FPGA Implementation of Elliptic Curve Cryptography Engine for Personal Communication Systems

M. B. I. REAZ, J. JALIL, H. HUSIAN, F. H. HASHIM
Department of Electrical, Electronic and Systems Engineering,
Universiti Kebangsaan Malaysia,
43600, UKM, Bangi, Selangor
MALAYSIA
mamun@vlsi.eng.ukm.my http://www.ukm.my

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. 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, ECC, FPGA, Synthesis, Hardware

1 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 due to the fact that these cryptosystems require a relatively long key length (> 500 bits) to be intractable [1].

The candidate remains is the Elliptic Curve Cryptosystem (ECC), which was first proposed in 1985 by N. Koblitz [2] and V. Miller [3]. ECC can be built with relatively shorter operand length of 130-200 bits as compared to RSA, which needs operands of 500-1024 bits [4]. This attractive feature makes ECC applicable in hardware-constrained environments such as hand phones and smartcards. Moreover, ECC is proven to be secured against known attacks as there are no sub-exponential time algorithms to attack cryptosystems in this group [5]. ECC is currently standardized by IEEE standards committee [6].

ECC has short key length with high cryptographic strength as compared to RSA, DSA and Elgamal [7,8,9]. There is no known Index Calculus Algorithm attack to the setting of ECC, while the RSA suffers from differential attack [10]. 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 [11]. ECC is considered to be 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 [11]. ECC provides enhanced security since the underlying curve can be freely chosen which allows a frequent change of the encryption function [12]. ECC provides wide variety of application such as key exchange, privacy through encryption, sender authentication and message integrity through digital signatures [12].

It is well recognized that hardware implementation of cryptographic ciphers provides better security and performance than software implementation [13]. However, the development

cost is higher and the flexibility is reduced as compared to software implementation.

The Field-programmable gate arrays (FPGA) offers a potential alternative to speed up the hardware realization [14-16]. From the perspective of computer-aided design, FPGA comes with the merits of lower cost, higher density, and shorter design cycle [17-18]. 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 [19-21]. This programmability and simplicity of FPGA made it favorable for prototyping digital system. FPGA allows the users to easily and inexpensively realize their own logic networks in hardware. FPGA also allows modifying the algorithm easily and the design time frame for the hardware becomes shorter by using FPGA [22-23].

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 [24-26]. 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 [27-28]. The method provides a systematic approach for hardware realization, facilitating the rapid prototyping of the Elliptic Curve Cryptography system. The performance of the system is investigated and compared to others implementation as well.

2. 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 [2] and V. Miller [3] in 1985.

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 [29], 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 [30], 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 [30].

2.1 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 $A(x) = a_nx^{n-1}+.....+a_0$ 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).

2.1.1 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:

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

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:

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

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

2.2 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 [31],

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

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.

2.3 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. In this section, the encryption protocol for Elliptic Curve Cryptography is given.

2.3.1 Encryption

System Setup: 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:

- Generate a random number $k \in n-1$
- Compute $Q = kP$
- Point Q is made Public.
- k is made private or secret key.

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 (M_1, M_2), $M_1 \in GF$, $M_2 \in GF$.
- Select a random integer a, such that $a \in n-1$.
- Compute the point $(X_1, Y_1) = aP$.
- Compute the point $(X_2, Y2) = aQ$.
- Calculate $C_1 = X_2 \times M_1$ and $C_2 = Y_2 \times M_2$.
- Transmit the data $C = (X_1, Y_1, C_1, C_2)$ to Bob.

2.3.2 Decryption

(Bob gets the text message C from Alice)

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

3. Design Overview

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.

