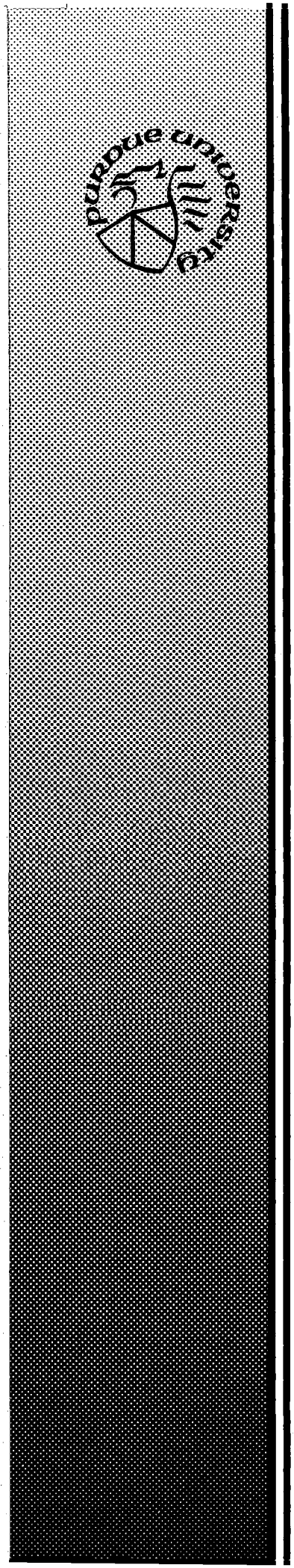
4-1-1990

Time-Optimal and Conflict-Free Mappings of Uniform Dependence Algorithms into Lower Dimensional Processor Arrays

Weijia Shang
Purdue University, shaug@ecn.purdue.edu

Jose A. B. Fortes
Purdue University, fortes@ecn.purdue.edu



Time-Optimal and Conflict-Free Mappings of Uniform Dependence Algorithms into Lower Dimensional Processor Arrays

Weijia Shang
Jose A. B. Fortes

TR-EE 90-29
April 1990

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

This research was supported in part by the National Science Foundation under Grant DC1-8419745 and in part by the Innovative Science and Technology Office of the Strategic Defense Initiative Organization and was administered through the Office of Naval Research under contracts No. 00014-85-k-0588 and No. 00014-88-k-0723.

TIME-OPTIMAL AND CONFLICT-FREE MAPPINGS OF UNIFORM DEPENDENCE ALGORITHMS INTO LOWER DIMENSIONAL PROCESSOR ARRAYS

Weijia Shang and Jose A. B. Fortes
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907
shang@ecn.purdue.edu and fortes@ecn.purdue.edu
(317) 494-3500 and (317) 494-3646

Key Words: processor array; time-optimal mapping; conflict-free; nested loops; bit-level algorithm.

Abstract

Most existing methods of mapping algorithms into processor arrays are restricted to the case where n-dimensional algorithms or algorithms with n nested loops are mapped into $(n-1)$-dimensional arrays. However, in practice, it is interesting to map n-dimensional algorithms into $(k-1)$-dimensional arrays where $k<n$. For example, many algorithms at bit-level are at least 4-dimensional (matrix multiplication, convolution, LU decomposition, *etc.*) and most existing bit level processor arrays are 2-dimensional. A *computational conflict* occurs if two or more computations of an algorithm are mapped into the same processor and the same execution time. In this paper, necessary and sufficient conditions are derived to identify all mappings without computational conflicts, based on the Hermite normal form of the mapping matrix. These conditions are used to propose methods of mapping any n-dimensional algorithm into $(k-1)$-dimensional arrays, $k<n$, without computational conflicts. When $k \geq n-3$, optimality of the mapping is guaranteed.

This research was supported in part by the National Science Foundation under Grant DC1-8419745 and in part by the Innovative Science and Technology Office of the Strategic Defense Initiative Organization and was administered through the Office of Naval Research under contracts No. 00014-85-k-0588 and No. 00014-88-k-0723.

List of Symbols

B: $(n-1)\times(n-1)$ matrix; $T=[B,\ \bar{b}]$.
$\bar{b}$: $(n-1)$-entry column vector; $T=[B,\ \bar{b}]$.
D: dependence matrix; Definition 2.1.
$detB$: determinant of matrix B.
$\bar{d}_i$: dependence (column) vector with n components; Definition 2.1 (4).
H: the Hermite normal form of mapping matrix T; Theorem 4.1.
I: identity matrix.
J: index set; Definition 2.1 (1).
$\bar{j}$: column vector; index point; Definition 2.1 (1).
k: number of rows of mapping matrix T; Definition 2.2.
m: number of dependence vectors in D; Definition 2.1 (4).
N: set of non-negative integers.
N^+: set of positive integers.
n: algorithm dimension or number of entries of index points in J; Definition 2.1 (1).
$rank(A)$: rank of matrix A.
S: space mapping matrix; Definition 2.2.
s_{ij}: entry of S at *ith* row and *jth* column.
T: mapping matrix; Definition 2.2.
U: multiplier of the Hermite normal form; Theorem 4.1.
u_{ij}: entry of U at *ith* row and *jth* column.
V: inverse of U; Theorem 4.1.
v_{ij}: entry of V at *ith* row and *jth* column.
Z: set of integers.
Π: row vector; linear schedule vector; Definition 2.2.
Π^o: row vector; optimal solution of Problem 2.2.
$\bar{\beta}$: $\bar{\gamma}=U\bar{\beta}$
β_i: *ith* entry of $\bar{\beta}$.
$\bar{\gamma}$: conflict vector; Definition 2.3.
γ_i: *ith* entry of $\bar{\gamma}$.
μ_i: the *ith* upper bound of J; Equation 2.5.
τ: linear mapping of algorithms into arrays; Definition 2.2.
$\varnothing$: empty set.
0: a column or row vector whose entries are all 0.
$|C|$: cardinality of set C.
$|\alpha|$: absolute value of α.

1. INTRODUCTION

Most existing methods of mapping algorithms into processor arrays are restricted to the cases where n-dimensional algorithms, or algorithms with n nested loops, are mapped into $(n-1)$-dimensional processor arrays [2-13]. For example, the 3-dimensional matrix multiplication algorithm is usually mapped into a 2-dimensional processor array by these methods [10], [21], [32]. This paper considers mappings of n-dimensional algorithms into $(k-1)$-dimensional, $k<n$, processor arrays. Procedures are proposed to find mappings without *computational conflicts*, which means no two or more computations of the algorithm are mapped into the same processor and execution time. When $k \geq n-3$, these mappings are time-optimal.

In simple terms, the algorithms under consideration in this paper are called *uniform dependence algorithms* and can model nested loop algorithms. They are represented as partially ordered subsets of a multidimensional integer lattice (called *index sets*). The points of this lattice correspond to (or index) computations, and the partial order reflects the data dependencies between these computations. These data dependencies are represented as vectors that connect points of the lattice. Informally, if a given dependence vector is always present when the vector difference between any two lattice points equals the dependence vector, then the dependence is said to be *uniform*. If all dependencies are uniform then the algorithm is said to be a *uniform dependence algorithm*. This algorithm model can be easily related to similar models and concepts in [1-13], [19] and several other works.

Examples of 2-dimensional bit-level processor arrays include GAPP [33], DAP [34], MPP [35], Connection Machine [31] *etc.* Many bit level algorithms are four or five dimensional, such as matrix multiplication, convolution, LU decomposition, *etc.* How to automatically map these algorithms into 2-dimensional bit level arrays is still a problem [28]. That is why in practice it is interesting to develop a method to map n-dimensional algorithms into $(k-1)$-dimensional processor arrays with $k<n$. This work was motivated by the implementation of RAB (Reconfiguration Algorithm for Bit level code) [26], an experimental tool which maps a class of algorithms programmed in 'C' into bit level arrays. In this approach, algorithms are first expanded into bit level algorithms, and second, the dependence relations are analyzed and the algorithm is uniformized. Then the global optimal solution, which maps often a four or five dimensional bit level algorithm into a 2-dimensional bit level processor array, is to be found.

Several attempts have been made to try to map algorithms into lower dimensional systolic arrays [15], [22], [23] [25]. In particular, important steps towards a formal solution to this problem were made in [23]. Based on the Lamport hyperplane transformation model [13], a procedure was proposed to find mappings of 3-dimensional algorithms into 1-dimensional or linear systolic arrays without computational conflicts and data link collisions. Five conditions were given to guarantee the correctness of the mapping. The first condition ensures that dependence

relations among different computations of the algorithm are respected; the second condition is about computational conflicts; the third and fourth conditions deal with the number of shift registers on links and the data travel directions; and the fifth condition is to avoid data link collisions. The concept of data link collisions and the conditions to avoid such collisions are introduced in this work. Detection of computational conflicts is basically by analysis of all computations of the algorithm and the optimality of the mapping is not guaranteed. In [22], further results are reported in mapping n-dimensional algorithms into $(k-1)$-dimensional processor arrays. A suboptimal solution for the reindexed transitive closure algorithm [17] [23] was found by the proposed procedure in [22] by which the total execution time is $\mu(2\mu+3)+1$ where μ is the problem size.

This paper describes a method of mapping n-dimensional uniform dependence algorithms into $(k-1)$-dimensional arrays, $k<n$, without any computational conflicts. Based on the Hermite normal form of the mapping matrix, simple and easy-to-use necessary and sufficient conditions are derived to guarantee a conflict-free mapping. These conditions are used to formulate the problem of finding time-optimal and conflict-free mappings as an integer programming problem. Optimality is always guaranteed for the mapping of n-dimensional algorithms into $(n-i)$-dimensional, $i=1, \ldots, 4$, processor arrays. Compared to the method in [22] and [23], the main contribution of this paper is the easy-to-use and closed form necessary and sufficient conditions for conflict-free mappings. In addition, based on these conditions, this paper formulates the problem of identifying time-optimal and conflict-free mappings as an integer programming optimization problem. For some algorithms such as the matrix multiplication algorithm and the transitive closure algorithm, the integer programming formulation can be further converted to linear programming problems. In Section 5, the method proposed in this paper is used to find the optimal solution for the reindexed transitive closure algorithm which improves the total execution time of $\mu(2\mu+3)+1$ in [22] to $\mu(\mu+3)+1$.

This paper is organized as follows. Section 2 presents basic terminology and definitions, introduces the concept of computational conflicts and provides statements of problems addressed in this paper. Section 3 discusses a simple case to illustrate different aspects of, and provide insight into, the conflict-free mapping problem. Section 4 discusses the conflict-free mapping problem in general. Section 5 presents an optimization procedure and integer programming problem formulations which find the time-optimal mapping without any computational conflicts. Section 6 concludes this paper and points out some future work.

2. TERMINOLOGY AND DEFINITIONS

Throughout this paper, *sets*, *matrices* and *row vectors* are denoted by capital letters, *column vectors* are represented by lower case symbols with an overbar and *scalars* correspond to lower case letters. The *transpose* of a vector $\bar{v}$ is denoted $\bar{v}^T$. The vector $\bar{0}$ denotes the row or column vector whose entries are all zeroes. The dimensions of vector $\bar{0}$ and whether it denotes a row or column vector are implied

by the context in which they are used. The symbol I denotes the identity matrix. The rank and the determinant of matrix A are denoted *rank(A)* and *detA*, respectively. The set of integers, the set of non-negative integers and the set of positive integers are denoted Z, N and N^+, respectively. The empty set is denoted $\varnothing$. The notations $|C|$ and $|\alpha|$ represent the cardinality or the number of elements of set C and the absolute value of scalar α, respectively. Let $\bar{v}$ and $\bar{u}$ be two vectors. Then $\bar{v} \geq \bar{u}$ means every component of $\bar{v}$ is greater than or equal to the corresponding component of $\bar{u}$. Finally, if x is an element of a set S, the notation $x \in S$ is used and this notation is also used to indicate that a column vector $\bar{m}_j$ (or row vector M_i) is a column (row) of a matrix M, i.e., $\bar{m}_j \in M$ ($M_i \in M$) means $\bar{m}_j$ (M_i) is a column (row) vector of matrix M.

Algorithms of interest in this paper are the so-called uniform dependence algorithms defined as follows.

Definition 2.1 (Uniform dependence algorithm): A *uniform dependence algorithm* is an algorithm that can be described by an equation of the form

$$v(\bar{j}) = g_{\bar{j}}(\, v(\bar{j} - \bar{d}_1),\ v(\bar{j} - \bar{d}_2),\ \ldots,\ v(\bar{j} - \bar{d}_m)\,) \tag{2.1}$$

where

(1) $\bar{j} \in J \subset Z^n$ is an index point (a column vector), J is the *index set* or *iteration space* of the algorithm and $n \in N^+$ is the *algorithm dimension* or the number of components of $\bar{j}$;

(2) $g_{\bar{j}}$ is the computation indexed by $\bar{j}$, i.e., a single-valued function computed "at point $\bar{j}$" in a *single unit of time*;

(3) $v(\bar{j})$ is the value computed "at $\bar{j}$", i.e., the result of computing the right hand side of (2.1) and

(4) $\bar{d}_i \in Z^n$, $i=1, \ldots, m$, $m \in N$ are *dependence vectors*, also called *dependencies*, which are constant (i.e., independent of $\bar{j} \in J$); the matrix $D=[\bar{d}_1, \ldots, \bar{d}_m]$ is called the *dependence matrix*.

