Bull. Korean Fish. Tech. Soc. 26(3), 1990 漁業技術 26(3), 1990

최적 시스토릭 어레이의 자동설계

성기택 · 신동석 · 이덕수*
부산수산대학교 · *한국해양대학

The Automatic Design of Optimal Systolic Arrays

Ki-Taek SEONG, Dong-Suk SHIN and Deok-Su LEE*
National Fisheries University of Pusan and *Korea Maritime University

In this paper, a methodology for the automatic design of the optimal systolic arrays is proposed. Algorithm transformation is the main mathematical tool on which this methodology is based. Also, technique for partitioning algorithm into systolic arrays is presented. Algorithm partitioning is essential when the size of the computational problem is larger than the size of the array.

This study results in (a) reduction of the design time of systolic arrays for given algorithms, (b) CRT display of the structures of systolic arrays, and (c) automatic designing of the optimal systolic array by the criteria such as the number of processing elements, bands, and communication paths. The procedure for these results was programmed using HP BASIC language on HP-9836 computer.

서 론

최근 VLSI 반도체 기술의 발달로 인하여 신호처리, 형태인식, 행렬계산, 동적 프로그래밍 및 데이타 베이스 연산등을 이행하는 특수목적의 프로세서 설계와 개발에 대한 연구가 진행되고 있다[1),2)]. 시스토릭 어레이는 이러한 연구의 중심과제의 하나로 H.T. Kung과 C.E. Leiserson[3),4),5)]에 의해 처음 제안된 이래 각종 알고리즘에 대해 개발되어왔다. 특히 알고리즘과 관련된 처리식이 순환적(recursive)이고 입출력에 비해서 계산량이 절대적으로 많은 알고리즘에서 시스토릭 어레이 기법이 효율적으로 사용된다. 이 기법은 파이프라인과 다중처리기법을 이용하여 'Von Neumann의 병목현상' 문제를 해결한다[6)]. 따라서 적절한 데이타 전송율로서 높은 계산력을 얻을 수 있다. 이러한 시스토릭 어레이는 모듈성(modularity), 규칙성, 지역적 상호연결, 파이프라이닝(pipelining), 동기화된 다중처리등의 특징을 가진다.

알고리즘을 시스토릭 어레이로 변형하는 체계적인 방법과 이것을 이용한 자동 설계에 대한 연구는 Moldovan[7)]등에 의해서 행하여져 왔다. Moldovan은 알고리즘과 시스토릭 어레이를 각각 모델화 하여 알고리즘 모델을 에레이 모델로 사상(mapping)하는 방법을 제안하였다. 이 방법에서는 가능한 이용행렬의 수는 데이타 의존행렬의 열의 수에 급수적으로 비례하므로 의존행렬의 크기가 큰 경우에는 설계시간이 많이 소요된다.

본 연구에서는 이용행렬에서 열의 수를 줄이고, 의존벡터가 같은 열벡터들을 단일화시켜 의존행렬의 크기를 줄이고, 시스토릭 어레이의 설계과정중 어레이를 VLSI로 구현하는 레이아웃 레벨(layout level)에서 적용될 수 있는 조건들을 이용하여 효과적으로 공간사상이 구해질 수 있도록 했다. 또한 알고리즘을 분할하여 고정된 크기의 시스토릭 어레이로 사상하였다.

구현된 최적의 시스토릭 어레이는 시스토릭 네트워크의 처리요소 수와 주어진 알고리즘을 처리하는 시간에 대해서 평가되었고 이에 대한 자동설계 소프트웨어 패케지를 개발하였다.

시스토릭 어레이의 설계

초기의 시스토릭 어레이는 설계자의 경험에 따라 알고리즘의 계산시간, 타이밍, 데이타의 전달특성 등을 관찰하여 설계되었다. 따라서 한 알고리즘에 대해서 다수의 시스토릭어레이들이 제안되었다. 일반적으로 시스토릭 어레이는 알고리즘의 특성에 따라 구현되므로 융통성이 없었다. 최근에는 알고리즘을 시스토릭 어레이에 사상하고, 합성하는 효과적인 방법[8]들이 연구되고 있으며 주어진 알고리즘에 대한 최적 시스토릭 어레이를 체계적인 방법으로 설계하기 위해서는 알고리즘과 시스토리 어레이의 모델을 설정할 필요가 있다. 여기서는 알고리즘과 어레이를 각각 프로그램의 입력이 가능한 형태로 모델화하여 알고리즘 모델을 어레이 모델로 사상하여 시스토릭 어레이를 설계한다.

