JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING **33,** 24–32 (1996)
ARTICLE NO. 0021

# Designing Lower-Dimensional Regular Arrays for Algorithms with Uniform Dependencies[1]

JONG-CHUANG TSAY AND PEN-YUANG CHANG

*Institute of Computer Science and Information Engineering, College of Engineering, National Chiao Tung University, Hsinchu, Taiwan 30050, Republic of China*

**In this paper, we will propose a polynomial-time method to design *m*-dimensional regular arrays for *n* ($n \geq m + 1$) dimensional algorithms with uniform dependencies, regular algorithms. The proposed method has two steps: the first one is to transform an independently partitioned regular algorithm to a new one, which has an identity dependence matrix. We call this step *identity transformation*, which is an affine one. In the second step, we propose a spacetime mapping in a fixed form to map the new regular algorithm to a lower-dimensional regular array. Thus, an *affine spacetime mapping* is constructed by combining the identity transformation and the fixed form's spacetime mapping together. With the proposed affine spacetime mapping, the original regular algorithm can be mapped to a lower-dimensional regular array in polynomial time, which depends only on the number of dimensions of the regular algorithm. Meanwhile, the designed regular array is asymptotically optimal in time and space.** © 1996 Academic Press, Inc.

## 1. INTRODUCTION

The systolic array or regular array [1, 2] is a special purpose parallel device, in which the processing elements (PEs) are regularly and locally connected. It follows that systolic architecture is very suitable for implementing on VLSI chips. Nevertheless, due to the limitations of current technology, most popular regular arrays have lower dimensions, such as 1-dimensional (1D) linear arrays [3] and 2D mesh ones [4].

Designing regular arrays from a system of recurrence equations includes two major steps: the first step is *regularization* [5], or *uniformization* [6]. After regularization, the original system of recurrence equations is transformed to an equivalent *system of uniform recurrence equations* (SURE) [7] or a *regular iterative algorithm* (RIA) [5]. An SURE can be further partitioned [8–10] to several independent ones, in which each one is independent on the others and can be treated apart.

The second step is to decide a *conflict-free* spacetime mapping [11, 12] to map the SURE to a regular array. The spacetime mapping is in general represented as a transformation matrix. The first row and the rest of the transformation matrix are respectively the time mapping vector and the space mapping matrix. Three kinds of conflict-free condition are necessary to be satisfied: Dependence conflict-free: if and only if the precedence constraints imposed by the SURE are satisfied; computation conflict-free: if and only if two different computations are not executed on the same PE at the same time; and link conflict-free: if and only if every dependence vector of the SURE is mapped to one and only one interconnection link of the array.

It is easy to check these conflict-free conditions for mapping an $n$D SURE to an $(n - 1)$D regular array [13]. So previous researchers put their efforts on how to find an optimal schedule vector [14] and an optimal processor assignment [15], as well as designing more efficient regular arrays [16]. Yet, when mapping an $n$D SURE to an $m$D regular array, $n > m$ and $m \neq n - 1$, the check of computation conflict-free condition cannot be done by only examining whether the dependence matrix is nonsingular. And link conflict may occur and should be detected. Such that, the designer should select two index vectors from the computation domain and then check the conditions of the computation and link conflict-free; then the above check is repeated until all pairs of index vectors in the computation domain have been examined. This makes the mapping problem more difficult and time-consuming.

Rao [5] proposed an iterative procedure (Procedure 3.3 in his dissertation) to design lower-dimensional regular arrays. He guarantees the designed regular array satisfying the conditions of dependence and computation conflict-free. However, all link conflict-free may occur in his design. Lee and Kedem [11] proposed the necessary and sufficient conditions of link conflict-free. Yet, they have to examine the whole computation domain to check whether the mapping is computation and link conflict-free. Shang and Fortes [12] derived some necessary and sufficient conditions for computation conflict-free. In some cases, the time-consuming examination of the whole computation domain can be avoided in their procedure. However, finding a closed form necessary and sufficient conditions on conflict-free mapping for general cases is still open and very difficult. Besides, the link conflict may occur in their designs. Ganapathy and Wah [17] designed the lower dimensional regular arrays by the parameter-based method rather than spacetime mapping. They described the conditions of con-

[1] This work was supported by the National Science Council, Republic of China, under Contract NSC85-2213-E-009-103.

flict-free mapping by bounding the parameters of period, velocity, and data distribution. The optimal design can be found in polynomial time. However, too many equations are established by these parameters when the dimension of SURE becomes large, which results in computation difficulty.

The major problem on designing lower dimensional regular arrays for SUREs is: *it is time-consuming to check conditions of computation and link conflict-free, because the whole computation domain is necessary to be examined.* In contrast to previous methods, we propose a two-step one that can avoid checking conditions every time. The first step is to transform an independently partitioned regular algorithm to a new one, which has an identity dependence matrix. We call this step *identity transformation.* Since the original regular algorithm does not always have a unimodular dependence matrix, the identity transformation is a nonunimodular one; thus, it is important to keep every index vector in the new regular algorithm with integral elements. So, the identity transformation is not linear but affine [18, 19], which has a linear part and a translation one. The translation part is to guarantee that the new regular algorithm has integral index vectors. In the second step, we propose a spacetime mapping in a fixed form to map the new regular algorithm to a lower dimensional regular array. Thus, an *affine spacetime mapping* is constructed by combining the identity transformation and the fixed form's mapping together. With the proposed affine spacetime mapping, the original regular algorithm can be mapped to a lower dimensional regular array in polynomial time, which depends only on the dimension of the regular algorithm. Meanwhile, the designed regular array is asymptotically optimal in time and space.

One thing should be mentioned here: a similar design concept has been proposed in [20] to design modular extensible linear arrays for regular algorithms. Yet, in that paper, we assume the original SURE has a unimodular dependence matrix. Thus, the affine transformation and affine spacetime mapping are not mentioned in that paper. On the other hand, only 1D (linear) arrays can be synthesized in that paper. Nevertheless, we can design not only 1D regular arrays but also other lower-dimensional ones here.

Here is an outline of following sections: some preliminary definitions and the design concepts are given in Section 2. In Section 3, we will propose an affine spacetime mapping, and show that it is correct and asymptotically optimal. Finally, the concluding remarks are in Section 4.

## 2. DEFINITION AND CONCEPT

In the following, we will give the definitions of SURE and regular array. Two types of SURE are identified: one is the independently partitioned SURE and the other is the SURE with identity dependence matrix. The concepts of affine transformation and affine spacetime mapping are also given in this section.

Definition 2.1 (SURE) [7, 21]. *A system of uniform recurrence equations* (SURE) is a set of recurrence equations of the form $v_2(\vec{j}_2) = f_2(..., v_1(\vec{j}_1), ...)$, where $\vec{j}_1, \vec{j}_2 (\in J^n \subset Z^n)$ are index vectors, $v_1, v_2$ variables, $f_2$ a function, and $\vec{j}_2 - \vec{j}_1 (\equiv \vec{d}_1 \in D)$ a dependence vector with constant integral elements.

