Information Technology Journal 5 (4): 742-748, 2006
ISSN 1812-5638

High Performance Elliptic Curve GF(2^m) Crypto-processor

[1]Turki F. Al-Somani, [2]M.K. Ibrahim and [1]Adnan Gutub
[1]Department of Computer Engineering,
King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia
[2]School of Engineering and Technology, De Montfort University, Leicester LE19BH, UK

Abstract: This study presents a high performance elliptic curve cryptoprocessor architecture over GF(2^m). The proposed architecture exploits parallelism at the projective coordinate level to perform parallel field multiplications. Comparisons between the Projective, Jacobian and Lopez-Dahab coordinate systems using sequential and parallel designs are presented. Results show that parallel designs gives better area-time complexity (AT^2) than sequential designs by 44-252% which leads to a wide range of design tradeoffs. The results also show that the Projective coordinate system gives the best AT^2 in parallel designs with the least number of multiplications levels when using 4 multipliers.

Key words: Elliptic curves cryptosystems, projective coordinate, parallel designs, normal basis

INTRODUCTION

Recently, Elliptic Curves Cryptosystems (ECC) (Koblitz, 1987; Menezes, 1993) has attracted many researchers and has been included in many standards (ANSI, 1998; IEEE, 2000; NIST, 2000; SEC, 2000a; SEC, 2000b). ECC is evolving as an attractive alternative to other public-key schemes such as RSA by offering the smallest key size and the highest strength per bit. Extensive research has been done on the underlying math, security strength and efficient implementations. Among the different fields that can underlie elliptic curves, prime fields GF(p) and binary fields GF(2^m) have shown to be best suited for cryptographic applications. In particular, binary fields allow for fast computation in software as well as in hardware. Small key sizes and computational efficiency make ECC not only applicable to hosts processing security protocols over wired networks, but also to small wireless devices such as cell phones, PDAs and Smartcards.

Inversion operations, which are needed in point addition over Elliptic Curves are the most expensive operation over Finite Fields (Blake, 1999; Gutub, 2002; 2003a, b). The approach adopted in the literature is to represent elliptic curve points in projective coordinate systems in order to replace the inversion operations with repetitive multiplications (Blake, 1999; Gutub, 2002; 2003a,b). Recently, several ECC processors have been proposed in the literature based on projective coordinate representation. There are many projective coordinate systems to choose from. In exiting architectures, the selection of a projective coordinate is based on the number of arithmetic operations, mainly multiplications. This is to be expected due to the sequential nature of these architectures where a single multiplier is used.

For high performance servers, such sequential architectures are too slow to meet the demand of increasing number of users. For such servers, high-speed crypto processors are becoming crucial. One solution for meeting this requirement is to exploit the inherent parallelism within Elliptic curve point operations in projective coordinate systems. Recently, ECC processor architectures have been proposed where the choice of the projective coordinate system used also depends on its inherent parallelism (Gutub 2003a,b). Since multiplication is the most dominant operation and most time consuming when computing point operations in projective coordinate, three multipliers that can work in parallel are used in the architectures (Gutub 2003a,b). These architectures give better area-time complexity (AT^2) than the architectures that are based in a single multiplier. The lower bound on the requirements of resources, area-time, is usually included in the performance metric (area) x $(\text{time})^{2\alpha}$, $0 \leq \alpha \leq 1$, where the choice of α determines the relative importance of area and time. Such lower bounds have been obtained for several problems, for example, discrete Fourier transform, matrix multiplication, binary addition and others (Thompson, 1980). Once the lower bound on the chosen performance metric is known, the designer attempts to devise an algorithm and a corresponding design which is optimal for a range of values of area and time. Even though a design might be

Corresponding Author: Turki F. Al-Somani, Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

optimal for a certain range of values of area and time, it is nevertheless of some interest to obtain a design for minimum values of time. In order to make a more meaningful comparison of the cost effectiveness of the proposed designs, AT^2 measure is used which is a better measure of how fast a design can compute the result.

In this study we are proposing an alternative parallel design using normal basis representation which is more suitable for hardware implementations. In addition, the complexity and parallelism in several homogenous and heterogeneous projective coordinate systems are given.