1. 알고리즘 및 VLSI 어레이 모델

알고리즘에서 변수의 수는 시스토릭 어레이의 입출력 채널의 수를 결정하며 데이타의 의존관계는 어레이에서 처리요소 사이의 데이타 전송을 의미한다. 알고리즘의 계산은 처러요소가 행하는 동작을 결정하며 인덱스는 어레이의 형태 및 처리요소의 수를 결정하는 중요한 정보가 된다. 따라서 알고리즘 모델은 다음과 같이 정의될 수 있다.

[정의1] 알고리즘은 다음의 3가지 항목으로 구성된다.

$$A=(J^n, C, D)$$

여기서 J^n은 $J^n \subset Z^n$인 알고리즘 인덱스 집합이며, C는 계산집합, D는 변수의 의존관계를 나타낸 의존행렬이다.

단 Z^n은 정수 Z의 n차 곱집합(Cartesian product).

시스토릭 어레이는 트리구조, 사각구조, 삼각구조등 여러가지 형태를 가질 수 있다. 메쉬(mesh)구조 어레이에서 적절한 통신방향을 취함으로서 다양한 어레이를 구성할 수 있다. 따라서 처리요소의 위치를 나타내는 인덱스와 통신방향을 표시하는 두 항목으로서 어레이는 모델화될 수 있다. 어레이구조는 다음과 같이 정의된다.

[정의2] 메쉬구조의 어레이는 두 항목으로 구성된다.

$$B=(I^{n-1}, P)$$

여기서 I^{n-1}은 어레이의 처리요소 위치를 나타내는 인덱스 공간이며 I^{n-1}과 Z^{n-1}은 $I^{n-1} \subset Z^{n-1}$의 관계로 나타낼 수 있고 P는 통신패스의 방향을 나타내는 행렬로서 $P \in Z^{(n-1)\times m}$의 조건을 만족한다. 단 m은 알고리즘의 의존행렬의 벡터수 이다.

본 연구에서는 알고리즘의 인덱스 공간은 J^3으로 고정한다(대부분의 설계된 어레이는 이 범주에 속함)[9]. n=3일때 어레이의 통신패스행렬 P는 식(1)로 표현될 수 있다.

$$P=\begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 0 & -1 & -1 & -1 \\ 0 & 1 & 1 & 0 & -1 & -1 & -1 & 0 & 1 \end{pmatrix} \quad (1)$$

어레이의 인덱스 공간을 다음식과 같이 두면 어레이의 형태는 Fig. 1과 같다.

$$I=\{(i_1, i_2) : 0 \leq i_1 \leq 2, 0 \leq i_2 \leq 2\} \quad (2)$$

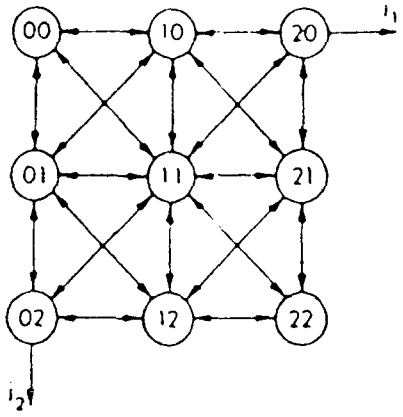

Fig.1. Array model(n=3).

2. 알고리즘에서 어레이로의 시간 및 공간사상

알고리즘의 인덱스된 변수가 어레이에서 계산되는 시각과 처리요소의 위치는 식(3)의 선형변환(Linear Transformation)에 의해서 결정된다[10].

$$T=\begin{bmatrix} \Pi \\ S \end{bmatrix} \tag{3}$$

여기서 Π와 S는 각각 시간변환, 공간변환을 의미하고 변환관계는 $\Pi : I^n \rightarrow Z$이고, $S : I^n \rightarrow Z^{n-1}$이다. 정확한 계산순서를 위하여 알고리즘의 모든 의존벡터(di)에 대하여 다음의 조건을 만족해야한다.

$$\Pi(d_i) > 0 \tag{4}$$

여기서 $\Pi(d_i)$는 데이타 의존벡터를 시간변환한 것이다.

