

Full Length Research Paper

FPGA implementation of elliptic curve cryptography engine for personal communication systems

Md. Syedul Amin^{1*}, Md. Mamun² and Jubayer Jalil¹

¹Department of Electrical, Electronic and Systems Engineering, Universiti Kebangsaan Malaysia, 43600, UKM, Bangi, Selangor, Malaysia.

²Smart Engineering System Research Group, Universiti Kebangsaan Malaysia, 43600, UKM, Bangi, Selangor, Malaysia.

Accepted 4 October, 2011

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.

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 Koblitz (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

*Corresponding author. E-mail: amin.syedul@gmail.com. Tel: +603-89216316. Fax: +603-89216146.

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). The proposed architecture is based on the López–Dahab elliptic curve point multiplication algorithm and uses Gaussian normal basis for GF field arithmetic. Their design is roughly 4.8 times higher than the work of Bajracharya et al. (2004). Chelton et al. (2008) proposed an efficient hardware implementation of ECC over GF. It introduces an improved Montgomery modular multiplication method for efficient hardware implementation and presents a fast mechanism of accelerating the elliptic curve point operation formulas by adopting modified Jacobian (Jm) coordinates. Their design is also in approximately millisecond range but the complexity is higher than this proposed research.

The Field-programmable gate arrays (FPGA) offers a potential alternative to speed up the hardware realization (Marufuzzaman et al., 2010; Reaz et al., 2007). From the perspective of computer-aided design, FPGA comes with the merits of lower cost, higher density and shorter design cycle (Akter et al., 2008). It comprises a wide

variety of building blocks. Each block consists of programmable look-up table and storage registers, where interconnections among these blocks are programmed through the hardware description language (Reaz et al., 2003). This programmability and simplicity of FPGA made it favorable for prototyping digital system. FPGA allows the users easily and inexpensively realize their own logic networks in hardware. FPGA also allows modifying the algorithm easily and the design time for the hardware becomes shorter by using FPGA (Yasin et al., 2004).

In this study, a unified framework for FPGA realization of ECC is designed by means of using a standard hardware description language VHDL for two different key sizes. The use of VHDL for modeling is especially appealing since it provides a formal description of the system and allows the use of specific description styles to cover the different abstraction levels (architectural, register transfer and logic level) employed in the design (Reaz et al., 2006). 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:

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

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, $a_6 \neq 0$.

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).

Elliptic curves over binary fields $GF(2^n)$

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, $GF(2^2)$ consists of 2^2 elements (00, 01, 10, 11). Every element in $GF(2^n)$ can be represented as a polynomial $A(x) = a_nx^{n-1} + \dots + a_0$ with coefficients $a_i \in \{0,1\}$. An elliptic curve with the underlying field $GF(2^n)$ is formed by choosing the curve coefficients a_2 and a_6 within $GF(2^n)$ (only condition is that a_6 is not 0).

Galois field arithmetic

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

Addition: Addition in $GF(2^n)$ 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:

$$C(x) = (C_{n-1}x^{n-1} + \dots + C_1x + C_0) + (A_{n-1}x^{n-1} + \dots + A_1x + A_0) + (B_{n-1}x^{n-1} + \dots + B_1x + B_0) \quad (4)$$

where, $A_i + B_i = C_i \in GF(2^n)$ and $C(x) \in GF(2^n)$

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

$$C(x) = A(x) \times B(x) \bmod P(x) \quad (5)$$

where $P(x)$ is the irreducible polynomial of the field $GF(2^n)$.

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:

$$B(x) = A^{-1}(x) \bmod P(x) \quad (6)$$

$$1 \equiv A(x) \times B(x) \bmod P(x) \quad (7)$$

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 $Q, P \in E$, find the integer, k , such that (Blake et al., 1999),

$$Q = kP \quad (8)$$

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.

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.

System Setup for Encryption: A Galois finite field $GF(2^n)$ 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.

