Parallel Computing 3 (1986) 217–229
North-Holland

The mapping of 2-D array processors to 1-D array processors *

C.B. YANG

Institute of Computer and Decision Sciences, National Tsing Hua University, Hsinchu, Taiwan 300

R.C.T. LEE

National Tsing Hua University, Hsinchu, Taiwan 300, and Academia Sinica, Nang Kang, Taipei, Taiwan 115

Received April 1985
Revised December 1985
Communicated by Professor D.J. Evans

Abstract. We consider the case of a 2-dimensional wavefront array processor where only one wavefront appears at any time. We show that in such a situation, this 2-dimensional wavefront processor can be mapped to a linear array processor if the wavefronts never backtrack. The mapping will not increase the number of registers in each processor element. Two examples, the spoken words recognition problem and the longest common subsequence problem, are given to demonstrate the feasibility of this method.

Keywords. VLSI, systolic algorithms, wavefront array processor, single-wavefront array processor, spoken word recognition problem, longest common subsequence problem.

1. Introduction

Because of the advanced elecctronic technologies, we can now use many processors to solve one problem in parallel. That is, this is the age of parallel processing. There are many architectures for parallel processing [10,13]. One of the most important parallel architectures is the systolic array processor architectures which is simple, regular and suitable for implementation by VLSI technology [15,16,20]. Many problems have been solved by systolic algorithms [1,3,4,6,14–16,19–22,26,28–30,32,34,35].

Based upon the idea of systolic array processors, Kung proposed the concept of wavefront array processors [17]. Figure 1 shows a typical wavefront array processor which consists of a matrix of processing elements (PE's for short). Each PE communicates with its neighbors only and as shown in Fig. 1, PE's behave like waves propogating. It was shown in [17] that the wavefront array processor is a tradeoff between the dedicated systolic array processor and the general-purpose data flow multi-processors [5,7,8,23].

In [34], single-wavefront algorithms were defined. In a single-wavefront algorithm, at time step i, only the PE's on diagonal L_i are activated, as shown in Fig. 1. Besides, there is no backtracking of wavefronts for a single-wavefront algorithm.

In this paper, we shall show that every single-wavefront algorithm can be implemented on a linear array processor. That is, we shall propose a general mapping method to map a 2-dimensional single-wavefront array processor to a linear array processor.

* This research work was partially supported by the National Science Council of the Republic of China under contract NSC73-0201-E007-01.

2. Single-wavefront algorithms

In this section, we shall introduce single-wavefront algorithms. Before doing that, we shall first introduce the longest common subsequence (LCS for short) problem. For a comprehensive review of this subject, consult [12].

A string consists of a sequence of symbols. A subsequence of a string is obtained by deleting zero or more than zero symbols, which are not necessarily consecutive, from the original string. For example, 'abe' is a subsequence of 'caadbec'. A string C which is a subsequence of both string A and B is called a common subsequence of A and B. For instance, 'ae' is a common subsequence of 'abe' and 'aace'. Two strings may have more than one common subsequence. The longest common subsequence problem is: Given two strings A and B, find a common subsequence of A and B which is the longest.

Dynamic programming approach is a strategy used to solve many problems [2,11]. In [34], it was shown that the longest common subsequence problem can be solved by systolic algorithms based upon the dynamic programming approach [9,12,27,31]. Let the two input strings for the LCS program be $A = a_1a_2 \dots a_m$ and $B = b_1b_2 \dots b_n$. Let $L(i, j)$ denote the length of an LCS of $a_1a_2 \dots a_i$ and $b_1b_2 \dots b_j$, where $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant n$. The dynamic programming approach to compute $L(i, j)$ is based upon the following equations:

$$\text{if } a_i = b_j, \quad \text{then } L(i, j) = L(i-1, j-1) + 1$$
$$\text{else } L(i, j) = \max(L(i-1, j), L(i, j-1)).$$

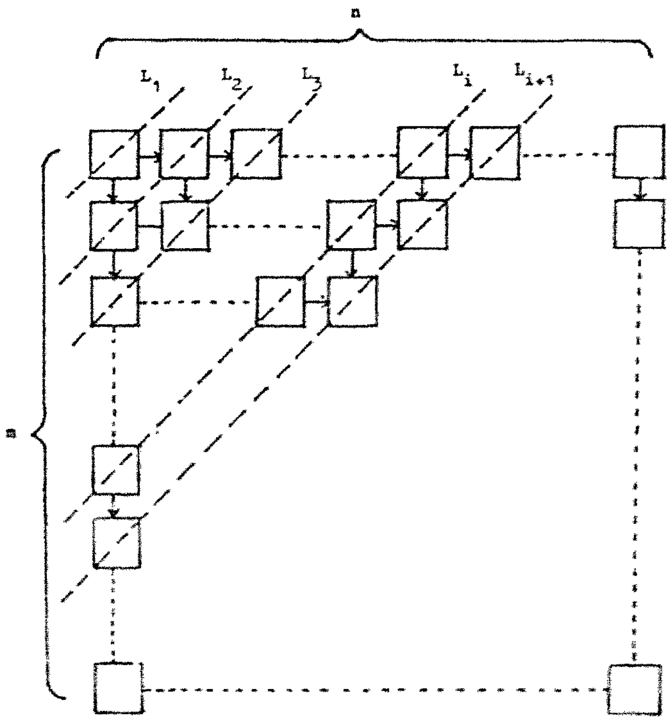


Fig. 1. A wavefront array processor.