Herein, a small letter with a head $\vec{}$ denotes a column vector. We use $\mathscr{A}_S = (J^n, D)$ to denote an SURE, because only its structural information is useful in here. $J^n$ is called the computation domain of $\mathscr{A}_S$. $D$ is in general represented by a matrix form, called *dependence matrix*, in which each column represents one dependence vector. det$D$ denotes the determinant of $D$. For simplicity, we only consider that $J^n = \{[j_1 j_2 \cdots j_n]^{\mathrm{T}} | 1 \leq j_i \leq N, 1 \leq i \leq n\}$ with $N \gg n$, and $D$ is an $n \times k$ matrix with $k \geq n$. We say that a dependence matrix $D$ is lexicographically positive if every dependence vector of $D$ is lexicographically positive. The dependence matrix $D$ of a computable SURE can be transformed by a unimodular matrix to a lexicographically positive one [5]. Moreover, Wolf and Lam in [22] showed that an SURE (or a loop nest) with a lexicographically positive $D$ can be made fully permutable by a skewing transformation (a unimodular matrix). Therefore, there exists a unimodular transformation matrix $X$, such that $X_{n\times n} D_{n\times k} = F_{n\times k}$, $\forall f_{ih} \in F, f_{ih} \geq 0$. It follows that $D = X^{-1}F \equiv BF$. Since every element in $F$ is greater than or equal to zero, columns of $B$ constitute a positive integral coordinate basis of $D$. That is, every dependence vector $\vec{d}_h \in D$ can be constructed by the positive integral combination of the column vectors $\vec{b}_i$'s of $B$, or $\vec{d}_h = \sum_{i=1}^{n} f_{ih}\vec{b}_i$. For example, if $D = \left[\begin{smallmatrix} 0 & 1 & 1 \\ 1 & -2 & -1 \end{smallmatrix}\right]$, then we may have $X = \left[\begin{smallmatrix} 1 & 0 \\ 2 & 1 \end{smallmatrix}\right]$. Thus, $B = X^{-1} = \left[\begin{smallmatrix} 1 & 0 \\ -2 & 1 \end{smallmatrix}\right]$ and $F = \left[\begin{smallmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{smallmatrix}\right]$, e.g., $\vec{d}_3 = \left[\begin{smallmatrix} 1 \\ -1 \end{smallmatrix}\right] = f_{13}\vec{b}_1 + f_{23}\vec{b}_2 = 1 \left[\begin{smallmatrix} 1 \\ -2 \end{smallmatrix}\right] + 1 \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$. Consequently, the basis matrix $B_{n\times n}$ can be used to replace $D_{n\times k}$ and becomes the new dependence matrix of the SURE. In the following, $D$ in $\mathscr{A}_S = (J^n, D)$ represents the basis matrix of the SURE $\mathscr{A}_S$ if its original dependence matrix is not a square one.

Now, we focus on the two types SURE: the independently partitioned one and the one with identity dependence matrix. An SURE $\mathscr{A}_S = (J^n, D)$ can be independently partitioned to several independent sets. The number of independent sets is equal to the absolute value of the det$D$, $|\det D|$. In the following, we define the *independently partitioned* SURE (IP-SURE), in which all index vectors belong to the same independent set. It is reasonable that each IP-SURE can be treated apart, because they are independent.

Definition 2.2 (IP-SURE). An SURE $\mathscr{A}_P = (J_p^n, D)$ is said to be *independently partitioned* from an $\mathscr{A}_S = (J^n, D)$ if $J_p^n = J^n \wedge \{\vec{j} | \vec{j} = \vec{j}_0 + D\vec{\mu}\}$, where $\vec{j}_0 \equiv [j_1^0 j_2^0 \cdots j_n^0]^{\mathrm{T}}$ is an index vector in an independent set of $\mathscr{A}_S$ and $\vec{\mu}$ is an integral column vector.

It follows that if $J_p^n$ and $J_q^n$ are two partitions from $\mathscr{A}_S$, then either $J_p^n = J_q^n$ or $J_p^n \wedge J_q^n = \varnothing$. The first step of our method is to transform an IP-SURE to the SURE with an identity dependence matrix. Since the SURE of matrix multiplication has an identity dependence matrix, we have the following definition for the *matrix-multiplication-like SURE* (MM-SURE).

DEFINITION 2.3 (MM-SURE). An SURE $\mathscr{A}_M = (J_m^n, D)$ is said to be *matrix-multiplication-like* if $J_m^n \subset Z^n$ and $D = I$, where $I$ denotes the identity matrix.

In the following, $e_{ih}$ denotes the $ij$th element in $I$, and $\vec{e}_i$ is a column vector corresponding to the $i$th column of $I$. With the definitions of IP-SURE and MM-SURE, the first step of our method can be described as: transforming an IP-SURE $\mathscr{A}_P = (J_p^n, D)$ to an equivalent MM-SURE $\mathscr{A}_M = (J_m^n, I)$. Since $\mathscr{A}_M$ has an identity dependence matrix, we call this step *identity transformation.* The following theorem tells us how to perform the identity transformation, and shows its existence and that every element of $\vec{h} \in J_m^n$ is an integral number. The proof of integral index vector in $J_m^n$ is important, because the identity transformation includes inverting a nonunimodular matrix.

THEOREM 2.1 (Identity Transformation). *An IP-SURE $\mathscr{A}_P = (J_p^n, D)$, with $J_p^n = J^n \wedge \{\vec{j} | \vec{j} = \vec{j}_0 + D\vec{\mu}\}$, can be transformed to an MM-SURE $\mathscr{A}_M = (J_m^n, I)$, with $J_m^n = \{\vec{h} | \vec{h} = D^{-1}(\vec{j} - \vec{j}_0) + \vec{j}_0\}$, where $\vec{j}_0 \in J_p^n$ is mapped to $\vec{h}_0 (= \vec{j}_0) \in J_m^n$.*

*Proof. Existence of* $D^{-1}$. Since $D$ is an $n \times n$ full rank matrix, its inverse matrix $D^{-1}$ must exist.

*Integral index vector in* $J_m^n$. Let $D = [d_{ik}]_{n \times n}$, $D^{-1} = [d_{ik}^{-1}]_{n \times n}$.

Substituting $j_k = j_k^0 + \sum_{l=1}^n d_{kl}\mu_l$ to $h_i = \sum_{k=1}^n d_{ik}^{-1}(j_k - j_k^0) + j_i^0$, we have

$$h_i = \sum_{k=1}^n d_{ik}^{-1}\left(\sum_{l=1}^n d_{kl}\mu_l\right) + j_i^0 = \sum_{l=1}^n \mu_l\left(\sum_{k=1}^n d_{ik}^{-1}\, d_{kl}\right) + j_i^0.$$

From

$$\sum_{k=1}^n d_{ik}^{-1}\, d_{kl} = \begin{cases} 1 & \text{if } i = l \\ 0 & \text{otherwise} \end{cases},$$

we have $h_i = j_i^0 + \mu_i$. It follows that $\vec{h} = \vec{j}_0 + \vec{\mu} \in Z$. ∎

DEFINITION 2.4 (Affine Transformation) [18]. An *affine transformation* $\langle T, \vec{t}_0 \rangle$ for transforming an $n \times 1$ column vector $\vec{j}$ to $\vec{h}$ is defined as $\vec{h} = T\vec{j} + \vec{t}_0$, where $T$ and $\vec{t}_0$ are respectively an $n \times n$ matrix (linear part) and an $n \times 1$ column vector (translation part) of the affine transformation.

The operation $T_l\langle T, \vec{t}_0 \rangle$ is defined as $\langle T_l T, T_l \vec{t}_0 \rangle$, where $T_l$ is an $(m + 1) \times n$ matrix. Obviously, the identity transformation is an affine one and can be represented as $\langle D^{-1}, -D^{-1}\vec{j}_0 + \vec{j}_0 \rangle$. The purpose of the affine transformation's translation part is to let all index vectors still be integral after inverting the nonunimodular matrix $D$.

