18 Years of Redundant Basis Multipliers over Galois Field

Roma Chourasia
M.Tech. Scholar
Electronics and Communication Department
M.A.N.I.T., Bhopal (M.P.)

Kavita Khare, PhD
Professor
Electronics and Communication Department
M.A.N.I.T., Bhopal (M.P.)

ABSTRACT
Redundant basis multipliers over Galois Field have gained huge popularity in elliptic curve cryptography mainly because of their negligible hardware cost for squaring and modular reduction. Different techniques used so far for the implementation of redundant basis multipliers over Galois Field are explored here. Based on review the Word Level Redundant Basis multiplier is the most efficient among all multipliers in terms of hardware utilization. Digit serial Redundant Basis multiplication in a bit level matrix vector form is most efficient in terms of area-time complexities.

General Terms
Redundant Basis Multiplication

Keywords
Galois Fields (GF (2^m)), Redundant Basis (RB) multiplier

1. INTRODUCTION
Multiplication over GF (2^m) is a basic operation frequently came across in modern cryptographic systems such as the Elliptic Curve Cryptography(ECC) and error control coding [1]–[3]. Also multiplication over a Galois field can be used to perform other field operations, e.g. division, exponentiation, and inversion [4]–[6]. Arithmetic operations in the GF (2^m) have several applications in computer algebra and theory of coding. Multiplication over GF (2^m) can be implemented on a general purpose machine, but it is expensive to use a general purpose machine to implement cryptographic systems in cost-sensitive consumer products. As compared to the order of GF (2^m) the word length of low end microprocessors used in cryptographic systems is too small, therefore, they cannot meet the real time requirements of different applications. Most of the real-time applications, therefore, need hardware implementation of GF (2^m) arithmetic operations for the benefits like low-cost and high-throughput rate. There are different types of bases to represent field elements, those are polynomial basis, normal basis, triangular basis and RB, and the choice of representation of field elements has a major impact on the performance of the arithmetic circuits [7]–[9], [15]. Several algorithms for basic arithmetic operations in GF (2^m) are suitable for both hardware and software implementations have been recently developed. Because of several advantages of the RB based multipliers [6], [10] they have gained significant attention in recent years. Like normal basis multipliers, RB multipliers offer free squaring, they also involve lower computational complexity and can be implemented in highly regular computing structures [10]–[14]. Several digit-level serial/parallel structures for RB multiplier over GF (2^m) have been reported in the last few years [10]–[14]. An efficient serial/parallel multiplier using redundant representation has been presented [10]. A bit-serial word-parallel (BSWP) architecture for RB multiplier has then been reported [11]. Several other RB multipliers have also been developed for reducing the complexity of implementation and for high-speed realization [12]–[14].

2. LITERATURE REVIEW
Multiplication is more complicated, whereas division or inversion can be broken down into a series of consecutive multiplication operations. Therefore in practice, a binary field (Galois field of characteristic two) multiplier becomes the key arithmetic unit and VLSI design core for the hardware systems, based on Galois field computations. The way in which GF (2^m) multiplication is performed is dependent on the representation bases in a binary field. Efficiency of Galois field multiplication depends on the choice of the basis to represent field elements. Bases that have been used for efficiently realizing Galois field multipliers include polynomial basis, normal basis (NB), dual basis, triangular basis, and redundant representation or redundant. Among these, redundant basis representation is especially interesting, because likewise normal basis multiplier it offers almost free squaring and also eliminates modular operation for multiplication. RB representation has high modularity and exhibits carry-free addition, which can be used to design high performance multipliers. The main idea of multiplication using redundant representation is to perform multiplication by embedding the field in a larger ring. The ring used here is a cyclotomic ring and has a very simple structure, such that the modular operation can be saved in a multiplication operation. The main drawback for redundant representation is that it uses more bits to represent an element as compared to other representation basis. The number of representation bits depends on the size of the cyclotomic ring. However, for the class of fields GF (2^m) for which there exists a type I optimal normal bases (ONB), the number of bits required for a redundant representation of a field element is slightly higher for large m—(m+1) bits compared to m bits used for the other bases. The work done in the field of redundant representation and RB multipliers over GF (2^m) in all these years is depicted in Table 1.

