A New Hardware Efficient Link Scheduling Algorithm to Guarantee QoS on Clusters *

J.M. Claver[1], M.C. Carrión[2], M. Canseco[1], M.B. Caminero[2], and F.J. Quiles[2]

[1] Dept. of Computer Science and Engineering. E.S.T.C.E.
Univ. Jaume I, 12071 - Castellón, SPAIN.
{claver,canseco}@icc.uji.es
[2] Dept. of Computer Science, Escuela Politécnica Superior.
Univ. de Castilla-La Mancha, 02071 - Albacete, SPAIN.
{carmen,blanca}@info-ab.uclm.es

Abstract. Contemporary router/switch technology for high-performance local/system area networks (LANs/SANs) should provide the capacity to fit the high bandwidth and timing requirements demanded by current applications. The MultiMedia Router (MMR) aims at offering hardware-based QoS support within a compact interconnection component. One of the key elements in the MMR architecture is the link scheduling algorithm. This algorithm must solve conflicts among data flows that share an input physical link. Required solutions are motivated by chances for parallelization and pipelining, while providing the necessary support both to multimedia flows and to best-effort traffic. In this work, a cost-aware link scheduling based on the temperature coding of priority value associated to every head flit is studied.

1 Introduction

Current applications include not only best-effort traffic such as ftp or e-mail but also multimedia QoS-aware applications. Numerous examples can be highlighted from web-based applications, streaming media, visualization, interactive simulations, virtual meeting and collaborative design environments.Clusters are being commonly used as back-end servers for these applications, so some QoS support is needed within the underlying interconnection network.

Router/switch technologies developed for high-speed multiprocessor interconnection networks or LAN/SANs were optimized for providing low latency to best-effort traffic. Available commercial SAN/LAN fabrics such as IBM SP2 [1], Myrinet2000 [2], or Quadrics [3] are not designed to permit concurrent guarantees of communication performance to multiple applications. Also, commercial Gigabit Ethernet switches with QoS support seem to be best suited to large enterprise backbone networks, and to interfacing with MANs and WANs, rather than to cluster environments [4]. QoS support in Ethernet switches for LAN environments is limited to providing two or four queues with different priority levels per port [5]. Also, while in the mid 90s several clusters where built around ATM interconnects, nowadays high-performance clusters [6] are built around

* This research was partially supported by the CICYT Projects TIC2003-08154-C06-04 and TIC2003-08154-C06-06

high-performance networks such as Myrinet [2], Quadrics [3] or InfiniBand [7] because ATM switches are too complex and slow for this environment.

The problem of providing architectural QoS support within switching elements in cluster and local area environments is still an open issue. Thus, the MultiMedia Router[3] (MMR) architecture [8] arises as a solution to provide hardware-based QoS support within an interconnection component targeted for use in cluster and LAN environments. The MMR organization is based on input queues and a multiplexed crossbar internal switch.

In order to achieve high link bandwidth utilizations and to provide the QoS needed by the applications the MMR requires efficient traffic scheduling algorithms. These algorithms decide which data must be transmitted at each time, so their behaviour will determine whether QoS guarantees are fulfilled or not. Most of the scheduling solutions appearing in the literature for such organization seek to maximize the link and internal switch utilization [9][10], and do not address QoS issues. Some recent research on scheduling algorithms tries to offer both high throughput and QoS support [11].However, these are almost theoretical solutions. Compact and fast hardware implementations of these algorithms are hardly feasible, which prevents their use in high-speed interconnection networks. Moreover, most scheduling solutions for input-buffered switches need to run at speeds higher than links to provide QoS guarantees and high link utilization, or lack the needed flexibility to concurrently accommodate different connection requirements.Thus, in this paper, a new link scheduling algorithm is presented, the Temperature-IABP (TIABP). The main features of this algorithm are shown to provide high-throughput and QoS guarantees to the different multimedia flows, according to their reservations, while being suitable for a simple low cost hardware implementation. Preliminary performance evaluation results and implementation on a Xilinx Virtex 2000E FPGA of a Temperature-IABP based link scheduling are presented. Due to the complexity of this device high level language like HandelC is used.