The class of uniform dependence algorithms is a simple extension of the class of algorithms described by uniform recurrence equations [1]. The main difference is that uniform dependence algorithms allow for different functions to be computed (in a unit of time) at different points of the index set. From a practical viewpoint, uniform dependence algorithms can be easily related to programs where (1) a single statement appears in the body of a multiply nested loop and (2) the indices of the variable in the left hand side of the statement differ by a constant from the corresponding indices in each reference to the same variable in the right hand side. Alternative computations can occur in each iteration as a result of a single conditional statement as long as data dependencies do not change. Nested loop programs with multiple statements can also use the techniques of this paper together with the alignment method discussed in [14] and [24]. Uniform dependence

algorithms occur frequently in scientific computing and digital signal processing applications.

For the purpose of finding time-optimal and conflict-free mappings, only structural information of the algorithm, i.e., the index set J and the dependence matrix D, is needed. Therefore, a uniform dependence algorithm with index set J and dependence matrix D is herein characterized simply by the pair (J,D). Each index vector $\bar{j}\in J$ corresponds to a computation; and computation $\bar{j}$ depends on computations $\bar{j}-\bar{d}_i\in J$, $i=1, \ldots, m$. It is assumed that, as in Definition 2.1, the letters n and m always denote the *algorithm dimension* and the number of dependence vectors, respectively.

Many models have been proposed to map algorithms into processor arrays. The *linear algorithm transformation* method proposed in [2], [12] and [32] is used in this paper and stated as follows.

Definition 2.2 (Linear algorithm transformation): A *linear algorithm transformation* maps an n-dimensional uniform dependence algorithm into a $(k-1)$-dimensional processor array according to the mapping:

$$\tau: J\rightarrow Z^k,\ \tau(\bar{j})=T\bar{j},\ \forall \bar{j}\in J \tag{2.2}$$

where $T=\begin{bmatrix} S \\ \Pi \end{bmatrix}\in Z^{k\times n}$ is the *mapping matrix*, $S\in Z^{(k-1)\times n}$ is the *space mapping matrix*, and $\Pi\in Z^{1\times n}$ is the *time mapping vector*, or *linear schedule vector*. The computation indexed by $\bar{j}\in J$ is executed at time $\Pi\bar{j}$ and at processor $S\bar{j}$. The mapping τ must satisfy the following conditions:

(1) $\Pi D > \bar{0}$.

(2) $SD=PK$ where $P\in Z^{(k-1)\times r}$ is the matrix of interconnection primitives of the target machine, $K\in Z^{r\times m}$ is such that

$$\sum_{j=1}^{r} k_{ji} \leq \Pi\, \bar{d}_i\ ,\quad i=1, \ldots, m. \tag{2.3}$$

(3) $\forall \bar{j}_1, \bar{j}_2\in J$, if $\bar{j}_1\neq\bar{j}_2$, then $\tau(\bar{j}_1)\neq\tau(\bar{j}_2)$ or $T\bar{j}_1\neq T\bar{j}_2$.

(4) The rank of T is equal to k or $rank(T)=k$.

Condition 1 in Definition 2.2 preserves the partial ordering induced by the dependence vectors. It is clear that if this condition is satisfied, then computation indexed by $\bar{j}\in J$ is scheduled to execute only after the executions of computations indexed by $\bar{j}-\bar{d}_i\in J$, $i=1, \ldots, m$ because $\Pi D>\bar{0}$, and therefore the dependence relation is respected.

The matrix of interconnection primitives P describes the connection links of processors in the array. For an array with each processor connected to its four nearest east, south, west and north neighbors, it has four interconnection primitives $[0, 1]^T$, $[0, -1]^T$, $[1, 0]^T$ and $[-1, 0]^T$ and matrix $P=\begin{bmatrix} 0 & 0 & 1 & -1 \\ 1 & -1 & 0 & 0 \end{bmatrix}$.

Condition 2 in Definition 2.2 guarantees that the space mapping can be implemented in a fixed systolic architecture with interconnection primitive matrix P. The summation in the left hand side of the inequality in (2.3) is the number of times of the usage of interconnection primitives to pass the datum caused by the dependence vector $\bar{d}_i$ from the source to the destination. The item in the right is the time units between the source usage and the destination usage of that datum. Assuming it takes one time unit for a datum to travel one interconnection primitive, the inequality must be satisfied to have the datum arrive before it is used. Condition 2 in Definition 2.2 may not be required when a new processor array is designed specially for the algorithm. It is required only when the algorithm is to be mapped into a processor array with a fixed interconnection structure.

Condition 3 is for avoiding computational conflicts because if $\tau(\bar{j}_1) = \tau(\bar{j}_2)$, then the computations indexed by $\bar{j}_1$ and $\bar{j}_2$ are mapped into the same processor and time and a conflict occurs. Condition 4 guarantees that the algorithm is to be mapped into a $(k-1)$-dimensional array but not a q-dimensional array, $q<k-1$. When $rank(T)=q+1<k$, there are exactly $q+1$ linearly independent rows in T, and all other rows of T are linear combinations of these $q+1$ linearly independent rows. Let T' be the matrix consisting of these $q+1$ linearly independent rows; then T can be transformed linearly to T' which means the algorithm is actually mapped into a q-dimensional processor array.

More constraints on the mapping τ are possible for some implementation requirements. In addition, different constraint forms from those in Definition 2.2 for the same implementation requirement can be used. For example, in [23], the inequality in (2.3) is required to be an equality which, means data must arrive right at the time of their usage and are not allowed to arrive before the usage. Also, in [23], constraints to avoid data link collisions are considered.

Because the execution of any computation needs one time unit as defined in Definition 2.1, the total execution time by the linear schedule vector Π is as follows:

$$t = \max \{ \Pi (\bar{j}_1 - \bar{j}_2) : \bar{j}_1, \bar{j}_2 \in J \} + 1 \tag{2.4}$$

For a class of practical algorithms, the loop bounds are constants. This kind of algorithm is characterized by the *constant-bounded index set* defined as

$$J=\{[j_1, ..., j_n]^T: 0\leq j_i\leq \mu_i,\ j_i\in Z,\ \mu_i\in N^+,\ i=1, ..., n\} \tag{2.5}$$

where zero and μ_i correspond to the lower and upper bounds of the *ith* loop, respectively. Upper bounds μ_i, $i=1, ..., n$, are called *problem size variables.* To simplify the problem, this paper is restricted to the algorithms with constant-bounded index sets. This assumption is summarized as follows.

Assumption 2.1: In this paper, the index sets under consideration are assumed to be constant-bounded defined formally by Equation 2.5.

Some other kinds of algorithms can be transformed into algorithms with constant-bounded index sets by a linear mapping of the index sets [12]. For an algorithm with a constant-bounded index set, because

$$\max\{\Pi(\bar{j}_1-\bar{j}_2): \bar{j}_1, \bar{j}_2 \in J\}=[\,|\pi_1|, \ldots, |\pi_n|\,]([\mu_1, \ldots, \mu_n]^T-\bar{0}), \quad (2.6)$$

the total execution time t in Equation 2.4 can be simplified to

$$t=1+\sum_{i=1}^{n} |\pi_i| \mu_i \quad (2.7)$$

It is clear from Equation 2.7 that the vector Π which minimizes the objective function t in Equation 2.7 is such that the absolute values of its entries $|\pi_i|$, $i=1, \ldots, n$ are as small as possible and with some constraints satisfied. In other words, if the absolute value of any one of the entries of the optimal Π is reduced by one, then the resulting vector is not a valid linear schedule vector. This conclusion is also indicated in [10] and [11] and is summarized in the following theorem.

Theorem 2.1 [10], [11]: The total execution time described in Equation 2.4 is a monotonically increasing function of $|\pi_i|$, $i=1, \ldots, n$, the absolute values of entries of vector Π.

A conflict occurs if two or more computations are mapped into the same processor and the same execution time. That is, for two distinct index points $\bar{j}_1$ $\bar{j}_2 \in J$, if $T\bar{j}_1 = T\bar{j}_2$, then there is a conflict. For the case where $k=n$ or T is a square matrix, that $rank(T)=n$ guarantees a conflict-free mapping because $T\bar{j}_1 = T\bar{j}_2$ if and only if $\bar{j}_1 = \bar{j}_2$. For the case where $k<n$, even when $rank(T)=k$, or matrix T has full row rank, there is at least one non-zero vector $\bar{\gamma}$ such that $T\bar{\gamma}=0$. Let $\bar{j}_1=\bar{j}_2+\bar{\gamma}$, then $T\bar{j}_1 = T\bar{j}_2$. If both $\bar{j}_1$ and $\bar{j}_2$ belong to the index set, then the computations indexed by $\bar{j}_1$ and $\bar{j}_2$, respectively, are mapped to the same processor and the same execution time and a conflict occurs. Therefore, it is much more difficult to find a mapping without conflicts when $k<n$ than for the case when $k=n$.

One possible way to avoid conflicts is to find the mapping matrix T such that, for any arbitrary index point $\bar{j} \in J$ and any $\bar{\gamma}$ that is a non-zero integral solution of equation $T\bar{\gamma}=0$, $\bar{j}+\bar{\gamma}$ does not belong to the index set J. This concept is illustrated by Figure 1 which shows a 2-dimensional index set $J=\{[j_1, j_2]^T: 0 \leq j_1, j_2 \leq 4, j_1, j_2 \in Z\}$. If $\bar{\gamma}$ is $\bar{\gamma}_1=[1, 1]^T$, then index points $\bar{j}=\bar{0}$ and $\bar{j}+\bar{\gamma}_1=[1, 1]^T$ both belong to index set J and computations indexed by $[0, 0]^T$, $[1, 1]^T$, $[2, 2]^T$, ..., $[4, 4]^T$ will be mapped into the same processor and the same execution time. Therefore, there is at least one conflict. However, if $\bar{\gamma}$ is $\bar{\gamma}_2=[3, 5]^T$, there will be no conflict at all because for any arbitrary $\bar{j} \in J$, $\bar{j}+\bar{\gamma}_2 \notin J$. Intuitively, if vector $[3, 5]^T$ is drawn with one end at $[0, 0]^T$ (or at any other index

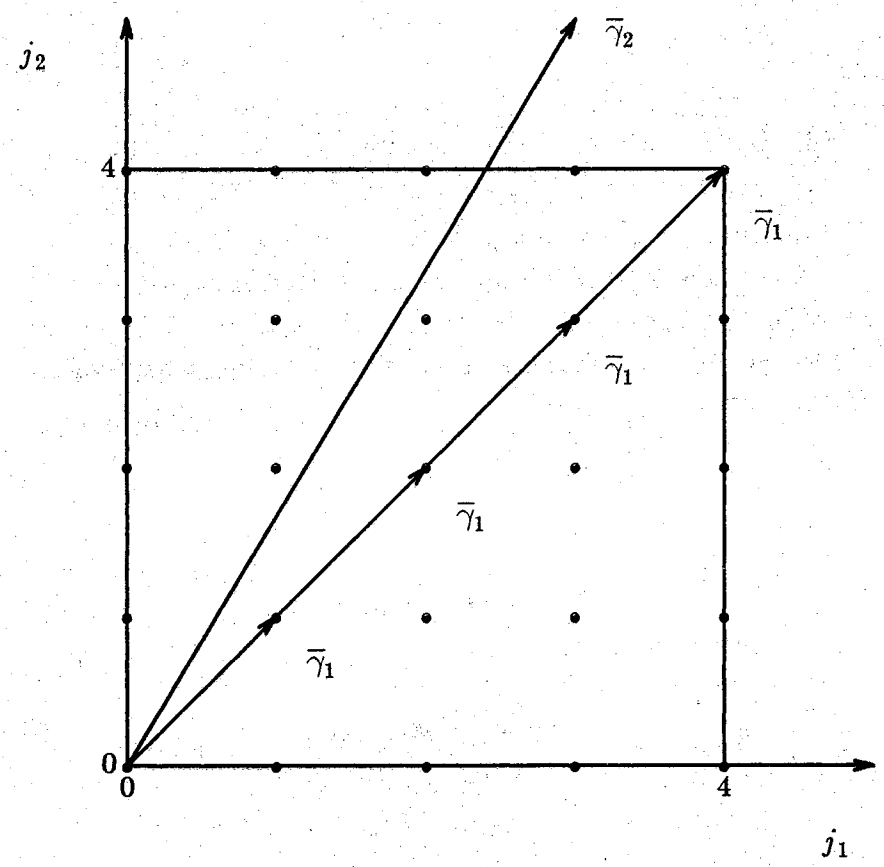


Figure 1 Non-feasible conflict vector $\overline{\gamma}_1$ and feasible conflict vector $\overline{\gamma}_2$. Vector $\overline{\gamma}_2$ does not meet any integral points inside the index set.

point of the index set), then the other end is out of the index set and vector $[3, 5]^T$ does not meet any integer points in the index set. Therefore, the mapping with this $\bar{\gamma}$ is conflict-free. To describe these concepts formally, the following definitions are introduced.

Definition 2.3 (Conflict vector, feasible and non-feasible conflict vectors and conflict-free mapping matrix): Given an algorithm (J, D) and a mapping matrix $T \in Z^{k \times n}$, an integral column vector $\bar{\gamma} = [\gamma_1, ..., \gamma_n]^T$ is a *conflict vector* of the mapping matrix T if and only if $T\bar{\gamma}=0$ and $gcd^{\dagger}(\gamma_1, ..., \gamma_n)=1$. If for any arbitrary index point $\bar{j} \in J$, $\bar{j}+\bar{\gamma} \notin J$, then $\bar{\gamma}$ is a *feasible conflict vector*. If there exists at least one index point $\bar{j} \in J$ such that $\bar{j}+\bar{\gamma} \in J$ then $\bar{\gamma}$ is called a *non-feasible conflict vector*. If all the conflict vectors are feasible, then this mapping matrix T is *conflict-free*.

