Available online at www.elixirpublishers.com (Elixir International Journal)

Cryptology

Elixir Cryptology 46 C (2012) 8511-8517

ISSN: 2229-712X

High Speed FPGA Based Elliptic Curve Cryptography Using Mixed Co-ordinates

Shylashree.N[1], A. Deepika[2] and V. Sridhar[1]
[1]Department of ECE, PESCE, Mandya, Karnataka, India.
[2]Department of ECE, RNSIT, Bengaluru, Karnataka, India

ARTICLE INFO

Article history:
Received: 30 April 2012;
Received in revised form:
15 May 2012;
Accepted: 25 May 2012;

Keywords
Elliptic Curve Cryptography (ECC), Finite field, affine coordinates, Projective coordinates, Mixed Co-ordinates, Scalar multiplication, FPGA (Field Programmable Gate Array), Xilinx.

ABSTRACT
Elliptic curve cryptography (ECC) plays an important role in communication and information security. ECC is an approach to public-key cryptography based on the finite fields. We propose a hardware architecture for scalar multiplication, which is the main operation of ECC based on mixed coordinates (Lopez-Dahab) over $GF(2^{163})$ in the Karatsuba multiplication polynomial representation. The proposed design is implemented using Verilog as a language and Xilinx virtex4Vlx15sf363-12 as a target device, and it is verified in Model Sim 6.3g. The results show that the proposed design occupies 30k slices with a delay of 35.086 ns.

Introduction

Elliptic curve cryptography (ECC) was proposed by Koblitz [1] and Miller in 1985 [2]. ECC is one of the public-key cryptography algorithms. Its attractive feature is lesser key size with the same level of security compared to other cryptography algorithms like RSA. Its drawback is slow when implemented in software. On the other hand, the hardware implementation of this algorithm is very fast and efficient. Therefore, ECC is a primary choice in hardware implemented cryptography [3]. Because of smaller key size, ECC is suitable for power constrained devices like mobile phones, smart cards, etc. A comparison based on key size of ECC and RSA is presented in Table 1.

Related work

In [3] lookup tables, high-speed operation has been achieved based on the proposed sharing scheme that reduces the field multiplications. A crypto-processor based on the Lopez–Dahab point multiplication is presented in Ref. [4] in which the curve arithmetic is based on Gaussian Normal Basis Arithmetic. High throughput has been achieved using two new word level arithmetic units and parallelized elliptic curve point doubling and addition algorithms. Parallelism is implemented during point doubling to reduce the latency.

Galois field multiplication can be performed using different methods based on finite elements representation. In Refs. [5–9], multiplication based on Karatsuba is used in which the elements are represented in polynomial form, and the finite reduction is based on irreducible polynomial which is recommended by NIST standards. Projective coordinates are considered as elements. By these methods, larger numbers are divided into smaller, and it is flexible for any key length. In Ref. [10], the authors proposed the processor which uses polynomial representation and applicable for arbitrary curves.

Point addition and doubling are key operations of ECC which decide the performance of ECC. In Refs. [11–13], architectures are proposed using parallelism and pipelining in both addition and doubling by using the projective coordinates. Exponentiation is achieved using the mixed coordinates. In Ref. [14], scalar multiplication based on window method is proposed which reduces delay by merging addition and doubling. Multiplication of finite fields takes more time than addition and squaring. In Refs. [15–17], authors proposed the serial multipliers, which are applicable for binary fields.

Reductions are defined within a multiplier unit to achieve high throughput. A high performance ECC processor based on the Lopez–Dahab EC point multiplication was proposed [18]. A dual field EC processor with projective coordinates adaptive to both the binary and prime fields, implementing the scalar multiplication architecture, was proposed [17]. The EC cryptographic processor proposed in this work has finite-field (FF) RISC cores and a main controller to achieve the instruction level parallelism (ILP) for EC point multiplication [19].

Mathematical background [20–23]

Elliptic curves are not ellipses. They are named so because they are described by cubic equations that are similar to the equations used to calculate the length of a curve in the circumference of an ellipse. The equation of elliptic curves over binary field is given by

$y^2+xy=x^3+ax^2+b$,

Where $a_i \in K$, $i=1,2,\ldots,6$, and K is a field.

