# OPTIMAL DESIGN OF SYSTOLIC ARRAYS FOR IMAGE PROCESSING

Guo-Jie Li and Benjamin W. Wah
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

## Abstract

Conventional design of systolic algorithms is based on the mapping of an algorithm onto an interconnection of processor elements in a VLSI chip. This mapping is done in an ad hoc manner, and the resulting configuration usually represents a feasible and not an optimal design. In this paper, we characterize systolic algorithms by three parameters: the velocity of data flow, the spatial distribution of data, and the period of computation; and then we formulate the design into a non-linear integer programming problem. Some examples in image processing applications, which include matrix multiplication, finite impulse response filtering, discrete Fourier transform and deconvolution are given.

**Keywords and Phrases:** data distribution, data flow, non-linear programming, parameter method, period, recurrence equations, systolic algorithms, velocity.

## 1. Introduction

The evolution in Very-Large-Scale-Integration (VLSI) technology has had a great impact on computer architecture [1]. Specialized algorithms can be implemented on a VLSI chip to exploit a greater potential of pipelining with array processing [2,4,19]. This type of array processor has been referred to as a *systolic processor* [1] which may be characterized as follows:

1) It consists of many processing elements (PEs) interconnected in a mesh fashion.
2) It may possess tens or hundreds of pipelines which extend in different directions. Several data items flowing along different pipes with the same or different rates may meet and interact. 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 is referred to as *systolic processing*.
3) The algorithms suitable for execution on a systolic processor can be written as recurrence processes.
4) A systolic processor can be implemented in VLSI.

Two different approaches to designing VLSI algorithms have been investigated. One approach is the synchronous systolic algorithms of H. T. Kung [1-6, 11] which maps high-level computations into special purpose VLSI chips. Various designs using this approach have been proposed [7, 8, 10]. The disadvantage of this approach is that it is inflexible because each algorithm has to be implemented differently. However, the most efficient implementation can be obtained. Studies have been made to design reconfigurable interconnections that are more flexible [20]. Another approach in designing VLSI algorithms is based on the concept of a computational wavefront which maps different VLSI algorithms into a fixed computer architecture. In order to tailor the algorithms to the architecture, special controls have to be included in the architecture. This is exemplified by the Matrix Data Flow Language of S. Y. Kung et al. [12,17]. This approach is flexible for the algorithms, but inflexible for the architecture. In this paper, we study the optimal design of systolic algorithms. This is useful if the algorithms are implemented directly using the first approach. Further, it provides a benchmark for comparing designs obtained by the second approach.

The design of specialized systolic architectures is very ad hoc at this time. Although research has been carried out with respect to the specification, analysis and synthesis of systolic algorithms [10,16,18], no formal method for designing optimal systolic architectures is available. In this paper, we characterize systolic algorithms by three parameters: velocity of data flow, spatial distribution of data, and period of computation; and we then find the relationship among these parameters (Section 3). The design is then formulated into a non-linear integer program. The objective function we have used is the completion time of the algorithm. By modifying the objective function, hardware complexity can also be taken into consideration. Examples illustrating the method are also shown (Section 4).

## 2. Characteristics of Recurrence Processes

As discussed previously, systolic processors are suitable for implementing recurrence processes. The following are some examples of recurrence formulae, each of which specifies a dynamic procedure of systolic processing [21].

1) Finite Impulse Response (FIR) filtering $(1 \leq i \leq n)$

$$y_i^0 = 0$$

$$y_i^k = y_i^{k-1} + a_k x_{i+k-1} \quad (1 \leq k \leq n, x_j = 0 \text{ for } j > n) \qquad (1)$$

2) Two-dimensional matrix multiplication $(1 \leq i,j \leq n)$

$$c_{i,j}^0 = 0$$

$$c_{i,j}^k = c_{i,j}^{k-1} + a_{i,k} b_{k,j} \quad (1 \leq k \leq n) \qquad (2)$$

3) Discrete Fourier Transform (DFT) $(0 \leq i \leq n-1)$

$$y_i^0 = 0$$

$$y_i^k = y_i^{k-1} \cdot w^i + x_{n-k} \quad (0 \leq k \leq n-1) \qquad (3)$$

This project was supported partially by a David Ross Grant from the Purdue Research Foundation and the National Science Foundation Grant ECS80-16580.

Workshop on Computer Architecture for Pattern Analysis and Image Database Management, Oct. 1983.

4) Polynomial multiplication ($0 \leq i \leq 2n-2$)

$c_i^0 = 0$

$c_i^k = c_i^{k-1} + a_{k-1} b_{i-k+1}$ ($1 \leq k \leq n$, $b_j = 0$ for $j < 0$ or $j \geq n$) (4)

5) Deconvolution ($1 \leq i \leq m$)

$y_i^0 = b_i$

$y_i^k = y_i^{k-1} - a_{m-k+1} x_{i-m+k}$ ($1 \leq k \leq m-1$) (5)

$x_i = y_i^{m-1} / a_1$ ($x_i = 0$ for $i \leq 0$)

6) Triangular matrix inversion ($1 \leq i \leq n, i \leq j \leq n$)

$w_{i,j}^{j+1} = 0$

$w_{i,j}^k = w_{i,j}^{k+1} - u_{i,k} v_{k,j}$ ($i < k \leq j$) (6)

$v_{i,j} = w_{i,j}^{i+1} / u_{i,i}$ ($i < j$)

$v_{i,i} = 1/u_{i,i}$

7) Two-dimensional comparison ($1 \leq i,j \leq n$)

$c_{i,j}^0 = \text{TRUE}$

$c_{i,j}^k = c_{i,j}^{k-1} \wedge (a_{i,k} = b_{j,k})$ ($1 \leq k \leq n$) (7)

Eq's 1, 3, 4, and 5 are one-dimensional linear recurrences; Eq's 2, 6 and 7 represent two-dimensional linear recurrences.

In general, linear recurrences can be expressed in the following form,

$$z^k(i,j) = f[z^{k-\delta}(i,j), x(i,k), y(k,j)] \quad (\delta = 1 \text{ or } -1) \quad (8)$$

where f is a function to be executed by a PE. $z^{k-\delta}(i,j)$ is the intermediate result of the last step of an iterative computation and is iteration dependent. x(i,k) and y(k,j) are functions to define the indices of x and y and are iteration independent. The functions x,y and z are linear in linear recurrences.