Secret Key Generation:

- (i) Generate a random number $k \in n-1$,
- (ii) Compute $Q = kP$
- (iii) Point Q is made Public.
- (iv) k is made private or secret key.

Encryption Process: (Suppose Alice sends a message m to Bob)

- (i) Look up Bob's Public Key: Q
- (ii) Represent the message m as a pair of the field elements (M_1, M_2), $M_1 \in GF$, $M_2 \in GF$.
- (iii) Select a random integer a , such that $a \in n-1$.
- (iv) Compute the point $(X_1, Y_1) = aP$.
- (v) Compute the point $(X_2, Y_2) = aQ$.
- (vi) Calculate $C_1 = X_2 \times M_1$ and $C_2 = Y_2 \times M_2$.
- (vii) Transmit the data $C = (X_1, Y_1, C_1, C_2)$ to Bob.

Decryption: Bob gets the text message C from Alice)

- (i) Compute the point $(X_2, Y_2) = k(X_1, Y_1)$, using its private key k .
- (ii) Recover the message by calculating $M_1 = X_2^{-1} \times C_1$ and $M_2 = Y_2^{-1} \times C_2$.

Design overview

Figure 1 shows the top level design of the elliptic curve encryption

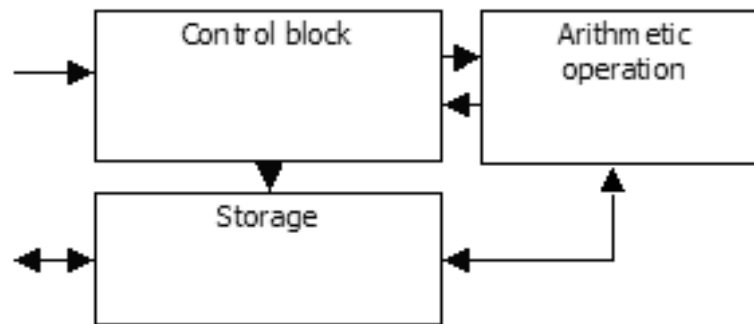


Figure 1. The top level design of the cryptosystem.

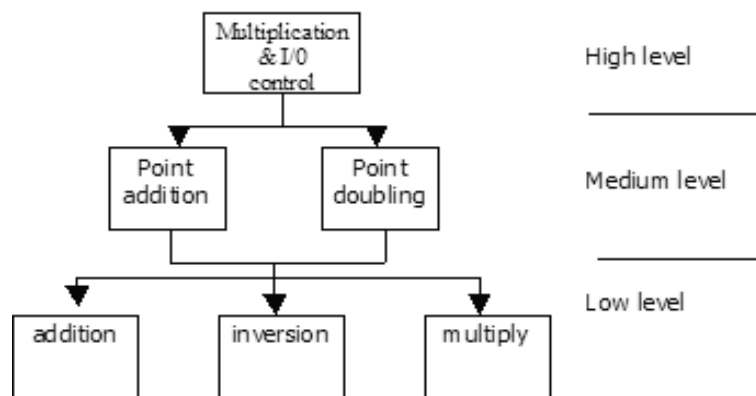


Figure 2. The design hierarchy of elliptic curve cryptosystem.

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.

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.

