The International Conference on Application Specific Array Processors,
297--308, Venice, 1993.

A New Formulation of the Mapping Conditions for the Synthesis of Linear Systolic Arrays

Jingling Xue
School of Electrical and Electronic Engineering
Nanyang Technological University
Singapore 2263

Abstract

This paper presents a new formulation for mapping algorithms into linear systolic arrays. The closed-form necessary and sufficient mapping conditions are derived to identify mappings without computation conflicts and data link collisions. These mapping conditions are easy to check because their constituent variables are the space-time mapping matrix and the problem size parameters. The design of optimal arrays is a mathematic programming problem, which can be solved by a systematic enumeration of its search space.

1. Introduction

Most existing systolic design methods are restricted to the cases where n-dimensional algorithms are mapped into $(n-1)$-dimensional systolic arrays. For detail, see [3, 4] and references therein. The mapping conditions are linear and have closed-form expressions. The design of optimal systolic arrays is an integer programming problem [4].

There have been several attempts on mapping n-dimensional algorithms into lower dimensional arrays [1, 2, 4, 5, 6]. Lee and Kedem [2] gave a set of necessary and sufficient conditions on mappings without computation conflicts and communication conflicts (data link collisions). A *computation conflict* occurs if more than one computation of an algorithm are mapped to the same processor and the same time step. A *communication conflict* (*data link collision*) occurs if more than one datum are so mapped that they travel along the same data link at the same time step. The two mapping conditions proposed in [2] to avoid these two conflicts are referred to here as the computation and communication constraints, respectively. However, their verification is done by analysis of all computations in the iteration space, which is too expensive to be practically feasible.

Shang and Fortes [6] presented a set of closed-form conditions on computation conflict-free mappings. In their methodology, the issue of communication conflicts was not addressed. Ganapathy and Wah [1] proposed a parameter-based method in which the operations of the array are captured by a set of parameters, and the constraints are derived to avoid computation and communication conflicts. However, the computation constraint is assumed to be implied by the communication constraint. It is therefore not part of their constraints for the correctness of a mapping, although it should be in certain cases. Their communication constraint ([1, Thm. 3]) does not have closed-form expressions and contains a few parameters that need to be further defined every time when it is applied to individual variables. How this is done in general was not mentioned.

This paper is concerned with the derivation of closed-form computation and communication constraints on the design of linear systolic arrays from three-dimensional algorithms. Depending on the shapes of the domains of input and output data, different closed-form communication constraints are provided. Often the absence of communication conflicts on one link implies the absence of computation conflicts. Conditions are provided that check

for this situation so that the computation constraint can be eliminated. If the conditions are not satisfied, we simply enforce the conditions by adding the communication constraint for an imaginary variable. So much of the text is devoted to the derivation of closed-form communication constraints.

2. Algorithm model

$\mathbb{Z}$ ($\mathbb{Q}$) denotes the set of integers (rationals). $\mathbb{Z}^n$ ($\mathbb{Q}^n$) denotes its n-fold Cartesian product. The starting point of our synthesis method is a single-assignment algorithm (called uniform dependence algorithm in [6]) adhering to the following format:

INPUT: $\text{in}_{V_1} = P_1$
$\cdots$
$\text{in}_{V_\ell} = P_\ell$

OUTPUT: $\text{out}_{V_1} = Q_1$
$\cdots$
$\text{out}_{V_\ell} = Q_\ell$

LOOPS: **for** x_1 **from** lb_1 **to** ub_1 **do**
$\cdots$
for x_n **from** lb_n **to** ub_n **do**
S_1
$\cdots$
S_r

where

- $V_1, \cdots, V_\ell$ are the variables of the algorithm; they form the variable set $\mathcal{V}$.
- Set $\text{in}_{V_i} \in \mathbb{Z}^n$ ($\text{out}_{V_i} \in \mathbb{Z}^n$) is called the *input* (*output*) *space* of variable V_i. $V_i(I)$ for all vectors I in in_{V_i} (out_{V_i}) represent the input (output) data of V_i. By convention, empty input and output spaces need not be declared in the algorithm.
- The loops define a space Φ of points $(x_1, \cdots, x_n)$ called the *iteration space* ($\Phi \subset \mathbb{Z}^n$):

$$\Phi \quad = \quad \{(x_1, \cdots, x_n) \mid (\forall\ i : 0 < i \leq n : lb_i \leq x_i \leq ub_i)\}$$

  The bounds lb_i and rb_i are linear expressions in the loop indices x_1 to x_{i-1} and in additional variables that specify the problem size. The loops have unit stride.
- Statement S_i in the loops has the following form:

$$V(I) \quad = \quad \begin{cases} I \in \Phi_1 \rightarrow f_1(W_1(I - \vartheta_1), \ldots) \\ \quad \cdots \\ I \in \Phi_s \rightarrow f_s(W_s(I - \vartheta_s), \ldots) \end{cases}$$

  $\Phi_j \in \mathbb{Z}^n$ and $\Phi_k \in \mathbb{Z}^n$ are disjoint if j and k differ. The three dots in the argument list of f_i stand for an arbitrary but fixed number of similar arguments. The constant vectors $\vartheta_i \in \mathbb{Z}^n$ are called *dependence vectors*. The graph whose vertices are the points of the iteration space and whose edges are the dependences between the points is called the *dependence graph*. If V is defined for the entire iteration space, we write

$$V(I) \quad = \quad f(W(I - \vartheta), \cdots)$$

Without loss of generality, we assume that each variable V in the algorithm is associated with one dependence vector, denoted ϑ_V.

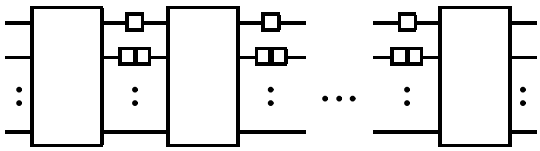

Figure 1: The linear systolic array model.

3. Array model

Fig. 1 displays the *linear systolic array*, which has the following properties:

- The execution of the array is governed by a global clock that ticks in unit time.
- Only the two border processors are connected to the host.
- Only neighbouring processors are connected directly with each other.
- If a variable is stationary, its elements are stored in the local processor memories of the array. If a variable is moving, its elements move along a fixed link with a constant velocity. A *link* is a line of channels all of which have the same number of delay buffers. A buffer retains a value for one time step, i.e., one clock tick.

