Error Control Methods for All-One Polynomial Based Galois Field Multiplier

Niviya. S
UG Scholar
Department of Electronics and Communication Engineering, Avinashilingam University for Women Coimbatore, Tamil Nadu, India.

Poorani. A
UG Scholar
Department of Electronics and Communication Engineering, Avinashilingam University for Women Coimbatore, Tamil Nadu, India.

Induja. R
UG Scholar
Department of Electronics and Communication Engineering, Avinashilingam University for Women Coimbatore, Tamil Nadu, India.

Sargunam. B
Assistant professor
Department of Electronics and Communication Engineering, Avinashilingam University for Women Coimbatore, Tamil Nadu, India.

***Abstract*— Finite field multiplier has wide applications in elliptic curve cryptography and error control coding systems. Error detection and correction is an effective way to mitigate attacks in cryptographic hardware. Security threats arising from injected soft (transient) errors into a cryptographic circuit can expose the secret information. In this paper, Triple Modular Redundancy (TMR) techniques and parity prediction technique based on Hamming for error detection and correction are used in all one polynomial based finite field multiplier. In triple modular redundancy technique there are three systems which perform same process and the results are processed by a majority voting system to produce a single output. Parity Prediction technique are used to detect and correct single error based on the Hamming principle. TMR techniques and parity prediction technique are compared based on area, delay and power consumption. Parity prediction technique is better than TMR in terms of area and power consumption.**

***Keywords*— finite field multipliers, irreducible AOP, error detection and correction, triple modular redundancy, parity prediction.**

IJERT

I. INTRODUCTION

The Finite field finds its main application in the field of computer algebra, error control coding theory, public key cryptosystem and elliptic curve cryptography [1] [2]. Among all the finite field arithmetic operations like addition, multiplication, exponentiation and multiplicative inversion, the multiplication occupies an important and irreplaceable part since exponentiation and multiplicative inversion can be carried out by iterative multiplication.

One important factor that affects the hardware efficiency in finite fields is the basis chosen to represent a field element. There are a few bases available for the finite field such as polynomial/standard/canonical basis, normal basis and dual basis [3]. Multipliers based on some popular polynomials such as All-One Polynomials (AOP) and trinomials have low circuit complexity [4-7]. The irreducible all one polynomial seems to be more efficient for both hardware and software implementations [8].

Fault based cryptanalysis attack is a recently developed cryptanalysis approach where faults are injected into cryptosystems. Inducing faults into the computation of elliptic curve scalar multiplication easily enables recovery of the secret key. There are many error detection approaches developed for private key cryptosystems and public key cryptosystems to check the output values. The error detection methods for polynomial basis multipliers employ parity checking.

A number of error correction schemes have been proposed in the literature [9-13] for error detection. The Hamming distance between the code words must be greater than one; otherwise it cannot be used to detect an error. J. Mathew and A. M. Jabir proposed a bit parallel GF multiplier [14] for single error detection and correction. The principal techniques that are being used for single error correction are: (1) error detection and retry, (2) error masking, and (3) using encoded operands.

In this paper we have used Triple Modular Redundancy (TMR) and Parity prediction (Hamming) techniques for error detection and error correction. In TMR there are three systems which perform the multiplication process and the result of the multiplication is given to a majority voting system to produce a single output. Since the output is processed by a majority voting circuit, even if one of the system fails there will be no errors in the output. In parity prediction, a parallel circuit (Hamming block) along with GF multiplier is used for error control. Hamming methodology corrects single error.

The rest of this paper is organized as follows. The algorithm for finite field multiplication over GF (2^m) based on AOP is derived in Section II. In Section III, the error control using triple modular redundancy technique is discussed. In Section IV, the error detection and correction using parity prediction (Hamming) technique is discussed. In section V the triple modular redundancy and parity prediction techniques for AOP based GF multipliers are compared based on area, power consumption and delay. Finally the conclusion is given in Section VI.

II. ALGORITHM FOR GF MULTIPLIER

A polynomial of the form $f(x) = f_0 + f_1x + f_2x^2 + f_3x^3 + f_4x^4 + \ldots\ldots\ldots + f_m\ x^m$ over $GF(2^m)$ is called all one polynomial when $f_i = 1$ for $i = 0,1,2,\ldots\ldots,m$.