When $\delta = -1$, the recurrence is called a *forward recurrence* and $z^k$ is defined in terms $z^{k+1}$,... 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 the others are backward recurrences. These two formulations are identical; that is, given one representation it can be converted into the other. Further, for each formulation, the evaluation order of the terms can be reversed. Therefore, there are four ways of representing a recurrence process. As an example, the following three recurrences are equivalent ways of representing the matrix multiplication problem in addition to Eq. 2.

$c_{i,j}^0 = 0$; $c_{i,j}^k = c_{i,j}^{k-1} + a_{i,n-k+1} b_{n-k+1,j}$ ($k = 1,\ldots,n$) (9)

$c_{i,j}^{n+1} = 0$; $c_{i,j}^k = c_{i,j}^{k+1} + a_{i,k} b_{k,j}$ ($k = n,\ldots,1$) (10)

$c_{i,j}^{n+1} = 0$; $c_{i,j}^k = c_{i,j}^{k+1} + a_{i,n-k+1} b_{n-k+1,j}$ ($k = n,\ldots,1$) (11)

In designing systolic algorithms, only the evaluation order of terms is important. Whether a recurrence is written in the forward or backward form is insignificant as long as the terms are evaluated in the same order. For example, Eq's 9 and 10 are equivalent for the design of systolic architectures. Similarly, Eq's 2 and 11 are equivalent. The complexities of the resulting design may depend on the order of the evaluation.

An observation from the recurrences discussed above is that the coefficients of i, j, k are 1 or -1 for x, y and z. This is the assumption taken in the discussion below. However, it will be extended in Section 5 so that the coefficient can be any integer.

The key to systolic processing is that the appropriate data must be in the appropriate place at the time they have to be processed. In other words, both timing and data distribution are very important and the relationship between them must be identified. In the general recurrence formula the index k couples time with space. 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, $c_{i,j}^k$ can be computed provided that $c_{i,j}^{k-1}$, $a_{i,k}$ and $b_{k,j}$ arrive at the same PE simultaneously. In other words, the k-th step of computing $c_{i,j}$ requires the intermediate result of the (k-1)-st step of computing $c_{i,j}$, an entry of matrix A on row i and column k, and an entry of matrix B on row k and column j. In the next section we define a set of parameters and prove a theorem to characterize the relationship between time and space with respect to systolic processing.

## 3. The Parameter Method for Designing Systolic Algorithms

We discuss the parameter method with respect to the matrix multiplication problem which can be represented in a recurrence $c^k(i,j) = f[c^{k-1}(i,j), a(i,k), b(k,j)]$ (Eq. 2). A number of systolic algorithms for matrix multiplication have been discussed [7,8,12,15]. The scheme proposed below is designed with the parameter method. This is the optimal systolic algorithm as far as completion time is concerned. In this scheme, as depicted in Figure 1, the data flow has three different direc-

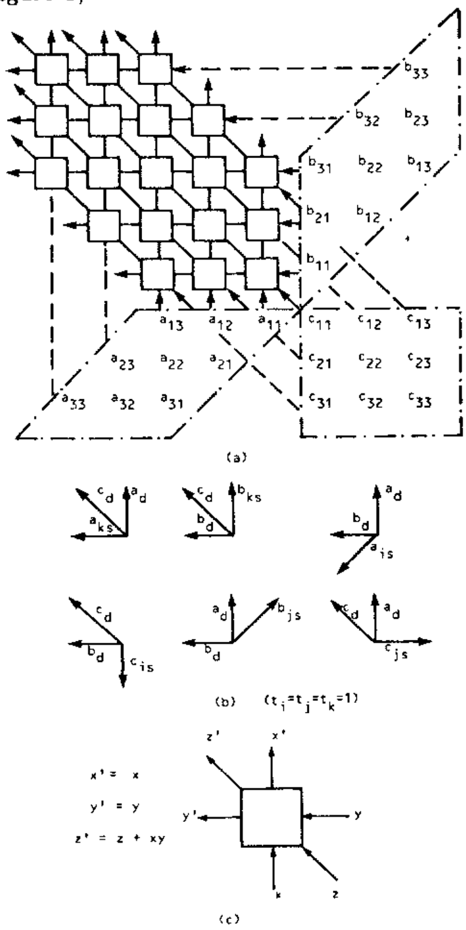

Figure 1. The systolic processor for 2-dimensional matrix multiplication.

tions. The rhomboidal data block A moves towards the north, the rhomboidal data block B moves towards the west and the square data block C moves towards the north-west. 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 PEs along their own pipes synchronously.

The ways that data are fed into the systolic processor are related and can be characterized by three parameters: velocity, space and time. If these parameters are known, the systolic algorithm can be determined completely.

Let us define the distance between two directly-connected neighboring PEs as unity. We define a clock cycle as a unit of time during which one iterative operation can be computed and data items can advance into neighboring PEs or buffers. When there are buffers between two adjacent PEs, the buffers are spaced equally between them. A datum can advance one buffer in a clock cycle. The first two parameters defined below are vectors[1] which possess both magnitude and direction. The third parameter is a scalar.

*Parameter 1. Velocity of data flow*

We call the directional distance passed by datum x during a clock cycle as the velocity of x and denote it by $\mathbf{x_d}$. If there are m-1 buffers between two neighboring PEs in the pipelining direction of x, then $|\mathbf{x_d}| = 1/m$ since the distance between two adjacent PEs is one unit. Therefore $|\mathbf{x_d}|$ is a rational number less than or equal to 1.