Example 2.1: Consider a 4-dimensional algorithm (J, D) where

$$J=\{\bar{j}: \bar{j} \in Z^4, 0 \leq j_i \leq 6, i=1, ..., 4\} .$$

Assume that this algorithm is to be mapped into a 1-dimensional or linear processor array and one possible mapping matrix is

$$T=\begin{bmatrix} 1 & 7 & 1 & 1 \\ 1 & 7 & 1 & 0 \end{bmatrix}. \tag{2.8}$$

Consider the following solutions of $T\bar{\gamma}=\bar{0}$: $\bar{\gamma}_1=[0, 1, -7, 0]^T$, $\bar{\gamma}_2=[7, -1, 0, 0]^T$ and $\bar{\gamma}_3=[1, 0, -1, 0]^T$. Clearly, $T\bar{\gamma}_1 = T\bar{\gamma}_2 = T\bar{\gamma}_3=\bar{0}$ and their greatest common divisors of their entries are unity. So $\bar{\gamma}_1$, $\bar{\gamma}_2$ and $\bar{\gamma}_3$ are conflict vectors of mapping matrix T. However vector $[2, 0, -2, 0]^T$ is also a solution of equation $T\bar{\gamma}=\bar{0}$ but is not a conflict vector of mapping matrix T because the greatest common divisor of its entries is not unity. Conflict vectors $\bar{\gamma}_1$ and $\bar{\gamma}_2$ are feasible because it can be checked that for any arbitrary index point $\bar{j} \in J$, $\bar{j}+\bar{\gamma}_i \notin J$, $i=1, 2$. Conflict vector $\bar{\gamma}_3$ is not feasible because for the index point $\bar{j}=[0, 0, 1, 0]^T \in J$, $\bar{j}+\bar{\gamma}_3 = [1, 0, 0, 0]^T \in J$. Therefore, T is not conflict-free. □

Given an arbitrary algorithm, if it is possible to identify the set of all vectors $\bar{\gamma}$, whose entries are relatively prime and such that for any arbitrary index point $\bar{j} \in J$, $\bar{j}+\bar{\gamma} \notin J$, then the mapping matrix T can be constructed subject to that all its conflict vectors must be in that set. Unfortunately, it is not always easy to find the set of all such vectors. For algorithms with constant-bounded index sets, the common characteristics of such vectors are described in the following theorem.

---

†: $gcd(a_1, ..., a_n)$ denotes the greatest common divisor of integers $a_1, ..., a_n$.

Theorem 2.2: For algorithms with constant-bounded index sets defined by Equation 2.5, a mapping matrix T is conflict-free if and only if for each of its conflict vectors $\bar{\gamma}=[\gamma_1, ..., \gamma_i, ..., \gamma_n]^T$ there exists an entry γ_i such that $|\gamma_i| > \mu_i$.

Proof: (=>). Because T is conflict-free, all the conflict vectors of T are feasible. Now suppose that $\bar{\gamma}$ is a conflict vector of T and $|\gamma_i| \leq \mu_i$, $i=1, ..., n$. Consider the index point $\bar{j} = [j_1, ..., j_n]^T$ where $j_i=0$ if $\gamma_i \geq 0$ and $j_i=-\gamma_i$ if $\gamma_i<0$. It is clear that both $\bar{j}$ and $\bar{j}+\bar{\gamma}$ belong to the index set J defined by Equation 2.5 because $|\gamma_i| \leq \mu_i$, $i=1, ..., n$. By Definition 2.3, $\bar{\gamma}$ is not feasible which is contrary to the assumption. Therefore, for each of the conflict vectors $\bar{\gamma}$, there must exist an entry γ_i such that $|\gamma_i| > \mu_i$.
(<=). Let $\bar{\gamma}$ be a conflict vector of mapping matrix T and consider an arbitrary index point $\bar{j}$ belonging to the index set defined by Equation 2.5. Let $\bar{j'} = \bar{j}+\bar{\gamma} = [j'_1, ..., j'_n]^T$. Because there exists an entry γ_i of $\bar{\gamma}$ such that $|\gamma_i| > \mu_i$ and $\mu_i \geq j_i \geq 0$, $j'_i=j_i+\gamma_i>\mu_i$ if $\gamma_i>0$ and $j'_i=j_i+\gamma_i<0$ if $\gamma_i<0$. In both cases, $\bar{j'}$ is not in the index set J and $\bar{\gamma}$ is feasible. This implies that T is conflict-free. □

According to Theorem 2.2, for the algorithms with constant-bounded index set, the set of feasible conflict vectors is $\{\bar{\gamma}: |\gamma_i| > \mu_i, i \in \{1, ..., n\}, gcd(\gamma_1, ..., \gamma_n)=1\}$. This solution space is not convex and is very difficult to analyze.

In practice, it is interesting to find optimal conflict-free mappings with respect to different criteria and based on different assumptions. To achieve this, one has to identify first all feasible mappings that are conflict-free. Then it is possible to choose an optimal one with respect to a certain criterion from these conflict-free mappings. The criterion could be the total execution time for the algorithm, the VLSI area taken to implement this algorithm including the number of processors and the length of the wiring, or the combination of the total execution time and the VLSI area. Two problems are addressed in this paper and are formulated below. The first is about identifying all conflict-free mappings and the second is about finding time-optimal mappings.

Problem 2.1 (Conflict-free mapping problem): Given an n-dimensional uniform dependence algorithm and a $(k-1)$-dimensional processor array, find necessary and sufficient conditions for mapping matrix $T \in Z^{k \times n}$ to be conflict-free, or equivalently, identify all conflict-free mapping matrices $T \in Z^{k \times n}$.

Problem 2.2 (Time-optimal and conflict-free mapping problem): Given an n-dimensional uniform dependence algorithm (J, D) and a feasible space mapping matrix $S \in Z^{(k-1) \times n}$, find an integral row vector $\Pi^o \in Z^{1 \times n}$ which minimizes

$$f = \max \{ \Pi (\bar{j}_1 - \bar{j}_2) : \bar{j}_1, \bar{j}_2 \in J \}$$

$$\text{subject to} \begin{cases} \Pi D > \bar{0} \\ \sum_{j=1}^{r} k_{ji} \leq \Pi \bar{d}_i, \quad \text{where } SD=PK \\ rank(T)=k \\ T=\begin{bmatrix} S \\ \Pi \end{bmatrix} \text{ is conflict-free} \end{cases}$$

In Problem 2.2, the objective function f differs by one from the total execution time t in Equation 2.4. Clearly, f is minimized if and only if t is minimized. P and K are as defined in Definition 2.2. In general, in Problem 2.2, space mapping matrix S is given and usually is not a function of problem size variables μ_i $i=1, \ldots, n$. The solution of a special case of Problem 2.1 is discussed in Section 3, and the general case is discussed in Section 4 followed by the discussion of Problem 2.2.

3. NECESSARY AND SUFFICIENT CONDITIONS FOR CONFLICT-FREE MAPPING MATRIX $T \in Z^{(n-1)\times n}$

This section discusses the solution of Problem 2.1 or how to identify all conflict-free mapping matrices $T \in Z^{(n-1)\times n}$ that map n-dimensional algorithms into $(n-2)$-dimensional processor arrays. This simplest case can illustrate and give an intuitive understanding of different aspects of the conflict-free mapping problem. So the reader can follow the general discussion in the next section more easily. Practical applications are the mapping of 4-dimensional convolution algorithm at bit-level [26] into a 2-dimensional systolic array and the mapping of the 3-dimensional matrix multiplication algorithm into a linear systolic array [23].

Let $\Pi \in Z^{1\times n}$, $S \in Z^{(n-2)\times n}$ and $rank(S)=n-2$. Consider the following equation

$$T\bar{\gamma}=\bar{0} \quad or \quad \begin{bmatrix} S \\ \Pi \end{bmatrix}\bar{\gamma}=\bar{0} \; . \tag{3.1}$$

Let's first assume that $rank(T)=n-1$. Later in this section, conditions on Π are given to guarantee that $rank(T)=n-1$. Clearly, there is only one linearly independent solution of Equation 3.1. Without loss of generality, let $T=[B, \bar{b}]$ where B contains the first $n-1$ columns of matrix T, $rank(B)=n-1$ and $\bar{b}$ is the last column of T. Also, let B^* and $detB$ be the adjugate or adjoint matrix and determinant of matrix B, respectively [18, pp. 170]. Then all solutions of Equation 3.1 can be expressed as

$$\bar{\gamma}=\lambda\begin{bmatrix}-B^{*}\bar{b}\\ detB\end{bmatrix}=\lambda\begin{bmatrix}f_1(\pi_1, \ldots, \pi_n)\\ f_2(\pi_1, \ldots, \pi_n)\\ \cdots\\ f_n(\pi_1, \ldots, \pi_n)\end{bmatrix} \tag{3.2}$$

where λ is a constant.

If the first non-zero entry of a conflict vector is assumed to be positive (this implies no loss of generality), then for the mapping matrix $T\in Z^{(n-1)\times n}$, there is only one unique conflict vector (otherwise, $-\bar{\gamma}$ would also be a conflict vector). This unique conflict vector $\bar{\gamma}$ is expressed by Equation 3.2 where λ is such that $\bar{\gamma}$ is integral, its entries are relatively prime and the first non-zero entry is positive. According to Theorem 2.2 if this unique conflict vector is feasible, then the corresponding mapping is conflict-free. In addition, if Π is such that there exists a non-zero entry $f_i(\pi_1, \ldots, \pi_n)$, $1\leq i\leq n$, then $rank(T)=n-1$ because $f_i(\pi_1, \ldots, \pi_n)$ is the determinant of the submatrix of T consisting of all columns except the *ith* one of T. These facts are summarized in the following theorem.

Theorem 3.1 (Necessary and sufficient condition 1): Let $\bar{\gamma}$ be defined in Equation 3.2 where the constant λ in Equation 3.2 is such that $\bar{\gamma}$ is integral, its entries are relatively prime and the first non-zero entry is positive. Then mapping matrix $T\in Z^{(n-1)\times n}$ is feasible if and only if vector $\bar{\gamma}$ is feasible. The rank of matrix T is $n-1$ if and only if there exists a non-zero entry $f_i(\pi_1, \ldots, \pi_n)$, $1\leq i\leq n$.

Proof: First, it is shown as follows that there is only one conflict vector if the first non-zero entry of the conflict vector is assumed to be positive. Suppose there are two conflict vectors $\bar{\gamma}_1$ and $\bar{\gamma}_2$ whose first non-zero entries are positive. Because there is only one linearly independent solution of Equation 3.2, $\bar{\gamma}_1$ and $\bar{\gamma}_2$ are linearly dependent. Thus $\bar{\gamma}_2=c\bar{\gamma}_1$, where c is a constant. If $c=1$, then $\bar{\gamma}_1=\bar{\gamma}_2$; if $c=-1$, then the first entry of one of the vectors is not positive; if c is a non-integral rational number, then $\bar{\gamma}_2$ is non-integral because the greatest common divisor of entries of $\bar{\gamma}_1$ is one; and if $c>1$ is integral, the greatest common divisor of $\bar{\gamma}_2$ is greater than unity. Therefore, in all the cases discussed above, $\bar{\gamma}_2$ is not a distinct conflict vector whose first entry is positive. So there is only one such conflict vector of mapping matrix T and, therefore, T is feasible if and only if $\bar{\gamma}$ is feasible.

It is trivial to show that if $rank(T)=n-1$, then there is a non-zero entry $f_i(\pi_1, \ldots, \pi_n)$ because otherwise, $rank(T)=n$. Now suppose there exists a non-zero entry $f_i(\pi_1, \ldots, \pi_n)$. Let

$$B^* = \begin{bmatrix} B_{11} & B_{21} & \cdots & B_{(n-1),1} \\ B_{12} & B_{22} & \cdots & B_{(n-1),2} \\ \cdots & \cdots & \cdots & \cdots \\ B_{1,(n-1)} & B_{2,(n-1)} & \cdots & B_{(n-1),(n-1)} \end{bmatrix} \tag{3.3}$$

where B_{ij}, $i, j=1, \ldots, n-1$, are the cofactors of matrix B [18, pp. 165]. Clearly, $f_i = [B_{1i}, \ldots, B_{(n-1),i}]\bar{b} = B_{1i}s_{1,n} + \ldots + B_{(n-2),i}s_{(n-2),n} + B_{(n-1),i}\pi_n$, $1 \leq i \leq (n-1)$. With little thought, it can be seen that f_i is the determinant of matrix B with its *ith* column being replaced by $\bar{b}$ [18, pp. 165] which is a submatrix of T. Therefore, there is a submatrix of T whose determinant is non-zero which means $rank(T)=n-1$. □

According to Theorem 2.2, the unique conflict vector $\bar{\gamma}$ in Equation 3.2 is feasible if and only if the absolute value of one of its entries is greater than a certain value. Therefore, given a mapping matrix T, to see if it is conflict-free or not, Equation 3.2 has to be used. Later in Section 5, Equation 3.2 is used to formulate Problem 2.2 as an integer programming problem. If functions f_i in Equation 3.2, $i=1, \ldots, n$, are linear, then the formulation is possibly an integer linear programming problem. In the following, it is shown that if the space mapping matrix S is given, functions f_i in Equation 3.2, $i=1, \ldots, n$, are linear functions of π_j, $j=1, \ldots, n$.

Proposition 3.2: Functions f_i, $i=1, \ldots, n$ in Equation 3.2 are linear functions of π_j, $j=1, \ldots, n$.

