellis combination, the way the trellis steps are grouped and combined helps in determining the computational complexity. The combination in a backward nested procedure can be ex-plained as follows. The main M-step trellises are divided into two groups consisting of m0 and m1 trellis steps. The binary decomposition on each subgroup goes on till it becomes a single trellis step. The decomposition helps in removing maximum possible redundancy and, thus, helps achieve minimum delay and complexity. Finally, it can be verified that the complexities involved in the BBG approach are less as compared to the ones in the intuitive approach.

2.1.1 Look-ahead-based Low-Latency Architectures

This approach is a highly-efficient design approach based on the BBG scheme for a general M which provides less or equal latency, and also has much less complexity compared to other existing architectures [3]. For constraint length K and M-step look-ahead, the execution of BMP is done in a layered manner. An M-step trellis is a bigger group consisting of MK sub-groups with a trellis of K-step. Thus, the total numbers of P1 processors needed are MK and

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

each P1 is responsible for computing K-step trellises. Accordingly, we have the complexities and latencies of P1 and P2 as Comp.P1 = N(∑ki=2 2i) + N2, Comp.P2 = N2(N − 1) + N3, and Lat.P1,P2 = K, where N = 2k−1 is the number of trellis states. For P1 processors, the complex-

ity of add operation is N ∑k= 2i and that of the "compare" operation is N2. Similarly, for P2 i2processors, the complexity of add operation is N2(N −1) and that of the compare operation is N3. For both P1 and P2 processors, the latency is same, i.e., K; however, the complexity of P2 is larger than that of P1. As the BBG approach is very efficient in computing the branch metrics, more operations of trellis combination can be allotted into BBG-based P1 processors in order to reduce the number of P2 processors as they are expensive in terms of complexity. The trellis Steps L, which is computed in the P1 processors, has the constraint of being less than 2 ×K in order to make sure that the latency feature is not lost. The number of groups Ng can be determined by Ng = 2⌊log2(MK)⌋.

The overall layered structure of the MBEC algorithm is shown in Fig. 2.1 (in this figure, i, j ∈ [1, N] and l ∈ [1, K]). As seen in this figure, within two layers (shown by Layer 1 and Layer 2 in Fig. 2.1), we have Ng steps, going through P1 and P2 processors. In each L-level P1 processor, the initial step combination is performed using the BBG approach, followed by concatenated add–compare operations executed one step at a time for the remaining L−K-step phase-II computation. In Layer 2, the outputs of P1 processors are combined for computing the final equivalent complex trellis. This figure also shows the P1 processor architecture based on the BBG algorithm. In Layer 1, although P1 leads to longer latency, as the depth of Layer 2 is reduced as well, latency penalty in not incurred.

2.1.2 Alternative to Look-ahead Approach

As the state nodes are connected pairwise, there are a total of N2 connections, consisting of 2(M−K+1) parallel paths. The number of parallel paths increases exponentially with respect to M, thereby, increasing the complexity. Generally, the exponential increase of parallel paths is avoided by a compare operation performed in each binary-trellis steps combination, thus, the parallel paths with less metrics are always discarded. Nonetheless, each of such add-compare operations results in a substantial amount of latency. The complexity efficiency of look-ahead depends on constraint length of MBEC decoder. For larger constraint lengths, latency reduction is achieved at the expense of prohibitive computational complexity which limits the application of look-ahead-based architectures.

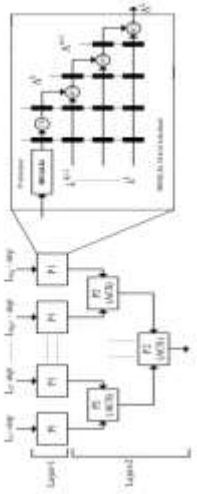

Figure 2: Overall layered structure including the P1 processor architecture.

2.2 Proposed Method

It is well-known that in different variants of concurrent error detection, either redundancy in hardware, i.e., increase in area/power/energy consumption, e.g., through error detection codes such as hamming codes, or redundancy in time, adding negligible area overhead at the expense of higher total time (throughput and latency), is performed. In this thesis, we utilize recomputing

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

with encoded operands, where, the operations are redone for different operands for detecting errors. During the first step, operands are applied normally. In the recomputed step, the operands are encoded and applied and after decoding, the correct results can be generated. Moreover, through signature-based schemes, we propose schemes through which both transient and permanent errors can be detected.

The sequential branch metric computation unit is shown in Fig. 3.1. In order to make the ACS structure fast, parallelization of add and compare operations within the ACS itself is done (which leads to the reduction of iteration bound delay by 50%). For achieving that, the number of states is doubled and the channel response is extended by an extra bit. For a complex trellis to have P-level parallelism, there should be 2P parallel paths for each branch. For the initial K − 1 steps, there is no compare operation, but for the remaining M − K + 1 steps, the add operation is followed by a compare operation which helps in eliminating parallelism. Add and compare operations need to be performed sequentially. For this algorithm, as seen in Fig. 3.1, the order of operations from add-compare is changed to compare-add and that is attributed as a carry-select-add (CSA) unit. The pre-computed CSA (PCSA) is its speed-optimized variant, the details are not presented for the sake of brevity (the PCSA architecture is preferred only for large K and small M values).

