A GENERALIZED BITONIC SORTING NETWORK

Kathy J. Liszka and Kenneth E. Batcher
Kent State University
Kent, Ohio 44242-0001
email: kliszka@cs.kent.edu email: batcher@cs.kent.edu

Abstract -- *The bitonic sorting network will sort $N=2^m$ keys in $O(\log^2 N)$ time with $O(N\log^2 N)$ comparators. Developments on the sorter enable the network to sort $N=pq$ keys, a composite number. However, there has been no general method for sorting a bitonic sequence of N keys, N a prime, or N a composite that decomposes into primes larger than 3. The odd-merge method removes this constraint while maintaining the same cost and delay, using a uniform and efficient decomposition.*

1. INTRODUCTION

In 1968, the bitonic sorter was introduced as an $O(\log^2 N)$ delay sorting network with $O(N\log^2 N)$ comparators[1]. This network, designed for 2^n keys, provides modularity, so the network may easily be partitioned and constructed. In the last two decades, much effort has been devoted to improve the network and map it to various supercomputer architectures. Nakatani et. al. [4], decomposed the bitonic sorter into a k-way sorter, where pk-keys, a composite number, may be sorted. In 1991, Wang, Chen and Hsu [5] mapped a reduced ascending-descending sorter of any size to an incomplete hypercube, based on work done by Batcher [2] in 1990. But this generalization does not work for bitonic sequences in general, only the subset of ascending-descending keys. Until now, there has been no general method for sorting a bitonic sequence of N keys, N a prime, or N a composite that decomposes into primes larger than 3. The odd-merge method presented here removes this constraint while maintaining the same cost and delay, using a uniform and efficient decomposition.

This paper is organized in the following way. Basic definitions are presented in the second section. The odd-merge bitonic sorter is described in section three. Performance is discussed in section four and conclusions are drawn in the last section.

2. TERMINOLOGY

The basic component of the bitonic sorter is a switching element (figure 1). Two input keys, a and b, are compared and output in ascending order with min(a,b) output to L, and max(a,b) output to H. All of the work on the bitonic sorter presented here uses the simple 2x2 comparator. In this paper, the terms switching element, switch, and comparator will be used interchangeably.

The bitonic sorter accepts a bitonic sequence as input to the network and produces a sorted sequence as output. For clarity, an *ascending sequence* (/)is a sequence of keys $k_0,k_1,\ldots,k_n$ such that $k_0 \le k_1 \le \ldots \le k_n$. A *descending sequence* (\) is a sequence of keys $k_0,k_1,\ldots,k_n$ such that $k_0 \ge k_1 \ge \ldots \ge k_n$. Note that a single key, k, is both an ascending sequence and a descending sequence.

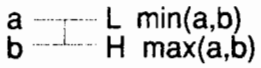

Figure 1
A Simple 2x2 Comparator

An ascending-descending sequence (/\) is an ascending sequence of keys followed by a descending sequence of keys, $k_0,k_1,\ldots,k_{i-1},k_i,k_{i+1},\ldots,k_n$ such that $k_0 \le k_1 \le \ldots \le k_{i-1} \ge k_i \ge k_{i+1} \ge \ldots \ge k_n$. Similarly, for a *descending-ascending sequence* (\/).

A bitonic sequence is the rotation of an ascending and a descending monotonic sequence. For example, (1 2 3 4 5 4 3 2 1) is an ascending monotonic sequence (1 2 3 4 5) and a descending monotonic sequence (4 3 2 1). It is irrelevant if the fifth key is associated with the ascending or the descending sequence. For instance, we could state that the previous sequence is an ascending monotonic sequence (1 2 3 4) and a descending monotonic sequence (5 4 3 2 1). Because this is true, note that either the ascending sequence or the descending sequence may be null.

