Volume 4 Issue IV, April 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

A High Speed FPGA Implementation of an ECSMA-Based Elliptic Curve Crypto Processor

P. Ravikumar[1], K.Giri[2]
[1]PG Scholar, ME (VLSI DESIGN), [2]Assistant Professor, Department ECE,
Mahendra Engineering College, Namakkal, Tamilnadu, India.

Abstract-- Elliptic Curve Cryptography (ECC), which allows smaller key length as compared to conventional public key cryptosystems, has become a very attractive choice in wireless mobile communication technology and personal communication systems. Any cryptosystem requires a very quick computation in very short time to get an optimal efficiency. One of the options could be to implement the overall cryptosystem into FPGA to reduce the execution time dramatically to pledge the efficiency. In this research, the ECC encryption engine has been implemented in Field Programmable Gate Arrays (FPGA) for two different key sizes, which are 131 bits and 163 bits. The cryptosystem, which has been implemented on Altera's EPF10K200SBC600-1, has taken 5945 and 6913 logic cells out of 9984 for the key sizes of 131 bits and 163 bits respectively with an operating frequency 43 MHz, and performs point multiplication operation in 11.3 ms and 14.9 ms for 131 bits and 163 bits implementation respectively. In terms of speed, the cryptosystem implemented on FPGA is 8 times faster than the software implementation of the same system.

Key words: Encryption, DES, 3DES, FPGA, synthesis, hardware.

I. INTRODUCTION

The Internet revolution in the last decade has enabled the success of e-commerce or electronic commerce over the world. The initial idea of e-commerce involves the conducting of business communication and transaction over remote computers. However, with the advent of new technology, e-commerce may no longer be limited to the use of computers, but involves small devices such as PDA, mobile phones, palmtop, and smartcard. The emergence of electronic commerce over the small devices implies that there is a greater need for faster and more secure transaction. Conventional public key cryptosystem such as RSA, Elgamal, and DSA may no longer be flexible to be implemented on these small, memory-constrained devices. This is because these cryptosystems require a relatively long key length (> 500 bits) to be intractable (Harper et al., 1992). The candidate remains is the Elliptic Curve Cryptosystem (ECC), which was first proposed by Miller (1986) and later by Koblitez (1987). ECC can be built with relatively shorter operand length of 130 to 200 bits as compared to RSA, which needs operands of 500 to 1024 bits (Guajardo, 1996).

This attractive feature makes ECC applicable in hardware-constrained environments such as hand phones and smartcards. Moreover, ECC is proven secured against known attacks, as there are no sub-exponential time algorithms to attack cryptosystems in this group (Menezes et al., 1993). ECC is currently standardized by IEEE standards committee (IEEE, 2000). ECC has short key length with high cryptographic strength as compared to RSA, DSA and Elgamal (Mazzeo et al., 2003; Swarup et al., 2004). There is no known Index Calculus Algorithm attack to the setting of ECC, while the RSA suffers from differential attack (Douglas, 2006). ECC hardware implementation use lesser transistor. Currently implementation of 155 bits ECC has been reported which uses only 11,000 transistors as compared to RSA 512-bits implantation, which used 50,000.

ECC is considered more secured than RSA. The largest size broken of ECC is 108 bits, which approximately needed 65,000 times as much as effort as breaking DES. Moreover, factoring of 512 bits RSA took only about 2% of the time required to break 108 bits ECC (Dhandapani and Kavita, 2010). ECC provides enhanced security since the underlying curve can be freely chosen which allows a frequent change of the encryption function. ECC provides wide variety of application such as key exchange, privacy through encryption, sender authentication and message integrity through digital signatures (Dhandapani and Kavita, 2010). BlackBerry is using ECC to ensure its security.

Market demands and the need for stronger security in BlackBerry drove the switch to 256-bit AES from 3DES. AES is the symmetric-key encryption algorithm recommended by NIST, and 256-bit is the current recommendation for classified government communications, which is necessitated BlackBerry to change in the matching public key algorithm, which is ECC. It is well recognized that hardware implementation of cryptographic ciphers provides better security and performance than software implementation (Coussy et al., 2009). However, the development cost is higher and the flexibility is reduced as compared to

software implementation. Kim et al. (2008) has proposed a FPGA implementation of high performance ECC processor over Galois Fields (GF).