*Parameter 2. Data distribution*

If a group of data x is a two-dimensional array, then the directional distance between x(i,j) and x(i+1,j) when data are fed into the systolic processor is referred to as the *row vector of x* and is denoted by $\mathbf{x}_{\mathbf{i}\bullet}$; the directional distance between x(i,j) and x(i,j+1) is referred to as the *column vector of x* and is denoted by $\mathbf{x}_{\mathbf{j}\bullet}$. For a one-dimensional array, $\mathbf{x}_\bullet$ is the *item vector*. Referring to the general recurrence formula (Eq. 8), index k exists in the access of array x. We define $\mathbf{x}_{\mathbf{k}\bullet}$ as the row/column vector depending on whether k is the row/column index. $\mathbf{x}_{\mathbf{k}\bullet}$ is the row vector between x(k,j) and x(k+1,j) or is the column vector between x(i,k) and x(i,k+1). For a one-dimensional array, the item vector between x(k) and x(k+1) is $\mathbf{x}_{\mathbf{k}\bullet} = \mathbf{x}_\bullet$. The direction of data distribution is defined along a subscript-increasing direction. Notice that data distributions are not changed during systolic processing; hence, $\mathbf{x}_{\mathbf{i}\bullet}$ and $\mathbf{x}_{\mathbf{j}\bullet}$ are iteration independent. The magnitude of all data distributions must be a non-zero rational number due to pipelining.

*Parameter 3. Period*

Suppose the time that a computation is performed is $\tau(\cdot)$. We define the *periods of i and j* as follows:

$$t_i = \tau(z^k(i+1,j)) - \tau(z^k(i,j)) \tag{12}$$

$$t_j = \tau(z^k(i,j+1)) - \tau(z^k(i,j)) \tag{13}$$

In computing $z^k(i,j) = f[z^{k-1}(i,j), x(i,k), y(k,j)]$ and $z^{k+1}(i,j) = f[z^k(i,j), x(i,k+1), y(k+1,j)]$, items x(i,k) and x(i,k+1) are accessed sequentially. Similarly, y(k,j) and y(k+1,j) are accessed in sequence. The time difference between the access of two consecutive items of array x or y in increasing order of indices in a backward recurrence is defined as the *period of x or y with respect to k* and is denoted by $t_{kx}$ or $t_{ky}$. For accesses in reverse order, the corresponding period is negated. Referring to the recurrence for deconvolution (Eq. 5), the access of array a is in decreasing order of indices ($a_j$ is accessed before $a_{j-1}$). The period $t_{ka}$ is, therefore, negative. On the other hand, array x is accessed in increasing order of indices. Therefore, $t_{kx}$ is positive. For forward recurrences, the signs of $t_{kx}$ and $t_{ky}$ are negated. For example, for the recurrence in Eq. 11, a and b are accessed with decreasing functions of indices. But since this is a forward recurrence, $t_{ka} = t_{kb}$ are positive. We also define the *period of iterative computation* as:

$$t_k = \tau(z^{k+1}(i,j)) - \tau(z^k(i,j)) \tag{14}$$

From this definition, $t_k$ is positive for backward recurrences and negative otherwise. Since the data needed in the computation of $z^{k+1}(i,j)$ must be assembled in time $t_k$, it is true that,

$$|t_{kx}| = |t_{ky}| = |t_k|. \tag{15}$$

All the periods are non-zero integers due to the integrality of clock cycles. In order for the periods to be zero, there must be a bus which can broadcast data items. This is unsuitable for systolic processing and is excluded from our consideration.

There are a total of 13 parameters for two dimensional linear recurrences, three of which are for the velocities of data flow, $\mathbf{x_d}, \mathbf{y_d}, \mathbf{z_d}$, six are for data distributions, $\mathbf{x}_{\mathbf{i}\bullet}, \mathbf{x}_{\mathbf{j}\bullet}, \mathbf{y}_{\mathbf{i}\bullet}, \mathbf{y}_{\mathbf{j}\bullet}, \mathbf{z}_{\mathbf{i}\bullet}, \mathbf{z}_{\mathbf{j}\bullet}$, and four are for the periods, $t_{kx}, t_{ky}, t_i, t_j$. For one dimensional problems, only 8 parameters exist. The following theorem states the relationship among these parameters.

**THEOREM OF SYSTOLIC PROCESSING:** Suppose recurrence computation $z^k(i,j) = f[z^{k-1}(i,j), x(i,k), y(k,j)]$ is performed in a systolic processor, then the velocities, data distributions and periods must satisfy the following vector equations:

$$t_{kx}\mathbf{x_d} + \mathbf{x}_{\mathbf{k}\bullet} = t_{kx}\mathbf{z_d} \tag{16}$$

$$t_{ky}\mathbf{y_d} + \mathbf{y}_{\mathbf{k}\bullet} = t_{ky}\mathbf{z_d} \tag{17}$$

$$t_i\mathbf{x_d} + \mathbf{x}_{\mathbf{i}\bullet} = t_i\mathbf{y_d} \tag{18}$$

$$t_i\mathbf{z_d} + \mathbf{z}_{\mathbf{i}\bullet} = t_i\mathbf{y_d} \tag{19}$$

$$t_j\mathbf{y_d} + \mathbf{y}_{\mathbf{j}\bullet} = t_j\mathbf{x_d} \tag{20}$$

$$t_j\mathbf{z_d} + \mathbf{z}_{\mathbf{j}\bullet} = t_j\mathbf{x_d} \tag{21}$$

*Proof:* Without loss of generality, we can use orthogonally connected PEs with diagonal connections in our proof. We also assume that $t_{kx} = t_{ky} = t_k > 0$. In Figure 2, A,B,C,D represent four PEs 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(i,k+1) is in PE B, y(k+1,j) is in PE D (Figure 2a). Thus, by definition, $\vec{CB}$ indicates $\mathbf{x}_{\mathbf{k}\bullet}$, $\vec{CD}$ indicates $\mathbf{y}_{\mathbf{k}\bullet}$. $\mathbf{x}_{\mathbf{k}\bullet}$ can be either $\mathbf{x}_{\mathbf{i}\bullet}$ or $\mathbf{x}_{\mathbf{j}\bullet}$ depending on the subscript involved, and similarly for $\mathbf{y}_{\mathbf{k}\bullet}$. In accordance with the characteristics of systolic processing, the operands needed in the next computational step must arrive at the same PE and at the same clock cycle after $t_k$ units of time. $z^k(i,j)$, x(i,k+1) and y(k+1,j) will arrive at PE A which computes $z^{k+1}(i,j) = f[z^k(i,j), x(i,k+1), y(k+1,j)]$ as shown in Fig. 2(b). Therefore, we have $\vec{CA} = t_k\mathbf{z_d}$, $\vec{BA} = t_k\mathbf{x_d}$, $\vec{DA} = t_k\mathbf{y_d}$. According to the principle of vector composition $\vec{BA} + \vec{CB} = \vec{CA}$, $\vec{DA} + \vec{CD} = \vec{CA}$, so Eq's 16 and 17 are proved. The other combinations in which $t_{kx}$ and

[1] All vectors in this paper are indicated with symbols in bold letters. The magnitude (absolute value) of a vector is enclosed in vertical bars. A negative value indicates that the direction of the vector is reversed.