The rest of the paper is organized as follows. First, Section 2 outlines the main characteristics of the Multimedia Router architecture. Then, in Sections 3 and 4 the new resource scheduling algorithm proposed for the MMR and its hardware architecture features are explained. Extensive evaluation results follow, which reveal the effectiveness of the proposed link scheduling algorithm. Finally, some conclusions are given.

2 The Multimedia Router

Figure 1 depicts the general organization of the Multimedia Router. In the following paragraphs, the basic building components will be briefly described [8].

a) Input Buffers: To support a large number of multimedia connections, the storage buffers at each input link are organized as a set of virtual channels. One virtual channel is provided per connection, in order to consider the QoS of each flow. This approach also avoids HOL-blocking [12].

[3] The MultiMedia Router is devised as a link-layer interconnection element. The term "router" is inherited from the interconnection elements used in multicomputer and multiprocessor networks, rather than from the IP world.

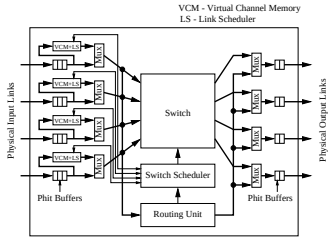


Fig. 1. MultiMedia Router organization.

The MMR avoids losing data due to buffer overflow by using *per connection flow control* at the link level. The selected scheme is *credit-based flow control*,as InfiniBand^{SM} does [7]. The flow control unit will be referred to as a *flit*.

b) Routing Unit: The MMR uses a *hybrid switching technique*: a connection-oriented scheme (Pipelined Circuit Switching (PCS) [13]) for the multimedia flows, and Virtual Cut-Through (VCT) [14] for best-effort messages. Then, the routing unit computes the path to follow a flow according to the Exhaustive Profitable Backtracking (EPB) routing algorithm [15]. On the other hand, best-effort messages are routed according to a fully adaptive routing algorithm [16].

c) Internal Switch: Due to the large number of virtual channels, the MMR internal switch is a *multiplexed crossbar* with as many ports as physical channels. This crossbar organization implies several arbitration tasks. First, *the LS module* solves conflicts at the input side selecting the virtual channel that will use the crossbar input port in the next flit cycle. Then, *the Switch Scheduler module* does a second arbitration, because several input channels might request the same output link for the same flit cycle. These arbitration tasks are carried out by the *link and switch scheduling algorithms*, respectively.

In the MMR, link bandwidth and switch port bandwidth are split into *flit cycles* [4]. The number of flit cycles in a round is an integer multiple K ($K > 1$) of the number of virtual channels per link. The allocated flit cycles will be assigned to the requesting connection every round.

The choice of the link and switch scheduling algorithms are critical parameters for the MMR. Both scheduling algorithms must cooperate to guarantee that the bandwidth allocated to each connection is available during data transmission. Besides, these algorithms must be well suited to parallelization and pipelining. In order to meet these design goals, the link and switch scheduling algorithms proposed for the MMR are partitioned into three basic decisions: *Candidate Selection*, *Port Ordering* and *Arbitration*. The Candidate Selection phase is carried out by the link schedulers. Thus, it is performed in parallel for every input link. Its purpose is to select a set of one or more virtual channels, with flits ready for transmission, called *candidates*. The switch scheduler, also known as crossbar arbitration, tackles the other two phases. They are aimed at selecting a set of conflict-free input/output port matchings among the candidates chosen by link schedulers.

[4] A flit cycle is the time taken for a flit to be transmitted through the router and across the physical link. Flit cycles are grouped into *rounds*.

3 A Cost-effective link scheduling

The link scheduling algorithm carries out the Candidate Selection phase of the scheduling problem, thus is, it chooses a small set of candidates per every physical input link. Selection will be based on a *biased priority scheme*associated to every head flit. The key point in our scheme is that priorities are biased according to the ratio between the QoS a flit is receiving and the one it should receive.

