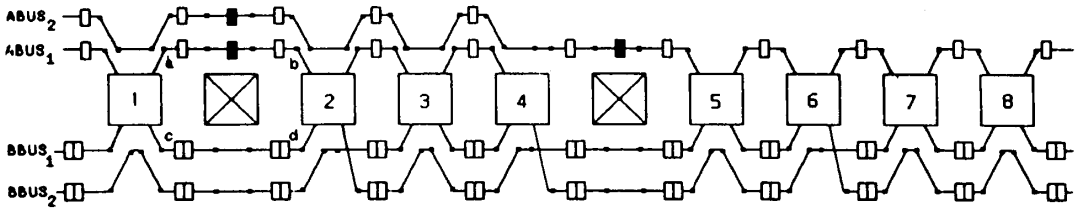


Fig. 5. Configured array in the presence of faulty PE's ($n = 4$).

achieving fault-tolerant systolic algorithms was proposed independently in [9] and [19]. If a systolic algorithm is comprised of only unidirectional data streams (that is all the data values move in one direction only as in our matrix multiplication algorithm), then in order to preserve relative alignment of data in all these streams, retiming requires that the *additional* delay encountered by data elements in each stream when bypassing a faulty PE be identical. For instance, in Fig. 5, a signal on $ABUS_1$ takes three cycles to reach point b from point a whereas in a fault-free array (see Fig. 3) it would require one cycle. Similarly, a signal on $BBUS_1$ requires four cycles to travel from c to d whereas it requires only two cycles to do so in a fault-free array. The additional delay of two cycles encountered in both these buses (and by the signals in all the other buses when bypassing this faulty PE) is the same. Thus, the single additional buffer on the ABUS's ensures the correctness of the algorithm when the faulty PE's are bypassed.

V. Concluding Remarks

In this paper, we have developed a fault-tolerant array for matrix multiplication that explicitly incorporates mechanisms for easy testability and reconfigurability. More importantly all signals in our array travel only a constant distance (independent of array size) in any clock cycle. On this model, we designed an optimal-time algorithm for multiplying matrices. The algorithm is an efficient simulation of a 2-D systolic algorithm for multiplying matrices.

Generalization of the simulation technique presented in this paper appears in [20] wherein we present a parameterized family of algorithms on this model (parameterized by the number of buses and storage per PE). The asymptotic processor and time complexities of all linear-array matrix multiplication algorithms and of the algorithm described in this paper are obtained as special cases of the parametrized algorithm.

