Fault Detection Multipliers in Polynomial and Normal Basis

Siddharth Shelly
ECE Department
Viswajyothi College of Engineering and Technology
Vazhakulam, Muvattupuzha, Ernakulam, Kerala, India.

Babu T Chacko
EEE Department
Viswajyothi College of Engineering and Technology
Vazhakulam, Muvattupuzha, Ernakulam, Kerala, India.

ABSTRACT
With significant advances in wired and wireless technologies and also increased shrinking in the size of VLSI circuits, many devices have become very large because they need to contain several large units. This large number of gates and in turn large number of transistors causes the devices to be more prone to faults. These faults especially in sensitive and critical applications may cause serious failures and hence should be avoided. In many cryptographic schemes, the most time consuming basic arithmetic operation is the finite field multiplication and its hardware implementation for bit parallel operation may require millions of logic gates. Some of these gates may become faulty in the field due to natural causes or malicious attacks, which may lead to the generation of erroneous outputs by the multiplier. New architectures are developed to detect erroneous outputs caused by certain types of faults in bit-serial polynomial basis multipliers and digit-serial normal basis multipliers over finite fields of characteristic two. In particular, parity prediction schemes are developed for detecting errors due to single and certain multiple stuck-at faults.

Keywords
Finite fields, Polynomial basis, Normal basis, Error detection

1. INTRODUCTION
With the rapid expansion of the Internet and wireless communications, more and more digital systems are becoming increasingly equipped with some form of cryptosystems to provide various kinds of data security. Many such cryptosystems rely on computations in very large finite fields and require fast computations in the fields. Finite field arithmetic operations are also used in error control coding, VLSI testing, and digital signal processing. To satisfy the high speed requirements of many such applications, there is a need to develop an efficient architecture for finite field multiplication which is suitable for VLSI implementation.

A fault-tolerant system is one that can continue the correct performance of its specified tasks in the presence of hardware and/or software faults. Fault detection is the process of recognizing that a fault has occurred. Fault detection is often required before any recovery procedure can be implemented. Recently a number of schemes have been developed for the detection of errors in hardware implementation of some arithmetic operations, which have applications in cryptography.

Among the basic arithmetic operations over finite fields $GF(2^m)$, multiplication is the one which has received the most attention in the literature. This is mainly because the implementation of a multiplier is much more complex compared to a finite field adder and, by using a multiplication operation repeatedly one can perform other difficult field operations, such as inversion and exponentiation, which are extensively used in cryptographic systems. Finite field multiplication is quite different from its counterparts in integer and floating-point number systems. For today's cryptographic applications, the field size can be very large and each input of the multiplier can be 160 to 2,048 bits long. Such a multiplier may require millions of logic gates and it is possible that errors will occur in the computation due to faults in the field. If one can have a multiplier which is capable of detecting error online at the presence of certain faults, cryptographic schemes can be operated more reliably.

Additionally, a number of schemes for detecting errors in arithmetic operations of the symmetric block ciphers are there. These schemes are mostly based on parity and/or residue codes. Here error detection of the arithmetic operations over binary extension fields are based on parity prediction technique. These schemes are more generic in the sense that they can be applied to different implementations, e.g., bit-serial, bit-parallel and/or digit-serial, and also they can be applied to different bases for the field representation such as polynomial, dual and optimal normal bases. Additionally, the schemes presented here have high error detection capability, e.g., based on our simulations, the majority of them have a percentage of error detection higher than 99% with a moderate amount of redundancies.

2. TRADITIONAL MULTIPLIER
Bit-Parallel Polynomial Basis Multipliers

Let the monic irreducible binary polynomial that defines the field $GF(2^m)$ be

$$F(z) = z^m + \sum_{i=0}^{m-1} f_i z^i$$

of degree m, where $f_i \in GF(2^m)$ for $i = 0, 1, \ldots m - 1$. Let $\alpha \in GF(2^m)$ be a root of $F(z)$, i.e., $F(\alpha) = 0$. Then the set $\{1, \alpha, \alpha^2, \ldots \alpha^{m-1}\}$ is known as the polynomial (or standard) basis and each element of $GF(2^m)$ can be written with respect to this basis, i.e., if A is an element of $GF(2^m)$, then

$$A = \sum_{i=0}^{m-1} a_i \alpha^i, \qquad \alpha \in \{0,1\},$$

where a_is are the coordinates of A with respect to polynomial basis (PB). For convenience, these coordinates will be denoted in vector notation as

$$a = [a_0, a_1, a_2, \ldots\ldots a_{m-1}]^T$$

where T denotes the transposition of a vector.

