Memory Optimization in Single Chip Network Switch Fabrics [*]

David Whelihan, Herman Schmit
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh,PA 15213
{whelihan,herman}@ece.cmu.edu

ABSTRACT

Moving high bandwidth (10Gb/s+) network switches from the large scale, rack mount design space to the single chip design space requires a re-evaluation of the overall design requirements. In this paper, we explore the design space for these single chip devices by evaluating the ITRS. We find that unlike ten years ago when interconnect was scarce, the limiting factor in today's designs is on-chip memory. We then discuss an architectural technique for maximizing the effectiveness of queue memory in a single chip switch. Next, we show simulation results that indicate that a more than two order of magnitude improvement in dropped packet probability can be achieved by re-distributing memory and allowing sharing between the switch's ports. Finally, we evaluate the cost of the optimized architecture in terms of other on-chip resources.

Categories and Subject Descriptors

B.3.2 [**Harware**]: Design Styles

General Terms

Design, Performance

1. INTRODUCTION

This paper explores the architecture of single chip network switch fabrics. Single-chip network switches are less costly and consume less power than today's rack-level and board-level switches, which will mean higher bandwidth and profitability for network users and suppliers. The levels of integration achievable by modern process technologies significantly alter the design space for network switches, however. This paper first evaluates the new realities for switch design. We conclude that unlike a rack-level system, wires are abundant. The limited resource in a single-chip switch is memory, which is used for queuing data throughout the chip.

[*]This work was funded by the Pittsburgh Digital Greenhouse under contract EDT-99-1.

This work was inspired by Dally's exploration of topology in message-passing networks[1]. Since our research goal was the exploration of the design space for single chip network switches, we re-evaluated the resources available to an SOC switch designer, establishing which are critical for single a chip network switch now and in the next 7 years. We accomplish this by analyzing data provided by the International Technology Roadmap for Semiconductors(ITRS)[10] with the needs of a scalable, high bandwidth switch architecture in mind. We then describe an architectural technique by which a designer could optimize the efficiency of critical on-chip resources. Based on this, we present a parameterizable class of network switch fabrics and a simulator capable of evaluating instances of this class of fabrics. We then use this simulator to determine the optimal memory organization, using dropped packets caused by overflowing queues as an optimization metric. Finally, we show simulation data supporting our design choices.

2. CRITICAL RESOURCES

A network switch, in its most basic form, can be viewed as a large conglomeration of inputs and outputs. The inputs absorb data packets of either fixed length(as in ATM) or variable length(IP)[7], and send them to the proper output. Further, such a system would require that a memory be placed at each output port to buffer in the case where two or more input packets are switched to the same output. Figure 1 illustrates this

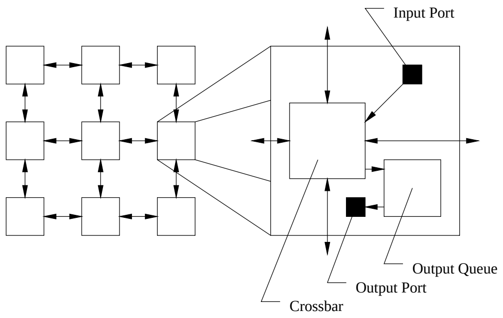


Figure 1: Array model of a single-chip network switch comprised of individual *nodes*

model, in which input and output blocks are grouped together with output queue memory and a switch-box unit. These *nodes* are then grouped in a grid with metal interconnect between them. This is consistent with earlier work[1] and due to its regularity supports scalability. For our chip resource analysis, we concentrate on three main areas: interconnect, memory, and I/O bandwidth.

2.1 I/O Bandwidth

The amount of on-chip bandwidth is irrelevant if the switch cannot be fed data from off chip. The number of I/O pads allocated to a particular port will depend on the incoming data rate, the maximum speed of an I/O Buffer, and the requirements of the signaling protocol used. As for the latter, we assumed Low Voltage Differential Signaling(LVDS)[6], in which each individual input or output requires two I/O pads. This was chosen because it is currently capable of GHz speeds, and it allowed us to be conservative in our IOB utilization estimates. Figure 2a shows the maximum number of nodes for ports running at both 2.4Gb/s(OC-48) and 10Gb/s(OC-192).[1] These values indicate the absolute bandwidth ceiling. Though this does not take into account control and strobe inputs, it is still true that by 2008 the IOB bandwidth will allow chips with over 100 bi-directional ports running at 10Gb/s[10].