References

[1] J. Bentley and T. Ottmann, "The power of one-dimensional vector of processors," Universitat Karlsruhe, Bericht 89, Apr. 1980.
[2] B. Chazelle and L. Monier, "A model of computation for VLSI with related complexity results," *J. ACM*, vol. 32, pp. 573–588, July 1985.
[3] A. L. Fisher, personal communication.
[4] W. M. Gentleman, "Some complexity results for matrix computations on parallel processors," *J. ACM*, vol. 25, pp. 112–115, 1978.
[5] J. W. Greene and A. Gamal, "Area and delay penalties in restructurable wafer-scale arrays," *J. ACM*, Oct. 1984.
[6] K. S. Hedlund, "Wafer scale integration on parallel processors," Ph.D. dissertation, Comput. Sci. Dep., Purdue Univ., Aug. 1982.
[7] A. V. Kulkarni and D. W. L. Yen, "Systolic processing and an implementation for signal and image processing," *IEEE Trans. Comput.*, vol. C-31, pp. 1000–1009, Oct. 1982.
[8] I. Koren, "A reconfigurable and fault-tolerant VLSI multiprocessor array," in *Proc. Eighth Annu. Symp. Comput. Architecture*, Aug. 1982, pp. 262–264.
[9] H. T. Kung and M. Lam, "Wafer-scale integration and two-level pipelined implementation of systolic arrays," in *Proc. MIT Conf. Adv. Res. VLSI*, Jan. 1984.
[10] H. T. Kung and C. E. Leiserson, "Systolic arrays (for VLSI)," in *Sparse Matrix Proc. 1978*, I. S. Duff and G. W. Stewart, Eds., SIAM, 1979, pp. 256–282.
[11] F. T. Leighton and C. E. Leiserson, "Wafer-scale integration of systolic arrays," *IEEE Trans. Comput.*, vol. C-34, pp. 448–461, May 1985.
[12] F. P. Preparata and J. E. Vuillemin, "Area-time optimal VLSI networks for multiplying matrices," *Inform. Processing Lett.*, vol. 11, no. 2, pp. 77–80, 1980.
[13] J. I. Raffel, "On the use of nonvolatile programming links for restructurable VLSI," in *Proc. Caltech Conf. VLSI*, Jan. 1979.
[14] I. V. Ramakrishnan, D. S. Fussell, and A. Silberschatz, "Systolic matrix multiplication on a linear array," in *Proc. Twentieth Annu. Allerton Conf. Comput., Contr., Commun.*, Oct. 1982.
[15] I. V. Ramakrishnan and P. J. Varman, "Modular matrix multiplication on a linear array," *IEEE Trans. Comput.*, vol. C-33, pp. 952–958, Nov. 1984.
[16] A. Rosenberg, "The Diogenes approach to testable fault-tolerant networks of processors," *IEEE Trans. Comput.*, vol. C-32, pp. 902–910, Oct. 1983.
[17] J. E. Savage, "Area-time tradeoffs for matrix multiplication and related problems in VLSI models," *J. Comput. Syst. Sci.*, vol. 20, no. 3, pp. 230–242.
[18] P. J. Varman and I. V. Ramakrishnan, "Optimal matrix multiplication on fault-tolerant VLSI arrays," Tech. Rep., State Univ. New York at Stony Brook, Aug. 1985, (also appeared in *12th ICALP*, Springer Verlag LNCS, vol. 194, July 1985, pp. 487–496).
[19] P. J. Varman, "Wafer-scale integration of linear processor arrays," Ph.D. dissertation, The Univ. Texas, Austin, Aug. 1983.
[20] P. J. Varman and I. V. Ramakrishnan, "A fault-tolerant VLSI matrix multiplier," in *Proc. 1986 Int. Conf. Parallel Processing*, Aug. 1986, pp. 351–357.

K-Way Bitonic Sort

TOSHIO NAKATANI, SHING-TSAAN HUANG, BRUCE W. ARDEN, AND SATISH K. TRIPATHI

***Abstract*—This paper presents *k-way bitonic sort*, which is a generalization of Batcher's bitonic sort. This algorithm is based on a *k*-way decomposition instead of a two-way decomposition. We prove that Batcher's bitonic sequence decomposition theorem still holds with this**

Manuscript received July 15, 1986; revised July 23, 1987. S.-T. Huang was supported by the National Science Council of the Republic of China under Contract NSC-76-0408-E007-07.

T. Nakatani is with the Department of Computer Science, Princeton University, Princeton, NJ 08544.

S.-T. Huang is with National Tsing Hua University, Institute of Computer and Decision Sciences, Hsinchu, Taiwan 30043, Republic of China.

B. W. Arden is with the College of Engineering and Applied Science, University of Rochester, Rochester, NY 14627.

S. K. Tripathi is with the Department of Computer Science, University of Maryland, College Park, MD 20742.

IEEE Log Number 8820891.

multiway decomposition. This leads to applications of sorting networks with bitonic sorters of arbitrary or mixed sizes.

Index Terms—**Bitonic sort, parallel processing, parallel sorting.**

I. Introduction

Batcher's bitonic sort (Batcher [1] and Knuth [9]) has been studied extensively. Stone has described the bitonic sort on the single-stage shuffle-exchange network (Stone [18]). He has also implemented it on the STAR vector processor (Stone [19]). Orcutt has implemented the bitonic sort on the ILLIAC-IV (Orcutt [15]). Thompson and Kung have shown improved time complexity for the bitonic sort on a mesh-connected processor by adopting the shuffle-row-major ordering (Thompson and Kung [20]). Nassimi and Sahni have also made a different adaptation of the bitonic sort on a mesh-connected processor achieving the same time complexity based on the row-major ordering (Nassimi and Sahni [13]). Meertens has studied the bitonic sort on the Ultracomputer (Meertens [12]). Jayanata and Hsiao have used the bitonic sort for a database machine design (Jayanata and Hsiao [8]). Chung, Luccio, and Wong have studied the bitonic sort for magnetic bubble memory systems (Chung, Luccio, and Wong [4]).