Table 1. RB multipliers over GF (2^m) and their features

Reference Number	Year	Scheme	Remarks
[6]	1998	Redundant representation introduction	Time and area complexities reduced as compared to normal and other bases
[18]	2001	Redundant basis multiplication introduction	Low power computation
[10]	Nov. 2002	Smallest value of GF variable is defined for practical uses Bit serial and bit parallel architectures for multipliers	Time and space complexities reduced
[17]	Dec. 2002	Optimal representation on Redundant basis	Representation and corresponding application's results improved
[19]	2003	Elements of GF are represented as elements in a ring	Time efficient multiplier with little increase in number of gates(XOR and AND gates)
[12]	2008	SIPO multiplier architecture and algorithm	All complexities are reduced
[13]	2009	Word level RB multiplication algorithm	Reduced delay
[14]	2012	New word level RB multiplication algorithm	Hardware utilization efficiency increased, and this multiplier is faster as compared to previous word level RB multiplier. Critical path delay independent of the size of the field (m in GF (2^m)) and word (w).
[16]	2015	Lesser number of partial products are generated using bit matrix form	Number of registers and adders is reduced, also calculation time is reduced

3. REDUNDANT BASIS REPRESENTATION

3.1 Polynomial Ring Representation

Polynomial Ring Representation is the first redundant representation of GF (2^m) [6]

Consider the residue polynomial ring $F_2(x)/ (x^n +1)$, where gcd (n, 2) = 1, and let h(x) be an irreducible factor of $x^n +1$ in $F_2(x)$, deg [h(x)] = m. The polynomial g(x) = $(x^n +1)/h(x)$ generates a minimal ideal (binary irreducible cyclic code) in $F_2(x)/ (x^n +1)$, denoted by g(x). The irreducible cyclic code has block length of n, dimension m, and parity check polynomial h(x). It is well known that g(x), together with addition and multiplication modulo $x^n +1$, is isomorphic to the field $F_{2^m} = F_2(\alpha)$, where α^{-1} is a root of h(x). In particular, the following are field isomorphism:

Ψ: {g(x)} $\longrightarrow F_{2^m} = F_2(\alpha)$

Ψ: $\alpha(x) \rightarrow \alpha(\alpha^{-1})$

Φ: $F_{2^m} = F_2(\alpha) \rightarrow$ {g(x)}

Φ: $\rho \rightarrow \sum_{i=0}^{n-1} \mathrm{Tr}(\rho\alpha^i)x^i$

Where Tr denotes the GF (2^m) linear trace mapping from $F_{2^m} = F_2(\alpha)$ onto F_2. Ψ and Φ are easily seen to be inverses of each other. The identity in the field {g(x)} is the primitive idempotent e(x) ϵ g(x).

Under the isomorphism Ψ and Φ, the elements of F_{2^m} are written as polynomials in x of degree n − 1 or less lying in g(x). The representation is called polynomial ring representation.

The elements of F_{2^m} take the form $a_0 + a_1x + \ldots. + a_{n-1}x^{n-1}$ where $a_i \epsilon F_2$. The representation is redundant because it uses n components, instead of m < n components as in conventional representations.

3.2 Optimal redundant Representation

The optimal redundant representation of GF (2^m) [17], [18] is useful when implementing GF (2^m) arithmetic on hardware.

