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

Turki F. Al-Somani, M.K. Ibrahim and Adnan Gutub

Department of Computer Engineering,
King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia

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 give 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 (time)^p, 0 < p < 1, where the choice of p 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
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, AT² 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.

\[ B = \sum_{i=0}^{m-1} b_i \beta^i, \]

the product \( C = A^*B, \) is given by:

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

then multiplication is defined in terms of a
multiplication table \( \lambda_i \{ 0, 1 \} \)

\[ c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \lambda_i a_{ij} b_{jk} \] \ (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 \( \lambda_i. \)
This value is \( 2m-1 \) and since it allows multiplication with
minimum complexity, such a basis would normally lead to
efficient hardware implementations.

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

\[ Aa^{-1} = 1 \mod 2^m \]

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

\[ a^{-1} = a^{2^m-2} = (a^2)^{2^m-1} \]

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

**ELLIPIC 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 \( P = (x_2, x_2 + y_2) \). For all \( P \in E, \)
\( O + P = P + O = P \). If \( Q = (x_3, y_3) \in E \) and \( Q \neq P \), then \( P + Q =
(x_5, y_5) \).
form \((x, y) = (XZ, Y/Z)\), while the Jacobian coordinate system takes the form \((x, y) = (XZ', Y/Z')\) and the Lopez-Dahab coordinate system takes the form \((x, y) = (XZ, Y/Z')\).

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 \(GF(2^n)\) 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 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 \((AT)\). 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>
<thead>
<tr>
<th>Projective coordinate (Pr)</th>
<th>Jacobian coordinate (J)</th>
<th>Lopez-Dahab coordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Addition</strong></td>
<td><strong>Doubling</strong></td>
<td><strong>Addition</strong></td>
</tr>
<tr>
<td>A = XZ + Z2</td>
<td>1M</td>
<td>A = XZ2</td>
</tr>
<tr>
<td>B = X2Z</td>
<td>1M</td>
<td>B = XZ + Z4</td>
</tr>
<tr>
<td>C = A+B</td>
<td>1M</td>
<td>C = AX + X4</td>
</tr>
<tr>
<td>D = Y + Z</td>
<td>1M</td>
<td>D = Y + Z2</td>
</tr>
<tr>
<td>E = Y + Z</td>
<td>1M</td>
<td>E = Y + Z2</td>
</tr>
<tr>
<td>F = D+E</td>
<td>1M</td>
<td>F = D + E</td>
</tr>
<tr>
<td>G = C+F</td>
<td>1M</td>
<td>G = X + C</td>
</tr>
<tr>
<td>H = Z + Z</td>
<td>1M</td>
<td>H = X + Z2</td>
</tr>
<tr>
<td>I = C + aH + C + HFG</td>
<td>5M</td>
<td>I = Z + G</td>
</tr>
<tr>
<td>X2 = C</td>
<td>1M</td>
<td>X2 = Z</td>
</tr>
<tr>
<td>Z2 = H + C</td>
<td>1M</td>
<td>Z2 + H + C</td>
</tr>
<tr>
<td>Y2 = X + C2[FX + CY2]</td>
<td>4M</td>
<td>Y2 + H + G</td>
</tr>
</tbody>
</table>

**TOTAL:** 16M, 7M, 15M, 5M, 14M

474
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^?

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
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 systems 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).

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 range 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


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


