HIGH RADIX PARALLEL ARCHITECTURE FOR GF(P) ELLIPTIC CURVE PROCESSOR

Adnan Abdul-Aziz Gutub and Mohammad K. Ibrahim

Computer Engineering Department
King Fahd University of Petroleum and Minerals
Dhahran 31261, SAUDI ARABIA
Email: {gutub,ibrahimm}@ccse.kfupm.edu.sa

ABSTRACT

A new GF(p) cryptographic processor architecture for elliptic curve encryption/decryption is proposed in this paper. The architecture takes advantage of projective coordinates to convert GF(p) inversion needed in elliptic point operations into several multiplication steps. Unlike existing sequential designs, we show that projecting into (X/Z,Y/Z) leads to a much better improved performance than conventional choice of projecting into the current (X/Z^2,Y/Z^3). We also propose to use high radix modulo multipliers which give a wide range of area-time trade-offs. The proposed architecture is a significant challenger for implementing data security systems based on elliptic curve cryptography.

1. INTRODUCTION

Elliptic Curve Cryptosystem (ECC) was proposed by Niel Koblitz and Victor Miller in 1985 [1,2,3,4,5,6,7,8,9]. Although critics are still skeptical as to the reliability of this method, to date, no significant breakthroughs have been made in determining weaknesses in the algorithm, which is based on the discrete logarithm problem over points on an elliptic curve. The fact that the problem appears so difficult to crack means that key sizes can be reduced in size considerably, even exponentially [2,5,8], especially when compared to the key size used by other cryptosystems. This made ECC become a challenge to the RSA, one of the most popular public key methods known. ECC is showing to offer equal security to RSA but with much smaller key size (128-256bits) [2].

Several ECC processors have been proposed in the literature recently for GF(p) including GF(2^k) [4,7,16]. The design of these processors is based on representing the elliptic curve points as projective coordinate points [1,4,7,9,16] in order to eliminate division, hence inversion, operations. It is well known that adding two points over an elliptic curve would require a division operation, which is the most expensive operation over GF(p). There are several projective coordinates systems candidates. The choice thus far has been based on selecting the system that has the least number of multiplication steps, since multiplication over GF(p) is next most time consuming and common operation in ECC.

In this paper we propose that the choice of the projective coordinates system should also depend on its inherent parallelism. High-speed crypto processors are now crucial for multimedia applications. It is clear that parallelism is one solution for meeting this requirement. We also propose to use high radix GF(p) multipliers reported in [15] since they have better AT characteristics than conventional radix-2 GF(p) multipliers, and they can lead to wide range of trade-offs between area and time. They can also be implemented in digit serial fashion which is more efficient than both unpipelined and pipelined parallel multipliers for algorithms with repeated multiplications such that found in ECC. It is worth noting that using pipelined parallel multipliers is not efficient for ECC where the multiplication of an iteration cannot commence before the multiplication operation of the previous iteration is finished.

2. ENCRIPTION AND DECRYPTION

It will be assumed that the reader is familiar with the arithmetic over elliptic curve. For a good review the reader is referred to [9]. There are many ways to apply elliptic curves for encryption/decryption purposes. In it most basic form, users randomly chose a *base point (x, y)*, lying on the elliptic curve E. The plaintext (the original message to be encrypted) is coded into an elliptic curve point (x_m, y_m). Each user selects a private key 'n' and computes his public key $P = n(x, y)$. For example, user A's private key is n_A and his public key is $P_A = n_A(x, y)$.

For any one to encrypt and send the message point (x_m, y_m) to user A, he/she needs to choose a random integer k and generate the ciphertext $C_m = \{k(x, y)\ ,\ (x_m, y_m)+ kP_A\ \}$. The ciphertext pair of points uses A's public key, where only user A can decrypt the plaintext using his private key.

To decrypt the ciphertext C_m, the first point in the pair of C_m, $k(x,\ y)$, is multiplied by A's private key to get the point: $n_A\ (k(x, y))$. Then this point is subtracted from the second point of C_m, the result will be the plaintext point (x_m, y_m). The complete decryption operations are:

$$((x_m,y_m)+kP_A)\ -\ n_A(k(x,y))=(x_m,y_m)+k(n_A(x,y))-n_A(k(x,y))=(x_m,y_m)$$

The most time consuming operation in the encryption and decryption procedure is finding the multiples of the base point, (x,y). The algorithm used to implement this is discussed in the next section.

