모듈러 곱셈연산을 위한 비트레벨 시스토릭 어레이

최 성 욱
한국해양대학교 대학원 전자통신공학과

Bit-Level Systolic Array for Modular Multiplication

Sung Wook Choi
Dept. of Elect. & Comm. Eng., Graduate School, Korea Maritime Univ.

Abstract

In this paper, the bit-level 1-dimensionl systolic array for modular multiplication are designed. First of all, the parallel algorithms and data dependence graphs from Walter's Iwamura's methods based on Montgomery Algorithm for modular multiplication are derived and compared. Since Walter's method has the smaller computational index points in data dependence graph than Iwamura's, it is selected as the base algorithm. By the systematic procedure for systolic array design, four 1-dimensional systolic arrays are obtained and then are evaluated by various criteria. Modifying the array derived from [0,1] projection direction by adding a control logic and serializing the communication paths of data A, optimal 1-dimensional systolic array is designed. It has constant I/O channels for modular expandable and is good for fault tolerance due to unidirectional paths. And so, it is suitable for RSA Cryptosystem which deals with the large size and many consecutive message blocks.

I. 서 론

현대 암호시스템에서 공개키암호시스템은 키의 분배와 관리가 쉽고, 인증 및 디지탈서명이 가능한 장점을 가지고 있으나 512비트이상의 큰 수에 대해 모듈러 지수연산을 요구함으로써 기본연산인 모듈러 곱셈연산의 고속화를 위한 VLSI 하드웨어 구현이 필요하게 되었다.[1,5]

RSA와 같은 공개키암호시스템의 암호화 및 복호화과정에서 주요 연산은 M^e mod N 의 모듈러 지수연산으로서 모듈러 곱셈연산이 전체 계산의 대부분을 차지하며, 그중에서 몽고메리 모듈러 곱셈연산 알고리즘이 가장 효과적인 것으로 알려져 있다.[2-5]

이러한 모듈러 곱셈연산은 문제규모가 크고 많은 수의 메세지블럭이 연속적으로 계산되어야 하므로 입출력 채널의 수를 상수개로 고정시켜서 VLSI 칩으로 제작할 경우, 문제 크기에 관계없이 입출력 핀의 수를 고정시킬 수 있는 모듈러 확장가능성의 개념이 매우 중요하다.

1992년 Sauerbrey[8]는 Atrubin-Array를 이용하여 두개의 시스토릭 어레이에 기반을 둔 Montgomery 모듈러 곱셈연산기를 구성하여 512비트의 모듈러 지수연산기로 구현시 1.2 bps/gate (215Kbps/178Kgate)의 구현효율을 보였고, 1992년 Iwamura, 등[9]은 몽고메리알고리즘을 이용하여 일차원 시스토릭 어레이를 설계하여 512비트의 모듈러 지수연산기로 구현시 2.0 bps/gate (50Kbps/25 Kgate)의 고속처리가 가능하였으며, 1993년 Walter[10]는 모듈러 곱셈연산을 위한 이차원 시스토릭 어레이를 제안하였다.

본 논문에서는 체계적인 시스토릭 어레이 설계기법[6-7]을 적용하여 Iwamura와 Walter가 제안한 몽고메리알고리즘을 근거로 병렬알고리즘으로 유도한 후 데이타의존그래프로 표현하고 이를 시간 및 공간변환을 수행하여 일차원 시스토릭 어레이를 설계하였다. 그리고 설계한 일차원 시스토릭 어레이를 모듈러 확장가능성을 고려하여 여러가지 평가기준에 따라 비교, 분석하였으며, 처리요소를 설계하였고, 컴퓨터시뮬레이션을 수행하여 설계의 정확성과 유효함을 검증하였다.

Ⅱ. Montgomery 병렬알고리즘

1. Montgomery 알고리즘

P. L. Montgomery는 주어진 정수 N에 대한 모듈러 곱셈연산알고리즘을 제안했다. 이에 관련된 몇가지 정의는 다음과 같다. [3,5]

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

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

임의의 a, b ∈ Z_N의 f (a), f (b) ∈ 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 환동형사상을 갖는다.

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

여기서 $f(a) = aR$ 이다.

각 환은 함수 f 가 일대일대응사상으로 역함수 f^{-1}를 가지며 각 환에서 정의된 연산은 다음의 성질을 만족한다.

임의의 a, b ∈ Z_N 에 대하여

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

따라서 정의 3의 두집합간의 환동형사상은 그림1과 같다.

