An Approach to Checking Link Conflicts in the Mapping of Uniform Dependence Algorithms into Lower Dimensional Processor Arrays

Jenn-Yang Ke and Jong-Chuang Tsay

Abstract—In this paper, we propose an enumeration method to check link conflicts in the mapping of n-dimensional uniform dependence algorithms with arbitrary convex index sets into k-dimensional processor arrays. Previous methods on checking the link conflicts had to examine either the whole index set or the I/O spaces whose size are $O(N^{2n})$ or $O(N^{n-1})$, respectively, where N is the problem size of the n-dimensional uniform dependence algorithm. In our approach, checking the link conflicts is done by enumerating integer solutions of a mixed integer linear program. In order to enumerate integer solutions efficiently, a representation of the integer solutions is devised so that the size of the space enumerated is $O((2N)^{n-k})$. Thus, our approach to checking link conflicts has better performance than previous methods, especially for larger k. For the special case $k = n-2$, we show that link conflicts can be checked by solving two linear programs in one variable.

Index Terms—Uniform dependence algorithms, lower dimensional arrays, space-time mapping, link conflict, mixed integer linear programming, Hermite normal form, Smith normal form.

1 Introduction

REGULAR processor arrays, such as *systolic arrays* introduced by Kung [1], which have regularly and locally connected interconnections via data links between processing elements (PEs) are very suitable for implementation on VLSI chips. This type of processor arrays supports the parallel implementation of algorithms, especially the uniform dependence algorithms, from signal or image processing and scientific computation applications [2].

Uniform dependence algorithms, termed by Shang and Fortes [3], include those described by single uniform recurrence equations [2], [4], [5], [6], [7], [8], [9], [10] and those described by programs with nested loops [11]. They are characterized by uniform data dependencies and unit-time computation. Informally, a uniform dependence algorithm is represented by a subset (called *index set*) of multidimensional integer points (called *index points*) and a finite set of data dependence vectors. The index set of an algorithm is a finite convex subset of $\mathcal{Z}^n$ [12]. The minimal convex polytope or convex hull R bounding the index set is usually a nondegenerated convex polytope in $\mathcal{R}^n$. An index set is a *hyperparallelepiped index set* if R is a hyperparallelepiped. We call n the dimension of the uniform dependence algorithm.

Most researches on synthesizing [13], [14], [15], [16], [17], [18] processor arrays from the uniform dependence algorithms focus on finding a space-time mapping (linear transformation) to the algorithm such that the transformed algorithm represents a regular processor array. The space-time mapping is, in general, represented as a transformation matrix. The first row and the rest of the transformation matrix are the time mapping vector, called *linear schedule* vector, and the space mapping matrix, called *allocation matrix*, respectively. In other words, the space-time mapping transforms an index point of the uniform dependence algorithm into a time step and a PE location in the processor array. Thus, a processor array is a k-dimensional array if the transformation matrix is $(k+1) \times n$ integer matrix. For mapping a uniform dependence algorithm into a k-dimensional processor array, three kinds of *conflict-free* mapping conditions must be satisfied. They are precedence, computation, and data link conflict-free conditions. We say that a *computational conflict* occurs if more than one computation of a uniform dependence algorithm are mapped to the same processor and the same time step, and that a *link conflict* occurs if more than one datum are mapped such that they travel along the same data link at the same time step. In this paper, we address the problem of checking link conflicts in the mapping of n-dimensional uniform dependence algorithms into k-dimensional processor arrays with $0 < k < n-1$.

There have been several attempts on the problem of checking the link conflicts [13], [14], [16], [17], [18], [19], [20]. One of these methods [13], [14] has to examine the whole index set of size $O(N^{2n})$, where N is the problem size. Other methods [16], [17], [18], [19], [20] use the I/O spaces concept to check link conflicts. Each I/O space is associated with a data dependence vector $\bar{d}$. An I/O space can be an input space or an output space and is defined as the set $\{\bar{i} - \bar{d}_\gamma \mid \bar{i} \in J \text{ and } \bar{i} - \bar{d}_\gamma \notin J\}$ and $\{\bar{i} \mid \bar{i} \in J \text{ and } \bar{i} + \bar{d}_\gamma \notin J\}$, respectively, where J is the index set. However, an *exact* link conflict checking cannot be obtained by their methods if the I/O space is a nonconvex one. In [17], [18], a procedure was proposed to map nonconvex I/O space into convex one. But, the projection of nonconvex I/O space may introduce superfluous points which are not needed in the computation of the algorithm, into the projected I/O space. An example of such projection of nonconvex I/O space is shown in Fig. 1. Thus, to check link conflict exactly, enumeration of the I/O space of size $O(N^{n-1})$ is required.

The rest of the paper is organized as follows: In the next section, we define the algorithm model and the array model used in this paper. Section 3 is devoted to the formulation of the link conflict checking problem as a mixed integer linear program. In Section 4, a representation of the integer solutions of the mixed integer linear program is given. Based on this representation, we can enumerate the integer solutions in an efficient way. For the special case $k = n-2$, we also show that the link conflicts can be checked by solving two linear programs in one variable. In Section 5, we estimate the size of the enumeration space and show that the time complexity for the representation is polynomial. Finally, a conclusion is given in Section 6.

2 Algorithm Model and Array Model

In this section, the models for the algorithms and processor arrays used in this paper are introduced. We briefly describe them as follows.

2.1 Algorithm Model

Algorithms under consideration are the uniform dependence algorithms. A uniform dependence algorithm is a single-assignment algorithm [2], which can be described as n-nested loops of the form:

```
for i_1 = l_1 to u_1 by 1 do
  for i_2 = l_2 to u_2 by 1 do
    ...
    for i_n = l_n to u_n by 1 do
      stat_1;
      stat_2;
      ...
      stat_m;
    endfor
```

- *J.-Y. Ke is with the Department of Applied Mathematics, Tatung Institute of Technology, Taipei, Taiwan 10401, Republic of China. E-mail: jyke@math.ttit.edu.tw.*
- *J.-C. Tsay is with the Department of Computer Science and Information Engineering, College of Electrical Engineering and Computer Science, National Chiao Tung University, Hsinchu, Taiwan 30050, Republic of China. E-mail: jctsay@csie.nctu.edu.tw.*

Manuscript received 9 June 1997.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 105236.

