저면적 RSA를 위한 효율적인 Montgomery 곱셈기 하드웨어 설계

Richard B. Nti · 류광기
한밭대학교 정보통신전문대학원

Hardware Design of Efficient Montgomery Multiplier for Low Area RSA

Richard B. Nti · Kwangki Ryoo
Graduate School of Information and Communication, Hanbat National University
E-mail : ntiboatengrichard@gmail.com, kkryoo@hanbat.ac.kr

요 약

공개 키 암호화에서 RSA 알고리즘은 연산시간이 높은 modular 지수 연산을 사용한다. RSA의 modular 지수 연산은 반복되는 modular 곱셈을 통해 연산한다. 빠른 해독 및 암호화 속도를 가지는 높은 효율의 RSA 알고리즘을 위해 수년간 빠른 modular 곱셈 알고리즘이 연구되었다. 그러나, Montgomery 곱셈은 추가적인 피연산자(반복 루프가 있는 3개의 피연사자)에 의해 캐리 전파 지연이 발생되는 단점이 있다. 본 논문에서는 RSA 암호화 시스템의 가벼운 어플리케이션을 위한 Montgomery 곱셈의 면적을 줄이는 하드웨어 구조를 제안한다. 제안된 하드웨어 구조는 90nm 셀 라이브러리 공정에서 합성한 결과 884.9MHz에서 84k 게이트 수를 가지며, 250MHz에서 56k 게이트 수를 가진다.

ABSTRACT

In public key cryptography such as RSA, modular exponentiation is the most time-consuming operation. RSA's modular exponentiation can be computed by repeated modular multiplication. To attain high efficiency for RSA, fast modular multiplication algorithms have been proposed to speed up decryption/encryption. Montgomery multiplication is limited by the carry propagation delay from the addition of long operands. In this paper, we propose a hardware structure that reduces the area of the Montgomery multiplication implementation for lightweight applications of RSA. Experimental results showed that the new design can achieve higher performance and reduce hardware area. A frequency of 884.9MHz and 250MHz were achieved with 84K and 56K gates respectively using the 90nm technology.

키워드
Public key cryptography, RSA, modular multiplication, CSA, Montgomery multiplication

Ⅰ. INTRODUCTION

Public key cryptosystems are vitals as far as information security is concerned. RSA is the most widely used public key algorithm[1]. RSA requires repeated modular multiplication to compute for modular exponentiation. Modular multiplication with large numbers is time-consuming. The Montgomery algorithm is used as the core algorithm for cryptosystems based on modular arithmetic. Montgomery algorithm determines the quotient by replacing the trial division by modulus with a series of additions and shift operations[2]. However, the three operand addition in the iteration loop causes carry propagation. The propagation delay influences the performance of the RSA. To avoid the delay, several approaches have been proposed to speed up the operation based on carry-save addition[3-5]. In this paper, we focus on the hardware design of efficient Montgomery multiplier with a two-level adder. A simplified Q_logic was designed for bit operation which accounted for a reduction in area.

Section II reviews radix-2 Montgomery algorithms. We proposed an efficient design of the Montgomery algorithm in section III. In Section IV, we compare different Montgomery multipliers. Finally, concluding remarks are drawn in Section V.

II. MODULAR MULTIPLICATION

The Montgomery multiplication is an algorithm used to compute the product of two integers A and B modulo N. Algorithm 1 shows the radix-2 version of the Montgomery multiplication algorithm. Given two integers a and b; where a, b < N (i.e. N is the k-bit modulus), R can be defined as 2^k modN where $2^{k-1} \leq N < 2^k$. The N-residue of a and b with respect to R can be defined as (1)

$$A = (a * R) \bmod N,\ B = (b * R) \bmod N \quad (1)$$

Algorithm 1: Radix-2 Montgomery Multiplication

```
Inputs: A, B, N (modulus)
Output: S[k]
1. S[0] = 0;
2. for i = 0 to k - 1 {
3. qi = (S[i]0 + Ai * B0) mod 2;
4. S[i+1] = (S[i] + Ai * B + qi * N)/2;}
5. if(S[k] ≥N ) S[k] = S[k] - N;
6. return S[k]
```

Based on (1), the Montgomery modular product S of A and B can be obtained as (2)

$$S = (A * B * R^{-1}) \bmod N \quad (2)$$

where R^{-1} is the inverse of R modulo N, i.e. $R * R^{-1} = 1$ (mod N). Since the convergence range of S in Montgomery Algorithm is $0 \leq S < 2N$, an additional operation $S = S - N$ is required to remove the oversized residue if $S \geq N$. The critical delay of algorithm 1 occurs during the calculation of S. This lead to variant modification of algorithm 1 to reduce the carry propagation delay.

III. PROPOSED HARDWARE DESIGN

In this section, we propose an efficient hardware design of Montgomery algorithm with reduce area complexity. The algorithm implemented is as proposed by Walter[3] shown in Algorithm 2.

