Interconnection Network Organization and Its Impact on Performance and Cost in Shared Memory Multiprocessors *

Sunil Kim
IBM (MS 4305)
11400 Burnet Road
Austin, TX 78758
skim@austin.ibm.com
512-838-5318 (phone)

Alexander V. Veidenbaum
Department of EECS
University of Illinois at Chicago
Chicago, IL 60607-7053
alexv@eecs.uic.edu
312-413-9657 (phone)
312-413-0024(fax)

August 21, 1998

Abstract

Interconnection network organization has a significant impact on multiprocessor system performance. However, detailed and comprehensive studies of the performance and the effect of different organizations are virtually non-existent. In this study, three interconnection network organizations are evaluated as part of a cache-based shared memory multiprocessor system: tori, multistage and single-stage shuffle-exchange networks. A system size is limited to 256 or fewer processors, a range where wiring constraints can be largely ignored. The performance impact of topology choice and switch size and channel width are studied under three different constraints: fixed switch size and channel width, constant number of switch pins, and constant network cost. The cost model reflects switch size and channel width. We find that after a certain point the performance advantage of wider channels becomes small, and the network topology and switch size become the determining parameters. Our results show that the multistage network is the best network topology if cost is not the main limiting factor. Otherwise, the single-stage network is the most cost-effective network topology. 2-D torus networks are seriously limited in terms of performance and cost. The dimension of torus networks needs to be larger than 2 in order to have a reasonable relative performance vis-à-vis networks of other topologies.

1 Introduction

Advances in VLSI technology dramatically increased the speed of processors and memory and the overall system performance in the last decade. Parallel processing is the only way to increase the performance even further for a given state of VLSI technology. This approach uses multiple state-of-the-art uniprocessors to work in parallel on a given program and reduce the

*This work was supported in part by the National Science Foundation under Grant No. NSF 89-20891 and the State of Illinois.

total execution time. Multiple processing nodes in such a system are attached to interconnection networks. The nodes communicate through an interconnection network and system performance is significantly affected by the network organization.

This study concentrates on interconnection networks for shared memory parallel systems. A shared memory is formed by the *union* of memories in all the nodes and is directly addressable by all processors. A processor accesses shared memory by sending and receiving requests through an interconnection network. Such shared memory accesses tend to incur a much larger latency than memory accesses in uniprocessor systems due to interconnection network traversals.

The memory access latency consists of a network latency and a memory latency. Network latency is a delay experienced by a message traversing a network, and the latency is a function of many different parameters such as topology. Memory latency occurs when memory is accessed and it is largely determined by commodity DRAM performance. Both latencies increase with system size, the former due to increased communication distance and both of them due to contention. Network bandwidth is to allow pipelining and multiple outstanding memory accesses.

Short latency and high bandwidth networks are essential to achieve high performance in large multiprocessor systems. However, there is no consensus on the best network organization. The choice depends on the underlying system architecture, design parameters, programming model, and application domain. Consequently, many different network architectures have been implemented. For example, IBM SP-2 uses a bidirectional multistage shuffle-exchange network [33, 32], Cray T3D uses a 3-D bidirectional torus network [23], Thinking Machines CM-5 uses a fat-tree network [27] and Intel Paragon uses a 2-D mesh network [5].

Defining an optimal network organization for a given system requires an in-depth study of various aspects of network design and trade-offs in a system context. Some of the trade-offs are determined by physical implementation constraints, such as a limited number of pins and wires. Network scalability also has to be considered. Finally, a high performance network is expensive and often a major cost factor in a large-scale multiprocessor system. Network cost as well as performance needs to be optimized in a constrained design.

Comprehensive and detailed studies addressing both performance and cost as well as design trade-offs for different networks in a given system are virtually non-existent. It is also impossible to extract and compare network performance data across existing systems due to their total incompatibility. The networks have been compared analytically, but such studies abstract away some important system characteristics. For example, networks are usually studied in isolation, replacing the rest of the system - processors, caches, memory, and parallel programs and their interaction by a constant request arrival rate because accurate analytical modeling of all these components and their interaction is extremely difficult. Detailed studies of some networks have been performed via system simulation [31, 34]. Such simulation studies also have limitations and, in particular, typically do not cover a wide range of topologies and other system parameters due to experimental complexity.

This paper presents a detailed comparative study of different network organizations, behavior, and design trade-offs with respect to performance and cost for a shared memory multiprocessor. The same overall system organization, parallel benchmarks, and basic router design are used while the network organization is varied to allow direct comparison. This is accomplished via detailed simulation of a complete multiprocessor system executing a parallel benchmark program. Specifically, the performance and cost of multidimensional torus (TORUS), multistage shuffle-exchange (MSX), and single-stage shuffle-exchange (SSX) networks are analyzed and compared, where network size varies from 64 to 256. SSX networks are included as a cost-effective alternative in the range of system sizes considered and because of its appealing packaging properties.

As in other recent network studies [1, 2, 3, 12, 24], physical implementation constraints, such as a fixed number of available pins and/or wires, are considered in this study. Such constraints limit the *switch* (router) size and *channel* (data path) width. Wider channels increase network bandwidth allowing more data to be transferred per unit time and thus reduce network contention and delays at each switch. The switch size, i.e. the number of ports in a crossbar or router which is used to construct the network, affects the *average message path length* in the network. The average message path length is a number of switches (also called *hops*) a message traverses to reach its destination averaged over all messages sent during the execution of an application. A longer average message path length directly increases network latency and tends to cause more contention. For a given network topology, increasing the switch size can result in a shorter average message path length.

Switch size and channel width use the same resources, namely package pins and wires. Hence, when such resources are limited, a trade-off exists between the switch size and channel width such that increased channel width results in a smaller switch size and vice versa. In previous studies [1, 2, 3, 12], the effects of such trade-offs were evaluated for a given network topology and a technology-induced constraint. For example, a fixed number of pins (wires) was used constraining possible switch size and channel width for a network, or a bisection width of the network was fixed to a given value across all networks. Only the presence of such constraints makes it possible to compare different network organizations and also has a major impact on the results.

This study largely follows the approach used by Agarwal [3] in the first two constraints and adds an overall network cost as another constraint. A non-zero *node delay* (minimal switch latency) is used and is greater than the wire delay between nodes. All constraint models basically assume an *unlimited* silicon area and a limited number of IC package pins. Specifically, the constraint models are:

1. *Fixed switch size and fixed channel width:*. this constraint implies the same crossbar IC used for all networks.

2. *Constant number of switch pins:*. this constraint implies the same crossbar IC package with a fixed number of pins, e.g. a packaging constraint, used for all networks.

3. *Constant network cost:*. this constraint implies the same cost to build each network.

All the constraint models reflect the pin cost and largely ignore the wiring constraint. This is justified by the fact that all networks are pin-limited but, *within the system sizes we study (64 to 256)*, are not limited by wiring area. As pointed in [3, 11, 16] the pin constraint is often a more serious limitation than wiring area. Thus the bisection width constraint is ignored in this study. Network cost, on the other hand, can be a significant fraction of the overall system cost if custom-designed crossbar switch ICs are needed to achieve high-performance.

The main points differentiating this study are:

- *Uniform, system-level network comparison:* the same functional and timing models for switches, processors, caches, and memory are used with all networks.

- *Use of a Cost Constraint* to compare different networks.
- *Single-stage networks* are studied and compared with other networks.
- *Detailed multiprocessor architectural simulation* is performed.
- *Execution-driven traffic generation* by each thread of a parallel program: this includes timing which is affected by caches and a cache coherence scheme and processor stalls for memory accesses.

The rest of this paper is organized as follows. The evaluation methodology, system organization, and trace-driven simulation are described in Section 2. Performance metrics and the constraint models are discussed in Section 3 and 4, respectively. Experimental results are in Section 5. The summary and conclusions of this study are in Section 6.

2 Evaluation Methodology

We use detailed trace-driven simulations to evaluate network performance and design tradeoffs. This section presents the programming and cache coherence model used in this study and describe our benchmarks, trace generation and system models.

2.1 Programming Model and Cache Coherence

We use a shared-memory programming model with DOALL loops [1] (*a DOALL programming model*). In this programming model, a processor executes a subset of loop iterations within a DOALL as a single task. *COBEGIN and COEND* statements start and terminate a set of parallel tasks comprising a DOALL. A loop iteration may have private variables which are placed in local node's memory and their access does not require network traversal. Shared memory is interleaved across all nodes and accessed through the interconnect. The system uses weak consistency [15] and does not enforce cache coherence in hardware. Weak consistency allows memory requests to be issued and completed out of program order, when legal.