2.2 Interconnect

The object of this particular analysis was to determine to a reasonable degree of accuracy the maximum *bi-section bandwidth* of a routing channel (in this case, 1mm was chosen). In this analysis, the first two metal layers are assumed to be used for local routing, and power and ground rails are routed on high level auxiliary layers. The intermediate layers are allocated evenly to north-south and east-west routing with the following rule:

$$N_{layers} = \left\lfloor \frac{N_{int} - 2}{2} \right\rfloor \tag{1}$$

where N_{int} is the total number of intermediate wiring layers predicted by the ITRS. Figure 2b shows the bandwidth ceiling for a 1mm bisection of a chip through 2008. We have also graphed two demand lines, calculated by multiplying the number of ports per year from Figure 2a by their respective line speeds. The demand lines show that a 1mm wide routing channel could transmit all of the required data bandwidth for such a chip. This means that we are IO limited and not bound by on-chip wiring bandwidth. This has been made possible by the substantial decrease in wire pitch, the increase of on-chip speeds (easily 2500 MHz by 2008), and the increase in the number of metal layers (9 intermediate layers by 2008).

2.3 Memory

Currently, due to advances in embedded DRAM technology, it is possible to efficiently integrate both DRAM and processing logic efficiently on the same die. In [11], a 1MB embedded DRAM with a footprint of $19mm^2$ is described. This device has a large line width(1024 bits) as well as a 3.7ns address access time, putting the per bit access time well within the limits for 10Gb/s port speeds. The DRAM cells, while about 2.5 times larger than a comparable cell made in a DRAM only process[14][15], still achieve roughly seven times the density of an SRAM cell.

Using this work as a baseline, to first order we can graph memory sizes for the maximum chip size predicted by the ITRS by extrapolating the cell scaling for both the DRAM and the hypothetical SRAM that is seven times larger. Figure 2c shows curves for both SRAM and DRAM on an $800mm^2$ die for four process nodes.[2] The cell size in future processes was extrapolated by simply scaling based on the shrinking feature size.

[1]The jump in 2008 is due to the IOB speed increasing such that it is almost an exact multiple of the data frequency.

[2]The ITRS specifies 800 mm^2 as the maximum chip size through 2008.

2.4 Analysis

While the absolute IOB bandwidth limit does provide a hard cap on the throughput of our device, it still allows for a high aggregate bandwidth (1Tb+). Since this is a hard technology barrier, it does not lend itself well to architectural improvement.

As for on-chip interconnect, it is apparent that the situation of ten years ago is no longer the case. Cross-chip metal interconnect running at well over 2 GHz will be in abundance. While it is true that this is only a first order approximation, it seems obvious that even with wires at greater than minimum pitch, and with wire speeds halved the bi-section bandwidth still outpaces the bandwidth capability of the IOBs.

We have determined that *memory* is the most critical chip resource today. We justify this by pointing out that commercial high bandwidth switches routinely have more than 1 MB per port of output queue memory[18]. Also, the worst case traffic at any one output port will increase with the number of input ports in the system, which in today's embedded DRAM technology would occupy the majority of an $800mm^2$ chip. The memory requirement for queueing under this assumption is plotted as a dashed line in figure 2c for all process nodes through 2008 [3]. This line was calculated by multiplying the number of possible ports from Figure 2a by 1MB. We see that while there is always enough DRAM availability, 1MB per port generally requires at least half of an $800mm^2$ chip. This problem is compounded by the fact that memory storage is required for other tasks on a switch:

- Keeping track of packet priority when service guarantees have been made.
- Keeping track of packet time-stamps to ensure that the oldest packet goes out first.
- Routing tables. Storage for upwards of 500k routes is not unreasonable in a high bandwidth switch.

