[14] D. Klappholz, K. Psarris, and X. Kong, "On the perfect accuracy of an approximate subscript analysis test," in *Proc. 1990 Int. Conf. Supercomput.*, Nov. 1990, pp. 201–212.
[15] D. Levine, D. Callahan, and J. Dongarra, "A comparative study of automatic vectorizing compilers," Argonne Nat. Lab., Tech. Rep., MCS-P218-0391, Apr. 1991.
[16] Z. Li, "Array privatization for parallel execution of loops," in *Proc. 1992 Int. Conf. Supercomput.*, July 1992, pp. 313–322.
[17] D. E. Maydan, S. P. Amarasinghe, and M. S. Lam, "Data dependence and data-flow analysis of arrays," in *5th Workshop Languages and Compilers for Parallel Comput.*, also in Yale Univ., Tech. Rep. YALEU/DCS/RR-915, pp. 283–292, Aug. 1992.
[18] ——, "Array data-flow analysis and its use in array privatization," in *ACM '93 Conf. Principles of Programming Languages*, Jan. 1993.
[19] D. E. Maydan, "Accurate analysis of array references," Ph.D. dissertation, Comput. Syst. Lab., Stanford Univ., Sept. 1992.
[20] K. S. McKinley, "Dependence analysis of arrays subscripted by index arrays," Dept. of Comput. Sci., Rice Univ., Tech. Rep. RICE COMP TR91-162, Dec. 1990.
[21] D. Oppen, "A $2^{2^{2^{pn}}}$ upper bound on the complexity of presburger arithmetic," *J. Comput. Syst. Sci.*, vol. 16, no. 3, pp. 323–332, July 1978.
[22] W. Pugh, "The Omega test: A fast and practical integer programming algorithm for dependence analysis," *Commun. ACM*, vol. 8, pp. 102–114, Aug. 1992.
[23] ——, "Definitions of dependence distance," *Lett. Programming Languages Syst.*, Sept. 1993.
[24] W. Pugh and D. Wonnacott, "Eliminating false data dependences using the Omega test," in *SIGPLAN Conf. Programming Language Design and Implementation*, San Francisco, CA, June 1992, pp. 140–151.
[25] ——, "Going beyond integer programming with the Omega test to eliminate false data dependences," Dep. of Comput. Sci., Univ. Maryland, College Park, Tech. Rep. CS-TR-3191, Dec. 1992. An earlier version of this paper appeared at the *SIGPLAN PLDI '92 Conf.*.
[26] ——, "An evaluation of exact methods for analysis of value-based array data dependences," in *Sixth Annu. Workshop Programming Languages and Compilers for Parallel Comput.*, Portland, OR, Aug. 1993.
[27] ——, "Static analysis of upper and lower bounds on dependences and parallelism," *ACM Trans. Programming Languages Syst.*, 1993. Accepted for publication.
[28] H. Ribas, "Obtaining dependence vectors for nested-loop computations," in *Proc. 1990 Int. Conf. Parallel Process.*, Aug. 1990, pp. II-212–II-219.
[29] C. Rosene, "Incremental dependence analysis," Ph.D. dissertation, Dep. of Comput. Sci., Rice Univ., Mar. 1990.
[30] R. E. Shostak, "On the sup-inf method for proving presburger formulas," *J. ACM*, vol. 24, no. 4, pp. 529–543, Oct. 1977.
[31] Z. Shen, Z. Li, and P. Yew, "An emperical student of array subscripts and data dependences," in *Proc. 1989 Int. Conf. Parallel Process.*, Aug. 1989.
[32] V. V. Voevodin, *Mathematical Foundations of Parallel Computing*. New York: World Scientific Publishers, 1992; World Scientific Series in Computer Science, vol. 33.
[33] ——, "Theory and practice of parallelism detection in sequential programs." *Programming and Comput. Software* (*Programmirovaniye*), vol. 18, no. 3, May 1992.
[34] M. Wolfe, "The tiny loop restructuring research tool," in *Proc. 1991 Int. Conf. Parallel Process.*, 1991, pp. II-46–II-53.
[35] H. Zima and B. Chapman, *Supercompilers for Parallel and Vector Computers*. New York: ACM Press, 1991.

