International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:10, 2007

Implementation and Analysis of Elliptic Curve Cryptosystems over Polynomial basis and ONB

Yong-Je Choi, Moo-Seop Kim, Hang-Rok Lee and Ho-Won Kim

***Abstract*—** Polynomial bases and normal bases are both used for elliptic curve cryptosystems, but field arithmetic operations such as multiplication, inversion and doubling for each basis are implemented by different methods. In general, it is said that normal bases, especially optimal normal bases (ONB) which are special cases on normal bases, are efficient for the implementation in hardware in comparison with polynomial bases. However there seems to be more examined by implementing and analyzing these systems under similar condition. In this paper, we designed field arithmetic operators for each basis over GF(2^{233}), which field has a polynomial basis recommended by SEC2 and a type-II ONB both, and analyzed these implementation results. And, in addition, we predicted the efficiency of two elliptic curve cryptosystems using these field arithmetic operators.

***Keywords*—** Elliptic Curve Cryptosystem, Crypto Algorithm, Polynomial Basis, Optimal Normal Basis, Security.

I. Introduction

ELLIPTIC Curve Cryptosystems (ECCs) were proposed in 1985 by Neal Koblitz[3] and Victor Miller[4]. The security of ECC is based on the discrete logarithm problem over the points on an elliptic curve. The discrete logarithms problem of ECC is known as more difficult problem than the prime number factorization problem of RSA algorithm. The security problems of cryptosystems, such as the discrete logarithms and prime factorization, are closely related to the key length of cryptosystems. If the security problem is more difficult then smaller key length can be used with sufficient security. In fact, it is known that the elliptic curve cryptosystem over GF(2^{233}) provides same security as of 2,048-bit RSA cryptosystem[1]. This smaller key length makes ECCs suitable for practical applications such as embedded systems and wireless applications.

ECCs in general are implemented over prime fields or binary fields. Arithmetic operations used in prime fields are similar to arithmetic operations used in RSA cryptosystems. For binary fields they are different according to bases used. Two of the most common bases used in binary fields are the polynomial basis and the normal basis. Any bases in both can be used for ECCs, however, some special cases such as trinomial bases, pentanomial bases and optimal normal bases (ONBs) are, in practice, used for the purpose of efficient operations. The ONBs especially are known to be more efficient for hardware implementation than polynomial bases because the multiplication operation can be performed very efficiently and inversion can be achieved by repeated multiplication typically using the method of Itoh and Tsujii and doubling can be executed by only one cyclic shift operation. However these facts seem to be more examined by implementing analyzing these systems under similar condition.

Manuscript received November 15, 2005

Y.J. Choi, M.S. Kim, H.R Lee and H.W. Kim are with Electronics and Telecommunications Research Institute (ETRI), 161 Gajeong-Dong Yuseong-Gu, DaeJeon, 305-350, KOREA, Tel : +82-42-860-1327, / FAX : +82-42-860-5611, e-mail : {choiyj,gomskim,neogauss,khw}@etri.re.kr

For this purpose, we designed field arithmetic operators for each basis over GF(2^{233}), which field has a polynomial basis recommended by SEC2[3] and a type-II ONB both, and analyzed these implementation results. And, in addition, we also predicted the efficiency of two elliptic curve cryptosystems using these field arithmetic operators.

II. Implementation of Finite Field Arithmetic Operations

As we said upper, we will deal with two bases which are polynomial basis and normal basis. Polynomial basis representation over GF(2^m) is as follow.

$$\{\alpha, \alpha^2, \alpha^3, \cdots, \alpha^{m-1}\},\ \alpha \in GF(2^m) \tag{1}$$

And, normal basis representation over GF(2^m) is as follow.

$$\{\beta, \beta^2, \beta^{2^2}, \cdots, \beta^{2^{m-1}}\},\ \beta \in GF(2^m) \tag{2}$$

The addition over GF(2^m) is simply logical XOR operation of the corresponding coefficients regardless of basis. Arithmetic operations for elliptic curve cryptosystems such as multiplication, squaring and division except addition are achieved by different methods according to bases used. In this chapter, we will describe operation methods in each basis and its implementation.

