循環 알고리즘의 Processor Array에로의 合成 및 具顯

李 德 秀* · 申 東 錫**

The Synthesizing Implementation of Iterative Algorithms on Processor Arrays

Deog-Su Lee · Dong-Suk Shin

〈目 次〉

Abstract

A systematic methodology for efficient implementation of processor arrays from regular iterative algorithms is proposed. One of the modern parallel processing array architectures is the Systolic arrays and we use it for processor arrays on this paper. On designing the systolic arrays, there are plenty of mapping functions which satisfy necessary conditions for its implementation to the time-space domain. In this paper, we use a few conditions to reduce the total number of computable mapping functions efficiently. As a result of applying this methodology, efficient designs of systolic arrays could be done with considerable saving on design time and efforts.

* 정회원, 한국해양대학
** 부산수산대학교

1. 서 론

컴퓨터 시스템의 성능을 향상 시키려는 노력으로서, 단일 프로세서 시스템이 가지는 기본적인 제약점을 극복하기 위해 다중 프로세서를 이용한 병렬 처리 방식에 관한 연구가 지속되어 왔다.[1] 병렬 처리 방식은 최근 VLSI의 급진적인 발전에 의하여 여러가지의 하드웨어적인 형태로 실현되고 있다. 대표적인 병렬 처리 방법에는 파이프라이닝, 어레이 프로세서 및 멀티 프로세서들이 있고, 보다 새로운 병렬 처리 기법으로서 시스토릭 프로세서(Systolic processors), 데이타 흐름 모델(Data flow models), Logic inference models 등이 있다.[2]

멀티 프로세서에 의한 병렬 처리 방식은 단일 프로세서에 비하면 그 성능이 현저하게 개선 되었지만 자원의 공유와 프로세서간, 혹은 프로세서와 I/O간에 요구되는 통신량이 대단히 많아져 통신 병목현상으로 인한 처리의 한계에 부딪히게 되었다. 이러한 현상을 해결하기 위하여 프로세서간의 통신을 지역적인 통신만으로 제한하고 프로세서간의 상호 연결이 규칙적으로 되어 있는 시스토릭 어레이 시스템이 카네기 멜론 대학에서 제안되어 수많은 알고리즘에 대한 시스토릭 구조(systolic architecture)가 개발되어 왔다.[3,6] 시스토릭 어레이는 처리되는 각 데이타 항목들의 통신을 국부적이고 지역적인 통신으로 제한 시키면서 구조가 간단한 다수의 처리요소(processing elements)들을 사용하여 고도의 파이프라이닝(Pipelining) 및 병렬성(Parallelism)을 실현한다.

하나의 알고리즘에 대한 시스토릭 어레이의 구현은 사용되는 cell의 갯수, 처리율, 총 실행 시간, cell간의 상호 연결 패턴 및 데이타의 흐름 방향 등에 따라 다양한 모양으로 설계된다. 본 논문에서는, 어떤 주어진 순환형의 알고리즘에 대하여 이를 인덱스 영역으로 효율적으로 모델화하는 방법과 데이타의 진행 방향 및 데이타 요소들의 순서들을 고려하여 실제적으로 필요한 사상들 중 진성의 사상들만을 선택하여 효과적으로 시스토릭 어레이를 설계하는 방법을 제시한다. 설계의 예로써 행렬곱 계산을 위한 시스토릭 어레이의 구현이 실행되는데 이것은 Moldovan[4]등에 의하여 제안된 방법과는 달리 수백개의 계산 가능한 사상들 중에서 불과 7개의 진성 사상만을 가려내어 설계에 사용하게 되므로 어레이 설계의 시간을 대폭 감축하게 된다.

2. 알고리즘의 어레이 사상

하나의 알고리즘에 관계되는 제반 요소들에 대하여 다음과 같이 정의한다.

R : $\{d^1, d^2, d^3, \cdots\cdots, d^m\}$
⇒ m개의 데이타 항목을 가진 알고리즘을 나타낸다.

