An Arithmetic System over Finite Fields

Chun-Myoung Park, *Member, KIMICS*

Abstract— **This paper propose the method of constructing the highly efficiency adder and multiplier systems over finite fields. The addition arithmetic operation over finite field is simple comparatively because that addition arithmetic operation is analyzed by each digit modP summation independently. But in case of multiplication arithmetic operation, we generate maximum k=2m-2 degree of α^k terms, therefore we decrease k into m-1 degree using irreducible primitive polynomial. We propose two method of control signal generation for the purpose of performing above decrease process. One method is the combinational logic expression and the other method is universal signal generation. The proposed method of constructing the highly adder/multiplier systems is as following. First of all, we obtain algorithms for addition and multiplication arithmetic operation based on the mathematical properties over finite fields, next we construct basic cell of A-cell and M-cell using T-gate and modP cyclic gate. Finally we construct adder module and multiplier module over finite fields after synthesizing α^k generation module and control signal CSt generation module with A-cell and M-cell. Next, we constructing the arithmetic operation unit over finite fields. Then, we propose the future research and prospects.**

Index Terms— arithmetic operation, finite field, polynomial, control signal, cell, module, adder, multiplieretc.

I. INTRODUCTION

IN many fields of digital logic systems and computer application, the arithmetic operation is important role[1-2]. Specially, in modern time, the multimedia and its application fields necessary to complex arithmetic operation and massive data manipulation. Therefore highly efficiency arithmetic operation and its systems are researched in previous time[3-8]. In specially, the arithmetic operation is effectively analyzed in finite fields fields[9-10].

The finite fields is used to the mathematical background for encryption/decryption, error correcting code, digital image processing, digital signal processing, switching function of digital logic systems etc.

This paper's construction is as following. Section2 discuss the important mathematical properties of finite fields and section3 discuss construct the adder module over finite fields that imply addition algorithm, basic A-cell.

Section4 discuss the multiplier module over finite fields that imply multiplication algorithm, basic M-cell, α^k generation module, control signal CSt generation module, universal control signal CSt generation module. Also,

in section5, we propose a method of arithmetic operation unit over finite fields. Finally, in section6, we summary the proposed arithmetic operation unit finite fields, and we compare proposed method with earlier method. Also we prospect future demand research and prospect.

If we will construct the logical operation unit, it will be able to construct the arithmetic & logical operation unit(ALOU).

Manuscript received July 22, 2011; revised August 5, 2011; accepted August 10, 2011.

Chun-Myoung Park is with the Department of Computer Engineering, Chungju National University, Chungju, 380-702, Korea (Email: cmpark@cjnu..ac.kr)

II. ARITHMETIC

Arithmetic or arithmetics the oldest and most elementary branch of mathematics, used by almost everyone, for tasks ranging from simple day-to-day counting to advanced science and business calculations. It involves the study of quantity, especially as the result of combining numbers.

In common usage, it refers to the simpler properties when using the traditional operations of addition, subtraction, multiplication and division with smaller values of numbers. Professional mathematicians sometimes use the term higher arithmetic when referring to more advanced results related to number theory, but this should not be confused with elementary arithmetic. These artifacts do not always reveal the specific process used for solving problems, but the characteristics of the particular numeral system strongly influence the complexity of the methods. The hieroglyphic system for Egyptian numerals, like the later Roman numerals, descended from tally marks used for counting. In both cases, this origin resulted in values that used a decimal base but did not include positional notation.

Although addition was generally straightforward, multiplication in Roman arithmetic required the assistance of a counting board to obtain the results.. Any set of objects upon which all four arithmetic operations (except division by zero) can be performed, and where these four operations obey the usual laws, is called a field. Addition (+) is the basic operation of arithmetic. In its simplest form, addition combines two numbers, the addends or terms, into a single number, the sum of the numbers.

Adding more than two numbers can be viewed as repeated addition; this procedure is known as summation and includes ways to add infinitely many numbers in an infinite series; repeated addition of the number one is the most basic form of counting. Addition is commutative and associative so the order the terms are added in does not matter. The identity element of addition (the additive identity) is 0, that is, adding zero to any number yields that same number. Also, the inverse element of addition (the additive inverse) is the opposite of any number, that is, adding the opposite of any number to the number itself yields the additive identity, 0 Addition can be given geometrically as follows: If a and b are the lengths of two sticks, then if we place the sticks one after the other, the length of the stick thus formed is a + b. Subtraction (−) is the opposite of addition.

