An Adaptive Low Latency Low Complexity Architecture for Matching of Information Coded with Error–Correcting Codes

Khadeejah Neelofer Roohi
PG Scholar
Department of ECE
Vaagdevi College of Engineering
Bollikunta, Warangal-506001

M.A.Himayath Shamshi
Asst. Professor
Department of ECE
Vaagdevi College of Engineering
Bollikunta, Warangal-506001

P Prasad rao
Professor & HoD
Department of ECE
Vaagdevi College of Engineering
Bollikunta, Warangal-506001

Abstract: A computing system is one, where an input data needs to be compared with a stored data to locate the matching entry, e.g., translation look-aside buffer and cache tag array lookup matching. In this paper we propose a new architecture to reduce complexity and latency for matching the data protected with an error-correcting code (ECC). It is based on the fact that the codeword of an ECC generated by encoding is usually represented in a systematic form, and it consists of the raw data and the parity information. The proposed architecture parallelizes the comparison of the data and that of the parity information. To further reduce the latency and complexity, in addition, a new butterfly-formed weight accumulator (BWA) is proposed for the efficient computation of the Hamming distance. The proposed architecture examines whether the incoming data matches the stored data if a certain number of erroneous bits are corrected.

1. Introduction

Data comparison circuit is a logic that has many applications in a computing system. For example, to check whether a piece of information is in a cache, the address of the information in the memory is compared to all cache tags in the same set that might contain that address. Error correction codes (ECC) are the one, most commonly used to protect standard memories and circuits [6], while more sophisticated codes are used in critical applications such as space [6]. ECC are widely used to enhance the reliability and data integrity of memory structures in modern microprocessors. For example, caches on modern microprocessors are protected by ECC [3]. If a memory structure is protected with ECC, a piece of data is encoded first and the entire codeword including the ECC check bits are written into the memory array. When the input data is loaded into the system, it has to be encoded and compared with the data stored in the memory and corrected if errors are detected to obtain the original data.

Data comparison circuit is usually in the critical path of a pipeline stage because the result of the comparison determines the flow of the succeeding operations. When the memory array is protected by ECC, it exacerbates the criticality because of the added latency due to ECC logic. In the cache tag matching example, the cache tag directory must be accessed first. After the tag information is retrieved, it must go through ECC decoding and correction before the comparison operation can be performed. At the mean time, the corresponding data array is waiting for the comparison result to decide which way in the set to load the data from.

The most recent solution for the matching problem is the direct compare method [5], which encodes the incoming data and then compares it with the retrieved data that has been encoded as well. Therefore, the method eliminates the complex decoding from the critical path. In performing the comparison, the method does not examine whether the retrieved data is exactly the same as the incoming data. Instead, it checks if the

retrieved data resides in the error correctable range of the codeword corresponding to the incoming data. As the checking necessitates an additional circuit to compute the Hamming distance, i.e., the number of different bits between the two code words, the saturate adder (SA) was presented [5] as a basic building block for calculating the Hamming distance.

However, it does not consider an important fact that a practical ECC codeword is usually represented in a systematic form in which the data and parity bits are completely separated from each other. In addition, SA contributes to the increase of the entire circuit complexity as it always forces its output not to be greater than the number of detectable errors by more. In brief, we renovate the SA-based direct compare architecture to reduce the latency and hardware complexity by resolving the drawbacks. More specifically, we consider the characteristics of systematic codes in designing the proposed architecture and propose a low-complexity processing element that computes the Hamming distance faster. Therefore, the latency and the hardware complexity are decreased considerably compared with the SA based architecture. The rest of this brief is organized as follows. Section II reviews previous works. The proposed architecture is explained in Section III, and evaluated in Section IV. Finally, concluding remarks are made in Section V.

2. Previous Works

This section describes the conventional decode-and-compare architecture based on the direct compare method. For the sake of concreteness, only the tag matching performed in a cache memory is discussed in this brief, but it is said that the proposed architecture can be applied to similar applications without loss of generality.