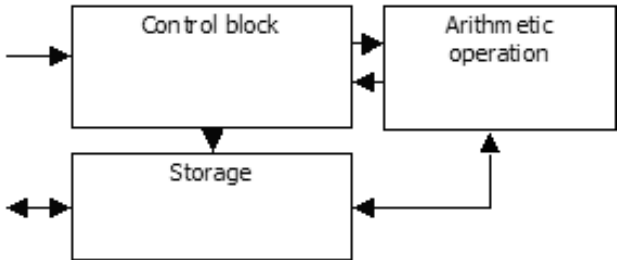

Fig. 1: The top level design of the cryptosystem

3.1 The Design Hierarchy

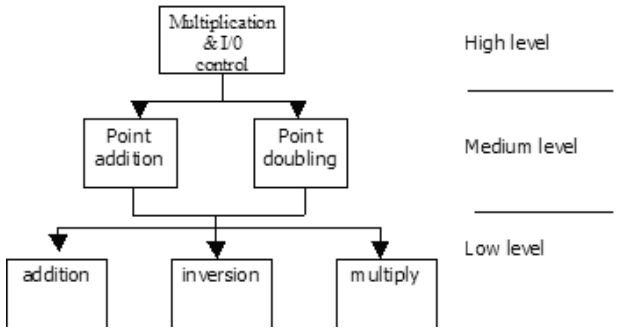

Fig. 2: The design hierarchy of elliptic curve cryptosystem

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 [31]. 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.

3.2 Results and Discussion

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

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

3.3 Timing Simulation

3.3.1 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,1999D474942D947B7EDE10F8F83631D21,77D3A7D446D15B295791566A912F91D79*)

Input:
$P = (1, 2)$, $Q = 2P = (6, D)$, plain message = (A, B), secret key, $a = 6$

Calculation:
$6P=(X_1,Y_1)=(4B24C3FB55749194B24C3FB5574919424,586C934F00F2E57BFF2EAA89E8B02E53B)$
$6Q=(X_2,Y_2)=(3C3C3758BDFB68A6D9B657E6B3F8F8307,69A74E89C0FBB1049E82C20F67046712)$
$C_1=X_2 \times M_1$
$=3C3C3758BDFB68A6D9B657E6B3F8F8307 \times A$
$=1999D474942D947B7EDE10F8F83631D21$
$C_2=Y_2 \times M_2$
$=69A74E89C0FBB1049E82C20F670467126 \times B$
$= 77D3A7D446D15B295791566A912F91D79$

Output:
Encrypted data=(X_1, Y_1, C_1, C_2) $=(4B24C3FB55749194B24C3FB5574919424,586C934F00F2E57BFF2EAA89E8B02E53B,1999D474942D947B7EDE10F8F83631D21,77D3A7D446D15B295791566A912F91D79)$.

3.3.2 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 (X_1, Y_1, C_1, C_2) = $(4B24C3FB55749194B24C3FB5574919424,586C934F00F2E57BFF2EAA89E8B02E53B,1999D474942D947B7EDE10F8F83631D21,77D3A7D446D15B295791566A912F91D79)$.
Decryption key, $n = 2$

Calculation:
$(X_2, Y_2) = 2(X_1, Y_1)$ = $(3C3C3758BDFB68A6D9B657E6B3F8F8307, 69A74E89C0FBB1049E82C20F670467126)$
$X_2^{-1} = 407288F2DF187A49DC4E01F56E0ED720D$
$Y_2^{-1} = 55ACCBB167058CC9869E3AF1E945804EF$
$M_1 = X_2^{-1} \times C_1 = A$
$M_2 = Y_2^{-1} \times C_2 = B$

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

3.4 Performance Analysis

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 11.3 ms. 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:

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

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 μs
	Finite field inversion	47.8 μs
	Point addition	53.3 μs
	Point doubling	59.9 μs
	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 μs
	Finite field inversion	50.5 μs
	Point addition	56.3 μs
	Point doubling	63.2 μs
	Point multiplication	14.9 ms
	Encryption	29.8 ms
	Decryption	15.9 ms

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 is 10.9 kbits / s.

4 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 14.9 ms. 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 which 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] G. Harper, A. Menezes and S. Vanstone, Public-key Cryptosystems with Very Small Key Lengths, Advances in Cryptology - Eurocrypt '92, Lecture Notes in Computer Science 658, Springer-Verlag Berlin, Vol. 658/1993, 1993, pp. 163-173.