Proof: Let B^* be defined as in Equation 3.3. Clearly, $f_i = [B_{1i}, \ldots, B_{(n-1),i}]\bar{b} = B_{1i}s_{1,n} + \ldots + B_{(n-2),i}s_{(n-2),n} + B_{(n-1),i}\pi_n$, $1 \leq i \leq (n-1)$. Cofactors B_{li}, $1 \leq l \leq n-2$, are the linear functions of π_j, $j=1, \ldots, n-1$ because B_{li} is the determinant of the submatrix of B obtained from removing the *lth* row and the *ith* column of matrix B. Thus, $B_{li}s_{l,n}$ are linear functions of π_j, $j=1, \ldots, n-1$. Cofactor $B_{(n-1),i}$ is independent of π_i, $i=1, \ldots, n$ because it is the determinant of the submatrix of B obtained by removing the *ith* column and the $(n-1)th$ row which is Π. Thus $B_{(n-1),i}b_{n-1} = B_{(n-1),i}\pi_n$ is a linear function of π_n. Therefore, f_i, $i=1, \ldots, n-1$ are linear functions of π_j, $j=1, \ldots, n$. The last entry $f_n=detB$ is also a linear function of π_j, $j=1, \ldots, n-1$ because it is the determinant of matrix B. □

Example 3.1: Consider algorithm (J, D) used in [23] where dependence matrix D and index set J are as follows.

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad J = \{\bar{j}: \bar{j} \in Z^3, 0 \leq j_i \leq \mu, i=1, \ldots, 3\} . \tag{3.4}$$

Actually, this uniform dependence algorithm can model the matrix multiplication algorithm. How the dependence matrix and the index set are derived from the Fortran code of the matrix multiplication algorithm is shown in [21], [23] and [32]. Let the matrix multiplication algorithm compute $C=AB$, where $C=[c_{ij}]$, $A=[a_{ij}]$ and $B=[b_{ij}]$. By [23], dependence vectors $\bar{d}_1$, $\bar{d}_2$ and $\bar{d}_3$ are induced by B, A and C, respectively. In other words, the computation indexed by $\bar{j}$ needs data A from index point $\bar{j}-\bar{d}_2$, B from index point $\bar{j}-\bar{d}_1$ and C from index point $\bar{j}-\bar{d}_3$. If the space mapping matrix is chosen as the one used in [23] $S=[1, 1, -1]$, then mapping matrix T and its conflict vector $\bar{\gamma}$ are as follows.

$$T=\begin{bmatrix} 1 & 1 & -1 \\ \pi_1 & \pi_2 & \pi_3 \end{bmatrix}, \quad \bar{\gamma}=\lambda \begin{bmatrix} -\pi_2-\pi_3 \\ \pi_1+\pi_3 \\ \pi_1-\pi_2 \end{bmatrix} \tag{3.5}$$

It is clear that $T\bar{\gamma}=\bar{0}$. If Π is chosen such that $-\pi_2-\pi_3\neq 0$ or $\pi_1+\pi_3\neq 0$ or $\pi_1-\pi_2\neq 0$, then $rank(T)=n-1=2$. □

Example 3.2: Consider another algorithm (J, D) used in [22] where dependence matrix D and index set J are as follows.

$$D=\begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & -1 & -1 & 0 \\ 1 & 0 & -1 & 0 & -1 \end{bmatrix} \quad J=\{\bar{j}: \bar{j}\in Z^3,\ 0\leq j_i\leq \mu,\ i=1, \ldots, 3\}. \tag{3.6}$$

This uniform dependence algorithm can model the reindexed transitive closure algorithm. How the dependence matrix and the index set are derived from the Fortran code of the transitive closure algorithm is shown in [17] and [23]. If the space mapping matrix is chosen as the one used in [22] $S=[0, 0, 1]$, then mapping matrix T and its conflict vector $\bar{\gamma}$ are as follows.

$$T=\begin{bmatrix} 0 & 0 & 1 \\ \pi_1 & \pi_2 & \pi_3 \end{bmatrix}, \quad \bar{\gamma}=\lambda \begin{bmatrix} \pi_2 \\ -\pi_1 \\ 0 \end{bmatrix} \tag{3.7}$$

It is clear that $T\bar{\gamma}=\bar{0}$. If Π is chosen such that $\pi_2\neq 0$ or $\pi_1\neq 0$ then $rank(T)=n-1=2$. □

4. GENERAL CASE--NECESSARY AND SUFFICIENT CONDITIONS FOR CONFLICT-FREE MAPPINGS

This section discusses and presents the solution to the general case of Problem 2.1, i.e., it provides necessary and sufficient conditions for conflict-free mappings where n-dimensional algorithms are mapped into $(k-1)$-dimensional processor arrays. In these mappings, $T\in Z^{k\times n}$, $T=\begin{bmatrix} S \\ \Pi \end{bmatrix}$, $\Pi\in Z^{1\times n}$ and $S\in Z^{(k-1)\times n}$.

Consider the equation

$$T\bar{\gamma}=\bar{0} \qquad or \qquad \begin{bmatrix} S \\ \Pi \end{bmatrix}\bar{\gamma}=\bar{0} \tag{4.1}$$

If $rank(T)=k$, then there are $n-k$ linearly independent solutions of Equation 4.1. Let $\bar{\gamma}_1$, ..., $\bar{\gamma}_{n-k}$ be the linearly independent integral solutions of Equation 4.1, whose entries are relatively prime, then all solutions $\bar{\gamma}$ of Equation 4.1 can be represented as linear combinations of the $n-k$ linear independent vectors as follows

$$\bar{\gamma}=\lambda_1\bar{\gamma}_1+\cdots+\lambda_{n-k}\bar{\gamma}_{n-k} \tag{4.2}$$

Clearly, $\bar{\gamma}_1$, ..., $\bar{\gamma}_{n-k}$ are conflict vectors of T.

In general, the mapping matrix T has more than $n-k$ conflict vectors when $k<n-1$ because a linear combination of these $n-k$ conflict vectors may be a different integral vector whose entries are relatively prime and therefore another conflict vector of T. This new conflict vector may or may not be feasible. Thus, unlike the mapping matrix $T\in Z^{(n-1)\times n}$ described in Section 3, it is not guaranteed that all conflict vectors of T are feasible even if the $n-k$ linearly independent solutions $\bar{\gamma}_i$, $i=1, ..., n-k$, of equation $T\bar{\gamma}=\bar{0}$ are all feasible. This is illustrated by the following example.

Example 4.1: Consider the 4-dimensional algorithm of Example 2.1 and the mapping matrix T in Equation 2.8. Let $\bar{\gamma}_1=[0, 1, -7, 0]^T$ and $\bar{\gamma}_2=[7, -1, 0, 0]^T$. Clearly, $T\bar{\gamma}_1 = T\bar{\gamma}_2=\bar{0}$, $\bar{\gamma}_1$ and $\bar{\gamma}_2$ are linearly independent, and they are feasible conflict vectors of T. Let $\bar{\gamma} = 1/7\bar{\gamma}_1 + 1/7\bar{\gamma}_2 = [1, 0, -1, 0]^T$. Vector $\bar{\gamma}$ is also a solution of equation $T\bar{\gamma}=\bar{0}$ and its entries are relatively prime. By Definition 2.3 and Theorem 2.2, $\bar{\gamma}$ is a non-feasible conflict vector of T. Therefore, as mentioned above, for a given mapping matrix $T\in Z^{k\times n}$ with $k<n-1$, there are possibly more than $n-k$ conflict vectors, and T may not be conflict-free even if there are $n-k$ linearly independent feasible conflict vectors of T. □

From Example 4.1, an interesting observation is that one difficulty in making all conflict vectors of mapping matrix T feasible is that non-feasible conflict vectors can result from rational linear combinations of the $n-k$ linearly independent feasible conflict vectors $\bar{\gamma}_1$, ..., $\bar{\gamma}_{n-k}$ like $\bar{\gamma} = 1/7\bar{\gamma}_1 + 1/7\bar{\gamma}_2$ in Example 4.1. Let's consider another way to select the $n-k$ linearly independent conflict vectors of T such that constants λ_i, $i=1, ..., n-k$ in Equation 4.2 must be integral in order for $\bar{\gamma}$ to be integral. To achieve this, the notion of the Hermite normal form is introduced.

Theorem 4.1 (Hermite normal form [29, pp. 45]): Let $T\in Z^{k\times n}$ and $rank(T)=k$. Then there exists a unimodular† matrix $U\in Z^{n\times n}$ such that $TU=H=[L, \mathbf{0}]$ ($\mathbf{0}$ denotes a zero-entry matrix) where $L\in Z^{k\times k}$ is a nonsingular and lower

triangular matrix. Matrix H is called the *Hermite normal form* of T.

The definition of the Hermite normal form used here is slightly different from the one used conventionally and in [29], where each diagonal element of matrix L is required to be positive and be the maximum of all absolute values of elements in that same row. This is because for the purpose of this paper it is enough to know that matrix T can be transformed into a lower triangular matrix $[L, \mathbf{0}]$ by right multiplication of a unimodular matrix U.

For a given mapping matrix T, let H be the corresponding Hermite normal form and $T=HV$ where $V=U^{-1}$, $U=[\bar{u}_1, \ldots, \bar{u}_n]$ and $V=[\bar{v}_1, \ldots, \bar{v}_n]$. Then Equation 4.1 can be rewritten as $HV\bar{\gamma}=\bar{0}$. Let $\bar{\beta}=V\bar{\gamma} = [\beta_1, \ldots, \beta_n]^T$ and $\bar{\gamma}=U\bar{\beta}$. Then the following statements are true.

Theorem 4.2:

(1) $H\bar{\beta}=\bar{0}$ if and only if $\beta_1, \ldots, \beta_k$ are zero.

(2) Vector $\bar{\gamma}$ is integral if and only if $\bar{\beta}$ is integral.

(3) Vector $\bar{\gamma}$ is a conflict vector of mapping matrix T if and only if

$$\bar{\gamma}=\left[\bar{u}_{k+1}, \ldots, \bar{u}_n\right]\begin{bmatrix}\beta_{k+1}\\ \cdots \\ \beta_n\end{bmatrix} \tag{4.3}$$

where β_i, $i=k+1, \ldots, n$, are arbitrary integers which are not all zero and are relatively prime.

Proof: (1) Because $H=[L, \mathbf{0}]$ and $H\bar{\beta} = L[\beta_1, \ldots, \beta_k]^T$ where $L\in Z^{k\times k}$ is a non-singular lower triangular matrix, $\beta_1, \ldots, \beta_k$ are zero if and only if $H\bar{\beta}=\bar{0}$.

(2) By definition, a matrix is unimodular if and only if it is integral and the absolute value of its determinant is unity. So U is unimodular means that matrix $V=U^{-1}$ is also unimodular. Therefore, $\bar{\gamma}$ is integral implies that $\bar{\beta}$ is integral and vice versa.

(3) By Theorem 4.2 (1) and (2), all integral solutions $\bar{\gamma}$ of equation $T\bar{\gamma}=\bar{0}$ are represented by Equation 4.3 where β_i, $i=k+1, \ldots, n$, have to be arbitrary integers because non-integral values of β_i, $i=k+1, \ldots, n$ result in a non-integral vector $\bar{\gamma}$. Next it is shown that the greatest common divisor of β_i, $i=1, \ldots, n$ is unity if and only if the greatest common divisor of γ_i, $i=1, \ldots, n$ is unity. Suppose $gcd(\beta_1, \ldots, \beta_n)=1$ and $gcd(\gamma_1, \ldots, \gamma_n)=c>1$. Then $\bar{\gamma}=c\bar{\gamma}'$ where $\bar{\gamma}'$ is integral and its entries are relatively prime. Because $\bar{\beta}=V\bar{\gamma} = cV\bar{\gamma}'$ where, obviously $V\bar{\gamma}'\in Z^n$, the greatest common divisor of β_i, $i=1, \ldots, n$ is at least $c>1$. This is contrary to the assumption. So, the greatest common divisor of β_i, $i=1, \ldots, n$ is unity implies the

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

greatest common divisor of γ_i, $i=1, \ldots, n$ is unity. With similar reasoning, the reverse can be shown. Therefore, the β_i, $i=k+1, \ldots, n$, in Equation 4.3 have to be relatively prime integers, otherwise the greatest common divisor of entries of vector $\bar{\gamma}$ is greater than one, which is not a conflict vector by Definition 2.3. □

What Theorem 4.2 implies is that all conflict vectors of mapping matrix T can be represented by Equation 4.3 where $\beta_{k+1}, \ldots, \beta_n$ are arbitrary integers which are not all zero and relatively prime. Notice that a non-integral value of any one of the $\beta_{k+1}, \ldots, \beta_n$ results in a non-integral vector $\bar{\gamma}$ according to Theorem 4.2. So in this representation, the case where a new conflict vector of T can be obtained by a non-integral linear combination of the $n-k$ linearly independent solutions of Equation 4.1 is avoided.

Example 4.2: The Hermite normal form of the mapping matrix T in Equation 2.8 is

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

where

$$U=\begin{bmatrix}1 & -1 & -1 & -7\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\end{bmatrix} \quad \text{and} \quad V=U^{-1}=\begin{bmatrix}1 & 7 & 1 & 1\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}$$

All conflict vectors of T are the integral combinations of the third and fourth columns of matrix U as follows:

$$\bar{\gamma}=\begin{bmatrix}-1 & -7\\ 0 & 1\\ 1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\beta_3\\ \beta_4\end{bmatrix},$$

where β_3 and β_4 are integers which are not all zero and relatively prime. □

So far, a better representation of all conflict vectors of T has been found which requires integral combinations of $n-k$ linearly independent conflict vectors of mapping matrix T. However, it is still not guaranteed that all conflict vectors are feasible unless matrix U satisfies some conditions. The following six theorems describe these necessary and sufficient conditions for mapping matrix T to be conflict-free.

