The Design of Optimal Systolic Arrays

GUO-JIE LI, STUDENT MEMBER, IEEE, AND BENJAMIN W. WAH, MEMBER, IEEE

Abstract — **Conventional design of systolic arrays is based on the mapping of an algorithm onto an interconnection of processing elements in a VLSI chip. This mapping is done in an ad hoc manner, and the resulting configuration usually represents a feasible but suboptimal design. In this paper, systolic arrays are characterized by three classes of parameters: the velocities of data flows, the spatial distributions of data, and the periods of computation. By relating these parameters in constraint equations that govern the correctness of the design, the design is formulated into an optimization problem. The size of the search space is a polynomial of the problem size, and a methodology to systematically search and reduce this space and to obtain the optimal design is proposed. Some examples of applying the method, including matrix multiplication, finite impulse response filtering, deconvolution, and triangular-matrix inversion, are given.**

Index Terms — **Data distribution, data flow, parameter method, period, recurrence equation, systolic array, velocity.**

I. Introduction

THE evolution in very large scale integration (VLSI) technology has had a great impact on computer architecture. Specialized algorithms can be implemented on a VLSI chip using multiple, regularly connected processing elements (PE's) to exploit the great potential of pipelining and multiprocessing. This type of array processor has been referred to as a *systolic array* [11]. One of the many advantages of this approach is that each input data item can be used a number of times once it is accessed, and thus, a high computation throughput can be achieved with only modest bandwidth. Other advantages include modular expandability, simple and regular data and control flows, and use of simple and uniform cells.

Systolic arrays have been classified into (semi-) systolic arrays with global data communications and (pure) systolic arrays without global data communications [11]. In *semi-systolic arrays*, a data item accessed from memory is broadcast to and used by a number of possibly nonidentical cells concurrently. Although this approach is potentially faster than systolic arrays without data broadcast, providing (or collecting) a data item to (or from) all the cells in each cycle requires the use of a global bus that may eventually slow down the processing speed as the number of cells increases. On the other hand, a *pure systolic array* eliminates the use of broadcast buses and implements the algorithm in pipelines extending in different directions. Several data items flowing along different pipes with the same or different rates may meet and interact. The PE's operate synchronously, that is, each data item must stay in a PE for one and only one clock cycle, and all the necessary operands to be processed by a PE in each computational step must arrive at this PE simultaneously. This mode of pipelining is referred to as *systolic processing*.

In general, systolic arrays are somewhat inflexible to implement because they must be algorithmically specialized. Studies have been made to design reconfigurable interconnections for the pure systolic array approach in order to provide the flexibility needed while retaining the benefits of uniformity and locality [17]. Another approach is based on the mapping of different algorithms onto a fixed computer architecture. This is exemplified in the mapping of algorithms onto wavefront array processors [12] and linear array of processors [16]. Special controls are included in order to tailor the algorithms to the architectures.

The efficient mapping of algorithms onto architectures is very important in a direct implementation, and also essential as a benchmark for comparison when algorithms are mapped onto a fixed architecture. Previously, the mapping of algorithms has been done in a rather ad hoc fashion. The proposed designs usually represent feasible but suboptimal solutions. Previous studies can be classified into four categories. In the first type, transformations are performed at the algorithm-representation level, and a direct mapping is made from this level to the architecture [8], [18], [26]. In the second type, transformations are performed at the algorithm-model level, and there are procedures for deriving the model from the algorithm representation and for mapping the model into hardware [1], [2], [4], [15], [16], [21]–[25], [28]–[30]. In the third type, transformations are performed on a previously designed architecture to obtain a new architecture [27]. In the last type, transformations are performed to map the systolic architecture into the function implemented and to prove the correctness of the design [19]. A critical review of the methodologies can be found in [20].

For the above methods, very little can be said about the optimality of the resulting design. Most of the transformational methods are heuristic in nature. Although minimum execution-time algorithms can be systematically derived and heuristics for optimizing space are studied [4], no formal method for optimally mapping the transformed algorithms onto systolic architectures is proposed.

The objective of this paper is to provide a *systematic methodology* for the *design of optimal pure planar systolic arrays* for algorithms that are representable as *linear recurrence processes*. Planar systolic arrays are those in which the interconnections can be laid out in a plane without crossing each other. Linear recurrence processes (to be defined later) with one- or two-dimensional inputs are suitable for imple-

Manuscript received April 4, 1983; revised January 25, 1984. This work was supported in part by a David Ross Grant from the Purdue Research Foundation and by the National Science Foundation under Grant ECS80-16580.

The authors are with the School of Electrical Engineering, Purdue University, West Lafayette, IN 47907.

0018-9340/85/0100-0066$01.00

mentation on these systolic arrays, and inputs with a larger number of dimensions have to be partitioned first. The merit of a design is measured by the completion time (T) or the product of the VLSI chip area and the completion time ($A \times T$) or the product of the VLSI chip area and the square of the completion time ($A \times T^2$). However, area is a complex function of the number of PE's, the number of buffers, the interconnection pattern, and the available technology, and has to be assessed for each configuration separately. The discussion on area measurement is outside the scope of this paper, and the number of PE's is used as an indicator for the area required.

Systolic designs are characterized by three classes of parameters: velocities of data flows, spatial distributions of data, and periods of computation. The relationships among these parameters are represented as constraint equations, and the completion time and hardware complexity of a design can be expressed in terms of these parameters (Section III). A systematic methodology to solve the design problem as an optimization problem is discussed in Section IV. Examples illustrating the method are shown in Section V.

II. Characteristics of Recurrence Processes

In this section, the characteristics of recurrence processes and their relationships to systolic processing are discussed. The following are examples of linear recurrences.

1) Finite Impulse Response (FIR) Filtering: FIR filtering, an important technique in signal processing [3], [5], [10], can be considered as a matrix–vector multiplication

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} a_1 & \cdots & a_m & \cdots & 0 \\ & a_1 & \cdots & a_m & \vdots \\ \vdots & & \ddots & & a_m \\ 0 & \cdots & & & \vdots \\ & & & & a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad n > m \quad (2.1)$$

where the matrix is an upper-triangular Toeplitz band matrix. $y_i, 1 \le i \le n$, is a summation of product terms, the kth term of which is denoted as y_i^k. The operation can be written as a recurrence equation

$$\begin{aligned} y_i^0 &= 0 && 1 \le i \le n \\ y_i^k &= y_i^{k-1} + a_k x_{i+k-1} \\ & && 1 \le i \le n,\ 1 \le k \le m, x_j = 0 \quad \text{for } j > n. \end{aligned} \quad (2.2)$$

2) Two-Dimensional Matrix Multiplication [5], [7], [9], [11]: Each term in the multiplication of two two-dimensional matrices, $C = A \times B$, is a summation of product terms. The kth term of $c_{i,j}$ can be computed as

$$\begin{aligned} c_{i,j}^0 &= 0 && 1 \le i, j \le n \\ c_{i,j}^k &= c_{i,j}^{k-1} + a_{i,k} b_{k,j} && 1 \le i, j, k \le n. \end{aligned} \quad (2.3)$$

3) Discrete Fourier Transform (DFT): DFT is usually regarded as a matrix–vector multiplication, $Y = \Omega \times X$, in which Ω contains power terms of $\omega = e^{2\pi\sqrt{-1}/n}$, the nth root of unity. Each term of Y can be expressed as a summation of product terms: $y_i = \sum_{k=0}^{n-1} x_k \omega^{ik}, 0 \le i \le n-1$. This form of representation requires additions, multiplications, and power operations. Rewriting it in a different way, the kth term of y_i is

$$\begin{aligned} y_i^0 &= 0 && 0 \le i \le n-1 \\ y_i^k &= y_i^{k-1}\omega^i + x_{n-k} && 1 \le k \le n,\ 0 \le i \le n-1. \end{aligned} \quad (2.4)$$

This representation requires multiplications and additions only.

4) Polynomial Multiplication: The multiplication of two polynomials $(\sum_{i=0}^{n-1} a_i x^i)(\sum_{j=0}^{n-1} b_j x^j) = (\sum_{i=0}^{2n-2} c_i x^i)$ can be reduced to finding the coefficients of the resulting polynomial. The kth term of c_i is expressed as

$$\begin{aligned} c_i^0 &= 0 && 0 \le i \le 2n-2 \\ c_i^k &= c_i^{k-1} + a_{k-1} b_{i-k+1} && 0 \le i \le 2n-2, \\ & && 1 \le k \le n,\ b_j = 0 \\ & && \text{for } j < 0 \text{ or } j \ge n. \end{aligned} \quad (2.5)$$

5) Deconvolution: This is the inverse of FIR filtering [see (2.1)] that solves for vector X given vector Y and the Toeplitz matrix. By equating the product on the left-hand side with vector Y on the right-hand side, $a_1 x_i = y_i - \sum_{k=1}^{m-1} a_{m-k+1} x_{i+m-k}, 1 \le i \le n$. This can be expressed as a recurrence with a temporary variable z_i

$$\begin{aligned} z_i^0 &= y_i && 1 \le i \le n \\ z_i^k &= z_i^{k-1} - a_{m-k+1} x_{i+m-k} \\ & && 1 \le k \le m-1,\ 1 \le i \le n,\ x_j = 0 \\ & && \text{for } j > n \\ x_i &= \frac{z_i^{m-1}}{a_1} && 1 \le i \le n. \end{aligned} \quad (2.6)$$

6) Triangular Matrix Inversion: Given an upper-triangular matrix U such that $u_{i,j} = 0, i > j, V = U^{-1}$ (the inverse of U) can be expressed as a recurrence by using a temporary variable $w_{i,j}$ [6]

$$\begin{aligned} w_{i,j}^{j+1} &= 0 && 1 \le i < j \le n \\ w_{i,j}^k &= w_{i,j}^{k+1} - u_{i,k} v_{k,j} && 1 \le i < k \le j \le n \\ v_{i,j} &= \frac{w_{i,j}^{i+1}}{u_{i,i}} && 1 \le i < j \le n \\ v_{i,i} &= \frac{1}{u_{i,i}} && 1 \le i \le n. \end{aligned} \quad (2.7)$$

7) Two-Dimensional Tuple Comparison: Given two two-dimensional matrices A and B, the operation of finding whether the ith row of A is identical to the jth row of B can be expressed as a recurrence [9]

$$\begin{aligned} c_{i,j}^0 &= \text{TRUE} && 1 \le i, j \le n \\ c_{i,j}^k &= c_{i,j}^{k-1} \wedge (a_{i,k} = b_{j,k}) && 1 \le i, j, k \le n. \end{aligned} \quad (2.8)$$

Equations (2.2) and (2.4)–(2.6) above are one-dimensional

recurrences, while others represent two-dimensional recurrences.