[2] N. Koblitz, Elliptic Curve Cryptosystems, Mathematics of Computation, 1987, Vol. 48, No. 177, pp. 203-209.

[3] V. Miller, Use of Elliptic Curves in Cryptography, Advances in Cryptology - Crypto '85 proceedings, Lecture Notes in Computer Science, Springer-Verlag Berlin, Vol. 218/1986, 1986, pp. 417-426.

[4] J. Guajardo and Christof Paar, Efficient Algorithms for Elliptic Curve Cryptosystem, Advances in Cryptology — CRYPTO '97, Lecture Notes in Computer Science, Springer-Verlag Berlin, Vol. 1294/1997, 1997, pp. 342-356.

[5] A. Menezes and S. Vanstone, Elliptic curve cryptosystems and their implementation, Journal of Cryptography, Vol. 6, No. 4,1993, pp. 209-224.

[6] IEEE Standard Specifications for Public-Key Cryptography. IEEE-SA Standards Board. Approved 30 January 2000, 227 pages.

[7] A. Mazzeo, L. Romano, G. P. Saggese and N. Mazzocca, FPGA-based Implementation of a serial RSA processor, In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'03), Munich, Germany, 2003, pp. 582-587.

[8] Rajat Swarup, Sumanth Adluru, ElGamal Cryptosystem - Applications, Performance and Comparison with RSA, 2004. http://www-rcf.usc.edu/~mdhuang/cs556/Applications-ElGamal.ppt.

[9] Dr. Gerhard Schabhüser, How to find the socially accepted minimal Key length for Digital Signature Algorithms, 2002. URL:http://www.cacr.math.uwaterloo.ca/conferences/2002/ecc2002/schabhueser.pdf

[10] Douglas R. Stinson, Cryptography: Theory and Practice, 2nd Edition, CRC Press, 1995. ISBN:1584882069

[11] Amit N. Gathani, Implementation of Elliptic Curve Cryptosystem in Embedded System: A brief overview, 2001. http://eref.uqu.edu.sa/files/Others/Elliptic%20Curves/Implementation%20Of%20Elliptic%20Curve%20Cryptography%20In%20Embedded%20Sy.pdf

[12] T. Kerins, E.M. Popovici, A. Daly and W. Marnace, Hardware Encryption Engines for E-Commerce, Irish Signals and Systems Conference 2002, Ireland, 24- 26 June, 2002, pp.89-94.

[13] Abdullah Al Noman, Roslina Mohd Sidek, Abdul Rahman Ramli, Hardware Implementation of RC4A Stream Cipher, International Journal of Cryptology Research, Vol. 1, Issue2, 2009, pp 225-233.

[14] P. Coussy, D. D. Gajski, M. Meredith and A. Takach, An Introduction to High-Level Synthesis, IEEE Design & Test of Computers, Vol. 26, No. 4, 2009, pp. 8-17.

[15] Mohd. Marufuzzaman, M. B. I. Reaz, M. S. Rahman and M.A. Mohd. Ali, Hardware

Prototyping of an intelligent current dq PI controller for FOC PMSM drive.In Proceedings of the 6th International Conference on Electrical and Computer Engineering (ICECE 2010), Dhaka, Bangladesh,18-20 December, 2010, pp.86-88.

[16] M. B. I. Reaz, F. Choong, M. S. Sulaiman and F. Mohd-Yasin, Prototyping of Wavelet Transform, Artificial Neural Network and Fuzzy Logic for Power Quality Disturbance Classifier, Journal of Electric Power Components and Systems, Vol. 35, No.1, 2007, pp.1-17.

[17] F. Choong, M. B. I. Reaz, F. Mohd-Yasin; Power Quality Disturbance Detection Using Artificial Intelligence: A Hardware Approach. In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05), Denver, USA , 4-5 April, 2005, pp. 146a-146a.