Theorem 2.1 states the fact that the time complexity of our method's first step is only a polynomial function of the number of dimensions $n$ of the IP-SURE, and is independent on the problem size parameter $N$. Now, we will turn our attention to the second step of our method: mapping an MM-SURE to a lower dimensional regular array. First, the definition of regular array is given as follows:

DEFINITION 2.5 (Regular Array). An $m$-dimensional *regular array* is defined as $\mathscr{A}_Y = (P^m, L)$, where $P^m = \{\vec{p} | \vec{p} = [p_1 p_2 \cdots p_m]^{\mathrm{T}}, 1 \le p_i \le N_i\}$ is a set of PEs and $L = \{\vec{l}_1, \vec{l}_2, \cdots, \vec{l}_\ell | \vec{l}_i, 1 \le i \le \ell, \text{ is an } m \times 1 \text{ column vector}\}$ a set of links connecting neighboring PEs.

The regular arrays we want to design have $N_i = c_i N$, where $c_i$ is a constant value and $1 \le i \le m$. One formal method of designing regular arrays for SUREs is by spacetime mapping. In the past, it is a linear transformation matrix, which only includes a schedule vector and a processor allocation matrix. Now, we extend the linear spacetime mapping to an affine one.

DEFINITION 2.6 (Affine Spacetime Mapping) [19]. An affine spacetime mapping $\langle T, \vec{t}_0 \rangle$ is a mapping that maps an SURE $\mathscr{A}_S = (J^n, D)$ to a regular array $\mathscr{A}_Y = (P^m, L)$ by the affine transformation $\langle T, \vec{t}_0 \rangle = \langle [\vec{\pi}\, S^{\mathrm{T}}]^{\mathrm{T}}, [\pi_0\, \vec{s}_0^{\mathrm{T}}]^{\mathrm{T}} \rangle$, where $\vec{\pi}$ is an $n \times 1$ column vector, $\pi_0$ an integer, $S^{\mathrm{T}}$ an $n \times m$ matrix, and $\vec{s}_0^{\mathrm{T}}$ an $1 \times m$ row vector. $\vec{\pi}^{\mathrm{T}}\vec{j} + \pi_0$ and $S\vec{h} + \vec{s}_0$ are respectively the time and space mapping of the index vector $\vec{j}$.

An affine spacetime mapping $\langle T, \vec{t}_0 \rangle$ can be reduced to a *linear* one $T$, if $\vec{t}_0$ is a zero vector. By an affine spacetime mapping $\langle T, \vec{t}_0 \rangle$, an index vector $\vec{h}$ is executed when time $t$ and at PE $[p_1 p_2 \cdots p_m]^{\mathrm{T}}$, where $[t\, p_1 p_2 \cdots p_m]^{\mathrm{T}} = T\vec{j} + \vec{t}_0$. In addition, a dependence vector $\vec{d}$ becomes a link $\vec{l}$, where $\vec{\pi}^{\mathrm{T}}\vec{d}$ and $S\vec{d}$ represent the link delay and link vector, respectively. A spacetime mapping is said to be *correct*, if it satisfies three conflict-free conditions: computation, dependence, and link. The following theorem states the necessary and sufficient conditions of a correct mapping.

THEOREM 2.2. *For an affine spacetime mapping $\{[\vec{\pi}\, S^{\mathrm{T}}]^{\mathrm{T}}, [\pi_0\, \vec{s}_0^{\mathrm{T}}]^{\mathrm{T}}\rangle$ to be correct on mapping an IP-SURE $\mathscr{A}_P = (J_p^n, D)$ to a regular array $\mathscr{A}_Y = (P^m, L)$, the necessary and sufficient conditions are as follows*:

- *Dependence conflict-free: if and only if $\forall \vec{d}_i \in D$, $\vec{\pi}^{\mathrm{T}}\vec{d}_i > 0$.*
- *Computation conflict-free: if and only if $\forall \vec{j}_1, \vec{j}_2 (\vec{j}_1 \ne \vec{j}_2) \in J_p^n$, if $S\vec{j}_1 = S\vec{j}_2$ then $\vec{\pi}^{\mathrm{T}}\vec{j}_1 \ne \vec{\pi}^{\mathrm{T}}\vec{j}_2$.*
- *Link conflict-free: if and only if $\forall \vec{j}_1, \vec{j}_2 (\vec{j}_1 \ne \vec{j}_2) \in J_p^n$, for each $\vec{d}_i \in D$, if $\vec{\pi}^{\mathrm{T}}(\vec{j}_1 - \vec{j}_2)S\vec{d}_i = S(\vec{j}_1 - \vec{j}_2)\vec{\pi}^{\mathrm{T}}\vec{d}_i$ then $\vec{j}_1 - \vec{j}_2 = k\vec{d}_i$ or $S\vec{d}_i = \vec{0}$.*

*Proof.*

- Dependence conflict-free: the precedence constraints imposed by the SURE are satisfied. That is, if there exists a $\vec{d}_i \in D$, such that $\vec{j}_2 = \vec{j}_1 + \vec{d}_i$, then $\vec{\pi}^{\mathrm{T}}\vec{j}_2 + \pi_0 > \vec{\pi}^{\mathrm{T}}\vec{j}_1 + \pi_0$. It follows that $\vec{\pi}^{\mathrm{T}}(\vec{j}_2 - \vec{j}_1) = \vec{\pi}^{\mathrm{T}}\vec{d}_i > 0$.

• Computation conflict-free: two different computations are not executed on the same PE at the same time. That is, if $S\vec{j}_1 + \vec{s}_0 = S\vec{j}_2 + \vec{s}_0$ then $\vec{\pi}^{\mathrm{T}}\vec{j}_1 + \pi_0 \neq \vec{\pi}^{\mathrm{T}}\vec{j}_2 + \pi_0$. It follows that if $S\vec{j}_1 = S\vec{j}_2$ then $\vec{\pi}^{\mathrm{T}}\vec{j}_1 \neq \vec{\pi}^{\mathrm{T}}\vec{j}_2$.

• Link conflict-free: every dependence vector of the SURE is mapped to one and only one interconnection link of the array. From the formula derived by Lee and Kedem in [11], for a linear spacetime mapping, the condition of link conflict-free is: if $(S\,\Delta\vec{j})(\vec{\pi}^{\mathrm{T}}\vec{d}_i) = (\vec{\pi}^{\mathrm{T}}\,\Delta\vec{j})(S\vec{d}_i)$ then $\Delta\vec{j} = k\vec{d}_i$, $k \neq 0$, where $\Delta\vec{j} = \vec{j}_1 - \vec{j}_2$. In the if part, the four items $S\,\Delta\vec{j}$, $\vec{\pi}^{\mathrm{T}}\vec{d}_i$, $\vec{\pi}^{\mathrm{T}}\,\Delta\vec{j}$, and $S\vec{d}_i$ represent respectively (after linear spacetime mapping) two index vectors' processor difference vector, a link delay, two index vectors' time delay, and a link vector. Now, for an affine spacetime mapping, these four items are still $S\,\Delta\vec{j}$, $\vec{\pi}^{\mathrm{T}}\vec{d}_i$, $\vec{\pi}^{\mathrm{T}}\,\Delta\vec{j}$, and $S\vec{d}_i$, respectively, since the effect of an affine transformation's translation part is eliminated from two vectors' difference. ■

To design regular arrays by finding a candidate of correct spacetime mapping, it is necessary to check all conflict-free conditions (dependence, computation, and link). These examinations will take a lot of time when the problem size parameter $N$ becomes very large. To avoid doing these examining procedures for shortening design time, our solution is first to find an affine transformation $\langle T, \vec{t}_0\rangle$, to transform the given IP-SURE to an MM-SURE. Then, we will propose (in Section 3) a conflict-free (linear) spacetime mapping matrix in a fixed form, say $T_l$, to map the MM-SURE to a lower dimensional regular array. The following theorem not only shows that this method is valid but also states how to obtain a transformation matrix from $\langle T, \vec{t}_0\rangle$ and $T_l$ for mapping the given IP-SURE to a regular array.