For an all one polynomial to be irreducible $(m+1)$ should be a prime number and 2 is the primitive modulo of $(m+1)$. The values of *'m'* satisfying the above conditions for irreducibility ($m \le 100$) are 2, 4, 10, 12, 18, 28, 36, 52, 58, 60, 66, 82 and 100. The algorithm for AOP based GF multiplier is referred from [4] and [5].

Let 'α' be the root of $f(x)$, then $f(\alpha) = 1 + \alpha + \alpha^2 + \alpha^3 + \alpha^4 + \ldots + \alpha^m = 0$. Therefore,

$$f(\alpha) + \alpha f(\alpha) = (1 + \alpha + \alpha^2 + \ldots + \alpha^{m-1} + \alpha^m) + \alpha(1 + \alpha + \alpha^2 + \alpha^3 + \alpha^4 + \ldots + \alpha^{m-1} + \alpha^m)$$
$$\alpha^{m+1} = 1 \quad (1)$$

Since *'α'* is the root of an irreducible all one polynomial of degree *'m'*. Any element 'y' in the Galois field $GF(2^m)$ can be represented as $y = y_0 + \alpha y_1 + \alpha^2 y_2 + \ldots\ldots + \alpha^{m-1}y_{m-1}$ where $y_i \in GF(2)$ for $(0 \le i \le m-1)$ and $\{1, \alpha, \alpha^2, \ldots\ldots\ \alpha^{m-1}\}$ forms the polynomial basis for degree $m-1$.

Any element 'Y' in the Galois field $GF(2^m)$ can also be represented as $Y = Y_0 + \alpha Y_1 + \alpha^2 Y_2 + \ldots + \alpha^{m-1} Y_{m-1} + \alpha^m Y_m$ where $Y_i \in GF(2)$ for $(0 \le i \le m)$ and $\{1, \alpha, \alpha^2, \ldots\ldots\ \alpha^{m-1}, \alpha^m\}$ forms the extended polynomial basis for degree m.

The elements in Galois field a, b and c where c is the product of elements a and b, are represented in polynomial basis as,

$$a = a_0 + a_1\alpha + a_2\alpha^2 + \ldots + a_{m-1}\alpha^{m-1} \quad (2)$$
$$b = b_0 + b_1\alpha + b_2\alpha^2 + \ldots + b_{m-1}\alpha^{m-1} \quad (3)$$

$$c = ab \ (\mathrm{mod}\ f(\alpha)) \quad (4)$$
$$c = c_0 + c_1\alpha + c_2\alpha^2 + \ldots + c_{m-1}\alpha^{m-1} \quad (5)$$

where $a_i \in GF(2)$ for $(0 \le i \le m-1)$, $b_j \in GF(2)$ for $(0 \le j \le m-1)$ and $c_k \in GF(2)$ for $(0 \le k \le m-1)$ and $\{1, \alpha, \alpha^2, \ldots \alpha^{m-1}\}$ forms the canonical basis. The coefficients of c, c_k can be calculated only by considering the extended polynomial basis representation. The algorithm is referred to [4].

The elements A, B are represented in extended polynomial basis as,

$$A = A_0 + A_1\alpha + A_2\alpha^2 + A_3\alpha^3 + \ldots + A_m\alpha^m \quad (6)$$
$$B = B_0 + B_1\alpha + B_2\alpha^2 + B_3\alpha^3 + \ldots + B_m\alpha^m \quad (7)$$

where $A_i \in GF(2)$ for $(0 \le i \le m)$ and $B_j \in GF(2)$ for $(0 \le j \le m)$.

If C is the product of two elements A and B, then

$$C = AB \bmod [f(\alpha)] \quad (8)$$
$$C = (A_0 + A_1\alpha + \ldots + A_m\alpha^m)(B_0 + B_1\alpha + \ldots + B_m\alpha^m)\ [\mathrm{mod}\ f(\alpha)]$$

