Scheduling a System of Nonsingular Affine Recurrence Equations onto a Processor Array

Yoav Yaacoby
Department of Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106

Peter R. Cappello[1]
Department of Computer Science
University of California
Santa Barbara, CA 93106

[1]This material is based upon work supported by the Office of Naval Research under contract nos. N00014-84-K-0664 and N00014-85-K-0553.

Abstract

Most work on the problem of scheduling computations onto a systolic array is restricted to systems of uniform recurrence equations. In this paper, this restriction is relaxed to include systems of affine recurrence equations. In this broader class, a sufficient condition is given for the system to be computable. Necessary and sufficient conditions are given for the existence of an affine schedule, along with a procedure that constructs the schedule vector, when one exists.

Key Words: affine recurrence equation, computability, concurrent computation, data dependence, parallel computation, processor array, scheduling, systolic array, uniform recurrence equation.

1 Introduction

1.1 Statement of the problem

A system of uniform recurrence equations, as defined by Karp, Miller, and Winograd [8, 9], maps especially well onto a systolic/wavefront array. Many researchers have either linearly mapped systems of uniform recurrence equations into spacetime, or translated them to systolic/wavefront arrays [14, 1, 15, 2, 19, 13, 5, 3, 21, 16, 20, 10].

One aspect of scheduling is to find an *affine schedule*, a vector $\pi \in \mathbf{Z}^n$ and a set of constants $\{c_i \in \mathbf{Z}\}$, one for each array, such that **for all problem sizes**, if a variable $a_j(x_2)$ depends on a variable $a_i(x_1)$ (not necessarily directly), then $\pi^T x_1 + c_i < \pi^T x_2 + c_j$. The existence of an affine schedule implies a valid execution ordering. The case of uniform recurrences has been considered by several authors (see, e.g., [6], [21], [4, § 7], [17], [22]). We are concerned with a generalization of uniform recurrence equations, called affine recurrence equations. Three related problems are considered:

1. Which systems of affine recurrence equations are computable?
2. Which systems of affine recurrence equations have an affine schedule?
3. When a system of affine recurrence equations has an affine schedule, how can it be computed?

1.2 Summary of the main results

Given a system of affine recurrence equations, we provide:

1. Sufficient conditions for it to be computable (Thms. 3.6 and 3.7).
2. Necessary and sufficient conditions for the existence of an affine schedule (Thms. 3.1, 3.2, 3.4, and 3.5).
3. A procedure which constructs a schedule vector, if one exists (§3.2, relying on Thms. 3.2 and 3.3).

1.3 Significance to processor arrays

Some affine recurrence equations may be formulated such that a variable needs to be 'broadcast'. In such cases, the recurrence equations may be reformulated so that the 'broadcast' variable becomes

a 'propagating' variable. Broadcast removal is **not** discussed in this paper (see, e.g., Leiserson and Saxe [12, 11], Miranker and Winkler [13], and Wong and Delosme [23] for details concerning broadcast removal).

Not all systems of affine recurrence equations can be converted to systems of uniform recurrence equations [24]. In these cases, they must be dealt with directly. We follow, in spirit, the work of Karp, et al. [8]; detecting computability, and computing an affine schedule, are crucial to the automatic implementation of systems of affine recurrence equations on processor arrays. Although not discussed in this paper, processors can be allocated using the same method as that for systems of uniform recurrence equations described in [21]. In case the system of affine recurrence equations cannot be converted to a system of [quasi-]uniform recurrence equations (see explanation below), Rao's projection-based method for processor allocation results in either wire lengths or local memory sizes that depend on the problem size. This case thus may benefit from more research.

Both the computability condition, and the procedure that computes an affine schedule, depend only on the affine dependence maps. Their computational complexity thus is independent of the problem size.

We also consider the case when the system 1) has an affine schedule, and 2) can be converted to an equivalent system of quasi-uniform recurrence equations [24]. In this case, one can eliminate the time dimension, simplifying the conversion of the system of affine recurrence equations into a system of quasi-uniform recurrence equations [25].

After broadcast removal, some existing algorithms for solving Toeplitz systems [7] are SAREs.

2 Preliminary Definitions

Example 1

The system of recurrence equations (SRE) below factors a symmetric Toeplitz matrix and its inverse into LDL^T:

$$1 \le i \le N, \qquad a_1(i,0) \equiv t_{i-1} \tag{1}$$

$$2 \le i \le N, \qquad a_2(i,0) \equiv t_{i-1} \tag{2}$$

$$1 \le j \le N-1, \qquad a_3(j,j) = a_2(j+1,j-1)/a_1(j,j-1) \tag{3}$$

$$j+1 \le i \le N, \qquad a_3(i,j) = a_3(i-1,j) \tag{4}$$

$$j+1 \le i \le N, \qquad a_1(i,j) = a_1(i-1,j-1) - a_3(i-1,j)a_2(i,j-1) \tag{5}$$

$$j+2 \le i \le N, \qquad a_2(i,j) = -a_3(i-1,j)a_1(i-1,j-1) + a_2(i,j-1) \tag{6}$$

$$a_2(1,0) = 1 \tag{7}$$

$1 \le j \le N-1$,

$$1 \le i \le j-1, \qquad a_3(i,j) = a_3(i+1,j) \tag{8}$$

$$1 \le i \le j+1, \qquad a_2(i,j) = -a_3(i+1,j)a_2(j+2-i,j-1) + a_2(i,j-1) \tag{9}$$

These recurrence equations are used to illustrate some of the following definitions, which are related to an SRE.

Index set: The set of points where an array is computed or used.

Domain of computation: The set of points C_i where an array a_i is computed (e.g., $C_2 = \{(i,j) | 1 \le j \le N-1,\ 1 \le i \le j+1\}$ in Eq. (9)).

Dependence map: A function δ_{ij} from the domain of computation of array a_j to the index set of a_i, on which the computation of a_j depends (e.g., $\delta_{32}(p) = p + \begin{pmatrix} 1 & 0 \end{pmatrix}^T$ in Eq. (9)).

Affine dependence: A dependence map of the form: $\delta_{ij}(p) = D_{ij}p + d_{ij}$ where $D_{ij} \in \mathbf{Z}^{n \times n}$, and $d_{ij} \in \mathbf{Z}^n$ (e.g., $\delta_{22}(p) = \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix} p + \begin{pmatrix} 2 \\ -1 \end{pmatrix}$ in Eq. (9)).

In the remainder of this paper, we assume that D_{ij} is nonsingular and integer, unless specified otherwise[1].

