Radix-4 Modified Booth 알고리즘과 CSA를 이용한 고속 RSA 암호시스템의 FPGA 구현

박 진 영, 서 영 호, 김 동 욱
광운대학교 전자재료공학과
전화 : 02-940-5167 / 핸드폰 : 011-375-7975

FPGA Implementation of High Speed RSA Cryptosystem Using Radix-4 Modified Booth Algorithm and CSA

Jin Young Park, Young Ho Seo, Dong Wook Kim
Dept. of Electronic Material , Kangwoon University
E-mail : codesign@explore.kangwoon.ac.kr

Abstract

This paper presented a new structure of RSA cryptosystem using modified Montgomery algorithm and CSA(Carry Save Adder) tree. Montgomery algorithm was modified to a radix-4 modified Booth algorithm. By appling radix-4 modified Booth algorithm and CSA tree to modular multiplication, a clock cycle for modular multiplication has been reduced to (n+3)/2 and carry propagation has been removed from the cell structure of modular multiplier. That is, the connection efficiency of full adders is enhanced.

I. 서론

현재 유/무선 인터넷과 개방형 네트워크가 발달함에 따라서 네트워크상에서 생성되거나 전달 혹은 저장되는 정보의 안전성과 보호가 매우 중요시되고 있다. 이때 정보보호는 컴퓨터와 각종 통신을 사용하는 모든 분야, 이를테면 위성통신, CATV 등을 비롯하여 각종 통신 이용 산업 및 인터넷, 전자문서 교환을 포함하는 전자상거래 등 거의 모든 정보통신 관련 사업 분야를 망라한다. 또한 스마트카드의 사용과 같은 새로운 형태의 데이터 통신에서 정보의 교환의 안정성이 매우 중요시되고 있다. 이러한 정보 보호기능의 문제를 해결하기 위해 암호화 알고리즘과 그것의 효율적인 구현에 대한 연구가 활발히 이루어지고 있다. 특히 공개키 구조의 RSA 암호 시스템은 SET(Secure Electronic Transaction), SSL(Secure Socket Layer) 등의 프로토콜, 가상가설망, X.509 인증메카니즘, 그리고 PGP(Pretty Good Privacy) 및 PEM(Privacy Enhance Mail)등의 전자우편 보안을 비롯한 여러 분야에서 활발히 사용되고 있고, 계속적으로 개선되고 있다.

본 논문은 RSA 암호 시스템에서 Montgomery 알고리즘[1]에 radix-4 modified Booth 알고리즘[2]을 적용하여 모듈러 곱셈과정에 있어서 클럭 주기를 반으로 줄이고, 모듈라 곱셈기를 구성하는 셀 구조에 있어서 carry 전파를 제거하기 위해 4단계의 CSA(Carry Save Adder) Tree 구조[2]의 도입을 제안한다. 즉, 기존방식에서 보이고 있는 각 단위 셀 내부의 전가산기 연결에 효율성을 높이는 것이 본 논문의 목적이다.

II. Radix-4 Modified Booth 알고리즘과 CSA를 이용한 RSA

2.1 RSA 암호 시스템

RSA 연산은 암호화와 전자서명 모두를 제공할 수

있으며, 소인수분해의 어려움에 안전도의 근거를 두고 있다. RSA 암호시스템은 n-bit(일반적으로 1024 bit를 사용하고 스마트 카드에서는 2048 bit까지 사용하고 있다)의 길이를 갖는 평문 메시지 블록(M), 암호화 키(E), 복호화 키(D), 모듈러스(N)를 사용하여 암호화와 복호화 연산을 수행한다. 이 암호 시스템 알고리즘은 모듈러 지수 연산이 주를 이루고, 모듈러 지수 연산은 모듈러 곱셈의 반복으로 수행된다.

