J Supercomput (2012) 59:920–940
DOI 10.1007/s11227-010-0479-y

OTIS-MOT: an efficient interconnection network for parallel processing

Prasanta K. Jana · Dheeresh K. Mallick

Published online: 9 September 2010

Abstract Mesh of trees (MOT) is well known for its small diameter, high bisection width, simple decomposability and area universality. On the other hand, OTIS (Optical Transpose Interconnection System) provides an efficient optoelectronic model for massively parallel processing system. In this paper, we present OTIS-MOT as a competent candidate for a two-tier architecture that can take the advantages of both the OTIS and the MOT. We show that an n^4_- processor OTIS-MOT has diameter $8\log n^* + 1$ (The base of the logarithm is assumed to be 2 throughout this paper.) and fault diameter $8\log n + 2$ under single node failure. We establish other topological properties such as bisection width, multiple paths and the modularity. We show that many communication as well as application algorithms can run on this network in comparable time or even faster than other similar tree-based two-tier architectures. The communication algorithms including row/column-group broadcast and one-to-all broadcast are shown to require $O(\log n)$ time, multicast in $O(n^2 \log n)$ time and the bit-reverse permutation in $O(n)$ time. Many parallel algorithms for various problems such as finding polynomial zeros, sales forecasting, matrix-vector multiplication and the DFT computation are proposed to map in $O(\log n)$ time. Sorting and prefix computation are also shown to run in $O(\log n)$ time.

Keywords Interconnection network · Optical transpose interconnection systems (OTIS) · Mesh of trees (MOT) · Topological properties · Communication algorithms

P.K. Jana (✉)
Department of Computer Science and Engineering, Indian School of Mines, Dhanbad 826 004, India
e-mail: prasantajana@yahoo.com

D.K. Mallick
Department of Computer Science and Engineering, Birla Institute of Technology, Mesra, Ranchi 835 215, India
e-mail: dkmallick@gmail.com

1 Introduction

An interconnection network is the heart of an SIMD model of a massively parallel processing system. Node degree, diameter, bisection width, fault tolerance and decomposability are the common issues that are generally considered for the evaluation of an interconnection network. However, among all, the diameter is the most important factor that puts the lower bound on the time complexity for many parallel algorithms, especially for those which require communication between arbitrary pair of nodes. MOT (Mesh of Trees) is an efficient interconnection network which is well known for its small diameter, large bisection width and area universality [22, 30, 36]. In the recent years, MOT has created a lot of interest among researchers. MOT has been shown as an efficient interconnection network for single-chip parallel processing by Balkan [3–5]. It has also been studied for FPGA [27] and MFPGA [28] architectures by Marrakchi et al. This network can provide logarithmic time algorithms for many problems such as matrix-vector multiplication, sorting, packet routing, prefix computation, minimum spanning trees, convex hull and so on [16, 22, 30]. Algorithms for other computations, including image compression, broadcasting and gossiping and Euclidean distance [1, 21, 37], have been efficiently developed on this architecture.

OTIS (Optical Transpose Interconnection System) [29] is an efficient model of optoelectronic parallel computers that has drawn enormous attention. This network (also known as swapped interconnection network [31]) uses both the electronic and the optical links. Several parallel algorithms have been developed on different models of OTIS for various operations such as matrix multiplication [39], polynomial interpolation and root finding [17], prefix computation [19], conflict graph construction [23], image processing [40], basic operations [41], BPC permutations [34], randomized routing, selection and sorting [33]. Topological properties of various OTIS-networks have also been studied by several authors that can be seen in [9, 10, 31, 32, 35, 44].

In this paper, we present OTIS-MOT as efficient two-tier architecture that bridges between the OTIS and the MOT network. We show that several parallel algorithms can be efficiently mapped to run in comparable or even faster time than other two-tier similar architectures. We do not propose here the OTIS-MOT as a new interconnection network as it has already been referred to in [35, 42]; rather, we further study to explore many interesting topological properties of the OTIS-MOT. However, our primary contribution of this paper is to map various communication and application algorithms on this architecture. To the best of our knowledge, no separate research papers either on the topological properties or on the parallel algorithms have been published for OTIS-MOT except for the parallel prefix [26] and the parallel enumeration sorting [24] reported by us.

Many two-tier architectures such as Mesh-Connected Trees [12], Multi-Mesh of Trees [18] and OMULT [38] have been proposed that are built around binary trees. However, processor interconnectivity following the transpose rule of OTIS (discussed later) can make the OTIS-MOT more significant than other tree-based networks in terms of topological properties and algorithm mapping. We show that the diameter of an n^4-processor OTIS-MOT is $8\log n + 1$ and the single-node fault diameter is $8\log n + 2$. The total number of links is $\frac{n^2(5n+1)(n-1)}{2}$ and the bisection width is $\frac{n^4}{4}$. The decomposition of the network is shown as a result of removal of $\frac{n^2(n^2-1)}{2}$ links. We develop various communication as well as application algorithms on OTIS-MOT as follows. The communication algorithms such as row/column-group broadcast, one-to-all broadcast are all shown to run in $O(\log n)$ time. Other communication algorithms including multicast and the bit-reverse permutation are mapped in $O(n^2 \log n)$ and $O(n)$ time respectively. Many application algorithms including summation, polynomial root finding, matrix-vector multiplication, forecasting, prefix computation and the sorting are shown to run in $O(\log n)$ time.

The paper is organized as follows. The related works are discussed in Sect. 2. The graph topology of the OTIS-MOT is described in Sect. 3. The topological properties are discussed in Sect. 4. Communication algorithms and the application algorithms are presented in Sects. 5 and 6 respectively. Section 7 concludes our paper.

2 Related works

A great deal of work has been carried out on OTIS architecture with respect to topological properties and algorithm mapping. OTIS architecture was initially proposed by Marsden et al. [29]. Later Zane et al. [44] showed how the OTIS connections can be exploited to develop a large-scale parallel processing system with a given network topology using a small copies of similar topology. Accordingly, they presented the design of OTIS-Mesh, OTIS-Hypercube and OTIS-expander in their paper. Khaled Day et al. [9] studied topological properties of an arbitrary OTIS network and developed one-to-one and optimal broadcasting algorithms. Behrooz Parhami [31] studied OTIS-networks which he called swapped interconnection networks and developed routing, Hamiltonicity and other topological properties. A number of parallel algorithms have been reported on OTIS network. However, we report only the works related to our research in this paper. Wang and Sahni [41] presented various basic operations on OTIS-Mesh of n^4 processors. Their summation algorithm was shown to require $8(n-1)$ electronics moves + 1 OTIS move on the SIMD and $4n$ electronic moves + 1 OTIS move on the MIMD model of the OTIS-Mesh, respectively. Their broadcast and prefix sum algorithms were shown to map in $4(n-1)$ electronic moves + 1 OTIS move and $7(n-1)$ electronic moves + 2 OTIS moves, respectively. The authors also proposed various matrix multiplications on the same architecture, i.e., n^4-processor OTIS-Mesh that can be found in [39]. Their matrix-vector multiplication requires $4(n-1)$ electronic moves + 1 OTIS move using GRM and $8(n-1)$ electronic moves + 2 OTIS moves using GSM mapping scheme. Their matrix-matrix multiplication requires $(8n^2 + O(n))$ electronic moves + (n^2+1) OTIS moves using GRM and $(4n^2 + O(n))$ electronic moves + $O(n)$ OTIS moves using GSM mapping. Parallel prefix on optical platform has been reported by several researchers. The authors in [41] developed algorithms for n-point parallel prefix computation with $(8n^{1/4}-1)$ electronic moves and 2 OTIS moves for both SIMD and MIMD models of an n-processor OTIS-Mesh. In the same paper, they also modified their algorithm to run in $(7n^{1/4}-1)$ electronic moves and 2 OTIS moves. Jana and Sinha [19] reported an improved algorithm with $(5.5n^{1/4}+3)$ electronic moves and 2 OTIS moves on the same model. Sinha and Bandyopadhyay [38] proposed a parallel prefix algorithm on optical multi-trees (OMULT) with $O(\log n)$ electronic moves + 5 optical moves for n^2 data points using $(2n^3 - n^2)$ processors. Parallel sorting algorithms have been also developed on OTIS platform. The sorting algorithm [41] on OTIS-Mesh is based on Leighton's column sort [22] and was shown to run in $(22n^{1/2} + o(n^{1/2}))$ electronic moves + $O(n^{3/8})$ OTIS moves for SIMD model and $(11n^{1/2} + o(n^{1/2}))$ electronic moves + $O(n^{3/8})$ OTIS moves for MIMD model. Rajasekaran and Sahni [33] developed randomized sorting algorithm on OTIS-Mesh with $(8n^{1/2} + o(n^{1/2}))$ steps. Parallel algorithms for polynomial root finding can be found in [17]. The algorithms are based on Durand–Kerner method and implemented on an OTIS-Mesh. For n-degree polynomial, the first version was shown to require $6(n^{1/2} - 1)$ electronic moves + 2 OTIS moves per iteration using n^2 processors. However, with the assumption that data points are already stored in the processors, the same algorithm requires $2(n^{1/2} - 1)$ electronic moves + 1 OTIS move. The second version requires $4(n^{1/2} - 1)$ electronic moves + 1 OTIS move per iteration. Assuming that the initial data points are already stored, this algorithm was shown to require $2(n^{1/2} - 1)$ electronic moves + 1 OTIS move.