Main operations involved in ECC are:

(a) *Point addition*: Let P and Q be two points on the curve with coordinates (x_1,y_1) and (x_2,y_2). Then adding the two points results in a third point $R=(P+Q)$. The addition is performed by drawing a line through P and Q as shown in Fig. 1. The point at which the line intersects the curve is $-(P+Q)$. The inverse of this

Tele:
E-mail addresses: shylashashi@gmail.com

is $R= (P+Q)$. Let the coordinates of R be (x_3,y_3), which are given by

$x_3=\Delta^2+\Delta+x_1+x_2+a$,

$y_3=\Delta(x_1+x_3)+x_3+y_1$,

where $\Delta=(y_1+y_2)/(x_1+x_2)$, if $P=-Q$.

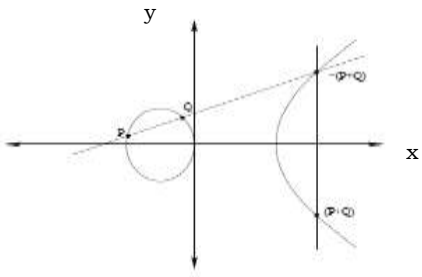


Fig. 1. Graphical representation of point addition for the curve $y^2+xy=x^3+ax+b$.

(b) *Point doubling (2P)*: Let P be a point on the curve with coordinates (x_1,y_1) and $P\neq -P$. The double of P is the point $2P=(x_3,y_3)$ obtained by drawing a tangent to the curve through P as shown in Fig. 2.

$x_3=\Delta^2+\Delta+a$,

$y_3=x_1^2+\Delta x_3+x_3$,

where $\Delta=x_1+(y_1/x_1)$.

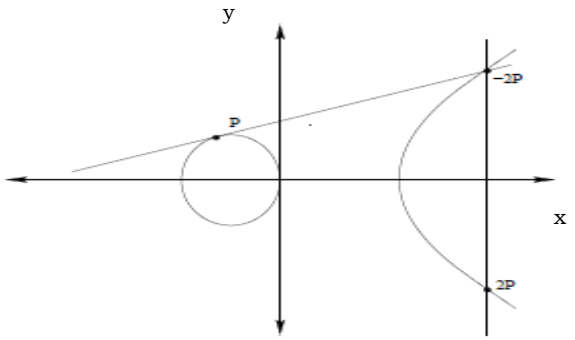
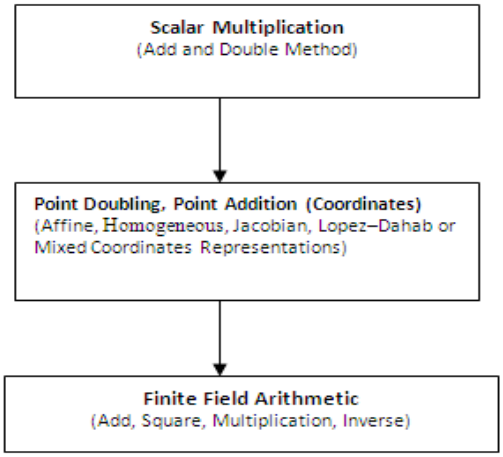


Fig. 2. Graphical representation of point doubling

(c) *Scalar multiplication (KP)*: Multiplying a point by a scalar is known as point multiplication, which is performed using point addition and doubling to increase the speed of encryption and decryption.

Example: If k=15, then $15P=2(2(2P+P)+P)+P$.

Hierarchy of ECC operations:



Finite fields

Elliptic curves are mainly defined over two finite fields:

- Prime field GF(P)
- Binary field $GF(2^n)$

Elliptic curve equation over prime field is given by y^2 mod p=(x^3+ax+b) mod P, where a and b are the parameters, and x and y are the points on curves. *Binary field equation is* $y^2+xy=x^3+ax^2+b$. ECC over binary field achieves the high performance without considering the carry and modular reduction. These fields are optimal for the use in hardware in terms of area and speed. On the other hand, the prime field has disadvantage in considering the numbers in the range from 0 to p–1, but has the advantage that the arithmetic operations can be reused.

In binary field, addition is XOR operation and multiplication is polynomial based, and the result is reduced by using the irreducible polynomial. Squaring is achieved by shift operation. So multiplication is performed based on the hybrid Karatsuba multiplier [7,8].

We primarily focus on ECC over binary field based on the short weierstrass equation.

Hybrid Karatsuba multiplier

The hybrid Karatsuba multiplier (combination of simple and general Karatsuba multiplier) [24] divides a larger number into smaller numbers, and the result is bring to the range by modulus. Hybrid Karatsuba Multiplier for 163-bits as shown in fig (3).

