Parallel implementation for ECCP based on Montgomery ladder algorithm

Shaimaa Abu Khadra¹, Salah Eldin S E Abdulrahman² and Nabil A Ismail²

¹Al-Mahala High Institute of Engineering, Al-Mahala, Egypt.
²Dept. of Computer Science and Engineering, Faculty of Electronic Engineering, Menoufia University, Menoufia 32952, Egypt.

E-mail: eng.abukhadraa@yahoo.com

Abstract. An Elliptic Curve Crypto-Processor (ECCP) is a favourite public-key cryptosystem. It is used for embedded systems due to its small key size and its high security arithmetic unit. It is applied in constrained devices which often run on batteries and have limited processing, storage capabilities and low power. A finite field polynomial multiplier takes the most implementation effort of an ECCP because it is the most consuming operation for time and area. So, it is preferable to optimize this operation especially for light devices where the small area is needed. This research introduces a hardware design for parallel ECCP binary implementation that is based on Montgomery ladder algorithm. This implementation is targeted for GF(2¹⁶³) and GF(2⁴⁰⁹) where the executed time are 2.9 μs and 29 μs respectively. The implementation is performed on Xilinx ISE Virtex6.

1. Introduction

With advanced technology there is a necessity of secure multiple means of communications and data transmission between devices [1]. The insurance is importance for embedded systems that ranged from light devices to satellites around the earth [2]. Each embedded system contains many devices connected with each other. Embedded systems are found in all products that use electricity from home products to medical systems to cars, as all of these contain microprocessor equipped within it and equipped also with modern devices and technology like WIFI, GSM, GPS, Bluetooth or other device that used in communication and remote control [3]. The core operation in a public key cryptosystem is depend on encrypt data using a public key and decrypt data using a private one. Cryptography means use a mathematical formula (algorithm) to change text from one form to another. This operation needs to execute very fast, however, in the same time it must kept a balance between security need and energy consumption and area.

ECCPs have occupied the center stage of the public key cryptographic research. In addition, ECCPs have also been included in IEEE 1363 [4] and NIST [5]. Hardware may be the fastest/most efficient way to implement cryptographic algorithms. So, the use of ECCP is preferred comparing with classical cryptosystems because of using ECCP guarantees the same security level but with shorter key size. The elliptic curve cryptography is one of the public key schemes that use finite field (Galois Field (GF)) arithmetic to do its operation as introduced in figure 1, which indicated the ECCP layer. So, it's useful to optimize the finite field operations for ECCPs in order to reduce area and power consumption [6]. The finite field includes four operations, which are addition, subtraction, multiplication and inversion.
The carry property is removed from the implementation of the binary field which makes the finite field implementation running in a very fast way. So, the implementation of the binary field is the easiest design between other fields. This work focuses on the implementation of the binary field over $\text{GF}(2^{163})$ and $\text{GF}(2^{409})$. This work is concerned with the implementation of an optimized ECCP for resource restricted environments in terms of hardware usage. This work also incorporates area, and computation time trade off through an optimized implementation process, which maps the optimized data path and controller analysis in the form of highly efficient hardware implementation.

The parts of the paper are presented as follows: Elliptic curve arithmetic is discussed in the next part. Parallel operation architecture design is presented in part 3. Related work is appearing in part 4.
Elliptic curve arithmetic

The elliptic curve arithmetic is identified in terms of underlying field operations [8]. A generic standard binary EC $E(GF(2^m))$ can be represented as:

$$E(GF(2^m)): y^2 + xy = x^3 + ax^2 + b$$  \hfill (1)

Where $a, b$ are in $GF(2^m)$ and $b \neq 0$

**Algorithm 1** Montgomery ladder PM for $GF(2^m)$ [7]

| Input: $k = (k_{m-1}, \ldots, k_1, k_0)$ with $k_{m-1} = 1, P = (x_p, y_p) \in GF(2^m)$. |
| Output: $Q = (x_k, y_k) = k \cdot P$. |

Step 1 (Initializations):