A Multiway Merge Sorting Network

De-Lei Lee and Kenneth E. Batcher

***Abstract*— A multiway merge sorting network is presented, which generalizes the technique used in the odd-even merge sorting network. The merging network described here is composed of m k-way mergers and a combining network. It arranges k ordered lists of length n each into one ordered lists in $T(k) + \lceil \log_m n \rceil \lceil \log_2 k \rceil \lceil \log_2 m \rceil$ steps, where $T(k)$ is the number of steps needed to sort k keys in order; and k and m are any integers no longer restricted to 2.**

***Index Terms*— Sorting networks, odd-even merge, multiway merge, parallel processing.**

I. Introduction

The subject of parallel sorting networks has been studied extensively in the past not only because parallel sorting is an interesting topic to explore in its own right but also because its important applications in switching networks, parallel processing systems, multiaccess memories, and so on.

One very popular parallel sorting network is the odd-even sorting network discovered by Batcher [1]. The odd-even merging network is a two-way merger that merges two ordered lists of lengths p and q each into one ordered list in $1 + \lceil \log_2 \max\{p, q\} \rceil$ steps. The odd-even merge produces the minimum time delay and uses minimum number of comparators, when $p = q$ [3].

In this paper, we present a multiway merging network (or k-way merger), which generalizes the technique used in the odd-even merge. The merging network described here merges k ordered lists into one ordered list, where k is not restricted to 2, hence the name. A k-way mergers can be constructed from m small k-way mergers and a combining network, where m is not restricted to 2. The k-way merging network becomes a generalized odd-even merging network, when $k = 2$. The generalized odd-even merger uses m small odd-even mergers to merge 2 ordered lists into one ordered list, where m is not restricted to 2. The multiway merge does not reduce the number of comparators nor does it reduce the delay time over the odd-even merge. However, it does provide a comprehensive extension and generalization of the odd-even merge method, allowing more flexible constructions of merge sorting networks.

The paper is organized as follows. The next section contains some basic definitions. Section III discusses the odd-even merge. The multiway merge is described in Section IV. Conclusions are drawn in Section V.

II. Definitions

The basic building block used by both the odd-even merge and the multiway merge is a 2-by-2 switching element, or comparator, as shown in Fig. 1. It receives two keys over its inputs A and B and presents their minimum on its L output and their maximum on its H output. The keys may enter and leave the comparator bit serially

Manuscript received May 21, 1993; revised August 30, 1993. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grant NSERC-OGP0009196.

D.-L. Lee is with the Department of Computer Science, York University, North York, Ont. M3J 1P3, Canada.
K. E. Batcher is with the Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242 USA.

IEEE Log Number 9407146.

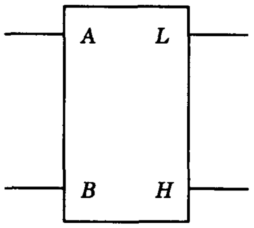


Fig. 1. A comparison-exchange element.

most significant bit first, or bit parallel. While the choice between bit-serial and bit-parallel comparator is largely an engineering decision, we shall assume the comparator will take one time delay to perform the comparison-exchange operation.

The odd-even merge accepts as input two ascending sequences of keys, and produces one ascending sequence as output. The k-way merge accepts as input k ascending sequences of keys, and produces one ascending sequence as output. As ascending sequence is defined as a sequence of keys $\langle K_1, K_2, \dots, K_n\rangle$ such that $K_1 \le K_2 \le \cdots \le K_n$.

III. Odd-Even Merge

Let the two ascending sequences to be merged be $\langle x_1, x_2, \dots, x_p\rangle$ and $\langle y_1, y_2, \dots, y_q\rangle$. A p-by-q odd-even merger can be constructed recursively as follows.