Z^n : n차원 공간상의 정수좌표로 표시되는 점들의 집합

C : n차원 인덱스 공간상의 계산점을 나타내는 정수 좌표
$(C \subset Z^n)$

순환 반복형의 계산 집약적인 알고리즘의 실례로서 C=A·B의 행렬곱을 이용하기로 한다. 여기서 A, B, C는 각각 NxM, MxQ, 그리고 NxQ의 크기를 가진 행렬이고 그 요소들은 a_{ij}, b_{jk}, c_{ik}로 표시한다. 이 행렬곱을 수식적으로 표현하면

$$c_{ik}=\sum_{j=1}^{M} a_{ij} \cdot b_{jk} \quad 1 \le i \le N,\ 1 \le k \le Q \quad \cdots\cdots (1)$$

이 된다. 이 수식을 등가의 순환 방정식으로 바꾸면

$$\begin{aligned} c_{ijk} &= 0 && 0 \le j \le (M-1) \\ c_{ijk} &= c_{i(j-1)k} + a_{ij} \cdot b_{jk}, && 1 \le j \le M \quad \cdots\cdots (2) \\ c_{ik} &= c_{iMk} \end{aligned}$$

가 되며, 이것을 다시 파이프라이닝 하면 다음과 같다.

```
begin
 for i:=1 to N do
  for k:=1 to Q do
   for j:=1 to M do
    A(i, j, k):=A(i, j, k-1)
    B(i, j, k):=B(i-1, j, k)
    C(i, j, k):=A(i, j-1, k)
```

```
    endfor ;
  endfor ;
endfor ;
end
```

2. 1 인덱스 영역상의 데이타의 의존 및 배열관계

1) 의존벡터(Dd) … 인덱스 공간에서 어느 한 데이타 항목에 대하여 동일한 인덱스값을 가지는 두개 이상의 계산점이 있을 경우 이들 사이에는 데이타의 의존관계가 성립한다.[3,8] 데이타 의존관계가 성립하는 두개의 계산점에 대하여 그 공간 좌표의 差벡터의 크기가 최소가 되는 벡터를 그 데이타 항목의 의존벡터라 하고 Dd로 표시하며, 각 데이타 항목들의 의존벡터들을 요소로 하는 의존행렬을 D로 표시한다. 의존행렬은 알고리즘에 있어서 인덱스에 따른 각 변수들의 데이타 의존 관계를 종합적으로 나타낸다.

2) 배열벡터(Fd) … 인덱스 공간에서 각 데이타 항목에 속하는 데이타 요소들이 어떻게 배열되는가를 나타내주는 벡터를 배열벡터라 하며 이는 인덱스값이 1씩 증가하는 관계에 있는 데이타 요소간의 공간 좌표의 差벡터로서 Fd로 표시한다. 각 데이타 항목들의 배열벡터들을 요소로 하는 배열행렬을 F로 표시한다.

식 (3)은 3×3의 정방 행렬들에 의한 C=A · B의 행렬곱의 요소들을 인덱스에 의해 나타낸 것으로 이것을 파이프라이닝하면 Table 1과 같은 순서에 따라 계산이 이루어지게 되며 각각의 계산점들을 인덱스 공간상에 표시하면 Fig. 1과 같다.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \quad (3)$$

이 3차원 인덱스 공간상에 표시되는 계산점들의 상호 관계를 파악하여 데이타 항목 A에 대한 의존벡터와 배열벡터를 구하면 다음과 같다.

$$\begin{aligned} D_{ai} &= C4 - C1 = (1, 1, 2) - (1, 1, 1) = (0, 0, 1) \\ D_{aj} &= C4 - C1 = (1, 1, 2) - (1, 1, 1) = (0, 0, 1) \end{aligned} \quad \cdots\cdots (4)$$

$$\begin{aligned} F_{ai} &= C10 - C1 = (2, 1, 1) - (1, 1, 1) = (1, 0, 0) \\ F_{aj} &= C2 - C1 = (1, 2, 1) - (1, 1, 1) = (0, 1, 0) \end{aligned} \quad \cdots\cdots (5)$$

각 데이타 항목들에 대한 의존 및 배열벡터들을 요소로 하는 행렬은 다음과 같다.

$$D = \begin{bmatrix} 0\,0 & 1\,1 & 0\,0 \\ 0\,0 & 0\,0 & 1\,1 \\ 1\,1 & 0\,0 & 0\,0 \end{bmatrix} \quad \cdots\cdots (6)$$

$$F = \begin{bmatrix} 1\,0 & 0\,0 & 1\,0 \\ 0\,1 & 1\,0 & 0\,0 \\ 0\,0 & 0\,1 & 0\,1 \end{bmatrix} \quad \cdots\cdots (7)$$

Table 1. Allocation of computation points on index domain.

