A Method for Error Detection and Correction of Fault Secure OLS Decoder

[1]Sahad B, [2]Chithra M
[1] PG Student [VLSI and ES] [2] Assistant professor,
Department of ECE, TKM Institute of Technology, Kollam

***Abstract*— Reliability is one of the major issues of advanced electronic circuits. In recent years there are several error correction codes (ECC) developed to protect the memories and registers in electronic circuits. But the encoder and decoder circuit may also suffer errors, for these reasons concurrent error detection (CED) and correction technique for orthogonal Latin square (OLS) decoder is proposed. This technique is strongly fault secured for single stuck at faults. The most significant advantage is that it achieves 100 percentage fault coverage for the whole CED circuit. The CED and correction is applicable to both binary and non binary OLS codes also. The proposed method can be achieved by performing the checking in parallel with the majority voting, syndrome generator and error corrector and orthogonal generator in the case of the decoder. Synthesis can be done in Xilinx ISE design suite 13.2 and simulation can be done with ModelSim.**

***Index Terms*— Error Correction Codes (ECC), Concurrent Error Detection (CED), Orthogonal Latin Square (OLS).**

I. INTRODUCTION

Memories are used to store information bits, but they are susceptible to defects due to transient errors, power supply noise etc. Error correcting codes (ECC) are used to protect the data integrity of the memory, i.e.; means of introducing redundancy in the data. Hamming codes, BCH codes and Reed Solomon codes are the commonly used error correcting codes but they have many limitation occurs in correcting more number of bits. Commonly used error correcting codes are more complex codes and also they can correct more errors, but it increases delay, power and complexity. One step majority logic decodable (OSMLD) are recently used to overcome these issues and it can be decoded with low latency, therefore protect memories. Orthogonal latin squares (OLS) code is another type of code that use OS-MLD, OLS codes for inter-connections, memories, and caches because of their modularity.

In the case of ECCs using encoders and decoders for controlling the errors in memory system. But the encoder and decoder circuitry around the memory blocks become susceptible to soft errors, due to the increase in soft error rate of logic circuits. Control of these errors is usually accomplished by concurrent error detection (CED) and used to enhance system dependability, also protect the entire circuit of the OLS parallel decoders.

II. LITERATURE REVIEW

Scaling down of semiconductor devices to nanoscale causes threshold voltage variation, negative bias temperature instability, short channel, single bit upsets and multiple cell upsets. Thus to make memory cells as fault tolerant as possible, Error Correction Codes (ECCs) are used. An important issue is that the encoder and decoder circuits needed to use ECCs can also suffer errors. CED scheme used to protect these surrounding circuits from errors. Double modular redundancy (DMR) is one of the most-used CED schemes, in which circuits are simply duplicated and their output compared. The main disadvantage of this technique is large hardware overhead due to additional checking hardware for the two copies. Other kind of CED scheme exists for specific and commonly used circuits with Reed-Solomon codes and EG-LDPC codes are reviewed. CED scheme for OLS code are available which protect whole encoder, but only part of decoders, i.e. the syndrome generators . Hence a CED scheme for the entire OLS decoder is required. This paper presents a CED scheme that protects the entire circuits of the OLS parallel decoders at 100% coverage of single stuck at fault along with multiple bit error correction.

III. PROPOSED SYSTEM

Orthogonal Latin Square codes (OLS) used in error detection and correction because of their modularity and low delay implementations due to the presence simplicity of decoding algorithm. Concurrent error detection (CED) and correction technique also protect memory and surrounding systems.

A. *CED scheme for OLS code*

Errors occur in a memory system is usually corrected by using Error correction codes (ECCs); however, these ECCs do not protect the faults occur in the decoder and encoder. In order to protect these surrounding systems from errors CED technique is used. The detailed design and analysis of the CED shows that it is strongly fault secure (SFS) for single stuck at faults. Here, error represent an error occurring in the memory that errors must be controlled by the ECC, and not by the CED. Whereas fault represent a fault occurring in the encoder or in decoder. Faults must be detected by the CED. By using property of OLS code properly CED technique is implemented with high speed and also multiple-bit error correction is possible.

