Parallelization of Two Arithmetic Operations over Finite Field GF(2^n)

Yongnan Li[1] and Limin Xiao[1]

[1] *School of Computer Science and Engineering, Beihang University, Beijing, 100191, China*
liyongnan1984@cse.buaa.edu.cn xiaolm@buaa.edu.cn

Abstract

This paper presents two parallel algorithms of basic arithmetic operations concerning multiple-precision integers over finite field GF(2^n). The parallel algorithms of reduction operation and inversion-multiplication operation are designed by analyzing their data dependencies. Time complexities of the parallel algorithms and the sequential algorithms are calculated to make the quantitative comparison. The performance evaluation shows high efficiencies of the proposed parallel algorithms.

Keywords: *parallel algorithm; finite field GF(2^n); reduction; inversion-multiplication*

1. Introduction

The finite field GF(2^n) is one of the common used mathematical sets for constructing some cryptosystems including conic curves cryptosystem [1, 2] and elliptic curves cryptosystem [3,4]. In recent years, the parallel algorithms of some fundamental operations over finite field GF(2^n) have received considerable attention due to massive computation caused by the increased security demands.

However, there is less deep study on fast parallel algorithms concerning multiple-precision integers over finite field GF(2^n). Our previous works have designed several parallel algorithms about some basic operations of multiple-precision integers over two other mathematical sets [5-9]. This paper proposes two efficient parallel algorithms of multiple-precision over finite field GF(2^n) to accelerate the speed of basic operations over finite field GF(2^n).

The rest of this paper is organized as follows. Next section proposes the parallel algorithms concerning multiple-precision integers over finite field GF(2^n). The performance evaluation is presented in section 3. The last section concludes the whole paper and points out some future works briefly.

2. Parallel Algorithms over Finite Field GF(2^n)

This section discusses the proposed parallel algorithms for two arithmetic operations over finite field GF(2^n). We take one time clock as the computation time unit to make the quantitative evaluation of the efficiencies of parallel algorithms and sequential algorithms.

2.1 Parallel Reduction over GF(2^n)

The following algorithm is the traditional algorithm for computing reduction over GF(2^n).

Reduction over GF(2^n)

Input: module $f(Z) = Z^n + r(Z)$, polynomial $C(Z) = C_{2n-2}Z^{2n-2} + \cdots + C_1 Z + C_0$.
Output: $C(Z) mod\ f(Z)$.
1. for *i* from $n-2$ to 0, repeat:
  1.1 if $C_{i+n} = 1$, then
    $j \leftarrow \lfloor i/W \rfloor, \quad k \leftarrow i - Wj, \quad C\{j\} \leftarrow u_k(Z) \oplus C\{j\}$.
2. return $(C[t-1], \cdots, C[1], C[0])$.

The parameter *W* represents the word length of the computer and the parameter *n* denotes the degree of the modular polynomial $f(Z)$. The parameter $u_k(Z)$ means $Z^k r(Z)$ where $0 \le k \le W-1$. We define $m = \lceil (2n-1)/W \rceil$, so the polynomial $C(Z)$ could be expressed as

$$C(Z) = (C[m], \cdots, C[1], C[0]). \quad (2.1)$$

The parameter $C\{j\}$ means

$$C\{j\} = (C[m], \cdots, C[j+1], C[j]). \quad (2.2)$$

In the first step, one comparison must be executed to decide whether or not to execute the two arithmetic operations and one logical operation. The average number for computing the other three operations in all rounds of the first step is $(n-1)/2$ for the reason that the probability of C_{i+n} equal to 1 is 0.5. In substep 1.1, operation $j \leftarrow \lfloor i/W \rfloor$ denotes computing the word length of $\{C(i+n), \ldots, C(n)\}$ and it could be calculated in one time clock. Operation $k \leftarrow i - Wj$ is used to get the number of the bits in the highest word of $\{C_i Z^i + \cdots + C_1 Z^1 + C_0\}$, so the operation of multiplication and operation of deduction are not needed to be computed. Only one time clock is needed to obtain the result of this operation. Operation $C\{j\} \leftarrow u_k(Z) \oplus C\{j\}$ means executing XOR for every bit of $u_k(Z)$ and $C\{j\}$ from the lowest bit to the highest bit and it also needs one time clock. To sum up, the total runtime of the sequential procedure for computing reduction over GF(2^n) is

