論文2002-39SD-8-5

가상 캐리 예측 덧셈기와 PCI 인터페이스를 갖는 분할형 워드 기반 RSA 암호 칩의 설계

(A Scalable Word-based RSA Cryptoprocessor with PCI Interface Using Pseudo Carry Look-ahead Adder)

權 宅 元 *, 崔 峻 林 **

(Taek-Won Kwon and Jun-Rim Choi)

요 약

본 논문에서는 가상 캐리 예측 덧셈기(pseudo carry look-ahead adder)를 사용하여 분할형 워드 기반 RSA의 구현에 관한 방법을 제안하고 검증하였다. 효율적인 모듈라 곱셈기의 설계를 위해 병렬 2단 CSA(carry-save adder) 구조를 사용하였으며 마지막 덧셈의 고속 처리를 위하여 캐리 발생과 지연시간이 짧은 가상 캐리 예측 덧셈기를 적용하였다. 제안한 모듈라 곱셈기는 분할형 워드를 기반으로하여 다음 모듈라 연산을 위해 매 클럭마다 쉬프트와 정렬 연산이 필요없기 때문에 하드웨어를 줄일 수 있으며 고속 모듈라 곱셈 연산을 가능하게한다. 제안한 연산 구조를 PCI 인터페이스를 갖는 FPGA로 기능을 검증한 후 0.5 ㎛ 삼성 gate array 공정을 사용해서 256 워드 모듈라 곱셈기를 기반으로 한 1024-bit RSA 암호프로세서를 단일 칩으로 구현하였다.

Abstract

This paper describes a scalable implementation method of a word-based RSA cryptoprocessor using pseudo carry look-ahead adder. The basic organization of the modular multiplier consists of two layers of carry-save adders (CSA) and a reduced carry generation and propagation scheme called the pseudo carry look-ahead adder for the high-speed final addition. The proposed modular multiplier does not need complicated shift and alignment blocks to generate the next word at each clock cycle. Therefore, the proposed architecture reduces the hardware resources and speeds up the modular computation. We implemented a single-chip 1024-bit RSA cryptoprocessor based on the word-based modular multiplier with 256 datapaths in 0.5㎛ SOG technology after verifying the proposed architectures using FPGA with PCI bus.

Key Word : RSA, word-based modular multiplier, pseudo carry look-ahead adder

* 學生會員, 慶北大學校 電子工學科
(School of Electrical Engineering, Kyungpook National University)
** 正會員, 慶北大學校 電子電氣工學部
(School of Electrical Engineering, Kyungpook National University)
※ 본 논문은 ITRC 사업과 IDEC 사업의 지원으로 수행되었다.
接受日字:2002年1月30日, 수정완료일:2002年6月25日

Ⅰ. 서 론

개방 네트워크의 신뢰성과 안전성을 제공하는 많은 기술들이 있지만 그 중에서도 공개키 암호 기술은 소인수분해의 어려움에 안전도의 근간을 갖고 있으며 암호화와 전자 서명을 위한 RSA[1] 암호 시스템이 가장 널리 사용되고 있다. RSA 암호 시스템은 안전도를 높이기 위해 키 값을 1024 비트 이상으로 높이고 있는 추세이며 이러한 키 값으로 인해 반복적인 모듈라 곱

셈 연산 구조를 갖는 하드웨어 구현을 매우 어렵게 만들고 있다. 이러한 문제를 해결하기 위해 몽고메리 모듈라 곱셈 알고리즘이 널리 사용되고 있으며,[2] 하드웨어 구현과 변형된 몽고메리 알고리즘 가운데 실제 하드웨어 구현 시 칩 면적을 줄이고 동작 속도를 개선할 수 있는 방안이 연구되고 있다.[3,4] 본 논문에서는 실제 하드웨어 구현에 적합하고 칩 면적을 줄일 수 있는 최적의 워드 기반 몽고메리 모듈라 곱셈기 구조와 마지막 덧셈 연산을 고속으로 수행하는 가상 캐리 예측 덧셈기를 제안한다. 본문 Ⅱ장에서는 일반적인 워드 기반 몽고메리 모듈라 곱셈 알고리즘[5]과 이러한 알고리즘을 최적으로 구현하기 위한 병렬 2단 CSA(Carry Save Adder)를 사용한 워드 기반 모듈라 곱셈 알고리즘을 제안한다. 본문 Ⅲ장에서는 병렬 2단 CSA를 사용한 워드 기반 모듈라 곱셈기 구조와 가상 캐리 예측 덧셈기(PCLA)를 제안한다. 본문 Ⅲ장에서 제안된 두 가지 구조를 PCI 인터페이스를 갖는 FPGA로 기능을 검증 후 0.5㎛ 삼성 gate array 공정을 사용하여 칩으로 구현하였으며 칩의 면적과 연산 속도 차이를 본문 Ⅳ장에서 비교 검토하였다.