$$C = C_0 + C_1\alpha + C_2\alpha^2 + \ldots + C_m\alpha^m \quad (9)$$

where $C_k \in GF(2)$ for $(0 \le k \le m)$

Therefore, the coefficient of 'C' is given by,

$$C_k = \textstyle\sum_{i+j=k(\mathrm{mod}\ m+1)} A_i\, B_j\ (\mathrm{mod}\ 2)$$

According to the theorem quoted in [4], let $y(x) = y_0 + \alpha y_1 + \alpha^2 y_2 + \ldots + \alpha^{m-1} y_{m-1}$ and $Y(x) = Y_0 + \alpha Y_1 + \alpha^2 Y_2 + \ldots + \alpha^{m-1} Y_{m-1} + \alpha^m Y_m$ be the polynomials over $GF(2^{m-1})$ and $GF(2^m)$. If $Y(x)$ and $y(x)$ satisfy the condition, $Y(x) = y(x)\ (\mathrm{mod}\ p(x))$, where $p(x)$ is an AOP of degree 'm', then the coefficients of $y(x)$ is given by $y_k = Y_k \oplus Y_m\ (\mathrm{mod}\ 2)$ for $(0 \le k \le m-1)$

Thus, the coefficients of c [$= ab(\mathrm{mod}\ f(\alpha)$], $c_k = C_k \oplus C_m$ for $(0 \le k \le m-1)$. The coefficient of c gives the final polynomial equation of GF multiplier output using AOP. The above algorithm is followed for the AOP based Galois field multiplier.

III. TRIPLE MODULAR REDUNDANCY

Triple Module Redundancy (TMR) is a very common fault tolerance technique. The principle of TMR is to triplicate the hardware circuit and a voter is added to the outputs. In this paper we are using TMR for standard AOP based GF multiplier. All standard AOP based GF multiplier are tripled and its respective outputs are connected to a voter. The voter will select the output of the majority of the modules. So, if one module fails, the error will not be reflected in the voter output.

A triple modular redundancy circuit with one voter for an all one polynomial based Galois field multiplier is shown in Fig.1.

IJERT

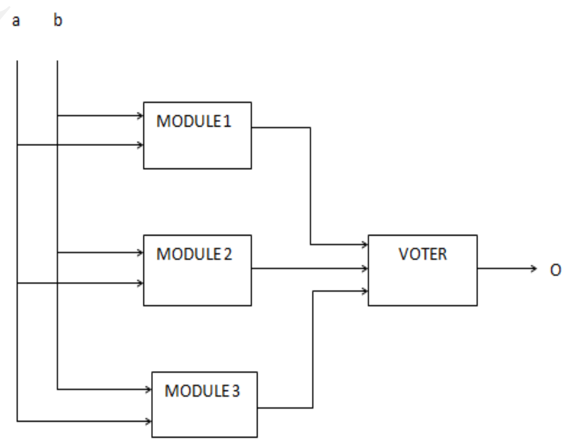

Fig.1 TMR with single voter

'a' and 'b' are the two inputs (each of four bits) given to the modules 1, 2 and 3. Each module is made up of All One Polynomial (AOP) based GF multiplier circuit. The output of each module is given simultaneously to the voter circuit.

Voter circuit does the majority gate function. It compares the output of the three modules and provides the majority output. In simple words, it performs the logical operation similar to the carry function derived in a full adder circuit. The Fig.2 explains the functionality of voter circuit.

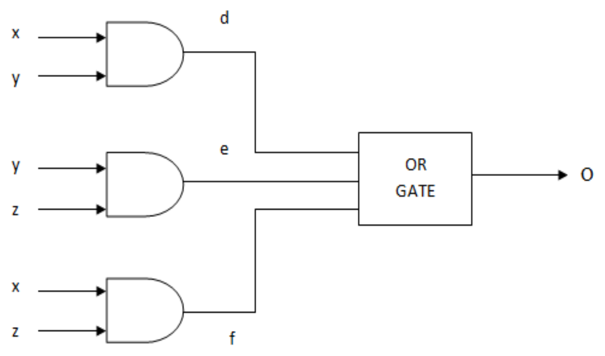

Fig.2 Block diagram of single voter circuit

The voter performing carry logical function compares the output of each GF multiplier on a bit-by-bit basis and is given by,

$$o = x.y + y.z + x.z$$

where o is the output of voter and x, y and z are the outputs of modules 1, 2 and 3. Thus even if fault occurs in any one module, there occurs no error in final output.

Although in some cases the majority gate could fail or the systems with only one voter could itself fail. Due to this the complete system will fail. In order to mask this, voters can be tripled. Three voters are used, one for each copy of the next stage of TMR logic. In such systems there is no single point of failure.

The Fig.3 explains TMR with four voters. In order to avoid occurrence of error in a single voter circuit from the previous concept, three more voter circuits are added to it. Here each voter (V_1, V_2 and V_3) compares the AOP multiplier output provided by three modules. The final fourth voter circuit (V_a) chooses the majority voter output from V_1, V_2 and V_3.