Theorem 4.3 (Necessary condition 2): Let v_{ij} be the entry of matrix V at the *ith* row and the *jth* column. If mapping matrix T is conflict-free, then at least one

of the first k entries of each and every column of V must be non-zero, that is, the following condition holds.

$$\begin{aligned}
&(v_{11}\neq 0 \vee v_{21}\neq 0 \vee \ldots \vee v_{k1}\neq 0)\wedge \\
&(v_{12}\neq 0 \vee v_{22}\neq 0 \vee \ldots \vee v_{k2}\neq 0)\wedge \\
&\ldots \\
&(v_{1n}\neq 0 \vee v_{2n}\neq 0 \vee \ldots \vee v_{kn}\neq 0)
\end{aligned} \tag{4.4}$$

Proof: Let $\overline{\gamma}$ be an arbitrary conflict vector of mapping matrix T. If T is conflict-free, then $\overline{\gamma}$ is feasible and it has at least two non-zero entries $\gamma_i\neq 0$ and $\gamma_j\neq 0$. Next, it is shown that $\overline{\gamma}$ has two non-zero entries if and only if Equation 4.4 holds.
($\Leftarrow$). Suppose that $\overline{\gamma}$ has only one non-zero entry, then $\overline{\gamma} = [0, \ldots, 0, 1, 0, \ldots, 0]^T$. By assumption, $\overline{\beta} = V\overline{\gamma} = \overline{v}_i$, the *ith* column of matrix V. According to Equation 4.4, there exists a non-zero element $v_{li}\in\{v_{1i}, \ldots, v_{ki}\}$ which means $\beta_l\neq 0$, $1\leq l\leq k$. This is contrary to Theorem 4.2 (1) that $\beta_i=0$, $i=1, \ldots, k$. Therefore, $\overline{\gamma}$ has at least two non-zero entries.
($\Rightarrow$). Suppose there exists a column $\overline{v}_i$ of matrix V whose first k entries are all zero, then $\overline{v}_i = [0, \ldots, 0, v_{k+1,i}, \ldots, v_{ni}]^T$. Let $\overline{\beta}=\overline{v}_i$. Because the first k entries of $\overline{v}_i$ are zero, $H\overline{\beta}=\overline{0}$ and $\overline{\gamma}=V^{-1}\overline{\beta}$ is a conflict vector of mapping matrix T. However, $\overline{\gamma}=V^{-1}\overline{v}_i = [0, \ldots, 0, 1, 0, \ldots, 0]^T$ whose entries are all zero except that the *ith* entry is unity. So, mapping matrix T has a conflict vector with only one non-zero entry which is contrary to the assumption. This means Equation 4.4 holds. □

Theorem 4.4 (Necessary condition 3): If mapping matrix T is feasible, then $\overline{u}_{k+1}, \ldots, \overline{u}_n$ are feasible conflict vectors.

Proof: Mapping matrix T is feasible implies that all conflict vectors $\overline{\gamma}$ of T are feasible. Such a conflict vector $\overline{\gamma}$ is represented by Equation 4.3 where $\beta_{k+1}, \ldots, \beta_n$ are arbitrary integers which are relatively prime and not all zero. Let $\overline{\beta}$ be a vector whose entries are all zero except that the *ith* entry, $k+1\leq i\leq n$, is unity. Then the corresponding conflict vector $\overline{\gamma} = \overline{u}_i$, $i=k+1, \ldots, n$. Therefore, $\overline{u}_i$ is a conflict vector and must be feasible because T is feasible by assumption. □

Theorem 4.5 (Sufficient condition 4): Mapping matrix T is conflict-free if the following conditions are met.

(1) There exist $i_1, \ldots, i_{n-k}\in\{1, \ldots, n\}$ such that

$$gcd(u_{i_1,k+1},\ u_{i_1,k+2},\ \cdots,\ u_{i_1,n})>\mu_{i_1}$$
$$gcd(u_{i_2,k+1},\ u_{i_2,k+2},\ \cdots,\ u_{i_2,n})>\mu_{i_2}$$
$$\cdots$$
$$gcd(u_{i_{n-k},k+1},\ u_{i_{n-k},k+2},\ \cdots,\ u_{i_{n-k},n})>\mu_{i_{n-k}}$$

$$(2)\qquad \det\begin{bmatrix} u_{i_1,k+1} & u_{i_1,k+2} & \cdots & u_{i_1,n} \\ u_{i_2,k+1} & u_{i_2,k+2} & \cdots & u_{i_2,n} \\ \cdots & \cdots & \cdots & \cdots \\ u_{i_{n-k},k+1} & u_{i_{n-k},k+2} & \cdots & u_{i_{n-k},n} \end{bmatrix} \neq 0$$

Proof: Let $\bar{\gamma}$ be an arbitrary conflict vector of mapping matrix T represented by Equation 4.3 where β_{k+1}, ..., β_n are arbitrary integers which are not all zero and relatively prime. Because

$$\det\begin{bmatrix} u_{i_1,k+1} & u_{i_1,k+2} & \cdots & u_{i_1,n} \\ u_{i_2,k+1} & u_{i_2,k+2} & \cdots & u_{i_2,n} \\ \cdots & \cdots & \cdots & \cdots \\ u_{i_{n-k},k+1} & u_{i_{n-k},k+2} & \cdots & u_{i_{n-k},n} \end{bmatrix} \neq 0$$

and β_{k+1}, ..., β_n are not all zero, there exists $i\in\{i_1, \ldots, i_{n-k}\}$ such that

$$\begin{bmatrix} u_{i,k+1},\ u_{i,k+2},\ \cdots,\ u_{i,n} \end{bmatrix}\begin{bmatrix} \beta_{k+1} \\ \beta_{k+2} \\ \cdots \\ \beta_n \end{bmatrix} = \gamma_i \neq 0\,. \qquad (4.5)$$

According to condition (1), $gcd(u_{i,k+1}, \ldots, u_{i,n}) = \alpha_i \geq \mu_i+1$. If $|\gamma_i| < \mu_i+1$, then α_i does not divide γ_i which means, according to [27], Equation 4.5 has no integral solution and β_{k+1}, ..., β_n are not all integral. Thus $\bar{\gamma}$ is not integral and not a conflict vector. Therefore, it must be $|\gamma_i| \geq \mu_i+1$. By Definition 2.3 and Theorem 2.2, $\bar{\gamma}$ is feasible and T is conflict-free. □

Theorem 4.6 (Sufficient condition 5 for $T\in Z^{(n-2)\times n}$): Mapping matrix $T\in Z^{(n-2)\times n}$ is conflict-free if the following conditions are met.

(1) There exists $i\in\{1, \ldots, n\}$ such that $gcd(u_{i,n-1}, u_{i,n}) \geq \mu_i+1$.

(2) Let β_{n-1} and β_n be relatively prime integers, not both zero and such that $\beta_{n-1}u_{i,n-1} + \beta_n u_{i,n}=0$, there exists $j\in\{1, \ldots, n\}$, $j\neq i$, such that $|\beta_{n-1}u_{j,n-1} + \beta_n u_{j,n}| > \mu_j$.

Proof: Consider all integral values for β_{n-1} and β_n which are not both zero, relatively prime and $\beta_{n-1}u_{i,n-1} + \beta_n u_{i,n}\neq 0$. Let the corresponding conflict vectors be $\bar{\gamma}$. Because $\gamma_i\neq 0$ and $gcd(u_{i,n-1}, u_{i,n})=\alpha_i \geq \mu_i+1$, $|\gamma_i| \geq \mu_i+1$. Otherwise, α_i does

not divide γ_i, and equation $\beta_{n-1}u_{i,n-1} + \beta_n u_{i,n} = \gamma_i$ has no integral solution [27]. Therefore, for these values of β_{n-1} and β_n, the corresponding conflict vectors are feasible. Now let's consider the integral values for β_{n-1} and β_n which are not both zero, relatively prime and $\beta_{n-1}u_{i,n-1} + \beta_n u_{i,n}=0$. For the corresponding conflict vector $\bar{\gamma}$, because there exists $j\in\{1, ..., n\}$, $j\neq i$, such that $|\beta_{n-1}u_{j,n-1} + \beta_n u_{j,n}| > \mu_j$, $|\gamma_j|$ is greater than μ_j and $\bar{\gamma}$ is feasible. Therefore, mapping matrix T is conflict-free because all of its conflict vectors are feasible. □

Theorem 4.7 (Necessary and sufficient condition 6 for $T\in Z^{(n-2)\times n}$): Mapping matrix $T\in Z^{(n-2)\times n}$ is conflict-free if and only if the following conditions are met.

(1) There exists $i\in\{1, ..., n\}$ such that $u_{i,n-1}\cdot u_{i,n}\geq 0$ and $|u_{i,n-1} + u_{i,n}| > \mu_i$.

(2) There exists $j\in\{1, ..., n\}$ such that $u_{j,n-1}\cdot u_{j,n}\leq 0$ and $|u_{j,n-1} - u_{j,n}| > \mu_j$.

(3) $\bar{u}_{n-1}$ and $\bar{u}_n$ are feasible conflict vectors.

Proof: Because $T\in Z^{(n-2)\times n}$, its conflict vectors are described by Equation 4.3 where $k=n-2$, β_{n-1} and β_n are arbitrary integers that are not both zero and relatively prime.

(<=): Suppose conditions (1), (2) and (3) are met. Let's consider the case where one of β_{n-1} and β_n is zero. Then the corresponding conflict vector $\bar{\gamma}$ equals either $\bar{u}_{n-1}$ or $\bar{u}_n$ and is feasible in either case because of condition (3). Let's consider the second case where $\beta_{n-1}\neq 0$ and $\beta_n\neq 0$ have the same sign and the corresponding conflict vector $\bar{\gamma}$. By condition (1), there exists $i\in\{1, ..., n\}$ such that $u_{i,n-1}$, $u_{i,n}$ have the same sign and $|u_{i,n-1} + u_{i,n}| > \mu_i$. Because u_{n-1} has the same sign as u_n and β_{n-1} has the same value as β_n, $|\bar{\gamma}_i| = |\beta_{n-1}u_{i,n-1} + \beta_n u_{i,n}| = |\beta_{n-1}u_{i,n-1}| + |\beta_n u_{i,n}| \geq |u_{i,n-1} + u_{i,n}| > \mu_i$. Therefore, $\bar{\gamma}$ is feasible. Now let's consider the third case where $\beta_{n-1}\neq 0$ and $\beta_n\neq 0$ have opposite signs and the corresponding conflict vector $\bar{\gamma}$. According to condition (2), there exists $j\in\{1, ..., n\}$ such that $u_{j,n-1}$, $u_{j,n}$ have opposite signs and $|u_{j,n-1} - u_{j,n}| > \mu_j$. Because β_{n-1} and β_n have different signs, $|\gamma_j| = |\beta_{n-1}u_{j,n-1} + \beta_n u_{j,n}| \geq |u_{j,n-1} - u_{j,n}| > \mu_j$. Therefore, $\bar{\gamma}$ is feasible.

(=>): Suppose T is conflict-free and therefore all its conflict vectors are feasible. Suppose conditions 1 in Theorem 4.7 does not hold. Thus, there does not exist $i\in\{1, ..., n\}$ such that $u_{i,n-1}$, $u_{i,n}$ have the same sign and $|u_{i,n-1} + u_{i,n}| > \mu_i$. Let $\beta_{n-1}=1$ and $\beta_n = 1$ and consider the conflict vector $\bar{\gamma} = \beta_{n-1}\bar{u}_{n-1} + \beta_n\bar{u}_n = u_{n-1}+u_n$. Because condition (1) does not hold, $|\gamma_i| \leq \mu_i$, $i=1, ..., n$ which means $\bar{\gamma}$ is not feasible and T is not conflict-free. Similarly, it can be shown that condition 2 holds. By Theorem 4.4, condition 3 holds. □

Theorem 4.7 provides necessary and sufficient conditions for mapping matrices $T\in Z^{(n-2)\times n}$ which can be applied to map 5-dimensional algorithms into 2-dimensional processor arrays and can be used to handle most frequently

occurring mappings in RAB [26] such as the mappings of a bit level matrix multiplication algorithm and a bit level LU decomposition algorithm. The reasoning used in Theorem 4.7 can also be used to derive similar necessary and sufficient conditions for mapping matrices $T \in Z^{(n-3)\times n}$, $T \in Z^{(n-4)\times n}$ and so on. These conditions are stated as follows for mapping matrices $T \in Z^{(n-3)\times n}$ without proofs, which is similar as the proof of Theorem 4.7.

Theorem 4.8 (Necessary and sufficient condition 7 for $T \in Z^{(n-3)\times n}$): Let the sign of the number zero be defined as either positive or negative. Then mapping matrix $T \in Z^{(n-3)\times n}$ is conflict-free if and only if the following conditions are met.

(1) There exists $i1 \in \{1, ..., n\}$ such that $u_{i1,n-2}$, $u_{i1,n-1}$ and $u_{i1,n}$ have the same sign and $| u_{i1,n-2} + u_{i1,n-1} + u_{i1,n} | > \mu_{i1}$.

(2) There exists $i2 \in \{1, ..., n\}$ such that $u_{i2,n-2}$ and $u_{i2,n-1}$ have the same sign and $u_{i2,n}$ has an opposite sign from $u_{i2,n-2}$ and $u_{i2,n-1}$. Also, $| u_{i2,n-2} + u_{i2,n-1} - u_{i2,n} | > \mu_{i2}$.

(3) There exists $i3 \in \{1, ..., n\}$ such that $u_{i3,n-2}$ and $u_{i3,n}$ have the same sign and $u_{i3,n-1}$ has an opposite sign from $u_{i3,n-2}$ and $u_{i3,n}$. Also, $| u_{i3,n-2} - u_{i3,n-1} + u_{i3,n} | > \mu_{i3}$.

(4) There exists $i4 \in \{1, ..., n\}$ such that $u_{i4,n-1}$ and $u_{i4,n}$ have the same sign and $u_{i4,n-2}$ has an opposite sign from $u_{i4,n-1}$ and $u_{i4,n}$. Also, $| -u_{i4,n-2} + u_{i4,n-1} + u_{i4,n} | > \mu_{i4}$.

