Design of an Efficient Elliptic Curve Cryptography Coprocessor

B.MuthuKumar[1], Dr.S.Jeevananthan[2]

[1]*Research scholar , Sathyabama University,India.*

[2]*Asst. Professor, Department of Electrical and Electronics Engineering, Pondicherry Engineering College, India.*

[1]anbmuthusba@yahoo.co.in

[2]drsj_eee@pec.edu

Abstract - **Elliptic curve cryptography plays a crucial role in networking and communication security. FPGA based architecture of elliptic curve cryptography coprocessor is proposed in this paper. The coprocessor contains the operation over binary finite fields, point adding, doubling and scalar multiplication on elliptic curve. In this coprocessor a new type of FPGA-based modular multiplier architecture is proposed. Experiment results show that coprocessor designed can achieve high performance. The coprocessor embedded in PCI adaptor is realized for encryption and decryption.**

I. INTRODUCTION

First introduced in the 1980s, elliptic curve cryptography (ECC) has become popular due to its superior strength per bit compared to existing public key algorithms such as RSA [1]. This superiority translates to equivalent security levels with smaller keys, bandwidth savings, and faster implementations [1], making ECC very appealing. The IEEE proposed standard P1363-2000 [3] recognizes ECC-based key agreement and digital signature algorithms. In [4], a list of secure curves is given by the U.S. Government National Institute of Standards and Technology (NIST).

Intuitively, there are numerous advantages of using field-programmable gate-array (FPGA) technology to implement in hardware the computationally intensive operations needed for ECC. Indeed these advantages are comprehensively studied and listed by Wollinger, *et al.* in [5]. In particular, performance, cost efficiency, and the ability to easily update the cryptographic algorithm in fielded devices are very attractive for hardware implementations for ECC.

The remainder of this paper is organized as follows. Section 2 introduces hierarchy of the ECC. In section 3 elements operations over finite field are described. Section 4 describes point arithmetic over elliptic curve group in detail. Section 5 proposes scalar multiplication scheme and architecture of coprocessor. Section 6 provides hardware experiment results and conclusions.

II. HIERARCHY OF ECC ARITHMETIC

Points on the elliptic curve defined over a finite field are along with the point addition as group operation. The basic operation in elliptic curve cryptosystems is computation of kP, where P is a curve point and k a large integer. The Arithmetic of ECC is primarily based on several elements operations over finite field, which have unique advantage in terms of hardware implementation. Elements operations over finite fields, such as elements modular addition, subtraction, modular squaring and modular multiplication, are quite necessary. However, the computing performance is greatly affected by the element representation basis, two of which are widely used in binary finite field [10] are polynomial basis and normal basis.

The mathematic foundation of ECC is based on Weierstrass equation. Nowadays the research work on the equation is focused on affine and projective coordinates form. The equation in affine coordinates is

$$y2+xy = x3+ax2+b \ (b?0) \quad (1)$$

Several projective coordinate systems for elliptic curve equation have been proposed in order to avoid the time-consuming inversion operation [7]. Projective coordinate system proposed by Lopez and Dahab is suitable for hardware implementation, and it is called L-D projective coordinates in this paper.

Scalar multiplication is the foundation of ECC, and it directly contributes to discrete logarithm problem and several cryptographic schemes.

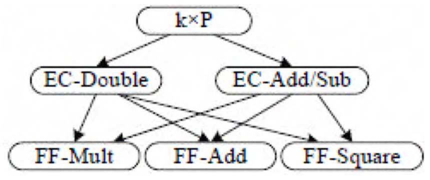

Fig 1. Computing hierarchy of ECC arithmetic

The hierarchy of arithmetic for elliptic curve scalar multiplication is depicted in Figure 1, where *FF-Mult* represents finite field multiplication, *FF-Add* represents finite field adding, *FF-Square* represents finite field squaring, *EC-Add* represents addition on elliptic curve, and so on [11].

III. Arithmetic over Binary Finite Field

The most important arithmetic over binary finite field is modular multiplication; hence it is necessary to design an efficient multiplier, which depends on the given finite field. Considering resource consumption, both shifted canonical basis and type II optimal normal basis are utilized in our design to achieve the efficient multiplier.

Let E be an elliptic curve over finite field defined by equation (1) where a and b are in finite field and b is not equal to 0. Let P = (x; y) be a point on elliptic curve E, and let k is greater than 0 be an integer. We may write k as a binary expansion

$$K = (K_{s-1}K_{s-2} \ldots K_1K_0)_2$$
$$= K_{s-1}2^{s-1} + k_{s-2}2^{s-2} + \ldots K_{1s} + k_0, \quad (2)$$
where $k_i = 0,1$ with $k_{s-1} = 1$

We can compute kP using the following L´opez-Dahab algorithm [8]. Two main advantages of the L´opez-Dahab algorithm are: 1) A number of inversions are avoided by using projective coordinate representation and 2) The same operations are performed in every iteration of the main loop. These two benefits mean that, faster implementation is possible with potentially increasing resistance to timing attacks.