GF(2^m) ARITHMETIC BACKGROUND

The finite GF(2^m) field has particular importance in cryptography since it leads to particularly efficient hardware implementations. Elements of the field are represented in terms of a basis. Most implementations use either a Polynomial Basis or a Normal Basis (Lidl, 1994). For the implementation described in this paper, a normal basis is chosen since it leads to more efficient hardware implementations. Normal basis is more suitable for hardware implementations than polynomial basis since operations are mainly comprised of rotation, shifting and exclusive-OR operations which can be efficiently implemented in hardware. A normal basis of *GF*(2^m) is a basis of the form

$$(\beta, \beta^2, \beta^4, \beta^8, \ldots\ldots \beta^{2^{(m-1)}}), \text{ where } \beta \in GF(2^m)$$

In a normal basis, an element $A \in GF(2^m)$ can be uniquely represented in the form

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i},$$

where $a_i \in \{0, 1\}$.

GF(2^m) operations using normal basis are performed as follows:

Addition and subtraction: Addition and subtraction are performed by a simple bit-wise exclusive-OR (XOR) operation.

Squaring: Squaring is simply performed by a rotate left operation.

Multiplication: $\forall A, B \in GF(2^m)$, where

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i}$$

and

$$B = \sum_{i=0}^{m-1} b_i \beta^{2^i},$$

the product C = A*B, is given by:

$$C = A * B = \sum_{i=0}^{m-1} c_i \beta^{2^i}$$

then multiplication is defined in terms of a multiplication table λ_{ij} $\{0, 1\}$

$$C_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_{ij} a_{i+k} b_{j+k} \qquad (2.1)$$

An Optimal Normal Basis (ONB) (Mullin *et al.*, 1988) is one with the minimum number of terms in (2.1), or equivalently, the minimum possible number of nonzero λ_{ij}. This value is 2m-1 and since it allows multiplication with minimum complexity, such a basis would normally lead to more efficient hardware implementations.

Inversion: February 16, 2006 Inverse of $a \in GF(2^m)$, denoted as a^{-1}, is defined as follows.

$$Aa^{-1} = 1 \bmod 2^m$$

Most inversion algorithms used are derived from Fermat's Little Theorem:

$$a^{-1} = a^{2^m-2} = (a^{2^{m-1}-1})^2$$

for all $a \neq 0$ in GF(2^m).

ELLIPTIC CURVES

Here we present a brief introduction to elliptic curves. Let GF(2^m) be a finite field of characteristic two. A non-supersingular elliptic curve E over GF(2^m) is defined to be the set of solutions $(x, y) \in GF(2^m) \times GF(2^m)$ to the equation,

$$y^2 + xy = x^3 + ax^2 + b,$$

where a and $b \in GF(2^m)$, $b \neq 0$, together with the point at infinity denoted by O. It is well known that E forms a commutative finite group, with O as the group identity, under the addition operation known as the tangent and chord method. Explicit rational formulas for the addition rule involve several arithmetic operations (adding, squaring, multiplication and inversion) in the underlying finite field. In affine coordinate system, the elliptic group operation is given by the following.

Let $P = (x_1, y_1) \in E$; then $-P = (x_1, x_1 + y_1)$. For all $P \in E$, O+P = P+O = P. If $Q = (x_2, y_2) \in E$ and $Q \neq -P$, then P+Q = (x_3, y_3),

where

$$x_3 = (\frac{y_1 + y_2}{x_1 + x_2})^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a$$

$$y_3 = (\frac{y_1 + y_2}{x_1 + x_2}) \cdot (x_1 + x_3) + x_3 + y_1$$

if $P \neq Q$ and

$$x_3 = x^2_1 + \frac{b}{x^2_1}$$

$$y_3 = x^2_1 + (x_1 + \frac{y_1}{x_1})x_3 + x_3$$

if P = Q.

Computing P+Q is called elliptic curve point addition if $P \neq Q$ and is called elliptic curve point doubling if P = Q. Scalar multiplication is the basic operation for ECC. Scalar multiplication in the group of points of an elliptic curve is the analogous of exponentiation in the multiplicative group of integers modulo a fixed integer m. Computing dP can be done with the straightforward double-and-add approach based on the binary expression of $d = (d_{l-1},\ldots,d_0)$ where d_{l-1} is the most significant bit of d. However, several scalar multiplication methods have been proposed in the literature. A good survey is presented by Gordon (1998).