3 Graph topology of OTIS-MOT

An $n \times n$ OTIS-MOT is built around n^2 groups, each group (also called block) being an $n \times n$ MOT. Therefore, there are in total n^4 processors laid in a two-dimensional lattice as shown in Fig. 1. The processors within each group are connected by usual electronic links whereas the processors of different groups are interconnected by optical links. Optical links have larger bandwidth than electronic links and transfer times including latency along them are different. The links are shown by solid and the dashed lines respectively in Fig. 1. Let the group placed in the ith row and the jth column be denoted by $\mathrm{G}(i, j)$. Then the processor placed in the kth row and the lth column within the group $\mathrm{G}(i, j)$ is denoted by $\mathrm{P}(i, j, k, l)$ for $1 \le i, j, k, l \le n$.

Definition 1 (Intra-block links) These are electronic links that connect the processors within a group following MOT topology described as follows.

(1) The processors in each row are connected to form a binary tree (called row-tree) rooted at $\mathrm{P}(i, j, k, 1)$, i.e., $\forall i, j$, $1 \le i, j \le n$, $\mathrm{P}(i, j, k, l)$ is directly connected to the processors $\mathrm{P}(i, j, k, 2l)$ and $\mathrm{P}(i, j, k, 2l + 1)$, $1 \le k, l \le n$, if they exist.
(2) Similarly, the processors in each column are also connected to form a binary tree (column-tree) rooted at $\mathrm{P}(i, j, 1, l)$, i.e., $\forall i, j$, $1 \le i, j \le n$, $\mathrm{P}(i, j, k, l)$ is directly connected to the processors $\mathrm{P}(i, j, 2k, l)$ and $\mathrm{P}(i, j, 2k + 1, l)$, $1 \le k, l \le n$, if they exist.

As an example, the intra-block connectivity for $n = 5$ is shown in Fig. 2 in which two indices are used to show the binary tree connectivity for the row-tree as well as the column-tree.

Definition 2 (Inter-block links) These are the optical links by which the processors of a group are connected to the processors of other groups following the transpose rule, i.e., the processor $\mathrm{P}(i, j, k, l)$ is directly connected to the processor $\mathrm{P}(k, l, i, j)$.

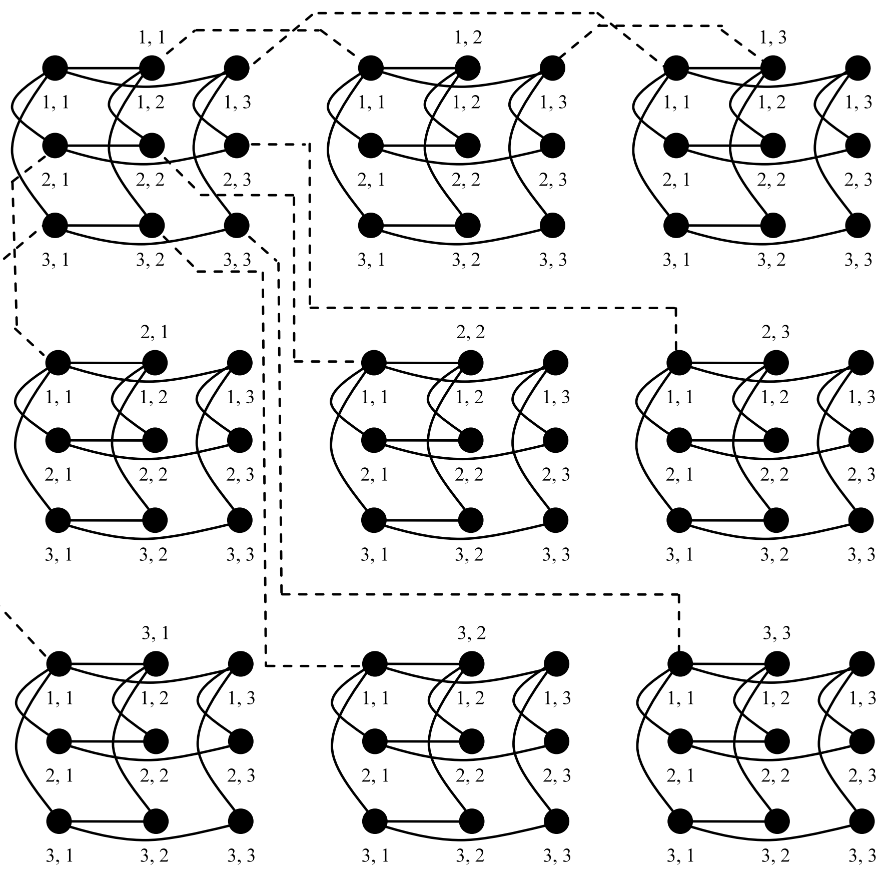


Fig. 1 Graph topology of 3 × 3 OTIS-MOT. Each group is labeled with two indices shown above the group. Similarly, processor nodes within each group are labeled with two indices shown below the nodes. Optical links and the electronic links are the dashed and solid lines respectively. For cleanness, the optical links only for the block G(1, 1) are shown in this figure

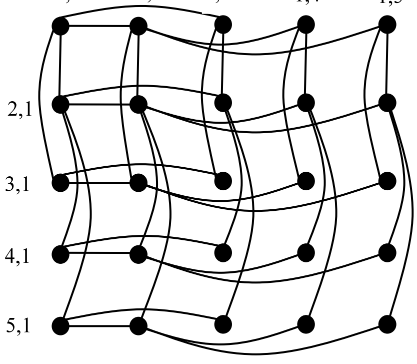


Fig. 2 Interconnectivity of 5 × 5 mesh of trees. For cleanness, the optical links only for the block G(1, 1) are shown in this figure

Definition 3 (Transit node) This is a node v which has an OTIS link with a node lying inside a group other than v's group. Note that a node which has the same index for both the processor and the group fails to be a transit node. Therefore, every block has a total of $n^2 - 1$ transit nodes.