A simple software-initiated mechanism is used to maintain cache coherence [35]. Each processor's cache is invalidated and its write buffer is flushed to memory at the end of a parallel task (i.e. when it executes a *COEND* statement). The serial processor's cache is flushed at the end of a serial task (i.e. when it executes a *COBEGIN* instruction). The invalidate/flush operation blocks the processor until it is completed, including performing the writes in memory. A write-through cache guarantees that 1) all the data is posted to memory before the next task starts, and 2) that no stale data remains in the processor's cache. It has been previously shown that this software coherence scheme perform well on codes similar to the ones used in this study [9].

2.2 Benchmarks

Our application domain is computation-intensive numerical applications. These applications usually exhibit large data set sizes and large amounts of data communication between computations. Most of them can be parallelized using DOALL parallel loops.

[1] similar to FORALLs in HPF without requiring data partitioning

Benchmark	Subroutines traced	Global Read Access (M)	Global Write Access (M)	Read Message Generated(M)[2]	Write Message Generated(M)
FLO52	EFLUX	0.725123	0.263424	0.160802	0.263424
MDG	INTERF	4.550483	1.329371	1.098652	1.329371
TRFD	Whole Prog	7.628555	0.267484	0.353330	0.267484
ARC3D	STEP	2.675342	0.561258	0.438169	0.561258
MG3P	Whole Prog	3.355304	0.192632	0.161779	0.192632

Table 1: Benchmark Characteristics

We select our benchmarks among PERFECT [10] and NAS benchmarks [4] : ARC3D, FLO52, TRFD, MDG and MG3P. ARC3D is used to solve 3-D CFD problems. The next three are used to solve 2-D CFD problems, two-electron integral transform, and molecular dynamics problems, respectively. The last benchmark implements a 3-D multigrid algorithm.

Subroutines which account for a large fraction of a benchmark's execution time are selected and traced. We started with a parallel version of the programs that ran on CEDAR [17] multiprocessor. The benchmarks were further modified to increase the amount of parallelism and improve cache performance through the introduction of local variables, loop interchange and unrolling. All parallelism is expressed in terms of DOALL loops. Table 1 shows the characteristics of each benchmark. The number of read messages is shown for a 128 processor system since it changes with the number of caches in a system. The other characteristics remain the same regardless of system size.

The trace size for our benchmarks may appear small. Consider the case of FLO52 where we traced only one call to a subroutine EFLUX. However, EFLUX accounted for over 30% of the serial execution time and is called 2514 times during the FLO52 execution and with the same parameters. The memory system is flushed by invalidating caches and waiting for request completion at the end of the last DOALL loop before EFLUX is called. Thus, each execution of EFLUX tends to exhibit the same cache, network and memory behavior. In such cases, we traced only one execution of the subroutine.

2.3 Trace generation

Traces are obtained via instrumentation and execution of benchmark programs. A Program is instrumented by our MIPS tracing tool [8], compiled with the MIPS optimizing Fortran compiler (with "-o2 -mips1" flags) and executed on a MIPS system. The instrumented code generates during its execution memory reference addresses and instruction execution time between such references, as well as information about parallel loop execution. The trace is then processed by a reordering tool which reorders loop iterations to emulate a static loop scheduling algorithm by assigning iterations to processors: the ith processor is assigned the ith, $(i+N)$th, $(i+2*N)$th .... loop iterations. All sequential execution is performed by processor 0.

[2]for system size 128, 8K direct-mapped, write-through caches, line size 32 bytes.

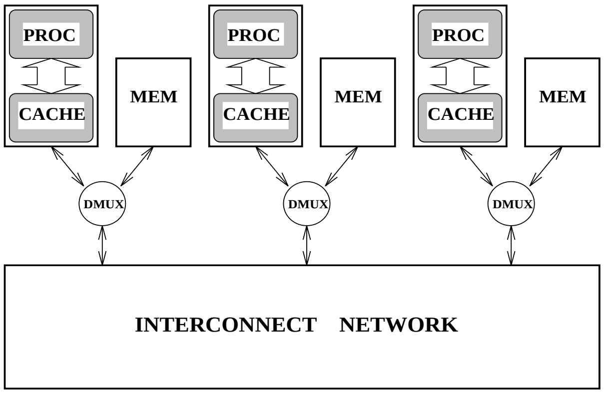

Figure 1: System Architecture

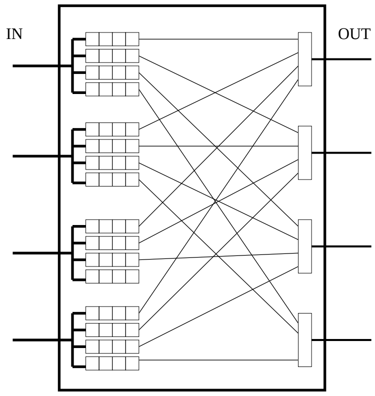

Figure 2: Non-Blocking Switch

2.4 Simulation Model

An interconnection network is part of shared memory system. Other system components affect message traffic and make it very important to model all the components. This section presents details of the overall system organization. The system consists of scalar processors, caches, memories and an interconnection network as shown in Figure 1. In this study, we vary the interconnection network organization while the rest of the system remains the same.

2.4.1 Shared Memory and Cache

The shared memory is physically distributed and directly addressable by each processor. Weak consistency is used with every write acknowledged when it has been performed in memory. The memory system is composed of pipelined, high-speed memory modules which are interleaved by the cache line size.

Memory module operation is pipelined to perform a memory access while accepting and assembling the next memory request message. Memory module latency is 4 processor cycles and is the time to send the first flit of a reply message header to the network after starting to process the request. The reply header generation is overlapped with DRAM access. The first data word from the DRAMs is ready when the entire message header is disassembled into flits and sent to the network. The memory module latency is quite low which we consider to be a requirement for high-performance multiprocessor memory systems. Techniques such as DRAM and memory caches and multi-bank design are assumed to be used to achieve this.

The memory cycle time depends on the message length and channel width. It consists of receiving a request message, performing a memory access, and sending a reply message. The memory module can buffer request messages while processing a request message to maximize throughput.

A direct mapped, write-though, no-write-allocate 8KB processor data cache is used with the line size of 4 words, and the word size of 8 bytes. The cache takes one cycle to service a cache hit. A processor stalls on a read miss until the request is satisfied. A 64-word write buffer is used and can accept a word of data every clock cycle. A processor continues as soon as the write buffer accepts the write. The processor stalls on both writes and reads when the write buffer becomes full. A compiler-initiated cache invalidation method is used to maintain the cache coherence.

Only cache misses and writes generate request messages to the network. At most one outstanding read and one write request per node are allowed simultaneously in the network. Read requests have priority over write requests on accessing the network. A read address matching a write address in the write buffer is not issued to the network until all the writes in the write buffer are issued, performed, and acknowledged.

A synchronization operation is started only after all pending read and write requests are *completed*. This includes issuing all write requests in the write buffer to the network and receiving acknowledgments. This is required to maintain the weak consistency.

All messages in the network have an 8Byte message header. A read request message and a write reply message consist of only the header. A write request message consists of a header and an 8Byte data word. A read reply message carries a header and one 32Byte cache line. The data word read by a processor is sent back first by the memory.

A request message is disassembled into flits and sent out to the network. The size of a flit is determined by the network data path width. A message is assembled from its flits when it arrives back at the requesting node. On a read reply, the word requested by a processor is forwarded to the processor as soon as it is assembled from the network flits. The processor resumes operation without waiting for an entire cache line to be assembled and written into the cache.

All non-shared data and instructions are stored in a separate part of memory *local* to a node. This memory is a large contiguous part of program's address space private to each node. Thus, only memory references to shared data are accessed through the network.

2.4.2 Scalar processor

Each node contains a standard RISC processor. MIPS R3000/R3010 is used to obtain the instruction timing information. Instructions execute in one cycle with the exception of integer multiplication and division, floating-point addition, multiplication and division, and a few other instructions. An *extension* to the MIPS3000 is the use of the 64-bit word size, address and data buses.

2.4.3 Crossbar Switch

A non-blocking switch is used [20, 25, 30] (see Figure 2) with a number of queues equal to the number of output ports per each input port. The message header contains the destination tag which selects the output port and the message is stored in the corresponding input queue. This eliminates head-of-line (HOL) blocking.

The switch supports the virtual cut-through flow control [21]. Thus, if there is no contention and no previous messages in a queue, the message experiences only the minimum switch latency of 3 cycles. A message is buffered in a switch if it can not move forward. A full-custom design of such a fast switch was shown in [13].

In this study, unlimited size queues are used. Other network studies have found a small difference in the performance of networks with unlimited queue size and with a reasonable queue size [6, 14].