Let C be the product of any two elements A and B of $GF(2^m)$. Then, C can be obtained with respect to the PB using two steps of polynomial multiplication and modular reduction as follows:

$$S = A \cdot B = A \cdot \sum_{i=0}^{m-1} b_i \alpha^i = \sum_{i=0}^{m-1} b_i (A\alpha^i),$$

$$C = S \bmod F(\alpha) = \sum_{i=0}^{m-1} b_i \cdot ((A\alpha^i) \bmod F(\alpha))$$

$$= \sum_{i=0}^{m-1} b_i \cdot X^{(i)}$$

Where

$X^{(i)}=\alpha \cdot X^{(i-1)} \bmod F(\alpha),\ 1 \le i \le m-1$

And $X^{(0)}=A$

A traditional bit-parallel architecture for $GF(2^m)$ multiplication using above equation is shown in Fig. 1. It mainly consists of three types of modules, namely, the sum, pass-thru, and α modules. The sum module (denoted as a double circle with a plus inside) is to simply add two $GF(2^m)$ elements and it can be realized in hardware using m two-input XOR gates. The pass-thru module (denoted as a double circle with a dot inside) is to multiply a $GF(2^m)$ element by a $GF(2)$ element, i.e., if $X^{(i)} \in GF(2^m)$ and $bi \in GF(2)$ are two inputs to a pass-thru module, then its output is

$$b_i X^{(i)} = \begin{cases} X(i) \text{ if } bi = 1, \\ 0 \quad \text{if } bi = 0 \end{cases}$$

In hardware, each pass-thru module consists of m two input AND gates. In Fig. 1, the third module (i.e., the rectangular shape α module) multiplies its input, which is an element of $GF(2^m)$, by α and reduces the result modulo $F(\alpha)$. Thus, this module is to essentially realize in hardware. Since α is a root of $F(z)$.

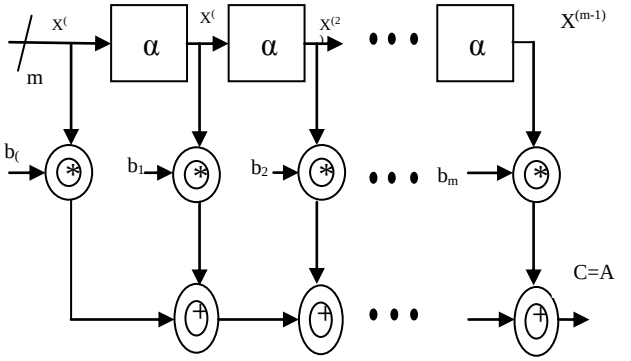


Fig.1 Multiplication of two elements in $GF(2^m)$ in polynomial basis

4.1. FAULT DETECTION STRATEGY

The fault detection strategy is done by parity prediction technique. This method is shown in Fig. 2, where the CUT (circuit under test) block can be either a complete finite field multiplier or a part of it, with A and B as inputs and Y as output, where A,B $\in GF(2^m)$. In this figure, the parity generation (PG) block produces the actual parity of Y, i.e. ,$Py=\sum_{i=0}^{k-1} yi$ where y_i are the coordinates of Y . The actual parity P_G is then compared with the predicted parity P_P using a single XOR gate, as shown in the figure. This comparison is monitored by an error indicator flag eCUT, where eCUT = 0 indicates that no error has been detected and eCUT = 1 flags the detection of errors. The parity prediction (PP) block predicts the parity of the output Y using a function which depends only on the inputs of the CUT as P_P = FCUT(A,B).