Subtraction finds the difference between two numbers, the minuend minus the subtrahend. If the minuend is larger than the subtrahend, the difference is positive; if the minuend is smaller than the subtrahend, the difference is negative; if they are equal, the difference is zero. Subtraction is neither commutative nor associative. For that reason, it is often helpful to look at subtraction as addition of the minuend and the opposite of the subtrahend, that is $a - b = a + (-b)$. When written as a sum, all the properties of addition hold. There are several methods for calculating results, some of which are particularly advantageous to machine calculation. For example, digital computers employ the method of two's complement. Of great importance is the counting up method by which change is made.

Although the amount counted out must equal the result of the subtraction P − Q, the subtraction was never really done and the value of P − Q might still be unknown to the change-maker. Multiplication is the second basic operation of arithmetic. Multiplication also combines two numbers into a single number, the product.

The two original numbers are called the multiplier and the multiplicand, sometimes both simply called factors. Multiplication is best viewed as a scaling operation. If the real numbers are imagined as lying in a line, multiplication by a number, say x, greater than 1 is the same as stretching everything away from zero uniformly, in such a way that the number 1 itself is stretched to where x was. Similarly, multiplying by a number less than 1 can be imagined as squeezing towards zero.

Multiplication is commutative and associative; further it is distributive over addition and subtraction. The multiplicative identity is 1, that is, multiplying any number by 1 yields that same number. Also, the multiplicative inverse is the reciprocal of any number (except zero; zero is the only number without a multiplicative inverse), that is, multiplying the reciprocal of any number by the number itself yields the multiplicative identity. The product of a and b is written as $a \times b$ or $a \bullet b$. When a or b are expressions not written simply with digits, it is also written by simple juxtaposition: ab. In computer programming languages and software packages in which one can only use characters normally found on a keyboard, it is often written with an asterisk: a * b. Division is essentially the opposite of multiplication. Division finds the quotient of two numbers, the dividend divided by the divisor. Any dividend divided by zero is undefined. For positive numbers, if the dividend is larger than the divisor, the quotient is greater than one, otherwise it is less than one (a similar rule applies for negative numbers).

Ⅲ. INTRODUCTION MATEHEMATICAL PROPERTIES OF FINITE FIELD

In this section, we review the important mathematical properties over finite fields[9-10], these mathematical properties used in build up this paper. Any other mathematical properties except these mathematical properties refer to references.

3.1 Finite Fields

Finite fields is defined by any prime number P and integer m, namely finite fields $GF(P^m)$. In generally finite fields is organized by 5-tuple $\{S, +, \cdot, 0, 1\}$, where S is set of elements, + and · are binary operation over S, 0 and 1 are each identity element for addition and multiplication arithmetic operation. Also finite fields are classified into ground fields GF(P) and extension fields $GF(P^m)$. The number of elements over ground fields GF(P), P is the prime number more than 1, are $\{0,1,2,\ldots\ldots,P-1\}$.

3.2 Important mathematical properties

The important mathematical properties over finite fields are as following.

<P1> Commutative law :
(1) $a+b=b+a$ (2) $a\cdot b=b\cdot a$ $(\forall a,b \in GF(P^m))$

<P2> Associative law :
(1) $a+(b+c)=(a+b)+c$ (2) $a\cdot(b\cdot c)=(a\cdot b)\cdot c$
$(\forall a,b,c \in GF(P^m))$

<P3> Distributive law :
$a\cdot(b+c)= a\cdot b+a\cdot c$ $(\forall a,b,c \in GF(P^m))$

<P4> Zero element **0** exist. $a+\mathbf{0}=\mathbf{0}+a=a$
$(\forall a \in GF(P^m))$

<P5> Unit element **1** exist. $a\cdot\mathbf{1}=\mathbf{1}\cdot a=a$
$(\forall a \in GF(P^m))$

<P6> Inverse element exist.
additive inverse element.
$a+(-\mathbf{a})=0$ multiplicative inverse element .
$a\cdot(\mathbf{a}^{-1})=1$ $(\forall \mathbf{-a}, \mathbf{a}^{-1} \in GF(P^m))$

<P7> $0\cdot a=a\cdot 0=0$ $(\forall a \in GF(P^m))$.

