RZ 3665 (# 99665) 06/01/2006
Electrical Engineering 8 pages

# Research Report

## An Input Queueing Implementation for Low-Latency Speculative Optical Switches

Raj Krishnamurthy and Peter Müller

IBM Research GmbH
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

# An Input Queueing Implementation for Low-Latency Speculative Optical Switches

*Raj Krishnamurthy, Peter Müller*

IBM Research GmbH
Rüschlikon, Switzerland
email: rajk_gt@ieee.org, pmu@zurich.ibm.com

## ABSTRACT

Switching fabrics using speculative scheduling are emerging as a promising way to tackle the latency problem in optical computer interconnects. This paper presents an input queueing architecture and hardware to process regular request/grant and speculative arbitration in an integrated fashion. The OSMOSIS research demonstrator co-built by Corning and IBM is a 64 port optoelectronic switch with a Broadcast & Select optical switching fabric and central arbiter, running at 40 Gbps per port. The demonstrator is built using Xilinx Virtex II Pro and Altera Stratix FPGAs. A speculative protocol bypasses the latency of the control channel by transmitting a cell from input port adapters to the optical switching fabric, ahead of the corresponding grant from the central scheduler. This paper discusses various input queueing architecture alternatives. FPGA sizing and speed results from synthesis of the selected queueing architecture to Altera Stratix are presented. A novel dual-tree round-robin queue scheduler hardware structure is proposed and evaluated to overcome the poor mapping properties of priority-arbiter schedulers to FPGAs. The dual-tree structure outperforms the conventional priority-arbiter structure by a factor of two when integrated in a queueing architecture.

## 1. INTRODUCTION

Bandwidth gains have dominated latency improvements in many domains of the computer system [1]. For processors, memory, network and disk, in the time that bandwidth doubles, latency improves only by a factor of 1.2 to 1.4 [1].

Latency is critical for computer interconnect networks. This is particularly relevant for high-performance computers where cache misses to local memory must be served by fetching cache blocks from remote memory. The OSMOSIS (*O*ptical *S*hared *M*em*O*ry *S*upercomputer *I*nterconnect *S*ystem) effort is a joint research project between Corning and IBM to leverage optics for construction of low-latency computer interconnects [2, 3]. The program is supported by the US Department of Energy. Optics can be beneficial for high performance computers because of (i) lower power and (ii) the ability to cover long distances. These factors are particularly relevant for large supercomputer installations. The OSMOSIS interconnect fabric tackles the latency problem using four novel provisions -

- Technology. The OSMOSIS switching fabric uses Corning SOAs (Semiconductor Optical Amplifiers) that can now switch in the range of a few nanoseconds.
- Protocols. The OSMOSIS switching fabric uses speculative protocols. Under certain conditions, cells can bypass the latency of a conventional request-grant cycle and directly proceed for transfer across the switch fabric [4]. More details are provided later in this paper.
- Switch Scheduler. The OSMOSIS switch uses a central scheduler FLPPR (Fast Low-latency Parallel Pipe lined Arbitration) that can match input ports and output ports with low-latency operation [2].
- Switch Architecture. The switching fabric supports ports with dual receivers. If there is contention for an output port in a given cell slot, cells can still pass through to the output port, with cells directed separately to each receiver.

The OSMOSIS prototype demonstrator is built using a combination of Altera and Xilinx FPGAs. The reader is referred to [3] for photographs of prototype chips and boards. The Xilinx Virtex II Pro FPGAs form the core of the central scheduler, while link queues and interfaces are built using Altera Stratix FPGAs. This allows the demonstrator to be used as a programmable research vehicle and limits ASIC NRE costs.

