

A Mesh-of-Trees Interconnection Network for Single-Chip Parallel Processing*

Aydin O. Balkan

Gang Qu

Uzi Vishkin

University of Maryland Institute for Advanced Computer Studies (UMIACS)
University of Maryland Department of Electrical and Computer Engineering
{balkanay, gangqu, vishkin}@umd.edu

Abstract

There is a recent surge of interest in single-chip parallel processors. In such machines, it is crucial to implement a high-throughput low-latency interconnection network to connect the on-chip components, especially the processing units and the memory units. In this paper, we propose a new mesh of trees (MoT) implementation of the interconnection network and evaluate it relative to metrics such as wire area, total switch delay and maximum throughput taking into account latency-throughput trade-offs. We show that on-chip interconnection networks can provide higher bandwidth between processors and shared first-level cache than previously considered possible, facilitating greater scalability of memory architectures that require that. MoT is also compared, both analytically and experimentally, to some other traditional network topologies, such as hypercube, butterfly, fat trees and butterfly fat trees. When we evaluate a 64-terminal MoT network at 65nm technology, concrete results show that MoT provides higher throughput and lower latency especially when the input traffic (or the on-chip parallelism) is high, at the cost of larger area. A recurring problem in networking and communication is that of achieving good sustained throughput in contrast to just high theoretical peak performance that does not materialize for typical work loads. Our quantitative results demonstrate a clear advantage of the proposed MoT network in the context of single-chip parallel processing.

1. Introduction

The advent of the Billion-transistor chip era coupled with a slow down in clock rate improvement brought

about a growing interest in parallel computing. Ongoing expansion in the demands of scientific and commercial computing workloads also contributes to this growth in interests. To date, the outreach of parallel computing has fallen short of expectations. This has primarily been attributed to programmability shortcomings of parallel computers. The Parallel Random Access Model (PRAM) is an easy model for parallel algorithmic thinking and for programming. Current multi-chip multiprocessor designs that aim to support the PRAM (such as Tera/Cray MTA [1] and SB-PRAM [2]), although interesting, are constrained by interchip interconnections. Latency and bandwidth problems have limited their success in supporting PRAM. With the continuing increase of silicon capacity, it becomes possible to build a single-chip parallel processor as is being demonstrated in the Explicit Multi-Threading (XMT) project [18, 23] that seeks to prototype a *PRAM-On-Chip* vision.

Parallel computing generally requires a larger number of memory accesses than serial computation per clock. A standard method for hiding access latencies is using multiple hardware threads, implying a steady and high demand for memory accesses. To facilitate concurrent accesses by many processors, memory is normally partitioned on parallel machines [12].

To handle the high level of parallelism needed for a PRAM on-a-chip, XMT uses a memory architecture where partitioning of data memory starts from the first level of the on-chip cache. A better designed interconnection network may create fewer on-chip queuing bottlenecks, when concurrent read and/or write requests are issued to the memory. This will significantly affect overall system performance.

Hypercube, butterfly and fat tree topologies and their variations have been popular for interconnection networks in parallel computing studies (e.g. [7, 11, 15, 21, 25]) as well as in machine architectures (e.g. [1, 9, 16]). In all these topologies, data packets between

*Partially supported by grant 0325393 from the National Science Foundation.

different sources and different destinations can interfere with one another. For background, note also that source to destination traffic is expected to be balanced. This is because that the use of universal hashing [9,17] will prevent hot spots. On the other hand, *Crossbar* networks provide one standard type of high-throughput interconnection networks, where packets do not interfere. They achieve this by scheduling the switches based on the global state of the network that may cause significant overheads.

In this paper, we study the interconnection network design problem for a certain memory architecture that is designed to achieve high single-chip parallelism. We first analyze the existing interconnection network models and conclude that they do not meet the latency and throughput requirements for such interconnection network. We then propose a new approach based on the concept of mesh of trees (MoT). We conduct simulation to evaluate the performance of both the existing approaches and the proposed MoT network.

The paper makes two contributions. First, on-chip implementations of several of the most popular interconnection network topologies are compared in terms of total wire area, total switch delay, maximum throughput, and trade-offs between throughput and latency. We are unaware of a similar comprehensive comparative study in the literature. Second, although MoT is a well-known concept for parallel algorithms and architectures, it has not been applied to build interconnection network. We describe the MoT network structure and demonstrate, through both theoretical analysis and simulation, that the proposed MoT network can achieve competitive throughput and low latency, especially when the input traffic (or the on-chip parallelism) is high. Assuming $1GHz$ clock rate at $65nm$ technology, a 64-terminal MoT network can provide a 4Tbps throughput, with a total (from source to destination) switch delay of 280ps.

