論文96-33B-9-6

# 비트 레벨 일차원 시스톨릭 모듈러 승산

# (Bit-level 1-Dimensional Systolic Modular Multiplication)

崔 盛 旭 *, 禹 鍾 鎬 **

(Sung-Wook Choi and Chong-Ho Woo)

요 약

본 논문에서는 모듈러 승산을 위한 비트레벨 일차원 시스톨릭 어레이를 설계한다. 먼저, 어레이 설계에 적합한 Walter가 제안한 몽고메리 알고리즘을 근거하여 병렬 알고리즘과 데이타 의존 그래프를 유도한다. 체계적인 시스톨릭 어레이의 설계과정에서 4가지의 일차원 시스톨릭 어레이가 구해지고 이를 다양한 평가 기준에 따라서 평가한 결과, 투영방향 [0,1]에서 제어로직을 하나 추가하고 데이타 A의 방향을 직렬로 수정함으로써 최적의 일차원 시스톨릭 어레이가 설계된다. 이는 확장가능한 모듈로 구성된 상수개의 입출력채널을 가지며 통신패스가 단일방향이므로 결함허용성이 용이하다. 이는 연속적이고 큰 수의 모듈러 승산이 필요한 RSA 암호시스템에 적합하다.

**Abstract**

In this paper, the bit-level 1-dimensional systolic array for modular multiplication is designed. First of all, the parallel algorithm and data dependence graph from Walter's method based on Montgomery Algorithm suitable for array design for modular multiplication is derived. By the systematic procedure for systolic array design, four 1-dimensional systolic arrays are obtained and then are evaluated by various criteria. As it is modified the array which is derived from [0,1] projection direction by adding a control logic and it is serialized the communication paths of data A, optimal 1-dimensional systolic array is designed. It has constant I/O channels for expansile module and it is easy for fault tolerance due to unidirectional paths. It is suitable for RSA Cryptosystem which deals with the large size and many consecutive message blocks.

## I. 서 론

현대의 고도 정보화 사회가 진전됨에 따라 다양하고 많은 정보의 전송 및 저장이 필요하게 되었고, 이러한 정보를 취급함에 있어서 컴퓨터 시스템의 내부에서 또는 각 시스템 상호간의 통신에서 각종 정보에 대한 보호기술이 중요하게 대두되었다. 공개키 암호시스템의 하나인 RSA 암호시스템은 키의 분배와 관리가 쉽고, 인증 및 디지탈 서명이 가능한 장점을 가지고 있으나 512 비트 이상의 큰 수에 대해 모듈러 멱승을 요구함으로써 기본연산인 모듈러 승산의 고속화를 위한 VLSI 하드웨어 구현이 필요하게 되었다.

RSA 암호시스템의 암호화 및 복호화과정에서 주요 연산은 $M^e$ mod N 의 모듈러 멱승으로서 모듈러 승산이 전체 계산의 대부분을 차지하며, 그중에서 몽고메리 모듈러 승산 알고리즘이 가장 효과적인 것으로 알려져 있다[1-2].

이러한 모듈러 승산은 문제 규모가 크고 많은 수의 메세지 블럭이 연속적으로 계산되어야 하므로 입출력 채널의 수를 상수개로 고정시켜서 VLSI 칩으로 제작

* 正會員, 韓國海洋大學校 電子通信工學科
(Dept. of Ele. & Com. Eng. Korea Maritime Univ.)
** 正會員, 釜慶大學校 컴퓨터工學科
(Dept. of Computer Eng. Pukyong Nat'l Univ.)
接受日字:1995年9月1日, 수정완료일:1996年8月12日

할 경우, 문제 크기에 관계없이 입출력 핀의 수를 고정시킬 수 있는 확장 가능한 모듈을 구성하는 개념이 매우 중요하다.

Sauerbrey는 Atrubin-Array를 이용하여 두개의 시스톨릭어레이에 기반을 둔 몽고메리 모듈러 승산기를 구성하여 512 비트의 모듈러 멱승기로 구현시 1.2 bps/gate (215Kbps/178Kgate)의 구현 효율을 보였고, Iwamura 등은 몽고메리 알고리즘을 이용하여 일차원 시스톨릭어레이를 설계하여 512 비트의 모듈러 멱승기로 구현시 2.0 bps/gate (50Kbps/25Kgate)의 고속처리가 가능하였으며, Walter는 모듈러 승산을 위한 이차원 시스톨릭어레이를 제안하였다[3-5].