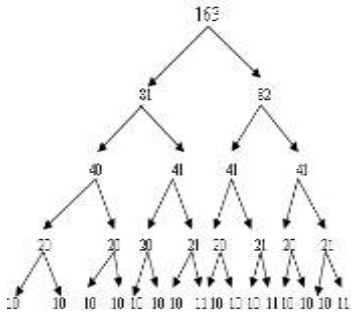


Fig (3) 163-bit Hybrid Karatsuba Multiplier

The finite field multiplication of two elements in the field $GF(2^n)$ is defined as $C(x)=A(x)B(x)\text{mod}P(x)$, where $A(x)$, $B(x)$ and $C(x)$€$GF(2^n)$ and $P(x)$ is the irreducible polynomial of degree n which generates the field $GF(2^n)$. Implementing the multiplication requires two steps. First, the polynomial product $C'(x)=A(x)B(x)$ is determined, and then the modulo-operation is done on $C'(x)$. The Karatsuba algorithm is used for the polynomial multiplication. The Karatsuba algorithm splits the n-bit multiplicands into two 2-term polynomials:

$A(x)=A_hx^{n/2}+A_l$ and $B(x)=B_hx^{n/2}+B_l$.

Here 'h' and 'l' represents higher and lower parts of multiplicand and multiplier.

The multiplication is then done using three $n/2$ bit multiplications. The three $n/2$ bit multiplications are then implemented recursively:

$C'(x)=(A_hx^{n/2}+A_l)(B_hx^{n/2}+B_l)$,

$A_hB_hx^n+(A_hB_l+A_lB_h)x^{n/2}+A_lB_l$.

ECC Co-ordinate system [25]

Elliptic curves are defined over two types of coordinates, i.e., affine coordinate and projective coordinate system. Affine coordinates are normal Cartesian coordinates (x,y). The affine coordinate system requires the inversion during point addition and point doubling operations, which are costly in terms of time and area. To overcome this drawback of the affine coordinate system, the projective coordinate system is used, because

projective Co-ordinate does not require the inversion, but with the increased cost of multiplication. There are three types of projective coordinate systems:

- Homogeneous ($X/Z,Y/Z$),
- Jacobian ($X/Z^2,Y/Z^3$),
- Lopez–Dahab ($X/Z,Y/Z^2$) [26].

To overcome this drawback of the projective coordinate system, a mixed Co-ordinate system is used. While calculating the point addition, we consider one point as affine coordinate and another point as projective coordinate, and that of the coordinate system is called as mixed Co-ordinates. The mixed coordinates have the advantage of lesser number of multiplications.

A comparison of mixed Co-ordinates and pure projective Co-ordinates for point addition is presented in Table 2. From Table 2, we can conclude that the mixed Co-ordinates requires lesser number of multiplications, additions and squaring compared to pure projective so it is faster and occupies lesser area.

Point addition using pure projective coordinates

Let us consider first point $P(X_1, Y_1, Z_1)$ and $Q(X_4,Y_4 ,Z_4)$. Addition of these two points results in a new point $R(X_3,Y_3, Z_3)$. Point addition using the pure projective Co-ordinates is presented below:

$A=Y_1*Z_4^2$, $H=E*F$,
$B=X_4*Z_1$, $Z_3=G*Z_1*Z_4$,
$C=Y_4*Z_1^2$, $X_3=D*(C+B^2)+B*(D^2+A)$,
$D=X_1*Z_4$, $Y_3=(D*H+F*A)*F+(H+Z_3)*X_3$,
$E=Y_1*Z_4^2+Y_4*Z_1^2$,
$F=B+D$,
$G=C^2+A^2$.

Proposed method for point addition using Lopez–Dahab mixed coordinates

In this method, one point is in projective Co-ordinate and another point is an affine Co-ordinate. The resulting point will be in projective Co-ordinate which avoids the inversion operation. In these equations 'a' & 'b' are considered as parameters of elliptic curve. Let us consider first point $\mathbf{A(x_2,y_2)}$ and $Q(X_4,Y_4,Z_4)$. Addition of these two points results in a new point $R(X_3,Y_3,Z_3)$. Point addition using the Mixed Co-ordinates is presented below:

1. $A=Y_4+y_2*Z_4^2$;
2. $B=X_4+x_2*Z_4$;
3. $C=B*Z_4$;
4. $Z_3=C*C$;
5. $D=x_2*Z_3$;
6. $E=A+B*B+aC$;
7. $X_3=A*A+C*E$;
8. $I=D+X_3$;
9. $J=A*C+Z_3$;
10. $F=I*J$;
11. $K=Z_3*Z_3$;
12. $Y_3=F+x_2*K+y_2*K$.

