Fast Exponentiaion over GF(2^m) Based on Cellular Automata

Kyo-Min Ku [1], Kyeoung-Ju Ha [2], and Kee-Young Yoo [3]

[1] Mobilab Co.Ltd, 952-3, 4F Plus Bldg., DongChun-Dong, Buk-Gu, Daegu, Korea, 702-250
kmku@nate.com
[2] KyungSan University, 75 San, JumChon-Dong, Kyungsan, KyungPook, Korea, 712-715
kjha@kyungsan.ac.kr
[3] KyungPook National University, 1370 Sankyuk-Dong, Puk-Gu, Daegu, Korea 702-701
yook@knu.ac.kr

Abstract. In this paper, we present a new exponentiation architecture and multiplier/squarer which are the basic operations for exponentiation on GF(2^m). The proposed multiplier/squarer is used as kernal architecture of exponentiation and processes the modular multiplication and squaring at the same time for effective exponentiation on GF(2^m) using a cellular automata. Proposed architecture can be used efficiently for the design of the modular exponentiation on the finite field in most public key crypto systems such as Diffie-Hellman key exchange, ElGamal, etc. Also, the cellular automata architecture is simple, regular, modular, cascadable and therefore, can be utilized efficiently for the implementation of VLSI.

1 Introduction

For the past 30 years, studies on finite fields have been conducted in many areas, including crypto systems[1], and most public key crypto systems, such as Diffie-Hellman key exchange and ElGamal, are based on modular exponentiation computations in a finite field[2][3]. Such modular exponentiation uses a modular multiplier as the basic structure for its implementation. The Elliptic Curve Cryptosystem is also based on constant multiplication[4]. Examples of the algorithms used to implement multipliers include the LSB-first multiplication algorithm[5], MSB-first multiplication algorithm[6], and Montgomery algorithm[7]. Previous research and development on modular multiplication is as follows: First, for a one-dimensional systolic array, in the case of an LSB-first algorithm, the modular multiplication is performed within 3m clock cycles using m cells[5]. While in the case of an MSB-first algorithm, the modular multiplication can be performed within 3m clock cycles using m cells[6]. With an LFSR structure, the modular multiplication can be performed within 2m clock cycles using m cells[10], the modular multiplication can be performed within m clock cycles using m cells and the modular squaring can be performed within m clock cycles using m cells[11]. The structures proposed in [5, 6, 10, 12] are simple modular multipliers.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2658, pp. 841–850, 2003.

However, when computing exponentiation, such structures must be repeated twice for modular multiplication and squaring. In case of [11], the structures of multiplication and squaring must be used together to simultaneously perform the modular multiplication and squaring.

The purpose of the current paper is to reduce the time and the space, and to investigate and develop a simple, regular, modular, and cascadable architecture for the VLSI implementation of exponentiation in GF(2^m) based on cellular automata, which is the basic computation in any public key crypto system.

Accordingly, this paper proposes a fast exponentiation architecture over GF(2^m) based on a cellular automata. The proposed architecture uses the basic architecture that can simultaneously perform multiplication and squaring in m clock cycles using $3m$-1 AND gates, $3m$-1 XOR gates, and $4m$-1 registers. Based on the properties of LSB-first multiplication, the parts of modular multiplication and squaring that can be performed in common are identified, then the remainder is processed in parallel. As a result, the multiplication and squaring can be performed much more efficiently as regards time and space compared to repeating the structure as proposed in [5, 6, 10] and can be performed much more efficiently as regards space compared to repeating the structure as proposed in [11,14]. Furthermore, the performance of the exponentiation is much more efficient than that of the [16] as regards time and space.

The remainder of the paper is as follows: Chapter 2 gives an overview of the concept of cellular automata, while Chapter 3 reviews the general exponentiation algorithm in GF(2^m). Chapter 4 introduces the structure of the proposed multiplier/squarer for efficient exponentiation using a cellular automata. Chapter 5 gives the exponentiation architecture over GF(2^m). Finally, Chapter 6 offers some conclusions.