The smallest n ϵ IN with GF (2) [x]/ ($x^n +1$) containing an isomorphic copy of GF (2^m), m>1, is n: min {k ϵ IN: $x^k +1$ has an irreducible divisor in GF (2) [x] of degree d ϵ IN such that m|d}

n is odd because, for n=2.n', $x^n +1 = (x^{n'} +1)^2$. Hence, in contradiction to the minimality of n, the polynomial $x^{n'}+1$ would contain an irreducible divisor of the required degree. The polynomial x^n+1 are square-free because $x^n +1$ has no multiple roots. Thus, from the Chinese Remainder Theorem, we know that GF (2) [x]/ ($x^n +1$) is isomorphic to a direct product of fields:

GF(2)[x]/($x^n +1$) $\cong k_1$X……Xk_s.; where k_i =GF(2)[x]/($f_i(x)$) for some irreducible factor $f_i(x)$ of $x^n +1$ ($1 \le i \le s$). If m|deg (f_i) for some irreducible factor $f_i(x)$ of $x^n +1$, then GF (2^m) is a subfield of k_i and consequently contained in GF (2) [x]/ ($x^n +1$).

Conversely, let F: =GF (2^m) be contained in GF (2) [x]/ ($x^n +1$) Š

For each $1 \le i \le s$, let Π_i be the projection

Π_i : F $\rightarrow$ GF (2) [x]/ ($x^n +1$) $\cong k_1$ X …. X $k_s \rightarrow k_i$: Each Π_i is either the zero map or injective since F and k_i are fields. But, they cannot all be the zero maps since F injects into the

product k_1 X …. Xk_s. Therefore, at least one Π_i is injective, so F is a subfield of some k_i

3.3 New Ring Representation

The new ring representation of RB representation of GF (2^m) [19] when used to implement multipliers, increases hardware but makes them faster.

In cryptographic applications, fields based on low weight irreducible polynomials are desired. The weight of a polynomial is the number of terms in it. For example, the weight of ($x^7 + x^3 + x$+1) is 4. About half the irreducible, minimum weight polynomials of degree less than 10,000 have weight 3 (these are called trinomials) and the rest have weight 5 (these are called pentanomials).

The new rings are

GF (2) [x]/ ($x^n + x^k$+1) and GF (2) [x]/ ($x^n + x^{k1} + x^{k2}$ +1).

In each of these rings, there is an isomorphic copy of the field defined by pentanomials. Multiplication in the rings is faster than multiplication in a field defined by pentanomials because the new rings are based on 3 and 4-term polynomials.

Let $f_2(x)$ be an irreducible polynomial of degree m. In what follows, there are polynomials, $f_2(x)$ such that either $x^n + x^k$+1 = $f_1(x)$ X f_2 (x) or $x^n + x^{k1} + x^{k2}$ +1 = $f_1(x)$ X f_2 (x) is satisfied. In both cases, the degree of $f_1(x)$ must be as low as possible in order to make multiplication in rings based on 3-term and 4-term polynomials faster than multiplication in fields based on pentanomials. The two new rings satisfy the following:

GF (2) [x]/ ($x^n + x^k$+1) = GF (2) [x]/ $f_1(x)$ xor GF (2) [x]/ $f_2(x)$

Or

GF (2) [x]/ ($x^n + x^{k1} + x^{k2}$ +1) = GF (2) [x]/f_1(x) xor GF (2) [x]/$f_2(x)$

Note that, in each of the above equations, $f_1(x)$ is different. Therefore, the procedure to find a ring which contains an isomorphic copy of a field, GF (2^m), in which multiplication must be done, is as follows: Assume that no irreducible trinomial exists of degree m.

1. If an irreducible All One Polynomials of degree m exists, then perform the multiplication in GF (2) [x]/ (x^n+1), where n =m + 1.

2. If no ring GF(2)[x]/(x^n+1) can be found such that n=(m+1), then find a ring GF(2)[x]/($x^n + x^k$+1) such that $x^n + x^k$+1 = $f_1(x)$X$f_2(x)$ for some irreducible $f_2(x)$ of degree m. Note that n should be as small as possible.

3. Find a ring GF (2) [x]/ ($x^n + x^{k1} + x^{k2}$ +1) such that $x^n + x^{k1} + x^{k2}$ +1 = $f_1(x)$ X$f_2(x)$ for some irreducible $f_2(x)$ of degree m.

