Low Complexity Cross Parity Codes for Multiple and Random Bit Error Correction

Mahesh Poolakkaparambil[1], Jimson Mathew[2], Abusaleh M. Jabir[1], and Saraju P. Mohanty[3]
[1]Department of Computer Science and Electronics, Oxford Brookes University, UK.
[2]Department of Computer Science, University of Bristol, UK.
[3]Department of Computer Science and Engineering, University of North Texas, USA.

***Abstract*— Error detection and correction which has been used in communication and memory design is becoming increasingly important in fault tolerant logic circuit design. As a result of the aggressive technology scaling, the current high-density integrated circuits are easily succumbed to faulty operations generated from many sources including stuck-at-faults, radiation induced faults, or malicious eavesdropper attacks. The currently used techniques like low-density parity-check (LDPC) and Hamming code based fault masking to mitigate bit flips in the digital circuits are either single bit error correcting or multiple error correctable with Bose-Choudhury-Hocquenghem (BCH) and Reed-solomon based methods with very large overheads. This paper introduce a novel cross code based method that can correct multiple errors with minimal compromise in error correction capability and area. The key idea of the novel method proposed in this paper is that *do not correct all the errors but minimize their probability being escaped*. Experimental results of the proposed methods show that the following: (1) area overhead is 101% for Hamming cross code and 106% for BCH cross code for a 90-bit finite field multiplier and (2) 150% for Hamming cross code and 170% for BCH cross codes for practically used 163-bit digit serial polynomial basis multiplier. Thus, the proposed methods are significantly efficient compared to Triple Modular Redundancy (TMR), LDPC, Hamming based methods in terms of area overhead and also the first attempted approach to a low complexity multiple error correctable digit serial multiplier to the best of the authors knowledge.**

Keywords: Polynomial Basis Bultiplier, Concurrent Error Detection, Single Error Correction, N-Modular Redundancy, Bose-Choudhury-Hocquenghem Code.

I. Introduction

The need for high density integration of CMOS circuits drove the semiconductor industry to a 22nm feature size that was supposed to be unthinkable previously. The chip vendors are able to integrate more and more devices on to the single die to reduce the cost of computation. One of the negative effects that is inevitable due to miniaturization of the integrated devices is their faulty operations when subjected to radiations, or stuck-at-faults [1]. Previously, the radiation interference towards the digital circuit operations was mainly due to decay of the packaging. Other probabilities of such situations are when the integrated circuits are put into space related operations where they are continuously in contact with cosmic rays. When cryptographic application specific integrated (ASIC) are used to delicately perform the authentication operations with much higher speed pulled the attraction of eves droppers who are interested in leaking information [2], [3]. These kind of attacks based on radiation bombardment are widely know as transient attacks.

Due to the random nature of these attacks, it is quite hard to model and mitigate such malicious eavesdropping. Due to the globalization of semiconductor industries and taking into the consideration that most of the ASIC designs are often manufactured in a third party vendor, possibilities of adding intruder circuits also called as hardware trojans that makes the circuit temporarily faulty to help the attacker to gather the hidden data or a security key that is been protected by the cryptogrphic chip [4].

The cryptographic processors are present in vast majority of day-to-day applications such as TV set-top boxes, bank ATM machines, credit cards, mobile communications and digital right management [5]. Most of these applications are meant to be executing one among the many cryptography algorithms at a much faster rate. They often hide information such as a secret key and some secret or important data. An attacker with the help of a prominent laboratory set up can subject such dedicated cryptography chips to radiations under controlled manner. Such transient attacks that are proved to be effective for revealing a secret key or the design intellectual property (IP) itself that can be easily cloned to make unauthorized version of the hardware. Various implementations of cryptographic processors designed for various cryptographic algorithms are widely researched and it reveals that the multiplier circuits are most complex unit of such a processor [6]. Hence they are undoubtedly stay as the main notion of attack for an attacker. Thus, it is evident that such functional arithmetic circuits or the digital circuits in general must be made attack tolerant to prevent malicious attacks or assure their fault free performance in a radiation prone environment.

