Modular Divider for Elliptic Curve Cryptographic Hardware Based on Programmable CA*

Jun-Cheol Jeon, Kee-Won Kim, Jai-Boo Oh, and Kee-Young Yoo**

Department of Computer Engineeing, Kyungpook National University,
Daegu, 702-701 Korea
{jcjeon33, nirvana, jboh}@infosec.knu.ac.kr,
yook@knu.ac.kr

Abstract. This study presents an efficient division architecture using irreducible trinomial in GF(2^n), based on programmable cellular automata (PCA). The most expensive arithmetic operation in elliptic curve cryptosystems (ECC) is division, which is performed by multiplying the inverse of a multiplicand. The proposed architecture is highly regular, expandable, and has reduced latency. The proposed architecture can be efficiently used in the hardware design of crypto-coprocessors.

1 Introduction

Finite field GF(2^n) arithmetic operations have recently been applied to a variety of fields, including cryptography and error-correcting codes [1]. A number of modern public key cryptography systems and schemes, for example, Diffie-Hellman key pre-distribution, ElGamal cryptosystem, and ECC, require division and inversion operations [2].

The main operation of ECC is the inverse/division operation, which can be regarded as a special case of exponentiation [3]. Since division, however, is quite time consuming, efficient algorithms are required for practical applications. Division operations can generally be classified into two approaches: a fast architecture design or a novel algorithm development. This current study focuses on the former approach.

Cellular automata have been used in evolutionary computations for over a decade. They have been used in a variety of applications, such as parallel processing and number theory. CA architecture has been used in the design of arithmetic computations that Zhang [4] proposed architecture with programmable cellular automata, Choudhury [5] designed an LSB multiplier based on CA, and Jeon [6] proposed simple and efficient architecture based on periodic boundary CA.

This paper proposes efficient hardware architecture for division based on PCA. We focused on the architecture in ECC, which uses restricted irreducible polynomials, especially, trinomials. The structure has a time complexity of $n(n\text{-}1)(T\__{AND} + T\__{XOR} + T\__{MUX})$ and a hardware complexity of (nAND+(n+2)XOR+ nMUX+ 4nREGISTER). In addition, our architecture can easily be expanded for other public key cryptosystems with additional (n-2)XOR gates. Our architecture focuses on both area and time complexities.

* This work was supported by the Brain Korea 21 Project in 2006.
** Corresponding author.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part IV, LNCS 3994, pp. 661 – 668, 2006.
© Springer-Verlag Berlin Heidelberg 2006

The rest of this paper is organized as follows: The theoretical background, including finite fields, ECC, and CA, is described in Section 2. Section 3 presents the proposed division architecture based on PCA, and we present our discussion, together with a comparison of the performances between the proposed architecture and previous research, in Section 4. Finally, the conclusion is presented in Section 5.

2 Preliminary

In this section, we discuss the mathematical background in the finite field and ECC, and the characteristics and properties of cellular automata.

2.1 Finite Fields

A finite field or Galois Field (GF), which is a set of finite elements, can be defined by commutative law, associative law, distributive law and it contains facilitates for addition, subtraction, multiplication, and division. A number of architectures have already been developed to construct low complexity bit-serial and bit-parallel structures by using various irreducible polynomials to reduce the complexity. Since a polynomial basis operation has regularity and simplicity, the ability to design and expand it into high-order finite fields, with a polynomial basis, is easier to realize than with other basis operations [7].

A finite field can be viewed as a vector space of dimensions n over GF(2^n). That is, there exists a set of n elements $\{1, \alpha, \ldots, \alpha^{n-2}, \alpha^{n-1}\}$ in GF(2^n) such that each $A \in$ GF(2^n) can be written uniquely in the form $A = \sum A_i \alpha^i$, where $A_i \in \{0,1\}$. This section provides one of the most common bases of GF(2^n) over GF(2), which are polynomial bases [7, 8]. Let $f(x) = x^n + \sum_{i=0}^{n-1} f_i x^i$, where $f_i \in \{0,1\}$, for $i = 0, 1, \ldots, n-1$, be an irreducible polynomial of degree n over GF(2). For each irreducible polynomial, there exists a polynomial basis representation. In such a representation, each element of GF(2^n) corresponds to a binary polynomial of less than n. That is, for $A \in$ GF(2^n) there exist n numbers $A_i \in \{0,1\}$ such that $A = A_{n-1}\alpha^{n-1} + \ldots + A_1\alpha + A_0$. In many applications, such as cryptography and digital communication applications, the polynomial basis is still the most widely employed criterion [9-11]. In the following, we confine our attention to computations that use the polynomial basis.