At each output port, a *snapshot* of all messages requesting the output port access is taken once the port becomes free, and then all are serviced in a given fixed order. Fairness is enforced

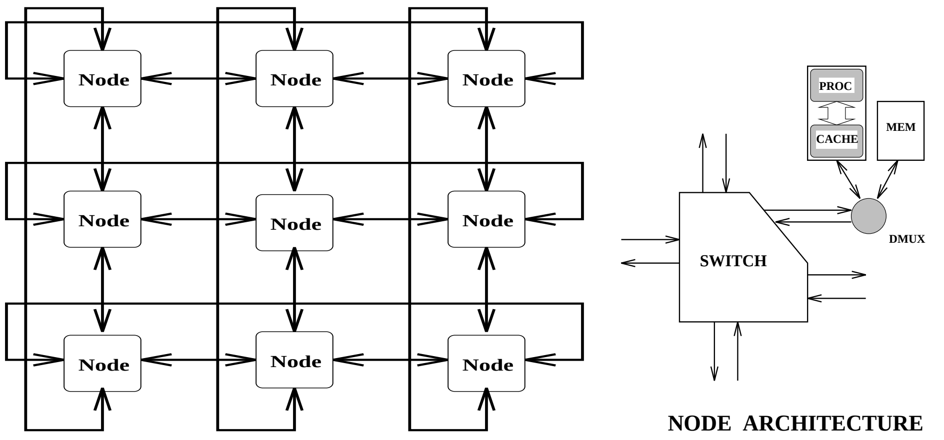

Figure 3: Multidimensional Torus Network System

between snapshots because no other message is serviced until all messages recorded in a snapshot are serviced.

2.4.4 Networks

Three different network topologies are studied: a multidimensional torus (TORUS), a multistage shuffle-exchange (MSX), and a single-stage shuffle-exchange (SSX). A node is connected to a network through a multiplexor and demultiplexor module (DMUX).

A torus network (Figure 3) uses bidirectional links implemented by a pair of unidirectional links. Each switch has an extra link to the DMUX. Dimension-order network routing is used.

A multistage shuffle-exchange network system (Figure 4) uses one unidirectional multistage network for both the request and reply traffic between processors and memories. A node is connected through a DMUX to the first and last stage of the network. A tag-based distributed routing algorithm proposed by Padmanabhan [28] is used.

A single-stage shuffle-exchange network system is shown in Figure 5. This construction has been used in previous studies [7, 26]. The shortest path routing algorithm proposed by the authors [22] is used.

Wires connecting switches incur a one clock cycle delay. The DMUX causes an additional one-cycle delay and also has infinite queues.

3 Performance Metrics

Read miss latency is used as the main performance metric. The processor stalls on a read miss and the read miss latency seen at the processor directly adds to the total execution time. The read miss latency also represents the performance of the entire memory system. Write latency is not as important because writes are buffered. Synchronization latency is also negligible in these DOALL loop based program.

Read miss latency (L) is the sum of the delays experienced in the network, memory and cache interface as shown in the following equation:

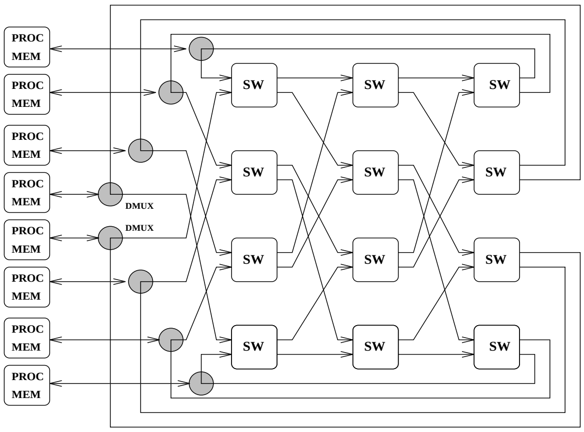

Figure 4: Multistage Shuffle-Exchange Network System

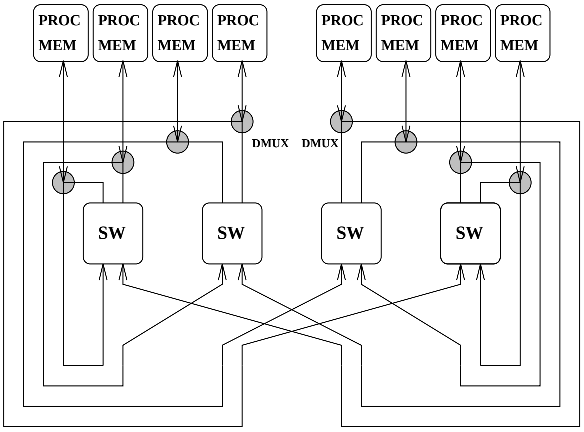

Figure 5: Single-Stage Shuffle-Exchange Network System

$$L = D_s \times PL + D_m + T_a$$

where
- L : Read miss latency
- D_s : Stage delay
- PL : Average message path length (hops) for a round trip to memory
- D_m : Memory delay, $D_m = T_{mw} + L_{ml}$
  - T_{mw} : Memory waiting time
  - L_{ml} : Memory module latency (4 cycles)
- T_a : Assembly time at a cache seen by a processor, $T_a = 128/W_c$
- W_c : Channel width

Network delay is the sum of stage delays (D_s) for a round trip to memory. A stage delay is the time required for a message to advance to the next switch in the message path. It contains a minimum switch latency, a switch queuing delay, and a wire delay.

Memory delay (D_m) has two components: *memory module latency* (L_{ml}) and *memory waiting time* (T_{mw}). The memory module latency is fixed, The memory module latency is the delay from when the memory module starts servicing a request until it starts sending out the corresponding reply message. The memory waiting time is the delay while a message waits in a queue before it is serviced by the memory. The memory module latency is fixed, whereas the memory waiting time is affected by *memory cycle time* and *contention at a memory module.*

T_a, the delay to receive and assemble the message header and the first word of the cache line for which the processor is waiting, depends on the channel width (flit size) as shown in the above equation. The size of the message header and the first word is 128 bits.

We measure read miss latency and each of its delay components for all network configurations and benchmarks through trace-driven simulation. Among the delays, L_{ml} is fixed to 4 cycles, and T_a depends only on the channel width. D_s and T_{mw} vary with network configuration as well as benchmarks.

The message path length PL in a MSX network is constant and is determined by the switch size and network size. For TORUS and SSX networks PL is a function of the source and destination of a message as well as the network and switch size. The term $D_s \times PL$ represents

switch size	TORUS			switch size	MSX			SSX		
	system size				system size			system size		
	64	128	256		64	128	256	64	128	256
5x5	9.96	13.98	17.98	4x4	6	8	8	5.2	6.42	7.17
7x7	7.96	9.98	11.99	8x8	4	6	6	3.69	4.77	5.37
9x9	7.96	8.98	9.99	16x16	4	4	4	3.44	3.72	3.86

Table 2: Average Message Path Length for ARC3D[3]

the network traversal time of the first flit under the virtual cut-through routing. Table 2 shows the average message path length for different networks as measured in simulation. *PL* for other benchmarks are almost the same in TORUS and SSX networks.

4 Network Constraint Models

The implementation of interconnection networks is constrained by the availability of physical resources such as pins and wires. The network implementation uses multiple VLSI crossbar chips and the number of chip pins is often considered a more serious limitation than wiring area [3, 11, 16]. The packaging cost of a large switch tends to become a dominant factor of switch cost. Thus the pin requirements of an interconnection network are a very important measure of network cost.

In [1, 3, 29], a constant pin limit was imposed on a node, and network performance was evaluated varying the node degree and channel width. Another important criterion is the total network cost in which both packaging constraints and total number of components in a network are considered together (this model is described in detail below). To recap, the interconnection networks are examined in this paper under three different constraint models:

- Fixed switch size and channel width.
  This model implies the same crossbar IC used for all networks.
- Constant number of switch pins.
  This model implies the same crossbar IC package with a fixed number of pins used for all networks.
- Constant network cost.
  This model implies the same cost to build each network. The network cost model is described in the next section.

4.1 Network and Switch Cost

The network cost model is based on a cost of the switch assuming the following:

- The same technology is used across all networks.

[3]The other benchmarks have almost the same average message path length

- A switch is implemented as a single IC.
- The switch cost is a major cost factor in a network.
- The switch cost is linearly proportional to the number of pins.

The actual cost of a switch chip depends on many parameters, such as wafer cost, yield, chip area, and packaging, making an accurate estimation difficult. We assume a linear relation between the switch cost and the number of pins based on the following observations. A portion of chip area is allocated for I/O pads, and in most cases of interest to us, the required number of I/O pads defines a die size large enough to fit all the required logic. In addition, for a wide range of IC package sizes, the packaging cost tends to increase linearly in proportion to the number of pins [19]. Thus, both silicon area and package costs can be approximated as a linear function of the number of pins.