We assume that all the links are bidirectional. For the rest of the paper, we use group and block interchangeably unless stated otherwise.

4 Topological properties

4.1 Diameter

Theorem 1 *The diameter of an $n \times n$ OTIS-MOT network is* $8\log n + 1$.

Proof Let $\mathrm{P}(\alpha_1, \beta_1, i_1, j_1)$ and $\mathrm{P}(\alpha_2, \beta_2, i_2, j_2)$ be any two nodes denoting the source and the destination nodes. Then we term $\mathrm{G}(\alpha_1, \beta_1)$ and $\mathrm{G}(\alpha_2, \beta_2)$ as the source and the destination block respectively. Note that any block in the OTIS-MOT is directly connected to all other blocks through a single optical link. Therefore, if $\alpha_1 \neq \beta_1$ and $\alpha_2 \neq \beta_2$, we can reach the destination block $\mathrm{G}(\alpha_2, \beta_2)$ from the source block $\mathrm{G}(\alpha_1, \beta_1)$ just by traversing only one inter-block link (optical). Now each block is an $n \times n$ MOT having diameter $4\log n$. Therefore, we can reach from the source node $\mathrm{P}(\alpha_1, \beta_1, i_1, j_1)$ to the transit node $\mathrm{P}(\alpha_1, \beta_1, \alpha_2, \beta_2)$ in $4\log n$ time. Then from $\mathrm{P}(\alpha_1, \beta_1, \alpha_2, \beta_2)$, we can reach $\mathrm{P}(\alpha_2, \beta_2, \alpha_1, \beta_1)$ through a single optical link and eventually reach the destination node $\mathrm{P}(\alpha_2, \beta_2, i_2, j_2)$ by traversing another path of length $4\log n$. Thus, it requires traversing a total path of length $8\log n$ (electronic) + 1 (OTIS) from the source node to the destination node. □

4.2 Fault diameter

Let any source–destination pair be denoted by $\langle P(i_1, j_1, k_1, l_1), P(i_2, j_2, k_2, l_2)\rangle$ and for the brevity assume $i_1 \neq i_2$ and $j_1 \neq j_2$. Then we have the following results.

Lemma 1 *If the faulty processor lies on the same row (column)-tree to which the source processor* $\mathrm{P}(i_1, j_1, k_1, l_1)$ *belongs, then the diameter remains same, i.e.,* $8\log n + 1$.

Proof As the faulty processor lies on the row (column)-tree, we can bypass it by traversing the shortest path between the source and the destination through the column (row)-tree of the source processor as follows.

$$\mathrm{P}(i_1, j_1, k_1, l_1) \xrightarrow{\text{Step 1}} \mathrm{P}(i_1, j_1, i_2, j_2) \xrightarrow{\text{Step 2}} \mathrm{P}(i_2, j_2, i_1, j_1) \xrightarrow{\text{Step 3}} \mathrm{P}(i_2, j_2, k_2, l_2)$$

where $\mathrm{P}(i_1, j_1, i_2, j_2)$ is the transit node, and the steps are:

Step 1. Traverse the column (row)-tree followed by the row (column)-tree in $4\log n$ electronic moves.

Step 2. Perform one OTIS move.
Step 3. Traverse the row (column)-tree followed by the column (row)-tree in $4\log n$ electronic moves.

Therefore it requires traversing the path of length $8\log n$ (electronic) + 1 (OTIS). □

Lemma 2 *If the transit node* $\mathrm{P}(i_1, j_1, i_2, j_2)$ (*defined in the proof of Lemma* 1) *is faulty*, *then the diameter is increased by at most one OTIS link*, *i.e.*, *the fault diameter in this case is* $8\log n + 2$.

Proof As the transit node $\mathrm{P}(i_1, j_1, i_2, j_2)$ is faulty, we can reach the destination node from the source node $\mathrm{P}(i_1, j_1, k_1, l_1)$ via other block to bypass faulty transit node. The successive steps are as follows.

Step 1. Perform one OTIS move to reach the node $\mathrm{P}(k_1, l_1, i_1, j_1)$.
Step 2. Traverse the column (row)-tree followed by the row (column)-tree in $4\log n$ electronic moves to reach a transit node $\mathrm{P}(k_1, l_1, i_2, j_2)$ of the block other than that of the source block.
Step 3. Perform another OTIS move to reach the node $\mathrm{P}(i_2, j_2, k_1, l_1)$.
Step 4. Finally, traverse the row (column)-tree followed by the column (row)-tree in $4\log n$ electronic moves to reach the destination node $\mathrm{P}(i_2, j_2, k_2, l_2)$.

Thus it requires traversing a path of length $8\log n$ (electronic) + 2 (OTIS). □

Theorem 2 *The diameter of the OTIS-MOT in the presence of a single node failure is* $8\log n + 2$.

Proof Directly follows from Lemma 1 and Lemma 2. □

4.3 Number of links

Lemma 3 *An* $n \times n$ *OTIS-MOT has* $2n^3(n-1)$ *intra-block* (*electronic*) *links.*

Proof Each block of the OTIS-MOT is an $n \times n$ MOT with binary tree connection in every row and every column having $n-1$ links for each. There are n such row-trees and n column-trees in each block; therefore the total number of links is $2n(n-1)$ in each block. However, there are n^2 blocks in the OTIS-MOT. Therefore, total number of intra-block links is $2n^3(n-1)$. □

Lemma 4 *An* $n \times n$ *OTIS-MOT has* $\frac{n^2(n^2-1)}{2}$ *inter-block links.*

Proof Each block of the OTIS-MOT has n^2-1 OTIS links that connect other n^2-1 blocks. There are n^2 such blocks. Therefore, the total number of inter-block links is $\frac{n^2(n^2-1)}{2}$. □

Theorem 3 *There are in total* $\frac{n^2(5n+1)(n-1)}{2}$ *links in an* $n \times n$ *OTIS-MOT network.*

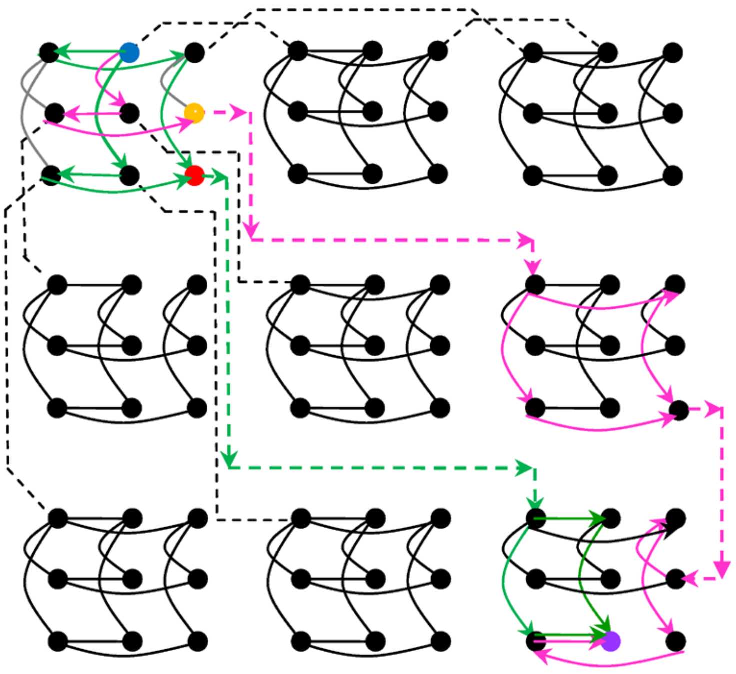


Fig. 3 Multiple paths from the source P(1, 1, 1, 2) (*colored blue*) to destination P(3, 3, 3, 2) (*colored violet*). Paths through a single OTIS link are shown by green color and the paths via an intermediate block (through two OTIS links) are shown by *pink color*