Point doubling using pure projective coordinates

The point doubling operation is to add a point on the elliptic curve with itself. In these equations 'a' & 'b' are considered as parameters of elliptic curve.

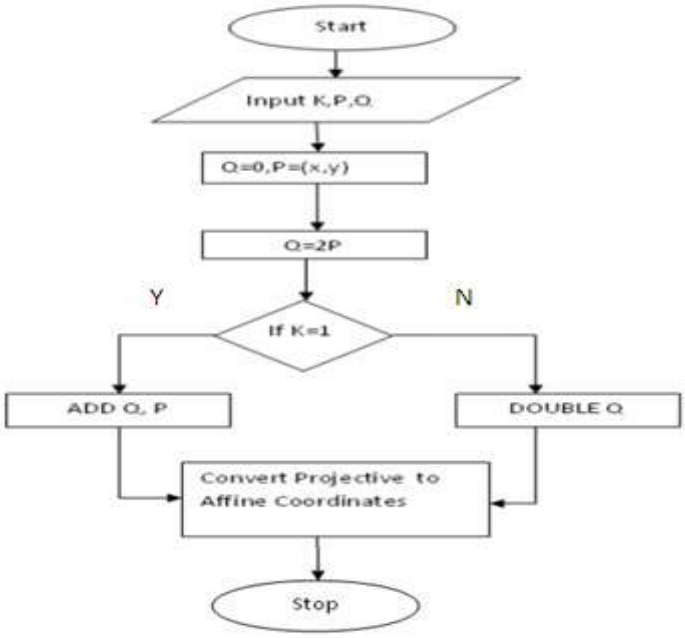
Input (X_1,Y_1,Z_1) and output (X_4,Y_4,Z_4)

$Z_4=Z_1^2*X_1^2$,
$X_4=X_1^4+bZ_1^4$,
$Y_4=(Y_1^2+aZ_4+bZ_1^4)*X_4+Z_4*bZ_1^4$.

Scalar multiplication algorithm

Scalar multiplication is performed using repeated point addition and doublings. Based on the value of 'k', either point addition or doubling is selected. In this algorithm, scalar 'k' is represented in binary form. It takes lesser number of doublings compared to normal binary method. This algorithm checks each bit, and if it is not LSB it will double the point or else it will perform the point addition. Algorithm pseudo-code and also flowchart for scalar multiplication is given below.

Input: An integer $k>0$, point $P(x,y)\in K$
Output=kP
  If $k=0$ or $x=0$ then (0,0) stop
 Set $k=(k_{l-1},k_{l-2},\ldots,k_1,k_0)$,
 Set $Q=O$ (Point at Infinity),
 Set LSB=k_0;
  For $k=l-1$ down to 0
   if $k_i=1$ then
   $P=P+Q$;
  else
  $P=Q$;
  if ($i\neq$LSB)
   $Q=2P$
   End for
  End if



Flowchart for scalar multiplication

Architecture for Scalar multiplication using Pure Projective Co-ordinates

The overall architecture of Pure Projective co-ordinates for scalar multiplication is presented below in which the inputs are projective co-ordinates for point addition and Point doubling. In this scalar, 'k' is represented in binary form. Multiplexer selects original or inverted point based on select line 'sel'. Projective Co-ordinates are given to Point doubling block to perform the point doubling operation. The output of point doubling and initial coordinates are serving as inputs to the point addition block. In both the blocks, key operations are multiplication, squaring and addition, and these are performed on binary fields. In this architecture the parameters of elliptic curve 'a' & 'b' are assumed as one (a=1 & b =1).

