HIGH SPEED PIPELINED ASIP PROCESSOR FOR ECC USING FPGA TECHNOLOGY

MuthuKumar .B[1], Jeevananthan .S[2]
[1]Research scholar , Sathyabama University, INDIA
[2]Asst. Professor, Department of Electrical and Electronics Engineering, Pondicherry Engineering College, INDIA
E-mail : [1]anbmuthusba@yahoo.co.in

Abstract

In recent years, Elliptic curve cryptography has gained widespread exposure and acceptance and has already been included in many security standards. Engineering of ECC is a complex, interdisciplinary research field encompassing such as Mathematics, Computer Science and Electrical Engineering. ASIP-based implementations constitute a key trend in SoC design enabling optimal tradeoffs between performance and flexibility. This paper details the design of a High speed pipelined ASIP processor for ECC using FPGA technology. A seven stage pipeline has been applied to the design, and pipeline stalls are avoided via instruction reordering and data forwarding. Three complex instructions are introduced to reduce the latency by reducing the overall number of instructions.The architecture was programmed in Verilog and synthesized to Xilinx Vertex II Pro devices .Simulation was done with Modelsim XE 6.1e, VLSI simulation software from Mentor Graphics Corporation especially for Xilinx devices.

Keywords: FPGA, Elliptic Curve Cryptography, ASIP

I. INTRODUCTION

An application-specific instruction-set processor (ASIP) is a component used in system-on-a-chip (SOC) design. The instruction set of an ASIP is tailored to gain a specific application. This specialization of the core allow for a tradeoff between the flexibility of a general purpose CPU and the performance of an ASIC. Application Specific Instruction Set Processor (ASIP) technology is a new digital signal processing system design approach developed from ASIC and DSP technology [1]. The key feature of ASIP technology is the development of a specific instruction set and architecture for a particular application or for a set of applications [6, 5]. The digital signal processing systems using ASIP approach are very suitable for them to be implemented in FPGAs, due to FPGA's reconfigurable ability, large scale circuit density and ample in-chip functional resources such as RAMs, arithmetic cells, and clock management cells, etc. Particularly, it's rather easy to implement multiple ASIP processing elements (PE) into a single FPGA chip, thus high parallelism and high processing speed can be realized with parallel process-unit array architecture. In fact, the concept of the application-specific instruction set processor (ASIP) [4] constitutes the appropriate solution for fulfilling the flexibility and performance constraints of emerging and future applications as shown in [5] and [6].

Elliptic Curve Cryptography (ECC)[16] is one of the more advantageous type of public key cryptography with the shorter key size for the same level of security provided by the other types of public key cryptography. For example, ECC requires only 160 bits where RSA needs 1024 bits with the same level of security.

In ECC, the fast or bit-parallel algorithms and architectures over finite fields has been performed due to the increasing use of cryptographic techniques in computer and communication network systems. The elliptic curve cryptosystems also have the advantage of their high cryptographic strength relative to the key size and, thus, they are especially attractive in applications such as the financial industry, smart cards, and wireless areas where power and bandwidth are limited.

The computationally intensive operation needed for ECC[3] was implemented in hardware using FPGA technology has numerous advantages. The optimization goal is to reduce the latency of a point multiplication in terms of the number of required cycles. Some of the design techniques used in modern high performance processors were incorporated into the design of an application-specific instruction set processor (ASIP) for ECC. Pipelining was applied to the design, giving improved clock frequencies. Data forwarding and instruction reordering were incorporated to exploit the inherent parallelism of the Lopez and Dahab point multiplication algorithm, reducing pipeline stalls.

II. ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic curve cryptography [ECC] is a public-key cryptosystem just like RSA. In this type of cryptography, every user has a public and a private key. Public key is used for encryption/signature verification. Private key is used for decryption/signature generation.