1) If $pq = 1$, the odd-even merger is simply a comparator.
2) If $pq > 1$, merge the odd sequences $\langle x_1, x_3, \dots, x_{2\lceil p/2\rceil - 1}\rangle$ and $\langle y_1, y_3, \dots, y_{2\lceil q/2\rceil - 1}\rangle$, obtaining the sorted result $\langle v_1, v_2, \dots, v_{\lceil p/2\rceil + \lceil q/2\rceil}\rangle$; and merge the two even sequences $\langle x_2, x_4, \dots, x_{2\lfloor p/2\rfloor}\rangle$ and $\langle y_2, y_4, \dots, y_{2\lfloor q/2\rfloor}\rangle$, obtaining the sorted result $\langle w_1, w_2, \dots, w_{\lfloor p/2\rfloor + \lfloor q/2\rfloor}\rangle$. Finally, apply the comparison-exchange operations
$$w_1 : v_2, w_2 : v_3, \dots, v_q : v_{q+1} \tag{1}$$
to the sequence
$$\langle v_1, w_1, v_2, w_2, \dots, v_q, w_q, \dots, v_p\rangle \tag{2}$$
to obtain the final sorted sequence.

An odd-even merging network thus constructed is composed of two small odd-even mergers to produce the two sorted sequences $\langle v_1, v_2, \dots, v_{\lceil p/2\rceil + \lceil q/2\rceil}\rangle$ and $\langle w_1, w_2, \dots, w_{\lfloor p/2\rfloor + \lfloor q/2\rfloor}\rangle$, and a column of comparators to carry out the comparison-exchange operations in (1). It merges the two ascending sequences into an ascending sequence in $1 + \lceil \log_2 \max\{p, q\}\rceil$ comparison exchange steps. The correctness proof of the odd-even merge can be found in Batcher's original paper [1].

Fig. 2 shows the odd-even merger, when $p = q = 8$. The recursive nature of the network is highlighted by the two boxes, where the first produces the sorted sequence $\langle v_1, v_2, \dots, v_8\rangle$, the second the sorted sequence $\langle w_1, w_2, \dots, w_8\rangle$, and the last column of comparators performs the comparison-exchange operations $w_1 : v_2, w_2 : v_3, \dots, w_7 : v_8$ to the sequence $\langle v_1, w_1, v_2, w_2, \dots, v_8, w_8\rangle$.

In the next section, we shall extend and generalize the theory of the odd-even merge to multiway merge.

IV. Multiway Merge

A k-way merging network of size n is a merger that merges k ascending sequences of n keys each into one ascending sequence. Fig. 3 shows a symbol for such a merging network in which the n keys of each ascending sequence, $a_{i,1}, a_{i,2}, \dots, a_{i,n}$, where $1 \le i \le k$, are presented over n inputs simultaneously. The kn outputs of the merging network represent the kn keys of the merged sequences in ascending order $c_1, c_2, \dots, c_{kn}$. A k-way merger of size 1 is simply a sorting network of order k, which can be constructed from comparison-exchange elements [1], [2]. Larger networks can be built by using the iterative rule shown in Fig. 4. A k-way merging network of size n can be built by presenting $a_{i,j}$ such that $j \bmod m = p$ to the pth k-way merger of size $\frac{n}{m}$ and then arranging the outputs of the m k-way merges, i.e., m ascending sequences, into one ascending sequence via a combining network.

We shall begin describing the detailed construction of the merging network by first giving two useful properties below on the outputs of the m small k-way mergers, which are in turn the input of the combining network.

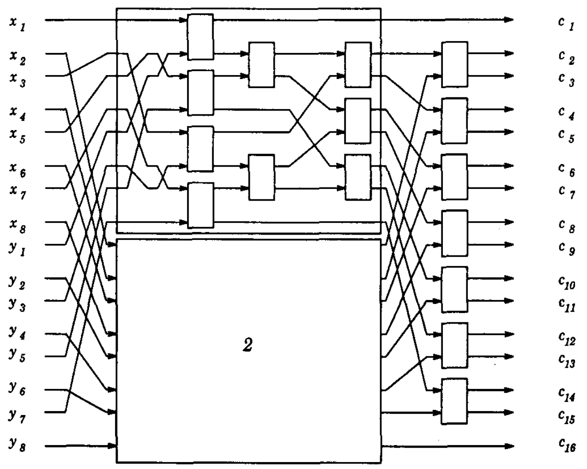
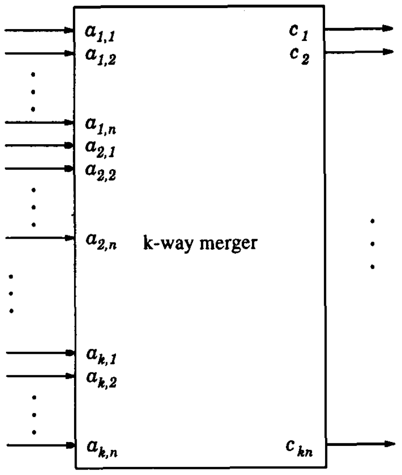