1. $X_1 = x_p, Z_1 = 1, X_2 = x_p^4 + b, Z_2 = x_p^2$  

Step 2 (PM):

for ($i = m - 2$ down to 0) loop

| if ($k_i = 1$) | $P = P + Q$ Return $P(X_1, Z_1)$ | $P = 2P$ Return $Q(X_2, Z_2)$ |
| Return $P(X_2, Z_2)$ |

| if ($k_i = 1$) | $P = P + Q$ Return $Q(X_1, Z_1)$ | $P = 2P$ Return $Q(X_2, Z_2)$ |
| Return $Q(X_1, Z_1)$ |

| $R_1 = X_2Z_1$ | $Z_2 = Z_2^2$ | $R_1 = X_2Z_1$ |
| $R_2 = X_2Z_2$ | $Z_2 = Z_2^2$ | $R_2 = X_2Z_2$ |
| $Z_1 = R_2+R_1$ | $R_2 = bR_2$ | $Z_2 = Z_2^2$ |
| $Z_2 = X_2Z_2$ | $R_1 = R_1R_2$ | $Z_1 = X_1Z_1$ |
| $R_1 = X_2R_2$ | $R_2 = x_pZ_2$ | $X_1 = X_1^2$ |
| $R_2 = x_pZ_1$ | $X_2 = R_2+R_1$ | $X_1 = X_2+X_1$ |

Step 3 (Reconversion):

$x_k = \frac{x_pZ_2X_1}{x_pZ_1Z_2}$  
$y_k = \frac{(x_p + x_k)[(X_1 + x_pZ_1)(X_2 + x_pZ_2) \pm (x_p^2 + y_p)Z_1Z_2]}{x_pZ_2Z_2} + y_p$

An ECCP is designed and build based on the main operation which is called scalar or Point multiplication (PM) $Q = k \cdot P = P + P + \ldots + P$ and it can be done using repeated Point Additions (PA) and Point Doubling (PD), for example $11P = 2(5P) + P = 2(2(2P) + P) + P$. The Lopez-Dahab (LD) projective coordinates which are used to calculate the PM of the binary ECCP in equation (1), are resulted from applying algorithm 1.

All operations of the PM require finite field operations like inversion, squarer, polynomial multiplier and addition [8]. This work adopted with $GF(2^{163})$ and $GF(2^{409})$ binary fields. The $GF(2^m)$ field is more suitable for hardware design in which the addition operation requires only XOR unit and eliminates the needed for carry propagation. The square operation done with no area and it is meant by inserting zero between bits. A Polynomial multiplier may be implemented in a bit serial or a bit parallel multiplier [9], A bit serial multiplier is a good choice for area, but a bit parallel are a good choice for time. Both the polynomial squarer and the polynomial multiplier are needed to follow with irreducible polynomial. The irreducible polynomial for $a(z) \mod p(z)$ was meant by the reminder of a long division of $a(z)$ by $p(z)$. The irreducible polynomial was implemented in hardware by using shift and XOR operation. Finally, the inversion is the most complicated unit as it takes a large area and has
a slow implementation. The Itoh-Tsujii [10] algorithm is one of the inversion algorithms that convert the inversion operation to run based on two finite field units which are a multiplier and squarer units, then it takes only nine multiplications for $GF(2^{163})$ or ten multiplications for $GF(2^{409})$ and $(m-1)$ repeated squaring operations.

3. Parallel operation architecture design
ECCP cryptosystem can be designed by using 3 multiplier, and 5 squarer and 2 adder unit, so the first stage which is the conversion from affine to projective takes 1 clock cycle to execute, second stage which is the group operation of PA and PD takes two steps, where the number of clock cycle for PA and PD are depending on the polynomial multiplier, so if the polynomial multiplier takes one clock cycle, then takes two clock cycles to execute. ECCP Scalar multiplication is designed by using liveness analysis, bypass scheduling, and implementation of parallel operation specially between different unit [11]. The final stage of algorithm is the conversion from projective to affine are depending on the execution of inversion. The three stages of the PM are discussed in the next sections.

3.1 Convert from affine to projective
The first stage in Algorithm 1 aims to compute the four values $X_1$, $Z_1$, $X_2$, $Z_2$. These values are computed from the equations: $X_1 = x$, $Z_1 = 1$, $X_2 = x^4 + b$, $Z_2 = x^8)$. These values represent the inputs of the second stage of Algorithm 1. The second stage is the group operation of computing the Point Addition (PA) and the Point Doubling (PD) in the projective coordinates. The first stage that converts from affine to projective takes only one clock cycle for computation.

