IMPLEMENTATION OF MULTI BIT ERROR DETECTION AND CORRECTION USING SYNDROME CHECKING

[1]Manideep Gadireddi, [2]Mrs V. Sravanthi
[1]M.Tech Student, [2]Assistant Professor
DEPARTMENT OF ECE
Siddhartha Institute of Technology & Sciences, Telengana

ABSTRACT
On behalf of technology scaling, on-chip memories in a die undergoes bit errors because of single events or multiple cell upsets by the ecological factors such as cosmic radiation, alpha, neutron particles or due to maximum temperature in space, leads to data corruption. Error detection and correction techniques (ECC) recognize and rectify the corrupted data over communication channel. In this paper, an advanced error correction 2-dimensional code based on divide-symbol is proposed to weaken radiation-induced MCUs in memory for space applications. For encoding data bits, diagonal bits, parity bits and check bits were analyzed by XOR operation. To recover the data, again XOR operation was performed between the encoded bits and the recalculated encoded bits. After analyzing, verification, selection and correction process takes place.

I. INTRODUCTION

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].

A top-level architecture for MBEC decoders is shown in Fig. 1.1. As seen in this figure, MBEC decoders are composed of three major components: branch metric unit (BMU), add-compare-select (ACS) unit, and survivor path memory unit (SMU). BMU generates the metrics corre-sponding to the binary trellis depending on the received signal, which is given as input to ACS which, then, updates the path metrics. The survival path is updated for all the states and is stored in the additional memory. SMU is responsible for managing the survival paths and giv-ing out the decoded data as output.

BMU and SMU units happen to be purely forward logic. ACS recursion consists of feedback loops; hence, its speed is limited by the iteration bound [11]. Hence, the ACS unit becomes the speed bottleneck for the system. M-step look-ahead technique can be used to break the iteration bound of the MBEC decoder of constraint length K [12–18]. A look-ahead technique can combine several trellis steps into one trellis step, and if M > K, then throughput can be increased by pipelining the ACS

architecture, which helps in solving the problem of iteration bound, and is frequently used in high-speed communication systems.

Branch metric precomputation (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 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.

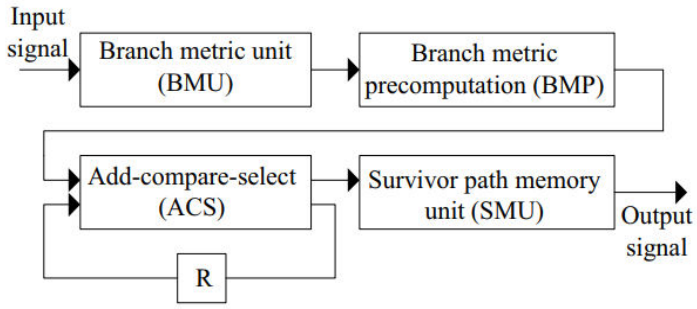


Figure 1.1: MBEC decoder block diagram.

Although MBEC algorithm architectures are used commonly in decoding convolutional codes, in the presence of very-large-scale integration (VLSI) defects, erroneous outputs can occur which degrade the accuracy in decoding of convolutional codes.

1.1. 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 wear-ing out or breaking of components. Temporary faults can be referred to as soft faults and these faults can be classified as intermiitent 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 af-fects 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. 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.

We summarize the contributions of this thesis as follows:

- We propose error detection methods for the modified MBEC decoder with the consider-ation of objectives in terms of performance metrics and reliability. The error detection approaches along with the modifications help achieving high error coverage and through the proposed improvements, performance boost can be achieved. Variants of recomput-ing with encoded operands on a number of architectures within the modified MBEC decoder as well as signature-based approaches (including modified self-checking based on two-rail encoding) are presented as well. The mechanisms for making the proposed structures immune to faults have not been presented before.
- We have extensively simulated the proposed error detection architectures and the ob-tained results help in benchmarking the error coverage. The results of our simulation show that the reliability of the proposed architecture can be ensured.
- Finally, our proposed error detection MBEC decoders incorporating the error detection approaches are implemented on application-specific integrated circuit (ASIC) [32nm li-brary] and field-programmable gate array (FPGA) [Xilinx Virtex-6 family]. The results indicate that the architectures can be reliably used. The proposed approaches can be utilized based on reliability objectives and performance/implementation metrics degra-dation tolerance.