Due to space restrictions, we refer the interested reader to [3] for an extended version of this paper, especially on the detailed analysis of other interconnection networks.

2. Background and Related Work

In this section we first briefly describe the existing interconnection network models, then summarize the underlying memory architecture for our network.

An n -dimensional **hypercube**, Q_n , interconnects $N = 2^n$ nodes by connecting each node to n other nodes. If we label the nodes from 0 to $N - 1$ in binary, a pair of nodes are connected directly if and only if their labels differ by one bit [7, 11, 14].

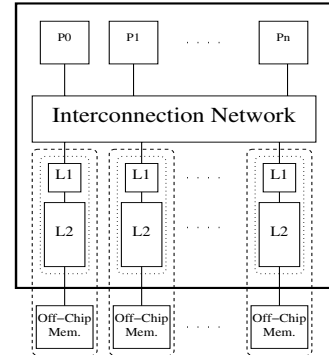


Figure 1. Partitioned global memory.

A binary **butterfly** network also connects $N = 2^n$ nodes in $\log_2 N$ levels. At level i , half of the signals in a group of 2^i nodes are shuffled with each other, creating a path from each source to each destination [7, 24].

Fat tree network provides multiple paths between each pair of nodes. A disadvantage of fat tree is its large switch size. Two structures were proposed to overcome this disadvantage. (i) The k -ary n -tree [21] connects $N = k^n$ PCs with a fat tree of n levels, where each internal node has k children and k parents. (ii) The *Butterfly fat tree* connects $N = 4^n$ nodes, where each internal node has 4 children and 2 parents [10, 19].

We assume that all of the above networks use *virtual channels (VCs)* in switch nodes to improve throughput. VCs act as buffers for incoming data packets that are stalled due to contention in later stages. A packet is stored in a virtual channel in the switch until an output port and physical channel toward its destination becomes available. More details on VCs and their use in interconnection networks can be found in [6, 7, 10, 21].

Using multiple memory modules (or banks) has been a common approach to increase memory bandwidth. In general, the global memory space is partitioned over the modules, and accesses to different modules are handled concurrently. In the XMT single-chip parallel architecture, the globally shared memory space is partitioned into multiple memory modules [18] (Figure 1). Each memory module consists of on-chip cache and off-chip memory portions. A universal hashing function is used to avoid pathological access patterns [1, 2, 9].

This architectural choice poses some significant design challenges. (i) The network is placed between the processors and the first level of data cache. In order to satisfy the steady and high demand of processors, the network needs to provide high throughput. (ii) The network needs to provide low on-chip communication latency. This will improve overall performance and simplify the design of processing units. (iii) The network needs to take as little chip area as possible.

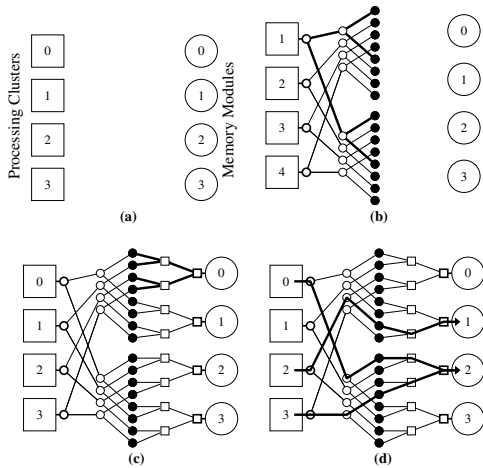


Figure 2. Mesh of Trees with 4 Clusters and 4 Memory Modules.

Hypercube, fat trees, and butterfly networks cannot provide the high throughput and low latency requirements for such memory structure designated for single-chip parallel processing. This is mainly due to their inability to avoid or reduce packets interference. In such networks, packets between different sources and different destinations can interfere with each other and reduce overall performance.

In this paper, we propose a new interconnection network implementation based on mesh of trees (MoT). The MoT structure guarantees that unless the memory access traffic is extremely unbalanced, packets between different sources and destinations will not interfere. We will demonstrate, both analytically and experimentally, that the proposed MoT network can provide high throughput with low latency.