본 논문에서는 체계적인 시스톨릭어레이 설계 기법[6]을 적용하여 Walter가 제안한 몽고메리 알고리즘을 근거로 병렬 알고리즘으로 유도한 후 데이타 의존 그래프로 표현하고 이를 시간 및 공간 변환을 수행하여 일차원 시스톨릭어레이를 설계하였다. 그리고 설계한 일차원 시스톨릭어레이를 확장가능한 모듈을 고려하여 여러가지 평가기준에 따라 비교·분석하였으며, 처리요소를 설계하였고, 컴퓨터 시뮬레이션을 수행하여 설계의 정확성과 유효함을 검증하였다.

## Ⅱ. 몽고메리 병렬 알고리즘의 유도

### 1. 몽고메리 알고리즘

P. L. Montgomery는 주어진 정수 N에 대한 모듈러 승산알고리즘을 제안했다. 이에 관련된 몇가지 정의는 다음과 같다[1-2].

정의 1 : 환 $<Z_N, +, \bullet>$는 집합 $\{0,1,\cdots,N-1\}$에 모듈러 합과 모듈러 곱이 정의된 정규 모듈러 수체계이다.

정의 2 : 환 $<RZ_N, \oplus, \odot>$는 집합 $\{0,1,\cdots,N-1\}$에 모듈러 합과 모듈러 곱이 아래와 같이 정의된 몽고메리 모듈러 수체계이다.

임의의 $a, b \in Z_N$ $f(a)$, $f(b) \in RZ_N$ 에 대하여

$f(a) \oplus f(b) = (f(a) + f(b))$,

$f(a) \odot f(b) = (f(a) \bullet f(b)) \bullet R^{-1}$ 이다.

정의 3 : 환 $<Z_N, +, \bullet>$과 $<RZ_N, \oplus, \odot>$는 아래에서 정의되는 함수 f 에 의해 서로 환 동형사상을 갖는다.

$f : Z_N \rightarrow RZ_N$ , $a \pmod N \rightarrow aR \pmod N$

여기서 $f(a) = aR$ 이다.

각 환은 함수 f가 일대일 대응사상(one to one correspondence)으로 역함수 $f^{-1}$를 가지며 각 환에서 정의된 연산은 다음의 성질을 만족한다.

임의의 $a, b \in ZN$ 에 대하여

$f(a+b) = f(a) \oplus f(b)$, $f(a \bullet b) = f(a) \odot f(b)$.

따라서 정의 3의 두 집합간의 환 동형사상은 그림 1과 같다. 정의 1, 2, 3 을 만족하는 몽고메리 모듈러 정수 환에서 $GCD(R,N) = 1$, $R > N$, $R = 2^{vn} = r^n$, N이 홀수일 때, 임의의 정수 a, b에 대하여 몽고메리 모듈러 승산 MP(a,b,N,R)는 정수가 되며 다음 (1)과 같다.

$$MP(a,b,N,R) = (a \cdot b + M \cdot N) / R \equiv a \cdot b \cdot R^{-1} \bmod N \quad (1)$$

여기서, $M = (a \bullet b \bullet (-N^{-1} \bmod R)) \bmod R$ 이다.

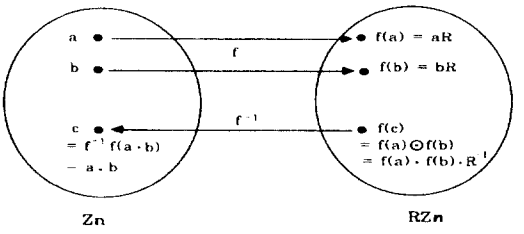



그림 1. 몽고메리 모듈러 승산을 위한 환 동형사상
Fig. 1. Ring isomorphism for Montgomery modular multiplication.

몽고메리 모듈러 정수 환에서 어떤 큰 수를 $r^n$으로 나누었을 때 몫과 나머지를 구하는 것은 각각 단지 그 수의 n자리 이상의 수를 취하는 것과 n자리 미만의 수를 취하는 것이다. $r^n$에 대한 나눗셈은 매우 쉽다는 성질을 이용한다.

### 2. 몽고메리 모듈러 승산 알고리즘

몽고메리 모듈러 승산 알고리즘은 그림 2와 같고 A, B, N, R, P는 다정도 정수이며 $a_i$, $p_i$는 A, P의 i번째 자리수이고 $P_i$는 i번째 누적되는 중간값이다[2].

여기서 모듈러스의 배수 $m_i$는 $p_i$에 의하여 계산되고 그 결과를 $P_i$의 계산에 이용하며 $m_i$를 뺄셈 연산하기보다는 덧셈 연산을 수행한다. 또한 피승수 A의 처리순서를 역전시켜 A의 낮은 자리수부터 계산하고, 각 부분 결과를 윗자리로 이동시키지 않고 아래자리로 이동시킨다.