A. Multiplication

Multiplication in polynomial basis is achieved to two steps. The first step is to multiply two binary polynomials together. Then it yields a product polynomial degree up to 2m-2. Next, this result should be reduced modulo the irreducible polynomial degree of m.

As multiplication methods in polynomial basis, there are bit-serial methods [5, 8] and bit-parallel methods [6, 7, 9]. The bit-serial methods can be implemented with a small size of hardware, but it must repeat operations more than m times and these iterations reduce the system performance. On the other hand, the bit-parallel method may expect a high performance, but according to enlargement of the degree m, its hardware area increases asymptotically with m^2. For example, a 233-bit parallel multiplier of Sunar and Koc[6] requires 54,288 XOR gates and 54,289 AND gates, and it is too large to implement on a general commercial FPGA.

For this reason, we proposed a hybrid multiplier for GF(2^m) defined by an irreducible trinomial $x^m + x^n + 1$ ($n \le m/2$) in [11], which can be constructed in variable structures depending on the performance-area trade-off. The structure of it is shown in a following picture 1. The proposed hybrid multiplier requires km AND gates, km+2k-1 XOR gates and $T_X + \lceil m/k \rceil (\max\{T_A, T_X\} + \lceil \log_2(k+1) \rceil T_X)$ delays, where T_A is an AND gate delay and T_X is a XOR gate delay.

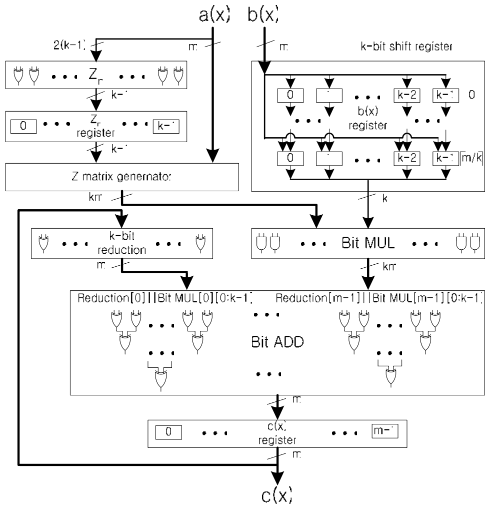

Fig. 1 Structure of the hybrid polynomial multiplier

Multiplication in a general normal basis can be achieved by Massey-Omura multiplication method[13]. For ONB, which has special cases so called Type I and Type II and performs efficiently multiplication, it is implemented some different way. Multiplication for Type I ONB is achieved by Modified Massey-Omura multiplication method. For Type II ONB, GF(2^{233}) of this paper is applicable to this case, Sunar and Koc [12] proposed a parallel multiplier optimized for area. Sunar-Koc's multiplication method is executed as follows. It firstly changes the bases of two multiplication elements to canonical bases for efficient calculation. And then it multiplies two elements changed in canonical basis. After multiplication, the result is converted back to the normal basis. Because basis conversion is implemented by only logic wiring, it needs no additional logics.

A hybrid multiplier for Type II ONB was proposed by Wu-Hasan in [9]. After basis conversion as like [12], they extend degree of the multiplier b(x) to 2m-1 and save it at the register. The multiplier b(x) saved and the multiplicand a(x) is performed bit-wise operations as like following picture 2. The operation logic of the picture outputs one bit among multiplication results and this operation logic can be used in parallel as many as need for whole multiplication operation depending on the performance-area trade-off.

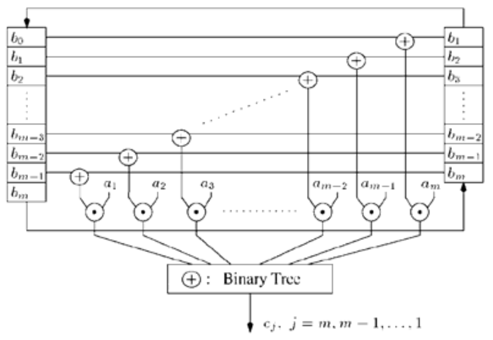

Fig. 2 Bit operation block of the Wu-Hasan multiplier

The following picture 3 is the structure of the hybrid multiplier for Type II ONB. This is implemented with registers to save inputs and results, (2m-1)k XOR gates, and km AND gates, and its operation time is $\lceil m/k \rceil (T_A + (1 + \lceil \log_2 m \rceil) T_X)$.