Ⅳ. ADDER

4.1 Addition Algorithm

We put any two element over GF(P^m), $F(\alpha)=\sum_{i=0}^{m-1} a_i\alpha^i$ and $G(\alpha)=\sum_{j=0}^{m-1} b_j\alpha^j$, also $A(\alpha)=\sum_{k=0}^{m-1} A_k\alpha^k$ which is the element after adding them. Then we represent relationship among these elements as following.

$$F(\alpha) + G(\alpha) = \sum_{i=0}^{m-1} a_i\alpha^i + \sum_{j=0}^{m-1} b_j\alpha^j = \sum_{i,j=0}^{m-1} (a_i + b_j)\,\alpha^i = \sum_{k=0}^{m-1} A_k\alpha^k = A(\alpha) \qquad (1)$$

where, a_i, b_j, $A_k \in GF(P)=\{0,1,\ldots,P-1\}$ $(i,j,k=0,1,\ldots,m-1)$, $A_k= a_i + b_j$, $\sum$ and + means modP summation.

Also, we represent above expression (1) to vector space, it is expression (2).

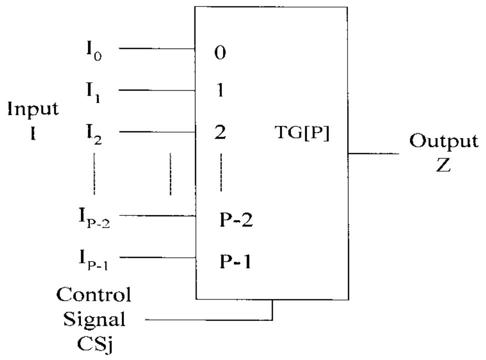
$$\begin{aligned}
F(\alpha)&=\underline{F(\alpha)}=[a_{m-1}, a_{m-2}, \ldots\ldots, a_1, a_0]= \underline{F(\alpha)}[a_V]\\
G(\alpha)&=\underline{G(\alpha)}=[b_{m-1}, b_{m-2}, \ldots\ldots, b_1, b_0]= \underline{G(\alpha)}[b_V]\\
A(\alpha)&=\underline{A(\alpha)}=[A_{m-1}, A_{m-2}, \ldots\ldots, A_1, A_0]= \underline{A(\alpha)}[A_V]\\
F(\alpha) + G(\alpha)&=\underline{F(\alpha)}[a_V] + \underline{G(\alpha)}[b_V]= \underline{A(\alpha)}[A_V]
\end{aligned}$$

Where, a_V, b_V, $A_V \in GF(P)(V=0,1,\ldots, m-1)$ (2)

4.2 Basic A-cell

In order to construct adder, first we construct basic adder cell(A-cell) using data selector T-gate and modP cyclic gate. The following expression (3) represent T-gate operation and fig.1 depict T-gate, expression (4) represent modP cyclic gate operation and fig.2 depict modP cyclic gate.

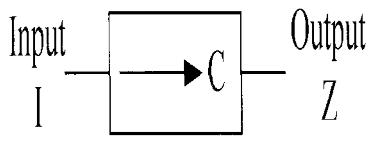
$$Z= I_i \quad \text{iff} \quad I_i= CS_j \qquad (3)$$



Where, I_i, Z, $CS_j \in GF(P)$ and i,j=0,1,......,P-1

Fig. 1. The block diagram of T-gate.

$$Z=I^{\rightarrow C}=(I+C)\ \mathrm{modP} \qquad (4)$$



Where, $1 \le C \le P-1$(C=integer)

Fig. 2. The block diagram of modP cyclic gate.

As we see above contents, because of $A_k=a_i+b_j(i=j=k)$, A_k is obtained as following. The coefficient a_i use as T-gate input after passing modP cyclic gate, also b_j use as T-gate control signal. Therefore we construct A-cell, fig.3, and its characteristic operation is expression (5).

$$A_k=a_i^{\rightarrow b}=(a_i+b_j)\ \mathrm{modP} \qquad (5)$$

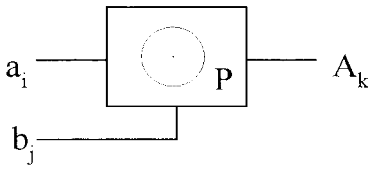


Fig. 3. The block diagram of A-cell.