Ⅱ. 워드 기반 몽고메리 곱셈 알고리즘

Algorithm 1. WMM(A, B, N)
Inputs : 승수 (A), 피승수 (B), 모듈러스 (N)
Step 1 : $R = 0$
Step 2 : **for** i = 0 **to** n-1 **do** {
  2a. $R^0 = R^0 + A_i \cdot B^0$
  if $R_0^0 = 1$ then
  2b. $R^0 = R^0 + N^0$
  Left shift and align R^0
  for j = 1 **to** w-1 **do** {
  $R^j = R^j + A_i \cdot B^j + N^j$
  Left shift and align R^j }
  else
  for j = 1 **to** w-1 **do** {
  $R^j = R^j + A_i \cdot B^j$
  Left shift and align R^j }
  end if
  }
Step 3 : **if** (R>N) R = R − N
Step 4 : **return** (R)
Output : R(remainder) = ABr^{-1} mod N, $0 \le R < N$, $r=2^n$

그림 1. 일반적인 워드 기반 몽고메리 모듈라 곱셈 알고리즘[5]
Fig. 1. A general word-based Montgomery multiplication algorithm.[5]

Algorithm 2. A CSA-based WMM
Inputs : 승수 (A), 피승수 (B), 모듈러스 (N)
Step 1 : R = (C, S) = 0 (initialization)
Step 2 : **for** i = 0 **to** n+b-1 **do** {
2a. $(C^0, S^0) = C^0 + S^0 + A_i \cdot B^0$, temp1 = C_{b-1}^0
  if $S_0^0 = 1$ then
2b. $(C^0, S^0) = (C_{b-2}^0 \cdots C_0^0) + S^0 + N^0$
  S_{b-1}^0 = temp1 OR C_{b-1}^0, temp2 = temp1 • C_{b-}^0
  $S^0 = (S_{b-1}^0 \cdots S_0^0)$, $C^0 = (0\ C_{b-2}^0 \cdots C_0^0)$
  for j = 1 **to** w-1 **do** {
  $(C^j, S^j) = C^j + S^j + A_i \cdot B^j$, temp1 = C_{b-}^j
  $(C^j, S^j) = (C_{b-2}^j \cdots C_0^j$ temp2) + $S^j + N^j$
  S_{b-1}^j = temp1 OR C_{b-1}^j
  temp2 = temp1 • C_{b-1}^j, $C_{b-1}^{j-1} = S_0^j$
  $S^j = (S_{b-1}^j \cdots S_0^j)$, $C^j = (0\ C_{b-2}^j \cdots C_0^j)$ }
  else
  for j = 1 **to** w-1 **do** {
  $(C^j, S^j) = C^j + S^j + A_i \cdot B^j$, temp1 = C_{b-}^j
  $(C^j, S^j) = (C_{b-2}^j \cdots C_0^j$ temp2) + S^j
  S_{b-1}^j = temp1 OR C_{b-1}^j
  temp2 = temp1 • C_{b-1}^j, $C_{b-1}^{j-1} = S_0^j$
  $S^j = (S_{b-1}^j \cdots S_0^j)$, $C^j = (0\ C_{b-2}^j \cdots C_0^j)$ }
  end if }
Step 3 : **return** R = S + C
Output : R(remainder) = ABr^{-1} mod N, $0 \le R < N$, $r=2^{n+b}$

그림 2. CSA 기반 WMM 알고리즘
Fig. 2. A CSA-based WMM algorithm.

일반적으로 1024-bit RSA 암호프로세서의 모듈라 곱셈기를 1024-bit 워드 기반으로 설계하면 높은 fan-out으로 인해 전파지연을 초래하고 많은 면적을 차지하게 된다. 이러한 문제를 해결하기 위해 본 논문에서는 1024-bit를 짧은 워드 길이로 분할한 워드 기반 모듈라 곱셈 알고리즘을 제안한다. 분할형 워드 기반 모듈라 곱셈 알고리즘은 n-bit 길이의 승수(A), 피승수(B), 그리고 모듈러스(N)과 b-bit 길이를 갖는 워드에 대하여 w(⌈n/b⌉)개의 워드가 필요하며 승수 A는 비트 단위로, 피승수(B)와 모듈러스(N)는 워드 단 위로 스캔된다. 일반적인 워드 기반 알고리즘은 그림 1과 같으며 X의 k번째 워드에서 i번째의 비트 위치를 X_i^k로 표현하

였다.