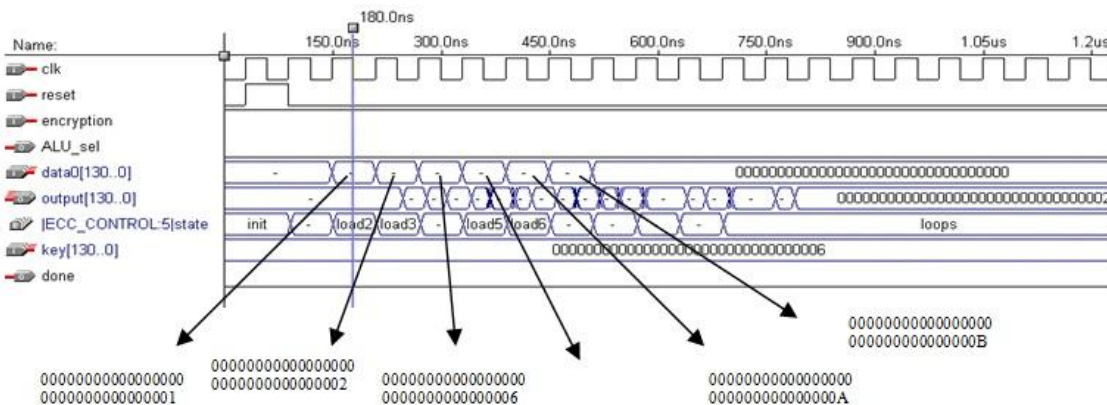
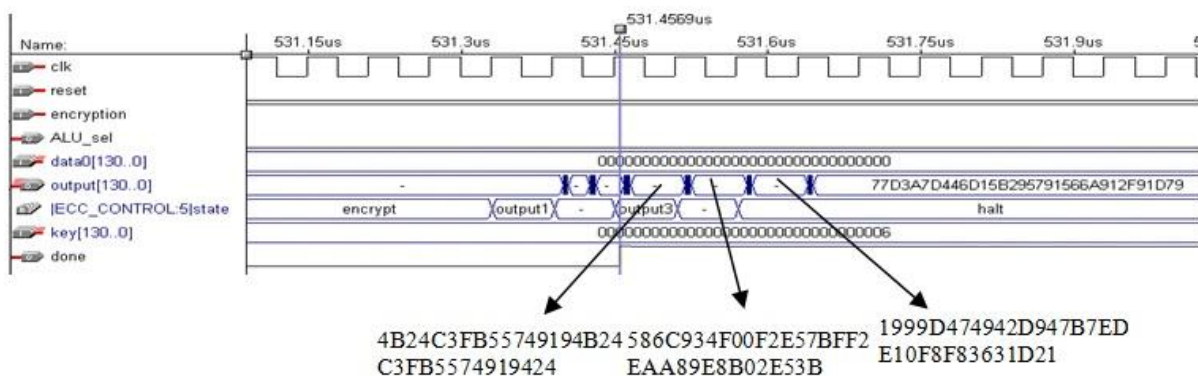
RESULTS

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.

Table 1. The synthesis result of the final design.

Key Length (bits)	Area (LC)	Clock period (ns)	Maximum operating frequency (MHz)
131	5945/9984 (59.8%)	21.8	45.87
163	6913/9984 (69.2%)	23.0	43.48

**Figure 3.** Timing simulation for encryption process (part 1).**Figure 4.** Timing simulation for encryption process (part 2).

However, the speed of the system (maximum frequency) does not degrade much with the increase size of the key.

Timing simulation

Encryption: The timing simulation for encryption process is shown in Figure 3 and 4. The “encryption” port is used to determine which operation to be performed. When it is ‘1’, the encryption process was carried out. Conversely, when it is ‘0’, the decryption process is executed. For encryption process, the input parameters are $P = (1, 2)$, $Q = 2P = (6, D)$ and the original message is (A, B) . After

the encryption process, the encrypted data is:

(4B24C3FB55749194B24C3FB5574919424, 586C934F00F2E57BFF2EAA89E8B02E53B, 1999D474942D947B7EDE10F8F83631D21, 7D3A7D446D15B295791566A912F91D79)

Input: $P = (1, 2)$, $Q = 2P = (6, D)$, plain message = (A, B) , secret key, $a = 6$
 Calculation: $6P = (X_1, Y_1) = (4B24C3FB55749194B24C3FB5574919424, 586C934F00F2E57BFF2EAA89E8B02E53B)$
 $6Q = (X_2, Y_2) =$