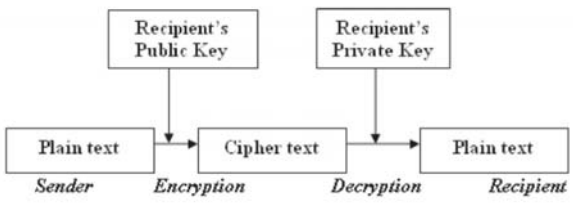

Fig. 1. Public Key Encryption/Decryption Scheme

In Figure1 shows the communication process of both encryption and decryption. The sender sends the plain text with the help of public key the secret data to be converted into cipher text. The reverse process to be performed on the receiver end.The main reason for choosing ECC is its shorter key size and it is faster than RSA.

A. Discrete Logarithm Problem

The Security of ECC depends on the difficulty of Elliptic Curve Discrete Logarithm [6] Problem. Let P and Q be the two points on an elliptic curve such that Q=k.P Where Q is the output point , P is the Input point, k is a Scalar .Given P and Q, it is computationally infeasible to obtain k, if k is sufficiently large. k is the discrete logarithm of Q to the base P. Hence the main operation involved in ECC is point multiplication. i.e. multiplication of a scalar k with any point P on the curve to obtain another point Q on the curve.

B .Point Multiplication

In point multiplication a point P on the elliptic curve is multiplies with a scalar k using elliptic curve equation to obtain another point Q on the same elliptic curve. Q=k.P Point multiplication is achieved by two basic elliptic curve operations. Point addition, adding two points P1 and P2 to obtain another point, P3 =P1+P2. Point doubling, adding a point P to itself to obtain another point R=2P.

In Fig2 shows the operation of the scalar multiplication. The point addition and point multiplication to be performed. Consider the p1 and p2 points; get the p4 point in the curve by performed the operation point adding.

Example:

Let P be a point on an elliptic curve. Let k be a scalar that is multiplied with the point P to obtain another point Q on the curve. i.e. to find Q=k.P .If k = 23 then kP = 23.P = 2 (2 (2 (2P) +P) +P) +P.

Thus point multiplication uses point addition and point doubling repeatedly to find the result. The above method is called 'double and add' method for point multiplication.

III. ELLIPTIC CURVES

An *elliptic curve* over a field ***K*** is a nonsingular cubic curve in two variables, *f(x,y) =0* with a rational point (which may be a point at infinity). The field ***K*** is usually taken to be the complex numbers, reals, rationals, algebraic extensions of rationals, p-adic numbers, or a **finite field**.

An *elliptic curve* is a plane curve defined by an equation of the form $y = X3 + ax + b$

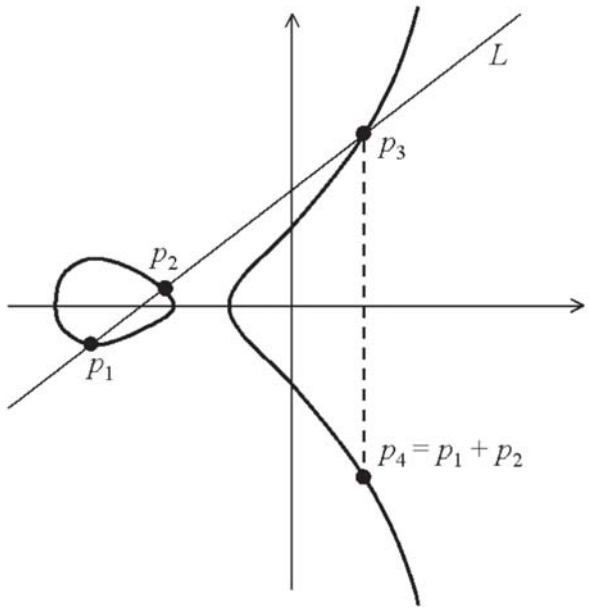

Fig. 2. Elliptic Curve

Operations over the real numbers are slow and inaccurate due to round-off error. Cryptographic operations need to be faster and accurate. To make operations on elliptic curve accurate and more efficient, the curve cryptography is defined over two finite fields. Prime field $\boldsymbol{F_p}$ and Binary field $\boldsymbol{F_2^m}$.