Direct Compare Method

The key idea behind direct compare scheme is to utilize the information carried by the incoming data (referred to as input) to circumvent the necessity of decoding and correction of the stored codeword which may or may not have errors. For input protected with ECC, in most scenarios, the corrupted codeword is the only copy of that contains the original information. Without redundancy provided by ECC there is no other way to retrieve it. However, for data comparison, the absolute values of the stored information are not that important, but rather the relative value to the incoming data is important for deriving a comparison result. Considering the number of input bits (N) to be 31, i.e., N=31 a circuit for direct compare method is proposed with full adders and saturate adders.
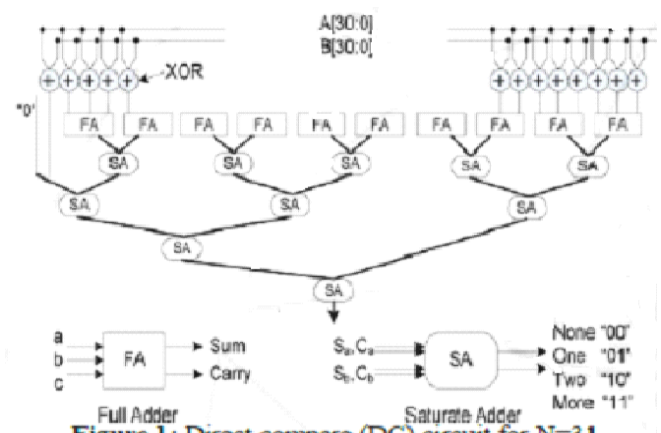

Figure 1: Direct compare (DC) circuit for N=31

Decode-and-Compare Architecture

Let us consider a cache memory where an n-bit codeword is stored after being encoded by a (n, k) code. In the decode-and-compare architecture depicted in Fig. 2, the n-bit retrieved codeword should first be decoded to extract the original k-bit tag. The extracted k-bit tag is then compared with the k-bit tag field of an incoming address to determine whether the tags are matched or not. As the retrieved codeword should go through the decoder before being compared with the incoming tag, the critical path is too long to be employed in a practical cache system designed for high-speed

ISSN No: 2348-4845

International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)

July 10, 2015 - Hyderabad, India

Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

access. Since the decoder is one of the most complicated processing elements, in addition, the complexity overhead is not negligible. Grounded on the fact of implementing the decoding architecture in hardware, it results in increase of hardware complexity, since the decoding technique includes large no of gates when implemented.
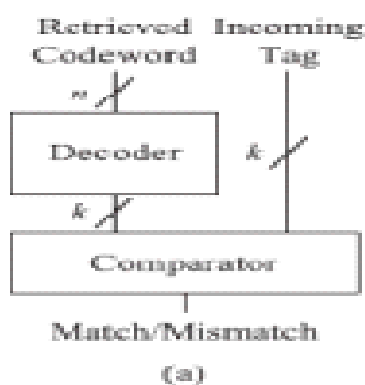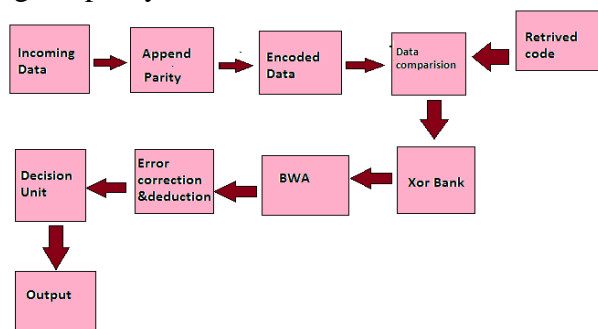

Figure 2: Decode-and-Compare architecture

3. Proposed Architecture

This section presents a new architecture that can reduce the latency and complexity of the data comparison by using the characteristics of systematic codes.

A. Block Diagram

The Fig.3 describes the flow of the proposed architecture. The incoming data is encoded by appending the parity bits.

Then the encoded data is compared with the data in the memory which can be retrieved. The XOR bank and Butterfly formed weighted accumulator

is used to find the number of bit changes and to calculate the number of ones which are fed into error correction and error deduction unit. Thus the output is obtained from the decision unit.

B. Data Path Design

In the SA-based architecture [5], the comparison of two codeword is performed after the incoming tag is encoded. Therefore, the critical path consists of a series of the encoding and the n-bit comparison. However, the fact that, in practice, the ECC codeword is of a systematic form in which the data and parity parts are completely separated is not taken into the account. As the data part of a systematic codeword is exactly the same as the incoming tag field, it is immediately available for comparison while the parity part becomes available only after the encoding is completed. Grounded on this fact, the comparison of the k-bit tags can be started before the remaining $(n-k)$-bit comparison of the parity bits. In the proposed architecture, therefore, the encoding process to generate the parity bits from the incoming tag is performed in parallel with the tag comparison, resulting in the reduction of the overall latency.