PROJECTIVE COORDINATE SYSTEMS IN GF(2^m)

The projective coordinate systems are to eliminate the need for performing inversion. For elliptic curve defined over GF(2^m), many different forms of formulas are found (Blake, 1999; Lopez, 1998) for point addition and doubling. The projective coordinate system (Pr), so called homogeneous coordinate system, takes the form (x,y) = (X/Z,Y/Z), while the Jacobian coordinate system takes the form $(x,y) = (X/Z^2,Y/Z^3)$ and the Lopez-Dahab coordinate system takes the form $(x,y) = (X/Z,Y/Z^2)$. Table 1 demonstrates only the multiplications needed in the Projective, Jacobian and Lopez-Dahab coordinate system since other field arithmetic operations requires negligible time as compared to multiplication. This is because of the nature of normal basis over GF(2^m) which performs addition and subtraction simply by an XOR operation and performs squaring by a single rotation.

ECC CRYPTO-PROCESSOR ARCHITECTURE

Generic ECC Crypto-processor architecture with Multi-multipliers: The basic idea is based on the parallelism of projective coordinate systems multiplications proposed by Gutub (2003a, b). Three multipliers were employed to provide parallelism to provide better (AT^2). The study reported by Gutub (2003a, b) was represented in polynomial basis and squaring was considered to be a multiplication, which can be negligible in normal basis. This makes a big difference in the number of multiplication cycles as is discussed in the next section. The proposed generic crypto-processor architecture uses 2-4 multipliers, a cyclic shift register to perform squaring, an XOR unit for field addition and a register file. Only one cyclic shift register and XOR unit is used since both squaring and field addition requires only one clock cycle and hence it can be reused several times while a single multiplication operation is computed. Each of these arithmetic units can get operands from the register file and store the result in the register file. The controller generates control signals for all the arithmetic units and the register file (Fig. 1).

Table 1: Multiplications within different coordinate systems

Projective coordinate (Pr)				Jacobian coordinate (J)				Lopez-Dahab coordinate			
Addition		Doubling		Addition		Doubling		Addition		Doubling	
$A = X_1Z_2$	1M	$A = X_1Z_1$	1M	$A = X_1Z_2^2$	1M	$Z_3 = X_1Z_1^2$	1M	$A_0 = Y_1^2Z_1^2$	1M	$Z_3 = Z_1^2X_1^2$	1M
$B = X_2Z_1$	1M	$B = bZ_1^4+X_1^4$	1M	$B = X_2Z_1^2$	1M	$A = bZ_1^2$	1M	$A_1 = Y_1Z_2^2$	1M	$X_3 = X_1^4+bZ_1^4$	1M
C = A+B		$C = AX_1^4$	1M	C = A+B		$B = X_1+A$		$B_0 = X_2Z_1$	1M	$Y_3 = bZ_1^4Z_3+X_3(aZ_3+Y_1^2+bZ_1^4)$	3M
$D = Y_1Z_2$	1M	$D = Y_1Z_1$	1M	$D = Y_1Z_2^3$	2M	$X_3 = B^4$		$B_1 = X_1Z_2$	1M		
$E = Y_2Z_1$	1M	$E = X_1^2+D+A$		$E = Y_2Z_1^3$	2M	$C = Z_1Y_1$	1M	$C = A_0+A_1$			
F = D+E		$Z_3 = A^3$	1M	F = D+E		$D = Z_3+X_1^2+C$		$D = B_0+B_1$			
G = C+F		$X_3 = AB$	1M	$G = Z_1C$	1M	$E = DX_3$	1M	$E = Z_1Z_2$	1M		
$H = Z_1Z_2$	1M	$Y_3 = C+BE$	1M	$H = FX_2+GY_2$	2M	$Y_3 = X_1^4Z_3+E$	1M	F = DE	1M		
$I = C^3+aHC^2+HFG$	5M			$Z_3 = GZ_2$	1M			$Z_3 = F^2$			
$X_3 = CI$	1M			I = F+Z3				$G = D^2(F+aE^2)$	2M		
$Z_3 = HC^3$	1M			$X_3 = aZ_3^2+IF+C^3$	3M			H = CF	1M		
$Y_3 = GI+C^2[FX_1+CY_1]$	4M			$Y_3 = IX_3+HG^2$	2M			$X_3 = C_2+H+G$			
								$I = D^2B_0E+X_3$	2M		
								$J = D^2A_0+X_3$	1M		
								$Y_3 = HI+Z_3J$	2M		
Total	16M		7M		15M		5M		14M		5M