정의 1, 2, 3 을 만족하는 Montgomery 모듈러 정수환에서 GCD(R,N) = 1, R > N, $R = 2^{vn} = r^n$, N이 홀수일 때, 임의의 정수 a, b에 대하여 Montgomery 모듈러 곱셈연산 MP(a,b,N,R)는 정수가 되며 다음과 같다.

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

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

Montgomery 모듈러 정수환에서 어떤 큰 수를 r^n으로 나누었을 때 몫과 나머지를 구하는 것은 각각 단지 그 수의 n자리 이상의 수를 취하는 것과 n자리 미만의 수를 취하는 것이다. r^n에 대한 나눗셈은 매우 쉽다는 성질을 이용한다.

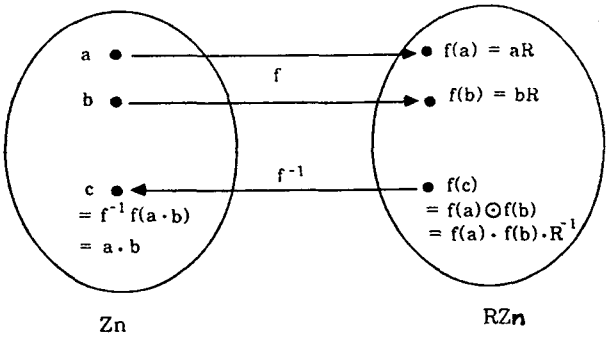


그림 1. 몽고메리 모듈러 곱셈연산을 위한 환동형사상

2. Montgomery 모듈러곱셈연산알고리즘

Montgomery 모듈러 곱셈연산 알고리즘은 그림 2와 같고 A, B, N, R, P는 다정도 정수이며 a_i, p_i는 A, P의 i번째 자리 수이다. [3-5]

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

Step 1 : $n_0' \leftarrow -n_0^{-1} \bmod r$
Step 2 : $P_{-1} \leftarrow 0$ // Initial value.
Step 3 : for $i \leftarrow 0$ to $n-1$
$P_i \leftarrow P_{i-1} + a_i \times B \times r^i$
$m_i \leftarrow p_i \times n_0' \bmod r$
$P_i \leftarrow P_i + m_i \times N \times r^i$
Step 4 : $P \leftarrow P_i \text{ div } R$
Step 5 : return P

그림 2. Montgomery 모듈러 곱셈연산 알고리즘

여기서 모듈러스의 배수 m_i는 p_i에 의하여 계산되고 그 결과를 P_i의 계산에 이용하며 m_i를 별셈연산하기 보다는 덧셈연산을 수행한다. 또한 피승수 A의 처리순서를 역전시켜 A의 낮은자리수부터 계산하고, 각 부분 결과를 윗자리로 이동시키지 않고 아래자리로 이동시킨다.

3. Walter 방식의 병렬알고리즘

Walter는 그림 2의 Step3에 해당하는 순환부분을 아래와 같이 변형하였다.

for $i \leftarrow 0$ to $n-1$
$m_i \leftarrow ((P_{i-1} \bmod r) + a_i \times b_0) \times n_0' \bmod r$
$P_i \leftarrow (P_{i-1} + a_i \times B + m_i \times N) \text{ div } r$

그림 2의 알고리즘과 비교하면 비트레벨(bit level)에서 처리가 가능하도록 최종 결과값을 나눗셈연산을 하는 대신 각각의 부분결과값에서 나눗셈 연산을 수행하였으며 병렬처리가 용이하도록 각 계산점에서 의존관계의 수를 줄였다. 이 알고리즘을 근거로 병렬알고리즘을 구성하면 그림 3과 같으며 비트레벨에서 짝수 r과 서로소인 n_0'의 값은 항상 1이 되므로 이 연산은 생략된다. 또한 캐리비트는 웨이트를 가진 상위캐리비트와 웨이트가 없는 하위캐리비트로 분리되어 계산되며 부분결과값은 하위캐리비트만을 이용한다. 초기값 및 계산과정에서 이전단계의 값을 저장하기 위하여 변수 $\overline{p}_{i,j}$, $\overline{ca0}_{i,j}$, $\overline{ca1}_{i,j}$를 사용하였다.