3. POINT OPERATION ALGORITHM

The ECC algorithm used for calculating *nP* from *P* is based on the binary method, since it is known to be efficient and practical to implement in hardware [2,5,7,9,10]. This binary method algorithm is shown below:

Define *k:* number of bits in *n* and n_i: the *i*th bit of *n*
Input: *P* (a point on the elliptic curve).
Output: *Q* = *nP* (another point on the elliptic curve).

```
if $n_{k-1} = 1$, then $Q:=P$ else $Q:=0$;
for $i = k-2$ down to $0$;
    { $Q := Q+Q$ ;
      if $n_i = 1$ then $Q:= Q+P$ ; }
return $Q$;
```

Basically, the binary method algorithm scans the bits of *n* and doubles the point *Q* *k*-times. Whenever, a particular bit of *n* is found to be one, an extra operation is needed. This extra operation is *Q+P*.

As can be seen from the description of the above binary algorithm, adding and doubling elliptic curve points are the most basic operations in each iteration. As mentioned earlier, the points operations over elliptic curve requires inversion [9]. As in the crypto processor in [6,16], inversion is eliminated using projective coordinates as elaborated in the next section.

4 PROJECTIVE COORDINATES IN GF(P)

The projective coordinates are used to eliminate the need for performing inversion. For elliptic curve defined over *GF(p)*, two different forms of formulas are found [1,9] for point addition and doubling. One form projections $(x,y)=(X/Z^2,Y/Z^3)$ [9], while the second projects $(x,y)=(X/Z,Y/Z)$ [1].

The two forms procedures for projective point addition of *P+Q* (two elliptic curve points) is shown below:

$$P=(X_1,Y_1,Z_1);Q=(X_2,Y_2,Z_2);P+Q=(X_3,Y_3,Z_3);\ where\ P \neq \pm Q$$

$\mathbf{(x,y)=(X/Z^2,Y/Z^3) \rightarrow (X,Y,Z)}$		$\mathbf{(x,y)=(X/Z,Y/Z) \rightarrow (X,Y,Z)}$	
$\lambda_1 = X_1Z_2^2$	*2M*	$\lambda_1 = X_1Z_2$	*1M*
$\lambda_2 = X_2Z_1^2$	*2M*	$\lambda_2 = X_2Z_1$	*1M*
$\lambda_3 = \lambda_1 - \lambda_2$		$\lambda_3 = \lambda_2 - \lambda_1$	
$\lambda_4 = Y_1Z_2^3$	*2M*	$\lambda_4 = Y_1Z_2$	*1M*
$\lambda_5 = Y_2Z_1^3$	*2M*	$\lambda_5 = Y_2Z_1$	*1M*
$\lambda_6 = \lambda_4 - \lambda_5$		$\lambda_6 = \lambda_5 - \lambda_4$	
$\lambda_7 = \lambda_1 + \lambda_2$		$\lambda_7 = \lambda_1 + \lambda_2$	
$\lambda_8 = \lambda_4 + \lambda_5$		$\lambda_8 =\lambda_6^2 Z_1Z_2-\lambda_3^2\lambda_7$	*5M*
$Z_3 = Z_1Z_2\lambda_3$	*2M*	$Z_3 = Z_1Z_2\lambda_3^3$	*2M*
$X_3 = \lambda_6^2 - \lambda_7\lambda_3^2$	*3M*	$X_3 = \lambda_8\lambda_3$	*1M*
$\lambda_9 = \lambda_7\lambda_3^2 - 2X_3$		$\lambda_9 = \lambda_3^2 X_1Z_2 - \lambda_8$	*1M*
$Y_3 = (\lambda_9\lambda_6 - \lambda_8\lambda_3^3)/2$	*3M*	$Y_3 = \lambda_9\lambda_6 - \lambda_3^3 Y_1Z_2$	*2M*
	-----		-----
	16 M		***15M***

Similarly, the two forms of formulas for projective point doubling is shown below:

$$P = (X_1,Y_1,Z_1); P+P = (X_3,Y_3,Z_3)$$

$\mathbf{(x,y)=(X/Z^2, Y/Z^3) \rightarrow (X,Y,Z)}$		$\mathbf{(x, y) = (X/Z, Y/Z) \rightarrow (X,Y,Z)}$	
$\lambda_1 = 3X_1^2 + aZ_1^4$	*4M*	$\lambda_1 = 3X_1^2 + aZ_1^2$	*2M*
$Z_3 = 2Y_1Z_1$	*1M*	$\lambda_2 = Y_1Z_1$	*1M*
$\lambda_2 = 4X_1 Y_1^2$	*2M*	$\lambda_3 = X_1Y_1\lambda_2$	*2M*
$X_3 = \lambda_1^2 - 2\lambda_2$	*1M*	$\lambda_4 = \lambda_1^2 - 8\lambda_3$	*1M*
$\lambda_3 = 8Y_1^4$	*1M*	$X_3 = 2\lambda_4\lambda_2$	*1M*
$\lambda_4 = \lambda_2 - 2X_3$		$Y_3=\lambda_1(4\lambda_3-\lambda_4)-8(Y_1\lambda_2)^2$	*3M*
$Y_3 = \lambda_1\lambda_4 -\lambda_3$	*1M*	$Z_3 = 8 \lambda_2^3$	*2M*
	------		-----
	10M		***12M***