Fig. 2. The odd-even merger, when $p = q = 8$.



$$a_{i,1} \le a_{i,2} \le \cdots \le a_{i,n}, \text{ for } 1 \le i \le k; \text{ and}$$
$$c_1 \le c_2 \le \cdots \le c_{kn}.$$

Fig. 3. A symbol for the k-way merging network.

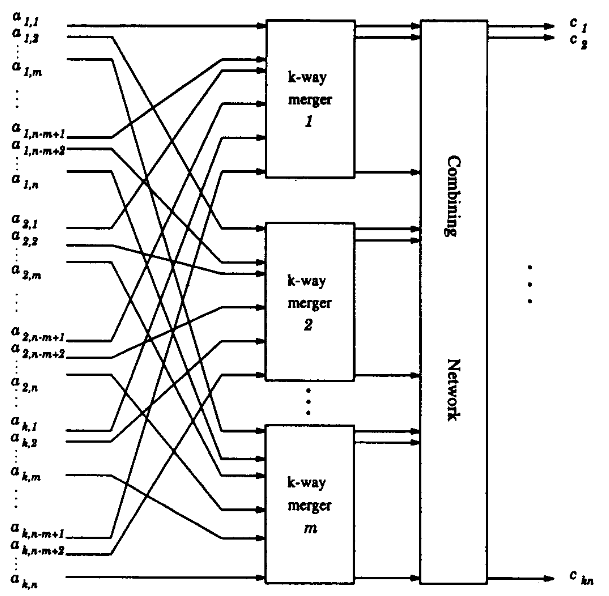


Fig. 4. Iterative construction rule for the merging network.

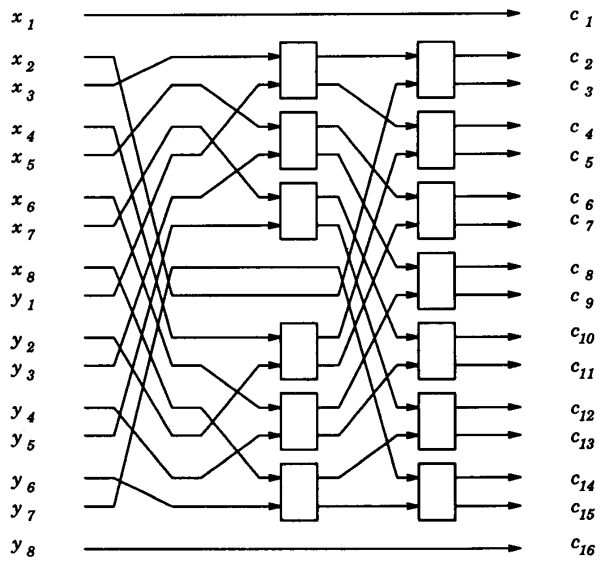


Fig. 5. The merger from Lemma 1, when $p = q = 8$ and $u = 4$.

The zero-one principle [3] will be used in our proofs: If a network with n input lines sorts all 2^n sequences of 0's and 1's into nondecreasing order, it will sort any arbitrary sequence of n numbers into nondecreasing order.

Theorem 1: Let the sequence $\langle b_{i,1}, b_{i,2}, \ldots, b_{i,l}\rangle$ be the output of the ith k-way merger in ascending order, where $1 \leq i \leq m, 1 \leq j \leq l$, and $l = \frac{kn}{m}$, then

$$b_{i,j} \leq b_{2,j} \leq \cdots \leq b_{m,j}, \text{ for all } 1 \leq j \leq l; \tag{3}$$

and

$$b_{m,j} \leq b_{1,j+k}, \text{ for all } 1 \leq j \leq l-k. \tag{4}$$

Proof: To prove this theorem, we use the zero-one principle, testing it on all sequence of 0's and 1's.