C(i, j, k)	Operation
1(1 1 1)	$c_{11}=c_{11}+a_{11} \cdot b_{11}$
2(1 2 1)	$c_{11}=c_{11}+a_{12} \cdot b_{21}$
3(1 3 1)	$c_{11}=c_{11}+a_{13} \cdot b_{31}$
4(1 1 2)	$c_{12}=c_{12}+a_{11} \cdot b_{12}$
⋮	⋮
7(1 1 3)	$c_{13}=c_{13}+a_{11} \cdot b_{13}$
⋮	⋮
27(3 3 3)	$c_{33}=c_{33}+a_{33} \cdot b_{33}$

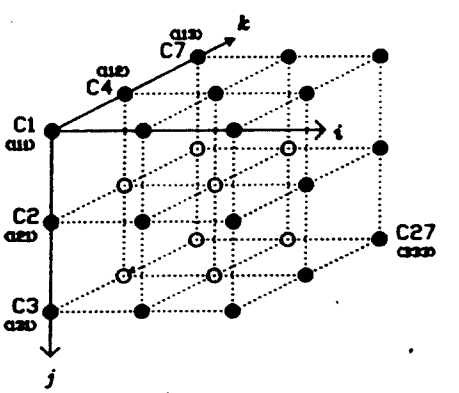


Fig. 1. 3−dimensional index space of matrix multiplication.

2.2 시간-공간 영역상의 데이타의 진행 및 배열관계

시스토릭 어레이의 구현은 알고리즘의 인덱스 공간상에서 일어나는 순차적인 계산들을 어레이 평면상의 처리 요소에 사상시키는 것이다.[6] 3차원의 인덱스 공간을 2차원의 어레이 평면상에 사상하는 것은 Fig. 2와 같이 인덱스 공간을 하나의 시간축을 포함하는 3차원의 시간-공간 영역으로 사상하는 것을 의미한다. 이때 x, y축에 의한 평면이 어레이 평면이 되며 이 어레이 평면상에 실제적인 처리 요소들이 자리잡게 된다. 동일한 x-y 평면상의 모든 처리 요소에서의 계산은 동시적으로 실행되며 단위 시간이 증가함에 따라 동일한 처리요소에서는 의존 관계가 성립하는 데이타의 입출력이 행해지면서 매번 새로운 계산이 이루어진다.

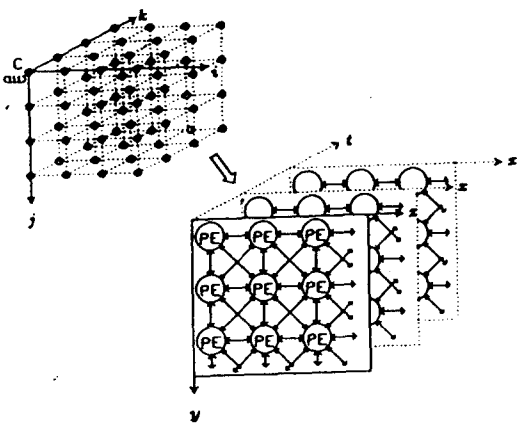


Fig. 2. The mapping of index domain into the time-space domain.

시간-공간 영역을 나타내는 세개의 축요소인 x, y, t의 상관 관계에 따라 하나의 어레이에 대하여 다음과 같이 벡터들을 분석할 수 있다.

1) 데이타 항목의 진행벡터

시간-공간 영역에서 어느 한 데이타 항목의 요소들에 대하여 단위 시간이 증가함에 따라 진행하는 방향을 나타내는 것으로 D'_d로 표시한다. 이것은 동일한 항목에 속하면서 인덱스값이 같은 계산점간의 差벡터이다.

2) 시간-공간 영역에서의 데이타 항목의 배열벡터

시간-공간 영역에서 어느 한 데이타 항목의 데이타 요소들이 어떻게 배열되는가를 나타내주는 것으로서 F'_d로 표시한다. 이것은 동일한 데이타 항목에 속하면서 단위 시간의 증가에 따라 인덱스값이 1씩 증가하는 관계에 있는 계산점간의 差벡터이다.

3) 공간 영역에서의 데이타 항목의 배열벡터

한 단위 시간내에 (x, y) 평면상에서 어느 한 데이타 항목의 데이타 요소들의 배열 형태를 나타내주는 것으로서 S_d로 표시한다. 이것은 동일한 시간 동안에 인덱스값이 1씩 증가하는 관계를 가진 x-y 평면상의 두 계산점 사이의 差벡터이다.