Proof Follows from Lemma 3 and Lemma 4. □

4.4 Multiple paths

Multiple paths between two arbitrary nodes are important to provide fault tolerance of an interconnection network. To count the multiple paths for the OTIS-MOT, we proceed as follows (however, we restrict counting these paths going through at most two OTIS links). Let us consider the source node P(1, 1, 1, 2) and the destination node P(3, 3, 3, 2) as depicted by blue and violet colors in Fig. 3. Let P(1, 1, 3, 3) and P(1, 1, 2, 3) be the two transit nodes as shown by red and yellow colors in this figure. Given a source node, we can reach to any transit node within the same block by traversing two possible shortest paths. As an example, two such shortest paths from the source node P(1, 1, 1, 2) to the transit node P(1, 1, 3, 3) are shown by green colors in Fig. 3. From the transit node P(1, 1, 3, 3) we can reach to the destination block through a single OTIS link, whereas from P(1, 1, 2, 3) the destination block is reachable via some other block going through two OTIS links as shown by pink color.

Now, within a block, there exist two shortest paths between any two nodes. Therefore, there exist four possible paths between any two source and the destination nodes going through a single OTIS link. In the other case, there exist six possible paths going through two OTIS links via other block. As every block contains $n^2 - 1$ transit nodes among which one is directly reachable to the destination block and the others can reach using two OTIS links, there are $6(n^2 - 2) + 4$ possible paths between any arbitrary pair of source and destination nodes. This leads the following theorem.

Theorem 4 *There exist* $6(n^2 - 2) + 4$ *possible paths between any two source and destination nodes going through at most two OTIS links.*

4.5 Bisection width

The size of the data set divided by bisection width puts the lower bound on the complexity of a parallel algorithm. Therefore, large bisection width is a desirable property of an interconnection network. This is especially significant for those computations which require large amounts of data movement. To compute the bisection width of the OTIS-MOT, we observe that there are $n^2 - 1$ inter-block links fanning out from each block of the network. Half of such links are actually connected to half of the total n^2 blocks of the OTIS-MOT. Therefore, we require removing $\frac{n^4}{4}$ optical links to bisect the whole network, assuming n is even. In other words, the bisection width of the $n \times n$ OTIS MOT is $O(n^4)$. This high bisection width along with the low diameter can make the OTIS-MOT very attractive to manipulate a massive volume of data for diameter-based algorithms.

4.6 Modularity

Modularity is an important property of an interconnection network that can be exploited by hardware as well as software implementation. For example, the decomposition of OTIS-MOT into identical modules (MOTs) can be useful for its VLSI layout. Moreover, modularity supports concurrency of identical local computations using MOT interconnectivity to integrate them for obtaining the solution of a given problem. For the decomposition, we require to remove all the inter-block links from all the modules (groups). As an $n \times n$ OTIS-MOT network has a total of $\frac{n^2(n^2-1)}{2}$ inter-block links (refer to Lemma 4), removal of all these links decomposes the OTIS-MOT into n^2 copies of MOT of size $n \times n$.

5 Communication algorithms

In this section, we present some communication algorithms, namely group-row broadcast, group-column broadcast, one-to-all, multicast and the bit-reverse permutation. These algorithms can be useful to solve various problems including matrix multiplication, prefix computation, polynomial root finding, Lagrange interpolation, enumeration sort and DFT computation discussed in the next section. We show that OTIS-MOT is very efficient in performing such operations. To analyze our proposed algorithm, we count the number of data movements on electronic channels as electronic moves and that on optical channels as OTIS moves separately.

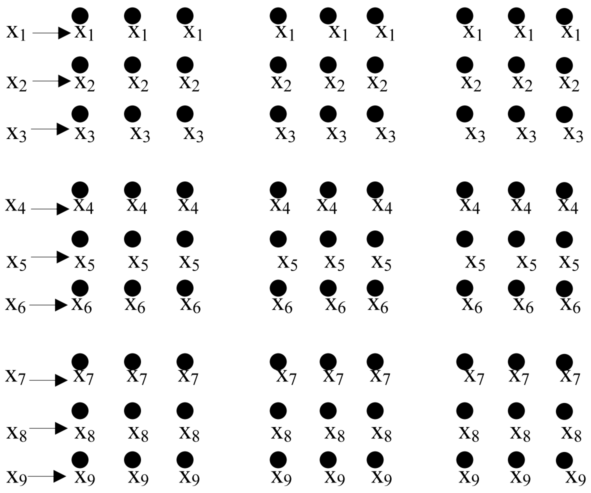


Fig. 4 Contents of A-Registers after row-group broadcast

5.1 Row-group broadcast (X, A)

This operation stores the data elements x_i, $1 \le i \le n^2$, in the ith row of the two-dimensional layout of the OTIS-MOT as shown in Fig. 4 in which data input is indicated by arrows. The steps are as follows.

```
Step 1. $\forall i, j, 1 \le i, j \le n$, do in parallel
        $A(i, 1, j, 1) \leftarrow x_{(i-1)n+j}$    /* Data input */
Step 2. $\forall i, j, 1 \le i, j \le n$, broadcast the contents of $A(i, 1, j, 1)$ (stored in Step 1)
        row-wise using row-tree link. $x_i$'s are now stored in all the processors of the
        first column blocks of the OTIS-MOT.
Step 3. Perform OTIS move on the contents stored in Step 2. $x_i$'s are now stored in
        all the processors of the first column of each block of the OTIS-MOT.
Step 4. $\forall i, j, 1 \le i, j \le n$, broadcast the contents of the A register (stored in Step 3)
        row-wise within each block.
Step 5. Perform OTIS move on the data received in Step 4. The data elements are
        now stored in the pattern shown in Fig. 4.
Step 6. Stop.
```

Time complexity: Each of the Steps 2 and 4 requires $\log n$ electronic moves. Steps 3 and 5, each requires one OTIS move. Hence the above row-group broadcast takes $2 \log n$ electronic moves $+$ 2 OTIS moves.

5.2 Column-group broadcast (X, B)

This operation stores x_j, $1 \le j \le n^2$ in the B registers of the jth column of the two-dimensional layout of the OTIS-MOT as shown in Fig. 5 and can be similarly developed as follows.

```
Step 1. $\forall i, j, 1 \le i, j \le n$, do in parallel
        $B(1, i, 1, j) \leftarrow x_{(i-1)n+j}$    /* Data input */
```

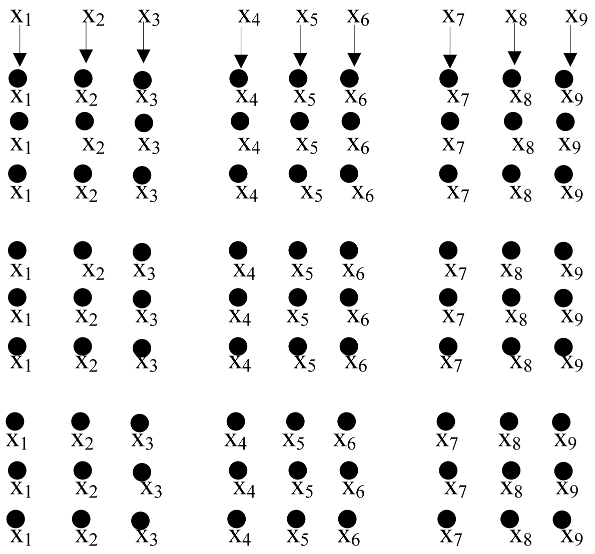


Fig. 5 Contents of A-Registers after column-group broadcast

Step 2. $\forall i, j$, $1 \le i, j \le n$, broadcast the contents of $\mathrm{B}(1, i, 1, j)$ (stored in Step 1) column-wise using column-tree link.
Step 3. Perform OTIS move on the contents stored in Step 2.
Step 4. $\forall i, j$, $1 \le i, j \le n$, broadcast the contents of the B register (stored in Step 3) column-wise within each block.
Step 5. Perform OTIS move on the data received in Step 4.
Step 6. Stop.