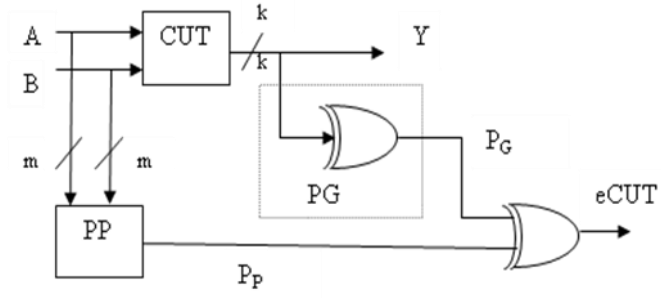


Fig 2:Parity prediction technique

The parities of A and B (i.e., pA and pB, respectively) are available or they can be reliably computed while loading the coordinates of A and B into the multiplier. Here we assume that the PP and PG blocks can be made fault-free or any fault in them can be detected using a suitable mechanism since these blocks are simple and/or regular. The basic assumptions for the fault model that we use in the entire paper are: 1) a fault in a logical gate (i.e., XOR, AND, etc.) results in one of its inputs or the output being fixed to either a logic 0 (stuck-at-0 or s-a-0 in short) or a logic 1 (stuck-at-1 or s-a-1), respectively, and 2) a fault can be either permanent or transient.

3. FAULT DETECTION IN TRADITIONAL BIT PARALLEL MULTIPLIER

The traditional bit parallel multiplier can be divided in to three modules namely α module, sum modules and pass through modules. So we need to develop parity prediction technique for these three modules.

4.2. Parity Prediction In α Module

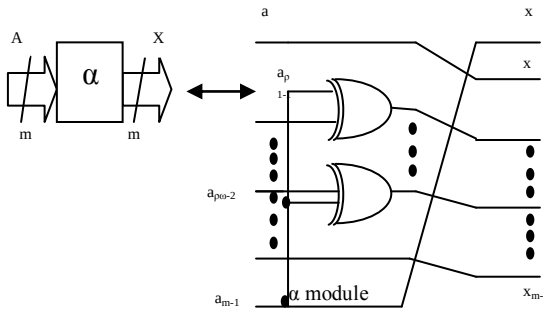
Let ω be the Hamming weight of the irreducible polynomial $F(z)$. Then, $f(z)$ can be written as

$$F(z)= 1 + \sum_{j=0}^{\omega-1} z^{\rho j} + z^m$$

where ρjs are powers of z with $f_{pj} = 1$, $1 \le j \le \omega-2$.Then, we have $1 \le \rho 1 < \rho 2 < \rho 3 < \ldots\ldots\ldots \rho_{\omega-2}$ and above equation can be written as

$$X_i = \begin{cases} a_{\rho j-1} + a_{m-1}\, i = \rho_j,\ 1 \le j \le \omega-2 \\ a_{i-1} \bmod m \ \text{ otherwise} \end{cases}$$

Using the above equation the α module diagram is shown in fig 3.



Let $X = A \cdot \bmod F(\alpha)$ and $P_A = \sum a_i$ is the parity bit of A then the predicted parity bit of X is given by

$$\wedge Px = p_a + a_{m-1} \in GF(2)$$

4.3. Parity Prediction In Sum And Pass-Thru Module

The sum module is a finite field adder which produces the sum of the two elements of $GF(2^m)$ at its output. Let $A = (a_{m-1}, \ldots\ldots a_1, a_0)$ and $B=(b_{m-1},\ldots\ldots b_1, b_0)$ be two inputs to this module. Then, the output is D=(A + B)= $(d_{m-1},\ldots\ldots d_1, d_0)$ where $d_i= a_i + b_i$ for $0 \le i \le m-1$. Then, the parity bit of the output $p_d = \sum_{i=0}^{m-1} d_i$ is predicted by

$$P_d = P_a + P_b$$

The pass-thru module of Fig. 1 multiplies an element $A \in GF(2^m)$ by a single bit $b \in GF(2)$ which can be implemented using m two-input AND gates. Let $G \in GF(2^m)$ be the output of such a module with inputs of A and b. Thus, the output of this module G is zero when b = 0 and A when b = 1.Then the parity of the output is predicted by

$$P_g = b \cdot P_a$$

Thus combining the above two results ie parity prediction of the α module and the parity prediction technique used in the sum and the pass-through modules the architecture of the bit parallel multiplier is modified as shown below.