3. Mesh of Trees Network

In earlier sections the memory architecture and its implementation challenges were overviewed. We now present the MoT network in the context of single-chip parallel processing.

3.1. Topology

The *mesh of trees* (MoT) concept is discussed in [8,13,14]. In an interconnection network based on their approach, processing units and memory modules will be placed at the leaves of the trees, and a communication path involves climbing up and down some part of the tree. This approach is not interference free and would create performance bottlenecks.

We implement the MoT network in a different way as

shown in Figure 2 with four processing clusters (PCs) and four memory modules (MMs). We treat the PCs as sources for packets and the MMs as destinations for convenience. Unlike the above conventional MoT approach, we put the PCs and MMs at the roots of the trees instead of the leaves. The network consists of two main structures, a set of fan-out trees and a set of fan-in trees¹. Figure 2(b) shows the binary fan-out trees, where each PC is a root and connects to two children (we call them *up child* and *down child*), each child will have two children of their own. The 16 leaf nodes also represent the leaf nodes in the binary fan-in trees that have MMs as their roots (Figure 2(c)).

There are two interesting properties of our MoT network. First, there is a unique path between each source and each destination. This simplifies the operation of the switching circuits and allows faster implementation which translates into improvement in throughput when pipelining a path. Second, packets between different sources and destinations will not interfere, unless the traffic is heavily unbalanced.

3.2. Routing

Routing is the process of finding a path for each packet from its source to its destination. Figure 2(d) gives the communication paths from PCs to MMs for three memory requests. Each memory request will travel from the PC (source) through a fan-out tree and then a fan-in tree before it reaches the MM (destination). There is no routing decision to be made in the fan-in trees as all packets move toward the root. In fan-out trees, routing decision is trivial from the binary representation of the destination. This simple routing scheme also ensures that the fan-out tree part of the network is non-interfering. Similarly, packets with different destinations will not interfere in the fan-in trees.

3.3. Flow Control

Flow control mechanisms manage the allocation of channels and buffers to packets. Figure 3 illustrates the switching primitives in our MoT network. Each node in the fan-out and fan-in trees of the network will be implemented as the fan-out or fan-in (arbitration) primitives as shown in Figure 3 (a) and (b). The pipeline primitive in Figure 3(c) is used to divide long wires into multiple short segments.

In general, packets do not compete for resources in the fan-out trees. They stall only when the fan-in trees

¹They are called row and column trees in [14]. Here we use the names of fan-out and fan-in trees to convey the data flow direction.

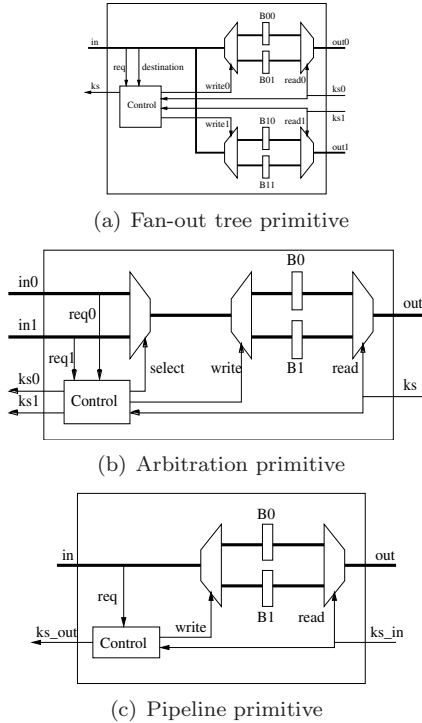


Figure 3. Switch primitives for MoT network. Data paths are marked with thick lines. Control paths are simplified. Signals: *req*: Request; *ks*: Kill-and-Switch; *write/read*: Write and Read pointers; *B*: Storage Buffer; *select*: Result of Arbitration; *destination*: Destination address.

stall and the stall propagates to the fan-out trees. This occurs rarely when traffic pattern is extremely unbalanced. Arbitration in fan-in trees is designed to be fair. If a request from one child loses the arbitration to the other child's request in one cycle, the request is guaranteed to win the arbitration in the next cycle.