4. Perform multiplication of elements in a field, GF (2^m), defined by a pentanomial.

5. Choose one of the representations from Steps 2, 3, or 4 above for which the multiplier complexity is minimal.

A polynomial of the form $f_2(x) = c_0 + c_1x + c_2x_2 + \ldots + c_m x_m$ is written as (c_m,c_{m-1},…., c_1, c_0). The coefficients c_i are in GF (2).

To satisfy $x^n + x^k$+1 = $f_1(x)$X$f_2(x)$, where $f_1(x)$=x^2 +x+1, $f_2(x)$ must be one of the following two forms: (1;101;101;...;101;1;101;101;...;101;1) (call this type A) or (1;101;101;...;101;10;101;101;...;101;1) (call this type B).

These forms have the pattern (1, 0 or more 101s, 1 or 10, 0 or more 101s, 1). If $f_2(x)$ is of the form (type A) (1, x repetitions of 101, 1, y repetitions of 101, 1), then it follows that k=n-3x-2 or 3y+2. If $f_2(x)$ is of the form (type B) (1, p repetitions of 101, 10, q repetitions of 101, 1), then it follows that k =n-3p-4 or 3q+1.

To satisfy $x^n + x^{k1} + x^{k2}$ +1= $f_1(x)$X$f_2(x)$, where $f_1(x)$=x+1, $f_2(x)$ must be of the form (1;1;1;1;...1 ;0;0;0;...0 ;1;1;...;1).

If $f_2(x)$ is of the form (p repetitions of 1, q repetitions of 0, r repetitions of 1), then it follows that k1=n-p and k2=r.

4. MATHEMATICAL FORMULATIONS

Redundant Basis Multipliers offers negligible hardware cost for squaring, provides lower computational complexities, and also can be implemented in highly regular computing structures.

4.1 Digit Serial RB Multiplier

In Digit serial RB multiplier [12],[16] digit wise partial products for the digit serial multiplication where operands A and B are decomposed into a number of digits, and then the addition of those partial products is done to compute the product word.

Assuming x to be a primitive nth root of unity, elements in GF (2^m) can be represented in the form:

$$A= a_0 + a_1 x + \ldots. + a_{n-1}x^{n-1} \quad (1)$$

Where $a_i \in$ GF (2), for 0≤ i ≤ n-1, such that the set {1, x,x^2,….., x^{n-1}} is defined as the RB for GF (2^m) elements, where n is a positive integer not less than m [6], [10].

For a GF (2^m), when (m+1) is prime and 2 is a primitive root modulo (m+1), there exists a type I optimal normal basis (ONB) [10], where x is an element of GF (2^m), and n=m+1.

Let A, B ϵ GF (2) be expressed in RB representation as

$$A= \sum_{i=0}^{n-1} a_i x^i \quad (2)$$

$$B= \sum_{i=0}^{n-1} b_i x^i \quad (3)$$

where a_i, b_i ϵ GF (2). Let C be the product of A and B, which can be expressed as follows

$$\begin{aligned} C= A\cdot B &= \sum_{i=0}^{n-1}(x^i b_i)\cdot A \\ &= \sum_{i=0}^{n-1}(\sum_{j=0}^{n-1} b_i x^{(i+j)})a_j \\ &= \sum_{j=0}^{n-1}(\sum_{i=0}^{n-1} b_{(i-j)n} x^i)a_j \\ &= \sum_{i=0}^{n-1}(\sum_{j=0}^{n-1} b_{(i-j)n}\ a_j)x^i \end{aligned} \quad (4)$$

where (i-j)n denotes modulo n reduction. Define C=$\sum_{i=0}^{n-1} c_i x^i$ where c_i ϵ GF (2), then

$$c_i = \sum_{i=0}^{n-1} b_{(i-j)n}\ a_j \quad (5)$$

4.2 Word Level RB Multiplier

In Word Level RB Multiplier [13], both the operand A and B are decomposed into number of blocks to achieve digit serial multiplication, and after that the partial products corresponding to these blocks are added together to obtain the desired product word.