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 : $b_{i,j} \leftarrow b_j$, $n_{i,j} \leftarrow n_j$, $\overline{p}_{i,j} \leftarrow 0$
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 : $\overline{p}_{i,j} \leftarrow 0$
9 : else
10: $\overline{p}_{i,j} \leftarrow p_{i-1,j+1}$
11: end if
12: end if
13: if $j = 0$ then
14: $a_{i,j} \leftarrow a_i$, $\overline{ca0}_{i,j} \leftarrow 0$, $\overline{ca1}_{i,j} \leftarrow 0$
15: $m_{i,j} \leftarrow ((\overline{p}_{i,j} \bmod r) + a_i \times b_{i-1,j}) \bmod r$
16: else
17: $a_{i,j} \leftarrow a_{i,j-1}$, $m_{i,j} \leftarrow m_{i,j-1}$, $\overline{ca0}_{i,j} \leftarrow ca0_{i,j-1}$, $\overline{ca1}_{i,j} \leftarrow ca1_{i,j-1}$
18: end if
19: $p_{i,j} \leftarrow (\overline{p}_{i,j} + a_{i,j} \times b_{i,j} + m_{i,j} \times n_{i,j} + \overline{ca0}_{i,j}) \bmod r$

20: $ca0_{i,j} \leftarrow (\bar{p}_{i,j} + a_{i,j} \times b_{i,j} + m_{i,j} \times n_{i,j} + \overline{ca0}_{i,j} + \overline{ca1}_{i,j} \times r)$ div r mod r
21: $ca1_{i,j} \leftarrow (\bar{p}_{i,j} + a_{i,j} \times b_{i,j} + m_{i,j} \times n_{i,j} + \overline{ca0}_{i,j} + \overline{ca1}_{i,j} \times r)$ div r^2
22: all for
23:end

그림 3. Walter방식 병렬알고리즘

4. Iwamura 방식의 병렬알고리즘

Iwamura, 등은 그림 2의 순환부분을 아래와 같이 변형하였다.

for $i \leftarrow 0$ to n

$m_i \leftarrow (P_{i-1} \mod r) \times n_o' \mod r$

$P_i \leftarrow (P_{i-1} + a_i \times B \times r + m_i \times N)$ div r

여기서 모듈러스의 배수 m_i를 계산하기 위하여 요구되었던 a_i, b_0의 곱셈연산부분이 제거되었고 승수 B값을 인덱스 j인 2차원 계산공간으로 확장한 상태에서 LSB의 한자리수를 left-shift하여 입력하거나 $r=2^v$ 일때 $a_{i,j-1} \times b_{i+1,j}$의 결과값을 MSB방향으로 v비트 쉬프트하여 계산점내에서 처리된다. 이것을 병렬알고리즘으로 구성하면 그림 4와 같다.

1 :begin
2 : for all $0 \le i \le n-1, 0 \le j \le n+2$ do in parallel
3 : if $i = 0$ then
4 : $b_{i,j} \leftarrow b_j$, $n_{i,j} \leftarrow n_j$, $\bar{p}_{i,j} \leftarrow 0$
5 : else
6 : $b_{i,j} \leftarrow b_{i-1,j}$, $n_{i,j} \leftarrow n_{i-1,j}$
7 : if $j = n+2$ then
8 : $\bar{p}_{i,j} \leftarrow 0$
9 : else
10: $\bar{p}_{i,j} \leftarrow p_{i-1,j+1}$
11: end if
12: end if
13: if $j = 0$ then
14: $a_{i,j} \leftarrow a_i$, $\overline{ca0}_{i,j} \leftarrow 0$, $\overline{ca1}_{i,j} \leftarrow 0$
15: $m_{i,j} \leftarrow \bar{p}_{i,j} \mod r$
16: else
17: $a_{i,j} \leftarrow a_{i,j-1}$, $m_{i,j} \leftarrow m_{i,j-1}$, $\overline{ca0}_{i,j} \leftarrow ca0_{i,j-1}$, $\overline{ca1}_{i,j} \leftarrow ca1_{i,j-1}$
18: end if
19: $p_{i,j} \leftarrow (\bar{p}_{i,j} + a_{i,j} \times b_{i,j} \times r + m_{i,j} \times n_{i,j} + \overline{ca0}_{i,j}) \mod r$
20: $ca0_{i,j} \leftarrow (\bar{p}_{i,j} + a_{i,j} \times b_{i,j} \times r + m_{i,j} \times n_{i,j} + \overline{ca0}_{i,j} + \overline{ca1}_{i,j} \times r)$ div r mod r
21: $ca1_{i,j} \leftarrow (\bar{p}_{i,j} + a_{i,j} \times b_{i,j} \times r + m_{i,j} \times n_{i,j} + \overline{ca0}_{i,j} + \overline{ca1}_{i,j} \times r)$ div r^2
22: all for
23:end