II. 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 m_0 and m_1 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.

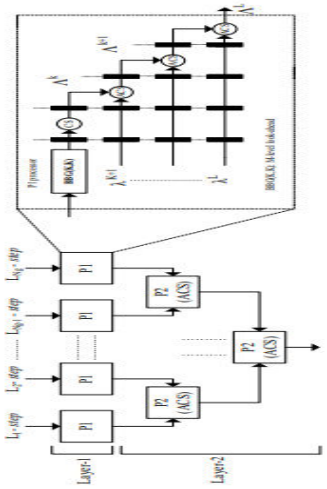


Figure 2.1: Overall layered structure including the P1 processor architecture.

III. 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 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.

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 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.

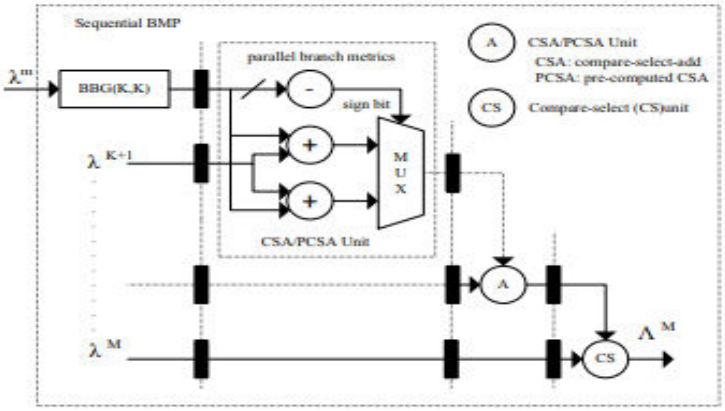


Figure 3.1: Sequential branch metric computation unit including CSA (PCSA) structures.

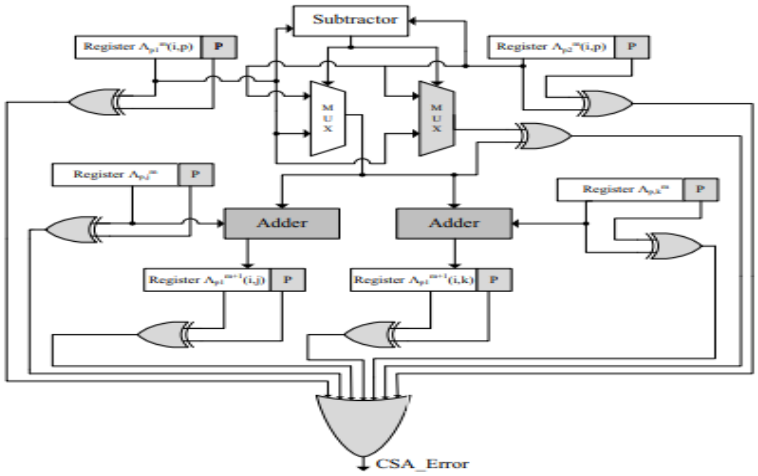


Figure 3.2: The CSA signature-based error detection approach (the shaded adders are variants of the original ones with the proposed error detection schemes).

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.

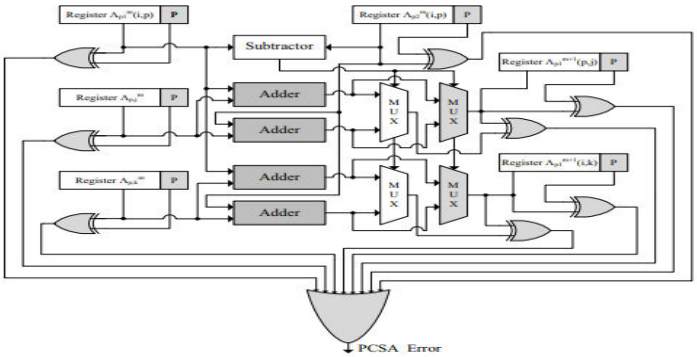


Figure 3.3: Signature-based PCSA error detection (the shaded adders include the proposed error detection schemes)

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.

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.

