ISSN: 2349- 5197
Impact Factor (PIF): 2.138

INTERNATIONAL JOURNAL OF RESEARCH SCIENCE & MANAGEMENT

FPGA DESIGN AND SIMULATION OF MODIFIED MODULAR INVERSE FOR ELLIPTIC CURVE CRYPTOGRAPHY

Mahesh Kumar A S[*1], Jithendra P R Nayak[2], Naveen Kumar M S[3]
[1*] M Tech Scholar, Department of Electronics and Communication, VTU-RC, PG Studies, Mysuru-570029, INDIA
[2, 3] Assistant professor, Department of Electronics and Communication, GMIT, Mandya-571422, INDIA
Correspondence Author*:* as.mahesh.ec@gmail.com

Keywords: Field programmable gate arrays (FPGA), Elliptic curve cryptography (ECC), Binary Inversion Algorithm (BIA), and GF (p) arithmetic operators.

Abstract
Elliptic Curve Cryptography is a public-key cryptography based on the algebraic structure of elliptic curves over finite fields. Elliptic Curve Cryptography recently gained a lot of attention in security areas. ECC contrasted with RSA is that it utilizes littler keys to accomplish equivalent level of security, in this manner decreasing handling overhead. With Elliptic Curve Cryptography abnormal state of security can be accomplish with minimal effort, little key size and littler equipment acknowledgment. The fundamental operation of ECC is point multiplication. This point multiplication includes operations like point addition and point doubling. These operation in turn include addition, subtraction, multiplication, inversion and squaring over a prime number. Out of these operation inversion is the most expensive modular operation with data dependent delays in ECC processor. A review on various algorithms to perform modular inverse on prime fields more effectively has been carried out. There are several inversion algorithms like Euclidean Algorithm, Extended Euclidean Algorithm, Montgomery algorithm, Binary Inversion Algorithm and Little Fermat thermo. Out of this Binary Inversion Algorithm division operation performed by cheap shift and adds operation. This work replace both adder and shifter unit by rotate unit. With that replacement can achieve 25% reduction in hardware results in improved performance, speed, area and also reduce power consumption This makes BIA even more well suitable for implementation in any hardware platform. Hence the main objective of the paper is to design the modified Binary Inversion Algorithm in FPGA for NIST recommended prime field p521.

Introduction
Cryptography is a method of storing and transmitting information in a particular form so that only those for whom it is intended can read and process it.
The data security has become an crucial and urgent need for modern applications such as health care information, confidential communication, hand held device storage and financial services. The public key cryptosystem is the most effective for the secure data transmission. The challenge to implement the most popular public key cryptosystem, RSA is the rapidly . growing key size. Elliptic Curve cryptography has been considered an alternative to RSA.
The main advantage of cryptography is security. It is estimated that security level of 160 and 224 bits ECC cryptosystem is equivalent to the 1024 and 2048 bits RSA respectively.The application of Elliptic curves in public-key cryptography was proposed by Koblitz[1] and Miller[2] in 1985. Since then, enormous amount of work has been done on elliptic curve cryptography(ECC).The attractiveness of using elliptic curves is that similar level of security can be achieved with considerably shorter keys than in methods based on the difficulties of solving discrete logarithms over integers or integer factorizations. Therefore ECC has become final choice in smart cards, credit cards and mobile phones due to its strength to provide equivalent security compared to RSA. The fundamental operation ECC of is point multiplication. This point multiplication involves the computation of point addition and point doubling. Each of these operations requires arithmetic operation like addition, subtraction, multiplication, squaring and inversion. Inversion operation is well known to be the slowest operation among all other modular arithmetic operations in ECC [1]. If this inversion operation consumes too much time, it will affect the performance of the whole ECC system. To have a fast modular inverse calculation is one of the main reasons to do inversion in hardware instead of software [7-9]. The other main reason to implement the modular inverse operation in hardware is security [1]. For cryptographic applications, it is better to have all the computations handled in hardware, inside an integrated chip for example, instead of mixing some computations performed in software with others processed in hardware. Software implementations are supported by operating systems in which the underlying processor is not optimized for the instruction set of ECC, which can be interrupted and trespassed by intruders and this way system may compromise the application security. This is not so easily attained in hardware implementations [1].In 2000, J.Goodman and A. Chandrakasan, "An energy efficient reconfigurable public-key cryptography processor architecture in Cryptographic Hardware and Embedded Systems (CHES)", it gives an idea to achieve parallelization in point multiplication. In 2008 Kendall Anayi and Hamad proposed " On Parallelization High-Speed Processors for Elliptic Curve Cryptography", designing a flexible ECC processor for performing additions, subtractions, multiplications and inversions over