3.2 Group operation for PA and PD
The PA and PD depend on using GF arithmetic units, Polynomial multipliers are the most time/area consuming operations, so the cost of PA and PD depends on the cost of multiplier unit. As seen in figure 2, the group operation for PA and PD is designed to use three GF multiplier units to run in parallel. The scheduling operation for PA and PD that appeared in the equation (2) lead the computation to execute in two steps.

The Point Addition $(Z_a, X_a)$ is given by

$$Z_a = (X_1 Z_2 + X_2 Z_1)^2,$$

$$X_a = x_p Z_a + (X_1 Z_2)(X_2 Z_1)$$

And Point Doubling $(Z_d, X_d)$ is given by

$$Z_d = X_1^2 Z_1^2,$$

$$X_d = X_1^4 + b Z_1^4. \quad (2)$$

The operation that used to find $X_2 Z_1, X_1 Z_2, X_1^2 Z_1^2$ are performed in the first step using multipliers M1, M2 and M3 respectively, where the result for this step is the computation for $(Z_a, Z_d)$ and the value of M1 and M2 are stored in two registers (R1,R2) in order to passes for the next step. The operation that used to find the multiplication of (R1R2), $x_p Z_a$ and $b Z_1^4$ are computed by the same multipliers to compute $(X_a, X_d)$. Figure 2 illustrates the operation of the PA and PD and indicates the implementation for equation (2). As appearing in figure 2, the number of clock cycles which needed to compute PA and PD depend on the number of clock cycles to compute the polynomial multiplier which takes four clock cycles to complete its calculation. If the PA and PD are build based on the parallel multiplier that take one clock cycle to run, then the computation for PA and PD takes two clock cycles rather than eight clock cycles. The proposed structure for the computation of PA and PD needs four registers (R0,R1,R2,R3), where (R1,R2) are used two times inside the PA and PD to store the value of M1,M2, and then are reused to store the value of $(X_a, X_d)$. Again the four registers are used to repeat the operation of the PA and PD as appearing in figure 2.
\[ x_k = \frac{x_p z_2 x_1}{x_p z_1 z_2}, \quad y_k = \frac{(x_1 + z_2)(x_1 + x_p z_2) + (x^2 + y_p z_1 z_2)}{x_p z_1 z_2} + y_p \]  

It is indicated in equation (3) that it requires to apply one GF inversion calculation, in order to find the result for \((x_1 z_2)^{-1}\). The cost for the conversion from projective to affine is calculated based on number of multiplication and inversion operations, where it needs ten multiplications and one inversion operation. Itoh-Tsujii algorithm proposed in [6] is used to implement inversion operation. The inversion unit is implemented based on two finite field units which are multiplications and squaring operations, so no need for extra unit to perform GF inversion operation. Table 1 indicates the details of multipliers usage based on equation (3). Multiplexers are used for selecting the related input signals in the computation steps, where there are three signal used with the multiplexers for selecting the related input, it is denoted as \(c_2, c_1, c_0\) where it may be used all, or some in each clock cycle, for example, in the first clock cycle, the multiplexers for M1 uses two signal \(c_2, c_1\) to select the related input, and as same as M2 and M3 uses the appropriate signal to select to the related input.
Table 1. Scheduling operation for conversion stage.

| Signal | M1 | M2 | M3 |
|--------|----|----|----|
| \(c_2c_1c_0 = 000\) | \(x_pZ_1\) | \(R_0 = x_pZ_2\) | \(Z_1Z_2\) |
| \(c_2c_1c_0 = 010\) | \((M_1 + X_2) (M_2 + X_1)\) | \(Z_2 * M_1\) | \((x_p^2 + y_p)M_3\) |
| \(c_2c_1c_0 = 100\) | \(inv = inv(M_2)\) | \(R_0 * X_1\) | \(R_1 = M_3 + M_1\) |
| \(c_2c_1c_0 = 110\) | \(x_k = inv * M_2\) | \(inv * R_1\) |
| \(c_2c_1c_0 = 111\) | \(y_k = (M_1 + x_p) * M_2 + y_p\) |