For VLSI implementation, Preparata and Vuillemin have studied the adaptation of the bitonic sort on the cube-connected-cycles network (Preparata and Vuillemin [17]). Nath, Maheshwari, and Bhatt have adapted the bitonic sort to the orthogonal tree (or mesh-of-trees) [14]. Bonuccelli and Pagli have taken a similar approach on the mesh-of-trees for external sorting (Bonuccelli and Pagli [3]). Bilardi and Preparata have designed an optimal VLSI architecture for the bitonic sort (Bilardi and Preparata [2]). Thompson has made an extensive survey for parallel sorting including various bitonic sorting schemes in terms of VLSI complexity (Thompson [21]). Loui has studied the bitonic sort in the context of distributed computing (Loui [11]). More recently, several people have implemented the bitonic sort using reduced hardware (Ja'Ja' and Own [7], Own and Ja'Ja' [16], Hsiao and Shen [5], and Tseng, Hwang, and Kumar [22]).

All of these efforts are based on the two-way decomposition of the original Batcher's bitonic sort. In this paper, we present k-way bitonic sort, which is the generalization of Batcher's bitonic sort. K-way bitonic sort is based on a k-way decomposition scheme instead of two-way decomposition. We prove that Batcher's bitonic sequence decomposition theorem still holds with a multiway decomposition. The proof is based on the zero–one principle: if an algorithm can sort an arbitrary zero–one sequence, using only comparisons applicable to arbitrary values, then it can sort any sequence (see [9, p. 224]).

This result leads to networks composed of bitonic sorters of varying size. In Section II, we describe mathematical notation and definitions. In Section III, we prove the main theorem and several corollaries.

II. Mathematical Notation and Definitions

In this section, notation and definitions are described for later use.

Definition 1: A sequence of real numbers A is bitonic if

a) it is a concatenation of a monotonically increasing sequence B and a monotonically decreasing sequence C, or if

b) the sequence A can be shifted cyclically so that condition a) is satisfied.

Moreover, a sequence A is called *regular-bitonic* if B and C are the same size. If B or C is empty, A is called *monotonic*.

Example: In the zero–one context, 0000011111000000 is bitonic as are any cyclic shifts of this binary string. There are either no 0–1 transitions or two 0–1 transitions in the cyclical string.

Definition 2: An *n-regular-bitonic-merger* permutes a regular-bitonic sequence of length n into monotonic sequence. Similarly, an *n-bitonic merger* permutes a bitonic sequence of length n into monotonic sequence. An *n-sorter* puts an arbitrary sequence of length n into monotonic order (Fig. 1).

Definition 3: If every element of the sequence A is less than or equal to every element of the sequence B, then $A \leq B$.

Definition 4: A sequence of zeros and ones is called *clean* if it is all zeros or all ones. Otherwise, a sequence is called *dirty*.

III. Fundamental Theorem for k-Way Bitonic Sort

In this section, we prove the generalization of the original theorem (i.e., $k = 2$) outlined by Batcher [1] and Stone [18]. As mentioned, the proof uses the zero–one principle, and more specifically, a binary bitonic sequence is represented in row-major order in a $k \times n$ array. For, example, the sequence shown earlier is represented in a 4 $\times$ 4 array as follows:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A' = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Theorem: (k-way bitonic sort). Let $\{a_i\}$ for $i = 0, \cdots, N-1$ be a bitonic sequence of length $N = kn$. Let $A = (a_{ij} | a_{ij} = a_{in+j}, 0 \leq i \leq k-1, 0 \leq j \leq n-1)$ and $A' = A'_0, \cdots, A'_{n-1}$, the sorted (nondecreasing) columns of A, then $B_0, \cdots, B_{k-1}$, the rows of A', are bitonic and $B_0 \leq B_1 \leq \cdots \leq B_{k-1}$.