Uniform dependence: An affine dependence of the form: $\delta_{ij}(p) = p + d_{ij}$ (i.e., $D_{ij} = I$).

A system of affine [uniform] recurrence equations (SARE [SURE]): An SRE where the dependence maps are affine [uniform]. Moreover, *in this paper*, each array is computed in one recurrence equation for its entire domain of computation (e.g., Eqs. (8,9) are an SARE).

A system of quasi-uniform recurrence equations: An SRE which is uniform except for boundary points (see [24] for more details).

Convertible SARE: An SRE that can be converted to a system of quasi-uniform recurrence equations (Eqs. (8,9) are convertible, as is shown in [24]).

[1] Although nonsingularity is *not* required to prove Thms. 3.1 — 3.3, it is not useful to apply these theorems to affine dependences with singular linear parts.

Reduced dependence graph (RDG): A directed multigraph[2] with a node for each array in the SRE, and an arc from a_i to a_j for each dependence map δ_{ji} in the SRE [21, 8]. For example, the RDG of Ex. 1 is shown in Fig. 1.

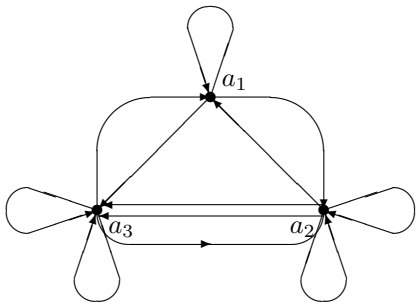

Figure 1: The RDG for the system of recurrence equations of Ex. 1.

Cycle dependence map: A composition of dependence maps associated with the arcs of a directed cycle in the RDG (not necessarily simple[3]). In most cases, we denote by δ_i a cycle dependence map which starts at a_i, and by δ_{ij} a direct dependence map of a_j on a_i (e.g., δ_{21} and δ_{12} in Ex. 1 constitute a cycle dependence map denoted δ_1).

3 Affine Schedule for an SARE

We establish necessary and sufficient conditions for the existence of an 'affine schedule' for an SARE. By *affine schedule* we mean the existence of a vector $\pi \in \mathbf{Z}^n$ and a set of constants $\{c_i \in \mathbf{Z}\}$, one for each array, such that **for all problem sizes**[4], if a variable $a_j(x_2)$ depends on a variable $a_i(x_1)$ (not necessarily directly), then $\pi^T x_1 + c_i < \pi^T x_2 + c_j$. This condition ensures that there exists a valid execution ordering. Array variable $a_i(x)$ is computed at time[5] $\pi^T x + c_i$. Vector $s = (c_1, c_2, \ldots, c_r, \pi^T)^T$ is called the *schedule vector*, where the SARE has r array variables. The above conditions are important, since not every SARE can be converted to an SURE. For example,

[2] A *directed multigraph*, $G = (N, A)$, is a directed graph that may have more than one arc directed from node v to node u, for any $v, u \in N$.

[3] A cycle is *simple* when it cannot be decomposed into smaller cycles.

[4] Rao [21], for example, has considered affine schedules that depend on the problem size. Since this leads to a serial execution ordering (in the direction where the problem size is used), we do not consider this case.

[5] By 'time' we really mean an ordering of the steps.

using the theorems proved in [24], it follows that the binary tree summation, $a(i) = a(2i) + a(2i+1)$, cannot be converted to an SURE. A discussion concerning the scheduling of uniform dependences has been considered by several authors (see, e.g., [6], [21], [4, § 7], [17], [22]).

3.1 A sufficient condition for the existence of an affine schedule

The following theorem characterizes the existence of an affine schedule for an SARE.

Theorem 3.1 *There exists an affine schedule for an SARE if and only if there exist $\pi \in \mathbf{Z}^n$ and a set $\{c_i \in \mathbf{Z}\}$ such that for all problem sizes, and for all direct dependences δ_{ij},*

$$\forall x \in C_j, \;\; \pi^T (D_{ij} - I)x + \pi^T d_{ij} + c_i - c_j < 0. \tag{1}$$

Proof. Property (1) can be rewritten as

$$\forall x \in C_j, \;\; \pi^T \delta_{ij}(x) + c_i < \pi^T x + c_j. \tag{2}$$

Let $\pi \in \mathbf{Z}^n$ and a set $\{c_i \in \mathbf{Z}\}$ be such that for all problem sizes, and for all direct dependences δ_{ij}, property (1) is true. Let $a_j(x_2)$ depend on $a_i(x_1)$ via $k+1$ direct dependences $\delta_{l_k j}, \ldots, \delta_{l_1 l_2}, \delta_{i l_1}$. Then $x_1 = \delta_{i l_1}(\delta_{l_1 l_2}(\ldots \delta_{l_k j}(x_2) \ldots))$. Multiplying both sides on the left by π^T, and using (2), we conclude that $\pi^T x_1 + c_i < \pi^T x_2 + c_j$. Thus, π and $\{c_i\}$ constitute an affine schedule.

The other direction is simple. We are given that an affine schedule exists. That is, there exist π, $\{c_i\}$, such that for every variable $a_j(x_2)$ depending on $a_i(x_1)$, $\pi^T x_1 + c_i < \pi^T x_2 + c_j$. Let the variable $a_j(x_2)$ depend directly on $a_i(x_1)$: $x_1 = \delta_{ij}(x_2)$. The π and $\{c_i\}$ of the affine schedule satisfy property (2), and thus (1) also is satisfied. ■

The proof of the previous theorem is not constructive. Finding a constructive characterization for the existence an affine schedule is an open problem. The following theorem provides a sufficient condition for the existence of an affine schedule which can be used to construct a schedule.

Theorem 3.2 *Let S be an SARE. If $\exists\, \pi \in \mathbf{Z}^n$ and $\{c_i \in \mathbf{Z}\}$ such that 1) $D_{ij}^T \pi = \pi$, for D_{ij}, the linear part of any direct dependence map, and 2) $\pi^T d_{ij} + c_i - c_j < 0$, for d_{ij}, the translation part of any direct dependence map, then $s = (c_1, c_2, \ldots, c_r, \pi^T)^T$ is a schedule vector for S.*

Proof. If π satisfies the premises, then

$$\pi^T (D_{ij} - I)x + \pi^T d_{ij} + c_i - c_j = (\pi^T D_{ij} - \pi^T)x + \pi^T d_{ij} + c_i - c_j = \pi^T d_{ij} + c_i - c_j < 0.$$

Therefore, π and $\{c_i\}$ satisfy property (1) in Thm. 3.1: $s = (c_1, c_2, \ldots, c_r, \pi^T)^T$ is a schedule vector. ■

The above condition (or criterion) is not necessary as is shown by the following example (a binary tree) $1 \leq i \leq N$, $a_1(i) = a_1(2i) + a_1(2i+1)$. In this SARE, $D_{11} = [2]$, and there is no π that satisfies $\pi^T D_{11} = \pi^T$, however, the schedule $\pi = [-1]$ is valid. The problem of finding a weaker sufficient condition that will include this case is open.