Fig. 3은 상기 세가지의 데이타 진행 및 배열 벡터들의 관계를 나타내고 있으며 각 경우에 대한 벡터들의 값은 식 (8)과 같다.

[Y : 시간-공간 영역에서의 하나의 계산점, Y(x, y, t)]

$$\begin{aligned}&\text{(a) } D'_A = Y_2 - Y_1 = (2,1,2) - (1,1,1) = (1,0,1)\\&\text{(b) } F'_B = Y_2 - Y_1 = (3,1,2) - (1,1,1) = (2,0,1)\\&\text{(c) } S'_A = Y_2 - Y_1 = (3,1,1) - (2,1,1) = (1,0,0)\end{aligned} \quad \cdots\cdots (8)$$

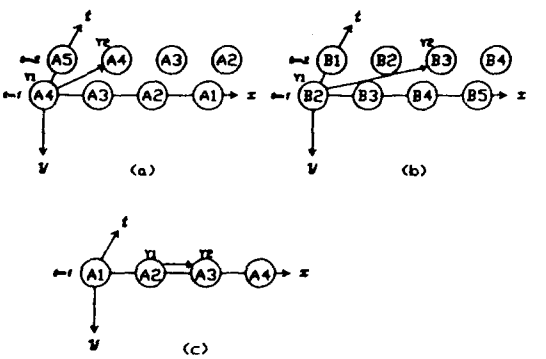


Fig. 3. (a)Propagation vector in time-space domain (b)Sequence vector in time-space domain (c)Sequence vector in space domain.

2.3 시스토릭 어레이로의 사상

알고리즘에 대한 시스토릭 어레이의 설계는 주어진 알고리즘의 인덱스 공간상의 계산점 C와 시간-공간 영역의 계산점 Y 사이에 Y=F(C)의 관계가 성립될 때 이루어지며 이때 F는 일대일 대응 관계의 함수로서 M으로 표시한다.[8] M에 의한 상호 관계를 정리하면 다음과 같다.

$$D'_d = M \cdot D_d, \quad F'_d = M \cdot F_d \quad \cdots\cdots (9)$$

$F'_d(t)$와 $D'_d(t)$를 각각 F'_d와 D'_d의 시간 요소라 할 때, 스칼라 a와 b를 $a \cdot F'_d(t) = b \cdot D'_d(t)$를 만족하는 정수라 하면 다음과 같은 관계가 성립한다.

$$a \cdot F'_d - b \cdot D'_d = a \cdot S'd \quad \cdots\cdots (10)$$

1) 시간사상(M_t)

시간사상(M_t)은 시스토릭 어레이의 특성을 만족하면서 동시에 다음의 조건을 만족해야 한다.

$$M_t \cdot D > 0 \quad \cdots\cdots (11)$$

여기서 $M_t \cdot D$는 데이타 의존벡터를 시간 변환한 것이며 식(11)의 조건을 만족하는 최적의 M_t는 주어진 알고리즘에서 처리 시간이 최소인 값이다.

2) 공간사상(M's)

공간사상은 다음의 수식들을 이용하여 구한다.

식(10)의 조건식인 $a \cdot F'_d(t) = b \cdot D'_d(t)$에서, α, β를 각각 스칼라 a, b의 행렬 표현이라 하면

$$\alpha \cdot M_t \cdot F - \beta \cdot M_t \cdot D = 0 \quad \cdots\cdots (12)$$

이 되며, 식 (10)을 M을 포함시켜서 정리하면

$$M \cdot \alpha \cdot F_d = \alpha \cdot S'_d + \beta \cdot D'_d \quad \cdots\cdots (13)$$

이 된다. M에 대한 조건식인 식 (9)와 식 (13)을 결합하면

$$M[Aug(D_d, \alpha \cdot F_d)] = Aug(D'_d, (\alpha \cdot S'_d + \beta \cdot D'_d)) \quad \cdots\cdots (14)$$

이다. 여기서 Aug(X, Y)는 X, Y 두개의 행렬을 컬럼 방향으로 결합하는 것을 나타낸다. 식 (14)의 우변에서 M_t를 나타내는 t의 행을 제외시키면

$$Ms \cdot [Aug(D_d, \alpha \cdot Fd)] = Aug_s(D'_d, (\alpha \cdot S'_d + \beta \cdot D'_d)) \quad \cdots\cdots (15)$$

이 되며, 따라서

$$M_s = Aug_s(D'_d, (\alpha \cdot S'_d + \beta \cdot D'_d)) \cdot Aug(D_d, \alpha \cdot F_d)^{-1} \quad \cdots\cdots (16)$$

이다.