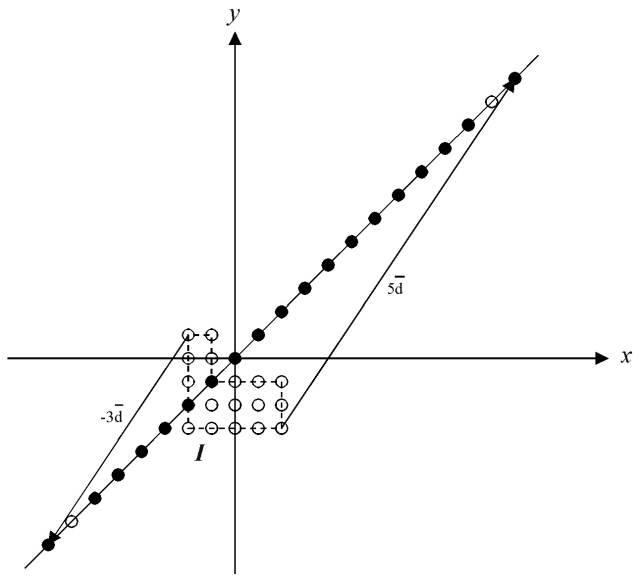

Fig. 1. Projection of a nonconvex I/O space I along the direction $\overline{d} = (2,3)^T$ introduces two superfluous points.

endfor
endfor

l_j and u_j are integer-valued linear expression involving $i_1, i_2, \cdots, i_{j-1}$, $1 < j \leq n$.

The loops define a space J, called the index set $J \subset \mathcal{Z}^n$, of points $(i_1, i_2, \cdots, i_n)$ as follows:

$$J = \{(i_1, i_2, \cdots, i_n) \mid l_j \leq i_j \leq u_j \text{ for } j = 1, \cdots, n; \text{ and } i_j \in \mathcal{Z}\}.$$

Statement $stat_j$ is of the form:

$$\mathcal{V}(\overline{i}) = f_j(\mathcal{W}(\overline{i} - \overline{d}_{\mathcal{W}}), \cdots, \mathcal{V}(\overline{i} - \overline{d}_{\mathcal{V}}), \cdots, \mathcal{X}(\overline{i} - \overline{d}_{\mathcal{X}})),$$

where index point $\overline{i} \in J$ and $\overline{d}_Z \in \mathcal{Z}^{n \times 1}$, $Z = \mathcal{V}, \mathcal{W}, \mathcal{X}$. f_j is a single-valued function of a fixed number of arguments. $\mathcal{V}, \mathcal{W}$, and $\mathcal{X}$ are *variables* of the uniform dependence algorithm. The constant vectors $\overline{d}_Z \in \mathcal{Z}^{n \times 1}$ are the data dependence vectors for the variables $Z = \mathcal{V}, \mathcal{W}, \mathcal{X}$, respectively. In the form of statement $stat_j$, computing the value $\mathcal{V}(\overline{i})$ of variable $\mathcal{V}$ at index point $\overline{i}$ needs the value $\mathcal{V}(\overline{i} - \overline{d}_{\mathcal{V}})$ of variable $\mathcal{V}$ at index point $\overline{i} - \overline{d}_{\mathcal{V}}$. Thus, we say that the value $\mathcal{V}(\overline{i} - \overline{d}_{\mathcal{V}})$ is used and the value $\mathcal{V}(\overline{i})$ is generated at index point $\overline{i}$ for variable $\mathcal{V}$. And, we say that data dependence vector $\overline{d}_{\mathcal{V}}$ is associated with variable $\mathcal{V}$. Now, we use the notation $token_{\mathcal{V}}(\overline{i})$ as a token of variable $\mathcal{V}$ whose value is used and generated in all index points of the form $\overline{i} + z\overline{d}_{\mathcal{V}}$, where $\overline{i} \in J$ and $z \in \mathcal{Z}$. From this, the input space and output space of variable $\mathcal{V}$ associated with data dependence vector $\overline{d}_{\mathcal{V}}$ are defined as the set $\{\overline{i} - \overline{d}_{\mathcal{V}} \mid \overline{i} \in J \text{ and } \overline{i} - \overline{d}_{\mathcal{V}} \notin J\}$ and $\{\overline{i} \mid \overline{i} \in J \text{ and } \overline{i} + \overline{d}_{\mathcal{V}} \notin J\}$, respectively.

Since the array model, described in the next paragraph, considered in this paper is the same as that in [13], [17], [18], we also impose the same restriction on the uniform dependence algorithms as in their works. The restriction is stated as follows.

Restriction [13, p. 66]. *For every data dependence vector*

$$\overline{d}_{\mathcal{V}}^T = (d_1, d_2, \cdots, d_n),$$

we have $gcd(d_1, d_2, \cdots, d_n) = 1$.

Under this restriction, there is no integer point between index point $\overline{i}$ and index point $\overline{i} + \overline{d}_{\mathcal{V}}$. Thus, $token_{\mathcal{V}}(\overline{i})$ and $token_{\mathcal{V}}(\overline{i}')$ represent two different tokens if and only if $\overline{i} \neq \overline{i}' + z\overline{d}_{\mathcal{V}}$ for $z \in \mathcal{Z}$.

2.2 Array Model

We assume that the execution of the processor array is synchronous with a global clock that ticks in unit time, and the evaluation of a computation by a PE takes unit time.