To be more precise we proposed the *Inter-Arrival Biased Priority (IABP)* scheme. In this case, the priority of a flit is computed as the ratio between the queuing delay, and the inter-arrival time (IAT) for the flits in the connection. The effect is that the priority grows as queuing delay grows. Moreover, priority grows faster for those flits belonging to high-bandwidth consuming connections, that is, there are more chances that they will be forwarded sooner through the switch.

In a first approach, looking for a more practical implementation biasing function, the *Simple IABP (SIABP) algorithm* has been devised. The idea is to apply the same rationale introduced by the IABP algorithm, that is, to relate the bandwidth required by the connection to the experienced queuing delay, but replacing the division with some other less expensive operation. Equations 1 and 2 show how the priority of the header flit is computed and updated:

$$Priority(0) = NumCycles \tag{1}$$

$$Priority(t) = Priority(0) << n \tag{2}$$

where *NumCycles* is related to the bandwidth reserved for a connection and n is the position of the highest significant bit set to 1 in the register that holds the queuing delay. In this way the QoS needed (represented by the initial priority value) is also related to the QoS received by the flit (the queuing delay).

An important hardware area reduction is got by using this algorithm. Nevertheless, and focusing as well on hardware reduction we propose a new link scheduling: the *Temperature IABP, TIABP*. The purpose of the TIABP algorithm is not only to get low area cost but also to improve some critical aspects of the SIABP algorithm. Considering equations 1 and 2, which describes how is computed and updated the SIABP priority, we can be aware that inversion priority can happen. The priority of the header flit allocated in a virtual channel increases with the time being waiting for the output port. Nevertheless, if the header flit remains there for a long time, its priority could decrease. In particular, whether the inicial priority value is a power of two, the priority updated can down to zero. In order to avoid priority inversion, extra hardware must be added to control overflow by freezing the priority of a virtual channel when the most significant bit of the register that holds the priority is equal to 1. However, this is not a solution if priorities are not power of 2. Being aware of this problem, the TIABP priority biasing function is computed as follows:

a) A counter stores the queuing delay of a flit and it is updated every router cycle, in the same way as in the IABP and SIABP algorithms.

b) The initial value for the priority is $(2^{k+1} - 1)$, where k is the position of the highest significant bit set to 1 of the bandwidth required by the connection

expressed as the flit cycles per scheduling round reserved to service the average bandwidth of the connection ($MSB(NumCycles)$).

c) Next, the priority of the flit is computed as the product of the queuing delay times the bandwidth requirements. But, in order to achieve a simpler hardware design, and similar to SIABP, the product is replaced by shifting operations. More precisely, the priority value is updated by shifting to the left its current value (i.e., it is multiplied by 2) and by setting the less significant bit. We should remark that the shifter operation is done each time the queuing delay becomes greater than $1, 2, 4, \ldots, 2^n$, i.e., every time a bit in the queuing delay counter is set for the first time since it was reset.

The TIABP algorithm is summarized in equations 3 and 4:

$$Priority(0) = 2^{k+1} - 1,\ k = MSB(NumCycles) \tag{3}$$

$$Priority(t) = (Priority(0) << n) + (2^n - 1) \tag{4}$$

where n is the position of the highest significant bit set to 1 in the register that holds the queuing delay.

As in the SIABP algorithm, TIABP implementation is just reduced to a shifter and some combinational logic. Nevertheless, using TIABP priority biasing function, priority inversion is not possible because a priority value can not ever decrease. If the priority of the header flit allocated in a virtual channel increases, with the time being waiting for the output port, the maximum priority value is $2^{r+1} - 1$, where r is the priority register bitwidth. So, extra hardware to control register overflow is not necessary. Moreover, when the priority is increased on SIABP, a minimum of 2 bits change their values before overflow. By using TIABP, a maximum of 1 bit changes its value every time the priority is increased. This behavior has direct effects on power dissipation.

The priority temperature coding used on the TIABP also simplifies the sorting stage circuitry of the link scheduler which generates the *candidate vector* that is sent to the *Switch Scheduler*. This stage, called SORT, is a sorting bitonic network, and their basic elements are comparators. By using temperature coding, the design of this comparators are less complex.