4. The space-time mapping

A space-time mapping, Π, that describes a linear systolic array is an $2 \times n$ matrix:

$$\Pi \quad = \quad \begin{bmatrix} \lambda \\ \sigma \end{bmatrix} \quad = \quad \begin{bmatrix} \lambda_1 & \cdots & \lambda_n \\ \sigma_1 & \cdots & \sigma_n \end{bmatrix}$$

where $\lambda \in \mathbb{Z}$ is called the *scheduling vector* and $\sigma \in \mathbb{Z}$ the *allocation vector*. A point I in the iteration space is computed at processor σI at time step λI.

We write $\mathsf{flow}(V)$ for the velocity with which the elements of a variable V travel:

$$\mathsf{flow}(V) \quad = \quad \sigma \vartheta_V / \lambda \vartheta_V$$

Thus, the number of buffers associated with a channel for variable V is $\frac{1}{|\mathsf{flow}(V)|} - 1$, where the 1 accounts for the fact that evaluation of a point takes unit time. A variable V is *moving* if $\mathsf{flow}(V) \neq 0$ and is *stationary* if $\mathsf{flow}(V) = 0$. We make the convention that a variable V moves to the right if $\mathsf{flow}(V) > 0$ and to the left if $\mathsf{flow}(V) < 0$. The coordinates of the two border processors of the array can be calculated as follows:

$$p_{\min} = (\min \ I : I \in \Phi : \sigma I) \qquad p_{\max} = (\max \ I : I \in \Phi : \sigma I)$$

The time steps at which data are input and output are defined as follows.

- Function input specifies the steps at which input data are input into the array:

$$\begin{aligned} &\mathsf{input} : \ \mathcal{V} \to \mathrm{in}_V \to \mathbb{Z} \\ &\mathsf{input}(V)(I) \ = \ \begin{cases} \mathsf{flow}(V) > 0 & \to \quad \lambda I - (\sigma I - p_{\min})/\mathsf{flow}(V) \\ \mathsf{flow}(V) < 0 & \to \quad \lambda I - (\sigma I - p_{\max})/\mathsf{flow}(V) \end{cases} \end{aligned}$$

- Function output specifies the steps at which output data are output from the array:

$$\begin{aligned} &\mathsf{output} : \ \mathcal{V} \to \mathrm{out}_V \to \mathbb{Z} \\ &\mathsf{output}(V)(I) \ = \ \begin{cases} \mathsf{flow}(V) > 0 & \to \quad \lambda I - (\sigma I - p_{\max})/\mathsf{flow}(V) \\ \mathsf{flow}(V) < 0 & \to \quad \lambda I - (\sigma I - p_{\min})/\mathsf{flow}(V) \end{cases} \end{aligned}$$

The mapping conditions for the correctness of space-time mappings for linear systolic arrays are given below. The notation $f : D \hookrightarrow R$ denotes that function f is injective from domain D to range R. For $x, y \in \mathbb{Z}$, we write $x|y$ iff x divides y.

Definition 1 (Mapping conditions) A space-time mapping, Π, is *correct* iff

- $(\forall\, V : V \in \mathcal{V} : \lambda\vartheta_V > 0)$ (Precedence Constraint)
- $\Pi : \Phi \hookrightarrow \mathbb{Z}^2$ (Computation Constraint)
- $(\forall\, V : V \in \mathcal{V} : \mathsf{flow}(V) \neq 0 \implies \sigma\vartheta_V | \lambda\vartheta_V)$ (Delay Constraint)
- $(\forall\, V : V \in \mathcal{V} : \mathsf{flow}(V) \neq 0 \implies \mathsf{input}(V) : \mathrm{in}_V \hookrightarrow \mathbb{Z})$
  $(\forall\, V : V \in \mathcal{V} : \mathsf{flow}(V) \neq 0 \implies \mathsf{output}(V) : \mathrm{out}_V \hookrightarrow \mathbb{Z})$ (Communication Constraint)

These mapping conditions are essentially the same as those proposed by Lee and Kedem [2], except that our communication constraint is both a necessary and sufficient condition on communication conflict-free mappings.

The precedence constraint ensures that the dependences prescribed in the algorithm are respected. It remains the same as in the synthesis of $(n-1)$-dimensional arrays. The computation constraint ensures that the mapping is computation conflict-free. The delay constraint states that the number of buffers on any channel for a moving variable is a non-negative integer. The communication constraint ensures that the mapping is communication conflict-free. It is redundant in the synthesis of $(n-1)$-dimensional arrays.

Both the computation and communication constraints depend on the index vectors and do not have closed-form expressions. Any formulation of finding correct and optimal mappings based directly on these conditions can only be effective for the problems of small sizes. In what follows, we focus on the derivation of the closed-form and index vector-independent computation and communication constraints. We shall drop the antecedent $\mathsf{flow}(V)$ in the delay and the communication constraints. It is understood that the two constraints are only applicable for moving variables. Similarly, whenever we apply functions input and output to a variable V or write $\sigma\vartheta_V|\lambda\vartheta_V$, we mean that V is a moving variable.

5. Simplifying the computation and communication constraints

We present our results in a more general setting by extending the functions input and output so that they are of types $\mathcal{V} \to \mathbb{Q}^n \to \mathbb{Q}$ and $\mathcal{V} \to \mathbb{Q}^n \to \mathbb{Q}$, respectively. The main results are summarised in three theorems. Thm. 1 presents the condition under which the computation constraint can be eliminated. Thms. 2 and 3 present the condition under which the communication constraint can be simplified. For ease of presentation, we first present two technical lemmata, which are useful in the proof of Thms. 1 – 3. The correctness of the two lemmata can be proved by a simple algebraic calculation. We write $I \overset{\vartheta_V}{\longleftrightarrow} J$ if $I = J + m\vartheta_V$ for some $m \in \mathbb{Q}$.