In general, *linear recurrences* for the computation of a two-dimensional result Z from two two-dimensional inputs X and Y can be expressed as

$$z^k_{i,j} = f[z^{k-\delta}_{i,j}, x(i,k), y(k,j)] \qquad \delta = 1 \text{ or } -1 \qquad (2.9)$$

where f is a function to be executed by a PE, and k is a positive integer bounded by a linear function of i, j, and the problem size. The size of a problem is characterized by a finite set of integers. For example, the size of the FIR-filtering problem [see (2.2)] is characterized by $\{n, m\}$. $z^{k-\delta}_{i,j}$ is the intermediate result of the last step of iterative computation at position (i, j). $x(i,k)$ and $y(k,j)$ are linear functions in linear recurrences to define the indexes of X and Y in the kth step of iteration. In the following discussion, the coefficients of i, j, and k in $x(i,k)$ and $y(k,j)$ are 1 or -1. This assumption will be extended in Section V so that the coefficients can be any integer. δ is defined within a recurrence to indicate the order of evaluation. When $\delta = -1$, the recurrence is called a *forward recurrence*, and z^k is defined in terms of z^{k+1}. When $\delta = 1$, the recurrence is termed a *backward recurrence*, and z^k is defined in terms of z^{k-1}. The triangular matrix inversion recurrence defined above is a forward recurrence, while others are backward recurrences.

There are four ways of representing a recurrence process when the operations performed in computing $z_{i,j}$ are commutative: forward recurrence, backward recurrence, and the corresponding recurrences in which the evaluation order of terms are reversed. As an example, the following three recurrences are equivalent ways of representing matrix multiplication in addition to (2.3):

$$c^0_{i,j} = 0; \quad c^k_{i,j} = c^{k-1}_{i,j} + a_{i,n-k+1}b_{n-k+1,j} \quad k = 1, \cdots, n \qquad (2.10)$$

$$c^{n+1}_{i,j} = 0; \quad c^k_{i,j} = c^{k+1}_{i,j} + a_{i,k}b_{k,j} \quad k = n, \cdots, 1 \qquad (2.11)$$

$$c^{n+1}_{i,j} = 0; \quad c^k_{i,j} = c^{k+1}_{i,j} + a_{i,n-k+1}b_{n-k+1,j} \quad k = n, \cdots, 1. \qquad (2.12)$$

These four formulations represent only two unique ways of computing $c_{i,j}$. Equations (2.10) and (2.11) are equivalent as far as the evaluation order of terms is concerned. The same can be said about (2.3) and (2.12). In designing systolic algorithms, only the evaluation order is important, and it is insignificant whether a recurrence is written in the forward or backward form. In this paper, it is assumed that the two alternate ways of solving the same problem are expressed as backward recurrences. Since the complexities of the resulting design may depend on the order of evaluation, the two alternative recurrences must be studied.

The key to systolic processing is that the appropriate data must be in the appropriate place at the time they have to be processed. That is, both timing and data distributions are very important, and the relationship between them must be identified. The coupling between time and space is provided by index k in the general recurrence formula. The superscript k is concerned with time, that is, the number of steps of iterative operation, whereas the subscript k is concerned with data distribution. For example, in matrix multiplication, $c^k_{i,j}$ can be computed provided that $c^{k-1}_{i,j}$, $a_{i,k}$, and $b_{k,j}$ arrive at the same PE simultaneously. In other words, the kth step of computing $c_{i,j}$ requires the intermediate result of the $(k-1)$st step of computing $c_{i,j}$, an entry in row i and column k of matrix A, and an entry in row k and column j of matrix B.

III. Parameters for Designing Systolic Arrays

In this section, a set of parameters characterizing the behavior and the correctness of systolic arrays are defined [13]. These parameters are illustrated with respect to the matrix-multiplication problem that can be represented in a backward recurrence $c^k_{i,j} = f[c^{k-1}_{i,j}, a(i,k), b(k,j)]$ [see (2.3)]. A possible systolic design is depicted in Fig. 1. The data flows of the three rhomboidal data blocks are in three directions: A moves towards the north, B moves towards $-120°$ north, and C moves towards $-60°$ north. During a clock cycle, each PE receives three data items from three different pipes and executes a multiply–add operation. These data items advance into neighboring PE's along their own pipes synchronously in the next clock cycle. This design is for illustration and requires the minimal completion time but not the minimal number of PE's. If hardware or #PE $\times T$ or #PE $\times T^2$ is to be optimized, one of the matrices has to be stationary; however, the concept of vector composition of zero vectors cannot be depicted clearly.

A systolic array consists of a mesh of interconnected PE's. The distance between two directly connected neighboring PE's is defined to be unity. If buffers exist between two adjacent PE's, they are equally spaced along the link. A clock cycle is a unit of time during which one iterative operation is computed in a PE and data advance into neighboring PE's or buffers, or a datum advances from one buffer into the next stage.

The ways that data are fed into a systolic array are related and can be characterized by three parameters: velocity, space, and time. If these parameters are known, the systolic array can be completely determined. The first two parameters defined below are vectors;[1] the third parameter is a scalar.

Parameter 1—Velocity of Data Flow: The *velocity* of a datum x is defined as the directional distance passed by x during a clock cycle and is denoted as $\vec{x}_d$. Since the distance between adjacent PE's is unity, and buffers, if they exist, are equally spaced between PE's, the magnitude of $\vec{x}_d$ must be a rational number of the form i/j where i and j are integers and $i \leq j$. This means that in j clock cycles, x has propagated through i PE's and $j - i$ buffers. If $i = 1$, then there are $j - 1$ buffers between two neighboring PE's in the pipelining direction of x. Otherwise, there are $j - i$ buffers between $i + 1$ PE's, and their positions have to be determined from the recurrence process. The locations of these buffers for one of the input directions can be chosen randomly, and these determine uniquely the locations of buffers for the other input and output directions.

[1] All vectors in this paper are indicated by symbols in bold letters with an arrow on top (e.g., $\vec{x}$). The magnitude (absolute value) of a vector is enclosed in vertical bars. A negative value indicates that the direction of the vector is reversed.

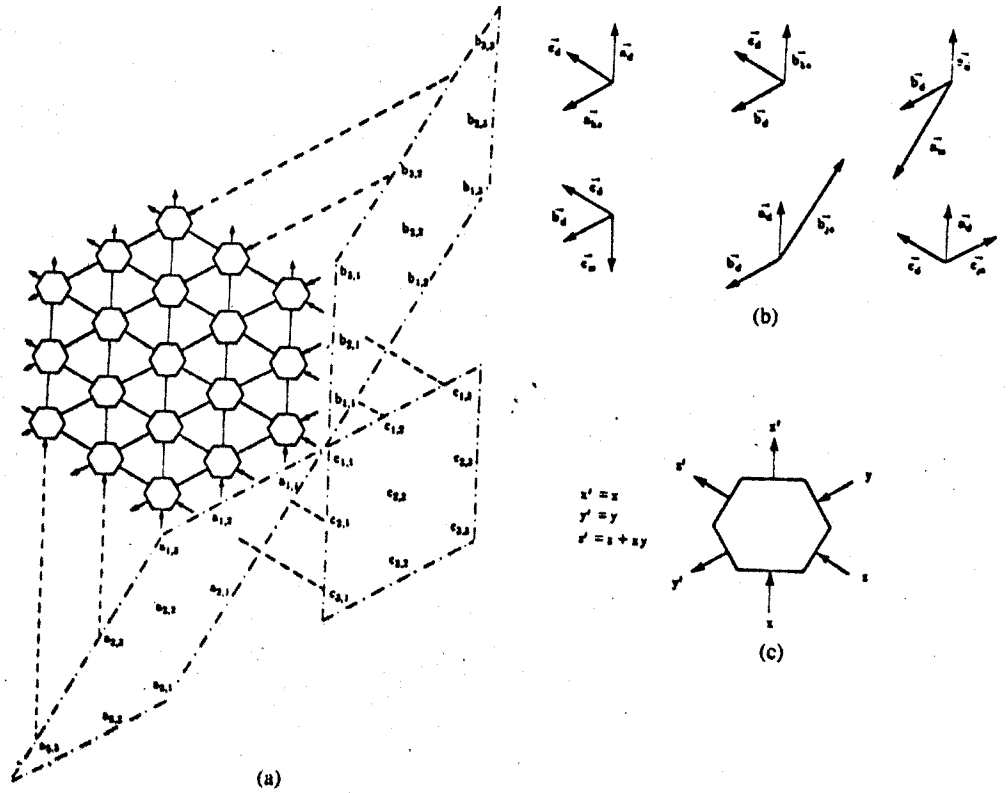

Fig. 1. The systolic processor for two-dimensional matrix multiplication [$c^0_{i,j} = 0$; $c^k_{i,j} = c^{k-1}_{i,j} + a_{i,k}b_{k,j}$, $1 \le i, j, k \le n$]. (a) Systolic processor. (b) Vector equations with ($t_i = t_j = t_k = 1$). (c) Structure of cell.