The use of local flow control mechanisms reduces the communication overhead. However, in the absence of a global *stall* signal, a node has to get the stall information from its immediate successor. A toggled *kill-and-switch* (*ks*) signal allows the subsequent data packet to appear at the output port. In order to prevent data loss, our implementation uses a slightly modified version of a *relay station* [4]. Each output port has two buffers, and a small control circuit ensures that consecutive packets are stored in different buffers.

3.4. Floorplan

Figure 4 depicts our proposed floorplan for the *MoT networks*. We first explain the layout of the fan-out and fan-in trees. Both the fan-out and fan-in trees are

placed in pairs for better area utilization. Figure 4 (a) shows such a pair of 8-leaf fan-out trees for an MoT network with $N = 8$ clusters. The two root nodes of the two fan-out trees are connected to the source clusters by the thick lines. Plain empty circles are internal nodes and filled circles are leaf connections. Figure 4 (c) shows the same layout for a pair of 32-leaf fan-out trees. Figure 4 (d) shows a pair of 8-leaf fan-in tree. The filled circles are leaves. They are connected to internal nodes represented by the empty squares. Roots of the two fan-in trees are connected to the destination clusters through the connections with arrowhead. Figure 4 (e) gives the layout of one 32-leaf fan-in tree.

Figure 4 (b) shows how the fan-out and fan-in trees are placed between the eight clusters. Each pair of the fan-out trees is placed vertically and each pair of fan-in trees is laid out horizontally. At the top and the bottom, there is only one single fan-in tree each. The leaves of fan-out tree are connected to the leaves of fan-in trees. The path of a packet from cluster 6 to cluster 1 is highlighted.

4. Evaluation

We evaluate the proposed MoT network and compare it with other networks in the following four categories: Wire area, total switch delay, maximum throughput and throughput-latency relation. Table 1 describes the symbols we used in these evaluations.

We present the derivation of our results for the proposed MoT network. Due to space restrictions, we summarize the results for other networks, and refer the interested reader to [3] for their derivation.

Symbol	Description
A_w	Wire area
b	Number of bits per channel
d_w	Wire pitch (technology dep. parameter)
$FO4$	Technology independent <i>Fan-Out-4</i> delay
H_{chip}	Height of the chip
N	Number of terminals
v	Number of virtual channels
W_{chip}	Width of the chip

Table 1. List of symbols.

4.1. Area Evaluation

The network area will be determined by the wires, and the switches and other silicon resources can be laid underneath the wires. For simplicity we use two metal