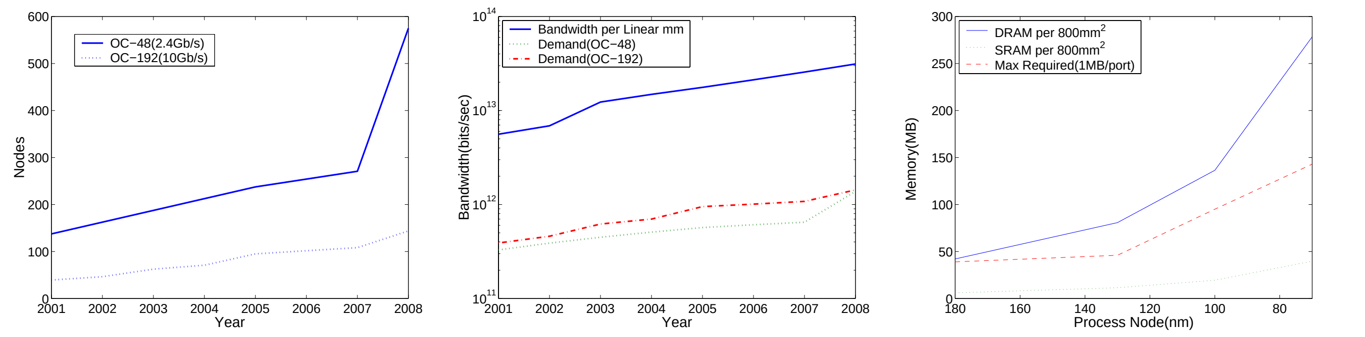
3. ARCHITECTURES

Within the last few years, several research teams have devised and fabricated single chip network switches and message routers [5][8][9][17]. Most of the previous work has used a shared buffer approach, in which a centralized block of memory is partitioned into multiple *output queues*. This is the most efficient use of memory, as a particular output does not truly have a fixed queue allocation (up to the limits of the physical memory). One could envision this as one output queue *borrowing* unused memory from other, less heavily utilized queues. An unfortunate side affect of this is that an output that fills the entire memory with its data at time *t* could starve the other outputs for queue memory at time *t+1* unless per port maximum allocations are enforced. Scalability is yet another problem that the shared buffer approach must contend with. As the number of Input/Output ports increases on a switch, so too does the chance of contention for the shared memory. One solution to this problem is to add more ports[4] to the shared memory, allowing multiple simultaneous accesses. Unfortunately, the area and power dissipation increase prohibitively when many simultaneous memory accesses are necessary.[12].

The other extreme involves the allocation of separate output queue memory to every output port. This has the benefit of

[3]2000:180nm,2002:130nm,2005:100nm,2008:70nm

[4]this is different from the definition of *port* previously discussed.



a. b. c.

Figure 2: a) Maximum number of bi-directional ports. b) Maximum bandwidth over a 1mm bi-section. c) Memory limits for an 800mm^2 chip.

more predictable output performance, since each output port is guaranteed a certain amount of memory. The major problem with this scheme is that one must add enough memory to *all* of the queues to ensure that any *one* can handle the target peak load, resulting in tremendous inefficiency.

We decided that a regular structure of identical nodes (k-ary n-cubes) best fit our scalability requirement, but these structures necessitate spreading out memory resources[5], leading to inefficiency in silicon real estate allocation. We would rather devote on-chip space to more advanced Quality Of Service(QOS) hardware[6].

The memory organization we used in our test architecture was based on the concept of wormhole routing[2][3][4]. In wormhole routing, flows of data are broken up into blocks called *flits.* A flit is defined as the smallest bufferable chunk of data. Wormhole routing is a circuit switched protocol in which a path from the input port to the output port is established, and persists until the packet has been received by the destination port in the network. The path across chip is comprised of a number of *hops* in intermediate nodes. The nodes only buffer data if a blockage occurs along the path, either at the terminal end (due to a lack of buffer space), or along the route (due to a lack of routing channels). In either case, the flits are buffered along the route in the intermediate nodes. The tendency to buffer en route makes wormhole routing optimal for our situation. If the output queue memory at the terminal node overflows, a packet need not be dropped, since it can temporarily share memory with the ports along its path. Our model therefore incorporates *transit buffers*, which allow wormhole buffering along a packet's path through the fabric. The buffers are only accessible to packets whose destination is *not* the current node.