The remainder of this paper is organized in the following manner. Section II explains the state of the art fault tolerant designs to mitigate the transient errors. The proposed novel cross code parity based multiple error correction is introduced in the Section III with a simple design example. This section also derives the closed form expressions of the design. Section IV discusses the experimental results and quantitative analysis. The conclusion and future extension of the proposed research in Section V.

II. Related Prior Research

There are few existing techniques presented in current literature to mitigate such transient errors. One approach for detecting erroneous calculations in finite field circuit is based on space redundancy. One example for such a space redundant scheme is Triple Modular Redundancy (TMR). In TMR, the actual functional block is replicated three times and the output is compared for correctness with a voter [7]. If two out of three circuits agree to one results to be correct, the voter stick to that as the final result of the circuit. The major drawback of TMR is that, the hardware overhead is always 200%. Another issue of TMR is that the reliability depends on the voter also the assumption is that error happens only in one functional block out of three. Another approach for error detection is based on time redundancy. This approach is also called as Concurrent Error Detection (CED) [8], [9]. In CED, an additional error monitoring block is hooked to the actual circuit that flags the occurrence of an error. Once the error flag is active, the functional block rolls back and recomputes. This induces a high delay penalty to the calculation that is unpleasant in many applications. In [10] authors present a method to protect memories against multiple bit upsets and to improve manufacturing yield. This method combines Hamming and Parity codes to assure the improvement of reliability and yield of the memory chips in the presence of high defects and multiple bit-upsets. However, the approach is specifically to memory design.

Some approaches are reported for double error detection and single error correction known shortly as SEC/DED schemes. SEC/DED are based on hamming or LDPC codes that can correct only single bit errors in the calculations [11], [12]. But analysis shows that transient error occurs at one critical node can cause multiple output errors due to large fan-out of the current large circuits. Other known but the least explored approaches are based on inherent properties of the functional blocks. One among such method is based on implication based error detection. Implication exist in any circuits and their violation can be used to detect error occupance [13].

It is evident from the above discussions that the research for handling multiple error detection and correction resulting form radiation induced transient errors or stuck-at-faults is still lacking. The current paper presents a novel transient error correcting technique based on cross parity scheme. The idea of introducing such a viable scheme is to make a trade of between area overhead and the fault tolerance capability. The key idea is to detect and correct as many errors as possible with less area overhead and less errors being escaped.

III. Proposed Cross Parity Codes for Multiple Error Correction

The very basic and easy method to do multiple bit error detection and correction is to use the well known forward error correction codes. The forward error correcting codes are generally meant to correct erroneous data in communication related applications. Hence the main challenge of applying these methods for fault tolerant circuit design is often complex and tricky. Also, the complexity associated with decoding the error information in order to perform the correction always consume comparatively higher area overhead though they give potential freedom in correcting fixed multiple bit errors [14]. In certain applications where area overhead is a criteria, one can design circuits with a trade of between the number of bit error correction and the area overhead. In this section hence we introduce a novel methodology for multiple bit error correction in logic circuits by using only the error detection features of the well known Hamming and BCH codes cross coupled with simple output parity prediction to save the area complexity contribution from the decoding circuitry of these codes. By doing so one can easily achieve a trade of between area and fault tolerance by simply avoiding the complex decoding implementations of the error correction codes. In this section we have explained our proposed method with two multiplier architecture mainly bit parallel multipliers and later with a 163-bit digit serial or world level multiplier that is used in Elliptic curve cryptography (ECC). The basic block diagram of the cross parity based code is as shown in Fig. 1. The major blocks are the functional block that need to be transient error hardened, cross parity predictor to detect error occurance and a simple error correction block.