2 Cellular Automata(CA)

Cellular automata consist of numbers of interconnected cells arranged spatially in a regular manner[8][9]. A cell of the CA has the "0" state or "1" state at a certain time. The next state of a cell depends on the present states of 'k' of its neighbors, for a k-neighborhood CA. The neighbor in CA means the cell which affect the state of cell update. Fig.1 is the example of the two-way CA.

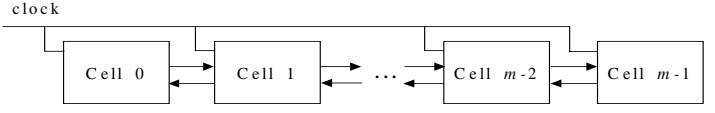

Fig. 1. m-cell two-way CA

The m-cell of the two-way CA is operated at the same time by a clock. The state of a cell at time t is determined by the states of the neighbors at time t-1. In this paper, we assume that the leftmost cell and rightmost cell are adjacent.

We use the modified CA which is the one-way CA(OCA) to solve the problem. Fig.2 shows the OCA structure. It is the same as two-way CA in Fig.1 except that the data stream is one-way.

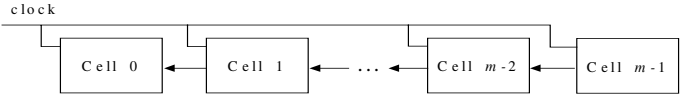

Fig. 2. *m*-cell OCA

The characteristic matrix shows the entire rules of the CA. An example of the characteristic matrix with 4-cell of which state is renewed by the state of its right adjacent cell is as follows:

$$D=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{bmatrix}$$

In the above example, it is shown that element "1" of the matrix on the *i*th line of the *j*th row shows that the *i*th cell is dependent on the neighbor of the *j*th cell.

3 General Algorithm for the Exponentiation over GF(2^m)

In this chapter, general algorithms for obtaining $M(x)^E$ mod $P(x)$ on GF(2^m) are illustrated[12].

Polynomial $P(x)$ of arbitrary degree with coefficients from GF(2) is called an irreducible polynomial if $P(x)$ is not divisible by any polynomial over GF(2) of degree greater than 0 but less than the degree of $P(x)$ [1]. Let $P(x)=x^m+p_{m-1}x^{m-1}+\ \dots\ +p_1x^1+p_0$ be an irreducible polynomial over GF(2) and α be a root of $P(x)$.

Let's suppose that $A(x)$ and $B(x)$ are the elements on GF(2^m). Then two polynomials $A(x)$, $B(x)$ are as follows:

$$A(x)=a_{m-1}x^{m-1}+\ \dots\ +a_1x^1+a_0 \quad (1)$$
$$B(x)=b_{m-1}x^{m-1}+\ \dots\ +b_1x^1+b_0 \quad (2)$$

Firstly, computation of $M(x)^E$ mod $P(x)$ is divided into the LSB-first method and MSB-first method according to the method of processing of the exponent E, [e_{m-1}, e_{m-2}, …,e_1, e_0], where the method of computation is as follows:

LSB-first exponentiation: Compute in the order of from e_0 to e_{m-1}

$$M(x)^E = M(x)^{e_0}\,(M(x)^2)^{e_1}\,(M(x)^4)^{e_2}\,...(M(x)^{2^{m-1}})^{e_{m-1}}$$

MSB-first exponentiation: Compute in the order of from e_{m-1} to e_0

$$M(x)^E = (..((M(x)^{e_{m-1}})^2 M(x)^{e_{m-2}})^2 ...M(x)^{e_1})^2 M(x)^{e_0}$$

In this chapter, general algorithm for the method of LSB-first exponentiation is reviewed [13], the structure of a multiplier using the cellular automata that can perform its computation efficiently is proposed in the next chapter.

