논문 02-27-3C-7 한국통신학회논문지 '02-3 Vol.27 No.3C

RSA 암호 시스템을 위한 고속 모듈라 곱셈 알고리즘

준회원 조 군 식*, 정회원 조 준 동**

High Speed Modular Multiplication Algorithm for RSA Cryptosystem

Koon-Shik Cho* *Associate Member*, Jun-Dong Cho** *Regular Member*

요 약

본 논문에서는 sign estimation technique [3]을 기초로 한 radix-4 모듈라 곱셈 알고리즘을 제안한다. Sign estimation technique은 carry와 sum의 형태로 표현되는 수에서 부호를 알아내는 것이다. 이 방법은 5비트 carry look-ahead adder로 구현이 가능하다. RSA와 같은 암호화 시스템에서는 모듈라 곱셈이 하드웨어의 성능을 좌우한다. 제안한 알고리즘은 modulus가 n 비트인 경우, 모듈라 곱셈 수행시 일반적인 알고리즘의 약 반 클럭 (n/2+3) 사이클만 필요하다. 그래서 매우 큰수의 modulus 사용하는 RSA 암호시스템에서 모듈라 멱승 연산에 매우 효율적이다. 또한 모듈라 곱셈의 하드웨어 성능을 향상하기 위해, CSA (Carry Save Adder)의 맨 마지막 출력에 사용되는 CPA (Carry Propagation Adder) 대신 고속 덧셈기[7]를 사용하였다. 모듈라 멱승 계산이 n 클럭이 소요되는 RL binary 방법을 적용하여 1024 비트 데이터를 RSA 암호화하는데 n(n/2+3) 클럭 사이클만 소요된다.

ABSTRACT

This paper presents a novel radix-4 modular multiplication algorithm based on the sign estimation technique [3]. The sign estimation technique detects the sign of a number represented in the form of a carry-sum pair. It can be implemented with 5-bit carry look-ahead adder. The hardware speed of the cryptosystem is dependent on the performance modular multiplication of large numbers. Our algorithm requires only (n/2+3) clock cycle for n bit modulus in performing modular multiplication. Our algorithm out-performs existing algorithm in terms of required clock cycles by a half. It is efficient for modular exponentiation with large modulus used in RSA cryptosystem. Also, we use high-speed adder [7] instead of CPA (Carry Propagation Adder) for modular multiplication hardware performance in final stage of CSA (Carry Save Adder) output. We apply RL (Right-and-Left) binary method for modular exponentiation because the number of clock cycles required to complete the modular exponentiation takes n cycles. Thus, One 1024-bit RSA operation can be done after n(n/2+3) clock cycles.

I. Introduction

The data security will play a crucial role in many future computer and communication systems [1]. The fundamental security requirements include confidentiality, authentication, data integrity, and non-repudiation. To provide such security services, most systems use public key cryptography. Among the various public key cryptography algorithms, RSA cryptosystem [2] is widely used public key cryptosystem today. The main objective of public key encryption is to provide privacy or confidentiality.

In public key cryptography algorithms, the essential arithmetic operation is modular multiplication, which is used to calculate modular exponentiation.

However, modular exponentiation on numbers

* 성균관대학교 전기전자컴퓨터공학부 koonshik@vada1.skku.ac.kr, ** 성균관대학교 전기전자컴퓨터공학부 jdcho@yurim.skku.ac.kr
논문번호 : 010308-1026 접수일자 : 2001년 10월 26일

of thousands of bits (1,024 bits or higher) makes it difficult for the RSA algorithm to attain high throughput. Also, public key encryption schemes are typically substantially slower than symmetric key encryption algorithm such as DES [13]. Fast exponentiation is becoming increasingly important with the widening use of encryption. Whereas the most startling improvements in speed are achieved through the use of dedicated hardware for multiplication, some small gains can also be made through a good choice of algorithm for organizing the order of multiplications at the hardware level.

The previous works on modular multiplications are as follows. The two systolic architectures is presented to speed up the computation of modular multiplication [14]. It is the main operation of montgomery's algorithm. The radix-4 modular multiplication algorithm based on Montgomery's algorithm is presented [15]. This algorithm runs four times faster than those based on original montgomery algorithm. The montgomery algorithm computes $R=ABr^{-1} mod\ N$ [13]. The result R involves an undesired factor r^{-1} (i.e. extra factor). However, the montgomery algorithm has an extra advantage. The correction involves some precomputation based on the divisor and an extra modular multiplication. Also, a precomputation constant [1] should be calculated or stored to remove the extra factor.

