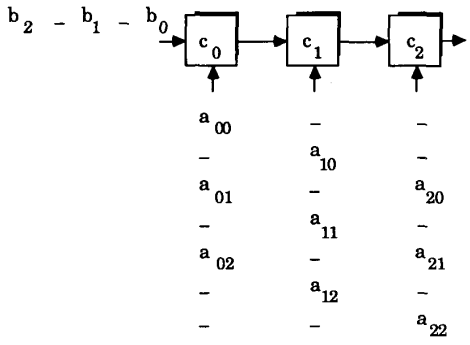

Fig. 7. The systolic array executing the matrix–vector multiplication by expanding the interval of the data items.

APPENDIX

Proof of Theorem 1: 1) The distance from the position at time t to the source is assumed to take q time steps for the data item with flow velocity $\vec{\Gamma}$, that is,

$$\boldsymbol{p}_t - \boldsymbol{p}_s = q\vec{\Gamma}.$$

or

$$\begin{bmatrix} i(t) \\ j(t) \\ k(t) \end{bmatrix} - \begin{bmatrix} i(s) \\ j(s) \\ k(s) \end{bmatrix} = q\vec{\Gamma}.$$

but the position vector $\boldsymbol{p}_t$ on the dependence graph is a multiple of the dependence vector plus the source, then

$$\left(r\vec{d} + \begin{bmatrix} i(s) \\ j(s) \\ k(s) \end{bmatrix} \right) - \begin{bmatrix} i(s) \\ j(s) \\ k(s) \end{bmatrix} = q\vec{\Gamma}.$$

Letting $c = r/q$ gives (4).

2) Measuring the time to go through the distance by the timing schedule Π, the time needed is q in 1), then by the above

$$q = \Pi(r\vec{d}) = \Pi(q\vec{\Gamma}) = q(\Pi \cdot \vec{\Gamma}),$$

and gives (5). By (4), substitute $\vec{d}$ with $c^{-1}\vec{\Gamma}$ into (2), it becomes

$$\Pi c^{-1}\vec{\Gamma} \geq 1.$$

Since $\Pi \cdot \vec{\Gamma} = 1$, $c \leq 1$. Q.E.D.

ACKNOWLEDGMENT

The authors are grateful to C.-M. Liu for his delightful help and the anonymous referees for their constructive suggestions.

REFERENCES

[1] S. Horiike, S. Nishida, and T. Sakaguchi, "Systematic design of systolic arrays using mapping algorithm," in *Proc. IEEE Int. Symp. Circuit Syst.*, 1988, pp. 2505–2508.
[2] H. V. Jagadish and T. Kailath, "A family of new efficient arrays for matrix multiplication," *IEEE Trans. Comput.*, vol. 38, pp. 149–155, Jan. 1989.
[3] H. V. Jagadish, S. K. Rao, and T. Kailath, "Array architectures for iterative algorithms," *Proc. IEEE*, vol. 75, pp. 1304–1321, Sept. 1987.
[4] R. M. Karp, R. E. Miller, and S. Winograd, "The organization of computers for uniform recurrence equations," *J. ACM*, vol. 14, pp. 563–590, July 1967.
[5] H. T. Kung and C. E. Leiserson, "Algorithms for VLSI processor arrays," in *Introduction to VLSI Systems*, C. Mead and L. Conway, Eds. Reading, MA: Addison-Wesley, 1980.
[6] C. L. Liu, *Introduction to Combinatorial Mathematics*, New York: McGraw-Hill, 1968.
[7] C. M. Liu and C. W. Jen, "Design of the algorithm-based fault-tolerant VLSI array processor," *IEE Proc.*, vol. 136, pt. E, pp. 539–547, Nov. 1989.
[8] D. I. Moldovan, "On the analysis and synthesis for VLSI algorithms," *IEEE Trans. Comput.*, vol. C-31, pp. 1121–1126, Nov. 1982.
[9] ——, "On the design of algorithms for VLSI systolic arrays," *Proc. IEEE*, vol. 71, pp. 113–120, Jan. 1983.
[10] P. Quinton, "Synthesizing systolic arrays using DIASTOL," in *Proc. Int. Workshop Systolic Arrays*, Univ. Oxford, July 1986, pp. 4.1–4.12.
[11] S. K. Rao and T. Kailath, "Regular iterative algorithms and their implementation on processor array," *Proc. IEEE*, vol. 76, pp. 259–269, Mar. 1988.
[12] J. C. Tsay and Y. C. Hou, "On equivalent systolic designs of matrix multiplication and its algebraic representation," in *Proc. Int. Comput. Symp.*, Taiwan, Dec. 1988, pp. 247–252.
[13] ——, "Generating function and equivalent transformation for systolic arrays," *Parallel Comput.*, vol. 10, pp. 347–356, May 1989.
[14] Y. Yaacoby and P. R. Cappello, "Scheduling a system of affine recurrence equations onto a systolic array," in *Proc. Int. Conf. Systolic Arrays*, 1988, pp. 373–382.

Some Combinatorial Aspects of Parallel Algorithm Design for Matrix Multiplication

Jong-Chuang Tsay and Sy Yuan

***Abstract*—In this paper, some combinatorial characteristics of matrix multiplication on regular two-dimensional arrays are studied. From the studies, we are able to design many efficient varieties of the cylindrical array and the two-layered mesh array for matrix multiplication. To design a cylindrical array for matrix multiplication, a systematic design procedure is proposed. In this design procedure, Latin square (a special type of matrix) plays an important role. To design a two-layered mesh array, we find that there is a transformation procedure to transform a cylindrical array to a two-layered mesh array.**

***Index Terms*—Cylindrical array, Latin square, matrix multiplication, parallel algorithm design, systolic array, two-layered mesh array.**

I. INTRODUCTION

Systolic array, as defined by Kung [3], is a synchronous operating parallel processing array consisting of simple processing elements interconnected in a local and regular manner. Systolic array has an unnoticed characteristic—its architecture should be planar, in other words, the communication lines among the processing elements should not cross over each other. If we relax this restriction, it is possible to obtain more efficient designs than systolic designs for a certain class of problems. Usually for this class of problems, operators used in their algorithm satisfy the associative and commutative properties. Typical examples in this class are matrix multiplication

Manuscript received April 28, 1989; revised November 11, 1990.