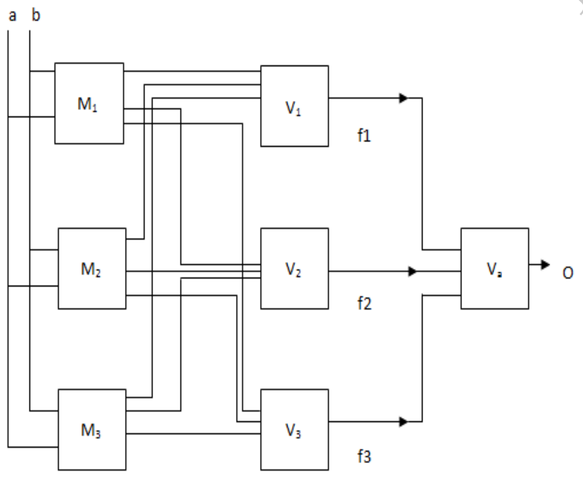

Fig.3 TMR with four voters

The Fig.4 explains TMR with seven voter circuit. To reduce the error occurrence caused by the voter circuit from the previous concept, three more voter circuits are added to it. The voter circuit V_1, V_2 and V_3 gives the majority output of the modules. The majority output of voters V_1, V_2 and V_3 is chosen by the voters V_a, V_b and V_c. The final voter V_f gives the majority output of V_a, V_b and V_c. Thus, even if an error occurs in any one of the voter, the output masks it and provides the correct output.

The advantage of TMR design is that it provides fault-tolerance without the need for separate detection and recovery functions. TMR approach provides real-time masking of the faulty-module outputs. Also TMR is faster than the code-based techniques since the voting speed is independent of the information bit length because the voting is always performed on three bits.

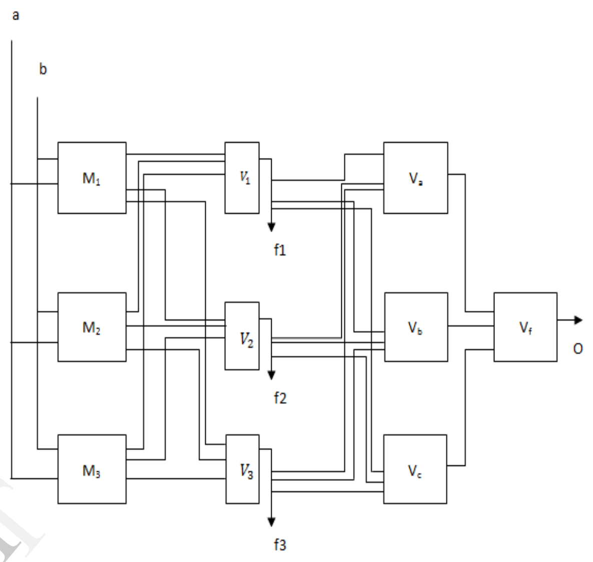

Fig.4 TMR with seven voters

IJERT

The traditional TMR voter masks the faults affecting only one module and the faulty module cannot be recovered in a traditional TMR system, as the system cannot identify the faulty module.

IV. PARITY PREDICTION

In this section the error detection and correction are done in a parallel manner by parity prediction technique where the parity bits are generated based on the Hamming principle. The Hamming codes are simplest of group of codes known as Linear Block codes. In Hamming, number of parity bits increases logarithmically with number of output bits which serves as the main advantage.

The principle difference between the Hamming codes which are used for memory application and our approach is that, instead of encoder we have parity prediction circuit and its size depends upon the number of input bits.

Parity prediction technique is one of the ways to detect errors. Since the parity prediction circuit runs in parallel with the multiplier, only the decoding and correction logic provides the delay penalty. The scheme is given in the Fig.5.

In the Galois Field multiplier block the standard AOP based GF multiplier is implemented. For a 4-bit input (A and B), the GF multiplier output is of 4 bits c_4, c_3, c_2 and c_1. In the multiple parity prediction circuit block polynomial multiplication takes place i.e. the two 4 bit multiplier inputs are multiplied and the resultant 7 bits are stored as d_0, d_1, d_2, d_3, e_0, e_1 and e_2.

Parity bits p_0, p_1, p_2 are calculated from the standard multiplier output.