During the encoding process, input data bits are provided to the OLS encoder. By using the parity check matrix and OLS code properties check bits can be obtained

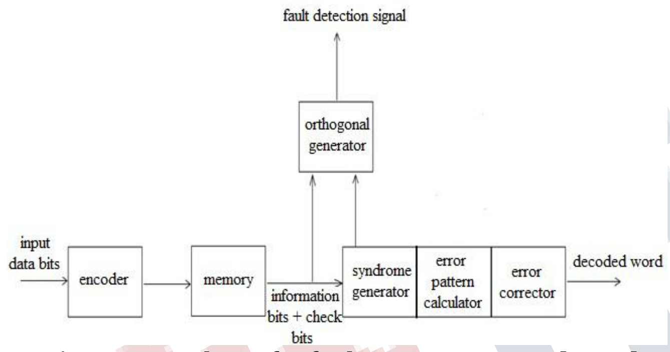


Fig. 1. CED scheme for fault secure OLS encoder and decoder.

From input data bits. Once encoding process gets completed, this code word is stored to the memory. If multiple-bit error occurs in memory block, this can be corrected during the decoding process. The basic block diagram for CED scheme of fault secure OLS encoder and decoder is as shown in Fig. 1.CED scheme of fault secure OLS encoder and decoder consists of a OLS encoder, memory unit and OLS decoder. The decoder consists of a syndrome calculator, error pattern calculator, error corrector and an orthogonal generator for fault detection.

B. *OLS encoder*

The CED technique used in the encoder is based on the use of parity prediction, which is one of the techniques commonly used to detect error in general logic circuits. Here the parity of the computed check bits c_i 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 2t ones, the null equation is obtained (see, H matrix, Fig. 5.). Therefore, the concurrent error detection (CED) scheme is simply to check

$$C_1 \oplus C_2 \oplus C_3 + \ldots\ldots \oplus C_{2tm} = 0 \qquad (1)$$

This enables an efficient implementation that is not possible in other codes. When a fault occurs in any one of the gates in encoder that can change at most one of the c_i check bits and does not satisfy. Hence this guarantees 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. Any circuits ensure the two property fault secure and self-testing is known as self-checking. These self-checking circuits are fully fault secured.

Let c_i be a sub-vector of check bits c corresponding to M_i. c_i is generated as $c_i = dMi^T$, where d denotes the information bits; so, $c = (c_1c_2\ldots..c_{2t})$ can be easily obtained using the H matrix and OLS code properties. For example $C_1 = d_{24} \oplus d_{23} \oplus d_{22} \oplus d_{21}\ldots\ldots\ C_{30} = d_{20} \oplus d_{19} \oplus d_{13} \oplus d_7 \oplus d_1$.

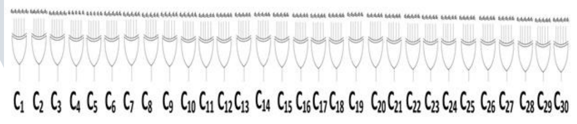

C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12 C13 C14 C15 C16 C17 C18 C19 C20 C21 C22 C23 C24 C25 C26 C27 C28 C29 C30