A significant property of a bitonic sequence is that it remains bitonic if it is split anywhere and the two parts are interchanged. Consider the simplest case, binary bits, where a string of zeros and ones are arranged in an ascending sequence. Then 00011111 is an /-sequence. If it is split between the fifth and sixth keys and the two

Proceedings of the International Conference on Parallel Processing (ICPP'93)
0-8493-8983-6/0-7 $20.00 © 1993IEEE

parts are interchanged, the sequence 11100011 results which is still bitonic. As a check for this bitonic property, one must simply rotate the sequence until a /\- sequence appears. If one cannot be created by the rotation, then the sequence is not bitonic.

The definition of a *bitonic sorter* is a network which arranges a bitonic sequence into a monotonic sequence. Since any two monotonic sequences may be joined to form a bitonic sequence, as either a strict /\-sequence or a \/-sequence, a bitonic sorter may be considered to be a sorting network since it arranges a bitonic sequence into a monotonic sequence. In fact, a simple 2x2 comparator is the smallest bitonic sorter.

The complexity and total number of comparators in a network are factors of cost. When calculating efficiency and speed, one needs to consider delay. The *delay* of a network is defined to be the number of comparators in the longest path through the network. Parallelism is applied when multiple comparators do not conflict in data inputs, and thus count as one step in delay.

Bitonic sorters are modular, meaning that larger networks may be constructed from smaller bitonic sorters. A bitonic sorter for 4 keys is constructed from the 2x2 comparator. A sorter for 8 numbers is constructed from a 4- key bitonic sorter. Figures 2 and 3 show bitonic sorters for N = 4 and 8 keys respectively. Throughout this paper, uppercase N will always refer to the number of keys input to (and output from) the sorting network.

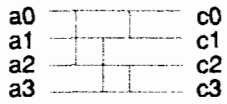

Figure 2
4-Key Bitonic Sorter

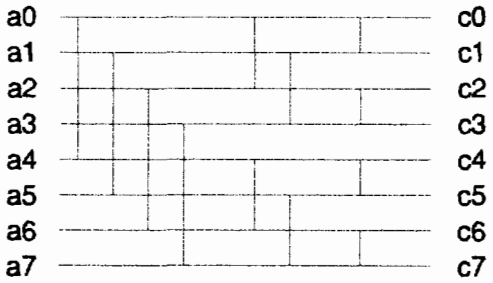

Figure 3
8-Key Bitonic Sorter

The bitonic sorter described thus far, is designed to sort keys where $N = 2^k$. In 1989, Nakatani, Huang, Arden and Tripathi [4] presented work that decomposed the bitonic sorter where N = pq, a composite number. They call it a k-way bitonic sorter, based on a k-way decomposition, instead of a two-way decomposition. Fundamentally, the bitonic sequence may be thought of in a pxq matrix form. A p-key bitonic sorter is applied to each column, then a q-key bitonic sort is applied to the rows.

In general, Batcher's original constructions showed a 2k-key sorter that could be decomposed as k 2-key bitonic sorters or 2 k-key bitonic sorters. Nakatani, et al. showed a pq-key bitonic sort that could be decomposed as p q-key bitonic sorters or q p-key bitonic sorters. The problem with the pq-sorter is one where N decomposes into primes greater than 3. For example, if N=30, then N=2x3x5. There is no bitonic sorter defined for N=5, therefore, there is no bitonic sorter defined for the composite number 30.

3. ODD-MERGE METHOD

The odd-merge method decomposes the keys based on indices as follows where N = 2n+1, an odd number.

1. Divide the bitonic sequence into 2 bitonic sequences based on odd/even indices. By definition, the two subsequences are still bitonic. Simply removing an element does not affect this property.

2. Sort each subsequence. This requires an n-key bitonic sorter and an (n+1)- key bitonic sorter.

3. A simple interleave of the two subsequences will produce a final sorted sequence.

Figure 4 shows an odd-merge bitonic sorter for 7 keys. It is composed of a 4-key bitonic sorter on the keys with even indices, a 3-key bitonic sorter on the keys with odd indices, and a simple interleave of 3 keys with 4 keys.