4.3 Adder module

We construct the adder module(A-module) using above sections. The fig.4 shows block diagram of A-module.

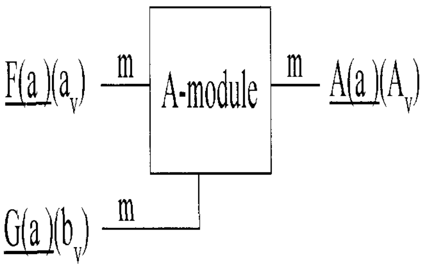


Fig. 4. The block diagram of adder module

V. MULTIPLIER MODULE

There are 2m—2 term of α for any two element multiplication over Galois Fields, that time we convert α term of α^k, $m \le k \le 2m-2$, into less standard basis representation α term less than m-1 degree using irreducible primitive polynomial. Next we obtain the result that multiply two element after sum each α term. We named Mod F(X) for this processing.

[Definition 1] Let δ[(a0, a1, … ,am-2, am-1),(b0, b1, … ,bm-2, bm-1)]=Mk, *mapping function* δ *is binary* operation, δ:GF(P^m)×GF(P^m)→GF(P). Where Mk is the kth product result of (a0, a1, … ,am-2, am-1) and (b0, b1, … ,bm-2, bm-1), and ai, bj∈GF(P)(i,j=0,1,…,m-1) and 0≤k≤2m-2.

Also mapping relationship is decided by selection irreducible primitive polynomial.

5.1 Multiplication algorithm

We put any two element over GF(P^m), $F(\alpha)=\sum_{i=0}^{m-1} a_i\alpha^i$ and $G(\alpha)=\sum_{j=0}^{m-1} b_j\alpha^j$, also $M(\alpha)=\sum_{k=0}^{m-1} M_k\alpha^k$ that is the element after multiply them. Then we represent relationship among these elements as following.

$$F(\alpha)\bullet G(\alpha)=\sum_{i=0}^{m-1} a_i\alpha^i \bullet \sum_{j=0}^{m-1} b_j\alpha^j = a_{m-1}(\sum_{j=0}^{m-1} b_j\alpha^j)\,\alpha^{m-1+j} + a_{m-2}(\sum_{j=0}^{m-1} b_j\alpha^j)\,\alpha^{m-2+j} \ldots + a_1(\sum_{j=0}^{m-1} b_j\alpha^j)\,\alpha^{1+j} + a_0(\sum_{j=0}^{m-1} b_j\alpha^j)\,\alpha^j$$
$$=\sum_{i,j=0}^{2m-2} a_i\, b_j\, \alpha^{i+j} \qquad (6)$$

where, a_i,b_j∈GF(P)(i,j,k=0,1,…,m-1), + and Σ are modP summation, • is mod P product.

As we see the expression (6), we partition α^k term into m≤k1≤2m-2 and 0≤k2≤m-1. This is represent in expression (7).

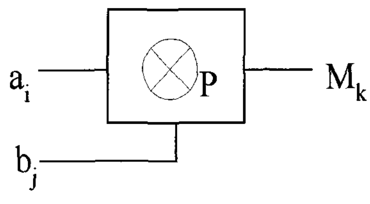
$$F(\alpha)\bullet G(\alpha)=\sum_{k1=m}^{2m-2} a_i\, b_j\, \alpha^{k1}\bullet \sum_{k2=0}^{m-1} a_i\, b_j\, \alpha^{k2}=\sum_{k=0}^{m-1} M_k\alpha^k=M(\alpha) \qquad (7)$$

where, k1= a_i b_j(k1=i+j=m,m+1, … ,2m-2) and k2= a_i b_j(k2=i+j=0,1, … ,m-1)

The other hand, these α^{k1} terms are used in input of control signal CSt.