The following lemmata and theorem give a property that is equivalent to property (2) of Thm. 3.2.

Let C be the connection matrix of the RDG[6], such that there is a column for every arc (representing a direct dependence), and a row for each node (representing an array). There is a $+1$ in $C(ij)$ if arc j is not a self loop and is directed into node i, a -1 if arc j is not a self loop and is directed out of node i, and zero otherwise.

In what follows we denote by $\mathbf{N}$ the set of natural numbers (i.e., $0, 1, 2, \ldots$).

Lemma 3.3.1 *Given vectors $d \in \mathbf{Z}^n$, $b \in \mathbf{Z}^m$, and a connection matrix C, the following primal integer linear programming problem*

$$min\ d^T x \ \ subject\ to\ \ C^T x \geq b \ \ and\ x \in \mathbf{Z}^n$$

has an optimal solution if and only if its dual

$$max\ t^T b \ \ subject\ to\ \ Ct = d \ \ and\ t \in \mathbf{N}^m$$

has an optimal solution.

Proof. In general, the primal linear program has an optimal solution if and only if the dual has one. Since the matrix C^T is totally unimodular [18, § 13.2], any optimal solution is integer. Thus, if the primal integer linear program has an optimal solution, it is also an optimal solution to the linear program without the integer restriction. The dual thus has an optimal solution. The latter solution is integer, since C is totally unimodular. The other direction follows from the fact that the dual of the dual is the primal. ■

Let S be a matrix associated with a directed graph, that has a row for each simple cycle, and a column for each arc, such that there is a 1 in $S(ij)$ if arc j participates in the simple cycle i, and zero otherwise.

[6]Matrix C is not to be confused with set C_i, the domain of computation of array a_i.

Lemma 3.3.2 *The connection matrix $C_{r\times k}$ of a directed graph G, and the matrix $S_{m\times k}$ defined above satisfy*

$$Ct = 0, \;\; t \in \mathbf{N}^k \quad \textit{if and only if } \; t = S^T l, \;\; l \in \mathbf{N}^m.$$

Proof. If $t = S^T l$ then $Ct = CS^T l$. But $CS^T = 0$ because summing the columns of C corresponding to a cycle is zero. So, $Ct = 0$. Since S and l have only natural components, so does t.

Now, if $Ct = 0$, then we want to show that t defines a multiset of cycles in the directed graph (i.e., there is a set of cycles whose arcs consist of t_i times arc i for all i). Suppose this is false. Interpreting the components of vector t as units of flow, this means that there is some node whose in-flow does not equal its out-flow. Thus, the corresponding components of C cannot sum to zero. Therefore, t defines a set of simple cycles which can be formulated as $t = S^T l$. ■

The following theorem is mentioned without proof in [21] for SUREs. It appears in [4], although only the 'if' part is proved. Also in [4], a different approach is taken that causes the condition to be somewhat stronger (i.e., $\pi^T d_i \leq -v_i \cdot g$, where g is the greatest common divisor of the components of π).

Theorem 3.3 *Let S be an SARE, and $\pi \in \mathbf{Z}^n$ be such that $D_{ij}^T \pi = \pi$, for D_{ij}, the linear part of any direct dependence map. There exist $\{c_i \in \mathbf{Z}\}$ such that the translation part d_{ij} of any direct dependence map satisfies $\pi^T d_{ij} + c_i - c_j < 0$ if and only if for all simple cycles, the translation part d_i of a cycle dependence map that starts in one of the nodes in that cycle, and the number of arcs in that cycle, v_i, satisfy $\pi^T d_i \leq -v_i$.*

Proof. Since we are given $\pi^T D_{ij} = \pi^T$, we have

$$\sum_{\text{cycle } w} \pi^T d_{ij} = \pi^T d_w, \tag{3}$$

where d_w is the translation part of a cycle dependence map starting in any node in cycle w. Notice that there may be several distinct d_w, but $\pi^T d_w$ will be the same for all of them (this follows from the fact that $\pi^T D_{l_1 l_2} \cdot D_{l_2 l_3} \cdots D_{l_k i} d_{ij} = \pi^T d_{ij}$).

Suppose $\pi^T d_{ij} + c_i - c_j < 0$, for d_{ij}, the translation part of any direct dependence map. Since all terms above are integers, we have for all direct dependence maps $\pi^T d_{ij} + c_i - c_j \leq -1$. Let v_i denote the number of arcs in simple cycle w (which starts in node i), and d_i denote the translation

part of the dependence map of this cycle. Then,

$$\sum_{\text{cycle } w} \pi^T d_{ij} + c_i - c_j \leq \sum_{\text{cycle } w} -1 \Leftrightarrow \pi^T \sum_{\text{cycle } w} d_{ij} + \sum_{\text{cycle } w} c_i - c_j \leq -v_i \Leftrightarrow \pi^T d_i \leq -v_i.$$

Now suppose that $\pi^T d_i \leq -v_i$, where the vectors d_i are the translation parts of simple cycle dependence maps picked arbitrarily, one for each simple cycle, and v_i is the number of arcs in that cycle. We use the following notation:

$C_{r\times k}$**:** The connection matrix of the RDG as defined above.

$F_{m\times n}$**:** A matrix whose rows are the vectors d_i^T, the translation parts of cycle dependence maps, one picked for each simple cycle.

$B_{k\times n}$**:** A matrix whose rows are the vectors d_{ij}^T, the translation parts of all direct dependence maps.

$S_{m\times k}$**:** A matrix composed of a row for each simple cycle, and a column for each arc, as defined above.

$v_{m\times 1}$**:** A vector whose ith component is v_i, the number of arcs in the simple cycle i.

$h_{k\times 1}$**:** $(-1, -1, \ldots, -1)^T$.

We now can restate the supposition as $F\pi \leq -v$. Also, $F\pi = SB\pi$ (which follows from Eq. (3) above), and $v = -Sh$. Thus, $SB\pi \leq Sh$, i.e., $S(B\pi - h) \leq 0$.