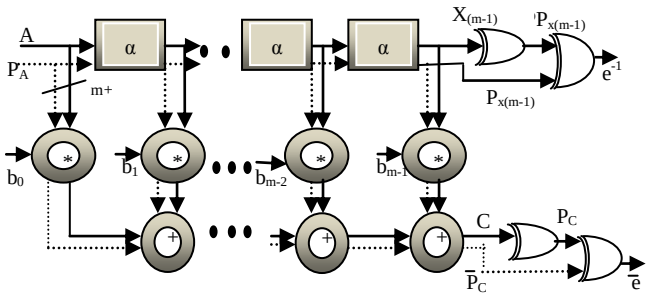


Fig 4: Multiplication of two elements using fault detection capability.

Since the output of any gate of the shaded pass-thru and sum modules in the fig 4 is connected to only one gate, the single stuck fault at any gate of these modules changes only one coordinate of the output of this multiplier. Therefore, a circuit that compares the actual parity P_C with the predicted $\hat{P}_C$ to generate ^e, which is shown at the bottom end of the figure, is capable of detecting any single fault in the shaded sum and pass-thru modules of the figure. Also, it is clear that any single fault in any XOR gate in the parity generation circuit P_C and the very last XOR gate can be detected by ^e. It is noted that such a comparison circuit is implemented using a binary tree of XOR gates which is done by the last three XOR gates at the bottom end of the figure.

5. FAULT DETECTION IN BIT SERIAL MULTIPLICATION

There are two types of bit-serial, namely, LSB-first and MSB-first, multipliers. Below, we consider these multipliers and present their fault detection architectures.

5.1 LSB First multiplier

The PB multiplier of can be realized in a bit-serial fashion as shown in fig 5. In this multiplier structure, both X = (x_{m-1} ….. x_1, x_0) and Y = (y_{m-1} …… y_1, y_0) are m bit registers. Let X(n) and Y(n) denote the contents of X and Y at the nth, $0 \le n \le m$, clock cycle, respectively. Suppose the X register is initialized with A, i.e., X(0) = A, then the content of this register at the nth clock cycle is X(n) = $X^{(n)}$, where $X^{(n)} \in GF(2^m)$. Also, suppose that the register Y is initially cleared, i.e., Y (0) = 0. Then, one can obtain the content of Y at the first clock cycle as Y(1) = b_0A and, in general, at the nth clock cycle as Y(n) = b_0A + $\sum_{i=1}^{n}$ b_iX(i), $1 < n \le m$ .Noting the fact that X(n) = $X^{(n)}$, one can determine that, after m clock cycles, Y contains C = AB $\in GF(2^m)$, i.e., Y(m) = C. Since the coordinates of B enter the multiplier from the least significant bit (LSB), i.e., b_0, this multiplier is referred to as the LSB first bit-serial multiplier.

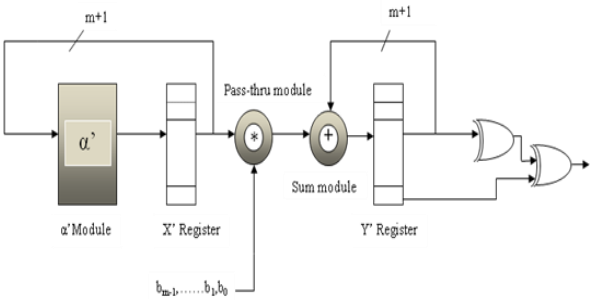


Fig 5: Multiplication of two elements in LSB multiplier

In order to detect fault two m-bit registers X = (x_{m-1},….. x_1 ,x_0) and Y = (y_{m-1}, …… y_1, y_0) are replaced with (m + 1) -bit registers X' = (x_m, x_{m-1}…. x_1, x_0) and Y'=(y_m, y_{m-1},… y_1, y_0).Now we initialize X' and Y' with X'(0) =(pA,am-1,……,a1,a0) and Y'(0)=(0,0,…..,0,0).After m clock cycles and in fault-free situation, the output of Y' will be Y'(m)=(pC, cm-1,…..,c1,c0) and e = $\sum_{i=0}^{m-1}$ ci + Pc = 0