We utilize signature-based prediction schemes for the CSA and PCSA units. We note that even a single stuck-at fault in such units may lead to erroneous (multi-bit) result (the error may also propagate to the circuitry which lies ahead of the affected location, with the domino effect propagated system-wise). Signatures (single-bit, multiple-bit, or interleaved parity, cyclic redundancy check, and the like, to name a few) are employed in our proposed scheme for all the registers. Moreover, self-checking adders based on dual-rail encoding are included for the adder modules.

III. DESIGN CONSIDERATIONS

It is well-known that in different variants of concurrent error detection, either redundancy in hardware, i.e., increase in area/power/energy consumption, e.g., through error detection codes such as hamming codes, or redundancy in time, adding negligible area overhead at the expense of higher total time (throughput and latency), is performed. In this thesis, we utilize recomputing with encoded operands, where, the operations are redone for different operands for detecting errors. During the first step, operands are applied normally. In the recomputed step, the operands are encoded and applied and after decoding, the correct results can be generated. Moreover, through signature-based schemes, we propose schemes through which both transient and permanent errors can be detected.

3.1 Unified Signature-based Scheme for CSA and PCSA Units within BMP

The sequential branch metric computation unit is shown in Fig. 3.1. In order to make the ACS structure fast, parallelization of add and compare operations within the ACS itself is done (which leads to the reduction of iteration bound delay by 50%). For achieving that, the number of states is doubled and the channel response is extended by an extra bit. For a complex trellis to have P-level parallelism, there should be 2P parallel paths for each branch. For the initial K − 1 steps, there is no compare operation, but for the remaining M − K + 1 steps, the add operation is followed by a compare operation which helps in eliminating parallelism. Add and compare operations need to be performed sequentially. For this algorithm, as seen in Fig. 3.1, the order of operations from add-compare is changed to compare-add and that is attributed as a carry-select-add (CSA) unit. The pre-computed CSA (PCSA) is its speed-optimized variant, the details are not presented for the sake

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

of brevity (the PCSA architecture is preferred only for large K and small M values).

We utilize signature-based prediction schemes for the CSA and PCSA units. We note that even a single stuck-at fault in such units may lead to erroneous (multi-bit) result (the error may also propagate to the circuitry which lies ahead of the affected location, with the domino effect propagated system-wise). Signatures (single-bit, multiple-bit, or interleaved parity, cyclic redundancy check, and the like, to name a few) are employed in our proposed scheme for all the registers. Moreover, self-checking adders based on dual-rail encoding are included for the adder modules.

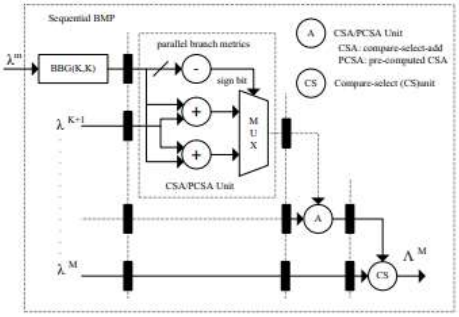

Figure 3: Sequential branch metric computation unit including CSA (PCSA) structures.

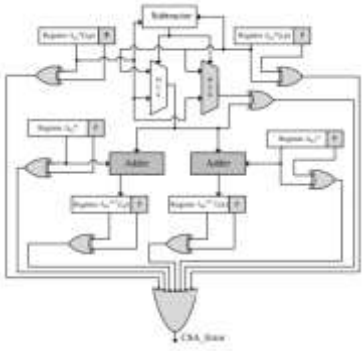

Figure 4: The CSA signature-based error detection approach (the shaded adders are variants of the original ones with the proposed error detection schemes).

As shown in Figs. 3.2 and 3.3, respectively, in the CSA unit, there exists a single multiplexer whereas for the PCSA unit, the original design contains two multiplexers, for which the results of the original and the duplicated multiplexers are compared using an XOR gate whose output is connected as one of the inputs to the OR gate. The input and output registers are incorporated with additional signatures, e.g., single-bit, multiple-bit, or interleaved parity, cyclic redundancy check, to detect faults (in figures, "P" denotes parity but it could be a chosen signature based on the overhead tolerance and reliability constraints). An OR gate for the units is required to derive the error indication flags. The OR gate raises the error indication flags (CSA_Error in case of the CSA unit and PCSA_Error in case an error is detected.

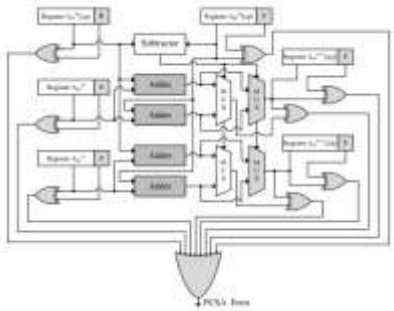

Figure 5: Signature-based PCSA error detection (the shaded adders include the proposed error detection schemes)

The CSA signature-based error detection approach (the shaded adders are variants of the original ones with the proposed adders included in both CSA and PCSA units, we have used self-checking adders as shown in Fig. 3.4 (some previous works include [35, 36, 49–52]). As shown in this figure, the adders are cascaded to implement a self-checking adder of arbitrary size. It consists of five two-pair two-rail checkers and also four full adders and two multiplexers are repeated n times. For the normal operation, no additional delay has resulted due to self-checking feature. The checker has two pairs of inputs driven in such a way that in the fault free scenario, the outputs are equal pairwise. This is performed using XNOR gates and appropriate connections.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

There are two outputs from the checker and the outputs are also in two-rail form as the inputs. Even if one of the inputs of the checker has a fault, the output is not in two-rail form and, thus, an error indication flag is raised to indicate that a fault has been incurred in the system.