Time complexity: The above column-group broadcast also requires $2 \log n$ electronic moves and $+\,2$ OTIS moves as Steps 2 and 4 take $2 \log n$ electronic moves and Steps 3 and 5 take 2 OTIS moves.

5.3 Bit-reverse permutation

Let $\pounds[0, 1, \dots, m-1]$ be a linear array where $m = 2^k$ for some integer $k > 0$. If we represent each index i of this array by k-bit binary form as $i = b_{k-1}b_{k-2}\cdots b_2b_1b_0$, where $i = \sum_{j=0}^{k-1} b_j 2^j$, then the bit reverse is defined by $\mathrm{BR}(b_{k-1}b_{k-2}\cdots b_2b_1b_0) = b_0b_1b_2\cdots b_{k-2}b_{k-1}$. In other words, $\mathrm{BR}(i) = \sum_{j=0}^{k-1} b_{k-j-1}2^j$. As an example, the bit reverse of $\mathrm{BR}(19) = (11001)_2 = 25$ since 5-bit binary form of decimal 19 is $(10011)_2$. Then the bit-reversal permutation can be defined as an operation which swaps data elements whose indices in binary representations are bit reverse of each other [6].

We assume that n^4 data points $d_0, d_1, \dots, d_{n^4-1}$ are initially stored in the OTIS-MOT such that processor $\mathrm{P}(i, j, k, l)$ contains the data $d_{n^3i+n^2j+nk+l}$.

Algorithm Bit_Reversal:

Step 1. Perform local bit-reversal on each row-tree.

Step 2. Perform local data exchange using transpose operation on each block.
Step 3. Perform local bit-reversal on each row-tree.
Step 4. Perform one OTIS move.
Step 5. Perform local bit-reversal on each row-tree.
Step 6. Perform local data exchange by transpose operation on each block.
Step 7. Perform local bit-reversal on each row-tree.
Step 8. Stop.

Theorem 5 *The Algorithm Bit_Reversal correctly permutes the n^4 data points in bit-reversed order.*

Proof Let the indices i, j, k, l of the processor $\mathrm{P}(i, j, k, l)$ be represented in bit forms as $i_{p-1}i_{p-2}\dots i_1i_0$, $j_{p-1}j_{p-2}\dots j_1j_0$, $k_{p-1}k_{p-2}\dots k_1k_0$ and $l_{p-1}l_{p-2}\dots l_1l_0$ respectively, where $p = \log n$ and n is a power of 2. Then the following data movements are taking place on the successive steps.

$$\mathrm{P}(i_{p-1}i_{p-2}\dots i_1i_0, j_{p-1}j_{p-2}\dots j_1j_0, k_{p-1}k_{p-2}\dots k_1k_0, l_{p-1}l_{p-2}\dots l_1l_0) \xrightarrow{\text{Step 1}}$$

$$\mathrm{P}(i_{p-1}i_{p-2}\dots i_1i_0, j_{p-1}j_{p-2}\dots j_1j_0, k_{p-1}k_{p-2}\dots k_1k_0, l_0l_1\dots l_{p-2}l_{p-1}) \xrightarrow{\text{Step 2}}$$

$$\mathrm{P}(i_{p-1}i_{p-2}\dots i_1i_0, j_{p-1}j_{p-2}\dots j_1j_0, l_0l_1\dots l_{p-2}l_{p-1}, k_{p-1}k_{p-2}\dots k_1k_0) \xrightarrow{\text{Step 3}}$$

$$\mathrm{P}(i_{p-1}i_{p-2}\dots i_1i_0, j_{p-1}j_{p-2}\dots j_1j_0, l_0l_1\dots l_{p-2}l_{p-1}, k_0k_1\dots k_{p-2}k_{p-1}) \xrightarrow{\text{Step 4}}$$

$$\mathrm{P}(l_0l_1\dots l_{p-2}l_{p-1}, k_0k_1\dots k_{p-2}k_{p-1}, i_{p-1}i_{p-2}\dots i_1i_0, j_{p-1}j_{p-2}\dots j_1j_0) \xrightarrow{\text{Step 5}}$$

$$\mathrm{P}(l_0l_1\dots l_{p-2}l_{p-1}, k_0k_1\dots k_{p-2}k_{p-1}, i_{p-1}i_{p-2}\dots i_1i_0, j_0j_1\dots j_{p-2}j_{p-1}) \xrightarrow{\text{Step 6}}$$

$$\mathrm{P}(l_0l_1\dots l_{p-2}l_{p-1}, k_0k_1\dots k_{p-2}k_{p-1}, j_0j_1\dots j_{p-2}j_{p-1}, i_{p-1}i_{p-2}\dots i_1i_0) \xrightarrow{\text{Step 7}}$$

$$\mathrm{P}(l_0l_1\dots l_{p-2}l_{p-1}, k_0k_1\dots k_{p-2}k_{p-1}, j_0j_1\dots j_{p-2}j_{p-1}, i_0i_1\dots i_{p-2}i_{p-1}).$$

Hence the proof. □

Time complexity: Each of the Steps 1, 3, 5 and 7 requires $O(n)$ electronic moves; Steps 2 and 6 each takes $O(\log n)$ electronic moves using $O(n)$ buffer and Step 4 takes a single OTIS move. Therefore the above algorithm can be implemented in $O(n)$ electronic moves + one OTIS move.

5.4 One-to-all broadcast

This operation can be developed similarly as described in [41]. The steps are as follows.

Let $\mathrm{P}(i, j, k, l)$ hold the message to broadcast.

Step 1. Perform local broadcast within the block $\mathrm{G}(i, j)$ using row-tree and column-tree links.

Step 2. Perform one OTIS move so as to send a copy of the message to the processor P(i, j, 1, 1) of each block.
Step 3. Perform local broadcast of the copy of the message on each block.
Step 4. Stop.

Time Complexity: Both the Steps 1 and 3 take $2\log n$ electronic moves and Step 2 takes one OTIS move. Hence one-to-all broadcast requires $4\log n$ electronic moves + one OTIS move.

5.5 Multicast

We consider here the multicast in which each processor holds one message and sends it to a group of other processors. At the end of the procedure, each processor receives n^2 elements. The multicast can be implemented by repeated invocation of the window broadcast [41] as follows. Each call of the window broadcast makes a copy of a specific block (window) to all other blocks of the OTIS-MOT. This is accomplished by (1) one OTIS move on the contents of the block, (2) one local broadcast on each block of the data elements received in Step 1, and (3) another OTIS move on each processor. Now a window broadcast takes $2\log n$ electronic moves + 2 OTIS moves and there are n^2 blocks in an $n \times n$ OTIS-MOT. Therefore, multicast can be performed using window broadcast of n^2 blocks one at a time. This requires a total of $2n^2 \log n$ electronic moves + $2n^2$ OTIS moves.