C. Construction of Low Delay Single-Error Correction Codes

The proposed method to construct SEC and SEC-DED codes tries to minimize the number of ones in each row and in each column of the parity check H matrix. Reducing the number of ones in the rows lowers the delay when computing the parity bits in the encoder. To minimize the number of ones the value $w = 2$ can be used to obtain SEC codes. It is also interesting to analyze the case $w = 3$ as in that case the code is SEC-DED. Since for the parity bits thecolumns have only a one, the condition is not met as other columns have a one in that bit. Therefore, this modification cannot correct errors in the parity bits. This is not an issue for registers as the
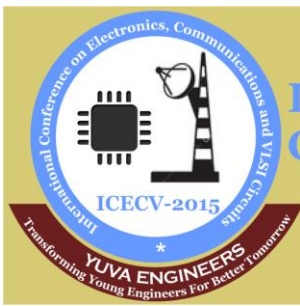
ISSN No: 2348-4845

International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)

July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

correction of parity bits is not normally needed. The method to construct the code starts by finding the smallest value of $n - k$ for which the following is true:

$$\binom{n-\kappa}{w} \geq k.$$

For $w = 2$, this value can be found analytically by solving (1) that is a quadratic equation in n. As the value of n has to be larger than k, only one of the two possible solutions of the equation is valid in our case. The value of $n - k$ obtained is

$$n - k \geq \left\lceil \frac{1 + \sqrt{8k+1}}{2} \right\rceil \qquad (2)$$

that shows a growth of the number of parity bits with square root of k that is larger than logarithmic growth of Hamming codes. This means that as k increases, the overhead of the proposed codes in terms of the number of additional parity bits compared to Hamming will also increase. Similarly, for $w = 3$, the solution to (1) is given by

$$n - k \geq \left(\frac{\sqrt{243k^2 - 1}}{3^{3/2}} + 3k \right)^{1/3} + \frac{1}{3\left(\frac{243k^2 - 1}{3^{3/2}} + 3 \right)}$$

that, as k is larger than one, it can be approximated by

$$n - k \geq ((6k)^{1/3}) + 1 \qquad (4)$$

The growth of the number of parity check bits with k is smaller than for $w = 2$, but is still larger than the logarithmic growth of traditional SEC-DED codes. In the second step to constructing the codes, a different combination of w of the $n-k$ added bits is used for each of the first k columns of the H matrix. Equation (1) guarantees that there are sufficient different combinations. The remaining $n - k$ columns form an identity matrix of size $n - k$. The reduction in the number of ones enables a lower encoding and decoding delay. In a

general case, a Hamming code will have rows with a number of ones that is roughly $k/2$. This compares with the proposed SEC codes ($w = 2$) for which the number of ones in a row is by design at most $n - k - 1$. Similarly, to locate an error a traditional SEC code requires an $n-k$ input AND gate compared with a simple two input AND gate in the proposed code. In practical implementations, this results in a significant reduction of the encoding and decoding delays.

One distinct feature of the proposed codes is that they correct errors on the data bits only. This is similar to other codes such as Orthogonal Latin square (OLS) codes [10]. However, in OLS codes, each pair of data bits participates in at most one shared parity check bit to ensure that majority logic decoding can be used. This is different from the proposed scheme in which the goal is to ensure that no data bit participates in all the parity check bits, in which another data bit participates.

This is then used to simplify the location and correction of an error, as described before. Another difference is that OLS codes are commonly used when multiple error correction capabilities are needed although SEC can also be implemented. The main issues with SEC OLS codes are that they are only implemented for a few block sizes and require a large number of parity check bits. Finally, it is worth mentioning that the parity check matrices of the proposed codes are similar to that of low density parity check (LDPC) codes commonly used in communication systems [11]. Nevertheless, since LDPC codes usually have large block size, and must provide multiple error correction, the encoding and decoding procedures are very different from our proposed codes and require complex logic circuitry [11].