It is impractical to allocate memory on a flit-by-flit basis, as the line widths for an embedded memory would most likely be larger than a flit. This is especially true for embedded DRAM, which would require a large line size to mitigate its slower speed. We would therefore allocate memory in blocks sized as an integer number of flits. Unfortunately, if the minimum block size is 20 flits, and a packet is only 10 flits long, another 10 flits worth of memory are wasted. Another problem is the choice of the amount of memory that any node can grant a transitory(not destined for that node) packet. Intuitively, since the goal of this architecture is to maximize memory sharing across a chip, the optimum situation would be one in which every packet was strung out evenly across its path length:

$$N_{bph} = \frac{L_{packet} - B_{oq}}{N_{hops}} \qquad (2)$$

where N_{bph} is the number of flits allocated per hop, L_{packet} is the packet length in flits, B_{oq} is the number of flits allocated in the output queue, and N_{hops} is the number of intermediate nodes a packet's circuit traverses.

4. SIMULATION

We used a software simulator written in C++ to explore the memory architecture/allocation space. Our simulator is capable of running systems with an arbitrary number of ports(*switch nodes*) and connectivity between the ports. The simulator currently only supports toroidal(k-ary n-cube[1]) topologies of arbitrary degree, though the overall simulator architecture does not preclude other topologies[16]. The simulator can achieve speeds of 50-100k cycles per minute on a 64 node switch.

The amounts and ratios of the two types of memory in our system(output queue and transit buffer) are parameterizable, as is the minimum block allocation for each. Figure 3 illustrates the functionality of our prototype network. The bottom of the figure indicates each node's position in the network. The spaces between the lightly shaded node boxes on the upper part of the figure represent interconnect with data in flight between nodes. In this example time moves forward as one moves down the figure. Packets are routed adaptively via minimal paths.

At time t_0, a header is received at node 1. After processing(assumed to take zero time for simplicity), the incident node chooses one of a number of *inter-node channels* to send the incoming packet out on. It is important to note that two adjacent nodes may have many such channels between them, but only one is used per packet. The incident node sends a *connect* message (along with the header) to the next node in this packets on-chip route. This message is propagated along the shortest path to the target node(8), until at t_3, a connection message reaches the destination node. At this point a circuit is established that will remain while the packet is being transmitted to the destination node. At t_1, the first data block or *flit* arrives at the input port. This flit is blocked pending the reception of an *unblock* message from the next node in the path, which is received at time t_2.

On reception of the unblock message, the first node transmits a data flit, while buffering the arriving data flit which is waiting for the connection to unblock. The transmitted flit arrives

[5]Unless memory is shared between clusters of nodes, which is not explored in this paper
[6]By this we mean structures such as per-session queues and class based scheduling

