변형된 RBA를 이용한 몽고메리 곱셈기와 하드웨어 구조

지성연*, 임대성*, 장남수*, 김창한**, 이상진*

*고려대학교 정보보호대학원, **세명대학교 정보보호학과

Montgomery Multiplier Base on Modified RBA and Hardware Architecture

Sung-Yeon Ji*, Dae Sung Lim*, Nam Su Jang*, Chang Han Kim**, Sangjin Lee*

*Graduate School Information Security, Korea Univ.

**Information & Communication System, Semyung Univ.

요 약

RSA 암호 시스템은 IC카드, 모바일 및 WPKI, 전자화폐, SET, SSL 시스템 등에 많이 사용된다. RSA는 모듈러 지수승 연산을 통하여 수행되며, Montgomery 곱셈기를 사용하는 것이 효율적이라고 알려져 있다. Montgomery 곱셈기에서 임계 경로 지연 시간(Critical Path Delay)은 세 피연산자의 덧셈에 의존하고 캐리 전파를 효율적으로 처리하는 문제는 Montgomery 곱셈기의 효율성에 큰 영향을 미친다. 최근 캐리 전파를 제거하는 방법으로 캐리 저장 덧셈기(Carry Save Adder, CSA)를 사용하는 연구가 계속 되고 있다. McIvor외 세 명은 지수승 연산에 최적인 CSA 3단계로 구성된 Montgomery 곱셈기와 CSA 2단계로 구성된 Montgomery 곱셈기를 제안했다. 시간 복잡도 측면에서 후자는 전자에 비해 효율적이다. 본 논문에서는 후자보다 빠른 연산을 수행하기 위해 캐리 전파 제거 특성을 가진 이진 부호 자리(Signed-Digit, SD) 수 체계를 사용한다. 두 이진 SD 수의 덧셈을 수행하는 잉여 이진 덧셈기(Redundant Binary Adder, RBA)를 새로 제안하고 Montgomery 곱셈기에 적용한다. 기존의 RBA에서 사용하는 이진 SD 덧셈 규칙 대신 새로운 덧셈 규칙을 제안하고 삼성 STD130 0.18μm 1.8V 표준 셀 라이브러리에서 지원하는 게이트들을 사용하여 설계하고 시뮬레이션 하였다. 그 결과 McIvor의 2 방법과 기존의 RBA보다 최소 12.46%의 속도 향상을 보였다.

I. 서론

1976년 W. Diffe와 M.E. Hellman이 공개키 암호 시스템의 개념을 제안하였다. 그 후 Rivest, Shamir, Adleman에 의해 구현된 RSA 암호 시스템은 키 분배와 관리의 용이, 인증과 디지털 서명이 가능한 장점으로 인해 널리 사용되고 있다. RSA 공개키 암호 시스템은 매우 큰 정수에 대한 소인수분해가 어렵고 해를 찾아내는데 많은 시간을 요구한다는 점을 이용한 암호 시스템이다. 그러나 최근 연산 환경의 발전 및 알고리즘의 효율성 개선 등으로 인수분해가 가능한 정수의 크기가 점차 커지고 있고 앞으로도 증가할 것이다[8]. 따라서 RSA 암호 시스템의 안전성을 위해서 키 길이는 길어지게 되고 상대적으로 연산량이 증가하게 된다. 그러므로 RSA 암호 시스템의 효율적인 구성은 매우 중요한 연구 과제이다.

RSA 암호 시스템은 모듈러 지수승 연산을 통하여 수행되고, 모듈러 지수승 연산은 반복적인 모듈러 곱셈을 통해 구성된다. 따라서 모듈러 곱셈기의 효율적인 구현은 RSA 암호 시스템의 효율성에 있어 중요한 문제이다. RSA 암호 시스템에서 사용되는 모듈러 연산을 위해 개발된 방법 중 Montgomery 모듈러 곱셈 알고리즘이 효율적이라고 알려져 있다[2]. Montgomery 모듈러 곱셈 알고리즘의 효율성은 세 입력 값에 대한 덧셈의 빠른 설계에 의존한다.