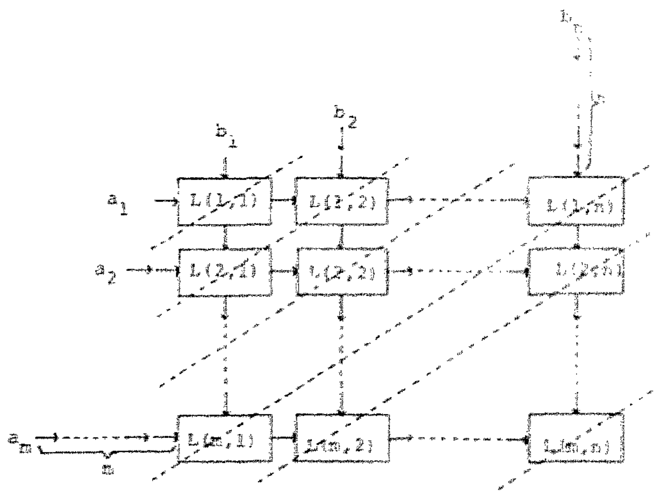


Fig. 2. The calculation of the L-Matrix for the Longest Common Subsequence Problem

The computing sequence of the L matrix is shown in Fig. 2. As soon as the values of $L(i-1, j-1)$, $L(i-1, j)$ and $L(i, j-1)$ have been calculted, the value of $L(i, j)$ can be calculated. For many purposes, only $L(m, n)$ is interesting. In this paper, we shall be interested in calculating $L(m, n)$ only. In general, all of the elements on the same diagonal of the L matrix can be calculated simultaneously.

The LCS problem, as shown in [34], can be solved easily in parallel scheme by using a 2-dimensional array processor as shown in Fig. 2. When the LCS problem is solved by a 2-dimensional wavefront array processor, there is a peculiar phenomon; at any time, there is one and only one wavefront propagating and the wavefronts never backtrack.

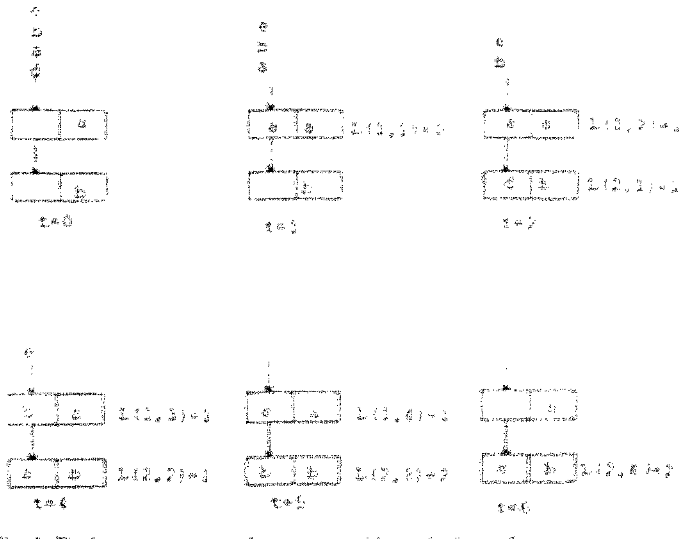


Fig. 3. The longest common subsequence problem solved on a linear array processor.

Consider a single-wavefront algorithm implemented on an $m * n$ 2-dimensional array processor. Let us assume that $m \leqslant n$. We observe that the following are true:

(1) At any time step, at most m processors are activated;

(2) At any time step, only one processor is activated for each row and each column.

Thus a natural question to ask is whether for a single-wavefront algorithm, a linear array processor suffices. This question will be answered in the following sections. Meanwhile, let us note that the longest common subsequence problem can be solved by using a linear array processor.

Consider the case of two strings $A =$ 'ab' and $B =$ 'babe'. Instead of using a 2-dimensional 2×4 array processor, we may use a linear array processor with two processors only. The entire calculation process of finding $L(2, 4)$ is illustrated in Fig. 3.

In the next section, we shall discuss a general model of register interconnection for a 2-dimensional wavefront array processor. Later, we shall show that this connection can be mapped so that a linear array processor suffices for a single-wavefront algorithm.

3. A general model for the register interconnections of 2-dimensional wavefront array processors

Consider the 2-dimensional array processor for a single-wavefront algorithm as shown in Fig. 4. There are three kinds of input data:

(1) Each a_{ij} is stored in PE(i, j);

(2) Each b_j is input from PE(1, j);

(3) Each c_i is input from PE(i, 1).

There are three kinds of outputs:

(1) Each g_{ij} is generated in PE(i, j);

(2) Each d_j flows out from PE (m, j);

(3) Each e_i flows out from PE(i, n).

