Efficient Storage Schemes for Arbitrary Size Square Matrices in Parallel Processors with Shuffle-Exchange Networks

Rajendra V. Boppana and C. S. Raghavendra
Department of Electrical Engineering-Systems
University of Southern California, Los Angeles, CA 90089-0781

Abstract. We present storage schemes based on linear permutations to store and access $N \times N$, for any integer N, data arrays in parallel processors with shuffle-exchange type interconnection networks. For parallel access of the most important templates, namely, row, column, main diagonal, and square block, simple criteria are given based on the linear permutations involved. This is the first such solution to provide access of the important templates of arbitrary size square matrices without memory and network conflicts using minimum number of memory modules and a shuffle-exchange type network.

Key words: data alignment, linear permutations, matrix storage, parallel memory systems, scrambled skewed storage, shuffle-exchange networks.

1 Introduction

In parallel processors, access to shared data in the memory modules plays an important role in the overall performance. For specific problems, knowing the data access patterns by processors, one can allocate data to memory modules initially so that high parallelism can be achieved in data access at run time. By the parallel access of data, we mean that the access of data is free of memory conflicts (that is, no memory module contains more than one element of the data to be accessed) and interconnection network conflicts (that is, one pass through the interconnection network is sufficient to transfer data from processors to memory modules or vice versa). In this paper we address the problem of storing an $m \times m$ matrix, $m \geq N$, in the N memory modules such that various N-(element)subsets (also called, *templates*) of the matrix are retrieved from memory modules without memory and interconnection network conflicts.

Previously known results [3, 4, 5, 7, 9, 10, 11, 12] either assume that N is a power of 2 and provide template access without memory and network conflicts or, for arbitrary N, discuss only the issue of template access without memory conflicts. In this paper, we present a class of scrambled schemes, called *clip* schemes, to store an $N \times N$ matrix for any integer N. These schemes are based on the class of *composite linear permutations*. As a main result of this paper, we specifically show some storage schemes in the proposed class which provide conflict free access of rows, columns, main and back diagonals, and square blocks (when N is square of an integer) of data arrays using the generalized shuffle-exchange networks.

2 Preliminaries

We assume that there are N processors and N memory units in the parallel processor system. Let $N = p_1^{r_1} \times \cdots \times p_k^{r_k}$, where $p_1, \cdots, p_k$ are distinct and primes, $k \geq 1$, $r_1, \ldots, r_k \geq 1$, and $n = r_1 + \cdots + r_k$. Let d_x, $0 \leq x < n$, be the xth digit of the sequence $p_1, \ldots, p_1(r_1 \text{ times}), p_2, \ldots, p_2(r_2 \text{ times}), \ldots, p_k$. Let $w_0 = 1$ and $w_x = \prod_{y=1}^{x} d_y$, $1 \leq x < n$. Each processor (memory module) is given a unique and distinct index i, $0 \leq i < N$, which can be represented in mixed radix form as $i_{n-1} \ldots i_0$ with d_x and w_x as the radix and weight of the xth digit, i_x. However, it is treated as an n-digit column vector $(i_0, \ldots, i_{n-1})^T$ in the matrix-vector computations defined below. Similarly , given an n-digit column vector $(j_0, \ldots, j_{n-1})^T$, its value is computed as $\sum_{x=1}^{x=k} j_x w_x$.

We use modulo arithmetic in the matrix-vector computations involving vectors and matrices defined in mixed radix form. Specifically, if two digits of radix p are added (or multiplied) then the result is given in modulo p. For example,the modulo addition and multiplication of the two digits 2 and 3 of radix 5 are 0 and 1, respectively. We never perform modulo arithmetic between digits of different radices. We use $\oplus$ to represent the modulo addition and juxtaposition to indicate multiplication.

Subsets of a matrix. A *template* T of an $N \times N$ data matrix is a set of N element positions $\{(\lambda_x, \mu_x) | 0 \leq x < N\}$, with $(\lambda_0, \mu_0) = (0, 0)$ [11]. By the access of a template T, we mean accessing of the N elements of the matrix whose positions are given by the template. Row (T_r), column (T_c), diagonal (T_d), and square block (T_s) templates are the four important templates reported in the literature [4, 9]. These templates are defined as follows.