Proof: Using the zero–one principle, consider a binary bitonic sequence a of length $N = kn$. The derivative $k \times n$ matrix A will have at most two dirty rows, since there are at most two 0–1 transitions in a binary, bitonic sequence. After column sorting, the matrix A' will have at most one dirty row. Assume that A has two dirty rows, otherwise the reduction is direct. Each row has only one transition and from left-to-right one is 0 to 1 and the other is 1 to 0. Again from left-to-right, the identical sequences (zeros or ones) have a gap (ones or zeros) between them, meet exactly, or overlap. With the gap or overlap, a clean row is produced or two clean rows when there is exact correspondence. Furthermore, the remaining dirty row, if any, will have at most two transitions, corresponding to the boundaries of the gap or overlap. Thus, the remaining dirty row is binary and bitonic. The rows preceding it are all zeros and trivially bitonic and the succeeding rows are all ones and bitonic. Therefore, the rows satisfy the bitonic inequality $B_0 \leq B_1 \leq \cdots \leq B_{k-1}$. □

Corollary 1: A bitonic sequence of length $N = kn$ can be merged by one stage of n k-bitonic mergers followed by one stage of k n-bitonic mergers [Fig. 2(a), (b), and (c)].

Proof: The columns of A are bitonic since they are subsequences of a single bitonic sequence. The column sort to produce A' can be a bitonic merge. □

Corollary 2: An arbitrary sequence of length $N = kn$ and $k = 2^h$ can be sorted by $h = \log k + 1$ steps: (Step 0) k n-sorters, (Step 1) $k/2$ $2n$-regular-bitonic mergers, (Step 2) $k/4$ $4n$-regular-bitonic mergers, $\cdots$, (Step $h = \log k$) a kn-regular-bitonic merger [Fig. 3(a)].

Proof: A regular-bitonic sequence of length $2n$ can be constructed from two monotonic sequences of length n, one ascending and one descending. Corollary 1 implies that there are two stages in each step except for the first step. □

Corollary 3: An arbitrary sequence of length $N = 2^{2m}$ can be sorted with $2m + 1$ stages of n n-sorters, where $n = 2^m$ [Fig. 3(b)].

Proof: Each of the $n \times n$ blocks in the $2m$ stages after the first of a network constructed according to Corollary 2 can be uniformly replaced by an *n-sorter*. □

IV. Multiple-Pass Bitonic Sorting

Stone [18] had shown that the bitonic sort could be realized with multiple passes of a perfect shuffle network as in Fig. 4(a). This can

