Split-Iterative and Sequential Multicast Scheduling For IQ Switches

Mohammed Shoaib
Department of Electrical Engineering
Indian Institute of Technology Madras
Chennai - 600036, India: **Email:** shoaib.m@iitm.ac.in

Abstract— **The objective of this work is to design and implement controlled (Weighted Round Robin Matching, WRRM) and sequential iterative schemes for weight based multicast traffic scheduling in input-queued (IQ) switches. Motivated by the practical synthesis of a scheduler for a 64-port optical crossbar switch, we demonstrate that limited and sequential iterations in the Weight Based Arbiter (WBA) lead to adjustable clock speeds and configurable designs with flexible control in the performance characteristics close to the conventional WBA. Our FPGA sizing experiments and clock speed evaluations show improvements of upto 46.48% and 19.12%, respectively, over the WBA. In addition, latency–throughput results for the proposed variations, highlight the trade-offs between fairness, throughput, hardware complexity and speed.**

I. Introduction and Background

Progressive integration of traditional communication services (Telephony, Radio, TV Broadcasting etc) into the internet has resulted in a number of challenges, one of them being the need for efficient multicast support. High performance computing cores and performance critical applications demand practical and efficient handling of multicast traffic. High efficiency and low hardware complexity are the watch words in current design methodologies. A number of architectures and implementations have been proposed to handle multicast traffic [3], [6]. We restrict our attention to input-queued switches in this design exploration. In particular, we consider how a scheduler can achieve a high throughput with efficient hardware complexity for multicast traffic queued in a FIFO fashion Sec.I-A.

Virtual Output Queues (VOQs) for multicast traffic have been proposed in literature [7]. VOQs avoid Head-of-line (HOL) blocking to a large extent but the issue with such a queueing scheme is that, it is impractical to maintain all the possible VOQs corresponding to all the outputs. For Example, in a switch with N output ports, $(2^N - 1)$ queues need to be maintained at the input. An alternative to this, called the multiple-queue architecture has been proposed [14], [9]. This uses k queues for N output ports, where $1 < k \ll (2^N - 1)$. This kind of a deisgn is unable to achieve high performance or run at high speeds. FIFO queues introduce HOL blocking but are more practical and hardware efficient [11], [8]. A practical and efficient way to integrate multicast and unicast scheduling was proposed in [13]. In this paper, we explore packet-switched, crossbar-based switches.

A. FIFO queues

FIFO Input-queued (IQ) switches typically operate on fixed-size data units called *cells*. R1, R2, and R3 are incoming multicast cells which are queued up in a FIFO fashion at each of the eight input ports. The structure of request R2 for the Input port 07 is shown enhanced in the Figure. The queue consists of a bitmap of N bits (N=8 here), N being the number of switch ports, where the bit in position i indicates whether the corresponding multicast cell requests output port i. There are N such FIFO queues corresponding to the N switch inputs in a N×M Switch.

B. Fanout

The total number of active requests in a particular bitmap constitutes the fanout of that particular cell. As the input queues are organized in a FIFO fashion, only the bitmaps at the heads of the FIFOs are considered by the multicast scheduler. Here fanout is used as a general term to stand for both the *constitution* and the *cardinality* of the input vector.

C. Scheduler

A critical component of the switch is the centralized scheduler Incoming traffic cells are scheduled (the scheduler computes a matching between the inputs and the outputs) and accordingly, the crossbar is configured at every cell cycle (time slot) or a multiple of a cell cycle, called the *matching epoch*. Our time slot target is 51.2ns (256-byte cells at a line rate of 40 Gb/s [4][5]). Known optimum multicast matching algorithms, e.g. the Concentrate and TATRA proposed in [1], are typically complex in hardware implementation. Hence, practically feasible, approximate algorithms are used. These algorithms employ independent selectors (also referred to as arbiters [10]). A popular approximate algorithm is the Weight Based Arbiter (WBA), proposed in [1]. It was shown to be hardware implementable and close to Concentrate in performance. We focus on WBA in the rest of our discussion.