공간변환 S는 $i_1, i_2 \in I^n$에 대하여

$$S(i_1)=S(i_2) \rightarrow \Pi(i_1) \neq \Pi(i_2) \tag{5}$$

가 되어야 한다. 즉 동시에 수행되는 계산은 같은 처리요소로 사상될 수 없다. 따라서 S를 구하는 것은 전치행렬[Π, S]^{T}가 비특이(non-singular)인 조건에서 알고리즘의 의존행렬을 어레이로 사상시키는 부정방정식(diophantine equation)의 해를 구하는 것과 같다.

$$S \cdot D = P \cdot K \tag{6}$$

여기서 행렬 K와 Z는 $K \in Z^{pxm}$로서 K의 각 요소는 다음의 조건을 만족해야 한다.

$K_{i,j}=0$ 혹은

$$1,\ 0 \leq \sum_i K_{i,j} \leq \Pi(d_j)\ \ 1 \leq i \leq p,\ \ 1 \leq j \leq m \tag{7}$$

여기서 p는 P행렬에서 열의 갯수이고 m은 알고리즘에서 주어지는 데이타 의존벡터의 수이다. 만일 식(7)의 조건을 만족하는 식 (6)의 해가 존재하지 않으면 Π를 다시 구하여 S를 구하는 과정을 반복한다. Moldovan의 방법에서는 어레이 설계의 대부분의 시간이 이용행렬을 발생시키는 데에 소요되어 의존벡터의 크기가 클 경우에는 많은 시간이 소요 되며 같은 형태를 나타내는 S가 중복되어 계산되므로 효율적이지 못하다. 우[11]는 시스토릭 어레이의 설계영역에서 적용할 수 있는 여러가지 사항들을 고려하여 효과적으로 S를 구하는 방법을 제안 했다. 이 방법을 적용하여 나타나는 처리요소의 형태는 Fig. 2의 (a)의 형태가 (b)의 형태와 같이되며 발생되는 이용행렬의 수는 식(8)과 같다.

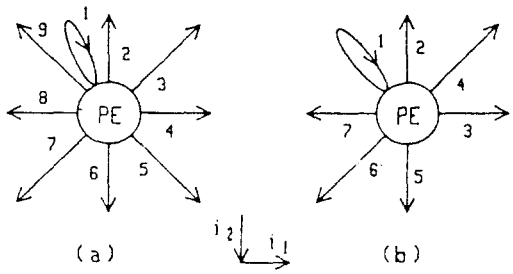

Fig.2. The communication paths of systolic array. (a) All possible paths. (b) The selected paths.

$$\begin{aligned} &\prod_{1 \leq j \leq m,\ 1 \leq i \leq \Pi d_j} \left(\sum {}_3C_{i-1} \right) \\ &\leq M \leq \prod_{1 \leq j \leq m,\ 1 \leq i \leq \Pi d_j} \left(\sum {}_7C_{i-1} \right) \end{aligned} \tag{8}$$

단 m'는 의존행렬을 단일화 시켰을 때 만들어진 새로운 의존행렬의 수이다.

알고리즘 분할에 대하여 Moldovan은 고정된 크기의 어레이에 알고리즘의 인덱스를 사상시키는 방법을 제안했다[12]. 즉 공간사상 S의 각 행벡터가 인덱스 공간을 sweep하는 하이퍼평면(hyperplane)을 형성하므로 하이퍼평면에 존재하는 인덱스는 동시에 처리될 수 있다. 따라서 S의 행벡터에 의해 형성되는 하이퍼평면은 하나의 밴드(band)를 형성한다. 각 밴드를 구별하기 위해서 설정되는 두좌표는 다음과 같다.

$$b_1 = \left\lfloor \frac{\Pi_{p_1} J}{N} \right\rfloor \tag{9}$$

$$b_2 = \left\lfloor \frac{\Pi_{p_2} J}{N} \right\rfloor \tag{10}$$

여기서 ⌊X⌋는 X보다 작거나 같은 최대 정수이고 Π_{p_i}(i=1, 2)는 S의 열벡터이고 N은 NxN형의 고정된 어레의 크기를 나타낸다. 따라서 b_1, b_2에 의해 알고리즘이 분할되었을 때 밴드의 수가 적을수록 처리시간이 짧게되므로 최소의 밴드를 갖게하는 S를 구하면 최적의 어레이가 설계될 수 있다. 알고리즘의 계산은 식 (11), (12)에 의해 결정되는 밴드에 분할되어 처리된다.

$$f_1 = \Pi_{p1} \cdot J \cdot mod\ N \tag{11}$$

$$f_2 = \Pi_{p2} \cdot J \cdot mod\ N \tag{12}$$

여기서 $\hat{J}_1$, $\hat{J}_2$는 밴드내의 처리요소의 위치를 나타내는 인덱스이다.