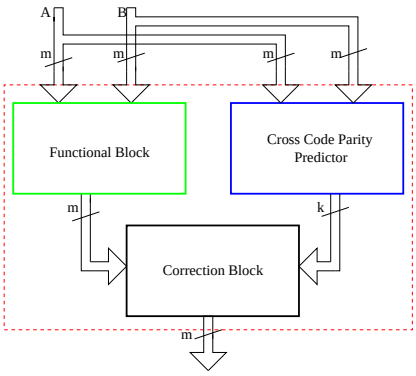

Fig. 1. General block diagram of cross parity based error correction

C0	C1	C2	C3	C4	Ham/BCH Parity -1
C5	C6	C7	C8	C9	Ham/BCH Parity -2
C10	C11	C12	C13	C14	Ham/BCH Parity -3
C15	C16	C17	C18	C19	Ham/BCH Parity -4
CP0	CP2	CP4	CP6	CP8	
CP1	CP3	CP5	CP7	CP9	

Fig. 2. Example of cross parity based error correction techniques

A. Multiple Error Detection

The structural model of such a cross parity technique is as shown in Fig. 2. Here we group the output bits of the

circuit in general uniform manner or in a random manner. Each rows of the grouped output bits are encoded either by using Hamming code or BCH codes, depending upon the number of error correction we need. Similarly the columns are encoded using simple output parity. Here we have considered simple Hamming code that can detect double errors and BCH code that will detect as much as 6 errors in each row. For better understanding of the row and column wise encoding we explain the procedure with an example circuit. We hence consider a 20-bit bit parallel finite field multiplier as example.

B. Error Detection Using Hamming Code Parity

Lets us assume the rows are encoded with Hamming codes then, each row is encoded with Ham(9,5) code. In other words, we require 4 bit parity to detect double error in one row. The 4 parity information for the first row is given by the following expression:

$$P1 = C0 \oplus C2 \oplus C4 \quad (1)$$
$$P2 = C1 \oplus C2 \oplus C3 \oplus C4 \quad (2)$$
$$P3 = C0 \oplus C3 \oplus C4 \quad (3)$$
$$P4 = C1 \oplus C2 \oplus C4 \quad (4)$$

Similarly, each row is encoded separately and treated as a different code word. The columns are encoded using the simple parity. Every two bits are protected by a column parity *CP* as shown in Fig. 2. The column parities of the first column are determined as shown in the equations below. Rest of the column parities are generated exactly the same was as that of *CP*0 to *CP*3, as represented in the following:

$$CP0 = C0 \oplus C10 \quad (5)$$
$$CP1 = C5 \oplus C15 \quad (6)$$
$$CP2 = C2 \oplus C12 \quad (7)$$
$$CP3 = C7 \oplus C17 \quad (8)$$

The set of equations from Eqn. (1) to Eqn. (4) helps to determine the occurrence the multiple error in each row. Similarly, the Eqn. (5) to Eqn. (8) computed for each column also predict the particular bit that being in error using the properties of cross parity. Some of the error patterns that our technique can correct are given in Fig. 3.

C. Multiple Error Correction

The above section explained how the errors are detected in both rows and columns. By just identifying the errors are not sufficient enough to correct them. Using classical error correction codes, they need a separate section often called as a decoder in order to identify the erroneous bit position and to correct it.

We eliminate the complex decoders using the fairly simple cross codes and use a simple *AND-XOR* logic to perform the correction. For example in Fig. 3 (a), suppose bits C0, C1, C5 and C6 are in error. One can easily predict erroneous bits C0 and C2 using the Hamming code of row1 and similarly the errors in C5 and C6 are detected by Hamming code of row2.

