International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017

A VLSI Design of a Novel Architecture for Orthogonal Latin Square Codes

Yenikapati Chanakya & K. Rajasekhar
yenikapati.chanakya@gmail.com[1] ; kasseyrajasekhar@gmail.com[2]
[1]PG Scholar, M.Tech (VLSI SD) Dept of ECE, ASR College Of Engineering, Tetali, Tanuku, West Godavari, Andhra Pradesh.
[2]Asscoiate Professor, Dept of ECE,ASR College Of Engineering, Tetali, Tanuku, West Godavari, Andhra Pradesh.

Abstract—*Reliability is a major concern in advanced electronic circuits. Errors caused for example by radiation become more common as technology scales. To ensure that those errors do not affect the circuit functionality a number of mitigation techniques can be used. Among them, Error Correction Codes (ECC) are commonly used to protect memories and registers in electronic circuits. When ECCs are used, it is of interest that in addition to correcting a given number of errors, the code can also detect errors exceeding that number. This ensuresthat uncorrectable errors are detected and therefore silent data corruption does not occur. Among the ECCs used to protect circuits, one option is Orthogonal Latin Squares (OLS) codes for which decoding can be efficiently implemented. In this paper, an enhancement of the decoding for Double Error Correction (DEC) OLS codes is proposed. The proposed scheme tries to reduce the probability of silent data corruption by implementing mechanisms to detect errors that affect more than two bits.*

Keywords—Concurrent error detection, error correction codes (ECC), Latin squares, majority logic decoding (MLD), parity, memory.

I. Introduction

The general idea for achieving error detection and correction is to add some redundancy which means to add some extra data to a message, which receiver can use to check uniformity of the delivered message, and to pick up data determined to be corrupt. Error-detection and correction scheme may be systematic or it may be non-systematic. In the system of the module non-systematic code, an encoded is achieved by transformation of the message which has least possibility of number of bits present in the message which is being converted. Another classification is the type of systematic module unique data is sent by the transmitter which is attached by a fixed number of parity data like check bits that obtained from the data bits. The receiver applies the same algorithm when only detection of the error is required to the received data bits which is then compared with its output with the receive check bits if the values does not match, there we conclude that an error has crept at some point in the process of transmission. Error correcting codes are regularly used in lower-layer communication, as well as for reliable storage in media such as CDs, DVDs, hard disks and RAM.

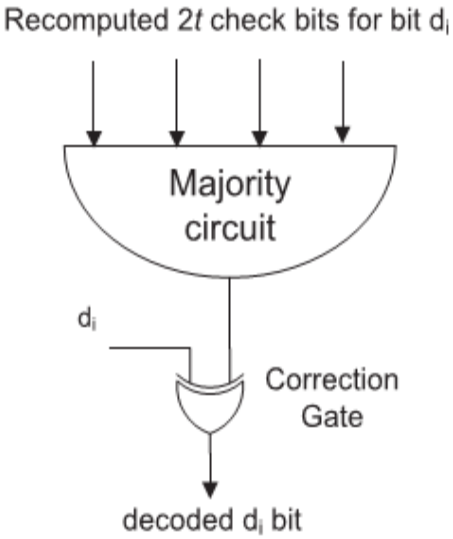


Fig.1. Illustration of OS-MLD decoding for OLS codes

Provision against soft errors that apparent they as the bit-flips in memory is the main motto of error detection and correction. Several techniques are used present to midi gate upsets in memories. For example, the Bose –Chaudhuri– Hocquenghem codes, Reed–Solomon codes, punctured difference set codes, and matrix codes has been used to contact with MCUs in memories. But the above codes mentioned requires more area, power, and delay overheads since the encoding and decoding circuits are more complex in these complicated codes. Reed-Muller code is another protection code that is able to detect and correct additional error besides a Hamming code. But the major drawback of this protection code is the more area it requires and the power penalties.
Reliability is a major issue for advanced electronic circuits. As technology scales, circuits become more