Since we use the space-time mapping $T = \begin{bmatrix} \Lambda \\ S \end{bmatrix} \in \mathcal{Z}^{(k+1) \times n}$ to map a uniform dependence algorithm with index set J to a processor array, the processor space is defined as the set $\{S\overline{i} \in \mathcal{Z}^{k \times n} \mid \overline{i} \in J\}$. Let data dependence vector $\overline{d}_{\mathcal{V}}$ be associated with a variable $\mathcal{V}$ in a uniform dependence algorithm. Under this space-time mapping, all tokens of the variable $\mathcal{V}$ must travel a distance $S\overline{d}_{\mathcal{V}}$ in $\Lambda\overline{d}_{\mathcal{V}}$ time steps. Thus, for each data dependence vector $\overline{d}_{\mathcal{V}}$ in the algorithm, there exists a directed path of links from PE with index p to PE with index $p + S\overline{d}_{\mathcal{V}}$. We assume that these links are connected as follows: Let g be the greatest common divisor of the entries of the vector $S\overline{d}_{\mathcal{V}}$. Then, two PEs with indices p_1 and p_2 are called *neighboring* PEs if there exists a directed link $(1/g)S\overline{d}_{\mathcal{V}}$ between the indices of those PEs, i.e., $p_1 - p_2 = (1/g)S\overline{d}_{\mathcal{V}}$. Assume that $g > 1$. Then, there are g directed links which connect neighboring PEs with index $p + (f/g)S\overline{d}_{\mathcal{V}}$, $f = 0, \cdots, g-1$ and PEs with index $p + ((f+1)/g)S\overline{d}_{\mathcal{V}}$, respectively. In addition, since the processor array is synchronous with a global clock, $\Lambda\overline{d}_{\mathcal{V}}$ must be equal to gz' for some positive integer z' so that a token can reach a PE at the time specified by the space-time mapping. Thus, there are $z' - 1$ shift registers in each directed link $(1/g)S\overline{d}_{\mathcal{V}}$. In other words, this array model does not allow data broadcasting. Let p be an index of a PE in the processor space, it is called an *input* (*output*) PE if $p - (1/g)S\overline{d}_{\mathcal{V}}$ ($p + (1/g)S\overline{d}_{\mathcal{V}}$) is not in the processor space. All I/O operations with the host are restricted to the input (output) PEs of the array. This array model is the same as that is adopted in [13], [14], [17], [18], [19], [20]. It conforms to the typical properties of VLSI processor arrays, namely, simple and regular interconnection pattern between PEs.

3 Formulation of the Link Conflict Checking Problem

In this section, we disscuss the formulation of the link conflict checking problem. A link conflict occurs if and only if two tokens of a variable arrive the same PE at the same time and move together contending the same link. Given a uniform dependence algorithm with data dependence matrix D and index set J, the formulation of the link conflict checking problem can be derived as follows:

Let $T = \begin{bmatrix} \Lambda \\ S \end{bmatrix}$ be the space-time mapping. Let variable $\mathcal{V}$ be associated with data dependence vector $\overline{d}_{\mathcal{V}}$. Since, for each data dependence vector $\overline{d}_{\mathcal{V}} = (d_1, \cdots, d_n)$, we restricted that $\gcd(d_1, d_2, \cdots, d_n) = 1$, there is no integer point between index point $\overline{i}$ and index point $\overline{i} + \overline{d}_{\mathcal{V}}$. Let g be the greatest common divisor of the entries of the vector $S\overline{d}_{\mathcal{V}}$. Obviously, the value of g may be greater than 1 for some allocation matrix S and data dependence vector $\overline{d}_{\mathcal{V}}$. Assume that $g > 1$. Then,

$(1/g)S\bar{d}_{\mathcal{V}}, (2/g)S\bar{d}_{\mathcal{V}}, \cdots, ((g-1)/g)S\bar{d}_{\mathcal{V}}$, and $S\bar{d}_{\mathcal{V}}$ are g integer vectors. Thus, rational points $\bar{i} + (f/g)\bar{d}_{\mathcal{V}}$ are mapped to PEs with indices $S\bar{i} + (f/g)S\bar{d}_{\mathcal{V}}$, $f = 1, \cdots, g-1$. For example, under the space-time mapping T given in Fig. 2, (2, 0, 0) + (1/2)(0, 0, 1) is the rational point that mapped to PE with index 1. And, index points $\bar{i}$ and $\bar{i} + \bar{d}_{\mathcal{V}}$ are mapped to PEs with indices $S\bar{i}$ and $S\bar{i} + S\bar{d}_{\mathcal{V}}$, respectively. This implies that $token_{\mathcal{V}}(\bar{i})$ travels from the PE with index $S\bar{i}$, passes through PEs with indices $S\bar{i} + (1/g)S\bar{d}_{\mathcal{V}}, \cdots, S\bar{i} + ((g-1)/g)(S\bar{d}_{\mathcal{V}})$ and arrives at the PE with index $S\bar{i} + S\bar{d}_{\mathcal{V}}$. Let $\bar{i} \neq \bar{i}' + z\bar{d}_{\mathcal{V}}$, for all $z \in \mathcal{Z}$, $\bar{i}, \bar{i}' \in J$. Then, by the Restriction of the algorithm, tokens $token_{\mathcal{V}}(\bar{i})$ and $token_{\mathcal{V}}(\bar{i}')$ represent two different tokens. Assume that there is a link conflict between tokens $token_{\mathcal{V}}(\bar{i})$ and $token_{\mathcal{V}}(\bar{i}')$. Then, $token_{\mathcal{V}}(\bar{i})$ and $token_{\mathcal{V}}(\bar{i}')$ arrive at the same PE at the same time and move together along the link.

In other words, there exist two rational points $\bar{i} + r_1\bar{d}_{\mathcal{V}}$ and $\bar{i}' + r_2\bar{d}_{\mathcal{V}}$, $r_1, r_2 \in \mathcal{Q}$, such that $S(\bar{i} + r_1\bar{d}_{\mathcal{V}}) = S(\bar{i}' + r_2\bar{d}_{\mathcal{V}})$ and $\Lambda(\bar{i} + r_1\bar{d}_{\mathcal{V}}) = \Lambda(\bar{i}' + r_2\bar{d}_{\mathcal{V}})$. After simple manupulation, we have

$$\Lambda(\bar{i} - \bar{i}' + (r_1 - r_2)\bar{d}_{\mathcal{V}}) = 0$$
$$S(\bar{i} - \bar{i}' + (r_1 - r_2)\bar{d}_{\mathcal{V}}) = 0.$$

Thus, we have $T\overline{\gamma} = 0$, where $\overline{\gamma} = (\bar{i} - \bar{i}' + q\bar{d}_{\mathcal{V}})$ is a rational vector and $q \in \mathcal{Q}$. Let integer vector $\overline{y} = \bar{i} - \bar{i}' = \overline{\gamma} - q\bar{d}_{\mathcal{V}} \in \mathcal{Z}^{n\times 1}$. Since $\bar{i} \neq \bar{i}' + z\bar{d}_{\mathcal{V}}$ for all $z \in \mathcal{Z}$, we have that if there exists a nonzero integer vector $\overline{y}$ such that $\overline{y} = \bar{i} - \bar{i}' \neq z\bar{d}_{\mathcal{V}}$ for all $z \in \mathcal{Z}$, then there exists a link conflict between $token_{\mathcal{V}}(\bar{i})$ and $token_{\mathcal{V}}(\bar{i}')$. Thus, the link conflict checking problem can be solved by checking the nonzero integer solution $\overline{y}$ of the following mixed integer linear programming problem.