Sunur and Koc have proposed an efficient full parallel multiplier with the above formula [12], but the efficient full serial modular multiplier and trade-off multiplication design have not been proposed yet. We design an efficient full serial modular multiplier in FPGA in this paper, and a trade-off multiplier design scheme is proposed for the first time. The multiplier uses both shifted canonical basis and type II optimal normal basis. The architecture of the trade-off version multiplier is depicted in Figure 2, in which symbol ? denotes logic AND operation and ? denotes

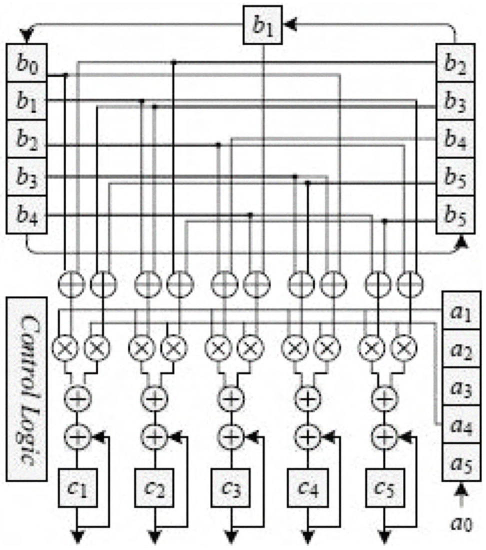

Fig 2. Trade-off design of modular multiplier

logic XOR operation, and the bit a_0 and b_0 in the figure 2 are zero. The multiplier consumes only half of the total clock cycles compared to full serial multiplier, that is, [m/2] clock cycles.

Addition of two elements over finite field is a different operation. Fortunately, the adding operation in binary finite field can be finished easily with exclusive XOR logic circuit. Element squaring is also a necessary arithmetic in the field arithmetic, to implement squaring arithmetic in FPGA just use register rotating.

IV. Point Operations on Elliptic Curve

Both point addition and subtraction operations are foundation of scalar multiplication. Point addition operation includes two different points adding and two same point doubling. In this section, we will discuss the implementation of point operations inside FPGA in detail based on coprocessor in L-D projective coordinates.

Since the modular inversion operation over field is a time consuming operation, points on elliptic curve should be represented in projective coordinates to avoid it. In the L-D projective coordinates, elliptic curve equation is the same as formula (2). The points addition formula that do not involve modular inversion operation can be derived by converting the point to affine projective as $x=X/Z$, $y=Y/Z_2$ at first, then adding the affine points with the formula, and finally clearing the denominators. With the method mentioned above, the distinct points adding and the same point doubling formula can be derived.

From algorithm 1, if we use one multiplier, six multiplications are required. By observing the algorithm 1,

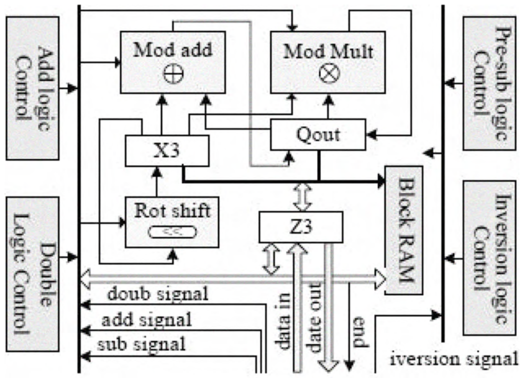

Fig 3. Architecture of point operation module

however, we can find that three multiplications are independently computed, i.e., the multiplications $X1Z2$ ($T1$), $X2Z1$

A. Algorithm 1: Point doubling & addition with uniform addressing

```
if $K_{l-2} = 1$ then Swap($X_1, X_2$), Swap($Z_1, Z_2$)
end if
for i = $l$-2 down to 0 do
$Z_3 = (X_1Z_2 + X_2Z_1)^2$
$X_2 = xZ_3 + (X_1Z_2)(X_2Z_1)$, $Z_2 = Z_3$,
$X_1 = X^4_1 + bZ^4_1$, $Z_1 = X^2_1Z^2_1$
```

```
if ($l$ != 0 and $k_l$ != $k_{l-1}$) or ($l$ = 0 and $k_1$ = 1) then
Swap($X_1$, $X_2$), Swap($Z_1$, $Z_2$)
end if
end for
```

(T_2), and X_1Z_1 can be computed at the first step and the multiplications T_1T_2, xZ_3, and bZ^4_1 can be computed at the second step respectively. From this observation, we can derive a new parallelized version corresponding to the algorithm 1, and a modified version is described in algorithm2.

