International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 5, May - 2014

# Reconfigurable Architecture For Efficient Elliptic Curve Scalar Multiplier

Geetha M
PG Scholar, TPGIT, Vellore

Dr. Rahila Bilal
Associate Professor of ECE, TPGIT, Vellore

*Abstract* - **Elliptic Curve Cryptography makes a good choice for implementing security services in constrained devices, like the mobile ones. However, the diversity of ECC implementation parameters recommended by international standards has led to interoperability problems among ECC implementations. This work presents the design and implementation results of a novel FPGA coprocessor for ECC than can be reconfigured at run time to support different implementation parameters and hence, different security levels. FPGA based architecture of elliptic curve cryptography coprocessor is proposed in this paper. Experiment results show that coprocessor designed in this paper can achieve high performance. In $GF(2^{163})$, we achieve a point multiplication in 13.38 *n*s in Xilinx Virtex-E. Using the modern Xilinx Virtex-5, the point multiplication $GF(2^{163})$ is achieved in 3.480*n*s, and it consumes less number of LUTs compared to other devices.**

## 1.INTRODUCTION

The rapid advances in information technology in the past few decades have led to intensive research on information security. Many technologies and crypto graphical systems have been developed, all to secure information and protect it from un authorized persons. Cryptography is the science of writing in secret code and is an ancient art and not only protects the data, but can also be used for user authentication. There are several ways of classifying cryptographic algorithms. They will be categorized based on the number of keys that are employed for encryption and decryption, and further defined by their application and use. The three types of algorithms are: Public key cryptography: Use a single key for both encryption and decryption. Symmetric key cryptography: Uses one key for encryption and another key for decryption. Hash functions: Uses a mathematical transformation to irreversibly "encrypt" information.

Public-key cryptography has been widely studied and used since 1975 when Rivest, Shamir, and Adleman invented RSA public key cryptography. This system heavily depends on integer factorization problem (IFP). In 1985, Koblitz and Miller used EC in cryptography using elliptic curves discrete logarithm problem (ECDLP) in [3] and [1]. In recent years, researchers have given more attention to develop the proposed ECC algorithms and improve their efficiency. Elliptic Curve Cryptography is a kind of cryptography that provides the security information services using shorter keys than other known public-key crypto-algorithms without decreasing the security level. Improving the efficiency of scalar multiplication in EC is one of the main interests of many researchers in the field of cryptology. The techniques proposed so far use different methods for representing the scalar *k,* which clearly shows different levels of computation speed and security.

ECC-based cryptographic schemes need to define a tuple $T$. Several tuples $T$ have been recommended for standards, like the National Institute of Standards and Technology NIST [5] or the Standards for Efficient Cryptography Group SECG [4]. The diversity of choices to implement ECC and the several tuples $T$ recommended by international standards has led to interoperability problems.

ECC implementations can be categorized into reconfigurable and non-reconfigurable classes. In a reconfigurable implementation, the Galois field, over which the elliptic curve is defined, can be changed without the need to change the design. In a non-reconfigurable design, the FPGA must be reprogrammed in order to change the field.

In this context, interoperability is understood as the ability of two ECC implementations (either in software or hardware) to work together and communicate, for example one ciphering and the other deciphering. However, most of the ECC hardware implementations of elliptic curve cryptography are focused on implementing efficiently the scalar multiplication operation $dP$ [6, 7, 8, 9, 10, 11].This work aims to provide a flexible solution that can dynamically switch to different

implementation parameters, instead of custom high performance solutions for a specific tuple $T$.

The rest of this paper is organized as follows. In section2 we describe the mathematical background of elliptic curve cryptography. Section 3 describes the architecture and point operation such as addition, doubling, scalar multiplication on EC. The results are discussed in section 4 and finally, concluding remarks and further directions are presented in section 5.

## 2. MATHEMATICAL BACKGROUND

### *2.1 Galios Field*

Galois field arithmetic plays a critical role in elliptic curve cryptography implementation because it's the core of ECC scalar multiplication. Galois field or Finite field (F) defines as $GF(p^m)$ which is a field with finite number of elements ($p^m$ elements with p is a prime number called characteristic of field) and two binary operation addition and multiplication. Furthermore, Order of Galois field is the number of elements on the Galois field [12, 13].

Galois fields suitable for ECC implementation divides into two categories: prime field where m = 1 and binary field where p = 2 and m > 1. Binary Galois field preferred in hardware because of free carry propagation property in hardware.

### *2.2 Binary Field*

Finite field of order $2^m$ is called binary field. Suppose Binary field ($F2^m$) and we have two elements A, B $\in F2^m$. Addition does not have any carry propagation and can be done by one n bit XOR operation, multiplication done by ordinary multiplication (a•b) modulo irreducible polynomial P(x) in $F2^m$, square operation done with no hardware resource rather than in ($F_p$) is cost as a general multiplication and faster Inversion operation in $GF(2^m)$.