Parameter 2—Data Distribution: For a two-dimensional array X used as input or output of a systolic array, the elements along a row or a column are arranged in a straight line and are equally spaced as they pass through the systolic array, and the relative positions of the elements are iteration independent. Other forms of data distributions are not considered in this paper. Suppose the row and column indexes of X are i and j, respectively. The *row displacement* of X is defined as the directional distance between $x_{i,j}$ and $x_{i+1,j}$ as X passes through the systolic array and is denoted by $\vec{x}_{is}$. Similarly, the *column displacement* of X ($\vec{x}_{js}$) is defined as the directional distance between $x_{i,j}$ and $x_{i,j+1}$. If X is a one-dimensional array, the index in accessing X is implicit, and the *item displacement* of X ($\vec{x}_s$) is defined as the directional distance between x_i and x_{i+1}. Since data are equally spaced along rows and columns, the row and column displacements are independent of the values of i and j. Note that the direction of data distribution is defined along a subscript-increasing direction, and the magnitudes of all data-distribution vectors are nonzero rational numbers due to pipelining.

Referring to Fig. 1, array A is referenced through indexes i and k in the recurrence formula; hence, the data distribution vectors are defined by $\vec{a}_{is}$ and $\vec{a}_{ks}$.

Parameter 3—Period: Suppose the time at which a computation is performed is defined by the function τ_c, and the time at which an input is accessed for a particular computation is τ_a. The *periods of i and j* for two-dimensional outputs are defined as

$$t_i = \tau_c(z^k_{i+1,j}) - \tau_c(z^k_{i,j}) \tag{3.1}$$

$$t_j = \tau_c(z^k_{i,j+1}) - \tau_c(z^k_{i,j}). \tag{3.2}$$

In computing $z^k_{i,j}$ ($= f[z^{k-1}_{i,j}, x(i,k), y(k,j)]$), it is assumed that the recurrence is expressed in a backward form or has been converted into a backward form, and hence, $z^{k+1}_{i,j}$ ($= f[z^k_{i,j}, x(i,k+1), y(k+1,j)]$) is evaluated after $z^k_{i,j}$. Define the *period of iterative computation* for two-dimensional outputs as

$$t_k = \tau_c(z^{k+1}_{i,j}) - \tau_c(z^k_{i,j}). \tag{3.3}$$

Note that t_k is always positive. In computing $z_{i,j}$, items $x_{i,k}$ and $x_{i,k+1}$ are accessed sequentially, and so are $y_{k,j}$ and $y_{k+1,j}$. Define the *periods of X and Y with respect to k* in the computation of $z_{i,j}$ as the time between accessing successive elements of X and Y. Formally,

$$t_{kx} = \tau_a(x_{i,k+1}) - \tau_a(x_{i,k}) \tag{3.4}$$

$$t_{ky} = \tau_a(y_{k+1,j}) - \tau_a(y_{k,j}). \tag{3.5}$$

t_{kx} and t_{ky} may be negative depending on the order of access defined in the subscript-access functions $x(i,k)$ and $y(k,j)$. Since data needed in the computation of $z^{k+1}_{i,j}$ after the computation of $z^k_{i,j}$ must be assembled in time t_k, it is true that

$$t_k = |t_{kx}| = |t_{ky}|. \tag{3.6}$$

As an example, the systolic array in Fig. 1 is implemented with $t_{ka} = t_{kb} = 1$ because $a_{i,k}$ is accessed one cycle before $a_{i,k+1}$, and so are $b_{k,j}$ and $b_{k+1,j}$. It is also seen that $t_i = t_j = 1$. As another example, elements of vector A are accessed in increasing order of indexes in the recurrence for polynomial multiplication [see (2.5)]. The period t_{ka} is, therefore, positive. On the other hand, vector B is accessed in decreasing order of indexes, and hence, t_{kb} is negative.

It is important to note that the computations in a systolic array are periodic, and hence, all the periods are independent

of i, j, and k. Furthermore, periods represent time intervals between computations, and their absolute values must be rational numbers greater than or equal to 1 because directly connected PE's are separated at unit distances, and computations can only be performed in PE's. When absolute values of periods are less than 1, there must be a bus that can broadcast data, and this is outside the scope of our present discussion.

There is a total of 13 parameters for two-dimensional linear recurrences, of which three are for the velocities of data flow, $\vec{x}_d, \vec{y}_d, \vec{z}_d$, six are for data distributions, $\vec{x}_{is}, \vec{x}_{js}$, $\vec{y}_{is}, \vec{y}_{js}, \vec{z}_{is}, \vec{z}_{js}$, and four are for the periods, t_{kx}, t_{ky}, t_i, t_j. For one-dimensional problems, only nine parameters $\vec{x}_d, \vec{y}_d, \vec{z}_d$, $\vec{x}_s, \vec{y}_s, \vec{z}_s, t_{kx}, t_{ky}$, and t_i exist. These parameters can be used in constraint equations to govern the correctness of the design and in performance measures to define the number of PE's needed and the completion time. The following theorem states the relationships among these parameters.

Theorem 1 (Theorem of Systolic Processing): Suppose a two-dimensional recurrence computation $z^k_{i,j} = f[z^{k-1}_{i,j}, x(i,k), y(k,j)]$ is implemented in a systolic array, then the velocities, data distributions, and periods must satisfy the following vector equations:

$$t_{kx}\vec{x}_d + \vec{x}_{ks} = t_{kx}\vec{z}_d \quad \text{(data movement for } X \text{ and } Z \text{ between computing } z^{k-1}_{i,j} \text{ and } z^k_{i,j}) \tag{3.7}$$

$$t_{ky}\vec{y}_d + \vec{y}_{ks} = t_{ky}\vec{z}_d \quad \text{(data movement for } Y \text{ and } Z \text{ between computing } z^{k-1}_{i,j} \text{ and } z^k_{i,j}) \tag{3.8}$$

$$t_i\vec{x}_d + \vec{x}_{is} = t_i\vec{y}_d \quad \text{(data movement for } X \text{ and } Y \text{ between computing } z^k_{i,j} \text{ and } z^k_{i+1,j}) \tag{3.9}$$

$$t_i\vec{z}_d + \vec{z}_{is} = t_i\vec{y}_d \quad \text{(data movement for } Y \text{ and } Z \text{ between computing } z^k_{i,j} \text{ and } z^k_{i+1,j}) \tag{3.10}$$

$$t_j\vec{y}_d + \vec{y}_{js} = t_j\vec{x}_d \quad \text{(data movement for } X \text{ and } Y \text{ between computing } z^k_{i,j} \text{ and } z^k_{i,j+1}) \tag{3.11}$$

$$t_j\vec{z}_d + \vec{z}_{js} = t_j\vec{x}_d \quad \text{(data movement for } X \text{ and } Z \text{ between computing } z^k_{i,j} \text{ and } z^k_{i,j+1}). \tag{3.12}$$

For one-dimensional problems, only (3.7)–(3.10) are necessary.

Proof: Without loss of generality, orthogonally connected PE's with diagonal connections are used in our proof. It is also assumed that $t_k = t_{kx} = t_{ky} > 0$. In Fig. 2, A, B, C, and D represent four PE's that do not have to be directly connected. While PE C is computing $z^k_{i,j} = f[z^{k-1}_{i,j}, x(i,k), y(k,j)]$, $x_{x(i,k+1)}$ is in PE B,[2] and $y_{y(k+1,j)}$ is in PE D [Fig. 2(a)]. Since $t_{kx} = t_{ky} > 0$, $\vec{CB}$ represents $\vec{x}_{ks}$, and $\vec{CD}$ represents $\vec{y}_{ks}$. According to the characteristics of systolic processing, the operands needed in the next iteration must arrive at the same PE simultaneously after t_k units of time. Hence, $z^k_{i,j}$, $x_{x(i,k+1)}$, and $y_{y(k+1,j)}$ arrive at PE A simultaneously that computes $z^{k+1}_{i,j} = f[z^k_{i,j}, x(i,k+1), y(k+1,j)]$ [Fig. 2(b)]. We have $\vec{CA} = t_k\vec{z}_d, \vec{BA} = t_k\vec{x}_d, \vec{DA} = t_k\vec{y}_d$. From the principle of vector composition, $\vec{BA} + \vec{CB} = \vec{CA}$ and $\vec{DA} + \vec{CD} = \vec{CA}$, so (3.7) and (3.8) are proved. The other cases in which t_{kx} and t_{ky} have different signs or are negative can be proved similarly.

To prove (3.9) and (3.10), suppose that while PE D is computing $z^k_{i,j} = f[z^{k-1}_{i,j}, x(i,k), y(k,j)]$, PE C is computing $z^p_{i+1,j}$, $p < k$, and PE B has $x_{x(i+1,k)}$ [Fig. 2(c)]. Therefore, $\vec{DC} = \vec{z}_{is}$, and $\vec{DB} = \vec{x}_{is}$. In accordance with the characteristics of systolic processing, $k - p$ steps of the iterative computation are performed after t_i units of time, and $z^{k-1}_{i+1,j}$, $x_{x(i+1,k)}$, and $y_{y(k,j)}$ arrive at PE A for the computation of $z^k_{i+1,j} = f[z^{k-1}_{i+1,j}, x(i+1,k), y(k,j)]$. Therefore, $\vec{BA} = t_i\vec{x}_d$, $\vec{CA} = t_i\vec{z}_d$, and $\vec{DA} = t_i\vec{y}_d$. From the principle of vector composition, $\vec{BA} + \vec{DB} = \vec{DA}$ and $\vec{CA} + \vec{DC} = \vec{DA}$; thus, (3.9) and (3.10) are proved. The cases for negative periods can be proved similarly.

Last, (3.11) and (3.12) can be proved similarly and will not be shown here. □

Performance of a systolic design can be expressed in terms of the defined parameters. The number of PE's required (denoted by #PE) is studied here, and the completion time can be expressed as a function of the PE configuration and velocity.