The sign estimation technique was used to obtain fast algorithms for multi-operand modular addition [4] and modular multiplication [3,5] operations. The improved sign estimation algorithm presented here correctly computes the sign of the number when the number is large enough in magnitude [6]. In this paper, improving the modified Koc's and Hung's algorithm [3], we propose a novel radix-4 modular multiplication algorithm based on the sign estimation technique. The radix-4 technique for multiplication is adopted to reduce the number of clock cycles needed in a 1024-bit RSA operation. Our algorithm is fast in computing large-bit addition and subtractions without using carry propagation (i.e., with using Carry Save Adder). Also our algorithm does not require an additional computation as in montgomery multiplication algorithm. [1,12,14,15]. In the next section, we describe modular multiplication based on sign estimation technique. We propose a novel radix-4 modular multiplication algorithm. Next, in Section 3, we show how to adopt modular exponentiation. In Section 4, hardware realization of our high-speed radix-4 modular multiplier is presented. Finally, Section 5 draws a conclusion.

Ⅱ. Modular Multiplication Algorithm

The RSA cryptosystem is the most widely used in the public-key cryptosystem. It may be used to provide both secrecy and digital signatures and its security is based on the intractability of the integer factorization problem.

In RSA, encrypt M into a ciphertext C by the rule,

$$C=M^e \ mod\ N$$

where M is the message such that $0 \leq M < N$, e is public key and the modulus N is a n-bit positive integer. The decryption operation is performed using the private key by computing

$$M=C^d \ mod\ N$$

Both encryption and decryption are modular exponentiation. Therefore, the fast modular exponentiation algorithm is required. For most hardware implementation, the simple square and multiply algorithm, often called binary method, is assumed to be the most effective because of the simple hardware structure and repetitive control. In the RL (Right-to-Left) binary method, n iterations are needed and each iteration need two modular multiplications. Also, these two modular multiplication can be done in parallel by RL (Right-to-Left) binary method. We reduced the number of partial products by using radix-4 Booth's algorithm for high performance, and designed a radix-4 modular multiplier based on Koc's modular multiplication algorithm by using sign estimation technique.

1. Review of Sign Estimation Technique

The many existing redundant representations, the carry save technique is rather inexpensive to implement, and thus, widely used [6]. In carry save adder, a partial sum(S) and carry(C) sequence are generated in the intermediate stages and the carry propagation occurs only at the last stage. The sign of a number in one's or two's complement representations is indicated by the most significant bit. However, in the carry save representation, the most significant bit (the sign bit) is not readily available. In order to compute the exact sign of the partial sum $C+S$, we have to sum C and S in full precision. To solve this problem, the sign estimation algorithm estimates the sign of a number represented by a carry-sum pair produced by a carry save adder [5].

The truncation function $T(X)$ is defined as the operation which replaces the least significant t bits of X with zeros, i.e. if X is an n bit number,

$$T(X) = X_{n-1}X_{n-2}\cdots X_t \underbrace{0\cdots 0}_{t\ zeros}$$

The parameter t controls the cost and quality of sign estimation. The following two inequalities can easily be proven [3]:

$$T(X) \le X \le T(X) + 2^t \quad (1)$$

$$T(S)+T(C) \le S+C \le T(S)+T(C)+2^{t+1} \quad (2)$$

The following algorithm [3] is modular multiplication algorithm by using sign estimation technique. The modular multiplication problem is defined as the computation of $P=AB(mod\ N)$ given the integer A, B and N. It is usually assumed that A and B is positive integers with $0 \le A, B < N$.

Algorithm 1.