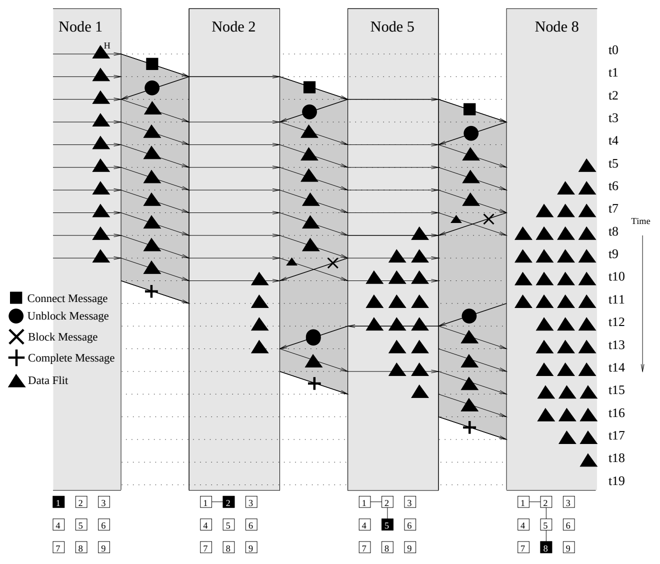


Figure 3: Network functionality

at the first intermediate node(2) before the entire connection is established through to the destination node. In this example, the packet is being buffered at the output node(8) because another packet is ahead of it in line to exit. Therefore, the first data flit is received and buffered at time t_5. This buffering continues until time t_7, when the packet is one flit away from filling its output queue allocation(in this case 4 flits) in node 8. At this time, the destination port(8) sends a *block* message to the second to last node in the path(5). Though it subsequently receives another flit at t_8, fully utilizing its output queue allocation, further transmissions are stopped and the next to last node begins to buffer in its transit memory. The second to last node(5) in turn buffers until its allocation is filled at time t_{10} when the second node in the path(2), after receiving a block message, begins to buffer.

One cycle later, at time t_{11}, node 2 receives a *completion* message from the first node(1). The first node has at that point finished transmitting data, and will subsequently deallocate the *channel* the packet was using, freeing it for use with other packets. The dark gray areas of the diagram represent the lifetime of the inter-node channels.

In this example, the output port on the last node(8) is freed simultaneously with the arrival of the completion message at the second node(2). An unblock message is propagated backwards to the first active node in the chain, while flits are being sent out the output port in order. If this had not been simultaneous, the packet and all of its allocated resources(channels and buffer memory) would have remained static until the output port was freed. At time t_{14}, the second node(2) transmits its final flit, which is followed one cycle later by a completion message, and the freeing of all resources allocated to the packet. The packet is transmitted in this manner until the last flit has passed through the output queue.

This example does not cover packet dropping. In our simulator, the packet in the previous example would have been dropped if the packet had been blocked until the first node had run out of transit space. At that time, there is no previous node to block, and the packet is dropped by sending a *drop* message forward from the incident port all the way to the end of the circuit. The simulator currently produces data such as the number of flits in, out and dropped, system utilization, average memory used per cycle, average latency, as well as detailed statistics on the characteristics of dropped packets (average length, number of hops, and memory allocation). The later statistics will be integral to the following analysis.

5. ANALYSIS

The purpose of this section is to answer the following three questions:

- What is the optimal ratio of output queue memory to transit buffer memory?
- What is the optimal policy for distributing transit memory to packets blocked at intermediate nodes?
- What do these policies cost us in terms of interconnect?

The first question aims to discover the amount of each of the two types of memory that must be at each node. To answer this, we ran simulations in which the total node memory stayed constant, but the ratio of output memory to transit memory changed. For the second question, we will show results for two different policies: *Distributed Packets*, in which the switch makes its best effort to spread a buffered packet across the chip, and *Greedy Packets*, in which a hard allocation is granted to each packet in every node it passes through, regardless of the packet's length. We will analyze the results of the two aforementioned studies to determine optimal points from the memory utilization, and dropped packet probability perspective. The third question will be answered by looking at wire utilization metrics produced by our simulator.

As we have been assuming that the the usage model for a chip of this type will be that of a high bandwidth IP switch, we used a packet length distribution from the Cooperative Association for Internet Data Analysis(CAIDA)[13], which is based on real network traces. Packet arrival rates were drawn from a Poisson process, and the destination distribution in the network of an incoming packet was uniform. The network for the following analysis is a 4-ary 3-cube(64 node 3-d torus), and the system utilization is 93%[7].

In the following analysis, we concentrate on the *hop length* of dropped packets. We define the hop length as the number of nodes that a packet must traverse before it reaches its output port. This value is important because we expect that a packet with a large number of intermediate nodes, and therefore a high probability of finding buffer space, will be less likely to have been dropped than a packet with a short path length when transit buffers are implemented.