6 Application algorithms

6.1 Summation

The summation is the representative of many other basic operations such as average, minimum, maximum, logical AND, logical OR, which are very useful in many applications. We assume that all the processors of the OTIS-MOT are initialized with n^4 data elements, a single element per processor. They can be summed up as follows. Perform the local summation on each group in parallel using row-tree followed by column-tree communications and store the partial results in the processors P(i, j, 1, 1), $1 \le i, j \le n$. This takes $2\log n$ electronic moves. Next, bring all the partial sums to the first block G(1, 1) by a single OTIS move and then perform another local summation on the first block in $2\log n$ electronic moves to generate the final sum at the processor P(1, 1, 1, 1). Therefore, summing n^4 elements is done in $4\log n$ electronic moves + one OTIS move.

Remark 1 The above procedure can be easily extended for summing more than n^4 data elements as follows. For the simplicity and without any loss of generality, we assume that kn^4 data elements are to be added where k is an integer greater than 1. Let us first partition the whole data set into k subsets, each having n^4 elements. Given the first data set, each group G(i, j) forms its local sum and stores it in the register A(i, j, 1, 1) for $1 \le i, j \le n$. The rest of the data sets are then successively fed. Each time a data set is fed to OTIS-MOT, it forms the local sum on each group and updates the contents of $A(i, j, 1, 1)$, $1 \le i, j \le n$, by adding the current sum. This is continued until the last data set is fed. Next, perform the OTIS move on $A(i, j, 1, 1)$ to bring them in the first block and add them locally to emerge the final sum from $A(1, 1, 1, 1)$. However, this requires $O(k \log n)$ electronic moves + one OTIS move.

6.2 Polynomial root finding

Durand–Kerner scheme [11] is an efficient method for finding polynomial roots, which has been studied for parallelization due to the following advantages. The method has inherent parallelism that can be suitably exploited by an SIMD machine. Moreover, it has local convergency with quadratic rate. Various parallel algorithms have been reported for Durnad–Kerner scheme, which can be found in [7, 13, 14, 17].

Let $P_N(x) = a_0x^N + a_1x^{N-1} + a_2x^{N-2} + \cdots + a_{N-1}x + a_N$ be an N-degree polynomial, where the coefficients a_i, $1 \le i \le N$, are assumed real. Then the iterative scheme for Durand–Kerner method [13] is as follows:

$$x_i^{(k+1)} = x_i^k - \frac{P_N(x_i^k)}{\prod_{\substack{j=1\\ j\neq i}}^{N} (x_i^k - x_j^k)}, \quad i = 1, 2, \ldots, N \tag{6.1}$$

where x_i^k denotes the kth approximation of the root. For the sake of simplicity, we denote $P_N(x_i^k)$ by P_i^k. Note that $\prod_{\substack{j=1\\ j\neq i}}^{N} (x_i^k - x_j^k)$, $1 \le i \le N$, is the principal computation of the Durand–Kerner method, which is the main target of parallelizing it. The idea is as follows. We first distribute initial data values row-wise and column-wise in the whole OTIS-MOT network following the row-group broadcast and the column-group broadcast discussed in Sect. 4. Each processor now subtracts these data values to create the factor of the form $(x_i - x_j)$. These factors are then used to form the local product row-wise in each group. The partial results (local products) are then brought by OTIS move to go another round of local products, which in turn forms the final computation. We now describe the algorithm stepwise as follows assuming $N = n^2$.

Parallel Durand–Kerner Algorithm:

```
Step 1. Call row-group broadcast (X, A).
        The contents of the A registers after this step are shown in Fig. 3.
Step 2. Call column-group broadcast (X, B).
        The contents of the B registers after this step are shown in Fig. 5.
Step 3. $\forall i, j$, $1 \le i, j \le n$, do in parallel
        If $A(i, j, *, *) \ne B(i, j, *, *)$ then
            $A(i, j, *, *) \leftarrow A(i, j, *, *) - B(i, j, *, *)$
        Else $A(i, j, *, *) \leftarrow 1$
Step 4. Form the local product with the contents of the A registers along the
        same row of each group using row-tree links and store it in $A(i, j, k, 1)$,
        $1 \le i, j, k \le n$.
Step 5. Perform an OTIS move on the contents of the A registers stored in Step 4.
Step 6. Form the local product with the contents of the A registers along the same
        row of each group G(∗, 1) using row-tree links and store it in A($i, 1, k, 1$),
        $1 \le i, k \le n$.
Step 7. Perform an OTIS move on the contents of the A registers stored in Step 6.
Step 8. $\forall i, k$, $1 \le i, k \le n$, do in parallel
        Input $y_{n(k-1)+i}$ to D($i, 1, k, 1$)
        Input $x_{n(k-1)+i}$ to C($i, 1, k, 1$)
Step 9. $\forall i, k$, $1 \le i, k \le n$, do in parallel
```

$$\mathrm{A}(i,1,k,l) \leftarrow \mathrm{C}(i,1,k,1) - \frac{\mathrm{D}(i,1,k,1)}{\mathrm{A}(i,1,k,1)}$$

```
Step 10. Stop.
```

Time Complexity: Each of the Steps 1 and 2 requires $2\log n$ electronic moves + two OTIS moves. Each of the Steps 4 and 6 takes $\log n$ electronic moves. Steps 5 and 7 require one OTIS move for each and the rest of the steps require constant time. Therefore, the above algorithm requires $6\log n$ electronic moves + 6 OTIS moves in total.

Remark 2 Given a set of tabulated values $y_1, y_2, \dots, y_N$ of a function $y = f(x)$ at some discrete points $x_1, x_2, \dots, x_N$, the N-point Lagrange formula for polynomial interpolation is as follows [15].

$$f(x) = \pi(x) \sum_{i=1}^{N} \frac{y_i}{(x - x_i)\pi'(x_i)}$$

where $\pi(x) = \prod_{i=1}^{N}(x - x_i)$, $\pi'(x_i) = \prod_{j=1, j\neq i}^{N}(x_i - x_j)$, $i = 1, 2, \dots, N$.

We note that the computation for Lagrange interpolation is similar to that of the Durand–Kerner method. Therefore the Lagrange interpolation can be similarly implemented on OTIS-MOT also in $O(\log n)$ time.

6.3 Matrix multiplication

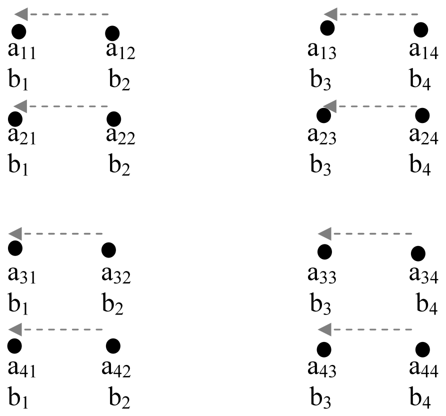
Let us first consider the implementation of the following matrix-vector multiplication:

$$c_{i1} = \sum_{j=1}^{N} a_{ij} b_j, \quad 1 \le i \le N \tag{6.2}$$

Assume that the matrix $(A)_{N\times N}$ where $N = n^2$ is already stored in the A registers in the row major order over the whole OTIS-MOT network. The vector $(b)_{N\times 1}$ can be stored in the B registers using column-group broadcast (b, B) as described in Sect. 4. The initialization of the matrix as well as the vector is shown in Fig. 6 for $N = 4$.

Now, each processor forms the product with the contents of their corresponding A and B registers in parallel. The products are then summed up to generate the final resultant vector $(C)_{N\times 1}$ by the following steps.

Fig. 6 Data initialization for matrix-vector multiplication



(1) Sum up the products row-wise (toward the first processor of each block as shown by an arrow in Fig. 6) for each group in $\log n$ electronic moves.
(2) Perform one OTIS move to bring all the partial sums to the appropriate group corresponding to c_{i1}.
(3) Again, sum up the partial results row-wise like in the Step 1 to generate the final c_{i1}s.
(4) Perform another OTIS move to store all the c_{i1}s in their appropriate processor. Therefore, the whole computation can be done in $2\log n$ electronic moves + 2 OTIS moves.