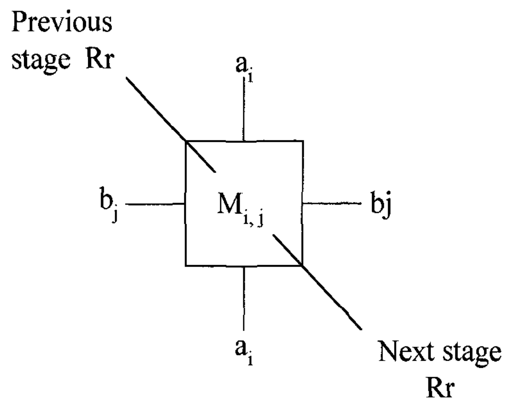
5.2 ModP multiplication gate and M-cell

This section discuss the modP multiplication processing device that is constructed by using T-gate, namely modP multiplication gate, it is depicted in fig.5. And we construct basic M-cell using by modP multiplication gate and adder basic cell A-cell, it is depicted in fig.6.

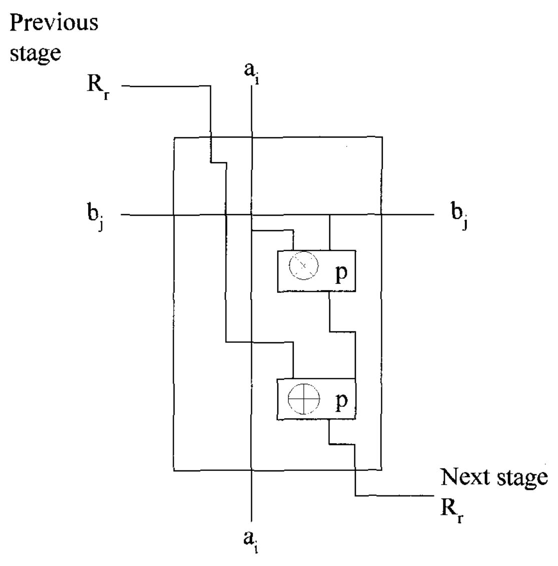


Where, a_i ,b_j , M_k∈GF(P)

Fig. 5. The block diagram of modP multiplication Gate.



(a) symbol



(b) internal circuit

where, a_i,b_j,R_s∈GF(P)

Fig. 6. Basic M-cell.

5.3 α^r generation module

The α^r generation module can be constructed by using M-cell, it is represented in fig.7.

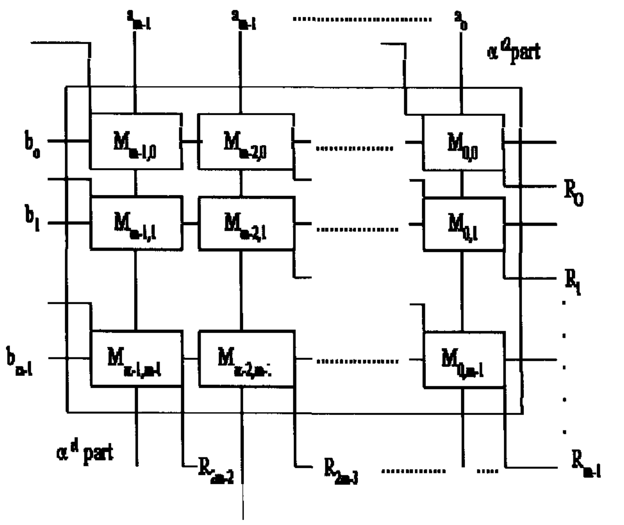


Fig. 7. α^r generation module

5.4 Control signal CSt generation module

The α^k term is generated in $m \le k1 \le 2m-2$ and $0 \le k2 \le m-1$, we can obtain multiplication result between two element using modP sum α^{k2} with result after decrease m-1 degree using irreducible primitive polynomial.

Therefore α^{k1} term is defined according to α^{k2}, we named α^{k2} to control signal CSt(t=0,1,2,...,m-1). This paper propose two algorithm of generating control signal CSt.

5.4.1 Combinational method

[STEP1] we select the proper irreducible primitive polynomial.

[STEP2] we construct basic control digit code $BCD_w(QQQ...Q)$ of α^k. Where w=m,m+1,...,2m-2 and $Q \in GF(P)$.