Proceedings of the International Conference on Parallel Processing (ICPP'93)
0-8493-8983-6/0-7 $20.00 © 1993IEEE

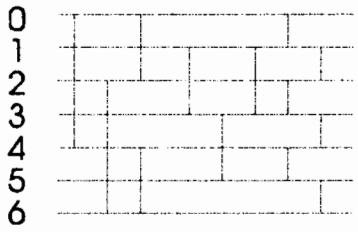

Figure 4
Odd-Merge Bitonic Sorter for N = 7
Bitonic (4) + Bitonic(3) + Interleave
Comparators = 13, Delay = 5

The proof of correctness for the odd-merge method must show that the simple interleave will suffice as the completion step for this method. Reference the work done in [3] for details of the proof to show that an N-key bitonic sorter can be constructed from an n-key bitonic sorter, an (n+1)-key bitonic sorter and 2n switches, where N=2n+1.

4. PERFORMANCE

The odd-merge used recursively with the bitonic merger demonstrates excellent results. It's performance, logically, does not vary significantly from the composite bitonic merger. First, examine the performance of the bitonic merger without the odd-merge method. N is composite, and therefore has a factorization of $N = r_1 \times \ldots \times r_k$, for $k \geq 2$. Select any factorization of $N = pq$ and construct a bitonic merger with q p-key mergers and p q-key mergers. P is a unique set of r_i; q is constructed from the remaining prime factors not included in p.

To construct a p-key merger, p may either be a composite number that must be decomposed further, or it is a terminating value that has a bitonic merger already defined (ex. 2, 3, 4, 8, 16, etc.). If it must be decomposed as a composite, then it may be factored into some unique pair p_1q_1 which are extracted from the set of r_i from the factorization of p. Similarly for a q-key merger, except if q is also a composite number that requires further decomposition, it must be factored into p_2q_2 which are constructed from the remaining factors, r_j of q, not used by p. Therefore, it is irrelevant how N is factored in the composite bitonic merger. The results will be the same. For example, given N = 60, we can choose pq to be (5,12), (3,20), (4,15), etc. If (5,12) is selected, the five q=12 size sorters will decompose further into either (3,4) or (2,6). Each will decompose into 3x2x2. Using the bitonic merger described in Nakatani [4] means that N will ultimately be reduced to a prime factorization of

$$2^{k1} \times 3^{k2} \times 5^{k3} \times \ldots$$

where each $k_j \geq 0$. To construct a bitonic sorter this way, we need

$$k_1 \times \text{bitonic}(2) + k_2 \times \text{bitonic}(3) + k_3 \times \text{bitonic}(5) + \ldots$$
or
$$q \times \text{bitonic}(p) + p \times \text{bitonic}(q)$$

comparators. Therefore, the entire decomposition of N will yield the same results. The performance in terms of delay, may be determined by:

$$\text{bitonic}(p) + \text{bitonic}(q).$$

Remember that the q p-key sorters may be performed in parallel as well as the p q-key sorters. However, there is no bitonic sorter defined for 5, 7, 11, 13, 17, etc. using this method of decomposition alone, so without a generalizing extension, the actual implementation of the sort for N is impossible.

When the odd-merge method is applied to the bitonic merger, for odd N as well as primes, it becomes relevant which factorization is selected for a composite N. The first positive N where this occurs is 105. The proper factors of 105 are 3, 5 and 7. Table 1 shows the results of the different combinations.

factors	cost	delay
3 x 35	468	13
5 x 21	468	13
7 x 15	468	12

Table 1
Results of Alternate Decompositions
of N = 105 Using the Composite Method

In the case where N = 105, the 7 x 15 decomposition yields a smaller delay because the odd-merge requires a delay of 7 for its (7,8) decomposition of 15 while the composite method requires a delay of 8 for its 3x5 decomposition. Optimizing delay, this occurs 22 times for N up to 1000. This means the odd-merge method may not only solve the problem of prime decomposition, it actually improves performance of the composite merger for some N.