J.-C. Tsay is with the Institute of Computer Science and Information Engineering, College of Engineering, National Chiao Tung University, Hsinchu, Taiwan 30049, Republic of China.

S. Yuan is with the Advanced Technology Center of Computer and Communication Research Laboratories, Industrial Technology Research Institute, Chutung, Hsinchu, Taiwan 31015, Republic of China.

IEEE Log Number 9102596.

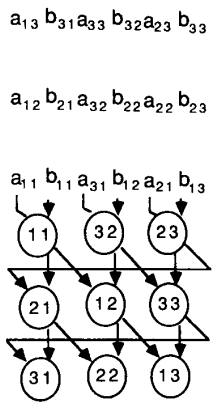

Fig. 1. A cylindrical array.

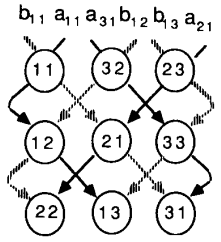

Fig. 2. A two-layered mesh array.

and convolution. For the matrix multiplication, most parallel algorithms are represented by graphs [1], [2], [4], [5], [7]–[9]. We study some combinatorial characteristics of the graph and extract their essential information, then formalize it on two tables—the *timing-level table* and the *processor assignment table.* Special characteristics of these tables enable us to design many efficient varieties of matrix multiplication algorithms which are executable on the cylindrical array [8] or the two-layered mesh array [1]. Fig. 1 and Fig. 2 are examples of these arrays, respectively.

II. Design of Cylindrical Arrays for Matrix Multiplication

The problem of matrix multiplication is to compute the product $\boldsymbol{C}$ of two matrices, $\boldsymbol{A}$ and $\boldsymbol{B}$. An equation to compute the product is

$$\boldsymbol{C} = \boldsymbol{A}_1\boldsymbol{B}_1 + \boldsymbol{A}_2\boldsymbol{B}_2 + \cdots + \boldsymbol{A}_n\boldsymbol{B}_n$$

where $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ are ith column and ith row of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively; and the product $\boldsymbol{A}_i\boldsymbol{B}_i$ denotes an "outer product." Therefore, the matrix multiplication can be carried out in n recursions (Each executes an outer product $\boldsymbol{A}_k\boldsymbol{B}_k$) [5],

$$c_{ij}^{(k)} = c_{ij}^{(k-1)} + a_{ik}b_{kj} \qquad \text{for } i, j, k = 1, 2, \cdots, n.$$

This recurrence equation leads to a dependence graph (DG) shown in Fig. 3, where each index point (or node) (i, j, k) in the index space $\{(i, j, k) | 1 \le i, j, k \le n\}$ of the DG performs the operation $c_{ij}^{(k)} = c_{ij}^{(k-1)} + a_{ik} * b_{kj}$. Since a_{ik} and b_{kj} are transmittent

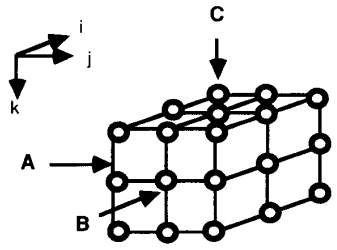

Fig. 3. Dependence graph of matrix multiplication.

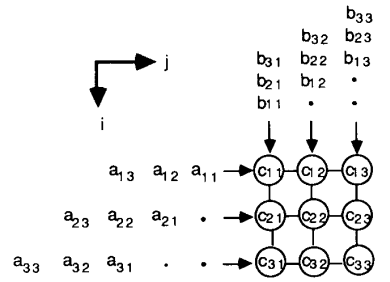

Fig. 4. A systolic algorithm for matrix multiplication.

data [5], their data flow direction can be reversed. Besides, since the addition operator is associative and commutative, data flow direction of c_{ij} can be reversed too. Consequently, the product of $a_{i1}b_{1j}, a_{i2}b_{2j}, \cdots, a_{in}b_{nj}$ can be added to c_{ij} in any order. Therefore, the recurrence equation can be generalized as

$$c_{ij}^{(k)} = c_{ij}^{(k-1)} + a_{il_k}b_{l_kj} \qquad \text{where } i, j, k = 1, 2, \cdots, n$$
$$\text{and } (l_1, l_2, \cdots, l_n) \text{ is a permutation of } (1, 2, \cdots, n).$$

On designing a parallel algorithm, there are two things we must determine—the timing schedule and the processor assignment of the algorithm. We determine the timing schedule of the algorithm first, then determine a processor assignment (or a projection direction) which is not conflicting with the timing schedule. The timing schedule and the processor assignment can be represented by a *timing-level table* (TLT) [7] and a *processor assignment table* (PAT), respectively. For example, the original systolic design, Fig. 4, proposed by Kung [4] to compute product of two $n \times n$ ($n = 3$) matrices A and B, can be described by a TLT and a PAT shown in Table I. In this design, [0 0 1] is the projection direction so that indexes i, j, and k in the tables corresponding to a *row*, a *column*, and a *layer*, respectively. In general, we use the indexes r, s, and q to represent a row, a column, and a layer in the tables, respectively. Assume that the index space is $I = \{(ijk) | 1 \le i, j, k \le n\}$. Depending on the projection directions, [0 0 1], [0 1 0], or [1 0 0], (rsq) is set to (ijk), (ikj), or (kji), respectively. The number t_{rsq} on position (r, s, q) of a TLT specifies that an operation $(c_{ij} = c_{ij} + a_{ik}b_{kj})$ is performed at time t_{rsq} and the number $p_{\alpha\beta}$ on position (α, β) of a PAT specifies that the above operation is performed by the processor (α, β), where $p_{\alpha\beta} = (r, s)$. In other words, all the nodes $\{(r, s, q) | q = 1, 2, \cdots, n\}$ of a DG are projected onto the same processor (α, β). For example, Table I specifies that the projection direction is [0 0 1] and the operation $c_{23} = c_{23} + a_{22}b_{23}$ is performed at time step $t_{rsq} = t_{ijk} = t_{232} = 5$ by the processor $(\alpha, \beta) = (2, 3)$.