Mixed Integer Linear Programming (MILP) problem:

$$\begin{bmatrix} \Lambda \\ S \end{bmatrix} \overline{\gamma} = 0$$
$$\overline{y} = \overline{\gamma} - q\bar{d}_{\mathcal{V}}$$
$$\overline{y} \in diff(J)$$
$$\overline{y} \in \mathcal{Z}^{n\times 1}$$
$$\overline{\gamma} \in \mathcal{Q}^{n\times 1}$$
$$q \in \mathcal{Q}$$

where $diff(J) = \{\bar{i}_1 - \bar{i}_2 \mid \bar{i}_1, \bar{i}_2 \in J\}$.

Clearly, $diff(J)$ is a convex polytope which is symmetric to the origin. Notice that a nonzero vector $\overline{y}$ such that $\overline{y} = \bar{i} - \bar{i}' = z\bar{d}_{\mathcal{V}} \in diff(J)$ for an integer z is always a solution of the MILP problem, since we can set $\overline{\gamma} = 0$ and choose $q = -z$. In this case, since $token_{\mathcal{V}}(\bar{i})$ and $token_{\mathcal{V}}(\bar{i}')$ represent the same token, there is no link conflict. If there exists a nonzero integer solution $\overline{y} = \bar{i} - \bar{i}' \neq z'\bar{d}_{\mathcal{V}}$, $z' \in \mathcal{Z}$, then a link conflict between tokens $token_{\mathcal{V}}(\bar{i})$ and $token_{\mathcal{V}}(\bar{i}')$ exists. Thus, to check the link conflicts for the tokens of the variable $\mathcal{V}$, all nonzero integer solutions of the MILP problem need to be enumerated.

4 Checking Link Conflicts

In this section, we give a method to enumerate the integer solutions $\overline{y}$ of the MILP problem so as to check link conflicts for a variable $\mathcal{V}$ with its associated data dependence vector $\bar{d}_{\mathcal{V}}$. As $\overline{y} = \overline{\gamma} - q\bar{d}_{\mathcal{V}}$ in the MILP problem where $\overline{\gamma} \in \mathcal{Q}^{n\times 1}$ and $q \in \mathcal{Q}$, enumeration of the integer solutions $\overline{y}$ is not a straightforward task. Thus, we derive a representation for the integer solutions $\overline{y}$. Based on this representation, all integer solutions $\overline{y} \in diff(J)$ can be enumerated systematically.

4.1 A Representation of the Integer Solutions

In order to enumerate the integer solutions $\overline{y}$, we represent $\overline{y}$ as an integer linear combination of some vectors. First, we expressed $\overline{\gamma} = \overline{y} + q\bar{d}_{\mathcal{V}}$ as a linear combination of some integer vectors as follows. Denote $NULL(T) = \{\overline{x} \mid T\overline{x} = 0\}$, the null space of the space-time mapping matrix T. Since $T\overline{\gamma} = 0$ in the MILP problem, we have $\overline{\gamma} \in NULL(T)$. Thus, $\overline{\gamma}$ can be expressed as a linear combination of the vectors of a basis spanning $NULL(T)$. In order to find a basis of the $NULL(T)$, we introduce the notion of the Hermite normal form.

Definition 4.1. *A matrix is* unimodular *if and only if it is integral and the absolute value of its determinant is one.*

Theorem 4.1 (Hermite normal form) [21, p. 45]. *Let $T \in \mathcal{Z}^{(k+1)\times n}$ and $rank(T) = k+1$. Then, there exists a unimodular matrix $W \in \mathcal{Z}^{n\times n}$ such that $TW = H = [L, \mathbf{0}]$ ($\mathbf{0}$ denotes a zero matrix), where $L \in \mathcal{Z}^{(k+1)\times(k+1)}$ is a nonsingular and lower triangular matrix. Matrix H is called the* Hermite normal form *of T.*

From Theorem 4.1, we have $T = HW^{-1}$. Let $W = [\overline{w}_1, \cdots, \overline{w}_n]$. Then, $T\overline{\gamma} = 0$ can be rewritten as $HW^{-1}\overline{\gamma} = 0$. Let $\overline{\zeta} = W^{-1}\overline{\gamma} = (\zeta_1, \cdots, \zeta_n)^T$. Then, we have the following lemma.

Lemma 4.2 [15].

$$\overline{\gamma} = (\overline{w}_{k+2}, \cdots, \overline{w}_n) \begin{bmatrix} \zeta_{k+2} \\ . \\ . \\ . \\ \zeta_n \end{bmatrix},$$

where $\zeta_i \in \mathcal{Q}$, $i = k+2, \cdots, n$.

Proof. After simple manipulation, the lemma follows. End of proof. □

Let $M = (\bar{d}_{\mathcal{V}}, \overline{w}_{k+2}, \cdots, \overline{w}_n) \in \mathcal{Z}^{n\times(n-k)}$. By Lemma 4.2, $\overline{w}_i \in NULL(T)$, $i = k+2, k+3, \cdots, n$. In addition, since $\mid det(W) \mid = 1$, $\overline{w}_{k+2}, \cdots, \overline{w}_n$ are linear independent vectors. Moreover, since $S\bar{d}_{\mathcal{V}} \neq 0$, we have $T\bar{d}_{\mathcal{V}} \neq 0$. Thus, $\bar{d}_{\mathcal{V}} \notin NULL(T)$. As a consequence, $rank(M) = n - k$. Let

$$\overline{r} = (-q, \zeta_{k+2}, \cdots, \zeta_n)^T \in \mathcal{Q}^{(n-k)\times 1}.$$

We have $\overline{y} = M\overline{r}$ where M is an integer matrix and $\overline{r}$ a rational vector. Since $\overline{y}$ is a *rational* linear combination of the columns of matrix M, vectors $\overline{y}$ cannot be enumerated. Thus, we must select another set of linear independent vectors such that vector $\overline{y}$ can be expressed as an integer linear combination of those vectors. To achieve this, the notion of the Smith normal form is introduced.