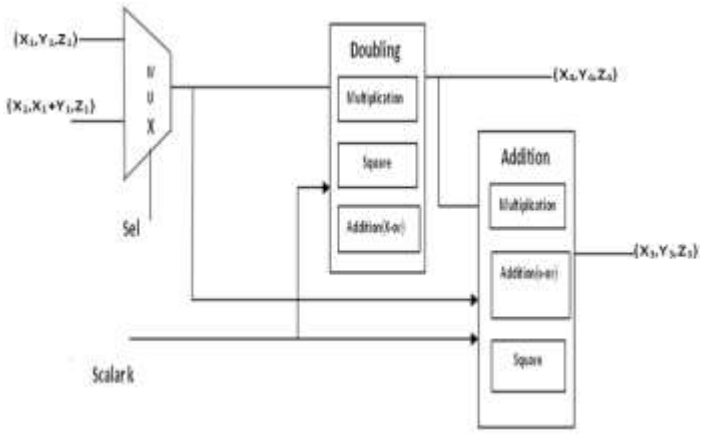


Fig (4) Overall architecture of Pure Projective Co-ordinates

Architecture for Scalar multiplication using Mixed Co-ordinates

The overall architecture of mixed co-ordinates for scalar multiplication is presented below in which the input is affine co-ordinates and it is converted to projective for point addition and doubling. In this scalar, '*k*' is represented in binary form. Multiplexer selects original or inverted point based on select line 'sel', and input affine coordinates (x_2,y_2) will be Converted to projective Co-ordinate (X_1,Y_1,Z_1) as $x_2=X_1$, $y_2=Y_1$ and $Z_1=1$. Converted Co-ordinates are given to Point doubling block to perform the point doubling operation. The outputs of Point doubling and initial coordinates are serving as inputs to point addition block. In both the blocks, key operations are multiplication, squaring and addition, and these are performed on binary fields. After scalar multiplication, it is again converted back to affine point. In this architecture the parameters of elliptic curve 'a' & 'b' are assumed as one (a=1 & b=1).

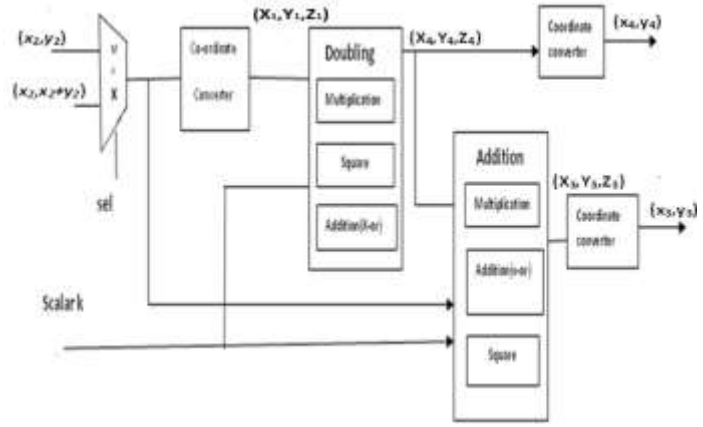


Fig (5) Overall architecture of Mixed Co-ordinates

Results and discussions

Synthesis is performed for point addition and doubling on both the mixed and pure projective coordinates for different bit sizes like 4, 8, 16 and 163 with Xilinx 9.1i and Virtex-4 with a speed grade of -12 as a target device. Synthesis results for point addition by using mixed coordinates are presented in Table 3. Pure projective coordinates syntheses results are given in Table 4. By comparing Tables 3 and 4, the mixed coordinate system occupies the less number of slices and high speed. Point doubling using pure projective coordinates is presented in Table 5. Finally, integrating point addition and point doubling in scalar multiplication for both pure projective coordinate and mixed coordinate are shown in Tables 6 and 7, respectively.

Comparison with previous work

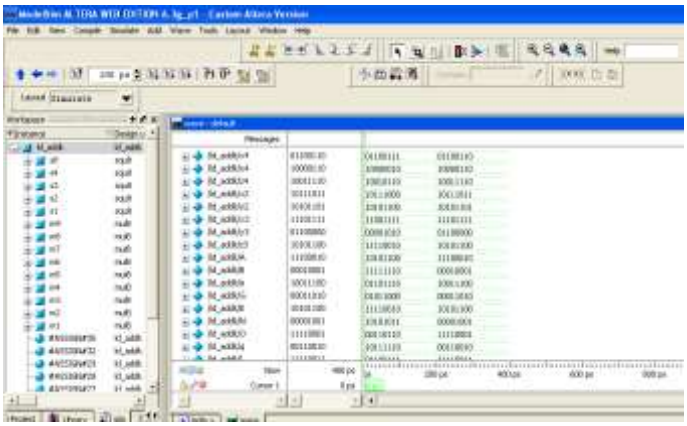
(1). Results are compared with the previous work on binary field with the Lopez–Dahab mixed coordinates system for point addition as shown in Table 8.

So we can conclude that it is twice faster than the previous work [27].