In the following, we shall describe the problem of how to select appropriate values of t_{rsq} and $p_{\alpha\beta}$ so that varieties of parallel

TABLE I
DESIGN OF THE PARALLEL ALGORITHM, FIG. 4, CAN BE DESCRIBED BY (a) A TIMING-LEVEL TABLE AND (b) A PROCESSOR ASSIGNMENT TABLE

$j=$ / $i=1$	1	2	3
1	1	2	3
2	2	3	4
3	3	4	5

$k=1$

$j=$ / $i=1$	1	2	3
1	2	3	4
2	3	4	5
3	4	5	6

$k=2$

$j=$ / $i=1$	1	2	3
1	3	4	5
2	4	5	6
3	5	6	7

$k=3$

(a)

$\beta=$ / $\alpha=1$	1	2	3
1	11	12	13
2	21	22	23
3	31	32	33

(b)

algorithms or arrays for matrix multiplication can be obtained. Before this, we introduce some basic terminologies.

In the following description, we use the notations L, R, O, P, I, D, F, and S to represent various attributes of a matrix or a sequence. They are abbreviations of *left, right, ordered, permutating, increasing, decreasing, first, and second,* respectively. For example, we use $\langle L, O\rangle$ to represent a matrix whose attributes are left and ordered.

Definition 1: Let $D = \{1, 2, \cdots, n\}$, $\boldsymbol{u} = [u_0 u_1 u_2 \cdots u_{n-1}] \equiv [u_i]$ be a sequence of integers belonging to D, and $u_i \neq u_j$ for $i \neq j$. A left (right) cyclic shift of $\boldsymbol{u}$ is defined as $\sigma\boldsymbol{u} = u_1 u_2 \cdots u_{n-1} u_0$ ($\sigma^{-1}\boldsymbol{u} = u_{n-1} u_0 u_1 \cdots u_{n-2}$). Thus, shift $\boldsymbol{u}$ right cyclically m times can be represented as $\sigma^{-m}\boldsymbol{u} = u_{n-m} u_{n-m+1} \cdots u_{n-1} u_0 u_1 \cdots u_{n-m-1}$, where $m \leq n$.

Definition 2: A sequence $\boldsymbol{u}$ of length n is *increasing* if it can be expressed by $\boldsymbol{u} = \sigma^l[12\cdots n]$, $l = 0, 1, 2, \cdots,$ or $n-1$. It is a *decreasing* sequence if it can be expressed by $\boldsymbol{u} = \sigma^l[nn-1\cdots 1]$.

Definition 3: An $n \times n$ *Latin square* (or a Latin square of *order n*) is an $n \times n$ matrix that has the numbers $1, 2, 3, \cdots, n$ as entries such that no number appears more than once in the same row or the same column [6].

Definition 4: Given a Latin square $\boldsymbol{L} = [l_{ij}]$ of order n, such that for each $p = 1, 2, \cdots, n-1$, $[l_{p+1,j}] = \sigma^{\varrho}[l_{p,j}]$ $\forall j$, where ϱ is either $+1$ or -1; we say $\boldsymbol{L}$ is an $\langle L\rangle$ Latin square (each row is a left rotation of its preceding row) if $\varrho = +1$, a $\langle R\rangle$ Latin square if $\varrho = -1$.

Definition 5: A Latin square $\boldsymbol{L} = [l_{ij}]$ of order n is an $\langle O\rangle$ Latin square if it is an $\langle L\rangle$ Latin square or a $\langle R\rangle$ Latin square and $[l_{1j}]$ is an increasing or a decreasing sequence. If $[l_{1j}]$ is increasing, $\boldsymbol{L}$ is an $\langle I\rangle$ Latin square. If $[l_{1j}]$ is decreasing, $\boldsymbol{L}$ is an $\langle D\rangle$ Latin square. For example, an $\langle I, R\rangle$ Latin square is an $\langle O\rangle$ Latin square which is increasing and right. We say $\boldsymbol{L}$ a $\langle P\rangle$ Latin square if it can be obtained by interchanging rows or columns of an $\langle O\rangle$ Latin square.

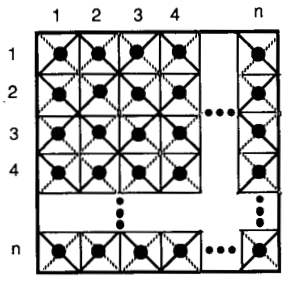

Fig. 5. Pattern of x-square (when n is even).

Definition 6: Two squares (or matrices) of order n, $\boldsymbol{M}_1 = [m_{ij}^{(1)}]$ and $\boldsymbol{M}_2 = [m_{ij}^{(2)}]$ can be superimposed to form a *composite square* $\boldsymbol{G} = [g_{ij}] = [(m_{ij}^{(1)}, m_{ij}^{(2)})]$, where g_{ij} is an ordered pair $(m_{ij}^{(1)}, m_{ij}^{(2)})$ (i.e., $g_{ij}(1) = m_{ij}^{(1)}$, and $g_{ij}(2) = m_{ij}^{(2)}$).

Definition 7: A composite square $\boldsymbol{G}$, composed from two squares of order n, $\boldsymbol{M}_1 = [m_{ij}^{(1)}]$ and $\boldsymbol{M}_2 = [m_{ij}^{(2)}]$, is a *z-square* if it satisfies (either $(x, y) = (1, 2)$ or $(x, y) = (2, 1)$)

a) $[m_{1j}^{(x)}]$ is a permutation of $(1, 2, \cdots, n)$,
b) for each $p = 1, 2, \cdots, n-1$, $m_{p+1,j}^{(x)} = m_{p,j}^{(x)}$ $\forall j$, and
c) $\boldsymbol{M}_y$ is an $\langle O\rangle$ Latin square. If $(x, y) = (1, 2)$, we say $\boldsymbol{G}$ is an $\langle S\rangle$ z-square; otherwise, it is an $\langle F\rangle$ z-square. Thus, an $\langle F, R\rangle$ z-square is a composite square whose first component is an ordered Latin square of right rotation type.