Fig. 2. Encoder for OLS code with k=25 and t=1

C. *OLS decoder*

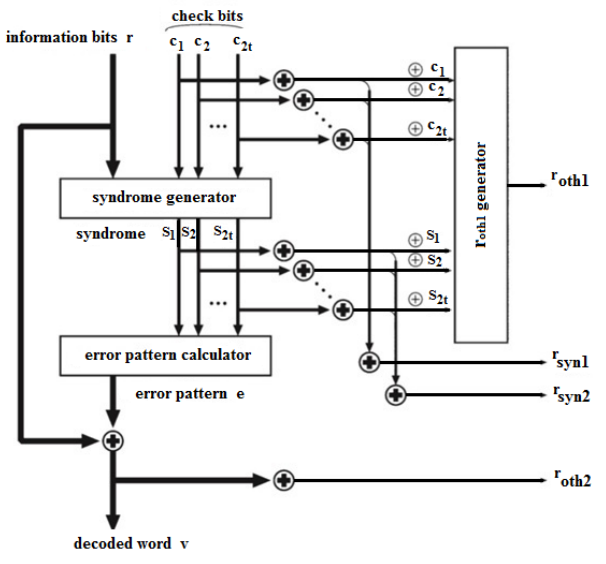


Fig. 3. Decoder design using CED

The output from the encoder is provided to the memory. Suppose if the information bits stored in the memory block, gets flipped it can be corrected during the decoding process . By using CED technique, single stuck at the input and output of the gates also protected. Fig. 3. shows the construction of the OLS decoder by the CED scheme; the decoder includes the syndrome generator, error pattern calculator and roth1 generator. CED technique requires two pairs of signals for fault detection: (r_{syn1}; r_{syn2}) and (r_{oth1}; r_{oth2}). If no fault occurs, ($r_{syn1} = r_{syn2}$) and ($r_{oth1} = r_{oth2}$). If a single stuck-at fault occurs, either or both ($r_{syn1} \neq$ rsyn2) and/or (roth1≠roth2), thus detecting the fault.

D. Syndrome generator

Syndrome generator compare the check bits of information bit received from the memory to the check bits of input data bits. If all the syndrome bits are 0, there is no error occurred in the memory. Else there is error exist in the received word. Let $\mathbf{C_{ri}}$ be a sub-vector of check bits $\mathbf{C_r}$ corresponding to $\mathbf{M_i}$. $\mathbf{C_{ri}}$ is generated as $\mathbf{C_{ri}} = d\mathbf{M_i}^{\mathbf{T}}$ where d denotes the information bits; so, $\mathbf{C_r} = (C_1\ C_2\ C_{2t})$. Let $\mathbf{C_r}$ be a sub vector of the syndrome S corresponding to $\mathbf{M_i}$. $\mathbf{S_i}$ is found as $S_i = r\mathbf{M_i}^{T} \oplus C_i$ where r is the information bits in received word s = ($S_1\ S_2\ S_{2t}$) and the length of $\mathbf{S_i}$ is m. For the syndrome computation, the parity prediction can be implemented by checking that the following two equations take the same value.

$$r_{syn1} = S_1 \oplus S_2 \oplus ... \oplus S_{2t} \qquad (2)$$

$$r_{syn2} = C_1 \oplus C_2 \oplus ... \oplus C_{2t} \qquad (3)$$

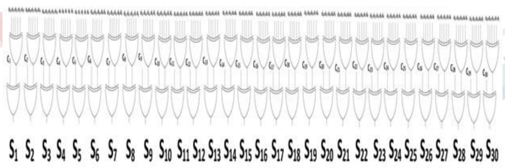

S_1 S_2 S_3 S_4 S_5 S_6 S_7 S_8 S_9 S_{10} S_{11} S_{12} S_{13} S_{14} S_{15} S_{16} S_{17} S_{18} S_{19} S_{20} S_{21} S_{22} S_{23} S_{24} S_{25} S_{26} S_{27} S_{28} S_{29} S_{30}

Fig. 4. Syndrome generator for ols code with k=25 and t=1

By using these fault detection signal simply detect the single stuck at fault of syndrome generator ($r_{syn1} \neq r_{syn2}$).

E. Orthogonal generator

The signal r_{oth1} is generated in the r_{oth1} generator; this consists of a MUX and two circuits, denoted as MAJ and EQ. The MAJ circuit is a 2t-input majority voter (and is also used in the error calculator of the OLS decoder). MAJ outputs a "0" if the number of "1"s is equal to that of "0". The EQ circuit outputs a "1" if and only if the number of "1" is equal to "0".

$$\gamma_{oth1} = \begin{cases} maj(\oplus c_1, \oplus c_2, ..., \oplus c_{2t}, 0) & (\sum_{i=1}^{2t} \oplus s_i \neq t) \\ (\oplus c_1) + (\oplus s_1) & (\sum_{i=1}^{2t} \oplus s_i = t) \end{cases}$$

where maj(…) is the majority of the input values; the following condition is then

$$maj(\oplus c_1, \oplus c_2, ..., \oplus c_{2t}, 0) = \begin{cases} 1 & (\sum_{i=1}^{2t} \oplus c_i > t) \\ 0 & (\sum_{i=1}^{2t} \oplus c_i \leq t) \end{cases}$$