Theorem 4.3 (Smith normal form) [21, p. 50]. *Given a matrix $A \in \mathcal{Z}^{n\times m}$, there exist two unimodular matrices $U \in \mathcal{Z}^{n\times n}$ and $V \in \mathcal{Z}^{m\times m}$ such that*

$$UAV = S(A) = \begin{bmatrix} s_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & s_2 & \cdots & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & s_{m'} & 0 & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

$S(A)$ is called the Smith normal form of matrix A, $S(A)$ is unique, $s_1, \cdots, s_{m'}$ are positive integers satisfying $s_1 \mid s_2 \mid \cdots \mid s_{m'}$, $\Pi_{i=1}^{k} s_i$, $k = 1, \cdots, m'$, is the greatest common divisor of subdeterminants of order k of the matrix A, and m' is the rank of the matrix A.

Now, let $S(M) \in \mathcal{Z}^{n\times(n-k)}$ be the Smith normal form of the matrix $M \in \mathcal{Z}^{n\times(n-k)}$. Then, there exist two unimodular matrices

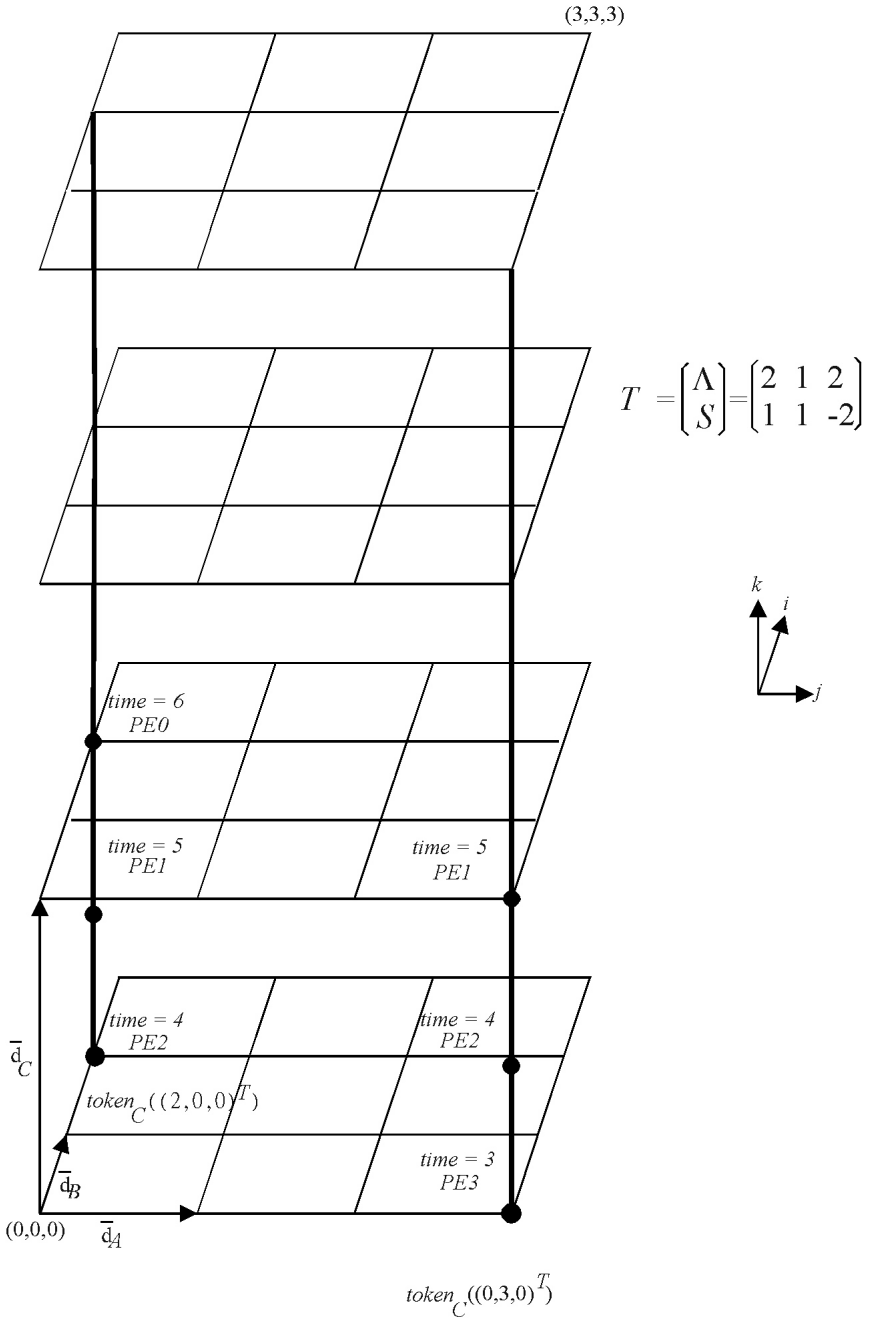

Fig. 2. The index set J and the data dependence vectors $\bar{d}_A$, $\bar{d}_B$, and $\bar{d}_C$ associated with variables A, B, and C, respectively.

$U \in \mathcal{Z}^{n\times n}$ and $V \in \mathcal{Z}^{(n-k)\times(n-k)}$ such that $S(M) = UMV$. Denote $U^{-1} = (\bar{u}_1', \cdots, \bar{u}_n')$. The following theorem shows that integer vector $\bar{y}$ can be expressed as an integer linear combination of the first $n-k$ columns of matrix U^{-1}.

Theorem 4.4.

$$\bar{y} = (\bar{u}_1', \cdots, \bar{u}_{(n-k)}') \begin{bmatrix} z_1 \\ \cdot \\ \cdot \\ \cdot \\ z_{n-k} \end{bmatrix},$$

where $z_i \in \mathcal{Z}$, $i = 1, \cdots, n-k$.