일반적인 이진 수 체계에서는 덧셈 수행 시 캐리 전파를 발생한다. 대부분 사용하는 캐리 전파 덧셈기(Carry Propagation Adder)의 종류는 비트 길이가 늘어날수록 연산 수행 시간이 증가한다. 따라서 이 문제를 해결하기 위한 방법으로 발생하는 캐리를 저장하는 CSA를 이용한 방법이 제안되었다. 하지만, CSA를 이용한 결과는 캐리와 합, 즉, 두 개의 값으로 표현되므로, 반복 연산 수행 시 입력 값의 개수가 증가한다. 최근 CSA를 이용한 5개의 입력 값과 2개의 출력 값을 가지는 곱셈기([3]에서 McIvor 외 세 명이 제안한 첫 번째 방

법: McIvor의 1 방법)와 4개의 입력 값과 2개의 출력 값을 가지는 곱셈기([3]에서 McIvor 외 세 명이 제안한 두 번째 방법: McIvor의 2 방법)가 제안되었다.[3] 전자의 곱셈기는 3단계의 CSA를 이용하지만 후자의 곱셈기는 사전 계산을 통해 값을 저장하여 MUX를 이용해서 사용하는 방법으로 5개의 입력 값을 4개로 줄여 2단계의 CSA로 구현이 가능하다. 4개의 입력 값은 캐리와 합으로 표현된 2개의 정수가 된다.

본 논문에서는 캐리 전파를 제거하기 위한 방법으로 Avizienis에 의해 [4]에서 처음으로 제안된 SD 수 체계를 사용한다. SD 수 체계에서는 동일한 수에 대해서 다른 표현 방법이 존재하므로 덧셈 설계 방법 또한 다양하다. 그러나 부호를 나타내기 위한 비트가 추가로 사용되어야 한다. 이진 SD 수 체계에서 고려하면 두 이진 SD 수의 덧셈은 부호와 값을 표현하는 비트가 각각 존재해야 하므로 총 4비트가 사용되며 결과 또한 부호와 값을 표현하는 2비트가 된다. 그러므로 2단계의 CSA와 동일하게 4개의 입력 값과 2개의 출력 값을 갖는 구조가 된다. 따라서 이 구조를 얼마나 효율적으로 설계하느냐가 전체 곱셈기의 효율성에 크게 영향을 미치게 된다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 지수승 연산을 효율적으로 수행할 수 있는 이진 Montgomery 곱셈 알고리즘과 CSA를 사용한 방법에 대한 설명과 두 이진 SD 수의 덧셈을 수행하는 새로운 이진 SD 덧셈 규칙을 제안하고 이를 이용하여 새로운 RBA를 설계한다. 그리고 제안하는 RBA의 속도를 McIvor의 2방법과 기존의 RBA[1,6,7]와 비교하고 3장에서 결론을 내린다.

II. Montgomery 곱셈 구조

1. Montgomery 곱셈 알고리즘

M을 n-비트 모듈러라 하자. 주어진 M보다 작은 정수 U, V에 대하여 Montgomery 곱셈을 수행하기 위해서 $R=2^n$을 이용하여 식 (1)을 통해 Montgomery 도메인의 원소인 X, Y로 변환시킨다.

$$X=U\times R(\mathrm{mod}M),\ Y=V\times R(\mathrm{mod}M) \quad (1)$$

R^{-1}을 모듈러 M에 대하여 R의 역원이라 할 때, X, Y의 Montgomery 곱셈은 식 (2)와 같이 정의한다.

$$Mont(X,Y)=X\times Y\times R^{-1}=U\times V\times R(\mathrm{mod}M) \quad (2)$$

곱셈을 단지 한 번 수행한다고 할 때 이 과정에서 소요되는 복잡도는 일반적인 곱셈에 비해 크다. 그러나

알고리즘 1 : 몽고메리 곱셈(X, Y, M)

입력 : $X=(x_{n-1}\cdots x_1x_0)_2$, $Y=(y_{n-1}\cdots y_1y_0)_2$, $M=(m_{n-1}\cdots m_1m_0)_2$, $R=2^n$, $\gcd(M,R)=1$,

$0\le X, Y< M$.

출력 : $A=XYR^{-1}\mathrm{mod}M=(a_{n-1}\cdots a_1a_0)_2$.

1. $A\leftarrow 0$.
2. For i from 0 to $n-1$ do :
   2.1. $u_i\leftarrow(a_0+x_iy_0)\bmod 2$.
   2.2. $A\leftarrow(A+x_iY+u_iM)/2$.
3. If $A\ge M$ then $A\leftarrow A-M$.
4. Return(A).