Considering equations (1) to (5) of Digit Serial RB Multiplier (3.1)

Let w denotes the word size. Then the operand A in RB representation in k= [n/w] words

$$A= \underbrace{a_0 \dots a_{w-1}}_{A_0} \underbrace{a_w \dots a_{2w-1}}_{A_1} a_{2w} \dots \underbrace{a_{(k-1)w} \dots a_{n-1} 0 \dots 0}_{A_{k-1}}$$

Note that a_j =0 if j > n-1

Replace j in (5) with hw+l

$$c_i = \sum_{l=0}^{w-1} \sum_{h=0}^{k-1} a_{hw+l} b_{i-hw-l} \quad ; \quad i = 0,1,\dots,n-1 \qquad (6)$$

Define new signal $d_{h,i}^{(-1)}$ as follows:

$d_{h,i}^{(-1)}$ =0 and

$$d_{h,i}^{(l)} = d_{h,i}^{(l-1)} + a_{hw+l} b_{i-hw-l} \quad ; \text{for } l=0,1,\dots\dots,w-1 \qquad (7)$$

then it follows from (7)

$$d_{h,i}^{(w-1)} = \sum_{l=0}^{w-1} a_{hw+l} b_{i-hw-l} \qquad (8)$$

comparing (6) with (8), it follows:

$$c_i = \sum_{h=0}^{k-1} d_{h,i}^{(w-1)}$$

This multiplier is faster than previous defined multipliers, but has larger area complexities.

4.3 Digit serial RB Multiplier using Bit level matrix vector form

In Digit serial RB Multiplier in a bit level matrix vector form [16], both the operands are decomposed into a number of blocks to achieve digit serial multiplication and after that the partial products corresponding to these blocks are added together to obtain desired product word.

Considering equations (1) to (4) of Digit Serial RB Multiplier (III.A)

Where (i-j)n denotes modulo n reduction. Define C in a bit level matrix vector form as:

$$\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} b_0 & b_{n-1} & \dots & b_1 \\ b_1 & b_0 & & b_2 \\ \vdots & \vdots & \ddots & \vdots \\ b_{n-1} & b_{n-2} & \cdots & b_0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} \qquad (5)$$

Looking at the structure of bit matrix in (5), n bit shifted (reduced) forms of operand B can be defined as follows

$$B^0 = \sum_{i=0}^{n-1} b_i^0 x^i = b_0 + b_1 x + \dots + b_{n-1} x^{n-1}$$
$$B^1 = \sum_{i=0}^{n-1} b_i^1 x^i = b_{n-1} + b_0 x + \dots + b_{n-2} x^{n-1} \qquad (6)$$
$$\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots$$
$$B^{n-1} = \sum_{i=0}^{n-1} b_i^{n-1} x^i = b_1 + b_2 x + \dots + b_0 x^{n-1}$$

where

$$b_0^{i+1} = b_{n-1}^i$$
$$b_j^{i+1} = b_{j-1}^i \quad ; \text{for } 1 \le j \le n-2 \qquad (7)$$

The recursions on (7) can be extended further to have

$$b_j^{i+s} = \begin{cases} b_{n-s+j}^i & \text{for } 0 \le j \le s-1 \\ b_{j-s}^i & \text{otherwise} \end{cases} \qquad (8)$$

where $1 \le s \le n-1$.