Proof. In this proof, we use the notation $A_{n\times m}$ to indicate that matrix A is of size $n \times m$. Since

$$S(M)_{n\times(n-k)} = U_{n\times n} M_{n\times(n-k)} V_{(n-k)\times(n-k)}$$

and

$$\bar{y} = M_{n\times(n-k)} \bar{r}_{(n-k)\times 1},$$

we have

$$U_{n\times n} \bar{y}_{n\times 1} = S(M)_{n\times(n-k)} \bar{\beta}_{(n-k)\times 1},$$

where

$$\bar{\beta}_{(n-k)\times 1} = V^{-1}_{(n-k)\times(n-k)} \bar{r}_{(n-k)\times 1}.$$

Since $\bar{y}_{n\times 1}$ must be an integer vector and $U_{n\times n}$ is a unimodular matrix, we have $U_{n\times n}\bar{y}_{n\times 1} \in \mathcal{Z}^{n\times 1}$. As a consequence, we have

$S(M)_{n\times(n-k)}\overline{\beta}_{(n-k)\times 1} \in \mathcal{Z}^{n\times 1}$. Remember that $rank(M) = n-k$. And, diagonal entries $s_i, i = 1,\cdots,n-k$ of the matrix $S(M)$ are positive integer numbers. This implies that $\beta_i = z_i/s_i$, $z_i \in \mathcal{Z}$, $i = 1,\cdots,n-k$. Thus, after simple manipulation, $\overline{y}$ can be expressed as an integer linear combination of the first $n-k$ columns of matrix U^{-1}. This completes the proof. □

Notice that we place the data dependence vector $\overline{d}_{\mathcal{V}}$ in the first column of the matrix M. Let g be the greatest common divisor of the entries of vector $\overline{d}_{\mathcal{V}}$. By the restriction of the algorithm, g is equal to one. Now, by the construction of Smith normal form $S(M)$ [21, p. 50], its (1, 1) entry s_1 should divide all entries of $\overline{d}_{\mathcal{V}}$. Therefore, $s_1 = 1$. Since $\overline{d}_{\mathcal{V}}$ is the first column of M with $g = 1$, vector $\overline{d}_{\mathcal{V}}$ can be brought into $\overline{e}_1 = (1, 0\cdots,0)^T$ after suitable row operations applied on M. In other words, there exists a unimodular matrix U such that the first column of UM is equal to $\overline{e}_1$. Thus, to make the first column of $S(M) = UMV$ equal to $\overline{e}_1$, we can choose a unimodular matrix V such that its first column is equal to $\overline{e}_1$. Now, since $U^{-1}S(M) = MV$, by comparing their first columns, we have the following theorem.

Theorem 4.5. *There exist two unimodular matrices U, V such that first columns of UMV and V are both equal to $\overline{e}_1$; and the first column $\overline{u}'_1$ of U^{-1} is equal to $\overline{d}_{\mathcal{V}}$.*

4.2 Enumeration of the Integer Solutions

In Theorem 4.4, we have expressed all integer vectors $\overline{y}$ in the MILP problem as an integer linear combination of integer vectors. But, only integer vectors $\overline{y} \in diff(J)$ need to be checked for link conflicts of two tokens of a variable. Notice that the index set J can be expressed as a set of linear inequalities, i.e.,

$$J = \{\overline{x} \mid l_i \le \overline{a}_i^T\overline{x} \le u_i, \overline{a}_i \in \mathcal{Z}^{n\times 1}, \overline{x} \in \mathcal{Z}^{n\times 1}, \\ l_i, u_i \in \mathcal{Z}, u_i > l_i, i = 1,\cdots,a\}.$$

Since $diff(J) = \{(\overline{x} - \overline{x}') \mid \overline{x}, \overline{x}' \in J\}$, we have

$$diff(J) = \{\overline{x} \mid l_i - u_i \le \overline{a}_i^T\overline{x} \le u_i - l_i, \overline{a}_i \in \mathcal{Z}^{n\times 1}, \\ \overline{x} \in \mathcal{Z}^{n\times 1}, l_i, u_i \in \mathcal{Z}, u_i > l_i, i = 1,\cdots,a\}.$$

Thus, we see that $diff(J)$ is symmetric with respect to the origin and can be expressed as the set

$$\{\overline{x} \mid A\overline{x} \le \overline{b}', \overline{b}' > 0, A \in \mathcal{Z}^{2a\times n}, \overline{b}' \in \mathcal{Z}^{2a}, \overline{x} \in \mathcal{Z}^{n\times 1}\}.$$

Since $\overline{y} \in diff(J)$, we have $A\overline{y} \le \overline{b}'$. Let $U' = (\overline{u}'_1,\cdots,\overline{u}'_{n-k})$, and $\overline{z} = (z_1,\cdots,z_{n-k})^T$. By Theorem 4.4, we have $\overline{y} = U'\overline{z}$. By substituting $\overline{y} = U'\overline{z}$ in $A\overline{y} \le \overline{b}'$, the integer vectors $\overline{y}$ can be enumerated by finding the bounds z_i^{min} and z_i^{max} of the values z_i, i.e., $z_i^{min} \le z_i \le z_i^{max}$ for $i = 1,,n-k$ such that $AU'\overline{z} \le \overline{b}'$. Since the convex polytope R defined by $AU'\overline{z} \le \overline{b}'$ is symmetric with respect to the origin, $0 \in R$ and $z_i^{min} = -z_i^{max}$. Since R is a convex polytope, the enumeration of the feasible $\overline{z}$ (or the integer solutions $\overline{y} = U'\overline{z}$ of the MILP problem) terminates in a finite number of steps. The enumeration procedure is terminated as soon as a nonzero integer solution $\overline{y}$ is found such that $\overline{y} \ne z'\overline{d}_{\mathcal{V}}$ for any integer z' since, in this case, a link conflict is found.

Notice that, as described in Section 3, $\overline{y} = z'\overline{d}_{\mathcal{V}}$ for some integers z' are solutions of the MILP problem. The relation of these $\overline{y}$s and the link conflicts is stated in the following theorem.

