LFCS

Laboratory for Foundations of Computer Science
Department of Computer Science - University of Edinburgh

On the Description and Development of One-Dimensional Systolic Arrays

On the Description and Development of One-Dimensional Systolic Arrays

by

Jingling Xue and Christian Lengauer

LFCS Report Series ECS-LFCS-90-116

LFCS
Department of Computer Science
University of Edinburgh
The King's Buildings
Edinburgh EH9 3JZ

July 1990

On the Description and Development of One-Dimensional Systolic Arrays

JINGLING XUE[0] AND CHRISTIAN LENGAUER

LABORATORY FOR FOUNDATIONS OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF EDINBURGH
EDINBURGH, SCOTLAND

ECS–LFCS–90–116
16 JULY 1990

Abstract

Previous work on the mapping of uniform recurrence equations to one-dimensional systolic arrays is extended. A number of properties of such mappings are stated and proved. Previously known mapping constraints are simplified and reduced. A previously known mapping algorithm is improved.

[0]Supported by an Overseas Research Students Award and a University of Edinburgh Postgraduate Fellowship.

Contents

1 Introduction

A systolic array is a collection of processors (or "cells") that are locally and regularly connected. This special-purpose computer architecture happens to support the parallel implementations of highly iterative algorithms in a variety of areas such as numerical analysis, signal or image processing and graph theory. These algorithms are often specified by uniform recurrence equations (UREs) [5] but can also be specified differently, such as by nested loops [8].

A number of synthesis methods have been proposed for mapping UREs [1, 11, 13, 16, 19] or loops [2, 3, 9, 10] to systolic arrays. They map the computations prescribed by n-dimensional UREs (or loops) to $(n-r)$-dimensional space and r-dimensional time, where r is chosen freely $(0<r<n)$.

When mapping n-dimensional UREs into a space of less than $n-1$ dimensions $(r>1)$, one must consider a multi-dimensional time domain. Wong [19] points out the possibility of implementing multi-dimensional time using a multi-dimensional clock. However, multi-dimensional clocks are costly and may operate inefficiently. Therefore one transforms multi-dimensional time into one-dimensional time. We distinguish two methods: one approaches a single-dimensional time domain directly [9, 14], the other does so via a multi-dimensional time domain [4, 16, 19].

In the synthesis method proposed by Lee and Kedem [9], one-dimensional time is derived directly. A two-dimensional space-time mapping, with one-dimensional time and one-dimensional space, is created by formulating a set of necessary and sufficient conditions. Various user-specified constraints and cost criteria can be taken into account during the search. However, the proof of this is not constructive and properties of the resulting space-time mappings are not extensively studied.

The second approach is represented by the synthesis method due to Rao, Jagadish and Wong [4, 16, 19]. In this method, one derives first an r-dimensional time and, if $r>1$, reduces the number of time dimensions successively. Rao presents an algorithm based on integer programming for mapping n-dimensional UREs into arrays of lower dimensionality. Similar ideas are also described in [4, 19]. In the derivation, resource limits like bounds on the number of cells, restrictions on cell connections and so on are not being considered.

In practice, one-dimensional arrays appear particularly attractive for several reasons. It has been argued that one-dimensional arrays have advantages such as 100% utilization of non-faulty cells on a wafer and a clock rate that is independent of the size of the array [6, 7, 14]. Also, one-dimensional arrays can be given a constant I/O bandwidth by requiring that I/O only be performed at the two border cells [7].

This paper is concerned with mapping UREs to one-dimensional arrays. We extend the work presented in [9]. Our example is matrix multiplication. In Sect. 2, we define the class of UREs that we are interested in. Sect. 3 presents two models of systolic arrays: one general model and one model of one-dimensional arrays. Sect. 4 contains a brief discussion of the main tool for mapping UREs to systolic arrays – the space-time mapping – and its properties. In Sect. 5, we derive one-dimensional time directly. We start with four conditions, due to Lee and Kedem [9], that are necessary and sufficient for the existence of a valid space-time mapping: the precedence constraint, the delay constraint, the computation constraint and the communication constraint. We simplify the communication constraint and show that it implies the computation constraint. In Sect. 6, we first derive $(n-1)$-dimensional time and then reduce it to one-dimensional time. A range of properties are proved that characterize the space-time mapping. It is shown that a proper extension and scaling of the index space guarantees that the computation constraint implies the communication constraint. This leads to a more constructive derivation of a valid space-time mapping if the precedence constraint can be satisfied. Sect. 7 evaluates the synthesis methods presented in Sects. 5 and 6.

2 Uniform Recurrence Equations

In what follows, the symbols $\mathbb{Z}$ and $\mathbb{Q}$ denote the set of integers and rationals. $\mathbb{Z}^+$ and $\mathbb{Q}^+$ denote the set of positive integers and rationals, $\mathbb{Z}_0^+$ and $\mathbb{Q}_0^+$ the non-negative integers and rationals. $\mathbb{Z}^n$ and $\mathbb{Q}^n$ denote the n-fold Cartesian product of $\mathbb{Z}$ and $\mathbb{Q}$. Following Quinton [12], we write a URE in the format: domain predicate $\longrightarrow$ recurrence equation.

Definition 1 A *system of uniform recurrence equations* consists of a number of equations each of which is of the following form:

$$I \in \Phi \rightarrow v(I) \quad = \quad f_v(w(I-\theta_{vw}), \ldots)$$

where:

- $I \in \Phi \subset \mathbb{Z}^n$.
- Φ is referred to as the *domain of computation* and is a set of integral points belonging to a bounded convex polyhedron of $\mathbb{Z}^n$ within which the system of UREs is defined.
- v and w are variable names belonging to a finite set V, $v(I)$ is called the *result* and $w(I-\theta_{vw})$ is called its *argument*. Each variable is defined at every integral point of Φ and takes on a unique value.
- The "..." indicates that there can be additional arguments of the same form.
- θ_{vw} is a constant integer vector of length n, called a *dependence vector*. It is defined as the difference between the index vectors of the result $v(I)$ and the argument $w(I-\theta_{vw})$.
- f_v is a k-ary function that is strictly dependent on each of its arguments [5, Sect. 2].

(*End of Definition*)

The data dependences in a URE can be represented by a dependence graph. A *dependence graph* has one node for each point of the domain and a directed arc from node J to node I if and only if a variable indexed by J is an argument in the equation for a variable indexed by I.

Data dependences can also be represented by a *dependence matrix* $D \in \mathbb{Z}^{n \times k}$; the columns of D are the dependence vectors; θ_i is the i-th column (we write $\theta_i \in D$).

Dependence vectors are associated with a variable name. For simplicity, we assume that, for each variable name $v \in V$, there is only one associated dependence vector, which is denoted θ_v, in D. Alternatively, for a dependence vector θ_i, we denote the corresponding variable name by v_i. We distinguish the variables that hold the input values and the variables that hold the output values.

Definition 2 The sets IN_v of *input variables* and OUT_v of *output variables* of a stream v are defined as follows:

$$\begin{aligned} IN_v &= \{v(I) \mid I \notin \Phi, J \in \Phi, J = I+\theta_v\} \\ OUT_v &= \{v(I) \mid I \in \Phi, J \notin \Phi, J = I+\theta_v\} \end{aligned}$$

(*End of Definition*)

We refer to v as the *stream* associated with dependence vector θ_v, to $v(I)$ $(I \in \Phi \cup IN_v)$ as the variables constituting the stream and to the set of index vectors of input variables as the *domain of input variables*: $\Omega = \{I \mid \forall\ v : v \in V : v(I) \in IN_v\}$.

Example: Matrix Multiplication

Let us now use the multiplication of $n \times n$ matrices as an example to illustrate some concepts presented in Def. 1. This example will be used for illustration throughout the paper.

Specification:

$$(\forall\ i,j : 0 < i,j \leq m :\ c_{i,j} = \sum_{k=1}^{m} a_{i,k} b_{k,j})$$

UREs:

$$\begin{array}{llll}
0<i\leq m, & 0<j\leq m, & k=m & \rightarrow c_{i,j} = C(i,j,k) \\
0<i\leq m, & 0<j\leq m, & 0<k\leq m & \rightarrow C(i,j,k) = C(i,j,k-1)+A(i,j-1,k)B(i-1,j,k) \\
0<i\leq m, & 0<j\leq m, & 0=k & \rightarrow C(i,j,k) = 0 \\
0<i\leq m, & 0<j\leq m, & 0<k\leq m & \rightarrow A(i,j,k) = A(i,j-1,k) \\
0<i\leq m, & 0=j, & 0<k\leq m & \rightarrow A(i,j,k) = a_{i,k} \\
0<i\leq m, & 0<j\leq m, & 0<k\leq m & \rightarrow B(i,j,k) = B(i-1,j,k) \\
0=i, & 0<j\leq m, & 0<k\leq m & \rightarrow B(i,j,k) = b_{k,j}
\end{array}$$

Domain of computation:

$$\Phi = \{(i,j,k) \mid 0 < i,j,k \leq m\}$$

Domain of input variables:

$$\Omega = \{(i,0,k) \mid 0<i,k\leq m\} \cup \{(0,j,k) \mid 0<j,k\leq m\} \cup \{(i,j,0) \mid 0<i,j\leq m\}$$

Variable set:

$$V = (A,B,C)$$

Streams:

$$A, B, C$$

Dependence matrix:

$$D = (\theta_A, \theta_B, \theta_C) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Dependence vectors:

$$\theta_A, \theta_B, \theta_C$$

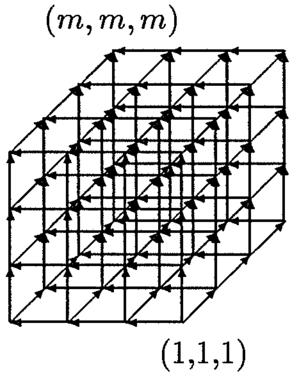
Input variables:

$$\begin{aligned} IN_A &= \{A(i,0,k) \mid 0<i\leq m,\ 0<k\leq m\} \\ IN_B &= \{B(0,j,k) \mid 0<j\leq m,\ 0<k\leq m\} \\ IN_C &= \{C(i,j,0) \mid 0<i\leq m,\ 0<j\leq m\} \end{aligned}$$