D. Architecture for Hamming Distance Computation

The proposed architecture grounded on the data path design is shown in Fig.4. It contains multiple butterfly-formed weight accumulators (BWAs) proposed for the Hamming distance computation. The basic function of the BWA is to count the number of 1's among its input bits. It consists of multiple stages of HAs as shown in Fig.5(a), where each output bit of a HA is associated with a weight. The HAs in a stage are connected in a butterfly form so as to accumulate the carry bits and the sum bits of the upper stage separately. In other words, both inputs of a HA in a stage, except the first stage, are either carry bits or sum bits computed in the upper stage. This connection method leads to a property that if an output bit of a HA is set, the number of 1's among the bits in paths reaching the HA is equal to the weight of the output bit.
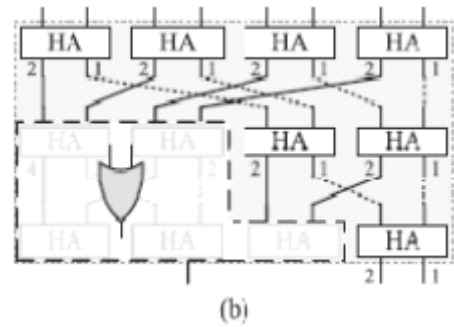

Figure 4:.Architecture of Hamming Distance Computation

Figure 5 (a):. General structure and **(b):** New revised structure.

In Fig. 5(a), for example, if the carry bit of the HA is set, the number of 1's among the associated input bits, i.e., A, B, C, and D, is 2. At the last stage of Fig. 5(a), the number of 1's among the input bits, d, can be calculated as $d = 8I + 4 (J + K + M) + 2 (L + N + O) + P$ (5)

Since what we need is not the precise Hamming distance but the range it belongs to, it is possible to simplify the circuit. When $r$max = 1, for example, two or more than two 1's among the input bits can be regarded as the same case that falls in the fourth range.

In that case, we can replace several HAs with a simple OR-gate tree as shown in Fig. 5(b). This is an advantage over the SA. Note that in Fig. 5 there is no overlap between any pair of two carry-bit lines or any pair of two sum-bit lines. As the overlaps exist only between carry-bit lines and sum-bit lines, it is not hard to resolve overlaps in the contemporary technology that provides multiple routing layers no matter how many bits a BWA takes. We now explain the overall architecture in more detail. Each of the number of gates in the critical path. bThe count of all gates. cThe critical path delay(CPD)in nanoseconds. XOR stage generates the bitwise difference vector for either data bits or parity bits.
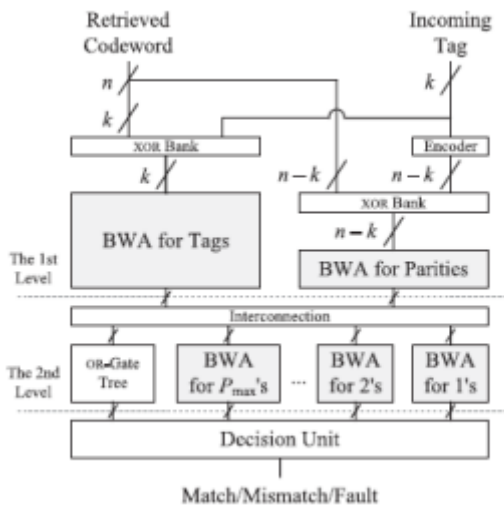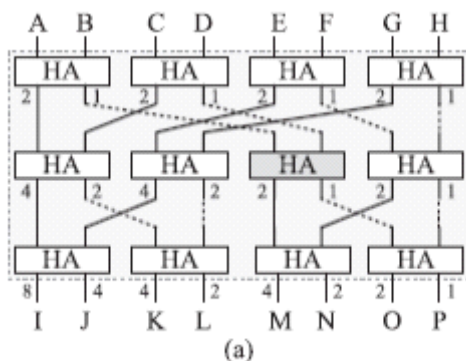
E. General Expressions for the Complexity

The complexity as well as the latency of combinational circuits heavily depends on the algorithm employed. It is unfortunately hard to derive an analytical and fully deterministic equation that shows the relationship between the number of gates and the latency for the proposed architecture The complexity of the proposed architecture, C, can be expressed as