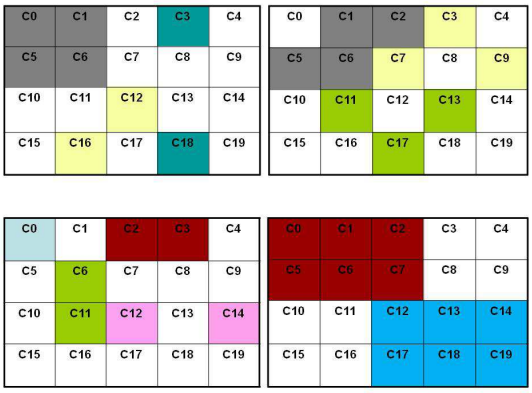

Fig. 3. Example patterns of hamminng based crossparity code based correction

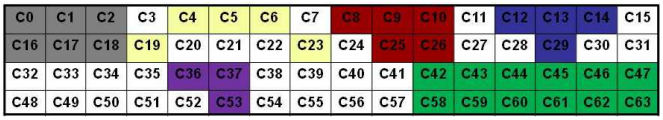

Fig. 4. Example patterns for hamming based cross parity code based correction for a 64-bit multiplier

But we just know 2 bits in row1 and row2 are in error but not their location. To find out which bits in each rows are in error can be easily done using the column parties as bit C0 is protected by CP0, bit C5 is protected by CP1. Similarly, the bits C2 and C7 are protected by CP2 and CP3. using the combination of both row and column parity, one can easily say which bits are in error.

We then use a simple *AND-XOR* logic to correct the detected errors. Fig. 4 shows some of the example patterns of the erroneous bits that can be corrected using the cross codes. A set of errors in Fig. 4 are denoted by same color indicator.

In similar way we can incorporate BCH codes for row error detection as it can detect more number of errors in each row. Fig. 4 shows example patterns of errors in a 64-bit finite field multiplier with BCH decoding in each row. But with a BCH(31,16) code, we can easily detect up to 6 errors per row that hence clearly increase the number of bits being corrected as compared to the simple hamming code.

D. Error Detection Using BCH Code Parity

The basic principle and design of the bit-parallel BCH code based multiple error detection is explained with an the same 20 bit multiplier as shown in Fig. 2. Let us consider a simple case of $BCH(15,5,7)$, where $n = 15$ and $k = 5$. In this fairly small example, we consider bit-parallel PB multiplier over GF(2^5). Let us consider the first row as a BCH code. Then, as $n = 15$ and $k = 5$, the following expression is obtained:

$$M(x) = C4x^4 + C3x^3 + C2x^2 + C1x + C0 \quad (9)$$
$$x^{n-k}M(x) = x^{n-k}(C4x^4 + C3x^3 + C2x^2 + C1x + C0)$$
$$= C4x^{14} + C3x^{13} + C2x^{12} + C1x^{11} + C0x^{10} \quad (10)$$

C0	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	C31
C32	C33	C34	C35	C36	C37	C38	C39	C40	C41	C42	C43	C44	C45	C46	C47
C48	C49	C50	C51	C52	C53	C54	C55	C56	C57	C58	C59	C60	C61	C62	C63

Fig. 5. Example patterns for BCH based cross parity code based correction for a 64-bit multiplier

The parity check bits are generated by the following:

$$P(x) = x^{n-k}M(x) \mod g(x). \quad (11)$$

Let us consider the generator polynomial to be $g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1$. Then parity expression for the first row for 6-bit detection will be,

$$P(x) = p_9x^9 + p_8x^8 + p_7x^7 + p_6x^6 + p_5x^5 + p_4x^4 + p_3x^3 + p_2x^2 + p_1x^1 + p_0 \quad (12)$$

We consider a 3 bit correcting BCH code hence it can detect 6 bit errors in a single code word. So to detect multiple errors in a 5 bit code, we need ten parity bits. The ten parity bits are given by, where, $p_0 = c_0+c_2+c_4$, $p_0 = d_0+d_2+d_4+e_0+e_1+e_2+e_3$, $p_1 = c_0+c_1+c_2+c_3+c_4$, $p_1 = d_0+d_1+d_2+d_3+d_4$, $p_2 = c_0+c_1+c_3$, $p_2 = d_0+d_1+d_3+e_1+e_2+e_3$, $p_3 = c_1+c_2+c_4$, $p_3 = d_1+d_2+d_4+e_0+e_2+e_3$, $p_4 = c_0+c_3+c_4$, $p_4 = d_0+d_3+d_4+e_0+e_2$, $p_5 = c_0+c_1+c_2$, $p_5 = d_0+d_1+d_2+e_2$, $p_6 = c_1+c_2+c_3$, $p_6 = d_1+d_2+d_3+e_0+e_3$, $p_7 = c_2+c_3+c_4$, $p_7 = d_2+d_3+d_4+e_1$, $p_8 = c_0+c_2+c_3$, $p_8 = d_0+d_2+d_3+e_0+e_1+e_3$, $p_9 = c_1+c_3+c_4$, $p_9 = d_0+d_3+d_4+e_0+e_2$.

