Switch Design for Fine-Grained Multicomputers-on-a-Chip*

Martin C. Herbordt

Department of Electrical and Computer Engineering
Boston University; Boston, MA 02215 USA
EMail: herbordt@bu.edu Phone: 1.617.353.9850

Kurt Olin

Hewlett Packard Corporation
Houston, TX 77070 USA

*This work was supported in part by the National Science Foundation through CAREER award #9702483, by the Texas Advanced Research Program (Advanced Technology Program) under grant #003652-952, and by a grant from the Compaq Computer Corporation.

Abstract: Previous studies in network switch design have generally not considered simultaneously both global communication performance and local effects such as critical timing path and chip area. Here a comparison is made among a large number of designs for the purpose of specifying cost-effective communication switches for multicomputers-on-a-chip. We obtain results using two methods: (i) RTL cycle-driven simulations to determine latency and capacity with respect to load, communication pattern, and packet size and (ii) hardware synthesis to a current technology to find the operating frequency and chip area. These are combined to yield performance/area measures for all of the designs; a number of interesting findings are made. One is a deeper understanding of virtual cut-through in terms of deadlock properties and the capability of dynamic load balancing among buffers. We find that virtual cut-through routing is preferable to wormhole routing in more domains than may have been previously realized. Another result is that, after factoring in operating frequency, having more than two lanes per physical channel is counterproductive. Buffer sharing among lanes was found to be useful, but only for certain simple designs under a hot-spot workload. The most important result, perhaps, is the finding that creating cost-effective switches requires that designs be evaluated with respect to both of the performance metrics.

Keywords: Communication switch design, multicomputer networks, systems-on-a-chip, network simulation.

For special issue on Networks on Chip

1 Introduction

With the continued advance of VLSI technology, the monolithic microprocessor no longer necessarily dominates parallel computer architecture. In particular, designs based on the replication of processor IP blocks on an ASIC have become a viable alternative, if not for general purpose computing, then at least for high-performance applications in signal and image processing and in bioinformatics and computational biology. As a result, classes of parallel computers that were discarded in the early 1990s for not being optimal with respect to the then-current technology are again promising. One of these is the fine-grained multicomputer, perhaps best exemplified by the J-Machine. The problem addressed here is the design of network switches for such systems-on-a-chip with an approximate range of 16 to 1K nodes, although many of our results are quite general.

Multicomputer switch design has been extraordinarily well-studied. However, most of those studies have either ignored technology or have focused on switches encompassing an entire chip. And the studies that have accounted for technology–to the point of looking at the timing implications of various design decisions–generally have not encompassed macro-level implications of those decisions on communication performance. In designing multicomputers-on-a-chip, however, we must simultaneously study the implications of design decisions on chip area, operating frequency, and cycle-level packet performance.

Because of the technological focus, we can make certain assumptions: the routing algorithm is deterministic and dimension order, input queuing is used, the topology is either a

2D mesh or torus, a single physical channel exists between nodes per direction per dimension, the packets are assumed to be relatively small and have fixed size, and inter-node flit propagation takes place within a single cycle. The less obvious of these assumptions are justified below. Still, there are a large number of variables to consider; these include: the switching mode (virtual cut-through [VCT] and wormhole [WH]), whether the network has wraparound connections or not (mesh or torus), is uni- or bi-directional, the number of lanes per channel, the size of the buffers, and the buffer sharing mechanism, if any.

The large body of previous work in multicomputer communication is referenced throughout the paper and summarized in a later section. The present work adds to those previous studies in that it accounts for both physical properties, such as critical path timing and layout area, and latency/capacity results from register transfer level simulations. However, there are other unique aspects: we account for (i) variations in buffer sharing among lanes and channels; (ii) not only bidirectional tori, but meshes and unidirectional tori as well; and (iii) static versus dynamic lane selection derived from deadlock considerations. Some of these issues have been considered previously but not all and not simultaneously as they are here. As such we have gained insight especially into some of the subtle differences between VCT and WH switching.

Some of the key results presented here are as follows:

- For equal numbers of lanes and lane size, VCT switching is likely to have better performance than WH switching. The reason is that the space guarantee associated with VCT switching is easy to implement for small packets and has powerful consequences in allowing additional load balancing among lanes and, to a lesser extent, in decreasing flow control latency.
- When operating frequency is factored in, increasing the number of lanes beyond two per physical channel is not likely to be cost-effective. This is in contrast to the cycle-only model where the benefits of increasing the number of lanes diminishes but does not reverse.
- One mechanism for buffer sharing among lanes was found to be cost-effective, but only under the hot spot workloads.

While investigating the design framework, it was necessary to work out some issues that are themselves contributions. One of these is the design of hybrid SRAM/register FIFOs that retain the speed advantage of registers and most of the size advantage of SRAMs. Perhaps our primary contribution, however, is to show that studying on-chip networks with accounting for the timing implications of design decisions can result in gross errors.

The rest of the paper is organized as follows. The next section describes the design space and high-level issues in WH and VCT switching. There follows a description of the basic hardware models and some details of the implementations. After that come the results followed by a review of some previous work and the conclusion.

2 Design Space

We now present the details of our network design space. Since some of the configurations are new and others have non-trivial motivation, we also discuss deadlock in WH and VCT networks and virtual channel selection in WH networks. We end this section with an analysis of the design consequences of the choice between WH and VCT switching.

2.1 Basic Network Assumptions and Parameters

In keeping with the technological focus of our study, we assume a two dimensional network and dimension order routing (DOR). The first assumption is justified through the obvious lay-out advantages of a two dimensional network for a single-substrate parallel computer. Although it is possible to embed higher dimensional networks on a chip, this necessarily results in long wires which substantially increase the cycle time in modern technologies. Another problem is a decrease in channel bandwidth due to packing constraints. The second assumption (DOR) is made because of its inherent attractiveness: DOR has low hardware cost [5] and its minimal number of turns reduces the possible collision points for each packet [15]. A parallel study has justified this design decision directly: under the technological constraints applied here, we have found that only the simplest of adaptive algorithms are ever beneficial and those only marginally and infrequently [13].

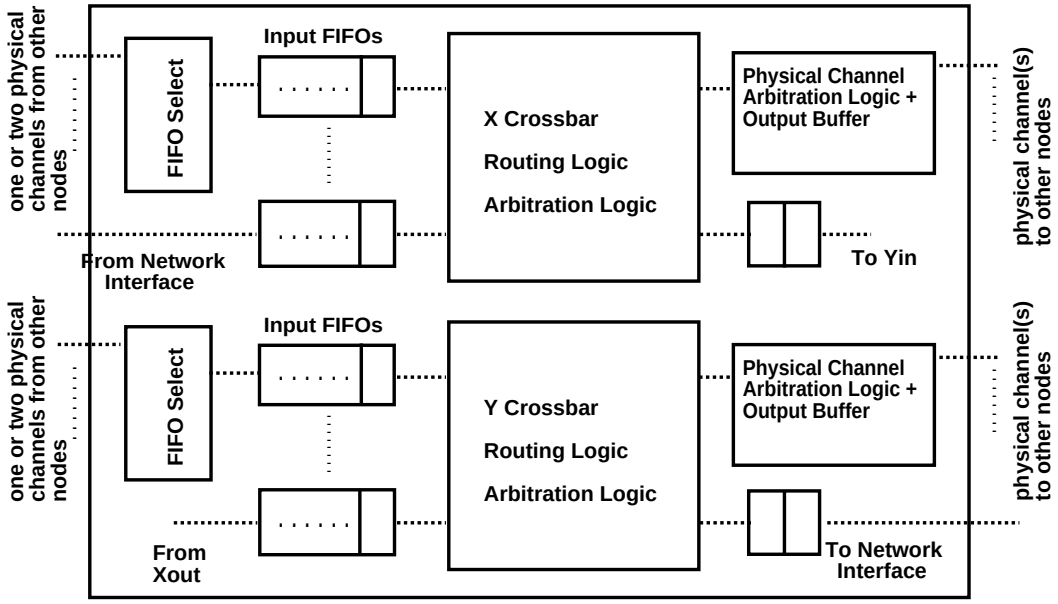

Figure 1: Canonical organization for the router switches evaluated in this study. The crossbars are cascaded.