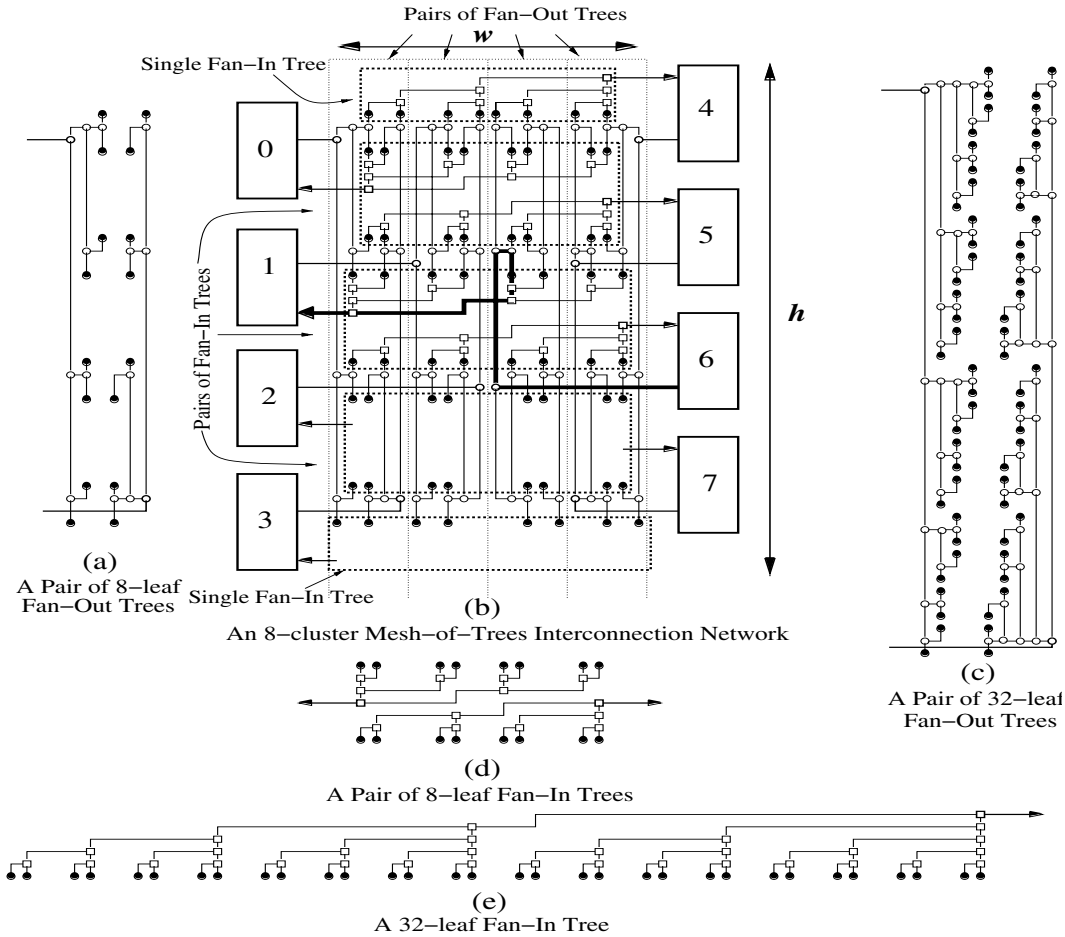


Figure 4. Detailed floorplan of the MoT interconnection network. (b): the 8-cluster (PC and co-located MM) MoT network floorplan; (a) and (d): details of a fan-out tree pair and a fan-in tree pair in (b); (c) and (e): layout of 32-leaf trees.

layers, one for horizontal and one for vertical wires. Future technology generations will allow around ten metal layers [22]. This would reduce the area cost of all networks. Figure 5(a) summarizes the area comparison of different networks with parameters for 65nm technology [22]. MoT network has more wires and less wire sharing to achieve high throughput and low latency. As expected, it requires more area, for example, 60% more than the most area efficient implementation of hypercube. The following elaborates how the area of MoT is derived.

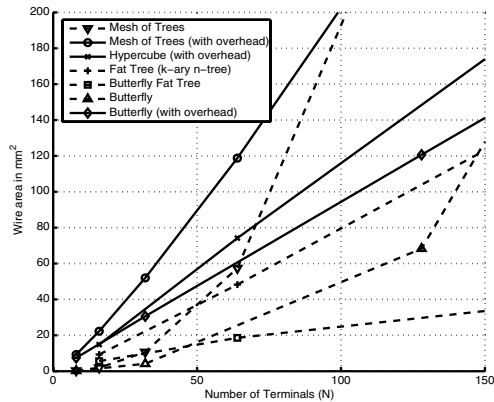
In the structure of Figure 4 (b), the width of wire area is $b \cdot d_w \cdot \frac{N}{2} \cdot (\log N + 2)$ and height is $b \cdot d_w \cdot (\frac{N}{2} \cdot (2 \log N + 1) + 1)$. Their product gives the wire area of MoT network. When we stretch the network along the vertical direction to the height of the chip in order to reach all the processing clusters, the area with such overheads becomes:

$$A_w = \frac{1}{2} \cdot b \cdot d_w \cdot H_{chip} \cdot N \cdot (\log N + 2)$$

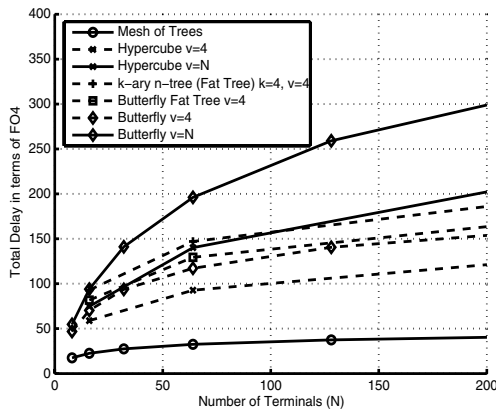
4.2. Total Switch Delay

The total switch delay is measured as the product of the *average number of hops* from source to destination and the switch delay of the switch circuit. Switch delay for MoT is derived from the synthesis results using Synopsys tools and others are computed following the formulas given in the literature [10,20]. This switch delay is measured in terms of *FO4*, a technology independent delay unit that represents the delay of an inverter driving four identical inverters. As Figure 5(b) shows, a 64-terminal configuration of the proposed MoT network has the smallest total switch delay, 2.8 to 6.0 times less than others. The following describes how we obtain the switch delay and average number of hops for MoT network.