3. 최적 어레이 설계조건

시스토릭 어레이를 설계할 때 최소의 칩면적과 최단의 처리시간을 갖는 것이 최적의 어레이라 할 수 있다. 칩면적의 대부분을 차지하는 처리요소의 수가 적을 수록 최소면적을 차지하게 된다. 알고리즘의 인덱스 집합이 어레이의 처리요소로 사상되므로 이때 전체 인덱스집합의 요소의 수에서 어레이로 사상될때 중복되는 수를 빼면 설계되는 전체 어레이의 처리요소 수가된다. 즉,

$$P_N = I_N - A_N \tag{13}$$

여기서 P_N은 설계되는 어레이의 처리요소의 수이고 I_N은 알고리즘에 존재하는 인덱스집합의 요소의 수이며 A_N은 중복되어 사상되는 인덱스의 수이다. 따라서 I_N은 알고리즘에 의해 고정되어 있으므로 A_N이 클수록 P_N의 수는 적게된다. 알고리즘 처리시간 t는 Π에 의해 변환된 인덱스에서 제일 먼저 처리될때의 시간차 이므로 최소의 처리시간을 요구하는 Π를 구하면 된다. 이를 식으로 나타내면 다음과 같다.

$$t = \left\lceil \frac{max(\Pi \cdot (J_1 - J_2)) + 1}{min(\Pi \cdot d_i)} \right\rceil \tag{14}$$

단 J_1, $J_2 \in J^n$이다. 여기서 $\lceil X \rceil$는 X보다 크거나 같은 최소 정수를 나타내며 Π를 구하는 알고리즘은 Fig. 3과 같다.

따라서 식(10)의 조건을 만족하는 Π를 구하여 최소의 P_N이 됨과 동시에 식(11), (12)에 의해 분할되는 밴드의 수가 최소인 S를 구하면 최적의 시스토릭 어레이를 설계할 수 있다.

4. 시스토릭 어레이의 자동설계

알고리즘으로 부터 시스토릭 어레이를 자동설계하는 소프트웨어 패케지를 앞의 이론을 이용하여 HP 9836 컴퓨터로써 구현하였다. 사용자와의 대화식 연결로서 설계요소의 변화에 따른 시스토릭 어레이의 상태가 CRT화면에 나타나도록하여 효과적인 설계작업이 수행되도록 하였다. 프로그램의 처리순서는 다음과 같다.

단계 1. 알고리즘의 인덱스공간의 의존행렬을 입력한다.

```
L1  :   Procedure Find-Π(D,Π,Ind)// D is the dependence matrix //
                                  //  and  Id is the index set. //
L2  :     Initilization ;
L3  :     M←3 // set the Π's element boundary  //
L4  :     for i=-M  to  M  do
L5  :       for  j=-M  to  M do
L6  :         for  k=-M to  M  do
L7  :           compose Π // Π=(i,j,k) //
L8  :           if Π · d >0 then  // d is the column vector of D //
L9  :             begin
L10 :               for  all  Id
L11 :                  begin
L12 :                    t ←(max (Π · (J1 - J2)) + 1)/(min(Π · d))
                          // J1,J2 are the elements of Id //
L13 :                  end
L14 :                all for
L15 :              end
L16 :            end if
L17 :            if t is the minimum then optimal←Π
L18 :          repeat
L19 :        repeat
L20 :      repeat
L21 :      write optimal
L22 :   end Find-Π
```

Fig.3. The algorithm for finding the time transformation(Π).

단계 2. 조건을 만족하는 최적의 시간사상을 구한다.

단계 3. 의존행렬에서 시간 인덱스를 제외한 나머지의 열벡터 중에서 같은 것이 있으면 단일화 하여 새로운 의존행렬을 구성한다.

