FAST ELLIPTIC CURVE CRYPTOGRAPHIC PROCESSOR ARCHITECTURE BASED ON THREE PARALLEL GF(2^k) BIT LEVEL PIPELINED DIGIT SERIAL MULTIPLIERS

Adnan Abdul-Aziz Gutub

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

ABSTRACT

Unusual processor architecture for elliptic curve encryption is proposed in this paper. The architecture exploits projective coordinates *(x=X/Z, y=Y/Z)* to convert GF(2^k) division needed in elliptic point operations into several multiplication steps. The processor has three GF(2^k) multipliers implemented using bit-level pipelined digit serial computation. It is shown that this results in a faster operation than using fully parallel multipliers with the added advantage of requiring less area. The proposed architecture is a serious contender for implementing data security systems based on elliptic curve cryptography.

1. INTRODUCTION

In 1985 Niel Koblitz and Victor Miller proposed the Elliptic Curve Cryptosystem (ECC) [1-9], a method based on the Discrete Logarithm problem over the points on an elliptic curve. Since that time, ECC has received considerable attention from mathematicians around the world, and no significant breakthroughs have been made in determining weaknesses in the algorithm. Although critics are still skeptical as to the reliability of this method, several encryption techniques have been developed recently using these properties. 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 [2].

Several crypto processors have been proposed in the literature recently [4,7,15]. A common feature of these processors is that they eliminate the need for an inversion circuit. It is well known that adding two points over an elliptic curve would require a division operation, and hence an inversion. Calculating the inverse is the most expensive operation over GF(2^k) [16,17]. To eliminate the need for performing inversion in *GF(2^k)*, designs replace the inversion by several multiplication operations by representing the elliptic curve points as projective coordinate points [1,4,7,9,15,18]. This approach is also adopted in the processor proposed in this paper.

The different crypto-processor designs differ mainly in the architecture of the basic GF(2^k) multiplier. Clearly it is impractical to use bit-parallel multipliers for large word length, i.e. k > 512. In [4] a n_d x m_d digit multiplier is used to implement the multiplication over GF(2^k), where $k > n_d$ and m_d. While in [7] a digit serial multiplier was adopted. A similar approach was used in the elliptic curve processor over GF(q^m) in [15]. There are two basic drawbacks with the existing processors. The first is that digit serial multiplication is not as efficient as sub-digit pipelined digit serial computation [13,14]. The second is that none of the existing designs exploit the inherent parallelism in the computation of the elliptic curve point operations. In this paper a new elliptic curve crypto processor architecture is proposed that takes an advantage of both of these aspects. It is strongly believed that these two aspects would lead to an even better trade off between the area and time of computation.

2. ENCRYPTION 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 compute 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 cipher text:

$$C_m = \{k(x, y) , (x_m, y_m)+ kP_A \}.$$

The cipher text pair of points uses A's public key, where only user A can decrypt the plaintext using his private key.

To decrypt the cipher text 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:

$$\begin{aligned} & ((x_m, y_m) + kP_A) - n_A(k(x, y)) \\ = \; & (x_m, y_m) + k(n_A(x, y)) - n_A(k(x, y)) \\ = \; & (x_m, y_m) \end{aligned}$$

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 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).

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

Basically, the binary method algorithm scans the binary 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 two elliptic curve points and doubling a point are the most basic operations in each iteration. As mentioned earlier, adding two points over elliptic curve requires inversion [9]. As in the crypto processor in [6], inversion is eliminated using projective coordinates as discussed in the next section.

4. POINT OPERATIONS OVER PROJECTIVE COORDINATES

Elimination of inversion is achieved by projecting the coordinates (x,y) into (X,Y,Z), where $x=X/Z$, and $y=Y/Z$. The projected elliptic curve equation is introduced in [18]; it is detailed below to generate the data flow graphs discussed afterward.