Based on the above the switch cost (C_{sw}) and the network cost (C_N) are defined as follows.

$$
\begin{aligned}
C_{sw} &= 2 \times K \times W_c \\
C_N &= \begin{cases} C_{sw} \times N & \text{for TORUS networks} \\ C_{sw} \times N/K & \text{for SSX networks} \\ C_{sw} \times (N/K) \times \lceil \log_K N \rceil & \text{for MSX networks} \end{cases}
\end{aligned}
$$

where K : Switch size
W_c : Channel width
N : System size, a multiple of K

For TORUS networks the local node's processor/memory port is not counted in the number of switch ports. Thus a 5x5 TORUS switch is considered equivalent to a 4x4 switch used in other networks to equalize the comparison.

5 Experiments and Analysis

System and network performance of the three network topologies is studied varying switch size, channel width, and system size. Switch sizes of 4x4, 8x8, and 16x16, channel widths of 1, 2, 4, and 8 bytes, and system sizes of 64, 128, and 256 processors are examined. The 16x16 switch with an 8 byte channel width may not be feasible in today's packaging technology but is include to help determine the bounds on results.

Table 3 lists network capacity for various networks defined as the product of the number of links and the channel width and estimates the maximum amount of message traffic a network can move per unit time. A network with a higher capacity is less likely to suffer from network contention and tends to cost more.

The memory waiting time (T_{mw}) and the stage delay (D_s) for different network configurations are examined first. The analysis of T_{mw} and D_s, coupled with path length effects, helps in understanding the network comparison results under the three different constraint models. Following this analysis, results of the network comparison under the three constraint models are presented.

switch size	TORUS			switch size	MSX			SSX		
	system size				system size			system size		
	64	128	256		64	128	256	64	128	256
5x5	2048	4096	8192	4x4	1536	4096	8192	512	1024	2048
7x7	3072	6144	12288	8x8	1024	3072	6144	512	1024	2048
9x9	4096	8192	16384	16x16	1024	2048	4096	512	1024	2048

Table 3: Network Capacity : Channel Width 1 Byte

5.1 Stage Delay and Memory Waiting Time

T_{mw} and D_s for ARC3D are shown in Figure 6 and Figure 7 for three system sizes. T_{mw} only is shown for MDG in Fig. 8 because the stage delay D_s of MDG is similar to that of ARC3D. The results for other benchmarks are qualitatively similar to that of ARC3D and are not shown to save space.

Channel width affects T_{mw} and D_s the most for a given system size, and switch size has a much smaller effect. The variable part of the delay is due to contention and resulting queuing delays which implies that channel width has more impact on *network contention* than switch size. T_{mw} also grows significantly with system size while D_s does not. Network topology has a significant effect on D_s only when a narrow channel is used. It does not affect T_{mw} significantly. The stage delay D_s of SSX networks is higher than that of other networks with the same channel width. The network capacity of SSX networks is the lowest among the three network topologies causing more network contention.

Increasing the channel width from 1 to 2 bytes and 2 bytes to 4 bytes considerably reduces T_{mw} and D_s. The improvement is the largest for SSX networks. For 8byte channel width the delay D_s approaches its minimum of 4 cycles for TORUS and MSX networks and of 5 cycles for SSX networks (due to an additional clock cycle of DMUX delay). The low D_s is in part due to caches which eliminate a large number of shared memory accesses reducing network contention.

T_{mw} is also near its minimum for channel widths of 4 or 8 bytes except for MDG which has a hot-spot in its memory access pattern. A parallelized inner loop of MDG contains shared scalar variable references. These references cannot be kept in a register by the compiler due to lack of inter-procedural analysis and optimization for library functions. With software coherence scheme flushing caches at synchronization points all processors have to refetch the shared scalar variable creating a hot spot.

T_{mw} increases as the system size increases, whereas D_s remains virtually the same, particularly with the channel widths equal to or larger than 4 bytes. This implies that once the channel width becomes sufficiently large (4 and 8 bytes in our case), average message path length, *PL* becomes a major factor in determining network performance for a given system size.

5.2 Fixed Switch Size and Channel Width

In this section the performance of different network topologies is compared when using the same switch size and channel width. This covers the case when all networks are built with the same crossbar switches. In this case the network topology determines the average message path

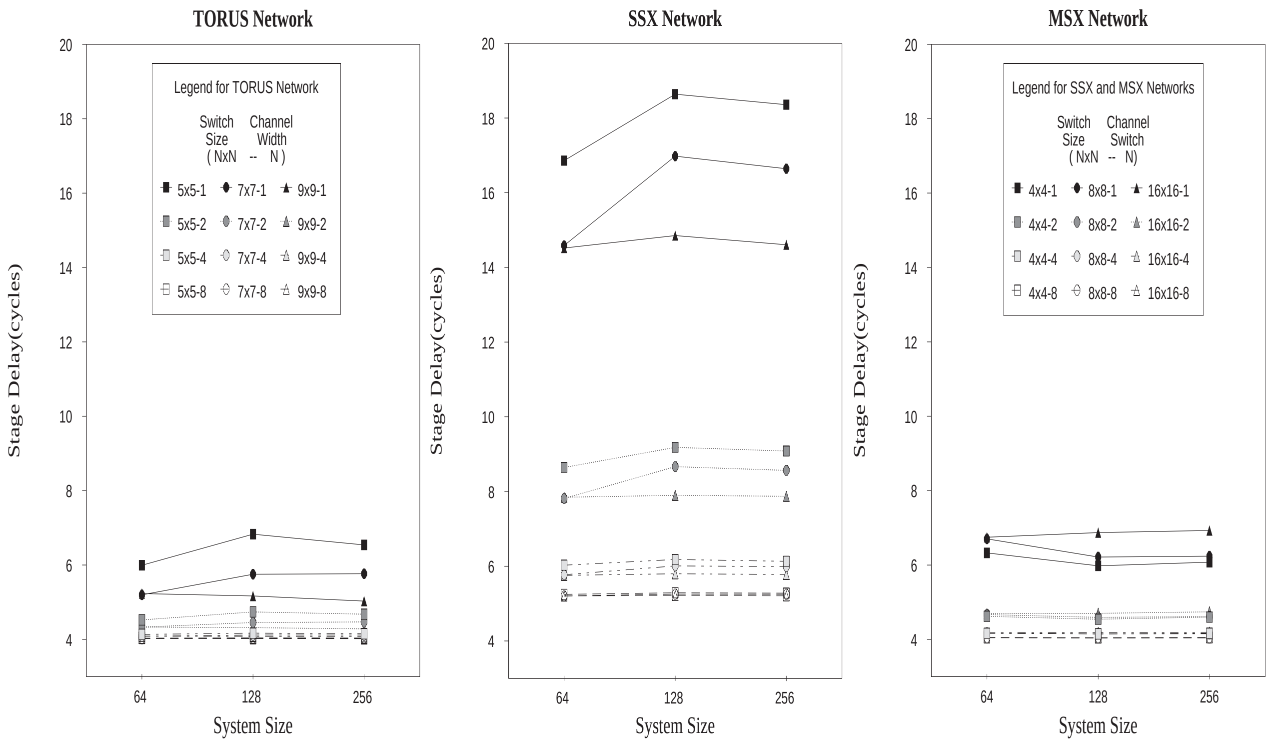

Figure 6: Stage Delay (ARC3D)