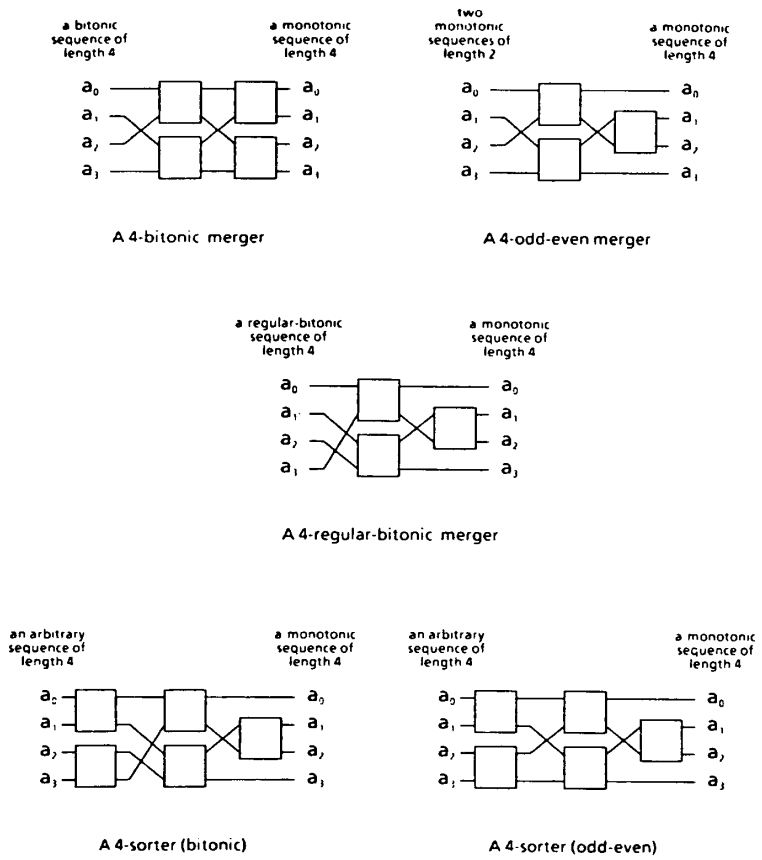
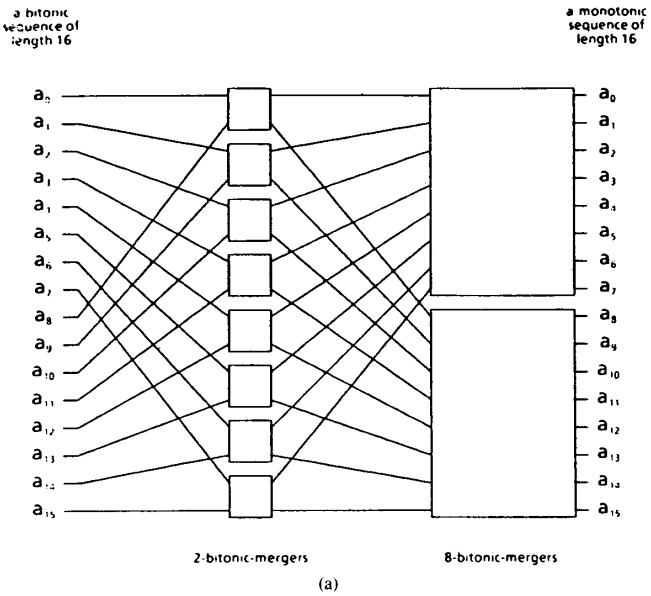
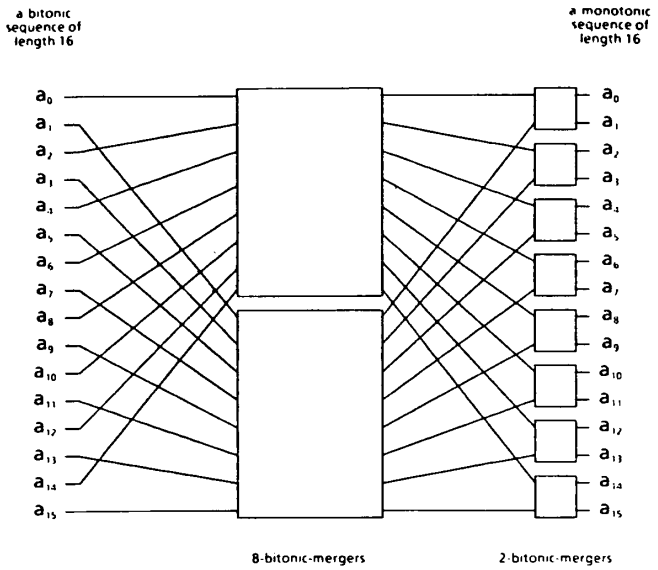


Fig. 1. A 4-regular-bitonic merger, 4-bitonic-merger, and 4-sorter.

