A Generalized Bitonic Sorting Technique for a Q-Dimensional Mesh Connected Computer

Kamalika Chaudhuri, Siddhartha Saha, P.Gupta
Department of Computer Science and Engineering
Indian Institute of Technology,
Kanpur, 208016, INDIA
Email: pg@cse.iitk.ac.in

Abstract

In this paper, we have proposed the generalization of the parallel bitonic sorting algorithm on a 2-Dimensional Mesh Connected Computer, due to Thompson and Kung [TK77], for a Q-Dimensional Mesh Connected Computer. The time complexity of our algorithm is $O(\sqrt[Q]{N})$, and the cost is $O(N\sqrt[Q]{N})$ in a Mesh Connected Computer with N processors arranged in the form of a Q Dimensional Mesh with $\sqrt[Q]{N}$ processors in each dimension. We have also modified the algorithm to get a cost optimal algorithm for performing bitonic sort in a Q-Dimensional Mesh Connected Computer.

I. Introduction

In our paper, we propose an algorithm for performing bitonic sort on a Q-Dimensional Mesh Connected Computer. The proposed algorithm is the generalization of the bitonic sorting algorithm of Thompson and Kung [TK77] for a 2-Dimensional Mesh Connected Computer. A mesh connected computer having a large number of processors is easy to implement because of the constant number of connections per processor. Another advantage of having a 2 dimensional or 3 dimensional mesh connected computer is that in this case, the processors can be readily arranged in a grid in such a way that the wire length for each connection is constant. This makes the choice of a mesh connected architecture particularly suitable for the implementation of large scale parallel machines. In Section 2, the parallel algorithm on Q-Dimensional Mesh Connected Computer has been proposed. Section 3 deals with the analysis of the algorithm. Conclusions are given in Section 4.

II. Algorithm

Let us assume the problem of sorting N numbers where $N = n^Q$ in a Q-Dimensional Mesh Connected Commuter having N processors. These processors are arranged in the form of Q-Dimensional mesh such that each dimension consists of n processors. Without any loss of generality let us assume that n is a power of 2.

A. Assumptions

Proceesors are assumed to be indexed following the Row Major Indexing Scheme. In other words, if a processor P_i has coordinates $(x_1, x_2, \ldots, x_Q)$, then the i^{th} element in the sorted sequence will be placed in that processor, where i is given by the relation $i = x_1 + (x_2 - 1)n_1 + (x_3 - 1)n_1 n_2 + \ldots + (x_Q - 1)n_1 n_2 \ldots n_{Q-1}$ where x_j varies from 1 to n_j. Further we assume that each processor has three registers, R_r, R_s, and R_t and can perform the following three operations

1) Register Exchange - interchanging the contents of two of its registers
2) Compare Exchange - comparing the contents of R_r and R_s and assign the larger element to R_s and the smaller to R_r or vice versa, depending on the value of the MASK array. This is explained in more detail later.
3) Routing Operation - transferring the contents of the R_r register to the R_r register of one of its immediate neighbors

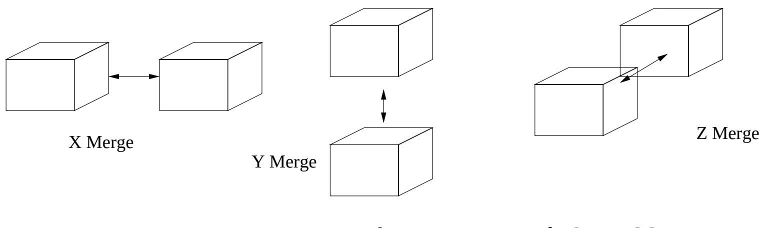

Fig. 1. X-Merge, Y-Merge and Z-Merge in case of a 3-D MCC

B. *Basic Idea*

Similar to the algorithm of Thompson and Kung [TK77], we define here the concept of X-Merge, Y-Merge and Z-Merge, which, as the name suggests, occur along the X, Y and Z axes respectively. We first start with $N = n^3$ bitonic sequences, each of size 1. In each step, we do a X-Merge, Y-Merge and Z-Merge in order. At the end of the i^{th} step, we thus get $N/8^i$ bitonic sequences, each contained in a cube of side 2^i. The process continues, until the entire sequence is sorted in Row Major Indexing order. Figure(1) will make the idea more clear.