The main contribution of this work is an optimization of the WBA algorithm. We propose to use the WBA algorithm in a sequential and iterative manner to compute the scheduling configuration. Our proposed variations yield sizeable improvements in hardware area and clock speeds while achieving a performance close to the WBA.

1–4244–0983–7/07/$25.00

II. The Weight Based Arbiter

The motivation for the choice of the WBA was the search for a multicast scheduling algorithm that could be implemented in an FPGA—specifically, in a Xilinx Virtex II Pro XC2VP100-6FF1704—for a 64-port switch. This required a greater collective effort for the organization of the queued packets and a more broader perspective of distribution in the scheduling process. The WBA, proposed in [1] has specific advantages, such as: it is simple to implement in hardware, fair in the scheduling of traffic and achieves a high throughput. However, the primary advantage is that, it formed a practical, approximate alternative to the concentrate algorithm, with reduced hardware complexity.

A. Design and Operation

The WBA scheduler assigns weights to the incoming input cells, depending on their age and fanout at the beginning of every time slot. Once the weights are assigned, each output is assigned the heaviest cell. In case of multiple requests with the same weight, the scheduler breaks ties randomly. For fair operation, a positive weight is given to age while fanout is weighted negatively to maximize throughput. Thus, the older the cell, the heavier it is and larger the fanout, the lighter it is. Basing cell grants on age and fanout results in a compromise between extremes of pure residue concentration and strict fairness. The weight w_i of input i is computed as

$$w_i = f * \text{fanout}(\text{HOL}(i)) + a * \text{age}(\text{HOL}(i)), \qquad (1)$$

where f and a are the weights assigned to fanout and age, respectively. It can be shown that for an M×N switch, no cell waits at the HOL for more than $(M + f * N/a - 1)$ cell times [1]. By means of the weights f and a, the weight calculation can prioritize either age or fanout. In particular, if we assign equal weights to age and fanout, no cell waits at the HOL for more than $(M + N - 1)$ cell times.

WBA employs *fanout splitting*, meaning that a cell can be served over a course of multiple time slots. In every slot a subset of the remaining fanout is served. This is the opposite of *one-shot* scheduling, in which every cell must be served in a single slot. As fanout splitting achieves a significant performance benefit at a small cost, it is usually preferred [15], [12].

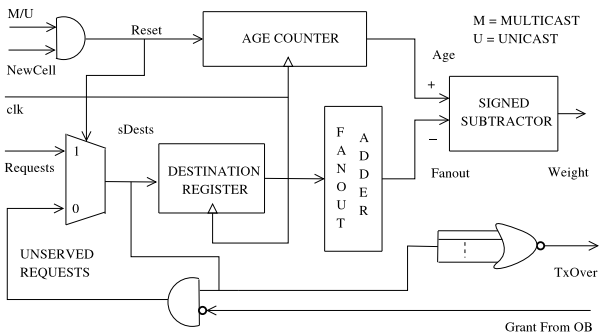

Fig. 1. 64 × 64 WBA Input Block (IB) Design Schematic.

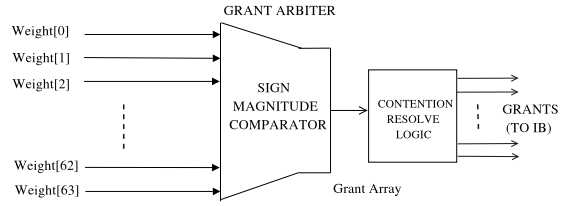

Fig. 2. 64 × 64 WBA Output Block (OB) Design Schematic.

Parallel operation of the WBA leads to two sections (or blocks) in the scheduler design:

- The input blocks (IB) for the weight computation.
- The output blocks (OB) for the weight comparison and grant selection.

The input block (IB): Fig. 1 shows the structure of an IB, which has to calculate the weight for each of the input cells based on their age and fanout. The age counter is reset with the advance of a new cell at the HOL, and is incremented in every time slot until the cell has been served completely. The fanout adder computes the cell fanout. The grants coming from the OBs are fed back to the IBs to update their age and fanout values for the next time slot.