THEOREM 2.3. *Designing a regular array $\mathscr{A}_Y = (P^m, L)$ for an IP-SURE $\mathscr{A}_P = (J_p^n, D)$ can be done by the affine spacetime mapping $T_l\langle T, \vec{t}_0\rangle$, where $\langle T, \vec{t}_0\rangle$ is the identity transformation for transforming $\mathscr{A}_P$ to an MM-SURE $\mathscr{A}_M = (J_m^n, I)$, and $T_l = [\vec{\phi}\ R^{\mathrm{T}}]^{\mathrm{T}}$ is an $(m+1) \times n$ correct linear spacetime mapping for mapping $\mathscr{A}_M$ to $\mathscr{A}_Y$, where $\vec{\phi}^{\mathrm{T}}$ and $R$ are the $1 \times n$ schedule vector and the $m \times n$ processor assignment matrix*, *respectively.*

*Proof.* From Theorem 2.1 we know that an IP-SURE $\mathscr{A}_P = (J_p^n, D)$ can always be transformed to an equivalent MM-SURE $\mathscr{A}_M = (J_m^n, I)$ by the affine transformation $\langle T, \vec{t}_0\rangle = \langle D^{-1}, -D^{-1}\vec{j}_0 + \vec{j}_0\rangle$. Now we want to prove that if $T_l = [\vec{\phi}\ R^{\mathrm{T}}]^{\mathrm{T}}$ is correct for mapping $\mathscr{A}_M$ to $\mathscr{A}_Y$, then $T_l\langle T, \vec{t}_0\rangle \equiv \langle[\vec{\pi}\ S^{\mathrm{T}}]^{\mathrm{T}}, [\pi_0\ \vec{s}_0^{\mathrm{T}}]^{\mathrm{T}}\rangle$ is also correct for mapping $\mathscr{A}_P$ to $\mathscr{A}_Y$.

• Dependence conflict-free: Since $T_l$ is conflict-free, $\forall \vec{e}_i \in I$, $\vec{\phi}^{\mathrm{T}}\vec{e}_i > 0$. Thus, from $\vec{e}_i = D^{-1}\,\vec{d}_i$, we have $\vec{\phi}^{\mathrm{T}}D^{-1}\,\vec{d}_i > 0$ or $\vec{\pi}^{\mathrm{T}}\vec{d}_i > 0$.

• Computation conflict-free: Since $T_l$ is conflict-free, $\forall \vec{h}_1, \vec{h}_2 \in J_m^n$, if $R\,\Delta\vec{h} = \vec{0}$ then $\vec{\phi}^{\mathrm{T}}\,\Delta\vec{h} \neq 0$, where $\Delta\vec{h} = \vec{h}_1 - \vec{h}_2$. Thus, from $\Delta\vec{h} = D^{-1}\,\Delta\vec{j}$, $\Delta\vec{j} = \vec{j}_1 - \vec{j}_2$, we have if $RD^{-1}\,\Delta\vec{j} = 0$ then $\vec{\phi}^{\mathrm{T}}D^{-1}\,\Delta\vec{j} \neq 0$, or if $S\,\Delta\vec{j} = 0$ then $\vec{\pi}^{\mathrm{T}}\,\Delta\vec{j} \neq 0$.

• Link conflict-free: Since $T_l$ is conflict-free, $\forall \vec{h}_1, \vec{h}_2 \in J_m^n$, for each $\vec{e}_i \in I$, if $\vec{\phi}^{\mathrm{T}}\,\Delta\vec{j}R\vec{e}_i = R\,\Delta\vec{h}\vec{\phi}^{\mathrm{T}}\vec{e}_i$, then $\Delta\vec{h} = k\vec{e}_i$ or $R\vec{e}_i = \vec{0}$, where $\Delta\vec{h} = \vec{h}_1 - \vec{h}_2$. Thus, if $\vec{\phi}^{\mathrm{T}}D^{-1}\,\Delta\vec{j}RD^{-1}\vec{d}_i = RD^{-1}\,\Delta\vec{j}\vec{\phi}^{\mathrm{T}}D^{-1}\,\vec{d}_i$ then $D^{-1}\,\Delta\vec{j} = kD^{-1}\,\vec{d}_i$ or $RD^{-1}\,\vec{d}_i = \vec{0}$, where $\Delta\vec{j} = \vec{j}_1 - \vec{j}_2$, or if $\vec{\pi}^{\mathrm{T}}\,\Delta\vec{h}S\,\vec{d}_i = S\,\Delta\vec{j}\vec{\pi}^{\mathrm{T}}\,\vec{d}_i$ then $\Delta\vec{j} = k\vec{d}_i$ or $S\vec{d}_i = \vec{0}$.

From the results of the above and Theorem 2.2, we know that $T_l\langle T, \vec{t}_0\rangle$ is a correct affine spacetime mapping for mapping $\mathscr{A}_P$ to $\mathscr{A}_Y$, if $T_l$ is a correct linear one for mapping $\mathscr{A}_M$ to $\mathscr{A}_Y$. ■

## 3. DESIGN OF REGULAR ARRAYS

In this section, we first give an affine spacetime mapping in a fixed form to design a regular array for an IP-SURE and prove the correctness of the mapping. Then, two examples will be given for illustration. Finally, we will show the regular array designed by our method is asymptotically optimal in space and time.

THEOREM 3.1. *Designing a regular array $\mathscr{A}_Y = (P^m, L)$ for an IP-SURE $\mathscr{A}_P = (J_p^n, D)$ can be done by the affine spacetime mapping $T_l\langle T, \vec{t}_0\rangle$, where $\langle T, \vec{t}_0\rangle = \langle D^{-1}, -D^{-1}\vec{h}_0 + \vec{h}_0\rangle$ is the affine transformation for transforming $\mathscr{A}_P$ to an MM-SURE $\mathscr{A}_M = (J_m^n, I)$, and $T_l = [\vec{\phi}\ R^{\mathrm{T}}]^{\mathrm{T}}$ $(R = [\vec{r}_1\,\vec{r}_2 \cdots \vec{r}_n])$ is the $(m+1) \times n$ linear spacetime mapping for mapping $\mathscr{A}_M$ to $\mathscr{A}_Y$, in which*

$$\phi_i = \begin{cases} H^{n-m-i} & 1 \le i \le n-m \\ 1 & \textit{otherwise} \end{cases},$$

$$\vec{r}_i = \begin{cases} \vec{0} & 1 \le i \le n-m \\ \vec{e}_{i-n+m} & \textit{otherwise} \end{cases},$$

*and $H = \max_{1 \le i \le n} \sum_{k=1}^{n} |d_{ik}^{-1}|N$ $(N \gg n)$.*

*Proof.* From Theorem 2.3, we only need to prove that $T_l$ is a correct spacetime mapping. Recall that, $\vec{h}$ denotes an index vector and $\vec{e}_i$ is a dependence vector in $\mathscr{A}_M$.

• Dependence conflict-free: $\vec{\phi}^{\mathrm{T}}\vec{e}_i > 0$, $\forall 1 \le i \le n$.