A. EC on Prime field F_p

The equation of the elliptic curve on a prime field $\boldsymbol{F_p}$ is $\mathbf{y^2 \bmod p = x^3+ax+b \bmod p}$, where $\mathbf{4a^3+27b^2 \bmod p \neq 0}$. Here the elements of the finite field are integers between 0 and p-1. All the operations such as addition, subtraction, division, multiplication involves integers between 0 and p-1. The prime number p is chosen such that there is finitely large of points on the elliptic curve to make the cryptosystem secure.

B. EC on Binary field F_2^m

The equation of the elliptic curve on a binary field $\boldsymbol{F_2^m}$ is $\mathbf{y^2+xy = x^3+ax^2+b}$, Where $\mathbf{b \neq 0}$. Here the elements of the finite field are integers of length at most m bits. These numbers can be considered as a binary polynomial of degree m-1. In binary polynomial the coefficients can only be 0 or 1. All the operation such as addition, subtraction, division, multiplication involves polynomial of degree m-1 or lesser. The m is chosen such that there is finitely large number of points on the elliptic curve to make the cryptosystem secure. SEC specifies curves with m ranging between 113-571 bits.

Here, we are choosing the binary field for cryptographic operations with the 163 bits i.e. m=163 that is between 113-571 bits according to the SEC.

IV. EC OPERATIONS ON AFFINE COORDINATE

Let $P_1 = (x_1, y_1) \in E$,and

$P_2 = (x_2, y_2) \in E$, then summing the two points is $P_1 + P_2 = P_3 = (x_3, y_3) \in E$, where

Point addition:

$$x_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)^2 + \frac{y_1 + y_2}{x_1 + x_2} + x_1 + x_2 + a, \qquad y_3 = \left(\frac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1 \text{ , } P_1 ? P_2$$

Point doubling:

$$x_3 = x_1^2 + \frac{b}{x_1^2} \text{ , } P_1 = P_2,$$

$$y_3 = x_1^2 + \left(x_1 + \frac{y_1}{x_1}\right)(x_3) + x_3 \text{ , } P_1 = P_2$$

Hence, when $P_1 = P_2$ we have the point-doubling operation (DBL), and when $P_1 \neq P_2$ we have the point-adding operation (ADD). These operations in turn constitute the crux of any ECC-based algorithm, known as point multiplication or scalar multiplication.

V. EC OPERATIONS ON PROJECTIVE COORDINATE

Due to the computational expense of inversion compared to multiplication, several projective coordinate methods have been proposed, which use fractional field arithmetic to defer the inversion operation until the end of the point multiplication.

Here the point (x, y, z) in projective coordinate corresponds to the point $(x/z, y/z^2)$ in affine coordinate. For point multiplication, convert the point (x, y) in affine coordinate to (x, y, 1) in projective coordinate. After multiplication the result (x, y, z) is converted back to the affine coordinate as $(x/z, y/z^2)$ where $z \neq 0$. If $z = 0$, then the point is considered as the point at infinity..

VI. POINT MULTIPLICATION IN PROJECTIVE COORDINATES

In this paper, the high-performance, generic projective-coordinate algorithm proposed by Lopez and Dahab is used, which is an efficient implementation of Montgomery's method for computing kP [9]. No precomputations or special field/curve properties are required.

In the projective coordinate version of the formulas, the x-coordinate of P_i is represented by X_i/Z_i , for i $\in$ {1,2,3}; Point doubling and Point addition are determined by the following equations. In shown in the Fig.3 and Fig. 4 respectively doubling architecture and addition architecture . The flow chart for Lopez – Dahab Point multiplication algorithm is shown in the Fig5.