vulnerable to error sources such as noise and radiation and also to manufacturing defects and process variations. A number of error mitigation techniques can be used to ensure that errors do not compromise the circuit functionality.Among those, Error Correction Codes (ECCs) are commonlyused to protect memories or registers. Traditionally, Single Error Correction (SEC) codes that can correct one bit error in a word are used as they are simple to implement and require few additional bits. A SEC code requires a minimum Hamming distance between code-words of three. This means that if a double error occurs, the erroneous word can be at distance of one from another valid word. In that case, the decoder will miss-correct the word creating an undetected error. To avoid this issue, Single Error Correction Double Error Detection (SEC-DED) codes can be used. Those codes have a minimum Hamming distance of four. Therefore, a double error can in the worst case cause the word to be at a distance of two of any other valid word so that miss-correction is not possible. More generally, for a code that can correct t errors, it is of interest to also detect t+1 errors. This reduces the probability of undetected errors that can cause Silent Data Corruption (SDC). SDC is especially dangerous as the system continues its operation unaware of the error and this can lead to further data corruption or to an erroneous behavior long after the original error occurred.

II. Literature Survey

Most prior work in memory ECC has focused on low failure rates present at normal operating voltages, and has not focused on the problem of persistent failures in caches operating at ultra low voltage where defect rates are very high.

For high defect rates, memory repair schemes based on spare rows and columns are not effective. Much higher levels of redundancy are required that can tolerate multi-bit errors in each cache line. In addition to the techniques in [Wilkerson 08] mentioned earlier, other prior work includes the twodimensional ECC proposed by [Kim 07] which tolerates multiple bit errors due to non-persistent faults, but is slow and complicated to decode.

Similarly the approach in [Kim 98] can tolerate as many faults as can be repaired by spare columns, which would be insufficient in the present context with high bit-error rate. In some cases, check bits are used along with spare rows and columns to get combined fault-tolerance. In [Stapper 92], interleaved words with redundant word lines and bit lines are used in addition to the check bits on each word. [Su 05] proposes an approach where the hard errors are mitigated by mapping to redundant elements and ECC is used for the soft errors. Such approaches will not be able to provide requisite fault tolerance under high bit error rates when there are not enough redundant elements to map all the hard errors.

The application of OLS codes for handling the high defect rates in low power caches as described in [Christi 09] provides a more attractive solution. While OLS codes require more redundancy than conventional ECC, the one-step majority encoding and decoding process is very fast and can be scaled up for handling large numbers of errors as opposed to BCH codes, which while providing the desired level of reliability requires multi-cycles for decoding [Lin 83]. The post-manufacturing customization approach proposed in this paper can be used to reduce the number of check bits and hence the amount of redundancy required in the memory while still providing the desired level of reliability. Note that the proposed approach does not reduce the hardware requirements for the OLS ECC as the whole code needs to be implemented on-chip since the location of the defects is not known until post-manufacturing test is performed.

III.Orthogonal Latin Squares codes

The concept of Latin squares and their applications are well known [12]. A Latin square of size m is an m * m matrix that has permutations of the digits 0,1,..m-1 in both its rows and columns. For each value of m there can be more than one Latin square. When that is the case, two Latin squares are said to be orthogonal if when they are superimposed every ordered pair of elements appears only once. Orthogonal Latin Squares (OLS) codes are derived from OrthogonalLatin squares [9]. These codes have k=m2 data bits and 2tm check bits where t is the number of errors that the code can correct. For a Double Error Correction (DEC) code t=2 and therefore 4m check bits are used. One advantage of OLS codes is that their construction is modular. This means that to obtain a code that can correct t+1 errors,simply 2m check bits are added to the code that can correct t errors. The modular property enables the selection of the error correction capability for a given word size.As mentioned in the introduction, OLS codes can be decoded using One Step Majority Logic Decoding (OS-MLD) as each data bit participates in exactly 2t check bits and each other bit participates in at most one of those check bits. This enables a simple correction when the number of bits in error is t or less. The 2t check bits are recomputed and a majority vote is taken, if a value of one is obtained, the bit is in error and must be corrected. Otherwise the bit is correct. As long as the number of errors is t or less this ensures the error correction as the remaining t-1 errors can, in the worst case affect t-1 check bits so that still a majority of t+1 triggers the correction of an erroneous bit. For an OLS code that can correct t errors using OS-MLD, t+1 errors can cause miss-corrections. This occurs for example if the errors affect t+1 paritybits in whichbit diparticipates as this bit will be miss-corrected. The same occurs when the number of errors is larger than t+1. Each of the 2t check bits in which a data bit participates is taken from a group of m parity bits. Those