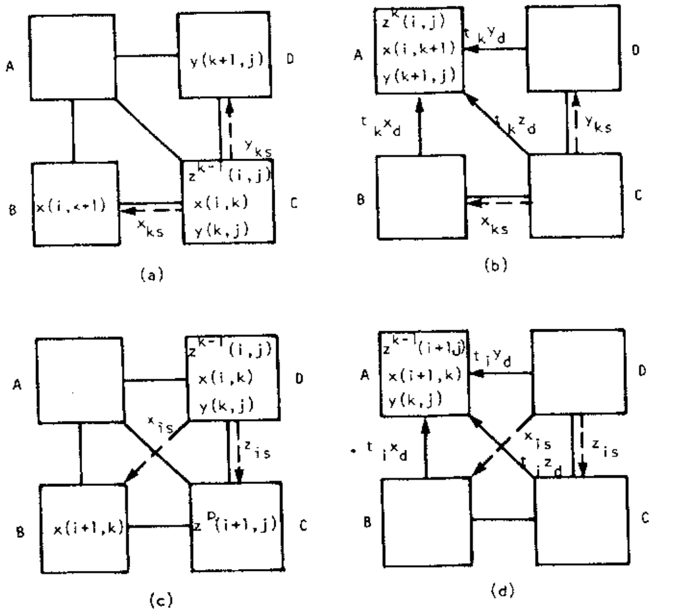

Figure 2. The proof of theorem of systolic processing.

$t_{ky}$ have different signs or are negative can be proved similarly.

In Fig. 2(c) suppose that while PE D is computing $z^k(i,j) = f[z^{k-1}(i,j), x(i,k), y(k,j)]$, $z^p(i+1,j)$, $p<k$, is being computed in PE C, and $x(i+1,k)$ is residing in PE B. So $\vec{DC} = \mathbf{z}_{is}$, $\vec{DB} = \mathbf{x}_{is}$. In accordance with the characteristics of systolic processing, after $t_i$ units of time, k-p steps of the iterative computation are performed, and $z^{k-1}(i+1,j)$, $x(i+1,k)$, $y(k,j)$ 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, $\vec{BA} = t_i\mathbf{x}_d$, $\vec{CA} = t_i\mathbf{z}_d$, $DA = t_i\mathbf{y}_d$. According to the principle of vector composition, $\vec{BA} + \vec{DB} = \vec{DA}$, $\vec{CA} + \vec{DC} = \vec{DA}$, thus Eq's 18 and 19 are proved. Eq's 20 and 21 can be proved similarly. For negative periods, the theorem can be proved using the same method. □

For one-dimensional problems, Eq's 20 and 21 do not exist. The above theorem has been written for the case such that x is independent of i and y is independent of j. This will be extended in Section 5 so that x and y are functions of both i and j.

The above theorem shows the fundamental space-time relationship in systolic processing. In order to design the optimal systolic algorithm, the design can be formulated into an optimization problem using Eq's 16-21 as constraints. The objective function depends on the design criterion and is a function of the characteristics of systolic algorithms and the size of the problem to be solved. Taking the minimum computation time criterion, the objective function may take the form $g(t_k, t_i, t_j, n_1, n_2)$ where $n_1$ and $n_2$ denote the problem size. Since the constraints are vector equations, and non-linear functions of sine and cosine have to be used when these equations are transformed into scalar equations, the design of optimal systolic algorithms is formulated as a non-linear integer program.

*Minimize* $g(t_k,t_i,t_j,n_1,n_2)$ (22)

*Subject to*
Equations 16-21

*and*

$$\begin{array}{lll} |t_k| \geq 1 & |t_i| \geq 1 & |t_j| \geq 1 \\ |\mathbf{x}_d| \leq 1 & |\mathbf{y}_d| \leq 1 & |\mathbf{z}_d| \leq 1 \\ |\mathbf{x}_{ks}| \neq 0 & |\mathbf{y}_{ks}| \neq 0 & |\mathbf{x}_{is}| \neq 0 \\ |\mathbf{y}_{js}| \neq 0 & |\mathbf{z}_{is}| \neq 0 & |\mathbf{z}_{js}| \neq 0 \\ |t_k|\,|\mathbf{z}_d| = k_1 & |t_i|\,|\mathbf{y}_d| = k_2 & |t_j|\,|\mathbf{z}_d| = k_3 \end{array}$$

All parameters are rational numbers of form $I_1/I_2$, $I_1$, $I_2$: integers; $t_k$, $t_i$, $t_j$, $k_1$, $k_2$, $k_3$ are integers.

The reason for the last three constraints, that is, $|t_k|\,|\mathbf{z}_d|$, $|t_i|\,|\mathbf{y}_d|$ and $|t_j|\,|\mathbf{z}_d|$ are integers is that these terms represent the distance traversed for the next computation (they exist on the RHS of Eq's 16-21). Since a computation must be performed in a PE, the distance traversed must coincide with the locations of PEs.

The objective function, based on the minimum computation time, can be obtained by analyzing the data dependence according to the recurrence formula. First we need to determine the item of the result that will be completed last. Then we can derive the total computation time in terms of $t_k$, $t_i$, $t_j$ and the problem size to be solved. For example, the objective function of multiplying two n by n matrices is:

$$n|t_k| + (n-1)|t_i| + (n-1)|t_j| \qquad (23)$$

since from Eq. 2, it needs $n|t_k|$ 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 this example, no constants have to be pre-loaded into the systolic processor. Therefore, Eq. 23 is the delay time of the algorithm. In some of the examples shown in Section 4, the load or drain time is non-zero, and it should be considered as a secondary objective in the optimization.

The above formulation is a complicated non-linear program which can be solved by conventional methods [22]. Since no restriction is put on the maximum values of parameters, the problem space is infinitely large. However, by examining the objective function, a preferred order of search usually can be defined. In this case, the first feasible solution that satisfies the constraints is the optimal solution.