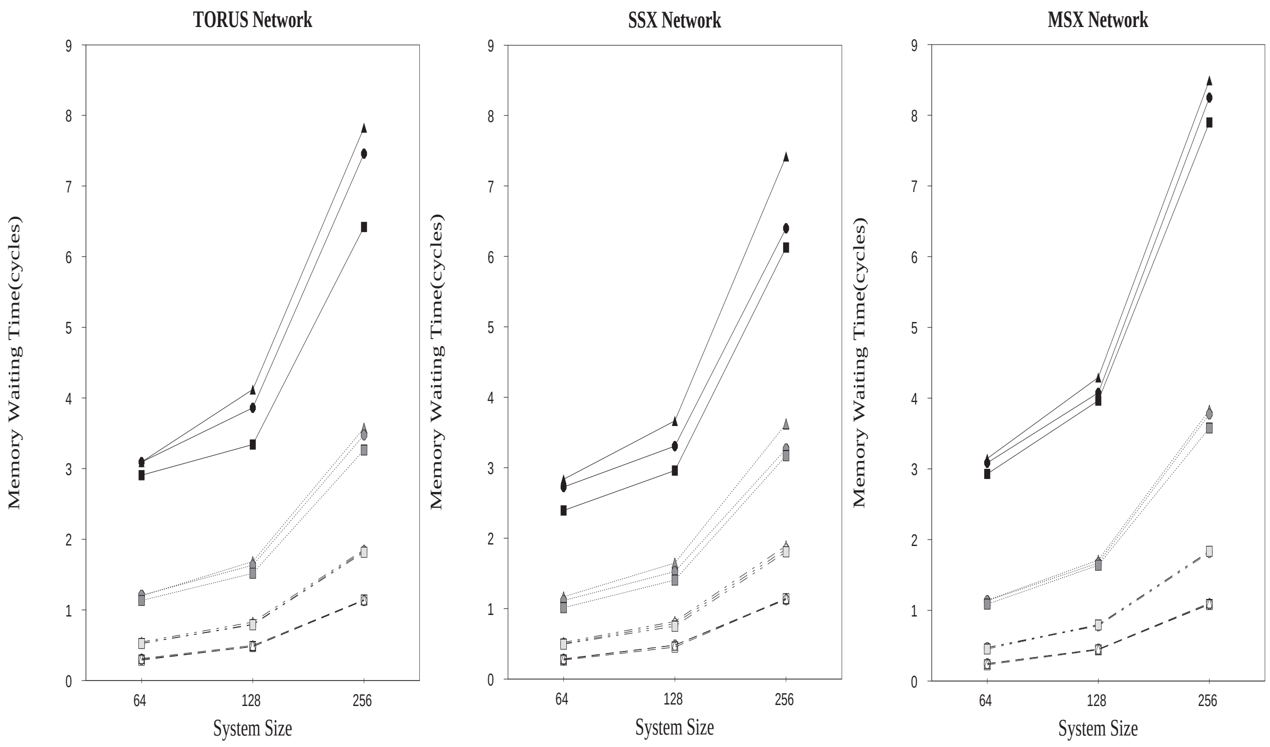

Figure 7: Memory Waiting Time (ARC3D)

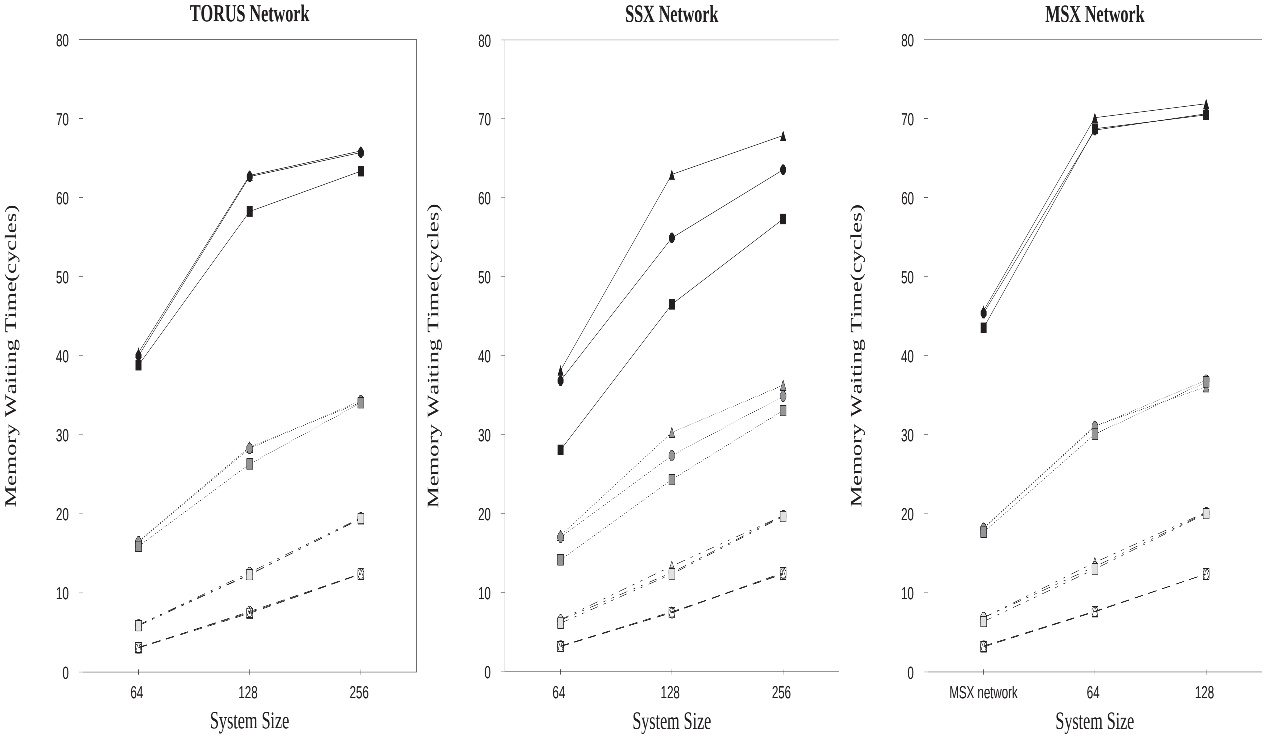

Figure 8: Memory Waiting Time (MDG)

length (PL) for a given system size. Table 4 shows performance difference between networks for the system size of 128. Other system sizes have similar trends and are not shown. A table entry is a relative difference in the read miss latency between two networks. For instance, the entry in column labeled "SSX/TORUS" is defined as $(L_{TORUS} - L_{SSX})/L_{TORUS} \times 100$, where L_{SSX} and L_{TORUS} are the read miss latency of SSX and TORUS networks, respectively. The column shows a read miss latency improvement of SSX networks over TORUS networks. 5x5 and 9x9 switch configurations of TORUS networks are considered under columns 4x4 and 8x8, respectively. Recall that TORUS networks we use do not have a configuration corresponding to the 16x16 switch.

MSX networks perform better than TORUS or SSX networks for all configurations. SSX networks have the worst performance for channel width of 1 byte. As the channel width increases, the relative performance of SSX networks improves significantly. It always exceeds that of TORUS networks for channel width of 4 bytes or more. The relative difference in latency for a given program, such as FLO52, can range from -100 % to +22.5 % , showing its sensitivity to channel width. As the channel width increases, the performance gap between SSX and MSX networks for most benchmarks narrows to below 15 % for an 8byte channel.

The performance difference between MSX and TORUS networks increases as the channel width increases, and the maximum difference reaches 35.2 %. On the other hand, the relative performance of TORUS improves when the switch size is increased from 4x4 to 8x8. This is due to a large decrease in PL for system size of 128 and 256, as shown in Table 2.

In summary, MSX networks are the best under the constraint of fixed switch size and channel

channel width (bytes)	SSX/TORUS 4x4	SSX/TORUS 8x8	MSX/TORUS 4x4	MSX/TORUS 8x8	MSX/SSX 4x4	MSX/SSX 8x8	MSX/SSX 16x16
ARC3D							
1	-26.2	-40.4	34.4	12.8	58	49.3	32.6
2	9.52	0.453	33.3	19.1	39.5	32.2	20.4
4	28.3	19.9	34	22.7	23.2	19	11.9
8	35.5	26.7	35	24.3	15.6	12.8	8.39
FLO52							
1	-56.5	-100	38.1	4.14	66.8	60	45.1
2	-30.4	-45.1	35.4	14.1	59	51.1	35.8
4	11.2	3.24	34.1	20.5	38.8	31.9	20.8
8	31.1	22.5	35	23.7	21.3	17.6	11.5
MDG							
1	-11.1	-20.5	15	1.56	32.6	24.7	11.9
2	1.08	-3.76	18.1	6.95	26.7	18.2	9.26
4	19.5	11.9	25	13.6	18.5	12.3	5.65
8	29.3	21.2	29	19.1	12.2	9.41	5.61
MG3P							
1	-35.4	-52.7	33.3	10	60.2	51.4	35
2	3.62	-3.82	32	17.6	42.9	34.6	21.2
4	26.7	18.3	33.3	21.7	24.4	19.4	11.9
8	34.8	26	34.6	23.8	15.7	12.9	8.41
TRFD							
1	-17.8	-33	30.2	10	49.6	42.3	29.8
2	3.91	-8.11	31.9	16.6	39.8	33.9	23.9
4	22.8	13.7	33.7	22	27.1	23.1	15.4
8	33.9	25.2	35.2	24.3	17.2	14.1	9.04

Table 4: Relative Read Miss Latency Difference(%) for System Size 128.

width. The relative performance of SSX networks largely depends on channel width. For wider channels, SSX networks perform better than TORUS networks and approach the performance of MSX networks. In all cases, the difference decreases for a larger switch size.

5.3 Constant Number of Switch Pins

This constraint model restricts the total number of switch pins and allows the switch size and channel width to change within this limit. This corresponds to the case when all networks use the same IC package for crossbar switches. A trade-off between the switch size and channel width under the constraint are examined. The minimum number of switch pins is 64 corresponding to a 4x4 switch with channel width of 1 byte. The maximum is 2048 corresponding to 16x16 switch with channel width of 8 bytes. Results are shown for ARC3D (Fig. 9) and MDG (Fig. 10). Other benchmarks are again qualitatively similar to that of ARC3D and are not shown. In a graph each point from left to right corresponds to 1, 2, 4 and 8 byte channels, respectively.