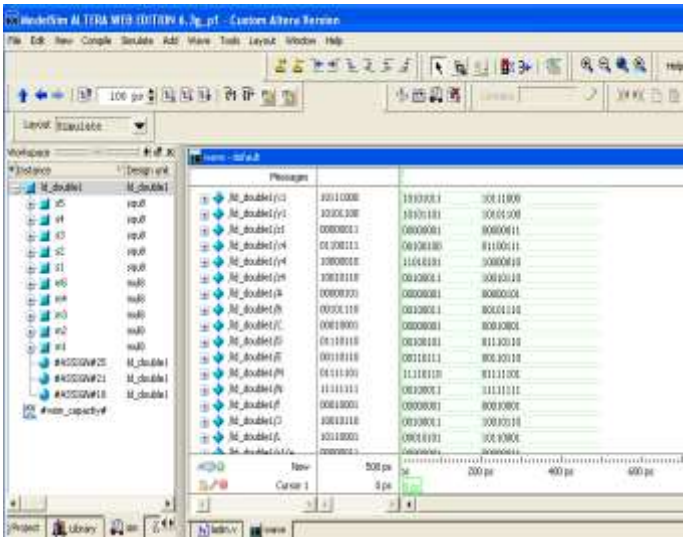
(2). Results are compared with the previous work on binary field with the Lopez–Dahab mixed coordinates system for scalar multiplication [19] (Table 9).

Simulation results

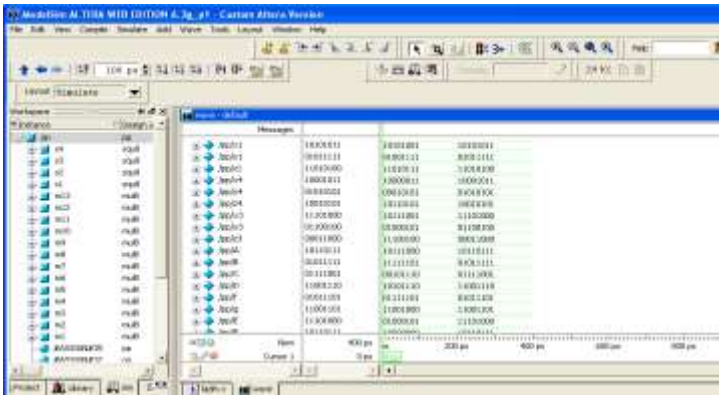
Lopez-Dahab Point addition using Mixed Co-ordinates for m=163 bits:



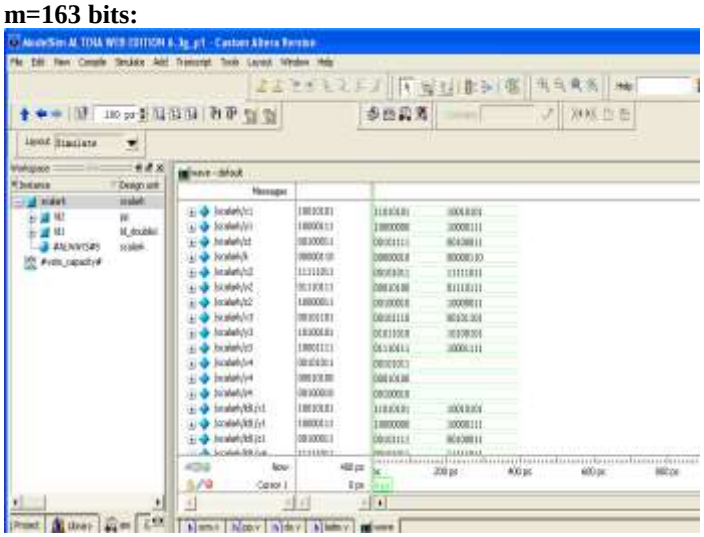
Lopez-Dahab Point doubling using pure projective for m=163 bits:



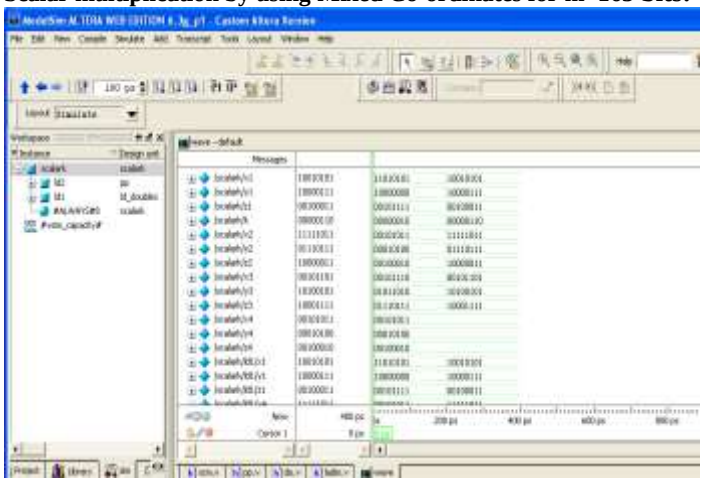
Lopez-Dahab Point addition using Pure Projective for m=163 bits:



Scalar multiplication using pure projective Co-ordinates for m=163 bits:



Scalar multiplication by using Mixed Co-ordinates for m=163 bits:



Conclusion

We proposed a high-speed hardware architecture of an elliptic curve point multiplication over $GF(2^{163})$ scheme using the mixed projective coordinate system. The point addition is performed on mixed coordinates and doubling using pure projective coordinates. Integration of these modules is done using the modified binary scalar multiplication algorithm. Synthesis is done using Xilinx 9.2i with virtex-4 as a target device. Simulation is done using Modelsim 6.2g. Our proposed design is two times faster than the previous work [27], and also this design is suitable for area and time constrained devices like smart cards and mobile devices.