For example, in the matrix multiplication problem, the objective function (Eq. 23) can be minimized when $|t_k|$, $|t_i|$ and $|t_j|$ are as small as possible. From earlier discussions, we recall that $t_k$, $t_i$ and $t_j$ are positive integers for the recurrence shown in Eq. 2. Therefore, the search should begin with $t_k = t_i = t_j = 1$. If no feasible solution is found, $t_k$, $t_i$ or $t_j$ is incremented by 1 and the search repeats. In this example, $t_k = t_i = t_j = 1$ results in a feasible solution satisfying the constraints of Eq. 22 which is depicted in Figure 1(b)[2]. By using these vectors, we can decide on the velocities of data flow and spatial distributions of data, and design a basic cell as shown in Figure 1(c). These cells are connected together into a mesh. In some of the cells, no computation is performed and they can be eliminated. The final VLSI structure is shown in Figure 1(a) for the multiplication of two 3 by 3 matrices. This is the fastest matrix multiplication scheme that can be completed in 3n-2 units of time with $3n^2 - 3n + 1$ cells.

The optimization problem (Eq. 22) can incorporate other design requirements such as the number of devices and I/O pins. The solution time for more complicated problems may be long because there may not be any pre-

[2] In the solution shown in Figure 1, the distance between two adjacent PEs in the northwest direction is not one. However, we can redraw the figure using hexagonal cells and the angle between two vectors in Figure 1(b) is always an integral multiple of 60°. In this case, all the adjacently connected cells are equidistant.

ferred order of search. In general, we can design an optimal systolic algorithm by solving a non-linear program. We refer to this method as the ***parameter method*** for designing systolic algorithms, or in short, the parameter method.

The procedure for designing an optimal systolic algorithm using the parameter method can be sketched as follows:

- Step 1 Write the recurrence formula for the problem to be solved.
- Step 2 Write the corresponding systolic processing equations using the theorem of systolic processing and constraints on the parameters.
- Step 3 Write the objective function based on the design requirements in terms of the systolic processing parameters and problem size.
- Step 4 Find the parameter values which minimize the objective function by solving the nonlinear program.
- Step 5 Design a basic cell for the systolic algorithm and find a possible interconnection of the cells based on the parameters obtained. Eliminate cells that do not perform any useful computations.

We illustrate in the next section the design of optimal systolic algorithms based on the above method. We compare our designs with designs proposed elsewhere and found that our designs are better (with respect to the objective function).

## 4. Examples Illustrating the Parameter Method

### (a) *FIR filtering*

FIR filtering is an important technique in signal processing [6,9,15]. It can be considered as a matrix-vector multiplication as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} = \begin{bmatrix} a_1 & \cdots & a_m & & \cdots & 0 \\ & & & & & \cdot \\ & a_1 & \cdots & a_m & & \cdot \\ & & & & & \cdot \\ \cdot & & & & & \\ \cdot & & & & & a_m \\ \cdot & & & & & \\ 0 & \cdots & & & & a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \qquad n > m$$

where the matrix is a band upper triangular Toeplits matrix. The operation can be represented as a one-dimensional linear recurrence $y^k(i) = f[y^{k-1}(i), x(k,i), a(k)]$ (Eq. 1). This recurrence can be written in another form such that the evaluation order of terms is reversed ($y_i^k = y_i^{k-1} + a_{m-k+1}x_{m-k+i}$). For both recurrences, $t_k = t_{k_a} = t_{kx}$. It takes $m|t_k|$ 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 time of the $a_i$s into the PEs, is $m|t_k| + (n-1)|t_i|$. The design problem can be formulated as:

*Minimize* $\quad m|t_k| + (n-1)|t_i| \qquad (24)$

*Subject to*

$$t_k\mathbf{x_d} + \mathbf{x_s} = t_k\mathbf{y_d} \qquad (25)$$

$$t_k\mathbf{a_d} + \mathbf{a_s} = t_k\mathbf{y_d} \qquad (26)$$

$$t_i\mathbf{x_d} + \mathbf{x_s} = t_i\mathbf{a_d} \qquad (27)$$

$$t_i\mathbf{y_d} + \mathbf{y_s} = t_i\mathbf{a_d} \qquad (28)$$

$$|t_k| \geq 1 \qquad |t_i| \geq 1 \qquad (29)$$

$$|\mathbf{a_d}| \leq 1 \qquad |\mathbf{x_d}| \leq 1 \qquad |\mathbf{y_d}| \leq 1 \qquad (30)$$

$$|\mathbf{a_s}| \neq 0 \qquad |\mathbf{x_s}| \neq 0 \qquad |\mathbf{y_s}| \neq 0 \qquad (31)$$

$$|t_k|\,|\mathbf{y_d}| = k_1 \qquad |t_i|\,|\mathbf{a_d}| = k_2 \qquad (32)$$

where $t_k$, $t_i$, $k_1$, $k_2$ are integers; all the other magnitudes are rational numbers.

We show the details of solving the non-linear program below. First, $t_k$ and $t_i$ are set to 1 because they minimize the objective function. However, this results in $|\mathbf{a_s}| = |\mathbf{y_s}| = 0$ which is an infeasible solution. Next, consider $t_k = -1$ and $t_i = 1$. Substituting into Eq. 25-32 results in 4 equations with 6 unknowns. Assume that the $a_i$s are statically placed in the PEs, $|\mathbf{a_d}| = 0$. From Eq. 26, it implies $\mathbf{a_s} = -\mathbf{y_d}$. From Eq. 31, $|\mathbf{y_d}| \neq 0$; and from Eq. 30, $|\mathbf{y_d}| \leq 1$. Therefore $|\mathbf{y_d}|$ can be 1 or -1. By choosing $|\mathbf{y_d}| = 1$, the remaining four unknowns can be solved: $|\mathbf{a_s}| = -1$, $|\mathbf{x_d}| = 1/2$ $|\mathbf{x_s}| = -1/2$ $|\mathbf{y_s}| = -1$ and it results in a one-dimensional solution (all the vectors are pointing in the same direction). It should be noted that $t_k = -1$ implies $t_{k_a} = t_{k_x} = -1$. This means that recurrence formula $y_i^k = y_i^{k-1} + a_{m-k+1}x_{m-k+i}$ $(1 \leq k \leq m, i \leq i \leq n)$ is used.