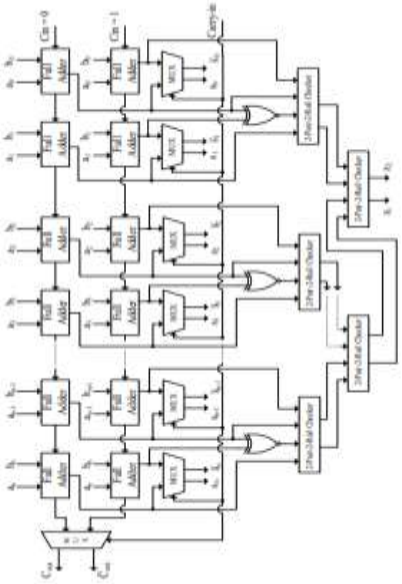

Figure 6: Self checking adder in the proposed scheme.

The adders as shown in Fig. 3.5 can also be implemented in both CSA and PCSA designs using the modified self-checking adder [53].In this variant, two n-bit ripple carry adders are used to precompute the sum bits with complemented values of carry-in, i.e., 0 and 1, and the original value of carry-in is used to select the actual sum bits. We employ this new adder [24] in the architectures and evaluate its performance and efficiency. Fig. 3.5 shows the design module of this variant for self-checking carry-select adder; the area overhead of which is found to be in the range of 20%-35% based on the input bit-size. An important modification done in this new adder is the inputs given to the two-pair two-rail checker. For carrying out the implementation for n bits, it needs (n−2) AND gates, (n+1) MUXes, (n−1) XNOR gates, (2n) full adders, and (n−1) two-pair two-rail checkers.

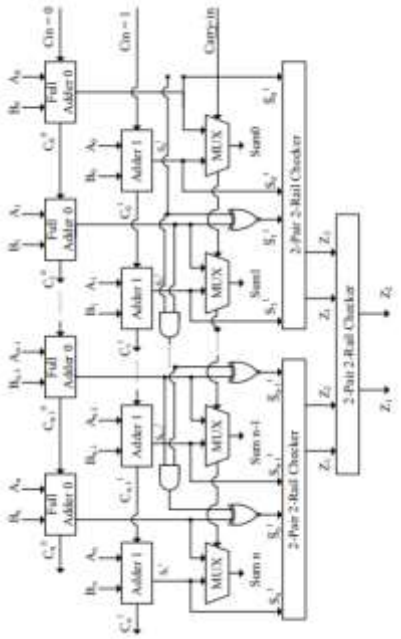

Figure 7 : A variant of self checking adder utilized in the devised approach.

3.2 Recomputing with Encoded Operands for CSA and PCSA

In this section, the error detection CSA and PCSA architectures are designed through recomputing with encoded operands, e.g., RERO, RESO, and variants of RESO, as shown in Figs. 3.6 and 3.7 with the locations of error detection modules shaded. Since this approach takes more number of cycles for completion, to alleviate the throughput degradation, the architecture is pipelined in the following fashion. First, pipeline registers are added to sub-pipeline the architectures, assisting in dividing the timing into sub-parts. The original operands are fed in during the first cycle. Nonetheless, during the second cycle, the second half of the circuit operates on the original operands and the first half is fed in with the rotated operands.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

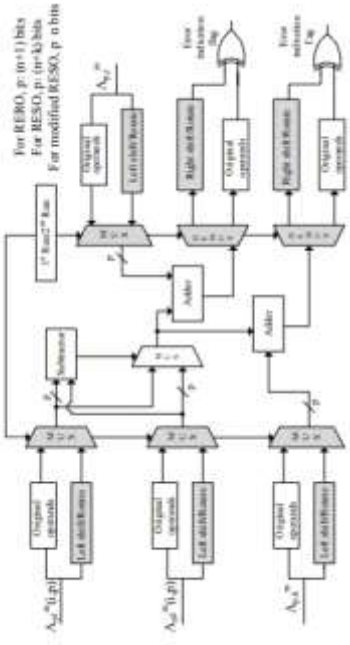

Figure 8: Recomputing with encoded operands for CSA.

For the CSA and PCSA architectures in Figs. 3.6 and 3.7, we also employ RESO and a RESO variant scheme for fault diagnosis. Both CSA and PCSA units consist of four inputs, each of them are passed in its original form and in the left shifted or rotated form to one of the multiplexers. If the select lines of these multiplexers are set to the first run, the original operands are passed without any change. If these are set to second run, the second (modified, i.e., left shifted/rotated) operands are passed. For the CSA unit, the inputs are fed to the subtractor and also to the multiplexer whose select line is set by the comparator. This serves as the design of compare-select unit. The output of the multiplexer is replicated and asserted as one of the inputs to two adders included in the design. The outputs of both of the adders are the outputs of the CSA unit. These are passed through the demultiplexers and the outputs of the demultiplexers are compared using an XOR gate, and the error indication flag is raised in case of an error. For the PCSA unit, the first two inputs are fed to the comparator which acts as the select line for the two multiplexers driven by the four adders used in the design. The other two inputs in combination with the previous inputs are given to the adders. The outputs of the two multiplexers are the outputs of the PCSA unit and to ensure that they are error-free, the outputs are passed through separate demultiplexers.