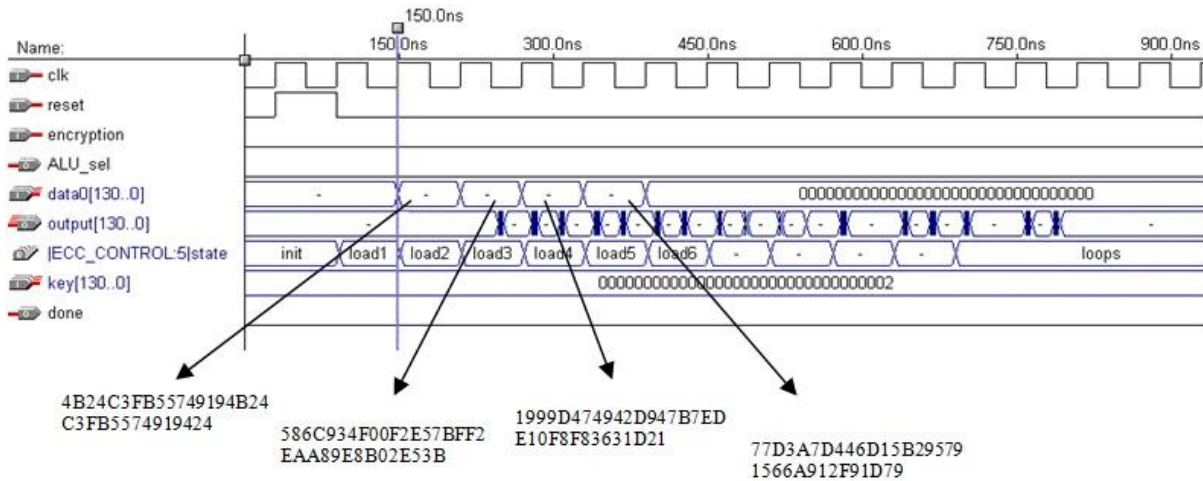


Figure 5. Timing simulation for decryption process (part 1).

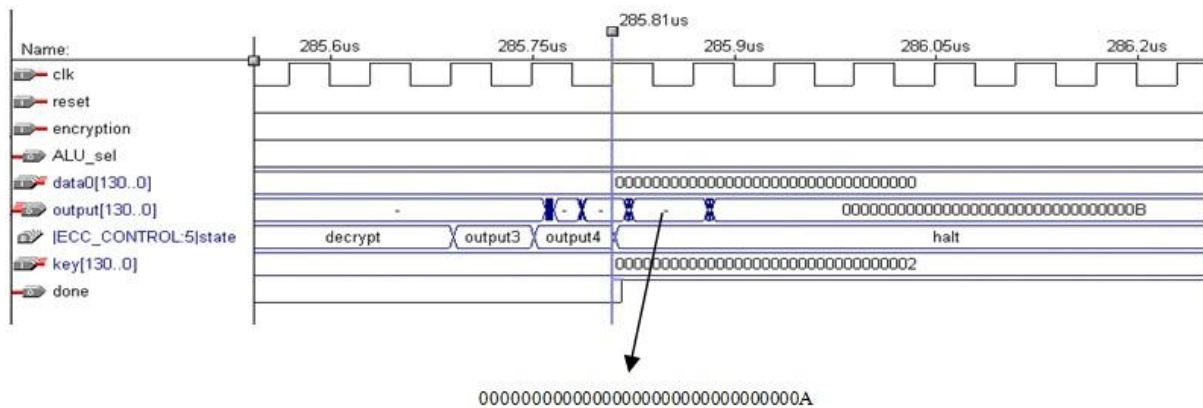


Figure 6. Timing simulation for decryption process (part 2).