All of the data in a PE can be only moved to be right and/or lower PE's. Each PE can only access the data within itself. In other words, the above model is a most general one. Note that some data or output data may be absent for some particular algorithms.

We shall further assume that each g_{ij} is computed by referencing previous results, as shown in Fig. 5(a). That is, each PE(i, j) references data which have been generated in PE($i - u$, $j - v$), $0 \leqslant u < k$, $0 \leqslant v < L$ and $u + v \neq 0$ to produce a new result. Hereafter, let R denote a register in a PE for storing some g's. Because $LK - 1$ data of g's must be retained in one PE to generate a new result, there must be LK registers R in each PE. These registers are numbered as $R(1, 1)$, $R(1, 2), \ldots, R(L, K)$ in each PE.

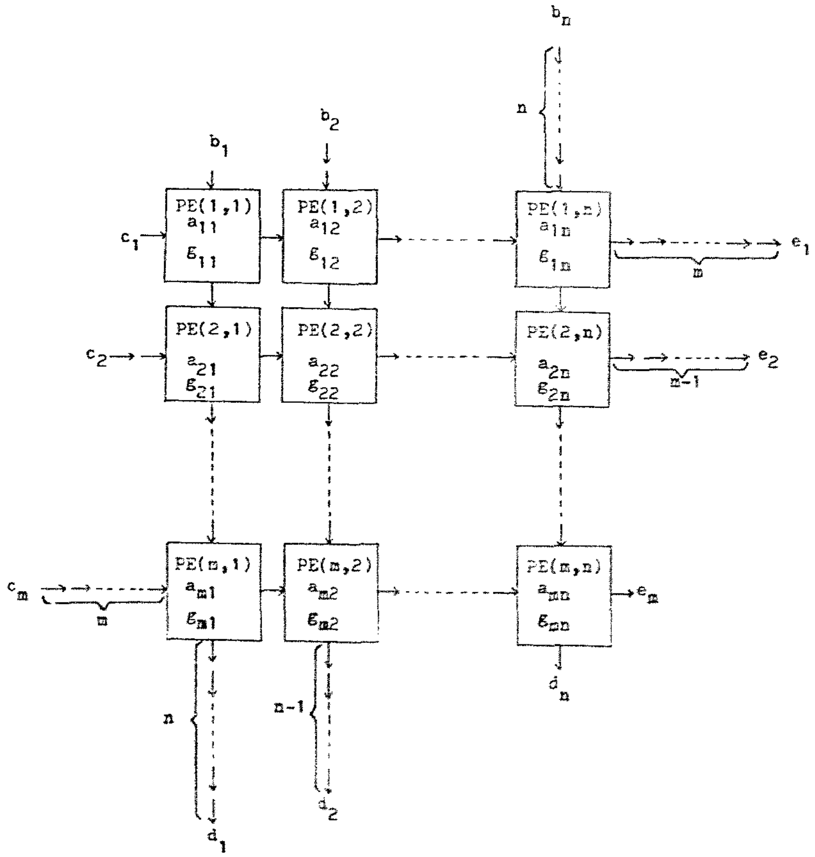
Figure 5(b) shows a general way to connect registers in each PE for a 2-dimensional array processor. We shall call this method of connecting registers Method A. Method A can be summarized as

(1) $R(1, 1)$ of each PE is used to store the newly generated data, which is marked with $\times$ in Fig. 5(b);

(2) For each i and j, $R(1, v)$ of PE($i - 1$, j) is connected to $R(1, v + 1)$ of PE(i, j), where $1 \leqslant v \leqslant k - 1$;

(3) For each i and j, $R(u, v)$ of PE(i, $j - 1$) is connected to $R(u + 1, v)$ of PE(i, j) where $1 \leqslant u \leqslant L - 1$ and $1 \leqslant v \leqslant K$.

Note that Method A is not the only way to connect registers. But the following lemma shows that Method A is feasible for wavefront algorithms. That is, the data needed by PE(i, j) will arrive at PE(i, j) when PE (i, j) is activated.



b_1
b_2
b_n
n
PE(1,1) a_{11} g_{11}
PE(1,2) a_{12} g_{12}
PE(1,n) a_{1n} g_{1n}
PE(2,1) a_{21} g_{21}
PE(2,2) a_{22} g_{22}
PE(2,n) a_{2n} g_{2n}
PE(m,1) a_{m1} g_{m1}
PE(m,2) a_{m2} g_{m2}
PE(m,n) a_{mn} g_{mn}
c_1
c_2
c_m
m
e_1
m
e_2
$m-1$
e_m
n
$n-1$
d_1
d_2
d_n

Fig. 4. The general model of 2-dimensional wavefront array processor with single nonbacktracking wavefronts.

Lemma 1. *If Method A is used in a 2-dimensional wavefront array processor, then the data generated in* PE$(i-v,\ j-u)$ *will be stored in* $(R(u+1,\ v+1)$ *of* PE$(i,\ j)$ *after* $u+v$ *steps where* $0 \leqslant u \leqslant L-1$ *and* $0 \leqslant v \leqslant K-1$.