**Salient Features of the OSMOSIS Switching Fabric** The OSMOSIS switching fabric element is shown in Figure 1. The switching element uses a bufferless optical core. Broadcast is an important capability of a parallel computer interconnect. The fabric has native support for broadcast, as switching is supported through a broadcast-and-select optical fabric. The OSMOSIS fabric supports 2048 nodes using a fat-tree in two-levels with 96 switch elements in each

This work was in part supported by the University of California under contract B527064

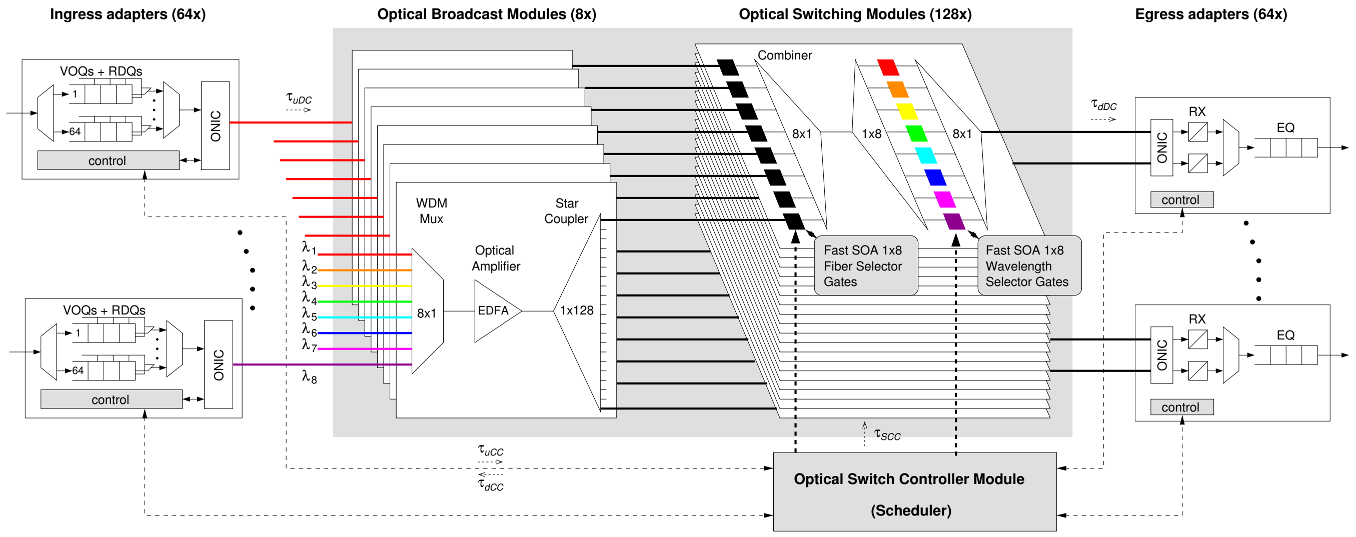


**Fig. 1**. OSMOSIS System Architecture

level. There are at most three hops between any two nodes. Each switch element has 64 ports with each port operating at 40 Gbps, built using SONET/SDH PHY components. There are 64 ingress port adapters and 64 egress port adapters.

The optical cell is 2048 bits long with a cell time of 51.2 ns. A central scheduler or arbiter matches input ports to output ports using a novel switch scheduling scheme called FLPPR [2, 5]. This allows a matching to be completed every cell-time of 51.2 ns. Cell arrival is communicated to the central scheduler using a control channel link shown in Figure 1, built using 2.5 Gbps Infiniband PHY components. This means that the input adapters must provide an O/E conversion. Grants are returned by the central scheduler when matchings are computed and cell transfer is possible. Cells are then transmitted to the switching fabric by ingress adapters, while the central scheduler activates SOAs for timely switching action. The current FPGA switch prototype latency is estimated at $\approx 1.2\mu$s (with 280 ns for full-custom ASIC).

Each switch element uses FEC (Forward Error Correction) in the port adapters along with acknowledgements to provide a BER (Bit Error Rate) of less than $10^{-21}$. The dual receiver capability to bypass receiver contention means that each OSMOSIS switching element is an asymmetric switch, with 64 input and 128 output ports. The 128 output ports are arranged in 64 egress adapters. Various optical and electronic sub-components are being rigorously tested and integration is underway.