2.2 Radix-4 Modified Booth Algorithm을 이용한 모듈러 곱셈기 알고리즘

RSA 암호 시스템의 모듈러 지수 연산은 많은 모듈러 곱셈 연산의 반복으로 수행되기 때문에 암호화의 연산 시간이 대칭키 알고리즘에 비교하여 매우 늦어진다. 이를 개선하기 위해 1985년에 P. L. Montgomery는 모듈러 곱셈의 효율적인 알고리즘을 제안했었다[1]. 그 후 많은 개선된 Montgomery 알고리즘들이 제안되어 연산 속도를 향상시켰다[3][4].

본 논문에서는 모듈러 곱셈의 반복의 수를 줄이기 위해 radix-4 Booth 알고리즘을 이용한 radix-4 Montgomery 알고리즘[4]에 4단계 CSA tree 구조를 적용하여 carry 전파를 제거하고 CSA의 각 단계 사이에 레지스터를 삽입하여 Y(n-bit) × X(n-bit) 모듈러 곱셈의 전체 $\lceil (n+3)/2 \rceil$ 클럭에서 매 클럭마다 중간 결과를 병렬로 출력할 수 있도록 하였고, 또한 덧셈 연산에서 CSA를 사용할 경우 모듈러 곱셈의 마지막 클럭 후에 일반적으로 CPA (Carry Propagation Adder)를 사용하는데[5] 이것을 제거하기 위해 다음과 같이 알고리즘을 개선하였다.

Modified Radix-4 Montgomery Algorithm(Y, X, N)

```
{
-- Output R = (A+B) · X · 4^-⌈(n+3)/2⌉ (mod N)
-- nY = nA(carry) + nB(sum)
-- Ri = carry + sum
   nn = ⌈ (n+3)/2 ⌉ ;
   R0 = 0 ;
   for(i=0 ; i<nn ; i++) {
      qi = Ri+nY (mod 4) ;
      Ri+1 = (Ri + nY + qiN)/4 ;
   }
}
```

개선된 Radix-4 Montgomery 알고리즘의 하드웨어구조가 그림 1에 보이고 있고 nY 신호 선택을 위한 radix-4 Modified Booth 알고리즘 recoding 규칙은 표 1에서 보이고 있다.

2.3 제안된 모듈러 곱셈기의 하드웨어 구조

Radix-4 모듈러 곱셈기 구조에서 CSA를 사용할 경우 모듈러 곱셈의 마지막 과정의 출력인 carry와 sum을 더하기 위해 일반적으로 n-bit CPA를 사용하여 매우 큰 연산 지연 시간을 갖는다. 이 지연 시간을 제거하기 위해 carry와 sum의 두 출력을 모듈러 지수 연산의 입력 값으로 사용하였고, 이 두 출력을 모듈러 곱셈기의 partial product(nA,nB)에 저장한 후 Booth recoding을 통하여 CSA tree의 입력 값으로 사용하였다.

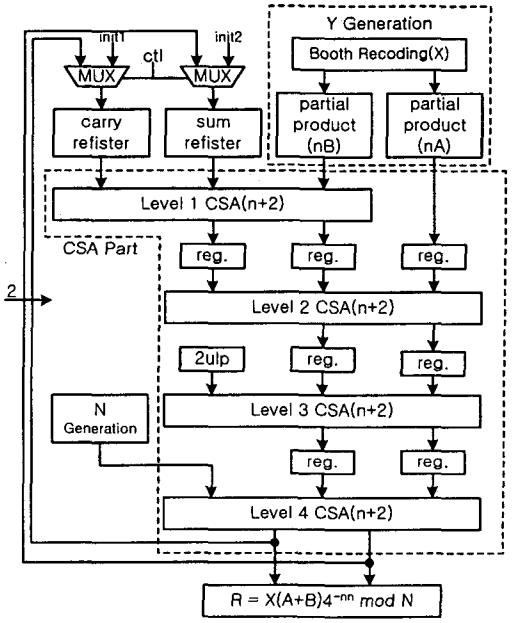

그림 1. CSA를 이용한 Radix-4 모듈러 곱셈기 구조

표 1. Radix-4 Modified Booth's 알고리즘

X_{i+1}	X_i	X_{i-1}	Booth Code	연산
0	0	0	00	0
0	1	0	01	+Y
1	0	0	1'0	-2Y
1	1	0	01'	-Y
0	0	1	01	+Y
0	1	1	10	+2Y
1	0	1	01'	-Y
1	1	1	00	0