We use as our canonical switch the familiar design shown in Figure 1, more detail is given

in the next section. [1] It has input FIFOs and output buffers with crossbars between them; a single physical channel per direction per dimension; and circuitry for routing (channel selection), lane selection, and physical channel arbitration among lanes. Since the routing is deterministic, source addressing is used to simplify routing. Lane selection uses a FIFO policy while physical channel arbitration uses a random policy; these policies were selected after extensive experiments showed them to be generally better in performance to the usual alternatives in addition to lending themselves to efficient implementations. The internode datapath is 32 bits with 32 bit paths for unidirectional networks and 16 bit paths for bidirectional networks. Varying datapath size affects all the designs studied here similarly; the decision for the single chip network will certainly depend much more on global resource allocation than on any of those variations. The networks are assumed to be 16 by 16, although we have confirmed that the results presented here do not differ significantly with network size.

There are two common crossbar configurations: a cascaded version with the output of the X crossbar being fed into the input of the Y crossbar and a single crossbar version with direct connections among most lanes. For two dimensional topologies, we have found the different configurations to be virtually a wash with the cascaded version allowing for a small reduction in operating frequency while causing a slight increase in the average packet latency in cycles. We present the cascaded version here.

The crossbars are assumed to have ports for all lanes and virtual channels. This is a reasonable assumption for the configurations we test: the fact that we find the configurations requiring large crossbars not very promising makes the issue moot.

With respect to the FIFOs and buffers, for hardware simplicity we again follow the standard convention of buffering the output. There is a small difference between the way WH and VCT outputs are handled which is explained below. When comparing buffers among designs, we sum the sizes of the output/input pairs. The FIFOs themselves are hybrid register/SRAM designs as explained in the next section.

We use input rather than output buffering because we found that input buffering is clearly superior in this domain. This somewhat anomolous result is again a consequence of technological constraints. Although output buffering has a clear advantage in congestion alleviation, it requires more complex routing. We found that this leads to at least a 20% increase in operating frequency in our designs [17]. Alternatively, we could modify our switches to route in three cycles rather than two. However, unlike in cabinet-sized parallel computers, here routing time dominates time-of-flight (2 or 3 to 1) rather than the other way around (e.g. 2 or 3 to 10). Either design change leads to a performance reduction that in not compensated for by the decrease in congestion.

We investigate three sharing modes for WH switching which together comprise a superset of most of the well-known techniques such as DAMQ [11]. These are: (i) sharing among lanes logically grouped for deadlock prevention, (ii) sharing among all the lanes associated with a physical channel, and (iii) sharing among all lanes in the switch.

[1] A note on the terminology: we avoid the use of the term *virtual channels*, since we feel that this unnecessarily infers a particular relationship with a physical channel, using instead the term *lanes*.

The parameters we vary are WH versus VCT switching, whether the topology is a mesh or torus, the number of lanes per channel, whether the packets can be sent only in single direction per dimension or in both, the method of virtual channel selection, the VCT deadlock prevention method, the size of the input FIFOs, and the type of buffer sharing. Since the output/input-FIFO pairs can be viewed as single buffers there is no need to vary both. Before giving the details of the configurations we first review issues in deadlock prevention and channel selection.

2.2 Deadlock Prevention

We begin by defining the switching modes:

Wormhole Routing – Packets are divided into flits which are transmitted contiguously. When the head of the packet is blocked, the flits remain in place. In practice, space for more than one flit per node is required. Two is usually the minimum needed to handle handshaking and communication of 'blocked-ness' back through the worm so as to prevent flits from being overwritten.

Virtual Cut-Through Routing – Packets are divided into flits and travel through the network as in WH routing. When blocked, however, the entire packet is queued as in packet-switched routing.

In practice WH routing networks usually have larger buffers than two. The term *buffered wormhole routing* was used to describe the IBM SP2 network in which a node can buffer several good-sized packets, but which provides no guarantee that an entire packet will be able to reside there.

In the simplest case, VCT implies that any packet arriving at a node can be queued there in its entirety. In parallel computers it is much more likely that the queue will be bounded to a fixed number of packets. We propose the term *virtual cut-through with limited buffers* to refer to networks with the following property: A blocked packet is guaranteed to be buffered in a single node in its entirety, but a packet is not guaranteed to be blocked only because a channel is unavailable.

Deadlock can occur when there is the possibility of a circular request for resources. For WH routing on torus networks, deadlock because of circular path requests is a well-known problem. With DOR, the simplest method for preventing deadlock is to partition the lanes associated with a physical channel (with at least two lanes being required) and statically allocating each packet to a particular set of lanes depending on its source and destination. To maintain a consistent nomenclature, we use the term *lane-set* to refer to a set of lanes in one of the partitions created to prevent deadlock. For mesh WH networks using DOR, there is no circular dependency and deadlock is not a problem.

2.3 Lane Selection

In an early work on deadlock prevention, Dally proposed the following selection algorithm: all packets that can proceed to their destination (in the dimension) without wraparound use lane 0 while the others use lane 1 until the wraparound point at which time they transfer to lane 0 to complete the route [9]. Bolding noticed that this algorithm leads to an extreme imbalance in buffer usage [2]. Scott and Thorson addressed this problem for bidirectional torus networks with the introduction of datelines and off-line lane selection [19]. We have extended this idea to unidirectional torus networks.

The method is a straightforward extension of the bidirectional technique with the exception that no pair of datelines exists such that all packets cross at most one dateline. We address this problem by requiring packets that cross both datelines to switch lanes once. This is not as elegant a solution as is available in the bidirectional case, but provides for a substantial improvement in load balance between lanes. The initial load (im)balance is shown in Table 1 and the load (im)balance after the application of the T3D technique is shown in Table 2. The resulting performance improvement is shown below.

Virtual	Node								Average	Max
Channel	0	1	2	3	4	5	6	7	Imbalance	Imbalance
0	0	0	1	3	6	8	15	21		
1	28	28	27	25	22	18	13	7		
Imbalance	1.0	1.0	.93	.79	.57	.46	.07	.5	.665	1.0

Table 1: Load imbalance in the standard virtual channel selection algorithm for the unidirectional wormhole torus. VC0 is the wrap-around channel and VC1 the direct. The counts indicate the number of source-destination paths that go through the particular channel in each node.

Virtual	Node								Average	Max
Channel	0	1	2	3	4	5	6	7	Imbalance	Imbalance
0	7	9	13	16	21	19	15	12		
1	21	19	15	12	7	9	13	16		
Imbalance	.50	.26	.07	.14	.50	.26	.07	.14	.24	.50

Table 2: Load imbalance in the T3D-inspired virtual channel selection algorithm for the unidirectional wormhole torus. VC0 has node 0 as a dateline and VC1 has node 4. The counts indicate the number of source-destination paths that go through the particular channel in each node.