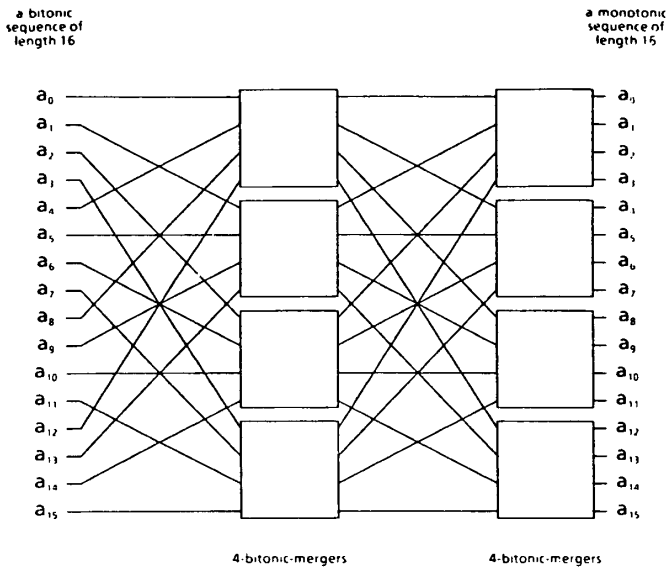


(a)

Fig. 2. (a) A 16-bitonic-merger by two-way decomposition (Batcher's construction).

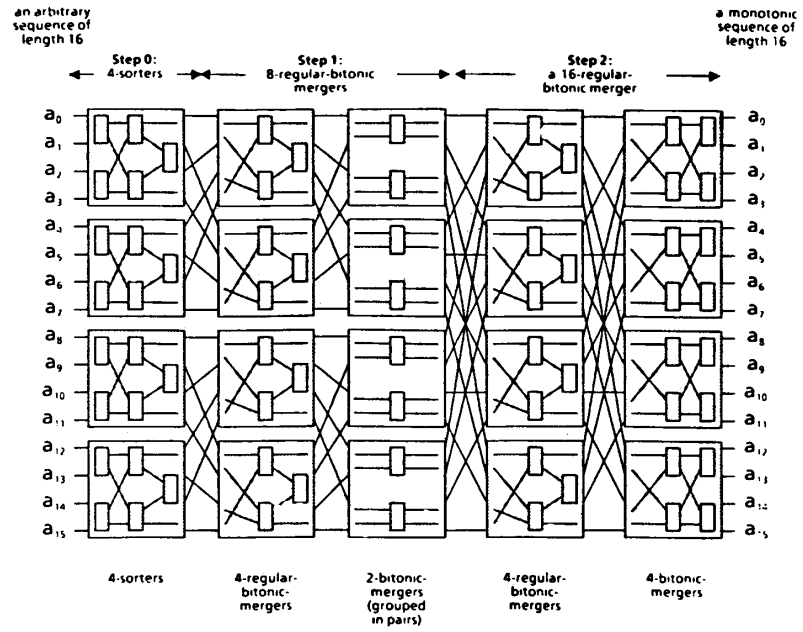


(b)

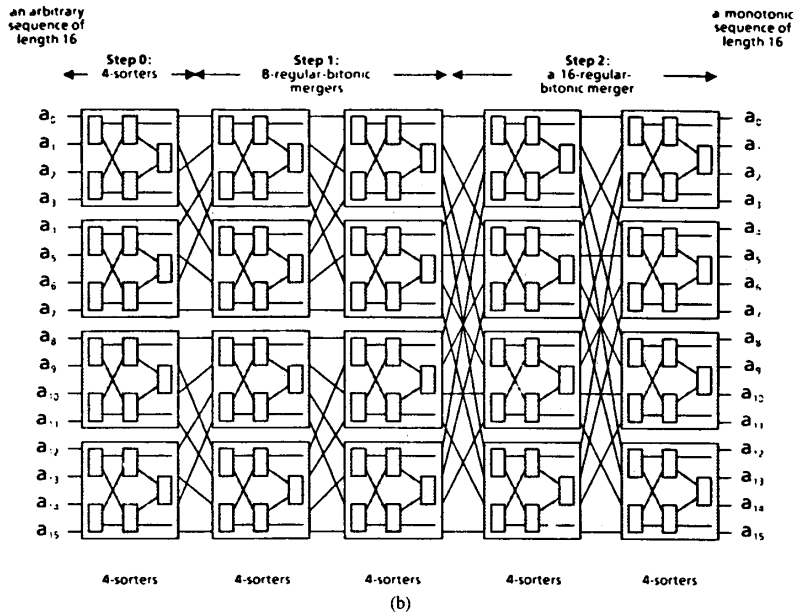


(c)

Fig. 2. (*Continued*) (b) A 16-bitonic-merger by eight-way decomposition. (c) A 16-bitonic-merger by four-way decomposition.