2.2 Elliptic Curve Cryptosystem

In ECC, computing kP is the most important arithmetic operation, where k is an integer and P is a point on the elliptic curve. This operation can be computed by the addition of two points k times. ECC can be done with at least two types of arithmetic, each of which gives different definitions of multiplication [12]. Two types of arithmetic are, namely, $\mathbf{Z}_p$ arithmetic (modular arithmetic with a large prime p as the modulus) and GF(2^n) arithmetic, which can be done with shifts and exclusive-ors. This can be thought of as the modular arithmetic of polynomials with coefficients mod 2.

We focused on GF(2^n) arithmetic operation. Let GF(2^n) be a finite field by definition. Then, the set of all solutions for equation E: $y^2 + xy = x^3 + a_2x^2 + a_6$, where a_2, $a_6 \in$ GF(2^n), $a_6 \neq 0$, together with special point called the point at infinity $\boldsymbol{O}$, is a non-supersingular curve over GF(2^n). Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be points in E(GF(2^n)) given in affine coordinates [13]. Assume that P_1 , $P_2 \neq \boldsymbol{O}$, and $P_1 \neq -P_2$. The sum $P_3 = (x_3, y_3) = P_1 + P_2$ is computed as follows; if $P_1 \neq P_2$ then $\lambda = (y_1 + y_2)/(x_1 + x_2)$, $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a_2$, $y_3 = (x_1 + x_3)\lambda + x_3 + y_1$, and if $P_1 = P_2$ (called point doubling), then $\lambda = y_1 / x_1 + x_1$, $x_3 = \lambda^2 + \lambda + a_2$, $y_3 = (x_1 + x_3)\lambda + x_3 + y_1$.

In either case, the computation requires one division, one squaring, and one multiplication. Squaring can be substituted by multiplication. From the point addition formula, it should be noted that no computation, except for addition, is performed at the same time due to data dependency. Therefore, sharing hardware between division and multiplication is more desirable than the separated implementation of division and multiplication [3, 8]

The additive inverse and multiplicative inverses in GF(2^n) can be calculated efficiently using the extended Euclidean algorithm. Division and subtraction are defined in terms of additive and multiplicative inverses: A-B is A+(-B) in GF(2^n) and A/B is $A \cdot (B^{-1})$ in GF(2^n). Here, the characteristic 2 finite fields GF(2^n) used should have $n \in$ {113, 131, 163, 193, 233, 239, 283, 409, 571} [3]. This restriction is designed to facilitate interoperability while enabling implementers to deploy efficient implementations that are capable of meeting common security requirements [13].

The rule that is used to pick acceptable reduction polynomials is the following: if a degree n binary irreducible trinomial, $f(x) = x^n + x^k + 1$, for $n > k \geq 1$ exists, then, the irreducible trinomial with the smallest possible k should be used. These polynomials enable the efficient calculation of field operations.

2.3 Programmable Cellular Automata

A CA is a collection of simple cells arranged in a regular fashion. CAs can be characterized based on four properties: a cellular geometry, a neighborhood specification, number of states per cell, and a rule to compute to a successor state. The next state of a CA depends on the current state and rules [14]. Only 2-state and 3-neighborhood CAs are considered in this paper. Table 1 shows all possible states and rules. Each mapping is called a 'rule' of the CA.

The next state transition for the ith cell can be represented as a function of the present states of the ith, (i+1)th, and (i-1)th cells for a 3-neighberhood CA: $Q_i(t+1) = f(Q_{i-1}(t), Q_i(t), Q_{i+1}(t))$, where '$f$' represents the combinational logic function as a CA rule, which is implemented by a combinational logic circuit (CL), and $Q(t+1)$ denotes the next state for cell $Q(t)$.

Table 1 specifies the three particular sets of transition from a neighborhood configuration to the next state. CA can also be classified as linear or non-linear. If the neighborhood is only dependent on an XOR operation, the CA is linear, whereas if it is dependent on another operation, the CA is non-linear. If the neighborhood is only dependent on an EXOR or EXNOR operation, then the CA can also be referred to as an additive CA.