Definition 1 $T_r = \{(0, j) | 0 \leq j < N\}$, $T_c = \{(i, 0) | 0 \leq i < N\}$, $T_d = \{(i, i) | 0 \leq i < N\}$, *and* $T_s = \{(i, j) | 0 \leq i, j < \sqrt{N}\}$.

It is clear that T_r, T_c, T_d, and T_s correspond to the positions given by, respectively, row 0, column 0, main diagonal, and the square block $S_{0,0}$ of the data matrix. The square block $S_{i,j}$ of an $N \times N$ matrix, N square of some integer, is the $\sqrt{N} \times \sqrt{N}$ submatrix with (i, j) in the top left position. Given a template $T = \{(0, 0), (\lambda_x, \mu_x) \mid 1 \leq x < N\}$, the set $\{(a, b), (a \oplus \lambda_x, b \oplus \mu_x) | 1 \leq x < N\}$, defines the affine template $T(a\ b)$ of T. For example, ith row is the affine row template, $T_r(i, 0)$. The back diagonal of a matrix is the diagonal starting at its upper right corner and ending at its lower left corner; it is an affine

International Conference on Parallel Processing, pp. 365-368, vol. 1, 1991

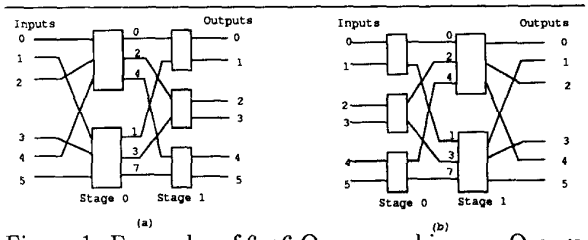

Figure 1: Examples of 6×6 Omega and inverse Omega networks. Addresses of lines before each stage are as indicated.

template of some template, for example, for $N = 2^n$, $T_d(0, N-1)$.

Processor-memory interconnection networks. We assume that $N \times N$ Omega or generalized shuffle-exchange networks [8, 1], which are less expensive compared to the alternatives such as the crossbar, are used for processor-memory interconnection. An important property of these networks is the ability to setup paths from inputs to outputs using only the output address with simple digit-controlled routing (at each stage, a particular digit of the address is used) [8]. An $N \times N$ Omega network [8] consists of n stages, numbered $0, \ldots, n-1$ left to right. Stage $(n-1-x)$, $0 \leq x < n$, contains N/d_x switches of size $d_x \times d_x$ which are interconnected to the preceding stage with a d_x-shuffle. In a p shuffle, i, $0 \leq i < N-1$, is mapped to $pi \bmod (N-1)$ and $(N-1)$ is mapped to itself. The Omega network, for $N = 2 \times 3$, and its inverse are shown in Figures 1(a) and 1(b).

Composite linear permutations. Using the concept of linear transformations [6], we define composite linear transformations on the set obtained by the cartesian product of vector spaces. For primes p, q and $r_1, r_2 > 0$, let ${}_pV_{r_1}$ and ${}_qV_{r_2}$ be two vector spaces over $GF(p)$ and $GF(q)$ respectively. Let $V = {}_pV_{r_1} \times {}_qV_{r_2}$. An element $i \in V$ is a $(r_1 + r_2)$-tuple with first r_1 elements (compactly denoted $i_{0:r_1-1}$) in $GF(p)$ and the remaining elements in $GF(q)$. Under the composite linear transformation, any $i \in V$ is mapped to some $j \in V$ given by the following matrix equation.

$$\left(\begin{array}{c} j_{0:r_1-1} \\ \hline j_{r_1:r_1+r_2-1} \end{array}\right) = \left(\begin{array}{c|c} {}_pQ_{r_1} & 0 \\ \hline 0 & {}_qQ_{r_2} \end{array}\right) \left(\begin{array}{c} i_{0:r_1-1} \\ \hline i_{r_1:r_1+r_2-1} \end{array}\right) \quad (1)$$

The matrix, denoted Q, in the above equation is called the characteristic matrix of the composite linear transformation; entries of its submatrix ${}_pQ_{r_1}$ are in $GF(p)$ and those of ${}_qQ_{r_2}$ are in $GF(q)$. A characteristic matrix is said to be nonsingular if and only if each of its submatrices corresponding to a particular radix is nonsingular. In that case, the mapping is called 'composite linear permutation'.

