A MODIFIED MESH INTERCONNECTION NETWORK FOR FASTER SORTING

Apurv Gupta, Asha Tarachandani and P. Gupta
Department of Computer Science & Engineering
Indian Institute of Technology,
Kanpur 208 016, India
e-mail: pg@iitk.ac.in

ABSTRACT

In this paper, we propose a modification of the well known $m \times m$ mesh interconnection network. The modification is done by adding some more connections to some of the processors. However a processor can have at most six connections which leads $O(m^2)$ number of connections. We have implemented bitonic sorting algorithm on the network. It is shown that the time complexity of the algorithm reduces to $O(\sqrt{m} \log m)$ due to this modification.

1. INTRODUCTION

The design of interconnection network has been one of the most fundamental challenges in parallel processing. A major problem in design of efficient parallel algorithms lies in devising effective mapping of different known sequential algorithms on suitable parallel machines.

In this paper, we propose a new interconnection network, which is essentially a modification of the well known 2-D MCC. A few extra links between processors have been added. However, the order of number of conncetions remains the same. By adding these connection intelligently, it is possible to reduce the time complexity of some of the widely implemented algorithms on the mesh network.

We use the Bitonic sort as an illustration of the power of our model. Batcher's bitonic sort is based on merging a bitonic sequence into a sorted one [1]. This algorithm has been implemented on 2-dimensional MCC in [2]. The parallel algorithm takes $O(m)$ time to sort m^2 elements on an $m \times m$ MCC. This algorithm can be implemented on our model in $O(\sqrt{m} \log m)$.

The rest of the paper is organized as follows. The modified model has been described in section 2. In the next section the bitonic algorithm on MCC has been discussed. We have implemented the bitonic sorting algorithm on the modified MCC in section 4 while in the next section we have tried to optimize the time complexity with respect to various constraints. We conclude in section 6.

2. THE MODIFIED 2-D MCC

In a two dimensional MCC, the processors are connected to form a two dimensional array of size $m \times m$. The processor $P_{i,j}$ denotes the processor in the column j of the row i. Each processor $P_{i,j}$ is connected to its neighbour processors $P_{i-1,j}$, $P_{i,j-1}$, $P_{i+1,j}$, $P_{i,j+1}$ if they exist. The index I of a processor $P_{i,j}$ is $i * m + j$ (i.e., processors are indexed left to right, top to bottom). In sorting problem each data item is to be routed to a distinct processor such that processor with index i would contain i-th smallest element, for $1 \leq i \leq n$.

For a mXm mesh, we propose to add a few more connections for each row and column, so that the time to route the elements from one processor to another reduces. These connections are laid so that the routing time in various stages of bitonic sort decreases. Since routing is the costliest instruction among all, reducing routing time reduces the overall time complexity of bitonic sort. The scheme of modified mesh is as follows. Apart from the connections that exist in an ordinary 2-D mesh, the following connections are laid. For $t =$ 0, 1,2, ..., $(2m - r)/(2s)$-1

1. processor $P(i, t * s + r/2)$ is connected to the processor $P(i, t * s + s - 1)$, for each row i,

2. processor $P(t * s + r/2, j)$ is connected to the processor $P(t * s + s - 1, j)$, for each column j,

where the r and s are power of 2, $1 \leq s \leq \frac{r}{2}$ and $2 \leq r \leq m$.

Thus, the number of extra connections being introduced are $(2m - r)/(2s)$ per row and per column. The total number of connections is $2 * m(m-1) + \frac{m}{s}(2m - r)$ which is also $O(m^2)$. Figure 1 illustrates some examples of the model. For simplicity we have depicted one row of 8×8 2-D MCC. It is easy to show that a processor may have at most 6 connections.

In the modified mesh, a row (or column) of m elements can be thought as $r/(2s)$ *disjoint* paths of $2ms/r$ elements each. There may be more than one path and with the help of these paths, the elements of a row (or column) can be routed.