$$T_s = 5(n-1)/2. \quad (2.3)$$

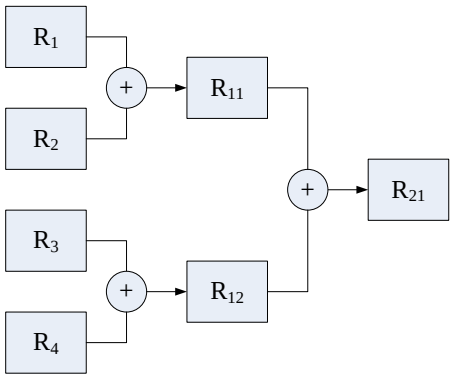

Figure 1. An Example of Incorporating the Intermediate Results

For the parallel procedure, every round of the first step can be calculated simultaneously except the operation of $C\{j\} \leftarrow u_k(Z) \oplus C\{j\}$. To obtain the final value of $C(Z) mod\ f(Z)$, the intermediate results of $u_k(Z)$ in substep 1.1 should be incorporated after the other operations finished. Obviously, the average number of the temporary result of $u_k(Z)$ is $(n-1)/2$. As depicted in Figure.1, we use the merging principle to incorporate the intermediate results of $u_k(Z)$ and it will cost $\lceil log_2((n-1)/2) \rceil$ time clocks. Then the total parallel runtime of reduction over GF(2^n) is

$$T_p = \lceil log_2((n-1)/2) \rceil + 2. \quad (2.4)$$

Therefore, the speedup is

$$S = \frac{5(n-1)/2}{\lceil log_2((n-1)/2) \rceil + 2}. \quad (2.5)$$

2.2 Parallel Inversion-multiplication over GF(2n)

The following algorithm is used to compute the inversion-multiplication over GF(2^n) and it is deduced from the Euclidean algorithm, which is used to calculate the inversion over GF(2^n). The terminology "inversion-multiplication" means the arithmetic operation of division over GF(2^n). In finite field GF(2^n), the operation of addition is same to the operation of XOR. The operation of shifting and operation of XOR both can be completed in one time clock.

Inversion-multiplication over GF(2^n)

Input: polynomial *a*, module *f*.
Output: $b \bullet a^{-1} mod\ f$.
1. $U \leftarrow a,\ V \leftarrow f,\ g_1 \leftarrow b,\ g_2 \leftarrow 0,\ k \leftarrow 0.$
2. when $U \neq 1$ and $V \neq 1$, repeat :
  2.1 when last bit of *U* is 0, repeat:
    $U \leftarrow U/2,\ g_2 \leftarrow 2g_2, k \leftarrow k+1.$
  2.2 when last bit of *V* is 0, repeat:
    $V \leftarrow V/2,\ g_1 \leftarrow 2g_1, k \leftarrow k+1.$
  2.3 if $deg(U) > deg(V)$, then $U \leftarrow U+V,\ g_1 \leftarrow g_1+g_2$;
    else, $V \leftarrow U+V, g_2 \leftarrow g_1+g_2$.
3. if $U = 1$, then $g \leftarrow g_1$; else, $g \leftarrow g_2$.
4. return $(g \bullet k) mod\ f$.

In the first step, there are only five operations of assignment. The five operations could be executed in one time clock in the parallel procedure while the sequential procedure needs five time clocks. Then we can get the parallel runtime and sequential runtime of the first step:

$$T_{p1} = 1. \quad (2.6)$$

$$T_{s1} = 5. \quad (2.7)$$

In the second step, substep 2.1 and substep 2.2 could be executed respectively in every round and the expression of $k = k+1$ is replaced by $k_1 = k_1 + 1$ and $k_2 = k_2 + 1$. In substep 2.3, the values of $U+V$ and g_1+g_2 could be computed simultaneously while calculating the additional equation $k = k_1 + k_2$. In substep 2.1, the probability of every bit in *U* is 0.5 and then the average number of the execution round is $1-(1/2)^n$, the sum of geometric sequence $1/2 + (1/2)^2 + \cdots + (1/2)^{n-1}$. The two shifting operations and one XOR operation cost only $1-(1/2)^n$ computing time units in every round in the parallel procedure. Obviously, the operations in substep 2.2 are same with the ones in substep 2.1 except the parameters. Two operations of comparison need to be considered before computing the substeps 2.1 and 2.2. In substep 2.3, the comparison operation must be computed firstly before the other three XOR operations (including $k = k_1 + k_2$) that are computed simultaneously in one time clock, so this substep costs two computing time units. The number of rounds in the second step depends on the number of bit '1' in coefficient of *U* or *V*. For every bit in coefficient of *U* or *V*, it obeys the binomial distribution and the mathematical expectation is *n/2*. Therefore, the average number of rounds in the second step is *n/2* and one computing time unit for executing the two comparisons in parallel would be cost at the beginning of every round. Then we can get the