Point doubling:

$$x(2P_i) = X_i^4 + b.Z_i^4$$

$$z(2P_i) = Z_i^2.X_i^2$$

Point addition:

$$Z_3 = (X_1.Z_2 + X_2.Z_1)^2$$

$$X_3 = x.Z_3 + (X_1.Z_2)(X_2.Z_1)$$

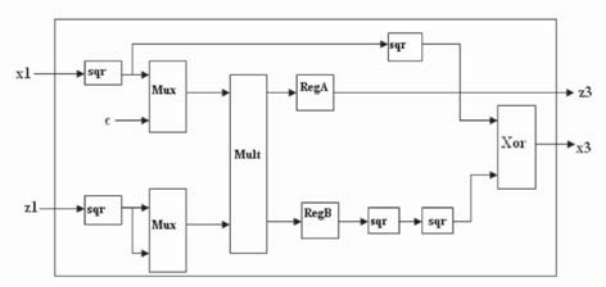

Fig. 3. Doubling Architecture

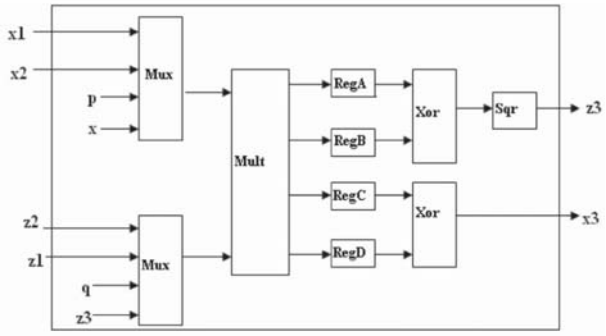

Fig. 4. Addition Architecture

Lopez-Dahab Point Multiplication Algorithm

Input: An integer $k \geq 0$ and a base point $P = (x, y) \in E$.**Output**: $Q = kP$

If $k = 0$ or $x = 0$ then output (0, 0) and stop.

Set $k \leftarrow (k_{l-1} k_{l-2} \ldots k_0)$,Set $X_1 \leftarrow x$, $Z_1 \leftarrow 1$, $X_2 \leftarrow x^4 + b$, $Z_2 \leftarrow x^2$.

For j from l-2 downto 0 do

If $k_j = 1$ then

$\text{ADD}(X_1, Z_1, X_2, Z_2), \text{DBL}(X_2, Z_2)$

Else $\text{ADD}(X_2, Z_2, X_1, Z_1), \text{DBL}(X_1, Z_1)$

Return (Q=Proj2Aff (X_1, Z_1, X_2, Z_2))

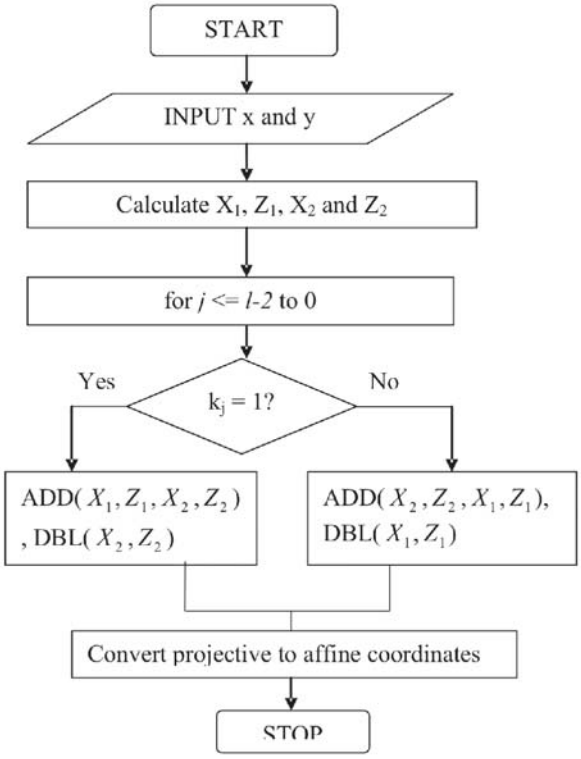

Fig. 5. Flow chart for Performing Point multiplication in Projective Coordinates

VII. PIPELINED ASIP FOR ECC

Complex instruction set computers (CISCs) reduces the number of instructions, and consequently the overall latency, by performing multiple tasks in a single instruction. Complex instructions could be used to reduce latency in the design of an ASIP for ECC[1],[10]. Three new instructions were introduced. Considering the projective-coordinate formula for point addition, an obvious instruction to combine operations is a multiply-and-accumulate instruction (MULAD), which will save two instruction executions.