From the above evaluations, there is a feasible solution that minimizes the objective solution. A little thought suggests the systolic algorithm as depicted in Figure 3. This algorithm can be completed in m+n-1 units of time assuming that $a_1, \ldots, a_m, x_1, \ldots, x_m$ are statically loaded into the pipe. Since the $x_i$s are problem-dependent, it may require m steps to preload them. The $a_i$s can be assumed to be fixed in the filter.

Another FIR systolic algorithm which requires n+2m-1 units of time to complete and no load time for the $x_i$s can be derived by choosing $t_i = 1$, $t_k > 1$ and $|\mathbf{a_d}| = 0$. This choice is reasonable because $n/m \sim 10$ and $t_i$ is the dominant factor in the objective function $g \sim m(t_k + 10t_i)$. From Eq. 25, $\mathbf{a_s} = t_k\mathbf{y_d}$ and since $|t_k|\,|\mathbf{y_d}|$ is an integer (Eq. 32), we can set $|\mathbf{a_s}| = 1$. Substituting the known values into Eq's 26 and 27, we

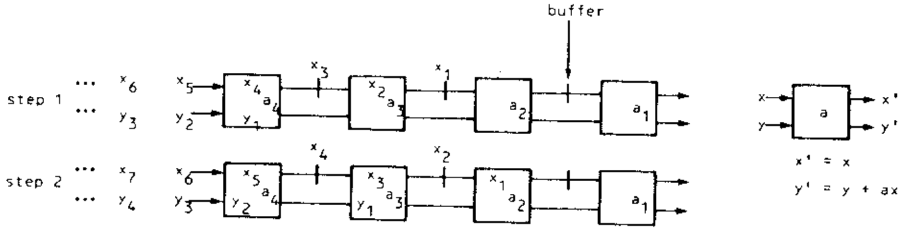

Figure 3. The systolic processor for FIR filter $(t_k = -1,\ t_i = 1)$

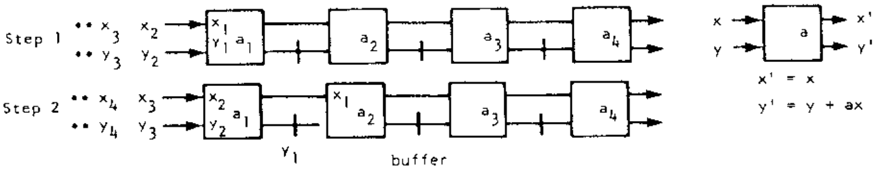

Figure 4. The systolic processor for FIR filter ($t_k = 2$, $t_i = 1$) (The cell used is the same as that of Figure 3)

obtain,

$$(t_k - 1)\mathbf{x_d} = t_k\mathbf{y_d} = \mathbf{a_s} \text{ (a one-dimensional solution)}$$

$$\Longrightarrow t_k = \frac{1}{|\mathbf{x_d}|} + 1$$

$$\Longrightarrow t_k \geq 2 \qquad \text{(since } |\mathbf{x_d}| \leq 1)$$

The above analysis implies a one-dimensional solution with all the vectors pointing in the same direction. By choosing $t_k = 2$, the remaining parameters can be solved: $|\mathbf{x_d}| = 1$, $|\mathbf{x_s}| = -1$, $|\mathbf{y_d}| = 1/2$, $|\mathbf{y_s}| = -1/2$ The corresponding systolic algorithm is depicted in Figure 4. We can assert that no faster systolic algorithm for FIR filtering exists if the filtering factors, $a_i$, are static during pipelining. When the $a_i$s can be moved, we may find faster systolic algorithms. However the number of PEs will increase dramatically.

Before leaving this example, recall that a bus structure is not involved in systolic processing. If a broadcast bus is allowed, that is, an input $x_i$ can be sent to all PEs simultaneously, then an n-point FIR filtering can be performed in m+n-1 units of time.

It is of interest to note that pattern matching problems have the same recurrence equation as FIR filtering. Therefore, pattern matching can be implemented in the same systolic scheme as depicted in Figures 3 and 4 except that each PE will carry out logical operations.

(b) *Discrete Fourier transform (DFT)*

DFT is usually regarded as a matrix-vector multiplication. In order to implement DFT as a systolic algorithm, we can consider DFT as the evaluation of polynomials, $y_i = \sum_{k=0}^{n-1} x_k w^{ik}$, $0 \leq i \leq n-1$, where $x_k$ is the coefficient and $w^i = \exp(\frac{2\pi ij}{n})$, $j = \sqrt{-1}$, is the variable. It can be represented as a recurrence $y^k(i) = f[y^{k-1}(i), w(i), x(k)]$ (Eq. 3).

The optimization problem can be formulated as:

*Minimize* $\quad n|t_k| + (n-1)|t_i| \qquad (33)$

*subject to*

$$t_k\mathbf{w_d} = t_k\mathbf{y_d} \qquad (34)$$

$$t_k\mathbf{x_d} + \mathbf{x_s} = t_k\mathbf{y_d} \qquad (35)$$

$$t_i\mathbf{w_d} + \mathbf{w_s} = t_i\mathbf{x_d} \qquad (36)$$

$$t_i\mathbf{y_d} + \mathbf{y_s} = t_i\mathbf{x_d} \qquad (37)$$

The constraints on the parameters are similar to those stated previously.

In Eq. 34, $\mathbf{w_s}$ does not appear because input w is independent of k.

In order to minimize the objective function, we set $t_k = t_i = 1$. Assume that the $w^0, w^1, w^2 \cdots w^{n-1}$ are statically loaded into a set of of linearly connected PEs as shown in Figure 5. In this case $|\mathbf{w_d}| = 0$ and $|\mathbf{w_s}| = 1$. By solving Eq's 34 to 37, we obtain $|\mathbf{y_d}| = 0$, $|\mathbf{y_s}| = 1$, $|\mathbf{x_d}| = 1$, $|\mathbf{x_s}| = 1$ and all the vectors point in the same direction. In accordance with these parameters, a feasible systolic implementation for DFT can be derived as shown in Figure 5. By using this scheme, a linear array of n PEs can compute an n point DFT in 2n-1 units of time. In addition, n units of time are needed to drain the results since $y_0, y_1 \cdots y_{n-1}$ are reserved in the corresponding PEs during the computation. It also requires w to be initially loaded into the PEs. All the w's, $0 \leq i \leq n-1$ can be computed in the first n iterations. We remark that this scheme is good because no input is necessary in the vertical direction and the number of I/O ports is small [23].