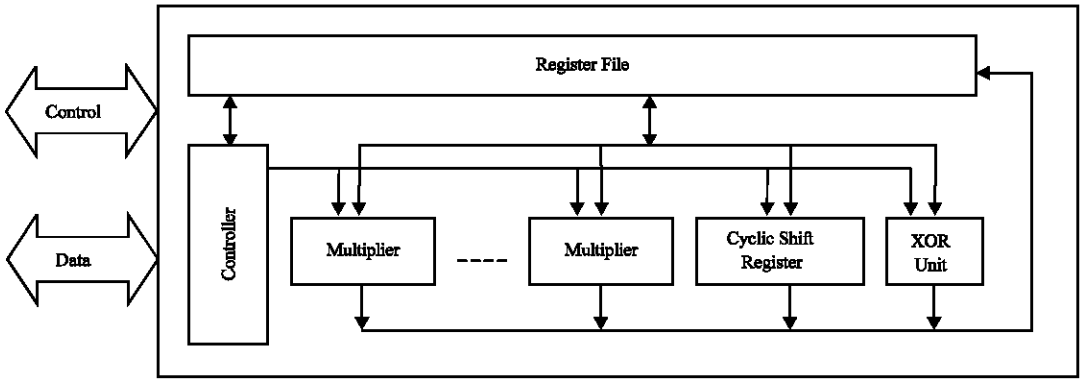

Fig. 1: The proposed architecture

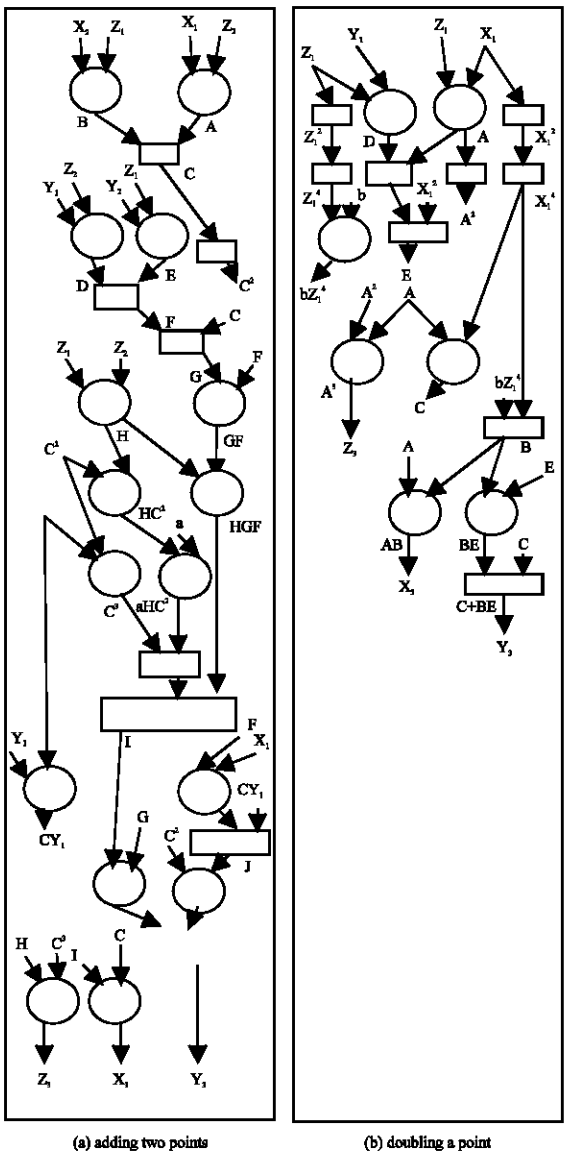

Fig. 2: Data flow graphs for projective coordinate (2 Multipliers)

Methodology used to find the number of multipliers: Since multiplication is the dominant operation in elliptic curve point operations in projective coordinate systems and since the computation time of multiplication is much higher than field squaring and addition, the emphasis in this paper is to speed up the computations of point operations in projective by performing more than one multiplication operation at any one time.