그림 1에서와 같은 워드 기반 모듈라 곱셈 알고리즘(word-based modular multiplication algorithm : WMM)은 피승수(B)와 모듈러스(N)을 워드 단위로 스캔하여 승수(A)의 각 비트에 대하여 b-bit 길이의 부분 나머지 합(R^j)을 구한다. 따라서 WMM을 고속으로 처리하기위해서 부분 나머지 합을 고속으로 계산할 수 있는 덧셈기가 필요하다. 만약 위의 덧셈 연산 구조를 병렬 2단 CSA 구조를 사용하여 구성한다면 피연산자의 워드 길이에 상관없이 한 클럭에 수행될 수 있다. 그러므로 매번의 반복 덧셈 연산 후 부분 나머지 합(R^j)은 캐리와 합의 벡터 형태로 출력할 수 있으며 마지막 워드에 대한 반복 덧셈 후 캐리와 합을 더함으로써 최종 나머지(R)값을 얻게 된다. 이러한 알고리즘의 가장 큰 문제점은 다음 워드에 대한 모듈라 곱셈 연산을 위해 캐리와 합으로 분할된 부분 나머지를 쉬프트와 정렬 연산을 수행해야 하는 어려움이 있다.[5]

본 논문에서는 그림 1에서의 단계2b에 필요한 쉬프트와 정렬 연산을 하는 블록과 그림 1에서의 단계3에 필요한 비교기를 제거하기 위해 CSA 기반의 WMM 구조를 제안한다. 먼저 복잡한 쉬프트와 정렬 연산을 하는 블록을 제거하기 위해 부가적인 논리 소자를 덧붙여 2단 병렬 CSA 구조를 채택하였으며 그림 1의 단계 3에서 필요한 비교 연산을 (n+b) 길이로 확장된 연산자를 사용하여 제거한 알고리즘을 그림 2에서 기술하였다. 만약 합(S)의 첫 번째 워드에서의 첫 번째 비트가 1이면 모듈라 연산은 매 클럭마다 피승수(B)와 모듈러스(N)의 모든 워드에 대하여 수행되며 만약 0이면 피승수(B)의 모든 워드에 대해서만 모듈라 연산이 수행된다. 그림 2에서 OR와 AND의 논리 소자는 다음 모듈라 연산을 위해 정렬된 합과 캐리(C, S)로 출력하므로 복잡한 쉬프트와 정렬 연산을 대체해준다. 따라서 그림 2에서 기술된 워드 기반 모듈라 곱셈 알고리즘은 w의 클럭이 요구되는 불필요한 쉬프트와 정렬 연산을 제거하여 하드웨어 소모를 줄일 수 있고 연산에 필요한 클럭을 줄여 연산 속도를 높일 수 있다.

Ⅲ. CSA 기반 WMM의 VLSI 구현

워드 기반 몽고메리 모듈라 곱셈기는 앞서 언급한 바와 같이 두 입력 승수 A, 피승수 B, 그리고 모듈러스 N에 대해서 $WMM(A, B, N) = A \cdot B \cdot r^{-1} \bmod N$을 출력하는 블록이며 전체 구조는 그림 3과 같다.

그림 3에서 CSA1은 그림 2의 워드 기반 몽고메리 알고리즘에서 $(C^j, S^j) = C^j + S^j + A_i \cdot B^j$ 연산을 처리하며, CSA2는 $(C^j, S^j) = C^j + S^j + S_0^0 \cdot N^j$ 연산을 처리한다. 연산 과정 중간값은 carry와 sum 값으로 나누어져 있기 때문에 본 논문에서는 고속 덧셈을 수행하기 위해 PCLA(pseudo carry look-ahead adder : 가상 캐리 예측 덧셈기)를 사용하여 마지막 덧셈을 수행한다. 그림 3과 같이 n 비트 피연산자 A, B, 그리고 N에 대하여 b 비트 워드 기반 모듈라 곱셈기는 $w = \lceil n/b \rceil$개의 캐리 레지스터와 합 레지스터가 필요하며 피승수 B와 모듈러스 N을 w개로 분할하여 저장한다. n 비트 승수 A에 대하여 총 $(n + b) \cdot w + a$ 번의 모듈라 곱셈 연산을 수행하는데 여기서 a는 b 비트 PCLA가 수행되는데 필요한 시간이다.