$$\mathbf{H} = \left[\begin{array}{ccccccccccccccccccccccccc c}
1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0& \\
0&0&0&0&0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0& \\
0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0& \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&0&0&0& \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1& \\
1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0& \\
0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0& \\
0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0& \\
0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0& \\
0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1& \\
1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0& \\
0&1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&0&0& \\
0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&1&0&0&0&1&0& \\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&1& \\
0&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0& \mathbf{I}_{30} \\
1&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0& \\
0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&1&0& \\
0&0&1&0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&1& \\
0&0&0&1&0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&0& \\
0&0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&0&0&1&0&0&1&0&0&0& \\
1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&1&0& \\
0&1&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1& \\
0&0&1&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&1&0&1&0&0&0&0& \\
0&0&0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&1&0&1&0&0&0& \\
0&0&0&0&1&0&1&0&0&0&0&0&0&1&0&1&0&0&0&0&0&0&1&0&0& \\
1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1& \\
0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0& \\
0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0& \\
0&0&0&1&0&0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0& \\
0&0&0&0&1&1&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&1&0&
\end{array}\right]$$

Fig. 5. H matrix

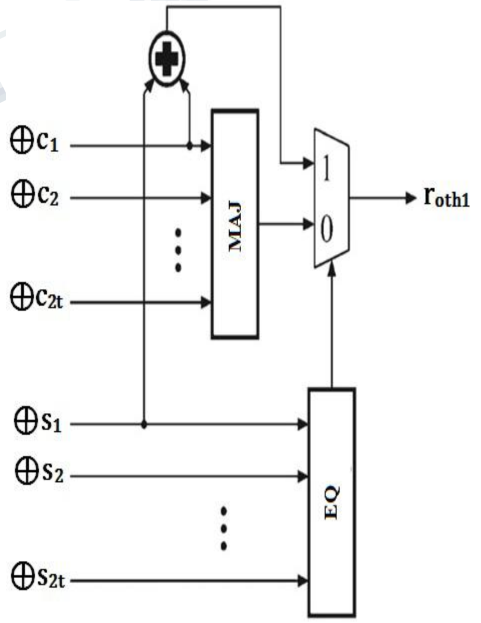


Fig. 6. r_{oth}1 generator

F. *Error pattern calculator and corrector*

OLS codes can be decoded using OSMLGD as follows. Let S_j be a vector that contains all i-th elements in a syndrome such that h_{ij} in the H matrix is 1. Suppose that a t-bit error occurs on a received word r. If the i-th bit in r is erroneous, then the values of at least (t + 1) bits in S_j are 1' s. If not, the values of at least t bits are 0's. Based on these conditions, errors can be corrected by flipping all received bits, such that at least (t + 1) bits in 0 are 1' s, i.e. by adding the majority of all values in 0 and a value a to all received bits.

Consider the H matrix shown in Fig. 5. suppose a code word with d = (01010-01010101010101010) is sent and an erroneous word r = (1011010101010101-101010) is received. In this example, a triple -bit error occurs on the 24th, 23rd and 22nd bit. Its syndrome is given by s= (100001110011100111001110011100). Initially, consider the decoding of the 24th bit in the received word, i.e. an erroneous bit. Since only $h_{1;24;}$ $h_{6;24}$; $h_{11;24}$; $h_{16;24}$; $h_{21;24}$ and $h_{26;24}$ are l's in the 24th column of H, then So = (111111). The majority of all values in so (i.e. 1, 1, 1, 1,1and 1) and a value 0 is l. Thus, the correct data is found as d_{24} = 0 by adding the majority 1 to the received bit r_{24} = l. Next, consider decoding of the 21st bit, i.e. a bit that is not erroneous. Since $h_{1,21}$, $h_{9,21}$, $h_{14,21}$, $h_{19,21}$, $h_{24,21}$ and $h_{29,21}$ are 1' s, then S6=(100000). The majority of 1, 0, 0, 0, 0, 0 and a value 0 is 0. Therefore, the correct data is given by adding the majority 0 to the received bit of the 21st bit, i.e. a bit that is not erroneous.