Lemma 1 (1) $(\forall\, V, I : V \in \mathcal{V} \wedge I \in \mathbb{Q}^n : I \overset{\vartheta_V}{\longleftrightarrow} J \implies \mathsf{input}(V)(I) = \mathsf{input}(V)(J))$
(2) $(\forall\, V, I : V \in \mathcal{V} \wedge I \in \mathbb{Q}^n : I \overset{\vartheta_V}{\longleftrightarrow} J \implies \mathsf{output}(V)(I) = \mathsf{output}(V)(J))$

This lemma has the following implication. The input (output) space of a variable V cannot contain two distinct points I and J such that $I \overset{\vartheta_V}{\longleftrightarrow} J$. Otherwise the two data $V(I)$ and $V(J)$ would be input (output) at the same time under any space-time mapping.

Lemma 2 $(\forall\, V, I : V \in \mathcal{V} \wedge I \in \mathbb{Q}^n : \mathsf{output}(V)(I) = \mathsf{input}(V)(I) + |(p_{\max} - p_{\min})/\mathsf{flow}(V)|)$.

This lemma says that a datum of V takes $|(p_{\max} - p_{\min})/\mathsf{flow}(V)|$ to travel across the array.

The sufficient conditions for the elimination of the computation constraint are presented below. Let X and Y be two sets and ϑ be a vector in $\mathbb{Z}^n$. We write $X \xrightarrow{\vartheta} Y$ if, when projected along ϑ onto a fixed hyperplane, the projection of X is a (not necessarily strict) subset of the projection of Y. That is, if $X \xrightarrow{\vartheta} Y$, then for every I in X there is J in Y such that $I \xleftrightarrow{\vartheta} J$. Assume that a mapping satisfies the precedence constraint for variable V and that $\Phi \xrightarrow{\vartheta_V} \mathrm{in}_V$ ($\Phi \xrightarrow{\vartheta_V} \mathrm{out}_V$) holds. Then the absence of communication conflicts for the input (output) data of V implies the absence of computation conflicts.

Theorem. 1 *Let* $\lambda\vartheta_V \neq 0$. (1) If $\Phi \xrightarrow{\vartheta_V} \mathrm{in}_V$, then $(\mathsf{input}(V) : \mathrm{in}_V \hookrightarrow \mathbb{Z}) \Longrightarrow (\Pi : \Phi \hookrightarrow \mathbb{Z})$
(2) If $\Phi \xrightarrow{\vartheta_V} \mathrm{out}_V$, then $(\mathsf{output}(V) : \mathrm{out}_V \hookrightarrow \mathbb{Z}) \Longrightarrow (\Pi : \Phi \hookrightarrow \mathbb{Z})$

Proof. We prove (1) by contradiction. The proof of (2) is similar. Let there be two distinct points $I, J \in \Phi$ such that $\Pi I = \Pi J$, i.e., $\lambda I = \lambda J \wedge \sigma I = \sigma J$. Since $\lambda\vartheta_V \neq 0$, $I \xleftrightarrow{\vartheta_V} J$ does not hold. By further using $\Phi \xrightarrow{\vartheta_V} \mathrm{in}_V$, we know that in_V must contain two distinct points I' and J' such that $I \xleftrightarrow{\vartheta_V} I'$ and $J \xleftrightarrow{\vartheta_V} J'$. We show that $\mathsf{input}(V)(I') = \mathsf{input}(V)(J')$ and thus achieve the impossible. By Lemma 1, $\mathsf{input}(V)(I) = \mathsf{input}(V)(I')$ and $\mathsf{input}(V)(J) = \mathsf{input}(V)(J')$. It suffices to show that $\mathsf{input}(V)(I) = \mathsf{input}(V)(J)$. This follows from the proof hypothesis $\lambda I = \lambda J \wedge \sigma I = \sigma J$ and the definition of input. □

Next, we consider the elimination of redundant communication constraints. In general, there are two communication constraints associated with a variable: one for the input space and the other for the output space. One of these two constraints can be eliminated if a certain relationship between the input and output spaces holds. The following theorem identifies such a relationship. Let X be a set in $\mathbb{Z}^n$ and $a \in \mathbb{Z}^n$, the set $\{x + a \mid x \in X\}$ is called the *translation* of X by a. Let X and Y be two sets and ϑ be a vector in $\mathbb{Z}^n$. We write $X \overset{\vartheta}{\longmapsto} Y$ if there exists a vector $o \in \mathbb{Z}^n$ such that $X \xrightarrow{\vartheta} (Y + o)$. If $\mathrm{out}_V \overset{\vartheta_V}{\longmapsto} \mathrm{in}_V$ ($\mathrm{in}_V \overset{\vartheta_V}{\longmapsto} \mathrm{out}_V$), then a mapping that is communication conflict-free for the input (output) data of V is also communication conflict-free for the output (input) data of V.

Theorem. 2 (1) If $\mathrm{out}_V \overset{\vartheta_V}{\longmapsto} \mathrm{in}_V$, then $(\mathsf{input}(V) : \mathrm{in}_V \hookrightarrow \mathbb{Z}) \Longrightarrow (\mathsf{output}(V) : \mathrm{out}_V \hookrightarrow \mathbb{Z})$
(2) If $\mathrm{in}_V \overset{\vartheta_V}{\longmapsto} \mathrm{out}_V$, then $(\mathsf{output}(V) : \mathrm{out}_V \hookrightarrow \mathbb{Z}) \Longrightarrow (\mathsf{input}(V) : \mathrm{in}_V \hookrightarrow \mathbb{Z})$

Proof. For reasons of symmetry, we only prove (1). Since $\mathsf{input}(V) : \mathrm{in}_V \hookrightarrow \mathbb{Z}$, it follows from algebraic calculations that $\mathsf{input}(V) : (\mathrm{in}_V + a) \hookrightarrow \mathbb{Z}$ for any vector a in $\mathbb{Z}^n$. By Lemma 2, we obtain $\mathsf{output}(V) : (\mathrm{in}_V + a) \hookrightarrow \mathbb{Z}$. The assumption that $\mathrm{out}_V \overset{\vartheta_V}{\longmapsto} \mathrm{in}_V$ means that there is an $o \in \mathbb{Z}^n$ such that $\mathrm{out}_V \xrightarrow{\vartheta_V} (\mathrm{in}_V + o)$. With the a being replaced by o, we have $\mathsf{output}(V) : (\mathrm{in}_V + o) \hookrightarrow \mathbb{Z}$. That $\mathrm{out}_V \xrightarrow{\vartheta_V} (\mathrm{in}_V + o)$ implies that, for every I in out_V, there is J in $\mathrm{in}_V + o$ such that $I \xleftrightarrow{\vartheta_V} J$. By Lemma 1, we have $\mathsf{output}(V)(I) = \mathsf{output}(V)(J)$ and consequently $\{\mathsf{output}(V)(K) \mid K \in \mathrm{out}_V\} \subseteq \{\mathsf{output}(V)(K) \mid K \in (\mathrm{in}_V + o)\}$. Since $\mathsf{output}(V) : (\mathrm{in}_V + o) \hookrightarrow \mathbb{Z}$, we conclude that $\mathsf{output}(V) : \mathrm{out}_V \hookrightarrow \mathbb{Z}$. □

The following theorem is a straightforward generalisation of Thm. 2.