The approach adopted in this study is:

- Analyzing the dataflow of point operations for each projective coordinate system in the following manner:
- Find the critical path which has the lowest number of the multiplication operations,
- Find the maximum number of multipliers that are needed to meet this critical path
- Varying the number of multipliers from one to the number of multipliers specified by the critical path to find the following:
- Find the best schedule of each dataflow using the specified number of multipliers
- Find the AT^2

The critical paths of the projective, Jacobian and Lopez-Dahab coordinate systems are listed in Table 2 for both the point addition and doubling. From Table 2, we can see that the total number of multiplications needed with the projective coordinate system is 16 and 7 for point addition and doubling, respectively. This means that using one multiplier gives an average of (16/2)+7 = 15 multiplications cycles since, on average, we perform doubling for all the bits in the key and perform point addition only for half of the key bits.

The dataflow of the projective coordinate system using two multipliers are shown in Fig. 2, where a circle represents a multiplication and rectangle represent either field addition or squaring. As can be seen from Fig. 2, using 2 multipliers makes the average number of

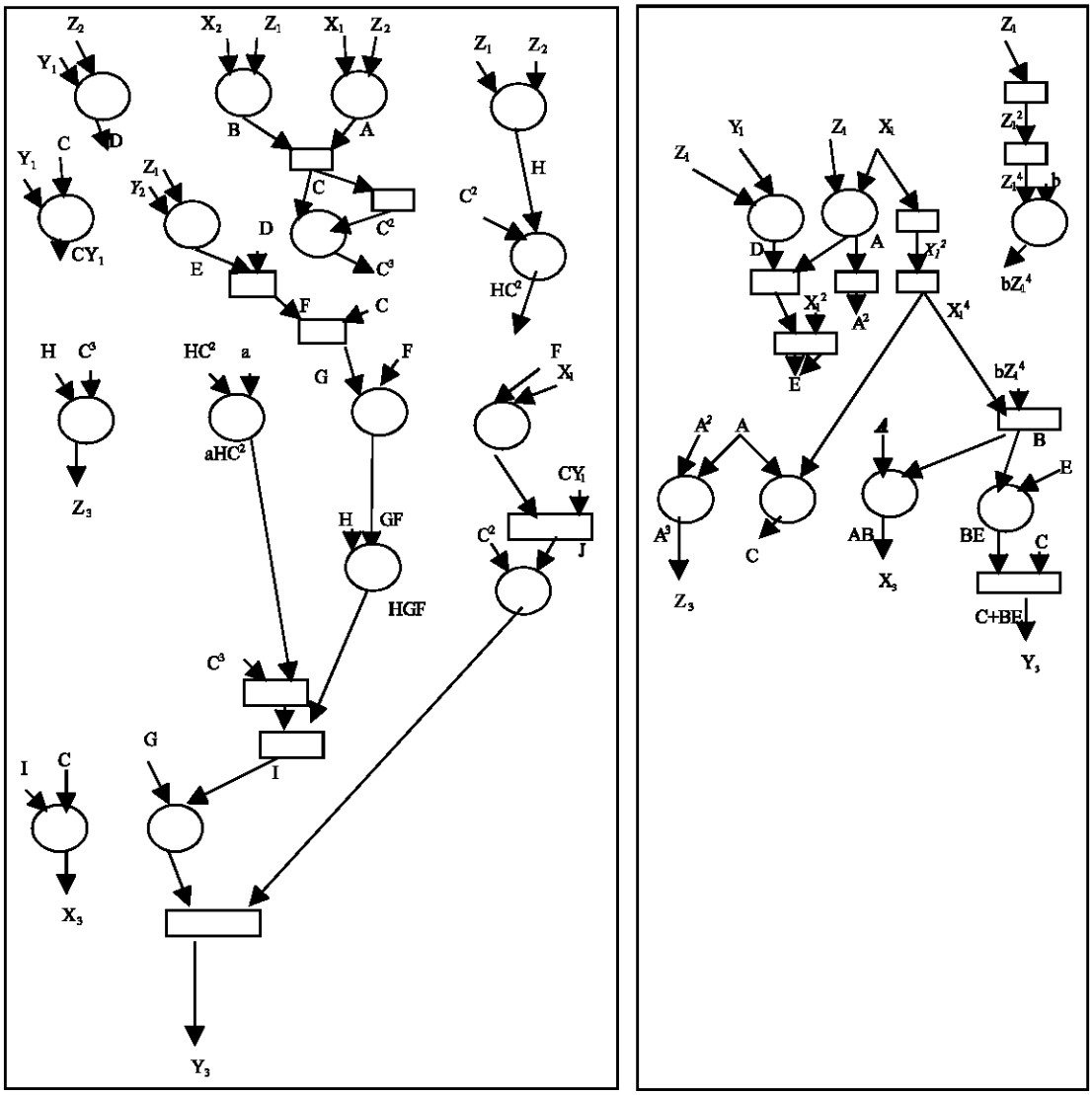

Fig. 3: Data flow graphs for projective coordinate (4 Multipliers)

Table 2: Multiplication cycles for the coordinate systems

	Critical path		Average No. of multiplication cycles			
Coordinate system	ADD	DBL	1 multiplier	2 multipliers	3 multipliers	4 multipliers
Projective coordinate	4	2	15.0	8.0	6.0	4.0
Jacobian coordinate	5	2	12.5	7.0	5.0	4.5
Lopez-Dahab coordinate	4	3	12.0	6.5	5.5	5.0

