Implementation of Elliptic Curve Scalar Multiplier on FPGA

V.Parthiban, Student
M. Tech - VLSI & Embedded System
B.S.Abdur Rahman University
Chennai, India
vjbparthi@gmail.com

Ms.S.Syed Rafiammal, Assistant Professor
Department of ECE
B.S.Abdur Rahman University
Chennai, India
syedrafiammal@gmail.com

***Abstract* — In this paper describes scalar multiplier using implementation of Elliptic Curve Cryptosystem (ECC) over the binary field GF (2^{163}).It illustrates suitable scheduling for performing point addition and doubling in Elliptic Curve Scalar Multiplier (ECSM).The detailed analyses, supported with experimental results are provided to design the fastest scalar multiplier over generic curves. It has been concluded from the comparison of proposed method is useful for high speed design and better utilization of LUT in FPGA.**

***Index Terms* — Elliptic Curve Cryptosystem, ECSM, LUT, FPGA, Galois Field.**

I. INTRODUCTION

Cryptography [1] uses mathematics to encrypt and decrypt data. It enables people to store or transmit sensitive information via insecure network. On the other hand, cryptanalysis is the science of breaking secure communication. There are two persons, Alice and Bob communicate via an insecure channel in a secure way. The third person who is eavesdropper should not be able to read the clear text or change it.

1. Symmetric key cryptography

2. Public key cryptography

A. Public key cryptography

In 1976, Diffie and Hellman presented public key cryptography (PKC), which is unlike traditional public-key. Diffie Hellman keys are not used to encrypt or decrypt the message. They are used to create a single shared secret key between the units.

Public key cryptography contains two keys, which are public and private keys. A situation is assumed where Alice wants to send a message to Bob. Alice uses Bob's Public key to encrypt a message and her private key to sign the message. Bob (receiver) uses his Private Key to decrypt the message and he uses Alice's Public Key to verify the signature. The standard bodies have set the key size of the encryption key, in order to provide the desired security. The key size decides the hardship of recovering the encrypted data computationally without the use of the secret key. A scenario of a public key is depicted in Figure 1 The key pair (e,d) is selected by Bob. 'e' is the public key that Bob sends to Alice over any channel but the private key 'd' is kept. Alice encrypts the sending message to Bob using Bob's public-key. Bob decrypts the cipher text 'c' using the 'd'.

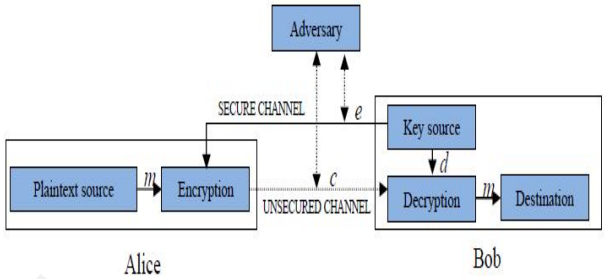

Fig. 1 Public-key cryptography

B. Elliptic curve Cryptography

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. This superiority translates to equivalent security levels with smaller keys, bandwidth savings, and faster implement- tations , making ECC very appealing. The IEEE proposed standard P1363-2000 recognizes ECC-based key agreement and digital signature algorithms. Elliptic curve-based cryptosystems are most closely related to algorithms like the Digital Signature Algorithm (DSA) which are based on the discrete logarithm problem. In the DSA, the parameters can be chosen to provide efficient implementations of the algorithm. In the same way, the parameters of ECC based cryptosystems can be selected to optimize the efficiency of the implementation. Unfortunately, the selection of the ECC parameters is not a trivial process and, if chosen incorrectly, may lead to an insecure system. In response to this issue NIST recommends ten finite fields, five of which are binary fields, for use in the ECDSA . For each field a specific curve, along with a method for generating a pseudo-random curve, are supplied. These curves have been intentionally selected for both cryptographic strength and efficient implementation. Such a recommendation has significant implications on design choices made while implementing elliptic curve cryptographic functions. In standardizing specific fields for use in elliptic curve cryptography (ECC), NIST allows ECC implementations to be heavily optimized for curves over a single finite field [2]. As a result, performance of the algorithm can be maximized and resource utilization,

whether it code size for software or logic gates for hardware, can be minimized.