1. Set $S^{(0)}=0$ and $C^{(0)}=0$ $M=-N$
2. Repeat Step 2a, 2b, and 2c for i = 1, 2, 3, ···, n
   2a. $(C^{(i)}, S^{(i)}) := 2(C^{(i-1)}+S^{(i-1)})+A_{n-i}B$
   2b. $(\overline{C}^{(i)}, \overline{S}^{(i)}) := C^{(i)}+S^{(i)}+2M$
   If $T(\overline{C}^{(i)})+T(\overline{S}^{(i)}) \ge 0$ then set $C^{(i)}=\overline{C}^{(i)}$ and $S^{(i)}=\overline{S}^{(i)}$
   2c. $(\overline{C}^{(i)}, \overline{S}^{(i)}) := C^{(i)}+S^{(i)}+M$
   If $T(\overline{C}^{(i)})+T(\overline{S}^{(i)}) \ge 0$ then set $C^{(i)}=\overline{C}^{(i)}$ *and* $S^{(i)}=\overline{S}^{(i)}$
3. $(\overline{C}^{(i)}, \overline{S}^{(i)}) := C^{(i)}+S^{(i)}+M$
4. Compute $P := C^{(n)}+S^{(n)}$ and $\overline{P} := \overline{C}^{(n)}+\overline{S}^{(n)}$
5. If $\overline{P} \ge 0$ then $P=\overline{P}$
6. Return P.

In algorithm 1, $\overline{C}^{(i)}$, $\overline{S}^{(i)}$ and $\overline{P}$ represent the temporary values of $C^{(i)}$, $S^{(i)}$ and P. It is used to transform T to estimate the sign of $C^{(i)}+S^{(i)}$ using the bits of the temporary carry and sum, starting from bit location $t=n-1$.

2. Radix-4 Modular Multiplication

For high-speed realization of modular exponentiation, a novel radix-4 modular multiplication algorithm is introduced. The algorithm is based on a sign estimation technique that estimates the sign of a number represented by a carry-sum pair produced by a carry save adder. By using radix-4 numbers for modular multiplication and modular reduction, the number of iteration can be reduced by half. The modulus N is n-bit number, which is equivalent to the number of partial products. Our algorithm reduces the number of operations and requires $n/2+3$ iterations. Booth encoding is also used to our algorithm.

The following algorithm computes $P=AB(mod\ N)$. In order to apply the Booth algorithm in performing the modular multiplication operations, let A be $n+3$ bit and B be $n+1$ bit 2's complement numbers and N is an n-bit odd integer, where $-N \le A, B < N$. Also, let

$BP_i = A_{Booth\ encoding} * B = (BP_i, BP_{i-1}, \cdots, BP_1, BP_0)$

where $0 \le i \le n/2+1$.

Algorithm 2. R4MM(A, B, N)

1. Set $S^{(0)}=0$ and $C^{(0)}=0$
2. for $i = 1$ *to* $\frac{n}{2}+2$
   $(C^{(i)}, S^{(i)}) := 4(C^{(i-1)}+S^{(i-1)})+BP_i$
   If $T(C^{(i)})+T(S^{(i)}) \ge 0$ then
   $(\overline{C}^{(i)}, \overline{S}^{(i)}) := C^{(i)}+S^{(i)}-2N$

If $T(\bar{C}^{(i)})+T(\bar{S}^{(i)})\geq 0$ then $C^{(i)}=\bar{C}^{(i)}$ and $S^{(i)}=\bar{S}^{(i)}$
$(\bar{C}^{(i)},\bar{S}^{(i)}):=C^{(i)}+S^{(i)}-2N$
If $T(\bar{C}^{(i)})+T(\bar{S}^{(i)})\geq 0$ then $C^{(i)}=\bar{C}^{(i)}$ and $S^{(i)}=\bar{S}^{(i)}$
end
end
$(\bar{C}^{(i)},\bar{S}^{(i)}):=C^{(i)}+S^{(i)}-N$
If $T(\bar{C}^{(i)})+T(\bar{S}^{(i)})\geq 0$ then $C^{(i)}=\bar{C}^{(i)}$ and $S^{(i)}=\bar{S}^{(i)}$
end
else
$(C^{(i)},S^{(i)}):=C^{(i)}+S^{(i)}+N$
If $T(C^{(i)})+T(S^{(i)})<0$ then
$(C^{(i)},S^{(i)}):=C^{(i)}+S^{(i)}+N$
end
end
3. $(\bar{C}^{(i)},\bar{S}^{(i)}):=C^{(i)}+S^{(i)}-N$
4. Compute $P:=C^{(n)}+S^{(n)}$ and $\bar{P}:=\bar{C}^{(n)}+\bar{S}^{(n)}$
5. If $\bar{P}\geq 0$ then $P=\bar{P}$
6. Return P.