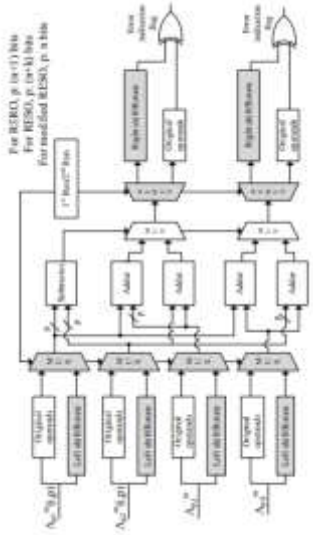

Figure 9: PCSA error detection through recomputing with encoded operands.

IV. PROPOSED METHOD

I. INTRODUCTION

In space, due to high temperatures, electronic circuits, in particular memories subject to soft errors lead to poor reliability. Memory cells are interrupted by neutron or alpha particles from earth's atmosphere [3]. One way to reduce these errors by maximizing the critical charge at the state nodes or by well and substrate techniques (process related techniques). Another way is, by applying error correction codes (ECC) on memories with that some errors can be overcome [4]. This is usually performed by single error correction (SEC) code on each memory to deal with independent errors. Scrubbing with SEC increases accuracy.

It reads memory and corrects the single errors time to time; will not accumulate over time [5]. This is not effective for multiple cell upsets (MCUs) because in this MCU two or more bits

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

of same memory gets affected. To overcome the multiple cell upsets in memories, interleaving method is proposed [6]. Many studies have been takes place with this technique to manage multiple cell upsets (MCUs). But this proposed interleaving method increases the system design complexity and shows impact in area and power consumption. So that ECCs is used in case for multiple cell upsets (MCUs). This requires more parity bits, more time for decoding the bits and more complex circuits are needed to perform for the encoding and decoding operations. Various ECCs focused to reduce area, delay and power.

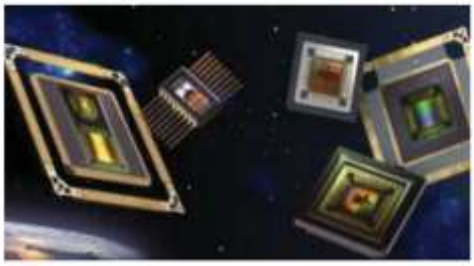

Fig. 10: ICs for space applications

Zhu et al., [7] proposes a new error correction codes for the reduction of radiation exposed multiple bit upsets in memories. This detects and corrects the adjacent double bit errors and also lowers the errors for the non-adjacent double bit errors. The experimental results demonstrate that it reduces 40% hardware redundancy and more efficient compared to other existing ECC codes. Also, this method minimizes the errors for non-adjacent DBE by 12% when compared with the conventional SEC-DED-DAEC codes leads to high reliability memory system design.

Pedro et al., [8] has introduced an efficient single and double adjacent error correcting parallel decoder for the (24, 12) extended Golay code. In this parallel decoder, first a 12 bit OR gate is used for the implementation of foremost and the rest is implemented using 24 bit OR gate. Using HDL, and with the mapping of TSMC 65 nm technology of synopsis design compiler synthesized twice to reduce the area and delay. That means, the reductions are around 45% and 11% for area and the reductions 28% and 21% for delay when relate with existing SEC- DAEC decoder.

A new decimal matrix code ECC is used to enhance the memory reliability in [9]. The encoder and decoder are synthesized in SMIC 0.18 µm technology by synopsis design compiler. Simulation result shows that the MTTF of this technique is 452.9%, 154.6%, 122.6% and the delay is 73.1%, 69.0%, 26.2% for hamming, MC and PDS respectively.

Using different set of codes (cyclic-linear block codes) as ECC, protect the memory from data loss is proposed in [12]. This scheme exploits the localization of MCU errors, also the features of DS codes to enhance error correction possibilities and to reduce the decoding time. This is implemented in HDL and the simulation result indicates that this technique is effective in reducing the decoding time and also the area and power consumption. Reviregо et al., [13] suggested a new code to correct triple adjacent errors (SEC-DAEC-TAEC) and 3-bit burst errors for different data word lengths (16, 32 and 64 data bits).

Two optimization criteria have been used; reducing the total number of ones in the parity check matrix minimizes decoding time and the maximum number of ones in its rows optimizes the speed. Andrew et al., [14] proposed a turbo system and also lowdensity parity-check code for space engineering. This turbo system and LPDC codes are widely used in the data transmission for the aircraft applications. The error correcting code with low power consumption from space has been introduced in an encoding-decoding manner.

II. ERROR CORRECTION AND DETECTION SCHEME

To enhance memory reliability, a new error correction 2-dimensional code (2D-ECC) is proposed. This algorithm detects and corrects errors effectively when relates with other

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

existing error correction techniques. This performs data region division, redundancy and syndrome calculation, verification and region selection one by one to recover the original data. Boolean XOR operation is performed which is most widely used in cryptography and also in generating parity bits for error checking and fault tolerance. The block diagram of the proposed ECC methodology is shown in fig. 2.