(3C3C3758BDFB68A6D9B657E6B3F8F8307,69A74E89C0FBB1049E82C20F670467126)

$$C_1 = X_2 \times M_1 =$$

$$3C3C3758BDFB68A6D9B657E6B3F8F8307 \times A$$

$$=$$

$$1999D474942D947B7EDE10F8F83631D21$$

$$C_2 = Y_2 \times M_2 =$$

$$69A74E89C0FBB1049E82C20F670467126 \times B$$

$$=$$

$$77D3A7D446D15B295791566A912F91D79$$

Output: Encrypted data = (X₁, Y₁, C₁, C₂) =
 (4B24C3FB55749194B24C3FB5574919424,586C934F0
 0F2E57BFF2EAA89E8B02E53B,1999D474942D947B7E
 DE10F8F83631D21,77D3A7D446D15B295791566A912
 F91D79)

Decryption: For decryption process, the timing simulation is shown in Figure 5 and 6. To decrypt the encrypted message form during encryption simulation, the decryption simulation is performed. From this simulation, the original message, which is (A, B) is recovered.

Input: Encrypted message (X₁, Y₁, C₁, C₂) =
 (4B24C3FB55749194B24C3FB5574919424,586C934F0
 0F2E57BFF2EAA89E8B02E53B,1999D474942D947B7E
 DE10F8F83631D21,77D3A7D446D15B295791566A912
 F91D79).

Decryption key, $n = 2$

$$\text{Calculation: } (X_2, Y_2) = 2(X_1, Y_1) =$$

$$(3C3C3758BDFB68A6D9B657E6B3F8F8307,$$

$$69A74E89C0FBB1049E82C20F670467126)$$

Table 2. Times for various elliptic curve operations.

Key length (bits)	Operations	Average timing
131	Finite field addition	40.0 ns
	Finite field multiplication	3.38 μ s
	Finite field inversion	47.8 μ s
	Point addition	53.3 μ s
	Point doubling	59.9 μ s
	Point multiplication	11.3 ms
	Encryption	22.6 ms
	Decryption	11.4 ms
163	Finite field addition	45.0 ns
	Finite field multiplication	3.57 μ s
	Finite field inversion	50.5 μ s
	Point addition	56.3 μ s
	Point doubling	63.2 μ s
	Point multiplication	14.9 ms
	Encryption	29.8 ms
	Decryption	15.9 ms

$$X_2^{-1} = 407288F2DF187A49DC4E01F56E0ED720D$$

$$Y_2^{-1} = 5ACCBB167058CC9869E3AF1E945804EF$$

$$M_1 = X_2^{-1} \times C_1 = A$$

$$M_2 = Y_2^{-1} \times C_2 = B$$

Output: $(M_1, M_2) = (A, B)$

$$\text{Throughput} = \frac{2 \times \text{key sizes}}{\text{encryptionspeed}} \quad (9)$$

It is noted that the estimation takes into consideration that in each encryption process, 2 pieces of data can be encrypted using coordinate x and y, which have size equivalent to the system key sizes. By using the above estimation, the throughput of 131 bits implementation is 11.6 kbits / s and for 163 bits implementation are 10.9 kbits / s.

DISCUSSION

The timing requirement for various elliptic curve operations is listed in Table 2. This timing requirement is measured by assuming that the system is operating on maximum frequency. This means that the clock period for 131 bits key system is 21.8 ns and 163 bits key system is 23.0 ns. It is noted that the timings presented in Table 2 are only average value because different point on elliptic curve will yield slightly different value.

From the result, the time required to compute a 131 bits point multiplication is about 11.3ms. This is estimated by assuming that the number of '1' and '0' in the secret key are the same. For 163 bits implementation, it requires about 14.9 ms to perform the same operation. This is about 8 times faster than software implementation, where an average of 123 ms is required to compute 176 bits multiplication.