Proof. We prove this lemma by induction on u and v. Note that this lemma is trivially true for $u=0$ and $v=0$.

It is obvious that the datum generated in PE$(i-v,\ j)$ is stored in $R(1,\ v+1)$, where $0 \leqslant v \leqslant k-1$. We omit the proof for this case.

Assume that this lemma is true for $u-1$ and v. That is, assume that the datum generated in PE$(i-v,\ j-u)$ is stored in $R(u,\ v+1)$ of PE$(i,\ j-1)$ after $u+v-1$ steps. In Method A, $R(u,\ v+1)$ of PE$(i,\ j-1)$ is connected to $R(u+1,\ v+1)$ of PE$(i,\ j)$. Therefore, the datum generated in PE$(i-v,\ j-u)$ will be stored in $R(u+1,\ v+1)$ of PE$(i,\ j)$ where $0 \leqslant u \leqslant L-1$ and $0 \leqslant v \leqslant K-1$. □

By using Lemma 1 and the properties of single-wavefront algorithms, we can easily obtain the following theorem.

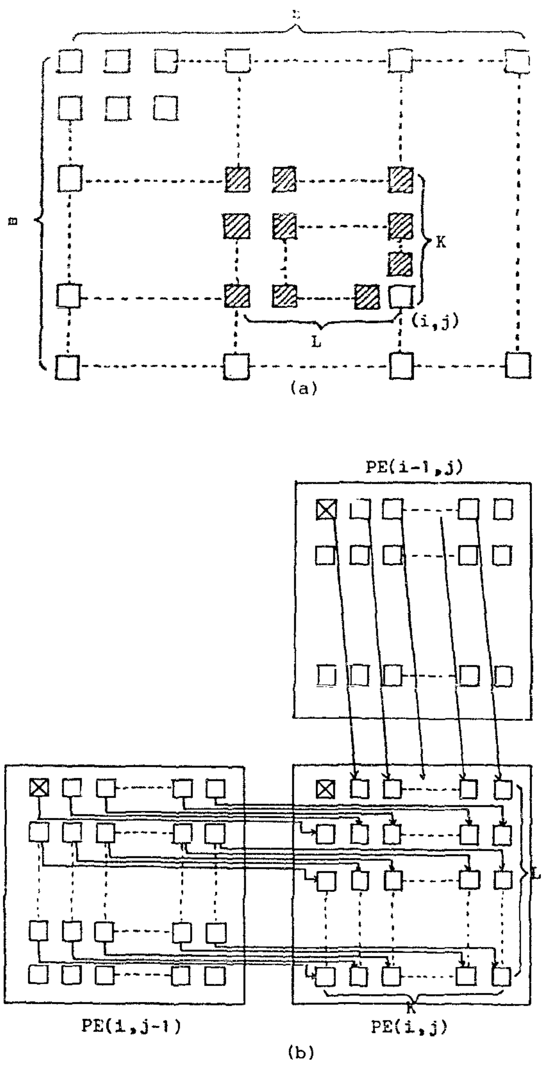


Fig. 5. Method A: a method to connect registers in a 2-dimensional wavefront array processor.

Theorem 1. *Method A is correct if the result of* PE(i, j) *is produced by referencing data previously generated in* PE($i-u$, $j-v$) *where* $0 \leqslant u < K$, $0 \leqslant v < L$, $u+v \neq 0$ *and a wavefront algorithm is used.*

4. The mapping of 2-dimensional array processors to linear array processors

For a single-wavefront algorithm, we note that at each time step, there is only one processor activated for each row. There are therefore at most m processors working concurrently at the same time. In the following, we shall show that a linear array consisting of m processors will be equivalent to a 2-dimensional array processor so far as a single-wavefront algorithm is concerned.