Figure 4 shows baseline data for a system in which nodes only have dedicated output queue memory(there are no transit buffers, and all buffering for an output port takes place within that port). In this simulation, as with all the rest, interconnect bandwidth is considered infinite, therefore the amount of dropped packets is related only to the output queue depth. Figure 4b shows the aggregate packet drop rate for the 64 node system for output queue depths ranging from 7000 flits(28kB) to 17000 flits(68kB). As we would expect, the packet drop rate decreases as the amount of available output queue memory increases. Figure 4a breaks down the dropped packets by their on-chip path length. Since our destination distribution is uniform, the longest path possible in a 4-ary 3-cube is six, and

[7]System utilization is calculated by dividing the number of flits into the system by the number of nodes times the number of cycles run

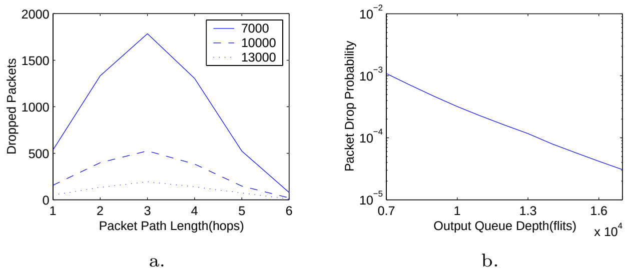


Figure 4: Baseline simulation; a) Dropped packets by path length for three different output queue depths(measured in flits). b) Dropped packet probability vs. output queue depth.

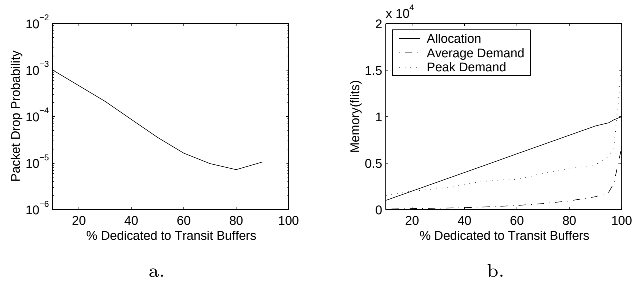


Figure 5: Distributed Packet Simulation; a) Dropped packet probability by memory ratio when total node memory is limited to 5k flits. b) Transit buffer utilization.

zero hop packets(packets whose ingress node is the same as their egress node) are disallowed, we see a Gaussian distribution centered at 3 hops for all three output queue depths.

5.1 Distributed Packets

To verify the assertion of equation (2), we allow intermediate nodes along a packet's path some extra information: the number of hops preceding them in the packet's path. When the terminal node in a 4 hop path blocks, the second to last node will only give enough buffer space to buffer 1/4 of the flits that have yet to be transmitted[8]. It is expected in this case that when the second to last node blocks, the preceding one will buffer 1/3 of the remaining traffic and so on, until the ingress node is required to buffer all of the rest of the packet in transit buffers (ideally 1/4 of the packet overflow). Figure 5a shows the dropped packet probability for different ratios of output queues to transit buffers when the *total* node memory is kept steady at 5k flits. We see a steady decline in the dropped packet probability until the 90% transit buffers point, where the probability has dropped down to the order of 10^{-6}. We see however, that a ratio of 1:4 output queues to transit buffers seems to be optimal. Interestingly, going beyond the 1:9 point, toward the 100% transit buffers point (no output buffering), the dropped packet probability sky-rockets to over 20%. Figure 5b shows both the peak and average transit buffer demands for this simulation, as well as the transit memory ceiling for each point defined by system parameters. For the majority of this plot, the peak remains below the maximum allocation. However,

[8]Since a four hop path will cross five nodes including the terminal one

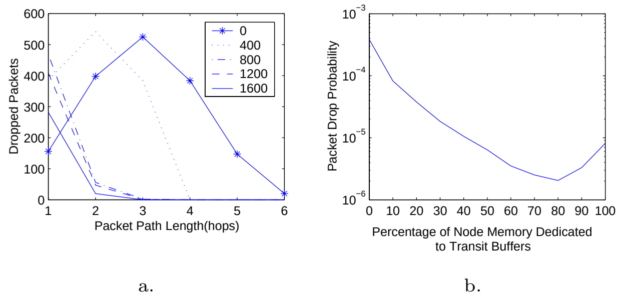


Figure 6: Greedy Packet Simulation; a) Packet drop rate vs. path length for 5 different per node transit buffer allowances at the 9:1 memory ratio. b) Packet drop probability for 11 memory ratios with total node memory 10k flits.

near the right side of the graph, we see a sharp, exponential increase in both the average and peak demands, ending with the peak at 15000 flits. It was at this point in the simulation that the packet dropping increased due to switch saturation.