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017

groups are bits 1 to m, m+1 to 2m, 2m+1 to 3m and 3m+1 to 4m.

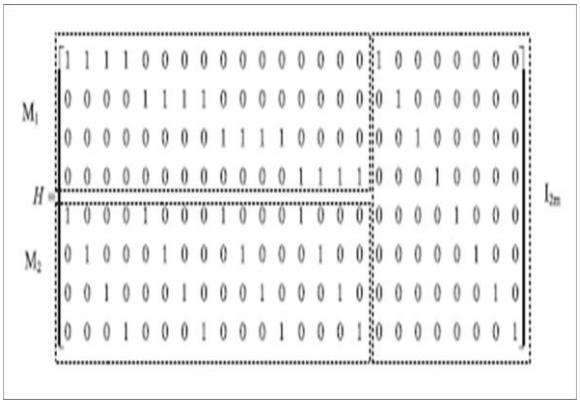


Fig 2: Parity check matrix for OLS code having k and t as 16&1

The 'H' matrix for OLS codes is build from their properties. The matrix is capable of correcting single type error. By the fact that in direction of the modular structure it might be able to correct many errors. They have check bits of number "2tm" in which, „t" stands for numeral of errors such that code corrects. If we wanted to correct a double bit then we have „2" as the value of t and thereby the check bits required are 4m.the H matrix , of Single Error Code „OLS" code is construct as :

$$H = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} \quad I_{2m} \qquad (1)$$

a. In the above, I2m is the identity matrix of size 2 m.
b. M1, M2 are the matrices of given size m × m2.
„"The matrix M1 have m ones in respective rows. For the rth row, the 1"s are at the position (r − 1) × m + 1,(r − 1)× m + 2,……..(r − 1) × m+ m − 1, (r − 1) × m + m". The matrix M2 is structured as:M2 = [ImIm . . . Im] (2). For the given value 4 for m, the matrices M1 and M2 can be evidently experiential in Fig. H Matrix in the check bits we remove is evidently the G Matrix

$$G = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix} \qquad (3)$$

On concluding the above mentioned, it is evident that the encoder is intriguing m2 data bits and computing 2tm parity check bits by using G matrix . These resulted from the Latin Squares have the below properties:
a. Exactly in 2t parity checks each info bit is involved.
b. Utmost one in parity check bits info bits takes participation.
We use the above properties in the later section to examine our proposed technique.