3. 효과적인 어레이의 설계

하나의 알고리즘에서 식 (6)과 식 (7)을 만족하는 공간사상은 그 수가 대단히 많으며 각 데이타 항목의 진행 방향과 데이타 요소들의 배열 순서에 따라 어레이의 모양이 다양하게 변하게 된다. 본 연구에서는 효과적으로 어레이를 설계하기 위하여 다음과 같은 방법을 이용하여 사용 가능한 사상의 수를 줄였다.

① 어레이 평면상의 처리요소들 사이의 통신 패스의 수를 최소화 한다.

② 임의로 주어지는 F'_d와 D'_d에 의해 계산된 사상 M 행렬에 대하여 Det(M)=0인 것들은 제외시킨다.

③ 상호 대칭, 회전 또는 대칭 및 회전의 기하학적인 관계를 야기하는 M에 대하여는 그 중 어느 하나만을 선택한다.

3.1 통신 패스의 최소화

Fig. 4 (a)는 인접한 처리요소간에 가능한 모든 방향의 통신 패스를 나타내고 있으며 이것을 행렬로 표현한 것이 식 (17)이다. 통신 패스의 방향이 바뀜에 따라 사상이 여러 형태로 바뀌게 되는데 VLSI 설계 영역에서 적용될 수 있는 사항들을 다음과 같이 미리 적용함으로써 통신 패스의 수를 줄일 수 있다.

① 선택된 패스들을 갖는 처리요소들로 구성된 어레이는 그 패스들을 45°씩 회전시킨 처리요소들로 구성된 어레이와 동일한다.

② 2차원 평면인 VLSI의 구현에 적합하도록

하기 위해서는 교차(crossing) 상태의 패스는 제거하여야 한다.

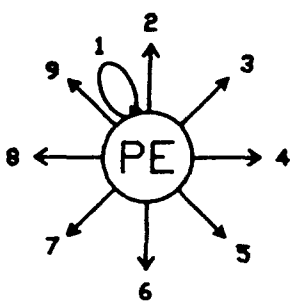
③ 처리요소에서 선택된 패스들이 좌우 대칭이면 모든 패스들을 좌우로 옮겨도 어레이는 변하지 않는다.

④ 가능한 서로 반대되는 방향으로의 패스들이 선택되지 않도록 주의한다.

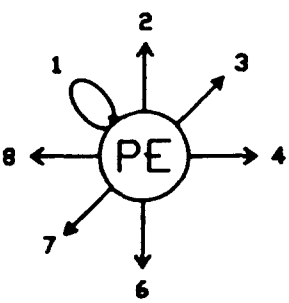
상기 조건들을 고려하게 되면 통신 패스는 Fig. 4 (b)와 같이 되며 그 행렬은 식 (18)과 같이 줄어들게 된다.

$$D'_d = \begin{array}{c} \begin{matrix} ① & ② & ③ & ④ & ⑤ & ⑥ & ⑦ & ⑧ & ⑨ \end{matrix} \\ \left| \begin{matrix} 0 & 0 & 1 & 1 & 1 & 0 & -1 & -1 & -1 \\ 0 & -1 & -1 & 0 & 1 & 1 & 1 & 0 & -1 \end{matrix} \right| \end{array} \quad \cdots\cdots (17)$$

$$D'_d = \begin{array}{c} \left| \begin{matrix} 0 & 0 & 1 & 1 & 0 & -1 & -1 \\ 0 & -1 & -1 & 0 & 1 & 1 & 0 \end{matrix} \right| \end{array} \quad \cdots\cdots (18)$$



(a)All possible comm. paths



(b)reduced comm. paths

Fig. 4.

3.2 眞性사상의 선택

최소화된 통신 패스들에 의해 구해진 사상들 중에는 시스토릭 어레이의 특성에 맞지 않아 실제적으로 어레이의 구현이 불가능한 것들이 있으며 또한 다른 구조를 대칭하거나 회전시키면 동일한 모양이 되어 중복되는 것들도 포함될 수 있다. 眞性사상이란 이러한 불필요한 사상들을 전부 제거하고 나서 남게되는 어레이 구현이 가능한 독자적인 형태의 것들을 말한다.