Algorithm 1 : LSB-first Exponentiation Algorithm[13]
Input : $A(x)$, E, $P(x)$
Output : $C(x)=A(x)^E \bmod P(x)$
STEP 1 : $C(x)=\alpha^0$, $T(x)=A(x)$
STEP 2 : for i=0 to m-1
STEP 3 : if e_i ==1 $C(x)= T(x)C(x) \bmod P(x)$
STEP 4 : $T(x)= T(x)T(x) \bmod P(x)$

The general method for implementation of Algorithm 1 is to design an exponentiation architecture by using two multipliers, or to use one multiplier and one squarer. However, in the next chapter, an efficient multiplier/squarer is designed by identifying the commonly computed part of two operations (modular multiplication and squaring) perform it at the same time, and processing the remaining operation in parallel. Proposed structure can perform exponentiation efficiently.

4 Multiplier/Squarer Using the Cellular Automata

In this chapter, we show the architecture that simultaneously process the modular multiplication and squaring over GF(2^m) in m clock cycles using a cellular automata[15].

According to Alg. 1 which is proposed in Chapter 3, in order to compute the LSB-first exponentiation, it is necessary to compute $M(x)= T(x)C(x)$ which is the modular multiplication used in Step 4 and $S(x)= T(x)T(x)$which is squaring used in Step 3.

These two computations can be expressed in a recurrence form again[15]. First, the recurrence form of the modular multiplication is as follows:

$$M^{(i)}(x)=M^{(i-1)}(x)+c_{i-1}T^{(i-1)}(x),\ T^{(i)}(x)=T^{(i-1)}(x)x \bmod P(x),\ \text{for } 1\le i\le m \qquad (3)$$

where $T^{(0)}(x)=T(x)$, $M^{(0)}(x)=0$, $M^{(i)}(x)=c_0T(x)+c_1[T(x)\ x \bmod P(x)] + c_2[T(x)\ x^2 \bmod P(x)] +\cdots+ c_{i-1}[T(x)x^{i-1} \bmod P(x)]$. For $i=m$, $M^{(m)}(x)=M(x)=T(x)C(x) \bmod P(x)$. In equation 3, two equations can be performed in parallel. And second, squaring can be also converted into the LSB-first recurrence form similar to equation 3 as follows:

$$S^{(i)}(x)=S^{(i-1)}(x)+t_{i-1}T^{(i-1)}(x),\ T^{(i)}(x)=T^{(i-1)}(x)x \bmod P(x),\ \text{for } 1\le i\le m \qquad (4)$$

where $T^{(0)}(x)=T(x)$, $S^{(0)}(x)=0$, $S^{(i)}(x)=t_0T(x)+t_1[T(x)\ x \bmod P(x)] + t_2[T(x)\ x^2 \bmod P(x)] + \ldots+ t_{i-1}[T(x)\ x^{i-1} \bmod P(x)]$. For $i=m$, $S^{(m)}(x)=S(x)=T(x)T(x) \bmod P(x)$. In equation 4, two equations can be performed in parallel.

The following bit-wise LSB-first algorithm both modular multiplication and squaring simultaneously can be derived from the above equation 3 and 4 :

Algorithm 2: MS($C(x)$, $T(x)$, $P(x)$)
Multiplication and Squaring Algorithm

Input : $C(x)$, $T(x)$, $P(x)$

Output : $M(x)=C(x)T(x)\ mod\ P(x)$, $S(x)=T(x)T(x)\ mod\ P(x)$

Step1 : $M^{(0)}(x)=0$, $T^{(0)}(x)=T(x)$, $S^{(0)}(x)=0$

Step2 : **for** $i=1$ **to** m

Step3 : $T^{(i)}(x)=T^{(i-1)}(x)x \bmod P(x)$

Step4 : $M^{(i)}(x)=M^{(i-1)}(x)+c_{m-i-1}T^{(i-1)}(x)$, $S^{(i)}(x)=S^{(i-1)}(x)+t_{m-i-1}T^{(i-1)}(x)$

Therefore, a structure in which the exponentiation can be computed efficiently on GF(2^m) by performing the modular multiplication and squaring simultaneously in the same amount of time as that of modular multiplication by computing $T^{(i)}(x)=T^{(i-1)}(x)x \bmod P(x)$ for $1\le i\le m$, which is the common part in equations 3 and 4, only once without duplicate computation, using the result, and obtaining the remaining part of equations 3 and 4 in parallel was proposed in [15].