ISSN: 2349- 5197
Impact Factor (PIF): 2.138

INTERNATIONAL JOURNAL OF RESEARCH SCIENCE & MANAGEMENT

prime finite fields GF(p).In 2012, Anil Kumar M .N " A Technique to Speed up the Modular Multiplicative Inversion over GF(P) Applicable to Elliptic Curve Cryptography", propose a module to speed up modular inversion calculation. The main contributions of this work include the following a modified architecture to speed up the computation of modular multiplicative inversion which uses Binary Inversion Algorithm. This technique can be used specifically to NIST recommended elliptic curve with modulus p521-1. The outline of this paper is as follows. In section 2, the back ground of Elliptic Curve Cryptography (ECC) is discussed. In section 3, how to speed up the modular inverse for Binary Inversion Algorithm is discussed, section 4 deals with the results and finally section 5 concludes the work.

Elliptic curve theory

Elliptic curves are described by cubic equations, similar to those used in ellipsis calculations. The general form for an elliptic curve equation is:

y2+axy+by=x3+cx2+dx+e.

There is also a single element named the *point at infinity* or the *zero point* denoted "ϕ". The point at infinity is computed as the sum of any three points on an EC that lie on a straight line. If a point on the EC is added to another point on the curve or to itself, some special addition rules are applied depending on the finite field being used and also on the type of coordinate system (affine or projective) it's applied to.

As mentioned earlier, a finite field is a set of elements that have a finite order (number of elements). There are many ways of representing the elements of the finite field. Some representations may lead to more efficient implementations of the field arithmetic in hardware or in software. The EC arithmetic is more or less complex depending on the finite field where the EC is applied and in which coordinate system the computation is performed. GF (p) and GF (2n), in affine and projective coordinates are considered in this research because they are the most used in ECC.

The finite field

The finite field Fp is the prime finite field containing p elements. Although there is only one prime finite field Fp for each prime p, there are many different ways to represent the elements of Fp. Here the elements of F p should be represented by the set of integers: {0, 1…p-1} with addition and multiplication defined as follows.

Addition

If a,b ϵ Fp, then $a+b = r$ in Fp, where r [0, p-1] is the remainder when the integer $a+b$ is divided by p. This is known as addition modulo p and written $a+b \equiv r \pmod{p}$.

Multiplication

If a, b ϵ Fp, then $a \times b = s$ in Fp, where s [0, p-1] is the remainder when the integer ab is divided by p. This is known as multiplication modulo p and written $a \times b \equiv s \pmod{p}$.

Addition and multiplication in Fp can be calculated efficiently using standard algorithms for ordinary integer arithmetic. In this representation of Fp, the additive identity or zero elements is the integer 0, and the multiplicative identity is the integer 1. It is convenient to define subtraction and division of field elements just as it is convenient to define subtraction and division of integers. To do so, the additive inverse (or negative) and multiplicative inverse of a field element must be described.

Additive inverse

If $a \epsilon$ Fp, then the additive inverse (- a) of a in Fp is the unique solution to the equation $a + x \equiv 0 \pmod{p}$.