Algorithm : MP(A,B,N,R)
Input : A, B, N, R.
Output : $P = A \times B \times R^{-1} \bmod N$

1 : $n_o' \leftarrow -n_o^{-1} \bmod r$; $P^{-1} \leftarrow 0$;
2 : for $i \leftarrow 0$ to n-1
$P_i \leftarrow P_{i-1} + a_i \times B \times r^i$;
$m_i \leftarrow p_i \times n_o' \bmod r$;
$P_i \leftarrow P_i + m_i \times N \times r^i$;
3 : $P \leftarrow P_i$ div R;
4 : return P;

그림 2. 몽고메리 모듈러 승산 알고리즘
Fig. 2. Montgomery modular multiplication algorithm.

3. 병렬 알고리즘의 유도

Walter는 그림 2의 2행에 해당하는 순환 부분을 다음 (2)와 같이 변형하였다[5].

for $i \leftarrow 0$ to n-1
$m_i \leftarrow ((P_{i-1} \bmod r) + a_i \times b_0) \times n_o' \bmod r$
$P_i \leftarrow (P_{i-1} + a_i \times B + m_i \times N)$ div r (2)

여기서 $P_{-1} = 0$ 은 P의 초기값이고, $P_{i-1}$는 for 루프 문장의 i번째 계산을 수행하기 이전 P의 부분결과이다. 수학적 귀납법에 따라 $r^i P_i = a_i B + m_i N$ 이 되며 P의 최종값 $P_n$은 $r^n P_n = AB + MN$이 되어 결과적으로 $P \equiv ABR^{-1} \bmod N$이 된다. 그림 2의 알고리즘에서는 P의 최종값에서 $R=r^n$을 나누어서 P를 구했으나 여기서는 비트레벨(bit level)에서 처리가 가능하도록 최종 결과값을 나눗셈 연산을 하는 대신 각각의 부분 결과값에서 나눗셈 연산을 수행하였으며 병렬처리가 용이하도록 각 계산점에서 의존 관계의 수를 줄였다.

이진수로 표현되는 비트레벨에서 처리되며 P를 생성하기 위하여 요구되는 입력값들에서 피승수 A와 모듈러스의 배수 M은 반복 루프(loop)에 해당하는 i인덱스로 두고, 승수 B와 모듈러스 N은 단일비트의 위치에 해당하는 j인덱스로 두어 인덱스공간을 2차원으로 확장한다. 여기서 결과값 P를 생성하는 한 부분인 비트레벨의 피승수 ai와 연산되는 승수 B를 단지 다정도상수로 연산하는 과정과 비트레벨로 인덱스를 확장한 $b_j$에서의 승산과정이 동일하다.

이러한 알고리즘을 근거로 병렬 알고리즘을 구성하면 그림 3과 같으며 비트레벨에서 짝수 r과 서로소인 $n_0'$의 값은 항상 1이 되므로 이 연산은 생략된다. 또한 캐리비트는 웨이트를 가진 상위 캐리비트와 웨이트가 없는 하위 캐리비트로 분리되어 계산되며 부분 결과값은 하위 캐리비트만을 이용한다.

1 : begin
2 : for all $0 \le i \le n-1$,
$0 \le j \le n+1$ do in parallel
3 : if i = 0 then
4 : $b_{i,j} \leftarrow b_j$; $n_{i,j} \leftarrow n_j$; $p_{i,j} \leftarrow 0$;
5 : else
6 : $b_{i,j} \leftarrow b_{i-1,j}$; $n_{i,j} \leftarrow n_{i-1,j}$;
7 : if j = n+1 then
8 : $p_{i,j} \leftarrow 0$;
9 : else
10: $p_{i,j} \leftarrow p_{i-1,j+1}$;
11: end if
12: end if
13: if j = 0 then
14: $a_{i,j} \leftarrow a_i$; $ca0_{i,j} \leftarrow 0$; $ca1_{i,j} \leftarrow 0$;
15: $m_{i,j} \leftarrow ((p_{i,j} \bmod r) + a_i \times b_{i-1,j}) \bmod r$;
16: else
17: $a_{i,j} \leftarrow a_{i,j-1}$; $m_{i,j} \leftarrow m_{i,j-1}$;
$ca0_{i,j} \leftarrow ca0_{i,j-1}$; $ca1_{i,j} \leftarrow ca1_{i,j-1}$;
18: end if
19: $p_{i,j} \leftarrow (p_{i,j} + a_{i,j} \times b_{i,j} + m_{i,j} \times n_{i,j} + ca0_{i,j}) \bmod r$;
20: $ca0_{i,j} \leftarrow ((p_{i,j} + a_{i,j} \times b_{i,j} + m_{i,j} \times n_{i,j} + ca0_{i,j} + ca1_{i,j} \times r)$ div r) mod r;
21: $ca1_{i,j} \leftarrow (p_{i,j} + a_{i,j} \times b_{i,j} + m_{i,j} \times n_{i,j} + ca0_{i,j} + ca1_{i,j} \times r)$ div $r^2$;
22: all for
23: end

그림 3. Walter방식에서 유도된 병렬 알고리즘
Fig. 3. Parallel algorithm derived from Walter's method.