[18] M. Akter, M. B. I. Reaz, F. Mohd-Yasin and F. Choong, Hardware Implementations of Image Compressor for Mobile Communications, Journal of Communications Technology and Electronics, Vol. 53, Issue 8, 2008, pp. 899-910.

[19] M. B. I. Reaz, F. Mohd-Yasin, M. S. Sulaiman, K. T. Tho and K. H. Yeow, Hardware Prototyping of Boolean Function Classification Schemes for Loss-less Data Compression, In Proceedings of the Second IEEE International Conference on Computational Cybernetics (ICCC 2004), Vienna, Austria, 30 August -01 September, 2004, pp. 47-51.

[20] M. B. I. Reaz, M. T. Islam, M. S. Sulaiman, M. A. M. Ali, H. Sarwar and S. Rafique, FPGA Realization of Multipurpose FIR filter, In Proceedings of the Parallel and Distributed Computing, Applications and Technologies (PDCAT), Chengdu, China, 27-29 August, 2003, pp. 912-915.

[21] M. B. I. Reaz, F. Mohd-Yasin, S. L. Tan, H. Y. Tan and M. I. Ibrahimy, Partial Encryption of Compressed Images Employing FPGA,In Proceedings of the IEEE Int. Symposium on Circuits and Systems (ISCAS 2005), Kobe, Japan, 23-26 May, 2005, pp 2385-2388.

[22] F. Choong, M. B. I. Reaz, T. C. Chin and F. Mohd-Yasin, Design and Implementation of a Data Compression Scheme: A Partial Matching Approach, In Proceedings of the Computer Graphics, Imaging and Visualisation: Techniques and Applications (CGIV'06), Sydney, Australia, 26-28 July, 2006, pp. 150 – 155.

[23] M. I. Ibrahimy, M. B. I. Reaz, M. A. Mohd Ali, T. H. Khoon and A. F. Ismail, Hardware Realization of an Efficient Fetal QRS Complex Detection Algorithm, WSEAS Transactions on Circuits and Systems, Vol. 5, No. 4, 2006, pp. 575-581.

[24] W. L. Pang, M. B. I. Reaz, M. I. Ibrahimy, L. C. Low, F. Mohd-Yasin and R.A. Rahim, Handwritten Character Recognition using Fuzzy Wavelet: A VHDL Approach, WSEAS Transactions on Systems, Vol. 5, No. 7, 2006, pp. 1641-1647.

[25] M. B. I. Reaz, F. Choong and F. Mohd-Yasin, VHDL Modeling for Classification of Power Quality Disturbance employing Wavelet Transform, Artificial Neural Network and Fuzzy Logic, SIMULATION: Transactions of the Society for Modeling and Simulation International, Vol. 82, No. 12, 2006, pp. 867-881.

[26] M. B. I .Reaz, M. I. Ibrahimy, F. Mohd-Yasin, C. S. Wei and M. Kamada, Single Core Hardware Module to Implement Encryption in TECB Mode, Informacije MIDEM - Journal of Microelectronics, Electronic Components and Materials, Vol. 37, No. 3, 2007, pp. 165-171.

[27] M. B. I. Reaz, P. W. Leong, F. Mohd-Yasin and T. C. Chin, Modeling of Data Compression using Partial Matching: A VHDL Approach,In Proceedings of the 6th World Wireless Congress (WWC), Palo Alto, USA, 25-27 May 2005, pp 411-416.

[28] M. B. I. Reaz, M. S. Sulaiman, F. M. Yasin and T. A. Leng, IRIS recognition using neural network based on VHDL prototyping, In Proceedings of the 2004 International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA 2004), Damascus, Syria, 19-23 April, 2004, pp. 463-464.

[29] G.B. Agnew, R.C. Mullin and S.A. Vanstone, An Implementation of Elliptic Curve Cryptosystems over F2155, IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, 1993, pp. 804-813.