IV. Proposed Method

The proposed method is based on the observation that by construction, the groups formed by the mparity bits in each Mi matrix have at most a one in every column of H.For theexample in Fig. 2, those groups correspond to bits (or rows) 1–4 (M1), 5–8 (M2), 9–12 (M3), and 13–16 (M4). Therefore, anycombination of four bits from one of those groups will at most sharea one with the existing columns inH. For example, the combinationformed by bits 1, 2, 3, and 4 shares only bit 1 with columns 1, 2, 3,and 4. This is the condition needed to enable OS-MLD. Therefore,combinations of four bits taken all from one of those groups canbe used to add data bit columns to theHmatrix. For the code withk=16 andt =2 shown in Fig. 2, we havem=4. Hence, one combination can be formed in each group by setting all the positions in thegroup to one. This is shown in Fig. 3, where the columns added arehighlighted. In this case, the data bit block is extended fromk=16 to
k=20 bits.
Before describing the proposed error detection techniques, the standard definition of self-checking circuits that are used in this section is presented. During normal, or fault-free, operation, a circuit receives only a subset of the input space, called the input code space, and produces a subset of the output space, called the output code space. The outputs that are not members of the output code space form the output error space. In general, a circuit may be designed to be self-checking only for an assumed fault set. In this brief, we consider the fault set ***F*** corresponding to the single stuck-at fault model. A circuit is self-checking [20] if and only if it satisfies the following properties: 1) it is self-testing, and 2) fault-secure. A circuit is self-testing if, for each fault *f* in the fault set ***F***, there is at least one input belonging to the input code space, for which the circuit provides an output belonging to the output error space.

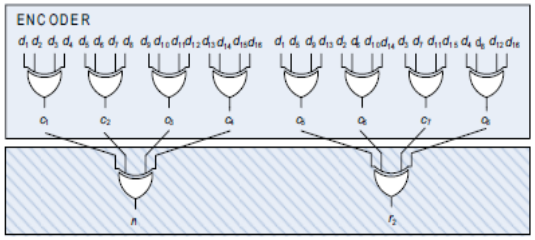


Fig. 3. Proposed self-checking encoder for OLS code with $k = 16$ and $t = 1$.

A circuit is fault-secure if, for each fault *f* in the fault set ***F*** and for each input belonging to the input code space, the circuit provides the correct output, or an output belonging to the output error space. The fault-secure property guarantees that the circuit gives the correct response, or signals the presence of a fault that provides an output in the error space. Faults are always detected, since there is an input that produces an output that identifies the

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017

presence of the fault.This property is related to the assumption that the interval between the occurrences of two faults is enough to permit to all the elements belonging to the input code space to appear as circuit inputs before the occurrence of the second fault. Thus, an output belonging to the output error space appears at the circuit output before the occurrence of the second fault.

The technique that we propose is based on the use of parityprediction, which is one of the techniques commonly used to detect error in general logic circuits. In our case, the problem is substantially simpler, given the structure of the OLS codes. For the encoder, it is proposed that the parity of the computed check bits (*ci)* is compared against the parity of all the check equations. The parity of all the check equations is simply the equation obtained by computing the parity of the columns in *G*. For OLS codes, since each column in *G* has exactly 2*t* ones, the null equation is obtained (see, for example, Fig. 1). Therefore, the concurrent error detection (CED) scheme is simply to check

$$c_1 \oplus c_2 \oplus c_3 \oplus \cdots \oplus c_{2tm} = 0.$$

This enables an efficient implementation that is not possible in other codes. For example, in a Hamming code a significant part of the columns in *G* has an odd weight and for some codes the number is even larger as they are designed to have odd weights. The input code space of the OLS encoder corresponds to the input space, since the encoder can receive all the possible 2*k*
input configurations. The output code space of the OLS encoder is composed by the outputs satisfying (4), while the output error space is the complement of the output code space. A fault that occurs in one of the gates composing the OLS encoder can change at most one of the *ci* check bits. When this change occurs, the OLS encoder provides an output that does not satisfy (4), i.e., an output belonging to the output error space.

Hence, this guarantees the fault-secure property for this circuit. Additionally, since the encoder is composed only by XOR gates, no logic masking is performed in the circuit. Therefore, when a fault is activated the error is propagated to the output. This ensures the self-testing property of the circuit. In order to check if the output of the OLS encoder belongs to the output code space or the output error space, a self-checking implementation of a parity checker is used. The checker controls the parity of its inputs and is realized with a repetition code. The two outputs (*r*1, *r*2*)* are each equal to the parity of one of two disjoint subsets of the checker inputs (c*i)*, as proposed . When a set of inputs with the correct parity is provided, the output
code {*r*1, *r*2} takes the values 00 or 11.