(5) $\bar{u}_{n-2}$, $\bar{u}_{n-1}$ and $\bar{u}_n$ are feasible conflict vectors. □

From the discussion above, it is clear that the multiplier matrix U is involved in most of the necessary and sufficient conditions for conflict-free mappings and the key point is to have matrix U satisfy certain conditions in order to make the mapping matrix T conflict-free. Given a mapping matrix T in numbers, there are several numerical methods to compute matrix U in polynomial time [20]. When the entries of T are symbols (variables), computing matrix U becomes more difficult. However, when the space mapping matrix S is known, it is possible to express matrix U as a function of Π. Proposition 8.1 in the appendix computes matrix U as a function of Π for the mapping matrix $T \in Z^{3\times 5}$.

5. TIME-OPTIMAL CONFLICT-FREE MAPPINGS

Solutions to Problem 2.1 have been discussed in the above sections. In this section, Problem 2.2 and its solutions are discussed: that is, how to find the optimal vector Π^o which contributes a conflict-free mapping matrix T and satisfies some other constraints. Two approaches are proposed, one of which is an intelligent search of the solution space by employing the basic strategy in [11] and the other is to formulate Problem 2.2 as an integer programming problem where the solution can be obtained by using existing standard algorithms for integer programming problems.

By Theorem 2.1, the total execution time increases monotonically with the sum of the weighted absolute values of entries of vector Π. Therefore, to find the solution of Problem 2.2, one simple way is to modify the method in [11] where candidates are enumerated in an increasing order of their total execution times. In the modified method, besides other considerations, each candidate should be checked to see if it contributes a conflict-free mapping. This method is described in the following procedure.

Procedure 5.1 (Finding optimal Π^o of Problem 2.2):

Input: Algorithm (J, D) and the space mapping $S \in Z^{(k-1)\times n}$.

Output: An integral row vector Π^o which is the solution of Problem 2.2.

Step 1: Set $l=1$. $C_0=\varnothing$

Step 2: Construct a candidate set $C_l=\{\Pi: \sum_{i=1}^{n} |\pi_i| \mu_i \leq x_l \in N^+\}$. $C=C_l - C_l \cap C_{l-1}$. Let $|C|=c$.

Step 3: Sort and assign indices to all candidates in C such that $t(\Pi_i) \leq t(\Pi_j)$, $0<i<j\leq c$, where $t(\Pi)$ is the function described in Equation 2.7.

Step 4: Set $i=1$.

Step 5: Consider candidate Π_i; set $T^i=\begin{bmatrix} S \\ \Pi_i \end{bmatrix}$ and check if Π_i meets the following conditions

(1) $\Pi D > \bar{0}$.

(2) $rank(T^i)=k$.

(3) T^i is conflict free. If $T^i \in Z^{(n-1)\times n}$, test the condition in Theorem 3.1; if $T^i \in Z^{(n-2)\times n}$, check the conditions in Theorem 4.7; if $T^i \in Z^{(n-3)\times n}$, check the conditions in Theorem 4.8 and otherwise, check the conditions in Theorem 4.5.

(4) $SD=PK$ and $\sum_{j=1}^{r} k_{ji} \leq \Pi \bar{d}_i$, $i=1, \ldots, m$. where P and K are as defined in Definition 2.2.

Step 6: If Π_i meets all the three conditions above, then $\Pi^o=\Pi_i$ and stop. If Π_i does not meet any one of the conditions above, ignore this candidate, set $i=i+1$.

Step 7: If $i>c$, set $l=l+1$, $x_l=x_l+\alpha$, $\alpha \in N^+$ and go to Step 2. If $i\leq c$, go to Step 5.

Clearly, for mapping matrices $T \in Z^{(n-1)\times n}$, $T \in Z^{(n-2)\times n}$ and $T \in Z^{(n-3)\times n}$, Procedure 5.1 finds the optimal solution because in Step 5 (3) it uses the necessary and sufficient conditions in Theorems 3.1, 4.7 and 4.8. An upper bound on the computational complexity of this procedure is $O(n^{2\mu+1})$ where $\mu=\min\{\mu_i: i=1, \ldots, n\}$. This is because the search starts at $\Pi=\bar{0}$ and the possible values for π_i, $i=1, \ldots, n$ are $0, \pm 1, \ldots, \pm\mu_i$. Obviously, the candidate Π where the absolute

values of π_i, $i=1, \ldots, n$, are small does not contribute a conflict-free mapping. So more sophisticated methods of finding the solution of Problem 2.2 may be possible. In general, together with Theorem 2.1, these necessary and sufficient conditions described in Theorems 3.1, 4.5, 4.6, 4.7 and 4.8, depending on the dimension of the mapping matrix T, should be used to guide the solution search which is under investigation. Usually, Procedure 5.1 is the last choice for finding the optimal solution. In the following, a much more preferable approach, integer linear programming approach is discussed.

For the mapping matrix $T \in Z^{(n-1)\times n}$, Problem 2.2 can be formulated as an integer programming problem as follows.

$$\min f = \sum_{i=1}^{n} |\pi_i| \mu_i \tag{5.1}$$

$$\textit{subject to} \begin{cases} (1)\ \Pi D > \bar{0} \\ (2)\ SD = PK \text{ and } \sum_{j=1}^{r} k_{ji} \leq \Pi \bar{d}_i,\ \ i=1, \ldots, m \\ (3)\ \exists i \quad |f_i(\pi_1, \ldots, \pi_n)| > \mu_i \\ (4)\ \Pi \in Z^{1\times n} \end{cases} \tag{5.2}$$

where $T = \begin{bmatrix} S \\ \Pi \end{bmatrix}$, S and P are given and f_i, $i=1, \ldots, n$ are as defined in Equation 3.2. The last two constraints required by Definition 2.2 are included implicitly in (5.2) because by Theorems 2.2 and 3.1 they are implied by constraint 3 in (5.2). Also, constraint 2 is not required if a new processor array is designed specially for the algorithm. By Proposition 3.2, constraint 3 in (5.2) is linear. So the formulation in (5.1) and (5.2) is an integer linear programming problem if, as in Examples 5.1 and 5.2, constraint 1 in (5.2) requires that $\pi_i > 0$, $i=1, \ldots, n$. Actually, this integer linear programming problem can be further converted to a linear programming problem for some applications as indicated by Examples 5.1 and 5.2. As a matter of fact, if π_i, $i=1, \ldots, n$ are required to be positive (usually by constraint 1 in (5.2)) and all entries of dependence matrix D and space mapping matrix S take values 0, 1 or -1, i.e., s_i, $d_{ij} \in \{-1, 0, 1\}$, $i=1, \ldots, n$ and $j=1, \ldots, m$, then the integer programming problem defined by (5.1) and (5.2) can always be converted to several linear programming problems. This is because in this case, the objective function is linear, the coefficients of π_i, $i=1, \ldots, n$, in the constraints of the integer linear programming problem in (5.1) and (5.2) are 1, 0 or -1, and therefore all extreme points of the solution sets are integral.

The general integer programming problem is NP-complete [29, pp. 245]. However, for each fixed natural number n, there exists a polynomial time algorithm which solves the integer linear programming problem [29, pp. 259]. This algorithm is a polynomial function of the number of constraints of the integer

linear programming problem and a logarithmic function of the problem size variables μ_i, $i=1, \ldots, n$. Many existing standard algorithms can be used to solve the formulation in (5.1) and (5.2) efficiently because in practice, the number of constraints and the algorithm dimension n are not large.

Example 5.1: Consider the matrix multiplication algorithm in Example 3.1 where the space mapping matrix is given as $S=[1, 1, -1]$. The dependence matrix D and index set J are shown in Equation 3.4. To satisfy condition 1 in Equation 5.2, each entry of the linear schedule vector Π must be positive or $\pi_i \geq 1$, $i=1, \ldots, 3$. Therefore, the problem of finding an optimal linear schedule vector for the matrix multiplication algorithm is formulated as an integer linear programming problem:

$$\min f=\mu(\pi_1+\pi_2+\pi_3)$$

$$\textit{subject to} \begin{cases} (1)\ \pi_i \geq 1, \quad i=1, 2, 3 \\ (2)\ SD=PK \text{ and } \sum_{j=1}^{r} k_{ji} \leq \Pi\, \bar{d}_i, \quad i=1, \ldots, 3 \\ (3)\ \pi_2+\pi_3 \geq \mu+1, \ \textit{or}\ \pi_1+\pi_3 \geq \mu+1, \ \textit{or}\ |\pi_1-\pi_2| \geq \mu+1 \\ (4)\ \Pi \in Z^{1\times 3} \end{cases} \tag{5.3}$$

where the inequalities in constraint 3 are derived in Example 3.1 and shown in Equation 3.5. As indicated in the appendix, this problem can be converted to three linear programming problems because all extreme points of the convex solution space are integral. In the appendix, techniques for solving linear programming problems are used to find the optimal solution. If the problem size variable $\mu=4$, then the optimal solution of this algorithm is either $\Pi_2=[1, \mu, 1]=[1, 4, 1]$ or $\Pi_3=[\mu, 1, 1]$. The derivation of the solution is shown in the appendix. Let's choose $\Pi^o=\Pi_2=[1, \mu, 1]$. The total execution time is $t=\mu(2+\mu)+1$ according to Equation 2.7. The matrix of interconnection primitives is $P=S=[1, 1, -1]$ and $K=I$, the identity matrix. Figure 2 shows the block diagram of the linear array and Figure 3 shows the execution of the matrix multiplication algorithm by mapping matrix $T=\begin{bmatrix} 1 & 1 & -1 \\ 1 & 4 & 1 \end{bmatrix}$. Computation $c_{j1,j2}=a_{j1,j3}\cdot b_{j3,j2}$ indexed by $\bar{j}=[j_1, j_2, j_3]^T$ is executed at processor $S\bar{j}$ and at time $\Pi\bar{j}$. Clearly, there are no computational conflicts. Also, as shown in Figure 3 and explained in the appendix, there is no link collision either which means no two data use the same link at the same time. As shown in Figure 2, three data links are used, one for data A traveling from left to right, one for data B traveling from left to right and one for data C traveling from right to left. Three buffers are needed for each processing element on the link for data A or for dependence vector $\bar{d}_2$. □

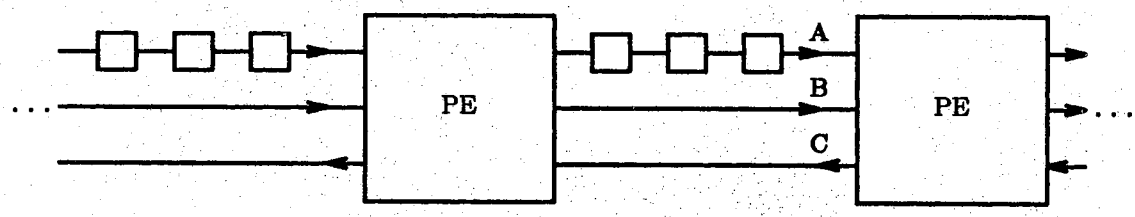


Figure 2: Block diagram of the linear array for the matrix multiplication algorithm. Data A and B travel from left to right and Data C travel from right to left. Three buffers are needed in data link for A.

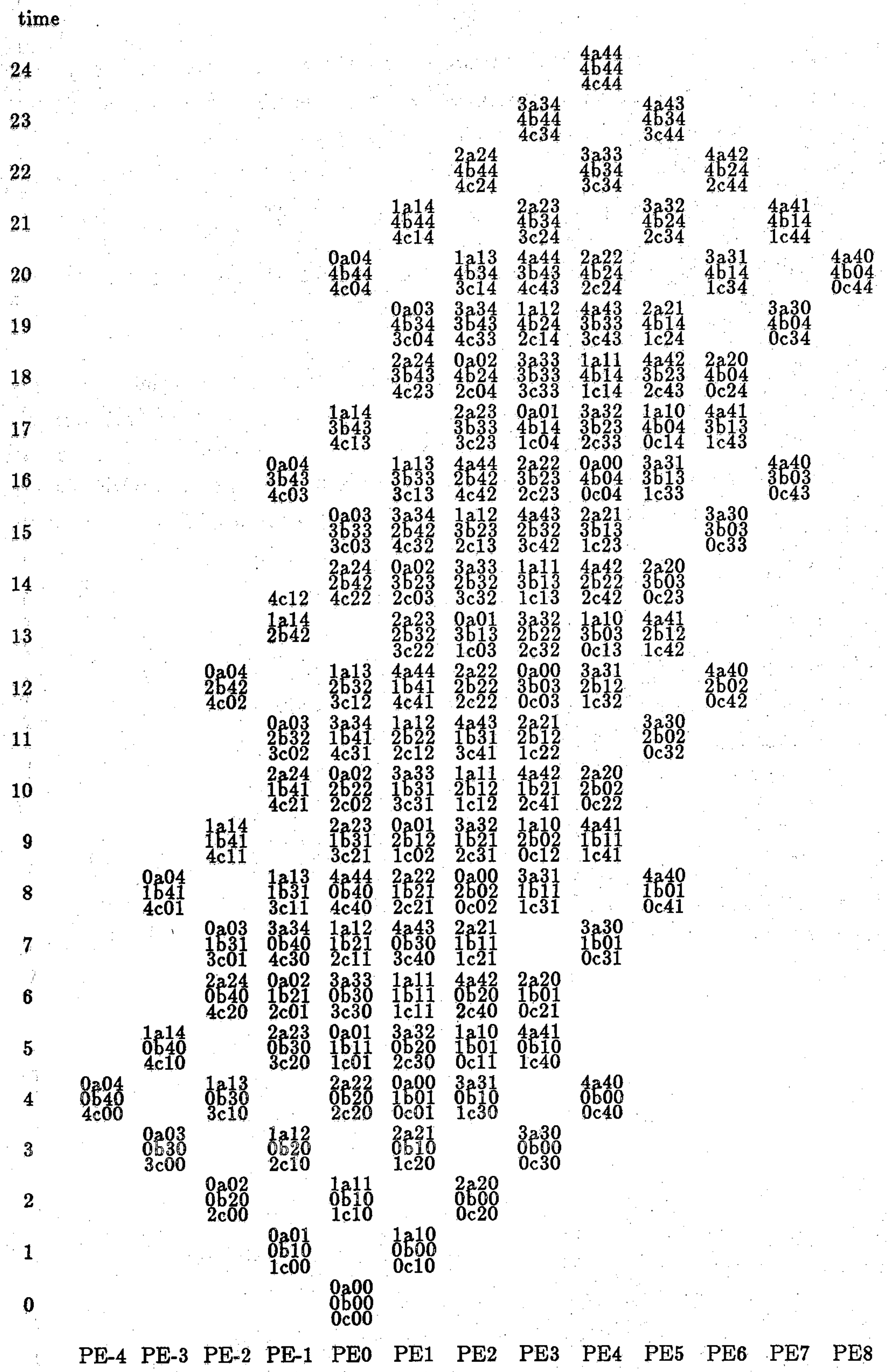


Figure 3: The execution of matrix multiplication algorithm. The small block with the left most column being $[j_1, j_2, j_3]^T$ corresponds to the computation $c_{j_1 j_2} = a_{j_1 j_3} \cdot b_{j_3 j_2}$ which is executed at processor $j_1+j_2-j_3$ and at time $j_1+4j_2+j_3$.