In order to utilize the carry save adders in performing the modular multiplication operations, we present the numbers as the carry save pairs (C, S), where the value of the number is the sum $C+S$. The carry-save adder is used to subtract N or $2N$ from $C^{(i)}+S^{(i)}$. The modulus is represented in 2's complement form. Thus, subtraction operation of algorithm 2 performs addition.

Assuming that we obtain $T(C^{(i)})+T(S^{(i)})\geq 0$ after K subtraction. If we perform one more subtraction and obtain $T(C^{(i)})+T(S^{(i)})<0$, then $T(C^{(i)})+T(S^{(i)})<-2^t$, because it must be a multiple of 2^t. There, after K subtraction, we have $0\leq T(C^{(i)})+T(S^{(i)})<N-2^t$ [3].

Equation (2) can be re-written as

$$0\leq C^{(i)}+S^{(i)}<N+2^t \qquad (3)$$

In the next step, we compute $C^{(i+1)}$ and $S^{(i+1)}$. Because $(C^{(i+1)}, S^{(i+1)}) := 4(C^{(i)}+S^{(i)}) + A_{n-i-1}B$, this gives,

$$0\leq C^{(i+1)}+S^{(i+1)}<4(N+2^t)+N=5N+2^{t+2} \qquad (4)$$

If we perform five subtraction ($K=5$), then $C^{(i+1)}+S^{(i+1)}$ will be less than 2^{t+2}. To satisfy the requirement of equation (3), we select $t+2=n$. We found that sign estimation parameter (t) is $t=n-2$. Booth encoding is applied to $A_{n-i-1}B$.

The modular reduction procedure in step 2 of algorithm 2 as described above subtracts N from (C, S) in each of the 5 iterations. Instead of subtracting N five times, $2N$, $2N$ and N are subtracted, respectively. After step 3 and step 4, the number of modular operations is reduced in the range $[0, N)$. In step 4, high-speed adder [7] is used because it is one of the most critical parts affecting the hardware performance of modular multiplication. The sign estimation procedure checks 5 most significant bits of $\overline{C}^{(i)}$ and $\overline{S}^{(i)}$ from the bit locations $n-1$ to $n+3$. The procedure is implemented with a 5-bit carry look adder.

Ⅲ. Modular Exponentiation Algorithm

The encryption process of the RSA is performed mainly by the modular exponentiation. The decryption can be done in the same way. A simply way to perform modular exponentiation is to repeat modular squaring and modular multiplication. The binary method for computing modular exponentiation has two variations depending on the direction by which the bits of exponent are scanned : Left-to-Right (LR) and Right-and-Left (RL). The LR binary method requires $2n$ clock cycles. The RL binary method requires n clock cycles [8]. We apply RL binary method by adopting radix-4 modular multiplication to modular exponentiation. The following RL binary method algorithm computes $C=M^e \bmod N$. Here we assume e is represented by an h-bit binary number $(e_{h-1}\ e_{h-2}\ \cdots\ e_1\ e_0)$.

ME(M, e, N)
1. $C := 1$
2.for $i = 0$ to $h-2$
if $e_i = 1$ then $R4MM(C, M, N)$
$M := R4MM(M, M, N)$
if $e_i = 1$ then $C := R4MM(C, M, N)$
3. return C

Each cycle of the loop potentially requires two modular multiplications, which may be executed in parallel. Also, the computing time is reduced to which is independent on the number of 1-bits in *e*.

Ⅳ. Hardware Implementation

Fig. 1 shows the overall architecture of a 1,024-bit RSA processor. Our design executes two radix-4 modular multiplications simultaneously. The number of clock cycles required to complete the modular exponentiation is *n* cycles.

Based on the presented modular multiplication algorithm, modular exponentiation can be accomplished by implementing three block : 1) the controller block, 2) register block and 3) the multiplication block. The controller block controls all input and outputs of the multiplication block. The register block stores parameters through a 32-bit input buffer to perform modular exponentiation. The multiplication block is described as below. The modular multiplication architecture is also given in Fig. 2.