Instead of the dual field approaches, ECC over binary field $GF(2^m)$ can achieve a high throughput inherently because there is no carry propagation in the arithmetic operations, resulting in fast and compact implementations proposed recently.

### *2.3 Point Addition And Doubling*

Any point multiplication will be done with a sequence of point additions, so to minimize the total cost one should consider both the point addition algorithm and the sequence in which the operations will be performed.

- Point Addition-ADD to sum two distinct points P,Q $\in E(K)$.
- Point doubling - ECC-Dbl to sum a point $\in (K)$ to itself.

### 2.4 Projective coordinate system

Several projective coordinate systems for elliptic curve equation have been proposed in order to avoid the time-consuming inversion operation [14]. Projective coordinate system proposed by Lopez and Dahab is suitable for hardware implementation, and it is called L-D projective coordinates in this paper.

Projective coordinates involve representing a curve point as a triplet $x, y, z \in GF(q)$, i.e., $P(x, y, z)$. In the L-D projective coordinates, *point* $(X{:}Y{:}Z)(Z\neq 0)$ is corresponding to *point* $(X/Z,\ Y/Z^2)$ in the affine coordinates, and the elliptic curve equation is simplified as below.

$Y^2+XYZ = X^3Z+aX^2Z^2+bZ^4$ (1)

3.Point Operation on Elliptic Curve

Both point addition and subtraction operations are foundation of scalar multiplication. Point addition operation includes addition of two different points and two same point doubling. In this section, we will discuss the implementation of point operations inside FPGA in detail based coprocessor in L-D projective coordinates.

Since the modular inversion operation over field is a time consuming operation, points on elliptic curve should be represented in projective coordinates to avoid it. In the L-D projective coordinates, elliptic curve equation is the same as formula (1). The points addition formula that do not involve modular inversion operation can be derived by converting the point to affine projective as $x=X/Z$, $y=Y/Z2$ at first, then adding the affine points with the formula, and finally clearing the denominators. With the method mentioned above, the distinct points adding and the same point doubling formula can be derived.

As for two distinct points adding, suppose the first point $P(X1,Y1,Z1)$ and the second point $Q(X2,Y2,Z2)$ are on elliptic curve, the adding result of the two different point is $R(X3,Y3,Z3)$ as following.

$(X_1,Y_1,Z_1)+(X_2,Y_2,Z_2)=(X_3,Y_3,Z_3)$ (2)

The two distinct points adding formula is shown as in detail [17], and it requires 13 field multiplications.

| | |
|---|---|
| 1. $A_0 = Y_2 \cdot Z_1^2$ | 9. $Z_3 = F^2$ |
| 2. $A_1 = Y_1 \cdot Z_2^2$ | 10. $G = D^2 \cdot (F+aE^2)$ |
| 3. $B_0 = X_2 \cdot Z_1$ | 11. $H = C \cdot F$ |
| 4. $B_1 = X_1 \cdot Z_2$ | 12. $X_3 = C^2 + H + G$ |
| 5. $C = A_0+A_1$ | 13. $I = D^2 \cdot B^0 \cdot E + X^3$ |
| 6. $D = B_0+B_1$ | 14. $J = D^2 \cdot A_0 + X_3$ |
| 7. $E = Z_1 \cdot Z_2$ | 15. $Y_3 = H \cdot I + Z_3 \cdot J$ |
| 8. $F = D \cdot E$ | |

It is necessary to note that when using the double and add method for scalar multiplication $k \cdot P$ the point $P$ is never modified in all distinct point adding. In mixed coordinates (one is projective point and the other is affine point) the point $P$ maintaining in affine coordinates will simplify the computing. The point adding in mixed coordinates

$(X_0, Y_0, Z_0)+(X_1, Y_1, 1) = (X_2, Y_2, Z_2)$

is computed by

| | |
|---|---|
| 1. $A = Y_1 \cdot Z_0^2 + Y_0$ | 6. $E = A \cdot C$ |
| 2. $B = X_1 \cdot Z_0 + X_0$ | 7. $X_2 = A^2 + D + E$ |
| 3. $C = Z_0 \cdot B$ | 8. $F = X_2 + X_1 \cdot Z_2$ |
| 4. $D = B^2 \cdot (C + aZ_0)$ | 9. $G = X^2 + Y_1 \cdot Z_2^2$ |
| 5. $Z_2 = C^2$ | 10. $Y_2 = E \cdot F + Z_2 \cdot G$ |

The point doubling operation is to add a point on the elliptic curve with itself. Suppose point $P(X_1,Y_1,Z_1)$ and the projective coordinate form of doubling formula is

$2P(X_1,Y_1,Z_1) = Q(X_2,Y_2,Z_2)$ (3)