The output block (OB): For a M×N switch, the output block has N comparators (corresponding to the N output ports). In every time slot, each comparator identifies the highest weight forwarded by the IBs and grants the input with the highest weight denying all other requests. See Fig.2.

B. Topology

To reduce implementation complexity, an input cell must wait in line (FIFO) until all the cells ahead of it have gained access to all of the outputs that they have requested. The topology of the broadcast algorithm is loop-back type and hence needs grant information sent from the OBs to the IBs.

III. Performance and implementation results

A. FPGA synthesis

The motivation for this synthesis is the implementation of a crossbar scheduler for a 64-port optical switch demonstrator (called OSMOSIS) with 40-Gb/s ports and a predefined scheduling cycle (time slot) of 51.2 ns for high performance computing applications [2][4][5]. A major challenge in this project is to implement a 64-port scheduler with this time slot duration in FPGAs, which were used for minimizing cost and improving the flexibility of the design. The FPGA synthesis results, shown in Table I, show that the 64-port WBA scheduler fits in the target FPGA device, nearly saturating it (84% full). The device used for implementation was "xc2vp100-6ff1704", a Xilinx Virtex-II Pro series FPGA with 8 M system gates (100 K logic cells†) and 1040 User IOs.

†Virtex logic cell = One, 4-Input LUT + One, Flip Flop + Carry Logic. One Virtex Slice = Two Virtex logic cells.

TABLE I
NxN WBA SCHEDULER – FPGA SYNTHESIS RESULTS IN XILINX VIRTEX-II-PRO[SPEED GRADE-6].

N	2	4	8	16	32	64
# slices	22	124	538	1995	8399	37024
% slices	0.05	0.28	1.22	4.52	19.05	83.96
Min. clock period (ns)	2.023	5.173	10.377	12.891	18.96	27.927
Max. clock freq. (MHz)	494.31	193.31	96.37	77.57	52.74	35.81

B. Performance simulations

The latency–throughput simulation results are shown in Fig.3. The mRRM (multicast Round Robin Matching) scheme is a simple multicast round robin arbiter granting the requesting cells in a round robin fashion. This is the simplest in implementation complexity but has poor latency–throughput performance. The Concentrate algorithm [1] gives the best performance score and the mRRM the worst. WBA has much lower implementation complexity than Concentrate, yet has comparable performance. On the other hand, WBA clearly outperforms mRRM.

IV. WBA Design Alternatives

A. Distributed WBA

The WBA has 64 IBs and 64 OBs in its core. This kind of a bulky hardware on a single chip introduces constraints on area and speed. An alternative to the WBA could be a distributed design where the IBs are kept outside the MC Scheduler FPGA which now has only the 64 OBs in it. The IBs are housed at the *test cards* from where the input requests to a particular output are generated. The weight computation is hence done at the backend and then the weights and requests are forwarded to the scheduler to make a decision. An issue with such an approach is the high latency and delays introduced by routing of large signal paths and also issues of IB-OB synchronization on different chips. To mitigate these drawbacks, we propose the sequential WBA design.

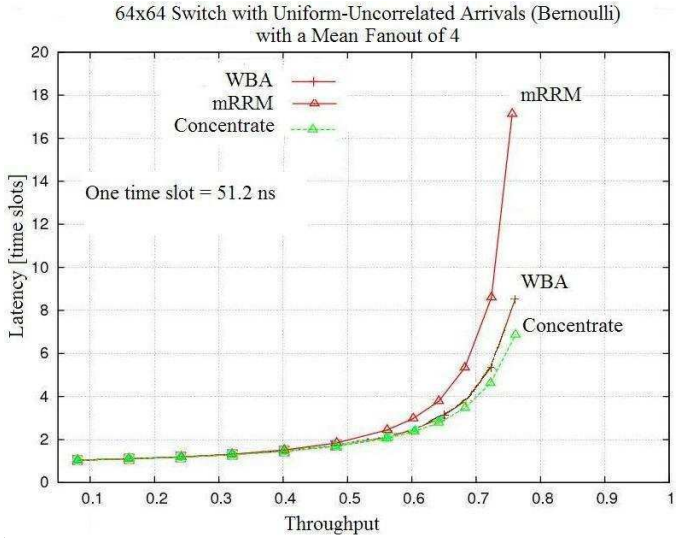

Fig. 3. Latency–throughput performance of the WBA scheduler.