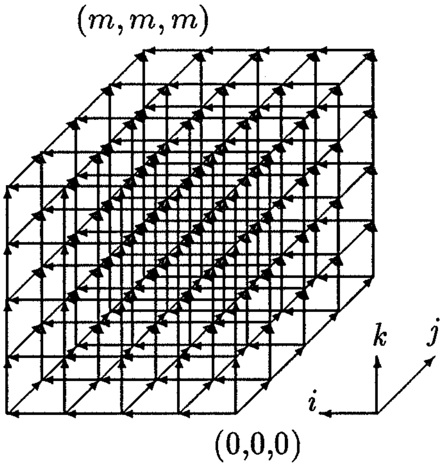
Output variables:

$$\begin{aligned} OUT_A &= \{A(i,m,k) \mid 0<i\leq m,\ 0<k\leq m\} \\ OUT_B &= \{B(m,j,k) \mid 0<j\leq m,\ 0<k\leq m\} \\ OUT_C &= \{C(i,j,m) \mid 0<i\leq m,\ 0<j\leq m\} \end{aligned}$$

Dependence Graph ($n=4$):



domain Φ



domain $\Phi \cup \Omega$

(*End of Example*)

The domain of computation Φ and dependence matrix D provide sufficient information from which systolic arrays can be synthesized. In what follows, we represent a system of UREs by (Φ, D) and presume that D is of size $n \times k$ (also denoted $D^{n\times k}$), i.e., the UREs are n-dimensional and D consists of k dependence vectors of length n.

3 Models of Systolic Arrays

We should talk about space-time mappings in terms of a model of systolic arrays. Actually, we shall refer to two different models (one of which will be an extension of the other). Without loss of generality, we assume that a computation takes one unit of time – this assures that a global clock ticks in unit time.

A simple and general model that has been the basis of many methods [3, 4, 10, 11, 16, 19] is defined in [16]:

Definition 3 (Qualitative Model) A *systolic array* is a network of cells that are placed at the grid points of a finite multi-dimensional lattice $\mathcal{L}$, satisfying the following two properties:

1. Postulate the existence of a directed connection from the cell at location l to the cell at location $l+d$, for some constant vector d. This postulate is either true for all $l \in \mathcal{L}$ or false for all $l \in \mathcal{L}$. A directed connection is also called a *channel*; it is an *input* channel to the cell at its destination and an *output* channel to the cell at its source.

2. If a cell receives a value on an input channel at time t, then it will receive a value on the same channel and send a value on the corresponding output channel at time $t+1$.

(*End of Definition*)

This model only characterizes the qualitative aspects, i.e., the topology and behaviour of the systolic array. Quantitative aspects are induced from the space-time mapping. Resource limits like bounds on the number of cells, restrictions on cell connections and so on are not specified.

When building systolic arrays, one will often want to comply with predefined design constraints. For example, Fortes and Moldovan consider implementations of algorithms on systolic arrays with predefined cell connections [2]. This requires an extension of the qualitative model.

The following quantitative model originates from initial attempts to map homogeneous graphs to one-dimensional arrays [14] and was later adopted by Lee and Kedem [9]. The first two conditions correspond with those of the qualitative model.

Definition 4 (Quantitative Model) A *one-dimensional systolic array* consists of a set of identical cells $\{PE_i \mid 0 < i \leq p\}$. PE_0 is the host computer (Fig. 1).

1. Each cell is connected with its two neighboring cells by a set of channels numbered 1 to k. To each channel j $(0 < j \leq k)$, a sequence of r_j registers $(r_j \in \mathbb{Z}_0^+)$ is connected. The registers function as delay latches in communication; a register retains a datum up to the next clock tick. The set of channels with the same number is referred to as a *link*.

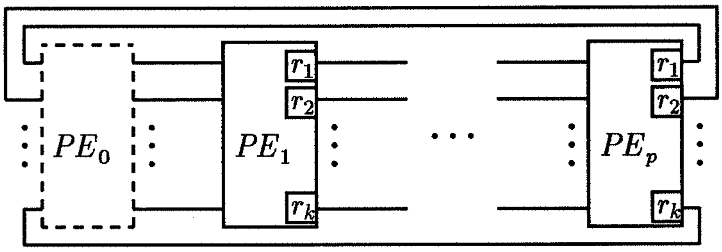


Figure 1: The one-dimensional systolic array model. The larger solid boxes represent the cells. The lines represent connecting channels. A small box inside a cell represent a sequence of delay registers at the respective channel.

2. If a cell PE_i receives a value via link j on its input channel at time t, then it will receive a value on the same channel and send a value on the corresponding output channel at time $t+r_j+1$.

3. I/O is performed only by the two border cells PE_1 and PE_p.

4. Each stream is allocated a distinct link. *(End of Definition)*

4 Space-Time Mappings

This section presents the principle of mapping UREs to the qualitative model of systolic arrays. The mapping assigns temporal and spatial coordinates to each computation in the UREs [10].

Definition 5 Consider the system of UREs (Φ, D). A space-time mapping $\Pi \in \mathbb{Z}^{n\times n}$ that maps a point in Φ to a point in $(n-r)$-dimensional space and r-dimensional time $(0<r<n)$ is given by:

$$\Pi = \begin{bmatrix} \Lambda \\ \Sigma \end{bmatrix} = \begin{bmatrix} \Lambda_1 \\ \vdots \\ \Lambda_r \\ \Sigma_1 \\ \vdots \\ \Sigma_{n-r} \end{bmatrix} = \begin{bmatrix} \Lambda_{1,1} & \Lambda_{1,2} & \cdots \Lambda_{1,n} \\ \vdots & \vdots & \vdots \\ \Lambda_{r,1} & \Lambda_{r,2} & \cdots \Lambda_{r,n} \\ \Sigma_{1,1} & \Sigma_{1,2} & \cdots \Sigma_{1,n} \\ \vdots & \vdots & \vdots \\ \Sigma_{n-r,1} & \Sigma_{n-r,2} & \cdots \Sigma_{n-r,n} \end{bmatrix}$$

1. Λ is the *time matrix*. Given a point $I \in \Phi$, ΛI specifies the time t_I at which the computation at I is to occur:

$$\Lambda I = t_I = \begin{bmatrix} t_1 \\ \vdots \\ t_r \end{bmatrix}$$

2. Σ is the *space matrix*. Given a point $I \in \Phi$, ΣI specifies the location c_I at which the computation at I is to be performed:

$$\Lambda I = c_I = \begin{bmatrix} c_1 \\ \vdots \\ c_{n-r} \end{bmatrix}$$

(*End of Definition*)

The image of dependence matrix D under mapping Π is given by:

$$\Pi D = (\delta_1, \delta_1, \cdots, \delta_k)$$

The image of dependence vector θ_i is given by:

$$\delta_i = \Pi \theta_i = \begin{bmatrix} t_{\delta_i} \\ c_{\delta_i} \end{bmatrix}$$

t_{δ_i} represents the delay in communication of elements of stream v_i between neighbouring cells; c_{δ_i} represents the channel used for the communication.

A space-time mapping is considered *valid* with respect to some model if data are mapped to the places where they are needed and the times when they are needed there [10], and the mapping satisfies the constraints of the model.

Theorem 1 *Satisfaction of the following two constraints is sufficient for the existence of a valid space-time mapping.*

Precedence Constraint: $(\forall\ i : \theta_i \in D : \Lambda\theta_i > \mathbf{0})$, *where "$>$" denotes the lexicographical ordering and* $\mathbf{0}$ *denotes zero vector of length* r.

Computation Constraint: $\text{rank}(\Pi) = n$, *i.e.,* Π *is non-singular.*

(Proof omitted [10].)

The precedence constraint preserves the semantics of the recurrence equations; the computation constraint ensures that no more than one computation is mapped to a cell per time step.

The following procedure converts an $r \times n$ time matrix Λ to a vector λ of length n [16, 19]:

Procedure 1 (Transformation of r-dimensional to one-dimensional time)

INPUT: A domain of computation Φ and a time matrix Λ.

OUTPUT: A vector λ.

1. $(\forall\ i : 0 < i \leq r : (\forall\ I, J : I, J \in \Phi :\ b_i = \max |\Lambda_i(I - J)| + 1))$.
2. $a_r = 1,\ (\forall\ i : 0 < i < r :\ a_i = a_{i+1} b_{i+1})$.
3. $(\forall\ k : 0 < k \leq n :\ \lambda_k = \sum_{i=1}^{r} a_i \Lambda_{i,k})$. (*End of Procedure*)

The transformation of Λ to λ preserves all interesting properties.

Theorem 2

1. $(\forall\, I, J : I, J \in \Phi : \Lambda I = \Lambda J \iff \lambda I = \lambda J)$.
2. $(\forall\, I, J : I, J \in \Phi : \Lambda I < \Lambda J \implies \lambda I < \lambda J)$.
3. $(\forall\, I, J : I, J \in \Phi : \Lambda I = \Lambda J + \Lambda\theta_i \implies \lambda I = \lambda J + \lambda\theta_i)$.

(Proof omitted [19].)

5 Mappings to One-Dimensional Time and Space

This section discusses the properties of space-time mappings with respect to the quantitative model. In our investigations in the rest of the paper, we shall relate space-time mappings exclusively to this model. Let us first introduce our assumptions and notations:

Assumptions

1. $step : \Phi \longrightarrow \mathbb{Z}, \quad step(I) = \lambda I, \quad \lambda \in \mathbb{Z}^n.$
2. $place : \Phi \longrightarrow \mathbb{Z}, \quad place(I) = \sigma I, \quad \sigma \in \mathbb{Z}^n.$
3. $\pi = \begin{bmatrix} \lambda \\ \sigma \end{bmatrix}, \quad (\forall\, i : \theta_i \in D : \sigma\theta_i \neq 0).$
4. $\gcd(\sigma_1, \sigma_2, \cdots, \sigma_n) = 1.$ *(End of Assumptions)*

To simplify matters, we allow that *step* and *place* map to the negative integers. Note also that *step* does not possess the additive constant that is required in the affine timing functions that are usually used [12, 16]. If there is an affine timing function but none of the form required of *step*, the constant can always be eliminated by application of an index transformation [16]. Note also that π is restricted to disallow stationary streams. Our results extend to stationary streams, but the inclusion of this special case complicates the presentation unduly. Ass. 4 normalizes the place function by assuring a consecutive numbering of the range.