In the computation of method, the problem is first divided into small pieces; each can be seen as a submodule in VHDL. Following the software verification of each submodule, the synthesis is then activated. It performs the translations of hardware description language code into an equivalent netlist of digital cells. The synthesis helps integrate the design work and provides a higher feasibility to explore a far wider range of architectural alternative. The method provides a systematic approach for hardware realization, facilitating the rapid prototyping of the Elliptic Curve Cryptography system. The system performance is investigated and compared to others implementation as well.

MATERIALS AND METHODS Background on elliptic curves Initially, elliptic curves have been used in the field of number theory to devise efficient algorithm for factoring integers and primality proving. The use of elliptic curve in the field of cryptography was proposed by N. Koblitz and V. Miller in 1986 and 1987 respectively. An elliptic curve is an equation of the form:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad (1)$$

From the above equations, the elliptic curves can be split into 2 classes, namely supersingular and non-supersingular curves. A supersingular elliptic curve is the set of solutions to the equations

$$y^2 + a_3y = x^3 + a_4x + a_6 \quad (2)$$

$$\text{where, } 4a_4{}^3 + 27a_6{}^2 \neq 0$$

A non-supersingular curve is the set of solutions to the equations:

$$y^2 + xy = x^3 + a_2x^2 + a_6 \quad (3)$$

where, . Since non-supersingular curve provides a far greater security than supersingular curve (Agnew et al., 1993), non-supersingular curve has been chosen for this research. By studying this kind of equation over various mathematical structures, such as real number, a ring or a field (Win et al., 1997), elliptic curve over a finite field has been considered. This is because calculations over the real numbers are slow and inaccurate due to round-off error and cryptographic applications require fast and precise arithmetic (Win et al., 1997). 0 6 □ a

II. RELATED WORK

Several hardware implementations to compute ECC scalar multiplication have been reported in the literature. Every technique has its pros and cons and requires fitting based on the application need. Many designs were dedicated for GF(2m) computation since it does not suffer the carry propagation problem. For example, in 1993, Agnew et al. Implemented ECC over GF(2155) normal basis finite field to be simple and gain efficient solution through an optimal multiplier. Their design used a programmable control processor that achieved high performance but limited to the finite field it is designed for. In 1998, Rosner worked on his thesis to develop a reconfigurable ECC crypto engine. His thesis hardware was dedicated for Galois fields GF(2n)m in standard His work proved that a full point multiplicationon ECC can be implemented on FPGAs although it is built for GF(2n)m.In 2000, Torii and Yokoyama [6] used efficient hardware techniques to implement ECC on adigital signal processor (DSP).

Their techniques improved modular multiplications based on Montgomery's multiplication method [14] but specified for pipeline processing on DSP. They devised an improved method for computing the number of multiplications and additions which enhanced computing the point doubling operation. Their ideas have been interesting but restricted to their targeted DSP hardware. In the same year, Bednara et al. presented a focus on field multiplications hardware analysis for ECC FPGA hardware implementation. They analyzed Montgomery field multipliers utilizing lookup tables to gain more efficiency. Their study compared Massey-Omura multipliers with LFSR in terms of area and speed. They evaluated different curve coordinate representations with respect to the number of operations within the fields. The best coordinate system matching their FPGA design was reported. In 2004, Saqib et al. described a parallel architecture for Computing Scalar Multiplication using Hessian Elliptic Curves over F(2191) on FPGA. The design aimed to be parallel in all levels and as general as possible without assuming any hardware type to gain the best possible speed. Their results have been interesting for GF(2m) parallel architecture.Several ECC hardware designs were introduced for GF(p) scalar multiplications, for example, in 2001, Orlando and Paar [21, 22] proposed an architecture for computation of point

multiplication for the ECC define over GF(p). Their architecture is scalable over area and speed and can easily be implemented on FPGA's. The processor used Montgomery multiplier (MM) for modular multiplications.