Table 1. State transition and functions according to different rules

Rules	Logical functions	State transition							
		111	110	101	100	011	010	001	000
90	$Q_{i-1}(t) \oplus Q_{i+1}(t)$	0	1	0	1	1	0	1	0
150	$Q_{i-1}(t) \oplus Q_i(t) \oplus Q_{i+1}(t)$	1	0	0	1	0	1	1	0
240	$Q_{i-1}(t)$	1	1	1	1	0	0	0	0

Furthermore, if the same rule applies to all cells in a CA, the CA is called a uniform or regular CA, whereas if different rules apply to different cells, it is called a hybrid CA. A CA can be divided into three patterns based on the boundary conditions: a null boundary CA, intermediate boundary CA, and periodic boundary CA (PBCA). For the remainder of this paper, a CA will be regarded as a PBCA, unless otherwise mentioned. A PBCA regards the leftmost and rightmost cells as neighbors.

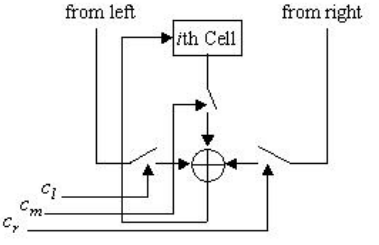

Fig. 1. A 3-neighborhood linear PCA cell structure

Table 2. The PCA corresponding rules according to the control signals

Control signals			Corresponding Rules	
C_l	C_m	C_r		
0	0	1	Q_r	170
0	1	0	Q_m	202
0	1	1	$Q_m \oplus Q_r$	102
1	0	0	Q_l	240
1	0	1	$Q_l \oplus Q_r$	90
1	1	0	$Q_l \oplus Q_m$	60
1	1	1	$Q_l \oplus Q_m \oplus Q_r$	150

A programmable CA (PCA) is a CA whose CL is not fixed for each cell, but is controlled by a number of signals such that different functions (rules) can be generated. Fig. 1 shows a 3-neighberhood linear PCA cell structure. A combination of control signals decides the rule of each PCA cell. The value of the control signals and corresponding rules of the PCA are presented in Table 2. If the '90' rule presented in Table 2 is used for the renewal of the next cell's value, the control signals of the C_l

and C_r values are '1' and C_m has a value of '0'. Also, if the '150' rule is used, the C_l, C_m, and C_r, all possess '1'values. We use the '170' rule which only depends on the right cell.

3 Division Architecture Based on PCA

This section presents $A(x)/B(x)$ architecture based on PCA. Finite field division in GF(2^n) can be performed using multiplication and inverse; that is, $A(x)/B(x) = A(x)B(x)^{-1}$. The division can be implemented efficiently by repeatedly applying $A(x)B(x)^2$ multiplications. Let us suppose that $A(x)$ and $B(x)$ are the elements on GF(2^n). Then, the two polynomials $A(x)$, $B(x)$ are as follows:

$$A(x) = A_{n-1}x^{n-1} + \ldots + A_1x^1 + A_0,\ B(x) = B_{n-1}x^{n-1} + \ldots + B_1x^1 + B_0$$

From the above equation, we have

$$B(x)^2 = B_{n-1}x^{2n-2} + B_{n-2}x^{2n-4} + \ldots + B_1x^2 + B_0$$

Then, $A(x)B(x)^2 \bmod T(x)$ can be induced from the two equations. The definitive algorithm for implementation is as follows:

$$\{\ldots[A(x)B_{n-1}x^2 \bmod T(x) + A(x)B_{n-2}]\ x^2 \bmod T(x) + \ldots + A(x)B_1\}\ x^2 \bmod T(x) + A(x)B_0$$

[Algorithm 1] $A(x)B(x)^2$ multiplication algorithm

Input : $A(x)$, $B(x)$, $T(x)$
Output : $A(x)B(x)^2 \bmod T(x)$

Step 1 : $M(x) = 0$
Step 2 : for $i = n$-1 to 0
Step 3 : $M(x) = M(x)\ x^2 \bmod T(x) + A(x)B_i$

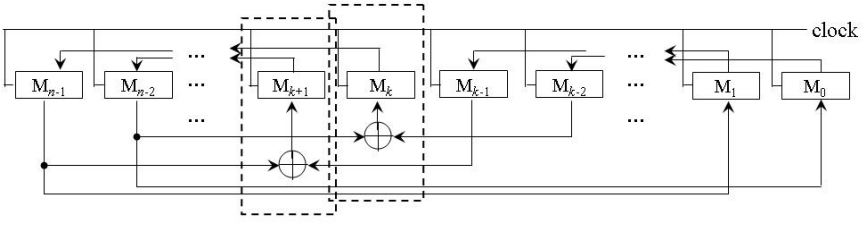

Fig. 2. Multiplication architecture using irreducible trinomials in Step 3 of Algorithm 1

Let $M(x)x^2 \bmod T(x)$ be $M_{n-1}x^{n-1} + \ldots + M_1x^1 + M_0$, where $T(x) = x^n + x^k + 1$. Then the following equation holds: $M(x)x^2 \bmod T(x) = M_{n-3}x^{n-1} + \ldots (M_{n-1} \oplus M_{k-1})x^{k+1} + (M_{n-2} \oplus M_{k-2})x^k + \ldots + M_{n-1}x^1 + M_{n-2}$. The equation is illustrated based on two PCA cell structures where C_l, $C_r = 1$ and $C_m = 0$.