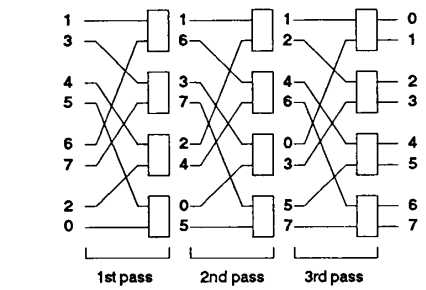


(a)

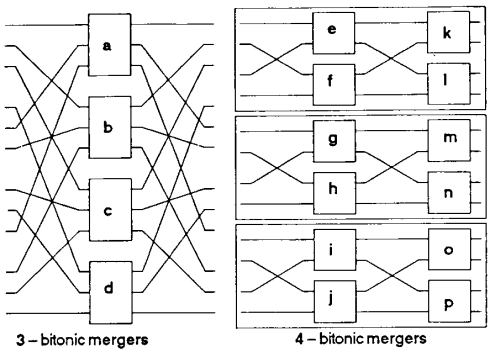


(b)

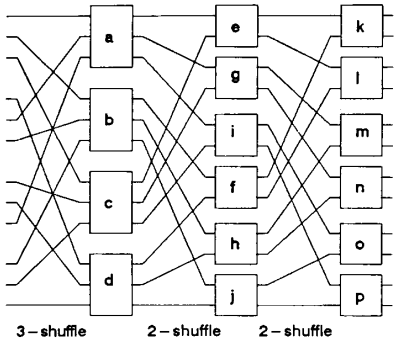
Fig. 3. (a) A 16-sorter based on four-way decomposition with reduced comparators. (b) A 16-sorter based on four-way decomposition using 4-sorters.



(a)



(b)



(c)

Fig. 4. (a) A three-pass perfect shuffle network to realize a bitonic sort for eight items. (b) A 12-bitonic merger. (c) A generalized omega network with $N = 3 \times 2 \times 2$.

also be generalized for k-way bitonic sort. For a specific example, Fig. 4(b) shows a 12-bitonic-merger consisting of one stage of 3-bitonic mergers followed by one stage of 4-bitonic mergers, the 4-bitonic merger, in turn, consisting of one stage of 2-bitonic mergers followed by one stage of 2-bitonic mergers; the network in Fig. 4(c) is a redrawing of the one in Fig. 4(b). The equivalence of these two networks can be verified by the verification technique developed in (Huang and Tripathi [6]) for permutation networks. From the diagrammatic point of view, the redrawn network is exactly a generalized omega network discussed in Lawrie [10]. A generalized omega network with $N (= k_1 k_2 \cdots k_n)$ inputs can be constructed as n stages of switches; the ith stage is k_i-switches, and the connection pattern before the ith stage is k_i shuffle. The generalization is that a generalized omega network with its k switches replaced by k-bitonic mergers can be used as a bitonic merger.

V. Conclusions

The networks obtained from Corollary 2 have fewer comparators than Batcher's original bitonic network but no less than the number in an odd–even sorting network (Batcher [1]). The networks from Corollary 3 have more comparators than the original. The uniform building blocks provide more sorting capability than is needed. However, the modular construction permits an iterative implementation, as described for two-input comparators in Stone [18] and is more attractive for possible VLSI implementation.