A. Scalar Multiplication Algorithm

The algorithm used for scalar multiplication is based on the binary method [34], since it is efficient for hardware implementation. The binary method algorithm is shown below:

```
Inputs:  k: a constant , P: point on the elliptic curve
Output: Q: another point on the elliptic curve, Q=k . P
Define: w: number of bits in k, where k_i is the i^th bit in k
If k_(w-1):=1 then Q := P else Q := 0;
for i:= w-2 down to 0 do
      Q := Q + Q;    Point Doubling
      If kw-1=1 then  Q := Q + P;    Point Addition
Return Q;
```

Basically, the binary method algorithm scans the bits of the constant k, in our case, from most to least bit and doubles the current point Q each time. After each point double operation, if the current k bit is one, then the algorithm adds the current point Q to the base point P. Each point operation, double or add, involves three elementary operations: modular multiplication, modular addition and modular multiplicative inverse.Finding multiplicative inverses in the field GF(p) is extremely slow, and is generally avoided as much as possible [7]. The use of coordinate systems other than the Affine coordinate system (will be illustrated later) greatly reduces the number of inversions required in the operations of the scalar multiplication on the expense of extra multiplications. ECC use effectively point doubling and addition operations in arithmetic execution. From many years of research, optimize formulae are available for the operations. Especially, by eliminating the costly field inversion from the main loop of the scalar multiplication, fast operations is achieved by using projective coordinates [32]. However, as in [33], the operation in projective coordinate involves more scalar multiplication than in affine coordinate and ECC on projective coordinate will be efficient only when the implementation of scalar multiplication is much faster than multiplicative inverse operation. Therefore, transfer is needed from one coordinate to another for avoiding the inversion process cost. The following section is dedicated for illustration of the coordinate systems structure used for these purposes.

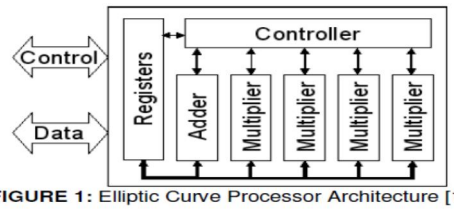

FIGURE 1: Elliptic Curve Processor Architecture [1]

III. ELLIPTIC CURVES OVER BINARY FIELDS GF(2N)

Finite Field or Galois Field is a set of finite number of elements,denoted as *GF(q)*. It shall be noted that *GF(q)* is a finite field consisting of *q* elements. For example, *GF(22)* consists of *22* elements (00, 01, 10, 11). Every element in *GF(2n)* can be represented as a polynomial $A(x) = anxn\text{-}1+\ldots.+a0$ with coefficients $ai \in \{0,1\}$. An elliptic curve with the underlying field *GF(2n)* is formed by choosing the curve coefficients *a2* and *a6* within *GF(2n)* (only condition is that *a6* is not 0).

A. Galois Field Arithmetic

Generally, there are 3 important arithmetic operations over the binary Galois Field (*GF(2n)*), which includes Addition, Multiplication and Inversion.

1) Addition: Addition in *GF (2n)* is a simple operation. Addition of 2 elements, $C(x) = A(x) + B(x)$, is performed by bitwise XORing the coefficients of the two polynomials, as follows:

2) Multiplication: The multiplication of 2 finite fields elements *A(x), B(x)* (2) $n \in GF$ can be performed as follows:

$C(x) = A(x) \cdot B(x) \bmod P(x)$ (5) where *P(x)* is the irreducible polynomial of the field *GF (2n).*

Volume 4 Issue IV, April 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

3) Inversion: Inversion is the most time consuming operation in Galois Field. The result is „1" for the multiplication operation between a field element and its inverse performed. The algorithm to get the inverse of an element:

B. Elliptic Curve Discrete Logarithm Problem (ECDLP)

At the foundation of every public key cryptosystem is a hard mathematical problem that is computationally infeasible to solve. The discrete logarithm problem is the basis for the security of many cryptosystems including the Elliptic Curve Cryptosystem. More specifically, the ECC relies upon the difficulty of the Elliptic Curve Discrete Logarithm Problem (ECDLP). In particular, for an elliptic curve *E*, the elliptic curve discrete logarithm problem (ECDLP) is given *Q, P* □□*E*, find the integer, k, such that (Blake et al., 1999), *Q = kP* (8) In fact, the security of the elliptic curve cryptosystem is based on
the presumed intractability of this problem. At present, the difficulty of the discrete logarithm on elliptic curve is orders of magnitude harder than others cryptosystems. This feature has made the Elliptic Curve Cryptosystem more powerful than others.