[30] Erik De Win and Bart Prenel, Elliptic Curve Public Key Cryptosystem - An Introduction, State of the Art in Applied Cryptography, Lecture Notes in Computer Science ,

Springer-Verlag,Berlin, 1998, Vol. 1528/1998, pp.131-141.

[31] Ian Blake, Gadiel Seroussi, and Nigel Smart, Elliptic Curves in Cryptography, London Mathematical Society Lecture Note Series: 265, 1st Edition, Cambridge University Press, United Kingdom, 1999. ISBN: 0521653746

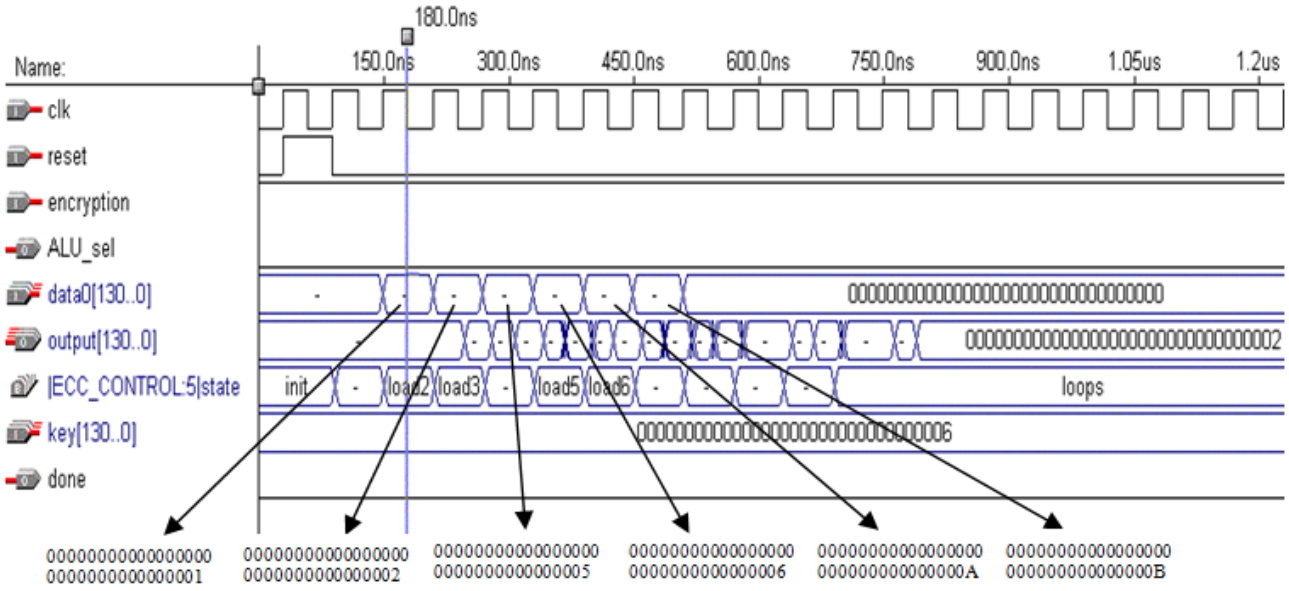

Fig. 3: Timing simulation for encryption process (part 1)