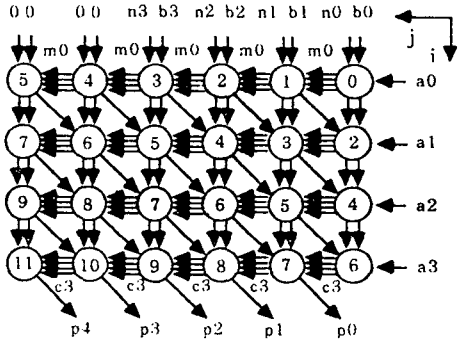
그림 4. Iwamura방식 병렬알고리즘

Ⅲ. 시스토릭 어레이의 설계

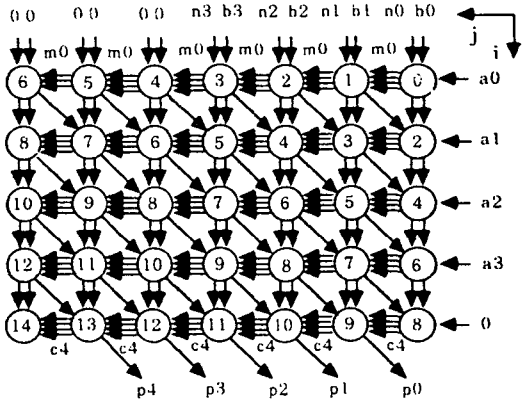
1. 데이타 의존 그래프

병렬알고리즘내에서의 데이타의존관계를 그래프로 나타낼 때 노드는 계산부분을, 아크는 데이타의존관계를 나타낸다. 이것을 VLSI 시스토릭어레이 구조로 구현하면 각 노드는 처리요소가 되고 데이타의존은 통신패스(path)가 된다.

그림 5의 (a)는 문제크기가 4인 경우 Walter방식에 의한 데이타의존그래프를 나타내며 j=0인 열의 계산점이 나머지 계산열과 다르고 인덱스 j의 방향에서 캐리 두비트를 계산에 포함시키기 위하여 문제의 크기보다 두개의 계산열이 더 요구된다. 그림 5의 (b)는 Iwamura방법에 의한 데이타의존그래프로서 (a)에 비해서 한 행과 한 열이 더 요구된다. 그러므로 두 방식 중 j=0인 계산열의 계산과정은 (b)가 단순하나 나머지 계산열은 동일하므로 전체적으로 볼 때 (a)의 Walter방식이 더 단순하고 계산행과 계산열의 수도 각각 한 개씩 적게 요구되어 시스토릭어레이에서 처리요소 수가 적어진다. 따라서 본 논문에서는 Walter방식을 기본알고리즘으로 채택하여 시스토릭어레이를 설계한다.



(a) Walter 방식 의존그래프



(b) Iwamura 방식 의존그래프

그림 5. 데이타의존그래프 (n=4)

2. 시간 및 공간 변환

알고리즘에서 시스토릭어레이로의 사상은 인덱스공간의 계산점 I와 어레이에서 시간 및 공간의 계산점 Y가 주어질 때 Y=T(I)의 관계를 가진다. 이때 T는 일대일의 사상으로 식(1)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 인덱스공간의 차수, Z는 정수의 집합이다.

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

그림 5(a)의 데이타의존그래프에서 각 변수들에 대한 데이타의존벡타들을 의존행렬(D)로 표시하면 식(2)와 같다.

$$D = [\hat{e}_a\ \hat{e}_b\ \hat{e}_{ca1}\ \hat{e}_{ca2}\ \hat{e}_m\ \hat{e}_n\ \hat{e}_p] = \begin{bmatrix}0&1&0&0&0&1&1\\1&0&1&1&1&0&-1\end{bmatrix} \tag{2}$$

그리고 식(1)의 조건에 따라 최적의 시간변환행렬(Π)과 투영 벡타($\hat{p}$)를 구하면 식(3)과 같다.

$$\Pi = [2\ 1],\ \hat{p}_1=[0\ 1],\hat{p}_2=[1\ 0],\ \hat{p}_3=[1\ -1],\hat{p}_4=[1\ 1] \tag{3}$$