**Queueing and Speculation** Consider an input port adapter with backlogged queues. Upon cell arrival, a descriptor is queued in VOQs (Virtual Output Queues) and a regular arbitration scheduling request is made to the central scheduler using the control channel. If in a cell-slot, the control channel is idle or a control channel cell does not contain the result of a prior arbitration, the OSMOSIS port adapter speculatively transmits an optical channel cell to the B&S (Broadcast-and-Select) fabric. Simultaneously, it makes a speculation request to the central arbiter. Note that this action bypasses the control channel latency or request-grant scheduling cycle. If the central arbiter can accommodate the speculative request, it returns an ACK (acknowledgement) or replies negatively with a NAK (negative acknowledgement). The central arbiter correlates this request with regular scheduling matchings, with priority given for regular arbitrated requests. An ACK means that the cell can be dequeued from the input port adapter, as it has been already transferred. A NAK means that optical cell must wait for a GRANT. OSMOSIS simulation studies have shown that speculation is effective for light to medium loads [5, 4]. Specifically, with an OCF (Oldest-Cell-First) scheduling policy at the input port adapters, the OSMOSIS speculative protocol completely eliminates the control path latency for loads upto 50%. The reader is referred to [4] for analytical modeling and simulation details.

The focus of this paper is the OSMOSIS input adapter queueing system. This paper considers various architectures to process regular arbitration (with requests/grants) and speculation in an integrated fashion. This is described in Section 3. Section 4 evaluates the scaling properties of the selected architecture with area and clock-rate synthesis results. A critical aspect of speculation is the choice of queue selection or queue scheduling algorithm. This paper develops scheduling hardware structures that map well to Altera Stratix FPGAs. This is also described in Section 4. Section

5 concludes the paper.

## 2. RELATED WORK

A number of VOQ queueing architectures and reference implementations have been developed by both Xilinx and Altera for their chips. Speculative schemes for pipelined routers were introduced in [6], but do not provide queueing structures or any evaluation with implementation technologies. [7] describes a distributed bus arbiter and analyzes the shortcomings of the centralized programmable priority encoder in [8] for round-robin arbitration. Both [7] and [8] describe ASIC implementations only. Link scheduling is exhaustively covered in [9]. The costs of implementing service-tag based schedulers is evaluated with Xilinx Virtex II FPGAs in [9]. The OCF (Oldest-Cell-First) queue scheduler described in this paper is a service-tag based scheduler with 19-bit service-tags (timestamps), but uses a comparator tree. The architecture in [9] uses a single-stage recirculating shuffle-exchange network for optimally mapping a range of scheduling algorithms. The comparator blocks in [9] compute complex decision rules and not simple comparisons as described in this paper.

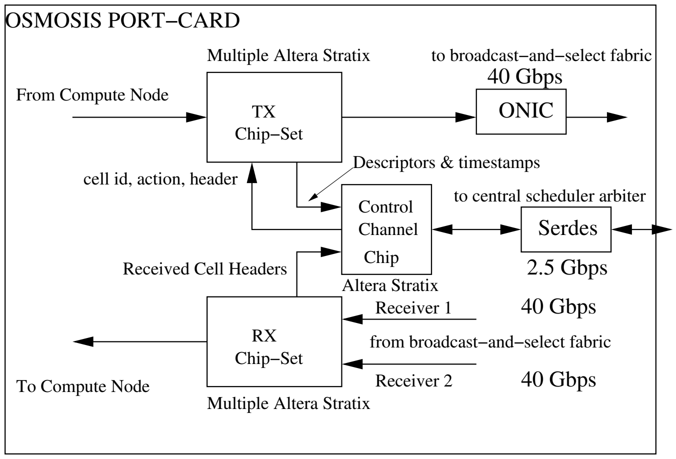


**Fig. 2**. OSMOSIS Port-card Architecture

## 3. QUEUEING ARCHITECTURE

Cells are queued in port-card adapters in transmitter (TX) datapath memory as shown in Figure 2. Cell descriptors are provided to the control channel chip also shown in Figure 2. The control channel chip contains a queueing engine that issues requests and receives responses from the central arbiter. The receiver (RX) datapath receives cells from other port-cards in the switching fabric. This section describes the architecture of the port-card, the control channel chip and the queueing engine.

### 3.1. Port-card Architecture

An OSMOSIS port-card consists of a transmitter (TX) chip-set, receiver (RX) chip-set and control channel chip. The TX chip-set consists of multiple Altera Stratix chips with interfaces to a 40 Gbps optical link. We used Stratix GX chips to allow use of the GX transceivers to interface with the 40 Gbps optical link through an ONIC (Optical Network Interface). The control channel chip is an Altera Stratix EP1S80 chip with a physical interface to a 2.5 Gbps optical control channel link. The RX chip-set consists of multiple Altera Stratix chips to handle two receiver interfaces. This allows upto two cells to be received at a port-card receiver from other port-card adapters in the same cell-slot.