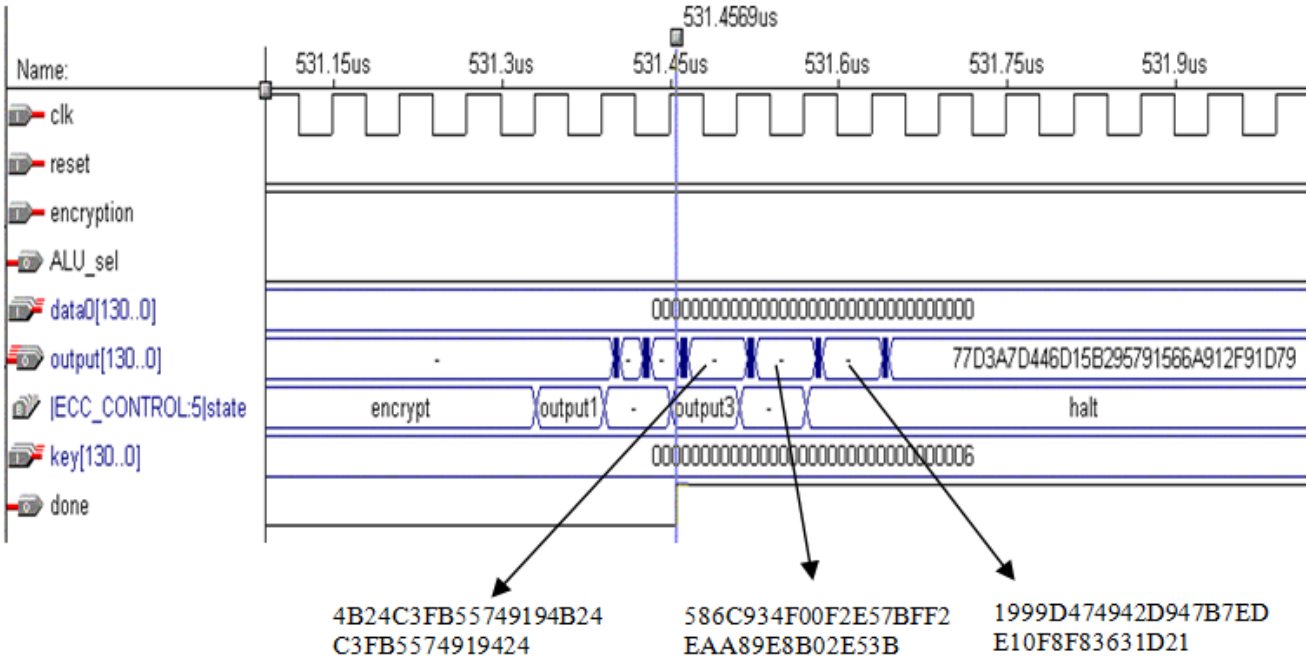

Fig. 4: Timing simulation for encryption process (part2)