5.2 Greedy Packets

Though the memory allocation technique just discussed reduced the dropped packet probability significantly, the requirement that intermediate nodes store the number of nodes preceding them in each packet chain might be a prohibitive factor in the design. Therefore, we sought to investigate the effectiveness of fixed limits at each node.

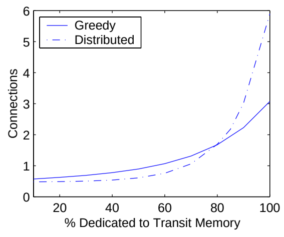
Figure 6a shows the results of this experiment where each node has 10k flits of memory instead of the 5k of the previous study. For comparison, we have included the data for the case in which there are only output queues (no transit buffers). The other four curves represent progressively larger transit allocations to each packet that passes through a node. We find that as we increase the allocation, effectively allowing packets to become greedier, clustering closer to their destination ports, the dropped packet rate decreases significantly. Also, we notice that the mean path length for dropped packets shifts to the left until nearly all dropped packets have traversed only one hop. This is a good indicator that this scheme is working properly, as the probability of finding transit buffer space increases with path length. The greatest percent decrease in dropped packets occurs when 200 flits are allocated per node per packet.[9]

Figure 6b shows the overall dropped packet probability for 11 ratios of output queue memory to transit memory. In this experiment, the maximum allocation per node is set to be an entire packet(400 flits or 1600 bytes). We see a 2+ orders of magnitude improvement over the case with only output queues when the ratio is 1:4 output buffers to transit buffers. The graph also trends upward at ratios beyond 1:4. All dropped packets in this region only traveled one hop. These packets were dropped due to an inopportunity to get enough queue memory along their path caused by their short path length and not due to saturation, as in the distributed study.

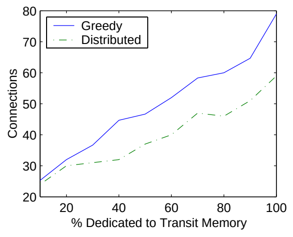
5.3 Interconnect Cost

One downside to circuit-switched transmission protocols is their tendency to hold wire resources longer. In a Store-and-

[9]The packet length distribution specifies that this is larger then 80% of injected packets



a.



b.

Figure 7: a)Average inter-node connections. b)Peak inter-node connections for greedy packet study.

Forward network, a path that is 3 hops long will hold 1/3 of the path at any one time, since packets are fully buffered at intermediate nodes before being granted a channel to the next node in the path.

Figure 7 shows the peak and average number of connections for the inter-node interfaces on-chip for both the distributed and greedy packet studies. When no transit buffers are allocated, and only output queues are used, we see a peak utilization of less than ten channels. As the memory ratio increases, so to does the peak channels used metric. The average number of connections(Figure 7a) increases exponentially as one moves in the direction of more transit buffers. It is important to note however, that the average utilization is quite low.

6. CONCLUSION

In this paper, we showed that wormhole routed architectures using distributed *transit buffers* for packet buffering allow us to attain a better packet drop probability by more than as 2 orders of magnitude over the case in which ports(nodes) only use output queues. This is due to the tendency to share transit memory between ports. We discussed two ways of allocating memory resources in this environment: *Distributed* and *Greedy*. We found that while the Distributed method achieved a better packet drop probability with less memory than the Greedy method, the distributed method did not react well to the memory saturation seen when the switch is using only transit buffers, and not output queues. In both cases, a switch with only transit memory did not perform as well as mixed designs. In both the Distributed and Greedy packet models, we saw that a 1:4 ratio of output queues to transit buffers is optimal. We also showed that the cost of such optimizations is more interconnect. However, the the number of physical/virtual channels actually *passing* data at any one time is quite small.