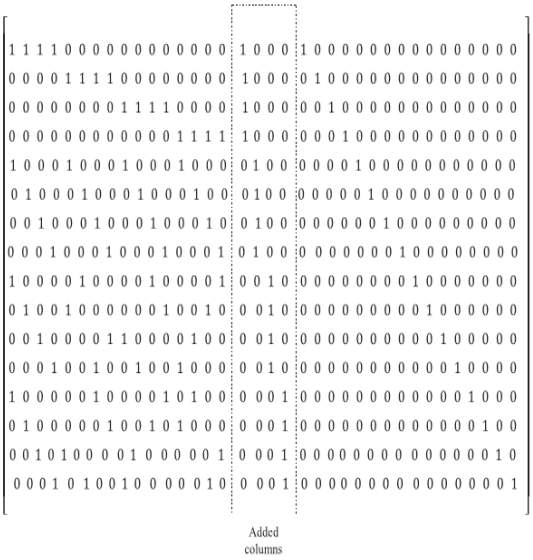


Fig. 4. Parity check matrix H for the extended OLS code with k=20 and t =2

The proposed method first divides the parity check bits in groupsofmbits given by the Mimatrices. Then, the second step is foreach group to find the combinations of 2t bits such that any pairof them share at most one bit. This second step can be seen asthat of constructing an OS-MLD code with mparity check bits.Obviously, to keep the OS-MLD property for the extended code,the combinations formed for each group have to share at most onebit with the combinations formed in the other 2t −1 groups. This isnot an issue as by construction, different groups do not share any bit.Whenmis small finding, such combinations is easy. For example,in the case considered in Fig. 4, there is only one possible combination per group. Whenmis larger, several combinations can be formedin each group. This occurs, for example, when m=8. In this case,the OLS code has a data block size k =64. With eight positionsin each group, now two combinations of four of them that shareat most one position can be formed. This means that the extendedcode will have eight (4×2) additional data bits. As the size of theOLS code becomes larger, the number of combinations in a groupalso grows. For the case m=16 and k =256, each group has16 elements. Interestingly enough, there are 20 combinations of fourelements that share at most one element. In fact, those combinationsare obtained using the extended OLS code shown in Fig. 3 in eachof the groups. Therefore, in this case, 4×20=80 data bits can beadded in the extended code. The parameters of the extended codes, wheren−k =2tm is the number of paritybits. The data block size for the original OLS codes (kOLS) is alsoshown for reference The method can be applied to the general case of an OLS code with k =m2 that can correct t errors. Such a code has 2tm parity bits that as before are divided in groupsofmbits. The code can be extended by selecting

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017

combinations of 2t parity bits taken from each of the groups. These combinations can be added to the code as longas any pair of the new combinations share at most one bit. When m is small, a set of such combinations with maximum size can be easilyfound. However, as m grows, finding such a set is far from trivial (as mentioned before, solving that problem is equivalent to designing an OS-MLD code with m parity bits that can correct t errors). An upper bound on the number of possible combinations can be derived by observing that any pair of bits can appear only in one combination. Because each combination has 2t bits, there are (2t 2) pairs in each combination. The number of possible pairs in each group of m bits is m 2. Therefore, the number of combinations NG in a group of m bits has to be such that

$$\binom{m}{2} \geq \binom{2t}{2} \times N_G \qquad (2)$$

which can be simplified as

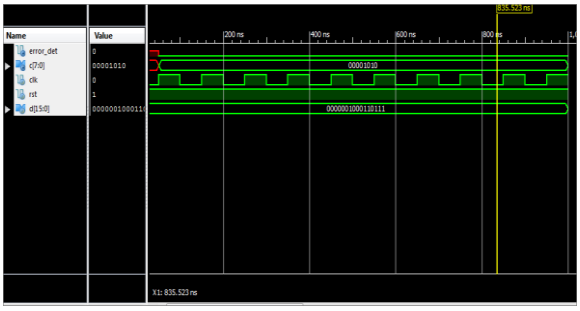
$$\frac{m^2 - m}{4t^2 - 2t} \geq N_G. \qquad (3)$$