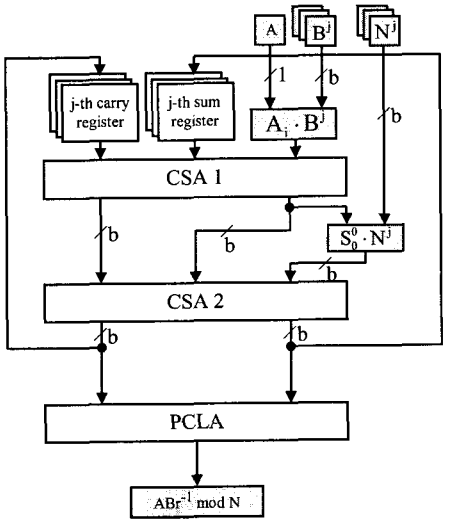


그림 3. WMM 모듈의 하드웨어 구조
Fig. 3. The architecture of WMM module.

1. CSA 기반의 WMM 구조

그림 1의 단계 2에서 캐리 전파 지연을 줄이기 위하여 병렬 2단 CSA 구조를 사용하여 모듈라 곱셈기를 구성하였다. CSA는 캐리와 합의 형태로 출력함으로써 워드 기반 모듈라 연산을 어렵게 하는 문제가 있다. 즉 승수 A의 i번째 비트에 대하여 $\lceil n/b \rceil$개의 워드에 대하여 모듈라 연산을 수행한 후 다음 승수 A의 i+1번째 비트에 대하여 모듈라 연산을 위한 쉬프트와 정렬 연산이 필요하게 된다. 그림 2의 알고리즘은 이러한 복잡

한 쉬프트와 정렬 연산을 제거할 수 있으며 따라서 고속 워드 기반 모듈라 곱셈을 가능하게 한다. 그림 2의 알고리즘을 예로써 설명하면 다음과 같다. n은 8 비트이며 워드 길이는 4비트일 때 승수 A가 133 (1000, 0101), 피승수 B가 137 (10001001), 그리고 모듈러스 N이 233 (1110,1001)인 경우에 R^{-1} mod N = 2^{-8} mod 233의 결과가 152가 되어 최종 나머지 결과는 133 • 137 • 152 mod 233 = 154가 된다. 이의 계산과정을 그림 2의 알고리즘을 사용하여 그림 4에 도시하였다.

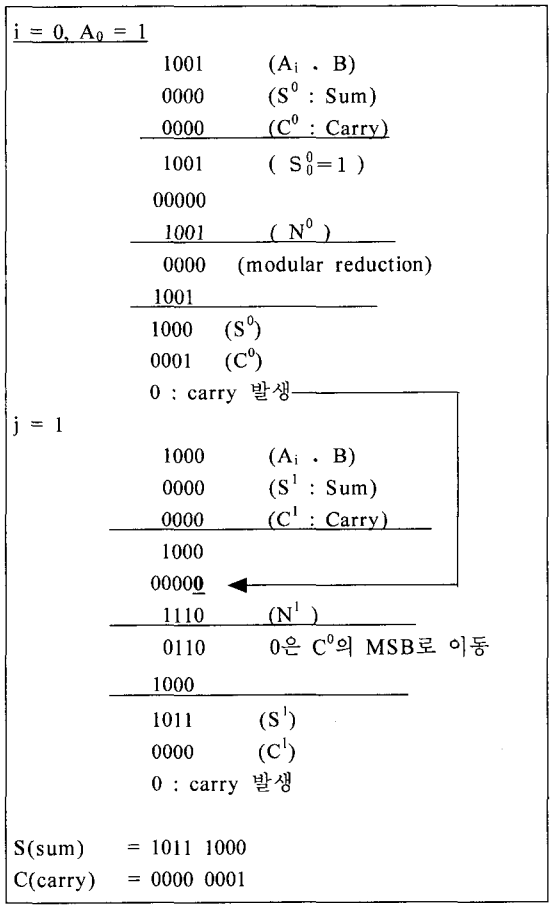


그림 4. CSA 기반의 WMM 알고리즘의 계산 예
Fig. 4. An example of computation of a CSA-based WMM algorithm.

그림 4와 같이 계산하면 승수 A의 8번째 비트 1에 대하여 합 S는 10011000이 되고 캐리 C는 00000010이 된다. 두 수를 덧셈하면 결과는 10011010이 되어 예측한 154를 얻을 수 있다. 그림 2의 알고리즘을 적용한 그림 4의 계산 방법을 하드웨어에 적용하면 그림 5와 같이 구성할 수 있다.