4. 데이타 의존 그래프

병렬 알고리즘내에서의 데이타 의존 관계를 그래프로 나타낼 때 노드는 계산 부분을, 아크는 데이타 의존 관계를 나타낸다. 이것을 VLSI 시스톨릭어레이 구조로 구현하면 각 노드는 처리요소가 되고 데이타 의존은 통신패스(path)가 된다.

따라서 문제 크기가 4인 경우 Walter방식에 의한 데이타 의존 그래프를 나타내면 그림 4와 같고 j=0인 열의 계산점이 나머지 계산열과 다르며 인덱스 j의 방향에서 캐리 두비트를 계산에 포함시키기 위하여 문제의 크기보다 두개의 계산열이 더 요구된다.

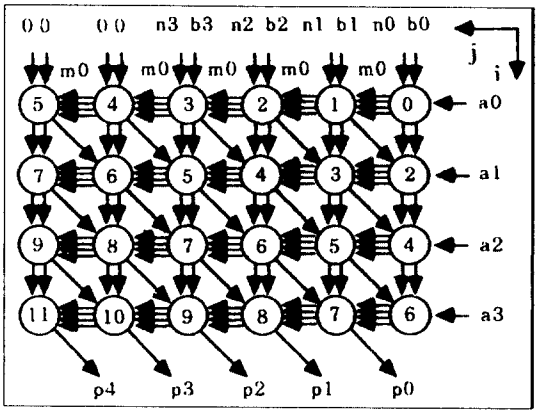



그림 4. 데이타 의존 그래프 (n=4)
Fig. 4. Data dependence graph (n=4).

## Ⅲ. 시스톨릭어레이의 설계

### 1. 시간 및 공간 변환

알고리즘에서 시스톨릭어레이로의 사상은 인덱스 공간의 계산점 I와 어레이에서 시간 및 공간의 계산점 Y가 주어질 때 Y=T(I)의 관계를 가진다. 이때 T는 일대일의 사상으로 식(3)과 같이 구성된다. 여기서 $T \in Z^{n\times n}$는 선형변환이고 Π와 S는 각각 $Z^{1\times n}$, $Z^{(n-1)\times n}$ 로 표현되는 시간 및 공간사상이며 n은 문제의 인덱스 공간의 차수, Z는 정수의 집합이다.

$$T=\begin{bmatrix}\Pi\\ S\end{bmatrix} \quad \text{여기서,} \quad \Pi\cdot\hat{e}_i\geq 1,\ \Pi\cdot\hat{p}_i\geq 1 \tag{3}$$

그림 4의 데이타 의존 그래프에서 각 변수들에 대한 데이타 의존 벡타들을 의존 행렬(D)로 표시하면 식(4)와 같다.

$$D=[\hat{e}_a\ \hat{e}_b\ \hat{e}_{ca1}\ \hat{e}_{ca2}\ \hat{e}_m\ \hat{e}_n\ \hat{e}_p]=\begin{bmatrix}0&1&0&0&0&1&1\\1&0&1&1&1&0&-1\end{bmatrix} \tag{4}$$

그리고 식(3)의 조건에 따라 최적의 시간 변환 행렬(Π)과 투영 벡타( $\hat{p}$)를 구하면 식(5), (6)와 같다.

$$\Pi=[2\ 1] \tag{5}$$

$$\hat{p}_1=[0\ 1],\ \hat{p}_2=[1\ 0],$$
$$\hat{p}_3=[1\ -1],\ \hat{p}_4=[1\ 1] \tag{6}$$

공간 변환(S)은 투영벡타와 직교하므로 식(7)과 같다.

$$S_1=[1\ 0],\quad S_2=[0\ 1],$$
$$S_3=[1\ 1],\quad S_4=[1\ -1] \tag{7}$$

그러므로 시간 및 공간 변환 함수로부터 변환된 의존 행렬($T_iD$)을 구하면 각각 식(8)과 같다.

$$T_1D=\begin{bmatrix}1&2&1&1&1&2&1\\0&1&0&0&0&1&1\end{bmatrix}$$

$$T_2D=\begin{bmatrix}1&2&1&1&1&2&1\\1&0&1&1&1&0&-1\end{bmatrix}$$

$$T_3D=\begin{bmatrix}1&2&1&1&1&2&1\\1&1&1&1&1&1&0\end{bmatrix}$$

$$T_4D=\begin{bmatrix}1&2&1&1&1&2&1\\-1&1&-1&-1&-1&1&2\end{bmatrix} \tag{8}$$

데이타 의존 그래프에서 식(7)에 의해 각 노드의 공간 변환을 수행하고 각각의 변환 $T_iD$에서 유도된 아크들의 방향으로 일차원시스톨릭어레이를 설계하면 그림 5와 같다.