그림 1의 CSA 부분은 CSA의 특징인 n-bit의 두 수를 덧셈하기 위해 CSA를 n번 반복하면 carry 전파 없이 덧셈결과를 얻는 원리를 이용하여 4단계의 CSA를 수행하고 모듈러 곱셈기의 매 중간 출력 carry와 sum을 직접 2bit 오른쪽 shift 연산하여 feedback할 수

있도록 하였다. 이때 4단계의 CSA를 사용하지 않는다면 2bit 오른쪽 shift 연산을 위해 carry와 sum의 LSB(Last Significant Bit) 2bit 덧셈에 의한 carry 전파 때문에 1024bit CPA 연산이 필요되므로 4단계 CSA를 사용하여 이 부분의 큰 지연시간을 제거할 수 있는 구조로 구현하였다. 또한 CSA의 각 단계마다 레지스터를 삽입하여 파이프라인이 가능하므로 radix-4 모듈러 곱셈기 구조는 4개의 모듈러 곱셈을 병렬적으로 처리할 수 있다. 따라서 Y(n-bit) × X(n-bit) 모듈러 곱셈은 전체 클럭에서 매 클럭 마다 R = (R+ nY + q_iN) / 4 연산은 하나의 전가산기를 수행하는 것과 동일한 지연시간(τ)을 갖고, 모듈러 지수 연산 과정은 n번 반복하므로 모듈러 곱셈기의 전체 클럭 주기 $\lceil (n+3)/2 \rceil$에 의해 RSA 암호 시스템의 전체 지연시간은 약 $n \cdot (n+3)/2 \cdot \tau + \beta$로 동작할 수 있도록 구현하였다. β(약 $4n \cdot \tau$)는 모듈러 지수 연산 입출력 영역에서 발생되는 약간의 지연시간이다. 4개의 모듈러 곱셈 입력으로 모듈러 지수 연산에서 모듈러 제곱 연산과 모듈러 단순 곱셈의 두 부분이 RL (Right-to-Left) 이진 방식에 의해 병렬로 처리되므로 2개의 모듈러 곱셈이 존재한다. 또한 RSA 암호 시스템은 임의의 길이 평문을 n-bit 블록으로 분리하여 모듈러 지수 연산의 입력으로 사용하기 때문에 본 논문에서는 2개의 n-bit 블록을 연산의 입력으로 사용한다. 따라서 총 4개의 모듈러 곱셈의 입력이 가능하여 향상된 속도로 암호화/복호화가 가능하다.

모듈러 곱셈기에 radix-4 modified Booth's 알고리즘을 이용하였을 때, 표 1에서 처럼 0, ±Y, ±2Y의 부분 곱이 발생하여 n+2 bit CSA 연산을 한다. MSB(Most Significant Bit) 2bit는 부호 bit로 사용되고 모듈러 곱셈의 중간 출력 값이 feedback되어 입력 값으로 들어갈 때 부호 bit가 확장된다. 그림 1의 Y 생성 부분은 Booth recoding 규칙에 의해 Y값 선택 신호(en[0..3])를 그림 4의 CSA 연산 부분과 독립적으로 생성하여 그림 2의 모듈을 통해 nY를 선택한다. 이 때 음수 값이 발생될 경우, 그 처리는 부분 곱 선택 모듈에서 XOR 연산에 의해 1의 보수화가 되고 단계 3의 CSA에서 2ulp(Unit in the last position)를 입력함으로써 2의 보수화를 수행하도록 구현하였다. 그리고 N 생성 부분은 2클럭 동안 qi를 계산하고 1클럭 동안 qi 신호에 의해 그림 3의 모듈을 통해 N의 값이 선택된다.