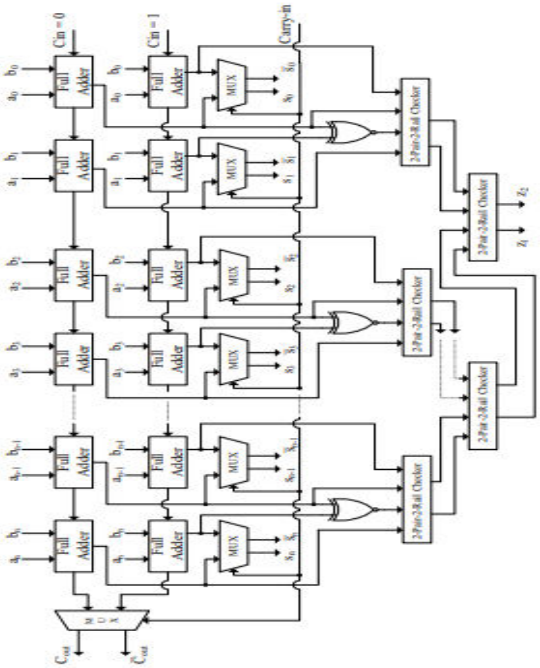


Figure 3.4: Self checking adder in the proposed scheme.

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.

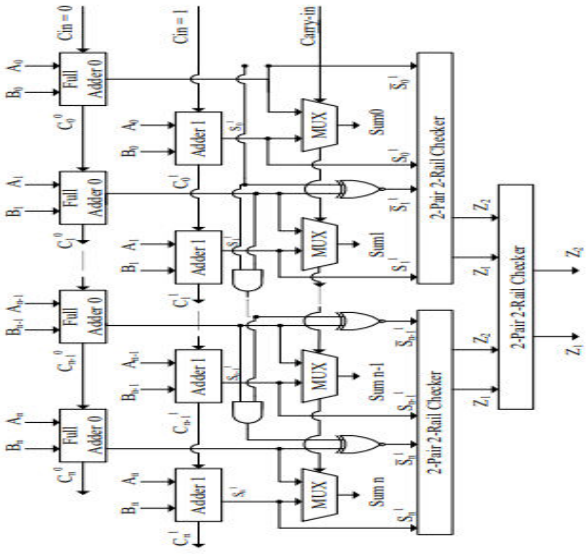


Figure 3.5: A variant of self checking adder utilized in the devised approach.

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.

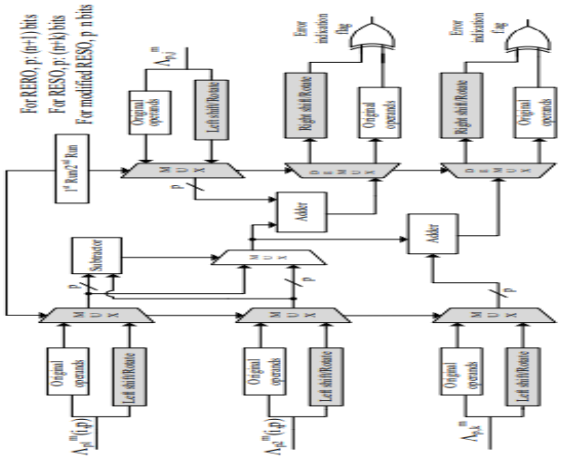


Figure 3.6: Recomputing with encoded operands for CSA.

IV. 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 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.

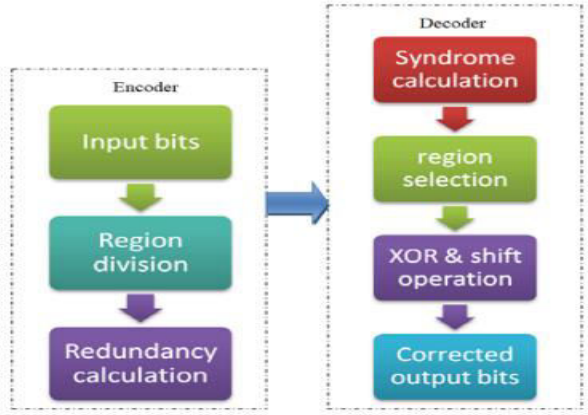


Fig. 4.1. ECC methodology

This 2-dimensional algorithm performs encoding-decoding process which codifies 16 bit input data into 32 bits in encoding and while decoding again the original 16 bit data is recovered.