average parallel runtime of the second step:

$$T_{p2} = (n/2)(5 - (1/2)^n). \quad (2.8)$$

In every round of substep 2.1 and substep 2.2, two shifting operations and one XOR operation are cost. In substep 2.3, we need to compute one comparison operation and two XOR operations. The expressions of $k = k_1 + k_2$, $k_1 = k_1 + 1$ and $k_2 = k_2 + 1$ don't need to be considered in the sequential procedure. Consequently, the total sequential runtime of the second step is

$$T_{s2} = (n/2)(13 - 6(1/2)^n). \quad (2.9)$$

In the third step, the operation of comparison and the operation of assignment must be executed sequentially because they have data dependencies. In the fourth step, *k* operations of shifting could be replaced by one assignment operation. Then we can get the parallel runtime and sequential runtime of the third step and the fourth step:

$$T_{p34} = 3. \quad (2.10)$$

$$T_{s34} = 3. \quad (2.11)$$

To sum up, the total parallel runtime and sequential runtime of inversion-multiplication over GF(2^n) are

$$T_p = 4 + (n/2)(5 - (1/2)^n). \quad (2.12)$$

$$T_s = 8 + (n/2)(13 - 6(1/2)^n). \quad (2.13)$$

Therefore, the speedup is

$$S = \frac{8 + (n/2)(13 - 6(1/2)^n)}{4 + (n/2)(5 - (1/2)^n)}. \quad (2.14)$$

3. Performance Evaluation

This section evaluates the performance of the parallel reduction algorithm and parallel inversion-multiplication algorithm over finite field GF(2^n). The parameter *n* in the equations of runtime is assigned into different values to make the quantitative comparison between the parallel algorithms and the sequential algorithms. The performance evaluation is depicted in Table 1 and the quantitative comparison between parallel algorithms and sequential algorithms is showed in Figure. 2 and Figure. 3. Be worth what carry is, the parallel algorithm of reduction achieves prominent efficiency improvement by contrast with the parallel algorithm of inversion-multiplication. The parallel algorithms proposed in this paper could significantly improve the performance of the two algorithms over finite field GF(2^n).

Table 1. Performance Evaluation

n	Tp_red	Ts_red	Tp_inv	Ts_inv
64	7	157.5	164	424
96	8	237.5	244	632
128	8	317.5	324	840
160	9	397.5	404	1048
192	9	477.5	484	1256
224	9	557.5	564	1464
256	9	637.5	644	1672