진성사상을 구하기 위한 조건들은 다음과 같다.

① 사상 M에 대하여 $\det(M) \neq 0$일 것.

② M_y가 서로 동일하면서 $M_{x1} + M_{x2} = 0$이면 상호 對稱 관계

③ $M_x = \pm M_y$이면서 $M_y = -M_x$이면 (대칭 및) 回轉 관계

조건 ①에서 $\det(M)=0$인 경우는 M의 대응하는 두 행(또는 열)의 요소가 비례하는 경우로서 어느 한점에서의 계산 활동이 단위 시간 내에 동시에 2회 이상 실시되어 어레이의 실제적인 구현이 불가능하게 되므로 배제한다. 조건 ②와 ③은 동일한 형태의 어레이가 설계되는 경우로서 이때에는 둘 중 어느 하나만을 선택한다.

이렇게 하여 구해진 사상 함수들에 의해서 어레이의 구현이 실현되며 최적의 어레이는 VLSI 구현의 최적화 요건에 따라 칩 면적(A)과 처리 속도(T) 등에 의하여 선택되어진다.[9] 그러나 사실상의 최적 시스토릭 어레이는 어레이의 응용 분야의 특수성에 따라 또는 계산의 내용에 따라 AT 또는 AT^2에 의한 판단 기준과 다를수도 있다.[5,7]

4. 시스토릭 어레이 설계의 實例

시스토릭 어레이 설계의 예로서, 알고리즘의 모델화에서 이용한 $A \cdot B = C$의 행렬곱 문제를 다루기로 한다. A, B 및 C의 각 행렬은 A=NxM, B=MxQ, C=NxQ의 크기를 가지고 있으며 인덱스 공간에서의 각 데이타 인덱스에 대한 의존행렬 및 배열행렬은 식(6)과 식(7)에서 이미 구한대로 다음과 같다.

$$D=\begin{bmatrix}00 & 11 & 00\\ 00 & 00 & 11\\ 11 & 00 & 00\end{bmatrix}\begin{matrix}i\\ j\\ k\end{matrix}\quad F=\begin{bmatrix}10 & 00 & 10\\ 01 & 10 & 00\\ 00 & 01 & 01\end{bmatrix}\begin{matrix}i\\ j\\ k\end{matrix}$$

4.1 시간사상(M_t)

사상 M의 행렬 요소 중에서[t1, t2, t3]를 시간사상 M_t의 요소라고 하자. 식(11)에 의해 $M_t \cdot D_A > 0$, $M_t \cdot D_B > 0$, $M_t \cdot D_C > 0$이 되어야 하는데 이것은 각 데이타 항목들이 최소한 하나 이상의 시간스탭을 취해야함을 의미한다. 위의 조건을 만족하며 가장 빠른 출력 전달을 달성하는 정수의 解는 $T_1=T_2=T_3=1$이 되며 따라서 $M_t=[1, 1, 1]$로 구하여진다.

4.2 공간사상(M_s)

사상 M은 어레이 평면상에서의 임의의 데이타 항목에 대한 진행 방향과 배열 형태가 주어져야 계산될 수 있다. 어떠한 데이타 항목을 설정하더라도 동일한 결과가 나오므로 여기서는 계산의 결과로서 산출되는 출력 변수 C를 선정하기로 한다. 행렬곱의 출력 변수인 C가 취할 수 있는 여러가지 다양한 진행 방향 및 배열의 형태를 조합적으로 사용하면 계산이 가능한 사상 함수는 모두 448가지가 됨을 알 수 있다.

출력 데이타 항목 C의 진행 방향을 아래에서 위로 향하게 하고 그에 따른 계산 가능한 사상을 구한다. 즉, $D'c=[0\ -1]^t$ 일 때 식(16)에 의거하여 C행렬의 다양한 배열인 S'c의 각각의 경우에 대하여 M_s가 구하이지는데 모두 64개의 사상 함수가 계산된다. 이 64개의 M_s값들 중에서 시스토릭 어레이의 구현 조건에 맞지 아니하는 사상 함수들 즉, Det(M)=0인 함수들을 제외하면 50개가 남게 되며 이들 중에서 다시 대칭 또는 회전의 관계에 있는 동일한 모양의 것들을 제외하면 최종의 眞性사상은 Table 2에 나열한 것과 같이 7개가 된다.