The *number of streams of data flow* of a matrix X in the direction of data flow is defined as the number of distinct lines that must be drawn in parallel to the direction of data flow so that each element of the matrix lies in exactly one line. For a one-dimensional vector with n elements, the number of streams can be one (elements are fed serially) or n (elements are fed in parallel). For a two-dimensional n-by-n matrix, this number depends on the directions of $\vec{x}_{is}$ and $\vec{x}_{js}$. If $\vec{x}_{is}$ and $\vec{x}_{js}$ are in the same or opposite directions, then the number of streams of data flow can be one (serial input) or n^2 (parallel input). If $\vec{x}_{is}$ and $\vec{x}_{js}$ are in different directions, then the number of streams is given by $n + (n-1)\ell$ where ℓ is an integer between 0 and n. To see that this is true, when ℓ is 0, each row or column of the matrix lies in a single stream, and the number of streams is n. The extreme case happens when each element of the matrix lies in a different stream, and the number of streams is n^2. As an example, each matrix in Fig. 1 has five streams of data flow ($n = 3, \ell = 1$).

#PE depends on the directions in which the inputs are moving. There are four possible cases. First, one of the input matrices or the output matrix is stationary, and the others are moving. Assuming that all the elements of the stationary matrix are used in the computation, #PE is given by the size of the stationary matrix. Second, both input and output matrices are moving in the same or opposite directions. #PE is given by the product of the minimum number of streams of data flow and the distance traveled between the time that the first elements of the input matrices meet and the time that the last elements of the input matrices meet. Third, there are two independent directions of data flow. If the two input matrices are flowing in the same or opposite directions and the output matrix is flowing in a different direction, #PE is given by the number of streams of data flow of the output matrix. If the two input matrices are flowing in different

[2] $x_{x(i,j)}$ represents the element of X defined by the subscript-access function $x(i,j)$.

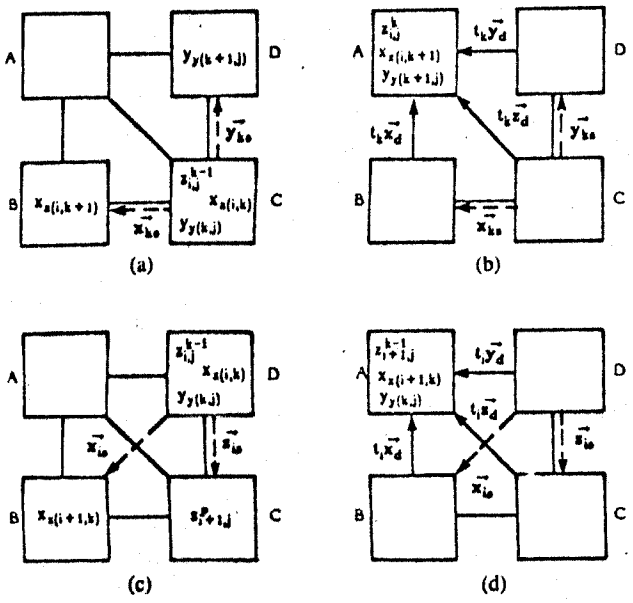

Fig. 2. Proof of Theorem 1.

directions and the output matrix is flowing in a direction of one of the inputs, and if each stream of data flow in an input matrix has to be interacted with every other stream of data flow in the other input matrix, #PE is given by the product of the numbers of streams of data flow of the input matrices. When the last assumption on the interaction of data flows is false, #PE reduced by a term that has to be determined from the recurrence. Lastly, there are three independent directions of data flow. In this case, #PE for the two input matrices can be computed as before. However, this number can be further reduced by the flow of the output matrix.

For example, matrices A, B, and C in Fig. 1 are flowing in three different directions and have five streams of data flow each. #PE for A and B is $5 \times 5 = 25$. However, matrix C is flowing in a different direction which cuts off two corners of the PE configuration. #PE is reduced to 19.

Based on the last item of the result that will be computed, the time required for all the computations in a systolic array can be derived in terms of t_k, t_i, t_j, and the problem size to be solved. However, any load and drain times of the input–output matrices would depend on the directions and distributions of data flows on a given configuration of the PE's, and have to be analyzed for each case individually. Other design requirements such as the number of input–output pins can also be expressed in terms of the PE configuration and the defined parameters.

For example, the time needed for multiplying two n-by-n matrices is $nt_k + (n-1)|t_i| + (n-1)|t_j|$ since, from (2.3), it takes nt_k steps to compute $c_{1,1}$, $(n-1)|t_i|$ steps from computing $c_{1,1}$ to $c_{n,1}$, and $(n-1)|t_j|$ steps from computing $c_{n,1}$ to $c_{n,n}$. In the design in Fig. 1, no load or drain time is necessary.

We have been less specific in the discussion on hardware and time complexities because they are dependent on the recurrence. In fact, closed-form expressions for these complexities under all possible situations are very complex, and it would be better to analyze the cases separately. This will be illustrated in the examples.

IV. Design Methodology for Systolic Arrays

The design of an optimal systolic array for a linear recurrence can be formulated into an optimization problem. The constraints of optimization are provided by Theorem 1, which shows the fundamental space–time relationships and governs the corrections in systolic processing. The objective function to be minimized can be expressed in terms of the parameters of systolic processing and the problem size. We first show the formulation of the optimization and then explain the various constraints. The design problem is formulated as

$$\text{minimize} \quad \#\text{PE} \times T^2 \quad \text{or} \quad \#\text{PE} \times T \quad \text{or} \quad T \tag{4.1}$$

subject to

(3.7)–(3.12)

and

$$\frac{1}{t_{j\max}} \le |\vec{x}_d| \le 1 \quad \text{or} \quad |\vec{x}_d| = 0 \tag{4.2}$$

$$\frac{1}{t_{i\max}} \le |\vec{y}_d| \le 1 \quad \text{or} \quad |\vec{y}_d| = 0 \tag{4.3}$$

$$\frac{1}{t_{k\max}} \le |\vec{z}_d| \le 1 \quad \text{or} \quad |\vec{z}_d| = 0 \tag{4.4}$$

$$1 \le |t_k| \le t_{k\max}; \quad 1 \le |t_i| \le t_{i\max}; \quad 1 \le |t_j| \le t_{j\max} \tag{4.5}$$

$$|t_k||\vec{z}_d| = k_1 \le t_{k\max}; \; |t_i||\vec{y}_d| = k_2 \le t_{i\max}; \; |t_j||\vec{x}_d| = k_3 \le t_{j\max} \tag{4.6}$$

$$|\vec{x}_{is}| \ne 0; \quad |\vec{x}_{ks}| \ne 0; \quad |\vec{y}_{ks}| \ne 0 \tag{4.7}$$

$$|\vec{y}_{js}| \ne 0; \quad |\vec{z}_{is}| \ne 0; \quad |\vec{z}_{js}| \ne 0 \tag{4.8}$$

$$t_k = |t_{kx}| = |t_{ky}|$$

recurrence determines relative signs of t_{kx} and t_{ky}. (4.9)

$k_1, k_2, k_3, t_{k\max}, t_{i\max}$, and $t_{j\max}$ are integers. All other parameters are rational numbers. Moreover, $t_{k\max}$, $t_{i\max}$, and $t_{j\max}$ are functions of the problem size and T_{serial}, the number of times that function f in the recurrence has to be executed in order to compute all the required results.

The terms in (4.6) represent the distances traversed between computations. Since a computation must be performed in a PE, the distance traversed must coincide with the locations of PE's. The upper bounds of k_1, k_2, and k_3 are the maximum values of t_k, t_i, and t_j because the maximum values of speeds are 1 [see (4.2), (4.3), (4.4)]. To derive these upper bounds, recall that the total computation time T is a function of t_k, $|t_i|$, and $|t_j|$. In order for systolic processing to be more efficient than a serial computation, it is necessary for $T \le T_{\text{serial}}$. By using the minimum values for two of the periods ($t_k = 1, |t_i| = 1, |t_j| = 1$) in the above inequality, the upper bound for the other period ($t_{k\max}$, $t_{i\max}$, or $t_{j\max}$) is obtained. As an example, in multiplying two 3-by-3 matrices, $T = 3t_k + 2|t_i| + 2|t_j| \le T_{\text{serial}} = 27$. By setting $|t_i| = |t_j| = 1$, $t_{k\max} = \lceil (27-4)/3 \rceil = 8$. Similarly, $t_{i\max} = t_{j\max} = 11$. Sometimes, T is not a function of a period, say t_i. According to the definition of t_i, this is the time between computing $z^k_{i,j}$ and $z^k_{i+1,j}$. Suppose i ranges from 1 to $n-1$; then $n|t_i|$ units of time are needed for the computation

from $z^t_{1,j}$ to $z^t_{n,j}$. The inequality $n|t_i| \le T_{\text{serial}}$ allows $t_{i\max}$ to be solved.

The constraints in (4.2), (4.3), (4.4) state that data cannot travel more than one unit per unit of time because directly connected PE's are separated at unit distances, and no broadcasting is allowed. The lower bounds on velocities are derived from (4.6). Velocities smaller than the given lower bounds do not have to be considered because there exists a more efficient design of computing the recurrence in a single PE (with time T_{serial}). The constraints in (4.5) follow directly from the definitions of t_k, t_i, and t_j.