A composite linear permutation on a set of N elements has its $n \times n$ characteristic matrix as follows.

$$\begin{pmatrix} {}_{p_1}Q_{r_1} & 0 & \cdots & 0 \\ 0 & {}_{p_2}Q_{r_2} & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & {}_{p_k}Q_{r_k} \end{pmatrix}$$

If the columns of Q are represented as $q_0, \ldots, q_{n-1}$, then $Q = (q_0, \ldots, q_{n-1})$ and $Qi = i_0 q_0 \oplus \cdots \oplus i_{n-1} q_{n-1}$. A composite affine linear permutations is of the form $j = Qi \oplus c$ for some $c \in V$. The composition of two composite affine linear permutations is again a composite affine linear permutation [2].

3 Inverse Omega passable composite linear permutations

Only a subset of the class of composite linear permutations are realized by a generalized Shuffle-Exchange network in one pass. Therefore, we characterize this subset of composite linear permutations so that network conflicts are eliminated in the data access by using the storage schemes based on such permutations.

Definition 2 *For a matrix $Q = (q_{i,j})_{n \times n}$, the "leading" principal submatrix $Q(x)$, $0 \leq x < n$, of Q is defined as follows.*

$$\begin{pmatrix} q_{0,0} & \cdots & q_{0,x} \\ \vdots & & \\ q_{x,x} & \cdots & q_{x,x} \end{pmatrix}$$

Theorem 1 *A composite linear permutation, π, with characteristic matrix Q on the set of N numbers is realized by an $N \times N$ inverse Omega network if only if all the leading principal submatrices of Q are nonsingular.*

For proof, see [2]. It is easy to show that if a composite linear permutation is passed by the inverse Omega network, then any affine composite linear permutation is passed by it. The composite linear permutations passable by an Omega network can be characterized in an analogous manner [2].

4 Clip schemes

The clip storage scheme: Element (i, j) of the data matrix is stored in location i of the memory unit with index $i \oplus \pi(j)$, where π is some composite linear permutation. ∎

The most important characteristic of a clip scheme is the composite linear permutation (hence, the corresponding 'Q' matrix) used. This composite linear permutation gives the complete information about the storage scheme. Therefore, a clip scheme, which defines an $N \times N$ mapping matrix, can be compactly represented using a much smaller $n \times n$ characteristic matrix, Q. In Figure 2(a), we give the mapping matrix for storing a 9×9 data matrix in 9 memory units, as specified by the clip scheme using the composite linear permutation with the characteristic matrix given

0	6	3	1	7	4	2	8	5
1	7	4	2	8	5	0	6	3
2	8	5	0	6	3	1	7	4
3	0	6	4	1	7	5	2	8
4	1	7	5	2	8	3	0	6
5	2	8	3	0	6	4	1	7
6	3	0	7	4	1	8	5	2
7	4	1	8	5	2	6	3	0
8	5	2	6	3	0	7	4	1

(a)

$$\begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}$$

(b)

Figure 2: The mapping and characteristic matrices of a clip scheme to store a 9×9 data matrix.

in Figure 2(b). With clip schemes, address computations can be done on-the-fly, since the arithmetic to compute the address of a data element is performed on individual digits of its indices. There is no propagation of carry from one digit to another during an address computation. When N is a power of 2, the address generation circuit consists of only exclusive-or gates [3]. In the general case, simple digit arithmetic circuits can be used [2].

Conflict-free access of templates of interest. For each template, T, of interest, we would like to know whether the template access is free of memory conflicts. If it is so, then we want to determine the data transfer function, $f(T)$, to be realized by the interconnection network in the access of that template or any of its affine template. For the row, column, diagonal, and square block templates, which are of interest to us here, this is typically done by showing that the index of the memory unit that a given processor has to access is given by some matrix-vector computation that is in the form of equation 1. Whenever the matrix appearing in the computation is non-singular, the template access is free of memory conflicts and the corresponding data transfer function is a composite affine linear permutation. From this matrix, we can also determine whether the data transfer function is realized by an Omega or inverse Omega network. The theorems below state the criteria for providing conflict-free access of rows, columns, main and back diagonals, and square blocks. Proofs of these results are given in [2].