The algorithm for the Q Dimensional case is based on this idea. In this case, we have, instead of X-Merge, Y-Merge and Z-Merge, $X_1^Q - Merge$,...,$X_Q^Q - Merge$ where $X_i^Q - Merge$ indicates that the procedure merges two Q-Dimensional blocks which are adjacent along the i^{th} dimension.

C. *Formal Algorithm*

In this section, we present formally our algorithm for bitonic sorting on a Q-Dimensional Mesh Connected Computer. To do so, let us define a few procedures which are to be used in the algorithm. Procedure $X_Q^Q - Merge(n_1, \ldots, n_Q)$ sorts a bitonic sequence stored in a Q-Dimensional block of size $(n_1, n_2, \ldots, n_Q)$ into increasing or decreasing order (depending on the value of MASK). The i^{th} dimension in the block has size n_i. Each processor in the block holds the element to be sorted in its R_r register.

```
**procedure** $X_Q^Q - Merge(n_1, n_2, \ldots, n_Q)$
    1) Let the sequence to be sorted be present in processors $P(i_1, i_2, \ldots, i_Q), \cdots, P(i_1 + n_1, i_2 + n_2, \ldots, i_Q + n_Q)$.
    2) if $n_Q = 1$, goto Step 8.
    3) Processors $P(i_1 + x_1, \ldots, i_Q + x_Q), x_j = 1, \cdots, n_j, for\ j = 1, \ldots, Q-1, x_Q = 1, \ldots, \frac{n_Q}{2}$ perform a register exchange operation between their $R_r$ and $R_s$ registers.
    4) Processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_Q + x_Q)$ route the contents of their $R_r$ registers to processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_Q + x_Q - \frac{n_Q}{2})$, where $x_j = 1, \cdots, n_j, for\ j = 1, \ldots, Q-1, x_Q = \frac{n_Q}{2}, \ldots, n_Q$.
    5) Processors $P(i_1 + x_1, \ldots, i_Q + x_Q), x_j = 1, \cdots, n_j, for\ j = 1, \ldots, Q-1, x_Q = 1, \ldots, \frac{n_Q}{2}$ perform a compare exchange operation.
    6) Processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_Q + x_Q)$ route the contents of their $R_r$ registers to processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_Q + x_Q + \frac{n_Q}{2})$,where $x_j = 1, \cdots, n_j, for\ j = 1, \ldots, Q-1, x_Q = 1, \ldots, \frac{n_Q}{2}$.
    7) Do the following in parallel:
        a) $X_Q^Q - Merge(n_1, n_2, \ldots, \frac{n_Q}{2})$, for processors $P(i_1 + x_1, \ldots, i_Q + x_Q), x_j = 1, \cdots, n_j, for\ j = 1, \ldots, (Q-1), x_Q = 1, \ldots, \frac{n_Q}{2}$
        b) $X_Q^Q - Merge(n_1, n_2, \ldots, \frac{n_Q}{2})$, for processors $P(i_1 + x_1, \ldots, i_Q + x_Q), x_j = 1, \cdots, n_j, for\ j = 1, \ldots, (Q-1), x_Q = \frac{n_Q}{2}, \ldots, n_Q$
    8) $X_{Q-1}^{Q-1} - Merge(n_1, n_2, \ldots, n_{Q-1})$
```