We have also made one change to the basic T3D technique for bidirectional wormhole routing: we perform interdimension lane selection dynamically based on availability. Since

the inter-dimensional channels do not form a ring, this does not alter the deadlock properties of the network. It does, however, improve network capacity by 3 to 5%.

2.4 Differences Between Wormhole and Virtual Cut-Through

Since we assume small fixed length packets and DOR, the differences between WH and VCT networks are small but very significant. In particular, in a VCT network, once a header has been transmitted to the next input FIFO there is a guarantee that there will always be a space for all succeeding flits in the packet. In a WH torus network: (i) deadlock must be prevented by partitioning lanes into lane-sets and (ii) the lane selection policy should balance buffer usage as much as possible.

It is important to note that almost everything else can remain identical. In particular, nothing prevents the use of multiple *lanes* per channel in VCT networks to improve performance, although this is one of the issues not discussed in previous comparisons between WH and VCT switching [18, 10].

If we assume similar buffer requirements for networks of either switching mode (an assumption we will show is very reasonable) and a simple packet header counting mechanism for VCT, then there are two principal differences. These are i) the lane partition requirement of WH tori and the concommitant increase in switching complexity and ii) the guarantee of flit space in VCT that allows us to group the physical channel arbitration with the lane arbitration.

Channel Selection
Since the physical channel bandwidth is not affected by the switching mode, the question arises whether the freedom from lane partitioning is actually an advantage: after all, lanes have been found to be effective for flow control [6], especially when used with an optimized load balancing strategy [19]. We show that the answer is often yes, and for two reasons:

1) Not requiring lane partitioning means that particularly low cost switches can be built for VCT networks for which there is no WH equivalent.

2) For the same total number of buffers, VCT switching allows for more *dynamic* load balancing among buffers. For example, assume pairs of VCT and WH switches with equal numbers of lanes per dimension per direction. One such pair would be i) a bidirectional WH torus with two lanes per direction per dimension and one lane per lane-set and ii) a bidirectional VCT torus with two lanes per direction per dimension but no lane partitioning. In the WH case, lane selection is static as determined by the deadlock prevention policy while in the VCT case it is dynamic and can be load dependent.

Combined Arbitration
Since, in VCT, a packet transfer can only be initiated if there is a guarantee of space for the entire packet in the next node, and because we assume synchronous switches, there is no need for internode flow checking beyond that for the header flit.

3 Implementation Sketch

A critical problem in network research is leveling the playing field among disparate designs by accounting for the consequences of hardware implementation. To build, or even lay out, each candidate design is impractical. In this study we have two unusual advantages: in our restricted space, much of the routing and arbitration logic is similar across all of the designs and many of the components in communication switches are easy to parameterize. We have synthesized all of the switches described here using Synopsys tools and a .18 micron LSI Logic cell-based technology G10-p [16] to derive their operating frequencies and layout areas.

Basics
We assume that the network runs on a single clock. This assumption is reasonable in the short term; longer term we may need to model globally asynchronous/locally synchronous signaling with slightly more complex interaction between nodes. The switches have 16 bit datapaths for the bidirectional and 32 bit datapaths for the unidirectional. Two cycles are required for a header to traverse from input to output: one for arbitration and one for transfer. Once a path has been opened, other flits only need the transfer cycle to advance from input to output. This asymmetry does not alter the 'time-of-flight' for the packet but does halve the injection and ejection times, which become important when the packet size is a significant fraction of the network diameter.

The arbitration cycle and transfer cycle both have on their critical paths flip-flop clock to output delay (T_{CO}), flip-flop setup time (T_{setup}), and clock parameters (T_{clk}). The arbitration cycle also has on its critical path the address decode and path request delay (T_{AD}) and the path arbitration delay (T_{PA}). The transfer cycle also has on its critical path the crossbar delay (T_{CB}) and the lane multiplex delay (T_{LM}). One of the differences between WH and VCT switches is the way the output physical channel flow control (T_{FC}) is handled. These delays are now described in turn.

Address Decode and Path Request
Each crossbar input has an address decode and path request module. We assume static routing with up to two extra bits to specify the virtual channel, if needed. The delay for the address check is constant for all node types except for a very slight increase if the selection bits need to be decoded as well.

Path Arbitration
The path arbitration module (associated with each crossbar output) determines the output lane. The logic considers whether the lane is in use and also what is happening in the next input FIFO. Priority is given to buffers that not only have space but which also are not blocked. The outcome of the arbitration is returned to the address decode and path request module.

Crossbar
The crossbars are implemented as multiplexors without hierarchical switching.

FIFOs
The FIFOs are based on a hybrid register/SRAM design. The SRAM is half as fast but

approximately five times smaller per bit than the register. In the basic non-shared design, the register part of the FIFO is five flits big with the rest of the space being taken with SRAM. The paths between registers and SRAM are two flits wide to hide the latency differential. There is a bypass to account for the case where the packet becomes unblocked just as it is being written to the SRAM.

The two cases where buffers are shared among channels and virtual channels are straightforward since the physical channel can only supply one flit per cycle. The SRAM is now simply shared among all of the lanes. The register part of each FIFO must be slightly larger than previously, however, to account for input contention. Linked lists keep track of which flits go with which lane.

The centralized queue case is substantially more complex since up to four flits can enter and exit every cycle. The SRAM datapaths are therefore 8 flits wide and the register portion of each FIFO is correspondingly larger as well. Extra circuitry is also needed to deal with contention.

Lane Multiplexor, Physical Channel, and Flow Control
We use the term flow control to denote the internode handshaking that determines whether a flit can be sent to the next node's input. In VCT, flits are always guaranteed space in the node ahead once packet transmission has begun and only need to deal with flow control during the arbitration cycle where that guarantee is given. In VCT, therefore, T_{FC} is on a path that is parallel to and shorter than the arbitration path. The multiplexor which determines which lane will have a flit transferred to the physical channel can be placed before a single output buffer. The lane selection inputs to the multiplexor (random bits) are generated continuously and in parallel to the data transfer and so are off of the critical path. In the WH case, there is no guarantee of space in the next node and so the lane multiplexor must come after the output buffers. The T_{FC} can also be placed on a parallel path, however, and adds only slightly to the delay.

	Unidirectional				Bidirectional				
Total Lanes	whSt	whT3D	vctStr	vctRel	meshwh	whSt	whT3D	vctStr	vctRel
1			1.54	1.54					
2	1.85	1.85	1.85	1.85	1.63			1.63	1.63
4	2.18	2.18	2.23	2.23	1.96	1.96	1.96	1.96	1.96
8	2.64	2.64			2.48	2.48	2.48	2.48	2.48
16						2.85	2.85		

Table 3: Shown are the cycle times in nanoseconds for the switch designs synthesized using an LSI Logic .18 micron cell-based technology. Since buffer size does not significantly affect the timing, configurations differing only in that aspect are not shown. Note that the timing differences between the corresponding wormhole and virtual cut-through switches are negligible.

The timing results are shown in Table 3 and were obtained from Synopsys critical path measurements. Since the circuits are relatively simple, the times should match those of the completed devices fairly closely; they are certainly valid for comparison purposes. There are

two key results: (i) doubling the number of buffers slows down the clock by roughly 20% and (ii) the hardware differences between VCT and WH switching result in only small changes in timing. Note that because the FIFO size has virtually no impact on operating frequency we do not give separate times for the various buffer configurations. The way sharing is implemented, there is no slowdown for sharing buffers among virtual channels and virtual channel pairs. For sharing among all lanes in the node, there is some slowdown due to output contention, but not enough to affect the overall results.