Definition 8: A composite square $\boldsymbol{G}$, composed from two squares of order n, $\boldsymbol{M}_1 = [m_{ij}^{(1)}]$ and $\boldsymbol{M}_2 = [m_{ij}^{(2)}]$, is an *x-square* if it satisfies (either $(x, y) = (1, 2)$ or $(x, y) = (2, 1)$)

a) Both of $[m_{1j}^{(1)}]$ and $[m_{1j}^{(2)}]$ are permutations of $(1, 2, \cdots, n)$,
b) for an even (odd) integer v, $1 \leq v \leq n$, all elements of $m_{1+d,f(v,d)}^{(x)}(m_{1+d,f(v,d)}^{(y)})$, $d = 0, 1, \cdots, n-1$ are equal, where if $v + d \leq n$, $f(v, d) = v + d$; otherwise $f(v, d) = 2n + 1 - (v + d)$,
c) for an odd (even) integer v, $1 \leq v \leq n$, all elements of $m_{1+d,g(v,d)}^{(y)}(m_{1+d,g(v,d)}^{(x)})$, $d = 0, 1, \cdots, n-1$ are equal, where if $v - d \geq 1$, $g(v, d) = v - d$; otherwise $g(v, d) = 1 - (v - d)$.

The above conditions a) and b) are illustrated in Fig. 5, where all elements of $\boldsymbol{M}_x(\boldsymbol{M}_y)$ on each solid-line (dash-line) paths are equal.

Next, we shall describe the design of cylindrical arrays [8] for matrix multiplication. First, we determine the timing schedule TLT, then determine a processor assignment PAT which is compatible (or not conflicting) with the TLT. The following lemma gives the conditions to be satisfied for a PAT to be compatible. In the following, we assume, without loss of generality, that $t_{rs1} \leq t_{rs2} \leq t_{rs3}$ for $1 \leq r, s \leq n$.

Lemma 1: On designing a cylindrical array, assume that we have correctly constructed a TLT ($\boldsymbol{T} = [t_{rsq}]$) of size $n \times n \times n$. Consider a PAT ($\boldsymbol{P} = [p_{\alpha\beta}]$) of size $n \times n$. If $\boldsymbol{P}$ is compatible with $\boldsymbol{T}$, then it should satisfy the condition that if $t_{rs1} = \alpha$ then $\exists\beta \ni p_{\alpha\beta} = (r, s)$ for $1 \leq r, s \leq n$.

Proof: For a cylindrical array, since the data are fed into the array from the first row of the array, computations of the processors in the first row (i.e., processors on positions $(1, j)$, $j = 1, 2, \cdots, n$) start one step earlier than the processors in the second row. In general, the processors in the ith row start one step later than the processors

in the $(i-1)$th row, for $i = 2, 3, \cdots, n$. Therefore, the processors on the αth row start computing at time step α. If $t_{rs1} = \alpha$, then the node $(rs1)$ should be executed at time step α. Hence, the element (r, s) should be placed in the αth row of the PAT. In other words, $\exists \beta \ni p_{\alpha\beta} = (r, s)$. □

An immediate consequence of this lemma is that $t_{rs1} = \alpha$ if $p_{\alpha\beta}(1) = r$ and $p_{\alpha\beta}(2) = s$.

Based on the lemma, we describe a design procedure for constructing a TLT and a PAT for matrix multiplication:

1) construct a TLT: Elements of the TLT should satisfy the following features:
   a) $[t_{rs1}]$ in the TLT is an $\langle O \rangle$ (ordered) Latin square.
   b) $[t_{rsq}] = [t_{rs1} + q - 1]$ for $q = 2, 3, \cdots, n$.
2) Given a TLT constructed above, construct the corresponding PAT as follows:
   a) Determine p_{11} of the PAT: Let Q be a set of all coordinates (x, y) which satisfy $t_{xy1} = 1$ in the TLT, then p_{11} can be chosen from any one coordinate of Q.
   b) Determine first column of the PAT: When the attribute of either an $\langle F \rangle$ or $\langle S \rangle$ has been selected for the PAT, using Lemma 1 and $[t_{xy1}]$, we can determine $[p_{\alpha 1}]$, $\alpha = 2, 3, \cdots, n$.
   c) Determine the PAT: Set the PAT to be either a $\langle R \rangle$ or $\langle L \rangle$ z-square. By using Definition 7, $[p_{\alpha\beta}]$ $\forall \alpha, \beta$ can be determined.

Note that the constructed PAT is a z-square of type $\langle F, R \rangle$, $\langle F, L \rangle$, $\langle S, R \rangle$, or $\langle S, L \rangle$.

The construction of a TLT is not difficult. But the construction of a PAT is not easy. It needs to be further described. We use a TLT whose $[t_{rs1}]$ is an $\langle I, R \rangle$ Latin square as an example. (For other cases, when $[t_{rs1}]$ is an $\langle I, L \rangle$, $\langle D, R \rangle$, or $\langle D, L \rangle$ ordered Latin square, there are similar construction procedures).

1) Determine the value of p_{11}: we select any coordinate (x, y) which satisfies $t_{xy1} = 1$ as the value of p_{11}.
2) Determine first column of the PAT: $[t_{rs1}]$ is $\langle I, R \rangle$ and $t_{xy1} = 1$, so we know $[t_{rs1}]$. If $\langle F \rangle$ is selected for the PAT, then $p_{n1}(2) = p_{n-1,1}(2) = \cdots = p_{21}(2) = p_{11}(2) = y$. Using Lemma 1, it can be deduced that $p_{i1} = ((x - i + 1) \bmod n, y)$ for $i = 2, 3, \cdots, n$, where $h \bmod n \equiv h \bmod n$ if $h \notin \{-n, 0, n\}$; otherwise $h \bmod n \equiv n$. If the PAT is selected to be $\langle S \rangle$, then $p_{n1}(1) = p_{n-1,1}(1) = \cdots = p_{11}(1) = x$. Using Lemma 1, we have $p_{i1} = (x, (y + i - 1) \bmod n)$.
3) Determine the PAT:
   a) If the PAT is selected to be an $\langle F, R \rangle$ z-square, then
   $$p_{ij} = ((x - i + j) \bmod n, (y + j - 1) \bmod n) \quad (1)$$
   b) If the PAT is selected to be an $\langle F, L \rangle$ z-square, then
   $$p_{ij} = ((x - i - j + 2) \bmod n, (y - j + 1) \bmod n) \quad (2)$$
   c) If the PAT is selected to be an $\langle S, R \rangle$ z-square, then
   $$p_{ij} = ((x - j + 1) \bmod n, (y + i - j) \bmod n) \quad (3)$$
   d) If the PAT is selected to be an $\langle S, L \rangle$ z-square, then
   $$p_{ij} = ((x + j - 1) \bmod n, (y + i + j - 2) \bmod n). \quad (4)$$

Lemma 2: The constructed TLT and PAT implement matrix multiplication in a cylindrical array.