• Computation conflict-free: If $R\,\Delta\vec{h} = \vec{0}$, then $\Delta h_i = 0$, $n-m+1 \le i \le n$. By contradiction, assume $\vec{\phi}^{\mathrm{T}}\,\Delta\vec{h} = 0$. Substituting $\Delta h_i = 0$, $n-m+1 \le i \le n$ to $\vec{\phi}^{\mathrm{T}}\,\Delta\vec{h} = 0$, we have $\sum_{i=1}^{n-m} \phi_i\,\Delta h_i = 0$. It follows that $H^{n-m-1}\,\Delta h_1 + H^{n-m-2}\,\Delta h_2 + \cdots + \Delta h_{n-m} = 0$. Since $\Delta h_i = \sum_{k=1}^{n} d_{ik}^{-1}\,\Delta j_k$, we have $|\Delta h_i| < H$, thus the above equation holds if and only if $\Delta h_i = 0$, $1 \le i \le n-m$. However, we already have $\Delta h_i = 0$, $n-m+1 \le i \le n$. Thus, $\Delta h_i = 0$, $1 \le i \le n$. It is contradictory to that there should exist at least one $\Delta h_i \neq 0$, $1 \le i \le n$.

• Link conflict-free: For $\vec{e}_k$, $1 \le k \le n-m$, no link is generated. For $\vec{e}_k$, $n-m+1 \le k \le n$, from $\vec{\phi}^{\mathrm{T}}$

$\Delta\vec{h}R\vec{e}_k = R\,\Delta\vec{h}\vec{\phi}^{\mathrm{T}}\vec{e}_k$, we can deduce $\Delta h_i = 0, n-m+1 \le i \le n$, except $i = k$, as well as $H^{n-m-1}\,\Delta h_1 + H^{n-m-2}\,\Delta h_2 + \cdots + \Delta h_{n-m} = 0$. Since $\Delta h_i = \sum_{k=1}^{n} d_{ik}^{-1}\,\Delta j_k$, we have $|\Delta h_i| \le H$, thus $\Delta h_i = 0,\ 1 \le i \le n-m$. It follows that $\Delta h_i = 0,\ 1 \le i \le n$, except $i = k$. Thus, if $\vec{\phi}^{\mathrm{T}}\,\Delta\vec{h}R\vec{e}_k = R\,\Delta\vec{h}\vec{\phi}^{\mathrm{T}}\vec{e}_k$, then $\Delta\vec{h} = p\vec{e}_k$, where $p$ is an integer. ■

For an affine spacetime mapping $\langle[\vec{\pi}\ S^{\mathrm{T}}]^{\mathrm{T}}, [\pi_0\ \vec{s}_0^{\mathrm{T}}]^{\mathrm{T}}\rangle = \langle T_l T, T_l\vec{t}_0\rangle = \langle T_l D^{-1}, -T_l D^{-1}\vec{j}_{\mathrm{o}} + T_l\vec{j}_0\rangle$, a dependence vector $\vec{d}_i$ is mapped to a link $\vec{l}_i$ with the delay $\vec{\pi}^{\mathrm{T}}\,\vec{d}_i$ and the vector $S\vec{d}_i$. When $T_l = [\vec{\phi}\ R^{\mathrm{T}}]^{\mathrm{T}}$, it follows that $\vec{\pi}^{\mathrm{T}}\,\vec{d}_i = \vec{\phi}^{\mathrm{T}}D^{-1}\,\vec{d}_i = \vec{\phi}^{\mathrm{T}}\vec{e}_i$ and $S\vec{d}_i = RD^{-1}\,\vec{d}_i = R\vec{e}_i$. If we design regular array by applying Theorem 3.1, then for every link ($R\vec{e}_i \ne 0$) we have $R\vec{e}_i = \vec{e}_{i-n+m}$, where $n-m+1 \le i \le n$. That is, the designed regular array must be locally connected that satisfies the definition of regular array. Besides, the delay ($\vec{\phi}^{\mathrm{T}}\vec{e}_i, n-m+1 \le i \le n$) for every link is equal to 1. Meanwhile, the buffer size required in each PE is $H^{n-m-1} = \mathcal{O}(N^{n-m-1})$. Note that, whether $H$ is an integer will determine if the execution time of each index vector is an integer. Thus, if $H$ is a fraction number, say $H_1N/H_2$, then we can either expand the computation domain so that $H_2|N$, or choose a minimal integral $H$ such that $H \ge \max_{1\le i\le n}\sum_{k=1}^{n}|d_{ik}^{-1}|N$. There are two examples to illustrate our method of designing lower dimensional regular arrays for IP-SUREs. The first example is to map a 4D IP-SURE to a 2D mesh array and an 1-D linear one. The second example is to design linear arrays for matrix multiplication and transitive closure.

Example 3.1. In this example, we will design a 2D mesh array and an 1-D linear one for an IP-SURE $\mathcal{A}_P = (J_p^4, D)$, with

$$D = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 2 & 0 \\ -1 & 0 & 1 & 2 \end{bmatrix}$$

and $J_p^4 = J^4 \wedge \{\vec{j}|\vec{j} = \vec{j}_0 + D\vec{\mu}\}$, where $\vec{j}_0 = [1\ 1\ 1\ 1]^{\mathrm{T}}$. The dependence graph of $\mathcal{A}_P$ is shown in Fig. 1a. First, we have

$$D^{-1} = \begin{bmatrix} 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}$$

According to Theorem 3.1 with $n = 4$ and $m = 2$, we have $H = 3N/2$ and

$$T_l = \begin{bmatrix} \frac{3N}{2} & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

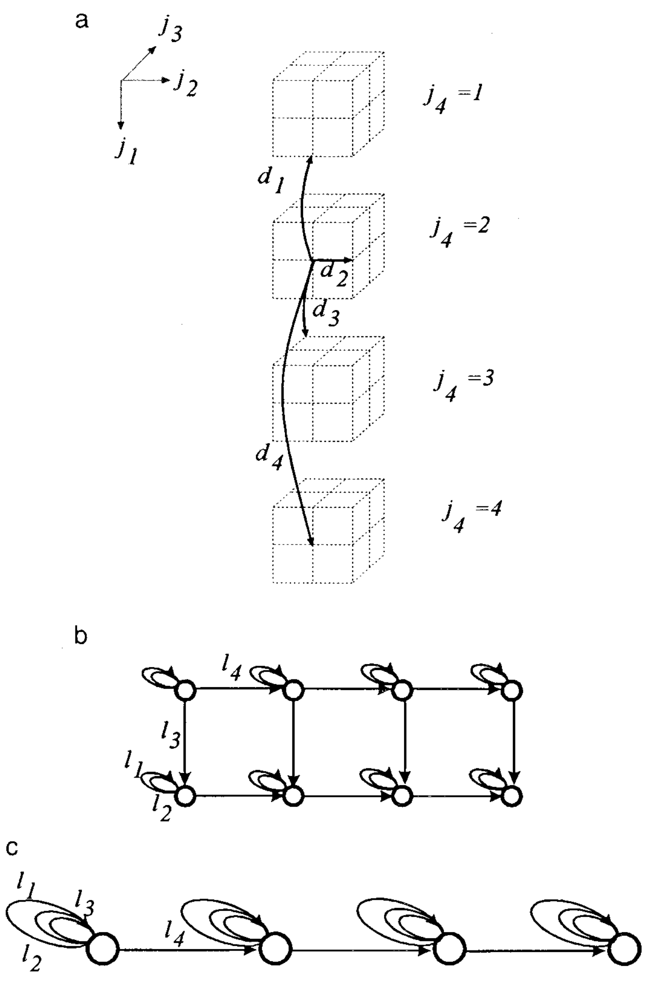


**FIG. 1.** (a) The dependence graph of Example 3.1. (b) The designed 2D mesh array ($N = 4$) for Example 3.1. (c) The designed 1D linear array ($N = 4$) for Example 3.1.