Montgomery 곱셈은 RSA 암호 시스템에서의 지수승 연산과 같이 반복적인 모듈러 곱셈을 요구되는 환경에서 효율적이다. 본 논문에서 이진 Montgomery 곱셈 알고리즘은 알고리즘 1과 같이 구성된다. Montgomery 곱셈 알고리즘을 보다 효율적으로 구성하여 복잡도를 줄이기 위해 다양한 방법들이 제안되었으며 이는 다음과 같다: 1) 반복적인 곱셈으로 이루어진 지수승의 경우 단계 3에서 수행되는 비교 연산과 뺄셈은 큰 연산량을 차지하게 되므로 단계 3을 제거하는 방법[9]; 2) Y를 한 비트 늘려 최하위 비트를 0으로 만들어 알고리즘 1의 단계 2.1에서 u_i를 계산할 때 중간 결과 A의 최하위 비트만으로 M의 덧셈 여부를 결정해 주는 방법[10]이 있다. Walter는 X와 Y의 범위를 기존의 $0\le X, Y< M$에서 $0\le X, Y< 2M$으로 대체하는 방법을 제안했다[9]. 이는 입력 값과 출력 값의 크기를 동일하게 하므로 출력 값을 다음 단계의 입력 값하여 반복적인 곱셈 연산을 수행할 수 있도록 한다. 따라서 X와 Y는 한 비트 늘어난 $n+1$비트가 되고, 반복 연산이 증가한다. 중간 결과 A의 크기를 $0\le A< 2M$로 만들기 위해 X의 최상위 비트 x_{n+1}을 0이라 하고 반복 연산을 한 번 더 수행한다. 알고리즘 1에서 단계 3을 제거하기 위해 반복 연산이 두 번 더 수행되지만 곱셈기의 구성상 기존 방법 보다 효율적이다[11].

2. CSA를 이용한 Montgomery 곱셈기

알고리즘 1에서 세 연산자의 덧셈과정에서 발생하는 캐리 전파를 처리하는 과정이 알고리즘 지연 시간의 대부분을 차지하므로, 알고리즘 1에서 단계 2.2의 세 입력 값에 대한 덧셈을 수행하는 과정이 알고리즘 1의 시간 복잡도에서 가장 큰 부분을 차지한다. 이 문제를 해결하기 위해 세 개의 입력 값에 대하여 덧셈을 수행

하고 두 개의 값(캐리: c_{i+1}, 합: s_i)을 출력하는 덧셈기인 CSA를 이용한 방법이 제안되었다[3].

알고리즘 1의 단계 2.2에서 $(A+x_iY+u_iM)$의 계산 결과는 CSA를 이용하여 캐리 A_C와 합 A_S으로 나타낼 수 있고, $A=(A_C+A_S)$는 다음 곱셈의 입력으로 사용되므로 Y 또한 $Y=(Y_C+Y_S)$으로 표현 한다. 따라서 $(A+x_iY+u_iM)$의 연산은 x_i를 X의 i번째 비트라 할 때 $((A_C+A_S)+x_i(Y_C+Y_S)+u_iM)$으로 나타낼 수 있으므로 다섯 개의 입력 값과 두 개의 출력 값을 갖는 McIvor의 1 방법이 된다[3].

$((A_C+A_S)+x_i(Y_C+Y_S)+u_iM)$의 연산에서 x_i와 u_i의 값에 따라 덧셈에 필요한 피 연산자의 조합은 달라진다. $x_i=u_i=0$인 경우 단지 (A_C+A_S)의 연산이 되고 $x_i=1, u_i=0$인 경우 $(A_C+A_S)+(Y_C+Y_S)$의 연산이 되며 $x_i=0, u_i=1$의 경우 $(A_C+A_S)+M$의 연산이 된다. 하지만 $x_i=y_i=1$의 경우 다섯 개의 피 연산자로 구성 된 $(A_C+A_S)+(Y_C+Y_S)+M$의 연산을 수행하게 된다. 피 연산자의 항의 개수가 증가하는 문제점을 해결하기 위하여 사전 연산을 통해 항의 개수를 줄인 McIvor의 2 방법이 제안되었다. 곱셈이 시작하기 전 사전 연산을 통하여 $(Y_C+Y_S)+M$의 연산을 수행하고 그 결과 캐리 D_C와 합 D_S를 저장한 후 MUX를 이용하여 x_i와 u_i의 경우에 선택적으로 사용하는 방법이다.

3. 제안하는 RB 곱셈기

RB 몽고메리 곱셈기는 이진 SD 수 체계를 사용하여 몽고메리 곱셈을 수행하는 곱셈기를 지칭한다. 본 논문에서 사용되는 $\wedge$, $\vee$, $\bar{}$는 각각 AND 연산, OR 연산, NOT 연산으로 정의한다.