Proof: If the projection direction is [0 0 1], then matrix $\boldsymbol{A}$ and matrix $\boldsymbol{B}$ are fed into the array, and the value of matrix $\boldsymbol{C}$ is stayed in the array. If the projection direction is [1 0 0], then matrix $\boldsymbol{A}$ and matrix $\boldsymbol{C}$ are fed into the array and the value of matrix $\boldsymbol{B}$ is stayed in the array. If the projection direction is [0 1 0], then matrix $\boldsymbol{B}$ and matrix $\boldsymbol{C}$ are fed into the array and the value of matrix $\boldsymbol{A}$ is stayed in the array. Assume that the projection direction is [0 0 1] and the PAT is an $\langle F, R \rangle$ z-square. Using (1), we see that $p_{11}, p_{12}, \cdots, p_{1n}$ are $((x, y), ((x + 1) \bmod n, (y + 1) \bmod n), \cdots, ((x + n - 1) \bmod n, (y + n - 1) \bmod n)$, respectively. Let the input data, $(a_{(x+j-1) \bmod n, t}, b_{t,(y+j-1) \bmod n})$, be arranged so that it is fed into the processor PE_{1j} for each j $(j = 1, 2, \cdots, n)$ at time step t $(t = 1, 2, \cdots, n)$.

Because we assume that the PAT is an $\langle F, R \rangle$ z-square, at time step $\tau = t + \alpha, 0 \le \alpha \le n - 1$, a's data fed into $(\alpha + 1)$th row of the array aremius9pt

$$\begin{aligned}\left(a_{\sigma-\alpha \boldsymbol{u}(1), t}, a_{\sigma-\alpha \boldsymbol{u}(2), t}, \cdots, a_{\sigma-\alpha \boldsymbol{u}(n), t}\right) \\ = (a_{(x-\alpha) \bmod n, t}, a_{(x+1-\alpha) \bmod n, t}, \cdots, \\ a_{(x+n-1-\alpha) \bmod n, t}) \quad (5)\end{aligned}$$

where $\boldsymbol{u} = [x, (x + 1) \bmod n, \cdots, (x + n - 1) \bmod n]$ is a sequence.

Since b's data move down along a vertical path, hence at time step τ,

$$\left(b_{ty}, b_{t,(y+1) \bmod n}, \cdots, b_{t,(y+n-1) \bmod n}\right) \quad (6)$$

arrive $(\alpha + 1)$th row of the array.

Furthermore, by (1), we find the data stayed in the $(\alpha + 1)$th row of the array are $c_{p_{\alpha+1,j}}, 1 \le j \le n$, i.e.,

$$\begin{aligned}(c_{(x-\alpha) \bmod n, y}, c_{(x+1-\alpha) \bmod n, (y+1) \bmod n}, \cdots, \\ c_{(x+n-1-\alpha) \bmod n, (y+n-1) \bmod n}). \quad (7)\end{aligned}$$

It follows from (5)–(7), $a_{(x-1-\alpha+j) \bmod n, t}, b_{t,(y+j-1) \bmod n}$, and $c_{(x-1-\alpha+j) \bmod n, (y+j-1) \bmod n}$, $1 \le j \le n$, interact properly at $\mathrm{PE}_{(\alpha+1),j}$ at time step $\tau = t + \alpha$. Hence, we conclude that the parallel array is a cylindrical array for matrix multiplication. For the other cases, when the constructed PAT is an $\langle F, L \rangle$, $\langle S, R \rangle$ or $\langle S, L \rangle$ type, the proof is similar. □

In the above constructions of $[t_{rsq}]$ and $[p_{\alpha\beta}]$, if we allow two columns (or rows) of $[t_{rs1}]$ (the $[p_{\alpha\beta}], [t_{rs2}], \cdots$, and $[t_{rsn}]$ should be updated accordingly) to be exchanged, then, based on the following lemmas, other cylindrical arrays can be designed. Now we prove it.

Lemma 3: Assume that $\boldsymbol{P}$ and $\boldsymbol{T}$ are, respectively, a PAT and a TLT which have been constructed by the design procedure. If $\boldsymbol{T}'$ is a TLT which is obtained by interchanging rows u and v $(1 \le u, v \le n)$ of $\boldsymbol{T}$, then it is possible to find a PAT $\boldsymbol{P}'$ compatible with $\boldsymbol{T}'$ such that $\boldsymbol{P}'$ is a z-square.

Proof: Since $\boldsymbol{T}'$ is obtained by interchanging rows u and v, we have $t_{uy1} = \alpha$ and $t_{vy1} = w$ iff $t'_{uy1} = w$ and $t'_{vy1} = \alpha$. Without loss of generality, assume that $\boldsymbol{P}$ is a z-square of $\langle F, R \rangle$ type. If we construct $\boldsymbol{P}'$ from $\boldsymbol{P}$ by interchanging all $p_{\alpha\beta}(1)$ which have value u with all $p_{\alpha\beta}(1)$ which have value v, then we have (the interchanging operations do not alter $p_{\alpha\beta}(2)$) $p'_{\alpha\beta}(1) = u$ if $p_{\alpha\beta}(1) = v$; $p'_{\alpha\beta}(1) = v$ if $p_{\alpha\beta}(1) = u$; and $p'_{\alpha\beta}(2) = p_{\alpha\beta}(2)$. Since $\boldsymbol{P}$ is compatible with $\boldsymbol{T}$, by Lemma 1, we have

$$\begin{aligned}t_{uy1} &= \alpha \quad \text{if } p_{\alpha\beta}(1) = u \quad \text{and} \quad p_{\alpha\beta}(2) = y; \\ t_{vy1} &= w \quad \text{if } p_{wb}(1) = v \quad \text{and} \quad p_{wb}(2) = y.\end{aligned}$$