We shall continue to denote a time (space) matrix by Λ (Σ) and a time (space) vector by λ (σ). We shall use Π for a space-time mapping in which either time or space is multi-dimensional; otherwise we shall write π.

A space-time mapping determines the communication between two cells:

Definition 6 *flow*(v) specifies the distance and direction that variables on the stream v travel at each step (compare [3]):

$$flow : V \longrightarrow \mathbb{Z}$$
$$flow(v) = \sigma\theta_v / \lambda\theta_v$$

(End of Definition)

Let $I = J+\theta_v$. Then $\lambda\theta_v$ indicates the temporal distance (in time steps) of the computations $v(I)$ and $v(J)$ at cells $PE_{\sigma I}$ and $PE_{\sigma J}$, whose spatial distance is given by $\sigma\theta_v$. Recall our assumptions that a computation takes one unit of time and a register delays for one unit of time (Sect. 3). To implement the flow of a stream, say v, we need $(1/\mid flow(v)\mid)-1$ registers at the respective channel. If $\sigma\theta_v > 1$, cells $PE_{\sigma I}$ and $PE_{\sigma J}$ are not neighbours and the cells in between function as delay registers. Note that stream v moves to the right if $\sigma\theta_v>0$ $(flow(v)>0)$ and to the left if $\sigma\theta_v<0$ $(flow(v)<0)$.

Restricting λ to satisfy $\gcd(\lambda\sigma_1/\sigma\theta_1, \lambda\sigma_2/\sigma\theta_2, \cdots, \lambda\sigma_k/\sigma\theta_k) = 1$ normalizes the step function in a similar way as we have normalized the place function with Ass. 4 previously. If the gcd is α, the throughput is reduced by a factor of α [7].

Definition 7 *pattern*$(v(I))$ specifies the location of $v(I)$ $(I\in\Omega)$ at the first execution step, where *fs* denotes the first execution step (compare [3]):

$$pattern : V\times\Omega \longrightarrow \mathbb{Z}$$
$$pattern(v(I)) = \begin{cases} \sigma I-(\lambda I-fs)(\sigma\theta_v/\lambda\theta_v), & \text{if } \sigma\theta_v>0,\\ \sigma I-(\lambda I-fs)(\sigma\theta_v/\lambda\theta_v), & \text{if } \sigma\theta_v<0 \end{cases}$$

(*End of Definition*)

Let us define $p_{\min} = \min\{\sigma I \mid I\in\Phi\}$ and $p_{\max} = \max\{\sigma I \mid I\in\Phi\}$. $PE_{p_{\min}}$ and $PE_{p_{\max}}$ are the leftmost and rightmost cells in the array. The following lemma, partly taken from [9], specifies the step at which an input variable is injected into the array and the step at which an output variable is ejected from the array.

Lemma 1 *Let a stream be v and its associated dependence vector be θ_v.*

1. *The step at which an input variable $v(I)\in IN_v$ is injected into the array is given by:*

$$T_{\text{in}}(v(I)) = \begin{cases} \lambda I-(\sigma I-p_{\min})(\lambda\theta_v/\sigma\theta_v), & \text{if } \sigma\theta_v>0,\\ \lambda I-(\sigma I-p_{\max})(\lambda\theta_v/\sigma\theta_v), & \text{if } \sigma\theta_v<0 \end{cases} \tag{1}$$

2. *The step at which an output variable $v(I)\in OUT_v$ is ejected from the array is given by:*

$$T_{\text{out}}(v(I)) = \begin{cases} \lambda I-(\sigma I-p_{\max})(\lambda\theta_v/\sigma\theta_v), & \text{if } \sigma\theta_v>0,\\ \lambda I-(\sigma I-p_{\min})(\lambda\theta_v/\sigma\theta_v), & \text{if } \sigma\theta_v<0 \end{cases} \tag{2}$$

(Proof omitted.)

The following two lemmata will play a technical role later on.

Two variables $v(I)$ and $v(J)$ satisfying $I = J+m\theta_v$, for a fixed $m\in\mathbb{Z}_0^+$, represent the same stream element at different points in the execution. The next lemma partially reflects this fact.

Lemma 2 *Let a stream be v and its associated dependence vector be θ_v.*

1. $v(J)\in IN_v$, $I = J+m\theta_v$, $m\in\mathbb{Z}_0^+ \implies T_{\text{in}}(v(I)) = T_{\text{in}}(v(J))$.

2. $v(I) \in OUT_v,\ I = J + m\theta_v,\ m \in \mathbb{Z}_0^+ \implies T_{\text{out}}(v(I)) = T_{\text{out}}(v(J))$.

Proof.

1. $\sigma\theta_v < 0 \vee \sigma\theta_v > 0$; without loss of generality, assume $\sigma\theta_v > 0$.

$$
\begin{aligned}
& v(J) \in IN_v,\ I = J + m\theta_v,\ m \in \mathbb{Z}_0^+ \\
\implies & \{v(J) \in IN_v, \text{ by Lemma 1, Equ. 1}\} \\
& T_{\text{in}}(v(J)) = \lambda J - (\sigma J - p_{\min})(\lambda\theta_v / \sigma\theta_v) \\
\implies & \{J := I - m\theta_v \text{ on the right side}\} \\
& T_{\text{in}}(v(J)) = \lambda(I - m\theta_v) - (\sigma(I - m\theta_v) - p_{\min})(\lambda\theta_v / \sigma\theta_v) \\
\iff & \{\text{algebraic simplification, linearity of } \lambda \text{ and } \sigma\} \\
& T_{\text{in}}(v(J)) = \lambda I - (\sigma I - p_{\min})(\lambda\theta_v / \sigma\theta_v) \\
\iff & \{\text{Lemma 1, Equ. 1}\} \\
& T_{\text{in}}(v(J)) = T_{\text{in}}(v(I))
\end{aligned}
$$

2. Similar. *(End of Proof)*

Consider a stream v. If its input variables are injected into the array at distinct time steps, its output variables will be ejected at distinct time steps. Intuitively, this follows from the fact that the elements of the stream move at a fixed speed.

Lemma 3 *Let $v(I_{\text{in}}), v(J_{\text{in}}) \in IN_v$ and $v(I_{\text{out}}), v(J_{\text{out}}) \in OUT_v$.*

$$
T_{\text{in}}(v(I_{\text{in}})) \neq T_{\text{in}}(v(J_{\text{in}})) \iff T_{\text{out}}(v(I_{\text{out}})) \neq T_{\text{out}}(v(J_{\text{out}}))
$$

Proof. Because we are dealing with convex sets, there is a bijection between IN_v and OUT_v. For element $v(I) \in IN_v$, there is a unique $m_{v(I)} \in \mathbb{Z}_0^+$ such that $v(I + m_{v(I)}\theta_v) \in OUT_v$. Then $I_{\text{out}} = I_{\text{in}} + m\theta_v,\ J_{\text{out}} = J_{\text{in}} + n\theta_v\ (m, n \in \mathbb{Z}_0^+)$.

$$
\begin{aligned}
& T_{\text{in}}(v(I_{\text{in}})) \neq T_{\text{in}}(v(J_{\text{in}})) \\
\iff & \{\text{by Lemma 2, Part (1), } T_{\text{in}}(v(I_{\text{in}})) := T_{\text{in}}(v(I_{\text{out}})) \text{ and } T_{\text{in}}(v(J_{\text{in}})) := T_{\text{in}}(v(J_{\text{out}}))\} \\
& T_{\text{in}}(v(I_{\text{out}})) \neq T_{\text{in}}(v(J_{\text{out}})) \\
\iff & \{\text{by Lemma 1, Equ. 1, substitute } T_{\text{in}}(v(I_{\text{out}})) \text{ and } T_{\text{in}}(v(J_{\text{out}}))\} \\
& \lambda I_{\text{out}} - (\sigma I_{\text{out}} - p_{\min})(\lambda\theta_v / \sigma\theta_v) \neq \lambda J_{\text{out}} - (\sigma J_{\text{out}} - p_{\min})(\lambda\theta_v / \sigma\theta_v) \\
\iff & \{p_{\min} := p_{\max}, \text{ maintaining the inequality}\} \\
& \lambda I_{\text{out}} - (\sigma I_{\text{out}} - p_{\max})(\lambda\theta_v / \sigma\theta_v) \neq \lambda J_{\text{out}} - (\sigma J_{\text{out}} - p_{\max})(\lambda\theta_v / \sigma\theta_v) \\
\iff & \{\text{Lemma 1, Equ. 2}\} \\
& T_{\text{out}}(v(I_{\text{out}})) \neq T_{\text{out}}(v(J_{\text{out}}))
\end{aligned}
$$

(End of Proof)

We shall use this lemma to reduce our analysis to either input or output variables as is convenient, and infer the same for the other.

The conditions due to Lee and Kedem [9] that are necessary and sufficient for the validity of a space-time mapping with respect to the quatitative model are stated next. The numbering scheme of Lee and Kedem is given in parentheses. We omit one condition, Cond. 4, because it has already been imposed as Cond. 4 of the quantitative model (Sect. 3).

Theorem 3 *Consider the system of UREs* (Φ, D). *Let* $I, J \in \Phi$ $(I \neq J)$. *A space-time mapping* π *is valid if and only if it satisfies the following four mapping constraints.*

1. *Precedence Constraint:* $\lambda\theta_i > 0$. *(Condition 1)*
2. *Delay Constraint:* $|\lambda\theta_i / \sigma\theta_i| \in \mathbb{Z}^+$. *(Condition 3)*
3. *Computation Constraint:* $\sigma I = \sigma J \implies \lambda I \neq \lambda J$. *(Condition 2)*
4. *Communication Constraint:*
$I - J \neq m\theta_i \implies (\lambda(I-J))\sigma\theta_i \neq (\sigma(I-J))\lambda\theta_i$. *(Condition 5)*

(Proof omitted [9].)