One particular case for which a simple solution can be found is when m=2t ×l. In this case, an OLS code with a smaller data block size (l2) can be used in each group. One example for t =2 is when m=16 (k=256) for which the OLS code with block size k=4^2 can be used in each group as explained before. Similarly, for t =2, whenk=1024 (m=32) the OLS code of size k =8^2 can be used in each group.

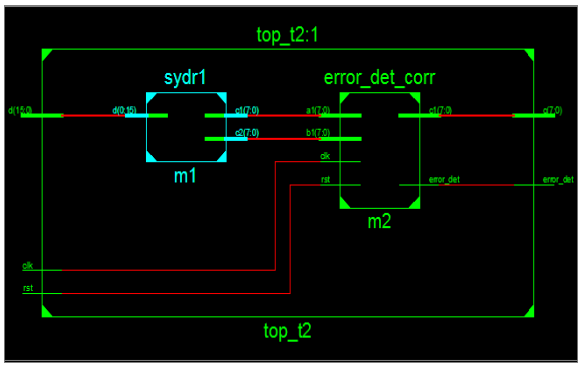
V. Results.

Proposed System.

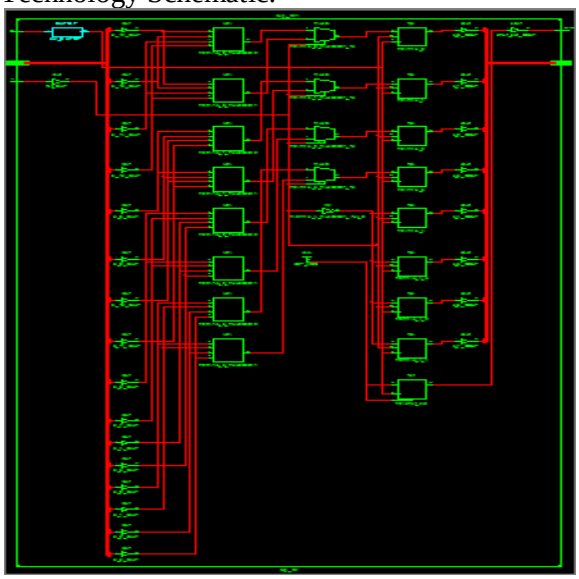
Simulation.



RTL Schematic.



Technology Schematic.



Design Summary.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	4	4656	0%
Number of 4 input LUTs	9	9312	0%
Number of bonded IOBs	27	232	11%
Number of GCLKs	1	24	4%

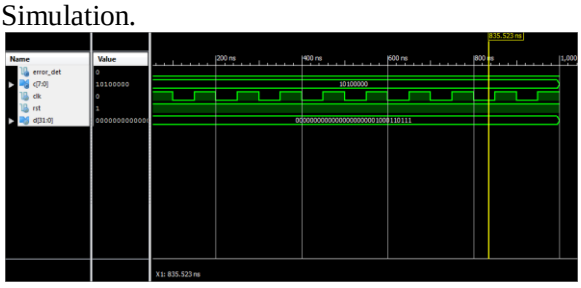
Timing Summary.