Implementation of the operation requires 4 field multiplications. The point doubling is computed by

$Z_2 = Z_1^2.X_1^2$
$X_2=X_1^4+b.Z_1^4$
$Y_2=bZ_1^4+X_2.(aZ_2+Y_1^2+bZ_1^4)$

The following algorithm implements a full point addition in the projective coordinates.

```
1.   If $Z_1=0$, then output
     $(X_3,Y_3,Z_3)=(X_2,Y_2,Z_2)$ and stop.
2.   If $Z_2=0$, then output
     $(X_3,Y_3,Z_3)=(X_1,Y_1,Z_1)$ and stop.
3.   Set $(X_3,Y_3,Z_3)$=Add[$(X_1,Y_1,Z_1)$ $(X_2,Y_2,Z_2)$].
4.   If $(X_3,Y_3,Z_3)=(0,0,0)$, then
     set $(X_3,Y_3,Z_3)$=Double[$(X_1,Y_1,Z_1)$].
5.   Output$(X_3, Y_3, Z_3)$.
```

## 4. ARCHITECTURE AND ALGORITHM

There are several different algorithms for performing elliptic curve scalar point multiplication. A survey of such algorithms can be found in [15] and [16]. The most used algorithms are the following: 1) double and add; 2) nonadjacent form (NAF) addition–subtraction chain; and 3) Montgomery ladder product.

Among these algorithms, the NAF method, without pre computing, is more suitable for hardware implementation for its concision and efficiency. The following NAF algorithm is adopted in our design of coprocessor.

```
1. set $h_l h_{l-1} ... h_1 h_0$ is binary representation
   of 3$k$.
2. set $k_l k_{l-1} ... k_1 k_0$ is binary representation
   Of $k$.
3. set $S=Q$.
4.For $i$ from $l$–1 Downto 1 Do
      4.1 set $S = 2S$.
      4.2 if ($h_i$=1 && $k_i$=0) $S=S+Q$;
      4.3 if ($h_i$=0 && $k_i$=1) $S=S-Q$;
5.output $S$.
```

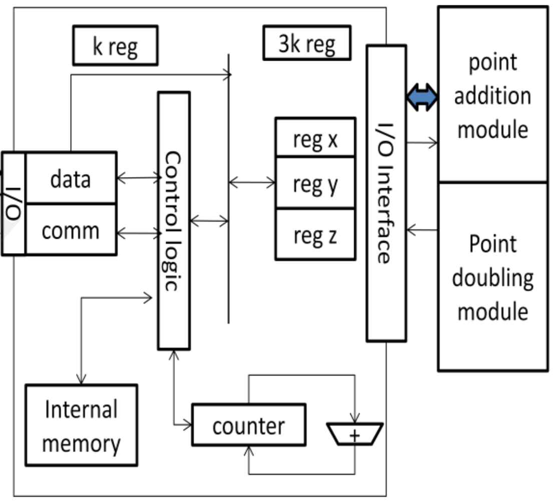

Fig.1 Architecture of scalar multiplier

The architecture of scalar multiplier module in FPGA device is depicted as Figure 1. The Point doubling and point addition operations are done by using their modules presented in the above architecture and control logic controls the all operations. Reg x, y ,z used for temperorary storage. A very small amount of clock cycles is required to read and write the data in the internal memory.

For example, two different elliptic curve points addition requires 9 field multiplications in mixed coordinate system, one is affine coordinate system and the other is LD projective coordinate system. A point doubling require 4 field multiplications.

## 4. EXPERIMENT AND RESULTS

The FPGA based architecture of EC Scalar Multiplier introduced above is described with VHDL. The VHDL description is correctly simulated in ModelSim Altera ana the simulation result is shown in Fig.2. The synthesis of the description is finished in Xilinx's integrated software environment. The placing and routing process are finished in Xilinx's Virtex-5 device.

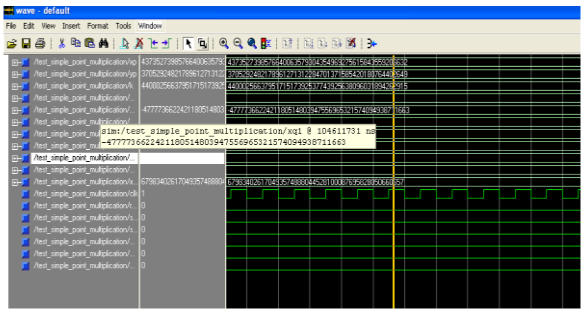
wave - default
File Edit View Insert Format Tools Window
sim:/test_simple_point_multiplication/xq1 @ 104611731 ns
-4777736622421180514803947556965321574094938711663
/test_simple_point_multiplication/clk

Fig.2 Scalar multiplication for Binary field 163 using VHDL code

The time to perform the scalar multiplication is given in table 1. This paper summarizes several coprocessor published in recent years and list them in Table 1 with our own design in it. The data in the table mainly shows time for scalar multiplication for 163 bit is 3.480ns.