Also, rewriting the projective-coordinate formula for point doubling, we have

$$x(2P_i) = (X_i.X_i)^2 + b.(Z_i.X_i)^2$$

$$z(2P_i) = (Z_i.X_i)^2$$

So, a multiply-and-square operation (MULSQ) would be beneficial, saving three instruction executions. The architecture for point multiplier is given in Fig. 6.

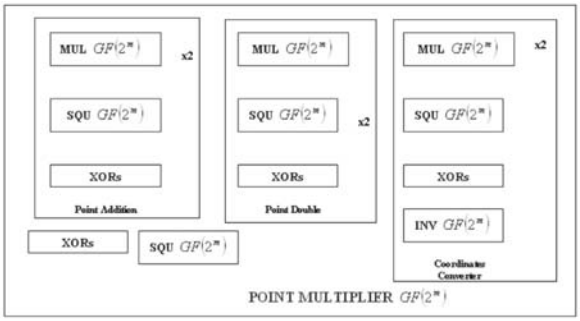

Fig. 6. Architecture for Point Multiplier

Based on the Montgomery point multiplication algorithm [11], the point multiplier is composed of point adder, point doublers, coordinates converter, squarer and XORs. We use two Galois field multipliers[12], one Galois field squarer and XORs to implement point adder. Point doublers is composed of two Galois field squarer's, one Galois field multiplier and XORs. The coordinate's converter is more complicated than point adder and point doublers. It consists of two Galois field multipliers, one Galois field squarer, one Galois field inverter and XORs. The addition unit in Galois field is straightforward to implement over binary field. It can be designed using an array of XOR gates.

The Itoh–Tsujii inversion algorithm is based on Fermat's little theorem and is used to compute the inversion at the end of the point multiplication. The algorithm uses addition chains to reduce the required number of multiplications, but it contains exponentiations that must be performed through repeated squaring operations—as many as 64 repeated squaring operations for the field. The relatively low latencies can be achieved using this algorithm if repeated squaring can be performed efficiently. Hence, a third application-specific instruction will be used to perform repeated squaring (RESQR) in order to accelerate the Itoh–Tsujii inversion algorithm.

Using these instructions, a combined algorithm to perform point doubling and point addition can be developed, which has only nine arithmetic instructions, shown in new combined algorithm to perform DBL and ADD .

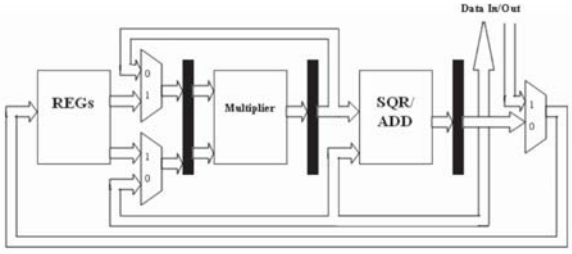

Fig. 7. Architecture for Pipelined ASIP Data path

New Combined Algorithm to Perform DBL and ADD

The combined algorithm to point addition and point doubling is given by the following instructions.

Input: x - coordinates X_1 / Z_1 for P_1 and X_2 / Z_2 for P_2; the curve parameter b;

x, the x - coordinate of the base point P.

Output: x - coordinates X_1 / Z_1 for $P_1 + P_2$ and X_2 / Z_2 for $2P_2$.

$T_1 \leftarrow MUL(X_2, Z_1)$

$T(2) \leftarrow MULAD(X_1, Z_2)$

$Z_1 \leftarrow RESQ(T_2)$

$T_3 \leftarrow MULSQ(Z_2, Z_2)$

$Z(2) \leftarrow MULSQ(X_2, Z_2)$

$T_1 \leftarrow MUL(T_1, T_2)$

$X_1 \leftarrow MULAD(Z_3, x)$

$X_2 \leftarrow MULSQ(X_2, X_2)$

$X_2 \leftarrow MULAD(T_3, b)$

To maintain high-throughput performance, high clock frequencies are required and one instruction/cycle or more is desirable. Therefore, the data path must be pipelined. The pipelined data path is shown in Fig.7.