The performance of an Elliptic Curve Cryptosystem is mostly determined by the efficient implementation of finite field arithmetic. In this work an efficient curve implementation for ECC over different approaches on finite field is presented. And we explained the Domain Parameters over prime field GF (p) and binary field GF (2^m)

II.MATHEMATICAL BACKGROUND

Elliptic curves over real numbers are defined as the set of points (x, *y)* which satisfy the elliptic curve equation of the form

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

Where a and b are real numbers. Each choice of a and b produces a different elliptic curve. The elliptic curve in forms a group if $4a^3+27b^2 \neq 0$. An elliptic curve group over real numbers consists of the points on the corresponding elliptic curve, together with a special point *O* called the point at infinity. Elliptic curve groups are an additive group that is, their basic function is addition. The negative of a point P = (x, y) is its reflection in the x-axis: the point - P is (x, -y). If the point P is on the curve, the point - P is also on the curve.

In elliptic curve cryptography we are only interested in elliptic curves defined over finite fields. This means that the coordinates of the points in the elliptic curve can only take values that belong to the finite field over which, the elliptic curve[3] has been defined. In particular we define elliptic curves over binary extension fields GF (2^m), using the following adjusted curve equation,

$$y^2+xy= x^3+ax^2+b \qquad (1)$$

Where a,b €GF (2^m) and $b \neq 0$. Once again, the elliptic curve includes all the points (x, y) that satisfy above equation in GF (2^m) arithmetic, plus the point at infinity *O*.

A. point addition

In elliptic curve P and Q are two different points. The Point addition is the line through two points to form the third point in elliptic curve[4], then tangent to the third point P+Q.

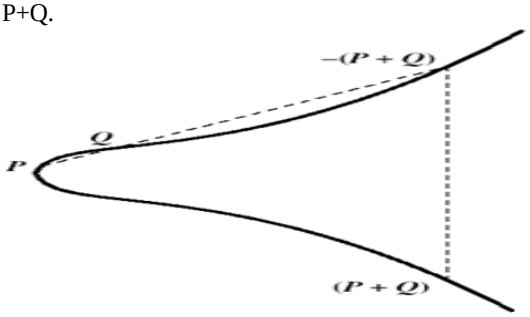

Fig.2 point addition

B.point doubling

In elliptic curve point is p.The point doubling is line through point form the second point ,then tangent to the second point R=2P.

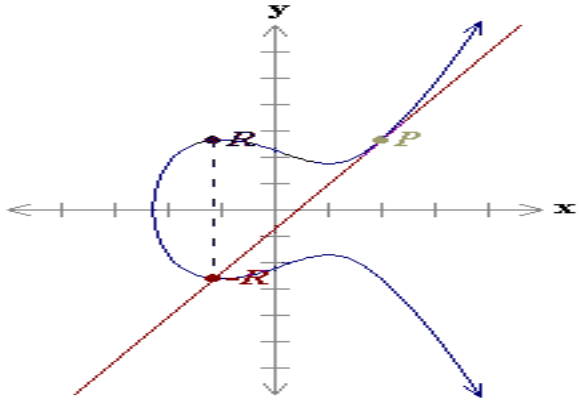

Fig.3 point doubling

C. Scalar multiplication in GF (2^m)

The elliptic curve scalar multiplier[5] (ECSM) following right –to-left scalar multiplication. The algorithm flow,

Y=k*p

k= $(k_{L-1}, k_{L-2}, k_{L-3}, \ldots, k_2, k_1, k_0)_2$

Y=0

S=P

for (i=0 t0 L-1)

{

if (k_i=1)

Y=y+s

S=2s

}

Above say that the scalar multiplication[10] performs using point addition and point doubling.