4. Related work

Several implementations of ECCP discuss the parallel architecture of ECCP design like [12, 13]. Since they use three multiplier units and two step to complete Point Addition (PA) and Point Doubling (PD) operations. Where in [12] it depends on Montgomery ladder for field GF(2\(^{163}\)) and GF(2\(^{233}\)) but it uses a pipeline digit-serial multiplier for the polynomial multiplier and Itoh-Tsujii for the inversion. In [13] it also depends on using the Montgomery ladder for field GF(2\(^{191}\)) but with karatsuba-Ofman for the polynomial multiplier and Extended Euclidian Algorithm (EEA) for the inversion. The work in [14] investigates the effect of digit size on area and time of the polynomial multiplier. It is also discusses different architecture for ECCP based on the Montgomery ladder algorithm when using either one or three multipliers. However, the work in [16] presented an ECCP architecture design for IoT security using the Montgomery ladder algorithm running sequentially. All the previous work results appeared in figure 3.

Figure 3. previous work comparison

5. Technical detail

ECCP used in many applications where some of them require high speed like web transaction and other require a minimum area like mobile application. Figure 4 is the proposed structure for Montgomery scalar multiplication based on LD projective coordinates. Figure 5 is the design implementation for PA and PD that needed to complete loop iteration for any scalar multiplication. The output from figure 4 is passed to the final step that makes the conversion from projective to affine that indicated in figure 6. The control signals are used to obtain the different operation for PM, where there is a signal to change the design from the group operation of PA and PD to the operation of conversion from projective to affine, also there are other signals to control the different operations inside PM.
Figure 4. Scalar multiplication architecture

Figure 5. Point Addition and Doubling architecture

Figure 6. Hardware architecture for conversion from projective to affine.

Tables 2, 3 and 4 indicate results for the implementation in virtex6 as illustrated in figure 5 and 6. All implementations are designed using the VHDL language. The implementation using a different size for polynomial multiplier indicated that there is a tradeoff between area and time. The cost of PM multiplication depends on the cost of GF($2^m$) multiplier unit, so the time improvement for the result in table 2 can be obtained by increasing the digit size (DG) for the GF($2^m$) multiplier unit.
Table 2. PM results for in Virtex-6 XC6VLX760.

| Parallel Montgomery PM based on LD projective coordinate | Digit size (DG) for Pipelined digit-serial multiplier | Frequency MHz | #occupied Slices | Time μs |
|----------------------------------------------------------|------------------------------------------------------|---------------|------------------|---------|
| for GF($2^{409}$)                                        | DG=24                                                | 317.140       | 12,630           | 59.7    |
| for GF($2^{409}$)                                        | DG=51                                                | 253.770       | 18,807           | 29      |
| for GF($2^{163}$)                                        | DG=5                                                 | 201.279       | 1,870            | 27      |
| for GF($2^{163}$)                                        | DG=41                                                | 370.001       | 7,734            | 3.930   |
| for GF($2^{163}$)                                        | DG=82                                                | 370.130       | 11,076           | 2.983   |

Table 3. Result for the Galois field operation($2^{409}$) (Virtex-6 XC6VLX760).

| GF algorithm | Digit Size | Frequency MHz | #occupied Slices | Time Ns |
|--------------|------------|---------------|------------------|---------|
| Bit parallel & Pipelined digit serial polynomial multiplier | DG=409 | ... | 17,560 | 5 |
| DG=204 | 276.095MHz | 9,907 | 7.5 |
| DG=102 | 281.597MHz | 4,813 | 15 ns |
| DG=51 | 310.915 | 2,880 | 32.5 |
| DG=24 | 275.057 | 1,475 | 65ns |
| Squaring | DG=409 | 0 | 0 | 0.345ns |
| Reduction | With word size =32 | 0 | 187 | 0.977ns |
| Itoh-Tsuji | DG=51 | 296.845MHz | 5,657 | 504 |

Table 4. Result for the Galois field operation($2^{163}$) (Virtex-6 XC6VLX760).

| GF algorithm | Digit Size | Frequency MHz | #occupied Slices | Time Ns |
|--------------|------------|---------------|------------------|---------|
| Bit parallel & Pipelined digit serial polynomial multiplier | G=163 | ... | 4,123 | 4.69 |
| G=41 | 408.69MHz | 1,497 | 9.78 ns |
| G=82 | 380.12MHz | 2,718 | 5.26ns |
| Squaring | G=163 | 0 | 0 | 0.345ns |
| Reduction | With word size =32 | 0 | 146 | 1.267ns |
| Itoh-Tsuji | Bit parallel multiplier | 458.6 | 4,602 | 315 ns |

6. Conclusion and future work
The parallel implementation for PM depends on the selection of the PM algorithm. The Montgomery ladder algorithm has an attractive result if it is designed to make parallel operation between PA and PD. Latency result of performing the inversion operation is very slow in comparison with other GF($2^m$) operation. The PM Implementation that is based on LD projective coordinate eliminates the
use of inversion operation. The binary implementation is more suitable when the time is needed rather than the prime implementation which was introduced in the previous work.