If a single fault (permanent or transient) occurs in the α' module and the register X' (except the x_m bit and the XOR gate connected to it) in then, in each clock cycle, the number of erroneous bits of Y' may increase. This is because, in the α' module , there is a cyclic shift with addition operation and it may increase the number of errors. Thus, at the end of the m^{th} clock cycle, there is a 50 percent chance that this fault is detected. If the single fault occurs in other parts of (including input and output of x_m, the output of the shaded pass-thru, and the sum modules as well as Y' and the PG circuit), then, at the end of the mth clock cycle, at most one bit of the register Y' will be in error and, thus, it will be detected by ^e. As a result, the total fault coverage is greater than 50 percent.

The probability of fault detection in the bit-serial multiplier can be increased by checking the contents of two registers in every clock cycle. Consider the above fig 5 before the triggering of the $(n+1)^{th}$ clock cycle. At this time ,the input and output of the X register are X(n+1) and X(n), respectively and using the above theorem the α module we have

$$P_{x(n+1)} = P_{x(n)} + x_{m-1}(n) = \sum_{i=0}^{m-1} x_i(n)$$

where $x_i(n) \in GF(2)$ is the i^{th} coordinate of X(n). In order to compare ^pX(n) with the actual value of pX(n), we need to store ^pX(n+1) into a 1-bit register D_X. Then, after the nth clock cycle, X(n) appears at the output of register X and the actual value of pX(n) is evaluated and compared with the value of D_X, i.e., ^pX(n), using the last XOR gate of figure that generates ^eX(n). a similar expression can be obtained for the Y register.Since Y(n+1) =Y(n)+ b_nX(n), then

$$P_{y(n+1)} = P_{y(n)} + b_n P_{x(n)}$$

This can be implemented and it is delayed by a 1-bit register D_Y to obtain ^pY (n). Then it is compared with the actual value.p_Y(n). If no fault occurs, after the first clock cycle, both $\hat{e}_{x(n)} = \hat{p}_{x(n)} + p_{x(n)}$ and $\hat{e}_{y(n)} = \hat{p}_{y(n)} + P_{y(n)}$ should be 0 and remain until the last clock cycle, ie ^eX(n) = 0.and ^eY (n)= 0 for $1 \le n \le m$ the existence of a fault is detected. For the cryptographic applications where m is a large number, it is most likely that either ^eX(n) or ^eY(n) is 1 in at least one clock cycle if a single permanent fault occurs. It can be seen that the probability of fault coverage here is more.

5.2 MSB FIRST MULTIPLIER

The PB multiplication of A and B in can also be written as

$$C=(((b_{m-1}A\alpha + b_{m-2}A)\alpha + b_{m-3}A) + \ldots\ldots + b_1A)\alpha + b_0A.$$

This equation can be realized by the below architecture, which is known as the most significant bit (MSB) first bit-serial multiplier,

the registers U and V are initialized with A=(a_{m-1},…..,a_1,a_0) and 0=(0,…..0), respectively.

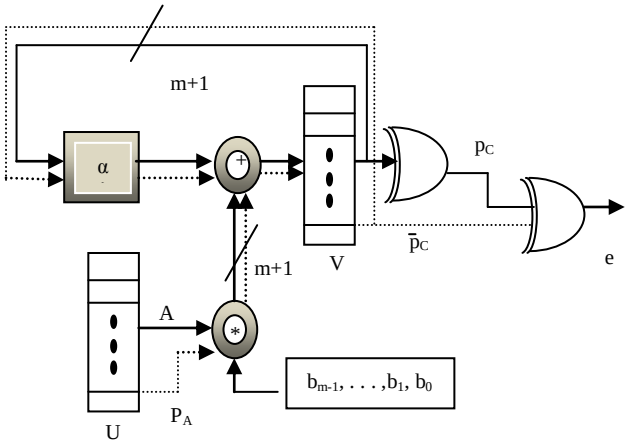


Fig 6: Multiplication of two elements in MSB multiplier

Let V(n) denotes the content of V=(v_{m-1},…….,v_1,v_0) at the nth, 0 ≤ n ≤ m, clock cycle. Then we can see that V(0)=0,V(1)=b_{m-1}A,V(2)=b_{m-1}Aα+b_{m-2}A,…