Example pattern for BCH code based cross parity code is as shown in Fig. 5. Here we considered the 6 bit error detectable BCH code in each row. In each column we used simple parity codes as that in case of the hamming based scheme. Hence it can detect 2 errors in each column and 6 errors on each row. This means that our technique can correct up to certain 12 bit errors. Some of the pattern examples are highlighted in colors in Fig. 5. Similar colour indicate the multiple error in the same group.

E. Cross Codes Over Digit Serial Multipliers

In this subsection we have extended our proposed cross parity scheme over more practical multiplier such as a word level multiplier or a digit serial multiplier. For experimental cases we have considered a 163-bit digit serial multiplier that is the standard size multiplier for secure ECC operations set by NIST and FIPS. As far the contribution concerned, this is the first approach to best of the authors knowledge in synthesizing a 163-bit multiple error correctable digit serial approach. This is because the known error detectable and correctable techniques are better suited for bit parallel multipliers structures as they give very huge area overhead because of the parallel complex error detection, decoding and correction part that runs parallel to the actual multiplier logic.

In this section we have made and attempt to evaluate the complexity of our proposed scheme over such a digit serial multiplier architecture to better understand the space complexity. The digit serial multiplication is designed using a single accumulator multiplier architecture. The multiplication algorithm is as shown in Algorithm 1 [15].

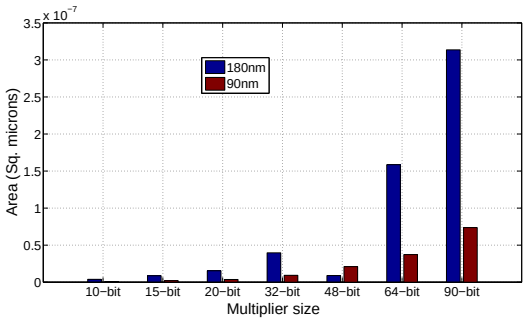

Fig. 6. Area for various multiplier sizes

Algorithm 1:
Input : $A(x) = \sum_{i=0}^{m-1} a_i.x^i$, $B(x) = \sum_{i=0}^{m-1} b_i.x^i$, P(x).
Output : $C(x) = A(x).B(x) mod P(x)$.
Step1: $C = 0$.
Step2: *for* $i = 0$ *to* $\lceil m/D \rceil$ - 1 do
Step3: $C = Bi.A + C$.
Step4: $A = A.\alpha^D$.
Step5: *end for*
Step6: return (C mod P(x))

IV. Experimental Results

This section explains the experimental results of our proposed cross parity based error correction method. The behavioral model of both Hamming and BCH based code are implemented using VHDL and checked for their functional correctness using Modelsim simulator. The schemes are checked and verified for bit parallel multiplier of various sizes such as 10, 15, 20, 32, 48, 64 and 90- bit multiplier structures. The designs are then synthesized using Synopsys design compiler. Variation in area, power of these designs are evaluated using both 180nm and 90nm TSMC technologies.

A. Area and Power Analysis of Proposed Implementation

Fig. 6 shows the space consumptions of bit parallel multipliers of various sizes. Fig. 7 and Fig. 8 reports the area of error correcting blocks (includes the parity generator) in both 180 and 90nm technology. It is very much obvious from Fig. 7 and Fig. 8 that the space consumption of BCH based technique is only slightly higher than the Hamming based cross code. This is because of the fact that the area intensive decoder sections of both the codes are replaced by simple cross parity based error detector and corrector as mentioned in Section III.