B. Sequential WBA

The 64×64 WBA saturates the chip and has a delay of about 28 ns. The Multicycle WBA design shown in Fig.4 is a proposed alternative to the WBA. This kind of a sequential design works in multiple cycles of the auxiliary clock to generate the grant array which is then forwarded to the input blocks for weight calculations in the next cell cycle. The advantage in such an approach is that, instead of having 64 OBs in the scheduler there needs to be only one comparator which runs in multiple cycles. A further advantage is that these iteration cycles over which the WBA runs could be customized. But an additional hardware investment is in a further addition of registers and clocks to the combinatorial design.

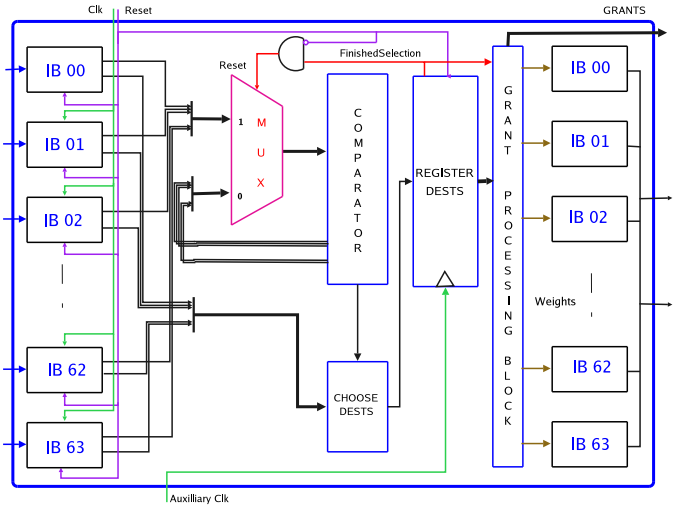

Fig. 4. 64×64 Multicycle-WBA Design Schematic.

V. Split Iterative Scheduling

The WBA computes matching for all the 64 OBs in a particular cell cycle, hence making it compeletely weight characterized. The split-iterative design provides a choice of iteration numbers, after which the matching is terminated and the incomplete grant array is combined with a parallel round robin matched array to generate a combined scheduling decision.

A. The WRRM

The Weighted Round Robin Matching (WRRM) algorithm has a design schematic shown in Fig.5. We catagorize such a design as a split iterative system. The IBs at the beginning of the scheduler, receive incoming traffic from the Head-Of-Line (HOL) of the FIFO queues. They then compute weights for the input ports and forward them along with the requests to the sole comparator at the nucleus of the design. This comparator

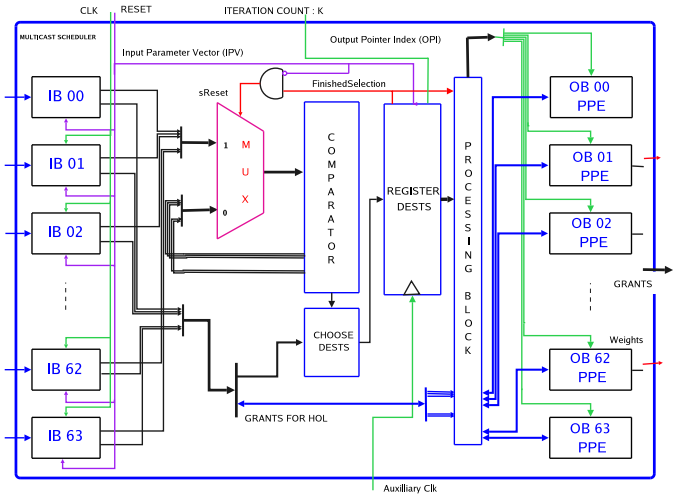

Fig. 5. 64×64 WRRM Design Schematic.