The area results are shown in Table 4. This measure is dominated by the FIFOs with the majority of the area of the larger designs being devoted to them. If register-based rather than hybrid FIFOs had been used, however, then the sizes would have been up to *five times* greater. The extent of the register size problem can be seen by looking at the sizes of the sharing configurations which require substantial register support to hide the latency of the sharing overhead.

lanes per dimension	flits per lane	Unidirectional		Bidirectional					
		uniwh	univct	meshwh	biwh no share	biwh share vc	biwh share ch	biwh share all	bivct
1	6								
1	12		1.09						
1	24		1.20						
1	48		1.40						
2	6			1.17					
2	12	1.84	2.17	1.30					1.30
2	24	2.01	2.37	1.57					1.57
2	48	2.35	2.77	2.11					2.11
4	6			2.38	2.07				
4	12	3.65	4.44	2.65	2.32		3.83	10.55	2.65
4	24	3.98	4.84	3.19	2.81		4.32	11.41	3.19
4	48	4.66	5.65	4.27	3.80		5.31	13.14	4.27
8	6			5.06	4.23				
8	12	7.50		5.60	4.72	7.75	8.66	26.03	5.60
8	24	8.18		6.68	5.71	8.73	9.64	27.77	6.68
8	48	9.53		8.84	7.69	9.85	10.76	29.71	8.84
16	6				8.36				
16	12				9.26	16.41	20.31	63.41	
16	24				11.06	17.53	21.44	65.38	
16	48				14.65	19.50	23.40	68.82	

Table 4: Shown are the areas of the switch designs in terms of 10000's of cells in an LSI Logic .18 micron cell-based technology.

4 Results

4.1 Methods

We use a register transfer level simulator to measure capacity and latency. Since our designs are synchronous, the simulator can be cycle-driven and validation with the hardware model is simple. We assume an internode routing time of one cycle as per our previous discussion.

We use three communication patterns: random, hot-spot, and random-near. For the random load, all destinations are equally probable. For the hot-spot load, we use a similar scheme as described in [3]: four destinations are four times as likely as the others. For the near-random load, the coordinates of the destination are chosen independently for each dimension: likelihood of a destination is inversely proportional to its distance from the source.

The load is presented in terms of the expected number of flits injected per node per cycle. We prefer this measure to that of fraction of overall capacity in that it forces separate evaluation for each communication pattern.

We use two packet sizes: 6 flits and 24 flits. These sizes were chosen to represent the transfer of a word and a small cache line transfer, respectively. Also, the smaller packets typically span a small number of nodes in transit while the larger packets span the entire path through the network. Together they offer two qualitatively different workloads.

Our primary choice of performance measure is network load capacity. One reason for this is its intrinsic importance. The other is the observation, repeated numerous times, that the switching mode, buffer size, number of lanes, and other parameters which are the objects of this study all have only a small effect on latency until saturation is approached [10] and then the effect is quite predictable. The latency/load graph in Figure 2 depicts this effect. There are three 'bundles' of series: one each for unidirectional WH, bidirectional mesh WH, and bidirectional torus WH. The bundles are formed around configurations with matching unloaded latency. The bidirectional and unidirectional VCT series, if shown, would be superimposed on their WH counterparts. Within each bundle, the capacity measure, which indicates where the series becomes vertical, is therefore sufficient to characterize the particular series.

A run consists of a single combination of network, load, communication pattern, and packet size. A run terminates either after 80,000 cycles or when the network goes into saturation. The first 50,000 cycles are used for transient removal; thereafter latencies are recorded. Generally, steady-state is reached after only a few thousand cycles: the extra cycles are used to increase the sensitivity of the saturation point measurement by making it more likely that a network running even slightly above capacity will saturate.

Saturation is determined to have taken place if the injector queue of any node overflows its capacity of 200 flits. This criterion is justified because it can only be caused by prolonged backpressure which is very unlikely to be caused by a local hotspot created in a load significantly less than the saturation point.

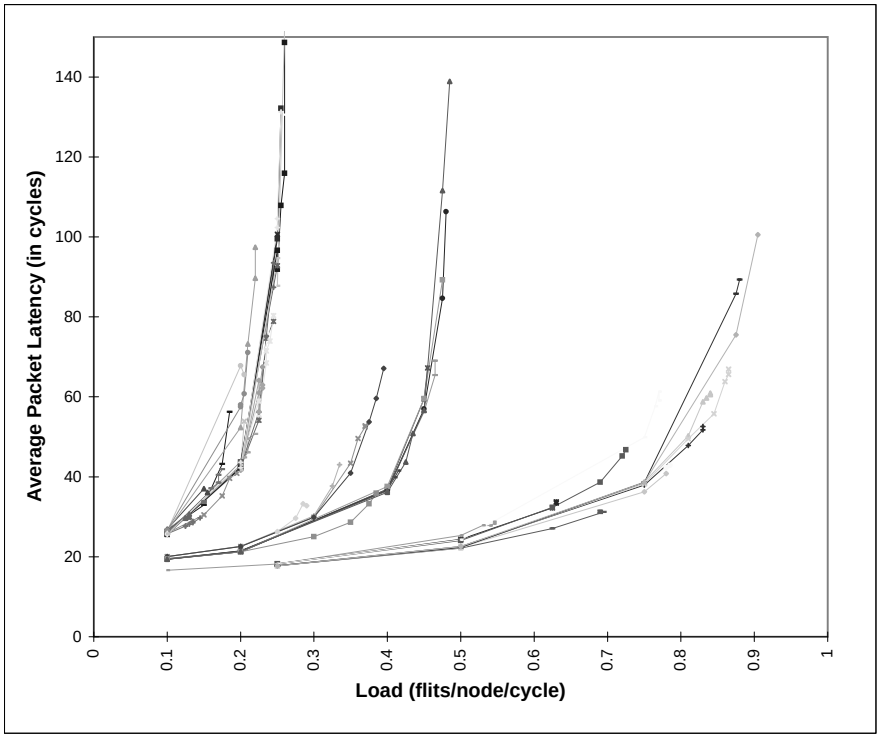

Figure 2: Shown is a graph of latency versus applied load for a number of switch designs with the random communication pattern, a packet size of 5 flits, and an 8 by 8 network. The three "bundles" of series correspond to the unidirectional torus wormhole, the mesh wormhole, and the bidirectional torus wormhole configurations, respectively.

Each combination of network, communication pattern, and packet size was simulated with respect to a number of loads (typically 12-15) which converged about the saturation point. Thus most of the latency/load points recorded are at and beyond the knees of the latency/load graphs where maximum sensitivity is required. The maximum load which does not cause saturation is determined to be the capacity of the network. The standard deviation on the capacity measure was found to be .011 flits per node per cycle.

4.2 Experiments and Observations

Size of buffers
The effect of varying buffer size on performance is shown in Figure 3. Increasing the buffer size has the predictably positive effect on capacity. The improvement starts to tail off significantly after about 200 flits per node in almost all cases. Especially for packets larger than 24 flits, bigger buffers may be called for. However, see also the later discussion: when chip area is accounted for such statements are no longer so obvious.