Hence, the affine spacetime mapping is

$$\langle T_l D^{-1},\ -T_l D^{-1}\vec{j}_0 + T_l\vec{j}_0\rangle$$

$$= \left\langle \begin{bmatrix} \frac{3N+1}{2} & 1 & \frac{3N+4}{4} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}, \begin{bmatrix} -\frac{3N}{4} \\ \frac{1}{2} \\ 0 \end{bmatrix} \right\rangle.$$

Thus, the spacetime mapping for an index vector $\vec{j} \in J_p^4$ is

$$[t\quad p_1\quad p_2]^{\mathrm{T}}$$

$$= \left[\frac{(6N+2)j_1 + 4j_2 + (3N+4)j_3 + 2j_4 - 3N}{4}\ \frac{j_3+1}{2}\ \frac{j_1+j_4}{2}\right]^{\mathrm{T}}.$$

The 2D mesh array is shown in Fig. 1b for $N = 4$. If the desired array is a linear one, then

$$T_l = \begin{bmatrix} \dfrac{9N^2}{4} & \dfrac{3N}{2} & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

It follows that the affine spacetime mapping is

$$\begin{aligned} &\langle T_l D^{-1}, -T_l D^{-1}\vec{j}_0 + T_l\vec{j}_0\rangle \\ &= \left\langle \begin{bmatrix} \dfrac{9N^2+2}{4} & \dfrac{3N}{2} & \dfrac{9N^2+6N+4}{8} & \dfrac{1}{2} \\ \dfrac{1}{2} & 0 & 0 & \dfrac{1}{2} \end{bmatrix}, \begin{bmatrix} -\dfrac{9N^2-6N+4}{8} \\ 0 \end{bmatrix} \right\rangle. \end{aligned}$$

Thus, the spacetime mapping for an index vector $\vec{h} \in J_p^4$ is

$$[t \quad p_1]^{\mathrm{T}} = \begin{bmatrix} \dfrac{(18N^2+4)j_1 + 12Nj_2 + (9N^2+6N+4)j_3 + 4j_4 - 9N^2 - 6N + 4}{8} & \dfrac{j_1+j_4}{2} \end{bmatrix}^{\mathrm{T}}.$$

The 1D linear array is shown in Fig. 1c.

EXAMPLE 3.2. Unidirectional linear arrays for the problem of matrix multiplication and transitive closure are designed as follows:

- Matrix multiplication: The dependence graph of matrix multiplication is shown in Fig. 2a. Since it has an identity dependence matrix, we have $T = I$ and $\vec{t}_0 = [0\,0\cdots 0]^{\mathrm{T}}$. According to Theorem 3.1 with $n = 3$ and $m = 1$, we have $T_l = [\begin{smallmatrix} N & 1 & 1 \\ 0 & 0 & 1 \end{smallmatrix}]$. It follows that the affine spacetime mapping is $\langle T_l T, T_l\vec{t}_0\rangle = \langle[\begin{smallmatrix} N & 1 & 1 \\ 0 & 0 & 1 \end{smallmatrix}], \vec{0}\rangle$, which is reduced to a linear one. We draw the complete spacetime diagram in Fig. 2c, where $[\begin{smallmatrix}\text{Time}\\\text{Space}\end{smallmatrix}] = [\begin{smallmatrix} Ni+j+k \\ k \end{smallmatrix}]$ and $1 \le i, j, k \le N = 4$. From this diagram, it is easy to see that in the case of matrix multiplication, the designed linear array is time–optimal, because every PE begins execution as soon as possible and does not stop until all computations are done. The designed linear array is shown in Fig. 2b for $N = 4$. The execution time is $T_e = (N + 1 + 1)(N - 1) + 1 = N^2 + N - 1$. The number of PEs used is $\|P\| = (N - 1) + 1 = N$.
- Transitive closure: The dependence graph of transitive closure is shown in Fig. 3a. Although its dependence matrix

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

is not a square one, $D$ can be made fully permutable by skewing. That is, $X_{3\times3}D_{3\times5} = F_{3\times5}$, where $X$ is the skew transformation matrix and $\forall f_{ih} \in F, f_{ih} \ge 0$. Here

$$X = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad\text{and}\quad F = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

Thus, we have $D = X^{-1}F \equiv BF$, where

$$B = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}.$$

The basis matrix $B$ is used to replace $D$ to construct the new dependence matrix of transitive closure. From

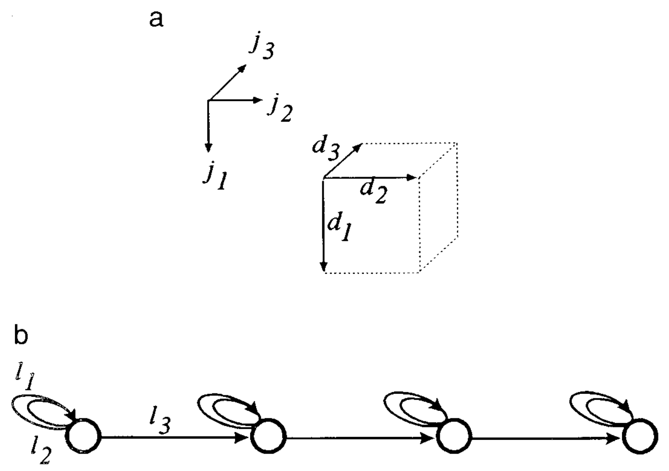
$$T = B^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix},$$

$\vec{t}_0 = [0\,0\cdots 0]^{\mathrm{T}}$, and $T_l = [\begin{smallmatrix} 2N & 1 & 1 \\ 0 & 0 & 1 \end{smallmatrix}]$, we have $\langle T_l T, T_l\vec{t}_0\rangle = \langle[\begin{smallmatrix} 2N & 1 & 2N+2 \\ 0 & 0 & 1 \end{smallmatrix}], \vec{0}\rangle$. The designed linear array is shown in Fig. 3b for $N = 4$. The execution time is $T_e = (2N + 1 + 2N + 2)(N - 1) + 1 = 4N^2 - N - 2$. The number of PEs used is $\|P\| = (N - 1) + 1 = N$. Note that, there are three basis vectors out of five dependence vectors. $\vec{d}_1(= \vec{b}_1)$, $\vec{d}_2(=\vec{b}_2)$ and $\vec{d}_3(=\vec{b}_3)$ are respectively mapped to the links $\vec{l}_1, \vec{l}_2$, and $\vec{l}_3$, which have delays $2N$, 1, and 1, and vectors $\vec{0}, \vec{0}$, and $\vec{1}$, respectively. On the other hand, the dependence vector $\vec{d}_4(\vec{d}_5)$ is a positive combination of $\vec{b}_2$ and $\vec{b}_3(\vec{b}_1$ and $\vec{b}_3)$. That is, we have the links $\vec{l}_4(= \vec{l}_2 + \vec{l}_3)$ for $\vec{d}_4$ and $\vec{l}_5(= \vec{l}_1 + \vec{l}_3)$ for $\vec{d}_5$. These two links are shown in thick and thin dash lines of Fig. 3b, respectively. The essence of $\vec{l}_i + \vec{l}_h$ for a dependence vector is: let $\vec{l}_i(\vec{l}_h)$ have link delay $\vec{\pi}^{\mathrm{T}}\vec{b}_i(\vec{\pi}^{\mathrm{T}}\vec{b}_h)$ and link vector $S\vec{b}_i(S\vec{b}_h)$. First, assuming that a computation result is generated by a processor $\vec{p}_k$ when $t_1$. Then, it is propagated to the processor $\vec{p}_k + S\vec{b}_i$ when $t_1 + \vec{\pi}^{\mathrm{T}}\vec{b}_i$ by the link $\vec{l}_i$. Finally, it reaches to the processor $\vec{p}_k + S\vec{b}_i + S\vec{b}_h$ when $t_1 + \vec{\pi}^{\mathrm{T}}\vec{b}_i + \vec{\pi}^{\mathrm{T}}\vec{b}_h$ by the link $\vec{l}_h$.