Our systolic array model imposes the restriction that I/O computations be performed only at the two border cells. The communication constraint guarantees that only one input variable per time step is injected. The following lemma restates the communication constraint to emphasize this.

Lemma 4 *Let* π *be a space-time mapping. Let* $v(I_{\text{in}}), v(J_{\text{in}}) \in IN_v$ $(I_{\text{in}} \neq J_{\text{in}})$. *Let* $I, J \in \Phi$ $(I \neq J)$.

$$T_{\text{in}}(v(I_{\text{in}})) \neq T_{\text{in}}(v(J_{\text{in}})) \iff (I - J \neq m\theta_v \implies ((\lambda(I-J))\sigma\theta_v \neq (\sigma(I-J))\lambda\theta_v)$$

Proof. $\sigma\theta_v < 0 \vee \sigma\theta_v > 0$; without loss of generality, assume $\sigma\theta_v > 0$.

$$\begin{aligned}
& T_{\text{in}}(v(I_{\text{in}})) \neq T_{\text{in}}(v(J_{\text{in}})) \\
\iff & \{\text{by assumption, } I_{\text{in}} \neq J_{\text{in}} \wedge I \neq J; \text{ by Lemma 2, Part (1),} \\
& (v(I_{\text{in}}) \in IN_v,\ I = I_{\text{in}} + p\theta_v,\ p \in \mathbb{Z}_0^+ \implies T_{\text{in}}(v(I)) = T_{\text{in}}(v(I_{\text{in}}))) \wedge \\
& (v(J_{\text{in}}) \in IN_v,\ J = J_{\text{in}} + q\theta_v,\ q \in \mathbb{Z}_0^+ \implies T_{\text{in}}(v(J)) = T_{\text{in}}(v(J_{\text{in}})))\} \\
& I - J \neq m\theta_v \implies T_{\text{in}}(v(I)) \neq T_{\text{in}}(v(J)) \\
\iff & \{\text{by Lemma 1, Equ. 1, substitute } T_{\text{in}}(v(I)) \text{ and } T_{\text{in}}(v(J))\} \\
& I - J \neq m\theta_v \implies \lambda I - (\sigma I - p_{\text{min}})(\lambda\theta_v/\sigma\theta_v) \neq \lambda I - (\sigma I - p_{\text{min}})(\lambda\theta_v/\sigma\theta_v) \\
\iff & \{\text{algebraic simplification, linearity of } \lambda \text{ and } \sigma\} \\
& I - J \neq m\theta_v \implies (\lambda(I-J))\sigma\theta_v \neq (\sigma(I-J))\lambda\theta_v
\end{aligned}$$

(End of Proof)

Example: Matrix Multiplication

For purpose of illustration, we choose the following space-time mapping:

$$\pi = \begin{bmatrix} \lambda \\ \sigma \end{bmatrix} = \begin{bmatrix} m^2 & m & 1 \\ m^2 & m & 1 \end{bmatrix}$$

Let us evaluate the four mapping constraints:

1. $\lambda\theta_A = m$, $\lambda\theta_B = m^2$, $\lambda\theta_C = 1$. Hence the precedence constraint is satisfied.
2. $\lambda\theta_A/\sigma\theta_A = \lambda\theta_B/\sigma\theta_B = \lambda\theta_C/\sigma\theta_C = 1$. Hence the delay constraint is satisfied.

3. σ is injective. Hence the computation constraint is satisfied.

4. $T_{\text{in}}(A(i,0,k)=T_{\text{in}}(B(0,j,k)=T_{\text{in}}(C(i,j,0)=m^2+m+1$ $(0<i,j,k\leq m)$.
Hence, the communication constraint is violated.

(*End of Example*)

If we eliminate Cond. 4 of the quantitative model, the communication constraint in Thm. 3 can be disregarded.

Theorem 4 *Consider the system of UREs* (Φ, D). *Let* $I, J \in \Phi$ $(I \neq J)$. *A space-time mapping* π *is valid with respect to the quantitative model without Cond. 4 if and only if it satisfies the following three mapping constraints.*

1. *Precedence Constraint:* $\lambda\theta_i > 0$.

2. *Delay Constraint:* $|\lambda\theta_i/\sigma\theta_i| \in \mathbb{Z}^+$.

3. *Computation Constraint:* $\sigma I = \sigma J \implies \lambda I \neq \lambda J$.

Proof. Thm. 3 and Lemma 1. (*End of Proof*)

Example: Matrix Multiplication

For the previous space-time mapping, each of the three streams A, B and C requires m^2 links, one for each input variable of A and B and output variable of C. The $2m^2$ inputs must be injected into the respective links at step m^2+m+1, the m outputs must be extracted from the respective links at step m^3+m^2+m.

(*End of Example*)

Let us return to the original qualitative model (with Cond. 4). The next lemma states that the communication constraint implies the computation constraint.

Lemma 5 *If* π *satisfies the precedence constraint and the communication constraint, it also satisfies the computation constraint.*

Proof. Let $I, J \in \Phi$ $(I \neq J)$.

Case 1. Assume $I-J = m\theta_i$ $(m \in \mathbb{Z}^+)$.

$$
\begin{array}{ll}
& \text{true} \\
\implies & \{\text{assumption}\} \\
& I-J = m\theta_i \\
\implies & \{\text{multiply both sides with } \sigma\text{, linearity of } \sigma\} \\
& \sigma(I-J) = m\sigma\theta_i \\
\implies & \{\sigma\theta_i \neq 0 \text{ by Ass. 3, } I \neq J\text{, } m>0\text{, linearity of } \sigma\} \\
& \sigma I \neq \sigma J \\
\implies & \{\text{propositional calculus}\} \\
& \sigma I = \sigma J \implies \lambda I \neq \lambda J
\end{array}
$$

Case 2. Assume $I - J \neq m\theta_i$, $(m \in \mathbb{Z}^+)$.

$$\begin{array}{ll} & \pi \text{ satisfies the precedence and communication constraints} \\ \Longrightarrow & \{\text{Thm. 3, Part (1) and (4)}\} \\ & \lambda\theta_i > 0 \ \wedge \ (I - J \neq m\theta_i \ \Longrightarrow \ (\lambda(I-J))\sigma\theta_i \neq (\sigma(I-J))\lambda\theta_i) \\ \Longrightarrow & \{\sigma\theta_i \neq 0 \text{ by Ass. 3, } I \neq J\} \\ & \sigma I = \sigma J \ \Longrightarrow \ \lambda I \neq \lambda J \end{array}$$

While the precedence constraint asserts $\lambda\theta_i > 0$, this proof requires only $\lambda\theta_i \neq 0$.

(End of Proof)

At this point, we have simplified Thm. 3 to the following:

Theorem 5 *Consider the system of UREs* (Φ, D)*. Let* $v_i(I), v_i(J) \in IN_{v_i}$ $(I \neq J)$*. A space-time mapping* π *is valid if and only if it satisfies the following three mapping constraints.*

1. *Precedence Constraint:* $\lambda\theta_i > 0$.
2. *Delay Constraint:* $|\lambda\theta_i / \sigma\theta_i| \in \mathbb{Z}^+$.
3. *Communication Constraint:* $T_{\text{in}}(v_i(I)) \neq T_{\text{in}}(v_i(J))$.

Proof. Thm. 3, Lemmata 4 and 5. *(End of Proof)*

Example: Matrix Multiplication

1. The precedence constraint requires: $(\forall\, i : 0 < i \leq 3 : \lambda_i > 0)$.
2. The delay constraint requires: $(\forall\, i : 0 < i \leq 3 : (\exists\, \alpha_i : \alpha_i \in \mathbb{Z}^+ : \lambda_i = \alpha_i \sigma_i))$.
3. Consider the communication constraint. Let us first consider the input variables of stream A. For variable $A(I) \in IN_A$, I can be expressed as $I = p\theta_B + q\theta_C$ $(0 < p, q \leq m)$. $T_{\text{in}}(A(I))$ is the step at which variable $A(I)$ is injected into the array.

$$\begin{array}{ll} & T_{\text{in}}(A(I)) \\ = & \{\text{Lemma 1, Equ. 1}\} \\ & \lambda I - (\sigma I - p_{\min})(\lambda\theta_A / \sigma\theta_A) \\ = & \{\text{algebra}\} \\ & \lambda I - \sigma I(\lambda\theta_A/\sigma\theta_A) + p_{\min}(\lambda\theta_A/\sigma\theta_A) \\ = & \{I := p\theta_B + q\theta_C\} \\ & \lambda(p\theta_B + q\theta_C) - \sigma(p\theta_B + q\theta_C)(\lambda\theta_A/\sigma\theta_A) + p_{\min}(\lambda\theta_A/\sigma\theta_A) \\ = & \{\text{inner product calculation and algebraic simplification}\} \\ & p\lambda_1 + q\lambda_3 - (p\sigma_1 + q\sigma_3)(\lambda_2/\sigma_2) + p_{\min}(\lambda_2/\sigma_2)) \end{array}$$

Similarly, we obtain for the input variables of streams B and C:

$$\begin{aligned} T_{\text{in}}(B(I)) &= p\lambda_2+q\lambda_3-(p\sigma_2+q\sigma_3)(\lambda_1/\sigma_1)+p_{\min}(\lambda_1/\sigma_1) \\ T_{\text{in}}(C(I)) &= p\lambda_1+q\lambda_2-(p\sigma_1+q\sigma_2)(\lambda_3/\sigma_3)+p_{\min}(\lambda_3/\sigma_3)) \end{aligned}$$

The communication constraint requires: $T_{\text{in}}(A(I))$, $T_{\text{in}}(B(I))$ and $T_{\text{in}}(C(I))$ each must be injective mappings to $\mathbb{Z}$.

(*End of Example*)

Lee and Kedem [9] provide a procedure for satisfying the necessary and sufficient conditions given by Thm. 3. In Thm. 5, we state equivalent but simpler necessary and sufficient conditions, which lead to a more efficient procedure. Both procedures do not provide a constructive means for finding space-time mappings. As Thm. 5 states it, the communication constraint is difficult to satisfy constructively.