$p_s = d_0$ xor d_1 xor d_2 xor d_3 xor e_0 xor e_1 xor e_2

$p_0 = p_s$ xor d_2 xor e_0

$p_1 = p_s$ xor d_1 xor e_0 xor e_2

$p_2 = p_s$ xor d_0 xor e_1

The parity bits p_0', p_1', p_2' are calculated from the GF multiplier output.

$p_0' = c_3$ xor c_4 xor c_1

$p_1' = c_4$ xor c_2 xor c_1

$p_2' = c_4$ xor c_3 xor c_2

These two sets of parity bits are XORed to generate syndrome bits s_0, s_1 and s_2.

If the syndrome bits are equal to zero then there is no error in the multiplication. If it is other than zero then the value denotes the position of the error.

The output of the Hamming block is formed from the syndrome bits.

h_1= (not s_0) and s_1 and (not s_2)

$h_2 = s_0$ and (not s_1) and s_2

$h_3 = s_0$ and (not s_1) and s_2

$h_4 = s_0$ and s_1 and s_2

The standard GF multiplier output and Hamming output are XORed to get the correct GF multiplication output.

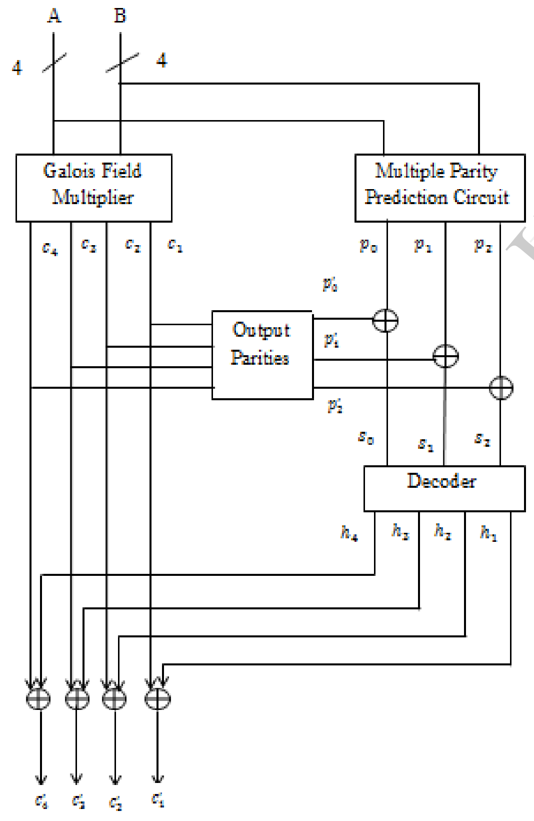

Fig.5. Galois Field multiplier with parity prediction technique (Hamming)

V. Results and Discussion

The standard AOP based GF multiplier, Error control for AOP based GF multiplier using TMR techniques and Error detection and correction for AOP based Galois Field multiplier using parity prediction technique (Hamming) are simulated to check the functionality. All the five GF multipliers (standard AOP, three TMR and parity prediction) are synthesized in FPGA and the results are tabulated in TABLE I

TABLE I COMPARISON OF ERROR CONTROL TECHNIQUES FOR AOP BASED GF MULTIPLIER

PARAMETERS	Standard AOP based GF multiplier	TMR with one voter	TMR with four voters	TMR with seven voters	Parity prediction (Hamming)
Gate count	84	234	732	2,226	207
Power(mW)	96	101	116	162	99
Delay(ns)	12.3	12.326	13.807	18.755	16.183

From the comparative results of gate count shown in Fig.6, it is seen that while using TMR techniques, a standard AOP based GF multiplier needs an additional increase of more than 175% of its actual area (i.e., area required by standard AOP multiplier), whereas parity prediction technique needs an area overhead of nearly 150%. Hence parity prediction requires less area than TMR.