Theorem. 3 *Let* V *and* W *be two variables such that* $\vartheta_V = m\vartheta_W$ $(m \in \mathbb{Q})$.

(1) If $\mathrm{in}_V \overset{\vartheta_V}{\longmapsto} \mathrm{in}_W$, then $(\mathsf{input}(W) : \mathrm{in}_W \hookrightarrow \mathbb{Z}) \implies (\mathsf{input}(V) : \mathrm{in}_V \hookrightarrow \mathbb{Z})$
(2) If $\mathrm{in}_V \overset{\vartheta_V}{\longmapsto} \mathrm{out}_W$, then $(\mathsf{output}(W) : \mathrm{out}_W \hookrightarrow \mathbb{Z}) \implies (\mathsf{input}(V) : \mathrm{in}_V \hookrightarrow \mathbb{Z})$
(3) If $\mathrm{out}_V \overset{\vartheta_V}{\longmapsto} \mathrm{in}_W$, then $(\mathsf{input}(W) : \mathrm{in}_W \hookrightarrow \mathbb{Z}) \implies (\mathsf{output}(V) : \mathrm{out}_V \hookrightarrow \mathbb{Z})$
(4) If $\mathrm{out}_V \overset{\vartheta_V}{\longmapsto} \mathrm{out}_W$, then $(\mathsf{output}(W) : \mathrm{out}_W \hookrightarrow \mathbb{Z}) \implies (\mathsf{output}(V) : \mathrm{out}_V \hookrightarrow \mathbb{Z})$

6. Transforming the computation into communication constraint

If there exists a moving variable V such that $\Phi \xrightarrow{\vartheta_V} \mathrm{in}_V$, then the computation constraint can be eliminated (Thm. 1). Otherwise, we replace the computation constraint by the following communication constraint: We imagine there exists a pseudo-variable, denoted Γ, such that $\Phi \xrightarrow{\vartheta_\Gamma} \mathrm{in}_\Gamma$. The dependence vector ϑ_Γ of the pseudo-variable Γ can be chosen to be any vector in $\mathbb{Z}^n$. The input space in_Γ is defined as

$$\mathrm{in}_\Gamma \quad = \quad (\Phi - \vartheta_\Gamma) \setminus \Phi$$

($\Phi - \vartheta_\Gamma$ is the translation of Φ by $-\vartheta_\Gamma$.) The very definition of in_Γ ensures that $\Phi \xrightarrow{\vartheta_\Gamma} \mathrm{in}_\Gamma$. In general, it is preferable to choose ϑ_Γ as a dependence vector, say ϑ_V. Since $\mathrm{in}_\Gamma \xmapsto{\vartheta_\Gamma} \mathrm{in}_V$ and $\mathrm{in}_\Gamma \xmapsto{\vartheta_\Gamma} \mathrm{out}_V$, the communication constraint for variable V can be eliminated (Thm. 3).

7. Transforming the communication constraint

As the central part of the paper, this section is concerned with the derivation of the closed-form necessary and sufficient conditions for mapping three-dimensional algorithms into linear arrays without communication conflicts (i.e., data link collisions).

The variables in systolic algorithms represent either the matrices or the vectors in the corresponding problem specifications. These matrices are usually diagonal, rectangular or triangular matrices. Vectors can be dealt with similarly as diagonal matrices. The type of a matrix determines the shape of the input and output spaces of the corresponding variable, which consequently dictates the derivation of the closed-form communication constraint. For reasons of symmetry, we only consider the input spaces. Often we shall speak of the communication constraint for a variable rather than for the input space of a variable. All results presented in this section apply for the output spaces.

We shall use the following notations. If x and y are integers, not both zero, then their *greatest common divisor*, $\gcd(x,y)$, is the largest positive integer that divides both x and y. We define $\gcd(0,0)=0$. The sign is the usual sign function. Let P and Q be two points in $\mathbb{Z}^n$. The notation $\overline{PQ}$ stands for the line connecting the two points P and Q, inclusive. Let V be a variable. Let e_V^s be a vector in $\mathbb{Z}^n$ where s is in the set $\{x,y,z\}$. We write δ_V^s for the time difference between the input of two input data $V(I+e_V^s)$ and $V(I)$:

$$\delta_V^s \quad = \quad \mathsf{input}(V)(I+e_V^s) - \mathsf{input}(V)(I) \quad = \quad \lambda e_V^s - \sigma e_V^s(\lambda\vartheta_V/\sigma\vartheta_V)$$

7.1. Lines

The input space in_V is a *line* if it is one-dimensional. Let e_V^x be a vector parallel to the line. The closed-form communication constraint follows from the definition of δ_V^x.

Theorem. 4 *Let* in_V *be a line. The communication constraint for* V *is satisfied iff* $\delta_V^x \neq 0$.

7.2. Parallelograms

The input space in_V is a *parallelogram* if it satisfies the following property (Fig. 2): It contains four distinct index points, denoted by O, P, Q and R. We write num_V^x and num_V^y for the number of index points on the edges $\overline{OP}$ and $\overline{OQ}$, respectively. We define $e_V^x = (P-O)/(\mathrm{num}_V^x - 1)$ and $e_V^y = (Q-O)/(\mathrm{num}_V^y - 1)$. Then

$$\mathrm{in}_V \quad = \quad \{O + (i-1)e_V^y + (j-1)e_V^x \mid 0 < i \leq \mathrm{num}_V^y \wedge 0 < j \leq \mathrm{num}_V^x\}$$

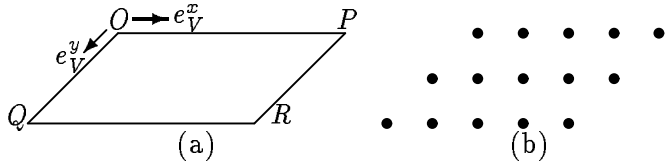

Figure 2: (a) The parallelepiped input space. (b) An instance: $\mathrm{num}_V^x = 5$ and $\mathrm{num}_V^y = 3$.