Theorem 4.6. *Let $T = \begin{bmatrix}\Lambda\\ S\end{bmatrix}$ be a space-time mapping with $S\overline{d}_{\mathcal{V}} \ne 0$. Let $\overline{y} = (\overline{u}'_1,\cdots,\overline{u}'_{n-k})(z_1,\cdots,z_{n-k})^T$ be a nonzero integer solution of the MILP problem. Then, there is no link conflict for the tokens of the variable $\mathcal{V}$ if and only if all of z_i for $i = 2,\cdots,n-k$ have zero values only.*

Proof. Notice that $\overline{y} = z_1\overline{d}_{\mathcal{V}} + (\overline{u}'_2,\cdots,\overline{u}'_{n-k})(z_2,\cdots,z_{n-k})^T$ by Theorems 4.4 and 4.5. Since $token(\overline{i})$ and $token(\overline{i}')$ represent two different tokens if and ony if $\overline{i} \ne \overline{i}' + z\overline{d}_{\mathcal{V}}$ for $z \in \mathcal{Z}$, and $\overline{y} = \overline{i} - \overline{i}'$ represents link conflicts of two tokens at any two index points $\overline{i}$ and $\overline{i}'$, the theorem follows. End of the proof. □

From Theorem 4.6, we see that if there exists a nonzero integer vector $\overline{z} = (z_1, z_2,\cdots,z_{n-k})$ which has a form other than $(*, 0,\cdots,0)$, then there is a link conflict. Let R be the convex polytope defined by $A\overline{y} = AU'\overline{z} \le \overline{b}'$. To check whether there is an integer vector $\overline{z}$ which has a form other than $(*, 0,\cdots,0)$, we need to enumerate the integer vectors $\overline{z}$ in R. Since R is symmetric with respect to the origin, only half space of R needs to be enumerated. To enumerate it, first, we construct a minimal hypercube C containing the half space of R. The minimal hypercube C is

$$C = \{(z_1, z_2,\cdots,z_{n-k}) \mid 0 \le z_1 \le \xi_1, -\xi_i \le z_i \le \xi_i, 2 \le i \le n-k\},$$

where $\xi_i = \lfloor z_i^{max}\rfloor$ and z_i^{max} is found by solving the following linear program: $\{max\ \ z_i \mid AU'\overline{z} \le \overline{b}', z_i \in \mathcal{Q}\}$. Then, integer vectors $\overline{z} \in C$ are enumerated in lexicographical order. For the special case $k = n-2$, the enumeration of integer vectors is not necessary as stated by the following theorem.

Theorem 4.7. *Link conflicts on mapping an n-dimensional uniform dependence algorithm into an $(n-2)$-dimensional processor array can be checked by solving two linear programs in one variable.*

Proof. Since the processor array is $(n-2)$-dimensional, solution space R is a convex polytope in the space spanning by vectors $\overline{d}_{\mathcal{V}}$ and $\overline{u}'_2$ by Theorem 4.4 and Theorem 4.5. Notice that $\overline{y} = z_1\overline{d}_{\mathcal{V}} + z_2\overline{u}'_2$ and $\overline{y} \in diff(J) = \{\overline{i} - \overline{i}' \mid \overline{i}, \overline{i}' \in J\}$. These observations imply that the convex polytope R defined by $AU'\overline{z} = A\overline{y} \le \overline{b}'$ have at least three integer solutions $\overline{y} = -\overline{d}_{\mathcal{V}}$, 0, and $+\overline{d}_{\mathcal{V}}$; otherwise, the computation in each index point in index set J uses (generates) a value of the variable to be input (output) from (to) the host. Since R is a convex polytope and symmetric with respect to the origin and contains three integer points $(-1, 0)^T$, $(0, 0)^T$, and $(1, 0)^T$, R must contains an integer point $(z_1, 1)$ if R contains rational point $(z_1, z_2)^T$ such that $\mid z_2 \mid \ge 2$. Thus, link conflicts can be checked by testing whether R contains an integer point $(z_1, 1)$. In other words, solve the following two linear programs: $\{min\ \ z_1 \mid AU'(z_1, 1)^T \le \overline{b}', z_1 \in \mathcal{Q}\}$ and $\{max\ \ z_1 \mid AU'(z_1, 1)^T \le \overline{b}', z_1 \in \mathcal{Q}\}$. If those linear programs have optimal solutions such that $\lfloor z_1^{max}\rfloor \ge \lceil z_1^{min}\rceil$, there is a link conflict. Otherwise, there is no link conflict. This completes the proof. □

5 Time Complexity Estimation

Now, we compare the time complexity of our method to that of Lee and Kedem [13], [14] and that of Xue [16], [18] or Ganapathy and Wah [19], [20]. Let N be the problem size parameter for the n-dimensional uniform dependence algorithm. To check link conflicts, Lee and Kedem [13], [14] enumerated all pairs of index points in the index set J. Since J is of size $O(N^n)$, the time complexity is $O(N^{2n})$. Xue [16], [18] (or Ganapathy and Wah [19], [20]) examined the I/O space instead of the whole index set. In order to check the link conflicts exactly, all index points in the I/O space need to be enumerated. Thus, the time complexity is $O(N^{n-1})$. By our approach, as argued below, only $O((2N)^{n-k})$ integer vectors needed to be enumerated, where k is the dimension of the processor array.

Recall that the convex polytope R is defined by $AU'\overline{z} \leq \overline{b}'$. Since each component of $\overline{b}'$ is of size $O(N)$ (see the first paragraph of Section 4.2) and AU' is an integer matrix, the number of z_is to be enumerated is $O(2N)$. Therefore, the total number of integer vectors to be enumerated is $O((2N)^{n-k})$. The additional cost that we pay for it is the time taken to find a representation of the integer solutions. To find the representation, we need to compute the Hermite normal form of space-time mapping matrix $T = \begin{bmatrix} \Lambda \\ S \end{bmatrix}$ and the Smith normal form $S(M)$ of matrix M corresponding to the data dependence vector $\overline{d}_v$. Using the algorithms in [22] and [23, appendix A], both forms can be computed in polynomial time. Consequently, the representation of the integer solutions can be found in polynomial time.

6 Conclusion

In this paper, a new formulation for the checking of link conflicts on mapping the uniform dependence algorithms into lower dimensional processor arrays with link connection between neighboring PEs is proposed. The formulation is a mixed integer linear program (MILP) and an integer solution of it can represent a link conflict of two tokens of a variable. To check the existence of nonzero integer solutions of the MILP, the integer solutions are represented as an integer linear combination of basis vectors. By this representation, we found that the integer vectors can be enumerated in $O((2N)^{n-k})$ time complexity, where N is the problem size, n and k are dimensions of the algorithms and processor arrays, respectively. (Previous methods to check link conflicts require an enumeration of the index set or I/O space whose size are $O(N^{2n})$ or $O(N^{n-1})$, respectively.) We have shown that the basis vectors can be found in ploynomial time complexity. For the special case, $k = n - 2$, we have shown that link conflicts can be checked by solving two linear programs in one variable.