Theorem 2 *For any clip scheme, T_r and T_c are free of memory conflicts and $f(T_r)$ and $f(T_c)$ are composite linear permutations, given by the matrices Q and I (identity matrix), respectively.*

Theorem 3 *For a clip scheme with characteristic matrix Q, T_d is free of memory conflicts if and only if the following matrix is non-singular.*

$$Q'' = Q \oplus I$$

Furthermore, whenever Q'' is non-singular, $f(T_d)$ is a composite linear permutation given by Q''.

Theorem 4 *For a clip scheme with characteristic matrix Q, the back diagonal is accessible free of memory conflicts if and only if the matrix $(I - Q)$ is nonsingular. Furthermore, when it is nonsingular, the data transfer function is a composite linear permutation.*

For the access of square block templates, we assume that N is square of an integer, that is, $r_1, \ldots, r_k$ are all even. The mixed radix form considered thus far does not use all the radices in representing values in the set $\{0, \ldots, \sqrt{N} - 1\}$. Therefore, it does not facilitate a systematic study of the square block templates, since both indices of an element of the template are less than $\sqrt{N}$. So, we slightly modify the representation of i, $0 \leq i < N$. In the modified form, d_x, $0 \leq x < n/2$, is the xth digit of the sequence $p_1, \ldots, p_1, p_2, \ldots, p_2, \ldots, p_k, \ldots, p_k$, where each p_y, $1 \leq y \leq k$, is repeated $r_y/2$ times; for $n/2 \leq x < n$, $d_x = d_{x-n/2}$. Weights w_x are computed as products of d_x as defined before. Then, i is represented in mixed-radix form $i_{n-1} \ldots i_0$ such that i_x, $0 \leq x < n$, is of radix d_x and of weight w_x. For example, for $N = 2^4 \times 3^2 \times 5^6$, 100 is represented in the earlier notation as $(000000)_5(20)_3(0100)_2$, here the subscript indicates the radix of the digits within the parentheses, and in the modified notation given above as $(000)_5(0)_3(00)_2(013)_5(1)_3(00)_2$. Similarly, rows and columns of a characteristic matrix expressed in the earlier notation are rearranged to obtain the corresponding characteristic matrix in the modified notation. Note that the modified notation does not affect the results for the conflict-free access of the various templates given above. Let I_n, or simply I when there is no confusion, is the $n \times n$ identity matrix; xth column of I is denoted by e_x.

Theorem 5 *For a clip scheme with characteristic matrix Q, T_s is free of memory conflicts if and only if the characteristic matrix Q' (obtained from Q such that, for $0 \leq i < n/2$, $q'_i = q_i$, and, for $n/2 \leq i < n$, $q'_i = e_{i-n/2}$) is nonsingular. Furthermore, whenever Q' is non-singular, $f(T_s)$ is a composite linear permutation given by Q'.*

All the above results obtained for various templates are applicable for any of the affine templates derived from these templates [2]. For example, if the transfer function for the access of column 0 (column template, T_c) is a composite affine linear permutation, then the transfer function for the access of any other column (an affine template of T_c) is also a composite affine linear permutation.

5 Some practical clip schemes

Thus far, we have described the criteria for the access of various templates when a clip scheme is used. For each template access of interest, we showed that the template is free of memory conflicts if and only if the characteristic matrix of the storage scheme or some matrix derived from it is nonsingular. Furthermore, conflict-free access of row and column templates is always guaranteed. It still remains to be shown whether

there are any characteristic matrices which satisfy the constraints given in theorems 3, 4 and 5. In such a case, to realize the data transfer functions of these templates without network conflicts, the characteristic matrix of a storage scheme and its derivatives should satisfy Theorem 1.

For any N, a square number, we use the mixed radix form discussed in the context of access of the square block templates. Let $L_{n/2}$ ($U_{n/2}$) be an $\frac{n}{2} \times \frac{n}{2}$ lower (upper) triangular characteristic matrix such that an entry on the diagonal has value $\lceil p/2 \rceil$ where p is the radix of that entry. Consider a characteristic matrix of the following form.

$$\Gamma_n = \left(\begin{array}{c|c} L_{n/2} & I_{n/2} \\ \hline I_{n/2} & 0 \end{array} \right) \qquad (2)$$

We may omit the subscript n if there is no confusion.