The area overhead of the proposed cross parity based method is depicted in Table I. It is observed for the experimental analysis that the area overhead for both BCH and Hamming based schemes are remarkably close. The area overhead for a very simple 10-bit multiplier is only 142%. As the multiplier size grows the percentage area overhead due to the parity generation circuit and the correction logic is getting smaller

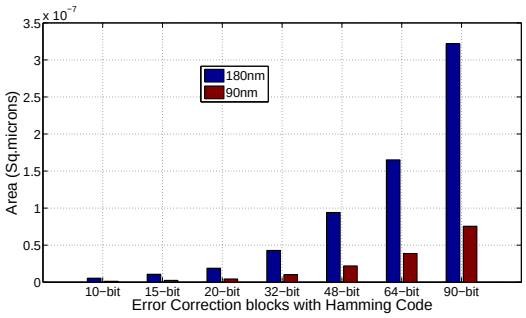

Fig. 7. Error detection and correction block area of hamming cross parity code

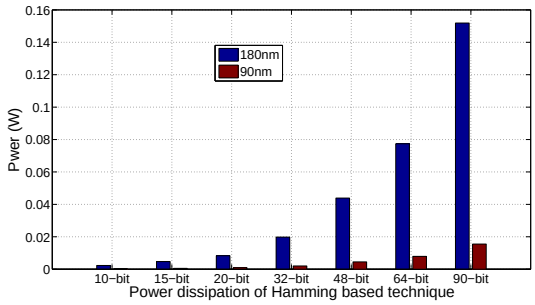

Fig. 9. Power consumption of hamming code based scheme

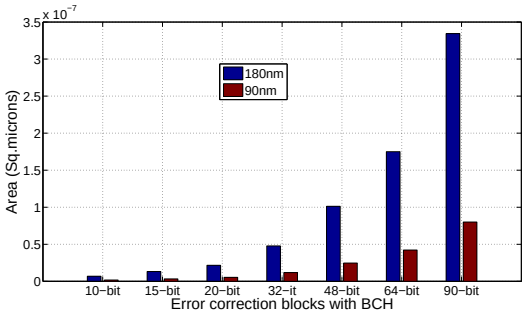

Fig. 8. Error detection and correction block area of BCH cross parity code

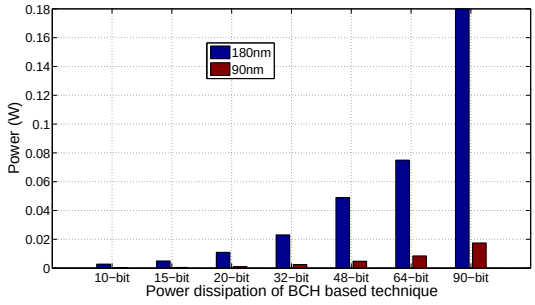

Fig. 10. Power consumption of BCH based cross parity scheme

and eventually for a 90-bit multiplier that can correct multiple error is just only 101%. This is quite smaller as compared to the classic multiple error correction schemes based on only single error correction code. Even though the design is not entirely dealing with all error patterns, it is very unlikely that some pattern that can occur outside the scope of the proposed scheme. It is because of the fact that the probability of the radiation particle interference that can cause multiple bit flip is for example only 1 in 1 million clock cycle. Hence our proposed scheme can provide excellent error masking capability with area overhead as less as 101%.

TABLE I
AREA OVERHEAD COMPARISION OF VARIOUS MULTIPLIER SIZES

No. of bits	Hamming	BCH
10	142%	160%
15	123%	152%
20	121%	140%
32	108%	120%
48	105%	116%
64	104%	114%
90	101%	106%

Table II compares our cross parity code approach with other error correction schemes available in open literature. For a fair comparison, we have used the 32-bit multiplier. It shows that our method can correct more number of errors with lesser area overhead as compared to the other well known designs.