Consider the following dual integer linear program:

$$\max\ t^T(B\pi - h)\ \text{ subject to }\ Ct = 0\ \text{ and } t \in \mathbf{N}^k.$$

According to Lemma 3.3.2, $Ct = 0$ if and only if $t = S^T l$ for some $l \in \mathbf{N}^m$. Thus, the above integer linear program can be restated as

$$\max\ l^T S(B\pi - h)\ \text{ subject to }\ l \in \mathbf{N}^m.$$

But as mentioned above, $S(B\pi - h) \leq 0$, so the maximum is zero (e.g., for $l = 0$), a finite optimum. According to Lemma 3.3.1, the corresponding primal linear program,

$$\min\ 0^T x\ \text{ subject to }\ C^T x \geq B\pi - h\ \text{ and } x \in \mathbf{Z}^r,$$

also must have an optimal solution. This primal linear program can be restated as finding a feasible integer solution to $C^T x \geq B\pi - h$. Replacing x by $-(c_1, c_2, \ldots, c_r)^T$ gives the following inequality

for every row of C^T (which corresponds to an arc in the RDG):

$$c_j - c_i \geq d_{ij}^T \pi + 1.$$

This is equivalent to

$$\pi^T d_{ij} + c_i - c_j \leq -1.$$

Since all terms are integers, this is equivalent to

$$\pi^T d_{ij} + c_i - c_j < 0,$$

the desired inequality. ■

3.2 Computing a schedule vector

Choosing the schedule vector is an important issue.[7] The case of SUREs is discussed by Rao [21], Delosme and Ipsen [4], and Shang and Fortes [22]. Here we use the same methodology as Rao [21], generalizing it to the case of SAREs. The schedule vector $s = (c_1, c_2, \ldots, c_r, \pi^T)^T \in \mathbf{Z}^{r+n}$ should satisfy the sufficient conditions of Thm. 3.2. Let the direct dependence maps have linear parts $D_{l_1,j_1}, D_{l_2,j_2}, \ldots, D_{l_k,j_k} \in \mathbf{Z}^{n \times n}$, with corresponding translations d_{l_i,j_i}. Since $D_{lj}^T \pi = \pi$, we have $A\pi = 0$, where

$$A = \begin{bmatrix} D_{l_1,j_1}^T - I \\ D_{l_2,j_2}^T - I \\ \vdots \\ D_{l_k,j_k}^T - I \end{bmatrix}_{kn \times n}.$$

We perform Gauss-Jordan reduction on the above matrix. If the system has a nontrivial solution, then after 1) Gauss-Jordan reduction, 2) a possible row and column permutation[8], and 3) deletion of zero rows, it is of the form:

$$\begin{bmatrix} I_{m \times m} & E \end{bmatrix}_{m \times n} \pi = 0.$$

The last $n - m$ components of π thus are free. Let $\hat{\pi}$ be the last $n - m$ components of π. We obtain

$$\pi = \begin{bmatrix} -E \\ I \end{bmatrix} \hat{\pi}. \tag{4}$$

[7] If the SARE will not subsequently undergo a nonlinear conversion, then the choice of affine schedule also can be optimized with respect to a cost function.

[8] For notational simplicity, we assume that no column permutation is needed.

The submatrix E is rational, since $D_{l_i,j_i} \in \mathbf{Z}^{n\times n}$, and Gauss-Jordan reduction does not introduce irrational numbers. As defined earlier, let B be a matrix whose rows are the vectors d_{l_i,j_i}^T, and C be the connection matrix of the RDG.

The sufficient condition for an affine schedule, requires that $[C^T B]s < 0$, where $s = (c_1, c_2, \ldots, c_r, \pi^T)^T$. Using Eq. 4, we have

$$\left[C^T\ B \cdot \left[\begin{array}{c} -E \\ I \end{array}\right]\right] \hat{s} < 0 \ \text{ where } \hat{s} = (c_1, c_2, \ldots, c_r, \hat{\pi}^T)^T.$$

A feasible solution for this system of strict inequalities can be obtained with the ellipsoid algorithm[18]. Since C, B, and s have only integer entries, we can solve equivalently for:

$$\left[C^T\ B \cdot \left[\begin{array}{c} -E \\ I \end{array}\right]\right] \hat{s} \leq h, \quad \text{where } h = (-1, -1, \ldots, -1)^T.$$

The desired vector π is

$$\pi = \left[\begin{array}{c} -E \\ I \end{array}\right] \hat{\pi}.$$

The resulting s is rational. By scaling, we obtain an integer primitive vector s (i.e., its components have greatest common divisor 1). This procedure, of course, can be used to compute an integer schedule vector for an SURE.

If we use Thm. 3.3, there is another way to compute an affine schedule: compute the matrix E as mentioned above, and then instead of solving simultaneously for

$$\left[C^T\ B \cdot \left[\begin{array}{c} -E \\ I \end{array}\right]\right] \hat{s} \leq h,$$

find all simple cycles in the RDG (this number can be exponential with respect to the number of nodes, but in practice is small), and then solve

$$F \left[\begin{array}{c} -E \\ I \end{array}\right] \hat{\pi} \leq -v$$

where v_i is the number of arcs in the simple cycle i in the RDG, and F has a row $[F]_i = d_i$ for a translation part d_i picked for any simple cycle. The desired vector π is

$$\pi = \left[\begin{array}{c} -E \\ I \end{array}\right] \hat{\pi}.$$

The solution π, can be made integer (by scaling it by a positive integer). We then can solve for the constants $\{c_i \in \mathbf{Z}\}$ by solving

$$C^T(c_1, c_2, \ldots, c_r)^T \leq h - B\pi.$$

Thm. 3.3 assures us that there is an integer set $\{c_i\}$ satisfying the above inequalities.

Example 2

Consider the following SARE[9]:

$1 \leq j \leq N-1$,

$$1 \leq i \leq j-1, \qquad a_3(i,j) = a_3(i+1, j-1) \tag{1}$$

$$1 \leq i \leq j+1, \qquad a_2(i,j) = -a_3(i+1,j)a_2(j+2-i, j-1) + a_2(i, j-1) \tag{2}$$

The dependence maps are:

$$\delta^1_{22}(i,j) = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 2 \\ -1 \end{pmatrix} \quad \delta^2_{22}(i,j) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

$$\delta_{32}(i,j) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \delta_{33}(i,j) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} i \\ j \end{pmatrix} + \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

The corresponding matrix $A = \begin{bmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$. After Gauss-Jordan reduction and deleting zero rows we get: $\begin{pmatrix} 1 & 0 \end{pmatrix}\pi = 0$. The last component is free, and the first component must be 0. The connection matrix C, and the matrix B in this case are:

$$C = \begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & -1 \\ 0 & -1 \\ 1 & -1 \\ 1 & 0 \end{bmatrix}.$$

The system of inequalities we have to solve is thus:

$$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & -1 \\ 0 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix} \hat{s} \leq \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \end{pmatrix}.$$

[9]Note that Eq. (1) is not the same as Eq. (8) of Example 1.