Multiplication inverse

If $a \epsilon$ F p, $a \neq 0$, then the multiplicative inverse a^{-1} of a in Fp is the unique solution to the equation $ax \equiv 1 \pmod{p}$. Additive inverses and multiplicative inverses in Fp can be calculated efficiently. Division and subtraction are defined in terms of additive and multiplicative inverses: $a - b \; mod \; p$ is $a + (-b) \pmod{p}$ and $a/b \bmod p$ is $a.(b-1) \; mod \; p$.

Point multiplication

Scalar multiplication Q= k·P is the fundamental operation in elliptic curve cryptography which is result of adding point P to itself (k-1) times

Q = k·P = P + P + ……. + P.

($k - 1$ Times)

Scalar point multiplication algorithms

ISSN: 2349- 5197
Impact Factor (PIF): 2.138

INTERNATIONAL JOURNAL OF RESEARCH SCIENCE & MANAGEMENT

The most commonly used point multiplication algorithms are Binary scalar multiplication algorithm, NAF scalar multiplication algorithm and JSF scalar multiplication algorithm.

Binary scalar multiplication algorithm

INPUT: A point P and an integer k, binary k with no leading 0's

OUTPUT: Q = k·P

1. Q←P

2. For j = |k| − 2… 1, 0

2.1 Q ← 2 Q

2.2 IF k_j = 1 THEN Q←Q + P

3. RETURN Q

NAF scalar multiplication algorithm

INPUT: A point P and an integer k, NAF k with no leading 0's

OUTPUT: Q = k·P

1. Q←P

2. For j = |k| − 2… 1, 0

2.1 Q ← 2 Q

2.2 IF k_j = 1 THEN Q←Q + P

IF k j = -1 THEN Q← -Q + P

3. RETURN Q

INPUT: A point P, JSF (r, s) with no leading$(0,0)$'s

OUTPUT: Q = rP + sP

1. R←2P
2. IF $(r_{|(r,s)|-1}, s_{|(r,s)|-1}) = (1,1)$ THEN Q ← R
   ELSE Q ← P
3. FOR l= $|(r, s)| -2, \ldots . 1, 0$
   3.1 Q ← 2Q

Point addition & point doubling in affine coordinates

Additions in GF (p) are controlled by the following rules:

$$O = -O$$
$$P(x, y) + O = P(x, y)$$
$$P(x, y) + P(x, -y) = O$$

The addition of two different points on the elliptic curve is computed as shown below.

$$\lambda = (y2 - y1)/(x2 - x1)$$
$$X3 = \lambda 2 - x1 - x2$$
$$y3 = \lambda(x1 - x3) - y1$$

The addition of a point to itself (point doubling) on the elliptic curve is computed as shown below

$$P(x1, y1) + P(x1, y1) = P(x3, y3);$$
$$\lambda = (3(x1)2 + a)/(2y1)$$
$$x3 = \lambda 2 - 2x1$$
$$y3 = \lambda(x1 - x3) - y1$$

Speeding up modular inverse computation

In this section, Binary Inversion Algorithm for modified modular inversion over GF (p) to speed up the modular inverse computation is discussed.

ISSN: 2349- 5197
Impact Factor (PIF): 2.138

INTERNATIONAL JOURNAL OF RESEARCH SCIENCE & MANAGEMENT

Binary Inversion Algorithm

The extended Euclidean algorithm uses the division operations to compute the inversion. The binary inversion algorithm replaces the divisions with cheaper shifts (divisions by 2) and subtractions.

The modular multiplicative inverse a-1 mod p of an integer a exists if and only if a and p are relatively prime, that is gcd (a,p) =1. One of the efficient modular inversion algorithms is Binary Inversion Algorithm shown below.