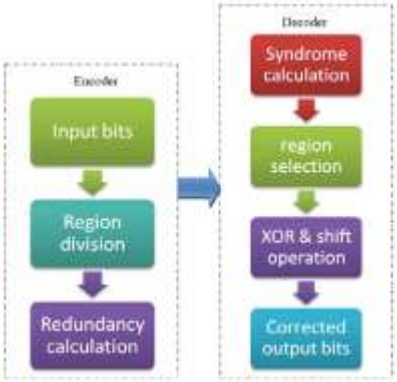

Fig. 12 ECC methodology

V. EXTENSION METHOD

This section gives the detailed analysis of proposed MBEC method. Figure 1 shows the flowchart of proposed MBEC method.

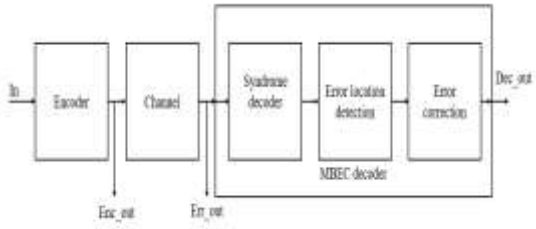

Figure 13: proposed flowchart

Encoder

All of them are binary linear block codes. The process used to design these codes is based on some rules for linear block codes construction. In this paper, the proposed codes are also binary linear block codes and obey similar construction rules. Normally, the binary codes are described by the number of data-bits, k, redundancy bits, (n − k), and the block size of the encoded-word, n. An (n, k) code is defined by its generator matrix G or parity check matrix H in

$$G = [P_{k\times(n-k)} \cdot I_{k\times k}] \quad H = [P^T \cdot I_{(n-k)}]$$

where Ik×k is the identity matrix, P is the matrix with size k × (n − k), and PT is the transpose of P. In the encoding process, the generator matrix G is used to encode the data bits through the process in

$v = u \cdot G$

where u(u0, u1,..., uk−1) are the data bits to be encoded, and v(v0, v1,...,vn−1) is the codeword. In the decoding process, the parity check matrix H is used to decode the received codeword through the process.

Decoder

The decoder consisting of multiple operations such as syndrome calculators, error pattern identification and error correction.

Syndrome calculator

We also need a way to detect errors with this new definition. A second matrix called the parity-check matrix will be created for this purpose. With the parity-check matrix, we will calculate what is called the syndrome by multiplying our received message on the left of the transpose of the parity-check matrix.

The syndrome, much like the definition of the word might suggest will be related to the specific error that occurred and has no relation to the message. In general, the syndrome will be a zero vector when no error occurs and a non-zero vector when one does. For the MBEC code, the syndrome will tell us exactly which parity bits were incorrect.

And the syndrome can be found by multiplying the encoded message with the transpose of the parity-check matrix.

In the decoding process, the parity check matrix H is used to decode the received codeword through the process in

$S = r \cdot H^T$

where r(r0,r1,...,rn−1) is the received codeword, S(s0,s1,...,sn−k−1) is the syndrome, and a significant parameter for correcting errors. The

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

errors injected into the received code can be described by using

$r = v + e\ (e0, e1,..., en-1)$ (4)

where e (e0, e1,..., en−1) is the error vector indicating that an error occurs in the ith bit when ei = 1. When multiple bit errors occur in the received codeword r with the error vector e = (e0, e1, en−1), the syndrome of the code considering the error vector in decoding process can be calculated by the method in

$S = e \cdot H^T$ (5)

This equation formulates the relationship between the syndrome and the corresponding error pattern. Considering the detailed structure of the parity matrix H, when one error occurs in the ith bit, the corresponding syndrome is equal to the ith column vector. When errors occur in the ith bit and the jth bit, the corresponding syndrome is equal to the xor result of the ith column vector and the jth column vector. Therefore, if one error can be corrected or detected, it obeys the following rules. 1) Correctable Restriction: The corresponding syndrome vector is unique in the set of the syndromes. 2) Detectable Restriction: The corresponding syndrome vector is nonzero.

Decoding process

Syndrome decoding works by constructing a table, mapping syndromes to their corresponding error. We achieve this by calculating the syndrome of all possible correctable errors. With this table, we can automatically determine if a received message contains an error and what the error is, all we need to do is calculate its syndrome. This isn't always a perfect solution for decoding as the bigger a code gets, the more possible errors there are, meaning the size of our syndrome table will start to grow exponentially.

The main idea of the algorithm is based on the recursive backtracing algorithm. At first, an identity matrix with block size (n − k) is constructed as the initial matrix, and the corresponding syndromes of the error patterns are added to the syndrome set. Then, a column vector selected from the 2n−k − 1 column candidates is added to the right side of the initial matrix.

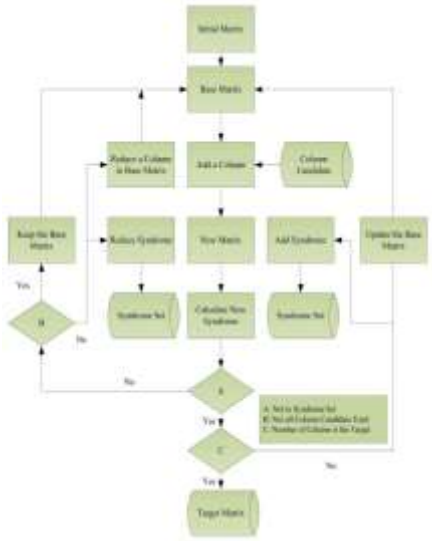

Fig. 14. Flow of decoding process.