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: No path found
   Minimum input arrival time before clock: 3.320ns
   Maximum output required time after clock: 4.040ns
   Maximum combinational path delay: No path found
```

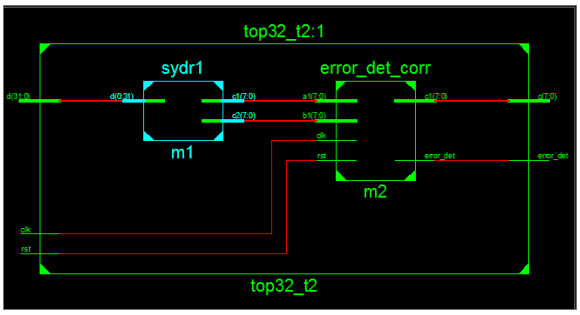
Extension.

Simulation.

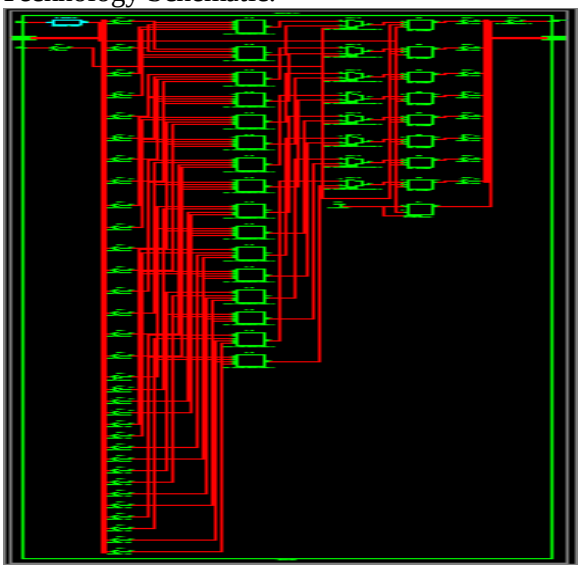


RTL Schematic.

International Journal of Research
Available at https://edupediapublications.org/journals
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 04 Issue 14
November 2017



Technology Schematic.



Design Summary.

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	8	4656	0%
Number of 4 input LUTs	16	9312	0%
Number of bonded IOBs	43	232	18%
Number of GCLKs	1	24	4%

Timing Report.

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: No path found
   Minimum input arrival time before clock: 2.894ns
   Maximum output required time after clock: 4.040ns
   Maximum combinational path delay: No path found
```

VI. Conclusion

In this brief, a CED technique for OLS codes encoders and syndrome computation was proposed. The proposed technique took advantage of the properties of OLS codes to design a parity prediction scheme that could be efficiently implemented and detects all errors that affect a single circuit node. The technique was evaluated for different word sizes, which showed that for large words the overhead is small. This is interesting as large word sizes are used, for example, in caches for which OLS codes have been recently proposed. The proposed error checking scheme required a significant delay; however, its impact on access time could be minimized. This was achieved by performing the checking in parallel with the writing of the data in the case of the encoder and in parallel with the majority voting and error correction in the case of the decoder.In a general case, the proposed scheme required a much larger overhead as most ECCs did not have the properties of OLS codes. This limited the applicability of the proposed CED scheme to OLS codes. The availability of low overhead errordetection techniques for the encoder and syndrome computation is an additional reason to consider the use of OLS codes in high-speed memories and caches.

REFERENCES

[1] 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.
[2] E. Fujiwara, Code Design for Dependable Systems: Theory and Practical Application. New York: Wiley, 2006.
[3] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in Proc. IEEE VLSI Test Symp., May 2007, pp. 349–354.
[4] R. Naseer and J. Draper, "DEC ECC design to improve memory reliability in sub -100nm technologies," in Proc. IEEE Int. Conf. Electron., Circuits, Syst., Sep. 2008, pp. 586–589.
[5] G. C. Cardarilli, M. Ottavi, S. Pontarelli, M. Re, and A. Salsano, "Fault tolerant solid state mass memory for space applications," IEEE Trans. Aerosp. Electron. Syst., vol. 41, no. 4, pp. 1353–1372, Oct. 2005.
[6] S. Lin and D. J. Costello, Error Control Coding, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2004.
[7] S. Ghosh and P. D. Lincoln, "Dynamic low-density parity check codes for fault-tolerant nano-scale memory," in Proc. Found.Nanosci., 2007, pp. 1–5.
[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.