ARC3D For a given number of switch pins there exists an MSX configuration with the best performance. With 64 switch pins, a SSX network has the worst performance. However, when 128 or more pins are allowed, SSX networks have at least one configuration that is better than all TORUS networks. As the pin limit approaches 2048, the performance difference between SSX and MSX networks becomes very small.

Given a small number of switch pins, their best use in MSX networks appears to be to increase the channel width using small switches. As the number of available pins grows, the trade-off between switch size and channel width favors a larger switch size. The same is generally true for SSX networks. For example, when the system size is 256, the read miss latency of the 4-ary MSX network with 4-byte channels becomes almost equal to that of the 8-ary MSX network with 2-byte channels at 256 pins. A MSX network using a larger switch size starts to have better performance than a network with a wider channel beyond that point.

TORUS networks behave differently. 4-D TORUS or 3-D TORUS networks are always better than 2-D TORUS networks over the entire range of switch pins in all but one case. This is explained by a large reduction in *PL* for increased dimensionality. For system size 64 and 128 pins the trade-off favors 2-D instead of 4-D network. This result are consistent to some degree with the results shown in [3], where a small system favors a lower dimensional mesh network whereas a larger system favors 3-D or 4-D mesh networks.

MDG MDG shows different trends than ARC3D and the other benchmarks. An MSX or SSX network with wider channels always performs better at a given number of switch pins for the entire range of system sizes. In TORUS networks the trade-off favors larger switch sizes between 2-D and 4-D TORUS when there are more than 512 pins available, which allows 2-D TORUS networks to have 8-byte channels.

This trend is explained by the dominant effect of T_{mw} on read miss latency as previously shown in Section 5.1. Wider channels improve performance much more than larger switches because T_{mw} is more affected by channel width.

The MDG results predict the performance in case a very slow memory was used. The slow memory will tend to increase T_{mw}, but will not affect D_s much due to the feedback effect. This

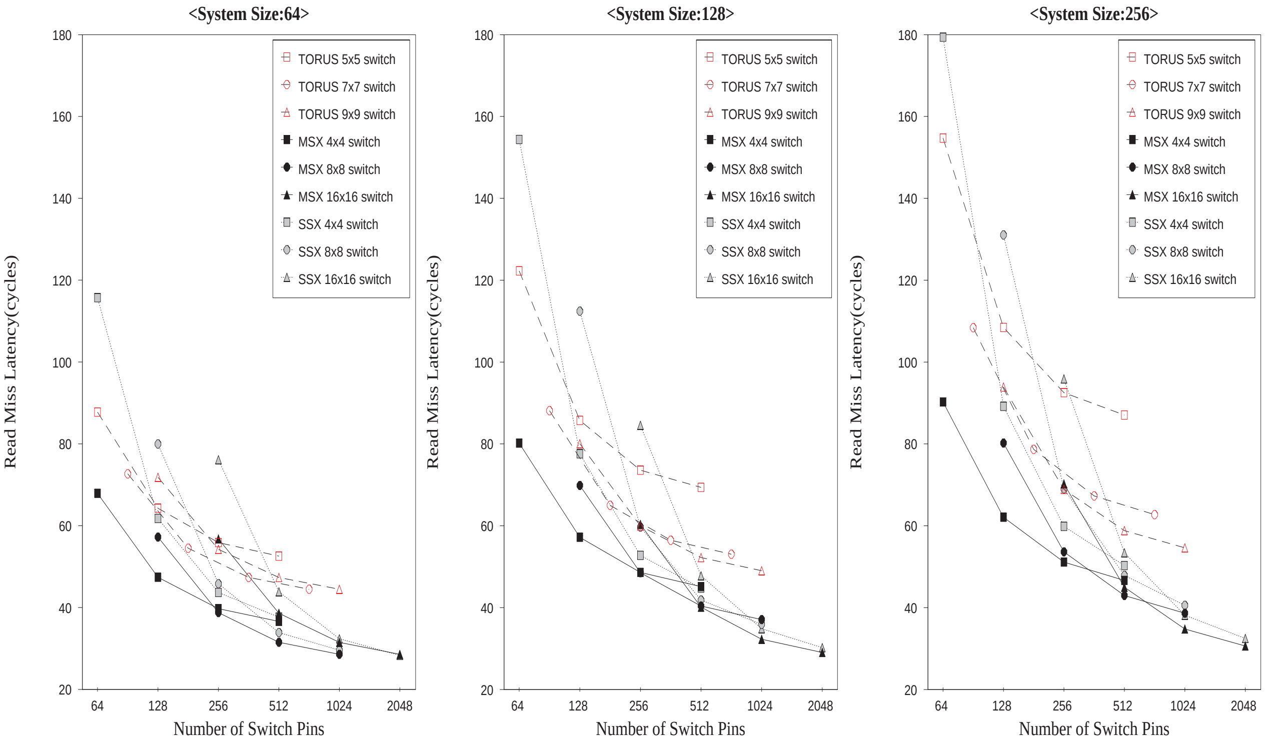

Figure 9: ARC3D: Read Miss Latency vs. Number of Switch Pins

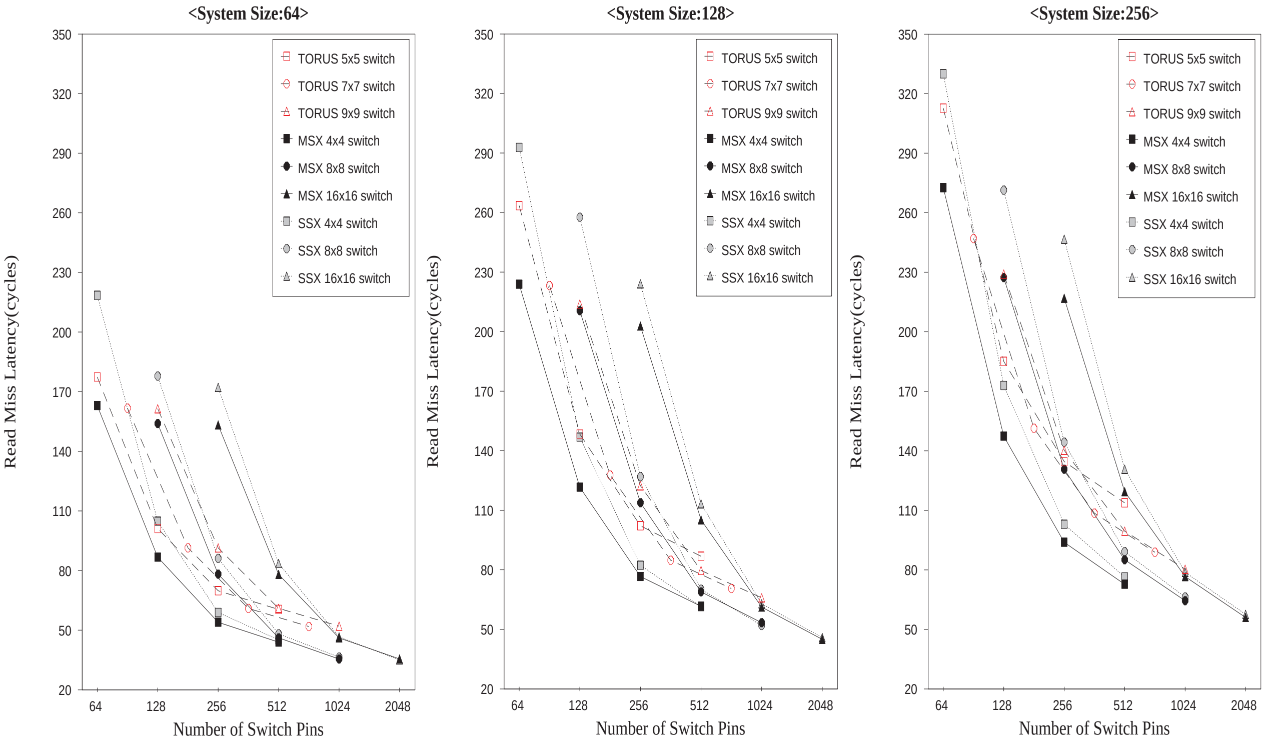

Figure 10: MDG: Read Miss Latency vs. Number of Switch Pins

requires higher network to memory and memory to network bandwidth and favors networks with wider channels.

5.4 Constant Network Cost

Networks have been evaluated so far under two design constraints: a fixed switch size and channel width and a constant number of switch pins. Another important question is which network performs better at a given cost. In this section, the cost issue is investigated using the model proposed in Section 4.1.