[STEP3] final control signal CSt is obtained as following. we disregard Q except corresponding R_{k2} and modP sum after each modP multiply. The drawback of this algorithm in according to selected irreducible primitive polynomial. Therefore, in using this algorithm, we select irreducible primitive polynomial type $X^m+(P-1)X^{m-1}+(P-1)X^{m-2}+ ......+ (P-1)X+(P-1)$.

5.4.2 Universal control signal CSt generation module

This proposed algorithm's advantage is usage of any irreducible primitive polynomial. That is not change basic control signal generation module, only input each α term coefficient in change the selected irreducible primitive polynomial.

We named this algorithm as universal control signal CSt generation module. This universal control signal CSt generation module operate modF(X). In order to obtain this function, we input coefficient of irreducible primitive polynomial to shift register, and shift each coefficient to next stage shift register in case of multiply α term in each time.

The fig.8 depicted universal control signal CSt generation module.

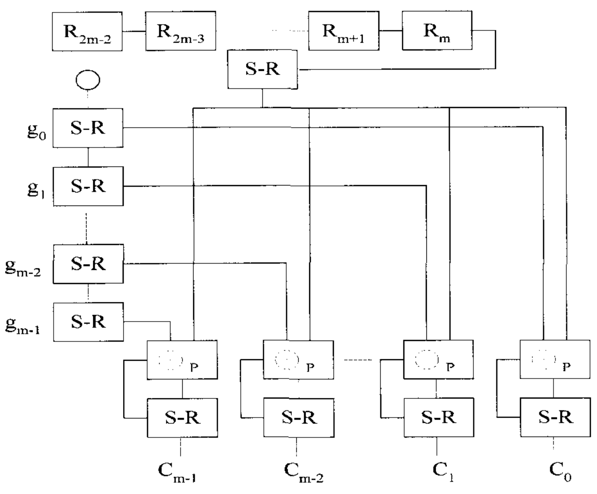


Fig. 8. Universal control signal CSt generation module.

5.5 Multiplier

This section discuss constructing the multiplication module over galois fields. We can construct this multiplication module in merging α^r generation module with control signal generation module CSt. Where, final multiplication result M_k(k=0,1,.....,m-1) between any two element over galois fields obtain R_{r2} of α^r mod P cyclic corresponding to control signal CSt. This is represented in expression (8).

$$M_k = R_{r2}^{\rightarrow CSt} \tag{8}$$

Then, the expression (8) is the same as expression in adder module. Therefore we use the adder module in this part, this block diagram depicted in fig.9.

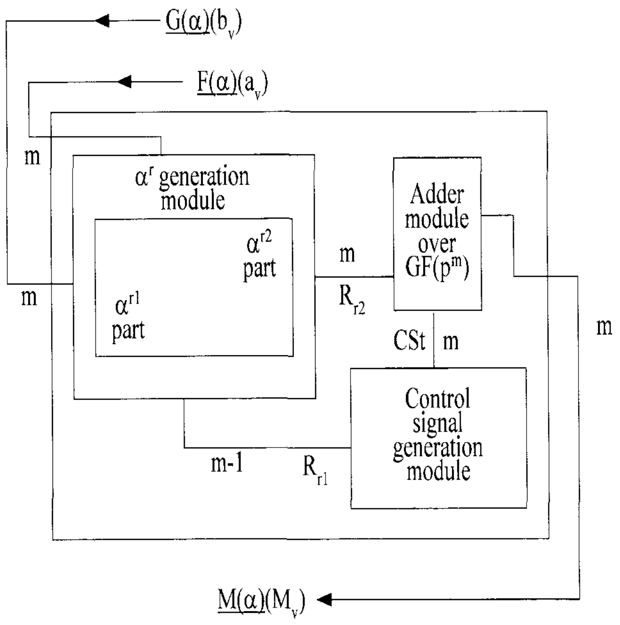


Fig. 9. The multiplier over galois fields.

VI. CONCLUSION

This paper propose the method of constructing the highly efficiency adder and multiplier systems over finite fields. The proposed highly adder/multiplier systems is more regularity, extensibility and modularity than any other research. Also, the proposed highly efficiency adder and multiplier systems is fabricated in VLSI type easily. The future demand research is the other arithmetic operation subtracter and divider, also need to constructing AOU(Arithmetic Operation Unit) in order to processing the four basic arithmetic operation. And we demanded more improvement ALOU(Arithmetic & Logical Operation Unit). The proposed highly efficiency adder and multiplier systems is able to apply modern multimedia hardware systems. The following table 1 represented several item that compare proposed highly adder/multiplier over finite fields with any other research result.