그림 4는 그림 1의 4단계 CSA tree 부분을 (2,2),(3,2) counter cell, 즉 반 가산기와 전 가산기 cell들로 나열하여 4bit RSA 암호 시스템의 모듈러 곱셈기(MRMA, Modified Radix-4 Montgomery Algorithm)를 보이고 있다. 이 cell들을 확장함으로써 1024bit RSA 암호 시스템을 구현하였다.

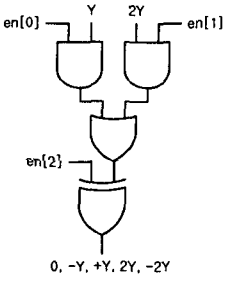

그림 2. 부분곱 선택모듈

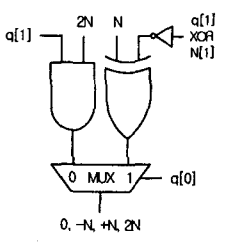

그림 3. 모듈러 수 선택 모듈

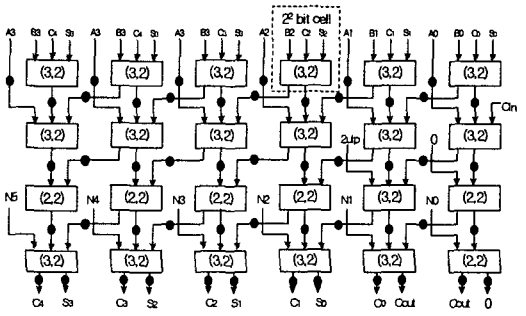

그림 4. 파이프라인된 4단계 CSA 형태의 MRMA

2.4 모듈러 지수 연산 알고리즘과 하드웨어 구조

모듈러 곱셈기를 호출하여 모듈러 제곱 연산과 모듈러 단순 곱셈을 병렬적으로 처리할 수 있는 RL 이진 방식을 이용한 모듈러 지수 연산 알고리즘은 다음과 같다.

Modular Exponentiation Algorithm(M,E,N) {
-- Output : ME = M^E (mod N)
-- Function : MRMA(X,Y,N)
-- Constant : C = $4^{2\lceil (n+3)/2 \rceil}$ (mod N)
  S_0 = MRMA(M,C,N) ;
  T_0 = MRMA(1,C,N) ;
  for (i=0 ; i<n-1 ; i++) {
    S_{i+1}=MRMA(S_i, S_i, N) ;
    T_{i+1}=MRMA(S_i, $e_i \times T_i$, N) ;
  }
  ME = MRMA(T_n, 1, N) ;
  if (ME<0) ME = N+ME ;
}

모듈러 제곱 연산과 모듈러 단순 곱셈은 서로 동작적으로 중첩되면서 파이프라인 되므로 실제 동작에 있어

서 병렬적이다. 모듈러 곱셈기 MRMA함수의 입력값과 함수값은 carry와 sum의 두 개의 값으로 입출력된다. 그리고 모듈러 지수 연산의 마지막 과정에서 M^E(mod N)을 출력하기 위해 두 개의 2bit CPA를 이용하여 carry과 sum을 더하고 radix-4 Booth 알고리즘을 사용했기에 음수 값이 발생될 경우 다시 그 결과에 N을 더하여 매 클럭 마다 1-bit의 암호문(또는 평문)을 연속적으로 출력한다. 따라서 본 논문에서 제안한 RSA 암호 시스템은 한 개의 모듈러 곱셈기(MRMA)와 제어부, 추가적으로 모듈러 N 생성 부분과 부분 곱 생성 부분 그리고 두개의 CPA(2bit)로 구현되었다. 그 구조는 그림 5와 같다.