As the initial k sequences of keys are already sorted, the ith sequence $\langle a_{i,1}, \ldots, a_{i,n}\rangle$ will consist of z_i 0's followed by $n - z_i$ 1's, for some z_i where $1 \leq i \leq k$. It follows from the iterative rule that the ith k-way merger receives no more 0's than does any k-way merger preceding it, i.e., $z'_1 \geq z'_2 \geq \cdots \geq z'_m$, where z'_i is the number of 0's in the sequence $\langle b_{i,1}, \ldots, b_{i,l}\rangle$. Thus, for any $1 \leq j \leq l$, the sequence $\langle b_{1,j}, b_{2,j}, \ldots, b_{m,j}\rangle$ consists of zero or more 0's followed by zero or more 1's, which implies property (1). Furthermore, the sequence $\langle b_{1,1}, \ldots, b_{1,l}\rangle$, which is the output of the first k-way merger, will consist of exactly $z'_1 = \sum_{i=1}^{k} \lceil \frac{z_i}{m} \rceil$ 0's, followed by 1's; and the sequence $\langle b_{m,1}, \ldots, b_{m,l}\rangle$, which is the output of the last k-way merger, will consist of $z'_m = \sum_{i=1}^{k} \lfloor \frac{z_i}{m} \rfloor$ 0's, followed by 1's; and $z'_1 - z'_m$, i.e., the difference between the number of 0's in the output of the first merger and the number of 0's in the last merger, is:

$$\sum_{i=1}^{k} \left\lceil \frac{z_i}{m} \right\rceil - \sum_{i=1}^{k} \left\lfloor \frac{z_i}{m} \right\rfloor \leq k$$

which implies property (2). This completes the proof. □

One way to interpret Theorem 1 is to view the outputs of the m small k-way mergers as an array with m rows and l columns, where the sequence $\langle b_{i,1}, \ldots, b_{i,l}\rangle$ comprises the ith row. Theorem 1 states that each row of the array is sorted in ascending order; each column is sorted in ascending order; and there exists an uncertainty window covering no more than k consecutive columns of the array such that any two columns may not be completely in order if they are covered by this window; otherwise, they will be already in order. The purpose of the combining network is to reduce the uncertainty window progressively to the width of one, thereby making the final sequence in order. The following lemma will be used to describe the construction of the combining network.

Lemma 1: Two ascending sequences $\langle x_1, x_2, \ldots, x_p\rangle$ and $\langle y_1, y_2, \ldots, y_q\rangle$ such that $x_j \leq y_j$ for all $1 \leq j \leq \min\{p, q\}$ and $y_j \leq x_{j+u}$ for all $1 \leq j \leq \min\{p-u, q\}$ can be merged into an ascending sequence in exactly $\lceil \log_2 u \rceil$ steps, where u is an integer $\leq \max\{m, q\}$.

Proof: We show that the last $\lceil \log_2 u \rceil$ stages of the p-by-q odd-even merger can be used to merge the two sequences into one sorted sequence.

To see this, observe that the last stage of the odd-even merger can merge two ascending sequences $\langle x_1, x_2, \ldots, x_p\rangle$ and $\langle y_1, y_2, \ldots, y_q\rangle$ such that $x_j \leq y_i$ for all $1 \leq j \leq \min\{p, q\}$ and $y_i \leq x_{j+1}$ for all $1 \leq j \leq \min\{p-1, q\}$ into an ascending sequence. The last two stages of the odd-even merger can merge two sequence $\langle x_1, x_2, \ldots, x_p\rangle$ and $\langle y_1, y_2, \ldots, y_q\rangle$ such that $x_j \leq y_j$ for all $1 \leq j \leq \min\{p, q\}$ and $y_j \leq x_{j+3}$ for all $1 \leq j \leq \min\{p-3, q\}$ into an ascending sequence, etc. Thus the last $\lceil \log_2 u \rceil$ stages of the odd-even merger can be used to merge the two ascending sequences given in the lemma into a sorted sequence. □

Fig. 5 illustrates the merger from Lemma 1, when $p = q = 8$ and $u \leq 4$, which is identical to the last 2 stages of the odd-even merging network of Fig. 2.