Number of lanes per channel
The effect of varying the number of lanes per channel is shown in Figures 4 and 5. In Figure 4—where we measure capacity in cycles—we see that, in most cases, the improvement in network capacity when the number of lanes per channel is increased from 1 to 2 was significant, while the improvement from 2 to 4 was slight, and, in some cases, negative. In

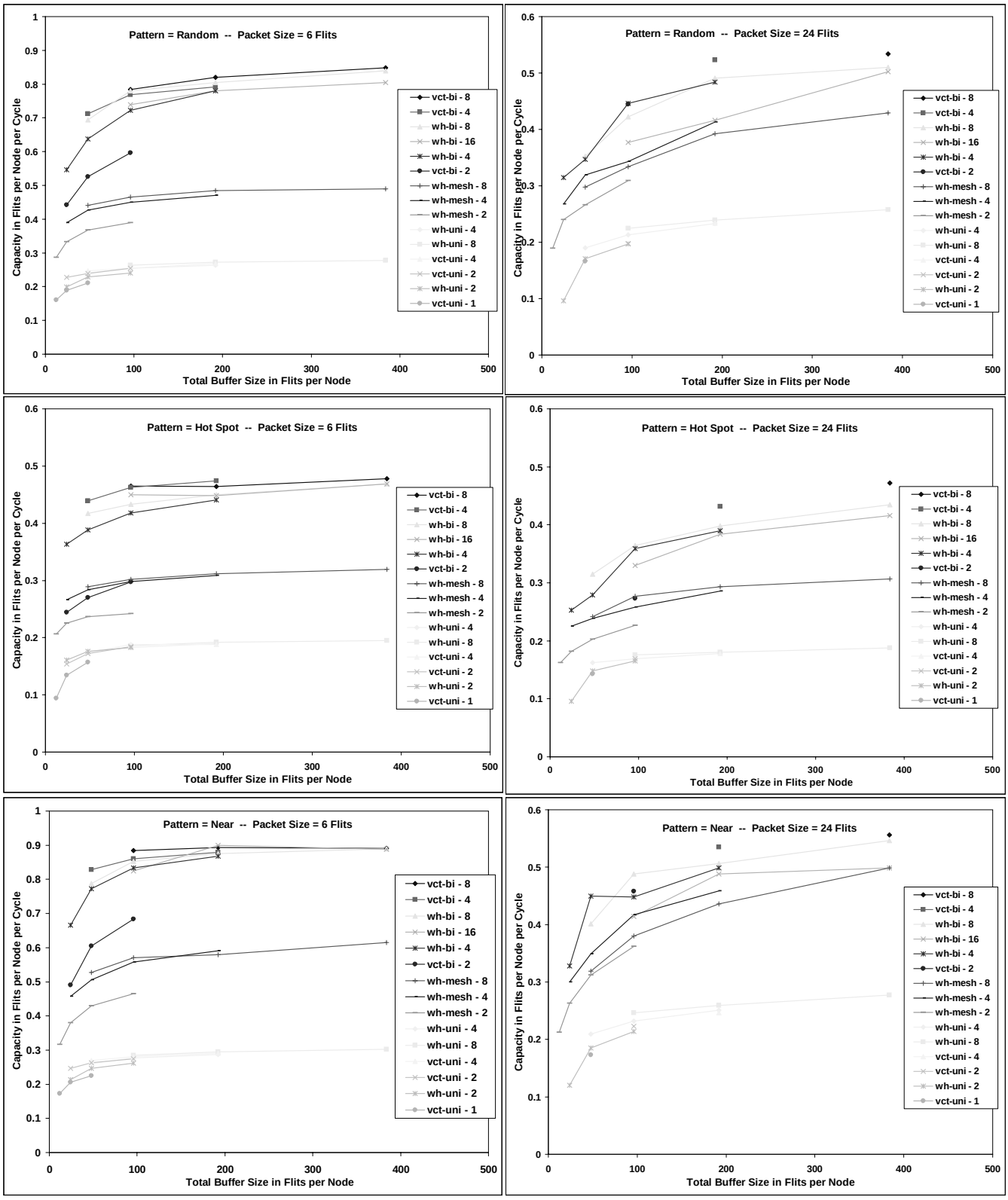

Figure 3: Shown are capacity versus buffer size graphs for various workloads, switching modes, topologies, and numbers of lanes. The legends have the format switching mode–topology–lanes per dimension.

Figure 5 we account for the variation in operating frequency with differing numbers of lanes per channel. Here the optimal number of lanes per channel is starkly apparent: exactly 2. Note that this is in contrast to studies that did not account for operating frequency where only *the benefit* decreases with number of lanes per channel, not the performance itself.

Topology
For reasons discussed above, the only latency results presented are those shown in Figure 2. With no congestion, latencies of the three topologies are related to the average path length: k for the unidirectional torus, about $2k/3$ for the mesh, and $1/2k$ for the bidirectional torus. However, since the injection time is a significant part of the latency (2 times the packet size or 10 in Figure 2) and identical for all three topologies, the average path length may not be the most critical aspect of changing topology for small networks. Rather, it is the effect on capacity.

There are three related reasons why topology affects capacity: (i) differences in total internode bandwidth (if the datapath is constant), (ii) the shorter path length gets packets out more quickly and so they interfere less with each other, and (iii) the intrinsic variation in quality among the buffer load balancing opportunities. These latency/bandwidth effects can be seen in Figure 2 as follows. The level of the 'flat parts' of each bundle are due to differences in average path length while the locations of the knees are given by differences in the capacity-related properties.

The effect of varying the topology is shown in Figures 3, 4 and particularly in 6. Clearly the unidirectional and mesh networks are not cost-effective except in the very simplest designs which have no bidirectional equivalent: e.g. VCT with one lane per channel, which equates to one lane per dimension. The rationale in trying out mesh and unidirectional tori, in addition to bidirectional tori, was their simplicity and that the doubling of flits per cycle due to the wider datapath (in the unidirectional case) might make up for the longer path length. This was obviously not borne out.

Accounting for area and operating frequency
Figure 6 contains the plots of network capacity in flits per node per nanosecond versus switch area in cells. The cost-effective designs are those toward the left and top of each graph. Some of the best designs for each combination of communication pattern and packet size are labeled from 1 to 10 and identified in Table 5. Figure 6 combines three sets of results: (i) that frequency is reduced by roughly 20% for every doubling in the number of buffers, (ii) that the buffer size is a large fraction of the switch area, and (iii) the capacity performance measures. Putting these together it is clear that a designer selecting a network for a multicomputer-on-a-chip will only consider candidates from the left edges of the graphs.

Buffer size again. The upper left graph in Figure 6 shows four connected series. These show designs that differ only in buffer size; the others are not shown so as not to obscure other points detailed below. The asymptotic behavior from Figure 3 should be apparent however: the series from that figure representing designs with higher numbers of lanes have been shifted to the right and down. The decision on the benefits of larger buffers is still not completely clear-cut, but doubling the buffer size only appears to yield a few percent increase in capacity.