In other words,

$$\begin{aligned}t'_{vy1} &= \alpha \quad \text{if } p'_{\alpha\beta}(1) = v \quad \text{and} \quad p'_{\alpha\beta}(2) = y; \\ t'_{uy1} &= w \quad \text{if } p'_{wb}(1) = u \quad \text{and} \quad p'_{wb}(2) = y.\end{aligned}$$

Moreover, all elements of $[p'_{\alpha\beta}]$ are distinct. Therefore, $\boldsymbol{P}'$ is compatible with $\boldsymbol{T}'$. Moreover, since $p'_{\alpha\beta}(2) = p_{\alpha\beta}(2)$ and all the $p'_{\alpha\beta}(1)$ on a southeast diagonal path are constant, we conclude that $\boldsymbol{P}'$ is a z-square of $\langle F, R\rangle$ type.

Lemma 4: Assume that $\boldsymbol{P}$ and $\boldsymbol{T}$ are, respectively, a PAT and a TLT which have been constructed by the design procedure. If $\boldsymbol{T}'$ is a TLT which is obtained by interchanging columns u and v $(1 \le u, v \le n)$ of $\boldsymbol{T}$, then it is possible to find a PAT $\boldsymbol{P}'$ compatible with $\boldsymbol{T}'$ such that $\boldsymbol{P}'$ is a z-square.

Proof: Since $\boldsymbol{T}'$ is obtained by interchanging columns u and column v, we have $t_{xu1} = \alpha$ and $t_{xv1} = w$ iff $t'_{xu1} = w$ and $t'_{xv1} = \alpha$. Without loss of generality, assume that $\boldsymbol{P}$ is a z-square of $\langle F, R\rangle$ type. If we construct $\boldsymbol{P}'$ from $\boldsymbol{P}$ by interchanging all $p_{\alpha\beta}(2)$ which have value u with all $p_{\alpha\beta}(2)$ which have value v, then we have $p'_{\alpha\beta}(2) = v$ if $p_{\alpha\beta}(2) = u$; $p'_{\alpha\beta}(2) = u$ if $p_{\alpha\beta}(2) = v$; and $p'_{\alpha\beta}(1) = p_{\alpha\beta}(1) = v$. Since $\boldsymbol{P}$ is compatible with $\boldsymbol{T}$, by Lemma 1, we have

$$t_{xu1} = \alpha \quad \text{if } p_{\alpha\beta}(1) = x \quad \text{and} \quad p_{\alpha\beta}(2) = u;$$
$$t_{xv1} = w \quad \text{if } p_{wb}(1) = x \quad \text{and} \quad p_{wb}(2) = v.$$

In other words,

$$t'_{xv1} = \alpha \quad \text{if } p'_{\alpha\beta}(1) = x \quad \text{and} \quad p'_{\alpha\beta}(2) = v;$$
$$t'_{xu1} = w \quad \text{if } p'_{wb}(1) = x \quad \text{and} \quad p'_{wb}(2) = u.$$

Moreover, all elements of $[p_{\alpha\beta}]$ are distinct. Therefore, $\boldsymbol{P}'$ is compatible with $\boldsymbol{T}'$. Moreover, since $p'_{\alpha\beta}(1) = p_{\alpha\beta}(1)$ and all the $p'_{\alpha\beta}(2)$ on a vertical path are constant, we conclude that $\boldsymbol{P}'$ is a z-square of $\langle F, R\rangle$ type. □

From Lemma 3 and Lemma 4, we have the following theorem.

Theorem: Starting from a Latin square of either $\langle O\rangle$ or $\langle P\rangle$ type, it is possible to construct a TLT $\boldsymbol{T}$ and find a PAT $\boldsymbol{P}$ compatible with $\boldsymbol{T}$ such that $\boldsymbol{P}$ is a z-square. Thus, by Lemma 2, a cylindrical array for matrix multiplication corresponding to the $\boldsymbol{P}$ and $\boldsymbol{T}$ can be constructed.

Now, we give some illustrative examples. In Table II, $[t_{rs1}]$ is an ordered Latin square of $\langle I, L\rangle$ type; $[t_{rs2} - 1] = [t_{rs3} - 2] = [t_{rs1}]$; and $[p_{\alpha\beta}]$ is an $\langle F, R\rangle$ z-square. If projection direction $\vec{d} = [001]$ is selected (i.e., $r \equiv i, s \equiv j$), then it has a corresponding parallel algorithm run on the cylindrical array of Fig. 1. In this design c_{ij} stays in the array. If $\vec{d} = [0\ 1\ 0]$ is selected, then a cylindrical array, where a_{ik} stays in the array can be obtained. There are other compatible z-squares which can be used for the PAT. Table III lists two examples. Table III(a) is an $\langle F, R\rangle$ z-square with $p_{11} = (3, 2)$. Table III(b) is an $\langle S, L\rangle$ z-square with $p_{11} = (1\ 1)$. Since three different projection directions can be selected, each z-square corresponds to three different designs of parallel algorithms.

III. Design of Two-Layered Mesh Arrays for Matrix Multiplication

Given a z-square $\boldsymbol{S}$ ($n \times n$ matrix), there is a method to transform the z-square to an x-square. Before we describe the method, we need to define an m-times transpositional network first.

Definition 9: An *m-times odd–even transpositional network* for n numbers is an m-level network having $m \times \lfloor \frac{n}{2} \rfloor$ $(m \times \frac{n}{2} - \lfloor \frac{m}{2} \rfloor)$ exchangers when $n =$ odd (even). Each exchanger accepts two input numbers and switches their positions. They are arranged in a brick-like pattern in the network. If the n numbers are fed downward, then the exchangers are placed at the odd locations on the odd rows and placed at the even locations on the even rows. For example, Fig. 6(a) shows a 3–times odd–even transpositional network for five numbers. Similarly, an *m-times even–odd transpositional network* can be defined. Fig. 6(b) is an example $(m = 3, n = 5)$.