C. Elliptic Curve Cryptography

The elliptic curve discrete logarithm problem can be used as the basis for various public key cryptographic protocols such as key exchange, digital signatures, and encryption. In this project, the encryption process is considered only.

D. System Setup For Encryption

A Galois finite field *GF(2n)* is chosen on an elliptical curve with a point *P* lying in *GF*, *n* denotes
the order of *P*. *GF*, *P* and *n* is made public.

E. Secret Key Generation

Generate a random number k □□n-1**,**
Compute Q = kP
Point Q is made Public.
k is made private or secret key.

F. Encryption Process

(Suppose Alice sends a message *m* to Bob)
Look up Bob"s Public Key: *Q*
Represent the message *m* as a pair of the field elements (*M1,M2*), *M1* □□*GF*, *M2*□*GF*.
Select a random integer a, such that *a* □□*n-1*.
Compute the point (*X1, Y1*) = *aP*.
Compute the point (*X2, Y2*) = *aQ*.
Calculate *C1 = X2* □□*M1* and *C2 = Y2* □□*M2*.
Transmit the data *C* = (*X1, Y1, C1, C2*) to Bob.

G. Decryption

Bob gets the text message C from Alice)
Compute the point (*X2, Y2*) = *k* (*X1, Y1*), using its private key *k*.
Recover the message by calculating *M1 = X2*

H. Design Overview

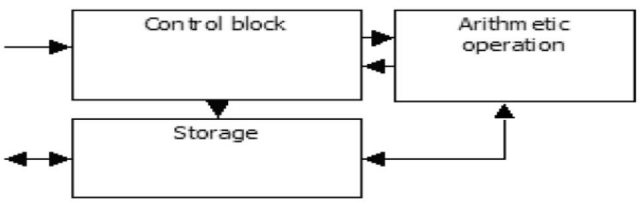

Figure 1. The top level design of the cryptosystem.

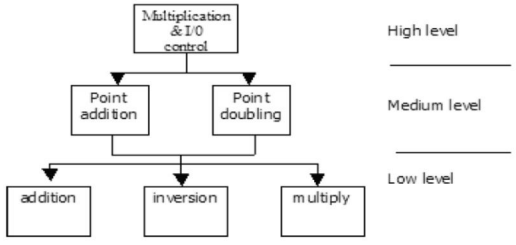

Figure 2. The design hierarchy of elliptic curve cryptosystem.

Volume 4 Issue IV, April 2016
ISSN: 2321-9653
International Journal for Research in Applied Science & Engineering Technology (IJRASET)

Figure 1 shows the top level design of the elliptic curve encryption engine. It consists of three major functional blocks, which are arithmetic operation block, control block, storage block. The arithmetic operation block is used to perform the arithmetic operation such as point doubling and point addition. The control block is used to control the arithmetic operation block in order to perform the encryption process. Lastly, the storage block is used to store the intermediate result from the arithmetic operation as well as the coefficients of the elliptic curve. **The Design Hierarchy** Figure 2 shows the design hierarchy of the elliptic curve encryption engine. The entire design process is divided into three levels. The low level defines the 3 basic finite field arithmetic operations, which are field addition, inversion and multiplication. By combining these operations, one can realize the operations of point doubling and point addition. The highest level of operation is point multiplication, which is the core operation in of the system.