4 FPGA Hardware Architecture

In order to implement a MMR on a single chip we have studied the design and implementation of some of their more important modules on an FPGA based board. As a result of this work, a more simple and lower power dissipation circuitry has been obtained.

We have centred our effort on the design of SIABP and SORT modules, which are the core of the *Link Scheduler (LS)*. Thus, we have developed a new version of LS using TIABP and TSORT (Temperature-SORT) modules.

4.1 Hardware Implementation

In order to evaluate the results of TIABP design, we have used the Celoxica RC1000 PCI based FPGA board [17]. The RC1000 is a PCI bus plug-in card for PC. It has one large Xilinx FPGA (in our case a Virtex 2000E, with 2 million

equivalent gates) with four banks of memory for data processing operations and two PCI Mezzanine Cards(PMC) for input/output with the outside world. The Virtex 2000E is based on slices that contain two 4-bit LUTs each one.

The FPGA Virtex has been programmed using HandelC [18] and the Celoxica DK1 environment. HandelC is a behavioral C based hardware description system developed by Celoxica that allows Co-simulation. Parallelism of process and synchronization are taken from the CSP model, in particular, from the Occam language. HandelC uses standard data types with user defined bitwidths. So, HandelC provides an efficient use of hardware resources.

The SIABP module updates the priority connection each time the queuing delay is increased to the next power of 2, following the HandelC macro

$$shl0(Priority) = ((Priority <- (width(Priority) - 1)@0b0,$$

while the TIABP module updates the priority following the HandelC macro

$$shl1(Priority) = ((Priority <- (width(Priority) - 1)@0b1,$$

where @ is the concatenation operator. Both, SIABP and TIABP modules, update *Priority* in 1 clock cycle.

The bitonic network of SORT and TSORT modules is builded recursively with basic sorter blocks which sort 2 virtual channels in function of its priority. These basic sorter blocks use a comparator, implemented as a boolean macro, called *Prio_greater*. For the SORT module this macro is expressed as

$$Prio\_greater(Priority1, Priority2) = (Priority1) > (Priority2),$$

while for the Temperature-SORT (TSORT), it is expressed as

$$Prio\_greater(Priority1, Priority2) = ((Priority1)|(Priority2)) == (Priority1).$$

A bitonic network needs $O(log_2(ncv)^2 ncv)$ modules of *Prio_greater* and $O(1/2log_2(ncv)^2)$ clock cycles to compute a *candidate vector*, where *ncv* is the number of virtual channels per input link.

5 Performance evaluation

This section presents the implementation results in Xilinx Virtex 2000E FPGA and the performance evaluation of the link scheduling proposed in this paper being used inside the MMR.

We first present preliminary scaling results expressed in terms of FPGA area and maximum clock rate for the SIABP/TIABP and SORT/TSORT modules. Then, the performance evaluation of the link scheduling is done by using a discrete-event C++ simulator.

5.1 Area/Delay Results

For the purpose of this evaluation, we scaled the design of SORT/TSORT modules to handle a different number of virtual channels from 4 to 128. The bitwidth used to represent priority of each virtual channel is 16 bits in all cases. For each result, we report the FPGA area (in equivalent NAND gates) and maximum clock rate provided by the Celoxica DK1 tools.

Figure 2 reports FPGA area/delay results obtained by SORT and TSORT designs for 4, 8, 16, 32, 64, and 128 virtual channels. As the number of virtual channels increase from 4 to 128, the area used by SORT and TSORT designs

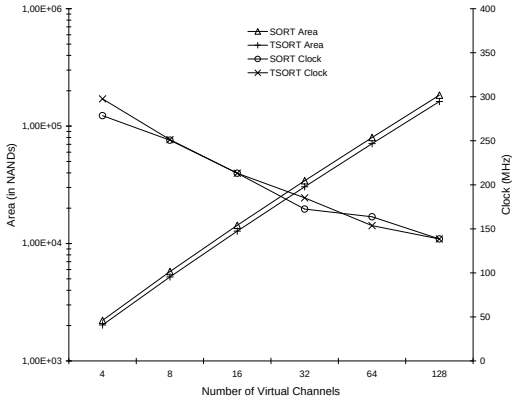


Fig. 2. Area/Clock Rate estimation for SORT and TSORT implementation with different number of Virtual Channels.