A cell is received in the TX chip-set and a 38 bit descriptor (6 bit port id, 10 bit cell id, 1 bit virtual lane id and a 21 bit timestamp) is provided to the control channel chip for queueing. Bits from the 10 bit cell id are used as sequence numbers for reliability purposes. The control channel chip provides a 43 bit (32 bit cell header, 10 bit cell id and 1 bit control signal) to the TX datapath for cell transmission. This happens when a grant arrives from the central scheduler or when the opportunity for speculative transmission exists. This is explained in detail in Section 3.3. Cells from other port-card adapters are received in the dual receiver datapath. 35 bit received cell descriptor (32 bit header and 3 bit control signals) are provided to the control channel chip for constructing acknowledgement cells.

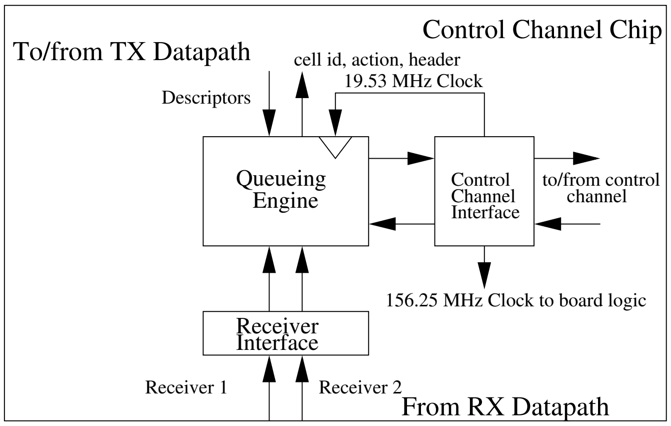


**Fig. 3**. OSMOSIS Control Channel Chip Architecture

### 3.2. Control Channel Chip Architecture

The control channel chip consists of a queueing engine, an RX (receiver) interface and a control channel interface. This is shown in Figure 3. All synchronization is distributed from a high precision clock on the central arbiter using the control channel. A phase-synchronized reference frequency (156.25

MHz, 8 clock cycles per cell) is distributed with a 51.2 ns cell start signal. The control channel interface consists of a transmit and receive FIFO that transmits and receives 96 bit control channel cells. Additionally, the control channel interface is also responsible for timing functions using a synthesizer PLL, a real time counter register (RT), a configuration unit and an RT delay analysis unit. The configuration unit presets the synthesizer PLL and the real-time counter based on programmable settings or the results of RT analysis. The RX interface processes received cell descriptors so that acknowledgement cells can be constructed for received optical data channel cells. The queueing engine receives optical channel cell descriptors for queued cells in the transmit datapath. The queueing engine also provides optical channel cell descriptors to the TX datapath for cells that have been granted passage through the optical fabric.