The set of distinct values of speeds and periods are related to the number of buffers among the PE's. Specifically, for $t_k, |\vec{z}_d|, k_1$, and $t_{k\max}, k_1$ represents the number of PE's traversed by a datum between two successive iterative computations, the maximum of which is $t_{k\max}$. Let η be the maximum number of iterations required for computing a result (η is usually a linear function of the problem size). For a given k_1, the maximum number of PE's in the pipeline is $(\eta - 1)k_1 + 1$ since the first iteration is computed in a single PE, and the remaining $\eta - 1$ iterations require $(\eta - 1)k_1$ PE's. The total number of buffers in the pipeline, b, satisfies

$$0 \le b \le [(\eta - 1)t_{k\max} + 1] - [(\eta - 1)k_1 + 1] = \eta(t_{k\max} - k_1). \quad (4.10)$$

Once b is chosen, $|\vec{z}_d|$ and $|t_k|$ can be determined

$$|\vec{z}_d| = \frac{(\eta - 1)k_1}{(\eta - 1)k_1 + b} \quad (4.11)$$

$$|t_k| = \frac{k_1}{|\vec{z}_d|} = k_1 + \frac{b}{\eta - 1}. \quad (4.12)$$

As a result, there are $O(\eta t^2_{k\max})$ combinations of values of t_k and $|\vec{z}_d|$. Similarly, for t_i and $|\vec{y}_d|$, t_j and $|\vec{x}_d|$, there are $O(\eta t^2_{i\max})$ and $O(\eta t^2_{j\max})$ combinations of values, respectively.

Data flow in the above optimization problem can be in one, two, or three independent directions. There are five possibilities: a) all inputs and outputs are flowing in the same or opposite directions; b) the inputs are flowing in the same or opposite directions, and the output is in a different direction; c), d) the inputs are flowing in two different directions, and the output is flowing in a direction of one of the inputs; and e) inputs and outputs are flowing in three different directions. The directions of vectors can be reversed, and velocities can be zero vectors. Due to the assumption of unit distance between directly connected PE's, when data are flowing in two independent directions, they must be orthogonal to each other; and when data are flowing in three independent directions, they must be in multiples of 120° to each other. Furthermore, the magnitudes of velocities and periods are chosen from a finite set. Therefore, the optimization of design of a systolic array for a given recurrence has a finite search space of complexity $O(\eta^3 t^2_{i\max} t^2_{j\max} t^2_{k\max})$.

The worst case complexity shown above is very large. There are two ways to reduce this complexity. First, instead of requiring that $T \le T_{\text{serial}}$, the requirement that $T \le O(T_{\text{serial}}/\#\text{PE})$ may be used. This reduces the values of $t_{k\max}$, $t_{i\max}$, and $t_{j\max}$, and in turn reduces the search complexity. Second, it is noted that systolic designs for linear recurrences are extendible because the systolic processing equations governing the correctness of the design (Theorem 1) are independent of the problem size. To reduce the search complexity, an optimal design for a smaller problem can be found and is used to extend to the systolic design for a larger version of the same problem. This method can also be applied when the maximum number of PE's that can be implemented in a single chip is smaller than that required by the optimal design. In general, this method will not lead to an optimal design for the original problem. This is due to the fact that the objective function is monotonically increasing with the problem size, and the fact that the objective function for one design is better at a given problem size does not imply that this design is better at a different problem size.

The optimal solution to the above design problem can be found by exhaustive enumeration. However, the search time for an arbitrarily large recurrence can be long, if not impossible. By recognizing that the completion time is a linear function of t_k, t_i, and t_j, their values can be ordered so that periods that do not lead to an optimal solution are eliminated from further consideration each time a feasible solution is found. This strategy is explained with respect to the minimization of completion time and $\#\text{PE} \times T^2$.

To minimize the completion time, the different directions of data flows are first determined (five possibilities). The maximum values of t_k, t_i, and t_j are found. A set of t_k, t_i, and t_j that minimize the completion time is selected from the set of possible values. The speeds of data flows are evaluated from (4.6) by using $k_1 = k_2 = k_3 = 1$. The six remaining unknowns on spatial distributions can be solved from the systolic processing equations (3.7)–(3.12). If no feasible solution is found, the procedure is repeated by finding another set of periods so that the completion time is increased by the least amount. The above steps are carried out for all the five combinations of data flow directions. If no feasible solution is found, one of the k_1, k_2, or k_3 is increased by 1, and the procedure repeats. It is obvious that the first feasible solution found is the optimal solution that minimizes the completion time.

To minimize $\#\text{PE} \times T^2$, it is necessary to know the lower bound on #PE. For linear recurrences with two-dimensional (n-by-n) inputs, the lower bound on #PE can be one (both inputs are serial), n (one input is serial and the other has n streams of data flow), or n^2 (both inputs have n streams of data flow). It is easy to prove from the systolic processing equations that serial inputs *usually* do not lead to feasible solutions, and the lower bound is n^2. Repeating the procedure in minimizing completion time, a feasible design is first found. Suppose this design requires P_1 PE's and T_1 clock cycles to complete, then it is true that any design with $T_2 \ge \sqrt{P_1 T_1}/n$ will not lead to a better solution (since $P_2 T_2 \ge n^2 T^2 \ge P_1 T_1^2$) and can be eliminated from consideration. The search is continued to find better solutions with completion time between T_1 and T_2.

By systematically enumerating and reducing the search space that is a polynomial function of the problem size, the optimal design can be solved very efficiently. The method of designing optimal systolic arrays using the parameters defined in this paper is referred to as the *parameter method*.

The steps in the design are sketched as follows.

Step 1: Write the recurrence formula for the problem to be solved.

Step 2: Write the corresponding systolic processing equations (Theorem 1) and the constraints on the values of parameters.

Step 3: Write the objective function based on the design requirements in terms of the systolic processing parameters and the problem size.

Step 4: Find the parameter values that minimize the objective function by enumerating over the limited search space.

Step 5: Design a basic cell for the systolic array and find a possible interconnection of cells from the parameters obtained. Eliminate cells that do not perform any useful computation.

In the 3-by-3 matrix-multiplication problem, recall that the computation time needed is $3t_k + (3-1)|t_i| + (3-1)|t_j|$. The completion time is minimized when t_k, $|t_i|$, and $|t_j|$ are as small as possible. The search is started with $t_k = t_i = t_j = 1$ on all the combinations of directions of data flows. If no feasible solution is found, the signs of t_i or t_j are negated, and the search repeats. In this example, when data are flowing in three different directions, $t_k = t_i = t_j = 1$ results in a solution that satisfies the constraints of (3.7)–(3.12) and (4.2)–(4.9) and minimizes the completion time. The corresponding vectors are depicted in Fig. 1(b). By using these vectors, the velocities and spatial distributions of data flows can be determined, and a basic cell design is shown in Fig. 1(c). These cells are connected together into a mesh. Some of the cells are eliminated because no computation is performed there. The final VLSI structure is shown in Fig. 1(a). This is the fastest matrix-multiplication scheme, which can be completed in 7 units of time with 19 cells.

On the other hand, if #PE $\times T^2$ is to be minimized, the search has to be continued to find out all the feasible designs with completion time less than $\sqrt{19 \times 7^2/3^2} = 10.2$. By assuming that the output matrix is stationary, a feasible design can be found with $t_k = t_i = t_j = 1$, 7 units of computation time, and 3 units of drain time. This reduces the search space further to finding all feasible designs with less than 10 units of completion time. In fact, this is the optimal solution that minimizes #PE $\times T^2$.

Note that the recurrence for two-dimensional tuple comparison [see (2.8)] is identical to the recurrence for matrix multiplication except for the operations performed in the PE's, and hence, the systolic array for matrix multiplication can be applied in this case.

V. Examples

A. Direct Applications of the Methodology

1) FIR Filtering: The operation can be represented as a one-dimensional linear recurrence $y_i^k = f(y_i^{k-1}, x(k,i), a(k))$ [see (2.2)]. Another recurrence that evaluates the terms in a reverse order can be written as $y_i^k = y_i^{k-1} + a_{m-k+1}x_{m-k+i}$, $1 \le i \le n, 1 \le k \le m$. In both recurrences, the inputs are accessed in the same order, and hence $t_{ka} = t_{kx}$. It takes mt_k units of time to compute y_1 (m is the window size of the FIR filter) and $(n-1)|t_i|$ units of time to compute the remaining y_i's. The total computation time, disregarding possible load and drain times, is $mt_k + (n-1)|t_i|$. For a problem with $m = 4$ and $n = 6$, $T_{\text{serial}} = 24$. From this, $t_{k\max} = \lceil 19/4 \rceil = 5$, $t_{i\max} = \lceil 20/5 \rceil = 4$. The design problem can be formulated as

$$\text{minimize} \quad \#\text{PE} \times [4|t_k| + 5|t_i| + \text{load time} + \text{drain time}]^2 \tag{5.1}$$

subject to

$$t_{kx}\vec{x}_d + \vec{x}_s = t_{kx}\vec{y}_d \tag{5.2}$$

$$t_{ka}\vec{a}_d + \vec{a}_s = t_{ka}\vec{y}_d \tag{5.3}$$

$$t_i\vec{x}_d + \vec{x}_s = t_i\vec{a}_d \tag{5.4}$$

$$t_i\vec{y}_d + \vec{y}_s = t_i\vec{a}_d \tag{5.5}$$

$$\frac{1}{4} \le |\vec{a}_d| \le 1 \quad \text{or} \quad |\vec{a}_d| = 0$$

$$\frac{1}{5} \le |\vec{y}_d| \le 1 \quad \text{or} \quad |\vec{y}_d| = 0$$

$$1 \le t_k \le 5 \qquad 1 \le |t_i| \le 4$$

$$|\vec{a}_s| \ne 0 \qquad |\vec{x}_s| \ne 0 \qquad |\vec{y}_s| \ne 0$$

$$|t_i||\vec{a}_d| = k_1 \le 4 \qquad t_k|\vec{y}_d| = k_2 \le 5$$

$$t_k = |t_{kx}| = |t_{ka}| \qquad t_{kx} = t_{ka}$$

where k_1, k_2 are integers; all the other magnitudes are rational numbers. It is not necessary to bound $|\vec{x}_d|$ because $\vec{x}_d$ is uniquely determined when t_k, t_i, $\vec{y}_d$, and $\vec{a}_d$ are set.