increases linearly. But in all cases the TSORT implementation uses between 9% and 13% less area than SORT. Respect to the maximum clock rate, both designs obtain similar results. Thus, maximum clock rate decrease in a logarithmical way when the number of virtual channels is increased. This similar delay between both SORT and TSORT implementations is due to the fact that TSORT in completely instantiated by the DK1 tool using FPGA LUTs, but SORT is instantiated using LUTs (in all cases a number greater than in TSORT) and other FPGA specific resources that accelerate the *Prio_greater* modules used in its design.

In the case of biased function, we have found that there is not important differences between SIABP (without support of overflow control) and TIABP. For an initial priority bitwidth of 12 bits, the SIABP module design utilizes 50 LUTs (53 LUTs if overflow control is included) while the TIABP module utilizes 51 LUTS. The maximum clock rate in both designs is 294 MHz.

5.2 Simulation Results

Having into account the previous results we are going to evaluate the performance of a single 4×4 MMR, with full-duplex 1.24 Gbps 16 bit-wide links. This gives a router cycle of 12.9ns. The number of virtual channels per input link is 128. Flits are 1024 bit long. The MMR buffers have capacity to store one flit per virtual channel. The scheduling round size, determined by the K parameter, has been been set to $K = 16$.

Constant and Variable Bit Rate traffics (CBR and VBR traffic, respectively) have been considered in simulations. The *CBR traffic model* is composed of a mix of synthetic connections with 64 Kbps, 1.54 Mbps, and 55 Mbps average bandwidth. These are representative of several applications, such as audio, video, and high-definition video transmission, respectively. The *VBR traffic model* is based on traces obtained from real MPEG-2 video sequences. This is a typical type of multimedia flow [19].

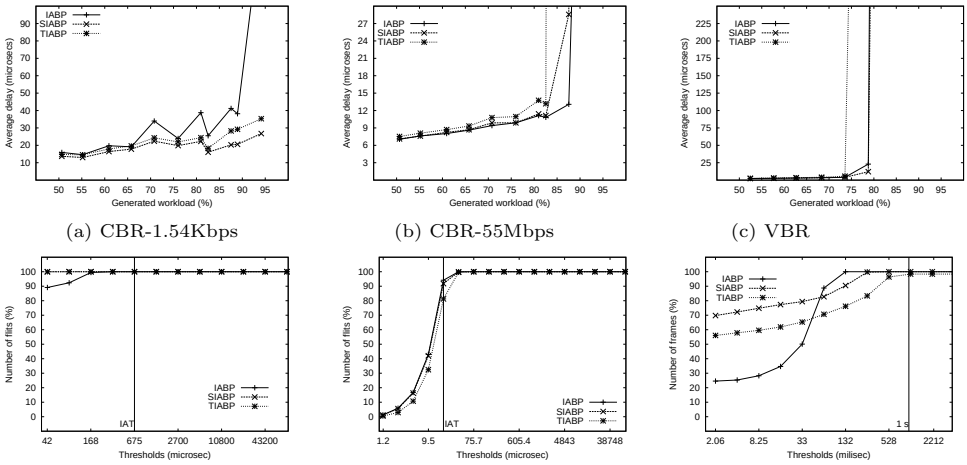
Traffic sources inject their flits into buffers located in the corresponding NIC attached to every input/output port. Input workload is measured as a percentage

of link bandwidth. Destination ports have been randomly selected using a uniform distribution. No statistical information is gathered until some scheduling rounds have been completed in order to get data only when the system is stable. Simulations have been carried out for long enough to record significant data on applications performance. Due to space limitations, only the most significant results are presented in this paper.

First, the TIABP scheme is compared with both the IABP and SIABP algorithms. Average delay since generation for the most demanding CBR flows and the VBR traffic are presented in Figures 3(a-c). Results show slightly difference when load is under saturation point. Close to saturation, the TIABP algorithm improves the other two proposals for low and medium CBR flows but, the performance of the TIABP algorithm degrades for high CBR and VBR traffics.