The power dissipation of our proposed scheme has been analysed. Fig. 9 and Fig. 10 compares the power consumption of both hamming and BCH based designs. As they have comparable area overhead, the power dissipation is roughly close to each other as well.

B. Experimental Analysis of 163-bit Digit Serial Multiplier

It is known that the bit parallel multipliers are mainly used in application that needs high speed smaller multiplier size. For much complex computation the classic bit parallel multipliers can not be used as the area complexity simple explodes as the multiplier size increases. Hence we use digit serial multipliers that brings up a trade of between the area and the performance. Hence our scheme has been verified over more realistic and practically applicable 163-bit digit serial multiplier. The area overhead of the 163-bit digit serial multiplier with both hamming and BCH code based cross parity scheme have been analyzed. Fig. 11 shows the bar chart of the area overhead for the 163 bit multiplier for different digit sizes. We have considered digit sizes of 2, 4, and 6. The overhead plot clearly indicates that the space overhead significantly reduces for higher digit size of the digit serial multiplier.

To complete the design flow, the proposed architecture is implemented using RTL synthesizable VHDL code. also the design is synthesized with 0.18 μm (1.8V supply voltage) technology using Synopsys design compiler tools. the back end process, place and route, is done using cadence Encounter^{TM} tool set. The final layout of the 163-bit multiplier design is shown in Fig. 12. The layout area , using 6 metal layer, is 1.84 mm^2. We have generated the physical layout of the 163-bit digit serial multiplier design with the hamming code using

TABLE II

COMPARISON WITH OTHER APPROACHES FOR 32-BIT MULTIPLIER

Property	Masoleh et al. 2004 [16]	Mathew et al. 2008 [12]	BCH [14]	Cross Parity (Ham)	Cross Parity (BCH)
#errors correction	single	single	3 Errors	up to 6 Errors	up to 12 Errors
Coding technique	Hamming	LDPC	Classic BCH	Hamming + Simple Parity	BCH + Simple Parity
Overhead	>100%	>100%	150.4%	108%	120.4%

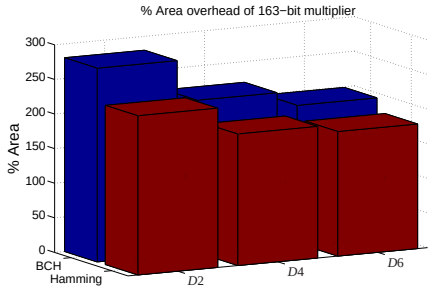

Fig. 11. Area overhead of error detection and correction block for 163 bit multiplier

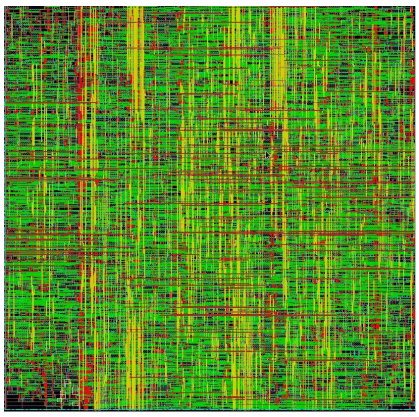

Fig. 12. Layout of the 163-bit multiplier with Cross Parity Code Correction Block

Cadence SoC Encounter. Fig. 12 shows the generated layout of the digit serial multiplier.

V. CONCLUSION AND FUTURE EXTENSION

This paper proposed a novel multiple error correction scheme based on cross parity codes in order to address the temporal fault in circuits mainly occurring due to the radiation interference. The authors have explained the proposed scheme by taking finite field multipliers as example circuits where radiation interference and attacks based on controlled radiation have been reported. With the new proposed scheme authors have come up with addressing the issue of multiple error correction capability and area overhead. With the cross parity codes, the complex decoding blocks have been replaced and the correction is done using the inherent error detection capabilities of the codes itself. With the new approach, the 90-bit multiple error correctable multiplier has just 101% area overhead. The authors have also analyzed their technique in practically applied digit serial multiplier circuit and the area overhead is 170% for a 163-bit digit serial multiplier with digit size 6. To the best of authors knowledge, this is the first approach for error correction in a digit serial multiplier that can correct more than 3 errors. The future extension of this work includes analyzing the fault coverage of the circuit also extend the method for a generic error correctable deign for a higher multiplier sizes.