The procedure for projective point addition of $P+Q$ (two elliptic curve points) is shown below:

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

$$\boldsymbol{(x,y)=(X/Z,Y/Z) \rightarrow (X,Y,Z)}$$

$A = X_1Z_2$	*1M*
$B = X_2Z_1$	*1M*
$C = A+B$	
$D = Y_1Z_2$	*1M*
$E = Y_2Z_1$	*1M*
$F = D+E$	
$G = C+F$	
$H = Z_1Z_2$	*1M*
$I=C^3+aHC^2+HFG$	*6M*
$X_3 = CI$	*1M*
$Z_3 = HC^3$	*1M*
$Y_3=GI+C^2[FX_1+CY_1]$	*5M*
	---
	17 M

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

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

$$\boldsymbol{(x, y) = (X/Z, Y/Z) \rightarrow (X,Y,Z)}$$

$A=X_1Z_1$	*1M*
$B= bZ_1^4+X_1^4$	*5M*
$C= AX_1^4$	*1M*
$D=Y_1Z_1$	*1M*
$E=X_1^2+D+A$	
$Z_3=A^3$	*2M*
$X_3=AB$	*1M*
$Y_3= C+BE$	*1M*

	12M

The squaring calculation over $GF(2^k)$ is assumed very similar to the multiplication computation. They are both denoted as M (multiplication) in the above.

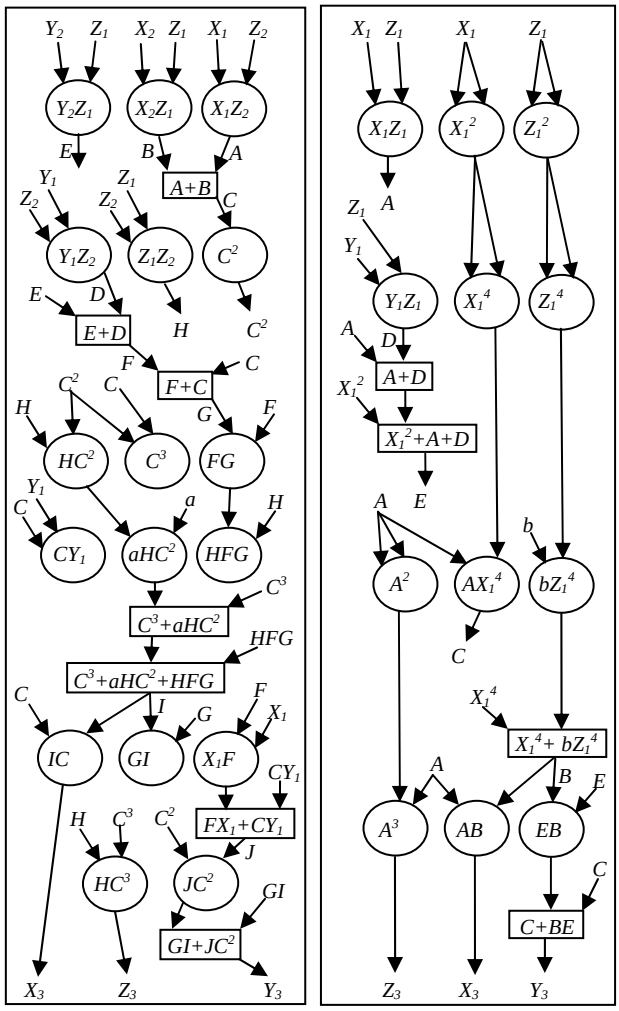
Figure 1a shows the data flow graph for adding two elliptic curve points. The hardware of this design if implemented as shown in Figures 1 would need seventeen multipliers and seven k-bit XOR gates. The complete data flow graph for doubling a point is shown in Figure 1b. It is made of twelve multipliers and four k-bit XOR gates.

Any elliptic curve crypto processor that uses projective coordinates must implements the dataflow graphs in Figures 1a and 1b iteratively.

5. PROPOSED CRYPTO ARCHITECTURE

The architecture of the new processor is shown in Figure 2. Unlike existing designs which use a single

multiplier, the new architecture has three multipliers. The reason for using more than one multiplier is discussed fully in section 6. However, the reason for using no more than three multiplier is now explained. As can be seen from Figures 1a and 1b, the corresponding critical path each dataflow diagram is effectively of 6 GF(2^k) multiplications and of 4 GF(2^k) multiplications, respectively. Here the time of GF(2^k) addition is ignored since it negligible compared to multiplication. Therefore, the lower bound of the minimum computation time to perform one elliptic point operation in the calculation of *nP* is ten GF(2^k) multiplications. It can be easily seen from Figures 1a and 1b that performing three multiplications in parallel will meet this lower bound. Furthermore the utilization of the three multipliers is very high almost the maximum. As can be seen from Figures 1a and 1b, all the three multipliers will be used in nine out of the ten steps, and in only one out of the ten cycles where a single multiplier is not used.