공간변환(S)은 투영 벡타와 직교하므로 식(4)와 같다.

$$S_1=[1\ 0],\ S_2=[0\ 1],\ S_3=[1\ 1],\ S_4=[1\ -1] \tag{4}$$

그러므로 시간 및 공간변환함수로부터 변환된 의존행렬(T_iD)을 구하면 각각 식(5)과 같다.

$$T_1D = \begin{bmatrix} 1 & 2 & 1 & 1 & 1 & 2 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix},\ T_2D = \begin{bmatrix} 1 & 2 & 1 & 1 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & -1 \end{bmatrix},\ T_3D = \begin{bmatrix} 1 & 2 & 1 & 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$T_4D = \begin{bmatrix} 1 & 2 & 1 & 1 & 1 & 2 & 1 \\ -1 & 1 & -1 & -1 & -1 & 1 & 2 \end{bmatrix} \qquad (5)$$

데이타의존그래프에서 식(4)에 의해 각 노드의 공간변환을 수행하고 각각의 변환 T_iD에서 유도된 아크들의 방향으로 일차원시스토릭어레이를 설계하면 그림 6과 같다.

그림 6(a)의 경우는 입력 a가 어레이내에 선적재되기 위해서 문제크기만큼의 입력채널수가 요구되고, 이 a의 값에 의해서 m, cal, ca2가 계산되어 처리요소내에 머물게 된다. 또한 그림 6의 (b)도 입력 b, n가 선적재되기 위하여 문제크기만큼의 입력채널수가 각각 요구된다. 여기서 계산결과 p는 입력값과 반대방향으로 전달되므로 입력값 a와의 충돌을 피하기 위하여 최종출력값의 길이에 해당하는 시간지연이 주어져야 한다. 그림 6의 (c)는 모든 변수값이 링크를 따라 전달되므로 선적재가 요구되지는 않으나, 출력 채널수가 문제의 크기만큼 요구된다. 그림 6의 (d)는 자기루프(self-loop)가 제거됨으로써 선적재가 불필요하고, 입출력채널수가 상수개로 고정된다.

따라서, 모듈러 확장가능성을 고려하여 볼때 그림 6의 (d)가 가장 유리하나 처리요소 및 데이타통신패스(path)의 수가 (a)의 경우보다 많다. 그러므로 처리요소의 수와 데이타통신패스의 연결형태 등에서 유리한 그림 6의 (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 블럭파이프라인의 주기는 '3'이 된다. 결과의 일차원 시스토릭어레이에서 데이타통신패스는 단일방향으로만 존재하며, 이는 어레이의 VLSI구현에서 결함허용성에 유리하다.

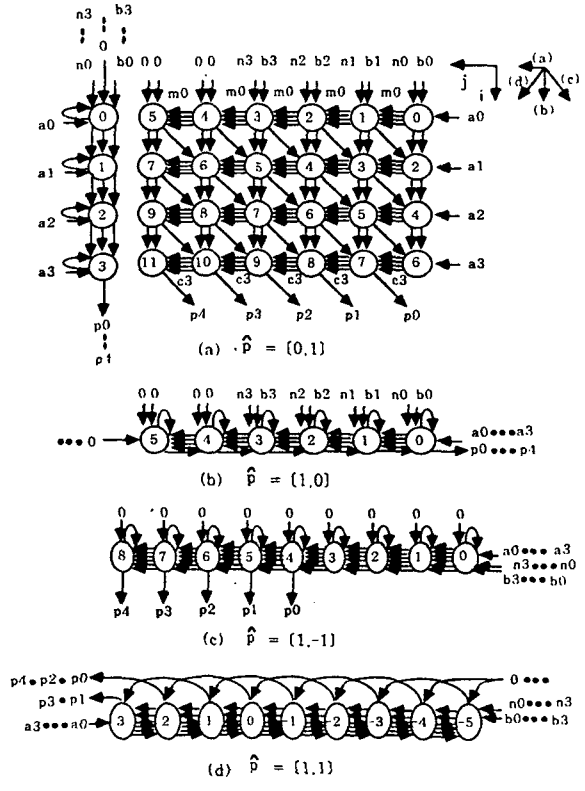


그림 6. 일차원 시스토릭어레이 설계

3. 처리요소의 내부설계

설계한 일차원 시스토릭어레이에서 각 처리요소의 내부구조는 투영방향이 [0,1]인 제어로직을 추가하여 그림 7과 같이 구성한다. 여기서 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 Half Adder, 2개의 Full Adder가 필요하며 이를 저장하기 위한 1개의 레지스터가 요구된다.