In [23], with the same space mapping matrix S, the linear schedule vector $\Pi'=[2, 1, \mu]$ is used and the corresponding conflict vector is $\bar{\gamma}=[-(\mu+1), 2+\mu, 1]^T$. When $\mu=3$, Π' is the optimal solution. However, when $\mu=4$, it is not. When $\mu=4$, the total execution time by Π' is $t'=\mu(3+\mu)+1$ which is longer than the optimal linear schedule Π^o. Also, the number of buffers is $\sum_{i=1}^{3}(\Pi'\bar{d}_i-1) = 4$. The systolic array designed in this paper only needs three buffers on data link for data A or dependence vector $\bar{d}_2$.

Example 5.2: Consider the transitive closure algorithm in Example 3.2 where the space mapping matrix is given as $S=[0, 0, 1]$. The dependence matrix D and index set J are shown in Equation 3.6. To satisfy condition 1 in Equation 5.2, it must have $\pi_2>0$, $\pi_3>0$ and $\pi_1-\pi_2>0$ or $\pi_1>\pi_2>0$. This means each entry of the linear schedule vector Π must be positive. Therefore, the problem of finding an optimal linear schedule vector for the transitive closure algorithm is formulated as an integer linear programming problem:

$$\min f=\mu(\pi_1+\pi_2+\pi_3)$$
$$\text{subject to} \begin{cases} (1)\ \pi_i\geq 1,\ i=2, 3,\ \pi_1-\pi_2-\pi_3\geq 1, \\ \quad \pi_1-\pi_2\geq 1,\ \pi_1-\pi_3\geq 1 \\ (2)\ SD=PK \text{ and } \sum_{j=1}^{r} k_{ji} \leq \Pi\,\bar{d}_i,\ i=1, \ldots, 3 \\ (3)\ \pi_2\geq\mu+1, \text{ or } \pi_1\geq\mu+1 \\ (4)\ \Pi\in Z^{1\times 3} \end{cases} \tag{5.4}$$

where constraint 3 is derived in Example 3.2 and shown in Equation 3.7. Again, as indicated in the appendix, this problem can be formulated as two linear programming problems. The optimal solution of this algorithm is $\Pi^o=[\mu+1, 1, 1]$ when $\mu\geq 2$. The derivation of the solution is shown in the appendix. The total execution time is $t=\mu(3+\mu)+1$ according to Equation 2.7. As mentioned in Section 1, the solution found by the heuristic procedure in [22] is $\Pi'=[2\mu+1, 1, 1]$ (notice that the lower and upper bounds for index points are 1 and n in [22] and therefore, $\mu=n-1$ in [22]) and the total execution time is $t'=\mu(2\mu+3)+1$ which is much longer than for Π^o. The matrix of interconnection primitives is $P=SD=[1, 0, -1, 0, -1]$ and $K=I$, the identity matrix. Clearly, there are no computational conflicts because the conflict vector of the mapping matrix $\begin{bmatrix} 0 & 0 & 1 \\ \mu+1 & 1 & 1 \end{bmatrix}$ is $\bar{\gamma}=[1, -(\mu+1), 0]^T$ which is feasible according to Theorem 2.2. Also, as explained in the appendix, there is no link collision either which means no two data use the same link at the same time. □

Problem 2.2 can be formulated as an integer programming problem in general if the multipier matrix U can be expressed as a function of Π. Proposition 8.1 in the appendix expresses matrix U as a function of Π for the mapping matrix $T \in Z^{3\times 5}$. Therefore, Problem 2.2 can be formulated as an integer programming problem as follows by using the necessary and sufficient condition in Theorem 4.7.

$$\min f = \sum_{i=1}^{n} |\pi_i| \mu_i \tag{5.5}$$

$$\textit{subject to} \begin{cases} (1)\ \Pi D > \bar{0} \\ (2)\ rank(T) = k \\ (3)\ \exists i \in \{1, \ldots, 5\}\ \ |u_{i4}+u_{i5}| > \mu_i,\ \ u_{i4}\cdot u_{i5} \geq 0 \\ (4)\ \exists l \in \{1, \ldots, 5\}\ \ |u_{l4}-u_{l5}| > \mu_l,\ \ u_{l4}\cdot u_{l5} \leq 0 \\ (5)\ \exists i' \in \{1, \ldots, 5\}\ \ |u_{i'4}| > \mu_{i'} \\ (6)\ \exists j' \in \{1, \ldots, 5\}\ \ |u_{j'5}| > \mu_{j'} \\ (7)\ SD = PK \text{ and } \sum_{j=1}^{r} k_{ji} \leq \Pi \bar{d}_i,\ \ i = 1, \ldots, 5 \\ (8)\ \Pi \in Z^{1\times 5} \end{cases} \tag{5.6}$$

where $T = \begin{bmatrix} S \\ \Pi \end{bmatrix}$. Clearly constraint 2 in (5.6) is linear because the determinant of T is a linear function of π_i, $i=1, \ldots, n$. If it is required that $\pi_i > 0$, $i=1, \ldots, n$, the objective function in Equation 5.5 is linear and the integer programming problem in (5.5) and (5.6) can be transformed to be linear because the constraint $u_{i4}\cdot u_{i5} \geq 0$ can be replaced equivalently by $\{u_{i4},\ u_{i5} \geq 0$ or $u_{i4},\ u_{i5} \leq 0\}$ and so can the constraint $u_{l4}\cdot u_{l5} \leq 0$. Constraints 5 and 6 ensure $\bar{u}_4$ and $\bar{u}_5$ are feasible conflict vectors as required by Theorem 4.7. Again, instead of mapping the algorithm into a target machine with fixed interconnections, if a new processor array is designed specially for that algorithm, constraint 7 can be removed as illustrated in Examples 5.1 and 5.2. This formulation is being used to find the optimal design of a 2-dimensional bit level processor array for the bit level matrix multiplication algorithm and the results will be reported in a separate paper due to limited space.

6. CONCLUSIONS AND FUTURE WORK

The main contributions of this paper are first, the necessary and sufficient conditions for computational conflict-free mappings, and second, the optimization procedure and the integer programming problem formulation of finding optimal solutions to map algorithms with n nested loops into lower dimensional processor arrays. Without these necessary and sufficient conditions for conflict-free mappings, the integer programming formulation is impossible, and even the optimization procedure has to enumerate all index points of the algorithm to see if there is a computational conflict. These techniques can be applied to map algorithms with

n nested loops into linear or 2-dimensional processor arrays with the total execution time minimized; they are especially useful for programming bit level processor arrays such that the total execution time is minimized.

Future work includes consideration of the number of buffers and length of wires required by the mappings and investigation of the following two problems.

Problem 6.1 (Space-optimal and conflict-free mapping problem): Given an n-dimensional uniform dependence algorithm and a linear schedule vector, find a space mapping matrix $S \in Z^{(k-1)\times n}$ such that $T=\begin{bmatrix} S \\ \Pi \end{bmatrix}$ is conflict-free and the number of processors plus the wire length of the array is minimized.

Problem 6.2 (Optimal conflict-free mapping problem): Given an n-dimensional uniform dependence algorithm and a $(k-1)$-dimensional processor array, find a conflict-free mapping matrix $T \in Z^{k\times n}$ such that a certain criterion is optimized.

In general, in Problem 2.2, space mapping matrix S is given and usually is not a function of problem size variables μ_i $i=1, \ldots, n$; in Problem 6.1, linear schedule vector Π is given, possibly by the optimization procedure proposed in [16], and usually is not a function of problem size variables; and in Problem 6.2, both S and Π are not given and possibly are both functions of problem size variables.

7. REFERENCES:

[1] R.M. Karp, R.E. Miller and S. Winograd, "The Organization of Computations for Uniform Recurrence Equations," JACM 14, 3, Jul. 1967, pp. 563-590.

[2] D.I. Moldovan and J.A.B. Fortes, "Partitioning and Mapping Algorithms into Fixed Size Systolic Arrays," IEEE Trans. Computers, Vol. C-35, No. 1, Jan. 1986, pp. 1-12.

[3] P.R. Cappello and K. Steiglitz, "Unifying VLSI Array Designs with Geometric Transformations," Proc. of 1983 Int'l Conf on Parallel Processing, pp. 448-457.

[4] P. Quinton, "Automatic Synthesis of Systolic Arrays from Uniform Recurrent Equations," Proc. 11'th Annual Symposium on Computer Architecture, 1984, pp. 208-214.

[5] S.K. Rao, "Regular Iterative Algorithms and Their Implementations on Processor Arrays," Ph.D Dissertation, Stanford University, Stanford, California, Oct. 1985.

[6] M. Chen, "A Design Methodology for Synthesizing Parallel Algorithms and Architectures," Journal of Parallel and Distributed Computing, Dec. 1986, pp. 461-491.

[7] J.-M. Delosme and I. C. F. Ipsen, "An Illustration of a Methodology for the Construction of Efficient Systolic Architectures in VLSI," Proc. Second Int'l Symposium on VLSI Technology, Systems and Applications, 1985, pp. 268-273.

[8] S. Y. Kung, "VLSI Array Processors," Prentice-Hall, Englewood Cliffs, N.J. 1987.

[9] C. Guerra and R. Melhem, "Synthesizing Non-Uniform Systolic Designs," Proc. 1986 Int'l Conf. on Parallel Processing, pp. 765- 771.

[10] G.-J. Li and B. W. Wah, "The Design of Optimal Systolic Arrays," IEEE Trans. Computers, Vol. C-34, Jan. 1985, pp. 66-77.

[11] M.T. O'Keefe and J.A.B. Fortes, "A Comparative Study of Two Systematic Design Methodologies for Systolic Arrays," Proc. 1986 Int'l Conf. on Parallel Processing, pp. 672-675.

[12] J.A.B. Fortes, F. Parisi-Presicce "Optimal Linear Schedule for the Parallel Execution of Algorithms," Proc. of 1984 Int'l Conf. on Parallel Processing, pp. 322-328.

[13] L. Lamport, "The Parallel Execution of DO loops," Comm. of the ACM, Vol. 17, No. 2, Feb. 1974, pp. 83-93.

[14] J.-K. Peir and R. Cytron, "Minimum Distance: A Method for Partitioning Recurrences for Multiprocessors," Proc. 1987 Int'l Conf. on Parallel Processing, pp. 217-225.

[15] V.P. Roychowdhury and T. Kailath, "Subspace Scheduling and Parallel Implementation of Non-Systolic Regular Iterative Algorithms," Journal of VLSI Signal Processing, 1, 1989.

[16] W. Shang and J.A.B. Fortes, "Time Optimal Linear Schedules for Algorithms with Uniform Dependencies," Proceedings of Int'l Conf. on Systolic Arrays, May 1988, pp. 393-402 (also to appear in IEEE Trans. on Computers).

[17] S. Y. Kung, S. C. Lo and P. S. Lewis, "Optimal Systolic Design for the Transitive Closure and the Shortest Path Problems," IEEE Trans. on Computer, Vol. C-36, May 1987, pp. 603-614.

[18] G. Strang, "Linear Algebra and its Applications," Second Edition, Academic Press 1980.

[19] R. Cytron, "Doacross: Beyond Vectorization for Multiprocessors (Extended Abstract)," Proc. of 1986 Int'l Conf. on Parallel Processing, pp. 836-844.

[20] R. Kannan and A. Bachem, "Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix," SIAM J. on Computing, Vol. 8, No. 4, Nov. 1979, pp. 499-507.

[21] H.T. Kung and C.E. Leiserson, "Algorithms for VLSI Array Processors," in C. Mead and L. Conway, "Introduction to VLSI Systems," Addison-Wesley, 1980, Section 8.3.

[22] P. Lee and Z. M. Kedem, "Mapping Nested Loop Algorithms into Multidimensional Systolic Arrays," IEEE Trans. on Parallel and Distributed Systems, Vol. 1, No. 1 January, 1990, pp. 64-76.

[23] P. Lee and Z. M. Kedem, "Synthesizing Linear Array Algorithms from Nested For Loop Algorithms," IEEE Trans. on Computers, Vol. 37, No. 12, December 1988, pp. 1578-1598.