단계 4. 특정 형태의 설계를 요구할 경우 의존행렬의 열벡터에 해당하는 변수에 대한 통신패스를 요구하고 어레이의 설계 가능성을 판별하고 최적 어레이의

```
L1 :    Procedure Find-S(Π,D)
L2 :      Initilization
L3 :      Kn←3 // Kn is the no.of comm. path //
L4 :      call Reduce-D(D,m,nm)  // Reduce the size of dependence //
                // matrix D if possible,nm is renewed value of m //
L5 :      for all K on condition (5),(6) do
                  //   K is the utilization matrix //
L6 :        generate K
L7 :        S ← P · K · D^T · (D · D^T)^-1
L8 :        if S is singular then return
L9 :          compose T  // T = [Π , S]^T //
L10:          compute the no. of PE's  and index's position
L11:          compute the index of partition band
L12:          display the result array's configuration on CRT
L13:          if other type of array is required then
L14:            if all K have generated   stop
L15:          else
L16:              Kn←Kn+1
L17:              return(kn)
L18:          end if
L19:        else
L20:        end if
L21:      all for
L22:    end Find-S

L22:    Procedue Reduce-D(D,m,nm)
L23:         initilization // i is the index of the vector of D //
L24:         for 1 ≤ i ≤ m do
L25:           compare all vector d_i;
L26:             if there exist the same vectors d_i in D then
                        // consider only row 2 and 3 of D //
L27:               find←1
L28:               n←n+1
L29:               d←max(Π · d_i)
L30:              else
L31:              return(i)
L32:             end if
L33:         all for
L34:              if find=1
L35:                then construct new D
L36:                nm←m-n
L37:                else
L38:                nm←m
L39:              end if
L40:    end Reduce-D
```

Fig.4. The algorithm for finding the transformation matrix(T).

설계조건을 지정한다(요구되는 처리요소의 수, 통신패스 분포도, 최소의 통신패스의 종류 등).

단계 5. 이용행렬 K를 발생시킨다.

단계 6. 공간 사상을 구하고 특이이면 단계 5로 간다.

단계 7. 이미 구해진 시간사상과 공간사상으로서 선형변환행렬 T를 구성하여 어레이를 설계하고 처리요소의 수, 분할밴드의 수 등을 계산하여 최적의 어레이를 판별하여 그 결과를 화면에 나타낸다.

단계 8. 다른 형태를 원할 경우 단계 5로 가서 다른 형태의 공간사상을 구하고 아니면 프로그램을 종료한다.

T를 구하는 알고리즘은 Fig. 4와 같다.

결과 및 고찰

작성된 소프트웨어를 동적 프로그래밍문제와 행렬곱문제, QR분해문제에 각각 적용하여, 각 알고리즘에 대한 선형변환 T 및 처리요소의 수, 처리시간, 밴드의 수등을 비교하여 최적의 시스토릭 어레이를 설계하였다. 또한 각 알고리즘 문제에 대하여 3×3 크기의 고정된 2차원 시스토릭 어레이로 분할하였다. 일반적으로 최적 시스토릭 어레이의 판별은 AT 혹은 AT^2(A : Area 즉 처리요소의 수, T : 처리시간)의 평가기준을 근거로하여 선택하므로 각 표에서 최적의 시스토릭 어레이는 처리시간과 처리요소의 수가 최소가 됨과 동시에 분할했을 경우 밴드의 수가 최소가 되는 것이다. 그 결과 아래의 각표에서 TYPE 1이 최적임을 알수있다.

Table. 1은 동적 프로그래밍문제(n=5)에 대한 3가지 형태의 시스토릭 어레이를 비교하였다. Table. 1에 대한 최적의 시스토릭 어레이는 Fig. 5와 같다.

Table. 2는 QR분해 알고리즘문제(n=9)에 대한 2가지 시스토릭 어레이를 비교하였다. Table. 2에대한 최적의 시스토릭 어레이는 Fig. 6과 같다.

Table. 3의 행렬곱문제는 5행 9열 행렬과 9행 5열 행렬의 곱을 대상으로 한 경우의 시스토릭

Table.1. Systolic Solutions for Dynamic programming.