Table 2. The essential mappings for matrix multiplication.

	S'c	M	
1	$\begin{bmatrix}-1 & 0\\ 1 & 1\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}-1 & 0 & 0\\ 0 & -1 & 0\\ 1 & 1 & 1\end{bmatrix}$	
2	$\begin{bmatrix}0 & -1\\ 1 & 1\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & -1\\ 0 & -1 & 0\\ 1 & 1 & 1\end{bmatrix}$	
3	$\begin{bmatrix}-1 & -1\\ 0 & 1\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}-1 & 0 & -1\\ -1 & -1 & 0\\ 1 & 1 & 1\end{bmatrix}$	
4	$\begin{bmatrix}-1 & -1\\ 1 & 0\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}-1 & 0 & -1\\ 0 & -1 & -1\\ 1 & 1 & 1\end{bmatrix}$	
5	$\begin{bmatrix}1 & -1\\ 0 & 1\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & -1\\ -1 & -1 & 0\\ 1 & 1 & 1\end{bmatrix}$	
6	$\begin{bmatrix}-1 & 1\\ 1 & 0\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}-1 & 0 & 1\\ 0 & -1 & -1\\ 1 & 1 & 1\end{bmatrix}$	
7	$\begin{bmatrix}-1 & 1\\ 1 & 1\\ 0 & 0\end{bmatrix}$	$\begin{bmatrix}-1 & 0 & 1\\ 0 & -1 & 0\\ 1 & 1 & 1\end{bmatrix}$	

4.3 어레이의 설계

Table 2에 열거한 여러가지의 S'c중에서 하나를 예로 들어 사상을 구하고 그에 따른 시스토릭 어레이를 설계한다.

배열 5를 고려하면

$$D'c=\begin{bmatrix}0 & 0\\ -1 & -1\\ 1 & 1\end{bmatrix}\quad S'c=\begin{bmatrix}1 & -1\\ 0 & 1\\ 0 & 0\end{bmatrix}$$

이다. 식(12)를 만족하는 a, b의 최소값은 1이며 이를 행렬로 표시하면

$$\alpha=\beta=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

이다. 식(16)에 의하여 M은 다음과 같다.

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

$M_t=[1\ \ 1\ \ 1]$이므로

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

식(9)에 의해 $D'=M\cdot D$이므로 각 데이타 항목들의 진행 방향을 나타내는 행렬 D'는 식(19)와 같으며 이것에 의하여 각 통신 패스의 방향을 그려보면 그림 5와 같다.

$$D'=M\cdot D=\begin{bmatrix}1 & 0 & -1\\ -1 & -1 & 0\\ 1 & 1 & 1\end{bmatrix}\cdot\begin{bmatrix}0&0&1&1&0&0\\0&0&0&0&1&1\\1&1&0&0&0&0\end{bmatrix}$$

$$=\begin{bmatrix}-1&-1&1&1&0&0\\0&0&-1&-1&-1&-1\\1&1&1&1&1&1\end{bmatrix}\qquad(19)$$

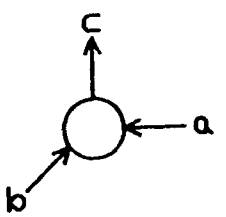


Fig. 5. Propagations of data items

또한, 식(10)에 의해 각 데이타 항목의 공간 영역상의 배열관계인 행렬 S'를 계산하면 식(20)과 같다.

$$S'=\begin{bmatrix}2&1&-1&-2&1&-1\\-1&-1&0&1&0&1\\0&0&0&0&0&0\end{bmatrix}\qquad(20)$$

시스토릭 구조의 설계를 위해 각 데이타 항목이 A=2×3, B=3×4, C=2×4의 크기를 가진 행렬이라 하면, 어레이 평면상에서의 데이타 항목들의 진행 방향과 배열 관계를 표시한 시스토릭 어레이는 Fig. 6과 같이 표시된다.