A solution to this set of inequalities is: $\hat{s} = (\ 1 \quad 0 \quad 1\ )^T$. The affine schedule is thus the vector $\pi = (\ 0 \quad 1\ )^T$ and the set $\{c_2, c_3\} = \{1, 0\}$.

3.3 A NECESSARY CONDITION FOR THE EXISTENCE OF AN AFFINE SCHEDULE

The premises of Thm. 3.2, in general, are not necessary. Some other criterion is needed to decide if an affine schedule does not exist. The following definitions are used.

Domain size parameters: Those parameters in the SRE which instantiate the domains of computation (e.g., N in Ex. 1).

Dependence length of an index point p with respect to a cycle dependence map δ_i, is the number of cycle compositions which can be made starting with $a_i(p)$ until (and including) a cycle on which an index of one of the arrays (not necessarily a_i) is outside its domain of computation. This dependence length is denoted by $\gamma_i(p)$. For a set S of index points, the notation $\gamma_i(S)$ is used to denote the minimum dependence length of points in S. If $H \subset \mathbf{R}^n$, then $\gamma_i(H) = \min_{p \in H \cap \mathbf{L}_i^n} \{\gamma_i(p)\}$[10].

n-dimensional system of recurrence equations: An SARE which satisfies the following property: For every cycle dependence map δ_i, $\forall k \in \mathbf{N}$, there exist domain size parameters such that the domain of computation C_i contains H, a k^n hypercube, and $\gamma_i(H) > 1$ (e.g., the SARE in Ex. 2 is 2-dimensional).

In the definition above, and throughout the paper, when we say "the domain of computation C_i contains hypercube H," we mean every point in $H \bigcap \mathbf{L}_i^n$ is in C_i.

We now focus the discussion to SAREs that can be converted to equivalent SREs which are quasi-uniform (i.e., they are uniform except for boundary points). These are called *convertible* SAREs. It is proved in [24] that, an SARE is convertible if and only if every cycle dependence map has a linear part which is a root of I (thus the linear part of every dependence map is nonsingular). The following lemmata and theorem establish a necessary condition for the existence of an affine schedule.

Lemma 3.4.1 *An n-dimensional SARE has the following property: For every nonzero vector v, and every constant $c \in \mathbf{R}^+$, and every domain of computation C_i, and every cycle dependence map δ_i, there exist domain size parameters and $x \in C_i$ such that $|v^T x| > c$, and $\gamma_i(x) > 1$.*

[10] $\mathbf{L}_i^n$ is the lattice on which array a_i is defined.

Proof. In the following proof we assume that $\mathbf{L}_i^n = \mathbf{Z}^n$. The generalization to any lattice is straight forward.

Since the SARE is n-dimensional, there exist domain size parameters such that there exists in C_i a hypersphere R^n, where $R = \frac{c+\|v\|\cdot\sqrt{n}}{\|v\|}$, and $\gamma_i(R) > 1$. Suppose the center of the sphere is r, and $v^T r > 0$. Then choose

$$x = r + \frac{c + \frac{1}{2}\sqrt{n}\|v\|}{\|v\|^2} \cdot v + u,$$

where u is a vector that is used to reach the nearest lattice point. Since the distance of any point to the nearest lattice point is at most $\frac{1}{2}\sqrt{n}$, x is in the hypersphere. Therefore,

$$|v^T x| = |v^T r + c + \tfrac{1}{2}\sqrt{n}\|v\| + v^T u| > c + \tfrac{1}{2}\sqrt{n}\|v\| - \|v\|\tfrac{1}{2}\sqrt{n} = c.$$

The inequality above follows from three facts: 1) $v^T r > 0$, 2) $\|u\| \leq \frac{1}{2}\sqrt{n}$, and 3) u, in the worst case, has the opposite direction of v.

If $v^T r < 0$, then choose

$$x = r - \frac{c + \frac{1}{2}\sqrt{n}\|v\|}{\|v\|^2} \cdot v + u.$$

According to the argument above, $|v^T x| > c$. If $v^T r = 0$, then choose the same x as the case for $v^T r > 0$. But if $v^T u = -\|v\|\frac{1}{2}\sqrt{n}$, then change u to $-u$ (i.e., choose the lattice point in the opposite direction). Again x is in the sphere, and $|v^T x| > c$. ■

Lemma 3.4.2 *If an SARE has a schedule vector $s = (c_1, c_2, \ldots, c_r, \pi^T)^T \in \mathbf{Z}^{r+n}$, then for all cycle dependence maps δ_i,*

$$\forall x \in C_i,\ \pi^T(D_i - I)x + \pi^T d_i \leq -v_i,$$

where v_i is the number of arcs in the RDG for cycle i.

Proof. Using Eq. 2 in Thm. 3.1 and the fact that all terms are integers, we have:

$$\pi^T(D_i - I)x + \pi^T d_i = \pi^T\delta_i(x) - \pi^T x = \pi^T\delta_{il_1}(\delta_{l_1l_2}(\ldots(\delta_{l_ki}(x))\ldots)) - \pi^T x$$

$$\leq c_{l_1} - c_i - 1 + \pi^T\delta_{l_1l_2}(\ldots(\delta_{l_ki}(x))\ldots) - \pi^T x \leq$$

$$c_{l_2} - c_i - 2 + \pi^T(\ldots(\delta_{l_ki}(x))\ldots) - \pi^T x \leq \cdots \leq -v_i + \pi^T(x) - \pi^T(x) = -v_i.$$

■

Theorem 3.4 *Let S be an n-dimensional and convertible SARE. If there exists an affine schedule for S, with a schedule vector $s = (c_1, c_2, \ldots, c_r, \pi^T)^T \in \mathbf{Z}^{r+n}$, then $D_i^T\pi = \pi$, for every D_i that is the linear part of a cycle dependence map, and $\pi^T d_i \leq -v_i$, for every d_i that is the translation part of a cycle dependence map, where v_i is the number of arcs in the RDG in that cycle.*