The search space in the above problem is quite reasonable. However, the problem can be solved very efficiently by using the defined search order. First, consider $t_{kx} = t_{ka} = -1$ and $t_i = 1$. Substituting into (5.2)–(5.5) results in four equations with six unknowns. From FIR-filtering applications, it is known that the a_i's are defined constants and can be fixed in the systolic array without preloading. Assume that the a_i's are statically placed in the PE's, $|\vec{a}_d| = 0$, and from (5.3), $\vec{a}_s = -\vec{y}_d$. Since $|\vec{a}_s| \ne 0$ and $|\vec{y}_d| \le 1$, $|\vec{y}_d|$ can be set to 1 or -1. Solving both cases results in a systolic design that requires four PE's, $m + n - 1$ units of computation time and the preloading of $x_1, \cdots, x_m$ into the pipe [14]. The completion time is, therefore, $2m + n - 1$ units.

On the other hand, if $t_{kx} = t_{ka} = 1$, $t_i = -1$, and $|\vec{a}_d| = 0$, then $|\vec{y}_d| = |\vec{y}_s| = 1$, $|\vec{x}_d| = |\vec{x}_s| = 1/2$, and $|\vec{a}_s| = 1$. This is a one-dimensional solution (all the vectors are pointing in the same direction). A feasible systolic design satisfying the above parameters is depicted in Fig. 3. This design does not require elements of X to be preloaded, and the completion time of the algorithm is 9 units. It should be noted that $t_{kx} = t_{ka} = 1$ implies that the recurrence formula in (2.2) is used.

To see that the design in Fig. 3 is optimal, the performance measure #PE $\times T^2$ for $n = 6, m = 4$ is 324. If the number of PE's is decreased to one, $T = T_{\text{serial}} = 24$, and #PE $\times T^2 = 576$. The lower bound on #PE is $m = 4$. The proposed design achieves this lower bound in the minimal completion time and, hence, is optimal.

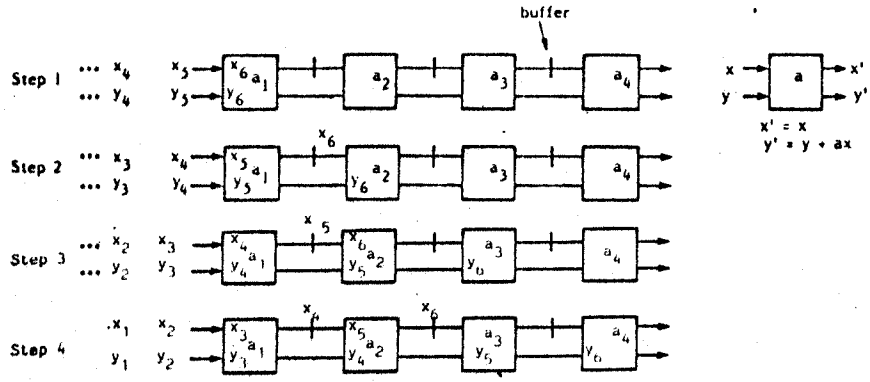

Fig. 3. Systolic array for FIR filtering ($n = 6, m = 4$).

It is of interest to note that discrete Fourier transform [see (2.4)], polynomial multiplication [see (2.5)], and many pattern-matching problems have the same form of recurrence equations as FIR filtering. The systolic design in Fig. 3 can be applied except that different functions are performed in the PE's [14].

2) Band-Matrix Multiplication: The systolic array for matrix multiplication (Fig. 1) can be used for band-matrix multiplication. However, the number of cells required is large considering that most of the terms in the result are zero. By recognizing that the result is also a band matrix, a different recurrence can be written to compute the elements in the band alone. The proposed methodology can be applied to obtain the optimal systolic design.

In this example, we illustrate the flexibility of our design method by showing that additional constraints (based on insights) can be included in the optimization. Given that the band width is m, the most efficient direction of sending the band matrix into a systolic array is along the diagonal. These required directions of data flows can be added as constraints to the original design problem

$$\vec{a}_{is} + \vec{a}_{js} = k_1 \vec{a}_d \qquad k_1 \text{ is a rational number} \qquad (5.6)$$

$$\vec{b}_{is} + \vec{b}_{js} = k_2 \vec{b}_d \qquad k_2 \text{ is a rational number.} \qquad (5.7)$$

The details of solution are not shown here. The resulting design for $m = 3$ has three independent directions of data flow (Fig. 4). The design is optimal and requires m^2 PE's and $m + n - 1$ units of completion time.

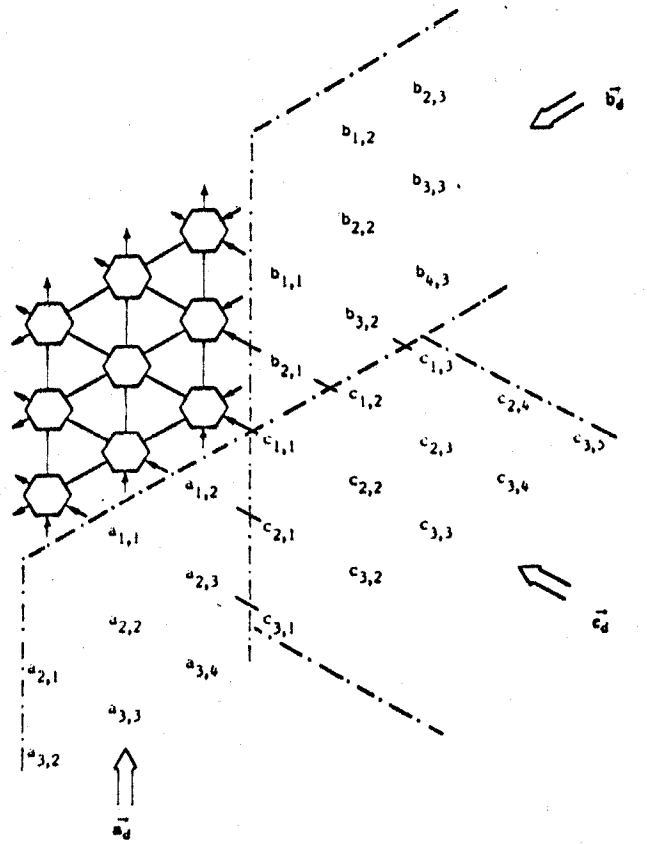

Fig. 4. Systolic array for band-matrix multiplication (the cell used is the same as that of matrix multiplication).

B. Recurrences with Feedback

In this section, the design of systolic arrays for recurrences with feedback are exemplified. When outputs are routed back into a systolic array, the direction of data flows has to be changed. The difficulties in designing such a systolic array are that feedbacks pose another direction of data flows in the inputs, and the correctness of feedbacks is hard to express in systolic processing equations because the format of feedbacks is governed by the way outputs are generated. One method is to treat the feedbacks as an independent input stream and to design the systolic array in the usual way. After an efficient design is obtained, the feedbacks have to be checked to determine whether they are generated before they are fed back into the systolic array. The process is repeated until an efficient and correct design is found.

1) Triangular-Matrix Inversion: The inversion of an upper-triangular matrix U into another upper-triangular matrix V has been represented in a forward recurrence [see (2.7)] and can be rewritten into a backward recurrence $w^k_{i,j} = f[w^{k-1}_{i,j}, u(i,k), v(k,j)]$. Consecutive elements of U and V are accessed in decreasing order of subscripts, and hence, $t_{ku} = t_{kv} < 0$.

By examining the recurrence, it is found that the evaluation of $w_{i,j}$ requires one more term than that of $w_{i,j-1}$. Therefore, $w_{i,j-1}$ can be completed before $w_{i,j}$, which implies that $t_j > 0$. On the other hand, the evaluation of $w_{i,j}$ requires one less term than that of $w_{i-1,j}$, which implies that $t_i < 0$. From the recurrence, it is also seen that the last item to be computed is $v_{1,n}$ rather than $v_{1,1}$. It takes $(n-1)|t_i|$ units of time from computing $v_{n,n}$ to $v_{2,n}$; and $n|t_{ku}|$ units of time to compute $v_{1,n}$.

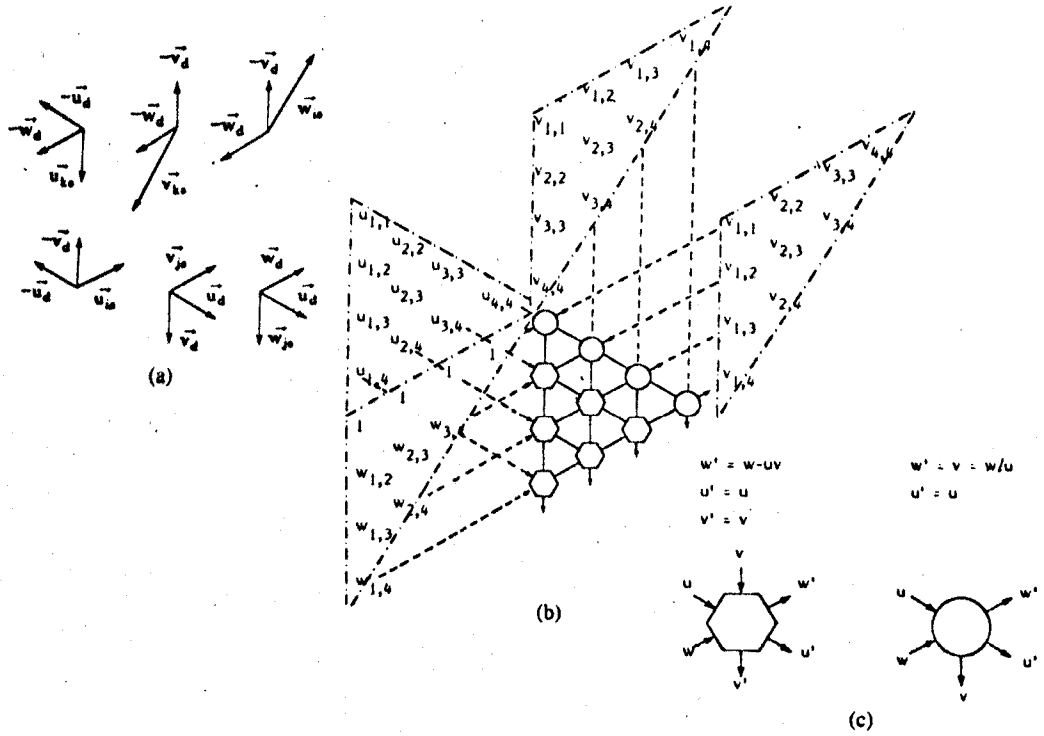

Fig. 5. Systolic array for triangular-matrix inversion ($n = 4$).