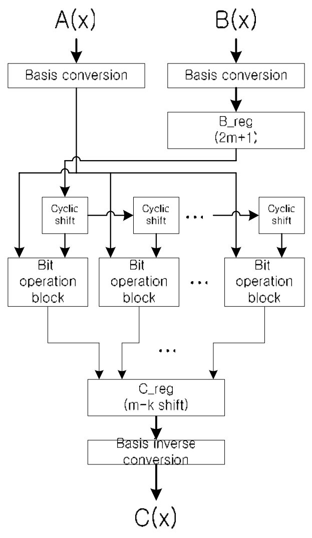

Fig. 3 structure of the hybrid multiplier for Type II ONB

B. Squaring

Squaring in polynomial basis can be performed using a multiplication block by inputting the same value at two inputs. Squaring, however, is used frequently in elliptic curve operations, therefore it using the multiplication block slows down the crypto systems. It in polynomial basis is achieved by bit extension and reduction. Bit extension is performed by inserting a '0' bit between consecutive bits of the binary representation of the input value. If an irreducible polynomial is not changed, then these procedures can be precomputed. In GF(2^{233}) with irreducible polynomial $x^{233} + x^{74} + 1$, squaring is executed with 153 XOR gates and its operation time is T_X. That is to say, squaring in polynomial basis is implemented in very small area with high performance.

As we said before, squaring in ONB can be achieved by only a cyclic shift operation. In hardware implementation, a cyclic shift operation is implemented by logic wiring with no additional logics.

C. *Inversion*

As inversion algorithms for polynomial bases, Extended Euclidean algorithm (EEA) and Almost Inverse Algorithm (AIA) are commonly used [6]. In software implementation, the running time of EEA is similar to AIA. AIA, however, is more desirable for hardware implementation because operations are calculated bit by bit and post-operations are predicted efficiently, although AIA needs separate operation multiplying x^{-k}. In MAIA (Modified AIA), this inverse reduction step is modified to perform directly. We enhanced the MAIA in point of parallel operation and prediction of post-operations at [10]. The following picture 4 is the structure of the inversion operator. Its operation time is about 250 cycles, which may be changed by input value.

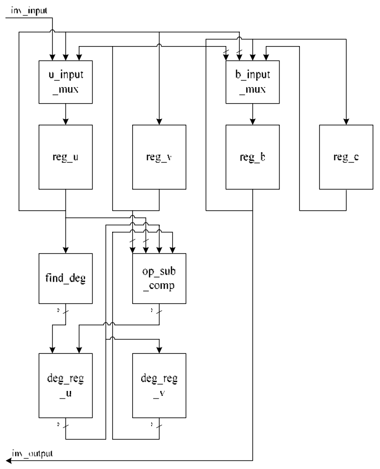

Fig. 4 structure of the inversion operator

Inversion in normal bases is achieved by Itoh and Tsujii algorithm. IT algorithm iterates multiplication and squaring. However squaring is only a shift operation; therefore it uses only multiplication logics in hardware implementation. It was enhanced to reduce the number of iterations by Chang and Takigi but it is applicable for some special cases. So in this paper we used IT algorithm for inversion. Picture 5 is the structure of the inversion operator. As you can see the picture, ONB inversion hardware is easily implemented by adding some multiplexers and registers to the multiplication logic. This ONB operator can achieve both multiplication and inversion according to a mode value. In GF(2^{233}), inversion needs 10 iterations of multiplication.

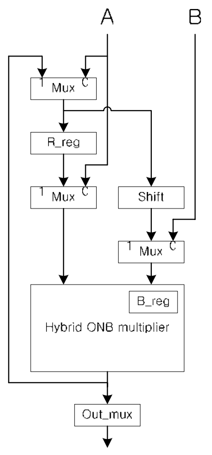

Fig. 5 structure of the inversion operator for ONB

III. Analysis of Implementation Results and Performance Prediction of ECC Systems

In this chapter, we will analyze the arithmetic operators implemented above and predict the performance of elliptic curve cryptosystems according to basis. Arithmetic operators in GF(2^{233}) is synthesized and verified at a xilinx virtex-2 8000 FPGA which is going to be used for our ECC system.