Proposed Algorithm

STEP 1: Read the input 16 bit data (A16 – A0)

STEP 2: Divide the input data into 4 groups

X_1	Y_1	Z_1	W_1
X_2	Y_2	Z_2	W_2
X_3	Y_3	Z_3	W_3
X_4	Y_4	Z_4	W_4

STEP 3: Analyze diagonal bits, parity bits and check bits using XOR operation.

i) Diagonal bits (D1, D2, D3, D4) using XOR operation as the 2×2 matrix,

$$D_1 = X_1 \oplus Y_2 \oplus Z_1 \oplus W_2$$
$$D_2 = X_2 \oplus Y_1 \oplus Z_2 \oplus W_1$$

ii) Parity bits (P1, P2, P3, P4) using XOR operation taking the first bits, second bits, third bits and the fourth bits from four groups

$$P_1 = X_1 \oplus Y_1 \oplus Z_1 \oplus W_1$$
$$P_2 = X_2 \oplus Y_2 \oplus Z_2 \oplus W_2$$

iii) Check bits (Cx, Cy, Cz, Cw) using XOR operation by taking the alternative bits

$$Cx_{13} = X_1 \oplus X_3$$
$$Cx_{24} = X_2 \oplus X_4$$
$$Cy_{13} = Y_1 \oplus Y_3$$
$$Cy_{24} = Y_2 \oplus Y_4$$

STEP 4: Calculate the syndrome values for diagonal, parity and check bits by performing XOR operation between the redundancy data stored and the recalculated redundancy bits (RDi, RPi, RCi)

$$SD_i = D_i \oplus RD_i$$
$$SP_i = P_i \oplus RP_i$$
$$SC_i = C_i \oplus RC_i$$

where i = 1, 2, 3, 4

STEP 5: Check the following conditions to identify the error that to be satisfied

i) SDi and SPi bits have atleast one value similar to 1

ii) ii) More than one SCi value was similar to 1

STEP 6: Perform region selection and change the erroneous data to get the corrected output.

Process of encoding First, divide the 16 input bits into four groups (Xi, Yi, Zi, Wi). The diagonal bits (Di), parity bits (Pi) and check bits (Ci) are determined using XOR operation. In the process of encoding, the input 16 bits gets converted into 32 bits (redundancy bits).

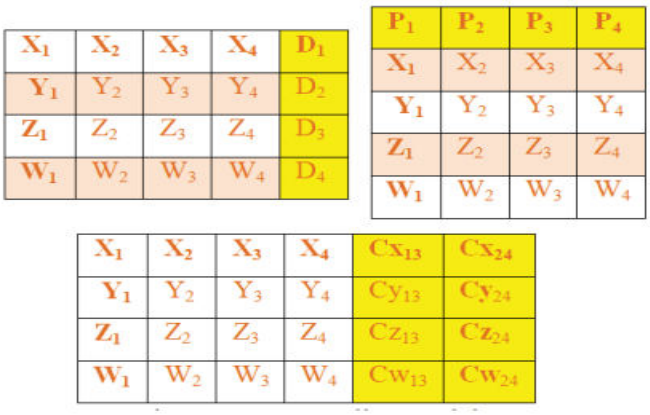


Fig. 4.2. Encoding model

Process of decoding

In decoding, the syndrome calculation has been analyzed with the encoded data and the recalculated encoded bits (SDi, SPi and SCi). After that, verification, region selection and correction can be performed.

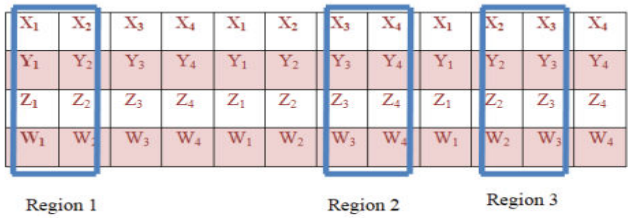


Fig.4.3.Different regions of data bits

Divide the data bits into regions 1, 2 and 3. This is formed by dividing the data bits by columns (1&2, 3&4, 2&3) to get efficient results.