The computation time is, thus, $n|t_{ku}| + (n-1)|t_i|$. For a problem with $n = 4$, $T_{\text{serial}} = 20$. By solving $T \leq T_{\text{serial}}$, we have $t_{k\max} = 5$ and $t_{i\max} = 6$. Since $v_{1,1}, \cdots, v_{1,4}$ have to be computed, we have another equality $4|t_j| \leq T_{\text{serial}}$ that results in $t_{j\max} = 5$.

To minimize #PE $\times T^2$, parameters are first chosen to minimize the computation time: $t_{ku} = t_{kv} = -1$, $t_i = -1$, and $t_j = 1$. From these, the following systolic processing equations are obtained:

$$-\vec{u}_d + \vec{u}_{ks} = -\vec{w}_d \tag{5.8}$$

$$-\vec{v}_d + \vec{v}_{ks} = -\vec{w}_d \tag{5.9}$$

$$-\vec{u}_d + \vec{u}_{is} = -\vec{v}_d \tag{5.10}$$

$$-\vec{w}_d + \vec{w}_{is} = -\vec{v}_d \tag{5.11}$$

$$\vec{v}_d + \vec{v}_{js} = \vec{u}_d \tag{5.12}$$

$$\vec{w}_d + \vec{w}_{js} = \vec{u}_d . \tag{5.13}$$

Constraints on parameter values are similar to those stated previously.

By searching through the set of feasible solutions, a set of solution vectors that minimize the completion time are shown in Fig. 5(a). In terms of these vectors, a systolic array for triangular-matrix inversion is depicted in Fig. 5(b). It should be noted that $w_{i,j}$ is considered as an intermediate output with initial value of 0. When $w_{i,j}$ arrives at one of the dividers denoted by circles, the computation of $w_{i,j}$ is finished, and $v_{i,j}$ is generated. $v_{i,j}$ is then fed back and moves downward. Note that in Fig. 5(b), the triangular block of V above the systolic processor represents a pseudo-input-matrix, whereas another triangular block of V to the right of the array shows the output sequence. In this scheme, a triangular array of $n(n+1)/2$ PE's can compute an nth-order triangular-matrix inversion in $2n - 1$ units of time. Since there is no feasible design with one or n PE's, the proposed design minimizes #PE $\times T^2$.

2) Deconvolution: In the previous examples, processing times in the PE's are assumed to be identical. This implies that the period of processing is independent of the PE along the direction of data flow.

In some applications, different operations may be performed in PE's along a direction of data flow. For example, the last operation in deconvolution is division, which may take more time than multiplication and may become the bottleneck of data flow. Referring to the recurrence for deconvolution $z_i^k = f[z_i^{k-1}, x(i,k), a(k)]$ [see (2.6)], the x_i's computed are fed back into the pipe for future computation. Let the delay of a division PE be w and the delay of other PE's be 1. The last iteration of computing z_i and the division of z_i to obtain x_i takes $(w+1)$ units of time. This is the period for the X inputs.

$$|t_i| = w + 1 . \tag{5.14}$$

Due to the extra time involved in division, the design of feedback signals is more complicated than that of triangular-matrix inversion. The fact that inputs and feedbacks are one-dimensional permits us to write down the relationships of data flows more formally in systolic processing equations [Fig. 6(a)]. Suppose PE A is the PE in which z_i^{m-1} is being computed using x_{i+1} before being sent to the division unit (PE D). z_{i-1}^p, $p < m - 1$ (for backward recurrences), exists in PE B. In $(1 + w)$ units of time, z_i^{m-1} is sent to PE D and is converted to x_i, which is then sent to PE C. At that time, z_{i-1}^{m-1} will be computed in PE C using x_i. From the above

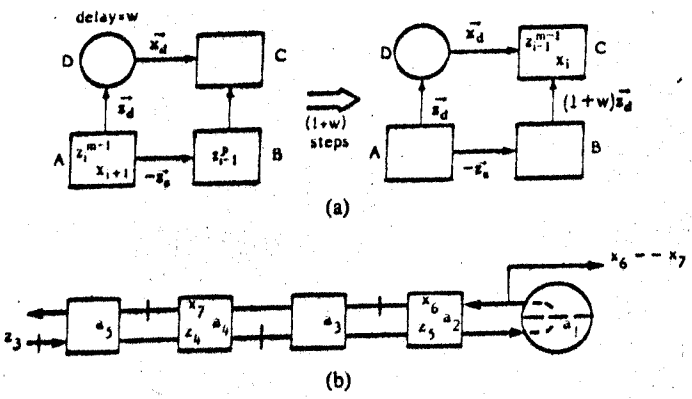

Fig. 6. Systolic array for deconvolution. (a) Relationships of data flow. (b) Systolic array for $n = 5, m = 4$ after six clock cycles.

discussion, we have another systolic processing equation

$$\vec{z}_d + \vec{x}_d = -\vec{z}_s + (1 + w)\vec{z}_d \Rightarrow \vec{x}_d = w\vec{z}_d - \vec{z}_s. \quad (5.15)$$

Equations (5.14) and (5.15) must be included in the optimization. Note that X is a one-dimensional vector, and there is one spatial-distribution parameter $\vec{x}_s$, although the subscript-access function is a function of i and k. By letting $w = 2, |\vec{a}_d| = 0, |\vec{a}_s| = -1$, and observing that $t_{kx} = t_{ka}$, the other parameters can be solved: $t_{ka} = -3/2, t_i = 3, |\vec{z}_d| = -2/3, |\vec{z}_s| = 2, |\vec{x}_d| = 2/3$ and $|\vec{x}_s| = -2$. These parameters satisfy the systolic processing equations, but the feedback X cannot be generated at the right time from Z. If $t_i = -3$, the values of the other parameters are $t_k = -3/2, |\vec{z}_d| = 2/3, |\vec{z}_s| = 2, |\vec{x}_d| = -2/3, |\vec{x}_s| = -2$. For $m = 4, n = 5$, the completion time $T = (m - 1 + w)t_k + (n - 1)|t_i|$ and #PE $\times T^2$ are minimized. It should be noted that the velocities of data flows are averaged over three clock cycles. By first inserting buffers for the Z inputs, the positions of buffers for the X inputs can be decided. A feasible assignment of buffers is shown in Fig. 6(b).

C. Generalizations on the Recurrence Formula

In this section, extended forms of the general recurrence formula [see (2.9)] are discussed. The systolic processing equations presented in Theorem 1 have to be modified.

1) Indexes in Subscript-Access Functions: In the original recurrence formula (2.9), x is a function of i and k, and y is a function of k and j. In general, x and y are functions of i, j, and k. The general recurrence formula can be expressed as

$$z^k_{i,j} = f[z^{k-1}_{i,j}, x(i, j, k), y(i, j, k)]. \quad (5.16)$$

Systolic processing equations (3.9)–(3.12) have to be changed

$$t_i\vec{z}_d + \vec{z}_{is} = t_i\vec{x}_d + \vec{x}_{is} \quad (5.17)$$

$$t_i\vec{z}_d + \vec{z}_{is} = t_i\vec{y}_d + \vec{y}_{is} \quad (5.18)$$

$$t_j\vec{z}_d + \vec{z}_{js} = t_j\vec{x}_d + \vec{x}_{js} \quad (5.19)$$

$$t_j\vec{z}_d + \vec{z}_{js} = t_j\vec{y}_d + \vec{y}_{js}. \quad (5.20)$$

The proof of (5.17) and (5.18) is similar to the proof shown in Theorem 1 except that an additional PE is needed where $Y_{y(i+1,j,k)}$ is stored (Fig. 2). $Y_{y(i+1,j,k)}$ is moved to PE A in t_i units of time. Similar arguments apply to (5.19) and (5.20). If a subscript-access function does not involve a particular index, the corresponding data distribution parameter is zero.

As an example, the recurrence for one-dimensional tuple comparison: c^0_i = TRUE, $c^k_i = c^{k-1}_i \wedge (a_{i,k} = b_{i,k}), 1 \le i \le n, 1 \le k \le m$, results in the following systolic processing equations:

$$t_{ka}\vec{a}_d + \vec{a}_{ks} = t_{ka}\vec{c}_d \quad (5.21)$$

$$t_{kb}\vec{b}_d + \vec{b}_{ks} = t_{kb}\vec{c}_d \quad (5.22)$$

$$t_i\vec{c}_d + \vec{c}_s = t_i\vec{a}_d + \vec{a}_{is} \quad (5.23)$$

$$t_i\vec{c}_d + \vec{c}_s = t_i\vec{b}_d + \vec{b}_{is}. \quad (5.24)$$

2) Coefficients of Indexes: The coefficients of i, j, and k used in the subscript-access functions x and y are 1 (regardless of sign). This means that all elements of X and Y are used in the course of computation of the recurrence. When these coefficients are greater than 1, some items in the inputs will be skipped. As an example, suppose we have the following recurrence for FIR filtering:

$$y^0_i = 0 \qquad 1 \le i \le n$$

$$y^k_i = y^{k-1}_i + a_{2k}x_{i+3k-1}$$

$$1 \le i \le n, 1 \le k \le m, x_j = 0 \quad \text{for } j > n. \quad (5.25)$$

In this case, every other item of A and every other two items of X will be skipped.