Proof. Let L be an integer such that $D_i^L = I$. (Since every cycle is a strongly-connected component, and the SARE is convertible, such an L must exist — see [24]). Let $M_i = \max\{|\pi^T(D_i^1 + D_i^2 + \ldots + D_i^{L-1})d_i|,\ |\pi^T d_i|\}$. Suppose that the theorem is false. If $\exists\ D_i,\ D_i^T\pi \neq \pi$, then $\pi^T(D_i - I) \neq 0$. By Lemma 3.4.1, $\exists\ x \in C_i$ that satisfies either $\pi^T(D_i - I)x > M_i$, or $\pi^T(D_i - I)x < -M_i$. If there exists a schedule vector s, then its linear part π satisfies one of the following:

(i) $\forall\ D_i, D_i^T\pi = \pi$ but $\exists\ d_j, \pi^T d_j > -v_i$

(ii) $\exists\ D_i, D_i^T\pi \neq \pi$, and $\exists\ x \in C_i$ such that $\pi^T(D_i - I)x > M_i$

(iii) $\exists\ D_i, D_i^T\pi \neq \pi$, and $\exists\ x \in C_i$ such that $\pi^T(D_i - I)x < -M_i$.

We prove that any π which satisfies one of the above cases contradicts the existence of an affine schedule, which according to Lemma 3.4.2 implies that for any cycle dependence map δ_i,

$$\forall x \in C_i,\ \pi^T(D_i - I)x + \pi^T d_i \leq -v_i. \tag{5}$$

In case (i), we have $\pi^T(D_j - I) = 0$. So $\pi^T(D_j - I)x + \pi^T d_j = \pi^T d_j > -v_i$. Therefore, (5) is not satisfied.

In case (ii), we have an x in the domain of computation C_i such that $\pi^T(D_i - I)x > M_i \geq -\pi^T d_i$. Therefore, (5) is not satisfied.

In case (iii), we have

$$\pi^T(D_i - I)x < -M_i \leq \pi^T(D_i^1 + D_i^2 + \ldots + D_i^{L-1})d_i. \tag{6}$$

According to Lemma 3.4.1, $\gamma_i(x) > L$ (taking δ_i^L in Lemma 3.4.1 to be the cycle dependence). For the remainder of the proof, we drop the subscript i. Let δ^i denote the i-fold composition of δ. Then

$$\sum_{i=0}^{L-1}(\pi^T(D - I)\delta^i(x) + \pi^T d) = L\pi^T d + \pi^T \sum_{i=0}^{L-1}(D\delta^i(x) - \delta^i(x))$$

$$= L\pi^T d + \pi^T \sum_{i=0}^{L-1}(\delta^{i+1}(x) - d - \delta^i(x)) = \pi^T(\delta^L(x) - \delta^0(x))$$

$$= \pi^T(D^{L-1} + D^{L-2} + \cdots + I)d > \pi^T(D - I)x + \pi^T d$$

(The inequality above follows from inequality 6). Subtracting the right hand side of the inequality from both sides,

$$\sum_{i=1}^{L-1}(\pi^T(D - I)\delta^i(x) + \pi^T d) > 0$$

At least one term in the above sum must be positive, implying that (5) is not satisfied. ■

As an example, consider the convertible SARE in Ex. 1 Eqs. (8,9). If there exists an affine schedule, then $D_2^T\pi = \pi$, where $D_2 = \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix}$. Thus $\pi^T = (\ 0 \quad 1\ )$ or a multiple of it. There is no other possibility. But, $d_3^T = (\ 1 \quad 0\ )$, so $\pi^T d_3 = 0$, which implies that the necessary conditions are not met. This SARE thus does *not* have an affine schedule. (Interestingly, the index sets of this SARE can be manipulated such that it has an affine schedule [24].)

3.4 Characterizing the existence of an affine schedule for a convertible SARE

Let A be an SARE with a strongly-connected RDG[11]. This kind of SARE is called a strongly-connected SARE. Affine transformations can be applied to the index sets of A's arrays such that the resulting RDG has a spanning tree whose arcs have uniform dependence maps associated with them. Such a transformation has been suggested in [4]. A similar procedure is as follows. Choose a spanning tree of the RDG[12]. Start from the root of the tree, and proceed down level by level, applying the affine transformation δ_{ij}^{-1} on the index set of the array a_i (i.e., $p \rightarrow D_{ij}^{-1}p - D_{ij}^{-1}d_{ij}$), where a_j is its parent[13]. All the appropriate dependence maps are updated after each transformation. This procedure is called a *tree conversion*. The tree is not unique.

Such a tree conversion is shown in Fig. 2. The linear parts of the dependence maps are shown

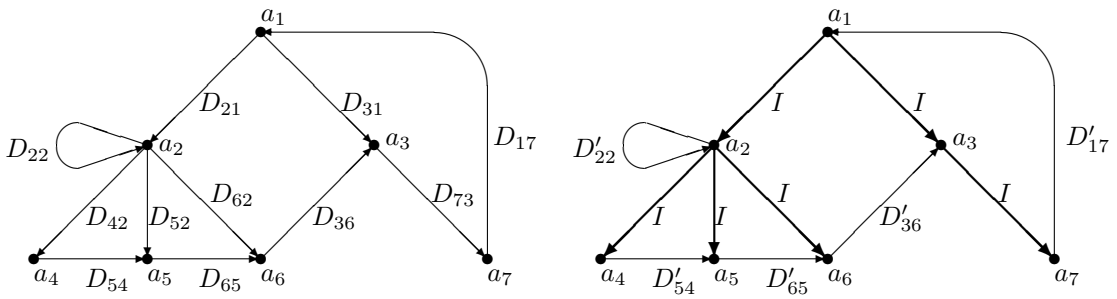

Figure 2: The RDG before (on the left) and after (on the right) the tree conversion.

near each arc. The array a_1 was chosen to be the root, and then the index set of the array a_2 is

[11] That is, an RDG where there is a directed path from every node to every other node.

[12] That is, a subgraph of the RDG that is a directed tree which contains all the RDG's nodes.