The squaring calculation over *GF(p)* is very similar to the multiplication computation. They both are noted as *M* (multiplication). It is worth noting that any EC crypto processor must implement the procedures of projective coordinates efficiently since they are the core steps of the point operation algorithm of ECC.

5. PROPOSED ARCHETICTURE

The architecture of the new processor is shown in Figure 5. This architecture can be used to implement ECC based on either of the two projective coordinate forms discussed in section 4. Unlike existing designs, which use a single multiplier, the new architecture has four multipliers to meet the high data rate demands of applications such as multimedia.

As will be explained now, four multipliers are sufficient to exploit the full parallelism inherent in projective coordinates. As can be seen from Figures 1 and 2, the corresponding critical path of each dataflow diagram is effectively of 4 GF(p) multiplications and of 3 GF(p) multiplications, respectively. Here the time of GF(p) addition and subtraction is ignored since it is very small compared to multiplication. Therefore, the lower bound of the minimum computation time to perform one elliptic point operation in the calculation of *nP* is 7 GF(p) multiplications. It can be easily seen from Figures 1 and 2 that performing four multiplications in parallel will meet this lower bound. Furthermore the utilization of the four multipliers is very high. As can be seen from Figures 1 and 2, all the four multipliers will be used in all of the steps. Similar comments can be made to the data flow in Figures 3 and 4.

6. COMPARSION WITH EXSITING DESIGN

In existing designs, a single multiplier is used to perform all the multiplications needed. The reason is that using more than one single multiplier is perceived to be too expensive.

Comparing the two projective forms, projecting *(x,y)* to $(X/Z^2,Y/Z^3)$ requires a less number of multiplications than projecting into *(X/Z,Y/Z)*. The later uses one less multiplication operation in adding two different elliptic points, however, it uses two more multiplication operations in doubling an elliptic point. For sequential implementation, i.e. using a single multiplier, projecting *(x,y)* into $(X/Z^2,Y/Z^3)$ has always been the candidate of choice for implementing ECC since it has the minimum number of multiplication operations. The crypto processor proposed in [16] is based on such a choice.

Although the proposed architecture can implement the procedures of both projective coordinate forms, the above analysis of the critical paths of both projective coordinates in section 5 indicates that for parallel implementation, projecting *(x,y)* to *(X/Z,Y/Z)* requires less number of cycles and hence it is faster than projecting *(x,y)* to *$(X/Z^2,Y/Z^3)$*. As can be observed from Table 1, using our proposed architecture with projection of *(x,y)* to *(X/Z,Y/Z)* is compared with two different implementations adopting the projection *(x,y)* to *$(X/Z^2,Y/Z^3)$*, using a single multiplier hardware (existing design [16]), and the proposed architecture in Figure 5. The time required by our design in projecting *(x,y)* to *(X/Z,Y/Z)* is less than one third the time of the sequential implementation in [16] and 23% faster than using projection *(x,y)* to *$(X/Z^2,Y/Z^3)$*. What is more significant observation from Table 1 is that the using the proposed architecture with projections *(x,y)* to *(X/Z,Y/Z)* is not only faster for parallel impregnation but it also leads to a better AT^2 performance than both alternatives.

Table 1. Comparison between the different designs

Procedure of Projecting *(x,y)* to	Hardware Design	Number of Multipliers (A)	Avg. Number of Multiplication Cycles (T)	AT^2
$(X/Z^2,Y/Z^3)$	Existing [16]	1	18	324
	Figures 3, 4	4	6.5	169
(X/Z,Y/Z)	Proposed	4	5	100

A final comment about the implementation of our proposed architecture is that we propose to use digit serial implementation of the high radix multiplication algorithms proposed in [15] in our architecture. Digit serial computation is more suitable for the elliptic curve crypto algorithm discussed above since the computation of elliptic point doubling, addition and the algorithm of computing multiples of the base point is such that the multiplication of one stage must be completed before starting the multiplication of the subsequent stage. Therefore even if a pipelined bit-parallel multipliers is used, the throughput of such a multiplier can not be exploited since the next multiplication operation can not commence until the multiplication operations in the previous stage has completed. As with regard to the GF(p) modulo adder, it is to be implemented in bit parallel fashion since the area is not significant compared to the multiplier and minimizing the addition time will reduce the overall multiply-add cycle time.