We synthesize the switch primitives described in Figure 3 in Synopsys. The timing analysis shows that the circuit delays vary between 2.21 *FO4* and 2.43 *FO4*. For simplicity we assume a conservative



(a) Wire Area



(b) Total Switch Delay

Figure 5. (a) Wire Area Comparison at 65nm Technology. Dashed lines represent wire areas without overheads. $H_{chip} = W_{chip} = 20mm$, $b = 80$, $d_w = 290nm$ according to [22]. (b) Technology independent total switch delay comparison for various networks with different parameters.

constant delay of 2.5 FO4. From the MoT topology, it is easy to see that the number of hops is $2 \log N + 1$ for all source and destination pairs.

4.3. Maximum Throughput

In order to evaluate the maximum throughput provided by each network model, we assume the maximum packet generation rate of one packet per cycle (1.0 *ppc*) at each input port of the network². At this generation

²Note that in several other studies of interconnection networks, long data packets are divided in shorter units, called *flits*. Since our interconnection network is in a critical path between the processors and the cache memory, we do not cut a packet in shorter flits and assume that the entire packet of b bits is injected

rate, the network will saturate with packets, and the injection and delivery rates will come to balance at the maximum throughput. We assume uniform traffic pattern, which is expected for the memory architecture described in Section 2, due to the use of hashing mechanism [1, 2, 9, 17].

We obtain the results for hypercube, butterfly, and MoT networks from theoretical analysis and simulation. The results for fat tree networks are from [21] and [19]. Table 2 shows that the proposed MoT network can provide the highest maximum throughput, which is 76% and 28% higher than butterfly and hypercube with $v = 4$ virtual channels, and 3% and 16% higher than butterfly and hypercube with 64 virtual channels, respectively.

In MoT network, each PC injects 1.0 *ppc* into its fan-out tree (Figure 2 (b)). Each leaf receives $1/N$ *ppc* for the uniformly injected traffic. In the fan-in tree, the $1/N$ *ppc* traffic accumulates to 1.0 *ppc* at the root node. Therefore, MoT network is capable of delivering the maximum 1.0 *ppc* at each destination port.

In simulation, due to temporary imbalances in traffic caused by the stochastic nature of the generation process, 1.0 *ppc* is not expected. As we increase N , the number of terminals, there will be more pipeline stages between the roots of fan-out tree and fan-in tree. This will relieve some of this imbalance and provide higher throughput. In fact, we observe the maximum throughput of 0.951 *ppc* per port for $N = 16$ goes up to 0.977 *ppc* when $N = 64$.

4.4. Throughput and Latency

As traffic in the network increases, packets will experience longer latencies. We follow the guidelines in [7] to design simulations in order to evaluate the throughput and latency of various network models under different input traffic. We are particularly interested in the case when the input traffic, or the on-chip parallelism, is high. Hypercube and butterfly networks are simulated on the simulator provided by [7] with $N = 64$ terminals, and different number of virtual channels, namely a typical $v = 4$ setting and an aggressive $v = 64$ setting. Router switches have three cycle switch latency per *speculative virtual channel router* design of [20]. MoT network is simulated using an RTL SystemC simulator we implemented.

First, to validate our own simulator, we implement a single-cycle butterfly switch similar to the switch primitives shown in Figure 3. We then use our simulator to simulate the butterfly network with $N = 64$ and this switch. The results are reported in Figure 6 at once [1].

Configuration	Max Throughput	Configuration	Max Throughput
Hypercube $N = 16 v = 4$	0.777	Butterfly $N = 16 v = 4$	0.602
Hypercube $N = 64 v = 4$	0.763	Butterfly $N = 64 v = 4$	0.553
Hypercube $N = 16 v = 16$	0.787	Butterfly $N = 16 v = 16$	0.861
Hypercube $N = 64 v = 64$	0.843	Butterfly $N = 64 v = 64$	0.946
Fat Tree $N = 256 k = 4 n = 4 v = 2$	0.55	BFT $N = 64 v = 4$	0.28
Fat Tree $N = 256 k = 4 n = 4 v = 4$	0.72	BFT $N = 64 v = 8$	0.30
Mesh of Trees (theoretical)	1.0	Mesh of Trees $N = 16$	0.951
Mesh of Trees $N = 32$	0.963	Mesh of Trees $N = 64$	0.977