그림 5의 (a)는 입력 a가 어레이내에 선적재되기 위해서 문제 크기만큼의 입력채널수가 요구되고, 이 a의 값에 의해서 m, ca1, ca2가 계산되어 처리요소내에 머물게 된다. 또한 그림 5의 (b)도 입력 b, n가 선적재되기 위하여 문제 크기만큼의 입력 채널수가 각각 요구된다. 여기서 계산결과 p는 입력값과 반대방향으로 전달되므로 입력값 a와의 충돌을 피하기 위하여 최종 출력값의 길이에 해당하는 시간지연이 주어져야 한다. 그림 5의 (c)는 모든 변수값이 링크를 따라 전달되므로 선적재가 요구되지는 않으나, 출력 채널수가 문제의 크기만큼 요구된다. 그림 5의 (d)는 자기루프(self-loop)가 제거됨으로써 선적재가 불필요하고, 입출력 채널수가 상수개로 고정된다.

따라서, 확장가능한 모듈의 구성을 고려하여 볼때 그림 5의 (d)가 가장 유리하나 처리요소 및 데이타 통신패스(path)의 수가 (a)의 경우보다 많다. 그러므로 처리요소의 수와 데이타 통신패스의 연결형태 등에서 유리한 그림 5의 (a)에서 입력 데이타 a의 흐름을 b, n와 같은 방향으로 직렬(serial)형태로 변형하고 입력 데이타 a는 한 단위시간의 전달지연으로 다음 처리요소로 이동되며 동작을 제어하기 위하여 두 단위시간의 전달지연을 갖는 제어신호(ctl)를 입력시켜 사용한다. 또한 데이타 a의 최초의 입력되는 두 비트는 두 단위시간의 지연이 요구되므로 블럭 파이프라인의 주기는 '3'이 된다. 따라서 결과의 일차원 시스톨릭어레이에서

데이타 통신패스는 단일방향으로만 존재하므로 확장가능한 모듈을 구성하고 어레이의 VLSI구현에서 결함허용성이 용이하다.

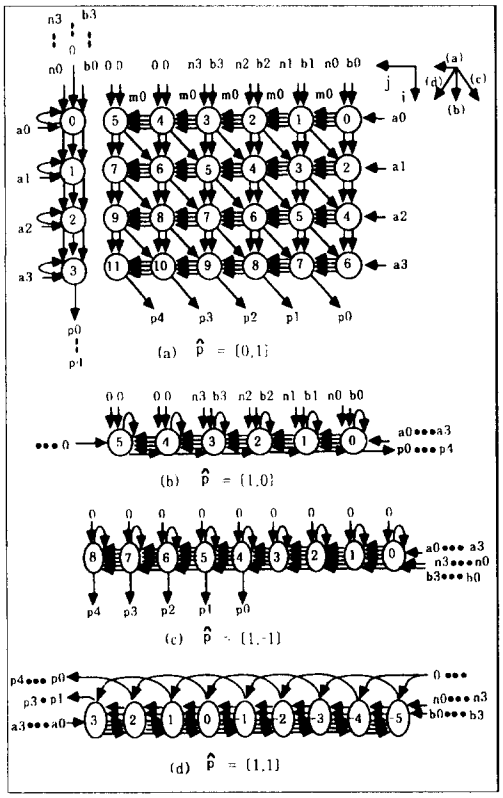



그림 5. 일차원 시스톨릭어레이의 설계 (n=4)
Fig. 5. The design of 1-D systolic array.

문제크기가 n인 경우, 기본적인 4가지 투영방향 ((a)~(d))과 제어로직을 추가하여 설계한 투영방향 [0,1] (a')의 일차원시스톨릭어레이를 여러가지 평가기준으로 고찰하여 보면 다음과 같다. 요구되는 처리요소(A)는 그림 5에서 직관적으로 구해지고, 처리시간(T)은 다음의 식(8)에서 구해진다[6].

$$T=\lceil (\max\{ \Pi(\bar{I}_1-\bar{I}_2)+1\} / \mathrm{disp}\Pi \rceil = 3n \quad (8)$$

여기서, disp$\Pi$는 1이고, $\bar{I}_1$와 $\bar{I}_2$는 문제의 임의의 두 인덱스점이다.