B. Algorithm 2: Parallelized version of point doubling & addition with uniform addressing

```
if $k_{l-2}$ =1 then
Swap($X_1$, $X_2$), Swap($Z_1$, $Z_2$) end if
for $i = l- 2$ down to 0 do
1. $T_1 = (X_1Z_2), T_2 = (X_2Z_1), Z_3 = (T_1 + T_2)^2$,
   $T_3 = (X_1Z_1)^2, Z_2 = Z_3$
2. $X_2 = T_1T_2 + xZ_3, X_1 = bZ^4_1 + X^4_1, Z_1 = T_3$
if ($l$ != 0 and $k_l$ != $k_{l-1}$) or ($l$ = 0 and $k_i$ = 1) then
Swap($X_1$, $X_2$), Swap($Z_1$, $Z_2$)
end if
end for
```

A very small amount of clock cycles is required to read and write the temporary data to and fro Block RAM. The architecture of the coprocessor is shown in Figure 3, in which Mod Mult is instantiated modular multiplication module in Verilog description, X_3, Z_3 and Q_{out} are 233- bit wide register, Add Logic Control and Double Logic Control are operation process finite state machine module.

On the field of arithmetic operation over GF(2^{233}) , the coprocessor can achieve two different point adding operation, point doubling operation and two point subtracting operation and so on. To improve computing performance the data bus inside coprocessor is 233-bit wide. All the computing operation is controlled by finite state machine.

V. SCALAR MULTIPLICATION

Many scalar multiplication algorithms for certain elliptic curve have been proposed, such as binary method, non-adjacent form (NAF) method, slide window NAF method, fixed comb method, Montgomery method, etc. Among these algorithms, the NAF method, without pre-computing, is more suitable for hardware implementation for its concision and efficiency. The following NAF algorithm is adopted in our design of coprocessor. The architecture of scalar multiplying module in FPGA device is depicted as Figure 4.

```
1.set $h_lh_{l-1}...h_1h_0$ is binary representation of 3k.
2.set $k_lk_{l-1...}k_1k_0$ is binary representation of k.
3.set S=Q.
4.For i from $l$–1 down to 1do
  4.1 set S = 2S.
  4.2 if (hi=1 && ki=0) S=S+Q;
  4.3 if (hi=0 && ki=1) S=S-Q;
5.output S.
```

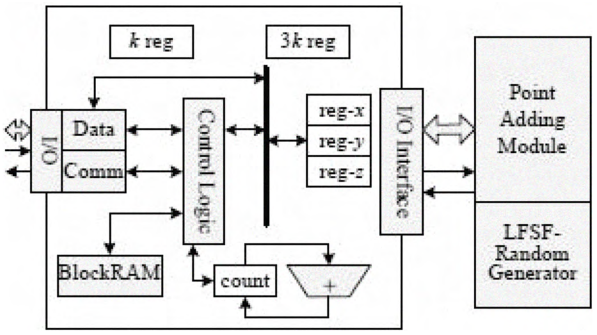

Fig 4. Architecture of scalar multiplying module

For example, two different elliptic curve points addition requires 9 field multiplications in mixed coordinate system, one is affine coordinate system and the other is LD projective coordinate system. A point doubling require 4 field multiplications and each multiplication requires [233/2] =117 clock cycles in GF(2^{233}). Suppose k is 233 bits and the hamming weight is approximately 70, so a general scalar multiplication requires about 183000 clock cycles. Since our ECC coprocessor can work at 80 MHz, that means 438 general scalar multiplications can be finished per second, and each scalar multiplication requires 2.28 ms.

TABLE 1
COMPARISON OF PUBLISHED DESIGN

Design	Field	Speed (MHz)	*K*p (ms)	Device
[13]	m=115	15	10.3	XC4020XL
[14]	m= 55	36	6.8	320KGates
[15]	m=233	30	2.94	XC4085XL
[16]	m=160	21	3.8	XCV800-4
[17]	m=161	166	4.7	XC2V100
[18]	m=233	37	13.2	XCV2000E
Our Work	m=233	80	2.28	XC3S1000

The performance comparison between different designing of processor is not always straightforward. There are so many aspects, such as field size, representing basis, point arithmetic, coordinate system selecting etc, influence the performances that it is hard to determine which one is more excellent. Yet we summarize several coprocessor published in recent years and list them in Table 1 with our own design in it. The data in the table mainly show time each scalar multiplication consuming.

VI. EXPERIMENT AND CONCLUSIONS

The FPGA based architecture of ECC arithmetic coprocessor introduced above is described with Verilog HDL. The Verilog description is correctly simulated in ModelSim and the synthesis of the description is finished in Xilinx's integrated software environment. The placing and routing process are finished in Xilinx's XC3S1000 device. The synthesis report shows that the coprocessor can work at over 100MHz. Experiment show that the chip works actually at 80-MHz successfully in the PCI card with the chip embedded in.