A. *Analysis of implementation results*

In implementing hardware not only logic delay but also routing delay affects performance. The routing delay is important especially at high-performance public key cryptosystems, which need operators to calculate over a hundred bit at the same time. In elliptic curve cryptosystems the multiplication logic dominates the hardware area. Table 1 shows the synthesis results of our hybrid polynomial multiplier. (The result may be changed minutely according to synthesis environment.) We examined the multiplier changing the operation bit from 59 to 233. When the operation bit is 59, it needs 4 operation clocks. As you can see in the table, according as the hardware area increases, the routing delay dominates the hardware performance.

TABLE I
SYNTHESIS RESULTS OF OUR HYBRID POLYNOMIAL MULTIPLIER

Operation Bit (operation cycle)	Synthesis Result Delay (ns)	Area (slices)	Theoretical Delay
233 (1 clock)	24	28,000	$\mathrm{Max}\{T_A, T_X\} + 9T_X$
117 (2 clocks)	20	13,300	$\mathrm{Max}\{T_A, T_X\} + 8T_X$
59 (4 clocks)	18	5,100	$\mathrm{Max}\{T_A, T_X\} + 6T_X$

Synthesizing our polynomial inversion hardware at FPGA, its hardware area is 2,124 slices and its operation time is below 19ns, so it can be operated over 50MHz. Squaring hardware is synthesized in 108 slices, and addition hardware is 84 slices.

As for ONB, addition logic is 108 slices like polynomial and arithmetic operator, which can perform both multiplication and inversion, is synthesized in 18,800 slices. At this time its operation bit is 117-bit. Only 117-bit ONB multiplier is implemented in 17,600 slices, therefore about 1,100 slices is added for inverse operation. The maximum delay is 21ns and its operation cycle is 2 clocks for multiplication and 30 clocks for inversion.

In comparison with a polynomial multiplier and an ONB multiplier, we can see that the former is a little superior to the later in the point of view of area and performance. This fact can be verified by observing the area and delay needed theoretically. Table 2 shows these comparisons.

TABLE 2 COMPARISON WITH A POLYNOMIAL MULTIPLIER AND AN ONB MULTIPLIER

	117 Bit		59 Bit	
	Poly	ONB	Poly	ONB
XOR Gate	27,494	54,405	13,629	27,435
AND Gate	27,261	27,261	13,514	13,514
Operation Time	$2\max\{T_A, T_X\} + 15T_X$	$2T_A + 18T_X$	$4\max\{T_A, T_X\} + 25T_X$	$4T_A + 36T_X$

B. Performance prediction of elliptic curve cryptosystems

Not only field arithmetic operators but also coordinates for elliptic curve operations and scalar multiplication methods dominate cryptosystem's performance. Two of the most common coordinates used in ECC are the affine coordinate and the projective coordinate. Scalar multiplication is the operation to compute kP, where k is a random integer and P is an elliptic curve point, and it can be defined the combination of additions of two points on an elliptic curve.

The most basic implementation method of ECC is to use an affine coordinate and a binary scalar multiplication method. The addition of two points on an elliptic curve over an affine coordinate is defined as following algorithm 1. (The elliptic curve over GF(2^m) given by the equation $y^2 + xy = x^3 + ax^2 + b$ and P1 and P2 are the points on the elliptic curve). This elliptic curve addition needs two field multiplications and one inversion.

```
Algorithms 1. Point Addition Equation in affine coord.

Input : $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$.
Output : $P_3 = P_1 + P_2 = (x_3, y_3)$.
1. If $P_1 = P_2$ (doubling)
      $x_3 = \lambda^2 + \lambda + a$,  $y_3 = x_1^2 + (\lambda + 1) x_3$
      where ($\lambda = x_1 + y_1 / x_1$)
2. Else if $P_1 \neq P_2$ (point addition)
      $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$,
      $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$
      where ($\lambda = (y_2 + y_1) / (x_2 + x_1)$)
3. Return ($x_3$, $y_3$)
```

The binary scalar multiplication method is shown in algorithm 2.

```
Algorithm 2. Binary method for scalar multiplication
Input : $k = (k_{t-1}, \dots, k_2, k_1, k_0)_2$, $P \in GF(2^m)$.
Output : kP.
1. Q ← O.
2. For i from t –1 downto 0 do
      2.1 Q ← 2Q.
      2.2 If $k_i = 1$ then Q ← Q + P.
3. Return (Q)
```

In the case of implementing an elliptic curve system using 117-bit hybrid multipliers and binary scalar multiplication method, its hardware area and operation time are as following table 3.