Region	Selection criteria
Region 1	SD1+SD2+SP1+SP2 > SD3+SD4+SP3+SP4
Region 2	SD1+SD2+SP1+SP2 < SD3+SD4+SP3+SP4
Region 3	SD1+SD2+SP1+SP2 = SD3+SD4+SP3+SP4

Table 1 Region selection criteria

In table 1, the equations derived from SDi and SPi. The region 1 and region 2 with more syndrome bits equal to 1 has the error bits and the region 3 for which both equations equal. This algorithm could be most widely used in space engineering for the error correction and detection of bits during transmission of information.

V. EXTENSION METHOD

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

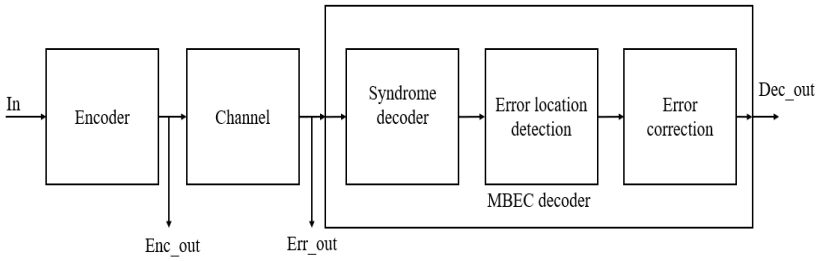


Figure 5.1: proposed flowchart

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 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.

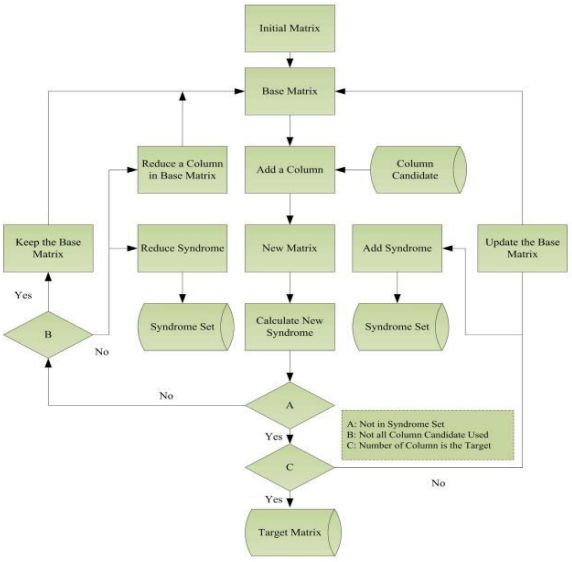
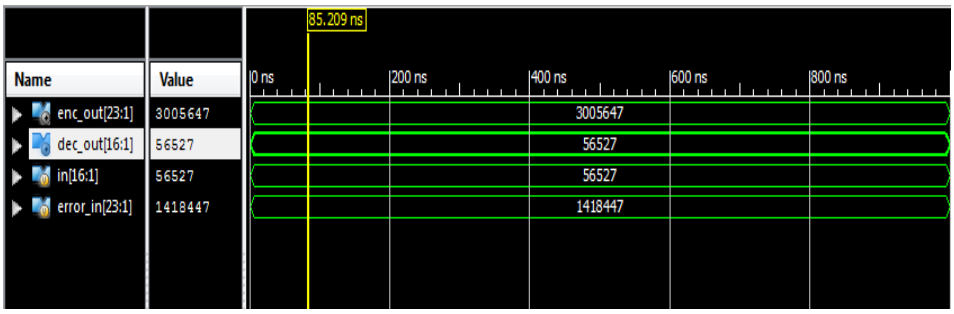


Fig. 2. Flow of decoding process.

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.

VI. RESULTS

6.1 Simulation Result



waveform

RTL schematic

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	55	63400	0%
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                      1.322ns (0.195ns logic, 1.127ns route)
                                     (14.7% logic, 85.3% route)
```

Time summary

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.

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.
[1] P. P. Vaidyanathan, Multirate Systems and Filter Banks, Englewood Cliffs, N.J., USA: Prentice Hall, 1993.
[2] A. Sibille, C. Oestges and A. Zanella, MIMO: From Theory to Implementation, New York, NY, USA: Academic, 2010.
[3] 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.
[4] M. Nicolaidis, "Design for soft error mitigation," IEEE Trans. Device Mater. Rel., vol. 5, no. 3, pp. 405–418, Sep. 2005.
[5] 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.
[6] 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.
[7] 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.
[8] 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.
[9] 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.
[10] 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.
[11] 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.