I. Point Multiplication Algorithm

The task of point multiplication is to compute *kP*, where *k* is a positive integer and *P* is a point on the elliptic curve. This operation, as mentioned earlier, forms the basis of public key cryptography using elliptic curve. The standard method for point multiplication is the double-and-add algorithm as given in (Blake et al., 1999). In this algorithm, all the bits in binary representation of k except the first one are traversed from left to right. For each „0", a point doubling operation will be performed, and for each „1", a point doubling followed by a point addition operation will be performed. Since for a random *n* bit number *k*, a average of *n/2* bits is „1", the total number of operations for a complete point multiplication is about *n* doublings and *n/2* addition.

IV. RESULTS

Results were gathered from Quartus II after the synthesis process. Since two different key lengths have been implemented, which are 131 and 163, the results for both fields are given so that a comparison can be made. The results are presented in terms of maximum operating frequency and number of logic cells (LC) required. The device chosen for all implementations is EPF10K200SBC600-1 from family FLEX10KE. Table 1 shows the synthesis result form the top level of the Elliptic Curve Cryptosystem. For 131 bits key, the required area is 5945 logic cells with a maximum operating frequency of 45.87 MHz. For 163 bits key, 6913 logic cells are required with a maximum frequency of 43.38 MHz. From this result, it shows that to increase the security of the system from 131 bits to 163 bits, an additional of about 1000 logic cells are required.

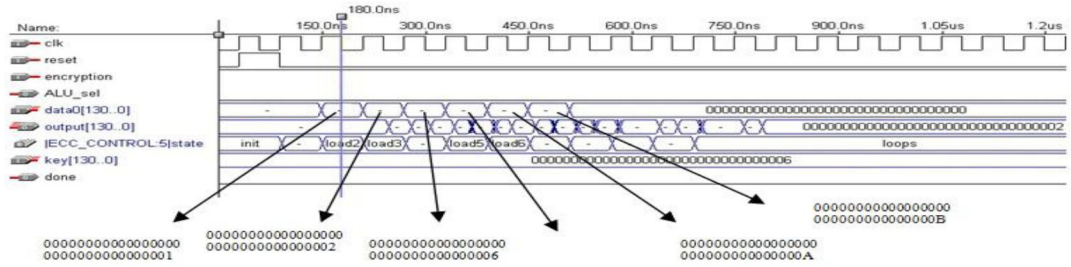

Figure 3. Timing simulation for encryption process (part 1).

Experimental Results and Analysis We have implemented and simulated the elliptic curve point multiplication with Xilinx's FPGA device. In order to show the effectiveness of hardware implementation over software based approach, we have also realized the design in software. We first provide the setups used in our work, then compare our FPGA based design with several previous works, and then show the difference between hardware and software implementations. 5.1 Software Implementation The software implementation of the elliptic curve point multiplication is done using C++ and LiDIA. LiDIA is a C++ library of computational number theory [17]. The simulation of the point multiplication in GF(2283) is based on Algorithm 1 and carried out on a Pentium4 2.8 GHz desktop with 1G memory. The source codes are compiled by GCC 4.1.1. The running time to perform a single Tate pairing operation is 9.6 ms.

V. HARDWARE IMPLEMENTATION

The purpose of the hardware implementation is to give some common platform and fair comparison between our proposed architecture and similar previous designs. The focus in this study is not targeted toward the details of the architecture implementation; instead our aim is to extract the hardware time and area parameters of the main blocks to build a fair comparison study between the designs. Therefore, our implementation exploration here is going to be limited to the level needed to serve this

comparison goal. We will implement the basic blocks of hardware that are commonly used to build all studied designs, i.e. our model here as well as similar previous architectures. The major common components needed by all designs are modular multiplier and modular adder. We described these designs in VHDL and synthesized them for Xilinx Spartan-3 FPGAs. The implementation features of the two basic components are detailed in this section.