7. FUTURE WORK

The intent of this study was to begin with first principles(basic silicon constraints), rising up to a level in which on-chip structures were created to realize network switch functionality. For this reason, the study concentrated only on memory utilization. Obviously, many other variables exist, such as the traffic model. A uniform distribution with Poisson arrival rates is good for a first order survey, but what about bursty models(representing FTP), or hot-spot models? We believe based on extrapolation of existing data that such traffic will require more memory, but that the transit buffers should still be better utilized than output queues due to the variability of these models.

We would like to experiment with topologies other than k-ary n-cubes. How does a mesh topology(non-toroidal) affect the overall performance? Higher and lower dimensional systems(2-d or 4-d) in addition to other topologies are also interesting avenues for exploration. We have shown that longer paths and high connectivity can utilize transit buffers effectively, but how does this affect message latency? If the head of all messages reaches the output queue without blocking due to sufficient inter-node bandwidth, the latency is just a function of the load and queue depth on that particular output port, resulting in a seemingly ideal fabric.

8. REFERENCES

[1] W.J. Dally *Performance Analysis of k-ary n-cube Interconnection Networks*, IEEE Trans. Comput., vol. 39 no. 6, pp. 775-785, Jun 1990.

[2] W.J. Dally and C.L. Seitz *The Torus Routing Chip,* J. Distributed Computing, vol. 1, no. 3, pp. 187-196, 1986.

[3] L.M. Ni and P.K. McKinley *A Survey of Wormhole Routing Techniques in Direct Networks*, IEEE Computer, vol. 26, issue 2, pp. 62-76, Feb 1993.

[4] J. Duato, et. al. *Interconnection Networks.*, Los Alamitos: IEEE Computer Society Press, 1997.

[5] J. Rexford, et. al. *A Router Architecture for Real-Time Communication in Multicomputer Networks*, IEEE Trans. Comput., vol. 47, no. 10, pp. 1088-1100, Oct 1998.

[6] A. Boni, et. al. *LVDS I/O Interface for Gb/s-per-pin Operation in 0.35-μm CMOS,* IEEE Journ. Solid-State Circ., vol. 36, no. 4, pp. 706-711 Apr 2001.

[7] A. Tanenbaum *Computer Networks*, Englewood Cliffs:Prentice Hall, 1989.

[8] G. Kornaros, et. al. *Atlas I: Implementing a Single-chip ATM switch with Backpressure,* IEEE Micro, vol. 19, issue 1, pp. 30-34, Jan-Feb 1999.

[9] D. Pnevmatikatos and G. Kornaros *Atlas II: Optimizing a 10Gbps Single-Chip ATM switch,* Proc. IEEE Intl. ASIC/SOC Conf., pp. 142-146, 1999.

[10] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors,* 1999.

[11] O. Takahashi, et. al. *1-GHz Fully Pipelined 3.7-ns Address Access Time 8kx1024 Embedded Synchronous DRAM Macro,* IEEE Journ. Solid-State Circ., vol. 35, issue. 11, pp. 1673-1679, Nov. 2000.

[12] S. Rixner, et. al. *Register Organization for Media Processing*, Proc. Intl. Symp. on High-Perf. Comp. Arch., pp. 375-386, 2000.

[13] Cooperative Association for Internet Data Analysis, http://www.caida.org

[14] P. W. Diodato *Embedded DRAM: More than Just a Memory,* IEEE Communications, vol. 38, issue 7, pp. 118-126, Jul. 2000.

[15] D. Keitel-Schultz and N. Wehn *Embedded DRAM development: Technolgy, physical design, and application issues,* IEEE Design & Test of Comp., vol. 18, issue. 3, pp. 7-15, May-Jun. 2001.

[16] F. Karim, et. al. *On-chip communication architecture for OC-768 network processors,* Proc. Design Autom. Conf. Las Vegas, NV, pp. 678-683, Jun. 2001.

[17] G.J. Jeong et. al. *Reconfigurable Shared Buffer ATM Switch,* Proc. IEEE APC Conf. on ASICs, pp. 327-330, 1999.

[18] Cisco Systems http://www.cisco.com/warp/public /cc/pd/ifaa/192/prodlit/cc1_ds.htm