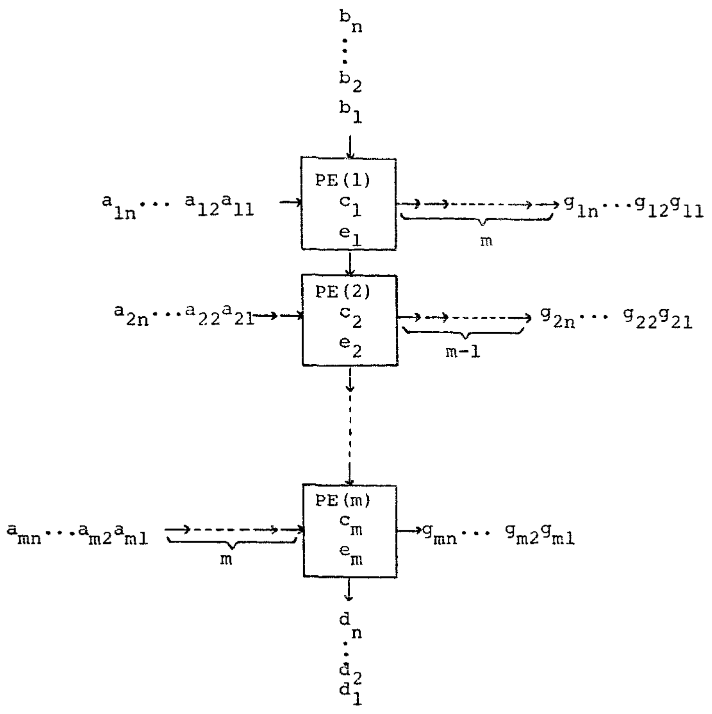

b_n
b_2
b_1
$a_{1n} \cdots a_{12} a_{11}$
PE(1)
c_1
e_1
$g_{1n} \cdots g_{12} g_{11}$
m
$a_{2n} \cdots a_{22} a_{21}$
PE(2)
c_2
e_2
$g_{2n} \cdots g_{22} g_{21}$
m-1
$a_{mn} \cdots a_{m2} a_{m1}$
m
PE(m)
c_m
e_m
$g_{mn} \cdots g_{m2} g_{m1}$
d_n
d_2
d_1

Fig. 6. The rearrangement of data for the linear array processor.

We have to rearrange the data as shown in Fig. 6. Note that so far as the data are concerned, the arrangement in Fig. 6 is equivalent to that in Fig. 4.

To rearrange the inter-register connection, let us note that the following is true for a 2-dimensional single-wavefront array processor:

(1) If PE(i, j) is activated at time step t, then PE(i, $j+1$) and PE($i+1$, j) will be activated at time step $t+1$;

(2) If PE(i, j) is activated at time step t, the datum generated in PE(i, j) will be moved to PE(i, $j+1$) and PE($i+1$, j) at time step $t+1$;

(3) At any time step t, the indices of i and j of all processor elements working concurrently satisfy the following equation:

$$i+j=t+1.$$

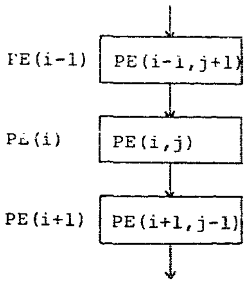


Fig. 7. The mapping of PE's to the linear array processor.

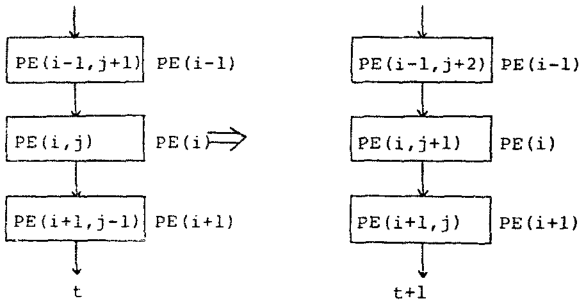


Fig. 8. The behavior of PE's in terms of time in a linear array processor.

Based upon the above observation, we may rearrange the processors as follows:

(1) There are m processors linked into a linear array, labled as PE(1), PE(2), ..., PE(m);

(2) PE(i) corresponds to the ith row of the original 2-dimensional array processor;

(3) If PE(i) acts as PE(i, j) of the original 2-dimensional single-wavefront array processor, then PE($i-1$) and PE($i+1$) act as PE($i-1$, $j+1$) and PE($i+1$, $j-1$) respectively as illustrated in Fig. 7;

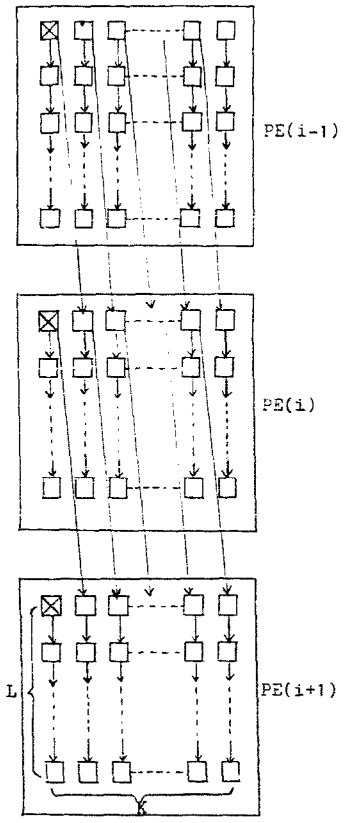


Fig. 9. Method B: a method of connecting registers in a linear wavefront array processor.

(4) If PE(i) acts as PE(i, j) at time step t, then at time step $t+1$, this processor acts as PE(i, $j+1$) as illustrated in Fig. 8;

(5) Since the data in PE(i, j) will be moved to PE($i+1$, j) in the 2-dimensional array processor, the data in PE(i) of the linear array will be moved to PE($i+1$). Similarly, since the data in PE(i, j) will be moved to PE(i, $j+1$) in the 2-dimensional array processor, the data in PE(i) will be stored in PE(i). The inter-connection is illustrated in Fig. 9.