***Future work*:** In ECC, point multiplication is more time consuming operation. In this architecture, it is performed using the mixed projective coordinates to achieve high-speed encryption/decryption, and it can be performed using the parallel architecture of point addition and doubling.

References

[1] N. Koblitz, "Elliptic curve cryptosystems", Mathematics of Computation, Vol.48, No 177,pp 203-209, 1987.

[2] V. Miller, "Uses of Elliptic Curve in Cryptography", Advances in Cryptology Crypto'85,Lecture Notes in Computer Science ,Vol. 218, pp. 417–426, 1986 ,Springer-verlag.

[3] Sining Liu, Brian King, Wei Wang, "Hardware organization to achieve high-speed elliptic curve cryptography for mobile devices", Mobile Networks and Applications, Vol.12, Issue 4, pp 271-279,2007, ACM.

[4] A. Kaleel Rahuman, G. Athisha, "Reconfigurable architecture for elliptic curve cryptography", Proceedings of the International Conference on Communication and Computational Intelligence, pp 461-466, 2010,IEEE.

[5] Yong-ping Dan, Xue-cheng Zou, Zheng-lin Liu, Yu Han, Li-hua Yi, "High-performance hardware architecture of elliptic curve cryptography processor over $GF(2^{163})$", Journal of Zhejiang University. Vol.10,No 2,pp 301-310,2009.

[6] Hao Li, Jian Huang, Philip Sweany, Dijiang Huang, "FPGA implementations of elliptic curve cryptography and Tate pairing over a binary field",Journal of Systems Architecture, Vol.54,Issue 12, pp 1077–1088,2008,Elsevier.

[7] Sameh M. Shohdy, Ashraf B. El-Sisi, Nabil Ismail, "Hardware implementation of efficient modified Karatsuba multiplier used in elliptic curves", International Journal of Network Security,Vol.11,Issue 3,2010,DOAJ.

[8] Zoya Dyka, Peter LangendoerferV, "Area Efficient Hardware Implementation of Elliptic Curve Cryptography by Iteratively Applying Karatsuba's Method", Proceedings of the conference on Design, Automation and Test in Europe, Vol.3, pp 73-75, 2005, ACM.

[9] Mohamed A. Fayed, M. Watheq, El-Kharashi Fayez Gebali, "A High-Speed, High-Radix, Processor Array Architecture for Real-Time Elliptic Curve Cryptography Over *GF* (2^m)", Signal processing and information technology, conference publications, pp 56-61, 2007, IEEE.

[10] Steffen Peter, Peter Langend¨orfer, "An efficient polynomial multiplier in *GF* (2^m) and its application to ECC designs" Design, Automation & Test in Europe Conference & Exhibition, conference publications, pp 1-6,2007, IEEE.

[11] William N. Chelton, "Fast elliptic curve cryptography on FPGA," IEEE Transactions on VLSI, Vol. 16, No. 2, February 2008,IEEE.

[12] Hans Eberle, Nils Gura, Sheueling Chang-Shantz, "A Cryptography Processor for Arbitrary Elliptic Curves Over $GF(2^m)$", Proceedings of Application-Specific Systems, Architectures, and Processors , pp 444 – 454,2003, IEEE.

[13] Henri Cohern, "Efficient Elliptic Curve Exponentiation Using Mixed Coordinates". Proceedings of the International Conference on the Theory and Applications of Cryptology and Information Security: Advances in Cryptology, pp 51-65,1998,ACM.

[14] Ray C.C. Cheung, "Customizable Elliptic Curve Cryptosystems", IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 13, No. 9, September 2005,IEEE.

[15] M. Prabu, Dr. Shanmugalakshmi, "A comparative and overview analysis of elliptic curve cryptography over finite fields", International Conference on Information and Multimedia Technology, pp 495-499,2009,IEEE.