works similar to the sequential WBA completing the grant array vector over multiple cycles (iterations). The matching is terminated after a customized number of iterations. For example, if the matching is terminated after j iterations, the matching is termed as $WRRM, I = j$. The advantage in such an approach over the Multicycle-WBA is that all the iterations need not complete based on the weight computations alone. Once the highest j weights are identified and granted over j iteration cycles, a pointer pointing to the index of the latest granted weight is forwarded to the output blocks along with the partially completed grant array vector. This pointer is referred to as the Output Pointer Index (OPI). In general, a priority scale factor could be assigned to the weight, age and fanout independantly as p, f and a respectively. The parameters can hence be weighted accordingly based on the preference given to specific input ports. I in equation 2 is the index of the weight granted in the latest iteration cycle.

$$OPI = I[p_j \times f * \text{fanout(HOL}(j)) + a * \text{age(HOL}(j))] \quad (2)$$

The difference of this arbiter design from the Multicycle WBA is that, the outputs from the centralized comparator in this design are forwarded to a Round Robin Arbiter unlike in the Multicycle WBA design, where the outputs were fed back to the IBs and the scheduling was carried out over multiple cycles until the grant array is complete.

The OPI, along with the partially completed grant array and the unserved requests from the centralized comparator is received by each the output blocks at the tail end of the scheduler. These output blocks comprise of Programmable Priority Encoders (PPE), which run a round robin grant algorithm on the incoming requests. The OPIs serve as cursor prioritizers for the Round Robin PPEs at the output. This facilitates a quick parallel computation of grants in a round robin fashion.The partially completed grant vector from the Centralized Function Block (CFB), housing the comparator is completed in multiple clock runs of the auxiliary clock and the corresponding requests for the output are masked out. This masked out request array is the one which is forwarded to the PPEs for grant array completion. The completed array is used for scheduling.

In this design, the WBA and Round Robin Matching (RRM) capabilities are combined in a hybrid fashion leading to a robust and highly reconfigurable system of multicast scheduling. In the WRRM and the Multicycle WBA design, the system clock is split into two fast running clocks and the iteration cycles are customized.

VI. Hardware Implementation Results

A. FPGA synthesis

The implementation results of the proposed schemes are shown in Table II and Fig.6. The Multicycle WBA has a reduction of FPGA area (improvement of 24.52%) as the number of comparators in the WBA design are reduced from Sixty-Four in the conventional design to one in the Multicycle design. The clock speed improvement is also substantial. It is seen that there is a "*clock split* ". The 27.93 ns WBA clock is split into two clocks of 10.54 and 20.28 ns. They are referred to as the primary and the auxiliary clocks respectively. The auxiliary clock could be run a multiple number of times to complete the grant array in the multicycle design. In the WRRM implementation, the amount of hardware required is higher than the WBA (19.09% increase). This additional hardware component can be understood to have been included to incorporate the 64 PPEs in the output blocks of the design. Apart from the additional hardware investment, the WRRM design, provides more flexibility in traffic scheduling, where the WBA could be mixed with the RRM after a specific number of iterations, thereby providing a dynamic design operative. The WRRM has again two clocks – The primary and the Auxiliary, which have min. periods of 19.04 and 14.95 ns respectively. The Primary clock in the WRRM is slightly faster than in the Multicycle WBA (MC-WBA) because, in the MC-WBA the iterations are run through a combinatorial loop until the grant array is completely matched.