The FPGA based architecture of ECC arithmetic coprocessor is proposed in this paper according to the computational hierarchy of ECC. The architecture of our coprocessor is described with Verilog HDL, in which a new design of trade-off modular multiplier is proposed. The new design can achieve more convenience and flexibility into modular mu ltiplier design over finite field. Moreover, the multiplier can achieve half or quarter of clock cycles consuming in the full bit-serial multiplier (m clock cycles in GF(2^m)). Our coprocessor can achieve a higher performance shown in above comparison. With the coprocessor embedded in, a PCI ECC adapter for data encryption and decryption is implemented.

REFERENCES

[1] N. Koblitz, A. Menezes, and S. Vanstone, "The state of elliptic curve cryptography," *Designs, Codes Cryptography*, vol. 19, pp. 173–193, 2000.

[2] N. Gura, S. C. Shantz, H. Eberle, S. Gupta, V. Gupta, D. Finchelstein, E. Goupy, and D. Stebila, "An end-to-end systems approach to elliptic curve cryptography," presented at theWorkshop Cryptographic Hardware .Embedded Syst. (CHES), Redwood Shores, CA, 2002.

[3] *Standard Specifications for Public Key Cryptography*, IEEE1363, 2000.

[4] NIST—National Institute of Standards and Technology, "Recommended elliptic curves for federal government use," 2000 [Online]. Available: http://csrc.nist.gov/encryption

[5] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGAs: State-of the- art implementations and attacks," *IEEE Trans. Embedded Comput.Syst.*, vol. 3, no. 3, pp. 534–574, Aug. 2004.

[6] J.Lopez, R.Dahab. "Improved Algorithm for Elliptic Curve Arithm etic in GF(2n) ". Selected Areas in Cryptography- SAC'98, LNCS 1556, 1999: pp. 201-212.

[7] IEEE Std P1363-2000. "IEEE Standard Specifications for Public-Key Cryptography". 2000.

[8] J. L´opez and R. Dahab, "Fast Multiplication on Elliptic Curves over GF(2m) without Precomputation," CHES 1999, Lecture Notes in Computer Science Vol. 1717, pp. 316-327, 1999.

[9] Michael Jung, "FPGA Based Implementation of an Elliptic Curve Coprocessor Utilizing Synthesizable VHDL code", Darmstadt University of Technology. Available at http://www.vlsi.informatik.tu-darmstadt.de/staff/mjung/publications/comprehensive.pdf

[10] A. Reyhani-Masoleh and M. A. Hasan, "Efficient Digit Serial Normal Basis Multipliers over GF(2m)", in proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2002), May 2002, pp. 781-784.

[11] Darrel Hankerson, Alfred J. Menezes, Scott Vanstone, "Guide to Elliptic Curve Cryptography". Springer-Verlag, New York. 2005.

[12] Sunar, B. Koc, C.K. "An Efficient Optimal Normal Basis Type II Multiplier". Computers, IEEE Transactions on, Jan 2001, pp.83-87.

[13] S. Sutikno, R. Effendi, A. Surya. "Design and Implementation of Arithmetic Processor GF(2155) for Elliptic Curve Cryptosystems", in: Proceedings of the IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS'98),1998, pp. 647~650.

[14] K.H.Leung, K.W.Ma, W.K. Wong, P.H.W.Leong, "FPGA implementation of a microcoded elliptic curve cryptographic processor", in: Proceedings of Field- Programmable Custom Computing Machines (FCCM'00), 2000, pp.68-76.

[15] Ernst, M, Henhapl, B.; Klupsch, S.; Huss, S. "FPGA based hardware acceleration for elliptic curve public key cryptosystems". Journal of Systems and Software Volume: 70, Issue: 3, March, 2004, pp. 299-313.

[16] Nele Mentens, Siddika Berna Ors, Bart Preneel. "An FPGA Implementation of an Elliptic Curve Processor over GF(2m)". in Proceedings of the 2004 ACM Great lakes Symposium on VLSI, GLSVLSI 2004: VLSI in the Nanometer Era. pp. 454-457

[17] Morales-Sandoval Miguel, Feregrino-Uribe Claudia. "On the Hardware Design of an Elliptic Curve Cryptosystem". Proceedings of the Fifth Mexican International Conference in Computer Science, ENC 2004: pp.64~70.

[18] T. Kerins, E. M. Popovici and W. P. Marnane. "An FPGA Implementation of a Flexible, Secure Elliptic Curve Cryptography Processor", istinguished paper, International Workshop on Applied Reconfigurable Computing-ARC 2005, IADIS press, pp.22-3.