[16] M. Morales-Sandoval, C. Feregrino-Uribe, R. Cumplido, I. Algredo-Badillo, "A Reconfigurable *GF*(2^m) Elliptic Curve Cryptographic Coprocessor", Progammable logic (spl) southern conference Proceedings, pp 209-214,2011, IEEE.

[17] B. Muthu Kumar, S. Jeevanathan, "High Speed Hardware Implementation of an Elliptic Curve Cryptography (ECC) Co-Processor", 2010, IEEE.

[18] Chang Hoon Kim, Soonhak Kwon, Chun Pyo Hong, "FPGA implementation of high performance elliptic curve cryptographic processor over *GF*(2^{163})", Journal of Systems Architecture, Vol. 54,Issue 10, pp 893–900, 2008,ACM.

[19] Yu Zhang, Dongdong Chen, Younhee Choi, Li Chen, Seok-Bum Ko, "A High Performance ECC Hardware Implementation with Instruction-Level Parallelism Over $GF(2^{163})$", Journal of microprocessors & microcontrollers, Volume 34, Issue 6, pp 228-236, Oct 2010 ,ACM.
[20] William Stallings, "Cryptography and Network Security", Third Edition, <www.williamstallings.com/Crypto3e.html>.
[21] G.-J. Lay, H.G. Zimmer, "Constructing elliptic curves with given group order over large finite fields",Proceedings of the First International Symposium on Algorithmic Number Theory, Lecture Notes in Computer Science, Vol. 877, pp 257-263,1998 Springer.
[22] M. Rosing, "Implementation Elliptic Curve Cryptography". 1999, Manning Publications.
[23] I.F. Blake, "Advances in Elliptic Curve Cryptography", London Mathematical Society Lecture Note Series, 2005 Cambridge University Press Series.
[24] Chester Rebeiro, Debdeep Mukhopadhyay, "High Performance Elliptic Curve Crypto-Processor for FPGA Platforms",12th VLSI Design and Test Symposium, 2008,IEEE.
[25] Darrel Hankerson, Alfred Menezes, Scott Vanstone, "Guide to Elliptic Curve Cryptography". 2004, Springer
[26] J. Lopez, R. Dahab, "Fast multiplication on elliptic curves over $GF(2^m)$", Cryptographic Hardware and Embedded Systems (CHES), Vol. 1717 , pp. 316–327, LNCS 1999, Springer.
[27] B. Muthu Kumar, S. Jeevanathan, "High speed pipe lined ASIP processor for FPGA technology", International Journal on Information Science and Engineering, Vol .4, No.1, 2010.

Table 1. Comparison of ECC and RSA in terms of key size

ECC key size	RSA key size	Key size ratio
163	1024	1:6
256	3972	1:12
384	7680	1:20
512	15,360	1:30

Table 2. Comparison of mixed and pure projective coordinates

Coordinates	No. of multiplications	No. of squaring	No. of additions
Pure projective coordinate (point addition)	13	6	8
Mixed coordinate (point addition)	8	5	5
Point doubling (pure projective)	3	3	4

Table 3. Point addition by using mixed coordinates

No. of bits	Delay (ns)	Slices
4	**15.746**	**75**
8	**20.446**	**254**
16	**26.122**	**1027**
163	**40.691**	**38k**

Table 4. Point addition using pure projective coordinates

No. of bits	Delay (ns)	Slices
4	**18.419**	**155**
8	**22.244**	**383**
16	**29.732**	**1719**
163	**47.271**	**65k**

Table 5. Point doubling using pure projective coordinates

No. of bits	Delay (ns)	Slices
4	**10.10**	**30**
8	**11.406**	**119**
16	**16.395**	**424**
163	**21.708**	**10k**

Table 6. Scalar multiplication using pure projective coordinates

No. of Bits	Delay (ns)	Slices
4	**21.577**	**194**
8	**23.270**	**502**
16	**34.343**	**2157**
163	**40.237**	**35k**

Table 7. Scalar multiplication by using mixed coordinates.

No. of Bits	Delay (ns)	Slices
4	**11.208**	**113**
8	**21.208**	**379**
16	**23.181**	**706**
163	**35.086**	**30k**

Table 8. Point addition by using mixed coordinates compared with previous work

No. of Bits	Our proposed work Time (ns)	Muthu Kumar et al.[27] Time (ns)
4	15.746	8.227
8	20.446	27.278
12	–	64.082
16	26.122	–
18	–	138.064

Table 9. Scalar multiplication by using mixed coordinates compared with previous work

Bits (m=163)	Our proposed work	Zhang et al. [19]
Delay	35.086 ns	7.7 µs