Here, the $A(x)B(x)^2$ operation can be used as an efficient method in division algorithms as follows [7]:

[Algorithm 2] $A(x)/B(x)$ Division algorithm

Input : $A(x)$, $B(x)$, $T(x)$
Output : $D(x) = A(x)/B(x) \bmod T(x)$

Step 1 : $D(x) = B(x)$
Step 2 : for $i = n$-2 to 1
Step 3 : $D(x) = B(x)D(x)^2 \bmod T(x)$
Step 4 : $D(x) = A(x)D(x)^2 \bmod T(x)$

The result is $D(x) = A(x)B(x)^{-1}$ and when $A(x) = 1$, the algorithm realizes the inverse operation $B(x)^{-1}$. In this case, the $A(x)B(x)^2$ operation can be used to compute the operations in Step 3 and 4. Fig. 3 shows the proposed architecture for division. Each initial value is such that cellular automata have all zeros, and that the B register and Shift register have $B(x)$ values.

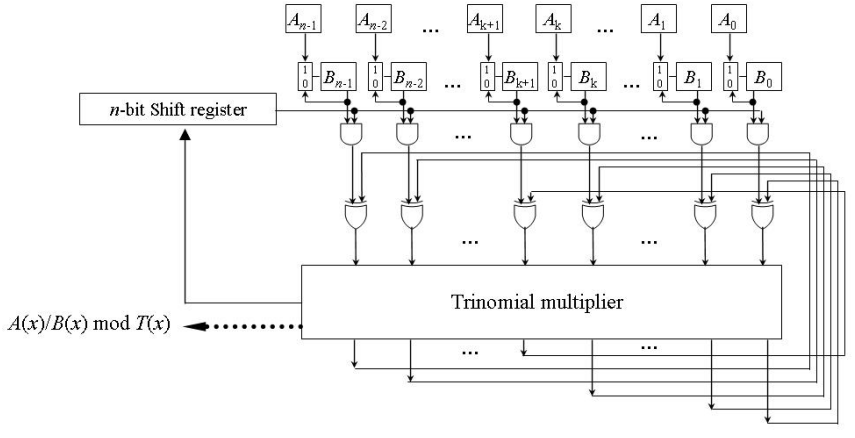

Fig. 3. Division architecture based on the trinomial multiplier in Fig. 2

After the $A(x)B(x)^2$ operation in Fig.2, the computed values transfer to the Shift register. $B(x)^{-1}$ is computed after the above mentioned process n-2 times. After the process, the system chooses $A(x)$ instead of $B(x)$ in the upper registers by Muxes for the final resultant values. It is possible to perform $A(x)/B(x) \bmod T(x)$ in $n(n\text{-}1)$ clock cycles by using n AND gates, n+2 XOR gates, n Muxes, and $4n$ bits registers, plus extra equipment such as control signals for the transfer of results in cellular automata to the shift register, and in order to change the values in the B register, immediately after deriving the values of $B(x)^{-1}$.

Moreover, our architecture can be easily applied to other public key cryptosystems by using general irreducible polynomials. In Fig. 3, by using additional n-2 XOR gates, the proposed architecture can perform a general division operation. Although the architecture is used as a general divider, it has the same latency as of that in Fig. 3 because of the parallel property.

4 Comparison and Analysis

A comparison of the proposed division architecture, with existing structures was performed, focusing on time and hardware complexity issues. As such, Wang's [10] and Kim's [11] division architecture was chosen.

Wang proposed parallel-in parallel-out division architecture, which has a latency of $n(2n\text{-}1.5)$ and a critical path of $(T_{\_AND} + 3T_{\_XOR})$ over GF(2^n). Kim proposed a serial-in serial-out divider, which has a latency of $2n(n\text{-}1)$ and critical path of a $(2T_{\_AND}+3T_{\_XOR} + T_{\_MUX})$. However, our serial-in parallel-out architecture has only a latency of $n(n\text{-}1)$ and a critical path of $(T_{\_AND}+T_{\_XOR} +T_{\_MUX})$ over GF(2^n).