To estimate the throughput of the cryptosystem, the following formula can be used:

Conclusions

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 from 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

- Agnew GB, Mullin RC, Vanstone SA (1993). An implementation of elliptic curve cryptosystems over $F_{2^{155}}$. *IEEE J. Selected areas in Comm.*, 11(5): 804-813.
- Akter M, Reaz MBI, Mohd-Yasin F, Choong F (2008). Hardware Implementations of Image Compressor for Mobile Communications. *J. Comm. Technol. Elect.*, 53(8): 899-910.
- Bajracharya S, Shu C, Gaj K, El-Ghazawi T (2004). Implementation of elliptic curve cryptosystems over $GF(2^n)$ in optimal normal basis on a reconfigurable computer. 12 International Symposium on Field Programmable Gate Arrays, pp. 259-259.
- Blake I, Seroussi G, Smart N (1999). *Elliptic Curves in Cryptography*. Cambridge University Press, United Kingdom.
- Chelton WN, Benaissa M (2008). Fast Elliptic Curve Cryptography on FPGA, *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 16(2): 198-205.
- Coussy P, Gajski DD, Meredith M, Takach A (2009). An Introduction to High-Level Synthesis. *IEEE Design & Test of Computers*, 26(4):8-17.
- Dhandapani S, Kaviitha C (2010). Network Security using Elliptic Curve Cryptography. *Technology Today*, 2(4): 209-217.
- Douglas RS (2006). *Cryptography: Theory and Practice*. CRC Press.
- Guajardo J (1996). Efficient Algorithms for Elliptic Curve Cryptosystem. Master's thesis, ECE Dept., Worcester Polytechnic Institute, Worcester.
- Harper G, Menezes A, Vanstone S (1992). Public-key cryptosystems with very small key lengths. *Advances in Cryptology - Eurocrypt '92*. pp. 163-173.
- IEEE Standard Specifications for Public-Key Cryptography. IEEE-SA Standards Board. Approved 30 January 2000.
- Kim CH, Kwon S, Hong CP (2008). FPGA implementation of high performance elliptic curve cryptographic processor over $GF(2^{163})$. *J. Syst. Archit.*, 54: 893-900.
- Koblitz N (1987). Elliptic curve cryptosystems. *Math. Comput.*, 48:203-209.
- Marufuzzaman M, Reaz MBI, Rahman MS, Ali MAM (2010). Hardware Prototyping of an intelligent current dq PI controller for FOC PMSM drive. 6th Int. Conf. Elect. Comput. Eng., pp. 86-88.
- Mazzeo A, Romano L, Saggese GP, Mazzocca N (2003). FPGA-based Implementation of a serial RSA processor. *Design, Automation and Test in Europe Conference and Exhibition (DATE'03)*, pp. 582-587.
- Menezes A, Vanstone S (1993). Elliptic curve cryptosystems and their implementation. *J. Cryptography.*, 6(4):209-224.
- Miller V (1986). Use of Elliptic Curves in Cryptography. *Advances in Cryptology - Crypto '85*, pp. 417-428.
- Reaz MBI, Choong F, Mohd-Yasin F (2006). VHDL Modeling for Classification of Power Quality Disturbance Employing Wavelet Transform, Artificial Neural Network and Fuzzy Logic. *Simulation: Trans. Soc. Model. Simul. Int.*, 82(12):867-881.
- Reaz MBI, Choong F, Sulaiman MS, Mohd-Yasin F (2007). Prototyping of Wavelet Transform Artificial Neural Network and Fuzzy Logic for Power Quality Disturbance Classifier. *J. Electric Power Compon. Syst.*, 35:1-17.
- Reaz MBI, Islam MT, Sulaiman MS, Ali MAM, Sarwar H, Rafique S (2003). FPGA realization of multipurpose FIR filter. *Parallel and Distrib. Comput. Appl. Technol.*, pp. 912-915.
- Swarup R, Adluru S (2011). ElGamal Cryptosystem - Applications, Performance and Comparison with RSA, <http://www-rcf.usc.edu/~mdhuang/cs556/Applications-ElGamal.ppt>.
- Win ED, Prenel B (1997). Elliptic Curve Public Key Cryptosystem-an introduction. *Lecture Notes Comput. Sci.*, pp. 131-141.
- Yasin FM, Tan AL, Reaz MI (2004). The FPGA prototyping of Iris recognition for biometric identification employing neural network. *Int. Conf. Microelectron.*, pp. 458-461.