Table 1. Comparison of Scalar Point Multiplication in various devices

| Devices | Spartan3E | Virtex E | Virtex 4 | Virtex 5 |
|---|---|---|---|---|
| No.of Slices | 4223 | 4274 | 4208 | 5968 |
| No.of LUTs | 7523 | 7590 | 7523 | 6611 |
| No.of Bonded IOBs | 334 | 334 | 334 | 334 |
| Time(ns) | 6.504 | 13.831 | 5.734 | 3.480 |

## 5.CONCLUSION

A new design of trade of scalar multiplier is proposed in this paper. This is scalable and programmable architecture that exploits the reconfigurabality to deliver optimized solutions for different elliptic curves and finite fields. The architecture of our design is described in VHDL simulated using Modelsim Altera and synthesized using Xilinx. This scalar multiplier achieves a better performance shown in above comparison. The architecture is presented using the modern Xilinx Virtex-5 FPGA device results 5560 occupied bit slices, 6611 LUTs with time delay 3.480*ns* for $2^{163}$ point multiplication.

## REFFERENCES

[1]. V. Miller, *Use of elliptic curves in cryptography*, in A. M. Odlyzko, editor, Advances in cryptology - CRYPTO '86, volume 263, of lecture notes in comput. sci. pages 417-426, Spriner - Verlag, 1987. Proceedings, Santa Barbr (USA), August 11-15, 1986.

[2]. W. Diffie and M. E. Hellman, *New directions in cryptograpgy*, IEEE Trans.Inform. Theory, IT-22(6), November 1976.

[3]. N. *Koblitz, Elliptic curve cryptosystems*, Math. Comp., 48(177): 203-209, January 1987.

[4]. SEC 1, "Elliptic curve cryptography: Standards for efficient cryptography group," 2000, http://www.secg.org.

[5]. NIST, "Recommended elliptic curves for federal government use," 1999,http://csrc.nist.gov/csrc/fedstandards.html.

[6]. L. Batina, N. Mentens, B. Preneel, and I. Verbauwhede, "Balanced point operations for side-channel protection of elliptic curve cryptography," in IEE Proceedings of Information Security, vol. 152, 2005, pp. 57–65.

[7]. M. Ernst, M. Jung, F. Madlener, S. Huss, and R. Blumel, "A Reconfigurable System on Chip Implementation for Elliptic Curve Cryptography over GF($2n$)," in Proc. Of CHES'2002, ser. LNCS, vol. 2523. Redwood Shores, CA: Springer, 2002, pp. 381–399.

[8]. N. Gura, S. C. Shantz, H. Eberle, S. Gupta, and V. Gupta, "An End to End Systems Approach to Elliptic Curve Cryptography," in Proc. of CHES 2002, ser. LNCS, vol. 2523. Springer, 2002, pp. 349–365.

[9]. N. Mentens, S. B. Ors, and B. Preneel, "An FPGA Implementation of an Elliptic Curve Processor GF($2m$)," in Proceedings of the 14th ACM Great Lakes symposium on VLSI, Boston, MA, 2004, pp. 454 457.

[10]. K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, "Superscalar coprocessor for high-speed curve-based cryptography," in Proc. of CHES 2006, ser. LNCS, vol. 4249. Springer-Verlag, 2006, pp. 415–429.

[11]. N. Saquib, F. Rodriguez, and A. Diaz, "A Parallel Architecture for Fast Computation of Elliptic Curve Scalar Multiplication over GF($2n$)," in Proc. of RAW'04, Sta. Fe, USA, 2004, pp. 26–27.

[12]. D. Hankerson, M. Alfred, and V. Scott, Guide to Elliptic Curve Cryptography, Springer-Verlag, ISBN 038795273, 2004.

[13]. R. J. McEliece, Finite Fields for Computer Scientists and Engineers, Kluwer Academic Publishers, 1987.

[14]. IEEE Std P1363-2000. "IEEE Standard Specifications for Public-Key Cryptography". 2000.8

[15]. J.-P. Deschamps, J. L. Imaña, and G. D. Sutter, Hardware Implementation of Finite-Field Arithmetic. New York: McGraw-Hill, 2009, ser. Electronic Engineering Series.

[16]. W. N. Chelton and M. Benaissa, "Fast elliptic curve cryptography on FPGA," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 2, pp. 198–205, Feb. 2008.

[17]. J.Lopez, R.Dahab. "Improved Algorithm for Elliptic Curve Arithmetic in GF(2n) ". Selected Areas in Cryptography- SAC'98, LNCS 1556, 1999: pp. 201-212.