IJERT

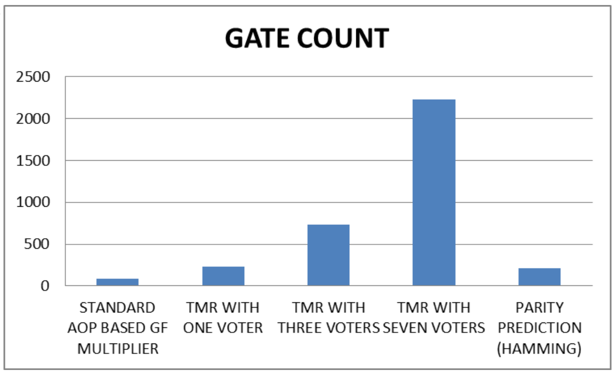

Fig. 6 Comparison of gate count

From the comparative results of power consumption shown in Fig.7, it is seen that while using TMR techniques for a standard AOP based GF multiplier, an additional increase of more than 5% of its actual power (i.e., power consumed by standard AOP multiplier)is needed, while parity prediction technique consumes an additional power of nearly 3% of its actual power. Hence parity prediction technique requires less power than TMR.

From the comparative results of delay shown in Fig.8, it is seen that when standard AOP based GF multiplier uses parity prediction circuit there is an additional delay of 31%. While using TMR with less number of voters there is an increase in delay of only 2-12%. An increase in the number of voters in TMR increases the delay abruptly.

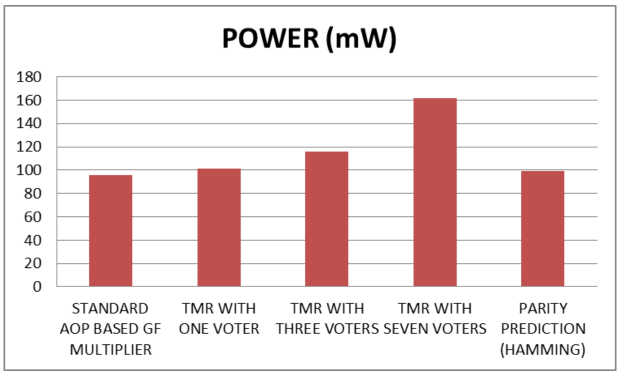

Fig. 7 Comparison of power consumption

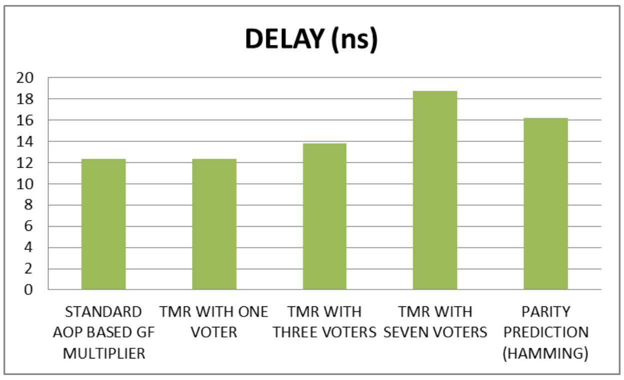

Fig.8 Comparison of time delay

VI. CONCLUSION

Thus all one polynomial based Galois field multiplier, error control using TMR techniques and parity prediction technique for GF multiplier are implemented using Xilinx ISE 8.1 and the results are compared. The comparative results of gate count proves that TMR techniques require an area overhead of more than 175%, while parity prediction technique requires an area overhead of only 150%. Similarly, power consumed by AOP based GF multiplier using TMR techniques are larger than the one with parity prediction technique. Hence, parity prediction technique (Hamming) is more efficient than Triple Modular Redundancy techniques for controlling error in AOP based GF multiplier considering the gate count and power as the important factors. TMR techniques lead to an increase in delay of 2-12% while parity prediction technique leads to an increase in delay of 31%. This can be taken up as a future work and the delay for parity prediction technique can be reduced.

IJERT

REFERENCES

[1] M. Ciet, J. J. Quisquater, and F. Sica, "A secure family of composite finite fields suitable for fast implementation of elliptic curve cryptography," *in Proc. 2nd Int. Conf. Cryptology.*, Vol. 2247, pp. 108–116, December 2001.

[2] H. Fan and M. A. Hasan, "Relationship between (2^m) montgomery and shifted polynomial basis multiplication algorithms," *IEEE Trans. Computers*, vol. 55, no. 9, pp. 1202–1206, Sep. 2006.

[3] E. D. Mastrovito, "VLSI Architectures for Computations in Galois Fields," *Linkoping studies in science and technology dissertation*, no. 242, 1991.

[4] T. Itoh and S. Tsujii, "Structure of Parallel Multipliers for a Class of Fields GF (2^m)," *in Proc. Inform. Computation*, Vol. 83, no. 1, pp. 21–40, October 1989.