Alternatively, one may specify bounds on the range of the coefficients in λ and σ and, based on Thm. 5, enumerate all space-time mappings within the specified bounds. A simple-minded enumeration procedure might be:

Procedure 2 (Construction of a space-time mapping by direct derivation of one-dimensional time)

INPUT: A system of UREs (Φ, D), bounds on the range of the coefficients in λ and σ, design constraints (such as the number of registers at some specified channels), a cost function with upper bound.

OUTPUT: All valid space-time mappings.

1. Find the next space-time mapping satisfying the specified bounds. If there are no more mappings, stop.
2. Verify the specified design constraints. If some constraint is violated, go to 1.
3. Verify the mapping constraints of Thm. 5. If some constraint is violated, go to 1.
4. Calculate the cost function; if it is within the specified upper bound, output the mapping. Go to 1.

(*End of Procedure*)

The time complexity of an enumeration procedure depends on the bounds given to it. Assume that the total number of space-time mappings within the bounds is b. Assume Φ is a hypercube of length s. Step 1 takes $O(b)$ time. Steps 2 and 4 take constant time. The time taken by Step 3 is dominated by the verification of the communication constraint. A hypercube has $2n$ surfaces with s^{n-1} points on each surface. Each surface takes $O((n-1)s^{n-1}\log s)$ time. Since there are $2n$ surfaces, Step 3 runs in $O(n(n-1)s^{n-1}\log s)$ time. There are b space-time mappings to be verified. Hence, Proc. 2 runs in $O(bn(n-1)s^{n-1}\log s)$ time.

The formulation of a cost function will be discussed in Sect. 7.

6 Space-Time Mappings and Hyperplanes

In this section, we transform multi-dimensional to one-dimensional time. The properties of space-time mappings and mapping constraints are studied further, based on the concept of hyperplanes [5, 8, 17]. This leads to a more constructive procedure for mapping UREs to the quantitative model.

Definition 8 Given $\beta \in \mathbb{Z}$ ($\mathbb{Q}$), a non-zero constant row vector $c \in \mathbb{Z}^n$ ($\mathbb{Q}^n$) and a column vector $x \in \mathbb{Z}^n$ ($\mathbb{Q}^n$), the set

$$H = \{x \mid cx = \beta\}$$

is called a *hyperplane* in $\mathbb{Z}^n$ ($\mathbb{Q}^n$) (H has $n-1$ dimensions.) *(End of Definition)*

Space-time mappings may be interpreted geometrically. The timing function λI and the place function σI slice the domain of a system of n-dimensional UREs into $(n-1)$-dimensional hyperplanes. Each hyperplane contains all points that are mapped to the same value (step number or location).

For both λ and σ we consider two hyperplanes special. For σ, they are the hyperplanes whose points are mapped to the border cells $PE_{p_{\min}}$ and $PE_{p_{\max}}$. For λ, they are the hyperplanes whose points are mapped to the first and the last step number.

Before we can identify these special hyperplanes, we must extend domain Φ such that the points that use the input variables or define the output variables become part of its boundary [4, 16].

Definition 9 The *convex hull* of a set X is defined as follows [17]:

$$\text{conv.hull } X = \{\textstyle\sum_{i=1}^{t} \lambda_i x_i \mid t \geq 1,\ (\forall\, i : 0 < i \leq t : x_i \in X \wedge \lambda_i \geq 0),\ \sum_{i=1}^{t} \lambda_i = 1\}$$

(End of Definition)

Definition 10 Consider the system of UREs (Φ, D). Let $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_n)$. The extended domain of computation Φ_{E} of Φ is defined as follows:

$$\begin{aligned}
\Phi_i &= \{I \mid J = I \pm m\theta_i,\ J \in \Phi,\ I \notin \Phi,\ m \in \mathbb{Z}^+,\ p_{\min} \leq \sigma I \leq p_{\max}\} \\
\Phi_{\mathrm{P}} &= \cup_{i=1}^{k} \Phi_i \\
\Phi_{\mathrm{E}} &= \{I \mid I \in \text{conv.hull}(\Phi \cup \Phi_{\mathrm{P}}) \wedge I \in \mathbb{Z}^n\}
\end{aligned}$$

To maintain uniformity, the definition of a system of UREs (Def. 1, Sect. 2) must be extended. We add to the already present recurrence equations:

$$I \in \Phi_{\mathrm{E}} \backslash \Phi \rightarrow (\forall\, v : v \in V : v(I) = v(I - \theta_v))$$

(End of Definition)

Φ_i contains the points that are added to the domain by extending dependence vector θ_i in both directions. Φ_{P} represents the pipelining of input variables from the two border cells into the array and of output variables from the array to the two border cells (also called "soaking" and "draining" [3]). When combining the original domain and its extension, we take the convex hull because the domain of computation must remain a convex polyhedron (Sect. 2).

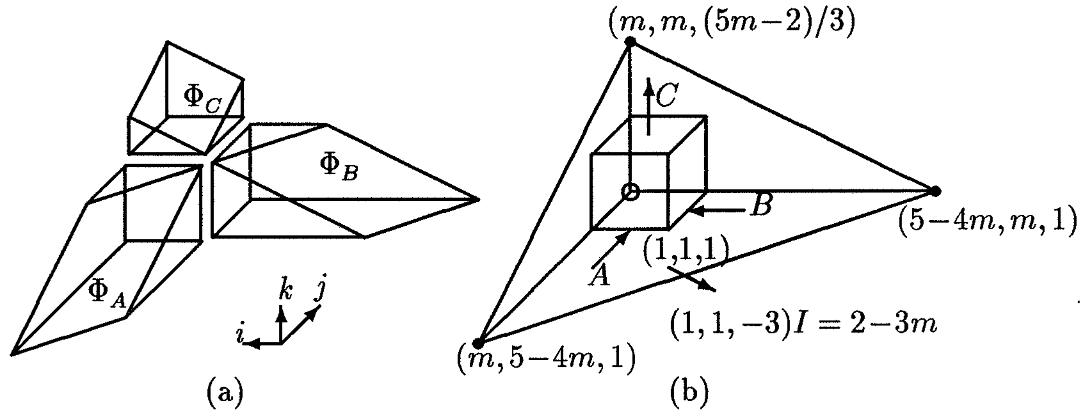


Figure 2: Extension of the domain of computation of matrix multiplication. In (a), Φ_{P} consists of three polyhedrons, Φ_A, Φ_B and Φ_C, that represent the extensions of Φ along dependence vectors θ_A, θ_B and θ_C. Φ^A_{min} ($\Phi^B_{\mathrm{min}}, \Phi^C_{\mathrm{min}}$) is the side of Φ_A (Φ_B, Φ_C) facing us. In (b), Φ is the cube inside the tetrahedron Φ_{E}. The minimum I/O plane Φ_{min} is the trangular plane facing us. Input variables in IN_A and IN_B enter and output variables OUT_C leave the tetrahedron through the minimum I/O plane. Three extreme points of Φ_{min} are highlighted with fat dots; they will be explained later in this section. The maximum I/O plane Φ_{max} degenerates to a single point; it is highlighted with a circle.

Definition 11 Consider the system of UREs (Φ, D). Let π be a space-time mapping.

$$\begin{aligned}
\Phi^i_{\mathrm{min}} &= \{I \mid \sigma I = p_{\mathrm{min}},\ J = I + m\theta_i\ J \in \Phi \cup \Phi_i,\ m \in \mathbb{Q}^+_0\} \\
\Phi^i_{\mathrm{max}} &= \{I \mid \sigma I = p_{\mathrm{max}},\ J = I - m\theta_i\ J \in \Phi \cup \Phi_i,\ m \in \mathbb{Q}^+_0\} \\
\Phi_{\mathrm{min}} &= (\forall\, i : 0 < i \leq k : \cup \Phi^i_{\mathrm{min}}) \\
\Phi_{\mathrm{max}} &= (\forall\, i : 0 < i \leq k : \cup \Phi^i_{\mathrm{max}})
\end{aligned}$$

We call the two hyperplanes Φ_{min} and Φ_{max} the *maximum* and *minimum* I/O plane. Their portions attributed to dependence vector θ_i are Φ^i_{min} and Φ^i_{max}. We call the points of Φ_{min} and Φ_{max} the *I/O points.*

(End of Definition)

Example: Matrix Multiplication

Assume that zeros can be generated inside cells to obtain values for IN_C and that OUT_A and OUT_B are not of interest. Choosing $\sigma = (1, 1, -3)$. By geometrical calculation, we obtain Φ_A, Φ_B, Φ_C, Φ_{P} and Φ_{E} as follows (Fig. 2):

$$\begin{aligned}
\Phi_A &= \{(i,j,k) \mid 0 < i \leq m,\ 2-3m-i+3k \leq j \leq 0,\ 0 < k \leq m\} \\
\Phi_B &= \{(i,j,k) \mid 2-3m-j+3k \leq i \leq 0, 0 < j \leq m,\ 0 < k \leq m\} \\
\Phi_C &= \{(i,j,k) \mid 0 < i \leq m,\ 0 < j \leq m,\ m < k \leq (3m-2+i+j)\ \mathbf{div}\ 3\} \\
\Phi_{\mathrm{P}} &= \Phi_A \cup \Phi_B \cup \Phi_C \\
\Phi_{\mathrm{E}} &= \{(i,j,k) \mid 5-4m \leq i \leq m,\ 5-3m-i \leq j \leq m,\ 0 < k \leq (3m-2+i+j)\ \mathbf{div}\ 3\}
\end{aligned}$$