Fig 7 shows the data path of the pipelined application specific instruction set processor (ASIP), which includes the acceleration technique data forwarding, a sub-pipelined multiplier and a register file that performs reads and writes independently and concurrently. As the multiplier is sub-pipelined in to four stages, the data path contains a total of seven pipeline stages: a read stage, four multiplication stages (MUL, MULAD, MULSQ and RESQ), the SQR/ADD stage and a write stage. Therefore, the given data path uses seven stage pipelining to improve the clock frequency.

SQR/ADD Block

The extra functionality required for the new instructions MULAD, MULSQ, and RESQ is provided by the SQR/ADD block [14],[15], which is appended to the multiplier output as shown in Fig.7.

A block diagram showing the functionality of the SQR/ADD block is shown in Fig. 8; note that the flip-flops are the pipeline flip-flops placed after the SQR/ADD block in the data path diagram shown in Fig.7. Selection 0 on the MUX shown in Fig.8 sets the data path to perform a standard multiplication over.

Selection 1 sets the data path to perform MULSQ. Selection 2 sets the data path to perform MULAD. Selection 3 sets the data path to perform RESQ; note that RESQ can be performed after any other operation, e.g., MULAD-RESQ is a valid instruction sequence.

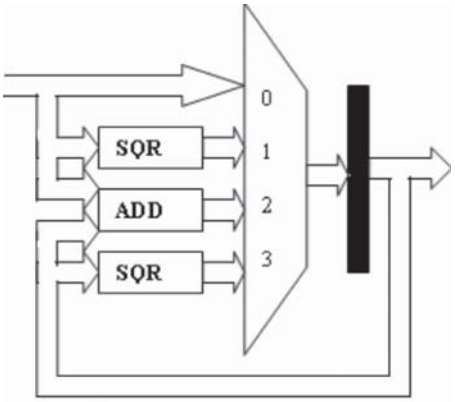

Fig. 8. SQR/ADD Block

Inversion by square and multiply

Input: Field element a

Output: b ? a^{-1}

$b \leftarrow a$;

for i = 1 to m-2 **do**

$b \leftarrow b^2 * a$;

$b \leftarrow b^2$;

The primary advantage to this inversion method is the fact that it does not require hardware dedicated specifically to inversion. The field multiplier can be used to perform all required field operations.

Table 1. Synthesis report for different bit sizes Affine Coordinates

Requirements	m=4	m=8	m=12	m=18
No. of Slices	11	318	808	1844
No. of 4 I/P LUTs	20	558	1418	3232
No. of IOs	17	33	49	73
No. of Bonded IOBs	17	33	49	73
No. of IOB Flipflops	8	16	24	36
No. of GCLKs	1	1	1	1
Time (ns)	3.506	27.222	45.991	77.480

Table 2. Synthesis report for different bit sizes in Projective Coordinates

Requirements	m=4	m=8	m=12	m=18
No. of Slices	27	153	481	1064
No. of 4 I/P LUTs	49	273	851	1873
No. of IOs	21	41	61	91
No. of Bonded IOBs	17	33	49	73
No. of IOB Flipflops	7	15	24	36
No. of GCLKs	1	1	1	1
Time (ns)	8.227	27.278	64.082	138.064

By using the above inversion process, the output point which is obtained in projective coordinates is converted back to affine coordinates.

VIII. FPGA IMPLEMENTATION

The architecture has been programmed in Verilog and synthesized to Xilinx Virtex II Pro FPGA devices. Xilinx ISE 8.2i is an Integrated Software Environment tool from Xilinx Inc. for complete project implementation with Xilinx specific devices is used in this project for synthesis purpose. Simulation is done with ModelSim XE III 6.1e, VLSI simulation software from Mentor Graphics Corporation specifically for Xilinx devices.

In this paper, the Lopez-Dahab Point multiplication algorithm and new combined algorithm to perform adding and doubling have been programmed in Verilog within the same module and simulated by using Modelsim XE III 6.1e.The simulated codings are then synthesized in Virtex II Pro device using Xilinx ISE 8.2i.