Figures 11 and 12 show the performance as a function of normalized network cost for all system sizes (ARC3D and MDG only). The cost is normalized to that of 4-ary SSX network with 1-byte channels and ranges from 1 to 64. The latter is the cost of a 4-D TORUS network with 8-byte wide channels. Also note that within a graph, each point from left to right corresponds to 1, 2, 4 and 8 byte channels, respectively.

ARC3D results show that for any system size the best networks at a given cost are SSX networks and the worst are TORUS networks. This is due in part to the fact that SSX cost is independent of switch size. The MSX cost is proportional to $log_K N$, while the TORUS cost is proportional to switch size K. $log_K N$ is a very small constant for system sizes considered here.

Among SSX networks the 16-ary networks have the best performance for the entire cost range, as is the true for MSX networks. 4-D TORUS networks are always better than 2-D TORUS networks for system size of 128 and 256. For system size of 64, 2-D TORUS networks initially perform better than 4-D TORUS networks. The trend is reversed as the network cost increases.

In general, read miss latency curves move towards better cost-performance with increase in switch size. A higher network cost does not guarantee a higher performance and the performance difference between the networks that have the same cost can be very high. For instance, for system size 256 and network cost of 8, the best network has a read miss latency of 32.5 cycles and the worst network has 108 cycles, a latency increase of over 300 %. Thus significant cost/performance trade-offs can be made. For example, for system size 128 an MSX network costs 16 for a read miss latency of 29 cycles while an SSX network has a read miss latency only 5.8 cycles longer (34.8 cycles) at one quarter of the MSX network cost.

MDG results are generally similar in this case. The difference is even more pronounced here, as the latency curves for different topologies do not cross each other. Due to hot spot contention at memory, the memory delay is large. This is similar to the case of using slower DRAMs. If a slower memory was used in the system, results for all benchmarks would be similar to those of MDG. This implies that SSX networks would be favored over other networks with slower memory at a given cost.

6 Conclusions

The performance and design trade-offs for three interconnection network topologies in cache-based shared memory multiprocessor systems were studied. Using execution-driven simulation an entire system rather than a network in isolation was modeled to understand network behavior. This allowed the effect of other system components, such as cache stalls, write buffer stalls, flow control, and processor read stalls, to be incorporated. Read miss latency was used as the

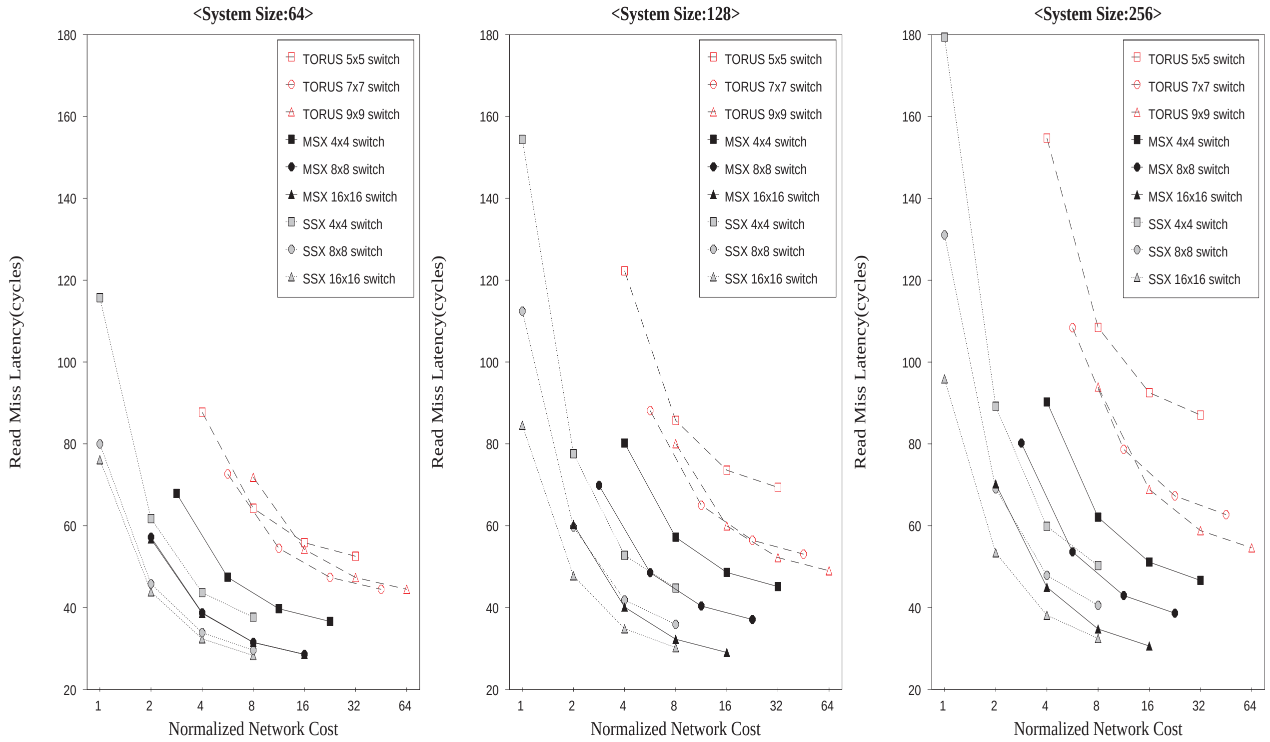

Figure 11: ARC3D: Read Miss Latency vs. Normalized Network Cost

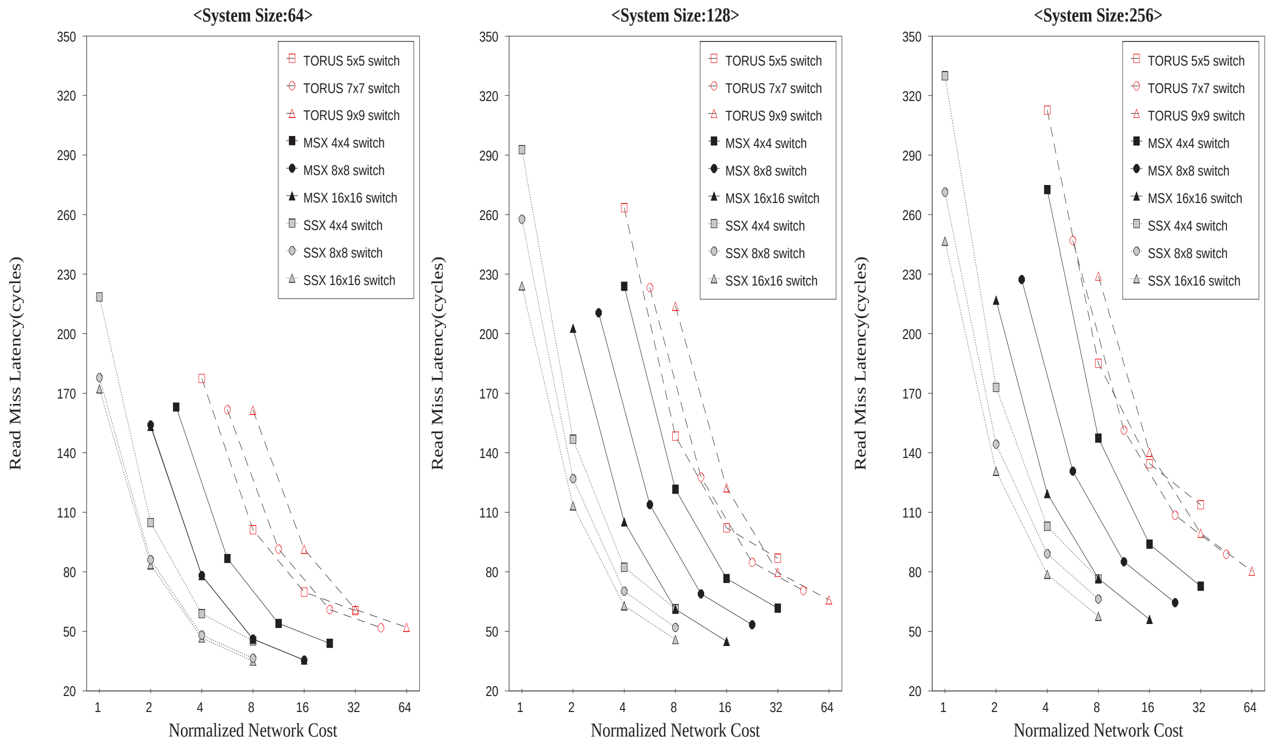

Figure 12: MDG: Read Miss Latency vs. Normalized Network Cost

main performance metric. Performance of different networks was compared using a common architecture, including basic router switch, under a common set of constraints. Three different constraint models were used reflecting the limitations imposed on network structure, IC packaging, and network cost.