The challenge in the next steps is to apply the principle of ECCPs in the application like the security of the smart card or the security of the medical information system or the security of the data transfer over the internet.

References

[1] Kumari S, Karuppiha M, Das A K, Li X, Wu F and Kumar N 2017 A secure authentication scheme based on elliptic curve cryptography for IoT and cloud servers. Supercomput. J. Springer. 74 6428–53.

[2] Lee C-I and Chien H-Y 2015 An Elliptic Curve Cryptography-Based RFID Authentication Securing E-Health System. International Journal of Distributed Sensor Networks. 2015.

[3] Gebotys C H 2010 Security in Embedded Devices ( New York: Springer).

[4] Institute of Electrical and Electronics Engineers 2000 Standard Specifications for public Key Cryptography (IEEE P1363: IEEE).

[5] National Institute of Standards and Technology (NIST) 2009 Digital Signature Standard (DSS) (NIST: NIST FIPS 186–4).

[6] Realpe P C and Velasco-medina J 2015 High-performance elliptic curve cryptoprocessors over GF(2m) on koblitz curve. Analog Integr Circ Sig Process. Springer. 85 129–138.

[7] Loi K C C and Ko S-B 2016 Parallelization of scalable elliptic curve cryptoprocessors in GF(2m). Microprocessors and Microsystems. Elsevier. 45 10–22.

[8] Hankerson D, Menezes A and Vanstone S 2004 Guide to Elliptic Curve Cryptography (New York: Springer-Verlag).

[9] Rashidi B, Farashahi R R and Sayed S M 2014 High-speed and pipelined finite field bit-parallel multiplier over GF(2m) for elliptic curve cryptosystems. 2014 11th Int ISC Conf on Information Security and Cryptology. Tehran. 15-20.

[10] Rashidi B, Farashahi R R and Sayedi S M 2017 High-performance and high-speed implementation of polynomial basis Itoh–Tsujii inversion algorithm over GF(2m). IET Information Security. 11-2 66–77.

[11] Ismail M N 2012 Towards Efficient Hardware Implementation of Elliptic and Hyperelliptic Curve Cryptography (Dept. of Electrical and Computer Engineering. University of Waterloo. Ontario. Canada: Ph.D. dissertation).

[12] Rashidi B, Farashahi R R and Sayedi S M 2016 High-speed hardware architecture of scalar multiplication for binary elliptic curve cryptosystems. Microelectronics J. Elsevier. 52 49–65.

[13] Shohdy S, El-Sisi A and Ismail N. 2009 FPGA Implementation of elliptic curve point multiplication over GF(2193). in Advances in Information Security and Its Applications (ISA). Lecture Notes in Computer Science. Springer-Verlag Berlin. Germany. 5576 619–634.

[14] Khan Z U A and Benaissa M 2017 High-Speed and Low-Latency ECC Processor Implementation Over GF(2m) on FPGA. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. Jan., 25-1 165-176.

[15] Imran M, Rashid M, Jafari A R and Najam ul Islam M 2018 ACryp-Proc: Flexible Asymmetric Crypto Processor for Point Multiplication. IEEE Access. 6 22778–793.

[16] Kudithi T, Sakthivel R 2019 High-performance ECC processor architecture design for IoT security applications. Supercomputer J. Springer. 75 447–474.

[17] Sutter G D, Deschamps J-P and Imaña J L 2013 Efficient Elliptic Curve Point Multiplication Using Digit-Serial Binary Field Operations. IEEE TRANS ON INDUSTRIAL ELECTRONICS. Jan. 60-1 217-225.

[18] Li L and Li S 2016 High-Performance Pipelined Architecture of Elliptic Curve Scalar Multiplication Over GF(2m). IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 24-4 1223-32.