The above matrix-vector multiplication can be repeatedly invoked to implement the matrix-matrix multiplication $(C)_{N\times N} = (A)_{N\times N} \times (B)_{N\times N}$ as follows. Like in the matrix-vector multiplication, we also assume here that the elements of A matrix are already stored in the A registers in the row major order. We now successively feed the columns of the B matrix using column-group broadcast and invoke the matrix-vector multiplication. Given the jth column, it produces c_{ij} for $1 \le i \le n$ successively. As this is the same task to be repeated n times, it can be completed in $O(\log n) + n - 1$ time in a pipelined fashion with a period of $O(\log n)$.

6.4 Forecasting

Forecasting means prediction of a future event from the knowledge of a set of past events. In the time series models, given an observed time series, say $d_1, d_2, \ldots, d_m$, where d_i represents the data value at the ith time period, $1 \le i \le m$, the problem of forecasting is to estimate $d_{m+\tau}$, where τ is a small positive integer which is usually 1. Among various time series models, the weighted moving average is a widely accepted technique for small-term forecasting in which the plot of the data exhibits a cyclical pattern around a constant trend. In this method, a set of positive weight vector, say $w_1, w_2, \ldots, w_n$, is given for the n most recent observations $d_t, d_{t-1}, \ldots, d_{t-n+1}$. Then the weighted moving average WM(t) is calculated by the following formula [43] and used as the forecast value at $t + \tau$ time:

$$\mathrm{WM}(t) = \frac{w_n d_t + w_{n-1} d_{t-1} + \cdots + w_1 d_{t-n+1}}{w_n + w_{n-1} + \cdots + w_1} \tag{6.3}$$

This means that if $\hat{d}_{t+\tau}$ denotes the forecast value at $t+\tau$, then $\mathrm{WM}(t)=\hat{d}_{t+\tau}$. Note that given n_i weights, there can be $m-n_i+1$ weighted moving averages by sliding the window of size n_i over a set of m data values. Each time a window size is chosen, the minimum square error (MSE) is calculated using the following formula:

$$\mathrm{MSE}=\sum_{n_i+\tau}^{m}\frac{[d_t-\hat{d}_t]^2}{(m-n_i-\tau+1)} \tag{6.4}$$

The value of n_i which produces the least value of MSE is chosen for the forecast value $\hat{d}_{t+\tau}$. Thus if one assumes $\tau=1$, $m-n+1$ weighted moving averages are calculated for a single iteration as follows:

$$\begin{aligned}
\mathrm{WM}(n)&=\frac{w_1d_1+w_2d_2+\cdots+w_nd_n}{w_1+w_2+\cdots+w_n}\\
\mathrm{WM}(n+1)&=\frac{w_1d_2+w_2d_3+\cdots+w_nd_{n+1}}{w_1+w_2+\cdots+w_n}\\
\mathrm{WM}(n+2)&=\frac{w_1d_3+w_2d_4+\cdots+w_nd_{n+2}}{w_1+w_2+\cdots+w_n}\\
&\vdots\\
\mathrm{WM}(m)&=\frac{w_1d_{m-n+1}+w_2d_{m-n+2}+\cdots+w_nd_m}{w_1+w_2+\cdots+w_n}
\end{aligned}$$

It is shown in [20] that the above calculation can be represented by

$$\mathrm{WM}(n+i)=\frac{\gamma(n+i)}{\lambda(n+i)}\quad\text{for } 0\le i\le m-n \tag{6.5}$$

where the denominator $\lambda(n+i)=w_1+w_2+\cdots+w_m$ remains same for all the values of i and the numerator $\gamma(n+i)$can be expressed by the following matrix-vector multiplication:

$$\begin{pmatrix}\gamma(n)\\ \gamma(n+1)\\ \gamma(n+2)\\ \vdots\\ \gamma(m)\end{pmatrix}=\begin{pmatrix}d_1 & d_2 & \cdots & d_n\\ d_2 & d_3 & \cdots & d_{n+1}\\ d_3 & d_4 & \cdots & d_{n+2}\\ \vdots & \vdots & \vdots & \vdots\\ d_{m-n+1} & d_{m-n+2} & \cdots & d_m\end{pmatrix}\begin{pmatrix}w_1\\ w_2\\ w_3\\ \vdots\\ w_n\end{pmatrix} \tag{6.6}$$

Therefore the parallel implementation of the above $m-n+1$ weighted moving averages can be accomplished in $O(\log n)$ time similarly to the matrix-vector multiplication (described in Sect. 6.3), together with the parallel computation of $w_1+w_2+\cdots+w_n$.

6.5 DFT computation

It is shown in [2] that the DFT computation is another form of matrix-vector multiplication and thus can be similarly implemented in $2\log n$ electronic moves + 2 OTIS moves.

Table 1 Comparison of the performances of MMT, OMULT and OTIS-MOT interconnection networks

Parameter	MMT		OMULT			OTIS-MOT (proposed)		
	Data size	Electronic	Data size	Electronic	Optical	Data size	Electronic	Optical
Diameter	n^4 (nodes)	$8\log n + 2$	$2n^3 - n^2$	$6\log n$	2	n^4	$8\log n$	1
Bisection width	n^4	$2n(n-1)$		–			$n^4/4$	
Row/column-group broadcast	n^2	$2\log n + 3$	n	$2\log n$	3	n^2	$2\log n$	2
One-to-all broadcast	1	$7\log n + 2$	1	$6\log n$	2	1	$4\log n$	1
Summation	n^4	$4\log n + 11$	n^3	$6\log n$	2	n^4	$4\log n$	1
Matrix-vector multiplication	$n^2 \times n^2$†	$4\log n + 6$		–		$n^2 \times n^2$	$2\log n$	2
Matrix-matrix multiplication	$n^2 \times n^2$	$O(\log n) + n - 1$	$n \times n$	$6\log n$	8	$n^2 \times n^2$	$O(\log n) + n - 1$	
Prefix computation		–	n^2	$6\log n$	5	n^4	$13\log n$	2
Enumeration sorting	n^2	$6\log n + 12$	n	$5\log n$	3	n^2	$8\log n$	3

†Size of the matrix

6.6 Prefix computation

The prefix computation can be used as a tool to solve many problems like job scheduling, knapsack, loop optimization, evaluation of polynomials, solving system of linear equations, polynomial interpolation and many others. Given a set of N data values, $d_1, d_2, \dots, d_N$, and an associative binary operation o, the problem of prefix is to compute $S_i = d_1 o d_2 o d_3 o \dots o d_i$, $1 \le i \le N$. We have shown that the prefix computation on OTIS-MOT for $N = n^4$ data elements requires $13 \log n + O(1)$ electronic moves $+$ 2 OTIS moves using n^4 processors. For details of the algorithm, the reader is referred to [26].

6.7 Sorting

We have shown in [24] that the enumeration sort for $N = n^2$ data elements can be implemented in $4.5 \log N$ electronic moves $+$ 5 OTIS moves. This is further improved to be $4 \log N$ electronic moves $+$ 3 OTIS moves [25].