That is, O, P, Q and R are the four vertices of the parallelogram and e_V^x and e_V^y are the vectors parallel to two adjacent edges. For ease of presentation, we define a matrix, named Δ_V and called the *input matrix*, of size $\mathrm{num}_V^y \times \mathrm{num}_V^x$ as follows:

$$\Delta_V \;=\; \begin{bmatrix} 0 & \delta_V^x & \cdots & (\mathrm{num}_V^x - 1)\delta_V^x \\ \delta_V^y & \delta_V^y + \delta_V^x & \cdots & \delta_V^y + (\mathrm{num}_V^x - 1)\delta_V^x \\ & & \cdots & \\ (\mathrm{num}_V^y - 1)\delta_V^y & (\mathrm{num}_V^y - 1)\delta_V^y + \delta_V^x & \cdots & (\mathrm{num}_V^y - 1)\delta_V^y + (\mathrm{num}_V^x - 1)\delta_V^x \end{bmatrix}$$

The (i,j)-th element of the input matrix, denoted $\Delta_V^{i,j}$, is the time difference between the input of $V(O)$ and $V(O + (i-1)e_V^y + (j-1)e_V^x)$, which is calculated as

$$\Delta_V^{i,j} \;=\; \mathsf{input}(V)(O + (i-1)e_V^y + (j-1)e_V^x) - \mathsf{input}(V)(O) \;=\; (i-1)\delta_V^y + (j-1)\delta_V^x$$

So the communication constraint for V is satisfied iff Δ_V contains no identical elements.

To ensure that all elements of Δ_V are distinct, the necessary and sufficient conditions must include $\delta_V^x \neq 0$ and $\delta_V^y \neq 0$. To derive the remaining conditions, we divide all input matrices into four groups by the case analysis: (1) $\delta_V^x > 0$ and $\delta_V^y > 0$, (2) $\delta_V^x > 0$ and $\delta_V^y < 0$, (3) $\delta_V^x < 0$ and $\delta_V^y > 0$ and (4) $\delta_V^x < 0$ and $\delta_V^y < 0$. We examine two input matrices from the first two groups. The analyses in (3) and (4) are similar to (1) and (2), respectively.

$$\Delta_V \;=\; \begin{bmatrix} 0 & 3 & 6 & 9 & 12 \\ 1 & 4 & 7 & 10 & 13 \\ 2 & 5 & 8 & 11 & 14 \end{bmatrix} \qquad \Delta_V \;=\; \begin{bmatrix} 0 & 2 & 4 \\ -4 & -2 & 0 \\ -8 & -6 & -4 \end{bmatrix}$$

(a) $\mathrm{num}_V^x = 5, \mathrm{num}_V^y = 3, \delta_V^x = 3, \delta_V^y = 1$ (b) $\mathrm{num}_V^x = 3, \mathrm{num}_V^y = 3, \delta_V^x = 2, \delta_V^y = -4$

The communication constraint in (a) is satisfied, since Δ_V has no identical elements. In general, since $\delta_V^x > 0$ and $\delta_V^y > 0$, all elements of the input matrix are distinct iff all elements in its first row and first column are distinct. If we scan the first row left to right and the first column top to bottom, the first pair of identical elements must be at the positions

$$(1, \frac{\delta_V^y}{\gcd(\delta_V^x, \delta_V^y)} + 1) \;=\; (1,2) \;\text{ and }\; (\frac{\delta_V^x}{\gcd(\delta_V^x, \delta_V^y)} + 1, 1) \;=\; (4,1)$$

and have the value

$$\Delta_V^{1,1} + \mathrm{sign}(\delta_V^x) \times \frac{\delta_V^x \delta_V^y}{\gcd(\delta_V^x, \delta_V^y)} \;=\; 3$$

In general, the input matrix does not contain identical elements iff

$$\frac{\delta_V^y}{\gcd(\delta_V^x, \delta_V^y)} \;\geq\; \mathrm{num}_V^x \;\text{ or }\; \frac{\delta_V^x}{\gcd(\delta_V^x, \delta_V^y)} \;\geq\; \mathrm{num}_V^y$$

The input matrix in (a) has no identical elements, because $3 = \frac{\delta_V^x}{\gcd(\delta_V^x, \delta_V^y)} \geq \mathrm{num}_V^y = 3$ holds.

Next, we consider the input matrix in (b). Since the input matrix contains identical elements 0 and -4, the communication constraint is violated. In general, since $\delta_V^x > 0$ and $\delta_V^y < 0$, all elements of the input matrix are distinct iff all elements in its first row and last column are distinct. If we scan the first row right to left and the last column top to bottom, the first pair of identical elements must be at the positions

$$(1, \mathrm{num}_V^x - \frac{-\delta_V^y}{\gcd(\delta_V^x, \delta_V^y)}) \;=\; (1,1) \;\text{ and }\; (\frac{\delta_V^x}{\gcd(\delta_V^x, \delta_V^y)} + 1, \mathrm{num}_V^x) \;=\; (2,3)$$

and have the value

$$\Delta_V^{1,\mathrm{num}_V^x} + \mathrm{sign}(\delta_V^x) \times \frac{\delta_V^x \delta_V^y}{\gcd(\delta_V^x, \delta_V^y)} \;\;=\;\; 0$$

In general, the input matrix does not contain identical elements iff

$$-\frac{\delta_V^y}{\gcd(\delta_V^x, \delta_V^y)} \;\geq\; \mathrm{num}_V^x \;\text{ or }\; \frac{\delta_V^x}{\gcd(\delta_V^x, \delta_V^y)} \;\geq\; \mathrm{num}_V^y$$

The input matrix in (b) contains identical elements, because neither $2 = -\frac{\delta_V^y}{\gcd(\delta_V^x, \delta_V^y)} \geq \mathrm{num}_V^x = 3$ nor $1 = \frac{\delta_V^x}{\gcd(\delta_V^x, \delta_V^y)} \geq \mathrm{num}_V^y = 3$ holds.