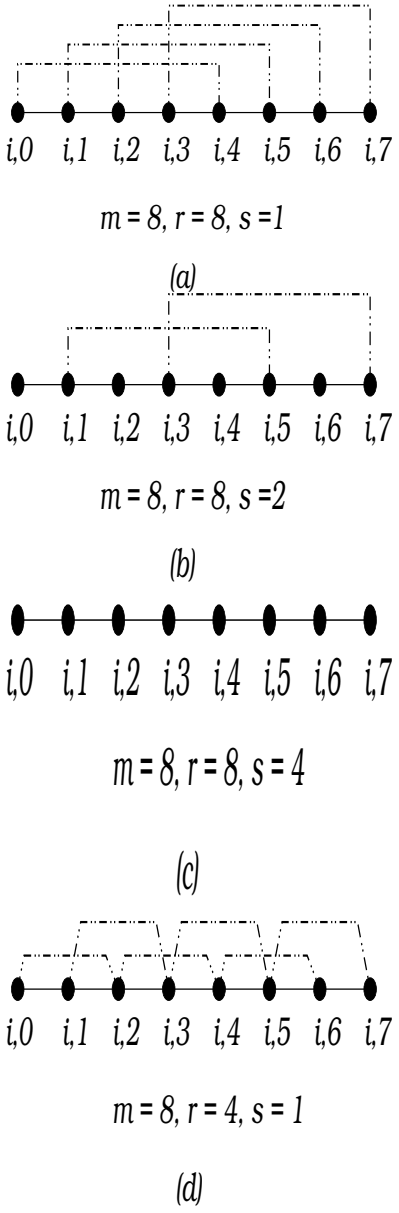

Figure 1. i-th Row of 8×8 Modified Mesh

3. BITONIC SORTING ON 2-D MCC

A sequence $X = (x_1, x_2 \ldots x_n)$ is said to be bitonic if either (i) there exists an index $i, 1 \leq i \leq n$, satisfying $x_1 \leq x_2 \leq \ldots \leq x_i \geq x_{i+1} \geq \ldots \geq x_n$ or (ii) a sequence can be obtained by cyclic shifting of the original sequence which satisfies the first condition. From the definition it can be shown that if X is a bitonic sequence then subsequences $D = (d_1, d_2, d_3, \ldots, d_{\frac{n}{2}})$ and $E = (e_1, e_2, e_3, \ldots, e_{\frac{n}{2}})$ are also bitonic where $d_i = \min(x_i, x_{\frac{n}{2}+i})$ and $e_i = \max(x_i, x_{\frac{n}{2}+i})$. Further, $\max(d_1, d_2, \ldots, d_{\frac{n}{2}}) \leq \min(e_1, e_2, \ldots, e_{\frac{n}{2}})$. Batcher's algorithm to sort a bitonic sequence X first defines the two subsequences D and E from X and then sorts recursively D and E. Any sequence $Y = (y_1, y_2, \ldots y_n)$ may be sorted recursively by sorting the subsequence $(y_1, y_2, \ldots y_{\lceil \frac{n}{2} \rceil})$ into non-decreasing order, the subsequence $(y_{\lceil \frac{n}{2} \rceil + 1}, y_{\lceil \frac{n}{2} \rceil + 2}, \ldots y_n)$ into non-increasing order (or vice-versa) and then sorting the resulting bitonic sequence $(y_1, y_2, \ldots y_n)$ into non-decreasing order using Batcher's method.

The idea of bitonic sort can be mapped directly on a $m \times m$ MCC. In the 1st stage the $m \times m$ array of input elements is treated as $\frac{m^2}{2}$ sub-arrays, each of size 1×2 that are sorted individually so that we get $\frac{m}{2} \times \frac{m}{2}$ subarrays each being a bitonic sequence of 4 elements. In stage 2 each of the $\frac{m}{2} \times \frac{m}{2}$ subarrays is sorted in such a way that we get $\frac{m}{2} \times \frac{m}{4}$ subarrays each being a bitonic sequence of 8 elements. This procedure is iterated till one gets the sorted sequence of m^2 elements.

In order to implement Batcher's bitonic sort on the 2-D MCC, Nassimi and Sahni [2] have assumed to have three registers to each processor and they are routing register R_r and two storage registers namely, R_t and R_s; each register is capable of storing one element of the sequence. Each processor can execute the following three instructions.

1. Register-Exchange instruction : for unconditional changing contents of two of its registers by each active processor. The same registers are used for all processors.

2. Route instruction : for routing the contents of their R_r registers of all active processors to their immediate neighbour in the same direction. Thus, this instruction simply shifts the entire R_r-array by unit-distance in one of the four directions viz., up, down, left or right.