[24] D. A. Padua, "Multiprocessors: Discussion of Theoretical and Practical Problems," Ph.D Thesis, Univ. of Illinois at Urb.-Champ., Rept. No. UIUCDCS-R-79-990, Nov. 1979.

[25] Y. Wong and J.-M. Delosme, "Optimal Systolic Implementation of N-dimensional Recurrences," IEEE Proc. ICCD, 1985, pp. 618-621.

[26] V.E. Taylor and J.A.B. Fortes, "Using RAB to Map Algorithms into Bit-level Systolic Arrays," Proc. of Int'l Conf. on Supercomputing, May 1987.

[27] L.J. Mordell, *Diophantine Equations*, Academic Press, New York, 1969, pp. 30.

[28] M.T. O'Keefe and J.A.B. Fortes, "Bit Level Processor Array: Current Architectures and a Design and a Programming Tool," 1988 Int'l Symposium on Circuit and System, Helsinki, Finland, June 1988, pp. 2751-2755.

[29] A. Schrijver, "Theory of Linear and Integer Programming," John Wiley & Sons, 1986.

[30] W. Shang, "Scheduling, Partitioning and Mapping of Uniform Dependence Algorithms on Processor Arrays," Ph.D Thesis, Purdue University, W. Lafayette, IN 47907, May, 1990.

[31] W.D. Hillis, "The Connection Machine," MIT Press: Cambidge, MA, 1985.

[32] Moldovan, D. I., "On the Design of Algorithms for VLSI Systolic Arrays," Proc. of IEEE, Vol. 71, No. 1, Jan. 1983, pp. 113-120.

[33] R. Davis and D. Thomas, "Systolic Array Chip Matches the Pace of High-Speed Processing," Electronic Design, Oct. 31, 1984.

[34] R.W. Hockney and C.R. Jesshope, "Parallel Computers: Architecture, Programming and Algorithms," Adam Hilger Ltd.: Bristol, 1981, pp. 178-192.

[35] K.E. Batcher, "Bit-Serial Parallel Processing Systems," IEEE Trans. on Computers, Vol. C-31, No. 5, pp. 377-384.

8. APPENDIX

Discussion of Example 5.1: Let's design a new linear systolic array for the matrix multiplication algorithm. Thus, constraint 2 in Equation 5.3 can be ignored at this moment. For an integer linear programming problem with convex solution set, if all of its extreme points are integral, then one of the extreme points is the optimal solution of that problem [29, pp. 232]. Now the solution set of the integer programming problem in Equation 5.3 is not convex because of constraint 3 although all extreme points are integral. One way to solve this problem is to partition the solution set as three convex subsets and then to find all optimal solutions for all the solution subsets. If the one with the smallest value of the objective function is satisfactory, then it is the optimal solution of the integer programming problem in Equation 5.3.

Now let's partition the solution set of the integer programming problem in Equation 5.3 as three subsets which are expressed as follows.

$$
\begin{aligned}
&(I)\ \min f=\mu(\pi_1+\pi_2+\pi_3)\\
&\textit{subject to}\begin{cases}(1)\ \pi_i\geq 1,\ \ i=1,\ 2,\ 3\\ (2)\ \pi_2+\pi_3\geq\mu+1\\ (3)\ \Pi\in Z^{1\times 3}\end{cases}\\
&(II)\ \min f=\mu(\pi_1+\pi_2+\pi_3)\\
&\textit{subject to}\begin{cases}(1)\ \pi_i\geq 1,\ \ i=1,\ 2,\ 3\\ (2)\ \pi_1+\pi_3\geq\mu+1\\ (3)\ \Pi\in Z^{1\times 3}\end{cases}\\
&(III)\ \min f=\mu(\pi_1+\pi_2+\pi_3)\\
&\textit{subject to}\begin{cases}(1)\ \pi_i\geq 1,\ \ i=1,\ 2,\ 3\\ (2)\ \mid\pi_1-\pi_2\mid\geq\mu+1\\ (3)\ \Pi\in Z^{1\times 3}\end{cases}
\end{aligned}
\tag{8.1}
$$

Each of the above problems is an integer linear programming problem with a convex solution set. Because the coefficients of π_i, $i=1, \ldots, 3$, in these inequalities are either 1, 0 or -1, every extreme point of the convex set is integral and one of the extreme points is the optimal solution of that problem. Let's first consider Formulation I in Equation 8.1; the convex solution set is defined by $\{\Pi=[\pi_1, \pi_2, \pi_3]: \pi_1\geq 1, \pi_2\geq 1, \pi_3\geq 1, \pi_2+\pi_3\geq\mu+1\}$. Each extreme point is the solution of three of the following four equations: $\pi_1=1$, $\pi_2=1$, $\pi_3=1$ and $\pi_2+\pi_3=\mu+1$. There are two such extreme points $\Pi_1=[1, 1, \mu]$ and $\Pi_2=[1, \mu, 1]$. Π_1 is not feasible because the corresponding conflict vector $[1, 1, 0]^T$ is not feasible. When μ is an even number, Π_2 is feasible and the corresponding conflict vector is $[-(\mu+1), 2, (1-\mu)]^T$. Similarly, the extreme points for the other two integer linear programming problems in Equation 8.1 are $\Pi_3=[\mu, 1, 1]$, $\Pi_4=[1, \mu+2, 1]$ and $\Pi_5=[\mu+2, 1, 1]$. Vectors $\Pi_2=[1, \mu, 1]$ and $\Pi_3=[\mu, 1, 1]$ have the same execution time and are the optimal solutions.

One more constraint $gcd(f_1, \ldots, f_n)=1$, where f_i, $i=1, \ldots, n$ are as defined in Equation 3.2, should be added to the formulation described by (5.1) and (5.2) to guarantee the greatest common divisor of the resulting conflict vector is unity. However, this makes the problem more difficult to solve. Hence, this constraint is ignored and the resulting conflict vector is checked to see if it is feasible. This is why Π_1 has a non-feasible conflict vector.

Let's design the linear systolic array for mapping matrix $T=\begin{bmatrix} S \\ \Pi_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & \mu & 1 \end{bmatrix}$. If $P=[1, 1, -1]$ is chosen as the matrix of interconnection primitives

and $K=I$ (the identity matrix), then $SD=PK$, $\sum_{j=1}^{3} k_{j1}=\Pi\,\bar{d}_1=1$, $\sum_{j=1}^{3} k_{j2}=1\leq \Pi\,\bar{d}_2=\mu=4$, $\sum_{j=1}^{3} k_{j3}=\Pi\,\bar{d}_3=1$ and constraint 2 in Equation 5.3 is satisfied. Because $\Pi\bar{d}_2-\sum_{j=1}^{3} k_{j2}=3$, three buffers are needed on the data link for $\bar{d}_2$ induced by data A. The systolic structure and the execution are shown in Figures 2 and 3, respectively. Notice that there is no data link collision because in every column of matrix K there is only one non-zero entry $k_{ii}=1$, $i=1, \ldots, 3$. This means that when data pass from the source to the destination, they use the data link just once. Data link collisions occur only if data use links more than once when passing from the source to the destination.

Discussion of Example 5.2: Similarly, the integer programming problem in (5.4) can be converted to the following two integer linear programming problems.

$$(I)\ \min f=\mu(\pi_1+\pi_2+\pi_3)$$
$$\textit{subject to}\begin{cases}(1)\ \pi_i\geq 1,\ i=2,3,\ \pi_1-\pi_2-\pi_3\geq 1,\\ \quad \pi_1-\pi_2\geq 1,\ \pi_1-\pi_3\geq 1\\ (2)\ \pi_2\geq\mu+1\\ (3)\ \Pi\in Z^{1\times 3}\end{cases}$$

$$(II)\ \min f=\mu(\pi_1+\pi_2+\pi_3)$$
$$\textit{subject to}\begin{cases}(1)\ \pi_i\geq 1,\ i=2,3,\ \pi_1-\pi_2-\pi_3\geq 1,\\ \quad \pi_1-\pi_2\geq 1,\ \pi_1-\pi_3\geq 1\\ (2)\ \pi_1\geq\mu+1\\ (3)\ \Pi\in Z^{1\times 3}\end{cases}\qquad (8.2)$$

Again, because all coefficients of π_i, $i=1, \ldots, 3$, in these inequalities are either 1, 0 or -1, all extreme points of the convex solution subsets are integral and techniques for linear programming can be applied, which check all extreme points of the solution subsets. One of the extreme points with the minimum value of the objective function f is the optimal one. Let's first consider Formulation II in Equation 8.2; the convex solution set is defined by $\{\Pi=[\pi_1, \pi_2, \pi_3]: \pi_2\geq 1, \pi_3\geq 1, \pi_1-\pi_2-\pi_3\geq 1, \pi_1-\pi_2\geq 1, \pi_1-\pi_3\geq 1, \pi_1\geq\mu+1\}$. Each extreme point is the solution of three of the following six equations: $\pi_2=1$, $\pi_3=1$, $\pi_1-\pi_2-\pi_3=1$, $\pi_1-\pi_2=1$, $\pi_1-\pi_3=1$, $\pi_1=\mu+1$. There are four such extreme points $\Pi_1=[\mu+1, 1, 1]$, $\Pi_2=[\mu+1, 1, \mu-1]$, $\Pi_3=[\mu+1, 1, \mu]$ and $\Pi_4=[\mu+1, \mu-1, 1]$. When $\mu\geq 2$, all the above linear schedule vectors satisfy constraint $\Pi D>\bar{0}$. The corresponding conflict vectors, according to Equation 3.2, are $\bar{\gamma}_1=[1, -(\mu+1), 0]^T$, $\bar{\gamma}_2=[1, -(\mu+1), 0]^T$,

$\bar{\gamma}_3=[1, -(\mu+1), 0]^T$ and $\bar{\gamma}_4=[\mu-1, -(\mu+1), 0]^T$. Vectors $\bar{\gamma}_i$, $i=1, ..., 3$, are feasible and $\bar{\gamma}_4$ is feasible if μ is an even number. Similarly, the extreme points for integer programming problem I in (8.2) can be found. The extreme point with the minimum value of the objective function f is $\Pi^o=\Pi_1 = [\mu+1, 1, 1]$ and the total execution time by Π_1 is $\mu(\mu+3)+1$ according to Equation 2.7.

If the matrix of interconnection primitives $P=SD = [1, 0, -1, 0, -1]$ is chosen and $K=I$, the identity matrix, then constraint 2 in (5.4) is satisfied. Similar to the design for the matrix multiplication algorithm, there is no data link collision because in every column of matrix K there is only one non-zero entry $k_{ii}=1$, $i=1, ..., 3$. This means that when data pass from the source to the destination, they use the data link just once. Data link collisions occur only if data use links more than once when passing from the source to the destination.

Proposition 8.1: Let mapping matrix $T=\begin{bmatrix} S \\ \Pi \end{bmatrix}$ where $\Pi = [\pi_1, ..., \pi_5] \in Z^{1\times 5}$ and $S=\begin{bmatrix} s_{11} & \cdots & s_{15} \\ s_{21} & \cdots & s_{25} \end{bmatrix} \in Z^{2\times 5}$. If $s_{11}=1$ and $s_{22}-s_{21}s_{12}=1$, then the last two columns of multiplier U are as follows:

$$\bar{u}_4=-\frac{h_{34}}{g_1}\begin{bmatrix} c_{13} \\ c_{23} \\ 1 \\ 0 \\ 0 \end{bmatrix}+\frac{h_{33}}{g_1}\begin{bmatrix} c_{14} \\ c_{24} \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{8.3a}$$

$$\bar{u}_5=-\frac{p_1 h_{35}}{g_2}\begin{bmatrix} c_{13} \\ c_{23} \\ 1 \\ 0 \\ 0 \end{bmatrix}-\frac{q_1 h_{35}}{g_2}\begin{bmatrix} c_{14} \\ c_{24} \\ 0 \\ 1 \\ 0 \end{bmatrix}+\frac{g_1}{g_2}\begin{bmatrix} c_{15} \\ c_{25} \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{8.3b}$$

where

$$h_{33}=-\pi_1(s_{12}s_{21}s_{13}-s_{12}s_{23}+s_{13})+\pi_2(s_{21}s_{13}-s_{23})+\pi_3, \tag{8.4a}$$

$$h_{34}=-\pi_1(s_{12}s_{21}s_{14}-s_{12}s_{24}+s_{14})+\pi_2(s_{21}s_{14}-s_{24})+\pi_4, \tag{8.4b}$$

$$h_{35}=-\pi_1(s_{12}s_{21}s_{15}-s_{12}s_{25}+s_{15})+\pi_2(s_{21}s_{15}-s_{25})+\pi_5, \tag{8.4c}$$

$$c_{13}=-s_{12}(s_{21}s_{13}-s_{23})-s_{13} \tag{8.5a}$$

$$c_{14}=-s_{12}(s_{21}s_{14}-s_{24})-s_{14} \tag{8.5b}$$

$$c_{15}=-s_{12}(s_{21}s_{15}-s_{25})-s_{15} \tag{8.5c}$$

$$c_{23}=s_{21}s_{13}-s_{23} \tag{8.5d}$$

$$c_{24}=s_{21}s_{14}-s_{24} \tag{8.5e}$$

$$c_{25}=s_{21}s_{15}-s_{25} \tag{8.5f}$$

$$g_1=gcd(h_{33},\ h_{34}),\quad p_1h_{33}+q_1h_{34}=g_1 \tag{8.6}$$

$$g_2=gcd(g_1,\ h_{35}),\quad p_2g_1+q_2h_{35}=g_2. \tag{8.7}$$

The proof can be found in Chapter 6 of [30]. Expression 8.3 is expected to be much simpler if the space mapping S is given in numbers and some special conditions are considered; therefore, it is easier to construct the multiplier U for the T with a specific space mapping matrix.