We usually try to find the design that will best satisfy a given set of requirements when we implement the arithmetic unit design. We consider construction simplicity, which is defined by the number of transistors that are needed for its construction and the time that is needed for the signal change to be propagated through the gates [15]. Table 3 shows a comparison of area-time products.

Table 3. A comparison of area-time products among dividers

Circuits	Area-time product
Proposed architecture	$128n^3(n\text{-}1)(30n\text{-}22.5)$
Wang et al. [15]	$92n(105n^2 \text{-}176n+69)(n\text{-}1)$
Kim et al.[16]	$336n(n\text{-}1)(4n+1)$

As in Table 3, our architecture has less complexity than the other architectures. In particular, the proposed architecture has $\boldsymbol{O}(n^3)$ area-time complexity whereas the other architecture have $\boldsymbol{O}(n^4)$ and $\boldsymbol{O}(n^5)$ complexity, respectively. We have shown that our architecture has less complexity than the serial or parallel architectures with regard to area and time. Our architecture only focuses on ECC, which is restricted by using irreducible trinomials. Our architecture, however, can be easily applied to other public cryptosystems with additional n-2 XOR gates, while existing systolic architectures including those of Wang's and Kim's, hardly reduce the level of complexity, although they apply irreducible trinomials for ECC. Moreover, our architecture does not influence latency after it has been applied to a general divider.

5 Conclusion

This paper has presented efficient hardware architecture in order to compute the $A(x)/B(x)$ modulo irreducible trinomials, which are restricted in the Certicom Standard for ECC. We have proposed a simple hardware architecture that is the most expensive

arithmetic operation scheme, such as inverse and division in ECC over GF(2^n). The proposed architecture includes the characteristics of both PCA and irreducible trinomials, and it has minimized both time and hardware complexity. Moreover, our architecture can be easily applied to a general division architecture with no additional latency needed. Therefore, we have shown that our architecture has outstanding advantages, as compared to typical structures.

References

1. T. R. N. Rao and E. Fujiwara, Error-Control Coding for Computer Systems, Englewood Cliffs, NJ: Prentice-Hall (1989)
2. W. Drescher, K. Bachmann, and G. Fettweis, "VLSI Architecture for Non Sequential Inversion over GF(2^m) using the Euclidean Algorithm," The International Conference on Signal Processing Applications and Technology, Vol. 2. (1997) 1815-1819
3. A.J.Menezes, Elliptic Curve Public Key Cryptosystems, Boston, MA: Kluwer Academic Publishers (1993)
4. C. N. Zhang, M. Y. Deng, and R. Mason, "A VLSI Programmable Cellular Automata Array for Multiplication in GF(2^n)," PDPTA '99 International Conference (1999)
5. P. Pal. Choudhury and R. Barua, "Cellular Automata Based VLSI Architecture for Computing Multiplication and Inverses in GF(2^m)," IEEE 7th International Conference on VLSI Design (1994) 279-282
6. Jun-Cheol Jeon and Kee-Young Yoo, "An Evolutionary Approach to the Design of Cellular Automata Architecture for Multiplication in Elliptic Curve Cryptography over Finite Fields," Lecture Notes in Artificial Intelligence PRICAI 2004: Trends in Artificial Intelligence (LNAI 3157), Springer-Verlag, Vol. 3157. (2004) 241-250
7. A. J. Menezs, Applications of Finite Fields, Boston, MA: Kluwer Academic Publishers (1993)
8. IEEE P1363, Standard Specifications for Public Key Cryptography (2000)
9. S. W. Wei, "VLSI architecture of divider for finite field GF(2^m)", IEEE International Symposium on Circuit and Systems, Vol. 2. (1998) 482-485
10. C. L. Wang and J. H. Guo, "New Systolic Arrays for C+ AB^2, inversion, and division in GF(2^m)", IEEE Trans. on Computer, Vol. 49, No. 10. (2000) 1120-1125
11. N. Y. Kim and K. Y. Yoo, "Systolic architecture for inversion/division using AB^2 circuits in GF(2^m)", Integration, the VLSI Journal, Vol. 35. (2003) 11-24
12. C. Kaufman, R. Perlman, and M. Speciner, Network Security private communication in a public world, New Jersey: Prentice Hall (2002)
13. SEC 1: Elliptic Curve Cryptography version 1.0, Certicom Reserch (2000)
14. O. Lafe, Cellular Automata Transforms: Theory and Applications in Multimedia Compression, Encryption, and Modeling, Kluwer Academic Publishers (2000).
15. D. D. Gajski, Principles of digital design : Prentice-Hall International Inc. (1997)