The characteristic matrix of the CA to operate the step 3 fo Algorithm 2 is as follows:

$$D=\begin{bmatrix}0&1&0&0\\0&0&1&0\\0&0&0&1\\1&0&0&0\end{bmatrix}$$

And the CA structure is in Fig.3.

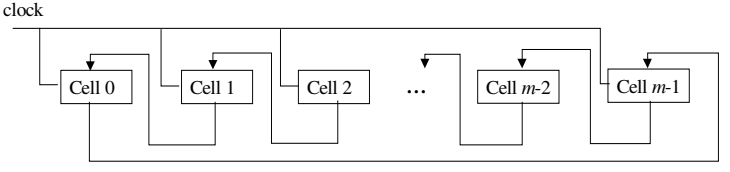

Fig. 3. OCA having the characteristic matrix D

The Fig.4 shows the entire suructure of common operation of multiplication and squaring. It accomplish the step 3 of Algorithm 2.

Now, how to perform the modular multiplication and squaring simultaneously is described. In order to perform step 4 of Algorithm 2, $M^{(i)}(x)=M^{(i-1)}(x)+c_{m-i-1}T^{(i-1)}(x)$ for $1\le i\le m$, the $c_{m-i-1}T^{(i-1)}(x)$ operation is reviewed firstly. For $c_{m-i-1}T^{(i-1)}(x)$ operation with

$1 \le i \le m$, the m bits obtained as a result of CA of Fig. 4 and c_{i-1} are inputted into m AND gates at i^{th} clock. The next result and $M^{(i-1)}(x)$ are subject to XOR, and the result is

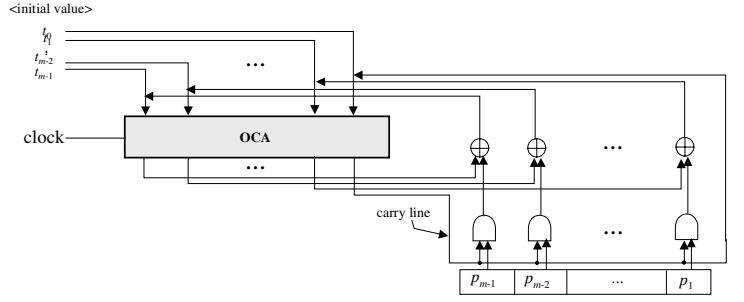

Fig. 4. Structure of $T^{(i-1)}(x)x \bmod P(x)$ operation

stored at $M^{(i-1)}(x)$ again. The value of each $M^{(0)}(x)$ register at the beginning is initialized to be 0. The structure for that is as shown in Fig. 5, which shows the computation in the ith clock for $1 \le i \le m$:

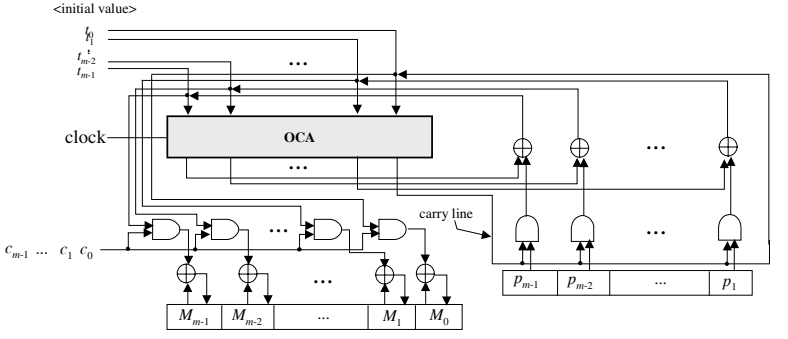

Fig. 5. Structure of modular multiplication

For squaring operation in step 4 of Algorithm 2, the $C(x)$, is substituted with $T(x)$. So the structure in which the modular multiplication and squaring can be performed simultaneously using CA, is shown in Fig. 6(at i^{th} clock). Each initial value is as follows:

- Initial values of OCA : $T(x) = t_{m-1} \ldots t_2 t_1 t_0$
- Initial values of the P register : $P(x) = p_{m-1} \ldots p_2 p_1 p_0$
- Initial values of the S, M register : all 0

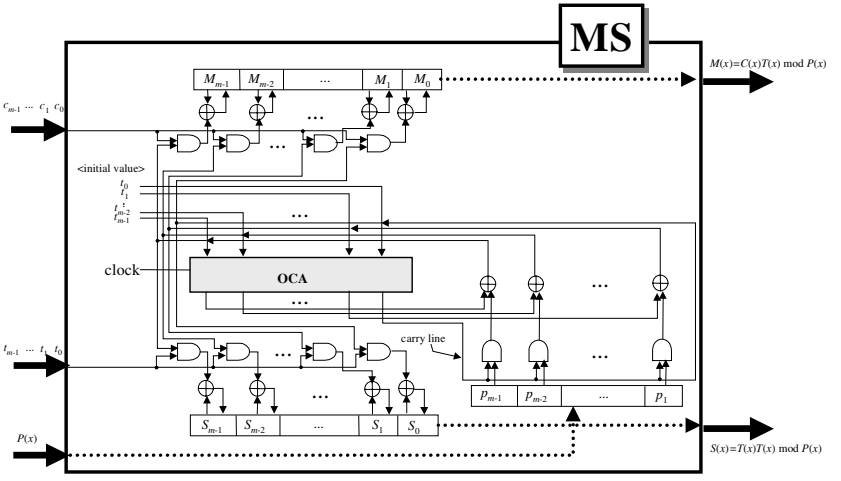

Fig. 6. Structure of simultaneously performing modular multiplication and squaring using CA

5 Exponentiation Architecture Using Multiplier/Squarer over GF(2^m)

Ordinary LSB-first exponentiation algorithm in [12] can be slightly changed to Algorithm 3. Exponentiation algorithm which uses a new multiplication/squaring algorithm, MS algorithms in chapter 4, as a sub function for exponentiation algorithm is as follows:

Algorithm 3 : EXP($A(x)$, E, $p(x)$)
Exponentiation Algorithm using MS Algorithm.
Input : $A(x)$, E, $P(x)$
Output : $C(x)=A(x)^E \bmod P(x)$
STEP 1 : $C(x)=\alpha^0$, $T(x)=A(x)$
STEP 2 : for i=0 to m-1
STEP 3 : if e_i ==1 $(C^{(i+1)}(x), T^{(i+1)}(x))$=MS$(C^{(i)}(x), T^{(i)}(x))$
else $C^{(i+1)}(x)=C^{(i)}(x)$, $T^{(i+1)}(x)$= MS$(C^{(i)}(x), T^{(i)}(x)$.

In step 3, MS algorithm is called with two parameters $C^{(i)}(x)$ and $T^{(i)}(x)$, and it returns two computation results of multiplication, $T^{(i)}(x)C^{(i)}(x)$ mod $P(x)$ and squaring $T^{(i)}(x)T^{(i)}(x)$ mod $p(x)$. Square result is stored to $T^{(i+1)}(x)$. But, the variable $C^{(i+1)}(x)$ stores a value depends on the value e_i.
If e_i has value 1 then returned multiplication result is passed to $C^{(i+1)}(x)$, but the other case, $C^{(i)}(x)$ is passed to $C^{(i+1)}(x)$.