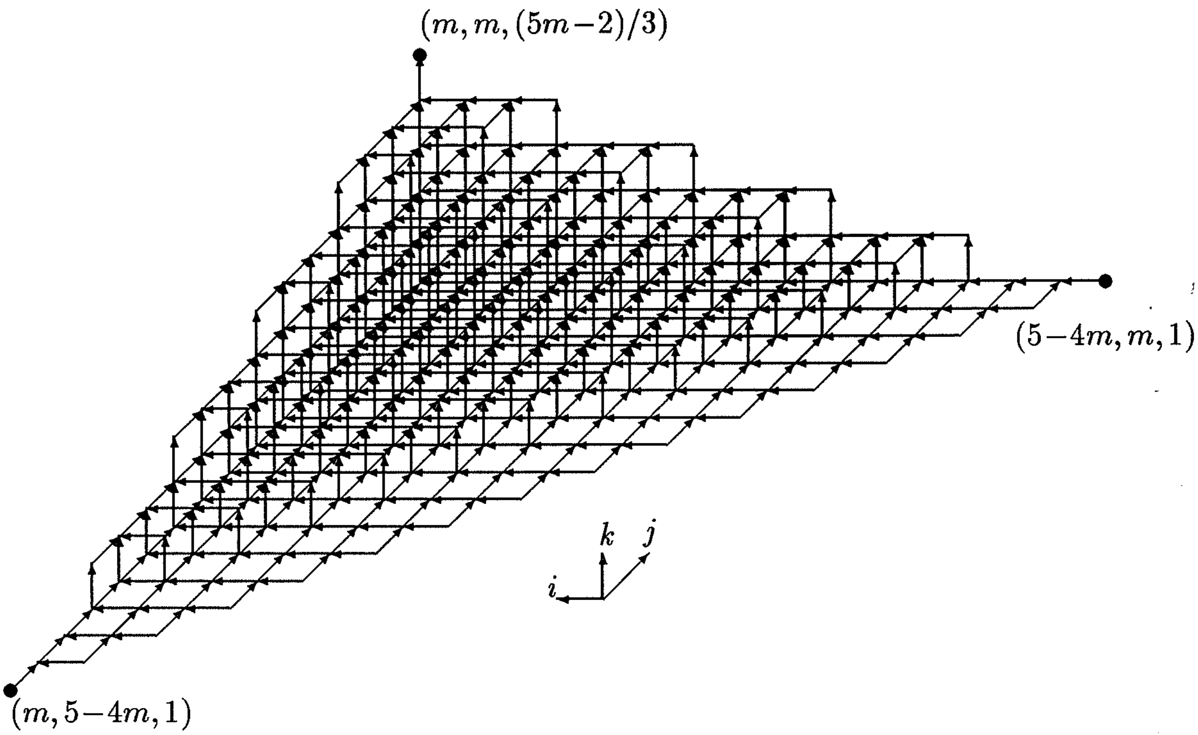


Figure 3: The dependence graphs of matrix multiplication in the extended domain $\Phi_{\rm E}$ with respect to the given vector $\sigma=(1,1,-3)$.

where **div** denotes integer division. The UREs for matrix multiplication in $\Phi_{\rm E}$ are obtained as follows. The equations defined on Φ remain unchanged (Sect. 2). The equations defined on the extensions Φ_A, Φ_B and Φ_C are:

$$\begin{array}{l} I\in\Phi_A \rightarrow A(i,j,k)=A(i,j-1,k) \\ I\in\Phi_B \rightarrow B(i,j,k)=B(i-1,j,k) \\ I\in\Phi_C \rightarrow C(i,j,k)=C(i,j,k-1) \end{array}$$

The dependence graph in the extended domain $\Phi_{\rm E}$ is displayed in Fig. 3. Choosing $\sigma=(1,1,-3)$ yields $p_{\rm min}=2-3m$ and $p_{\rm max}=2m-3$.

$$\begin{array}{l} \Phi^A_{\rm min} = \{(i,j,k) \mid 0<i\leq m,\ j=2-3m-i+3k,\ 0<k\leq m\} \\ \Phi^B_{\rm min} = \{(i,j,k) \mid i=2-3m-j+3k,\ 0<j\leq m,\ 0<k\leq m\} \\ \Phi^C_{\rm min} = \{(i,j,k) \mid 0<i\leq m,\ 0<j\leq m,\ k=(3m-2+i+j)/3\} \\ \Phi_{\rm min} = \Phi^A_{\rm min} \cup \Phi^B_{\rm min} \cup \Phi^C_{\rm min} \\ \Phi_{\rm max} = \{(m,m,1)\} \end{array}$$

Compare Φ_A (Φ_B, Φ_C) with $\Phi^A_{\rm min}$ $(\Phi^B_{\rm min}, \Phi^C_{\rm min})$.

(*End of Example*)

The following lemma provides a construction of I/O points.

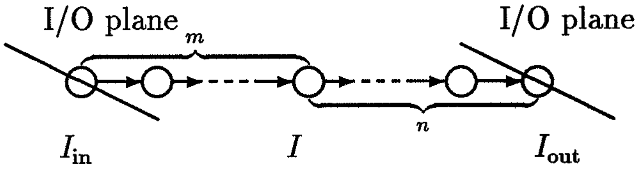


Figure 4: An example of one-dimensional I/O planes ($n=2$). The two I/O points I_{in} and I_{out} with respect to a point $I\in\Phi$ and its dependence vector θ_i are shown.

Lemma 6 *Consider the system of UREs (Φ, D). Take a vector σ and a point $I\in\Phi$ ($I=(I_1,I_2,\cdots,I_n)$). The two I/O points I_{in} ($I=I_{\text{in}}+m\theta_i$, $m\in\mathbb{Q}_0^+$) and I_{out} ($I_{\text{out}}=I+n\theta_i$, $n\in\mathbb{Q}_0^+$) with respect to dependence vector $\theta_i\in D$ ($\theta_i=(\theta_{i1},\theta_{i2},\cdots,\theta_{in})$) are completely determined by (Fig. 4):*

$$\begin{aligned} I_{\text{in}} &= (I_1+\theta_{i1}\frac{p_{\min}-\sigma I}{\sigma\theta_i}, I_2+\theta_{i2}\frac{p_{\min}-\sigma I}{\sigma\theta_i},\cdots,I_n+\theta_{in}\frac{p_{\min}-\sigma I}{\sigma\theta_i}) \\ I_{\text{out}} &= (I_1+\theta_{i1}\frac{p_{\max}-\sigma I}{\sigma\theta_i}, I_2+\theta_{i2}\frac{p_{\max}-\sigma I}{\sigma\theta_i},\cdots,I_n+\theta_{in}\frac{p_{\max}-\sigma I}{\sigma\theta_i}) \end{aligned}$$

(Proof by calculation omitted.)

Example: Matrix multiplication

The extreme point $(m, m, (5m-2)/3)$ of the minimum I/O plane is not integral if $m\neq 3k+1$ ($k\in\mathbb{Z}_0^+$).

(*End of Example*)

I/O points are rational ($\Phi_{\min}$, $\Phi_{\max}\subset\mathbb{Q}^n$) but need not be integral, as the previous example shows.

Next we restate Lemma. 4 (Sect. 5) using the concept of I/O planes.

Theorem 6 *Let π be a space-time mapping. Let $I_{\text{in}}, J_{\text{in}}\in\Phi^i_{\min}$ ($I_{\text{in}}\neq J_{\text{in}}$) and $I, J\in\Phi$ ($I\neq J$).*

$$\lambda I_{\text{in}}\neq\lambda J_{\text{in}} \iff (I-J\neq m\theta_v \implies (\lambda(I-J))\sigma\theta_v\neq(\sigma(I-J))\lambda\theta_v)$$

Proof. Lemmata 1, 4 and 6. (*End of Proof*)

Having completed our extension of the domain, we can now introduce in time similar concepts to $p_{\min}$ and $p_{\max}$ in space (Sect. 5). We define $t_{\min}=\min\{\lambda I\mid I\in\Phi_{\min}\cup\Phi_{\max}\}$ and $t_{\max}=\max\{\lambda I\mid I\in\Phi_{\min}\cup\Phi_{\max}\}$. $t_{\min}$ and $t_{\max}$ represent the first and last step number.

We can calculate $t_{\min}$ and $t_{\max}$ from $\Phi_{\min}$ and $\Phi_{\max}$ with techniques of integer programming. Integer programming is an NP-complete problem, also when applied to UREs, but in many cases it turns out to be quite simple [11, 16]. Our search space is reduced because only those extreme points in $\Phi_{\min}$ and $\Phi_{\max}$ qualify that satisfy the dependences imposed by λ.

Definition 12 Let π be a space-time mapping. We call the following two hyperplanes

$$\begin{aligned} \Upsilon_{\text{min}} &= \{I \mid \lambda I = t_{\text{min}},\ I \in \Phi_{\text{min}} \cup \Phi_{\text{max}}\} \\ \Upsilon_{\text{max}} &= \{I \mid \lambda I = t_{\text{max}},\ I \in \Phi_{\text{min}} \cup \Phi_{\text{max}}\} \end{aligned}$$

the *minimum* and *maximum time plane.* *(End of Definition)*

Example: Matrix Multiplication

The three extreme points of the minimum I/O plane are $(m, 5-4m, 1)$, $(5-4m, m, 1)$ and $(m, m, (5m-2)/3)$. The only point of the maximum I/O plane is at $(m, m, 1)$. An inspection of the dependence graph (Fig. 3) reveals that either $(m, 5-4m, 1)$ or $(5-4m, m, 1)$ must be mapped to t_{min}, and $(m, m, (5m-2)/3)$ must be mapped to t_{max}.

$$\begin{aligned} t_{\text{min}} &= \begin{cases} m\lambda_1 + (5-4m)\lambda_2 + \lambda_3 & \text{for } \Upsilon_{\text{min}} = \{(m, 5-4m, 1)\} \\ (5-4m)\lambda_1 + m\lambda_2 + \lambda_3 & \text{for } \Upsilon_{\text{min}} = \{(5-4m, m, 1)\} \end{cases} \\ t_{\text{max}} &= m\lambda_1 + m\lambda_2 + ((5m-2)/3)\lambda_3 \quad \text{for } \Upsilon_{\text{max}} = \{(m, m, (5m-2)/3)\} \end{aligned}$$

(End of Example)

When restricting I/O to the border cells, we are particularly interested in the number of steps spent on soaking and draining.