If a fault occurs in the error pattern calculator, then at most a bit in e flips. If there is no bit flipping in e, then the decoder outputs the correct v. If a bit flips in e, then the corresponding bit in v flips too; hence, r_{oth2} flips too. As the fault does not affect r_{oth1}, then the fault is detected. If a fault occurs on an XOR gate calculating u+v, it flips exactly one bit in v and then r_{oth2} flips too. Also in this case, r_{oth1} does not change and so, the fault is detected.

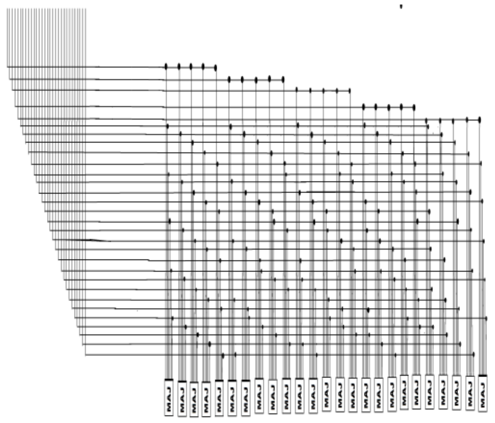


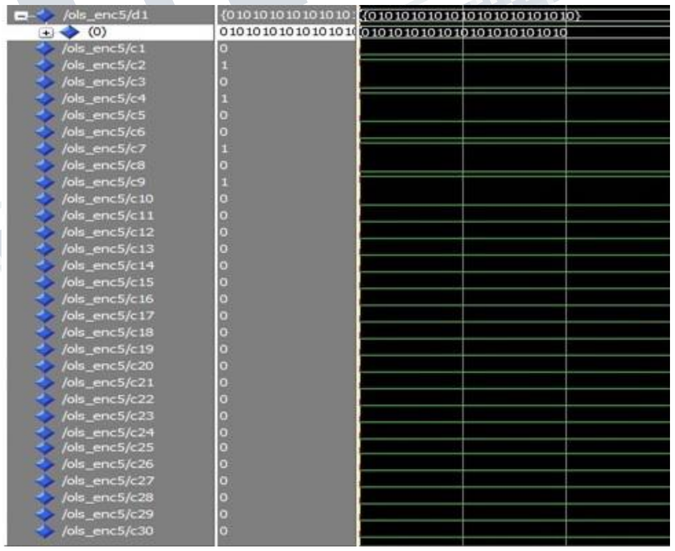
Fig. 7. Error pattern calculator for the (55,25) OLS code

IV. EXPIRIMENTAL RESULTS

A. *Simulation of OLS encoder*

In order to obtain triple bit error correction choose k=25 for input data bit .

d1 = 0101010101010101010101010



```
/ols_enc5/d1   {0101010101010101010101010}
  (0)          0101010101010101010101010
/ols_enc5/c1   0
/ols_enc5/c2   1
/ols_enc5/c3   0
/ols_enc5/c4   1
/ols_enc5/c5   0
/ols_enc5/c6   0
/ols_enc5/c7   1
/ols_enc5/c8   0
/ols_enc5/c9   1
/ols_enc5/c10  0
/ols_enc5/c11  0
/ols_enc5/c12  0
/ols_enc5/c13  0
/ols_enc5/c14  0
/ols_enc5/c15  0
/ols_enc5/c16  0
/ols_enc5/c17  0
/ols_enc5/c18  0
/ols_enc5/c19  0
/ols_enc5/c20  0
/ols_enc5/c21  0
/ols_enc5/c22  0
/ols_enc5/c23  0
/ols_enc5/c24  0
/ols_enc5/c25  0
/ols_enc5/c26  0
/ols_enc5/c27  0
/ols_enc5/c28  0
/ols_enc5/c29  0
/ols_enc5/c30  0
```

Fig. 8. Simulation result of OLS encoder

At the encoder when the information bit d is applied check bits are obtained. Checkbits obtained by the equation c = dM_i^T. By the proper use of H matrix of OLS code and XOR operations check bits can easily obtained. The simulated waveform is as shown in Fig. 8. Hence,