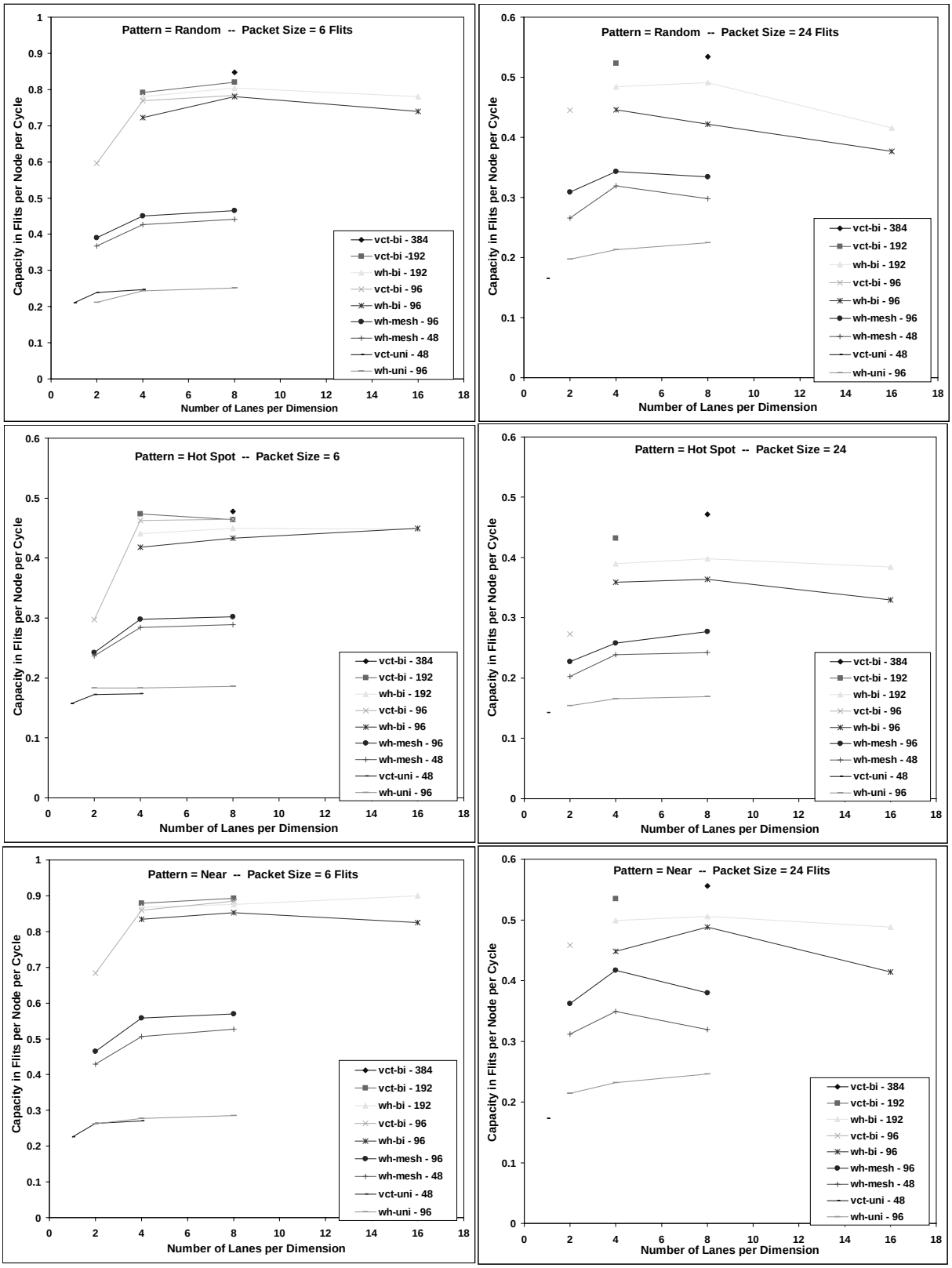

Figure 4: Shown are capacity versus lanes per dimension graphs for various workloads, switching modes, topologies, and buffer sizes for the entire node in flits. The legends have the format switching mode–topology–flits per node.

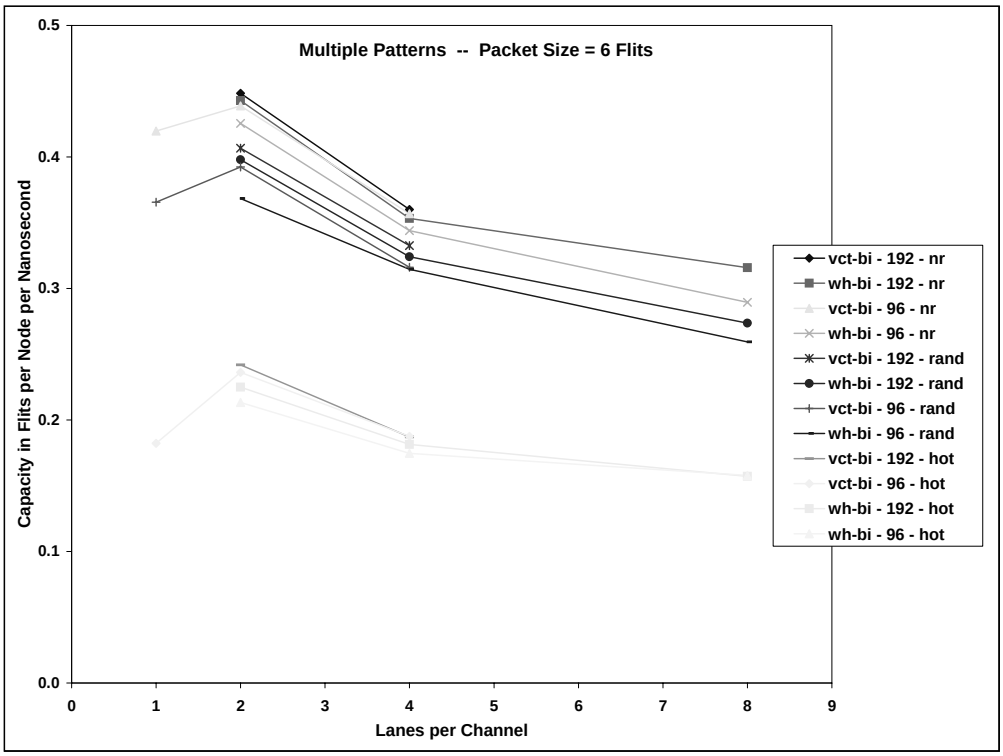

Figure 5: Shown are capacity versus lanes-per-dimension graphs for various workloads, switching modes, and buffer sizes for the entire node in flits. The legends have the format switching mode–flits per node–routing pattern. Note that capacity here is in nanoseconds rather than cycles.