References

[1] K. E. Batcher, "Sorting networks and their applications," in *1968 Spring Joint Comput. Conf. AFIPS Proc.*, vol. 32, Washington, DC, 1968, pp. 307–314.
[2] G. Bilardi and F. P. Preparata, "An architecture for bitonic sorting with optimal VLSI performance," *IEEE Trans. Comput.*, vol. C-33, no. 7, pp. 646–651, 1984.
[3] M. A. Bonuccelli and L. Pagli, "External sorting in VLSI," *IEEE Trans. Comput.*, vol. C-33, no. 10, pp. 931–934, 1984.
[4] K. M. Chung, F. Luccio, and C. K. Wong, "On the complexity of sorting in magnetic bubble memory systems," *IEEE Trans. Comput.*, vol. C-29, no. 7, pp. 553–565, 1980.
[5] C. C. Hsiao and N. Shen, "k-fold bitonic sort on a mesh-connected parallel computer," *Inform. Processing Lett.*, vol. 21, no. 10, pp. 207–212, 1985.
[6] S. T. Huang and S. K. Tripathi, "Finite state model and compatibility theory: New analysis tools for permutation networks," *IEEE Trans. Comput.*, vol. C-35, no. 7, pp. 591–601, 1986.
[7] J. Ja'Ja' and R. M. Owens, "VLSI sorting with reduced hardware," *IEEE Trans. Comput.*, vol. C-33, no. 7, pp. 668–671, 1984.
[8] B. Jayanata and D. K. Hsiao, "Parallel bitonic record sort—An effective algorithm for the realization of a post processor," Tech. Rep. OSUCISRC-TR-79-1, Ohio State Univ., Columbus Comput. Inform. Sci. Res. Center, 1979.
[9] D. E. Knuth, *The Art of Computer Programming, Vol 3: Sorting and Searching*. Reading, MA: Addison-Wesley, 1973.
[10] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, vol. C-24, no. 12, pp. 1145–1155, 1975.
[11] M. C. Loui, "The complexity of sorting on distributed systems," *Inform. Contr.*, vol. 60, pp. 70–85, 1984.
[12] L. G. L. T. Meertens, "Bitonic sort on ultracomputers," Tech. Rep. MC-IW-117-79, Mathematisch Centrum, Amsterdam, Netherlands, 1979.
[13] D. Nassimi and S. Sahni, "Bitonic sort on a mesh-connected parallel computer," *IEEE Trans. Comput.*, vol. C-27, no. 1, pp. 2–7, 1979.
[14] D. Nath, S. N. Maheshwari, and P. C. P. Bhatt, "Efficient VLSI networks for parallel processing based on orthogonal trees," *IEEE Trans. Comput.*, vol. C-32, no. 6, pp. 569–581, 1983.
[15] S. E. Orcutt, "Implementation of permutation functions in ILLIAC IV-type computers," *IEEE Trans. Comput.*, vol. C-25, no. 9, pp. 929–936, 1976.
[16] R. M. Owens and J. Ja' Ja', "Parallel sorting with serial memories," *IEEE Trans. Comput.*, vol. C-34, no. 4, pp. 379–383, 1985.
[17] F. P. Preparata and J. Vuillemin, "The cube-connected cycles: A versatile network for parallel computation," *Commun. ACM*, vol. 24, no. 5, pp. 300–309.
[18] H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.*, vol. C-20, no. 2, pp. 153–161, 1971.
[19] ——, "Sorting on STAR," *IEEE Trans. Software Eng.*, vol. SE-4, no. 3, pp. 138–146, 1978.
[20] C. D. Thompson and H. T. Kung, "Sorting on mesh-connected parallel computers," *Commun. ACM*, vol. 20, no. 4, pp. 263–271, 1977.
[21] C. D. Thompson, "The VLSI complexity of sorting," *IEEE Trans. Comput.*, vol. C-32, no. 12, pp. 1171–1184, 1983.
[22] P. S. Tseng, K. Hwang, and V. K. Prasanna Kumar, "A VLSI-based multiprocessor architecture for implementing parallel algorithms," in *Proc. IEEE Int. Conf. Parallel Processing*, 1985, pp. 657–664.