Consider the case in which the coefficients of k can be any integer (regardless of sign). The general form of the recurrence formula is

$$z^k_{i,j} = f[z^{k-1}_{i,j}, x(i, uk), y(vk, j)]$$

$$u, v \text{ are positive integers.} \quad (5.26)$$

In this case, systolic processing equations (3.7) and (3.8) have to be modified

$$t_{kx}\vec{x}_d + u\vec{x}_{ks} = t_{kx}\vec{z}_d \quad (5.27)$$

$$t_{ky}\vec{y}_d + v\vec{y}_{ks} = t_{ky}\vec{z}_d. \quad (5.28)$$

The other four systolic processing equations remain unchanged. The proof for these is very similar to that of Theorem 1. The data distribution parameters are augmented in this case because nonadjacent data are used. Similar changes have to be made on the other four systolic processing equations when the coefficients of i and j in the subscript-access functions are integers not equal to 1.

For the recurrence given in (5.25), systolic processing equations (5.2) and (5.3) have to be changed to (5.27) and (5.28) with $u = 2, v = 3$. Solving these with (5.4) and (5.5) while assuming $t_k = t_i = 1$, we obtain $|\vec{y}_d| = 1, |\vec{y}_s| = -1$, $|\vec{a}_d| = 0, |\vec{x}_d| = -1/2$, and $|\vec{a}_s| = |\vec{x}_s| = 1/2$. The interesting point is that the separation between the a_i's is 1/2 unit, which implies that only half of the a_i's reside in PE's. The other half do not have to be loaded since they are never used. The throughput of the system remains unchanged, that is, one output per clock cycle.

VI. Conclusion

Conventional approaches in mapping algorithms onto systolic arrays are usually done in an ad hoc fashion. The

resulting design represents a feasible, but not necessarily optimal, solution. In this paper, we propose a systematic methodology for the optimal mapping of algorithms that are represented as linear recurrences onto systolic arrays. The characteristics of systolic arrays are parameterized by the velocities of data flows, spatial distributions and periods of computation. The design problem is formulated into an optimization problem with objective function constraint equations that are functions of the defined parameters. An efficient order of searching the solution space is also proposed.

Numerous examples illustrating the methodology are shown. Cases including feedbacks and general recurrence formulas are discussed. Although the systolic-processing equations defined does not exhaust all possible situations, the theory developed can guide the designers in obtaining the necessary equations. For example, the mapping of recurrences onto a fixed architecture, such as the wavefront array processors, essentially restricts the search space of the optimization problem. Design requirements, such as hardware and I/O constraints, can also be included in the optimization. The art of designing systolic arrays is, therefore, reduced to a systematic methodology.

ACKNOWLEDGMENT

The authors are indebted to Prof. P. S. Xia and Prof. C. D. Han for their helpful comments and discussions. Thanks are also due to Prof. K. Hwang and Prof. H. T. Kung for providing valuable information.

REFERENCES

[1] P. R. Cappello and K. Steiglitz, "Unifying VLSI array designs with geometric transformations," in *Proc. Int. Conf. Parallel Processing*, 1983, pp. 448–457.

[2] H. D. Cheng, W. C. Lin, and K. S. Fu, "Space–time domain expansion Approach to VLSI and its application to hierarchical scene matching," in *Proc. 8th Int. Conf. Pattern Recognition*, Montreal, P.Q., Canada, Aug. 1984.

[3] A. L. Fisher, "Systolic algorithms for running order statistics in signal and image processing," in *VLSI Systems and Computations*, H. T. Kung *et al.*, Eds. Rockville, MD: Computer Science Press, Oct. 1981, pp. 265–271.

[4] J. A. B. Fortes, "Algorithm transformations for parallel processing and VLSI architecture design," Ph.D. dissertation, Univ. Southern California, Los Angeles, CA, Dec. 1983.

[5] E. Horowitz, "VLSI architecture for matrix computations," in *Proc. Int. Conf. Parallel Processing*, Aug. 1979, pp. 124–127.

[6] K. Hwang and Y.-H. Cheng, "VLSI computing structure for solving large scale linear system of equations," in *Proc. Int. Conf. Parallel Processing*, Aug. 1980, pp. 217–227.

[7] ——, "Partitioned matrix algorithms for VLSI arithmetic systems," *IEEE Trans. Comput.*, vol. C-31, pp. 1215–1224, Dec. 1982.

[8] L. Johnsson and D. Cohen, "A mathematical approach to modelling the flow of data and control in computational networks," in *VLSI Systems and Computations*, H. T. Kung *et al.*, Eds. Rockville, MD: Computer Science Press, Oct. 1981, pp. 213–225.

[9] H. T. Kung, "Highly concurrent systems," in *Introduction to VLSI System*, C. A. Mead and L. A. Conway, Eds. Reading, MA: Addison-Wesley, 1980.

[10] H. T. Kung, L. M. Ruance, and D. W. L. Yen, "A two-level pipelined systolic array for convolutions," in *VLSI Systems and Computations*, H. T. Kung *et al.*, Eds. Rockville, MD: Computer Science Press, Oct. 1981, pp. 255–264.

[11] H. T. Kung, "Why systolic architecture," *Computer*, pp. 37–46, Jan. 1982.

[12] S. Y. Kung, K. S. Arun, R. J. Gal-Ezer, and D. V. B. Rao, "Wavefront array processor: Language, architecture, and applications," *IEEE Trans. Comput.*, vol. C-31, pp. 1054–1066, Nov. 1982.

[13] G.-J. Li, "Array pipelining algorithms and pipelined array processors," M. Sc. thesis, Inst. Comput. Technol., Chinese Acad. Sci., Beijing, 1981.

[14] G.-J. Li and B. W. Wah, "The design of optimal systolic algorithms," in *Proc. Comput. Software Appl. Conf.*, 1983, pp. 310–319.

[15] D. I. Moldovan, "On the analysis and synthesis of VLSI algorithms," *IEEE Trans. Comput.*, vol. C-31, pp. 1121–1126, Nov. 1982.

[16] I. V. Ramakrishnan, D. S. Fussell, and A. Silberschatz, "On mapping homogeneous graphs on a linear array-processor model," in *Proc. Int. Conf. Parallel Processing*, 1983, pp. 440–447.

[17] L. Snyder, "Introduction to the configurable, highly parallel computer," *IEEE Computer*, vol. 15, pp. 47–56, Jan. 1982.

[18] V. Weiser and A. Davis, "A wavefront notion tool for VLSI array design," in *VLSI Systems and Computations*, H. T. Kung *et al.*, Eds. Rockville, MD: Computer Science Press, Oct. 1981, pp. 226–234.

[19] M. C. Chen and C. A. Mead, "Concurrent algorithms as space-time recursion equations," in *Proc. USC Workshop VLSI Modern Signal Processing*, Los Angeles, CA, Nov. 1982, pp. 31–52.

[20] J. A. B. Fortes, K. S. Fu, and B. W. Wah, "Systematic approaches to the design of algorithmically specified systolic arrays, Purdue Univ., W. Lafayette, IN, Tech. Rep. TR-EE-84-39, Sept. 1984.

[21] D. Gannon, "Pipelining array computations for MIMD parallelism: A functional specification," in *Proc. 1982 Int. Conf. Parallel Processing*, 1982, pp. 284–286.

[22] J. M. Jover and T. Kailath, "Design framework for systolic-type arrays," in *Proc. ICASSP*, 1984, pp. 8.5.1–8.5.4.

[23] R. H. Kuhn, "Optimization and interconnection complexity for parallel processors, single stage networks and decision trees," Ph.D. dissertation, Dep. Comput. Sci., Univ. Illinois, Urbana-Champaign, Rep. 80-1009, 1980.

[24] H. T. Kung and W. T. Lin, "An algebra for VLSI algorithm design," in *Proc. Conf. Elliptic Problem Solvers*, Monterey, CA, 1983.

[25] S. Y. Kung, "On supercomputing with systolic/wavefront array processors," *Proc. IEEE*, vol. 72, no. 7, pp. 867–884, 1984.

[26] M. Lam and J. Mostow, "A transformational model of VLSI systolic design," in *Proc. IFIP 6th Int. Symp. Comput. Hardware Descriptive Lang. Appl.*, Carnegie-Mellon Univ., Pittsburgh, PA, May 1983.

[27] C. E. Leiserson, F. M. Rose, and J. B. Saxe, "Optimizing synchronous circuitry by retiming," in *Proc. 3rd Caltech Conf. Very Large Scale Integration*, R. Bryant, Ed. Rockville, MD: Computer Science Press, 1983, pp. 87–116.

[28] W. L. Miranker and A. Winkler, "Space-time representations of computational structures," *Computing*, vol. 32, pp. 93–114, 1984.

[29] P. Quinton, "Automatic synthesis of systolic arrays from uniform recurrent equations," in *Proc. 11th Annu. Symp. Comput. Arch.*, 1984, pp. 208–214.

[30] D. A. Schwartz and T. P. Barnwell III, "A graph theoretic technique for the generation of systolic implementations for shift-invariant flow graphs," in *Proc. ICASSP*, 1984, pp. 8.3.1–8.3.4.

Guo-jie Li (S'83) graduated from Peking University, Beijing, China, in 1968 and received the M.S. degree in computer science and engineering from the University of Science and Technology of China and the Institute of Computing Technology, Chinese Academy of Science in 1981.

Currently, he is working towards the Ph.D. degree in electrical engineering at Purdue University, West Lafayette, IN. His research interests include parallel processing, computer architecture, and artificial intelligence.

Benjamin W. Wah (S'74–M'79) received the B.S. and M.S. degrees in electrical engineering and computer science from Columbia University, New York, NY, in 1974 and 1975, and the M.S. degree in computer science, and the Ph.D. degree in engineering both from the University of California, Berkeley, CA, in 1976 and 1979, respectively.

He is now an Associate Professor in the School of Electrical Engineering, Purdue University, West Lafayette, IN. His current research interests include parallel computer architecture, distributed databases, and theory of algorithms.

Dr. Wah has been a Distinguished Visitor of the IEEE Computer Society since 1983.