Table 2. Maximum throughput (in ppc per port) for different networks (BFT: Butterfly Fat Tree).

as the two curves marked *systemC BF*. The simulation results from [7] on the butterfly network with the $N = 64 v = 2$ configuration and single-cycle switch latency are also reported in Figure 6 as the curves marked *booksim BF*. One can see that these curves match reasonably well which indicates the accuracy of our simulator.

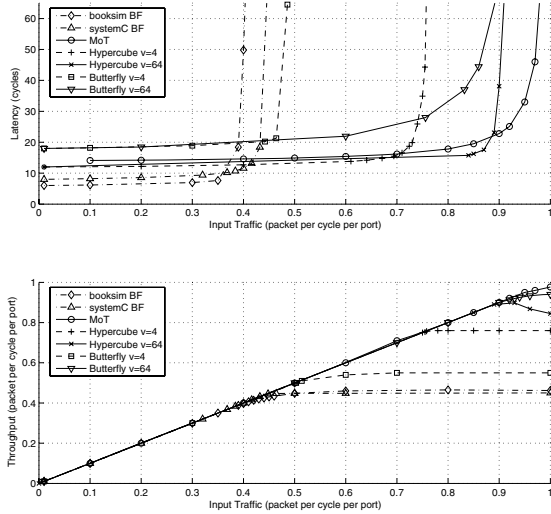


Figure 6. Throughput and latency of various networks for $N = 64$ terminals.

Simulation results are summarized in Figure 6. MoT network provides competitive throughput and latency and has a clear advantage over others when the input traffic is high. More importantly, MoT network has a more predicible latency when the input traffic varies. For example, when we increase the input traffic from 0.1 *ppc* per port to 0.9 *ppc*, the hypercube latency increases by a factor of 3.2, butterfly network latency increases by a factor of 3.9, while MoT latency increases only by a factor of 1.6.

5. Conclusion and Future Work

A memory hierarchy similar to Figure 1 was considered in [5]. However, [5] opined that while this strategy is possible for a multiprocessor-on-chip it applies to only a very small number of processors (2-8) since the interconnection network between processors and shared first-level cache will not be able to deliver the tremendous bandwidth needed to the multiprocessors accessing it simultaneously. The current paper shows that it can for a significantly larger number of processors.

We introduce a certain *Mesh of Trees* (MoT) interconnection network and compare it with traditional networks in terms of wire area, total switch delay, maximum throughput and latency-throughput relation. This network enables (i) support of easy parallel programming approaches, e.g., PRAM-like [18,23] that cannot be supported otherwise [5]; and (ii) feeding data to the functional units at a sufficient rate.

Our quantitative results demonstrate a clear advantage for the proposed MoT implementation. Particularly, at 65nm technology, a 64-terminal MoT network operating at 1GHz clock rate, provides up to 62G words (4T bits) per second throughput. Between the source and destination terminals, the total switch delay is approximately 280ps. Ongoing studies on wire delays will allow more accurate latency and throughput analysis.

The interconnection network of multi-chip Tera parallel computer has a bisection bandwidth of one 64-bit word per cycle per processor [1]. By providing the same per-cycle bandwidth and a matching actual throughput within the chip, MoT eliminates the clock cycle limitations of inter-chip communication. This results in much faster clock rates and significantly lower memory access latency.

It would be useful to better understand the cost-performance trade-offs of interconnection network designs for single-chip parallel processors with a large number of processing units. According to our results, the proposed MoT network outperforms all of the ex-

isting approaches in terms of latency and throughput, while this comes at a price of increased area. We plan to study opportunities for reducing the wire area, and the effects of wire delays on packet latency, as well as other cost factors such as the area cost of circuit resources and power requirements. The study and results presented in this paper lay a foundation for such future work.

Finally, the use of MoT network is not limited to parallel processors. As the number of functional units grow in system-on-chip applications, on-chip networks will be required to satisfy the communication needs. MoT network may provide an alternative solution, where high-throughput and low-latency communication is needed.