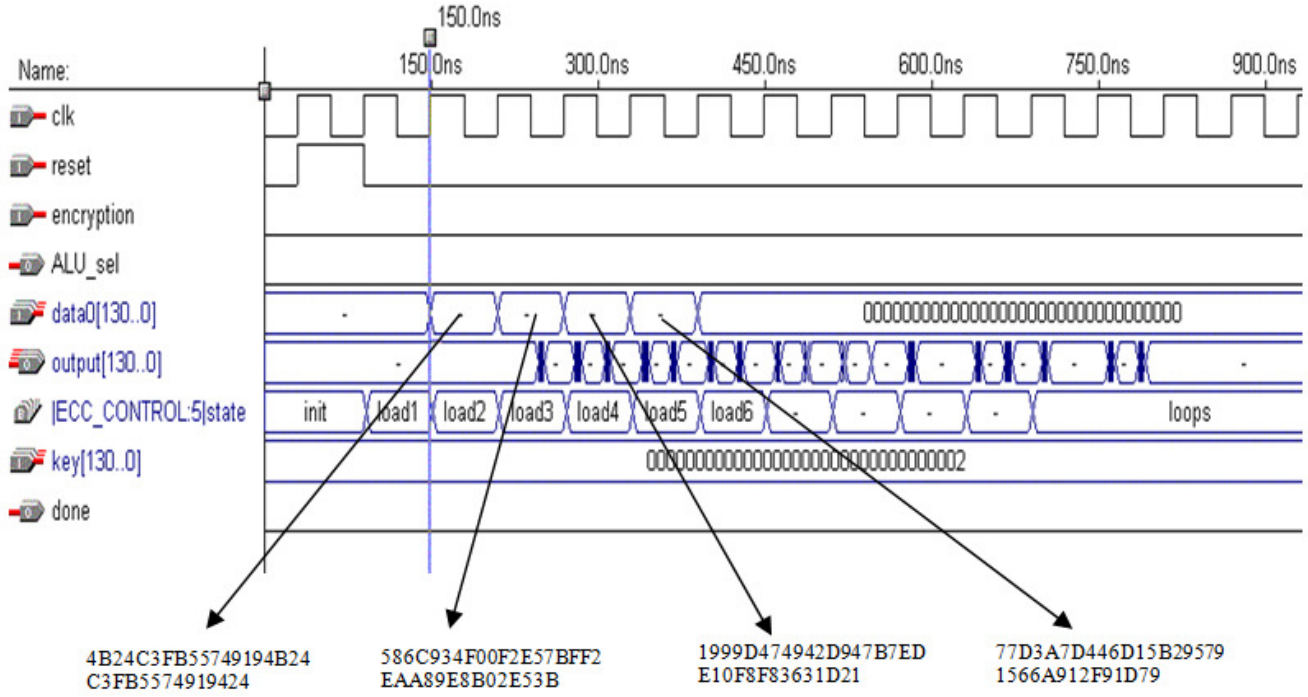

Fig. 5: Timing simulation for decryption process (part 1)

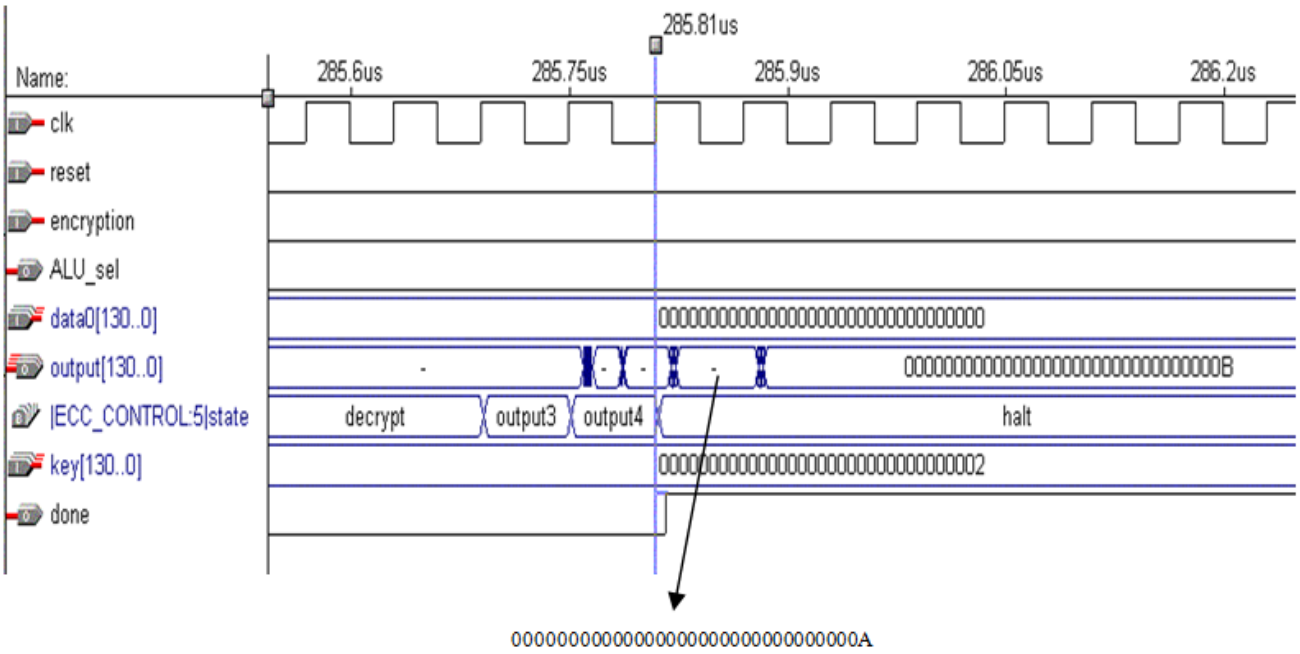

Fig. 6: Timing simulation for decryption process (part 2)