multiplication cycles decreases to 8 and 4 for point addition and doubling respectively. Which implies an average of (8/2)+4 = 8 multiplications cycles only. This dramatically speed up increases by employing more multipliers until reaching the maximum number of multipliers that satisfies the critical path which means adding more multipliers will increase the area only without any enhancements in the speed up (Fig. 3). The same procedure is applied to the Jacobian and Lopez-Dahab coordinate systems. Table 2 summarizes the average number of multiplications cycles required for point operations using 1, 2, 3 and 4 multipliers.

It is worth noting that unlike the work reported by Gutub (2003a, b) where polynomial basis is used, the

architecture proposed here is based on using normal basis. The advantage of using normal basis is that the computation time of squaring becomes negligible compared to multiplication. This makes a big difference in the number of multiplication cycles as can be seen from Table 2 and also has a significant impact on the utilization of multipliers.

RESULTS AND DISCUSSION

In Fig. 4, comparisons between the different coordinate system are shown. Four cases are covered in these comparisons:

- Single multiplier (Sequential),
- Two, multipliers (Parallel),
- Three multipliers (Parallel) as by (Gutub 2003a, b) and
- Four multipliers (Parallel).

It is clear from Fig. 4 that with the projective coordinate system, the enhancement in the AT^2 increases by employing more multipliers. The maximum number of multipliers that can be reached that satisfies the critical path was found to be 4 multipliers. The enhancements using parallel designs with the Projective coordinate, was found to be 76, 108 and 252% when using 2, 3 and 4 multipliers, respectively. However, the Projective coordinate system was giving better AT^2 than both Jacobian and Lopez-Dahab coordinate systems when employing 4 multipliers, while it was giving worse results by using less number of multipliers.

It can be also noticed that using 3 multipliers, (Gutub, 2003a,b), was giving better result than using 4 multipliers with the Jacobian coordinate system (Fig. 4).

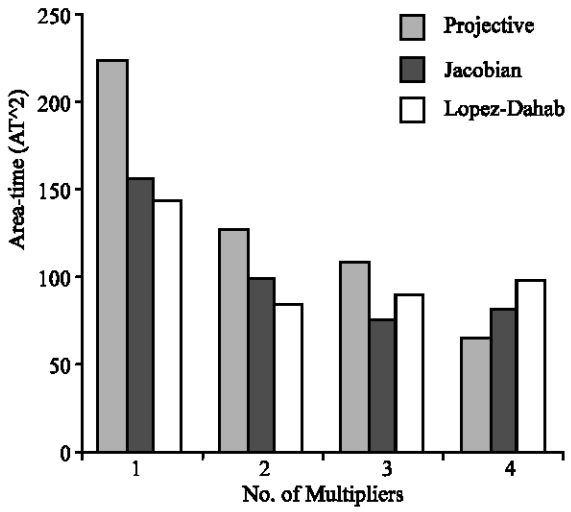

Fig. 4: Comparison between the different designs

This shows clearly that adding more multipliers does not necessarily increase performance.