C. Karatsuba algorithm

The basic step of Karatsuba algorithm[8] can be used to compute the product of two large numbers a and b using three multiplications of smaller numbers, each with about half as many digits as a or b along with some additions and digit shifts.

Let a and b represent n-digit strings in some radix R. For any positive integer m less than n, the two numbers can be divided as follows,

$$a=a_iR^m+a_0 \qquad (2)$$

$$b=b_iR^m+b_0 \qquad (3)$$

Where a_0 and b_0 are less than R^m. The product is

$$a*b=(a_1R^m+a_0)(b_1R^m+b_0) \qquad (4)$$

$$a*b=a_1b_1R^{2m}+(a_1b_0+a_0b_1)R^m+a_0b_0 \quad (5)$$

$$a*b=u_2R^{2m}+u_1R^m+u_0 \quad (6)$$

Where,

$$u_2= a_1b_1$$

$$u_1= a_1b_0+a_0b_1$$

$$u_{0=} a_0b_0$$

These formulae require four numbers of multiplications. But, it can be observed that the value of the product ab can be determined using only three numbers of multiplications, at the cost of a few more number of additions in the following manner.

After obtaining,

$$u_{2=} a_1b_1 \text{and } u_{0=} a_0b_0,$$

the value of u_1 can be determined as

$$u_1= (a_1+a_0)(b_1+b_0)- u_{2-} u_0 \quad (7)$$

$$u_1= a_1b_0+a_0b_1=(a_1b_{1+} a_1b_0+a_0b_{1+} a_0b_0) - a_1b_{1-} a_0b_0$$

$$u_1= (a_1+a_0)(b_1+b_0)-a_1b_{1-} a_0b_0 \quad (8)$$

III. Experimental Results

This section shows our experimental results. The karatsuba multiplier is successfully experimented using Xilinx ISE 14.5 Simulator. The analysis of pipelined multiplier is targeted and verified on Xilinx FPGA of family Spartan-3E. The constraint taken to consideration is analysis of delay. The number of pipelined stages increasing throughput also increased.

A Different Types of multiplier

The below table I shows an experimental results of various multipliers.

TABLE I. Comparison of different multipliers

Comparison Factor/ Multiplier	*4bit Multiplier using add & shift method*	*4bit Multiplier using full adder*	*4bit Multiplier using fast adder*
DELAY	14.281 ns	13.781 ns	13.657 ns
LUT'S	33/4896	29/4896	29/4896
MEMORY	253968 kb	254992 kb	254544 kb

B. RTL View of 2 stage pipelined multiplier

The below Fig.4 shows the RTL view of two stage pipelined multiplier.

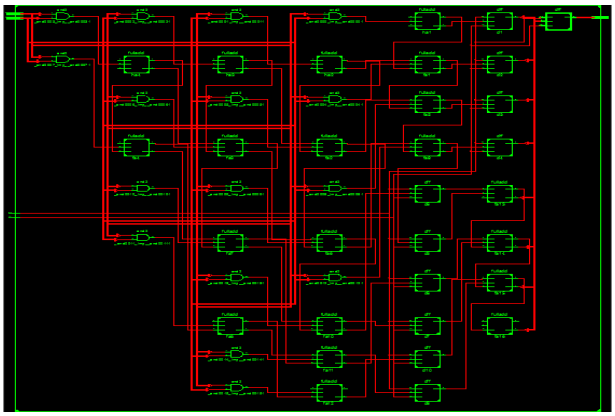

Fig. 4 RTL view 2 stage pipelined multiplier

C. Simulation of 2 stage pipelined multiplier

The below Fig. 5 shows the simulation result of two stage pipelined multiplier.

Fig. 5 output waveform of 2 stage pipelined multiplier

D. Simulation of karatsuba multiplier

The below Fig.6 shows the simulation result of karatsuba multiplier[9].