TABLE II
(a) A Timing-Level Table. (b) A Processor Assignment Table

$s =$	1	2	3
$r = 1$	1	2	3
2	2	3	1
3	3	1	2

$q = 1$

$s =$	1	2	3
$r = 1$	2	3	4
2	3	4	2
3	4	2	3

$q = 2$

$s =$	1	2	3
$r = 1$	3	4	5
2	4	5	3
3	5	3	4

$q = 3$

(a)

$\beta =$	1	2	3
$\alpha = 1$	11	32	23
2	21	12	33
3	31	22	13

(b)

TABLE III
Two z-Squares Compatible with Table II(a)

32	23	11
12	33	21
22	13	31

(a)

11	32	23
12	33	21
13	31	22

(b)

The procedure to transform a z-square $\boldsymbol{S}$ to an x-square is now given below (we name leftmost column as the first column):

1) For $n \ge 4$, insert the first column of $\boldsymbol{S}$ between the $(n-2)$th and $(n-1)$th columns, and insert the second column between the $(n-3)$th and $(n-2)$th columns, $\cdots$ until the $(\lfloor n/2 \rfloor - 1)$th column is inserted between the $(\lceil n/2 \rceil)$th and the $(\lceil n/2 \rceil + 1)$th columns. After this step, we obtain a new square $\boldsymbol{S}'$. When $n = 3$, we let $\boldsymbol{S}' = \boldsymbol{S}$.
2) the mth row of $\boldsymbol{S}'$ (where $m = 2, 3, \cdots, n$) is rearranged by the $(m-1)$-times transpositional network. Depending on the value of n and the type of the z-square $\boldsymbol{S}$, either the odd–even (even–odd) transpositional network should be selected (see Table IV). After all rows are processed, we obtain an x-square.

For example, after we apply the transformation procedure, a z-square of Table II will be transformed to an x-square of Table V. When we choose the projection direction [0 0 1], it has a corresponding design of two-layered mesh array [2] shown in Fig. 2.

Another example is given in Table VI. Table VI(a) is a 4×4 $\langle I, R\rangle$ Latin square which can be used for the $[t_{rs1}]$ of a TLT. Table VI(b) is an $\langle S, L\rangle$ z-square which is a PAT compatible with the TLT.

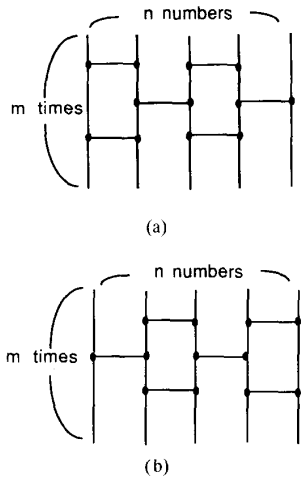

Fig. 6. (a) An m-times odd–even transpositional network on n numbers ($m = 3, n = 5$). (b) An m-times even–odd transpositional network on n numbers ($m = 3, n = 5$).

TABLE IV
SELECTION OF EVEN–ODD OR ODD–EVEN TRANSPOSITIONAL NETWORK

n	type of S	
	right-rotation	left-rotation
odd	odd-even	even-odd
even	even-odd	odd-even

TABLE V
A PROCESSOR ASSIGNMENT TABLE FOR THE TWO-LAYERED MESH ARRAY FIG. 2

11	32	23
12	21	33
22	13	31

Table VI(c) is an x-square derived from the z-square. The x-square is a feasible PAT for designing a two-layered mesh arrays.

If $[t_{rs1}]$ in the TLT is not an ordered Latin square or cannot be transformed from an ordered Latin square by interchanging rows or columns, then we cannot find its corresponding z-square or x-square. Table VII is an example of this type.

IV. CONCLUSION

Some combinatorial characteristics of parallel algorithms for matrix multiplication on regular two-dimensional arrays are studied. Studying its characteristics, we are able to design different parallel arrays, such as the cylindrical array, or the two-layered mesh array. Intuitively, we conjecture that the design procedure can be used to construct all the cylindrical arrays (of the form shown in Fig. 1) for matrix multiplication. From a given cylindrical array, we have described a transformation procedure which can be used to transform the cylindrical array to a two-layered mesh array. Finally, it is worthy to note that almost all the matrix multiplication algorithms designed in this paper use nonlinear timing schedules. This indicates that the design of matrix multiplication algorithms with nonlinear timing schedule can be formalized.

TABLE VI
(a) A 4×4 ORDER $\langle I, R \rangle$ LATIN SQUARE. (b) AN $\langle S, L \rangle$ z-SQUARE. (c) AN x-SQUARE

$s =$

$r =$	1	2	3	4
1	1	2	3	4
2	4	1	2	3
3	3	4	1	2
4	2	3	4	1

(a)

$\beta =$

$\alpha =$	1	2	3	4
1	11	22	33	44
2	12	23	34	41
3	13	24	31	42
4	14	21	32	43

(b)

$\beta =$

$\alpha =$	1	2	3	4
1	22	11	33	44
2	12	23	41	34
3	13	42	24	31
4	43	14	32	21

(c)

TABLE VII
A LATIN SQUARE WHICH HAS NO CORRESPONDING z-SQUARE OR x-SQUARE

1	3	2	4
2	4	1	3
4	1	3	2
3	2	4	1

REFERENCES

[1] S. C. Kak, "Multilayered array computing," in *Proc. 20th Annu. Conf. Inform. Sci. Syst.*, Princeton, 1986, pp. 436–441.
[2] ——, "A two-layered mesh array for matrix multiplication," *Parallel Comput.*, pp. 383–385, 1986.
[3] H. T. Kung, "Why systolic architectures?," *IEEE Comput. Mag.*, vol. 15, pp. 37–46, Jan. 1982.
[4] S. Y. Kung, "On supercomputing with systolic/wavefront array processors," *Proc. IEEE*, vol. 72, pp. 867–884, July 1984.
[5] ——, *VLSI Array Processor*. Englewood Cliffs, NJ: Prentice-Hall, 1988, ch. 3.
[6] C. L. Liu, *Introduction to Combinatorial Mathematics* New York: McGraw-Hill, 1968, ch. 11.