In hardware implementation every step in radix-4 modular multiplication algorithm is designed by four-level CSA. The CSA block performs modular reduction operations through the three-level CSA or four-level CSA (shown in loop part of algorithm 2). The controller1 controls carry, sum and modulus (*-N*, *-2N* or *N*) for CSA operation and also all inputs for high-speed adder. In the design of a high-speed adder [7], we employed a combination of carry-skip and carry-

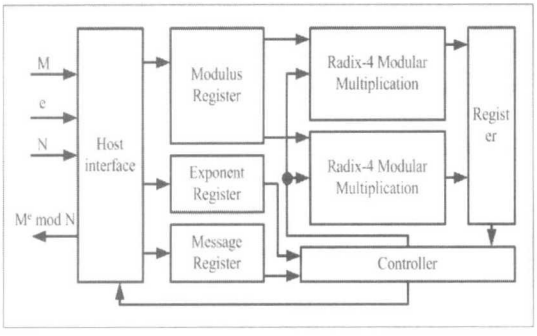


Fig. 1 Architecture of the 1,024-bit RSA processor

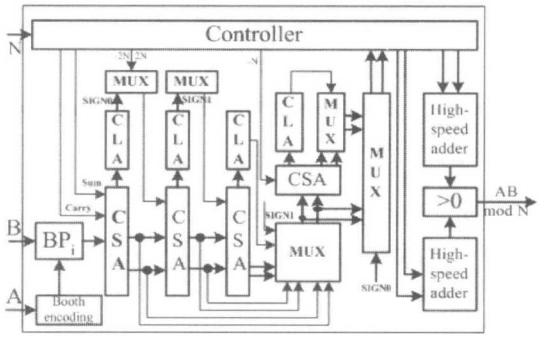


Fig. 2 The radix-4 modular multiplication block.

select techniques. Fig. 3 shows the basic structure of the adder. This adder performs 46.5ns delay for 1028 bit operation. Therefore, instead of carry propagation adder in step 4 of algorithm 2, we use high-speed adder to improve modular multiplication. The booth recoding rules for booth encoding block applied are shown in table 1 [9].

Table 1. Radix-4 Booth's recoding rules.

A_n	A_{n-1}	A_{n-2}	Recoded digit	Operation on B
0	0	0	0	0B
0	0	1	+1	+1B
0	1	0	+1	+1B
0	1	1	+2	+2B
1	0	0	-2	-2B
1	0	1	-1	-B
1	1	0	-1	-B
1	1	1	0	0B

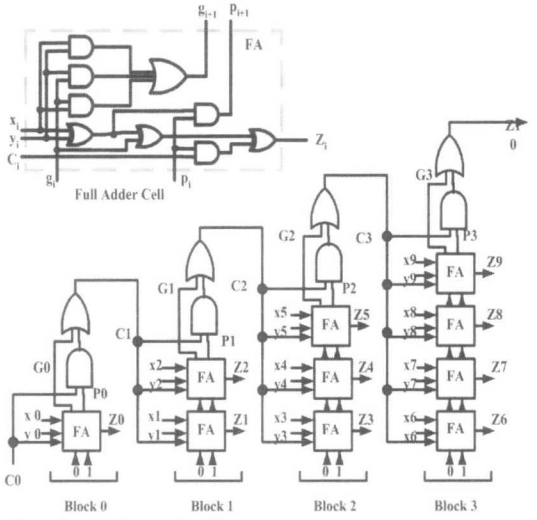


Fig. 3 High-speed adder

The total number of clock cycles required to complete the modular multiplication is *(n/2+3)* cycles to perform the desired operations. The loop part of algorithm 2 takes *(n/2+1)* cycles. The high-speed adder block performs only 2 clock cycles at 40*MHz*. One 1024-bit RSA operation can be done after *n(n/2+3)* clock cycles. As a result, the total operation time is 13ms at 40*MHz*. Table 2 shows a comparison of some RSA processor. We compare the operating time and clock frequency of I.C. in consumer electronic. It can be seen that our design has the shortest operating time.

Table 2. A comparison of some RSA processor

ITEM Circuit	IBM [10]	PIJNENBURG [11]	Our design
Clock frequency	33 MHz	25 MHz	40MHz
Gate counts	Crypto core + 10K gate logic	Unknown	230k
Number of bits	1024	1024	1024
Operating time	32ms	48ms	13ms

V. Conclusions

In this paper, we presented a novel hardware realization of a 1024-bit RSA processor. The proposed a novel radix-4 modular multiplication algorithm is based on sign estimation technique and improves existing modular multiplication algorithms significantly. The carry look-ahead logic is also used to detect the sign of the partial product during the CSA process. To complete the modular multiplication for desired operations, the number of clock cycles required is only *(n/2+3)*. The proposed algorithm is implemented and further reduced computational time for modular multiplication. The total number of gate was increased due to the additional one-level CSA and control part. However, our algorithm does not require an extra factor as in Montgomery multiplication algorithm [12]. Furthermore, precomputation constant [1] is not required in our algorithm. The baud rate of RSA operation is *78.8kbits/s*. The hardware evaluation demonstrated the efficiency of our algorithm and our algorithm can be effectively applied to the high-speed RSA cryptosystems.