그림 5의 모듈라 곱셈기는 그림 3에서 도시한 것과 같이 병렬 2단 CSA와 쉬프트와 정렬 연산을 간단히 수행할 수 있는 그림 2의 알고리즘에서 표현한 여분의 논리 회로(AND, OR)로 구성되어 있다. 이 구조에서 승수 A는 비트 단위로 스캔되어 입력되며 피승수 B와 모듈러스 N은 워드 단위로 스캔되어 입력된다. 연산자 A, B, 그리고 N에 의해 워드 길이 단위로 합(S)와 캐리(C)가 발생되며, 제어신호(control)는 처음 클럭에서는 '0'를 선택하고 다음 클럭에서는 이전 모듈라 연산에서 발생한 캐리를 전파하기 위하여 '1'을 선택한다. 승수 A가 모두 스캔되면 w= ⌈n/b⌉ 개의 합(S)과 캐리(C)가 레지스터에 저장된다. 다음 승수 A에 대하여 모듈라 연산을 위해서는 합(S)과 캐리(C)를 더해야하며 이를 위해 본 논문에서는 다음 절에서 설명할 고속 가상 캐리 예측 덧셈기를 이용한다.

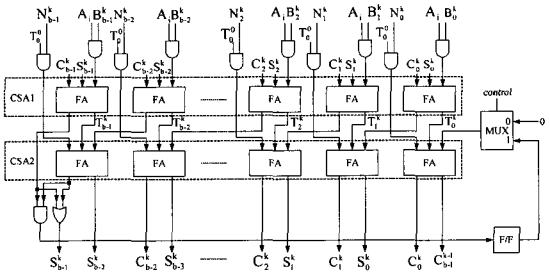


그림 5. CSA 기반 WMM 알고리즘의 하드웨어 구조
Fig. 5. The architecture of a CSA-based WMM algorithm.

2. Pseudo Carry Look-ahead Adder

워드 기반 WMM 알고리즘은 출력을 합(S)과 캐리(C)의 벡터형으로 출력하므로 최종 결과를 얻기 위해서는 합(S)과 캐리(C)를 더해야한다. 본 논문에서는 가상 캐리 예측 덧셈기(pseudo carry look-ahead adder : PCLA)를 사용하여 고속 덧셈을 수행한다. PCLA에서 입력을 합(S)과 캐리(C)라 하면 i번째 PCLA의 출력은 식 (1)과 같다.

$$Sum_i = H_{i-1} \cdot P_{i-1} \oplus \bar{P}_i \quad (1)$$

식 (1)에서 P_i (=C_i + S_i)는 캐리 전파 신호이며 $\bar{P}_i$는 $C_i \oplus S_i$이며, H_i는 가상 캐리이다. 모든 P_i와 $\bar{P}_i$ 신호는 병렬로 연산되며 H_i는 이전 식에 의해서 연산되

며 일반식은 식 (2)과 (3)과 같다.

$$H_0 = G_0 + C_{in}, \tag{2}$$

$$H_i = G_i + H_{i-1} \cdot P_{i-1}. \tag{3}$$

일단 H_i가 구해지면 합을 고속으로 얻을 수 있으며 식 (4)와 같이 선택기(multiplexer)로 구현할 수 있다.

$$Sum_i = \begin{cases} P_i, & \text{if } H_{i-1}=0 \\ P_{i-1} \oplus P_i, & \text{if } H_{i-1}=1 \end{cases} \tag{4}$$

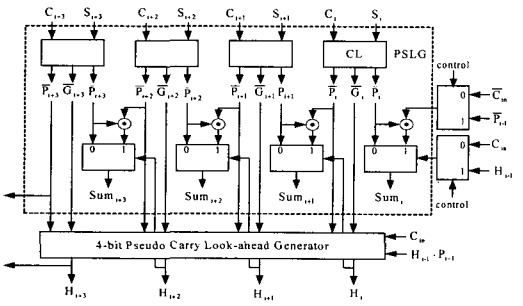


그림 6. 4 비트 가상 캐리 예측 덧셈기
Fig. 6. Four-bit pseudo carry look-ahead adder.

식 (4)에서 $P_{i-1} \oplus P_i$는 $\overline{P}_{i-1} \odot P_i$로 다시 쓸 수 있다. 식 (1)부터 (4)를 사용하여 4 비트 PCLA를 구성하면 그림 6과 같다. 일반적으로 캐리 예측 덧셈기를 설계할 때와 마찬가지로 긴 워드에 대하여 PCLA를 구성할 경우에는 그림 6에서와 같은 작은 블록들을 조합하여 다단계 그룹 PCLA를 구성할 수 있다.