	변환행렬 T	처리시간	처리요소의 수	밴드의 수
TYPE 1	$\begin{pmatrix} 1 & -2 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	10(Unit Time)	15	5
TYPE 2	$\begin{pmatrix} 1 & -2 & 1 \\ 1 & -1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$	10(Unit Time)	15	7
TYPE 3	$\begin{pmatrix} 1 & -2 & 1 \\ 1 & -1 & -1 \\ 0 & -1 & 0 \end{pmatrix}$	10(Unit Time)	25	6

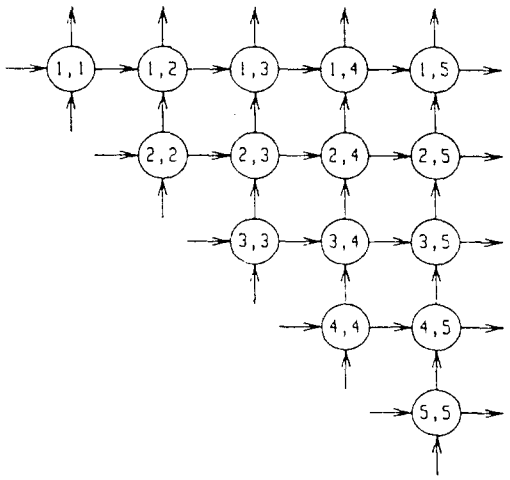

Fig.5. Systolic array for dynamic programming.

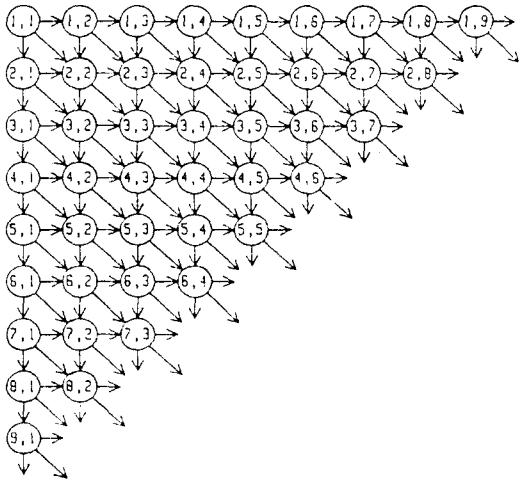

Fig.6. Systolic array for QR decomposition.

Table.2. Systolic Solutions for QR decomposition.

	변환행렬 T	처리시간	처리 요소의 수	밴드의 수
TYPE 1	$\begin{pmatrix} 2 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & -1 & 0 \end{pmatrix}$	15(Unit Time)	36	6
TYPE 2	$\begin{pmatrix} 2 & -1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	15(Unit Time)	36	9

Table.3. Systolic Solutions for Matrix multiplication.

	변환행렬 T	처리시간	처리 요소의 수	밴드의 수
TYPE 1	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$	17(Unit Time)	25	4
TYPE 2	$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$	17(Unit Time)	45	8
TYPE 3	$\begin{pmatrix} 1 & 1 & 1 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	17(Unit Time)	81	16

어레이를 비교하였다. Table.3에 대한 최적의 시스토릭 어레이는 Fig.7과 같다.

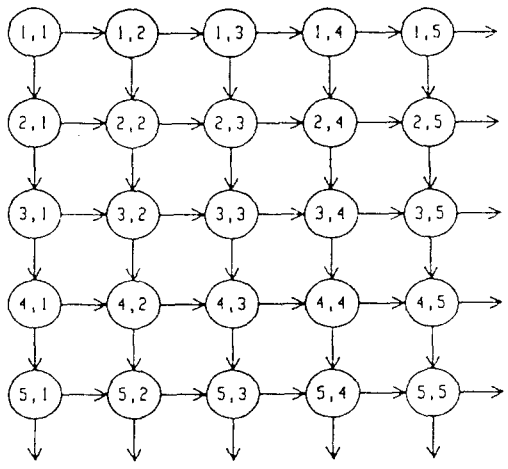

Fig.7. Systolic array for Matrix multiplication.

Moldovan의 방법과 본 연구에서 적용한 방법과의 시행횟수를 비교하면 Table.4와 같다.

Table.4에서 m은 알고리즘에서 의존행렬의 열벡터의 수이며 m'는 단일화 된 열벡터의 수이고 d_j는 의존행렬의 열벡터이다.

Table.4. The comparision of Modovan's and proposed method.