7. CONCLUSION

A novel GF(p) elliptic curve cryptographic processor is proposed in this paper. It does not need a GF(p) inverse module, because the inverse operation is converted into consecutive multiplication steps using a method known as projective coordinates. It is also shown that for parallel implementation projection of *(x,y)* to *(X/Z,Y/Z)* leads to a better implementation than the usually selected projection *(x,y)* to *$(X/Z^2,Y/Z^3)$*.

8. ACKNOWLEDGMENT

The authors would like to thank King Fahd University of Petroleum And Minerals for its support of this research work.

9. REFERENCES

[1] Miyaji A., "*Elliptic Curves over F_P Suitable for Cryptosystems*", Advances in cryptology- AUSCRUPT'92, Australia, December 1992.

[2] Stallings, W. "*Cryptography and Network Security: Principles and Practice*", 2nd Ed., Prentice Hall, NJ, 1999.

[3] Chung, Sim, and Lee, "*Fast Implementation of Elliptic Curve Defined over $GF(p^m)$ on CalmRISC with MAC2424 Coprocessor*", Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000, Massachusetts, August 2000.

[4] Okada, Torii, Itoh, and Takenaka, "*Implementation of Elliptic Curve Cryptographic Coprocessor over $GF(2^m)$ on an FPGA*", Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000, Massachusetts, August 2000.

[5] Crutchley, D. A., "*Cryptography And Elliptic Curves*", Master Thesis under Supervision of Prof. Gareth Jones, submitted to the Faculty of Mathematics at University of Southampton, England, May 1999.

[6] Orlando, and Paar, "*A High-Performance Reconfigurable Elliptic Curve Processor for $GF(2^m)$*", Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000, Massachusetts, August 2000.

[7] Stinson, D. R., "*Cryptography: Theory and Practice*", CRC Press, Boca Raton, Florida, 1995.

[8] Paar, Fleischmann, and Soria-Rodriguez, "*Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents*", IEEE Transactions on Computers, Vol. 48, No. 10, October 1999.

[9] Blake, Seroussi, and Smart, "*Elliptic Curves in Cryptography* ", Cambridge University Press: New York, 1999.

[10] Hankerson, Hernandez, and Menezes, "*Software Implementation of Elliptic Curve Cryptography Over Binary Fields*", Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000, Massachusetts, August 2000.

[11] G. A. Orton, M. P. Roy, P. A. Scott, L. E. Peppard, and S. E. Tavares. "VLSI implementation of public-key encryption algorithms", *Advances in Cryptology -- CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 277-301, 11-15 August 1986. Springer-Verlag, 1987.

[12] Scott, Norman R., "*Computer Number Systems and Arithmetic*", Prentice-Hall Inc., New Jersey, 1985.

[13] Tocci, R. J. and Widmer, N. S., "*Digital Systems: Principles and Applications*", Eighth Edition, Prentice-Hall Inc., New Jersey, 2001.

[14] Ercegovac, M. D., Lang, T., and Moreno, J. H., "*Introduction to Digital System*", John Wiley & Sons, Inc., New York, 1999.

[15] Mekhallalati, M, Ibrahim, M K, & Ashur, A, "Radix Modular Multiplication Algorithm", *Journal of Circuits and Systems, and Computers*, Vol.6, N0.5, pp547-567,1996.

[16] Orlando, and Paar, "A scalable GF(p) elliptic curve processor architecture for programmable hardware", *Cryptographic Hardware and Embedded Systems, CHES 2001*, May 14-16, 2001, Paris, France.