1. 이진 SD 수 체계에서의 RBA

이진 SD 수 체계를 사용하여 설계되고 있는 대부분의 RBA들은 $0,1,\bar{1}$에 대해서 각각 [0,0] 또는 [1,1], [1,0], [0,1]로 표현하고 (x_i^+,x_i^-)로 입력 값 및 출력 값을 나타낸다. 4개의 XOR 게이트와 두 개의 MUX를 사용하여 덧셈을 설계한 RBA1[1]의 경우 NAND 게이트에 비해 약 3배 큰 지연 시간을 가진 XOR 게이트 3단계로 구성되어 있으며 RBA2[6]는 주 입력 4비트, 캐리 입력 3비트, 캐리 출력 3비트, 주 출력 2비트로 단위 셀로는 입력과 출력의 개수가 많았고 NAND 게이트 2단계, XOR 게이트 2단계, NOT 게이트 1단계로 구성되어 있다. RBA3[7]는 다중 입력 게이트를 활용하여 속도를 향상시키고자 하였다. 그 결과 XOR 게이트 2단계, XNOR 게이트 1단계, NAND 게이트 1단계, OA21 게이트 1단계로 구성되어 있다. 또한 [1,6,7]의 RBA 중 삼성 STD130 0.18μm 1.8V 표준 셀 라이브러리에서 지원하는 게이트로 설계하여 지연 시간을 측정한 결과 가장 빠른 속도를 나타내었다.

곱셈기의 임계 경로 지연 시간 측면에서 세 피연산자의 덧셈의 효율적인 구성은 중요한 문제이다. 특히 이진 수 체계에서 지수승 연산에 적합하도록 McIvor가 제시한 구조는 이러한 측면을 고려할 때 효

알고리즘 2 : NRBA(X, Y)

입력 : $X=(X^S,X^V)=(x_{n+1}^S\cdots x_1^Sx_0^S, x_{n+1}^V\cdots x_1^Vx_0^V)$, $Y=(Y^S,Y^V)=(y_{n+1}^S\cdots y_1^Sy_0^S, y_{n+1}^V\cdots y_1^Vy_0^V)$.

출력 : $A=(A^S,A^V)=(a_{n+2}^S\cdots a_1^Sa_0^S, a_{n+2}^V\cdots a_1^Va_0^V)=X+Y$.

1. $A=(A^S,A^V)\leftarrow 0$.
2. Process : $(x_i^S,x_i^V,y_i^S,y_i^V)$

   Begin

   2.1. $c_{i+1}\leftarrow\overline{((x_i^S\vee x_i^V)\vee\overline{(x_i^S\vee y_i^S)})}$. $s_i\leftarrow\overline{\overline{(\overline{(x_i^V\wedge y_i^S)}\vee\overline{(x_i^V\vee y_i^S)})}}$.

   2.2. $(a_i^S,a_i^V)\leftarrow\overline{\overline{\overline{\overline{\overline{\overline{(s_i\wedge y_i^N)}\vee\overline{(s_i\vee y_i^N)})\wedge c_i)}}\vee\overline{\overline{\overline{\overline{(s_i\wedge y_i^N)}\vee\overline{(s_i\vee y_i^N)}}\vee\overline{c_i}}}}}$,
   $\overline{\overline{\overline{\overline{(s_i\wedge y_i^N)}\wedge\overline{(s_i\wedge c_i)}}\vee\overline{\overline{(s_i\wedge c_i)}\wedge\overline{(y_i^N\wedge c_i)}}}}}$.

   End process.

3. Return $A=(A^S,A^V)$.

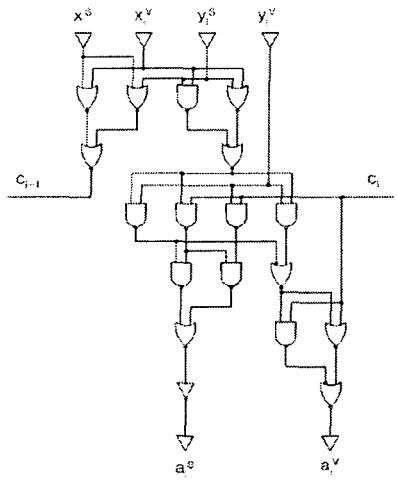


그림 1 제안하는 RBA의 하드웨어 구조

율적이다. 본 논문에서는 덧셈기의 효율적인 구성을 위하여 이진 SD 수 체계를 사용한다.