$$V(n+1)=V(n)\alpha + b_{m-n-1}A, \quad 0 \le n \le m-1$$

Thus after the m^{th} clock cycle,V(m)=C,ie the register contains the coordinates of C. Now to detect the fault its α modules, sum and pass through modules are replaced with the fault detecting corresponding modules we used for the bit parallel and LSB multiplier.

6. DIGIT-SERIAL NORMAL BASIS MULTIPLIERS

Consider an element B which belongs to the Normal Basis in $GF(2^m)$, where $B = (b_0,b_1,b_2....b_{m-1})$. Then the register B in the architecture diagram shown below contains

$$B_i = B^{2^{\wedge}(-di)} = (b_o\alpha,b_1\alpha^2,.....b_{m-1}\alpha^{2^{\wedge}(m-1)})$$

Consider an element A be a normal basis element in $GF(2^m)$. Let n = [m/d], where d is the selected digit size. Then the field element can be represented by

$$A = \sum_{i=0}^{n-1} A_i^{2^{\wedge}(id)}$$

Where

$$A_i = \sum_{j=0}^{d-1} a_{id+j}\alpha^{2^{\wedge}j}$$

Here the partial multiplier can be found by two ways. One by using Sunar and Koc multiplier method and other by Redundant basis method.

6.1 Sunar And Koc Multiplier Method

Here the multiplication takes place by three steps.

1) Convert the elements represented in the basis *M* to the the basis *N* using the permutation.
2) Multiply the elements in the basis *N*.
3) Convert the result back to the basis *M* using the inverse permutation.

Conversion of M basis elements to N basis elements.

Assume that the element A belongs to the basis M and its going to be converted to basis N and A in basis M is given as $A= a'_1\beta+a'_2\beta^2+a'_3\beta^4+……a'_m\beta^{2^{\wedge}(m-1)}$. After the conversion A will be given as $A=a_1\beta+a_2\beta_2+a_3\beta_3+.....a_m\beta_m$. So the elements are converted as $a_j=a'_i$ as

$$j = \begin{cases} k & if\ k \in [1,m] \\ (2m+1)\ if\ k \in [m+1,\ 2m] \end{cases}$$

where $k=2^{i-1}$ mod (2m+1) for i=1,2,….m

Multiply the elements in the basis *N*

The conversion does not requires any gate. But the multiplication requires the gate. So the elements represented in N basis are

$$A = \sum_{i=1}^{m} a_i\beta_i = \sum_{i=1}^{m} a_i(\partial^i + \partial^{-i})$$

$$B = \sum_{i=1}^{m} b_i\beta_i = \sum_{i=1}^{m} b_i(\partial^i + \partial^{-i})$$

Now C=A.B given by

$$C = A\,.\,B = \left(\sum_{i=1}^{m} ai(\partial^i + \partial^{-i})\right)\left(\sum_{i=1}^{m} bj(\partial^i + \partial^{-i})\right)$$

Here it is divided in to three parts C1,D1,D2 where C1 contains elements of ∂^{i-j} and D1,D2 contains elements of ∂^{i+j}. So C1,D1 and D2 can be calculated and represented in table format by

Table 1. Contents of C1

β_1	β_2	..	β_{m-2}	β_{m-1}	β_m
$a_1b_2 + a_2b_1$ $a_2b_3 + a_3b_2$ . . . $a_{m-2}b_{m-1}+a_{m-1}b_{m-1}$ $a_{m-1}b_m+ a_mb_{m-1}$	$a_1b_3 + a_3b_1$ $a_2b_4 +a_4b_2$ . . . $a_{m-2}b_m+a_mb_{m-2}$	. . . .	$a_1b_{m-1} + a_{m-1}b_1$ $a_2b_m + a_mb_2$	$a_1b_m + a_mb_1$	