Algorithm**:** Binary Inversion in $GF(p)$

Input: $p\ and\ a \in [0, p-1]$
Output: $a^{-1}\ mod\ p$

1. $u = a;\ v = p;\ x1 = 1;\ x2 = 0$
2. $\boldsymbol{while\ u = 1\ and\ v = 1\ do}$
2.1. $\boldsymbol{while\ u\ is\ even\ do}$
2.2.1. $u = u/2$
2.2.2. $if\ x1\ is\ even\ then\ x1 = x1/2\ else\ x1 = (x1+p)/2$2.3. $\boldsymbol{end\ while}$
2.4. $\boldsymbol{while\ v\ is\ even\ do}$
2.5.1. $v = v/2$
2.5.2 $if\ x2\ is\ even\ then\ x2 = x2/2\ else\ x2 = (x2+p)/2$2.6. $\boldsymbol{end\ while}$
2.7. $if\ u \geq v\ then\ u = u - v;\ x1 = x1 - x2$
2.8. $else\ v = v - u;\ x2 = x2 - x1$
3. $\boldsymbol{end\ while}$
4.1. $if\ u = 1\ then\ return\ x1\ mod\ p$
4.2. $else\ return\ x2\ mod\ p$

The binary modular inversion algorithm can be easily be modified to perform modular division b/a mod p by initializing x1 variable in step 1 by b instead of 1.In the subsequent sections iteration denote subtraction.

The diagram of the modular inverter is shown in Figure 1 Based on the Binary inversion algorithm, the modular inverter is composed of modular subtractor, subtractor unit, binary shifter, multiplexer, comparator, AND's and NOR's.

The *Start* signal controls the loading operation of u, v, *p*, x1 and x2 registers through multiplexers either by their initial values or by the intermediate results at the beginning of positive edge of every clock cycle.

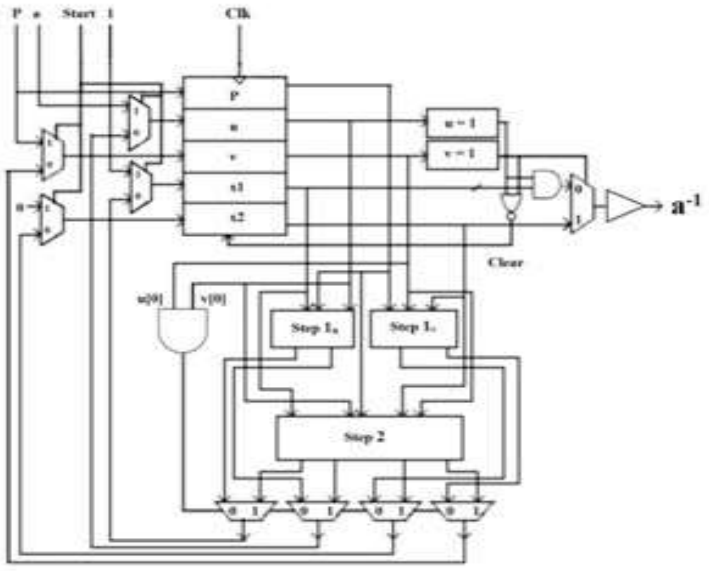

Fig. 1 Architecture of Modular Inverter

There are two comparators to compare the present values of u and v registers with 1. If any one of the comparator outputs becomes true then at the next clock all internal registers are freezed as the final result is available in the x1 or x2.The blocks Step-1_u and Step-1_v of the architecture perform the operations within two inner while loop of the algorithm. The operations designed in step 2.7 and 2.8 of the algorithm are performed by the Step-2 block of the architecture. The diagram of Step-1 and