알고리즘 1의 단계 2.2에서 $(A+x_iY+u_iM)$의 연산은 x_iY+u_iM을 한 개의 값으로 표현하면 두 개의 피연산자 덧셈으로 표현이 가능하지만, 양수인 두 값에 대하여 이진 수 체계에서는 캐리가 발생한다. 따라서 캐리 전파 제거 특성을 가진 이진 SD 수 체계를 사용하면 캐리 전파 없이 덧셈을 수행할 수 있다.

제안하는 RB 곱셈기는 McIvor의 구조와 유사한 형태로 이진 SD 수로 표현된 X, Y와 이진 수로 표현된 M을 입력받아 $Y+M$을 계산하여 저장하고 x_i와 u_i에 따라 MUX를 통해 $\{Y+M, Y, M, 0\}$에서 선택하여 덧셈을 수행한다.

2. 제안하는 RB 몽고메리 곱셈기

본 논문에서는 기존의 덧셈 규칙을 따르지 않고 고속 연산에 적합하도록 새로운 덧셈 규칙을 제안하고 이를 이용하여 새로운 RBA를 제안한다. 제안하는 새로운 RBA는 알고리즘 2와 같다.

제안하는 덧셈기를 NRBA(New Redundant Binary Adder)라 하자. 이진 SD 수는 부호를 표현해야 하므로 x_i, y_i를 부호 비트와 값 비트를 사용하여 각각 $[x_i^S, x_i^V]$와 $[y_i^S, y_i^V]$와 같이 나타내고 이진 SD 수는 $(1=[0,1], 0=[0,0], \bar{1}=[1,1])$로 나타낸다.

이진 SD 수를 S, 이진 수를 B, 0과 $\bar{1}$로 이루어진 수를 N이라 할 때, 기존의 RBA는 두 S를 입력받아 S를 출력하는 구조이다. 그러나 NRBA의 입력 값은 S, B, N의 형태이고 출력 값은 B, N의 형태이다. B와 N은 각각 한 비트로 표현이 가능하므로 기존의 RBA와 입출력 크기가 동일하다. NBRA는 두 단계로 구성되며 S+B를 먼저 수행하고 S+B의 결과와 N의 덧셈을 수행한다.

$$S^S+S^V+B+N \Rightarrow B+N+N \Rightarrow N+B$$

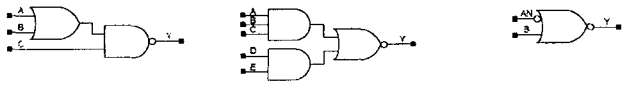
본 논문에서 사용하는 삼성 STD130 0.18μm 1.8V 표준 셀 라이브러리에서는 NAND 게이트나 NOR 게이트에 비해 AND 게이트, OR 게이트, XOR 게이트 등은 크기가 크고 지연 시간을 크게 증가시키므로 NAND 게이트, NOR 게이트, NOT 게이트 등으로만 구성된 하드웨어 구조를 설계하기 위하여 입력 값과 출력 값을 다음과 같이 바꿀 수 있다.

$$S^S+\overline{S^V}+\overline{B}+N \Rightarrow \overline{B}+N+N \Rightarrow N+\overline{B}$$

일반적인 입력 값과 출력 값을 고려한 경우 알고리즘으로 나타내면 하드웨어 구조는 그림 1과 같다. 그림 1에서 전체 비트를 고려한 입력 값 X^S, X^V, Y^S, Y^V는 각각 $S^S, \overline{S^V}, \overline{B}, N$의 형태가 된다. 출력 값 A^S, A^V 또한 동일한 방법으로 고려하면 각각 $N, \overline{B}$의 형태가 된다. 중간

표 1 삼성 STD130 0.18μm 1.8V 표준 셀 라이브러리에서 지원하는 게이트를 이용한 덧셈기 비교

	지연 시간(ns)	지연 시간 구성	게이트 구성
McIvor 2 방법	0.722	FA 2단계	2FA
RBA1[11]	0.676	XOR 3단계	4XOR + 2MUX
RBA2[6]	0.661	NAND 2단계 + XOR 2단계 + NOT 1단계	6NAND + 2XOR + 1MUX + 2NOT
RBA3[7]	0.634	XOR 1단계 + XNOR 1단계 + NAND 1단계 + OA21* 1단계	2NAND + 1NOR + 1XOR + 1XNOR + 1AO32** + 1OA21 + 1NOT
NRBA	**0.555**	**NOR 3단계 + NORb*** 3단계**	**7NOR + 7NAND + 3NORb + 1NOT**



OA21* AO32** NORb***

결과의 캐리 C는 $\overline{B}$의 형태를 가진다.