(a) adding two points (b) doubling a point

Figure 1. Data flow graphs for the elliptic curve point operations of projecting *(x,y)* to *(X/Z,Y/Z)*

In the crypto processor presented here we also propose to use bit-level pipelined GF(2^k) digit serial multipliers reported in [13,14]. It is significant to point out that these multipliers are in fact faster and use less area than their *un-pipelined* bit-parallel counterparts [13,14]. Moreover, sub-digit pipelining of digit serial computation leads to a much better performance than the conventional digit serial structures as shown in Table 1 [13].

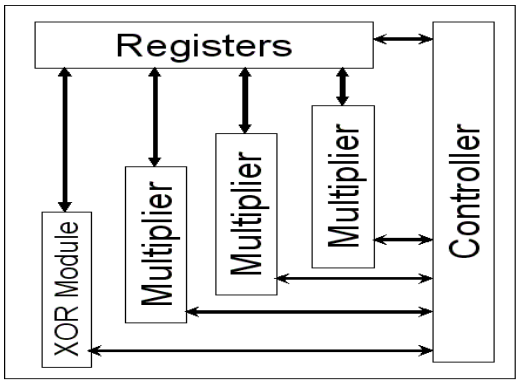


Figure 2. The elliptic curve point operations hardware

Bit-level 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(2^k) 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.

Table 1. Comparison of Area and Time of the pipelined digit-serial GF(2^k) multiplier in [14] for different number of sub-digit pipelining levels, K.

K	Area: $A_T(K)/A_T(1)$	Time: $T(K)/T(1)$
1	1	1
2	1.3	2
4	1.4	4
8	1.9	8

6. COMPARISON WITH EXISTING DESIGN

In existing designs, a single multiplier is used to perform all the multiplications needed in Figures 1a and 1b. The reason is that using more than one single multiplier is perceived to be too expensive. However, using three multipliers will lead to a better AT^2.

Observe Table 2, our proposed design is compared with an existing design demonstrated in [6]. The number of registers needed in the proposed hardware is not that much better than the existing one. However, the AT^2 of our design is the real achievement.

Table 2. Comparing the proposed design with the conventional one.

Hardware Design	Conventional	Proposed
Number of Multipliers (A)	1	3
Worst case No of Cycles	17 + 12 = 29	6 + 4 = 10
Avg. No. of Cycles (T)	12 + (17/2) = 20.5	4 + (6/2) = 7
Number of Registers	12	11
Cost: AT^2	420.25	147

7. CONCLUSION

A new $GF(2^k)$ elliptic curve crypto processor is proposed in this paper. It does not need a $GF(2^k)$ inverter, because the inverse operation is converted into successive multiplication steps using projective coordinates. It exploits the inherent parallelism in the computation of doubling and adding points over an elliptic curve as well as the sub-digit pipelined digit serial computation to achieve a better trade-off between area and time.

8. ACKNOWLEDGMENT

The Author would like to thank Professor *Mohammad K. Ibrahim* for his valuable suggestions and comments. The Author also acknowledges the support provided to this work from King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

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*", Second Edition, Prentice Hall Inc., New Jersey, 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: NY, 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] Ibrahim, M. K., Almulhem, A., "Bit-Level Pipelined Digit Serial $GF(2^m)$ Multiplier", *IEEE International Symposium on Circuits and Systems*, Sidney, Australia, 2001.

[14] Ibrahim, M. K., Junaid, A. K., Al-Abaji, R. H., Almulhem, A., "Trade-off analysis of a new sign digit serial GF multiplier", *Fifth World Multi-conference on Systemics, Cybernetics and Informatics SCI / ISAS 2001*. Volume XIV, Part II, pages 52-56. July 2001, Orlando, 2001.

[15] 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.

[16] Gutub, Adnan Abdul-Aziz, Tenca,A., and Koc,C., "Scalable VLSI architecture for GF(p) Montgomery modular inverse computation", *IEEE Computer Society Annual Symposium on VLSI*, pages 53--58, Pittsburgh, Pennsylvania, April 25-26, 2002.

[17] Gutub, Adnan Abdul-Aziz, Tenca,A.F., and Koc,C., "Scalable and Unified Hardware to Compute Montgomery Inverse in GF(p) and GF(2^n)", *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 485-500, August 13-15, 2002.

[18] Ernst, Klupsch, Hauck, and Huss, "Rapid Prototyping for Hardware Accelerated Elliptic Curve Public-Key

Cryptosystems", *The IEEE 12th International Workshop on Rapid System Prototyping*, Monterey, CL, June 25-27, 2001.