3. Compare-Exchange instruction : for comparing and interchanging contents of R_r and R_s by each active processor.

Note that the maximum number of routing steps will be required when first processor contains maximum and last processor contains minimum element. Exactly $2m - 1$ routing steps are needed to bring the maximum element from $P_{0,0}$ to $P_{m-1,m-1}$ and that many routing steps for bringing the minimum element from $P_{m-1,m-1}$ to $P_{0,0}$. Thus, total $4m - 2$ routing steps are required to interchange data between from $P_{0,0}$ and $P_{m-1,m-1}$. This implies, no algorithm can sort m^2 elements on an $m \times m$ MCC in fewer than $\Omega(m)$ steps.

The algorithm proposed by Nassimi and Sahni [2] used to call *Horizontal Merge* to sort bitonic sequences at the odd indexed iteration while it used to call *Vertical Merge* to sort the bitonic sequences at the even indexed iterations. They considered three types of time complexities and they are complexity due to *Comparison-Exchanges*, *Register-Exchanges* and *Routings*. Let $T^C_{VM}(J,K)$, $T^E_{VM}(J,K)$ and $T^R_{VM}(J,K)$ be the respective time complexities to sort the $J \times K$ bitonic sequence using *Vertical Merge* algorithm. Then, it is shown that

$T^C_{VM}(J,K) = \log(JK)$
$T^E_{VM}(J,K) = 2\log(JK)$
$T^R_{VM}(J,K) = 2(J+K)-4$

Again, let $T^C_{HM}(J,K)$, $T^E_{HM}(J,K)$ and $T^R_{HM}(J,K)$ be the time complexities due to comparison- exchanges, register-exchanges and routings respectively, to sort the $J \times K$ bitonic sequence using *Horizontal Merge* algorithm, one gets

$T^C_{HM}(J,K) = \log(JK)$
$T^E_{HM}(J,K) = 3\log J + 2\log K$
$T^R_{VM}(J,K) = 2(J+K)-4$

Finally, various time complexities to sort $m \times m$ arbitrary elements using 2-D MCC are obtained as follows:

$$\begin{aligned} T^C(m,m) &= 2\log^2 m + 5\log m - 4 \\ T^E(m,m) &= 4.5\log^2 m + 10.5\log m - 9 \\ T^R(m,m) &= 14(m-1) - 8\log m \end{aligned}$$

where T^C, T^E and T^R are time complexities due to comparison-exchanges, register-exchanges and routings respectively. Note that the time complexity of the algorithm to sort $m \times m$ elements on 2-D MCC is $O(m)$ which is due to the routings.

4. SORTING ON THE MODIFIED 2-D MCC

In this section we will discuss how the bitonic sorting algorithm can be implemented on the proposed model. We will follow the same algorithm presented by Nassimi & Sahni [2] but routing will be taken place using the new connections. Hence the number of comparison- exchange steps and register-exchange steps is of $O(\log m)$ and will not be affected by our change in routing scheme.

For routing $\frac{K}{2}$ elements in a row (or column) to the adjacent $\frac{K}{2}$ elements, the number of steps needed are:

$$R(K) = \begin{cases} \frac{K}{2}, K < r \\ \frac{sK}{r}, K \geq r \end{cases}$$

Clearly $R(K)$ is the time taken to route one half of the array of K numbers to the other half. Let us define

$$A(K) = \sum_{i=1}^{\log K} R(2^i) \quad \text{and}$$

$$B(K) = \sum_{i=1}^{\log K} A(2^i)$$

It is easy to derive the formulas for $A(k)$ and $B(K)$. These follow from simple summation. For $K < r$,

$$\begin{aligned} A(K) &= \sum_{i=1}^{\log K} R(2^i) \\ &= \sum_{i=1}^{\log K} 2^{i-1} \\ &= K-1 \end{aligned}$$

Again for $K \geq r$,

$$\begin{aligned} A(K) &= \sum_{i=1}^{\log K} R(2^i) \\ &= \sum_{i=\log r}^{\log K} R(2^i) + A(r/2) \\ &= \frac{r}{2} - 1 + \sum_{i=\log r}^{\log K} (s2^i/r) \\ &= s[\tfrac{2K}{r} - 1] + \tfrac{r}{2} - 1 \end{aligned}$$

Hence we have the following:

Theorem 1 *$A(K)$ satisfies*

$$A(K) = \begin{cases} K-1, K < r \\ s[\frac{2K}{r} - 1] + r/2 - 1, K \geq r \end{cases}$$

Summation of $B(K)$ is similar. For $K < r$,

$$\begin{aligned} B(K) &= \sum_{i=1}^{\log K} A(2^i) \\ &= \sum_{i=1}^{\log K} (2^i - 1) \\ &= 2K - \log K - 2 \end{aligned}$$

Further, $B(r/2) = r - \log r - 1$.