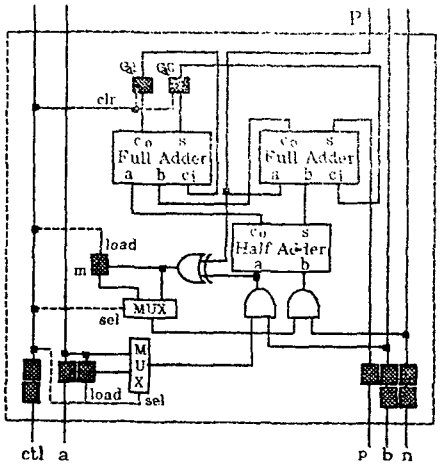


그림 7. 처리요소의 내부구조

Ⅳ. 결과 및 고찰

문제크기가 n인 경우, 투영방향 [0,1]에서 제어로직을 추가하여 설계한 일차원시스토릭어레이를 여러가지 평가기준[6-7]으로 고찰하여 보면 다음과 같다.

처리요소	처리시간	이용률	파이프라인 주기	블럭파이프라인 주기	입출력채널
n	3n	$\frac{n+2}{3n}$	1	3	$O(1)$

투영방향 [0,1]은 처리요소의 이용률이 가장 높으므로 입력채널을 제어신호로 조정함으로써 문제의 크기에 관계없이 입출력 채널의 수가 상수개로 고정된다. 한편 투영방향[1,1]의 경우는 [0,1]에 비하여 처리요소의 수정없이 설계가 간단하며 I/O 핀수가 고정되어 모듈러 확장이 용이하나 처리요소의 이용률이 떨어진다.

설계한 일차원 시스토릭어레이에서 각 처리요소의 내부동작은 그림 8과 같다. 여기서 단계1-5는 각입력값을 입력큐이거나 상위처리요소의 기억장소로부터 받을지를 결정하여 임시기억장소에 저장하는 부분이며, 단계6-22는 제어신호에 따라 결과값을 생성하는 주요부분이다. 그리고 단계23은 임시기억장소에 저장되었던 입력값을 다음 입력값 전달을 위하여 현재 입력값으로 재저장하는 부분이다.

Procedure of processing element located at (i), $0 \le i \le n-1$

1 : if i = 0 then
2 : ctl←ctlQ_0 , p← 0 , a←aQ_0 , b←bQ_0 , n←nQ_0
3 : else
4 : ctl←ctl$_{i,i-1}$, p←p$_{i-1}$, a←a0$_{i-1}$, b←b1$_{i-1}$, n← n1$_{i-1}$
5 : endif

```
6 : if  ctl = 1  then                  // Control_Logic enable
7 :   if  ctlQ0 = 1  then
8 :     cal_i ← 0
9 :   else
10:     ca0_i ← 0, al_i ← a
11:     m_i ←  p ⊕ (a∧b)        // Modulus multiple
12:     hs ← (a∧b) ⊕ (m_i∧n)  // Halfadder_Sum
13:     hc ← (a∧b) ∧ (m_i∧n)  // Halfadder_Carry
14:   endif
15: else                               // Control_Logic disable
16:   hs ← (al_i∧b) ⊕ (m_i∧n)
17:   hc ← (al_i∧b) ∧ (m_i∧n)
18:   fc ← (p∧hs ∨ (p⊕hs)∧ca0_i)  // Fulladder_Carry
19:   p_i ← p ⊕ hs ⊕ ca0_i             // Product
20:   ca0_i ← hc ⊕ fc ⊕ cal_i          // Lower_Carry
21:   cal_i ← hc∧fc ∨ (hc⊕fc)∧cal_i // Upper_Carry
22: endif
23: a0_i ← a, ctl1_i ← ctl0_i, ctl0_i ← ctl, bl_i ← b0_i, b0_i ← b, nl_i ← n0_i, n0_i ← n
```

그림 8. 처리요소의 내부동작과정

설계한 모듈러 곱셈연산을 위한 일차원 시스토릭어레이의 정확한 내부동작과 평가를 검증하기 위하여 컴퓨터시뮬레이션을 이행하였다. 시뮬레이션 과정에서의 화면구성은 그림 9와 같다. 여기서 좌측의 큰 사각형들은 각각 처리요소를 나타내며 그 밑의 수는 어레이에서의 위치이다. 또한 처리요소 내부의 작은 사각형의 셀은 플립플롭 또는 레지스터에 해당한다. 그림 9의 우측은 입출력큐를 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.