그림 6을 블록으로 구성하면 그림 7과 같이 합을 출력하는 4 비트 pseudo sum look-ahead generator (PSLG)와 가상 캐리를 출력하는 4 비트 pseudo carry look-ahead generator (PCLG)로 구성할 수 있으며 그림 7의 4 비트 PCLA를 이용하여 16 비트 2 단계 그룹 PCLA를 그림 8과 같이 구성할 수 있다. 그림 7과 8에서 Hg와 Ig는 식 (5), (6)과 같으며 다음 그룹 PCLA의 가상 캐리를 발생시킨다. Hg와 Ig에 의해 발생되는 그룹 가상 캐리는 식 (7)와 같다.

$$Hg = G_3 + G_2 + P_2 G_1 + P_2 P_1 G_0, \tag{5}$$

$$Ig = P_2 P_1 P_0 P_{i-1}, \tag{6}$$

$$H_{i+3} = Hg + Ig \cdot H_{i-1}. \tag{7}$$

그림 8의 게이트 지연을 살펴보면 다음과 같다.
1 gate delay(각 비트 위치에서 G와 P를 출력) + 2 gate delays(4 비트 블럭의 입력을 위한 가상 캐리 신호 H3, H7, 그리고 H11을 예측) + 2 gate delays(각 4 비트 블록마다 내부 가상 캐리 예측) + 2 gate delays(Hg와 Ig를 출력) + 2 gate delays(합을 계산) = 9 gate delays

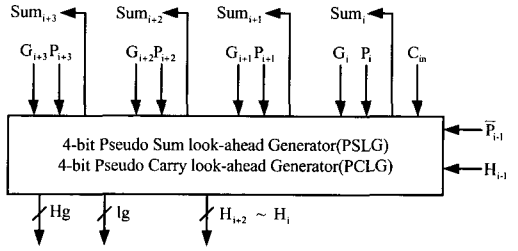


그림 7. 4 비트 그룹 PCLA 블록다이어그램
Fig. 7. Block diagram of 4-bit group PCLA.

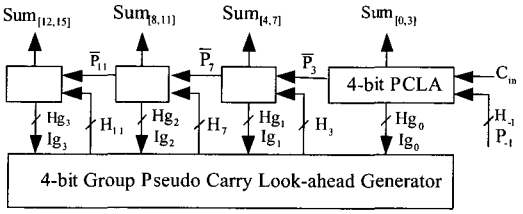


그림 8. 2 단계 16 비트 GPCLA
Fig. 8. Two-level 16-bit GPCLA.

표 1. 게이트 지연 비교
Table 1. The computation of gate delays(r is the group size and k is the width of the adder).

Adder	gate delay 일반식	gate delays 64 비트(r=4)
PCLA	$4\log_r k + 1$	13
CLA[6-7]	$4\log_r k + 1$	13
RCA	2k	128

표 1은 64 비트 리플 캐리 덧셈기(ripple-carry adder : RCA)와 캐리 예측 덧셈기(carry look-ahead adder : CLA)를 PCLA와 게이크 지연을 비교하였다. CLA와는 게이트 지연이 같지만 그룹 캐리(Cg)와 그룹 가상 캐리(Hg)를 구할 때 Cg는 4개의 fan-in, 10개의 항, 20개의 입력을 가짐에 비해,[6,7] Hg는 3개의 fan-in, 7개의 항, 14개의 입력을 가지므로 속도의 향상 및 면적을 절감할 수 있다.

본 논문에서는 b 비트의 워드 기반 모듈라 곱셈기를 위해 그림 9와 같이 PCLA를 구성하여 n 비트 크기의 연산자에 대해 $w=\lceil n/b \rceil$번을 반복 수행하여 $AB_r^{-1} \bmod N$의 최종 결과를 출력한다.

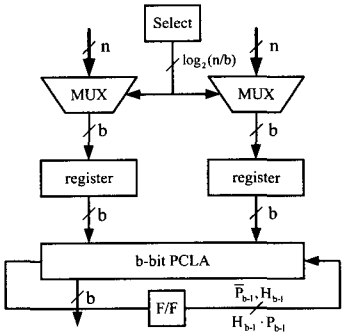


그림 9. PCLA 하드웨어 구조
Fig. 9. The architecture of PCLA.

표 2. L-R 이진 지수승 방식을 적용한 각 연산에 대한 클럭 사이클 (n= 1024)
Table 2. Clock cycles for each process using L-R binary method.