Let Q and P be two integers such that n = QP + r, where $0 \le r < P$. for simplicity of discussion, it is assumed that r=0, and decompose the input operand A into Q number of bit vectors A_u for u = 0,1,….., Q-1, as follows:

$$A_0 = [a_0\ a_Q\ \dots\ a_{n-Q}]$$
$$A_1 = [a_1\ a_{Q+1}\ \dots\ a_{n-Q+1}]$$
$$\dots\dots\dots\dots\dots\dots\dots\dots \qquad (9)$$
$$A_{Q-1} = [a_{Q-1}\ a_{2Q-1}\ \dots\ a_{n-1}]$$

Similarly, Q number of shifted operand vector B_u can be generated, for u = 0,1,…., Q-1

$$B_0 = [B^0\ B^Q\ \dots\ B^{n-Q}]$$
$$B_1 = [B^1\ B^{Q+1}\ \dots\ B^{n-Q+1}]$$
$$\dots\dots\dots\dots\dots\dots\dots\dots \qquad (10)$$
$$B_{Q-1} = [B^{Q-1}\ B^{2Q-1}\ \dots\ B^{n-1}]$$

The product C=A·B given by the bit level matrix vector product in (5) can be decomposed into Q inner products of vectors A_u and B_u for u = 0,1,…., Q-1 as:

$$C=A\cdot B = B_0 A_0^T + B_1 A_1^T + \dots + B_{Q-1} A_{Q-1}^T$$
$$C = \sum_{u=0}^{Q-1} B_u A_u^T = \sum_{u=0}^{Q-1} \overline{C_u} \qquad (11)$$

Where $\overline{C_u} = B_u A_u^T$

Note that each A_u for u = 0,1,….,Q-1 is a P point bit vector and each B_u for u = 0,1,….,Q-1 is a P point bit shifted forms of operand B. From (11) and (12) it can find that the desired multiplication can be performed by Q cycles of successive accumulation of $\overline{C_u}$ for u = 0,1….,Q-1 while each $\overline{C_u}$ can be computed as $\overline{C_u} = \sum_{v=0}^{P-1} B^{u+vQ} a_{u+vQ}$.

The partial products generated in this multiplier are lesser in numbers than those generated in above multipliers, which reduces the computation time, and also reduces register and adder complexities of RB multipliers.

5. COMPARISON OF MULTIPLIERS

In order to compare the hardware requirements of the above discussed multipliers some notations need to be considered. Let R be the number of multiplication nodes, S be the number of parallel arrays, n be the integer and n=m+1 in GF (2^m). The hardware requirements are compared on the number of AND gates, XOR gates and Registers required implementing the multiplication and it is represented in Table 2.

Table 2. Comparison of Hardware Requirements

Multipliers	Registers	AND Gates	XOR Gates
4.1	n	Rn	R(n-1)
4.2	(R+1)n	Rn	(2R-1)n
4.3	Rn+2n	Rn	Rn

According to (11), the RB multiplication can be represented by the 2 dimensional Signal Flow Graph (SFG), in which it is considered that $\boldsymbol{T_A}$ and $\boldsymbol{T_X}$ are the delay of an AND gate and an XOR gate respectively. The time duration of critical path is calculated from SFG represented by $\boldsymbol{T_{cp}}$, and with this Average Computation Time (ACT) is calculated for above discussed multipliers shown in Table 3.

Table 3. Comparison of Critical Path and ACT

Multipliers	Critical Path	ACT
4.1	$T_A + [\log_2 n]\, T_X$	ST_{cp}
4.2	$T_A + T_X$	$ST_{cp} + [\log_2 R]\, T_X$
4.3	$T_A + T_X$	ST_{cp}

6. CONCLUSION

RB multipliers over GF (2^m) are very popular in Elliptic Curve Cryptography because of their negligible hardware cost for squaring and modular reduction. Word Level RB multiplier is the most efficient among all multipliers in terms of hardware utilization. Digit serial RB multiplication in a bit level matrix vector form is most efficient in terms of area-time complexities. Future works can be done to find out new methods to obtain partial products in lesser time and with less hardware requirements.

7. REFERENCES

[1] Alessandro Cilardo, Luigi Coppolino, Nicola Mazzocca, and Luigi Romano, "Elliptic Curve Cryptography Engineering," proc. of IEEE, vol.94, no.2, pp.395-406, Feb.2006.