Procedure $X_i^Q - Merge(i \neq Q)$ merges a bitonic sequence of length $n_1 n_2 \cdots n_Q$ contained in a Q-Dimensional block of processors of dimensions $(n_1, n_2, \ldots, n_Q)$. The sequence is held in the R_r registers of these processors. The increasing part of the sequence is held in processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_i + x_i, \ldots, i_Q + x_Q)$, $x_j = 1, \ldots, n_j$, $for\ j \neq i$, $x_i = 1, \ldots, n_i/2$. The decreasing half of the sequence is held in the rest of the processors.

```
**procedure** $X_i^Q - Merge(n_1, n_2, \ldots, n_Q)(i \neq Q)$
    1) Let the increasing part of the sequence be present in processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_i + x_i, \ldots, i_Q + x_Q)$, $x_j = 1, \ldots, n_j, for\ j \neq i, x_i = 1, \ldots, n_i/2$, and the decreasing part in processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_i + x_i, \ldots, i_Q + x_Q), x_j = 1, \ldots, n_j, for\ j \neq i, x_i = n_i/2, \ldots, n_i$.
    2) Processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_i + x_i, \ldots, i_Q + x_Q), x_j = 1, \ldots, n_j, for\ j \neq i, x_i = 1, \ldots, n_i/2$ perform a register-exchange operation between their $R_r$ and $R_s$ registers.
    3) Processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_i + x_i, \ldots, i_Q + x_Q)$ route the contents of their respective $R_r$ registers to processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_i + x_i - n_i/2, \ldots, i_Q + x_Q)$, where $x_j = 1, \ldots, n_j, for\ j \neq i, x_i = n_i/2, \ldots, n_i$
    4) Perform a $Two - Column - Merge(n_Q)$, along the $X_Q$ dimension
    5) Send back the rejected elements of processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_i + x_i - n_i/2, \ldots, i_Q + x_Q)$ back to processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_i + x_i, \ldots, i_Q + x_Q)$, where $x_j = 1, \ldots, n_j, for\ j \neq i, x_i = n_i/2, \ldots, n_i$.
    6) Processors $P(i_1 + x_1, i_2 + x_2, \ldots, i_i + x_i, \ldots, i_Q + x_Q), x_j = 1, \ldots, n_j, for\ j \neq i, x_i = 1, \ldots, n_i/2$ perform a register-exchange operation between their $R_r$ and $R_s$ registers.
    7) For each of the $2n_Q$ $(Q-1)$-Dimensional blocks of dimensions $(n_1, \ldots, n_i/2, \ldots, n_Q)$
       perform $X_{Q-1}^{Q-1} - Merge(n_1, \ldots, n_i/2, \ldots, n_Q)$ in parallel.
```

D. *Main Algorithm*

The main algorithm applies merges alternately in the X_i dimension until the whole sequence gets sorted. As explained earlier, after the i^{th} stage, we get $N/2^{Qi}$ sequences, each of size 2^{Qi}, contained in a Q-Dimensional Block of side length 2^i. All the compare-exchanges are done according to the MASK function. The algorithm is given below.

```
for $i$ = 0 to $\log n - 1$
    $X_1^Q - Merge(2^{i+1}, 2^i, 2^i, \ldots, 2^i)$
    $X_2^Q - Merge(2^{i+1}, 2^{i+1}, 2^i, \ldots, 2^i)$
    $\vdots$
    $X_Q^Q - Merge(2^{i+1}, 2^{i+1}, \ldots, 2^{i+1})$
end for
```

E. *The MASK Function*

MASK function determines whether a sequence is to be sorted in increasing or decreasing order, as we need by the next step of Bitonic Sorting algorithm. The MASK function does so by affecting the result of the compare-exchange instruction of processor $P(i_1, \ldots, i_Q)$ as follows.

If $Mask((i_1, i_2, \ldots, i_Q, P) = 0$ then $R_r = max(a,b)$ and $R_s = min(a,b)$
Else $R_r = min(a,b)$ and $R_s = max(a,b)$

where P is the particular pass number of the algorithm. For example, the first $X_1^Q - Merge$ is said to be pass 1, $X_2^Q - Merge$ is said to be Pass 2 and so on. Thus if the MASK function is 0 for all processors in an array, the array will be sorted in increasing order. The MASK function generalized to Q-Dimensions can be given by

```
function $MASK(i_1, \ldots, i_Q, P)$
    $r = P\%Q$ ; if $i_{r+1}/2^{\lfloor P/Q \rfloor}$ is even return 0 else return 1
```

III. Analysis

Define t_r as the time required to perform one routing operation, t_c be the time required to perform a compare-exchange operation, and t_e be the time required to perform a register-exchange operation. We will express the complexity of our entire algorithm in terms of the time required to perform each of these operations, that is, in terms of t_r, t_e and t_c.

A. *Complexity of* $X_Q^Q - Merge$

Equipped with this notation, let us first determine the complexity of the $X_Q^Q - Merge$'s. Let $T_Q^r(n_1, n_2, \ldots, n_Q)$ be the number of routing operations involved in merging a block of dimension Q and of size $(n_1, n_2, \ldots, n_Q)$ which contains a bitonic sequence. We assume, in addition, that all the n_i's are powers of 2. Then, the recurrence relation holds that

$T_Q^r(n_1, n_2, \ldots, n_Q) = T_Q^r(n_1, n_2, \ldots, \frac{n_Q}{2}) + n_Q$,
$T_Q^r(n_1, n_2, \ldots, n_{Q-1}, 1) = T_{Q-1}^r(n_1, \ldots, n_{Q-1})$ and $T_1^r(1) = 0$.