7 Conclusion

In this paper, we have presented OTIS-MOT as an efficient interconnection network. We have derived various topological properties such as fault diameter, bisection width, multiple paths and modularity. We have shown that various communication algorithms such as row/column-group broadcast and one-to-all broadcast run in $O(\log n)$ time. The multicast and the bit-reverse permutation have been proposed in $O(n^2 \log n)$ and $O(n)$ time, respectively. Parallel algorithms for finding polynomial zeros, forecasting, matrix-vector multiplication and DFT computation have been presented in $O(\log n)$ time. Sorting and prefix computation have been shown to run in $O(\log n)$ time, too. The comparison of the performances of OTIS-MOT with other similar tree-based two-tier architectures, namely MMT and OMULT, has been shown in Table 1. It is obvious to note that OTIS-MOT is far better than MMT and OMULT with respect to topological properties, communication algorithms and parallel algorithms for various problems. Obviously, it is also better than other two-tier architectures, namely OTIS-Mesh [41] and Multi-mesh [8], each having $O(n^{1/2})$ diameter.

References

1. Agostino SD (2006) Lossless image compression by block matching on a mesh of trees. In: Proc of data compression conference, p 443
2. Akl SG (1989) The design and analysis of parallel algorithms. Prentice Hall, Englewood Cliffs
3. Balkan AO, Qu G, Vishkin U (2009) Mesh-of-trees and alternative interconnection networks for single-chip parallelism. IEEE Trans Very Large Scale Integr (VLSI) Syst 17:141–1432
4. Balkan AO, Qu G, Vishkin U (2006) A mesh-of-trees interconnection network for single-chip parallel processing. In: Proc of intl conf on application-specific systems, architectures and processors, pp 73–80
5. Balkan AO, Michael NH, Qu G, Vishkin U (2007) Layout-accurate design and implementation of a high-throughput interconnection network for single-chip parallel processing. In: Proc of the 15th IEEE symposium on high-performance interconnects, pp 21–28

6. Cormen TH, Leiserson CE, Rivest RL (1999) Introduction to algorithms. Prentice Hall of India, New Delhi
7. Cosnard M, Fraigniaud P (1990) Finding the roots of a polynomial on an MIMD multicomputer. Parallel Comput 15:75–85
8. Das D, De M, Sinha BP (1999) A new network topology with multiple meshes. IEEE Trans Comput 68:536–551
9. Day K et al (2002) Topological properties of OTIS-networks. IEEE Trans Parallel Distrib Syst 13:359–366
10. Day K (2004) Optical transpose k-ary n-cube networks. J Syst Archit 50:697–705
11. Durand E (1960) Solutions numeriques des equations algebriques. Mason, Paris. Tome I
12. Efe K (1996) Mesh-connected trees: a bridge between grids and meshes of trees. IEEE Trans Parallel Distrib Syst 7:1281–1291
13. Freeman TL (1989) Calculating polynomial zeros on local memory parallel computer. Parallel Comput 12:351–358
14. Freeman TL, Bane MK (1991) A synchronous polynomial zero-finding algorithms. Parallel Comput 17:673–681
15. Hildebrand FB (1956) Introduction to numerical analysis. McGraw-Hill, New Work
16. Hussain HM, Prasanna VK (1994) Efficient parallel computation on the reduced mesh of trees organization. J Parallel Distrib Comput 20:121–135
17. Jana PK (2006) Polynomial interpolation and root finding on OTIS-mesh. Parallel Comput 32:301–312
18. Jana PK (2004) Multi-mesh of trees with its parallel algorithms. J Syst Archit 50:193–206
19. Jana PK, Sinha BP (2006) An improved parallel prefix algorithm on OTIS-mesh. Parallel Process Lett 16:429–440
20. Jana PK, Sinha BP (1997) Fast parallel algorithms for forecasting. Comput Math Appl 34:39–49
21. Lee Y, Horng S, Kao T (1997) Parallel computation of the Euclidean distance transform on the mesh of trees and the hypercube computer. Comput Vis Image Underst 68:109–119
22. Leighton FT (1992) Introduction to parallel algorithms and architectures: array, trees and hypercubes. Morgan Kaufman, San Mateo
23. Lucas KT, Mallick DK, Jana PK (2008) Parallel algorithm for conflict graph on OTIS triangular array. In: Lecture notes in computer science, vol 4904. Springer, Heidelberg, pp 274–279
24. Lucas KT, Jana PK (2009) An efficient parallel sorting algorithm on OTIS mesh of trees. In: Proc of IEEE intl advance comp conf, 6–7 March, 2009, Patiala, India, pp 175–180
25. Lucas KT, Jana PK (2010) Sorting and routing on OTIS-Mesh of trees. Parallel Process Lett 20:145–154
26. Mallick DK, Jana PK (2008) Parallel prefix on mesh of trees and OTIS mesh of trees. In: Proc of the intl conf on parallel and distributed processing techniques and appl, Las Vegas, Nevada, USA, pp 359–364
27. Marrakchi Z, Mrabet H, Masson C, Mehrez H (2007) Efficient mesh of tree interconnect for FPGA architecture. In: Proc of the intl conf on field-programmable technology, pp 269–272
28. Marrakchi Z, Mrabet H, Masson C, Mehrez H (2007) Mesh of tree: unifying mesh and MFPGA for better device performances. In: Proc. of the 1st intl symposium on networks-on-chip, pp 243–252
29. Marsden GC, Marchand PJ, Harvey P, Esener SC (1993) Optical transpose interconnection system architectures. Opt Lett 18:1083–1085
30. Nath D, Maheshwari SN, Bhatt PCP (1983) Efficient VLSI networks for parallel processing based on orthogonal trees. IEEE Trans Comput C-32:569–581
31. Parhami B (2005) Swapped interconnection networks: topological, performance and robustness attributes. J Parallel Distrib Comput 65:1443–1452
32. Parhami B (2005) The Hamiltonicity of swapped networks built of Hamiltonian component networks. Inform Process Lett 19:441–445
33. Rajasekaran S, Sahni S (1998) Randomized routing, selection and sorting on the OTIS-mesh optoelectronic computer. IEEE Trans Parallel Distrib Syst 9:833–840
34. Sahni S, Wang CF (1997) BPC permutation on the OTIS-mesh optoelectronic computer. In: Proc of the 4th intl conf massively parallel processing using optical interconnections, pp 130–135
35. Sahni S (2001) Models and algorithms for optical and optoelectronic parallel computers. Int J Found Comput Sci 12:249–264
36. Salinger P, Tvrdik P (2002) Broadcasting in all-output-port mesh of trees with distance-insensitive switching. J Parallel Distrib Comput 62:1272–1294

37. Salinger P, Tvrdık P (2002) Optimal broadcasting and gossiping in one-port meshes of trees with distance-insensitive routing. Parallel Comput 28:627–647
38. Sinha BP, Bandyopadhyay S (2004) OMULT: an optical interconnection system for parallel computing. In: Proc of the intl conf on computers and devices for comm, Jan 1–3, 2004, Kolkata, India
39. Wang CF, Sahni S (2001) Matrix multiplication on the OTIS-mesh optoelectronic computer. IEEE Trans Comput 50:635–646
40. Wang CF, Sahni S (1998) Image processing on the OTIS-mesh optoelectronic computer. IEEE Trans Parallel Distrib Syst 11:97–109
41. Wang CF, Sahni S (1998) Basic operations on the OTIS-mesh optoelectronic computer. IEEE Trans Parallel Distrib Syst 12:1226–1236
42. Wang CF, Sahni S (1998) OTIS optoelectronic computers. In: Li K, Pan Y, Zhang SQ (eds) Parallel computation using optical interconnections. Kluwer Academic, Dordrecht
43. Wheelwright SC, Makridakis S (1980) Forecasting methods for management. Wiley, New York
44. Zane F, Marchand P, Paturi R, Esener S (2000) Scalable network architectures using the optical transpose interconnection system (OTIS). J Parallel Distrib Comput 60:521–538