ISSN: 2349- 5197
Impact Factor (PIF): 2.138

INTERNATIONAL JOURNAL OF RESEARCH SCIENCE & MANAGEMENT

Step-2 are shown in figure 2 and figure 3 respectively. The blocks Step-1_u will perform the same operation as that of Step-1_v for different inputs. Hence the blocks Step-1_u and Step-1_v have same hardware architecture as shown in figure 2.
The Step-1 block consists of a 521 bit rotate, two 521 bit right shifters and three 521 bit 2:1 multiplexers. For Step-1_u block inputs will be u, x1 and outputs are u_out and x1_out. To Step-1_v block v, x2 will be the inputs and v_out and x2_out were outputs. The output u_out depends if the input u is either even or odd value. LSB bit of u is given to select bit of multiplexer. The output u_out will be u if the u is odd value, if u is even the U_OUT= U\2.
Divide by 2 is achieved by performing bitwise right shift operation on u. The output x1_out depends on the inputs u and x1. Here input x1 is rotate and then shifted right once and x1 is divided by 2. If x1 is odd output of multiplexer will be output from rotate unit, if it is even x/2 is the output. The output x1_out is simply x1 or output of multiplexer depending on the value of u is odd or even. Same operation will be performed in Step-1_v with inputs being v, x2 and output were v_out and x2_out.

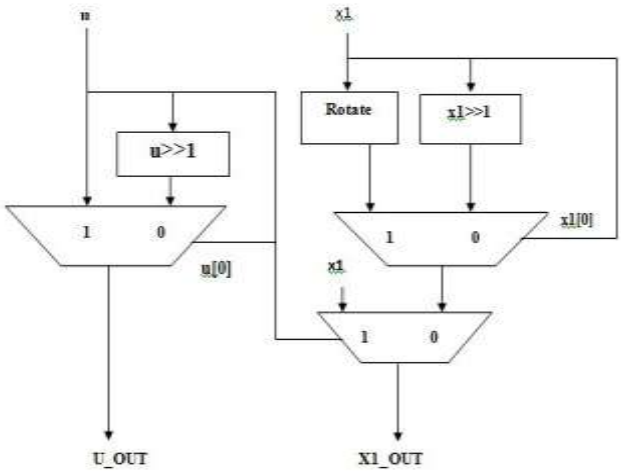

***Fig .2** Block diagram of Step-1 block*

The block diagram of Step-2 block is shown in figure 3. Step-2 module consists of two 521-bit magnitude subtractors, two 521-bit GF (P) subtractors and four 521-bit 2:1multiplexers.

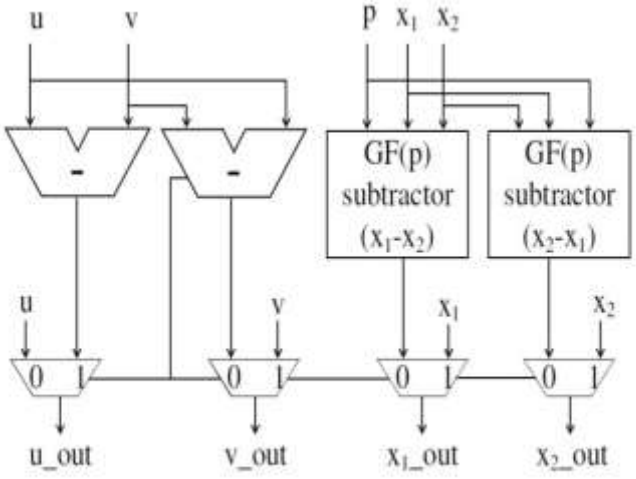

***Fig .3** Block diagram of Step-2 block*

Two magnitude subtractors concurrently perform U-V and V-U and at the same time two GF (P) subtractors perform (X1-X2) mod p and(X2-X1) mod p. The borrow out signal of V-U operation, that says whether the current value of U ≥V, used to select its final output.

ISSN: 2349- 5197
Impact Factor (PIF): 2.138

INTERNATIONAL JOURNAL OF RESEARCH SCIENCE & MANAGEMENT

Results

Simulation results of binary inverse

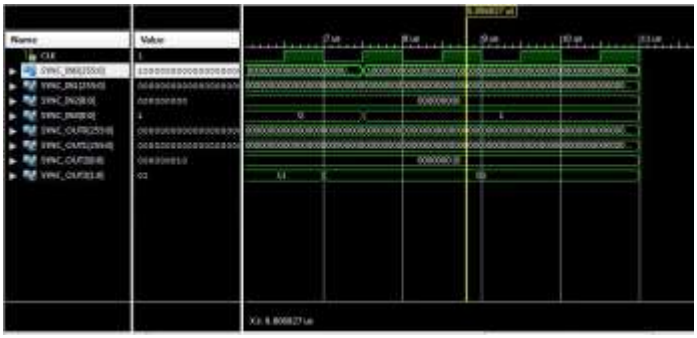

Fig .4 Simulation results of Binary Inverse module.