연산	클럭 사이클
Mapping	2×{(n+b) × w + n/b}
모듈라 지수승	1.5n×{(n+b) × w + n/b} : average 2n×{(n+b) × w + n/b} : worst
Re-mapping	1×{(n+b) × w + n/b}
총 수행시간	(2n+3)×{(n+b) × w + n/b}

Ⅳ. 성능 분석과 PCI 인터페이스를 통한 검증

1. 수행 시간과 하드웨어 면적 비교

본 논문에서 제시된 RSA 전체블록의 수행시간을 L-R 이진 지수승 방식을 기준으로 표 2에 나타내었다. 본 논문에서 구현한 L-R 이진 연산 방식의 n 비트 RSA 모듈라 지수승 연산기는 최악의 경우 2n+3, 그리고 평균의 경우엔 1.5n+3의 클럭수 동안 모듈라 곱셈을 수행한다. 여기서 3번의 모듈라 곱셈이 추가된 것은 mapping을 위한 두 번의 곱셈과 re-mapping을 위한 한번의 곱셈이 필요하기 때문이다. 그리고, 모듈라 곱셈을 수행하기 위해서는 (n+b) × w + n/b번의 클럭 사이클이 요구되는데, 여기서 b는 워드 길이이며 w는 워드 개수이다. 따라서 전체 연산을 수행하는데 소요되는 총 클럭 사이클은 최악의 경우 (2n+3) × {(n+b) × w + n/b}이다. 1024 비트 RSA를 L-R 이진 연산 방식과 256 비트 워드 기반 WMM으로 구성하면 13M 클럭 사이클이 소요된다.

표 3. 두 개의 1024 비트 RSA 프로세서의 특성
Table 3. The features of the two 1024-bit RSA processors.

비교 \ 공정		0.5㎛ 삼성 gate array	
워드 길이		256	1024
설계 면적	전체 면적	41.5 K	122 K
	모듈라 곱셈기	18 K	69 K
	버퍼	3 K	7 K
	선택기	3 K	6 K
	제어기	10 K	10 K
	입/출력 인터페이스	7.5 K	30 K
클럭 주파수		66MHz	33MHz
수행 시간		122ms	59ms

L-R 이진 지수승 방식을 적용한 1024 비트 RSA 코어를 0.5㎛ 삼성 gate array 공정을 사용하여 1024 비트와 256 비트를 적용한 워드 기반 몽고메리 모듈라 곱셈기를 설계하여 표 3에서 비교하였다. 1024 비트 모듈라 곱셈기는 256 비트의 워드 기반 모듈라 곱셈기에 비하여 약 4배의 게이트가 필요한 레지스터와 CSA 회로로 인하여 약 4배가 많은 게이트를 소모하였으며 전체적으로 약 3배 이상의 게이트를 소모하였다. 전체 RSA의 수행시간은 1024 비트 길이의 모듈라 곱셈기를 사용할 경우 많은 fan-out 신호로 클럭 주파수를 높일 수 없어 전체 수행시간은 256 비트의 워드 기반 모듈라 곱셈기보다 약 2배의 이득을 보임을 알 수 있다.

2. PCI 인터페이스를 통한 동작 검증 및 테스트

본 논문에서 설계된 분할형 워드 기반 1024 비트

RSA 암호프로세서의 사용을 위하여 PCI 인터페이스 보드를 부가적으로 설계하였다. PCI 인터페이스 보드는 로컬 버스에 132MB/sec의 버스트전송을 제공할 수 있는 PLX 9050 칩을 타겟으로 제작되었다.

PCI 인터페이스 보드는 그림 10에서와 같이 33MHz에서 동작하며 32-bit I/O, 입/출력 데이터 전송을 위한 데이터 어레이, 그리고 이러한 블록들을 제어하는 제어 블록으로 구성되어 있다. 1024 비트 RSA 암호프로세서는 그림 11과 같이 PCI 인터페이스 보드를 통하여 메시지 M, 공개키 e, 그리고 모듈러스 N을 전송하여 기능을 테스트하였다.

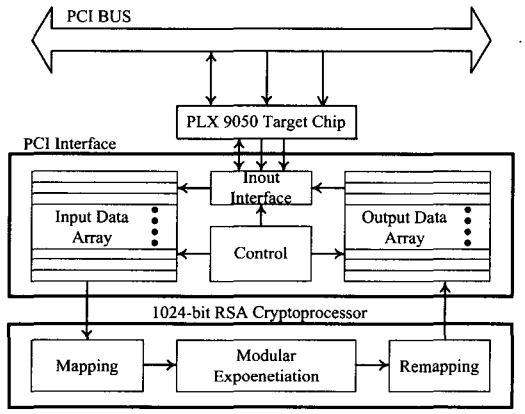


그림 10. 1024 비트 RSA 암호프로세서를 위한 PCI 인터페이스

Fig. 10. PCI interface for the 1024-bit RSA cryptoprocessor.