Table 2. Contents of D1

β1	β_2	β_3		β_{m-2}	β_{m-1}	β_m
	a1b1	a_1b_2 a_2b_1	… …	a_1b_{m-3} a_2b_{m-3} .. .. $a_{m-3}b_1$	a_1b_{m-1} a_2b_{m-3} .. .. $a_{m-3}b_2$ $a_{m-2}b_1$	a_1b_{m-1} a_2b_{m-2} .. .. $a_{m-3}b_3$ $a_{m-2}b_2$ $a_{m-1}b_1$

Table 3. Contents of D2

β_1	β_2	β_3	...	β_{m-2}	β_{m-1}	β_m
a_mb_m	$a_{m-1}b_m$ a_mb_{m-1}	$a_{m-2}b_m$ $a_{m-1}b_{m-1}$ $a_m b_{m-2}$	... ...	a_3b_m a_4b_{m-1} a_5b_{m-2} . . . $a_{m-1}b_4$ $a_m b_3$	a_2b_m a_3b_{m-1} a_4b_{m-2} . . . $a_{m-2}b_4$ $a_{m-1}b_3$ a_mb_2	a_1b_m a_2b_{m-1} a_3b_{m-2} . . . $a_{m-3}b_4$ $a_{m-2}b_3$ a_mb_1

So the final product can be obtained as C = C1+ D1+ D2. Now we want to reconvert this which is in N basis to M basis using the same permutations.

6.2 REDUNDANT BASIS MULTIPLIER METHOD.

Let define $C_i = A_iB_i$ then the product of C_i can be computed as

$$C_i = a_{id}\,\alpha B_i + (a_{(1+id)}\alpha B_i^{2^{\wedge}-1})^2 + ...(a_{(d-1+id)}\,\alpha\, B_i^{2^{\wedge}-(d-1)})^{2^{\wedge}(d-1)}$$

Now the αB is found by converting the B to redundant basis.

$$\alpha B = \sum_{j=0}^{m-1} (b_{0, F(2^j)} + b_{0, F(0)} \alpha^{2^j})$$

Where

$$\bar{b}_{0,F(j)} = b_{F(j-1)} + b_{F(j-2m)} + \ldots + b_{F(j-2^{(m(t-1)))}}$$

Here the $b_{F()}$ is the representation of B in redundant basis. If $A=(a_0,a_1,a_2\ldots\ldots a_{m-1})$ indicate a normal basis elements in $GF(2^m)$, then the redundant basis can be obtained by $F(2^i 2^{mj} \bmod p)=i$ where $0\leq I \leq m-1$ and $0\leq j \leq t-1$. For eg. If m=5 then p=2m+1=11 and if t=2 then the redundant basis representation becomes

$a_{F(1)}=a_0$ $a_{F(2)}=a_1$
$a_{F(3)}=a_3$ $a_{F(4)}=a_2$
$a_{F(5)}=a_4$ $a_{F(6)}=a_4$
$a_{F(7)}=a_2$ $a_{F(8)}=a_3$
$a_{F(9)}=a_1$ $a_{F(10)}=a_0$

In the initial step, both NB elements A and $B^{2^{(-d(n-1))}}$ are stored in registers A and B, respectively. During first round a_{n-1} and B_{n-1} is computed and then corresponding c_{n-1} is calculated and stored in register C. In next cycle both registers B and C must be cyclically shifted to the left and the right by d digits, respectively. The result produced is added to the register C in just before cycle. After n iteration, the final register c can obtain the whole normal basis multiplication. The architecture is shown in the fig 8.

7. PARITY PREDICTION IN DIGIT-SERIAL NORMAL BASIS MULTIPLIER

The digit-serial normal basis multiplier shown in the fig. 7 mainly consists of a partial multiplier and a sum module. So the parity prediction of both this partial multiplier and the sum modules should be found out.