A. FPGA Implementation

The hardware implementation is simulated by ModelSim XE and synthesized with Xilinx ISE 8.2i.The target device is Xilinx Virtex 4 XC4VFX140- FF1517-11. The optimization goal during synthesis is set as \speed", and the optimization e_ort is set to \normal". We have simulated the elliptic curve point addition, point doubling, coordinates converter and point multiplication in both software and hardware. The simulated latencies for these operations are shown in Table 2. Here, latency is the time to perform one specic arithmetic operation. The k values in our simulation have the same number of 1's and 0's in the binary representation. Point multiplication is the slowest module among other modules because it is composed of point addition, point doubling and coordinates converter. According to Table 2, the FPGA implementation of the point multiplication is 31.6 times faster than the software implementation. We compare the simulated latency with Leung's [10] and Ernst's work [9]
and show the results in Table 3. Our FPGA implementation of the point multiplication is 47 times faster than that in Leung's work (13.3 ms), and 22.5 times faster than that in Ernst's work (6.85 ms).

Design	Key Size	Latency
[10]	281	14.3 ms
[9]	270	6.85 ms
Our work	283	0.304 ms

Table 1: Comparisons of Latency of Point Multiplication.

VI. CONCLUSION

Hardware implementation of Elliptic Curve Cryptography encryption engine has been shown in this paper. The system is designed using VHDL, and implemented on a FPGA, EPF10K200SBC600-1 by Altera. For 163 bits key length, the system operates at a frequency of 43 MHz and performs the point multiplication operation in 14.9ms. This is much faster than the software implementation, where about 120 ms is required for the same operation. The cryptosystem is implemented in 2 different key lengths, 131 bits and 163 bits. From the synthesis result, increasing form 131 bits to 163 bits it only requires an additional of about 1000 logic cells in the FPGA, without degrading much on the timing performance. However, the security is gained by increasing 131 bits to 163 bits that is indeed the most attractive feature of elliptic curve cryptography. In summary, it is shown that elliptic curve cryptosystem can be efficiently implemented on a commercial FPGA, resulting in very flexible implementation with increased speed performance over the software solution.

REFERENCES

[1] N. Koblitz, "Elliptic curve cryptosystems", In Mathematics of Computation, volume 48, pages 203–209, 1987.
[2] V. Miller, "Use of elliptic curves in cryptography", Advances in Cryptology—CRYPTO'85, Vol. 218 of Lecture Notes in Computer Science, pages 417–426. Springer-Verlag, 1986.
[3] J.H. Cheon, H.J. Kim, S.G. Hahn, "Elliptic curve discrete logarithm and integer factorization",The Math Net Korea, Information Center for Mathematical Sciences (ICMS), February 7, 1999, http://mathnet.kaist.ac.kr/
[4] A Certicom Whitepaper, "The Elliptic Curve Cryptosystem", July 2000,http://www.certicom.com/
[5] Hitchcock, Yvonne Roslyn, "Elliptic Curve Cryptography for Lightweight Applications", Institution Queensland University of Technology, 2003. http://adt.library.qut.edu.au/adtqut/ public/adt-QUT20040723.150510/
[6] Naoya Torii and Kazuhiro Yokoyama, "Elliptic Curve Cryptosystem", FUJITSU Sci. Tech. Journal, Vol. 36, No. 2, pages 140-146, December 2000. www.fujitsu.com/downloads/MAG/vol36-2/paper05.pdf
[7] O. Al-Khaleel, C. Papachristou, F. Wolff, K. Pekmestzi, "An Elliptic Curve Cryptosystem Design Based on FPGA Pipeline Folding", 13th IEEE International On-Line Testing Symposium, IOLTS 07, pages 71 – 78, 8-11 July 2007.
[8] A.J. Menezes, T. Okamoto, S.A. Vanstone, S, "Reducing elliptic curve logarithms to logarithms in a finite field", IEEE Transactions on Information Theory, Volume 39, Issue 5, pages 1639 – 1646, Sept. 1993.
[9] T. Hasegawa, J. Nakajima, M. Matsui, "A practical implementation of elliptic curve cryptosystems over GF(p) on a 16-bit microcomputer", In Public Key Cryptography – PKC '98, Proceedings, volume 1431 of Lecture Notes in Computer Science, pages 182–194, Springer- Verlag, 1998.
[10] Scott Vanstone, "Crypto Column: The Importance of Good Crypto and Security Standards", Code & Cipher- Certicom's Bulletin of Security and Cryptography, Volume 1, Issue 4, 2004, http://www.certicom.com/codeandcipher