The construction of the combining network is now described. As will be seen, the number of steps the combining network needs is a function of the two constants m and k, but *independent* of n, the length of the lists being combined. We shall give a constructive proof to the following result.

Theorem 2: The outputs of the m k-way mergers can be combined into an ordered sequence in $\lceil \log_2 k \rceil \lceil \log_2 m \rceil$ steps.

Proof: The input of the combining network is the m sequences of keys having the properties stated in Theorem 1. That is,

$$z_1' \geq z_2' \geq \cdots z_m'; \tag{5}$$

and

$$z_1' - z_m' \leq k, \tag{6}$$

in terms of the zero-one principle, where z_i' is the number of 0's in the ith sequence. We give a procedure to combine the m sequences into an ascending sequence.

1) If $k \leq 1$, the sequence $\langle b_{1,1}, \ldots, b_{1,l}, b_{2,1}, \ldots, b_{2,l}, \ldots, b_{m,1}, \ldots, b_{m,l} \rangle$ is already in order. This takes no operations.
2) If $k > 1$, combine the m odd sequences $b_{i,1}, b_{i,3}, \ldots, b_{i,2\lceil l/2 \rceil - 1} \rangle$ for $1 \leq i \leq m$, obtaining the sorted result $\langle v_1, v_2, \ldots, v_{m\lceil l/2 \rceil} \rangle$; and combine the m even sequences $\langle b_{i,2}, b_{i,4}, \ldots, b_{i,2\lfloor l/2 \rfloor} \rangle$ for $1 \leq i \leq m$, obtaining the sorted result $\langle w_1, w_2, \ldots, w_{m\lfloor l/2 \rfloor} \rangle$. Finally, merge the two sequences $\langle v_1, v_2, \ldots \rangle$ and $\langle w_1, w_2, \ldots \rangle$, using the procedure described in Lemma 1. The result will be sorted.

To prove the procedure given above will combine the outputs of the m k-way mergers into a sorted sequence, we use the zero-one principle. It follows from (5) and (6) that $\langle v_1, v_2, \ldots, v_{m\lceil l/2 \rceil} \rangle$ will consist of $\sum_{i=1}^{m} \lceil \frac{z_i'}{2} \rceil$ 0's, followed by 1's; and the sequence $\langle w_1, w_2, \ldots, w_{m\lfloor l/2 \rfloor} \rangle$ will consist of $\sum_{i=1}^{m} \lceil \frac{z_i'}{2} \rceil$ 0's, followed by 1's. Hence, the sequence v will have at most m more 0's than does the sequence w. By Lemma 1, the two sequences can be merged into an ordered sequence in $\lceil \log_2 m \rceil$ steps.

Now, consider the m odd sequences $\langle b_{i,1}, b_{i,3}, \ldots, b_{i,2\lceil l/2 \rceil - 1} \rangle$ for $1 \leq i \leq m$. It follows from (5) and (6) that

$$\lceil \frac{z_1'}{2} \rceil \geq \lceil \frac{z_2'}{2} \rceil \geq \cdots \geq \lceil \frac{z_m'}{2} \rceil; \quad and$$
$$\lceil \frac{z_1'}{2} \rceil - \lceil \frac{z_m'}{2} \rceil \leq \lceil \frac{k}{2} \rceil$$

where $\lceil \frac{z_i'}{2} \rceil$ is the number of 0's in the ith odd sequence. Likewise, no even sequence has more 0's than does any even sequence preceding it; and the first even sequence has at most $\lceil \frac{k}{2} \rceil$ more 0's than does the last even sequence. As the difference between the numbers of 0's in m odd sequences (and m even sequences) is halved each time m new odd sequences (and m new even sequences) are obtained, the procedure will take at most $\lceil \log_2 k \rceil \lceil \log_2 m \rceil$ steps to combine the outputs of the m k-way mergers into an ascending sequence. This completes the proof. □

Let $T(k, m, n)$ be the number of steps used by the k-way merging network to merge k ascending sequences of n keys each into an ascending sequence. It follows from the iterative construction rule and Theorem 2 that

$$T(k,m,n) = \begin{cases} T(k, m, \lceil \frac{n}{m} \rceil) + \lceil \log_2 k \rceil \lceil \log_2 m \rceil & \text{if } n > 1 \\ T(k) & \text{otherwise,} \end{cases}$$