Algorithm 2: Modified Montgomery Multiplication

```
Inputs: A, B, N (modulus)
Output: S[k + 2]
1. S[0] = 0;
2. for i = 0 to k + 1 {
3. qi = ( S[i]0 + A[i] * B0 ) mod 2;
4. S[i+1]=(S[i]+A[i] * B + qi * N ) div 2; }
5. return S[k + 2]
```

A. Simplified Q_logic Unit

From line (4) of algorithm 2, the computation of S[i+1] depends on the pre-computation of qi which evaluates to 0 or 1. Evaluation of even or odd numbers(line 3) can simply be deduced from the LSB (0=even, 1=odd). Inference can be made that qi is influenced by A[i]. As a result, a logic circuit of bit operation can model the given mathematical equation on line 3 as shown in fig. 1.

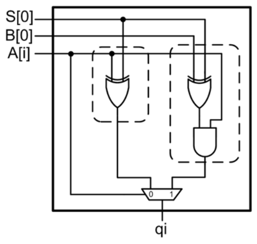

Fig. 1. Logic circuit of the simplified Q_logic

B. Architecture of Proposed Multiplier

Fig. 2 illustrates the block diagram of the proposed multiplier. Registers A, B and N store the inputs, S (shift reg) stores the intermediate computation and R represents the output. The arithmetic unit consists of two adders(two-level addition). Based on the control signals to multiplexers M1 and M2, different computations are carried out. Assuming both A[i] and qi equal zero(0), M1 and M2 route zero(0) to the output. As a result S[i]=S[i-1]. In case A[i] and qi equal one(1), M1 and M2 route B and N respectively. Therefore, S[i]=S[i-1]+B+N from line(4). At any given combination, the appropriate signals are routed to the two-level adder for computation. Note that S is a shift register which outputs a bit shift(right). The shifting accounts for the division by 2. The END signal controls the number of iterations before the final output is registered to R. All lines in short dashes represent bit signals.

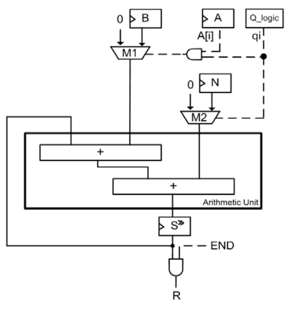

Fig.2 proposed multiplier structure

Table 1. Comparison of different Montgomery multiplier with 1024-bit key size

Multiplier	#Cycle	Delay (ns)	Area (um^2)	Throughput Rate(Mbps)
[4]	1049	5.60	406k	174.3
[5]	880	4.00	498k	290.9
proposed 1	1026*	4.00	313k	249.5
proposed 2	1026*	1.13	465k	883.2

Ⅳ. EXPERIMENTAL RESULTS

The synthesis of the proposed hardware design was done using TSMC 90nm CMOS cell library from Synopsys Design Compiler. Table 1 shows comparisons of different Montgomery multipliers. According to Kuang[5], #cycle represent the average clock cycles for completing one Montgomery operation measured through the simulation of 10000 random input patterns. On the other hand, the #cycle with asterisks (*) denote the worst case scenario to complete one Montgomery operation.

Our proposed multiplier design showed a reduction of about 37% in hardware area over the Kuang[5] at 250MHz. In addition, a clock speed of 884.9MHz was achieved with our design. For high-speed applications, our max frequency shows throughput enhancement by a factor of 3.04 and a reduction of about 6% in hardware complexity. A gate count of 56K and 84K were achieved at 250MHz and 884.9MHz operating frequency respectively.

Ⅴ. CONCLUSION

In this paper, we presented an alternative hardware design of Montgomery algorithm as modified by Walter. A simplified Q_logic was designed coupled with a compact arithmetic unit. Kuang[5] showed a reduction in clock cycles which was better than our approach. Synthesis results conclude that our multiplier has an improved performance for low area modular multiplication applications.

Acknowledgments

This research was supported by the MSI(Ministry of Science, ICT and Future Planning), Korea, under the Global IT Talent support program(IITP-2017-0-01681) and Human Resource Development Project for Brain scouting program(IITP-2016-0-00352) supervised by the IITP(Institute for Information and Communication Technology Promotion)

REFERENCE

[1] R. L. Rivest, A. Shamir, and L. Adleman, “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,” Commun. ACM, Vol. 21, No. 2, pp. 120-126, Feb. 1978.

[2] P. L. Montgomery, “Modular multiplication without trial division,” Math. Comput., Vol. 44, No. 170, pp. 519–521, Apr. 1985.

[3] C. D. Walter, “Montgomery exponentiation needs no final subtractions,” Electron. Lett., Vol. 35, No. 21, pp. 1831-1832, Oct. 1999.

[4] Y. Y. Zhang, A. Li, L. Yang, and S. W. Zhang, “An efficient CSA architecture for Montgomery’s modular multiplication,” Microprocessors Microsyst., Vol. 31, No. 7, pp. 456-459, Nov. 2007.

[5] S. R. Kuang, K. Y. Wu and R. Y. Lu, “Low-Cost High-Performance VLSI Architecture for Montgomery Modular Multiplication,” IEEE Trans. Very Large Scale Integration (VLSI) Syst., Vol. 24, No. 2, pp. 440-442, Feb. 2016.