References

- [1] R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, and B. Smith. The tera computer system. In *Proc. Int. Conf. On Supercomputing*, pages 1–6, 1990.
- [2] P. Bach, M. Braun, A. Formella, et al. Building the 4 processor SB-PRAM prototype. In *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, volume 5, pages 14–23, Jan. 1997.
- [3] A. Balkan, G. Qu, and U. Vishkin. Mesh-of-trees and alternative interconnection networks for single-chip parallel processing. Technical Report 2006-32, University of Maryland Institute for Advanced Computer Studies (UMIACS), 2006.
- [4] L. P. Carloni, K. McMillan, A. Saldanha, and A. L. Sangiovanni-Vincentelli. A methodology for correct-by-construction latency insensitive design. In *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 301 – 315, 1999.
- [5] D. E. Culler and J. P. Singh. *Parallel Computer Architecture*. Morgan Kaufmann, 1999.
- [6] W. J. Dally. Virtual-channel flow control. *IEEE Trans. Parallel Distrib. Syst.*, 3(2):194–205, Mar. 1992.
- [7] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, San Francisco, CA, 2004.
- [8] A. DeHon and R. Rubin. Design of FPGA interconnect for multilevel metallization. *IEEE Trans. VLSI Syst.*, 12(10):1038–1050, 2004.
- [9] A. Gottlieb, R. Grishman, C. Kruskal, K. McAuliffe, L. Rudolph, and M. Snir. The NYU ultracomputer—designing an MIMD shared memory parallel computer. *IEEE Trans. Comput.*, pages 175–189, Feb. 1983.
- [10] C. Grecu, P. P. Pande, A. Ivanov, and R. Saleh. Structured interconnect architecture: A solution for the non-scalability of bus-based SoCs. In *Proceedings of the Great Lakes Symposium on VLSI*, pages 192 – 195, 2004.
- [11] R. I. Greenberg and L. Guan. On the area of hypercube layouts. *Information Processing Letters*, 84:41–46, 2002.
- [12] D. J. Kuck. A survey of parallel machine organization and programming. *Computing Surveys*, pages 29–59, 1977.
- [13] F. T. Leighton. New lower bound techniques for VLSI. In *Proc. Of the 22nd IEEE Symposium on Foundations of Computer Science*, pages 1–12, 1981.
- [14] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann, 1992.
- [15] C. E. Leiserson. Fat trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput.*, 34(10):892–901, Oct. 1985.
- [16] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, et al. The network architecture of the connection machine CM-5. *Journal of Parallel and Distributed Computing*, 33(2):145–158, 1996.
- [17] K. Mehlhorn and U. Vishkin. Randomized and deterministic simulations of PRAMs by parallel machines with restricted granularity of parallel memories. *Acta Informatica*, 21:339–374, 1984.
- [18] D. Naishlos, J. Nuzman, C.-W. Tseng, and U. Vishkin. Towards a first vertical prototyping of an extremely fine-grained parallel programming approach. *Theory of Computer Systems*, 2003. Special Issue of SPAA 2001.
- [19] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh. Evaluation of MP-SoC interconnect architectures: A case study. In *IEEE International Workshop on System-On-Chip for Real-Time Applications*, pages 253 – 356, 2004.
- [20] L.-S. Peh and W. J. Dally. A delay model and speculative architecture for pipelined routers. In *Proceedings of the International Symposium on High-Performance Computer Architecture*, pages 255–266, 2001.
- [21] F. Petrini and M. Vanneschi. Performance analysis of wormhole routed k-ary n-trees. *International Journal of Foundations of Computer Science*, 9(2):157–178, 1998.
- [22] Semiconductor Industry Association. The international technology roadmap for semiconductors. <http://public.itrs.net/Files/2003ITRS/Home2003.htm>, 2003.
- [23] U. Vishkin, S. Dascal, E. Berkovich, and J. Nuzman. Explicit multi threading (XMT) bridging models for instruction parallelism. In *Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 140–151. ACM, 1998.
- [24] T. T. Ye and G. D. Micheli. Physical planning for on-chip multiprocessor networks and switch fabrics. In *Proceedings of the Application-Specific Systems, Architectures and Processors (ASAP)*, pages 97 – 107, 2003.
- [25] C.-H. Yeh. Optimal layout for butterfly networks in multilayer VLSI. In *International Conference on Parallel Processing (ICPP)*, pages 379 – 388, 2003.