처리요소이용률($\mu$)은 전체 처리요소(A)에 대하여 처리시간(T)동안 사용된 처리요소의 비로서 설계한 일차원 시스톨릭어레이에서는 $\mu = n(n+2)/AT$ 가 된다. 파이프라인주기($\alpha$)는 $\Pi\hat{p}_i$ 이고, 요구되는 입출력 채널수(c)는 그림 5에서 직관적으로 구해지며 VLSI 구현시 I/O핀 수에 해당하므로 중요한 평가기준이 된다. 따라서 이러한 평가기준으로 각각을 비교하면 표 1과 같다.

표 1. 일차원 시스톨릭어레이의 비교
Table 1. The comparison of 1-D systolic array.

| projection \ criteria | processing elements | processor utilization | pipeline period | I/O channel |
|---|---|---|---|---|
| (a) [0,1] | n | $\frac{n+2}{3n}$ | 1 | $O(n)$ |
| (a') [0,1] | n | $\frac{n+2}{3n}$ | 1 | $O(1)$ |
| (b) [1,0] | n+2 | $\frac{n}{3n}$ | 2 | $O(n)$ |
| (c) [1,-1] | 2n+1 | $\frac{n+2}{6n+3}$ | 1 | $O(n)$ |
| (d) [1,1] | 2n+1 | $\frac{n+2}{6n+3}$ | 3 | $O(1)$ |

여기서 (a')의 투영방향 [0,1]은 처리요소의 이용률이 가장 높으므로 입력 채널을 제어신호로 조정함으로써 문제의 크기에 관계없이 입출력 채널의 수가 상수개로 고정된다. 한편 (d)의 투영방향 [1,1]의 경우는 [0,1]에 비하여 처리요소의 수정없이 설계가 간단하며 I/O 핀수가 고정되어 모듈 확장이 용이하나 처리요소의 이용률이 떨어진다.

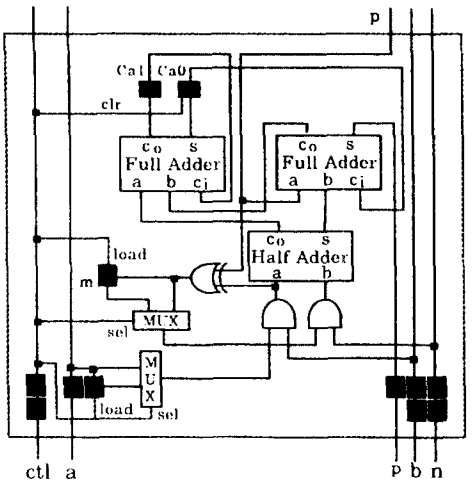
2. 처리요소의 내부설계

설계한 일차원 시스톨릭 어레이에서 각 처리요소의 내부구조는 투영방향이 [0,1]인 제어로직을 추가하여 그림 6과 같이 구성한다.

구성된 어레이에서 게이트를 통하여 모듈러 승산을 수행하고 제어값과 입력값을 쉬프트 연산으로 다음 PE로 전달한다. 여기서 제어신호에 따라 입력레지스터값(a0,b,n)들은 초기화하거나 이전 PE로부터 전달받으며 내부레지스터(a1,m,ca)의 값은 자기루프한다.

입력 a를 제어하기 위하여 제어신호 ctl를 지연시키기 위한 2비트 쉬프트레지스터와 2×1 멀티플렉서가 요구되고, b, n값은 2단위시간 지연후의 입력값으로 2개의 레지스터가 요구된다. 또한 제어신호 ctl에 따라 모듈러스 m을 저장하기 위한 플립플롭 1개, XOR 게이트 1개, 2×1 멀티플렉서 1개와 캐리 ca0, ca1을 저장하기 위한 플립플롭 2개가 요구된다. 전체적으로 결과값 p를 생성하기 위하여 2개의 AND 게이트, 1개의 HA, 2개의 FA가 필요하며 이를 저장하기 위한 1개의

레지스터가 요구된다.




where, input is ctl,a,b,n and ouput is p.

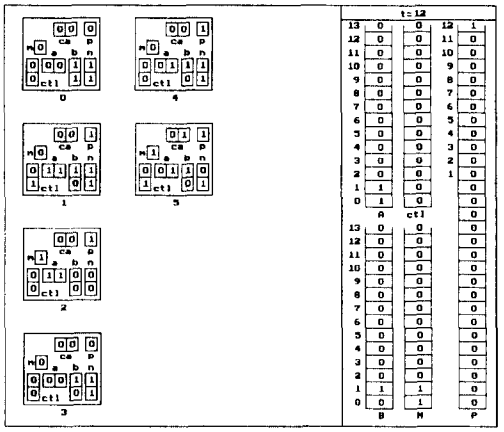
그림 6. 처리요소의 내부구조
Fig. 6. The internal structure of PE.