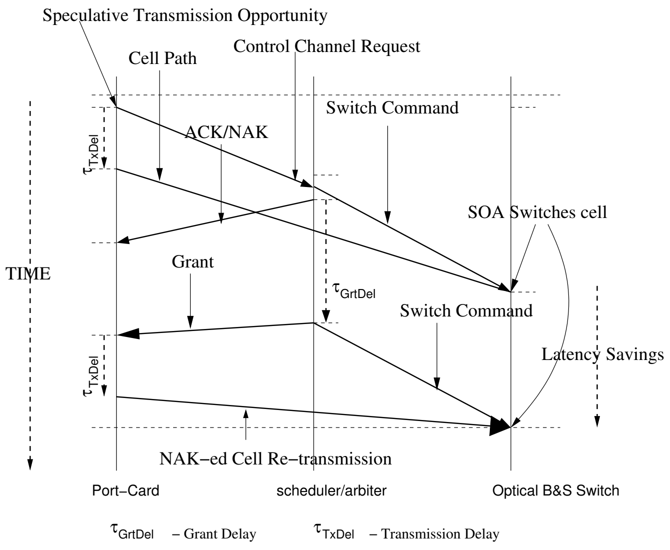


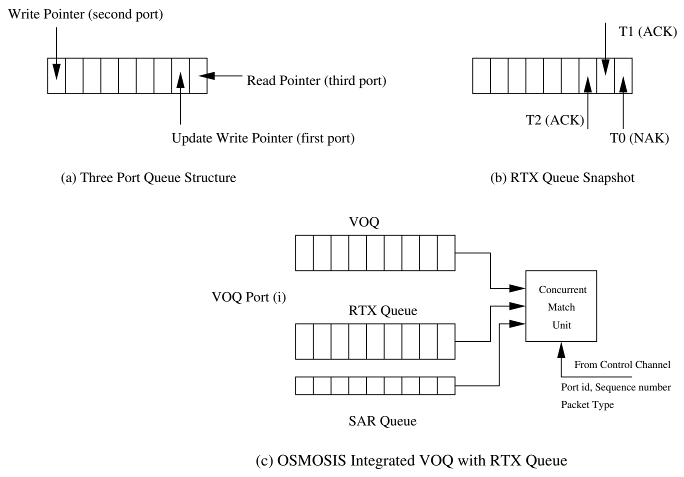
**Fig. 4**. OSMOSIS Control Channel Protocol with Speculation Actions

## 3.3. Queueing Architecture

This section describes the design choices and tradeoffs in the development of the OSMOSIS queueing architecture. The control channel protocol is summarized first, followed by a discussion of the design choices and tradeoffs.

### *3.3.1. Control Channel Protocol*

The control channel supports regular request-grant arbitration and speculative requests in an integrated fashion. The speculation actions in the control channel protocol are summarized in Section 1 and shown graphically in Figure 4. The OSMOSIS control channel cell format allows regular arbitration requests and speculative requests to be issued in the same cell slot from same or different queues. The reader is referred to [4] for further details of the protocol.



Write Pointer (second port)
Read Pointer (third port)
Update Write Pointer (first port)
(a) Three Port Queue Structure
T1 (ACK)
T2 (ACK)
T0 (NAK)
(b) RTX Queue Snapshot
VOQ
VOQ Port (i)
RTX Queue
SAR Queue
Concurrent Match Unit
From Control Channel
Port id, Sequence number
Packet Type
(c) OSMOSIS Integrated VOQ with RTX Queue

**Fig. 5**. Architecture Alternatives

### *3.3.2. Architecture Alternatives*

A Virtual Output Queue (VOQ) queues packets in FIFO order according to packet destination ports. There is a VOQ for every destination port. A VOQ for a port *i* can be realized by a write pointer for queueing and a read pointer for dequeueing. This maps well to Altera Stratix FPGAs that have embedded RAM memory with dual ports. This means that queueing and dequeueing can be performed concurrently in the same clock cycle. In order to support regular arbitration and speculative transmission in an integrated fashion, the following architectures are considered.

**Integrated VOQ with three memory access ports.** When a cell arrives, a descriptor is queued using a write port. When the opportunity for a speculative transmission exists, a second write pointer is used to set a bit (stx_bit) in the queue entry and the corresponding cell is speculatively transmitted. A read pointer is used to dequeue a cell. Figure 5 (a) shows this queue architecture with two write pointers and one read pointer. If one port is reserved for the read pointer, then the second port available in Altera embedded RAMs can be shared by the write pointers. If the read port clock frequency is *f*, then each of the write pointers must operate at *2f* to share the remaining memory port. When the speculative transmission is acknowledged, the stx_bit is reset. We chose not to use this architecture because of the complexity in maintaining multiple clock domains for read ports and write ports.