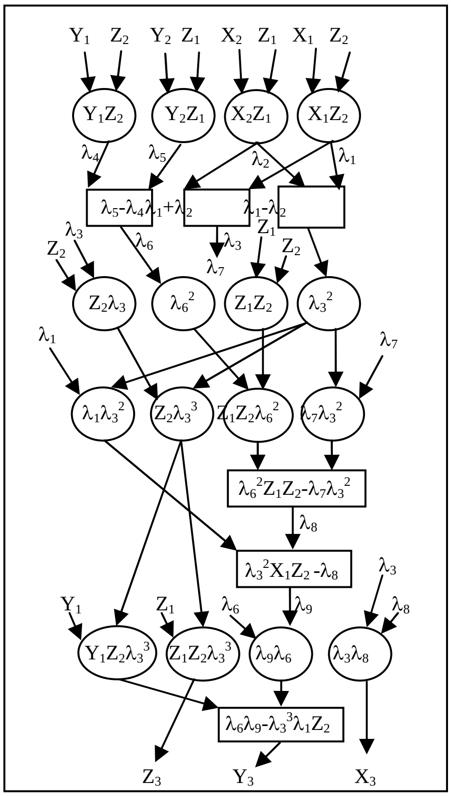

Fig 1. Projecting *(x,y)* to *(X/Z,Y/Z)* adding two points data flow

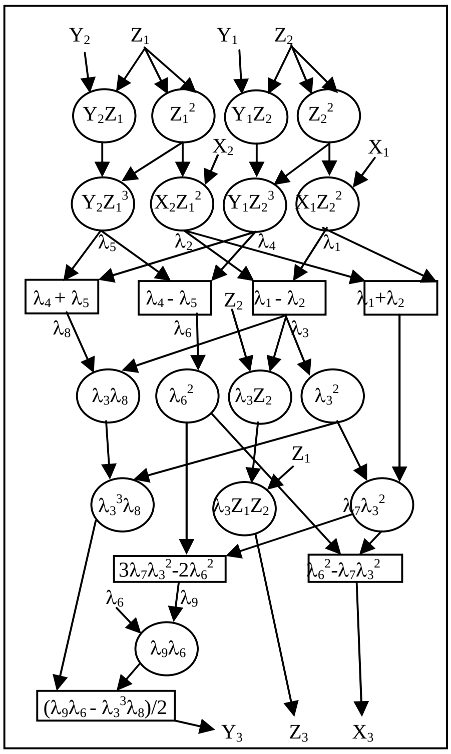

Fig 3. Projecting *(x,y)* to *(X/Z^2,Y/Z^3)* adding points data flow

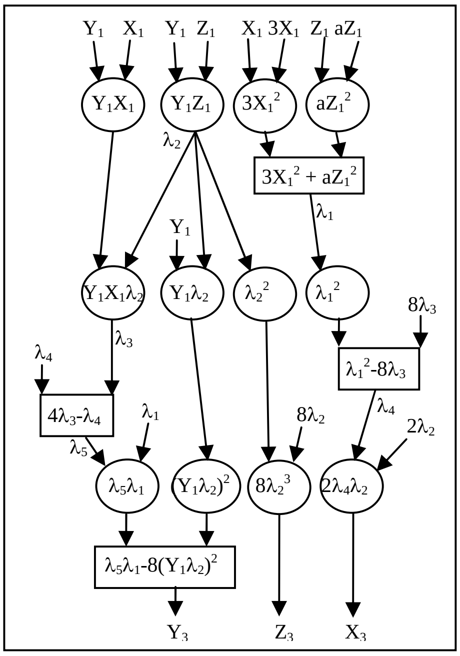

Fig 2. Projecting *(x,y)* to *(X/Z,Y/Z)* doubling a point data flow

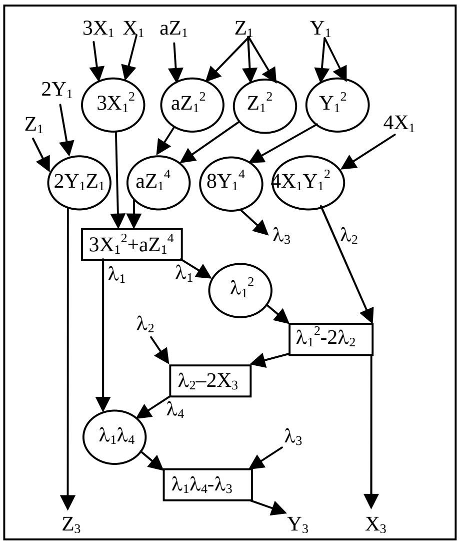

Fig 4. Projecting *(x,y)* to *(X/Z^2,Y/Z^3)* doubling a point data flow

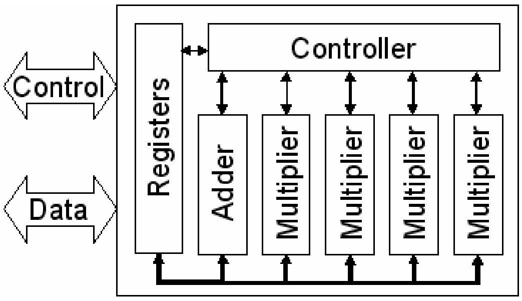

Fig 5. Proposed elliptic curve processor architecture