Definition 13 *Let π be a space-time mapping. Let t_{fst} denote the step of the first computation; let t_{lst} denote the step of the last computation.*

$$\begin{aligned} t_{\text{fst}} &= \min\{\lambda I \mid I \in \Phi\} \\ t_{\text{lst}} &= \max\{\lambda I \mid I \in \Phi\} \end{aligned}$$

The soaking time t_{soak}, draining time t_{drain} and computation time t_{comp} are given by

$$\begin{aligned} t_{\text{soak}} &= t_{\text{fst}} - t_{\text{min}} \\ t_{\text{drain}} &= t_{\text{max}} - t_{\text{lst}} \\ t_{\text{comp}} &= t_{\text{lst}} - t_{\text{fst}} + 1 \end{aligned}$$

(End of Definition)

Again, the points that are mapped to t_{fst} and t_{lst} must be extreme points of Φ and must satisfy the dependences imposed by λ.

Example: Matrix Multiplication

There are eight extreme points in Φ: $\{(i, j, k) \mid i = 1, m,\ j = 1, m,\ k = 1, m\}$. An inspection of the dependence graph defined at domain Φ (Sect. 2) reveals that point $(1, 1, 1)$ must be mapped to t_{fst} and point (m, m, m) must be mapped to t_{lst}.

$$\begin{aligned} t_{\text{fst}} &= \lambda_1 + \lambda_2 + \lambda_3 \\ t_{\text{lst}} &= m\lambda_1 + m\lambda_2 + m\lambda_3 \\ t_{\text{soak}} &= \begin{cases} (1-m)\lambda_1 + (4m-4)\lambda_2 & \text{for } \Upsilon_{\text{min}} = \{(m, 5-4m, 1)\} \\ (4m-4)\lambda_1 + (1-m)\lambda_2 & \text{for } \Upsilon_{\text{min}} = \{(5-4m, m, 1)\} \end{cases} \\ t_{\text{drain}} &= ((2m-2)/3)\lambda_3 \quad \text{for } \Upsilon_{\text{max}} = \{(m, m, (5m-2)/3)\} \\ t_{\text{comp}} &= (m-1)\lambda_1 + (m-1)\lambda_2 + (m-1)\lambda_3 + 1 \end{aligned}$$

(*End of Example*)

Let us denote the total number of time steps taken steps_π, the total number of cells needed cells_π, the total number of channels needed chans_π and the total number of registers needed regs_π (each of these values depends on the choice of space-time mapping π). The following lemma characterizes this dependence [9].

Theorem 7 *Consider the system of UREs* (Φ, D). *Let* π *be a space-time mapping that satisfies Thm. 5.* step_π, cell_π, chan_π *and* reg_π *are given by:*

1. $\text{steps}_\pi = t_{\text{max}} - t_{\text{min}} + 1$
2. $\text{cells}_\pi = p_{\text{max}} - p_{\text{min}} + 1$
3. $\text{chans}_\pi = k$
4. $\text{regs}_\pi = \text{cells}_\pi \sum_{i=1}^{\text{chans}_\pi} (\mid 1/\text{flow}(v_i) \mid -1)$

Proof.

1. Follows from the definitions of p_{min} and p_{max} (Sect. 5).
2. Follows from the definitions of t_{min} and t_{max} (Sect. 6).
3. Follows from Cond. 4 of the quantitative model (Sect. 3).
4. Follows from (2) and (3) and from the fact that the number of registers needed for propagation of the stream with respect to dependence θ_i is $\mid 1/\mathit{flow}(v_i) \mid -1$ (Sect. 5).

(*End of Proof*)

In the following theorem, we identify a sufficient condition under which the communication constraint implies the computation constraint.

Theorem 8 *Consider the system of UREs* (Φ_{E}, D). *Assume* $\Phi_{\text{min}} \subset \mathbb{Z}^n$. *If* π *satisfies the computation constraint, it also satisfies the communication constraint.*

Proof.

π satisfies the computation constraint.
$\Longrightarrow$ {Thm. 3, Part (3)}
$(\forall\ I, J \in \Phi_{\text{E}} : \sigma I = \sigma J \Longrightarrow \lambda I \neq \lambda J)$
$\Longrightarrow$ $\{\Phi_{\text{min}} \subset \Phi_{\text{E}}$, by Def. 10 and assumption$\}$
$(\forall\ I_{\text{in}}, J_{\text{in}} \in \Phi_{\text{min}} : \sigma I_{\text{in}} = \sigma J_{\text{in}} \Longrightarrow \lambda I_{\text{in}} \neq \lambda J_{\text{in}})$
$\Longrightarrow$ $\{\sigma I_{\text{in}} = \sigma J_{\text{in}} = p_{\text{min}}$, by Def. 11,$\}$
$(\forall\ I_{\text{in}}, J_{\text{in}} \in \Phi_{\text{min}} : \lambda I_{\text{in}} \neq \lambda J_{\text{in}})$
$\Longrightarrow$ $\{\Phi^i_{\text{min}} \subset \Phi_{\text{min}}$, by Def. 11,$\}$
$(\forall\ I_{\text{in}}, J_{\text{in}} \in \Phi^i_{\text{min}} : \lambda I_{\text{in}} \neq \lambda J_{\text{in}})$
$\Longrightarrow$ {Thm. 6}
The communication constraint is satisfied. (*End of Proof*)

(End of Proof)

Assuming $\Phi_{\min} \subset \mathbb{Z}^n$, Thm. 3 can be restated without the communication constraint.

Theorem 9 *Consider the system of UREs* $(\Phi_{\rm E}, D)$. *Assume* $\Phi_{\rm in} \in \mathbb{Z}^n$. *Let* $I, J \in \Phi_{\rm E}$ $(I \neq J)$. *A space-time mapping* Φ *is valid if and only if it satisfies the following three mapping constraints.*

1. *Precedence Constraints:* $\lambda\theta_i > 0$.
2. *Delay Constraints:* $|\lambda\theta_i / \sigma\theta_i| \in \mathbb{Z}^+$.
3. *Computation Constraint:* $\sigma I = \sigma J \implies \lambda I \neq \lambda J$.

Proof. Thms. 3 and 8. *(End of Proof)*

Example: Matrix Multiplication

Choosing $\sigma = (1, 1, -1)$, we obtain $|\sigma\theta_A| = |\sigma\theta_B| = |\sigma\theta_B| = 1$. Lemma 6 tells us that the I/O points are integral, i.e., $\Phi_{\min} \subset \mathbb{Z}^n$. Hence, the mapping constraints are given by Thm. 9.

Let us first pick a space-time mapping Π that satisfies the mapping constraints of Thm. 1:

$$\Pi = \begin{bmatrix} \Lambda \\ \sigma \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

and then transform Λ into λ by applying Proc. 1 to obtain the space-time mapping π:

$$\pi = \begin{bmatrix} \lambda \\ \sigma \end{bmatrix} = \begin{bmatrix} 2m-2 & 1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

By Thms. 1 and 2, π satisfies the precedence constraint and the computation constraint of Thm. 9; it also trivially satisfies the delay constraint of Thm. 9. Hence, the space-time mapping π is valid. This mapping is presented in [4].

(End of Example)

Theorem 10 *Assume* $\Phi_{\min} \not\subset \mathbb{Z}^n$. *For some system of UREs* (Φ, D), *there exists a space-time mapping* π *that satisfies the computation constraint but violates the communication constraint.*

Proof. The proof presents an example that validates the theorem. Consider the system of UREs $(\Phi, D^{3\times 4})$:

$$\Phi = \{(i, j, k) \mid 0 < i, j, k \leq 4\}$$

$$D = (\theta_A, \theta_B, \theta_C, \theta_X) = \begin{bmatrix} 0 & 1 & 0 & 3 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Choosing $\sigma = (1, 1, -1)$, we obtain

$$\Phi_{\rm E} \;=\; \{(i, j, k) \mid -5 \leq i \leq 4,\ -1-i \leq j \leq 4,\ 0 < k \leq (2+i+j)\}$$

We pick the space-time mapping:

$$\pi = \begin{bmatrix} \lambda \\ \sigma \end{bmatrix} = \begin{bmatrix} 6 & 1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

By Lemma 6 and calculation, $\Phi_{\text{min}} \not\subset \mathbb{Z}^n$ (because $\Phi^{\text{X}}_{\text{min}} \not\subset \mathbb{Z}^n$). π satisfies the precedence, delay and computation constraint, but:

$$T_{\text{in}}(X(1,3,4)^t) = T_{\text{in}}(X(3,1,2)^t) = 5.$$

Hence the communication constraint is violated.

(End of Proof)

If there are I/O points that are not integral, Thm. 9 requires an appropriate scaling of the domain to ensure the containment of all I/O points in $\mathbb{Z}$. We present a procedure that returns a valid space-time mapping if the precedence constraint can be satisfied. In the following, LCM stands for the least common multiple.

Procedure 3 (Construction of a space-time mapping by derivation of one-dimensional via multi-dimensional time)

INPUT: A system of UREs (Φ, D) and a vector σ.

OUTPUT: A vector λ.

1. Extend domain Φ to obtain the new domain Φ_{E} (with respect to σ) by Def. 10. Name the minimal I/O plane Φ_{min}.
2. Find the smallest scaling factors α_j $(\alpha_j \in \mathbb{Z}^+)$ such that $(\forall\, j : 0 < j \leq n : (\forall\, i : 0 < i \leq k : |\, \alpha_j(\theta_{ij}/\sigma\theta_i) \,| \in \mathbb{Z}^+_0))$ (see Lemma 6). Set $\alpha = \text{LCM}(\forall\, j : 0 < j \leq n : \alpha_j)$.
3. Set $\Psi_{\text{E}} = \{\alpha I \mid I \in \Phi_{\text{E}}\}$, i.e., scale the index points of Φ_{E} by a factor of α. The scaled version of Φ_{min} is $\Psi_{\text{min}} \in \mathbb{Z}^n$. Set $\Xi = \{I \mid I \in \text{conv.hull}\, \Psi_{\text{E}} \;\wedge\; I \in \mathbb{Z}^n\}$. Name the new system of UREs (Ξ, D). Name its minimal I/O plane Ξ_{min}.
4. Find a $(n-1) \times n$ time matrix Λ such that $\Pi = \begin{bmatrix} \Lambda \\ \sigma \end{bmatrix}$ satisfies the mappings constraints of Thm. 1.
5. Set $r = n-1$. Transform Λ to λ using Proc. 1 with Ξ, Λ and r as inputs.
6. Find the smallest β $(\beta \in \mathbb{Z}^+)$ such that $(\forall\, i : 0 < i \leq k : |\, \beta(\lambda\theta_i/\sigma\theta_i) \,| \in \mathbb{Z}^+)$, i.e., scale λ by a factor of β.