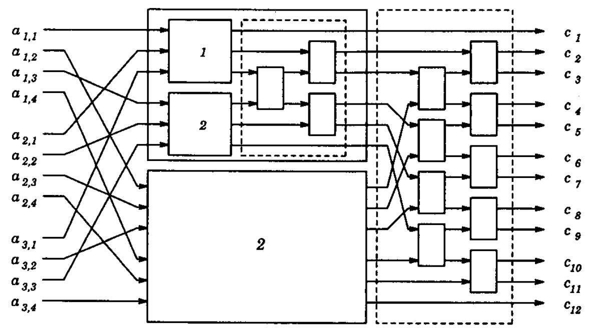


Fig. 6. The three-way merging network, when $m = 2$, and $n = 4$.

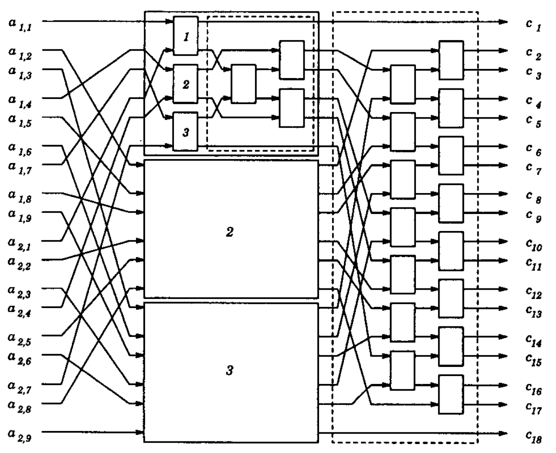


Fig. 7. The generalized odd-even merger, when $m = 3$ and $n = 9$.

where $T(k)$ is the number of steps needed to sort k keys. Solving the recurrence, we have

$$T(k,m,n) = T(k) + \lceil \log_m n \rceil \lceil \log_2 k \rceil \lceil \log_2 m \rceil.$$

Fig. 6 depicts a three-way merging network with $m = 2$ and $n = 4$, where combining networks are highlighted by the dot-lined boxes, and the iterative rule by the solid-lined boxes of different sizes. The network merges three ascending sequences of four keys each into one ascending sequence.

The k-way merge gives rise to a generalized odd-even merge, when $k = 2$, using a different modulus for merging two lists. Fig. 7 illustrates an example of such a generalized odd-even merge for $m = 3$ and $n = 9$.

While the above discussions have assumed that each of the m small k-way mergers is a recursive k-way merger, it is not necessary. In fact, a small k-way merge may use a different modulus from m to take advantages of less delay and/or comparators. For any given k and n, one can choose any value of m in the range from 2 to n to construct a multiway merging network. However, use of certain values of m results in less delay than others for the same value of n. One choice of the value of m is $m = n$. In this case, $T(k,n,n) = T(k) + \lceil \log_2 n \rceil \lceil \log_2 k \rceil$. Other values of m, which can achieve this time bound for the given value of n, are $m = 2^c$ with $c < \lceil \log_2 n \rceil$.

We have also assumed that there are n keys in each ascending sequence and m divides n for simplicity and clarity of exposition. In general, a multiway merging network can be constructed to merge k sorted lists of different lengths with the ith list having n_i keys not necessarily equal to n. One way of doing this is to construct a k-way merger of size n first using keys with the value of positive infinity to fill the difference, and then remove these positive infinity keys together with the comparison-exchange elements associated with them, resulting in a simpler k-way merging network. Thus, there are no restrictions on the property of n_i, k, and m.

A sorter can be constructed from k-way mergers: the keys are combined k at a time to form ordered lists of length k; these lists are merged k at a time to form ordered lists of length k^2, etc., until all keys are merged into one ordered list. To sort k^p keys using the mergers requires k^{p-1} mergers of size 1 followed by k^{p-2} mergers of size k followed by k^{p-3} mergers of size k^2 followed by k^{p-4} mergers of size k^3, etc., etc. The longest path will go through $\sum_{i=0}^{p-1} T(k,m,k^i) = pT(k) + \lceil \log_2 k \rceil \lceil \log_2 m \rceil (\sum_{i=1}^{p-1} \lceil i \log_m k \rceil)$ steps.