## Ⅳ. 컴퓨터 시뮬레이션 및 고찰

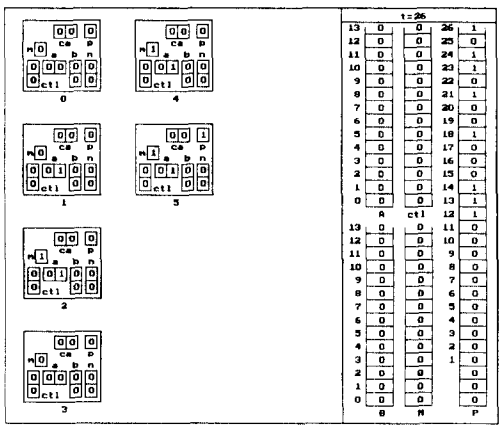
설계한 모듈러 승산을 위한 일차원 시스톨릭어레이의 정확한 내부동작과 평가를 검증하기 위하여 컴퓨터 시뮬레이션을 이행하였다. 시뮬레이션 환경으로 486 PC에서 C언어로 작성하였으며 화면 크기관계로 문제 크기는 n=12로 제한하였고 시뮬레이션 과정에서의 화면구성은 그림 7와 같다. 여기서 좌측의 큰 사각형들은 각각 처리요소를 나타내며 그 밑의 수는 어레이에서의 위치이다.

또한 처리요소 내부의 작은 사각형의 셀은 플립플롭 또는 레지스터에 해당한다. 그림 7의 우측은 입출력 큐를 표시하며 세부적으로 입력 데이타 A, B, N과 제어신호 ctl은 아래방향으로 어레이에 입력되고, 계산결과의 출력값 P도 역시 위에서 아래방향으로 출력된다. 제어신호 ctl은 입력큐에 있는 A의 첫비트가 처리요소내로 입력될 때마다 '1'과 함께 입력되며 나머지는 '0'으로 입력된다. 즉 문제의 크기 n = 6 일 때 ctl = 100000 이 된다. 여기서 문제의 크기는 입력데이타의 최대 비트수이다.

예를 들면 연속적인 입력데이타 A = 53 ; 54, B =42 ; 46, N = 61 ; 63 이고 예상되는 계산 결과의 출력 데이타 P = 71 ; 90 또는 10 ; 27 인 경우, 컴퓨터 시뮬레이션한 결과는 그림 7와 같다.



(a)



(b)

그림 7. 일차원 시스톨릭어레이의 snap-shots (n=6)
Fig. 7. The snap-shots of 1-D systolic array (n=6). (a) t = 12. (b) t = 26.

이때 t = 12, 26 일 경우 어레이내의 각 처리요소들과 입출력큐의 내용을 나타내면 각각 그림 7의 (a), (b)와 같고 입력 A, B, N 과 출력 P를 LSB에서 MSB 방향으로 비트단위로 표시하면 다음과 같다.

A : 10101100011011, B : 01010100011101,
N : 10111100111111, P : 1110001 0 0101101 또는
P' : 0101000 0 1101100

여기서 예상되는 출력값 P는 상위 캐리가 발생하였을 경우이고 P'은 상위 캐리가 발생하지 않을 경우이다. 블럭 파이프라인 주기가 '3' 이므로 입력데이타 A, B, N 내에서 연속되는 입력값을 처리하기 위하여 두 입력값사이에 두개의 '0'을 두어 입력을 지연시켰다. 최하위 비트부터 한 비트씩 입력 큐에서 처리요소로 입

력되도록 처리하며 최종 출력값에서 두 문제의 단위 출력값사이에 한 개의 의미없는 값 '0'이 존재한다.

그림 7의 (a)는 t = 12 단위시간에 각 처리요소의 상태를 보이고 첫번째 결과값의 첫비트가 마지막 처리요소에서 출력되어 출력큐에 저장되기 시작한다. (b)는 t = 26 단위시간에 최종결과값의 마지막 비트가 마지막 처리요소에서 출력되어 전체 계산결과의 내용이 출력큐에 저장되어 출력된다.