IX. COMPARISON BETWEEN AFFINE AND PROJECTIVE COORDINATES

When comparing the area required and time consumption for performing point multiplication in projective coordinates and affine coordinates which have shown in Table 1 and Table 2 respectively, the time consumption in the projective coordinates is considerably reduced with the increase in the required area.

The comparison between the affine and projective coordinates according to the time that it has been consumed and the area required to perform the point multiplication are tabulated in table 3.In figure 9 and Figure 10 shows the graph projective coordinates according to the time that it has been consumed and the area required to perform the point multiplication.

Table 3. Device Comparison for Proposed ASIP:

Field	Reduction Polynomial
$GF(2^{163})$	$F(x) = x^{163} + x^7 + x^6 + x^3 + 1$
$GF(2^{233})$	$F(x) = x^{233} + x^{74} + 1$
$GF(2^{283})$	$F(x) = x^{283} + x^{12} + x^7 + x^5 + 1$
$GF(2^{409})$	$F(x) = x^{409} + x^{87} + 1$
$GF(2^{571})$	$F(x) = x^{571} + x^{10} + x^5 + x^2 + 1$

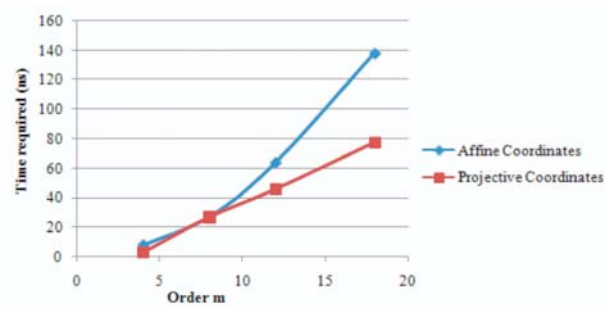

Fig. 9. Affine Vs Projective Coordinates (Time Consumption)

Fig. 10. Affine Vs Projective Coordinates (Area Required)

Consequently, the increase in required area also reduces when increasing the bit size. When considering the bit sizes, NIST recommends ten finite fields, five of which are binary fields, for use in the ECSDA. The binary fields include $GF(2^{163})$, $GF(2^{233})$, $GF(2^{283})$, $GF(2^{409})$ and $GF(2^{571})$ defined by the reduction polynomials

Table 4. NIST Recommended Finite Fields

Device	Time Taken (ns)			
	m=4	m=8	m=12	m=18
XC3S1000-FG456-4	6.483	58.092	100.067	170.415
XCV600E-FG900-8	5.783	55.794	94.740	177.465
XC2VP4-FG256-6	3.506	27.222	45.991	77.480
XC4VFX12-SF363-12	3.665	24.936	41.438	70.630

For each field a specific curve, along with a method for generating a pseudo-random curve, are supplied. These curves have been intentionally selected for both cryptographic strength and efficient implementation.

Such a recommendation has significant implications on design choices made while implementing elliptic curve cryptographic functions. In standardizing specific fields for use in elliptic curve cryptography (ECC), NIST allows ECC implementations to be heavily optimized for curves over a single finite field. As a result, performance of the algorithm can be maximized and resource utilization, whether it be in code size for software or logic gates for hardware, can be minimized.

Based on the NIST recommended binary fields and the reason for that recommendations, the binary field GF(2^{163}) has been selected to perform the elliptic curve operations. Then the Verilog coding for Lopez-Dahab algorithm for point multiplication has been developed with the binary field GF(2^{163}) and then it has been simulated by using Modelsim XE III 6.1e.

X. CONCLUSION AND FUTURE WORK

For ECC, point multiplication is the essential thing to convert the given plain text in to the cipher text and then the multiplier is considered to be the more resource and time sensitive operation in the field. The multiplication process can be carried with the projective coordinates by using the Lopez-Dahab point multiplication algorithm to reduce the number of inversions required. This is because, the direct computation of the point addition and point doubling according to the equations require five inversions for each input point. But in projective coordinates, it requires only two inversions – one for x coordinate and another for y coordinate. Therefore, the time taken to perform the inversions is reduced in projective coordinates when compared to affine coordinates.