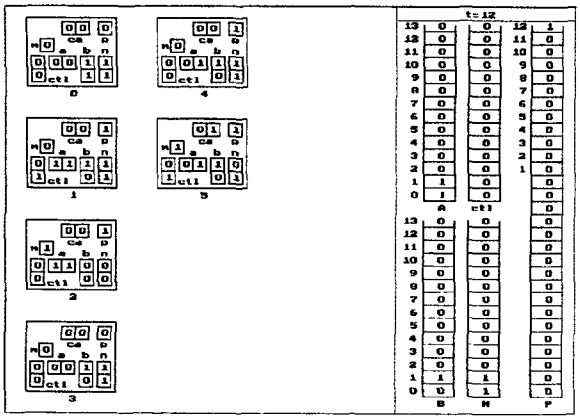
예를 들면 연속적인 입력데이타 A = 53 ; 54, B =42 ; 46, N = 61 ; 63 이고 예상되는 계산결과의 출력데이타 P = 71 ; 90 또는 10 ; 27 인 경우, 컴퓨터시뮬레이션한 결과는 그림 9와 같다. 이때 t = 12, 20, 26 일 경우 어레이내의 각 처리요소들과 입출력큐의 내용을 나타내면 각각 그림 9의 (a), (b), (c)와 같고 입력 A, B, N 과 출력 P를 LSB에서 MSB방향으로 비트단위로 표시하면 다음과 같다.

A : 101011 00 011011, B : 010101 00 011101, N : 101111 00 111111

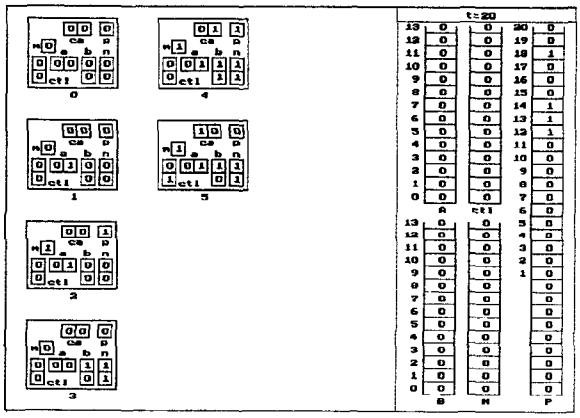
P : 1110001 0 0101101 또는 P' : 0101000 0 1101100

여기서 블럭파이프라인 주기가 '3' 이므로 입력데이타 A, B, N 내에서 연속되는 입력값을 처리하기 위하여 두 입력값 사이에 두개의 '0'을 두어 입력을 지연시켰다. 최하위비트부터 한 비트씩 입력큐에서 처리요소로 입력되도록 처리하며 최종출력값에서 두 문제의 단위출력값사이에 한 개의 의미없는 값 '0'이 존재한다. 그림 9의 (a)는 t = 12 단위시간에 각 처리요소의 상태를 보이고 첫번째 결과값의 첫비트가 마지막 처리요소에서 출력되어 출력큐에 저장되기 시작한다. (b)는 t = 20 단위시간에 두번째 결과값의 첫비트가 마지막 처리요소에서 출력되어 출력큐에 저장되기 시작한다. (c)는 t = 26 단위시간에 최종결과값의 마지막 비트가 마지막 처리요소에서 출력되어 전체 계산결과의 내용이 출력큐에 저장되어 출력된다.

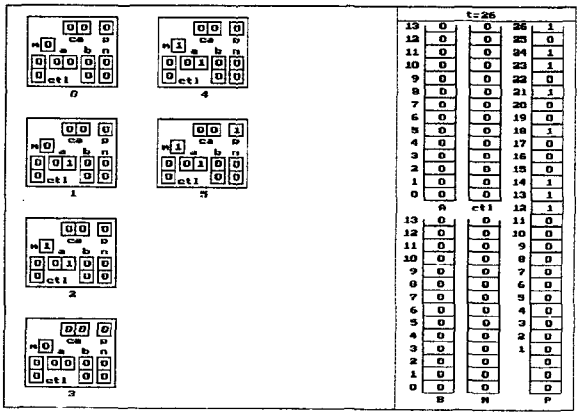
문제의 크기가 다른 여러가지 경우의 입력값에 대해서도 컴퓨터시뮬레이션을 수행하였고, 정확한 계산결과가 출력됨을 확인하였다. 본 논문에서 설계한 일차원 시스토릭어레이를 이용하여 적정규모의 VLSI 어레이를 제작한 후 필요한 개수대로 간단하게 연결함으로써, 모듈러 확장가능한 RSA암호시스템을 구성할 수 있다. 또한 고속처리를 위하여 처리요소의 이용율을 높힘으로써 하드웨어요구량과 복잡도를 줄일 수 있다.[11-12]