Switch size and channel width are two very important network organization parameters as they largely determine the average message path length and message size, respectively. They affect the network stage delay and memory waiting time, the two main components of the read miss latency. It was shown that stage delay was affected the most by the channel width, and memory waiting time was greatly affected by the channel width, system size and memory access pattern. As the channel width increased beyond 4 bytes, the difference in the stage delay and memory waiting time of different networks became small and the average message path length became a dominant cause of network performance difference.

The results also showed that for a fixed switch size and channel width multistage shuffle-exchange (MSX) networks performed better than the other networks. The performance difference between networks of different topologies varied dramatically with channel width. The increase in channel width resulted in the largest performance gain for single-stage shuffle-exchange (SSX) networks, boosting their performance close to that of MSX networks. The increase had the smallest effect in multidimensional torus (TORUS) networks.

An important trade-off in switch design is between switch size and channel width when the total number of switch IC package pins is limited. The results showed that it was best to increase the channel width in a small switch if only a small number of switch pins was available. As the number of available pins grew, it became more beneficial at some point to use them to increase the switch size. The number of pins at which this happens was dependent on the system size and switch size for each network topology. Overall, under this packaging constraint the MSX networks had the best overall performance. Given sufficient number of pins SSX network performance improved over the TORUS networks and approached that of the MSX networks.

Significant differences in the read miss latency were observed among the networks at a given cost. A substantial network cost reduction was possible when a small performance degradation could be tolerated, clearly demonstrating the importance of network choice. In general, the best networks at a given cost were SSX networks with TORUS networks showing the worst performance at the same cost.

In summary, under the conditions and parameters examined the MSX network was the best network topology when cost was not the main limiting factor. Otherwise, the SSX network was the best network topology when cost was a main limiting factor. the TORUS network was seriously limited by longer average message path length in spite of the fact that full-duplex bidirectional channels were used, and the TORUS networks had highest overall network capacity. Significant trade-offs are possible under a given set of constraints. Such an optimization requires detailed evaluation using multiple parameters.

References

[1] Seth Abraham. *Issues in the Architecture of Direct Interconnection Schemes for Multiprocessors.* PhD thesis, University of Illinois at Urbana-Champaign, march 1990.

[2] Seth Abraham and Krishnan Padmanabhan. Constraint based evaluation of multicomputer networks. In *International Conference on Parallel Processing*, volume I, pages 521–525, 1990.

[3] Anant Agarwal. Limits on interconnection network performance. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):398–412, October 1991.

[4] D. Bailey, H. Simon, J. Barton, and T.Lasinski. The NAS parallel benchmarks. Technical Report RNR-91-02, NASA Ames Research Center, 1991.

[5] Rudolf Berrendorf, Heribert C. Burg, Ulrich Detert, Ruediger Esser, Michael Gerndt, and Renate Knecht. Intel Paragon XP/S - architecture, software environment, and performance. Technical Report KFA-ZAM-IB-9409, KFA Research Centre, Juelich, 1994.

[6] P-Y. Chen, D. H. Lawrie, P-C. Yew, and D. A. Padua. Interconnection networks using shuffles. *IEEE Computers*, pages 55–64, 1981.

[7] Pin-Yee Chen, Pen-Chung Yew, and Duncan Lawrie. Performance of packet switching in buffered single-stage shuffle-exchange networks. In *International Conference on Distributed Computing Systems*, pages 622–627, 1982.

[8] Yung-Chin Chen. *Cache Design and Performance in a Large-Scale Shared-Memory Multiprocessor System.* PhD thesis, University of Illinois at Urbana-Champaign, 1993.

[9] Yung-Chin Chen and Alexander Veidenbaum. Comparison and analysis of software and directory coherence schemes. In *Supercomputing91*, pages 818–829, 1991.

[10] G. Cybenko, L. Kipp, L. Pointer, and D. Kuck. Supercomputer performance evaluation and the perfect benchmarks. In *International Conference on Supercomputing*, 1990.

[11] Robert Cyper. Theoretical aspects of VLSI pin limitation. *SIAM Journal on Computing*, 22(2):356–379, April 1993.

[12] William J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Transactions on Computers*, 39(6):775–784, June 1990.

[13] Andre DeHone, Frederic Chong, Matthew Becker, Eran Egozy, Henry Minsky, Samuel Peretz, and Jr. Thomas F. Knight. Metro: A router architecture for high-performance, short-haul routing networks. In *International Symposium on Computer Architecture*, pages 266–277, April 1994.

[14] Daniel M. Dias and J. Robert Jump. Analysis and simulation of buffered delta networks. *IEEE Transactions on Computers*, C-30(4):331–340, April 1981.

[15] M. Dubois, Scheurich C, and F. Briggs. Memory access buffering in multiprocessors. In *International Symposium on Computer Architecture*, pages 434–442, 1986.

[16] Mark A. Franklin, Donald F. Wann, and William J. Thomas. Pin limitations and partitioning of VLSI interconnection networks. *IEEE Transactions on Computers*, C-31(11):1109–1116, November 1982.

[17] D. Gajski, D. Kuck, D. Lawrie, and A. Sameh. Cedar - a large scale multiprocessor. In *International Conference on Parallel Processing*, pages 524–529, 1983.

[18] Anoop Gupta, John Hennessy, Kourosh Gharachorloo, Todd Mowry, and Wolf-Dietrich Weber. Comparative evaluation of latency reducing and tolerating techniques. In *International Symposium on Computer Architecture*, pages 254–263, May 1991.

[19] Linley Gwennap. Alternative packages emerge for processors. Microprocessor Report, October 1993.

[20] Mark J. Karol, Michael G. Hluchyj, and Samuel P. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Transactions on Communications*, COM-35(12):1347–1356, December 1987.

[21] Parviz Kermani and Leonard Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.

[22] Sunil Kim and Alexander Veidenbaum. On shortest path routing in single stage shuffle-exchange networks. In *Seventh Annual ACM Symposium on Parallel Algorithms and Architectures 95*, July 1995.

[23] R. Kent Koeninger, Mark Furtney, and Martin Walker. A shared memory MPP from Cray research. *Digital Technical Journal*, 6(2):8–21, Spring 1994.

[24] Smaragda Konstantinidou. *Deterministic and Chaotic Adaptive Routing in Multicomputers.* PhD thesis, University of Washington, may 1991.

[25] Manoj Kumar and J. R. Jump. Performance enhancement in buffered delta networks using crossbar switches and multiple links. *Journal of Parallel and Distributed Computing*, (1):81–103, 1984.

[26] D. H. Lawrie and D. A. Padua. Analysis of message switching with shuffle-exchanges in multiprocessors. In *The Proceedings of the Workshop on Interconnection Networks for Parallel and Distributed Processing*, pages 116–123, 1980.

[27] Charles Leiserson and et al. The network architecture of the connection machine CM-5. In *Symposium on Parallel Algorithms and Architectures92*, pages 272–285, June 1992.

[28] Krishnan Padmanabhan. Design and analysis of even-sized binary shuffle-exchange networks for multiprocessor. *IEEE Transactions on Parallel and Distributed Systems*, 2(4):385–397, October 1991.

[29] Krishnan Padmanabhan. On the tradeoff between node degree and communication channel width in shuffle-exchange networks. In *Proceedings, IEEE Symposium on Parallel and Distributed Processing*, pages 120–127. IEEE, December 1991.

[30] Ora E. Percus and Susan R. Dickey. Performance analysis of clocked-regulated queues with output multiplexing in three different 2x2 crossbar switch architectures. *Journal of Parallel and Distributed Computing*, (16):27–40, 1992.

[31] J. E. Smith and W. R. Taylor. Accurate modeling of interconnection networks in vector supercomputers. In *International Conference on Supercomputing*, pages 264–273, June 1991.

[32] Craig B. Stunkel, Dennis G. Shea, Bulent Abali, Monty M. Denneau, Peter H. Hochschild, Douglas J. Joseph, Ben J. Nathanson, Michael Tsao, and Philip R. Varker. Architecture and implementation of Vulcan. In *Proceeding of the 8th International Parallel Processing Symposium*, pages 268–274, April 1994.

[33] Craig B. Stunkel, Dennis G. Shea, Don G. Grice, Peter H. Hochschild, and Michael Tsao. The SP1 high-performance switch. In *Proceedings of the Scalable High Performance Computing Conference*, pages 150–157, May 1994.

[34] Steve Turner and Alex Veidenbaum. Performance of a shared-memory system in a vector multiprocessor. In *International Conference on Supercomputing*, pages 315–325, July 1988.

[35] Alexander Veidenbaum. A compiler-assisted cache coherence solution for multiprocessors. In *International Conference on Parallel Processing*, pages 1029–1036, 1986.