Proceedings of the International Conference on Parallel Processing (ICPP'93)
0-8493-8983-6/0-7 $20.00 © 1993IEEE

The odd-merge method requires a delay for N=2n+1 of

$$\max[\text{bitonic}(n), \text{bitonic}(n+1)] + 2.$$

The upper bound for delay is $(2\lceil \log N \rceil)-1$, where $\lceil x \rceil$ is the ceiling function. This may be calculated by looking at the worst case scenario where N decomposes completely by odd numbers. Starting at N=3, it requires a delay of 3 to sort. This is our base. To sort N=5 or N=7 requires bitonic(3)+2 for both, as they decompose into (2,3) and (3,4) respectively. To sort N=9,11,13,15 requires either bitonic(5)+2 or bitonic(7)+2, both being (bitonic(3)+2)+2 = bitonic(3)+4. To sort the odds from 17 to 31 will require a delay of 2 plus a bitonic of either 9, 11, 13 or 15, which requires a delay of ((bitonic(3)+2)+2)+2)=bitonic(3)+6. For each range for a power of 2, from $N = 2^m+1$ through 2^{m+1}, we increase delay by 2 for odd numbers. The delay remains constant for values of N within these ranges.

The general formula is (2logN)-1. The odd-merge used as the generalization technique for the bitonic merger is very stable in terms of delay for the upper bound. It is possible to do better than this, case in point being N=19. This decomposes into

$$\max[\text{bitonic}(9), \text{bitonic}(10)] + 2$$

which produces a delay of 8. Bitonic(10) requires a delay of 6, and so does bitonic(9) if the composite method is used to decompose it into (3x3) instead of the odd-merge decomposing it into (4,5) with a result of 7. An example of N meeting the upper bound is 31. This requires

$$\max[\text{bitonic}(15), \text{bitonic}(16)]+2$$

which is $2(\lceil \log 31 \rceil)-1=9$.

The cost for the composite bitonic merger may be calculated for N = pq:

$$p \times \text{bitonic}(q) + q \times \text{bitonic}(p).$$

For the odd-merge, the equation for N = 2n + 1 is:

$$\text{bitonic}(n) + \text{bitonic}(n+1) + (N-1).$$

The selected merger is constructed from the minimum result of the above two equations applied recursively.

5. CONCLUSION

For the odd-merge, there is only one choice for decomposition, and therefore, no conflict between optimizing either cost or delay. However, when a table for some range of N is constructed using the composite and odd-merge methods jointly, a decision must be made to minimize either cost or delay. There are cases where the odd-merge may outperform the composite method either in terms of cost or delay for some non-prime N. For example, if N = 65, the composite method requires 274 comparators while the odd-merge requires 254 comparators, a savings of 20. For delay, if N = 15, the odd-merge will take 7 steps compared to 8 using the composite merger. With exception of $N=2^m$, any N greater than 4 is dependent on efficient decompositions. The results of those decompositions must be selected based on the objectives of saving switching elements or time. The overall performance of the bitonic merger incorporating this generalization remains at $O(N\log^2 N)$ for cost and $O(\log^2 N)$ for delay when applied to sorting inputs of random keys.

REFERENCES

[1] Batcher, K. E., "Sorting Networks and Their Applications", *1968 Spring Joint Computer Conf., AFIPS Proc.*, Washington, D.C., (1968, Vol. 32), pp. 307-314.

[2] _____, "On Bitonic Sorting Networks", *Proceedings of the 1990 International Conference on Parallel Processing*, (1990, Vol. 1), pp. 376-379.

[3] Liszka, K.J., "Generalizing Bitonic and Odd-Even Merging Networks", *Doctoral Thesis*, Department of Mathematics and Computer Science, Kent State University, (1992).

[4] Nakatani, T., Huang, S., Arden, B. W., and Tripathi, S. K., "K-Way Bitonic Sort", *IEEE Transactions on Computers*, (February 1989,Vol. 32,), pp. 283-288.

[5] Wang, B. F., Chen, G. H., and Hsu, C. C., "Bitonic Sort with an Arbitrary Number of Keys", *Proceedings of the 1991 International Conference on Parallel Processing*, (1991, Vol. 3), pp. 58-61.

0-8493-8983-6/0-7 $20.00 © 1993IEEE