B. Performance simulations

Fig.7 shows the latency vs throughput characteristics of the proposed design schemes. The data plotted here is for a 64×64 scheduler, with uniform-uncorrelated arrivals (Bernoulli traffic) with a mean fanout of Four. In the graph, WRRM,I=j stands

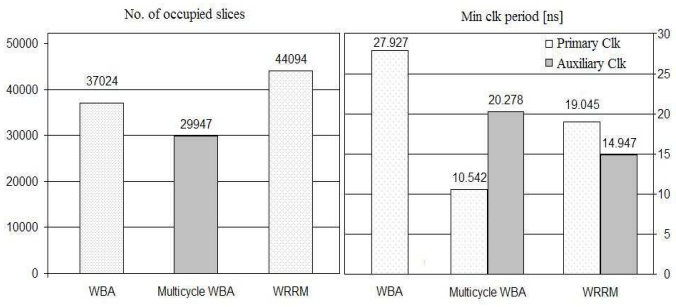

Fig. 6. FPGA area and clock speeds for a 64×64 scheduler.

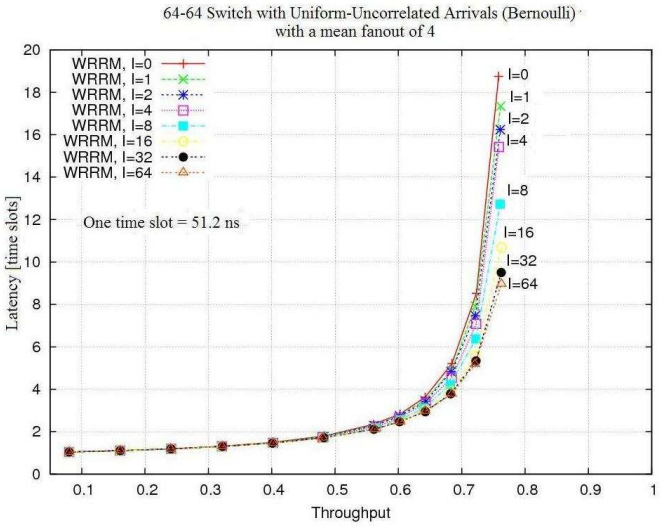

Fig. 7. Latency vs Throughput performance of the Age-Fanout WBA Schemes.

for WRRM scheduling of multicast traffic (WBA+RRM) with j iterations. After j iterations, the scheduler passes on the incomplete grant array along with the unserved requests to be completed by Round Robin Matching in the PPEs. In the figure, $j = 0$ stands for pure Round Robin Matching (RRM) and $j = 64$ stands for pure WBA. Between these extremes, the performance characteristics of the customizable scheduler with j iterations, $0 < j < 64$, is shown to lie. As the number of iterations are increased from 0 towards 64, there is a sizeable improvement in the latency of the scheduler. For eg. this kind of an improvement, may be sufficient after only 4 iterations in a specific application. Whence, the generic hardware design of the scheme could be customized to handle only 4 WBA iterations. After about 8 WBA iterations, the characteristics of the WRRM design are acceptably close in performance to the WBA. The hardware investment pointed above is further eclipsed by this design flexibilty and performance enhancement. Note that in the graph of Fig.7, the y-axis is in terms of time-slots (cell-cycles), each of which stand for 51.2 ns as explained in sec.II

VII. Conclusion

The WBA scheme proposed in [1] for scheduling of multicast cells was synthesized on the Xilinx Virtex II Pro FPGA. The scheduler had an area occupancy of 83.96% and a minimum clock period of 27.927 ns for a 64×64 switch. Improvement of clock speeds by $15 - 20\%$ and making the design more flexible led us to investigate newer and novel scheduling techniques - the Sequential WBA and the WRRM. The main contribution of this paper is to propose customizable scheduling policies and study the trade-offs in such a design approach. It is shown here that FPGA area reductions (of upto 24.52%) are attainable in the sequential designs with respect to the WBA. A little added complexity to integrate the WBA and the RRM is shown to acheive flexible scheduler designs with customizable number of iterations. An acceptable trade-off between performance and complexity makes them practical arbitrator choices in contemporary scheduler designs.