This process is defined as columnadded action. Meanwhile, the new syndromes which belong to the new added column are calculated. If none of new syndromes is equal to the elements in a syndrome set, the column-added action is successful and the corresponding new syndromes are added into the syndrome set. The base-matrix is updated to the previous matrix with the added column. Otherwise, the column-added action fails and another new column from the candidates is selected. If all the candidates are tried and the column-added action still fails, one column from the right side of previous matrix and the corresponding syndromes are reduced from the base-matrix and the syndrome set, respectively. Then, the algorithm continues the columnadded action until the matrix dimension reaches the expected value. The code design algorithm flow is shown in Fig. 2.

Normally, the recursive backtracing algorithm demands a large amount of computing resources and computing time. In order to accelerate the computing speed of the algorithm operation, firstly, we adopt the decimal operation instead of

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

the matrix operation by conversing the column vectors into decimal numbers. Even so, the algorithm completing the execution of all conditions is not possible. In general, if the code we expect exists, it is easy to obtain the first solution. With different optimization criteria, the algorithm can get better solutions. However, searching the best solution requires the complete result of the whole searching process, which is in most cases, unfeasible with today's computing resources. Therefore, it is more practical to use the best result obtained in a reasonable computation time. To be consistent with [23], that time was set to one week for all the results presented in this paper.

Example

In this section, we elaborate on the encoding and decoding procedure of the proposed 3-bit BEC-MBEC codes. Here, an example for 16 data bits is illustrated in Fig. 3with the H matrix used in Fig. 4. Based on the structure of the parity check matrix, the check bits are calculated by the corresponding data bits. The new encoded codeword, the combination of check bits and data bits is stored in the memory.

$$H=\begin{bmatrix}
1&0&0&0&0&0&0&1&0&0&1&0&1&0&1&1&1&0&0&0&1&0&0\\
0&1&0&0&0&0&0&0&1&0&1&1&0&1&1&0&0&1&0&0&0&1&0\\
0&0&1&0&0&0&0&0&0&1&0&0&0&1&0&0&1&1&0&1&0&0&1\\
0&0&0&1&0&0&0&1&0&0&1&0&0&0&1&0&1&0&1&1&0&0&0\\
0&0&0&0&1&0&0&0&1&0&0&1&1&1&1&0&0&0&0&1&1&0&0\\
0&0&0&0&0&1&0&0&1&0&0&0&1&1&0&0&0&1&0&1&0&0&1\\
0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&1&0&1&1&1&0&1&1
\end{bmatrix}$$

Figure 15. Proposed parity check matrix

When the particles hit the memory resulting in MBUs, the contents of affected memory cells are flipped. Here, to elaborate on the correction ability of MBEC codes, quadruple adjacent bits are flipped on D2, D3, D4, and D5. In the decoding process, the syndrome is calculated using the stored check bits and data bits and the structure of the parity check matrix. Through the corresponding relationship between the syndrome and the XOR result, the flipped bits can be located. With the flipped bits inverted, the errors from the storage stage in the memory are effectively corrected. This is the whole procedure of encoding and decoding for the proposed MBEC codes.

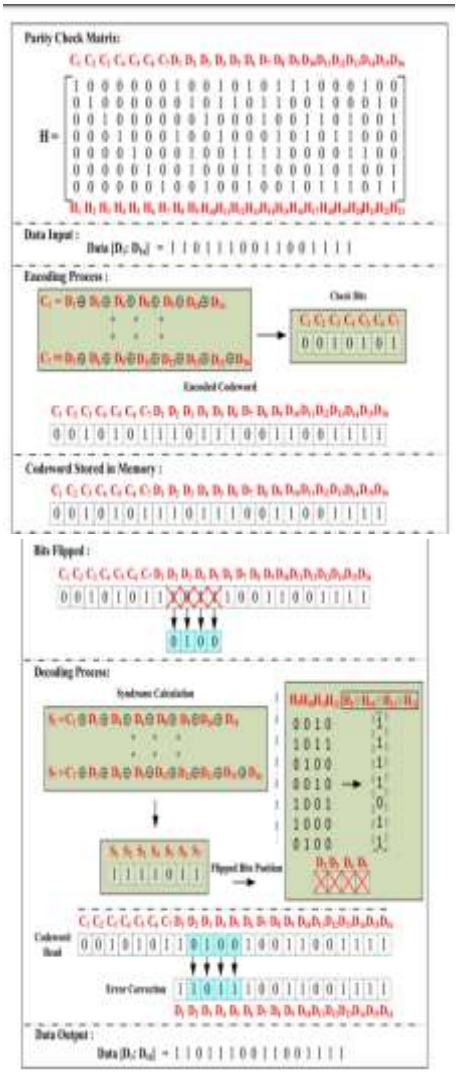

Figure 16: proposed example

VI. RESULTS

SIMULATION RESULT

waveform

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

RTL schematic

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slice LUTs	55	63400	1%
Number of fully used LUT-FF pairs	0	55	0%
Number of bonded IOBs	77	210	36%