## 5. Extension

In this section, we show three generalizations to the proposed parameter method.

*(a) Generalized Recurrence Formula*

In the recurrence formulae studied so far, 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 $z^k(i,j) = f[z^{k-1}(i,j), x(i,j,k), y(i,j,k)]$. Systolic processing equations 18 to 21 have to be changed:

$$t_i\mathbf{z_d} + \mathbf{z_{is}} = t_i\mathbf{x_d} + \mathbf{x_{is}} \qquad (38)$$

$$t_i\mathbf{z_d} + \mathbf{z_{is}} = t_i\mathbf{y_d} + \mathbf{y_{is}} \qquad (39)$$

$$t_j\mathbf{z_d} + \mathbf{z_{js}} = t_j\mathbf{x_d} + \mathbf{x_{js}} \qquad (40)$$

$$t_j\mathbf{z_d} + \mathbf{z_{js}} = t_j\mathbf{y_d} + \mathbf{y_{js}} \qquad (41)$$

The proofs of Eq's 38 and 39 are similar to the proof shown in Figure 2 except that an additional PE is needed where y(i+1,j,k) is stored. y(i+1,j,k) is moved to PE A in $t_i$ units of time. Similar arguments apply for Eq. 40 and 41. If a variable does not involve subscript i and/or j explicitly in a recurrence, the corresponding data distribution parameter is zero.

*(b) Non-uniform Periods and Feedback*

In the previous examples, we assume that processing times in the PEs are 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 PEs along a direction of data flow. For example, the last operation in deconvolution is division which takes longer time than multiplication and may

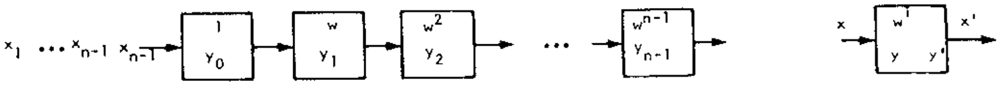

Figure 5. The systolic processor for DFT.

become the bottleneck of data flow. Referring to the recurrence for deconvolution $y^k(i) = f[y^{k-1}(i), x(i,k), a(k)]$ (Eq. 5), 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 PEs be 1. The last iteration of the computation for $y_i$ and the division of $y_i$ to obtain $x_i$ takes 1 + w units of time. This is the period for the x inputs.

$$t_i = 1 + w \qquad (42)$$

Due to the extra time involved in division, the design of feedback signals is more complicated. The relationship of data flow is shown in Figure 6(a). Originally, $y^{k-1}(i)$ is being computed in PE A using x(i−1). $y^p(i+1)$, $p<k-1$ (for backward recurrences), exists in PE B. $y^{k-1}(i)$ is sent to PE D and is converted to x(i) which is sent to PE C in 1 + w units of time. $y^{k-1}(i+1)$ will be computed in PE D. From the above discussion, we have another vector equation:

$$\mathbf{y_d} + \mathbf{x_d} = \mathbf{y_s} + (1+w)\mathbf{y_d}$$

Simplifying,

$$\mathbf{x_d} = w\mathbf{y_d} + \mathbf{y_s} \qquad (43)$$

Eq's 42 and 43 must be included in the original set of systolic processing equations. By letting w = 2, $|\mathbf{a_d}| = 0$, $|\mathbf{a_s}| = 1$ and observing that $t_{ka} = -t_{kx}$. We can solve for the other parameters: $t_{kx} = 3/2$, $t_i = 3$, $|\mathbf{y_d}| = 2/3$, $|\mathbf{y_s}| = -2$, $|\mathbf{x_d}| = -2/3$ and $|\mathbf{x_s}| = 2$ in order to minimize the objective function $(m-1+w)|t_k| + (n-1)|t_i|$. By inserting buffers properly, the systolic processor for deconvolution is shown in Figure 6(b). It should be noted that the velocities of data flow are averaged over three clock cycles. This is the fastest systolic processor that has a throughput of one output every three clock cycles with w = 2.

*(c) Coefficients of Variables*

The coefficients of i, j, k that have been used in the functions x and y are 1. This means that all the elements in x and y are used in the course of computation of the recurrence. When these coefficients are greater than 1, some items in the input will be skipped. As an example, suppose we have the recurrence for FIR filter as:

$$y_i^0 = 0 \qquad (1 \le i \le n)$$

$$y_i^k = y_i^{k-1} + a_{2k}x_{i+3k-1} \quad (1 \le k \le n, x_j = 0 \text{ for } j > 3n) \qquad (44)$$

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 (regardless of sign) can be greater than 1. The general form of the recurrence formula is represented as:

$$z^k(i,j) = f[z^{k-1}(i,j), x(i,mk), y(nk,j)] \qquad (45)$$

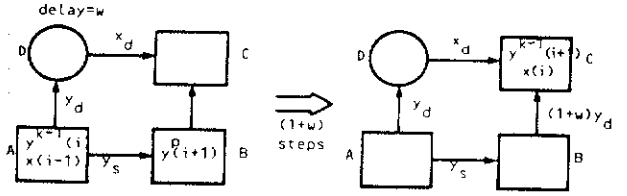

Figure 6a. Relationship of data flow

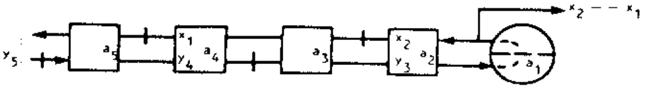

Figure 6b. Systolic processor for deconvoltution

when m,n are integers greater than or equal to 1. In this case, the first two systolic processing equations (Eq's 16, 17) have to be modified as:

$$t_{kx}\mathbf{x_d} + m\mathbf{x_{ks}} = t_{kx}\mathbf{z_d} \qquad (46)$$

$$t_{ky}\mathbf{y_d} + n\mathbf{y_{ks}} = t_{ky}\mathbf{z_d} \qquad (47)$$

The other four systolic processing equations remain unchanged. The proof for these is very similar to the previous proof (Figure 2a). In this case, the data distribution parameters have to be augmented because non-adjacent data are used. Similar changes have to be made on the other four systolic processing equations when the coefficients of i or j are integers greater than 1.