문제의 크기가 다른 여러가지 경우의 입력값에 대해서도 컴퓨터 시뮬레이션을 수행하였고, 정확한 계산결과가 출력됨을 확인하였다. 본 논문에서 설계한 일차원 시스톨릭어레이를 이용하여 적정규모의 VLSI 어레이를 제작한 후 필요한 개수대로 간단하게 연결함으로써, 확장가능한 모듈로 구성하여 그림 8의 RSA 암호시스템의 암호화 및 복호화과정에서 주요연산인 모듈러 멱승장치를 구성할 수 있다. 또한 고속처리를 위하여 처리요소의 이용율을 높힘으로써 하드웨어 요구량과 복잡도를 줄일 수 있다.

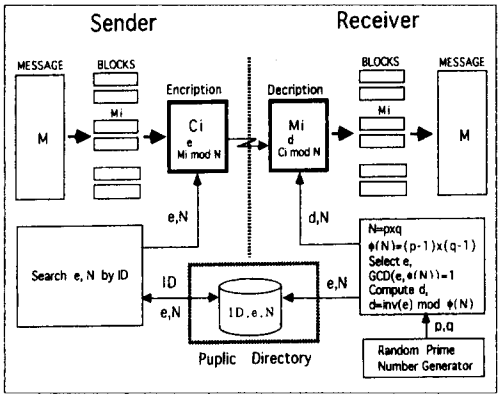



그림 8. RSA 암호시스템의 구성도
Fig. 8. The block diagram of RSA Cryptosystem.

## V. 결 론

RSA 암호시스템에 적합하며 모듈러 승산을 위한 확장가능한 모듈로 구성된 일차원 시스톨릭어레이를 설계하였다. 먼저 몽고메리 알고리즘에 근거한 Walter 방식을 병렬 알고리즘으로 재구성한 후 데이타 의존 그래프를 유도하였다. 그리고 시간 변환 및 공간 변환을 이행하는 체계적인 방법으로 일차원 시스톨릭어레이들을 설계하였다. 여기서 처리요소의 수, 데이타 통신패스의 연결형태와 확장가능한 모듈의 구성 등에서 유리한 투영방향 [0,1]의 어레이를 선택하여, 입력데이타 A의 흐름을 B, N과 같은 방향으로 직렬로 입력시키고 전달지연을 조정하기 위하여 1비트의 제어신호를 사용하여 어레이를 변형하여 문제의 크기에 의존하지 않는 입출력 채널을 요구하는 최적의 일차원 어레이를 설계하였다.

설계된 일차원 시스톨릭어레이에서 데이타 통신패스는 단일방향이므로 어레이의 VLSI 구현에서 결합허용성이 용이하다. 여러가지 평가기준에서 다른 방향의 시스톨릭어레이와 비교해 본 결과 확장가능한 모듈로 구성된 모듈러 승산을 위한 일차원 시스톨릭어레이가 설계되었다. 설계된 일차원 시스톨릭어레이는 특히 연속적이고 큰 수의 모듈러 승산이 필요한 RSA 암호시스템에 적합하다.

## 참 고 문 헌

[1] H. S. Hwang, et al., "An Implementation and Analysis of Modular Multiplication on PC," *KIISC Review*, Vol. 4, No. 3, pp. 34-60, 1994.

[2] S. R. Dussé, et al., "A Cryptography Library for the Motorola DSP 56000," *Proc. EUROCRYPT '90*, pp. 230-244, 1990.

[3] J. Sauerbrey, "A Modular Exponentiation Unit based on Systolic Arrays," *Abst. AUSCRYPT '92*, pp. 12.19-12.24, 1992.

[4] K. Iwamura, et al., "Systolic Arrays for Modular Exponentiation Using Montgomery Method," *Proc. EUROCRYPT '92*, pp. 477-481, 1992.

[5] C. D. Walter, "Systolic Modular Multiplication," *IEEE Trans. Comp.*, Vol 42-3, pp. 376-378, 1993.

[6] S. Y. Kung, "VLSI Array Processor", *Prentice Hall*, 1986.

## 저 자 소 개

**崔 盛 旭(正會員)**

1966년 4월 6일생 1992년 2월 동아대학교 공과대학 산업공학과 (공학사) 1995년 8월 부경대학교 산업대학원 전자 및 컴퓨터공학과 (공학석사) 1995년 9월 ~ 현재 한국해양대학교 대학원 전자통신공학과 (박사과정), 1994년 2월 ~ 1995년 2월 부경대학교 전자공학과 조교. 1995년 3월 ~ 1995년 8월 동명전문대학 시간강사(전자계산). 1995년 9월 ~ 현재 한국해양대학교 전자통신공학과 조교. 주관심분야는 암호·디지탈공학, 뉴로·퍼지 제어시스템공학

**禹 鍾 鎭(正會員)** 第 33卷 A編 第 5號 參照

현재 부경대학교 컴퓨터공학과 교수