TABLE I
THE COMPARISON TABLE

Comparison item	C.C. Wang[11]	C-Ling etal. [12]	S.T.J. Fenn etal. [13]	This paper
Basis	SDNB	SB	DB	NB
I/O Type	SISO	SIPO	P-I/O	P-I/O
AND	3m	$2m^2$	$2m^2$	2m
OR	2m	$2m^2$	$2m^2$	m
# of control signal	2m-1	2m-1	2m-2	m-1
Overall Type	M-O	S-A	S-A	S-A
Regularity/ Extensibility	●	◎	○	○

Remarks :
SISO : Serial Input Serial Output
SIPO : Serial Input Parallel Output
P-I/O : Parallel I/O, I/O : Input/Output
SDNB : Standard Dual Normal Basis
SB : Standard Basis
NB : Normal Basis, DB : Dual Basis
M-O : Massey-Omura
S-A : Systolic Array
○: Available
◎:some available
●: Disable

ACKNOWLEDGEMENT

The research was supported by a grant from the Academic Research Program of Chungju National University in 2011.

REFERENCES

[1] D. Green, *Modern Logic Design*, Addison-Wesley company, 1986.
[2] K. Hwang, *Computer Arithmetic principles, architecture, and design*, john Wiley & Sons,1979.
[3] H. Wu, M.A. Hasan, I.F. Blake and S. Geo, "Finite Field Multiplier Using Redundant Representation", *IEEE Trans. Comput.*, vol.51, no.11, pp.1306-1316, December 2002.
[4] W. Geiselmann and R. Steinwandt, "A Redundant Representation of GF(q^n) for designing Arithmetic Circuit", *IEEE Trans. Comput.*, vol.52, no.7, pp.848-853, July 2003.
[5] A. Reyhani-Masoleh and M.A. Hsan, "Fast Normal basis Multiplication Using general Purpose Processors," *IEEE Trans. Comput.*, vol.52, no.11, pp.1379-1390, November 2003.
[6] M.E. Kaihara and N. Takagi, "A Hardware Algorithm for Modular Multiplication/Division", *IEEE Trans. Comput.*, vol.54, no.1, pp.12-21, January 2005.
[7] C. Efstathiou, H.T. Vergos and D. Nikolas. "Modified Modulo 2^n-1 Multipliers", *IEEE Trans. Comput.*, vol.53, no.3, pp.370-374, March 2004.
[8] C.H. Wu, C.M. Wu, M.D. shieh and Y.T. Hwang, "High-Speed, Low-Complexity systolic Design of Novel Iterative Division Algorithms in GF(2^m)", *IEEE Trans. Comput.*, vol.53, no.3, pp.375-379, March 2004.
[9] E. Artin, *Galois Theory*, NAPCO Graphics arts, Inc., Wisconsin.1971.
[10] R. Lidi and H. Niederreiter, *Introduction to finite fields and their applications*, Cambridge University Press,1986.
[11] C.C. Wang," an algorithm to design finite field multipliers using a self-dual normal basis", *IEEE Trans. Comput.*, vol.38 ,no.10, pp.1457-1460, Oct.1989.
[12] C. Ling and J. Lung," Systolic array implementation of multipliers for finite fields GF(2^m)", *IEEE Trans. Cir. & Sys.*, vol.38, no.7, pp.796-800, Jul.1991.
[13] S.T.J.Fenn, M.Benaissa and D.Taylor," GF(2^m) multiplication and Division over dual basis", *IEEE Trans. Comput.*, vol.45, no.3, pp.319-327, Mar.1996.

Chun-Myoung Park received the B.S, M.S and Ph.D. degree from electronic engineering, Inha University, Incheon, Korea, in 1983, 1986 and 1994 respectively. He joined the faculty at Chungju National University at 1995, where he is currently a professor in the Dept. of computer engineering. From 2002 to 2003, he was a visiting scholar at UCI in USA. His research interests include the next generation digital logic systems & computer architecture, embedded computer systems, ubiquitous computing systems, e(U)-Learning , IT application etc.