Simulation results of step_1U module

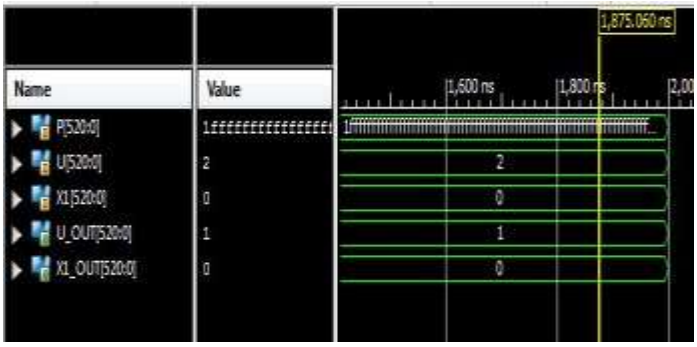

Fig .5 Simulation results of STEP_1U MODULE.

Simulation results of step_1V module

Fig .6 Simulation results of STEP_1V MODULE

Conclusion & future work

As more and more applications of cryptography rely on the modular inverse operation, the need of precious, fast modular inverter has to be designed. Implementation of the inversion algorithm in hardware results in efficiently increasing in performance and speed. In literature survey several modular inverse algorithms in GF (p) were analyzed to study their origins and to trace some of the evolutionary changes made to the basic algorithm. From study we can conclude that Extended Euclidean algorithm requires computationally expensive division operation, Also Montgomery modular inverse algorithm requires some extra arithmetic operations to represent inverse of integer in Montgomery domain to integer. Little Fermat theorem is very much suitable for smaller bit length.

But in the Binary Inversion Algorithm division operation is replaced by cheaper shifts (divisions by 2) and addition. In modified architecture replace the shifts and addition by rotate unit also it does not require any domain conversion. This makes modified

ISSN: 2349- 5197
Impact Factor (PIF): 2.138

INTERNATIONAL JOURNAL OF RESEARCH SCIENCE & MANAGEMENT

Binary Inversion Algorithm more suitable for hardware implementation. The project work was completed successfully by designing and simulation of architecture of modular inverse on the Xilinx version 14.3.Future enhancement will target speeding up computation of individual computational blocks, integration of the proposed architecture with these arithmetic modules to perform scalar multiplication and its FPGA implementation.