(a)



(b)



(c)

그림 9. 일차원시스토릭어레이의 snap-shots.

(a) t = 12 (b) t = 20 (c) t = 26

V. 결 론

RSA암호시스템에 적합하며 모듈러 곱셈연산을 위한 모듈러 확장가능한 일차원 시스토릭어레이를 설계하였다. 먼저 몽고메리 알고리즘에 근거한 Walter방식과 Iwamura방식을 각각 병렬알고리즘으로 재구성한 후 데이타의존그래프를 구하여 비교하였다. 그 결과 계산행과 계산열의 수가 한 개씩 적게 요구되어 시스토릭어레이의 설계에서 처리요소의 수가 적게 요구되는 Walter방식을 선택하였다. 여기에 시간변환 및 공간변환을 이행하는 체계적인 방법으로 일차원 시스토릭어레이들을 설계하였다. 여기서 처리요소의 수, 데이타통신패스의 연결형태와 모듈러확장가능성 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 입출력채널을 요구하는 최적의 일차원 어레이를 설계하였다.

설계된 일차원 시스토릭어레이에서 데이타통신패스는 단일방향이므로 어레이의 VLSI구현에서 결합허용성에 유리하다. 여러가지 평가기준에서 기존의 제안된 다른 시스토릭 어레이와 비교해 본 결과 최적인 모듈러 확장가능한 모듈러 곱셈연산을 위한 일차원 시스토릭어레이가 설계되었다. 설계된 일차원 시스토릭어레이는 특히 연속적이고 큰 수의 모듈러 곱셈연산이 필요한 RSA암호시스템에 적합하다.

정보보호의 중요성이 증가되는 현실에서 암호시스템에 대한 연구가 활발할 것이며 본 연구의 결과는 RSA 암호시스템에서 모듈러 지수연산장치의 VLSI 구현에 기초설계기술로 제공되며, 적절한 VLSI 설계 툴을 이용하여 실제의 FPGA(Field Programmable Gate Array) 등의 ASIC 구현이 가능하다.

참 고 문 헌

[1] D. E. R. Denning, "Cryptography and Data Security," *Addison-Wesley*, 1983.

[2] D. E. Knuth, "The Art of Computer Programming, Vol. 2, Seminumerical Algorithms, 2nd ed.," *Addison-Wesley*, 1981.

[3] P. L. Montgomery, "Modular Multiplication without Trial Division", *Mathemat. of Computat.*, Vol. 44, pp. 519-521, 1985.

[4] S. R. Dussé and B. S. Kaliski Jr. "A Cryptography Library for the Motorola DSP 56000," *Proc. EUROCRYPT '90*, pp. 230-244, 1990.

[5] H. S. Hwang, C. H. Lim, and P. J. Lee, "An Implementation and Analysis of Modular Multiplication on PC," *KIISC Review*, Vol. 4, No. 3, pp. 34-60, 1994.

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

[7] D. I. Modovan, "Modern Parallel Processing", *USC*, 1986.

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

[9] K. Iwamura, T. Matsumoto, and H. Imai, "Systolic Arrays for Modular Exponentiation Using Montgomery Method," *Proc. EUROCRYPT '92*, pp. 477-481, 1992.

[10] C. D. Walter, "Systolic Modular Multiplication," *IEEE Trans. Comp.*, Vol 42, No. 3, pp. 376-378, 1993.

[11] S. W. Choi, "Design of Systolic Array for Modular Multiplication suitable for RSA Cryptosystem," *Master thesis, Dept. Elect. and Comput. Eng., National Fisheries Univ. of Pusan, Korea*, 1995.

[12] 최성욱외 3인, "모듈러 곱셈연산을 위한 모듈러 확장가능한 시스토릭 어레이의 설계," *대한전자공학회 1995년도 하계종합학술대회 논문집*, pp. 353-356 , 1995.