We shall call the method of connecting registers as illustrated in Fig. 9 Method B. Method B can be summarized as follows:

(1) Each PE(i) contains excatly the same number of registers as in Method A;

(2) $R(1, 1)$ of each PE is used to store the newly generated result;

(3) In each PE(i), $R(1, v)$ of PE($i-1$) is connected to $R(1, v+1)$ of PE(i) where $1 \leqslant v \leqslant K-1$;

(4) In each PE(i), $R(u, v)$ of PE(i) is connected to $R(u+1, v)$ of PE(i), where $1 \leqslant u \leqslant L-1$ and $1 \leqslant v \leqslant K$.

In the following, we shall prove that our mapping method is correct.

Theorem 2. *Method B is correct for replacing a 2-dimensional single-wavefront array processor by a linear array.*

Proof. For a 2-dimensional single-wavefront array processor, at time step t, PE(i, j) is activated if and only if $i+j=t+1$. Thus, PE(i, $t-i+1$) is activated at each time step t. We shall now prove that for the linear array processor, each processor PE(i) performs exactly the same task as PE(i, $t-i+1$) of the 2-dimensional single-wavefront array processor. We do this by induction on t. For $t=1$, PE(1, 1) of the 2-dimensional array processor is activated. For $t=1$, PE(1) of the linear array processor is activated. The data generated by PE(1, 1) are all available in PE(1) of the linear array.

Assume that for some time t, PE(i) performs the same task as PE(i, $t-i+1$). Let the data which will be moved from PE(i, $t-i+1$) to PE(i, $t-i+2$) and PE($i+1$, $t-i+1$) be denoted as D_1 and D_2 respectively. For Method B, D_1 will be retained in PE(i) and D_2 will be moved to PE($i+1$). Thus at time step $t+1$, PE(i) and PE($i+1$) will obtain all of the data they need and perform the same tasks as PE(i, $t-i+2$) and PE($i+1$, $t-i+1$) respectively. That is, for every t, PE(i) of the linear array performs the same task as PE(i, $t-i+1$) of the 2-dimensional array processor. □

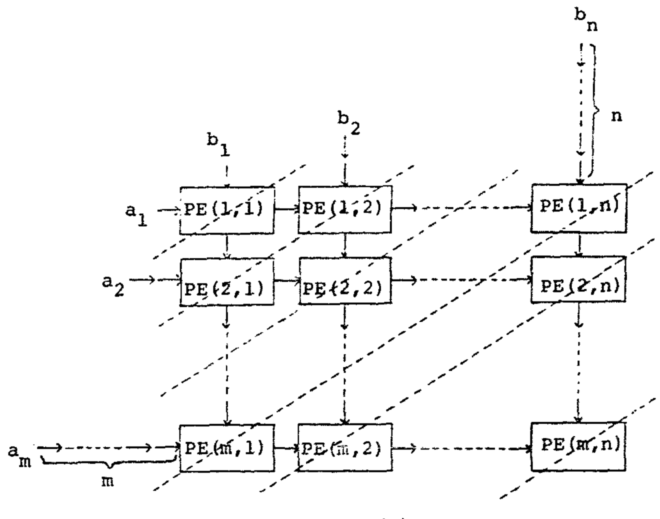
5. Examples of mapping

In this section, we shall show examples of mapping 2-dimensional single-wavefront array processors to linear array processors.

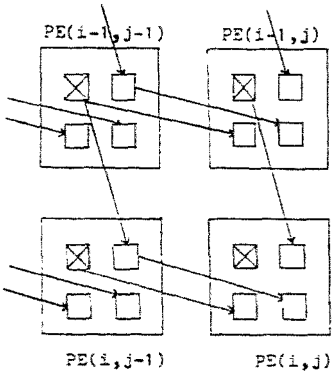
Example 1. *The Longest Common Subsequence Problem.* For this problem, our job is to find $L(i, j)$ which is determined by $L(i-1, j-1)$, $L(i, j-1)$ and $L(i-1, j)$. For the 2-dimensional wavefront processor, the data are arranged as in Fig. 10(a). The inter-connection of registers is shown in Fig. 10(b) by using Method A.

If Method B is used is used, the input data arrangement is illustrated in Fig. 11(a) and the inter-conncetion of registers is shown in Fig. 11(b).

Example 2. *The Spoken Word Recognition Problem.* The spoken word recognition problem was defined and discussed in [24,25]. We assume that there are two speech patterns: one is the



(a)



(b)

Fig. 10. Method A for the longest common subsequence problem.

reference pattern and the other is the input pattern. Each pattern is time-sampled. Let A and B represent the reference pattern and the input pattern respectively:

$$A = a_1 a_2 \dots a_m \quad \text{and} \quad B = b_1 b_2 \dots b_n$$

where a_i and b_j are feature vectors.

We shall not get into the details on how the feature vectors are obtained. This is not our concern. The spoken word problem is to determine whether B is sufficiently similar to A. To measure the distance between A and B, we can not use any ordinary distance function as presented in [18]. We shall try to match these two patterns as much as we can. If, after matching them as much as we can, we still conclude that these two patterns are vastly different, we have to make a painful conclusion that B is not similar to A after all.