3. 제안하는 RB 곱셈기의 효율성

제안하는 RB 곱셈기의 속도는 임계 경로 지연 시간에 따라 클럭의 주기가 결정되므로 가장 많은 시간이 소요되는 덧셈기의 효율성에 크게 의존하게 된다. 삼성 STD130 0.18μm 1.8V 표준 셀 라이브러리에서 지원하는 게이트[13]를 이용하여 McIvor의 2 방법 및 기존의 RBA 등과 속도를 비교하면 표 1과 같이 제안한 RBA는 McIvor 방법과 기존의 RBA들에 비해 최소 12.46%의 속도 향상을 보였다.

III. 결론

제안하는 곱셈기는 연산 속도를 향상시키기 위해 이진 SD 수 체계를 사용하여 임계 경로 지연 시간의 대부분을 차지하는 세 피연산자의 덧셈을 효율적으로 구성했다. 기존의 이진 SD 덧셈 규칙을 사용하지 않고 새로운 방법을 제안하여 속도를 향상시켰다. 삼성 STD130 0.18 μm 1.8V 표준 셀 라이브러리에서 지원하는 게이트로 덧셈을 설계 할 경우 기존의 RBA보다 12.46%의 속도 향상을 보였다. 이진 SD 수 체계는 캐리 전파 제거 특성 이외에도 음수를 표현하는 경우 2의 보수나 1의 보수 등으로 변환 시 요구되는 연산량이 필요하지 않고 바로 변환이 가능하므로 장점을 가진다. 본 논문에서 제안한 곱셈기는 연산을 빠르게 수행해야 하고 충분한 게이트들을 사용할 수 있는 서버 등에서 효율적이라 예상되며 여러 응용 분야에 활발히 사용될 수 있을 것으로 보인다.

참고문헌

[1] 홍종욱, "Redundant Binary 연산을 이용한 실수/복소수 승산기", *연세대학교 대학원 석사 학위 논문*, 전기・컴퓨터 공학과, 1999.

[2] S. E. Eldridge, C. D. Walter, "Hardware implementation of Montgomery's modular multiplication algorithm", *IEEE Transaction on Computers*, Vol. 42, pp. 693-699, July 1993.

[3] Ciaran McIvor, Maire McLoone, John V McCanny, Alan Daly, "Fast Montgomery modular multiplication and RSA cryptographic processor architectures", *ACCSC 2003*, pp 379-384, 2003.

[4] A. Avizienis, "Signed-digit number representations for fast parallel arithmetic", *IRE Trans. Electron. Comput.*, vol. EC-IO, no. 9, pp. 389-400, Sept. 1961.

[5] Naofumi Takagi, et. al., "High-Speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree", *IEEE Trans. on Computers*, Vol. C-34, No. 9, pp. 789-796, Sep. 1985.

[6] H. Makino,Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara, and K. Mashiko, "An 8.8-ns 54 × 54 bit multiplier with high speed redundant binary architecture", *IEEE J. Solid-State Circuit*, vol. 31, no. 6, pp. 773-783, June 1996.

[7] H. Edamatsu, T. Taniguchi, T. Nishiyaina and S. Kuninobu, "A 33 MFLOPS floating point processor using redundant binary representation", *Dig. Tech. Papers of 1988 ISSCC*. pp. 152-153, Feb. 1988.

[8] B. Kaliski, "TWIRL and RSA Key Size", *RSA Labs Tech Note*, May 2003.

[9] Walter C. D., "Montgomery Exponentiation Needs No Final Subtractions", *Electronics Letters*, 35(21) : pp. 1831-1832, 1999.

[10] Manochehri, K, Pourmozafari. S, "Modified radix-2 Montgomery modular multiplication to make it faster and simpler", *ITCC 2005*. pp. 598-602, 2004.

[11] Christof Paar, Thomas Blum, "High radix Montgomery modular exponentiation on reconfigurable hardware", *IEEE Transactions on Computers*, vol. 50, No. 7, pp. 759-764, 2001.

[12] I. Koren, "Computer Arithmetic Algorithms", *Englewood Cliffs*, NJ:Prentice-Hall, 1993.

[13] SAMSUNG STD130 0.18㎛ 1.8V CMOS Standard Cell Library for Pure Logic Products.