REFERENCES

[1] Y. Jin and Y. Makris, "Hardware Torjans in Wireless Cryptographic ICs," *IEEE Design & Test Computers*, vol. 27, no. 1, pp. 26–35, January/February 2010.

[2] A. R. Masoleh and M. A. Hasan, "Fault Detection Architectures for Field Multiplication Using Polynomial Bases," *IEEE Trans. Computers*, vol. 55, no. 9, pp. 1089–1103, 2006.

[3] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the Importance of Eliminating Errors in Cryptographic Computations," *J. Cryptology*, vol. 14, no. 2, pp. 101–119, 2001.

[4] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware Trojan: Threats and emerging solutions," *IEEE Proceedings on High Level Design Validation and Test Workshop*, pp. 166–171, 2009.

[5] S. P. Mohanty, "A Secure Digital Camera Architecture for Integrated Real-Time Digital Rights Management," *Journal of Systems Architecture - Embedded Systems Design*, vol. 55, no. 10-12, pp. 468–480, 2009.

[6] C. R. Moratelli, E. Cota, and M. S. Lubaszewski, "A Cryptography Core Tolerant to DFA Fault Attacks," *Journal Integrated Circuits and Systems*, vol. 2, no. 1, pp. 14–21, 2007.

[7] J. f. Wakerly, "Microcomputer reliability improvement using triple-modular redundancy," *IEEE Proceedings*, vol. 64, pp. 889–895, 1976.

[8] K. Wu, R. Karri, G. Kuznetsov, and M. Goessel, "Low Cost Concurrent Error Detection for the Advanced Encryption Standard," in *Proceedings of the International Test Conference*, 2004, pp. 1242–1248.

[9] O. Keren, "One to Many: Context Oriented Code for Concurrent Error Detection," *Journal of Electronic Testing*, vol. 26, no. 3, pp. 337–353, 2010.

[10] C. Argyrides, D. K. . Pradhan, and T. Kocak, "Matrix Codes for Reliable and Cost Efficient Memory Chips," *IEEE Trans. on VLSI*, vol. 19, no. 3, pp. 420–428, 2011.

[11] J. Mathew, A. M. Jabir, H. Rahaman, and D. K. Pradhan, "Single Error Correctable Bit Parallel Multipliers Over GF(2^m)," *IET Comput. Digit. Tech.*, vol. 3, no. 3, pp. 281–288, March 2008.

[12] J. Mathew, J. Singh, A. M. Jabir, M. Hosseinabady, and D. K. Pradhan, "Fault Tolerant Bit Parallel Finite Field Multipliers using LDPC Codes," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 2008, pp. 1684–1687.

[13] N. Alves, "State of the Art Techniques for Detecting Transient Errors in Electrical Circuits," *IEEE Potentials*, pp. 30–35, 2011.

[14] M. Poolakkaparambil, J. Mathew, A. M. Jabir, D. K. Pradhan, and S. P. Mohanty, "BCH Code Based Multiple Bit Error Correction in Finite Field Multiplier Circuits," in *Proceedings of the 12th IEEE International Symposium on Quality Electronic Design*, 2011, pp. 615–620.

[15] S. kumar, T. Wollinger, and c. Paar, "Optimum Digit Serial GF(2m) Multipliers for Curve Based Cryptography," *IEEE Trans. Computers*, vol. 55, no. 10, pp. 1306–1311, 2006.

[16] A. Reyhani-Masoleh and M. A. Hasan, "Low Complexity Bit Parallel Architectures for Polynomial Basis Multiplication over GF(2^m)," *IEEE Trans. Computers*, vol. 53, no. 8, pp. 945–959, 2004.