Reference

[1] T. Blum and C. Paar, "Montgomery modular exponentiation on reconfigurable hardware," 14th IEEE Symposium on Computer Arithmatic, pp.70 77, 1999.

[2] R. L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signature and public-key cryptosystems," Commun. ACM, vol.21, pp.120126, Feb. 1978.

[3] C. K. Koc and C. Y. Hung, "Carry save adders for computing the product AB modulo N," Electronics Letters, 26(13), pp.899-.900, June 1990.

[4] C. K. Koc and C. Y. Hung, "Multi-operand modulo addition using carry save adders," Electronics Letters, 26(6), pp.361-363, March 1990.

[5] C. K. Koc and C. Y. Hung, "Bit-level systolic arrays for modular multiplication," Journal of VLSI Signal Processing, 3(3), pp.215-223,1991

[6] C. K. Koc and C. Y. Hung, "Fast algorithm for modular reduction," IEE Proceedings - Computers and Digital Techniques, 145(4), pp.265-271, July 1998.

[7] A. Satoh, N. Ooba, K. Takano and E. D'Avignon, "High-speed MARS hardware," Third Advanced Encryption Standard (AES) Candidate Conference, April 2000 http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html)

[8] C. K. Koc. "*RSA Hardware Implementation.*" TR 801, RSA Laboratories, April 1996. http://www.rsa.com/rsalabs/html/tech_notes.html

[9] Parhami Behrooz "*Computer Arithmetic Algorithms and Hardware Designs*" Oxford University Press Inc. 2000

[10] IBM, MEAC1024 : High-performance crypto-

graphic engine http://www.tri.ibm.com/projects/rsa/meac144e.pdf

[11 PIJNENBURG Company, PCC201 High Speed Exponentiator, http://www.secure-a-link.com/ pcc201. html

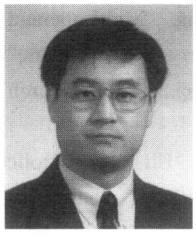
[12] P.L. Montgomery, "Modular multiplication without trial division," Mathematics of Computation, 44(170), pp519-521, April 1985.

[13] Alfred j. Menezes, " *Handbook of Applied Cryptography*," CRC Press, 1996

[14] W.C. Tsai, C.B. Shung and S.J.Wang, "Two systolic architecture for modular multiplication," IEEE Trans. on VLSI System. Vol.8, No.1, Feb. 2000.

[15] J. H. Hong and C. W. Wu, "Radix-4 modular multiplication and exponentiation algorithm for the RSA public-key cryptosystem," ASP-DAC pp. 565 570, 2000

조 준 동(Jun-Dong Cho) 정회원



1980년 2월 : 성균관대학교 전자공학과 졸업 (학사)

1983년 6월~1987년 8월 : 삼성전자 CAD 팀 근무 (연구원, 팀장)

1989년 9월 : Polytechnic University, Brooklyn, NY 전산학과 졸업 (석사)

1993년 7월 : Northwestern University 전산학과 졸업 (박사)

1995년 3월~현재 : 성균관대학교 전기전자컴퓨터공학부 (부교수)

IEEE Senior Member

<주관심 분야> 디지털통신, 무선통신, 이동통신, 저전력 설계 기술, VLSI CAD

조 군 식(Koon-Shik Cho) 준회원

1994년 2월 : 순천향대학교 정보통신공학과 졸업 (학사)

1996년 2월 : 순천향대학원 정보통신공학과 졸업 (석사)

1996년 1월~2000년 1월 : (주)콤텍시스템 기술연구소 근무

2000년 3월~현재 : 성균관대학교 전기전자컴퓨터공학부 박사과정

<주관심 분야> 암호프로세서, embedded system, 저전력 설계 기술