A closed related work on mapping the n-dimensional uniform dependence algorithms into lower dimensional arrays was proposed in [15]. However, there is a main difference in implementation requirements between our method and that in [15]. First, they allow the data arrive before the time of their usage at the PE. Thus, a large bandwith is required between neighboring PEs to support the necessary data movement. For our method, the data must arrive at the PE at the time of their usage. And, the buffer size between neighboring PEs can be derived when the design is complete. Second, if a large bandwith is not possible, their method does not guarantee free of link conflicts for some designs. For more details, see [24]. But, all the designs produced by our method are free of link conflicts.

The applicability of our method can be seen by the following arguments: First, our method can be applied to uniform dependence algorithms with arbitrary convex index sets. Second, using the approach proposed in [13], our method can also be applied to the case that the components of the dependence vector are not relative-prime. Their approach is the use of a modified array model. For details, see their paper [13, pp. 68-69]. Third, since synthesizing processor arrays from algorithms is divided into two main steps, uniformization step [25] and mapping step, our method can be used in the mapping step after the algorithm is uniformized.

Acknowledgments

This research was supported by the National Science Council of the Republic of China under contract NSC88-2213-E-036-003.

References

[1] H.T. Kung, "Why Systolic Architectures?," *Computer*, vol. 15, no. 1, pp. 37-46, Jan. 1982.
[2] S.Y. Kung, *VLSI Array Processor*. Englewood Cliffs, N.J.: Prentice-Hall, 1988.
[3] W. Shang and J.A.B. Fortes, "Time Optimal Linear Schedules for Algorithms with Uniform Dependencies," *IEEE Trans. Computers*, vol. 40, no. 6, pp. 723-742, June 1991.
[4] P.R. Cappello and K. Steiglitz, "Unifying VLSI Array Designs with Geometric Transformations," *Proc. Int'l Conf. Parallel Processing*, pp. 448-457, 1983.
[5] C. Guerra and R. Melhem, "Synthesizing Non-Uniform Systolic Designs," *Proc. Int'l Conf. Parallel Processing*, pp. 765-771, 1986.
[6] R.M. Karp, R.E. Miller, and S. Winograd, "The Organization of Computations for Uniform Recurrence Equations," *J. ACM*, vol. 14, pp. 563-590, July 1967.
[7] G.J. Li and B.W. Wah, "The Design of Optimal Systolic Arrays," *IEEE Trans. Computers*, vol. 34, no. 1, pp. 66-77, Jan. 1985.
[8] D.I. Moldovan and J.A.B. Fortes, "Partitioning and Mapping Algorithms into Fixed Size Systolic Arrays," *IEEE Trans. Computers*, vol. 35, no. 1, pp. 1-12, Jan. 1986.
[9] P. Quinton, "Automatic Synthesis of Systolic Arrays from Uniform Recurrent Equations," *Proc. Int'l Symp. Computer Architecture*, pp. 208-214, 1984.
[10] S.K. Rao, "Regular Iterative Algorithms and Their Implementations on Processor Arrays," PhD thesis, Standford Univ., 1985.
[11] L. Lamport, "The Parallel Execution of DO Loops," *Comm. ACM*, vol. 17, pp. 83-93, Feb. 1974.
[12] J.A.B. Fortes and F. Parisi-Presicce, "Optimal Linear Schedules for the Parallel Execution of Algorithms," *Proc. Int'l Conf. Parallel Processing*, pp. 322-329, 1984.
[13] P.Z. Lee and Z.M. Kedem, "Mapping Nested Loop Algorithms into Multidimensional Systolic Arrays," *IEEE Trans. Parallel and Distributed Systems*, vol. 1, no. 1, pp. 64-76, Jan. 1990.
[14] P.Z. Lee and Z.M. Kedem, "Synthesizing Linear Array Algorithms from Nested for Loop Algorithms," *IEEE Trans. Computers*, vol. 37, no. 12, pp. 1,578-1,598, Dec. 1988.
[15] W. Shang and J.A.B. Fortes, "On Time Mapping of Uniform Dependence Algorithms into Lower Dimensional Processor Arrays," *IEEE Trans. Parallel and Distributed Systems*, vol. 3, no. 5, pp. 350-363, May 1992.
[16] J. Xue, "Avoiding Data Link and Computational Conflicts in Mapping Nested Loop Algorithms to Lower-Dimensional Processor Arrays," *Proc. Int'l Conf. Parallel and Distributed Systems*, pp. 567-572, 1994.
[17] J. Xue, "Closed-Form Mapping Conditions for the Synthesis of Linear Processor Arrays," *J. VLSI Signal Processing*, vol. 10, pp. 181-199, 1995.
[18] J. Xue, "A Unified Approach to Checking Data Link and Computational Conflicts in the Design of Algorithm-Specific Processor Arrays," Technical Report 94-100, Dept. of Mathematics, Statistics, and Computing Science, Univ. of New England, 1994.
[19] K.N. Ganapathy and B.W. Wah, "Optimal Synthesis of Algorithm-specific Lower-Dimensional Processor Arrays," *IEEE Trans. Parallel and Distributed Systems*, vol. 7, no. 3, pp. 274-287, Mar. 1996.
[20] K.N. Ganapathy and B.W. Wah, "Synthesizing Optimal Lower Dimensional Processor Arrays," *Proc. Int'l Conf. Parallel Processing*, pp. III. 96-III. 103, 1992.
[21] A. Schrijver, *Theory of Linear and Integer Programming*, John Wiley & Sons, 1986.
[22] R. Kannan and A. Bachem, "Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix," *SIAM J. Computing*, vol. 8, no. 4, pp. 499-507, 1979.
[23] T.C. Hu, *Integer Programming and Network Flows*. Addison-Wesley, 1969.
[24] J.A.B. Fortes, B.W. Wah, W. Shang, and K.N. Ganapathy, "Algorithm-Specific Parallel Processing with Linear Processor Arrays," *Advances in Computers*, M. Yovits, ed. Academic Press, 1993.
[25] V. Van Dongen and P. Quinton, "Uniformization of Linear Recurrence Equations: A Step Towards the Automatic Synthesis of Systolic Arrays," *Proc. Int'l Conf. Systolic Array*, pp. 473-482, 1988.