Solving above equations we get the number of routing operations as

$T_Q^r(n_1, n_2, \ldots, n_Q) = 2(n_1 + n_2 + \ldots + n_Q) - 2Q$

Similarly, let the $T_Q^c(n_1, n_2, \ldots, n_Q)$ be the number of compare-exchange operations and $T_Q^e(n_1, n_2, \ldots, n_Q)$ be the number of register-exchange operations of $X_Q^Q - Merge$. For these, we have the recurrence relations

$T_Q^c(n_1, n_2, \ldots, n_Q) = T_Q^c(n_1, n_2, \ldots, \frac{n_Q}{2}) + 1$, $T_Q^c(n_1, n_2, \ldots, 1) = T_{Q-1}^c(n_1, n_2, \ldots, n_{Q-1})$ and $T_1^c(1) = 0$
$T_Q^e(n_1, n_2, \ldots, n_Q) = T_Q^e(n_1, n_2, \ldots, \frac{n_Q}{2}) + 2$,$T_Q^e(n_1, n_2, \ldots, 1) = T_{Q-1}^e(n_1, n_2, \ldots, n_{Q-1})$ and $T_1^e(1) = 0$

Solving above equations, we get

$T_Q^c(n_1, n_2, \ldots, n_Q) = (\log n_1 + \log n_2 + \ldots + \log n_Q)$ and $T_Q^e(n_1, n_2, \ldots, n_Q) = 2(\log n_1 + \log n_2 + \ldots + \log n_Q)$

B. *Complexity of* $X_i^Q - Merge(i \neq Q)$

Let $F_Q^{ri}(n_1, n_2, \ldots, n_Q)$ denote the number of routing steps taken in merging two blocks of size $(n_1, n_2, \ldots, n_Q)$, adjacent along the i^{th} dimension, where $i \neq Q$. One of the blocks contains the increasing part of a bitonic sequence and the other block the decreasing part.Then, the number of routing steps can be given by

$F_Q^{ri}(n_1, n_2, \ldots, n_Q) = n_i + 2(n_Q - 1) + T_{Q-1}^r(n_1, n_2, \ldots, \frac{n_i}{2}, \ldots, n_{Q-1})$

Here $2(n_Q - 1)$ is the number of routing steps in a Two Column Merge. The relations can be solved to give

$F_Q^{ri}(n_1, n_2, \ldots, n_Q) = 2(n_1 + n_2 + \ldots + n_Q) - 2Q$.

But this does not depend on the value of i at all, hence let us call $F_Q^{ri}(n_1, n_2, \ldots, n_Q)$ as $F_Q^r(n_1, n_2, \ldots, n_Q)$.
Similarly the recurrence relations expressing the number of compare-exchange and register-exchange operations are

$F^c_Q(n_1, n_2, \ldots, n_Q) = (1 + \log n_Q) + T^c_{Q-1}(n_1, \ldots, \frac{n_i}{2}, \ldots, n_{Q-1})$
$F^e_Q(n_1, n_2, \ldots, n_Q) = 3\log n_Q + T^e_{Q-1}(n_1, \ldots, \frac{n_i}{2}, \ldots, n_{Q-1}) + 2.$
Here $(1+\log n_Q)$ and $3\log n_Q$ are the number of compare exchange and register exchange operations in a two-column merge of column size n_Q.These equations can be solved to give
$F^c_Q(n_1, n_2, \ldots, n_Q) = \log n_1 + \log n_2 + \ldots + \log n_Q$
$F^e_Q(n_1, n_2, \ldots, n_Q) = 2(\log n_1 + \log n_2 + \ldots + \log n_{Q-1}) + 3\log n_Q$