The following three theorems will show that the regular arrays designed by Theorem 3.1 are asymptotically optimal in space and time.

THEOREM 3.2. *The execution time and the number of PEs of the mD regular array, designed by Theorem* 3.1 *for an nD IP-SURE, are $\mathcal{O}(N^{n-m})$ and $\mathcal{O}(N^m)$, respectively.*

*Proof.* From Theorem 3.1, we know that the execution time $T_e = (\vec{\phi}^{\mathrm{T}}D^{-1}\vec{j})_{\max} - (\vec{\phi}^{\mathrm{T}}D^{-1}\vec{j})_{\min} + 1$. Let $\vec{\pi}^{\mathrm{T}} = [\pi_1\ \pi_2 \cdots \pi_n] = \vec{\phi}^{\mathrm{T}}D^{-1}$ and $H = \max_{1\le i\le n}\sum_{k=1}^{n}|d_{ik}^{-1}|N \equiv hN$. Thus,



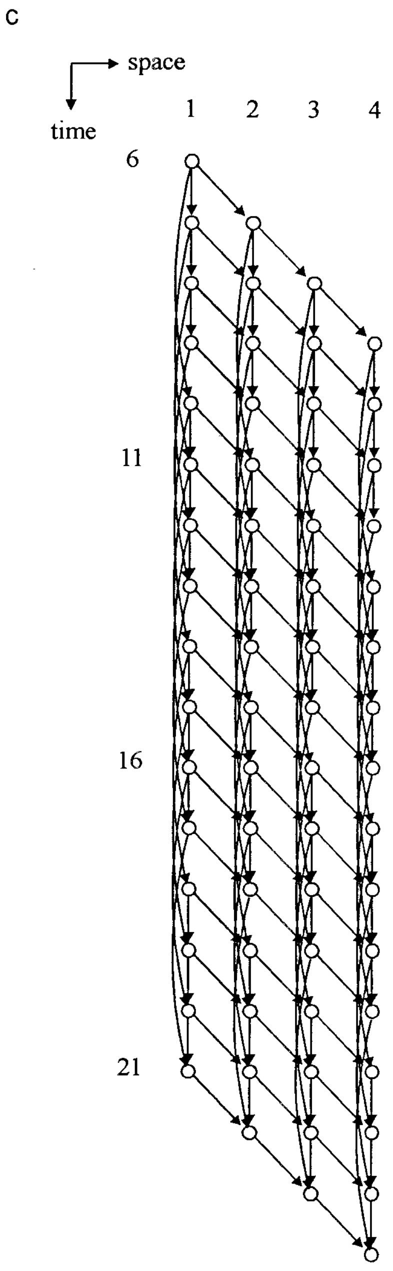


**FIG. 2.** (a) The dependence graph of matrix multiplication. (b) The spacetime diagram ($N = 4$). (c) The designed 1D linear array ($N = 4$) for matrix multiplication.

$\pi_k = \sum_{i=1}^{n} \phi_i d_{ik}^{-1} = d_{1k}^{-1} H^{n-m-1} + o(H^{n-m-1})$. Thus, the execution time of the $m$D regular array designed by our method is $T_e = (\sum_{k=1}^{n} |\pi_k|)(N - 1) + 1 = (\sum_{k=1}^{n} (|d_{1k}^{-1}| H^{n-m-1} + o(H^{n-m-1})))(N - 1) + 1 \equiv cN^{n-m} + o(N^{n-m}) = \mathcal{O}(N^{n-m})$, where $c$ is a positive constant value.

On the other hand, let $R = [\vec{r}_1 \vec{r}_2 \cdots \vec{r}_m]^{\mathrm{T}}$. We know that the number of PEs used $\|P\| = \Pi_{k=1}^{m} N_k$, where $N_k = (\vec{r}_k^{\mathrm{T}} D^{-1} \vec{j})_{\max} - (\vec{r}_k^{\mathrm{T}} D^{-1} \vec{j})_{\min} + 1$. It can be derived that $N_k = (\sum_{i=1}^{n} |d_{n-m+k,i}^{-1}|)\ (N - 1) + 1$. Thus, we have $\|P\| = \Pi_{k=1}^{m} |N_k| = cN^m = \mathcal{O}(N^m)$. ■

THEOREM 3.3. *Executing an IP-SURE $\mathcal{A}_P = (J_p^n, D)$ on an mD regular array, the lower bound of the execution time is* $\Omega(N^{n-m})$.

*Proof.* For a recurrence equation $v_h(\vec{h}_i + \vec{d}_i) = f_h(\cdots, v_i(\vec{h}_i), \cdots)$ in an IP-SURE, if $\vec{h}_i \notin J_p^n$, then the value of $v_i(\vec{h}_i)$ should be set by the initial value of the IP-SURE. For an $n$D IP-SURE, there are at least $c_1 N^{n-1}$ initial values for the variable $v_i$. Executing the IP-SURE on an $m$D regular array, there are at most $c_2 N^{m-1}$ input ports for each variable, because the input ports of the $m$D regular array are confined on the boundary PEs. Since the regular array should satisfy the link conflict-free condition, the time is at least $c_1 N^{n-1}/c_2 N^{m-1} \equiv N^{n-m}/c$ for inputting all $c_1 N^{n-1}$ initial values of a variable. Thus, the lower bound of the execution time is $\Omega(N^{n-m})$. ■

THEOREM 3.4. *By Theorem* 3.1 *an asymptotically optimal mD regular array can be designed in polynomial time for any IP-SURE.*

*Proof. Optimality.* The asymptotical optimality of time can be proved directly from Theorem 3.2 and Theorem 3.3. On the other hand, the proof of the asymptotical optimality of space is direct: it is obvious that the number of PEs supported by an $m$D regular array is at least $\Omega(N^m)$, because $N_i = c_i N$, where $1 \le i \le m$. From Theorem 3.2, the number of PEs used by Theorem 3.1 is $\mathcal{O}(N^m)$. Hence the space optimality is proved.

*Polynomial time.* The time complexity is polynomial for calculating the affine spacetime mapping $\langle T_l D^{-1}, -T_l D^{-1}\vec{j}_0 + T_l \vec{j}_0 \rangle$ proposed in Theorem 3.1, because we need only to calculate the inverse of $D$ and to compute products of martrices. It is easy to see that the design time only depends on the number of dimensions of the IP-SURE.