Also the pipelined ASIP results in better time consumption with the considerable increase in area to complete the entire process. The proposed architecture has been implemented for the binary fields 4,8,12 and 18 and its results were shown clearly in the Table 4. In future, it will be implemented for the binary field 233. Further work will continue to develop the better architecture by increasing the pipeline stages for reducing the time consumption as well as the area required.

REFERENCES

[1] K. Keutzer, S. Malik, and R. Newton, “From ASIC to ASIP:The Next Design Discontinuity,” In *IEEE International Conference on Computer Design*, pp. 301–304, January 2002.

[2] W. Chelton and M. Benaissa, “Fast Elliptic Curve Cryptography on FPGA”, IEEE Trans.VLSI systems, vol.16, No.2, pp. 198-205, Feb. 2008

[3] W. Chelton and M. Benaissa, “High-speed pipelined ECC processor over GF(2m),” presented at the IEEE Workshop Signal Process. Syst. (SiPS), Banff, Canada, 2006.

[4] A. Oraioglu, A. Veidenbaum, “Application Specific Microprocessors” (Guest Editors Introduction), IEEE Design& Test Magazine, Jan/Feb, 2003.

[5] O. Muller, A. Baghdadi, and M. Jézéquel, "ASIP-Based Multiprocessor SoC Design for Simple and Double Binary Turbo Decoding", in Proc. of DATE 2006, Munich, Germany, March 2006.

[6] T. Vogt, and N. Wehn, “A Reconfigurable Application Specific Instruction Set Processor for Viterbi and Log-MAP Decoding”. IEEE Workshop on Signal Processing (SIPS'06), pages 142-147, October 2006, Banff, Canada.

[7] N. Koblitz, A. Menezes, and S. Vanstone, “The state of elliptic curve cryptography,” Designs, Codes Cryptography, vol. 19, pp. 173–193, 2000.

[8] P. L. Montgomery, “Speeding the Pollard and elliptic curve methods of factorization,” 1987.

[9] T. Itoh and S. Tsujii, “A fast algorithm for computing multiplicative inverses in GF(2/sup m/) using normal bases,” Inf. Computation, vol. 78, pp. 171–177, 1988

[10] K. Jarvinen, M. Tommiska, and J. Skytta, “A scalable architecture for elliptic curve point multiplication,” presented at the ICFPT, Brisbane, Australia, 2004.

[11] F. Rodriguez-Henriquez, N. A. Saqib, and A. Diaz-Perez, “A fast parallel implementation of elliptic curve point multiplication over GF(2m),” Microprocessors Microsyst., vol. 28, pp. 329–339, 2004.

[12] H. Wu, "Bit-parallel finite field multiplier and squarer using polynomial basis," IEEE Trans. Comput., vol. 51, no. 7, pp. 750–758, Jul. 2002.

[13] J. Lutz and A. Hasan, "High performance FPGA based elliptic curve cryptographic co-processor," in *Proc.* Int. Conf. Inf. Technol.: Coding Comput. (ITCC), 2004, p. 486.

[14] R. C. C. Cheung, N. J. Telle, W. Luk, and P. Y. K. Cheung, "Customizable elliptic curve cryptosystems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 13, no. 9, pp. 1048–1059, Sep. 2005.

[15] NIST—National Institute of Standards and Technology, "Recommended elliptic curves for federal government use," 2000 [Online]. Available: http://csrc.nist.gov/encryption

[16] N. Koblitz. Elliptic Curve Cryptosystems. Mathematics of Computation, 48(177):203–209, November 1987.

Muthukumar .B, received M.C.A degree from the Manonmaniam Sundaranar University ,Thirunelveli,Tamil Nadu, India in 1999 and M.E degree from Sathyabama University, Chennai, Tamil Nadu, India in 2004. He has ten Plus years of experience in teaching. He is currently working as Senior Lecturer in Master of Computer Applications Department at sathyabama University and also pursuing his doctoral program in the Faculty of Computer Science and Engineering at Sathyabama University, Chennai, Tamil Nadu, India. His area of research is Networks Cryptography.