Theorem. 5 *Let* in_V *be a parallelogram. The communication constraint for* V *is satisfied iff*

$$\delta_V^x \neq 0 \text{ and } \max\left(\frac{|\delta_V^x|}{\mathrm{num}_V^y}, \frac{|\delta_V^y|}{\mathrm{num}_V^x}\right) \geq \gcd(\delta_V^x, \delta_V^y) \qquad (1)$$

Proof. The conditions of (1) imply $\delta_V^y \neq 0$. The proof is conducted by the case analysis: (1) $\delta_V^x \delta_V^y > 0$ and (2) $\delta_V^x \delta_V^y < 0$.

- $\delta_V^x \delta_V^y > 0$. The input matrix Δ_V contains identical elements iff there are identical elements in its first row and first column. If we scan the first row left to right and the first column top to bottom, the first pair of identical elements are located at the positions $(1, \frac{|\delta_V^y|}{\gcd(\delta_V^x, \delta_V^y)} + 1)$ and $(\frac{|\delta_V^x|}{\gcd(\delta_V^x, \delta_V^y)} + 1, 1)$ and have the value $\Delta_V^{1,1} + \mathrm{sign}(\delta_V^x) \times \frac{\delta_V^x \delta_V^y}{\gcd(\delta_V^x, \delta_V^y)}$. Hence, the input matrix Δ_V does not contain identical elements iff either $\frac{|\delta_V^y|}{\gcd(\delta_V^x, \delta_V^y)} \geq \mathrm{num}_V^x$ or $\frac{|\delta_V^x|}{\gcd(\delta_V^x, \delta_V^y)} \geq \mathrm{num}_V^y$, which is equivalent to $\max\left(\frac{|\delta_V^x|}{\mathrm{num}_V^y}, \frac{|\delta_V^y|}{\mathrm{num}_V^x}\right) \geq \gcd(\delta_V^x, \delta_V^y)$.
- $\delta_V^x \delta_V^y < 0$. The proof is similar to the first case (see the forgoing discussions). □

7.3. Trapezoids

The input space in_V is a *trapezoid* if it satisfies the following properties (Fig. 3):

1. Four index points, denoted O, P, Q and R, are the vertices of the trapezoid. Q and R may be identical.
2. $\overline{OP}$ is parallel to and contains no less index points than $\overline{QR}$.
3. Let num_V^x, num_V^y and num_V^z be the number of index points on $\overline{OP}$, $\overline{OQ}$ and $\overline{PR}$, respectively. Then $\mathrm{num}_V^y = \mathrm{num}_V^z$.

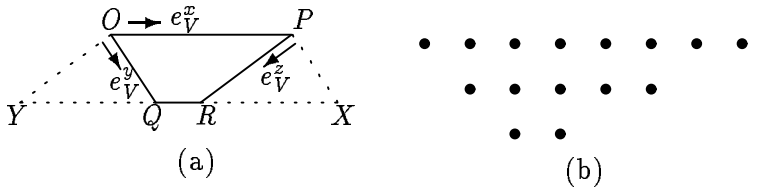

Figure 3: (a) The trapezoidal input space. (b) An instance: $\text{num}_V^x = 8$ and $\text{num}_V^y = \text{num}_V^z = 3$.

4. Let $e_V^x = (P-O)/(\text{num}_V^x - 1)$, $e_V^y = (Q-O)/(\text{num}_V^y - 1)$ and $e_V^z = (R-P)/(\text{num}_V^z - 1)$. (By using the conditions given so far, $e_V^y - e_V^z$ and e_V^x are co-linear.) Then

$$\text{in}_V \quad = \quad \{O + (i-1)e_V^y + (j-1)e_V^x \mid 0 < i \leq \text{num}_V^y \wedge 0 < j \leq \text{num}_V^x - (i-1)\frac{e_V^y - e_V^z}{e_V^x}\}$$

A trapezoid is a *triangle* if $Q = R$ and a parallelogram if $e_V^y = e_V^z$.

By definition, if the input space in_V is a trapezoid, $e_V^y - e_V^z$ must be an *integral* positive multiple of e_V^x. This multiple is the difference of the index point counts on any two adjacent line segments in the trapezoid that contain index points and are parallel to $\overline{OP}$.

Let X be the point such that $\overline{OP}$ and $\overline{QX}$ are parallel and $\overline{OQ}$ and $\overline{PX}$ are parallel. Let Y be the point such that $\overline{OP}$ and $\overline{YR}$ are parallel and $\overline{OY}$ and $\overline{PR}$ are parallel. Then, the trapezoidal input space in_V is the intersection of the parallelogram $OPXQ$ and the parallelogram $OPRY$. Therefore, the input matrix Δ_V for the trapezoidal input space in_V can be defined as either of the following:

(P1) Δ_V is the input matrix for the parallelogram $OPXQ$ with the last $\text{num}_V^x - (i-1)\frac{e_V^y - e_V^z}{e_V^x}$ elements in the i-th row being removed.

(P2) Δ_V is the input matrix for the parallelogram $OPRY$ with the first $\text{num}_V^x - (i-1)\frac{e_V^y - e_V^z}{e_V^x}$ elements in the i-th row are removed.