[7] Y. J. Ma, J. F. Wang, and J. Y. Lee, "Systolic array mapping of sequential algorithm for VLSI architecture," in *Proc. Int. Comput. Symp.*, Tainan, Taiwan, R.O.C., 1986, pp. 865–874.
[8] W. A. Porter and J. L. Aravena, "Cylindrical arrays for matrix multiplication," in *Proc. 24th Annu. Allernton Conf. Commun., Contr. and Comput.*, Monticello, 1986, pp. 595–602.
[9] W. A. Porter and J. L. Aravena, "Orbital architectures with dynamic reconfiguration," *Proc. IEE*, vol. 134, pt. E, no. 6, pp. 281–287, Nov. 1987.

PPMB: A Partial-Multiple-Bus Multiprocessor Architecture with Improved Cost-Effectiveness

Hong Jiang and Kenneth C. Smith

***Abstract*—This paper addresses the design and performance analysis of partial-multiple-bus interconnection networks. They are bus architectures that have evolved from multiple-bus structure by dividing buses into groups and reducing bus connections. Their effect is to reduce cost and alleviate arbitration and drive requirements without degrading performance significantly. One such structure, called processor-oriented partial-multiple-bus (or PPMB), is proposed. It serves as an alternative to the conventional structure called memory-oriented partial-multiple-bus (or MPMB) and is aimed at higher system performance at less or equal system cost. It has been shown, both analytically and by simulation, that a substantial increase in system bandwidth (up to 20%) is achieved by the PPMB structure over the MPMB structure. With very large systems, the results also imply a significantly improved cost-effectiveness over the conventional multiple-bus architecture.**

***Index Terms*—Cost-effectiveness, interconnection network, load-balancing arbitration, multiprocessor architecture, partial multiple-bus structures, performance evaluation.**

I. Introduction

Due to their reliability and cost-effectiveness, multiple-bus structures have assumed considerable importance in both research on, and applications of, interconnection networks in the multiprocessor arena. As a result, a great deal of work has been done in the performance analysis of multiple-bus systems. Such analysis shows that among the three major categories of interconnection networks (i.e., crossbar networks, multistage networks, and multiple-bus networks), multiple-bus structures are the most reliable and, under certain circumstances, the most cost effective [1]–[3], [5], [6], [8]. Nevertheless, multiple-bus structures might still be too costly for very large systems, due to the arbitration and drive requirements they entail.

Lang *et al.* [6] proposed, based on the conventional multiple-bus structure, a new network structure called a partial multiple-bus. The motivation for proposing the new structure was to reduce the cost of the system while trading off an acceptable and tolerable degree of performance degradation. This structure is derived from a conventional multiple-bus structure by dividing memory modules and buses into identical parts (or groups) while maintaining the connection of each processor to every bus. This partial-multiple-bus structure is shown in Fig. 1. As shown in [6], the performance degradation of a partial-multiple-bus is not significant. For a two-group partial-multiple-bus system of size 16 (i.e., $N = M = 16$, where N is the number of processors and M the number of memory modules), the decrease in performance (system bandwidth) is below 6%. For the sake of simplicity and consistency, we shall call this structure memory-oriented partial-multiple-bus, or MPMB.

A different partial multiple-bus structure is proposed in this paper as an alternative to the one proposed by Lang, and as one which provides higher system bandwidth and faster arbitration at lower or equal cost. Derived also from the conventional multiple-bus structure, this structure, called processor-oriented partial multiple-bus, or PPMB, divides processors and buses into identical groups while maintaining the connection of each memory module to every bus.

A notable difference between this structure and the one by Lang is that in it, a memory module has a maximum of B potential paths (where B is the number of buses) to processors while, in Lang's, a memory module has a maximum of only B/g potential paths to processors (where g is the number of groups of buses). This structural difference gives rise to a distinguishing feature of the PPMB structure, namely of having potential for load-balancing arbitration. Load balancing, aimed at fully exploiting the potential for higher bandwidth inherent in the structure, is able to provide a substantial improvement in system performance. As a matter of fact, analytical and simulation results have both shown a maximum of 20% increase in system bandwidth of the PPMB over MPMB. Meanwhile, the cost of a PPMB system has been shown in general to be less than or equal to that of an MPMB of the same size. Note that while the partial-multiple-bus structure, proposed by Lang, was motivated to reduce cost and arbitration time without reducing system bandwidth significantly, we have shown as well that the PPMB structure can lead to a substantial improvement in cost-effectiveness when system size is very large.

In the section that follows, details of the PPMB structure and its load-balancing feature are discussed on a comprehensive basis. Section III introduces probabilistic models for evaluating synchronous-system bandwidth of the structures under study and comparisons are made between PPMB and MPMB. The numerical results produced by them all lie within $\pm 3\%$ of the results of simulation, implying a high level of confidence in the models. Finally, some concluding remarks are given in Section IV.

II. Processor-Oriented Partial Multiple-Bus Structure (PPMB)

A. The Structure

In PPMB, shown in Fig. 2, N processors are divided into g groups with each group of (N/g) processors *fully* connected to a set of (B/g) buses, whereas all M memory modules are connected to all B buses. This is to be contrasted with MPMB in which the M memory modules are divided into g groups where each group of (M/g) memory modules is *fully* connected to a set of (B/g) buses, and all of the N processors are *fully* connected to all buses. For both MPMB and PPMB, g is assumed to be a factor of both B and M (or N).

In the rest of this paper on the study, we will refer to an $N \times M \times B/g$ system as a partial multiple-bus system that has B buses, M memory modules, N processors, and is divided into g groups. In addition, we will replace the notation M/g, N/g, and B/g with MG, NG, and BG, respectively.

Manuscript received May 10, 1989; revised October 23, 1990.
H. Jiang was with the Department of Computer Science, Texas A&M University, College Station, TX 77843. He is now with the Department of CS&E, University of Nebraska, Lincoln, NE 68588.
K. C. Smith is with the Department of Electrical Engineering and Computer Science, University of Toronto, Toronto, Ont., M5S 1A4 Canada.
IEEE Log Number 9102592.