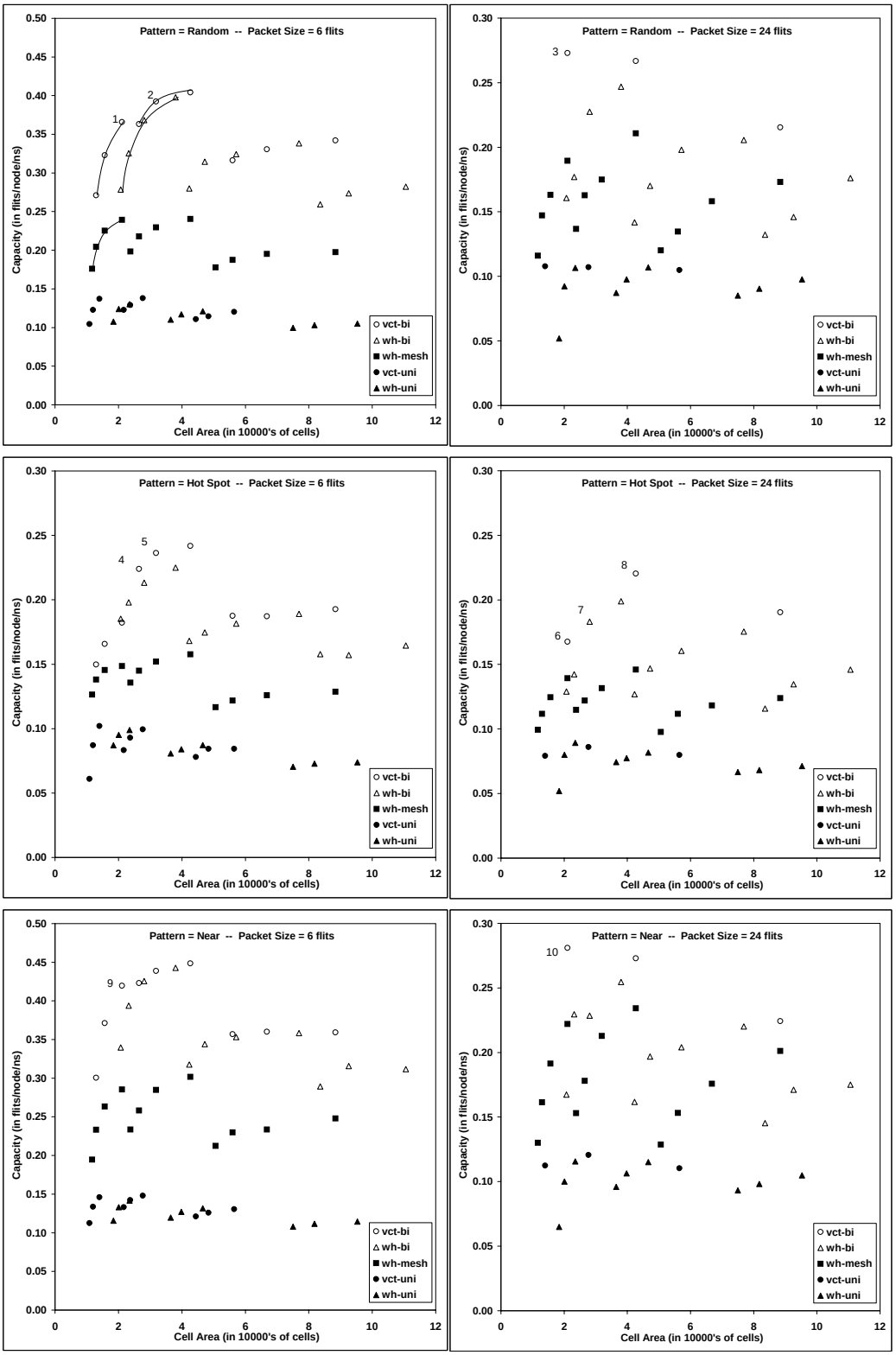

Figure 6: Shown are the capacity versus area graphs for many of the configurations described. Note that capacity is with respect to nanoseconds rather than cycles. The best designs are on the left and top with the points on the left/top frontier denoting suitable choices for implementation distinguished only by the area/capacity trade-off. Ten are labeled and described in Table 5.

Wormhole versus virtual cut-through switching. The effect of varying the switching mode is shown particularly in Figure 6 and Table 5, although it can be seen in the previous figures as well.

Recall that the basic difference in VCT and WH implementations is that VCT has additional flexibility in channel selection due to relaxed deadlock constraints. This manifests itself in several ways. First, if all other parameters are the same, the VCT switch will be bigger. This is because the additional routing flexibility requires a denser crossbar. Second, VCT networks can be created with a single buffer per node per ring. Such a WH network can only be constructed by either removing the wrap-around connections or adding buffers. Therefore cost-effective VCT networks exist trivially in niches where WH networks do not. Third, the additional flexibility results in higher capacity. For example, it is possible that a VCT switch can route a packet to an empty channels that the WH switch cannot because of deadlock-related constraints. These effects are all shown in Figure 6 and Table 5.

For this comparison, it only makes sense to examine the most cost-effective designs. Qualitatively, Table 5 shows the dominance of VCT. The actual quantitative benefit varies with workload, but VCT appears to have a capacity advantage of from 5% to 15% in most cases for comparably sized switches. An exception is the hot spot load with 24 flit packets where the cost-effective "frontier" has 2 VCT and 2 WH designs. However, VCT design 8 has a 12% capacity advantage over the WH alternative with a much smaller difference in area. As the number of lanes increases and the initial static selection becomes less important, WH and VCT performance converges. However, these are also the less cost-effective designs.

Label	Workload		Switch Configuration			
	Pattern	packet size	mode	topology	lanes/dimension	buffer size/lane
1	Random	6	VCT	bi-torus	2	48
2	Random	6	VCT	bi-torus	4	24
3	Random	24	VCT	bi-torus	2	48
4	Hot Spot	6	VCT	bi-torus	4	12
5	Hot Spot	6	VCT	bi-torus	4	24
6	Hot Spot	24	VCT	bi-torus	2	48
7	Hot Spot	24	WH	bi-torus	4	24
8	Hot Spot	24	VCT	bi-torus	4	48
9	Near	6	VCT	bi-torus	2	48
10	Near	24	VCT	bi-torus	2	48

Table 5: Most cost-effective designs in Figure 6.

Buffer Sharing

Figure 7 shows the cost-effectiveness of various buffer sharing mechanisms for bidirectional WH networks. Table 6 identifies the best alternatives. Sharing has virtually no benefit for random loads but some benefit for hot spot loads. This observation is similar to that of [3] although our sharing schemes retain the deterministic routing algorithm.