V. Conclusion

The multiway merge described in this paper merges k ascending sequences into an ascending sequence for any integer k. This differs from existing merging networks that merge only two ascending sequences into one. Furthermore, the k-way merge represents a complete generalization of the odd-even merge, when $k = 2$. In this case, it uses m small two-way mergers to merge two ascending sequences into one, where m is not restricted to 2. The odd-even merge is a special case of the k-way merge, when $k = 2$ and $m = 2$. The multiway merges does not reduce the number of comparators nor does it reduce the delay over the odd-even merge. However, it does provide a comprehensive extension and generalization of the odd-even merge method, allowing more flexible construction of merge sorting networks.

References

[1] K. E. Batcher, "Sorting networks and their applications," in *AFIPS Proc. Spring Joint Comput. Conf.*, 1968, pp. 307–314.
[2] K. E. Batcher, "On bitonic sorting networks," in *Proc. 19th Int. Conf. Parallel Process.*, 1990, pp. 376–379.
[3] D. E. Knuth, "Sorting and searching," *The Art of Computer Programming*. Reading, PA: Addison-Wesley, 1973, vol. 3.

Design of Efficient Regular Arrays for Matrix Multiplication by Two-Step Regularization

Jong-Chuang Tsay and Pen-Yuang Chang

***Abstract*—A two-step regularization method in which first permutation sequences and then broadcast planes are selected is proposed to design various regular iterative algorithms for matrix multiplication. The regular iterative algorithms are then spacetime mapped to regular arrays, such as mesh, cylindrical, two-layered mesh, and orbital arrays. The proposed method can be used to design regular arrays with execution time of less than N (problem size).**

***Index Terms*— Broadcast, cylindrical array, mesh array, orbital array, parallel algorithm design, permutation sequence, propagation, two-layered mesh array, VLSI architecture**

I. Introduction

The topic of designing 2-D regular arrays for matrix multiplication has been studied for over a decade. Most existing designs are based on the well-known sequential algorithm of $C = A \times B$ (A, B, C are all $N \times N$ matrices). These include, in chronological order, the hexagonal array (with execution time on the order of $5N$) proposed by Kung and Leiserson [1], the mesh array ($3N$) proposed by Kung [2], the hexagonal array ($3N$) of Li and Wah [3], the orbital array (N) of Porter and Aravena [4], the cylindrical array ($2N$) of Porter and Aravena [5], the two-layered mesh array ($2N$) of Kak [6], and the cylindrical array, two-layered mesh array, and mesh array ($\frac{3N}{2}$) proposed by Tsay and Chang [7]. Other designs are based on the *Winograd* algorithm. These include the mesh array ($\frac{5N}{2}$) proposed by Jagadish and Kailath [8] and the two-layered mesh array ($\frac{3N}{2}$) presented by Benaini and Robert [9].

Some of the regular array designs based on $C = A \times B$ were proposed in a rather ad hoc fashion. Although they are derived from the same sequential algorithm, no one has ever written regular iterative algorithms (RIA's) [10] for all of them and stated the relationship between them. To derive these RIA's in a unified way, in this paper we propose a *two-step regularization* method: in the first step a permutation sequence is selected for each index and in the second step a broadcast plane is selected for each variable. Then, by spacetime mapping, various regular arrays can be designed. Furthermore, with knowledge of the derivation of these RIA's, regular arrays with execution time of less than N, faster than any other designs we know of, can be obtained.

This paper is organized as follows: Section II presents some preliminary definitions. In Section III, the two-step regularization method is proposed. Using this method, we can design RIA's in a unified manner for mesh arrays, cylindrical arrays, and two-layered mesh arrays. In Section IV, we design regular arrays with execution time approaching and then equal to N. Section V studies orbital array derivations with execution time of less than N. Finally, concluding remarks are presented in Section VI.

Manuscript received February 18, 1992; revised April 1, 1994. This work was supported by the National Science Council of the R.O.C. under Contract NSC-81-0408-E-009-568.

The authors are with Institute of Computer Science and Information Engineering, College of Engineering, National Chiao Tung University, Hsinchu, Taiwan 30050, R.O.C.

IEEE Log Number 9406340.