Again for $k \geq r$, we get

$$\begin{aligned} B(K) &= \sum_{i=1}^{\log K} A(2^i) \\ &= \sum_{i=\log r}^{\log K} A(2^i) + B(r/2) \\ &= \sum_{i=\log r}^{\log K} A(2^i) + r - \log r - 1 \\ &= \sum_{i=\log r}^{\log K} (s[\tfrac{2^{i+1}}{r}) + \sum_{i=\log r}^{\log K} (r/2 - s - 1) \\ &\quad + r - \log r - 1 \\ &= 2s[\tfrac{2K}{r} - 1] + (r/2 - s - 1)[\log(K/r) + 1] \\ &\quad + r - \log r - 1 \end{aligned}$$

Thus we have

Theorem 2 *The value of $B(K)$ is*

$$B(K) = \begin{cases} 2K - \log K - 2, K < r \\ 2s[\frac{2K}{r} - 1] + (\frac{r}{2} - s - 1) \\ \qquad (\log(\frac{K}{r}) + 1) + r - 1 - \log r, K \geq r \end{cases}$$

Let us now obtain the number of routing steps needed by the sorting algorithm. It can be shown that to merge a $J \times K$ bitonic sequence the number of routing steps needed by *Vertical Merge* is $2A(J) + 2A(K)$ and that needed by *Horizontal Merge* is $2A(J) + 2R(K) + 2A(K/2)$. Thus the algorithm to sort $m \times m$ elements on 2-D modified MCC needs the $T^R(m \times m)$ routing steps where $T^R(m \times m)$ is

$$\begin{aligned} T^R(m \times m) &= \sum_{i=0}^{\log(m/2)} \{2A(2^{i+1}) + 2A(2^{i+1}) \\ &\quad + 2A(2^i) + 2R(2^{i+1}) + 2A(2^i)\} \\ &= 4B(m) + 4B(m/2) + 2A(m) \end{aligned}$$

Using Theorem 1 and Theorem 2, we get:

$$\begin{aligned} T^R(m \times m) &= 8s[\tfrac{2m}{r} - 1] + 4(\tfrac{r}{2} - s - 1)(\log(\tfrac{K}{r}) + 1) \\ &\quad + r - 1 - \log r + 8s[\tfrac{m}{r} - 1] \\ &\quad + 4(\tfrac{r}{2} - s - 1)(\log(\tfrac{m}{2r}) + 1) + r - 1 \\ &\quad - \log r + 2s[\tfrac{2m}{r} - 1] + r/2 - 1 \\ &= 28\tfrac{sm}{r} - 18s + 8(r - 1 - \log r) + \tfrac{r}{2} \\ &\quad - 1 + 4(\tfrac{r}{2} - s - 1) + 8(\tfrac{r}{2} - s - 1)\log(\tfrac{m}{r}) \end{aligned}$$

Note that this formula is valid only for $r \leq m/2$. Otherwise the different expressions for $A(K)$ and $B(K)$ must be used. This is not a problem as we can clearly see that we would never really use $r = m$ since we gain in only the largest routing step. Thus,

Theorem 3 *Bitonic sorting algorithm to sort m^2 elements using $m \times m$ modified mesh takes $O(\sqrt{m}\log m)$ time.*

5. OPTIMIZING TIME COMPLEXITY

Let us try to optimize this time with respect to the parameters r and s. Since the derivative of this equation is a transcendental one, which doesn't lend itself to a solution in closed form, we plot a graph of $T^R(m\times m)$ versus $\log r$ for various values of s. It is easy to see $s = O(1)$ and $r = \sqrt{m}$ lead to a time complexity of $T^R(m \times m) = O(\sqrt{m}\log m)$. Also it is interesting to note the quantity the gain in time complexity per extra connection.

6. CONCLUSION

In this paper we have explored a new interconnection network that can reduce the time complexity of bitonic sort on a mesh like interconnection network. By adding the same order of connection we have seen that we could gain the overall time complexity of the bitonic sorting algorithm on 2-D MCC. It remains open to study other problems on this network.

References

[1] Knuth, D. E., "The Art of Computer Programming, Vol 3, Sorting and searching", Addison-Wesley, Reading, Mass, 1973.

[2] Nassimi, D. & Sahni, S., "Bitonic Sort on A Mesh-connected Parallel Computer", *IEEE Trans. Comput.*, C-28(1), 2-7, 1979.