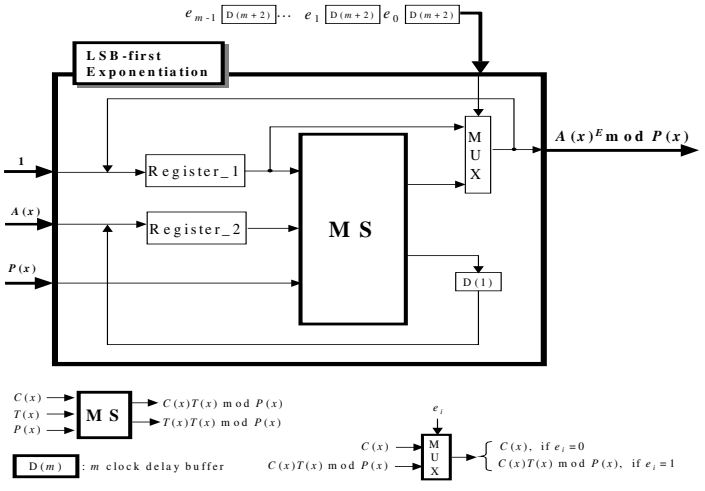

Fig. 7. A structure for performing an exponentiation using MS structure

Fig 7. represents a structure for performing an exponentiation using a new MS structure in Fig.6, which operates Algorithm 3. In Fig.7, the architecture includes a MS structure as its kernel architecture. This MS structure for multiplier/squarer is described in Fig. 6. While the multiplier/squarer computes the multiplication operation, it also processed the squaring operation concurrently forming the square term. The MUX depicted in Fig. 7 decides the multiplication result depends on the exponent.

6 Analysis and Conclusion

In this paper, we propsed a new fast exponentiation architecture over GF(2^m) using a cellular automata. The performance of the proposed archtecture is compared with that of previous study. Table 1 shows the result.

In conclusion, the proposed structure in this paper is much more efficient than systolic structure in view of the space and time. And we generally consider construction simplicity, defined by the number of transistors needed for its construction and the time needed for the signal change to propagate through the gate[17]. So the comparison of area-time product[17] for LSB first exponentiation is shown in Table 2. In Table 2, the proposed architecture in this paper is more efficient in view of the space and time than that of [16]. Even if when m=512, the proposed architecture get 24% speed up. In view of the space, the area complexity of the proposed architecture is O(m^2), but architecture based on systolic array is O(m^3).

Architecture proposed in this paper can be efficiently compute the exponentiation over GF(2^m) in public key cryptosystems.

Table 1. Comparison of performance

Structure	Systolic array [16]	Proposed paper
Operation	exponentiation	exponentiation
NO. of basic components	2(m-1) multipliers	1 MS*
NO. of AND gates	$4m^2(m-1)$	$3m-1$
NO. of XOR gates	$4m^2(m-1)$	$3m-1$
NO. of one bit latches	$14m^2(m-1)$	m^2+2m+1
NO.of MUXes	m	*1*
NO.of registers	0	m bit 5 | m-1 bit: 1
Execution time (clock cycles)	$2m^2+m$	m^2+3m

* MS：Structure of simultaneously performing modular multiplication and squaring

Table 2. Comparison of Area-Time Product for LSB first exponentiation

Structure	Systolic array [16]	Proposed paper
AREA	$4m^2(m-1)A_{2AND}$ $+4m^2(m-1)A_{2XOR}$ $+mA_{2MUX}$ $+(14m^3-14m^2)\ A_{1LATCH}$ $= (192m^3-192\ m^2+20m\ )\phi$	$(3m-1)A_{2AND}$ $+(3m-1)A_{2XOR}$ $+A_{2MUX}$ $+(6m-1)A_{1FF}$ $+\ (m^2+2m+1)\ A_{1LATCH}$ $= (8m^2+184m-10)\ \phi$
TIME	$(2m^2+m)(2T_{2AND}+4T_{2XOR})$ $= (13.6\ m^2+6.8\ m)\Delta$	$(m^2+3m)(T_{2AND}+T_{2XOR}+\ T_{1FF})=$ $(10.4\ m^2+31.2\ m)\ \Delta$
AREA × TIME	$(2611.2m^5-1305.6m^4-$ $1033.6m^3+136m^2)\ \phi\Delta$	$(83.2m^4+2163.2m^3+5636.8m^2-$ $312m)\ \phi\Delta$