Nevertheless, in order to know whether the multimedia flows are receiving the requested QoS, we have depicted the distribution of flit delays and frame delays for the CBR and VBR flows, respectively. This distribution is computed as the percentage of flits/frame that suffered a delay lower than a set of thresholds. It can be seen that lowest deadlines are fulfilled with the TIABP algorithm but for all the cases the scheme obtains reasonable deadlines for all the flits. Note that thresholds are related to the QoS needs of the connection.

From the analysis above, it can be concluded that the router with the TIABP scheme is able to provide QoS guarantees to the multimedia flows at a lower hardware cost than the IABP algorithm.



(a) CBR-1.54Kbps (b) CBR-55Mbps (c) VBR

(d) 85% load. CBR-1.54Kbps (e) 85% load. CBR-55Mbps (f) 85% load. VBR

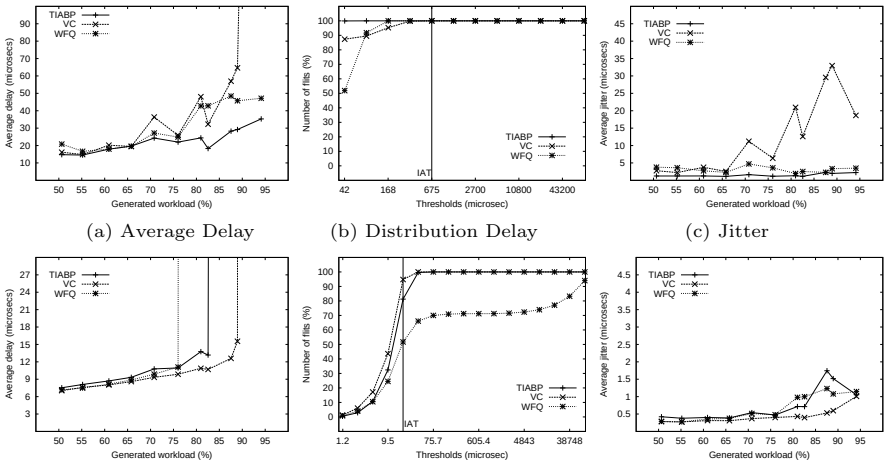
Fig. 3. IABP vs TIABP: Delay since generation (a)-(c) and Distribution delay(d)-(f).

For reference purposes, the performance obtained when using the practical TIABP algorithm is compared to that achieved when introducing a couple of classical algorithms acting as link schedulers. The algorithms chosen for this purpose are Virtual-Clock (VC) [20] and Weighted Fair Queuing (WFQ) [21]. Like TIABP does, both VC and WFQ assign priorities to flits according to the bandwidth requirements of the connection they belong to.

The plots shown in Figures 4(a) and 4(d) correspond to the average delay since generation obtained with the three link switch schedulers, for the CBR connections with medium and highest requirements. While TIABP outperforms the others algorithms for low and medium CBR flows, saturation is reached at lower loads than using VC algorithm for the most demanding CBR flows. On the other hand, considering the distribution delay of the algorithms, Figures 4(b) and 4(d), we can appreciate that little differences are found between the results obtained with VC and TIABP. Moreover, when WFQ is used a large amount of flits (30% of the generated flits, approximately) cannot fulfil even the most relaxed thresholds. Note that plots depict the distribution of flits delay for a workload of 85%.

The conclusion is that when WFQ is used, the connections with the highest bandwidth requirements cannot meet their QoS requirements, because they do not receive their share of bandwidth. On the other hand, VC and TIABP exhibit better behavior when operating in this way.

Last, Figures 4(c) and 4(f) show average jitter. TIABP is able to provide an almost constant average jitter over all the workload range, below or equal the values obtained for the other two algorithms.



(a) Average Delay (b) Distribution Delay (c) Jitter

(d) Average Delay (e) Distribution Delay (f) Jitter

Fig. 4. TIABP versus VC and WFQ: CBR 1.54Mbps (a)-(c) and CBR 55Mbps(d)-(f).