Hence the theorem is proved. ■

## 4. CONCLUSION

In this paper, we have proposed a polynomial time method to design lower-dimensional regular arrays for

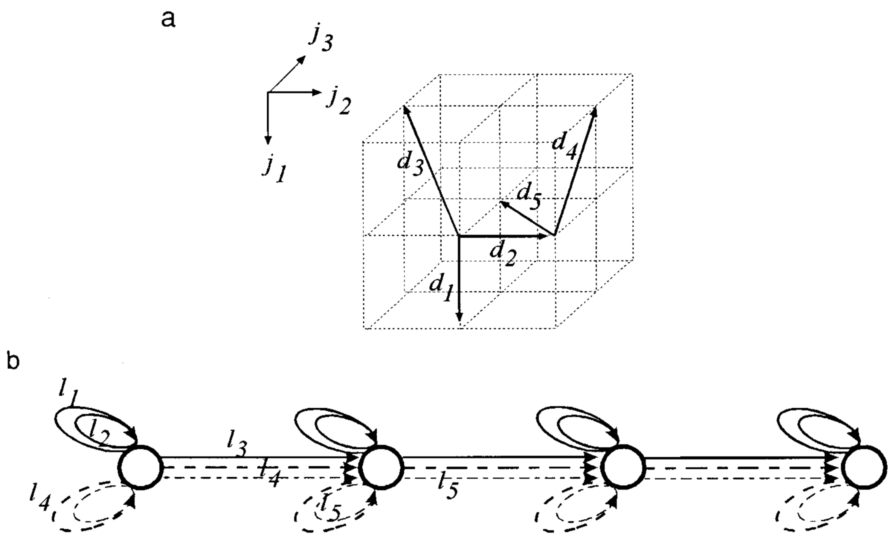


**FIG. 3.** (a) The dependence graph of transitive closure. (b) The designed 1D linear array ($N = 4$) for transitive closure.

SUREs. Previous researchers solved this problem by checking the conditions of dependence, computation, and/or link conflict-free for a candidate of spacetime mapping. However, this check step is time-consuming when the problem size of an SURE is very large. We avoid the previous time-consuming step by proposing a new two-step method. In the first step, an independently partitioned SURE (IP-SURE) $\mathcal{A}_P = (J_p^n, D)$ is transformed to a matrix-multiplication-like SURE (MM-SURE) $\mathcal{A}_M = (J_m^n, I)$. The transformation is an affine one $\langle T, \vec{t}_0\rangle$ that is equal to $\langle D^{-1}, -D^{-1}\vec{j}_0 + \vec{j}_0\rangle$, where $\vec{j}_0$ is an index vector in $J_p^n$. The purpose of the affine transformation is to let all index vectors in $J_m^n$ be integral ones, because $D$ is in general a nonunimodular matrix, which results in that elements in $D^{-1}$ are not integral. In the second step, the MM-SURE is mapped to an $m$D regular array $\mathcal{A}_Y = (P^m, L)$ by a linear spacetime mapping matrix $T_l$, which is in a fixed form. By combining $\langle T, \vec{t}_0\rangle$ and $T_l$ obtained in the above two steps, the affine spacetime mapping $\langle T_lT, T_l\vec{t}_0\rangle$ can be used to design an $m$D regular array for the $n$D IP-SURE, where $n \geq m + 1$. The contribution of this paper is that we propose a polynomial time method to design lower-dimensional regular arrays for SUREs; the time complexity of the method is independent on the problem size and dependent only on the number of dimensions of the regular algorithm. Meanwhile, the designed regular array is asymptotically optimal in space and time.

## REFERENCES

1. Kung, H. T., and Leiserson, C. E. Systolic arrays (for VLSI). *Proc. Sparse Matrix Symposium*, SIAM, Philadelphia, 1978, pp. 256–282.
2. Kung, H. T. Why systolic architectures? *Computer* **15,** 1 (1982), 37–46.
3. Kung, H. T. The Warp computer: architecture, implementation, and performance. *IEEE Trans. Comput.* **C-36,** 12 (Dec. 1987), 1523–1538.
4. Kung, H. T. *et al.* iWarp: An integrated solution to high-speed parallel computing. *Proc. of Supercomputing*, 1988, pp. 330–339.
5. Rao, S. K. Regular iterative algorithms and their implementations on processor arrays. Ph.D. thesis, Standford University, 1985.
6. Van Dongen, V., and Quinton, P. Uniformization of linear recurrence equations: A step towards the automatic synthesis of systolic arrays. *Proc. Int. Conf. Systolic Array,* 1988, pp. 473–482.
7. Quinton, P. Automatic synthesis of systolic arrays from uniform recurrent equations. *Proc. Int. Symp. Computer Architecture*, 1984, pp. 208–214.
8. Peir, J. K., and Cytron, R. Minimum distance: a method for partitioning recurrences for multiprocessors. *IEEE Trans. Comput.* **38,** 8 (Aug. 1989), 1203–1211.
9. Shang, W., and Fortes, J. A. B. Independent partitioning of algorithms with uniform dependencies. *IEEE Trans. Comput.* **41,** 2 (Feb. 1992), 190–206.
10. D'Hollander, E. H. Partitioning and labeling of loops by unimodular transformations. *IEEE Trans. Parallel Distrib. Systems* **3,** 4 (July 1992), 465–476.
11. Lee, P. Z., and Kedem, Z. M. Mapping nested loop algorithms into multidimensional systolic arrays. *IEEE Trans. Parallel Distrib. Systems* **1,** 1 (Jan. 1990), 64–76.
12. Shang, W., and Fortes, J. A. B. On time mapping of uniform dependence algorithms into lower dimensional processor arrays. *IEEE Trans. Parallel Distrib. Systems* **3,** 3 (May 1992), 350–363.
13. Moldovan, D. I., and Fortes, J. A. B. Partitioning and mapping algorithms into fixed size systolic arrays. *IEEE Trans. Comput.* **C-35,** 1 (Jan. 1986), 1–12.
14. Shang, W., and Fortes, J. A. B. Time optimal linear schedules for algorithms with uniform dependencies. *IEEE Trans. Comput.* **40,** 6 (June 1991), 723–742.
15. Tsay, J. C., and Chang, P. Y. Design of space-optimal regular arrays for algorithms with linear schedules. *IEEE Trans. Comput.* **44,** 5 (May 1995), 683–694.
16. Chang, P. Y., and Tsay, J. C. A family of efficient regular arrays for algebraic path problem. *IEEE Trans. Comput.* **43,** 7 (July 1994), 769–777.
17. Ganapathy, K. N., and Wah, B. W. Synthesizing optimal lower dimensional processor arrays. *Proc. Int. Conf. Parallel Processing,* 1992, pp. III.96–III.103.
18. Rockafellar, R. T. *Convex Analysis,* 2nd ed., Princeton Univ. Press, Princeton, NJ, 1972.
19. Van Dongen, V. Quasi-regular arrays: Definition and design methodology. *Proc. Int. Conf. on Systolic Arrays*, 1989, pp. 126–135.

20. Chang, P. Y., and Tsay, J. C. An approach to designing modular extensible linear arrays for regular algorithms. *IEEE Trans. Comput.*, to appear.
21. Yaacoby, Y., and Cappello, P. R. Scheduling a system of affine recurrence equations onto a systolic array. Technical Report TRCS87-19, Dep. of Electrical and Computer Engineering, UCSB, 1988.
22. Wolf, M. E., and Lam, M. S. A loop transformation theory and an algorithm to maximize parallelism. *IEEE Trans. Parallel Distrib. Systems* **2,** 4 (Oct. 1991), 452–471.

JONG-CHUANG TSAY was born in Taipei, Republic of China in 1943. He received the M.S. and Ph.D. degrees in computer science from National Chiao-Tung University in 1968 and 1975, respectively. Since 1968, he has been with the faculty of the Department of Computer Engineering, National Chiao-Tung University. Currently, he is a professor in the Department of Computer Science and Information Engineering. His current research interests include systolic algorithm design, parallel computations, and computer-aided typesetting.

PEN-YUANG CHANG was born in Kaohsiung, Republic of China in 1962. He received the M.S. and Ph.D. degrees in computer science from National Chiao-Tung University in 1986 and 1995, respectively. Since 1987, he has been an associate researcher in the Telecommunication Laboratories, Ministry of Communications. His research interests include systolic arrays and computer networks.

Received February 7, 1994; revised January 29, 1995; accepted July 30, 1995