(End of Procedure)

When designing multi-dimensional systolic arrays, we can choose either the layout in time or the layout in space before the other. Usually, one chooses the layout in time first because time is considered more valuable than space. Posing the restriction that the spatial layout be of one dimension only makes it sensible to choose the spatial layout first. Most importantly, the choice of σ determines the direction of projection of the spatial layout. Therefore the input of σ to the procedure. This is also the reason that we prefer to reason not in terms of *pattern* but in terms of its dual in time: $T_{\min}$.

It can be shown that $\Phi_{\min} \subset \mathbb{Z}^n \iff \Xi_{\min} \subset \mathbb{Z}^n$. After scaling, all I/O points are integral ($\Psi_{\min} \subset \mathbb{Z}^n$). Non-integral points in $\Xi_{\min}$ do not correspond to any computations.

Theorem 11 *Proc. 3 returns a space-time mapping that satisfies the constraints of Thm. 9.*

Proof. Steps 4 and 5 guarantee that the space-time mapping satisfies the precedence and computation constraint (Thms. 1 and 2). Scaling λ in Step 6 guarantees that the delay constraint is satisfied. It remains to prove that the communication constraint is satisfied:

π satisfies the computation constraint.
$\Longrightarrow$ {Thm. 3, Part (3)}
$(\forall\, I, J \in \Xi : \sigma I = \sigma J \Longrightarrow \lambda I \neq \lambda J)$
$\Longrightarrow$ $\{\Psi_{\min} \subset \Xi$ by Def. 10 and assumption$\}$
$(\forall\, I_{\text{in}}, J_{\text{in}} \in \Psi_{\min} : \sigma I_{\text{in}} = \sigma J_{\text{in}} \Longrightarrow \lambda I_{\text{in}} \neq \lambda J_{\text{in}})$
$\Longrightarrow$ $\{\sigma I_{\text{in}} = \sigma J_{\text{in}} = p_{\min}$ by Def. 11$\}$
$(\forall\, I_{\text{in}}, J_{\text{in}} \in \Psi_{\min} : \lambda I_{\text{in}} \neq \lambda J_{\text{in}})$
$\Longrightarrow$ $\{\Psi^i_{\min} \subset \Psi_{\min}$ by Def. 11$\}$
$(\forall\, I_{\text{in}}, J_{\text{in}} \in \Psi^i_{\min} : \lambda I_{\text{in}} \neq \lambda J_{\text{in}})$
$\Longrightarrow$ {Thm. 6}
The communication constraint is satisfied.

(*End of Proof*)

7 Evaluation

Example: Matrix Multiplication

Assume that input variables IN_C and output variables OUT_A and OUT_B are not communicated.

Tab. 1 lists several space-time mappings. The mappings in (a) have been derived by Proc. 2, the ones in (b) are taken from the literature and have been derived by Proc. 3 or similar techniques, followed by individual optimizations. (In the row labeled by $[15]^o$, n is odd; in the row labeled by $[15]^e$, n is even.) We also list the resource requirements calculated following Def. 13 and Thm. 7.

(*End of Example*)

Let us compare Procs. 2 and 3. When one enumerates all solutions, one has complete freedom to impose any design constraints one might like on the space of solutions. One can also synthesize space-time optimal one-dimensional arrays. One reasonable cost function for space-time mappings would be:

$$\text{cost}_\pi = \alpha_1 \text{steps}_\pi + \alpha_2 \text{cells}_\pi + \alpha_3 \text{chans}_\pi + \alpha_4 \text{regs}_\pi .$$

λ	σ	PEs	Registers	Soaking	Draining	Computing
(2,3,2)	(1,1,−1)	10	40	12	12	22
(2,6,4)	(1,2,−2)	16	64	21	18	37
(2,2,4)	(1,2,−4)	22	22	30	9	25
(1,2,6)	(1,1,1)	10	60	3	27	28
(1,6,4)	(1,1,2)	13	78	39	3	34

(a) Size: 4×4; method: enumeration.

	λ	σ	PEs	Registers	Soaking	Draining	Computing
[4]	$(2m-2,1,1)$	(1,1,−1)	$3m-2$	$6m^2-13m+6$	$4m^2-9m+5$	$2m-2$	$2m^2-2m+1$
[9]	$(2,1,m-1)$	(1,1,−1)	$3m-2$	$3m^2-5m+2$	$3m-3$	$2(m-1)^2$	m^2+m-1
$[15]^o$	$(2m,1,\frac{m+1}{2})$	$(m,1,\frac{-m-1}{2})$	$\frac{3m^2-1}{2}$	$\frac{3m^2-1}{2}$	m^2+m-2	m^2-1	$\frac{5m^2-2m-1}{2}$
$[15]^e$	$(2m-2,1,\frac{m}{2})$	$(m-1,1,\frac{-m}{2})$	$\frac{3m^2-3m+2}{2}$	$\frac{3m^2-3m+2}{2}$	m^2-1	m^2-m	$\frac{5m^2-7m+4}{2}$

(b) Size: $m\times m$; method: integer programming and others.

Table 1: Matrix multiplication; resource requirements.

where steps_π, cells_π, chans_π and regs_π are defined in Thm. 7 and the weights α_i $(0<i\leq4)$ depend on the application. By selecting different weights, one can synthesize A, T, AT and AT^2 optimal arrays [18]. The obvious disadvantage of enumeration is the dependence of its time complexity on the chosen bounds. In our setting, there is good reason to keep these bounds small: if dependence vectors are constants – and they usually are – large bounds on λ and σ lead to potentially large communication distances.

Proc. 3 is based on linear algebra and integer programming. It is more constructive than enumeration and its solution space is not restricted by (more or less) artificial bounds, but it is difficult to take design constraints into account. Moreover, the resulting solutions may be inefficient: while σ maps the I/O points in planes Ψ_{min} and Ψ_{max} to the same location p_{min} and p_{max}, Step 5 will map them to distinct steps. But the communication constraint of Thm. 6 only requires that the I/O points in Ψ^i_{min} (Ψ^i_{max}) for a fixed i be mapped to distinct steps (Def. 11). In other words, even though it is permitted to input distinct streams at a border cell in parallel, Proc. 3 prevents this.

8 References

[1] J. M. Delosme and I. C. F. Ipsen, "Systolic Array Synthesis: Computability and Time Cones", in *Parallel Algorithms & Architectures*, M. Cosnard, P. Quinton, Y. Robert and M. Tchuente (eds.), North-Holland, 1986, 295–312.

[2] J. A. B. Fortes and D. I. Moldovan, "Parallelism Detection and Algorithm Transformation Techniques Useful for VLSI Architecture Design", *J. Parallel and Distributed Computing 2*, 3 (Aug. 1985), 277–301.

[3] C.-H. Huang and C. Lengauer, "The Derivation of Systolic Implementations of Programs", *Acta Informatica 24*, 6 (Nov. 1987), 595–632.

[4] H. V. Jagadish, S. K. Rao and T. Kailath, "Array Architecture for Iterative Algorithms", *Proc. IEEE 75*, 9 (Sept. 1987), 1034–1320.

[5] R. M. Karp, R. E. Miller and S. Winograd, "The Organization of Computations for Uniform Recurrence Equations", *J. ACM 14*, 3 (July 1967), 563–590.

[6] V. K. Prasanna Kumar and Y.-C. Tsai, "Designing Linear Systolic Arrays", *J. Parallel and Distributed Computing 7*, 3 (Nov. 1989), 441–463.

[7] H. T. Kung and M. S. Lam, " Wafer-Scale Integration and Two-Level Pipelined Implementations", *J. Parallel and Distributed Computing 1*, 1 (Aug. 1984) 33–63.

[8] L. Lamport, "The Parallel Execution of DO Loops", *Comm. ACM 17*, 2 (Feb. 1974), 83–93.

[9] P. Lee and Z. Kedem, "Synthesizing Linear-Array Algorithms from Nexted for Loop Algorithms", *IEEE Trans. on Computers 37*, 12 (Dec. 1988), 1578–1598.

[10] D. I. Moldovan, "On the Design of Algorithms for VLSI Systolic Arrays", *Proc. IEEE 71*, 1 (Jan. 1983), 113–120.

[11] P. Quinton, "Automatic Synthesis of Systolic Arrays from Uniform Recurrent Equations", Proc. *11th Ann. Int. Symp. on Computer Architecture*, IEEE Computer Society Press, 1984, 208–214.

[12] P. Quinton and V. van Dongen, "The Mapping of Linear Recurrence Equations on Regular Arrays", *J. VLSI Signal Processing 1*, 2 (Oct. 1989), 95–113.

[13] S. V. Rajopadhye and R. M. Fujimoto, "Synthesizing Systolic Arrays from Recurrence Equations", *Parallel Computing 14*, 2 (June 1990), 163–189.

[14] I. Ramakrishnan, D. Fussell and A. Silberschatz, "Mapping Homogeneous Graphs on Linear Arrays", *IEEE Trans. on Computers C-35*, 3 (Nov. 1986), 189–209.

[15] I. Ramakrishnan and P. Varman , "Modular Matrix Multiplication on a Linear Array" *IEEE Trans. on Computers C-33*, 11 (Nov. 1984), 952–958.

[16] S. K. Rao and T. Kailath, "Regular Iterative Algorithms and their Implementations on Processor Arrays", *Proc. IEEE 76*, 2 (Mar. 1988), 259–282.

[17] A. Schrijver, *Theory of Linear and Integer Programming*, Series in Discrete Mathematics, John Wiley & Sons, 1986.

[18] J. D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, 1984, Chap. 2.

[19] Y. Wong and J. M. Delosme, "Optimal Systolic Implementations of N-Dimensional Recurrences", Proc. *IEEE Int. Conf. on Computer Design (ICCD 85)*, IEEE Press, 1985, 618–621. Also: Tech. Report 8810, Department of Computer Science, Yale University, New Haven, April 1988.