As is for the parallelepiped input space, the necessary and sufficient conditions must include $\delta_V^x \neq 0$, $\delta_V^y \neq 0$ and $\delta_V^z \neq 0$ if all elements of the input matrix are distinct. To derive the remaining conditions, we distinguish the two cases: (1) $\delta_V^x \delta_V^y > 0$ and (2) $\delta_V^x \delta_V^y < 0$. In the first case, the input matrix is regarded as being defined in (P1). The transformation of the communication constraint is carried out as if the input space is the parallelogram $OPXQ$. That is, the communication constraint is satisfied iff all elements of the first row and first column of the input matrix are distinct, i.e., iff

$$\max\left(\frac{|\delta_V^x|}{\text{num}_V^y}, \frac{|\delta_V^y|}{\text{num}_V^x}\right) \geq \gcd(\delta_V^x, \delta_V^y)$$

In the second case, the input matrix is regarded as being defined in (P2). The transformation of the communication constraint is carried out as if the input space is the parallelogram $OPRY$. That is, the communication constraint is satisfied iff all elements of the first row and last column of the input matrix are distinct, i.e., iff

$$\max\left(\frac{|\delta_V^x|}{\text{num}_V^z}, \frac{|\delta_V^z|}{\text{num}_V^x}\right) \geq \gcd(\delta_V^x, \delta_V^z)$$

Theorem. 6 *Let* in_V *be a trapezoid. The communication constraint for* V *is satisfied iff*

$$\begin{aligned} \delta_V^x \neq 0 \quad \text{and} \quad \delta_V^x \delta_V^y > 0 \implies \max\left(\frac{|\delta_V^x|}{\mathrm{num}_V^y}, \frac{|\delta_V^y|}{\mathrm{num}_V^x}\right) \geq \gcd(\delta_V^x, \delta_V^y) \\ \text{and} \quad \delta_V^x \delta_V^y < 0 \implies \max\left(\frac{|\delta_V^x|}{\mathrm{num}_V^z}, \frac{|\delta_V^z|}{\mathrm{num}_V^x}\right) \geq \gcd(\delta_V^x, \delta_V^z) \end{aligned} \tag{4}$$

Proof. The conditions of (4) imply that $\delta_V^y \neq 0$ and $\delta_V^z \neq 0$. The rest of the proof follows from the forgoing discussions. □

Thm. 6 has a special case in which δ_V^z is not needed.

Theorem. 7 *Let* in_V *is a trapezoid such that* $e_V^x + e_V^z = e_V^y$. *The communication constraint for* V *is satisfied iff*

$$\delta_V^x \neq 0 \text{ and } \max\left(\frac{|\delta_V^x|}{\mathrm{num}_V^y}, \frac{|\delta_V^y|}{\mathrm{num}_V^x}, \frac{|\delta_V^y - \delta_V^x|}{\mathrm{num}_V^x}\right) \geq \gcd(\delta_V^x, \delta_V^y)$$

7.4. Arbitrary input spaces

This section is very brief due to space limitations. We distinguish the two cases:

1. The input space in_V is a two-dimensional set. First, construct a minimal enclosing parallelogram or trapezoid of the input space. Then, apply the respective method for parallelograms (Sect. 7.2) or for trapezoids (Sect. 7.3).
2. The input space in_V is a three-dimensional set. First, project the input space along ϑ_V onto an arbitrarily chosen hyperplane (the projected input space is two-dimensional). Second, scale the projected input space such that the *scaled input space* is integral. Finally, we are back to the first case.

It makes no difference onto which hyperplane the input space is projected.

Theorem. 8 *The communication constraint for the input space of* V *is equivalent to the communication constraint for the scaled input space of* V.

Proof. A simple algebraic manipulation. □

8. A summary

Let us summarise the process for the derivation of the closed-form necessary and sufficient conditions on computation and communication conflict-free mappings. Let *IOspace* be the set of all input and output spaces, which are referred to as the *spaces* below.

1. Transform all spaces in *IOspace* into parallelograms or trapezoids (Sect. 7.4).
2. Transform the computation constraint (Sect. 6). Eliminate the computation constraint if possible, or replace it by the communication constraint for an imaginary variable Γ by adding the minimal enclosing parallelogram or trapezoid of in_Γ to *IOspace*.
3. Eliminate all redundant spaces in *IOspace* (Thms. 2 and 3).
4. Transform the communication constraints for the spaces in *IOspace* (Thms. 4 – 7).

To find optimal and conflict-free mappings for linear arrays, we solve the following:

Minimise	$\mathrm{cost}(\Pi)$
Subject to	The precedence constraint
	The delay constraint
	The transformed communication constraint

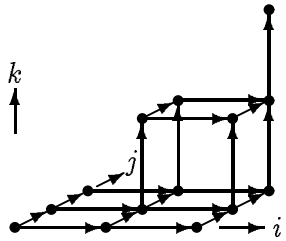

Figure 4: The dependence graph for LU decomposition ($m=3$).

where cost(Π) denotes some cost function (e.g., the latency) for a space-time mapping.

9. An example and related work

To compare our work with the related work, let us consider LU decomposition. LU decomposition is the unique decomposition of a non-singular $m \times m$ matrix A into a lower-triangular matrix L and an upper-triangular matrix U such that $A = LU$. The elements of the upper triangle of L and the elements of the lower triangle U are 0; the diagonal elements of L are 1. An algorithm for LU decomposition can be found in [7]. The corresponding dependence graph is displayed in Fig. 4. There are three variables A, L and U which are associated with the following dependence vectors:

$$\vartheta_A = (0,0,1), \quad \vartheta_L = (0,1,0), \quad \vartheta_U = (1,0,0)$$

The input and output spaces are:

$$\begin{aligned} \text{in}_A &= \{(i,j,0) \mid 0 < i,j \leq m\} \\ \text{out}_L &= \{(i,m,k) \mid k < i \leq m \wedge 0 < k \leq m\} \\ \text{out}_U &= \{(m,j,k) \mid k \leq j \leq m \wedge 0 < k \leq m\} \end{aligned}$$

The output space of A and the input spaces of L and U are of no interest.