1) Parity prediction of partial multiplier for computing A_iB_i

The partial multiplier consist of a α module and computes the value of αB. Depending upon this computation the calculation of parity also changes. Here we can compute this by using multiplication of Gaussian Normal basis of type t. Ie if the value of t=1 it becomes optimal normal basis of type I and if the value of t=2 it becomes optimal normal basis of type II. So the parity is different for both the type. If its value is computed by t=1 ie optimal normal basis of type I then the parity is given by

$$P_{\alpha B^{\wedge} i} = P_b + b_{m/2+i}$$

If its value of t=2 ie optimal normal basis of type II then the parity is given by

$$P_{\alpha B^{\wedge} i} = \sum_{i=0}^{m-1} b_i \ \{ \text{if } a_i = 1\}$$

Thus depending upon the type of computation that is used to perform the parity must be calculated.

2) Parity prediction of the sum module

The main functions taking place in this modules is the addition and the shifting. Here the shifting will not cause any change in the parity. So the parity is caused only due to addition. So the parity of this module is calculated by

$Pc = (((pc_{n-1}) + pc_{n-2}) + \ldots) + Pc_0$ So when the parity prediction is incorporated the new architecture becomes

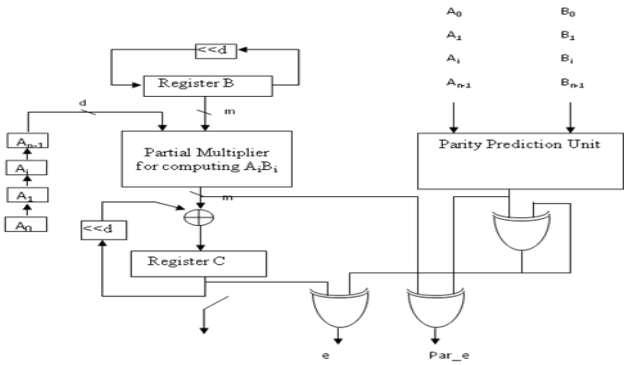


Fig 8: Parity prediction in Digit-Serial normal basis multiplier

8. RESEARCH WORK PROPOSED

The time and area overhead required for the additional parity prediction is less. The parity predictions used in these multipliers are capable of detecting error online at the presence of certain faults, which make cryptographic schemes more reliable. These parity prediction schemes predict errors due to single and certain multiple faults during the multiplication operation in the field. But in case of multiple faults due to malicious attacks or natural causes that result in an even number of errors at the output, the proposed structures are unable to detect these faults. The parity Prediction also cannot be used for detecting soft errors. So some other technique can be used to detect errors and which is more reliable like cyclic redundancy check (CRC), Hamming codes etc. We can also relay on the some of the algorithms that are using in the VLSI testing field like the fan algorithm. Apart from that these techniques can also be implemented in different kind of multipliers in both the normal and polynomial basis.

9. CONCLUSION

The proposed parity prediction technique will make the cryptographic systems more reliable as they are checking whether there is an error while the entire system is working. So works are going on to implement some error detection techniques in all types of cryptographic multipliers

10. REFERENCES

[1] S. Fenn, M. Gossel, M Benaissa, and D. Taylor, "On Line Error Detection for Bit Serial Multipliers in $GF(2^m)$." *Journal of electronic Testing: Theory and Applications, vol.13, pp.29-40, 1998*

[2] B. Sunar and C. K. Koc "An Efficient Optimal Normal Basis Type II Multiplier." *IEEE Trans.Computers,50(1), 83-87,Jan.2001*

[3] Siavash Bayat-Sarmadi and M. Anwar Hasan "On Concurrent Detection of Errors in Polynomial Basis Multiplication" *IEEE transactions on very large scale integration (vlsi) systems, vol. 15, no. 4, April 2007*

[4] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard*," IEEE Trans. Computers, special issue on cryptographic hardware and embedded systems, vol. 52, no. 4, pp. 492-505, Apr. 2003*

[5] Arash Reyhani-Masoleh and M. Anwar Hasan "Low complexity bit parallel architectures for polynomial basis multipliers over $GF(2^m)$."*IEEE Trans.Computers,vol.53,no.8,pp.945-959,AUG.2004*