Check bits ,c = 010100101000000000000000000000

Codeword=0101010101010101010101010100101001010000000 000000000000000

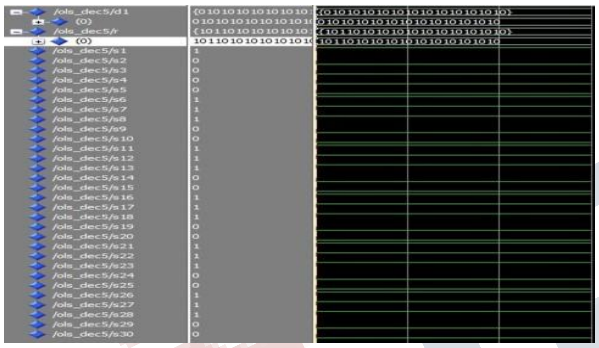
B. *Simulation of OLS syndrome generator*

To the syndrome generator section, check bits of the information bit from the memory and check bits of the encoder block are applied. Thus the syndrome bits s_i are obtained by the XOR operation of check bits obtained from the encoder block and that of the check bits obtained from the memory.

d_1 = 0101010101010101010101010

r = 10110101010101010101010

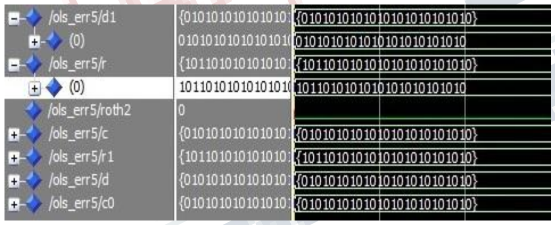
s = 100001110011100111001110011100



```
/ols_dec5/d1     {01010101010101010 {0101010101010101010101010}
  (0)            01010101010101010  0101010101010101010101010
/ols_dec5/r      {10110101010101010 {1011010101010101010101010}
  (0)            10110101010101010  1011010101010101010101010
/ols_dec5/s1     1
/ols_dec5/s2     0
/ols_dec5/s3     0
/ols_dec5/s4     0
/ols_dec5/s5     0
/ols_dec5/s6     1
/ols_dec5/s7     1
/ols_dec5/s8     1
/ols_dec5/s9     0
/ols_dec5/s10    0
/ols_dec5/s11    1
/ols_dec5/s12    1
/ols_dec5/s13    1
/ols_dec5/s14    0
/ols_dec5/s15    0
/ols_dec5/s16    1
/ols_dec5/s17    1
/ols_dec5/s18    1
/ols_dec5/s19    0
/ols_dec5/s20    0
/ols_dec5/s21    1
/ols_dec5/s22    1
/ols_dec5/s23    1
/ols_dec5/s24    0
/ols_dec5/s25    0
/ols_dec5/s26    1
/ols_dec5/s27    1
/ols_dec5/s28    1
/ols_dec5/s29    0
/ols_dec5/s30    0
```

Fig. 9. Simulation result of syndrome generator

C. *Simulation of OLS error correction*



```
/ols_err5/d1     {01010101010101010 {0101010101010101010101010}
  (0)            01010101010101010  0101010101010101010101010
/ols_err5/r      {10110101010101010 {1011010101010101010101010}
  (0)            10110101010101010  1011010101010101010101010
/ols_err5/roth2  0
/ols_err5/c      {01010101010101010 {0101010101010101010101010}
/ols_err5/r1     {10110101010101010 {1011010101010101010101010}
/ols_err5/d      {01010101010101010 {0101010101010101010101010}
/ols_err5/c0     {01010101010101010 {0101010101010101010101010}
```

Fig. 10. Simulation of OLS error correction

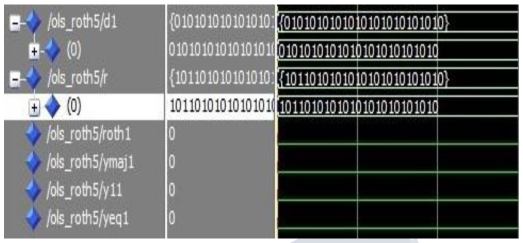
At the error corrector section the output information bit (output) is obtained as equivalent to the information bit (d) by proper usage of majority voter with syndrome bits and XOR operation of flipped information bit and that of the error pattern bits .