TABLE 3 COMPARISON OF ECC SYSTEMS USING AFFINE COORDINATE

	Area (slices)				Operation Time (App.)
	MUL	INV	ETC (App.)	Total (App.)	
Polynomial Basis	13,300	2,214	1,500	17,000	88,500 clocks
Optimal Normal Basis	18,800		1,200	20,000	11,500 clocks

Comparing the hardware areas, the elliptic curve cryptosystems for polynomial basis is implemented by small area than that of ONB. However considering the performance, ONB is far superior to polynomial system above table. It is caused because of difference of inversion performance of the two bases.

For high-performance system, projective coordinate and scalar multiplication methods using pre-computation are used. In projective coordinate, multiplication operation numbers are increased but inversion is achieved only one time. Because multiplication is very faster than inversion in hardware and multiplications can be achieved in parallel, the ECC system over projective coordinate is suited for the high performance cryptosystem. The addition of two points on an elliptic curve over a projective coordinate is defined as following algorithm 3. The results calculated in projective coordinate should be converted back to the affine coordinate. This operation is done by x/z, y/z^2. At this time, inversion is used only one.

```
Algorithms 3. Point Addition Equation in projective coordinate.
Input : $P_0 = (x_0, y_0, z_0)$, $P_1 = (x_1, y_1, 1)$.
Output : $P_2 = P_0 + P_1 = (x_2, y_2, z_2)$.
1. If $P_0 = P_1$ (doubling)
      $Z_2 = Z_0^2 * X_0^2$,
      $X_2 = X_0^4 + b * Z_0^4$,
      $Y_2 = b * Z_0^4 * Z_2 + X_2 * (a * Z_2 + Y_0^2 + b * Z_0^4)$.
2. Else if $P_0 \neq P_1$ (point addition)
      $A = Y_1 * Z_0^2, + Y_0$,          $B = X_1 * Z_0 + X_0$,
      $C = Z_0 * B$,                     $D = B^2 * (C + a * Z_0^2)$,
      $Z_2 = C^2$,                       $E = A * C$,
      $X_2 = A^2 + D + E$,               $F = X_2 + X_1 * Z_2$,
      $G = X_2 + Y_1 * Z_2$,             $Y_2 = E * F + Z_2 * G$,
3. Return ($x_3$, $y_3$, $z_3$)
```

Using two multipliers in parallel, the operation flow for elliptic curve addition over projective coordinate is shown in picture 6. This flow is applicable equally for both polynomial basis and ONB. If we use binary scalar multiplication method with two multipliers, then expectation results of hardware performance and area will be as following table 4. As you can see, we will expect that the difference of the ECC systems of two bases is slight if both performance and area are considered. Performance difference may be from the difference of only one inversion operation of each basis.

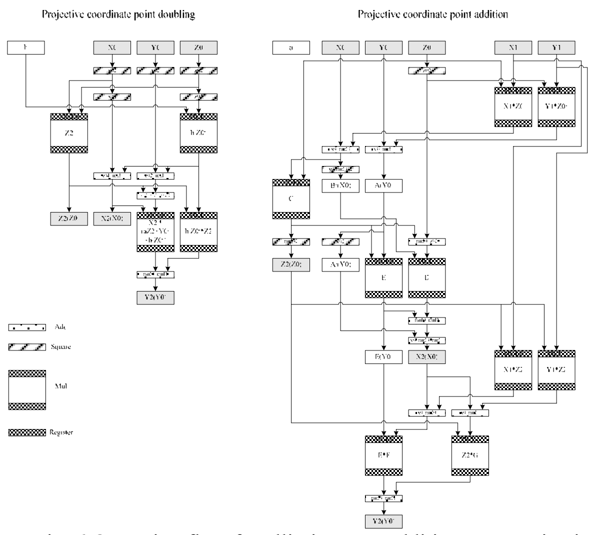

Fig. 6 Operation flow for elliptic curve addition over projective coordinate

TABLE 4 COMPARISON OF ECC SYSTEMS USING PROJECTIVE COORDINATE

	Area (slices)				Operation Time (App.)
	MUL	INV	ETC (App.)	Total (App.)	
Polynomial Basis	26,600	2,214	5,000	34,000	3,420 clocks
Optimal Normal Basis	18,800 + 17,700		3,000	39,500	3,200 clocks