The best AT^2 result reported by Lopez-Dahab coordinate system was found when using only two multipliers (Fig. 4). The AT^2 when using 2 multipliers is better than the one reported with the Projective coordinate system but not better than the Jacobian coordinate system using the same number of multipliers. However, the best results reported in Fig. 4 where found to be when using the Projective coordinate system with 4 multipliers. What is a more significant observation from Fig. 4 is that using the proposed architecture with Projective coordinate system is not only faster for parallel implementation but it also leads to a better AT^2 (cost) than other alternatives.

CONCLUSIONS

In this study we presented a high performance GF(2^m) elliptic curve cryptoprocessor. Parallelism was exploited at the projective coordinate level using 2, 3 and 4 multipliers to perform parallel field multiplications represented in normal basis. Comparisons between the Projective, Jacobian and Lopez-Dahab coordinate systems using sequential and parallel designs was also presented. The results show that using parallel designs gives better AT^2 than sequential designs by almost 44-252% which gives the designers a wide large of design tradeoffs. The results also show that the Projective coordinate system is the best in parallel designs and gives the least multiplications cycles using 4 multipliers and accordingly the best AT^2.

ACKNOWLEDGMENT

The authors would like to acknowledge the support King Fahd University of Petroleum and Minerals.

REFERENCES

ANSI, X9.62-1998. Public Key Cryptography for the Financial Services Industry: Curve Digital Signature Algorithm (ECDSA).

Blake, I., G. Seroussi and N. Smart, 1999. Elliptic Curves in Cryptography, Cambridge University Press: New York.

Gordon, D., 1998. A Survey of Fast Exponentiation Methods. J. Algorithms, pp: 129-146.

Gutub, A.A., A.F. Tenca, E. Savas and C.K. Koc, 2002. Scalable and unified hardware to compute Montgomery inverse in GF(p) and GF(2^n). Cryptographic Hardware and Embedded Systems-CHES 2002, August 13-15, pp: 484-499.

Gutub, A.A. and M.K. Ibrahim, 2003a. High performance elliptic curve GF(2^k) cryptoprocessor architecture for multimedia, proc. IEEE International Conference on Multimedia and Expo, ICME 2003, Baltimore, Maryland, USA, July 6-9, pp: 81-84.

Gutub, A.A. and M.K. Ibrahim, 2003b, High radix parallel architecture for GF(p) elliptic curve processor, proc. IEEE Conference on Acoustics, Speech and Signal Processing ICASSP 2003, Hong Kong, pp: 625-628.

IEEE P1363, 2000. IEEE Standard Specifications for Public-Key Cryptography.

Koblitz, N., 1987. Elliptic Curve Cryptosystems. Mathematics of Computation, 48: 203-209.

Lidl, R. and H. Niederreiter, 1994. Introduction to Finite Fields and Their Applications. Cambridge University Press, Cambridge, UK.

Lopez, J. and R. Dahab, 1998. Improved algorithms for elliptic curve arithmetic in GF(2^n). SAC'98, LNCS Springer Verlag, pp: 201-212.

Menezes, A.J., 1993. Elliptic Curve Public Key Cryptosystems, Kluwer Academic Publishers.

Mullin, R.C., I.M. Onyszchuk, S.A. Vanstone and R.M. Wilson, 1988. Optimal normal bases in GF(p^m). Discrete Applied Math., 22: 149-161.

National Institute of Standards and Technology (NIST), 2000. Recommended Elliptic Curves for Federal Government Use, Appendix to FIPS, 186-2.

Standards for Efficient Cryptography (SEC) Group/Certicom Research, 2000a. SEC 1: Elliptic Curve Cryptography, Version 1.0,< http://www. secg. org/>.

Standards for Efficient Cryptography (SEC) Group/Certicom Research, 2000b. SEC 2: Recommended Elliptic Curve Cryptography Domain Parameters, Version 1.0,<http://www.secg.org/>.

Thompson, C.D., 1980. A complexity theory for VLSI. Ph.D. Thesis, Carnegie Mellon University, Dep. Computer Science.