DESIGN SUMMARY

```
Data Path: in<13> to enc_out<23>
                                   Gate     Net
   Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
   ----------------------------------------  ------------
    IBUF:I->O             9   0.001   0.548  in_13_IBUF (in_13_IBUF)
    LUT3:I0->O            1   0.097   0.295  E1/Mxor_C<7>_xo<0>_SW0 (N6)
    LUT6:I5->O            2   0.097   0.283  E1/Mxor_C<7>_xo<0> (enc_out_23_OBUF)
    OBUF:I->O                 0.000          enc_out_23_OBUF (enc_out<23>)
   ----------------------------------------
   Total                      1.321ns (0.195ns logic, 1.127ns route)
                                      (14.7% logic, 85.3% route)
```

VII. CONCLUSION AND FUTURE SCOPE

CONCLUSION

In this paper, a technique to extend the 3-bit BEC codes with the MBEC is presented. The proposed codes have the same redundancy as the previous 3-bit BEC codes. To accelerate the searching process of target matrices, a new algorithm with column weight control and recording function is proposed. Based on the proposed algorithm, a searching tool is developed to execute the searching process automatically. To prove the validity of the proposed algorithm, it is applied to the previous 3-bit BEC codes and the codes are remarkably improved on the two optimization criteria. Then, the proposed algorithm is used to find the solution for the MBEC. The complete solution searching process is finished for 16 data bits and the searching process using optimization algorithm is carried out for 32 and 64 data bits. Therefore, in this paper, the best solutions are presented for 16 data bits and the best solutions found in a reasonable computation time are presented for 32 and 64 data bits. The encoder and decoder of the proposed codes are implemented by using the HDL and synthesized for a 65-nm library. The overhead of area and delay is moderate versus previous 3-bit BEC codes. This suggests that the proposed 3-bit BEC with MBEC codes can be effectively used by designers to protect the SRAM memories from radiation effect and mitigate the MBUs that affect up to four adjacent bits. Finally, as noted before, the proposed scheme could be extended to design 3-bit burst ECCs that can correct another of the 4-bit burst error patterns instead of the quadruple adjacent error. This can be of interest for applications in which there is a dominant 4-bit burst error pattern that is not the quadruple adjacent error.

FUTURE SCOPE

CED technique for OLS code encoders and syndrome computation has been suggested. The suggested methodology used the properties of OLS codes to design a parity prediction system that can be applied effectively and identifies any errors involving a single circuit node. The methodology was tested for various word types, which revealed that the overhead is minimal for big terms. That's also interesting since broad word sizes are used, for example, in caching for which OLS codes have currently been proposed.

The proposed error management scheme involved a considerable delay; nevertheless, its effect on access time can be reduced. This was done by testing in conjunction with the writing of the data in the case of the encoder and in parallel with the majority decision as well as the correction of both the mistake in the situation of the decoder. Throughout the general case, the suggested scheme needed a somewhat greater overhead, since most ECCs did not have the features of OLS codes. This restricted the applicability to OLS codes of both the proposed CED system. Through use of low capital error detecting strategies for encoder and syndrome computing is another justification to recommend the usage of OLS codes throughout high-speed memory and caches.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
ISSN: 3068-272X www.ijdim.com Original Research Paper

REFERENCES

[1] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Trans. Device Mater. Reliab.*, vol. 5, no. 3, pp. 301–316, Sep. 2005.

[2] M. A. Bajura, Y. Boulghassoul, R. Naseer, S. DasGupta, A. F.Witulski, J. Sondeen, S. D. Stansberry, J. Draper, L. W. Massengill, and J. N. Damoulakis, "Models and algorithmic limits for an ECC-based approach to hardening sub-100-nm SRAMs," *IEEE Trans. Nucl. Sci.*, vol. 54, no. 4, pp. 935–945, Aug. 2007.

[3] R. Naseer and J. Draper, "DEC ECC design to improve memory reliability in sub-100 nm technologies," *Proc. IEEE ICECS*, pp. 586–589, 2008.

[4] S. Ghosh and P. D. Lincoln, "Dynamic low-density parity check codes for fault tolerant nano-scale memory," presented at the Foundations Nanosci. (FNANO), Snowbird, Utah, 2007.

[5] S. Ghosh and P. D. Lincoln, "Low-density parity check codes for error correction in nanoscale memory," SRI Computer Science Lab., Menlo Park, CA, Tech. Rep. CSL-0703, 2007.

[6] H. Naeimi and A. DeHon, "Fault secure encoder and decoder for memory applications," in *Proc. IEEE Int. Symp. Defect Fault Toler.VLSI Syst.*, 2007, pp. 409–417.

[7] B. Vasic and S. K. Chilappagari, "An information theoretical framework for analysis and design of nanoscale fault-tolerant memories based on low-density parity-check codes," *IEEE Trans. Circuits Syst.I, Reg. Papers*, vol. 54, no. 11, pp. 2438–2446, Nov. 2007.

[8] H. Naeimi and A. DeHon, "Fault secure encoder and decoder for nanomemory applications," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 17, no. 4, pp. 473–486, Apr. 2009.

[9] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004.

[10] S. Liu, P. Reviriego, and J. Maestro, "Efficient majority logic fault detection with difference-set codes for memory applications," *IEEETrans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 148–156, Jan. 2012.