IV. Concluding Remarks

In this paper, we designed field arithmetic operators for polynomial basis and ONB over GF(2^{233}) and analyzed these implementation results. On the basis of results we predicted the efficiency of two elliptic curve cryptosystems using these field arithmetic operators.

When we compared the hybrid polynomial multiplier with the hybrid ONB multiplier under similar performance, the former was implemented with smaller hardware than the latter. As for inversion, ONB inversion operator had merits of area efficiency and high performance. Although ONB square was achieved only cyclic shift with no addition logics, polynomial square was also implemented with very small area and high performance, therefore it seems that the difference of the two is slight. If we design an elliptic curve cryptosystem using these arithmetic operators for each basis, the performance difference of the two systems will be caused by the performance difference of inversion operators of each basis. So, in affine coordinate which needs many inversion operations, the performance of the ONB cryptosystem will be superior to the other. For example, a polynomial ECC system using affine coordinate over GF(2^{233}) will be needed more about 76,000 clocks than ONB system. But, using projective coordinate, inversion is needed only one time, so it seems that the performance of the two cryptosystems will be similar and area efficiency of polynomial basis will be better than ONB. In future work, we are going to implement these elliptic curve systems using the above operators and verify the prediction of this paper.

References

[1] Certicom research , The Elliptic Curve Cryptosystem, Certicom, April 1997.

[2] Darrel Hankerson, Julio Lopez Hernandez, Alfred Menezes, Software Implementation of Elliptic Curve Cryptography over Binary Fields, CHES 2000, page 1-24. 2000.

[3] Certicom research, "SEC 2 : Recommended Elliptic Curve Domain Parameters", October 1999.

[4] Richard Schroeppel, Hilarie Orman, Sean O'Malley, "Fast Key Exchange with Elliptic Curve Systems", TR-95-03(Tucson, AZ: University of Arizona, Computer Sciences Department, 1995)

[5] Mastrovito, E. D. : 'VLSI architectures for computations in Galois fields'PhD Thesis, Linkoping University, Department of Electrical Engineering, Linkoping, Sweden, 1991.

[6] Sunar, B. and Koc, C. K.: 'Mastrovisto multiplier for all trinomials', IEEE Trans. Comput. 1999, 48, (5), pp. 522-527.

[7] Wu, H. : 'Bit-parallel finite field multiplier and square using polynomial basis', IEEE Trans. Comput., 2002, 51, (7), pp. 750-758.

[8] Chiou, C. W, Chou F. H. and Shu S. F : 'Low-complexity finite field multiplier using irreducible trinomials', Electron. Lett., 2003, 39, (24), pp. 1709-1711

[9] Huapeng Wu, Anwar Hasan, " Finite Field Multiplier Using Redundant Representation" , IEEE Transactions on Computers, Vol 51, No 11, November 2002 .

[10] HoWon Kim, Thomas Wollinger, YongJe Choi, Kyoil Chung, and Christof Paar, "Hyperelliptic Curve Coprocessors on a FPGA," The 5th International Workshop on Information Security Applications (WISA 2004), Aug. 23-25, 2004, JeJu, Korea (LNCS: SCI-E)

[11] Y.J. Choi, K.-Y. Chang, D.W. Hong and H.S. Cho, "Hybrid multiplier for GF(2m) defined by some irreducible trinomials" Electronics Letter, Volume 40 852-853, Number 14, 8th July 2004.

[12] C. K. Koc, and B. Sunar, "Low-Complexity Bit-Parallel Canonical and Normal Basis Multipliers for a Class of Finite Fields", IEEE Trans on Comp. Vol 47, No 3, March 1998.

[13] J. Omura and J. Massey, "Computational Method and Apparatus for Finite Field Arithmetic" U.S. Patent Number 4,587,627