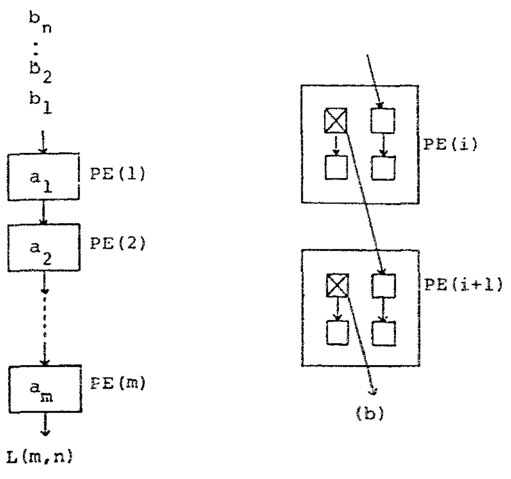


Fig. 11. Method B for the longest common subsequence problem.

In matching B with A, the following rules must be observed:

(1) a_1 must be matched with b_1,

(2) a_m must be matched with b_n,

(3) If b_j is matched with a_i, then only the following matching can be performed:

(a) b_j is matched with a_{i+1},

(b) b_{j+1} is matched with either a_i or a_{i+1}.

In other words, (3) stipulates that the matching in Fig. 12 is not allowed.

The spoken word recognition problem has been solved by the dynamic programming approach in [24,25]. West [33] designed special circuits to solve this problem based upon the concept of 2-dimensional wavefront array processor. In this example, we shall apply our mapping method to produce a linear array processor to solve this problem. Let $g(i, j)$ be the

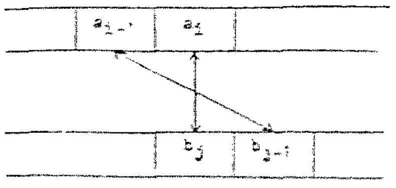


Fig. 12. Forbidden matching for the spoken word recognition problem.

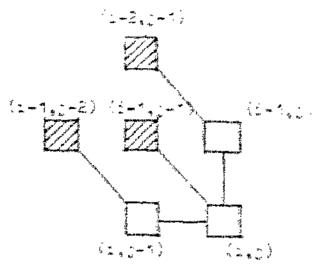


Fig. 13. Data behind referenced in a 2-dimensional wavefront array processor for the spoken word recognition problem.

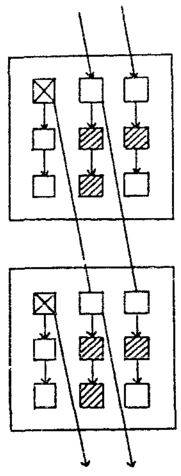


Fig. 14. Method B for the spoken word recognition problem.

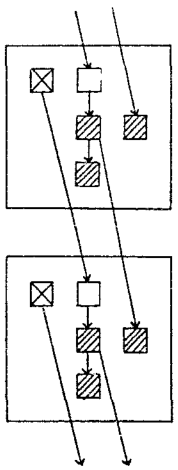


Fig. 15. A referred register connection for the spoken word recognition problem.

distance between two sequences of feature vectors $a_1a_2 \ldots a_i$ and $b_1b_2 \ldots b_j$. The algorithm for calculating $g(i, j)$ with slope constraint 1 is as follows. (Slope constraint was defined in [25], which prevents too steep or too gentle gradient in the matching between $a_1a_2 \ldots a_i$ and $b_1b_2 \ldots b_j$.)

$$g(i, j) = \min(g(i-1, j-2) + 2d(i, j-1) + d(i, j),\\ g(i-1, j-1) + 2d(i, j),\ g(i-2, j-1) + 2d(i-1, j) + d(i, j))$$

where

$$d(i, j) = \|a_i - b_j\|$$

and the initial condition is as follows:

$$g(1, 1) = 2d(1, 1).$$

The distance between two speech patterns A and B is

$$D(A, B) = g(m, n).$$

The solution for this spoken word recognition problem is quite similar to that of the longest common subsequence problem. Again, the dynamic programming strategy can be used. Figure 13 shows how data are referenced for computing $g(i, j)$ in a 2-dimensional array processor. Figure 14 shows how registers are connected in a linear array processor by using Method B. Since many registers are wasted, we may delete some of them as shown in Fig. 15.

6. Concluding remarks

The 2-dimensional wavefront array processor is a natural concept to many VLSI designers. Previous research emphasises the high-level architecture without discussing inter-register connection method. In this paper, we have shown that if the systolic algorithm is a single-wavefront algorithm, then a linear array processor is sufficient. A general inter-register connection method for this linear array is proposed. We have also shown that our mapping method can be applied to design linear array processors to solve the longest common subsequence problem and the spoken word recognition problem.

References

[1] A. Apostolico and A. Negro, Systolic algorithms for string manipulations, *IEEE Trans. Comput.* **33** (4) (1984) 361–364.

[2] R.E. Bellman, *Dynamic Programming* (Princeton University Press, Princeton, NJ, 1957).