[2] N.R.Murthy and M.N.S.Swamy, "Cryptographic applications of brahmagupta-bhaskara equation," IEEE Trans. Circuits Syst. I, Reg. Papers, vol.53, no.7, pp.1565-1571, 2006.

[3] L.Song and K.K.Parhi, "Low-energy digit-serial/parallel finite field multipliers," J.VLSI Digit. Process, vol.19, pp.149-166, 1998.

[4] P.K.Meher, "On efficient implementation of accumulation in finite field over GF (2^m) and its applications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.17, no.4, pp.541-550, 2009.

[5] L.Song, K.K.Parhi, I.Kuroda, and T.Nishitani, "Hardware / software codesign of finite field data path for low-energy Reed-Solomn codecs," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.8, no.2, pp.160-172, Apr.2000.

[6] G.Drolet, "A new representation of elements of finite fields GF (2^m) yielding small complexity arithmetic circuits," IEEE Trans. Comput., vol.47, no.9, pp.938-946, 1998.

[7] C.Y.Lee, J.S.Horng, I.C.Jou, and E.H.Lu, "Low-complexity bit parallel systolic Montgomery multipliers for special classes of GF (2^m)," IEEE Trans. Comput., vol.54, no.9, pp.1061-1070, Sep.2005.

[8] P.K.Meher, "Systolic and super-systolic multipliers for finite field GF (2^m) based on irreducible trinomials," IEEE Trans. Circuits Syst. I, Reg. Papers, vol.55, no.4, pp.1031-1040, May 2008.

[9] J.Xie, J.He, and P.K.Meher, "Low latency systolic Montgomery multiplier for finite field GF (2^m) based on pentanomials," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.21, no.2, pp.385-389, Feb.2013.

[10] H.Wu, M.A.Hasan, I.F.Blake, and S.Gao, "Finite field multiplier using redundant representation," IEEE Trans. Comut., vol.51, no.11, pp.1306-1316, Nov.2002.

[11] A.H.Namin, H.Wu, and M.Ahmadi, "Comb architectures for finite field multiplication in F_{2m}," IEEE Trans. Comput., vol.56, no.7, pp.909-916, Jul.2007.

[12] A.H.Namin, H.Wu, and M.Ahmadi, "A new finite field multiplier using redundant representation," IEEE Trans. Comput., vol.57, no.5, pp.716-720, May 2008.

[13] A.H.Namin, H.Wu, and M.Ahmadi, "A high-speed word level finite field multiplier F_{2m} in using redundant representation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no.10, pp.1546-1550, Oct.2009.

[14] A. H. Namin, H. Wu, and M. Ahmadi, "An efficient finite field multiplier using redundant representation," ACM Trans. Embedded Comput. Sys., vol. 11, no. 2, Jul. 2012, Art. 31.

[15] G.Yuvaraj, Anusree.G, Manopriya.K, Shobana.T, Yasodha.R Final, "Design of Word Level Multiplication Algorithm on reordered normal Basis," International Journal of Innovations in Engineering and Technology (IJIET),vol.1, Issue 4, pg.88-94, December 2012.

[16] Jiafeng Xie, P.K.Meher, Zhi-Hong Mao, "High Throughput Finite Field Multipliers Using Redundant Basis for FPGA and ASIC Implementations," IEEE Trans. Circuits and Systems, vol.62, no.1, pp.110-119, January 2015.

[17] W.Geiselmann, J.Muller-Quade, and R.Steinwandt, "On "A New Representation of Elements of Finite Fields GF (2^m) Yielding Small Complexity Arithmetic Circuits," IEEE Trans. Computers, vol. 51, no. 12, pp. 1460-1461, Dec. 2002.

[18] W.Geiselmann and H.Lukhaub, "Redundant Representation of Finite Fields," Proc. Public Key Cryptography, pp. 339-352, 2001.

[19] R.Katti and J.Brennan, "Low Complexity Multiplication in a Finite Field Using Ring Representation," IEEE Trans. Computers, vol. 52, no. 4, pp. 418-426, April 2003.