For the FIR filter given in Eq. 44, Eq's 25, 26 have to be changed accordingly. Eq's 27, 28 remain the same. Solving these equations while assuming $t_k = t_i = 1$, we obtain $|\mathbf{y_d}| = 1$, $|\mathbf{y_s}| = -1$, $|\mathbf{a_d}| = 0$, $|\mathbf{a_s}| = 1/2$, $|\mathbf{x_d}| = -1/2$, $|\mathbf{x_s}| = 1/2$. The interesting point is that the separation between $a_i$s is 1/2 unit which implies that only half of the $a_i$s reside in PEs. The other $a_i$s do not have to be loaded since they are never used. The throughput of the system is unchanged, that is, one output per clock cycle.

## 6. Conclusion

Conventional approaches in designing systolic algorithms is usually ad hoc. The resulting design represents a feasible, but not necessarily the optimal, solution. In this paper, we propose a systematic methodology for the design of systolic algorithms. The characteristics of systolic algorithms, which are represented as recurrence processes, are parameterized by the velocity of data flow, spatial distribution, and period of computation. Relationships among these parameters are investigated. The design problem can be formulated into a nonlinear integer program. By using the maximum completion time as the objective function, we found that the optimization problem can be solved very efficiently.

Numerous examples in image processing illustrating the methodology are shown. Special cases that include feedback, non-uniform processing time, and unused data are discussed. Although the systolic processing equations defined may not exhaust all the possible variations, the theory developed can guide the designers in obtaining the necessary equations. Other design requirements, such as hardware and I/O constraints, can also be included in the optimization. The art of designing systolic algorithms is, therefore, reduced to a systematic methodology.

## 7. Acknowledgements

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

## References

[1] H. T. Kung, "Why Systolic Architecture," *IEEE Computer*, Jan. 1982, pp. 37-46.

[2] H. T. Kung, "Highly Concurrent Systems," in *Introduction to VLSI System*, by C. A. Mead and L. A. Conway, Addison-Wesley, 1980.

[3] H. T. Kung and P. L. Lehman, "Systolic (VLSI) Arrays for Relational Database Operations," *Proc. ACM-SIGMOD 1980 Int'l Conf. Management of Data*, May 1980, pp. 105-116.

[4] K. Bromley, J. J. Symanski, J. M. Speiser, and H. J. Whitehouse, "Systolic Array Processor Develop-

ments," in *VLSI Systems and Computations*, H. T. Kung et al., eds, Computer Science Press, Oct. 1981, pp. 273-284.

[5] P. R. Cappello and K. Steiglitz, "Digital Signal Processing Applications of Systolic Algorithms," in *VLSI Systems and Computations*, H. T. Kung et al. eds, Computer Science Press, Oct. 1981, pp. 245-254.

[6] 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, Computer Science Press, Oct. 1981, pp. 255-264.

[7] K. Hwang and Y-H Cheng, "VLSI Computing Structure for Solving Large Scale Linear System of Equations," *Proc. Int'l Conf. on Parallel Processing*, Aug. 1980, pp. 217-227.

[8] K. Hwang and Y-H Cheng, "Partitioned Matrix Algorithms for VLSI Arithmetic Systems," *IEEE Trans. on Comp.*, Vol. C-31, No. 12, Dec. 1982, pp. 1215-1224.

[9] 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, Computer Science Press, Oct. 1981, pp. 265-271.

[10] V. Weiser and A. Davis, "A Wavefront Notion Tool for VLSI Array Design," in *VLSI Systems and Computations*, H. T. Kung et al. eds, Computer Science Press, Oct. 1981, pp. 226-234.

[11] M. J. Foster and H. T. Kung, "The Design of Special-purpose VLSI Chips," *IEEE Computer* Vol. 12, No. 1, Jan 1980, pp. 26-40.

[12] S. Y. Kung, K. S. Arun, D. V. B. Rao and Y. H. Hu, "A Matrix Data Flow Language/Architecture for Parallel Matrix Operation Based on Computational Wavefront Concept," in *VLSI Systems and Computations*, H. T. Kung et al. eds, Computer Science Press, Oct. 1981, pp. 235-244.

[13] A. Mukhopadhyay, "Hardware Algorithms for Nonnumeric Computation," *IEEE Tran. on Comp.*, Vol. C-28, No. 6, June 1979, pp. 384-394.

[14] A. . Kulkarni and D. W. L. Yen, "Systolic Processing and an Implementation for Signal and Image Processing," *IEEE Trans. on Comp.*, Vol. C-31, No. 10, Dec. 1982, pp. 1000-1009.

[15] E. Horowitz, "VLSI Architecture for Matrix Computations," *Proc. 1979 Int'l. Conf. on Parallel Processing*, Aug. 1979, pp. 124-127.

[16] L. Johnson 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, Computer Science Press, Oct. 1981, pp. 213-225.

[17] S.Y. Kung et al., "Wavefront Array Processor, Language, Architecture, and Applications" *IEEE Trans. on Comp.*, Vol. C-31, No. 11, Nov. 1982, pp. 1054-1065.

[18] D.I. Moldovan, "On the Analysis and Synthesis of VLSI Algorithms," *IEEE Trans. on Comp.*, Vol. C-31, No. 11, Nov. 1982, pp. 1121-1126.

[19] G-J Li, *Array Pipelining Algorithms and Pipelined Array Processors*, M.Sc thesis, Institute of Computing Technology, Chinese Academy of Science, 1981.

[20] L. Snyder, "Introduction to the Configurable, Highly Parallel Computer," *IEEE Computer*, Vol. 15, No. 1, pp. 47-56, Jan. 1982.

[21] A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[22] A. M. Geoffrion and R. E. Marsten, "Integer Programming Algorithms: A Framework and State-of-the-Art Survey," *Management Science*, Vol. 18, No. 9, May 1972, pp. 465-491.

[23] H. T. Kung, "Notes on VLSI Computation," *CREST Parallel Processing Systems Course*, Loughborough, England, 1980.