$C = CXOR + CENC + CBWA(k) + CBWA(n-k) + C2nd + CDU \leq n + CENC + 2CBWA(n) + CDU$ (6)

where $CXOR$, $CENC$, $C2nd$, CDU, and $CBWA(n)$ are complexities of XOR banks, an encoder, the second level circuits, the decision unit, and a BWA for n inputs, respectively. Using the recurrence relation, $CBWA(n)$ can be calculated as

$CBWA(n) = CBWA(n/2) + CBWA(n/2) + 2[n/2]$

F. General Expressions for the Latency

The latency of the proposed architecture, L, can be expressed as

$L \leq \max[LXOR + LBWA(k), LENC + LXOR + LBWA(n-k)] + L2nd + LDU$ (8)

where $LXOR$, $LENC$, $L2nd$, LDU, and $LBWA(n)$ are the latencies of an XOR bank, an encoder, the second level circuits, the decision unit, and a BWA for n inputs, respectively. Note that the latencies of the OR-gate tree and BWAs for $x \leq n$ inputs at the second level are all bounded by $\log 2n$. As one of BWAs at the first level finishes earlier than the other, some components at the second level may start earlier.

Similarly, some BWAs or the OR-gate tree at the second level may provide their output earlier to the decision unit so that the unit can begin its operation without waiting for all of its inputs. In such cases, $L2nd$ and LDU can be partially hidden by the critical path of the preceding circuits, and L becomes shorter than the given expression.

4. Results

Table 2: Comparison For Latency And Complexity

ECC	Architecture	Gate-Level Counting		Implementation Results	
		Latency[a]	Complexity[b]	CPD[c]	EGC[d]
(16, 11)	Conventional	14 (1.17)[e]	137 (1.10)	2.13 (1.16)	320 (1.20)
	SA-based	14 (1.17)	132 (1.06)	2.12 (1.16)	304 (1.14)
	Proposed	12 (1.00)	125 (1.00)	1.83 (1.00)	266 (1.00)
(24, 18)	Conventional	16 (1.23)	238 (1.24)	2.31 (1.16)	491 (1.19)
	SA-based	18 (1.38)	211 (1.10)	2.46 (1.23)	475 (1.15)
	Proposed	13 (1.00)	192 (1.00)	2.00 (1.00)	412 (1.00)
(31, 25)	Conventional	16 (1.23)	336 (1.29)	2.48 (1.24)	684 (1.22)
	SA-based	18 (1.38)	290 (1.11)	2.57 (1.29)	634 (1.13)
	Proposed	13 (1.00)	261 (1.00)	1.99 (1.00)	561 (1.00)
(40, 33)	Conventional	18 (1.20)	473 (1.38)	2.64 (1.20)	861 (1.21)
	SA-based	22 (1.47)	377 (1.10)	2.96 (1.35)	816 (1.15)
	Proposed	15 (1.00)	342 (1.00)	2.20 (1.00)	709 (1.00)

a The number of gates in the critical path.
b The count of all gates.
c The critical path delay(CPD)in nanoseconds.
d The equivalent gate count(EGC).
e The numbers in the parenthesis are normalized values

Table II shows the latencies and hardware complexities resulting from three architectures: 1) the conventional decode-and-compare; 2) The SA-based direct compare; and 3) the proposed ones. In [5], the latency is measured from the time when the incoming address is completely encoded. As the critical path starts from the arrival of the incoming address to a cache memory, the encoding delay must be, however, included in the latency computation. The latency values in Table II are all measured in this way. Besides, critical-path delays in Table II are obtained by performing post layout simulations and equivalent gate counts are measured by counting a two-input NAND as one. As shown in Table II, the proposed architecture is effective in reducing the latency as well as the hardware complexity even with considering the practical factors. Note that the effectiveness of the proposed architecture over the SA-based one in shortening the latency gets larger as the size of a codeword increases. The reason is that, the latencies of the SA-based architecture and the proposed one is dominated by SAs and

HAs, respectively. As the bit-width doubles, at least one more stage of SAs or HAs needs to be added. Since the critical path of a HA consists of only one gate while that of a SA has several gates, the proposed architecture achieves lower latency than its SA-based counterpart, especially for long code words.