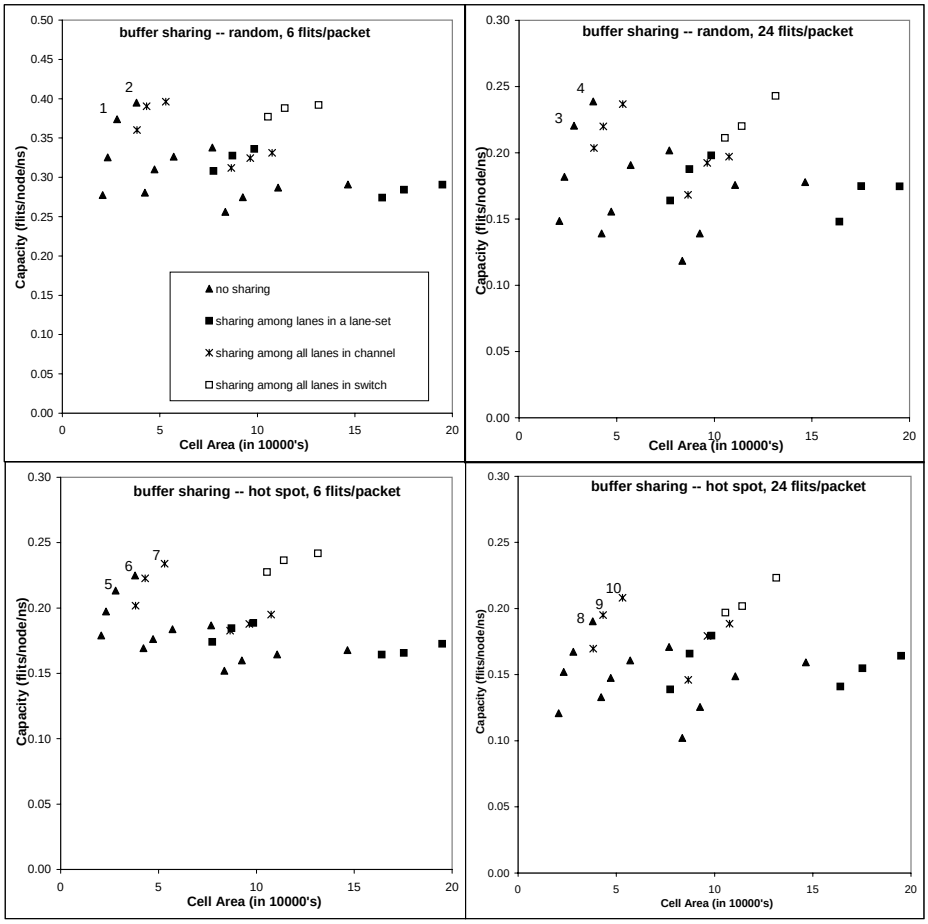

Figure 7: Shown are capacity versus area graphs for bidirectional wormhole routing networks with various sharing configurations. The best designs are on the left and top with the points on the left/top frontier denoting suitable choices for implementation distinguished only by the area/capacity trade-off. These labeled designs are described in Table 6.

The sharing designs that prove cost-effective under hot spot loads are those with the minimum complement of two lanes per channel and sharing restricted to precisely those two lanes. This scheme has relatively low area overhead and no added latency.

Label	Workload		Switch Configuration		
	Pattern	packet size	sharing type	lanes/dimension	buffer size/lane
1	Random	6	No Sharing	4	24
2	Random	6	No Sharing	4	48
3	Random	24	No Sharing	4	24
4	Random	24	No Sharing	4	48
5	Hot Spot	6	No Sharing	4	24
6	Hot Spot	6	No Sharing	4	48
7	Hot Spot	6	Channel	4	48
8	Hot Spot	24	No Sharing	4	48
9	Hot Spot	24	Channel	4	24
10	Hot Spot	24	Channel	4	48

Table 6: Cost-effective designs in Figure 7.

5 Sample of Previous Work

A sample of the extensive previous work in this area is now presented. For an excellent bibliography please see [10]. Emphasis is placed on work built on directly in this article.

The seminal papers for virtual cut-through and wormhole routing are by Kermani and Kleinrock [14] and by Dally and Seitz [8, 9], respectively. The virtual channel selection algorithm of the latter Dally and Seitz article is what we used in our basic virtual channel scheme. Dally also proposed partitioning channels into lanes for the purpose of improving performance [6]. Bolding noticed the extreme load imbalance in the standard virtual channel selection strategy [2] and a solution was devised by Scott and Thorson [19]. Rexford did extensive work comparing cut-through and wormhole routing [18] which has been augmented in [10].

Among the parallel processors using wormhole routing are the J-Machine [7], the Cray T3D [19], the Intel Teraflops router [4], and the SGI Spider [12]. A network that uses VCT is the Chaos router [3].

Innumerable RTL simulators have been created for these and other studies; however comparatively few studies have accounted for hardware implementation. Of particular note are those by Chien [5] and by Aoyama and Chien [1].

6 Conclusions and Work in Progress

In this work we have endeavored to explore exhaustively the design space of networks for multicomputers-on-a-chip including issues in switching, lane selection, buffer size, topology, buffer sharing, and hardware implementation. Preliminary experimentation allowed us to make basic assumptions restricting the topology, routing algorithm, and queuing regimen. The expected domain restricted our packet size to be relatively small.

Of particular interest is that VCT was examined and found to be both more and less like WH switching than previously described: more in that virtually all of the critical hardware including multi-lane channels can be identical to that of WH switches; less in that VCT is open to a different lane selection paradigm. It has been stated elsewhere that virtual cut-through versus wormhole routing is a tradeoff between buffering and congestion. We suggest that for small packets and equal buffering it is really mostly the *difference between static and dynamic lane selection.*

The most significant result is that we have demonstrated the inseparability of global performance (e.g. network capacity with respect to a routing pattern) from local consequence (e.g. the operating frequency and switch area). This was shown in particular with respect to the determination of the appropriate number of lanes per channel.

Acknowledgments
We wish to thank Jade Cravy for his help in generating the graphs and Shivshankar Sanikop for his work on an earlier version of the RTL simulator.

References

[1] Aoyama, K., and Chien, A. A. The cost of adaptivity and virtual lanes in a wormhole router. *Journal of VLSI Design* (1994).

[2] Bolding, K. Non-uniformities introduced by virtual channel deadlock prevention. Tech. Rep. UW-CSE-92-07-07, Dept. of Comp. Sci. and Eng., U. of Washington, Seattle, WA 98195, 1992.

[3] Bolding, K., Fulgham, M., and Snyder, L. The case for Chaotic Adaptive Routing. *IEEE Trans. on Computers C-46*, 12 (1997), 1281–1292.

[4] Carbonaro, J., and Verhoorn, F. Cavallino: The Teraflops router and NIC. In *Proc. of Hot Interconnects Symposium IV* (1996).

[5] Chien, A. A. A cost and speed model for k-ary n-cube wormhole routers. In *Proc. Hot Interconnects '93* (1993).

[6] Dally, W. J. Virtual channel flow control. *IEEE Trans. on Parallel and Distributed Systems 3*, 2 (1992), 194–205.

[7] Dally, W. J., and et al. The Message-Driven Processor: A multicomputer processing node with efficient mechanisms. *IEEE Micro 12*, 2 (1994), 194–205.

[8] Dally, W. J., and Seitz, C. L. The Torus Routing Chip. *Distributed Computing 1* (1986), 187–196.

[9] Dally, W. J., and Seitz, C. L. Deadlock free routing in multiprocessor interconnection networks. *IEEE Trans. on Computers C-36*, 5 (1987).

[10] Duato, J., Yalamanchili, S., and Ni, L. *Interconnection Networks: An Engineering Approach*. IEEE Computer Society Press, Los Alamitos, CA, 1997.

[11] Frazier, G. L. *Buffering and Flow Control in Communication Switches*. PhD thesis, Department of Computer Science, University of California, Los Angeles, Los Angeles, CA, 1995.

[12] Galles, M. Scalable pipelined interconnect for distributed endpoint routing: The SPIDER chip. In *Proc. of Hot Interconnects Symposium IV* (1996).

[13] Ge, J. *Evaluating the Cost-Effectiveness of Adaptive Wormhole Routers*. PhD thesis, Department of Electrical and Computer Engineering, University of Houston, 2002.

[14] Kermani, P., and Kleinrock, L. Virtual cut-through: A new computer communication switching technique. *Computer Networks 3* (1979), 267–286.

[15] Leighton, F. T. Average case analysis of greedy routing algorithms on arrays. In *Proc. 2nd Symp. on Parallel Algorithms and Architectures* (1990), pp. 2–11.

[16] LSI Logic Corp. *G10-p Cell Based ASIC Products*. Milpitas, CA, 1997.

[17] Olin, K. High-performance embedded multicomputer networks. Master's thesis, Department of Electrical and Computer Engineering, University of Houston, 2000.

[18] Rexford, J. *Tailoring Router Architectures to Performance Requirements in Cut-Through Networks*. PhD thesis, Dept. of Comp. Sci. and Eng, U. of Mich., 1996.

[19] Scott, S., and Thorson, G. Optimized routing in the Cray T3D. In *Proc. of the Workshop on Parallel Computer Routing and Communication* (1994), pp. 281–294.