[5] C.-Y. Lee, E.-H. Lu, and J.-Y. Lee, "Bit-Parallel Systolic Multipliers for GF(2^m) Fields Defined by All-One and Equally Spaced Polynomials," *IEEE Trans. Computers*, Vol. 50, no. 5, pp. 385–393, May 2001.

[6] R. S. Waters, E. E. Swartzlander, Jr., "A Reduced Complexity Wallace Multiplier Reduction," *IEEE Trans. Computers,* Vol. 59, no. 8, August 2010.

[7] Lee C. Y and Chiou C. W, "New Bit-Parallel Systolic Architectures for Computing Multiplication, Multiplicative Inversion and Division in GF(2^m) Under Polynomial Basis and Normal Basis Representations," *Signal Processing Systems. Journal*, Vol. 52, pp. 313-324, 2008.

[8] P. K. Meher, J. Xie and J. He, "Low-Complexity Multiplier for GF(2^m) Based on All-One Polynomials," *IEEE Trans. Very Large Scale Integration (VLSI) Systems.*, Vol. 21, no. 1, pp. 168-173, 2013.

[9] Lee C. Y, Chang C. C, Hou T. W, Lin J. M and Chiou C. W, "Concurrent Error Detection and Correction in Gaussian Normal Basis Multiplier over GF (2^m)," *IEEE Trans. Computer*, pp. 851-857, Vol. 58, No.6, 2009.

[10] Sargunam B and Dhanasekaran R, "Error Detection Schemes for Finite Field Multipliers," *American Journal. Applied Sciences*, pp.137-144, Vol.11, No.1, 2014.

[11] Mathew J, Jabir A. M and Pradhan D. K, "Design Techniques for Bit-Parallel Galois Field Multipliers with On-line Single Error Correction and Double Error Detection," *IEEE.14th Int. On-Line Testing Symposium*, pp.16-21, 2008.

[12] "Parity-based On-Line Detection for a Bit-Parallel Systolic Dual-Basis Multiplier over GF (2^m)," *in Proc. IEEE Int. Symposium*, 2006.

[13] Mathew J, Jabir A. M, Rahaman H, Pradhan D. K, "Single error correctable bit Parallel multipliers over GF(2^m)," *Computers and Digital Techniques*, pp.281-288, Vol.3, No. 3, 2009.

[14] Mathew J, Costas A, Jabir A. M, Rahaman H and Pradhan D. K, "Single Error Correcting Finite Field Multipliers Over GF (2^m)," *21st Int.Conf.VHDL Design*, pp. 33-38, 2008.

[15] Ping Yeh Yin, Yuan Ho Chen, Chih Wen Lu, Shian Shing Shyu, Chung Lin Lee, Ting Chia Ou, and Yo Sheng Lin, "A Multi-Stage Fault-Tolerant Multiplier with Triple Module Redundancy (TMR) Technique," *in Proc. 4th Int. Conf. Intelligent Systems, Modelling and Simulation*, pp. 636-641, 2013.

[16] Hentschke R, Marques F, Lima F, Carro L, Susin A, Reis R, "Analyzing Area and Performance Penalty of Protecting Different Digital Modules with Hamming Code and Triple Modular Redundancy," *in Proc. 15th Symposium on Integrated Circuits and Systems Design*, 2002.

[17] Rahaman H, Mathew J and Pradhan D. K, "Constant Function Independent Test Set for Fault Detection in Bit Parallel Multipliers in GF(2^m)," *20th Int. Conf. VLSI Design*, 2007.

[18] Rahaman H, Mathew J, Sikdar B. K, and Pradhan D. K, "Transition Fault Testability in Bit Parallel Multipliers over *GF(2^m)," in Proc. 25th IEEE VLSI Test Symposium*, 2007.

[19] Rizwan A, Ashraf, Ouns Mouri, Rami Jadaa and Ronald F and DeMara, "Design-For-Diversity for Improved Fault- Tolerance of TMR Systems on FPGAs," *in Proc. Int. Conf. Reconfigurable Computing and FPGAs*, pp. 99-104, 2011.

[20] Praveen Kumar Samudrala, "Selective Triple Modular Redundancy (STMR) Based Single-Event Upset (SEU) Tolerant Synthesis for FPGAs," IEEE Trans. Nuclear Science, Vol. 51, No. 5, pp. 2957-2969, 2004.