Theorem 6 *For any N not a multiple of 2 or 3, Γ defines a clip storage scheme for which row, column, main diagonal, back diagonal, and square block templates are free of memory conflicts. Furthermore, the data transfer functions for the access of all these templates are realized by an $N \times N$ inverse Omega network.*

This is easily proved [2] by showing that the characteristic matrix corresponding to the data transfer function of each template access is nonsingular and satisfies Theorem 1. With a little experimentation, many schemes satisfying Theorem 6 can be constructed. For example, the characteristic matrices with $I_{n/2}$ replaced by $U_{n/2}$ in (2) can be used for clip schemes with the same results. Also a diagonal entry of $L_{n/2}$ or $U_{n/2}$ could be any value other than 0,1 or $p-1$, where p is the radix of that entry, in the above example characteristic matrices without affecting the validity of the results.

For the case when N is a multiple of 2 or 3, the above schemes provide access of the templates without memory conflicts; however, to avoid network conflicts in the access of templates, an inverse Omega network is required for row, column, and square block accesses, and an Omega network is required for the main and back diagonal accesses. This is because the smallest leading submatrix, which is simply a 1×1 matrix, of the characteristic matrix can not take three nonsingular values for when the radix is 3 or 2. See [2] for more details. Clip schemes for conflict-free access of templates in parallel processors with Omega networks can be developed similarly.

6 Summary

In this paper, we considered a class of scrambled skewing schemes, called the clip schemes. These schemes have the following advantages: (a) use of simple and popular Omega type interconnection networks to realize the data transfer functions of a template, (b) compact representation of the storage scheme, and (c) efficient address generation methods. These aspects are crucial for the use of a storage scheme. The proposed schemes are flexible and could be used in storing multidimensional and larger size matrices [2]. The main contribution of the paper is the development of some special clip schemes for use in parallel processors with shuffle-exchange type interconnection networks.

Further work in this direction could be in developing systematic procedures to obtain schemes that allow conflict free access of the templates of interest. One can analyze the transfer functions for subsets that are regular but not affine templates and investigate methods to realize them using an Omega or a similar network. Also, further work is warranted in realizing composite linear permutations by Omega networks with binary switches.

References

[1] L. N. Bhuyan and D. P. Agrawal. Design and performance of generalized interconnection networks. *IEEE Trans. on Computers*, C-32(12):1081–1090, December 1983.

[2] R. V. Boppana and C. S. Raghavendra. Data access and alignment of arbitrary size square matrices in parallel processors with multistage dynamic and static interconnection networks. Technical report, Dept. of EE-Systems, Univ. of Southern Cal., Univ. Park, Los Angeles, CA 90089-0781, 1991. In preparation.

[3] R. V. Boppana and C. S. Raghavendra. Generalized schemes for access and alignment of data in parallel processors with self-routing interconnection networks. *J. of Parallel and Distributed Computing*, 11:97–111, 1991.

[4] P. Budnik and D. J. Kuck. The organization and use of parallel memories. *IEEE Trans. on Computers*, c-20(12):1566–1569, 1971.

[5] J. M. Frailong, W. Jalby, and J. Lenfant. XOR-Schemes: A flexible data organization in parallel memories. In *Proc. International Conf. on Parallel Processing*, pages 276–283, 1985.

[6] K. Hoffman and R. Kunze. *Linear Algebra*. Prentice-Hall, second edition, 1971.

[7] K. Kim and V. K. Prasanna Kumar. Perfect latin squares and parallel array access. In *Proc. International Symp. on Comput. Arch.*, pages 372–379, 1989.

[8] D. H. Lawrie. *Memory-Processor Connection Networks*. PhD thesis, University of Illinois, Urbana, 1973.

[9] D. H. Lawrie. Access and Alignement of Data in an Array Processor. *IEEE Trans. on Computers*, c-24(12), 1975.

[10] D.-L. Lee. Scrambled storage for parallel memory systems. In *Proc. International Symp. on Comput. Arch.*, pages 232–239, 1988.

[11] H. D. Shapiro. Theoretical limitations on the efficient use of parallel memories. *IEEE Trans. on Computers*, c-27(5):421–428, 1978.

[12] H. A. G. Wijshoff and J. V. Leeuwen. The structure of periodic storage schemes for parallel memories. *IEEE Trans. on Computers*, c-34(6):501–505, 1985.