Since $\Phi \xrightarrow{\vartheta_A} \text{in}_A$, the computation constraint can be eliminated (Thm. 1). The input space in_A is a parallelogram: $e_A^x = (1,0,0)$, $e_A^y = (0,1,0)$ and $\text{num}_A^x = \text{num}_A^y = m$. The output space out_L is a triangle: $e_L^x = (-1,0,0)$, $e_L^y = (0,0,1)$, $e_L^z = (1,0,1)$ and $\text{num}_L^x = \text{num}_L^y = \text{num}_L^z = m-1$. The output space out_U is a triangle: $e_U^x = (0,-1,0)$, $e_U^y = (0,0,1)$, $e_U^z = (0,1,1)$ and $\text{num}_U^x = \text{num}_U^y = \text{num}_U^z = m$. We obtain

$$\begin{array}{lll} \delta_A^x = \lambda_1 - \sigma_1 \frac{\lambda_3}{\sigma_3} & \delta_L^x = \sigma_1 \frac{\lambda_2}{\sigma_2} - \lambda_1 & \delta_U^x = \sigma_2 \frac{\lambda_1}{\sigma_1} - \lambda_2 \\ \delta_A^y = \lambda_2 - \sigma_2 \frac{\lambda_3}{\sigma_3} & \delta_L^y = \lambda_3 - \sigma_3 \frac{\lambda_2}{\sigma_2} & \delta_U^y = \lambda_3 - \sigma_3 \frac{\lambda_1}{\sigma_1} \\ & \delta_L^z = (\lambda_1 + \lambda_3) - (\sigma_1 + \sigma_3) \frac{\lambda_2}{\sigma_2} & \delta_U^z = (\lambda_2 + \lambda_3) - (\sigma_2 + \sigma_3) \frac{\lambda_1}{\sigma_1} \end{array}$$

The communication constraint for the input space in_A is given in Thm. 5. Since $e_L^x + e_L^z = e_L^y$ and $e_U^x + e_U^z = e_U^y$, the communication constraints for out_L and out_U are given in Thm. 7. (In this case, δ_L^z and δ_U^z are not needed.)

The problem of finding optimal and conflict-free mappings can be formulated as follows:

Minimise cost(Π)

Subject to $\lambda_1 > 0$, $\lambda_2 > 0$, $\lambda_3 > 0$ (Precedence Constraint)

$\sigma_1 | \lambda_1$, $\sigma_2 | \lambda_2$, $\sigma_3 | \lambda_3$ (Delay Constraint)

$\delta_A^x \neq 0$, $\max(|\delta_A^x|, |\delta_A^y|) \geq m \gcd(\delta_A^x, \delta_A^y)$,
$\delta_L^x \neq 0$, $\max(|\delta_L^x|, |\delta_L^y|, |\delta_L^x - \delta_L^y|) \geq (m-1) \gcd(\delta_L^x, \delta_L^y)$,
$\delta_U^x \neq 0$, $\max(|\delta_U^x|, |\delta_U^y|, |\delta_U^x - \delta_U^y|) \geq m \gcd(\delta_U^x, \delta_U^y)$
(Communication Constraint)

By using Lee and Kedem's mapping conditions, the problem formulation would be:

$$
\begin{array}{rll}
\textbf{Minimise} & \text{cost}(\Pi) & \\
\textbf{Subject to} & \lambda_1 > 0,\ \lambda_2 > 0,\ \lambda_3 > 0 & \text{(Precedence Constraint)} \\
& \sigma_1 | \lambda_1,\ \sigma_2 | \lambda_2,\ \sigma_3 | \lambda_3 & \text{(Delay Constraint)} \\
& (\Pi : \Phi \hookrightarrow \mathbb{Z}^2) & \text{(Computation Constraint)} \\
& (\forall\ I, J : I, J \in \Phi : \neg(I \xleftrightarrow{\vartheta_A} J) \implies \lambda(I-J)\sigma\vartheta_A \neq \lambda\vartheta_A\sigma(I-J)) & \\
& (\forall\ I, J : I, J \in \Phi : \neg(I \xleftrightarrow{\vartheta_L} J) \implies \lambda(I-J)\sigma\vartheta_L \neq \lambda\vartheta_L\sigma(I-J)) & \\
& (\forall\ I, J : I, J \in \Phi : \neg(I \xleftrightarrow{\vartheta_U} J) \implies \lambda(I-J)\sigma\vartheta_U \neq \lambda\vartheta_U\sigma(I-J)) & \\
& & \text{(Communication Constraint)}
\end{array}
$$

The computation constraint is redundant. In addition, both the computation and communication constraints depend on the index vectors and do not have closed-form expressions. Thus, the total number of constraints depends on the problem size. By using integer programming techniques, only problems of small sizes can be solved efficiently.

It is unclear how Ganapathy and Wah's method [1] can be used to obtain the problem formulation same as ours. Ganapathy and Wah's communication constraint [1] improves Lee and Kedem's [2] because the index vectors in the latter are replaced by a few parameters that depend on the problem-size parameters. But the issue of defining these parameters in general and then coming up with closed-form constraints was not addressed. By classifying the input and output spaces in terms of their shapes, this issue has been successfully addressed here for the case where three-dimensional algorithms are mapped to linear arrays.

Finally, Ganapathy and Wah proposed an enumeration method to systematically generate optimal mappings – the scheduling vector first followed by the allocation vector. One component in their objective functions is the completion time, which is taken as the time spent on performing all computations of an algorithm. It depends on the scheduling vector only. In lower dimensional (pure) systolic arrays, the time spent on the soaking (draining) of the input (output) data can be significant. If this is taken into account, the completion time also depends on the allocation vector. Then a new enumeration method is called for.

10. Acknowledgement

Thanks to the referees for helpful comments and to J. A. B. Fortes for discussions.

References

[1] K. Ganapathy and B. W. Wah. Synthesizing optimal lower dimensional processor arrays. In *Int. Conf. on Parallel Processing*, pages 96–103, Aug. 1992.

[2] P. Lee and Z. Kedem. Mapping nested loop algorithms into multidimensional systolic arrays. *IEEE Trans. on Parallel and Distributed Systems*, 1:64–76, Jan. 1990.

[3] P. Quinton and V. van Dongen. The mapping of linear recurrence equations on regular arrays. *J. VLSI Signal Processing*, 1(2):95–113, Oct. 1989.

[4] S. K. Rao. *Regular Iterative Algorithms and their Implementations on Processor Arrays.* PhD thesis, Department of Electrical Engineering, Stanford University, Oct. 1985.

[5] V.P. Roychowdhury and T. Kailath. Subspace scheduling and parallel implementation of non-systolic regular iterative algorithms. *J. VLSI Signal Processing*, 1(2):127–142, Oct. 1989.

[6] W. Shang and W. A. B. Fortes. On time mapping of uniform dependence algorithms into lower dimensional processor arrays. *IEEE Trans. on Parallel and Distributed Systems*, 3(3):350–363, May 1992.

[7] J. Xue and C. Lengauer. The synthesis of control signals for one-dimensional systolic arrays. *Integration*, 14(1):1–32, Nov. 1992.