5. Conclusion

To reduce the latency and hardware complexity, a new architecture has been presented for matching the data protected with an ECC. The proposed architecture examines whether the incoming data matches the stored data if a certain number of erroneous bits are corrected. To reduce the latency, the comparison of the data is parallelized with the encoding process that generates the parity information. It is based on the fact that the systematic codeword has separate fields for the data and parity. In addition, an efficient processing architecture has been presented to further minimize the latency and complexity. As the proposed architecture is effective in reducing the latency as well as the complexity considerably, it can be regarded as a promising solution for the comparison of ECC-protected data. Though this brief focuses only on the tag match of a cache memory, the proposed method is applicable to diverse applications that need such comparison.

References

[1] J.D. Warnock, Y.H. Chan, S. M.Carey, H.Wen, P. J. Meaney, G.Gerwig, H.H.Smith, Y.H.Chan, J. Davis, P. Bunce, A.Pelella, D.Rodko, P.Patel, T.Strach, D.Malone, F. Malgioglio, J. Neves, D. L. Rude, and W. V. Huott "Circuit and physical design implementation of the microprocessor chip for the zEnterprise system," IEEE J. Solid-State Circuits, vol. 47, no. 1, pp. 151– 163, Jan. 2012.

[2] B.Y Kong, Jihyuck Jo, Hyewon Jeong, Mina Hwang, Soyoung Cha, Bongjin Kim, and In-Cheol Park, "Low- Complexity Low-Latency Architecture for Matching of Data Encoded With Hard Systematic Error-CorrectingCodes," IEEE Trans. Very Large Scale Integr.(VLSI) *Syst.*, vol. 22, no. 7, pp. 1648 - 1652, July. 2014.

[3] H. Ando, Y. Yoshida, A. Inoue, I. Sugiyama, T. Asakawa, T. Muta, K.Morita, T. Motokurumada, S. Okada, H. Yamashita, Y. Satsukawa, A. Konmoto, R. Yamashita, and H. Sugiyama, "A 1.3 GHz fifth generation SPARC64 microprocessor," in *ISSCC. Dig. Tech. Papers*, 2003, pp. 246–247.

[4] AMD Inc., Sunnyvale, CA, "Family 10h AMD Opteron™ Processor Product Data Sheet," PID: 40036 Rev: 3.04, 2010. Available:http://support.amd.com/us/Processor_TechDocs/40036.pdf [Online]

[5] W.Wu, D. Somasekhar, and S.-L. Lu, "Direct compare of information coded with error-correcting codes," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 20, no. 11, pp. 2147–2151, Nov. 2012.

[6] Pedro Reviriego, Salvatore Pontarelli, Juan Antonio Maestro, and Marco Ottavi, "A Method to Construct Low Delay Single Error Correction Codes for Protecting Data Bits Only," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems.*, vol. 32, no. 3, pp. 479 - 483, March 2013.

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

[8] 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, 1984.

[9] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Fault tolerant solid state mass memory for space applications," *IEEE Trans. Aerospace Electron. Syst.*, vol. 41, no. 4, pp. 1353–1372, Oct. 2005.

[10]M. Y. Hsiao, D. C. Bossen, and R. T. Chien, "Orthogonal latin square codes," *IBM J. Res. Develop.*, vol. 14, no. 4, pp. 390–394, Jul. 1970.

[11] G. Li, I. J. Fair, and W. A. Krzymien, "Low-density parity-check codes for space-time wireless transmission," *IEEE Trans. Wirel. Commun.*, vol.5, no. 2, pp. 312–322, Feb. 2006.

[12]M. Y. Hsiao "A class of optimal minimum odd-weight column SECDED codes," *IBM J. Res. Develop.*, vol. 14, pp. 395–301, Jul. 1970.

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

[14] V. Gherman, S. Evain, N. Seymour, and Y. Bonhomme, "Generalized parity-check matrices for SEC-DED codes with fixed parity," in *Proc. IEEE On-Line Testing Symp.*, Jul. 2011, pp. 198–20.

Author-1:-

Khadeejah Neelofer Roohi completed her B.Tech from Balaji Institute of Engineering and Sciences; and Pursuing M.Tech in Vaagdevi College of Engineering.

Author-2:-

Mr. M.A.Himayath Shamshi is working as Asst.prof in Dept of ECE, Vaagdevi College of Engineering, Bollikunta, Warangal.

Author-3:-

Professor P Prasad Rao, Head of the Department of ECE, Vaagdevi College of Engineering Bollikunta, Warangal.