Acknowledgements. This research was supported by Kyungpook National University Research Team Fund, 2002.

References

[1] R.J. McEliece, *Finite Fields for Computer Scientists and Engineerings*, New York:Kluwer Academic, 1987

[2] W. Diffie and M.E. Hellman, “New directions in cryptography,” *IEEE Trans. on Info. Theory*, VOL. 22, pp.644–654, Nov. 1976.

[3] T. ElGamal. “A public key cryptosystem and a signature scheme based on discrete logarithms,” *IEEE Trans. on Info. Theory*, VOL. 31(4). pp. 469–472, July 1985.
[4] A.J. Menezes, *Elliptic Curve Public Key Cryptosystems*, Kluwer Academic Publishers,1993.
[5] C.–S. YEH, IRVING S. REED, T.K. TRUONG, “Systolic Multipliers for Finite Fields GF(2^m),” *IEEE TRANSACTIONS ON COMPUTERS*, VOL. C-33, NO. 4, pp. 357–360, April 1984.
[6] C.L. Wang, J.L. Lin, “Systolic Array Implementation of Multipliers for Finite Fields GF(2^m),” *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*, VOL. 38, NO. 7, pp. 796–800,July 1991.
[7] P.L. Montgomery, “Modular multiplication without trial division,” *Mathematics of Computation*, 44(170):519–521, April, 1985.
[8] M. Delorme, J. Mazoyer, *Cellular Automata*, KLUWER ACADEMIC PUBLISHERS 1999.
[9] STEPHEN WOLFRAM, *Cellular Automata and Complexity*, Addison-Wesly Publishing Company, 1994.
[10] ELWYN R. BERLEKAMP, “Bit-Serial Reed-Solomon Encoders,” *IEEE TRANSACTIONS ON INFORMATION THEORY*, VOL. IT-28, NO. 6, pp. 869–874, November, 1982.
[11] P.P. Choudhury, R. Barua, “ Cellular Automata Based VLSI Architecture for Computing Multiplication And Inverse In GF(2^m),” *IEEE Proceeding of the 7th International Conference on VLSI Design*, pp.279–282, January 1994.
[12] Knuth, *THE ART OF COMPUTER PROGRAMMING*, VOL. 2/Seminumerical Algorithms, ADDISON-WESLEY, 1969.
[13] P.A. Scott, S.J. Simmons, S.E. Tavares, and L.E. Peppard, “Architectures for Exponentiation in GF(2m)”, *IEEE J. Selected Areas in Comm*., VOL.6., NO. 3, pp. 578–586, April, 1988.
[14] C.Parr, “Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents,” *IEEE TRANSACTIONS ON COMPUTERS*, VOL.48, NO. 10, pp. 1025-1034, October, 1999.
[15] Kyo-Min Ku, Kyeoung-Ju Ha, Hyun-Sung Kim, Kee-Young Yoo, “New Parallel Architecture for Modular Multiplication and Squaring Based on Cellular Automata.”, *PARA 2002, LNCS 2367*, pp.359–369, 2002
[16] C.L. Wang, “Bit-Level Systolic Array for Fast Exponentiation in Gf(2^m)”, *IEEE TRANSACTIONS ON COMPUTERS,* VOL. 43, NO. 7, pp. 838–841, July, 1994.
[17] D. D. Gajski, *Principles of Digital Design*, Prentice-Hall International, Inc., 1997.