TABLE II

64×64 multicast scheduler – FPGA synthesis results in Xilinx VirtexII Pro. MC-WBA = Multicycle WBA, Aux = Auxiliary Clk

64 × 64	# slices	% slices	Min clk period ns		Max clk speed MHz	
			Primary	Aux	Primary	Aux
WBA	37024	83.96%	27.93	-	35.81	-
MC-WBA	27947	63.38%	10.54	20.28	94.86	49.31
WRRM	44094	83.96%	19.04	14.95	52.51	66.90

Acknowledgment

The author would like to thank François Abel and Cyriel Minkenberg, IBM Research GmbH, Zurich Research Laboratory, CH-8808, Rüschlikon, Switzerland. This work was performed while the author was at the IBM Zurich Research Laboratory. OSMOSIS is a contract for IBM and Corning to develop optically-switched interconnects for supercomputers and sponsored by the Department of Energy, USA.

References

[1] B. Prabhakar, N. McKeown, R. Ahuja, *Multicast scheduling for input queued switches* IEEE Journal of Select Areas in Communication, vol.15, no.5, pp.855–866, June 1997.

[2] R.P. Luijten, C. Minkenberg, B.R. Hemenway, M. Sauer, and R. Grzybowski, *Viable opto-electronic HPC interconnect fabrics* Proceedings of the CM/IEEE SC2005 Conference on High-performance Networking and Computing, p.18, Seattle, WA, USA, November 12-18 2005.

[3] T.T. Lee, *Non-Blocking copy networks for multicast packet switching* IEEE Journal of Select Areas in Communication, vol.6, no.5, pp.1455-1467, Dec 1988.

[4] B.R. Hemenway, R.R. Grzybowski, C. Minkenberg and R.P. Luijten, *An optical packet-switched interconnect for supercomputer applications* OSA Journal of Optical Networks, vol.3, no.12, pp.900–913, October 2004.

[5] C. Minkenberg and F. Abel, *Designing a Crossbar Scheduler for HPC Applications* IEEE Micro, vol.26, issue 3, pp.58–71, May-June 2006.

[6] J.S. Turner, *Design of a broadcast switching network* Proceedings of the IEEE INFOCOM, pp.667–675, 1986.

[7] M. Karol, M. Hluchyj, and S. Morgan, *Input versus output queueing on a space division switch* IEEE Transactions on Communication, vol.35, no.12, pp.1347–1356.

[8] M. Andrews, S. Khanna, and K. Kumaran, *Integrated scheduling of unicast and multicast traffic in an input-queued switch* Proceedings of the IEEE INFOCOM, pp.1144–1151, 1999.

[9] S. Gupta and A. Aziz, *Multicast scheduling for switches with multiple input queues* Proceedings of Hot Interconnects, pp.28-33, 2002.

[10] N. McKeown, *iSLIP scheduling algorithm for input-queued switches* IEEE Transaction on Networks, vol.7, no.2, pp.188–201, Apr 1999.

[11] N. McKeown, *A fast switched backplane for a gigabit switched router* Business Communication Review, vol.27, no.12, 1997.

[12] J.Y. Hui and T. Renner, *Queueing analysis for multicast packet switching* IEEE Transactions on Communication, vol.42, no.234, pp.723–731, Feb 1994.

[13] E. Schiattarella and C. Minkenberg, *Fair integreated scheduling of unicast and multicast traffic in an input queued switch* Proceedings of the IEEE ICC, Istanbul, Turkey, 2006.

[14] A. Bianco, E. Leonardi, F. Neri, C. Piglione, and P. Giaccone, *On the number of input queues to efficiently support multicast traffic in input queued switches* Proceedings of the IEEE HPSR, pp.111–116, Jun 2003.

[15] J.F. Hayes, R. Breault, and M. Mehmet-Ali, *Performance analysis of a multicast switch* IEEE Transactions on Communication, vol.39, no.4, pp.581–587, Apr 1991.