References

1. Jin – Hua Hong and Cheng-Wen Wu, " Cellular array modular multiplier for fast RSA public key cryptosystem based on modified Booth's algorithm", IEEE Transactions on Very Large Scale Integration Systems, Vol.11, No.3, June 2003.
2. Ching-Chao Yang, Tian – Sheuan Chang and Chein-Wei Jen, " A new RSA cryptosystem hardware design based on Montgomery's algorithm", IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 45, No.7, July 1998.
3. Andre Vandemeulebroecke, Etienne Vanzieleghem, Tony Denayer and Paul G A Jesperes, " A new carry free division algorithm and its application to a single chip 1024 bit RSA processor", IEEE Journal of Solid State Circuits, Vol. 25, No.3, June 1990.
4. Ming-Der Shieh, Jun-Hong Chen, Hao-Hsuan Wu and Wen-Ching Lin, " A new modular exponentiation architecture for efficient design of RSA cryptosystem", IEEE Transactions on Very Large Scale Integration Systems, Vol.16, No.9, September 2008.
5. Qiang Liu, Fangzhen Ma , Dong Tong and XuCheng, " A regular Parallel RSA Processor", The 47th IEEE International Midwest Symposium on Circuits and Systems.
6. C Mclvor, M.Mcloone and J V McCanny , " Modified Montgomery modular multiplication and RSA exponentiation techniques", IEE Proceedings online no.20040791.
7. Kendall Anayi, Hamad Alrimeih and Daler Rakhmatov, " Flexible hardware processor for elliptic curve cryptography over NIST prime fields", IEEE Transactions on Very Large Scale Integration Systems, Vol.17, No.8, June 2009.
8. Jyu-Yuan Lai and Chih-Tsun Huang, "Elixir: High-throughput cost effective dual field processors and the design framework for elliptic curve cryptography", IEEE Transactions on Very Large Scale Integration Systems, Vol.16, No.11, November 2008.
9. Kimmo Jarvinen and Jorma Skytta, " On parallelization of high speed processors for elliptic curve cryptography", IEEE Transactions on Very Large Scale Integration Systems, Vol.16, No.9, September 2008.
10. William N Chelton and Mohammed Benaissa, "Fast Elliptic Curve Cryptography on FPGA", IEEE Transactions on Very Large Scale Integration Systems, Vol.16, No.2, February 2008.
11. Santhosh Ghosh, Monjur Alam, Indranil Sen Gupta and Dipanwita Roy Chowdhury, " A robust GF(p) parallel arithmetic unit for public key cryptography", 10th Euromicro conference on digital system design architectures, methods and tools (DSD 2007).
12. Hamid Reza Ahmadi and Ali Afzali Kuhsa, "Low power flexible GF(p) elliptic curve cryptography processor",
13. Ciaran J McIvor, Maire McLoone and John V McCanny, " Hardware elliptic curve cryptographic processor over GF(p)", IEEE Transactions on circuits and systems –I: Vol. 53, No.9, September 2006.
14. Yadollah Eslami, Ali Sheikholeslami, P Glenn Gulak Shoichi Masui and Kenji Mukaida, " An area efficient universal cryptography processor for smart cards", IEEE Transactions on Very Large Scale Integration Systems, Vol.14, No.1, January 2006.
15. Sabel Mercurio Hernandez Rodrfguez and Francisco Rodrfguez, "An FPGA arithmetic logic unit for computing scalar multiplication using the half and add method", Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs -2005.
16. Ray C C Cheung, Nicolas Jean-baptiste Telle, Wayne Luk and Peter Y K Cheung, " Customizable Elliptic curve cryptosystems", IEEE Transactions on Very Large Scale Integration Systems, Vol.13, No.9, September 2005.
17. Alireza Hodjat , David D Hwang and Ingrid Verbauwhede, "A scalable and high performance elliptic curve processor with resistance to timing attacks', Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2005).
18. Jonathan Lutz and Anwarul Hasan, "High performance FPGA based elliptic curve cryptographic co-processor", Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC 2004).
19. Akashi Satoh and Kohji Takano, "A scalable dual field elliptic curve cryptographic processor", IEEE Transactions on Computers, Vol.52, No.4, April 2003.
20. Chang Shu, Kris Gaj and Tarek El Ghazawi, "Low latency elliptic curve cryptography accelerators for NIST curves over Binary Fields", ICFPT 2005.

ISSN: 2349- 5197
Impact Factor (PIF): 2.138

21. Philip H W Leong and Ivan K H Leung, " A microcoded elliptic curve processor using FPGA technology", IEEE Transactions on Very Large Scale Integration Systems, Vol.10, No.5, October 2002.
22. Essame Al-Daoud,Ramlan Mahmod, Mohammad Rushdan and Sdem Kilicman, "A new addition formula for elliptic curves over GF(2^n)", IEEE Transactions on Computers, Vol. 51, No.8, August 2002.
23. Adnan Abdul Aziz Gutub and Mohammad K Ibrahim, "High radix parallel architecture for GF(p) elliptic curve processor", ICASSP 2003.