6 Conclusion

The main goal pursued by the MultiMedia Router (MMR) project is to design a single-chip router able to efficiently handle multimedia flows and best-effort traffic in LAN/SAN environments. In order to achieve this goal, solutions to many difficult resource management and scheduling problems must be provided, while keeping into account that these solutions must be simple enough to permit effective single-chip implementation.

The link scheduling algorithm is one key element on the MMR design to provide QoS guarantees to the multimedia flows. Thus, in this work the proposed Temperature-IABP scheme is analysed. We have obtained a simplified

link scheduling design based on the temperature coding of the priority value associated to every head flit and used in the candidate selection phase. Thus, TSORT design uses about 10% less FPGA area than the previous SORT design. Then, the new TIABP link scheduling algorithm reduce the hardware cost while is available of giving QoS guarantees to multimedia flows.

References

1. C. B. Stunkel et al., "The SP-2 high-performance switch," *IBM Syst. Jour.: Scalable Parallel Computing*, vol. 34, no. 2, 1995.
2. Myricom, Inc, *Guide to Myrinet-2000 Switches and Switch Networks*, August 2001.
3. F. Petrini et al., "The Quadrics network: High-performance clustering technology," *IEEE Micro*, January/February 2002.
4. R. Froom, M. Flannagan, and K. Turek, *Cisco Catalyst QoS: Quality of Service in campus networks*, chapter Exploring QoS in Catalyst, Cisco Press, 2003.
5. A. Pandey and H.M. Alnuweri, "Quality of Service support over switched Ethernet," in *IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing*, 1999.
6. "Top 500 Supercomputer sites," URL http://www.top500.org/, November 2003.
7. G. Pfister, *High Performance Mass Storage and Parallel I/O*, chapter 42: An Introduction to the InfiniBand Architecture, IEEE Press and Wiley Press, 2001.
8. J. Duato, S. Yalamanchili, M. B. Caminero, D. Love, and F. J. Quiles, "MMR: A high-performance multimedia router - Architecture and design trade-offs," in *Intl. Symp. on High Performance Computer Architecture (HPCA-5)*, 1999.
9. Y. Tamir and H.C. Chi, "Symmetric crossbar arbiters for VLSI communication switches," *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 1, 1993.
10. N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches," *IEEE Trans. on Networking*, vol. 7, no. 2, 1999.
11. I. Stoica and H. Zhang, "Exact emulation of an output queuing switch by a combined input and output queuing switch," in *IEEE/IFIP Intl. Workshop on QoS (IWQoS'98)*, May 1998.
12. M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queuing on a space division packet switch," *IEEE Trans. on Communications*, December 1987.
13. P. T. Gaughan and S. Yalamanchili, "A family of fault-tolerant routing protocols for direct multiprocessor networks," *IEEE Trans. on Parallel and Distributed Systems*, May 1995.
14. P. Kermani and L. Kleinrock, "Virtual Cut-Through: A new computer communication switching technique," *Computer Networks*, vol. 3, 1979.
15. P. T. Gaughan and S. Yalamanchili, "Adaptive routing protocols for hypercube interconnection networks," *IEEE Computer*, May 1993.
16. F. Silla and J. Duato, "Improving the efficiency of adaptive routing in networks with irregular topology," in *Conf. on High Performance Computing (HiPC)*, 1997.
17. Celoxica, *RC1000 Software Reference Manual*, 2001.
18. Chapell S. Sullivan, C., "Handel-C for co-processing an co-design of field programmable systems on chip," in *Proc. of the JCRA'02*, 2002.
19. "Generic coding of moving pictures and associated audio. Rec. H.262. Draft Intl. Standard ISO/IEC 13818-2," 1994.
20. L. Zhang, "Virtual Clock: A new traffic control algorithm for packet switching networks," *ACM Trans. Comp. Sys.*, May 1991.
21. A. Demers, S. Keshav, and S. Shenker, "Analysis and simulations of a fair queuing algorithm," in *ACM SIGCOMM*, 1989.