[13] The transformation δ_{ij}^{-1} is used instead of D_{ij}^{-1}, ensuring that integer lattice points map to integer lattice points.

mapped by δ_{21}^{-1}. By doing so, the dependence map δ_{21} becomes uniform: its linear part becomes I as shown in the figure. The updating that should be done after this transformation is: $\delta_{22} \rightarrow \delta_{21}^{-1}\delta_{22}\delta_{21}$, $\delta_{42} \rightarrow \delta_{42}\delta_{21}$, $\delta_{52} \rightarrow \delta_{52}\delta_{21}$, $\delta_{62} \rightarrow \delta_{62}\delta_{21}$. The index set of a_2 also is updated. The next step is to change δ_{31} into a uniform dependence map. This is done by mapping the index set of a_3 by δ_{31}^{-1}. The same process is applied; this continues until the whole tree has uniform dependences associated with its arcs. The result also is shown in Fig. 2. The new linear parts which do not correspond to tree arcs are:

$$D'_{22} = D_{21}^{-1}D_{22}D_{21} \qquad D'_{54} = D_{21}^{-1}D_{52}^{-1}D_{54}D_{42}D_{21}$$
$$D'_{65} = D_{21}^{-1}D_{62}^{-1}D_{65}D_{52}D_{21} \quad D'_{36} = D_{31}^{-1}D_{36}D_{62}D_{21} \qquad D'_{17} = D_{17}D_{73}D_{31}$$

The translation parts are updated accordingly.

We now focus the discussion on a strongly-connected SARE that is convertible (i.e., every cycle dependence map has a linear part that is a root of I). We also assume that a tree conversion has been done on the SARE, as defined above. The following theorem establishes necessary and sufficient conditions for the existence of an affine schedule. This condition ensures that the SARE can be embedded in spacetime such that time is decoupled from space [25]. That is, there exists matrix M, such that for every linear part D_{ij} of direct dependence map δ_{ij}, $MD_{ij}M^{-1}$ is of the form $\begin{pmatrix} 1 & 0 \\ 0 & D'_{ij} \end{pmatrix}$, where $D'_{ij} \in \mathbf{Z}^{(n-1)\times(n-1)}$. Conversion of the SARE to one with quasi-uniform recurrence equations thus can be simplified.

Theorem 3.5 *Let S be a convertible strongly-connected SARE after a tree conversion. The following three statements are equivalent: 1) There exists an affine schedule for S, with a schedule vector $s = (c_1, c_2, \ldots, c_r, \pi^T)^T \in \mathbf{Z}^{r+n}$; 2) $D_i^T\pi = \pi$, and $d_i^T\pi \leq -v_i$ for all simple cycles dependence maps, where D_i is the linear part and d_i is the translation part and v_i is the number of arcs in the RDG for that cycle dependence map; 3) $D_{ij}^T\pi = \pi$, for D_{ij}, the linear part of any direct dependence map, and $\pi^T d_{ij} + c_i - c_j < 0$, for d_{ij}, the translation part of any direct dependence map.*

Proof. The proof that 1 implies 2 appears in Thm. 3.4. The proof that 3 implies 1 appears in Thm. 3.2. We now prove that 2 implies 3. Since the SARE is convertible, and has undergone a tree conversion, the set of distinct compositions of linear parts of dependence maps equals the set of the distinct linear parts of all cycle dependence maps (proved in [24]). Thus, for every linear part D_{ij} of a direct dependence map, there exists an equal linear part of a cycle dependence map, and therefore there exist a finite number of linear parts $D_1, D_2, \ldots, D_k$ of simple cycle dependence maps, such

that $D_{ij} = D_1 D_2 \cdots D_k$. Thus,

$$D_{ij}^T \pi = D_k^T \cdots D_2^T D_1^T \pi = \pi.$$

The other property follows directly by using Thm. 3.3. ■

The above theorem proves that the sufficient condition of Thm. 3.2 and the necessary condition of Thm. 3.4 are equivalent in the case of a convertible strongly-connected SARE after a tree conversion.

3.5 Computability in SAREs

The following theorem relates computability to the existence of an affine schedule. It establishes the set of SAREs with an affine schedule as a subset of the set of computable SAREs.

Theorem 3.6 *If an SARE has an affine schedule, then it is computable.*

Proof. If the SARE is not computable then there exists a point x in the domain of computation C_i, such that $a_i(x)$ depends on itself (not necessarily directly). This requires $\pi^T x < \pi^T x$, contradicting the definition of an affine schedule. ■

Computability is not equivalent to the existence of an affine schedule, as shown by the following examples:

$$\begin{aligned} \delta_{21}(p) &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} p + \begin{pmatrix} 0 \\ -1 \end{pmatrix} \\ \delta_{12}(p) &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} p + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned} \tag{7}$$

An SARE with the above dependence maps has no affine schedule, but is computable. After applying a tree conversion, there is an affine schedule.

Now consider an SARE S with the following two dependence maps.

$$\begin{aligned} \delta_{11}(p) &= p + \begin{pmatrix} 0 \\ -1 \end{pmatrix} \\ \delta_{11}(p) &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} p + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{aligned} \tag{8}$$

This SARE also lacks an affine schedule, but is computable. Here unlike the first example, there is no affine schedule even after conversion to an SRE with quasi-uniform dependences. There however

is a serial schedule. Fig. 3 depicts a dependence graph for an instance of this SRE. The numbers on the figure are the only valid schedule.

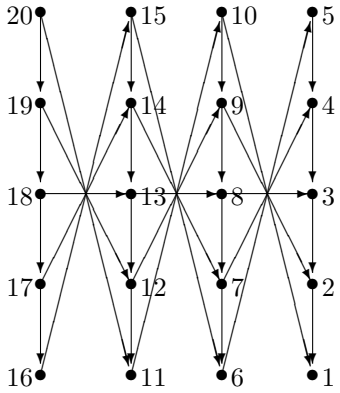

Figure 3: The dependence graph for SARE S. The numbers indicate a serial schedule.

The last example is an SURE which does not have an affine schedule, though is computable. Given by Rao, it is an algorithm for a two dimensional filter (for details — see [21]).

$$1 \le i \le N, \qquad a_1(i,0,0) = a_1(i,-1,,0) = a_1(i,0,-1) \equiv 0 \tag{1}$$

$$1 \le j \le M,$$

$$1 \le k \le M, \qquad a_2(N,j,k) \equiv 0, \;\; a_3(1,j,k) \text{ given} \tag{2}$$

$$1 \le i \le N,$$

$$1 \le j \le M,$$

$$1 \le k \le M, \;\; a_1(i,j+1,k+1) = f(a_1(i,j,k), a_2(i,j,k), a_3(i,j,k)) \tag{3}$$

$$a_2(i-1,j,k) = g(a_1(i,j,k), a_2(i,j,k)) \tag{4}$$

$$a_3(i+1,j,k) = h(a_1(i,j,k), a_2(i,j,k), a_3(i,j,k)) \tag{5}$$

The last two examples have schedules **that depend on the problem size**. (For the example in Fig. 3, we can use $\pi^T = [-5, 1]$, and $\pi^T = [-N, 1]$, in general. For the last example, Rao [21] provides a schedule that depends on the problem size.)