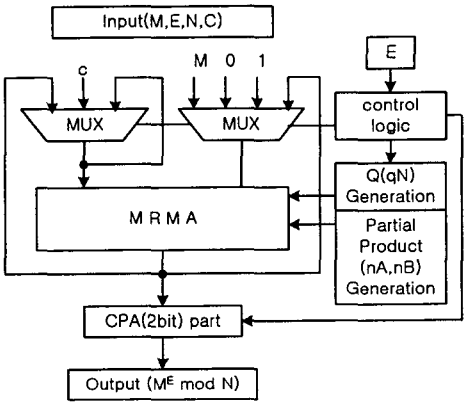

그림 5. 모듈러 지수 연산 구조

III. 실험 결과

본 논문에서 구현된 RSA 하드웨어는 칩의 면적에 있어서 CSA tree에서 n+2 bit 전가산기 3개, 반 가산기 1개와 각 단계의 레지스터를 사용하므로 기존에 제안된 CSA 구조가 개선되었고 면적에 있어서는 레지스터 부분만 추가되었다. 하지만 현재까지 제안된 구조와 하드웨어 처리속도를 비교한다면 매우 효율적이다. 모듈러 곱셈의 승수는 bit-serial로 피승수는 bit-parallel로 입력되어 연산이 이루어지고 레지스터와 시스템 제어부가 단순화되어 Altera의 FLEX10KA EPF250ABC600-1 칩에서 97%(11869/12160)의 자원 사용률을 보였다. 전체 하드웨어가 파이프라인되어 100MHz에서 동작함을 확인하였다. 하드웨어 구조적 지연시간이 4ns로 예상되나 FPGA 내부적인 블록 구성의 특성 때문에 부가적인 지연시간이 추가된 것으로 보인다. 또한 조합회로에 비해 상대적으로 레지스터를 많이 사용하므로 레지스터에 의한 셀의 점유에 의해 FPGA 칩 내에서 많은 자원 사용률을 보이지만 추후 ASIC으로 구현된다면 많은 면적의 최적화가 이루어 질 것으로 예상된다.

VI. 결론

본 논문은 비 대칭키 암호 알고리즘으로 가장 널리 사용되고 있는 RSA 암호 알고리즘을 하드웨어로 구현하였다. 그 구현에 있어 암호화 속도를 향상 하기 위해 단순 곱셈 부분에 radix-4 Booth recoding을 적용하여 부분 곱의 수를 약 n/2로 감소시켜 모듈러 곱셈에 적용함으로써 모듈러 연산의 반복 횟수 또한 $\lceil (n+3)/2 \rceil$로 줄이고, 그 과정에서 요구되는 덧셈 연산 부분에 CSA tree를 사용하여 carry 전파를 제거함과 동시에 파이프라인 하여 데이터 출력률을 향상 시켰으며, 또한 모듈러 곱셈기에서 CPA를 제거하여 RSA 암호 시스템의 연산 속도를 향상하였다.

본 논문은 기존 방식에 비해 임계 경로 길이(critical path)가 감소하여 향상된 속도를 가지고 최적화된 면적을 가지므로 추후 전자상거래를 비롯한 여러 응용분야에 적용이 가능할 것으로 사료된다.

참고문헌

[1] P. L. Montgomery, "Modular multiplication without trial division," Math. Computation, vol. 44, pp.519-521, 1985.

[2] Behrooz Parhami, "Computer arithmetic algorithms and hardware design," Oxford university press, 2000

[3] J. H. Guo, C. L. Wang, H. C. Hu, "Design and implementation of an RSApublic-key cryptosys tem," Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on , Volume 1,pp.504 -507, 1999

[4] J. H. Hong, C. W. Wu, "Radix-4 modular multiplication and exponentiation algorithms for the RSA public-key cryptosystem," Design Automation Conference, 2000. Proceedings of the ASP-DAC 2000. Asia and South Pacific, pp.565 -570, 2000

[5] Y. S. Kim, W. S. Kang, J. R. Choi, " implementation of 1024-bit modular processor for RSA cryptosystem,"ASICs, 2000. AP-ASIC 2000. Proceedings of the Second IEEE Asia Pacific Conference on , pp187 -190. 2000