Output, C0 = 0101010101010101010101010

D. *Simulation of OLS roth1 generator*

To the roth1 generator, parity bit of check bits and syndrome bits are applied .Thus the fault detection signal roth1 is obtained by using majority voter and EQ block.

roth1 = 0



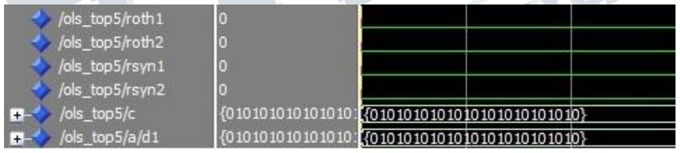
```
/ols_roth5/d1     {01010101010101010 {0101010101010101010101010}
  (0)             01010101010101010  0101010101010101010101010
/ols_roth5/r      {10110101010101010 {1011010101010101010101010}
  (0)             10110101010101010  1011010101010101010101010
/ols_roth5/roth1  0
/ols_roth5/ymaj1  0
/ols_roth5/y11    0
/ols_roth5/yeq1   0
```

Fig. 11. Simulation of OLS roth1 generator

E. *Simulation of fault detection modules*

Here, fault detection signal rsyn1 and rsyn2 are obtained by performing xor operations in all checkbits and all syndrome bits respectively.

r_{syn1} = 0 and r_{syn2} = 0.



```
/ols_top5/roth1   0
/ols_top5/roth2   0
/ols_top5/rsyn1   0
/ols_top5/rsyn2   0
/ols_top5/c       {01010101010101010 {0101010101010101010101010}
/ols_top5/a/d1    {01010101010101010 {0101010101010101010101010}
```

Fig. 12. Simulation of fault detection modules

V. CONCLUSION

CED technique takes the advantages of OLS codes to design a parity prediction scheme that could be efficiently implemented and detects any single stuck-at fault. OLS codes correct multiple bit and utilize high-speed parallel decoding by using one-step majority-logic decoding (OSMLGD). Self-checking property is the important features of OLS encoder and syndrome generator used in the CED technique, hence it strongly fault secured. By using additional fault detection signals, which is obtained from the check bits and syndrome bits lead to detect single stuck at faults. Fully separated majority voters are used for the error pattern calculation and with orthogonal generator together provide additional fault detection signals .Hence the CED scheme achieve 100 percentage fault coverage for the whole CED circuit, thus providing a very efficient and fully fault-tolerant implementation. In CED technique, the check bit size is closely depends on the data width. Therefore increase in data width increase the XOR gates, as a result it enlarges the system size, to reduce the size using a new PACC () architecture.

REFERENCES

[1] K. Nambe and F. Lombardi, ”Concurrent Error Detection of Binary and Nonbinary OLS Parallel Decoders”, IEEE Transactions on device and materials reliability, Vol. 14, No.1, March 2014.

[2] K. Namba and F. Lombardi, ”A novel scheme for concurrent error detection of OLS parallel decoders,” in Proc. IEEE Int. Symp. DFT VLSI Nano technol. Syst., New York, NY, USA, Oct. 2013.

[3] P. Reviriego, S. Pontarelli, and J. A. Maestro, ”Concurrent error detection for orthogonal Latin squares encoders and syndrome computation,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,vol. 21, no. 12 , Dec. 2013

[4] H. Y. Hsiao, D. C. Bossen, and R. T. Chien, ”Orthogonal Latin square codes,” IBM J. Res. Dev., vol. 14, no. 4, Jul. 1970.

[5] 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, Apr. 2009.

[6] G. C. Cardarilli, S. Pontarelli, M. Re, and A. Salsano, ”Concurrent error detection in ReedSolomon encoders and decoder,” in IEEE Trans. Very Large Scale Integr. (VLSI) Syst.,vol. 15, no. 7, Jul. 2007.

[7] J. E. Smith and G. Metze, ”Strongly fault secure logic networks,” IEEE Trans. Comput.,vol. C.27, no. 6, Jun. 1978.

[8] J. F. Wakerly, ”Error Detecting Codes, Self-Checking Circuits and Applications,”Amsterdam, The Netherlands: North Holland, 1978.