C. Total Complexity of the Algorithm

Given the results proved above, it is easy to get the total complexity of the algorithm. Let $S^r_Q(n)$ denote the total number of routing steps taken in the whole algorithm, when the sequence to be sorted is placed in a Q-Dimensional Mesh Connected Computer, with each dimension having size n. Suppose we are at the i^{th} step of the main algorithm and let $K = 2^i$. Then the number of routing operations in the stage i, N^r_i can be expressed as
$N^r_i = F^r_Q(2K, K, \ldots, K) + F^r_Q(2K, 2K, \ldots, K) + \ldots + F^r_Q(2K, 2K, \ldots, 2K, K) + T^r_Q(2K, 2K, \ldots, 2K)$
Solving this, we get $N^r_i = K(3Q^2 + Q) - 2Q^2$.
Hence $S^r_Q(n)$ can be given by $S^r_Q(n) = \Sigma_{i=1}^{\log n-1} N^r_i = (3Q^2 + Q)(n-1) - 2Q^2 \log n$. Let $N = n^Q$ be the total number of elements to be sorted. Then, the routing time complexity of the algorithm is $S^r_Q(N) = (3Q^2 + Q)(\sqrt[Q]{N} - 1) - 2Q\log N$i.
Similarly let $S^c_Q(n)$ denote the total number of compare-exchange steps in the total algorithm and $S^e_Q(n)$ denote the total number of register-exchange steps in the total algorithm. Let the number of compare-exchange operations and register-exchange operations in step i be N^c_i and N^e_i respectively. They are given by the relations
$N^c_i = F^c_Q(2K, K, \ldots, K) + F^c_Q(2K, 2K, \ldots, K) + \ldots + F^c_Q(2K, 2K, \ldots, 2K, K) + T^c_Q(2K, 2K, \ldots, 2K)$
$N^e_i = F^e_Q(2K, K, \ldots, K) + F^e_Q(2K, 2K, \ldots, K) + \ldots + F^e_Q(2K, 2K, \ldots, 2K, K) + T^e_Q(2K, 2K, \ldots, 2K)$
where $K = 2^i$. Putting in the values of the right hand side quantities, the equation becomes $N^c_i = Q^2 i + \frac{Q}{2}(Q+1)$ and $N^e_i = (2Q^2+1)i + (Q^2 + Q + 1)$. Hence $S^c_Q(n)$ and $S^e_Q(n)$ can be given by
$S^c_Q(n) = \Sigma_{i=1}^{\log n-1} N^c_i = \frac{Q^2}{2}(\log n)^2 + \frac{Q}{2}\log n$
$S^e_Q(n) = \Sigma_{i=1}^{\log n-1} N^e_i = \left(Q^2 + \frac{1}{2}\right)(\log n)^2 + \left(Q + \frac{1}{2}\right)\log n$
If we express the complexity in terms of N, the complexity is
$S^c_Q(N) = \frac{1}{2}(\log N)^2 + \frac{1}{2}\log N$
$S^e_Q(N) = \left(1 + \frac{1}{2Q^2}\right)(\log N)^2 + \left(1 + \frac{1}{2Q}\right)\log N$
Hence the total time complexity C of the sorting technique is
$C = ((3Q^2+Q)(\sqrt[Q]{N}-1) - 2Q\log N)t_r + \left(\frac{1}{2}(\log N)^2 + \frac{1}{2}\log N\right)t_c + \left(\left(1 + \frac{1}{2Q^2}\right)(\log N)^2 + \left(1 + \frac{1}{2Q}\right)\log N\right)t_e$
Assuming that it takes constant time to perform each of the operations routing, compare exchange and register exchange, the time complexity is $O(\sqrt[Q]{N})$. Therefore the cost is $O(N\sqrt[Q]{N})$.

D. Cost Optimal Algorithm

The algorithm presented above is not cost-optimal, however it can be made cost optimal if we consider model, which has more powerful but lesser number of processors. Suppose N element are to be sorted, and there are $P = p^Q$ processors for the purpose. The processors are arranged in a Q-Dimensional mesh, with each dimension containing p processors. In this case we assign N/P elements to each processor. Each processor first sorts sequentially the N/P elements given to it. Then the algorithm mentioned above is applied to perform bitonic merging of these sequences, until we get a completely sorted sequence.
Note that sequential sorting has the time complexity of $O((N/P)\log(N/P)$ and steps like routing, compare-exchange, and register-exchange operations take $O(Q^2 N \sqrt[Q]{[P]})$, $O((\log P)^2)$, $O(\frac{1}{Q^2}(\log P)^2)$ time respectively. Hence for $P\sqrt[Q]{P} \leq \log N$ or $P \leq (\log n)^{\frac{Q}{Q+1}}$, this algorithm becomes cost optimal.

IV. Conclusion

In this paper we have proposed a generalized bitonic sorting algorithm for sorting N elements on a Q-Dimensional Mesh Connected Computer. The time complexity of our algorithm is $O(\sqrt[Q]{N})$, and the cost is $O(N\sqrt[Q]{N})$ in a Mesh Connected Computer with N processors arranged in the form of a Q Dimensional Mesh, with $\sqrt[Q]{N}$ processors in each dimension. The algorithm has also been modified to get a cost optimal algorithm for performing bitonic sort in a Q-Dimensional Mesh Connected Computer.

References

[TK77] C. D. Thompson and H. T. Kung. Sorting on a mesh-connected parallel computer. *CACM*, 20, 1977.