Fig. 6 Output waveform of karatsuba multiplier

IV. CONCLUSION

In this paper, we designed a kartsuba multiplier and point addition and point doubling over GF(2^{163}). The comparison of delay, area parameters in the design of different various multipliers are analyzed. Experimental results have shown that our approach is very effective in reducing delay compare to previous method of multiplier. The pipelined multiplier is increasing throughput and reducing delay of the multiplier.

REFERENCES

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

[2] L. Song and K. K. Parhi, "Low-energy digit-serial/parallel finite field multipliers," *J. VLSI Signal Process.*, vol. 19, no. 2, pp. 149–166, Jul.1998.

[3] J. López and R. Dahab, "Fast multiplication on elliptic curves over*GF(*2^m*)* without precomputation," in *Proc. 1st Int. Workshop Cryptographic Hardw. Embedded Syst*., 1999, pp. 316–327.

[4] G. Orlando and C. Paar, "A high performance reconfigurable elliptic curve processor for *GF(*2^m*),*" in *Proc. 2nd Int. Workshop Cryptographic Hardw. Embedded Syst*., 2000, pp. 41–56.

[5] Ansari and M. Hasan, "High-performance architecture of elliptic curve scalar multiplication," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1443–1453, Nov. 2008.

[6] W. N. Chelton and M. Benaissa, "Fast elliptic curve cryptography on FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 2, pp. 198–205, Feb. 2008.

[7] Rebeiro and D. Mukhopadhyay, "High speed compact elliptic curve cryptoprocessor for FPGA platforms," in *Proc. 9th Int. Conf. Cryptol. India: Progress Cryptol.*, 2008, pp. 376–388.

[8] Rebeiro and D. Mukhopadhyay, "Power attack resistant efficient FPGA architecture for Karatsuba multiplier," in *Proc. 21st Int. Conf. VLSI Design*, Jan. 2008, pp. 706–711.

[9] G. Zhou, H. Michalik, and L. Hinsenkamp, "Complexity analysis and efficient implementations of bit parallel finite field multipliers based on Karatsuba-Ofman algorithm on FPGAs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 7, pp. 1057–1066, Jul. 2010.

[10] C. Rebeiro, S. S. Roy, D. S. Reddy, and D. Mukhopadhyay, "Revisiting the Itoh-Tsujii inversion algorithm for FPGA platforms," *IEEE Trans.Very Large Scale Integr. (VLSI) Syst*., vol. 19, no. 8, pp. 1508–1512,Aug. 2011.

[11] K. Järvinen, "On repeated squarings in binary fields," in *Selected Areas in Cryptography* (Lecture Notes in Computer Science), vol. 5867,M. Jacobson, V. Rijmen, and R. Safavi-Naini, Eds. Berlin, Germany:Springer, 2009, pp. 331–349.

[12] F. Rodríguez-Henríquez, N. A. Saqib, A. Díaz-Pèrez, and C. K. K¸ oc,*Cryptographic Algorithms on Reconfigurable Hardware* (Signals and Communication Technology). Secaucus, NJ: Springer-Verlag, 2006

[13] M. Bednara, M. Daldrup, J. von zur Gathen, J. Shokrollahi, and J. Teich,"Reconfigurable implementation of elliptic curve crypto algorithms," inProc. 16th Int. Parallel Distrib. Process. Symp., 2002, pp. 157–164

[14] J. Lutz and A. Hasan, "High performance FPGA based elliptic curve cryptographic co-processor," in *Proc. Int. Conf. Inf. Technol: Coding Comput.*, vol. 2. Apr. 2004, pp. 486–492.

[15] Q. Pu and J. Huang, "A microcoded elliptic curve processor for *GF(*2^m*)* using FPGA technology," in *Proc. Int. Conf. Commun., Circuits Syst.*,vol. 4. Jun. 2006, pp. 2771–2775.