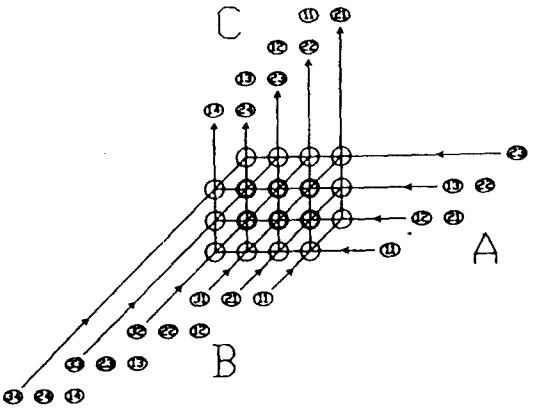


Fig. 6 Implementation of systolic array for matrix multiplication.

Table 2에 나열된 眞性사상들 중에서 어느 것이 최적의 어레이를 산출할 것인가 하는 문제는 최적 평가의 기준에 따라 다르다. 그 기준을 칩의 면적에만 국한 시킨다면 1의 경우가 6개의 처리 요소만을 필요로 하는 최적의 어레이가 되며 처리 시간만의 경우에는 3, 4, 5, 6의 경우가 해당된다. 실제적으로 최적 어레이의 설계는 특유의 해당하는 알고리즘의 적용 내용에 따라 모든 사항을 고려하여 선정하게 된다.

5. 결 론

본 논문에서는 순환 알고리즘을 시스토릭 어레이 구조로 구현시킴에 있어서 입출력 데이타 항목의 진행 방향과 배열의 순서를 이용하여 사상을 구하였다. 그리고 그 결과로서 나오는 많은 수의 계산 가능한 사상들 중에서 眞性사상만을 가려내어 효과적으로 어레이를 설계할 수 있는 방법을 제안하였고 실례를 통하여 설계 및 평가하였다.

데이타의 진행 방향과 입력 배치 순서의 조합이 매우 다양하므로 이에 따라 대량의 사상 함수를 계산해야 하는데 이들 중에는 시스토릭 어레이의 특성에 적합하지 않은 것도 있으며 상호 대칭 또는 회전의 관계로 인하여 결국에는 동일한 설계가 나오는 불필요한 것들도 많이 포함되어

있다. 본 연구에서는 어레이 프로세서의 설계 과정 중 VLSI의 레이아웃 레벨에서 고려될 수 있는 몇가지의 사항들을 데이타의 진행 방향 및 배치 순서의 선정에 미리 사용함으로써 어레이 평면상의 처리요소의 통신 패스의 수를 줄였다.

그 결과 자동 설계 프로그램 등에 의하여 일괄 계산이 가능한 사상 함수의 전체적인 수를 감축하였다. 그리고 眞性사상을 선택함으로써 기하학적으로 동일한 결과를 야기하는 사상들을 제거하여 시스토릭 어레이를 설계하는 시간과 노력을 단축시킬 수 있었다.

參 考 文 獻

1. H. T. Kung and C. E. Leiserson, "Systolic Arrays(for VLSI)," Sparse Matrix Proc., pp. 256−282, 1978.
2. S. Y. Kung, "VLSI array processors," Prentice Hall, pp. 195−285, 1988.
3. D. I. Moldovan, "Modern parallel processing," pp. 25−30, 1986.
4. D. I. Moldovan, "ADVIS : Software Package for the Design of Systolic Arrays," IEEE Trans. on Computers, vol. CAD−6, pp. 33−40, 1987.
5. A. Faroughi and M. A. Shanblatt, "Systematic generation and enumeration of Systolic Arrays from Algorithms," Proc. of International conference on Parallel Processing, pp. 844−847, Aug. 1987.
6. H. T. Kung, "Why Systolic Architectures ?," Computer magazine vol. 15, No. 2, pp. 37−46, 1982.
7. GUO−JIE LI and BENJAMIN W. WAH, "The Design of Optimal Systolic Arrays," IEEE Trans. on Computers, vol. c−34, No. 1, pp. 66−77, Jan. 1985.
8. Chung K. Ko and Omar Wing, "Mapping Strategy for Automatic Design of Systolic Arrays," IEEE International Conference on Systolic Arrays, pp. 285−194, 1988.
9. R. W. Hartenstein and K. Lemmert, "A CHDL−Based CAD System for the Synthesis of Systolic Architectures," IEEE International Conference on Systolic Arrays, 1988.
10. 우종호, 안광선, "Polyadic−nonserial 동적 프로그래밍 처리를 위한 시스토릭 어레이의 설계 및 효율적인 운영," 대한전자공학회논문집, vol. 26, No. 8, pp. 1178−1186, Aug. 1989.