적용 알고리즘	Moldovan의 방법	제안된 방법
Dynamic programming algorithm	$\prod_{1\le j\le m,\,1\le i\le \Pi d_j} (\sum {}_8C_{i-1})$	$\prod_{1\le j\le m,\,1\le i\le \Pi d_j} (\sum {}_3C_{i-1})$
QR decomposition algorithm	〃	$\prod_{1\le j\le m,\,1\le i\le \Pi d_j} (\sum {}_4C_{i-1})$
Matrix multiplication algorithm	〃	$\prod_{1\le j\le m,\,1\le i\le \Pi d_j} (\sum {}_3C_{i-1})$

결 론

본 연구에서는 시스토릭 어레이의 처리요소 수와 주어진 알고리즘을 처리하는 시간 두 평가기준에 대해서 최적의 시스토릭 어레이를 구현하기 위한 자동설계 소프트웨어 패케지를 개발하였다. 알고리즘의 크기에 맞는 시스토릭 어레이는 많은 처리요소를 요구하기 때문에 비효율적이므로 알고리즘을 분할하여 고정된 크기의 시스토릭 어레이로 사상시키는 방법을 이용했다. 시스토릭 어레이 설계과정에서 고려될 수 있는 여러가지 사항들을 고려하여 처리요소의 통신패스 방향의 수를 줄이고 의존행렬의 열벡터에서 값이 같은 열벡터는 단일화하여 의존행렬의 크기를 줄여 발생되는 이용행렬의 수를 크게 줄였다. 따라서 기존의 Moldovan에 의한 방법보다 시스토릭 어레이를 설계하는 시간을 단축시켰으며, 처리요소의 수, 알고리즘의 수행시간, 분할밴드의 수 등을 계산하여 최적의 시스토릭 어레이를 설계했다. 작성된 프로그램에 동적 프로그래밍 알고리즘, QR분해 알고리즘과 행렬곱 알고리즘을 적용하여 각각에 대한 최적의 시스토릭 어레이를 설계하였으며 설계된 어레이의 구성을 CRT에 나타내어 어레이의 형태를 쉽게 인식할 수 있게 했다.

본 연구의 결과는 빠른 응답을 요구하는 신호처리 및 데이타베이스 등에서 특수회로를 설계할 때 응용 될 수 있다. 그러나 본 연구에서의 시스토릭 어레이는 처리요소들이 분산되어 지역적으로 상호연결되어 있으므로 한 처리요소가 제대로 동작하지 않으면 전체결과가 잘못된다. 따라서 몇개의 처리요소가 동작되지 않을 경우에도 전체

시스템이 정확하게 동작할 수 있는 폴트톨러런스 시스템의 설계가 앞으로의 고려사항이다.

참고문헌

1) Synder, L., L.H. Jamieson, D.B. Gannon and H.J. Siegel(1985) : Algorithmically specialized parallel computers. Academic Press.
2) Kung, S.Y.(1988) : VLSI array processors. Prentice Hall Press. 110-294.
3) Kung, H.T. and C.E. Leiserson(1979) : Systolic Array(for VLSI). Sparse Matrix Symposium, 256-282.
4) Leiserson, C.E.(1982) : Area-efficient VLSI computation. The MIT Press.
5) Guibas, L.J., H.T. Kung and C.D. Thompson(1979) : Direct VLSI implementation of combinatorial algorithms. Proc. of CALTECH conference on VLSI, 509-525.
6) Kung, H.T.(1982) : Why Systolic architectures? IEEE Computer 15(1), 37-46.
7) Moldovan, D.I.(1987) : ADVIS ; a software package for the design of systolic arrays. IEEE trans. on computer-Aided Design 6(1), 33-40.
8) GUO-JIE Li and B.W. WAH(1985) : The Design of Optimal Systolic Arrays. IEEE Transactions on COMPUTER 34(1), 66-77.
9) 堀池 聰, 西田 正吾, 坂口 敏明(1987) : プロセツサ數の制限下におけるシストリツクアレーの 設計法. 電子情報通信學會論文誌 J70-D (5), 880-888.
10) Moldovan, D.I.(1983) : On the design of algorithms for VLSI systolic arrays. Proc. of IEEE 71(1), 113-120.
11) 우종호(1989) : Polyadic-nonserial 動的프로그래밍 處理를 爲한 시스토릭 構造. 慶北大學校 工學博士學位論文.
12) Moldovan, D.I. and A.B. Fortes(1986) : Partitioning and Mapping Algorithm into fixed size systolic arrays. IEEE Transactions on Computer 35(1), 1-12.