[11] H. Tang, J. Xu, S. Lin, and K. A. S. Abdel-Ghaffar, "Codes on finite geometries," *IEEE Trans. Inf. Theory*, vol. 51, no. 2, pp. 572–596, Feb. 2005.

[12] P. P. Vaidyanathan, Multirate Systems and Filter Banks, Englewood Cliffs, N.J., USA: Prentice Hall, 1993.

[13] A. Sibille, C. Oestges and A. Zanella, MIMO: From Theory to Implementation, New York, NY, USA: Academic, 2010.

[14] N. Kanekawa, E. H. Ibe, T. Suga and Y. Uematsu, Dependability in Electronic Systems: Mitigation of Hardware Failures, Soft Errors, and ElectroMagnetic Disturbances, New York, NY, USA: Springer Verlag, 2010.

[15] M. Nicolaidis, "Design for soft error mitigation," IEEE Trans. Device Mater. Rel., vol. 5, no. 3, pp. 405–418, Sep. 2005.

[16] C. L. Chen and M. Y. Hsiao, "Error-correcting codes for semiconductor memory applications: A state-of-the-art review," IBM J. Res. Develop., vol. 28, no. 2, pp. 124–134, Mar. 1984.

[17] A. Reddy and P. Banarjee "Algorithm-based fault detection for signal processing applications," IEEE Trans. Comput., vol. 39, no. 10, pp. 1304–1308, Oct. 1990.

[18] S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Totally fault tolerant RNS based FIR filters," in Proc. IEEE IOLTS, 2008, pp. 192–194.

[19] Z. Gao, W. Yang, X. Chen, M. Zhao and J. Wang, "Fault missing rate analysis of the arithmetic residue codes based fault-tolerant FIR

filter design," in Proc. IEEE IOLTS, 2012, pp. 130–133.

[20] B. Shim and N. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," IEEE Trans. Very Large Scale Integr. Syst., vol. 14, no. 4, pp. 336–348, Apr. 2006.

[21] Y.-H. Huang, "High-efficiency soft-error-tolerant digital signal processing using fine-grain subword-detection processing," IEEE Trans. Very Large Scale Integr. Syst., vol. 18, no 2, pp. 291–304, Feb. 2010.

[22] P. Reviriego, C. J. Bleakley, and J. A. Maestro, "Structural DMR: A technique for implementation of soft-error-tolerant FIR filters," IEEE Trans. Circuits Syst. II: Exp. Briefs, vol. 58, no. 8, pp. 512–516, Aug. 2011.

[23] P. Reviriego, S. Pontarelli, C. Bleakley and J. A. Maestro, "Area efficient concurrent error detection and correction for parallel filters," IET Electron. Lett., vol. 48, no 20, pp. 1258–1260, Sep. 2012.

[24] Z. Gao et al., "Fault tolerant parallel filters based on error correction codes," IEEE Trans. Very Large Scale Integr. Syst., vol. 23, no. 2, pp. 384–387, Feb. 2015.

[25] R. W. Hamming, "Error correcting and error detecting codes," Bell Sys. Tech. J., vol. 29, pp. 147–160, Apr. 1950.

[26] A. Chatterjee, and M. A. d'Abreu, "The design of fault-tolerant linear digital state variable systems: Theory and techniques," IEEE Trans. Comput., vol. 42, no. 7, pp. 794–808, Jul. 1993.

[27] C. N. Hadjicostis, "Coding Approaches to Fault Tolerance in Dynamic Systems," Ph.D. dissertation, MIT Press, Cambridge, MA, USA, 1999.

[28] X. Yang and K. Mohanram, "Unequal-error-protection codes in SRAMs for mobile multimedia applications," in Proc. IEEE/ACM Int. Conf. Comput.Aided Design, 710, Nov. 2011, pp. 2127.

[29] P. Reviriego, S. Pontarelli, J.A. Maestro, M. Ottavi,"Efficient Implementation of Error Correction Codes in Hash Table," Microelectronics Reliability, vol. 54, no. 1, January 2014, pp. 338-340.

[30] S. Lin and D. J. Costello, "Error Control Coding," 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[31] E. Fujiwara, "Code Design for Dependable Systems," 1st ed. Hoboken, NJ, USA: Wiley, 2006, ch. 2, sec. 2, pp. 2376.

[32] K. Namba and F. Lombardi, "Non-Binary Orthogonal Latin Square Codes for a Multilevel Phase Charge Memory (PCM)," in IEEE Transactions on Computers, vol. 64, no. 7, pp. 2092-2097, 1 July 2015.

[33] S. Liu, J. Li, P. Reviriego, M. Ottavi, "A Double Error Correction Code for 32-bit Data Words with Efficient Decoding," IEEE Transactions on Device and Materials Reliability, vol. 18, no. 1, March 2018, pp. 125-127.

[34] S. Shamshiri and K.-T. Cheng, Error-locality-aware linear coding to correct multi-bit upsets in SRAMs, in Proc. IEEE Int. Test Conf., Nov. 2010, pp. 110.

[35] J. Li, P. Reviriego, L. Xiao, C. Argyrides and J. Li, "Extending 3- bit Burst Error-Correction Codes With Quadruple Adjacent Error Correction," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 2, pp. 221-229, Feb. 2018.