Another interesting theorem is that a condition, weaker than the necessary condition for the existence of an affine schedule in convertible SAREs (see Thm. 3.4), also implies that the SARE is computable.

Theorem 3.7 *An SARE is computable if there exists a vector $\pi \in \mathbf{Z}^n$ such that $D_i^T \pi = \pi$, for every D_i that is the linear part of a cycle dependence map, and $\pi^T d_i < 0$, for every d_i that is the*

translation part of a cycle dependence map.

Proof. Suppose the SARE is not computable, then $D_i x + d_i = x$ for some linear part D_i, and translation part d_i of a cycle dependence map in the SARE, and $x \in C_i$. Multiplying by π^T from the left on both sides gives:

$$\pi^T(D_i x + d_i) = \pi^T x \Rightarrow \pi^T x + \pi^T d_i = \pi^T x \Rightarrow \pi^T d_i = 0,$$

which is a contradiction. ■

As can be shown by the three examples above, the sufficient condition of Thm. 3.7 is not necessary. It is open as to whether or not there is a characterization of computable SAREs, which is simple to compute.

The following theorem, a variation on a theorem by Delosme and Ipsen [4], identifies a property of computable SAREs. The property is weaker than the one established in Thm. 3.4, since computability is a weaker property than the existence of an affine schedule (Thm. 3.6).

Theorem 3.8 *If an n-dimensional SARE is computable and convertible, then the linear part D of any cycle dependence map is a root of I with at least one eigenvalue $\lambda = 1$.*

Proof. The essence of the short proof of this is given by Delosme and Ipsen [4]. ■

An SARE is not necessarily computable if 1 is an eigenvalue of all the linear parts D, and for all the linear parts we have $D^L = I$. For example, the dependence map $\delta(p) = Ip + 0$ has these properties, but clearly is not computable.

References

[1] Peter R. Cappello. *VLSI Architectures for Digital Signal Processing.* PhD thesis, Princeton University, Princeton, NJ, Oct 1982.

[2] Peter R. Cappello and Kenneth Steiglitz. Unifying VLSI array design with linear transformations of space-time. In F. P. Preparata, editor, *Advances in Computing Research*, pages 23–65, JAI Press, Inc., 1984.

[3] Jean-Marc Delosme and Ilse C. F. Ipsen. An illustration of a methodology for the construction of efficient systolic architectures in VLSI. In *Proc. 2nd Int. Symp. on VLSI Technology, Systems and Applications*, pages 268–273, Taipei, 1985.

[4] Jean-Marc Delosme and Ilse C. F. Ipsen. *Systolic Array Synthesis: Computability and Time Cones.* Technical Report Yale/DCS/RR-474, Yale, May 1986.

[5] José A. B. Fortes and Dan I. Moldovan. Parallelism detection and algorithm transformation techniques useful for VLSI architecture design. *J. Parallel Distrib. Comput*, 2:277–301, Aug. 1985.

[6] José A. B. Fortes and F. Parisi-Presicce. Optimal linear schedules for the parallel execution of algorithms. In *Int. Conf. on Parallel Processing*, pages 322–328, Aug. 1984.

[7] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The John Hopkins University Press, 1983.

[8] Richard M. Karp, Richard E. Miller, and Shmuel Winograd. The organization of computations for uniform recurrence equations. *J. ACM*, 14:563–590, 1967.

[9] Richard M. Karp, Richard E. Miller, and Shmuel Winograd. Properties of a model for parallel computations: determinacy, termination, queueing. *SIAM J. Appl. Math*, 14:1390–1411, 1966.

[10] Sun-Yuan Kung. *VLSI Array Processors.* Prentice-Hall, Englewood Cliffs, NJ, 1988.

[11] Charles E. Leiserson, Flavio M. Rose, and James B. Saxe. Optimizing synchronous circuitry by retiming. In *Proc. Third Caltech Conf. on VLSI*, Computer Science Press, Rockville, MD, 1983.

[12] Charles E. Leiserson and James B. Saxe. Optimizing synchronous systems. In *Proc. IEEE 22nd Annual Symp. Foundations of Computer Science*, Oct 1981.

[13] Willard L. Miranker and Andrew Winkler. Spacetime representations of computational structures. *Computing*, 32:93–114, 1984.

[14] Dan I. Moldovan. On the analysis and synthesis of VLSI algorithms. *IEEE Trans. Comput.*, C-31:1121–1126, Nov. 1982.

[15] Dan I. Moldovan. On the design of algorithms for VLSI systolic arrays. *Proc. IEEE*, 71(1):113–120, Jan. 1983.

[16] Dan I. Moldovan and José A. B. Fortes. Partitioning and mapping algorithms into fixed systolic arrays. *IEEE Trans. on Computers*, C-35(1):1–12, Jan. 1986.

[17] Mathew T. O'Keefe and José A. B. Fortes. A comparative study of two systematic design methodologies for systolic arrays. *Proc. Int. Conf. Parallel Processing*, 672–675, Aug. 1986.

[18] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc, Englewood Cliffs, NJ, 1982.

[19] Patrice Quinton. Automatic synthesis of systolic arrays from uniform recurrent equations. In *Proc. 11th Ann. Symp. on Computer Architecture*, pages 208–214, 1984.

[20] I. V. Ramakrishnan, D. S. Fussell, and A. Silberschatz. Mapping homogeneous graphs on linear arrays. *IEEE Trans. Computers*, C-35(3):189–209, Mar. 1986.

[21] Sailash K. Rao. *Regular Iterative Algorithms and Their Implementation on Processor Arrays.* PhD thesis, Stanford University, October 1985.

[22] Weija Shang and José A. B. Fortes. Time optimal linear schedules for algorithms with uniform dependencies. In *Int. Conf. on Systolic Arrays*, pages 393–402, San Diego, May 1988.

[23] Yiwan Wong and Jean-Marc Delosme. Broadcast removal in systolic algorithms. In *Int. Conf. on Systolic Arrays*, pages 403–412, San Diego, May 1988.

[24] Yoav Yaacoby and Peter R. Cappello. *Converting Affine Recurrence Equations to Quasi-Uniform Recurrence Equations.* Technical Report 18, Dept. Computer Science, UCSB, Santa Barbara, CA 93106, Feb. 1988.

[25] Yoav Yaacoby and Peter R. Cappello. *Decoupling the Dimensions of a System of Affine Recurrence Equations.* Technical Report 12, Dept. Computer Science, UCSB, Santa Barbara, CA 93106, Apr. 1988.