V. 결 론

본 논문에서는 n 비트 RSA에 대하여 b 비트 길이의 병렬 2단 CSA와 가상 캐리 예측 덧셈기(pseudo carry look-ahead adder), 그리고 PCI 인터페이스를 적용한 분할형 워드 기반 몽고메리 곱셈기를 제안하였다. 정렬 연산과 쉬프트 연산이 요구되는 워드 기반 몽고메리 모듈라 곱셈 알고리즘[5]은 본 논문에서 제안한 WMM 알고리즘보다 연산 수행시 n/b 배의 클럭이 더 필요한 단점이 있다. 본 논문에서는 이러한 단점을 기본 논리 소자를 이용하여 해결하였으며 n/b 배의 속도 향상을 얻을 수 있었다. 또한, b 비트 PCLA를 n/b번 사용하여 모듈라 연산의 결과를 고속으로 얻을 수 있었으며 n 비트 길이를 갖는 몽고메리 모듈라 곱셈기보다 하드웨어 면적을 n/b 만큼 줄일 수 있었다.

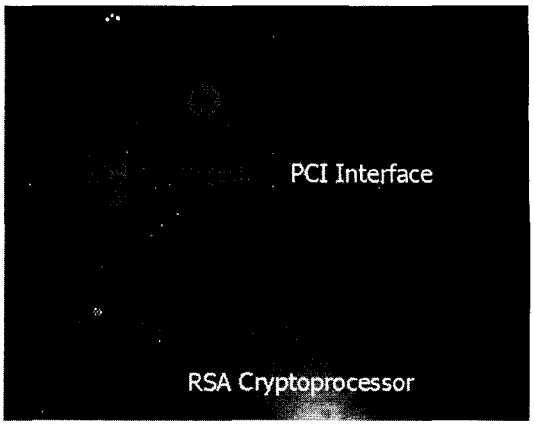


그림 11. 1024 비트 RSA 암호 프로세서의 보드 테스트

Fig. 11. The board test of the 1024-bit RSA cryptoprocessor.

참 고 문 헌

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Communications of the ACM, 21(2):120-126, February 1978.

[2] T. Blum and C. Paar, "Montgomery modular exponentiation on reconfigurable hardware," in Proc. 14th IEEE Symposium on Computer Arithmetic, 1999, pp. 70~77.

[3] C. K. Koc, "High-Speed RSA Implementation," Technical Report TR 201, RSA Laboratories, November 1994.

[4] C. K. Koc, "RSA Hardware Implementation," Technical Report TR 801, RSA Laboratories, April 1996.

[5] F. Tenca and C. K. Koc, "A scalable architecture for Montgomery multiplication," In C. K. Koc and C. Paar, editors, Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science No. 1717, pp 94~108, Springer, Berlin, Germany, 1999.

[6] H. Ling, "High speed binary adder," IBM Journal of Research and Development, Vol. 25, No. 3, pp. 156~166, May 1981.

[7] Lynch, T., and E. Swartzlander, "A spanning tree carry lookahead adder," IEEE Trans.

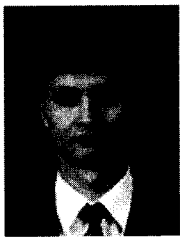
Computers, Vol. 41, No. 8, pp. 931~939, 1992.

[8] T. W. Kwon, J. R. Choi and etc., "Two implementation methods of a 1024-bit RSA cryptoprocessor based on modified Montgomery algorithm," Circuits and Systems, ISCAS 2001. Vol. 4, pp. 650~653, Sydney, 2001

저 자 소 개

權 宅 元(學生會員)

1998년 경북대학교 전자공학과 학사. 2000년 경북대학교전자공학과 석사. 2000년 3월~현재 경북대학교 전자공학과 박사과정



崔 峻 林(正會員)

1986년 연세대학교 전기공학과학사. 1988년 미국 Cornell 대학교 전자전기공학과 석사. 1991년 미국 Minnesota 대학교 전자전기 공학과 박사. 1991년 7월~1997년 2월 LG 전자기술원 책임 연구원. 1997년 3월~현재 경북대학교 전자전기공학부 조교수