[3] R.P. Brent and H.T. Kung, Systolic VLSI arrays for polynomial GCD computation, *IEEE Trans. Comput.* **33** (8) (1984) 731–376.

[4] B. Chazelle, Computational geometry on a systolic chip, *IEEE Trans. Comput.* **33** (9) (1984) 774–785.

[5] J.B. Dennis, Data flow supercomputers, *IEEE Comput.* (November 1980) 48–56.

[6] M.J. Foster and H.T. Kung, The design of special-purpose VLSI chips, *IEEE Comput.* (January 1980) 26–40.

[7] D.D. Gajski, D.A. Padua, D.J. Kuck and R.H. Kuhn, A second opinion on data flow machines and languages, *IEEE Comput.* (February 1982) 58–69.

[8] K.P. Gostelow and R.E. Thomas, Performance of a simulated dataflow computer, *IEEE Trans. Comput.* **29** (10) (1980) 905–919.

[9] D.S. Hirschberg, Algorithms for the longest common subsequence problem, *J. ACM* **24** (4) (1977) 664–675.

[10] R.W. Hockney and C.R. Jesshope, *Parallel Computers* (Adam Hilger, Bristol, 1981).

[11] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms* (Computer Science Press, Baltimore, MD, 1978).

[12] W.J. Hsu, New algorithms for the longest common subsequence problems, Ph.D. Thesis, National Chiao Tung University, Taiwan, 1983.

[13] K. Hwang and F.A. Briggs, *Computer Architecture and Parallel Processing* (McGraw-Hill, New York, 1984).

[14] A.V. Kulkarni and D.W. Len, Systolic processing and implementation for signal and image, *IEEE Trans. Comput.* **31** (10) (1982) 1000–1009.

[15] H.T. Kung, Notes on VLSI computation, in: D.J. Evans, ed., *Parallel Processing System* (Cambridge Univ. Press, London, 1982) 339–356.

[16] H.T. Kung, Why systolic architectures?, *IEEE Comput.* (January 1982) 37–46.

[17] S.Y. Kung, K.S. Arun, R.J. Galezer and D.V.B. Rao, Wavefront array processor: Language, architecture, and applications, *IEEE Trans. Comput.* **31** (11) (1982) 1054–1066.

[18] R.C.T. Lee, Clustering analysis and its applications, in: J.T. Tou, ed., *Advances in Information Systems Science*, Vol. 8 (Plenum, New York, 1981) Chapter 4.

[19] J.V. McCanng, J.G. McWhirter and K. Wood, Optimised bit level systolic array for convolution, *IEE Proc.* **131** (6) (1984) 632–637.

[20] C.A. Mead and L.A. Conway, *Introducation to VLSI Systems* (Addison-Wesley, Reading, MA, 1980) Section 8.3.

[21] D.I. Moldovan, On the analysis and synthesis of VLSI algorithms, *IEEE Trans. Comput.* **31** (11) (1982) 1121–1126.

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

[23] J.E. Rumbaugh, A data flow multiprocessor, *IEEE Trans. Comput.* **26** (2) (1977) 138–146.

[24] H. Sakoe and S. Chiba, Dynamic programming algorithm optimization for spoken word recognition, *IEEE Trans. Acoust. Speech. Signal Processing* **26** (1) (1978) 43–49.

[25] H. Sakoe, two-level DP-matching—A dynamic programming-based pattern matching algorithm for connected word recognition, *IEEE Trans. Acoust. Speech Signal Processing* **27** (6) (1979) 588–595.

[26] C. Savage, A systolic design for connectivity problems, *IEEE Trans. Comput.* **33** (1) (1984) 99–103.

[27] P.H. Sellers, An algorithm for the distance between two finite sequences, *J. Combin. Theory Ser. A* **16** (1974) 253–258.

[28] M. Steinby, Systolic trees and systolic language recognition by tree automata, *Theoret. Comput. Sci.* **22** (1983) 219–232.

[29] J.D. Ullman, *Computational Aspects of VLSI* (Computer Science Press, Rockville, MD, 1984) Chapter 5.

[30] R.B. Urquhart and D. Wood, Systolic matrix and vector multiplication methods for signal processing, *IEE Proc.* **131** (6) (1984) 623–631.

[31] R.A. Wagner, The string-to-string correction problem, *J. ACM* **21** (1) (1974) 168–173.

[32] C.R. Ward, A.J. Robson, P.J. Hargrave and J.G. Mcwhirter, Application of a systolic array to adaptive beamforming, *IEE Proc.* **131** (6) (1984) 638–645.

[33] N. Weste, D.T. Burr and B.D. Ackland, Dynamic time warp pattern matching using an integrated multiprocessing array, *IEEE Trans. Comput.* **32**, (8) (1983) 731–744.

[34] C.B. Yang and R.C.T. Lee, Systolic algorithms for the LCS problem, *Proc International Computer Symposium 1984*, Taipei, Taiwan (1984) 895–901.

[35] C.S. Yeh, I.S. Read and T.K. Troons, Systolic multipliers for finite field GF(2^m), *IEEE Trans. Comput.* **33** (4) (1984) 357–360.