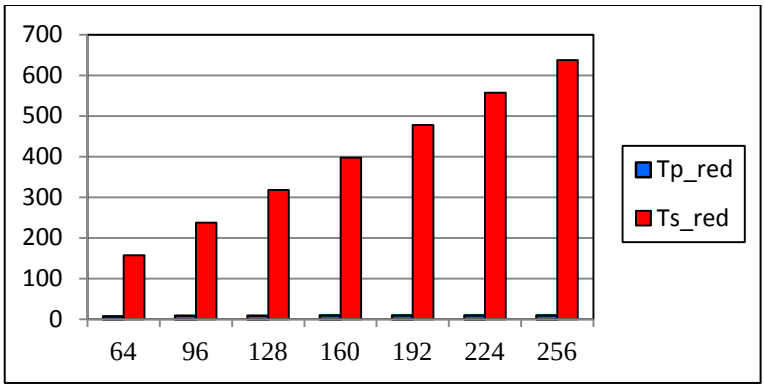

Figure 2. Performance Comparison of Reduction

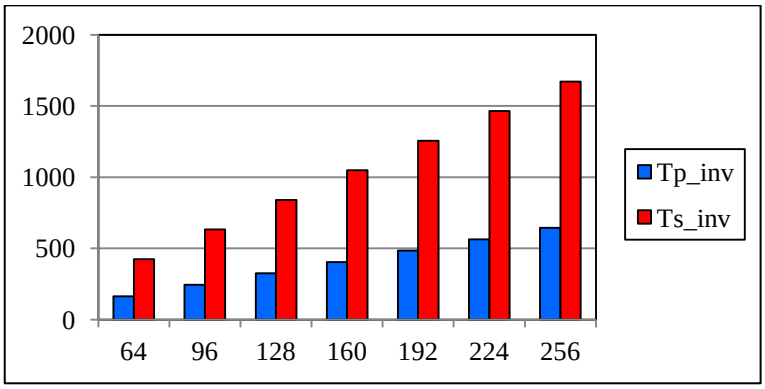

Figure 3. Performance Comparison of Inversion-multiplication

4. Conclusions

In this paper, we presented two parallel algorithms of multiple-precision over finite field GF(2^n). The parallel algorithms are designed by analyzing the data dependencies of the sequential algorithms. Time complexities and speedup ratios of the parallel algorithms and the sequential algorithms are discussed by measuring the time clocks of the operations. The performance evaluation and quantitative performance comparison demonstrate that our parallel algorithms of the two basic operations over finite field GF(2^n) achieve high efficiencies.

We only presented the methods for paralleling basic operations over finite field GF(2^n) in this paper. Future research efforts may focus on parallelization of point-operations for conic curves cryptosystem over finite field GF(2^n).

Acknowledgments

This study is sponsored by the National “Core electronic devices high-end general purpose chips and fundamental software” project under Grant No. 2010ZX01036-001-001, the Hi-tech Research and Development Program of China (863 Program) under Grant No. 2011AA01A205 and the National Natural Science Foundation of China under Grant No. 60973008.

References

[1] Z. Cao, "A public key cryptosystem based on a conic over finite fields Fp (in Chinese)", Advances in Cryptology: Chinacrypt98, Science Press, pp. 45–49 **(1998)**.

[2] Z. Cao, "Conic analog of RSA cryptosystem and some improved RSA cryptosystems", Natural Science Journal of Heilongjiang University, 16 (4), pp. 5–18 **(1999)**.

[3] N. Koblitz, "Elliptic curve cryptosystems", Mathematics of Computation, 48: pp. 203-209 **(1987)**.

[4] V. Miller, "Uses of elliptic curves in cryptography", Advances in Cryptology-Crypto'85. Lecture Notes in Comput. Sci, vol. 218, pp. 417–426, Springer-Verlag, Berlin **(1986)**.

[5] Y. Li, L. Xiao, Y. Hu, A. Liang and L. Tian, "Parallel algorithms for cryptosystem on conic curves over finite field Fp", Proceedings of 9th International Conference on Grid and Cloud Computing, pp. 163–167 **(2010)** November 1-5; Nanjing, China.

[6] Y. Li, L. Xiao, A. Liang and Z. Wang, "Parallel point-addition and point-double for cryptosystem on conic curves over ring Zn", Proceedings of 11th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 317–322 **(2010)** December 8-11; Wuhan, China.

[7] Y. Li and L. Xiao, "Parallel point-multiplication for conic curves cryptosystem", Proceedings of 3rd International Symposium on Parallel Architectures, Algorithms and Programming, pp.116–120 **(2010)** December 18-20; Dalian, China.

[8] Y. Li, L. Xiao, S. Chen, H. Tian, L. Ruan and B. Yu, "Parallel Extended Basic Operations for Conic Curves Cryptography over Ring Zn", Proceedings of 9th IEEE International Symposium on Parallel and Distributed Processing with Applications Workshops, pp. 203–209 **(2011)** May 26-28 May; Busan, Korea.

[9] Y. Li, L. Xiao, Z. Wang and H. Tian, "High Performance Point-Multiplication for Conic Curves Cryptosystem based on Standard NAF Algorithm and Chinese Remainder Theorem", Proceedings of 2011 International Conference on Information Science and Applications, **(2011)** April 26-29; Jeju, Korea.

Authors

Yongnan Li is a Ph.D. student at School of Computer Science and Engineering, Beihang University. He received his B.S. degree from Dalian Polytechnic University, China, in 2006 and M.S. degree from Northeastern University, China, in 2008, respectively. His main research areas are computer architecture, parallel computing and information security.

Limin Xiao is a professor of the Institute of Computer Architecture, Beihang University. He is also a senior membership of China Computer Federation. His main research areas are computer architecture, computer system software, high performance computing, virtualization and cloud computing.