**Integrated VOQ with separate RTX queue.** A simpler architecture to realize regular arbitration and speculative transmission in an integrated fashion is to use an RTX (speculative retransmission) queue along with each VOQ. The RTX queue holds cell descriptors that have been speculatively transmitted. Additionally, a cell that does not succeed in speculative transmission is also held in the RTX queue. Unsuccessful cells are transmitted again when a regular grant for the cell eventually arrives. There is an RTX queue and a VOQ for each port in the switch. When a cell arrives, a descriptor is placed in the VOQ corresponding to the destination output port. If the control channel does not carry any result of a prior arbitration request, then the head-of-line descriptor corresponding to a cell in a VOQ can be speculatively transmitted. When this happens, the head-of-line cell descriptor is dequeued and placed in the RTX queue. A copy of this descriptor is also provided to the TX datapath for transmission. When a grant arrives, the port corresponding to the grant is extracted along with a sequence number. A check is first made in the RTX queue and then in the VOQ corresponding to the extracted destination port. If there is a match of the port id and sequence number in the RTX queue, the head-of-line descriptor is dequeued and the cell is transmitted. If the grant precedes a positive acknowledgement (ACK) then a duplicate cell might be delivered to the destination output port. This is because a cell that waits for an ACK or NAK has already been speculatively transmitted[4]. The OSMOSIS prototype provides suitable logic to eliminate any duplicate cells in the output. If the extracted queue id and sequence number does not exist in the RTX queue, then the corresponding VOQ is checked. A match in the VOQ leads to the cell descriptor being dequeued and the cell transmitted by the TX datapath. The OSMOSIS queueing system matches extracted port ids and sequence numbers with the VOQ and RTX queue in the same cycle using a concurrent match unit (CMU) shown in Figure 5 (c). The RTX queue is given precedence over the corresponding VOQ using a priority encoder.

Consider the RTX queue snapshot in Figure 5 (b). The descriptor in the first position receives a NAK, the descriptor in the second position receives an ACK and the descriptor in the third position also receives an ACK. A FIFO is used for the RTX queue, instead of a linked list to allow simple hardware complexity. On receiving the NAK, the descriptor in the first position cannot be dequeued as it still needs to be transmitted upon receipt of a grant. When the next ACK is received, it cannot be matched with the second descriptor because of the first entry in the FIFO ahead of it. To allow this situation, all ACK/NAK entries are stored in a single-bit FIFO called Speculative Arbiter Response (SAR). The NAK resets the first entry with bit 0 corresponding to the head-of-line RTX cell descriptor. When the subsequent ACKs arrive, the successive bits are set with 1. Each entry is left in the RTX queue until the corresponding grants arrive. Note that each cell arrival results in a regular scheduled arbitration request and will be matched with a grant from the central arbiter. When the first grant arrives, the first descriptor is dequeued and provided to the TX datapath for transmission as it was previously NAK-ed. When the next grant arrives, a bit value of 1 in the corresponding SAR queue means that the corresponding cell was transmitted successfully by speculation. This means that the descriptor can be simply dequeued and any action for the grant is not needed.

We have used a simple 1-bit SAR queue instead of duplicating the RTX queue to hold any NAK-ed entries in a separate NAK queue. The VOQ and RTX are dimensioned to hold cell descriptors corresponding to the current switch prototype. The current latency estimates for the switch are about 1.224 $\mu$s with a cell slot of 51.2 ns, yielding an RTT of about 50 cell slots. We have dimensioned the queues for 64 cell descriptors.

The integrated VOQ structure with the RTX queue (Architecture B) requires more memory bits than the three-port VOQ structure (Architecture A). Architecture A requires memory port sharing domains with higher hardware complexity. We chose to implement Architecture B as we store descriptors in our queues and each queue is dimensioned to not exceed 64 entries. Additionally, each queue is realized using simple FIFO structures that map well to Altera Stratix dual-port RAMs.

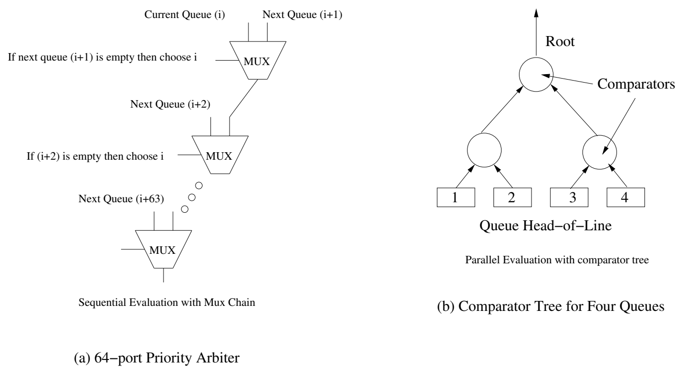

Current Queue (i)
Next Queue (i+1)
If next queue (i+1) is empty then choose i
MUX
Next Queue (i+2)
If (i+2) is empty then choose i
MUX
Next Queue (i+63)
MUX
Sequential Evaluation with Mux Chain
(a) 64−port Priority Arbiter
Root
Comparators
1
2
3
4
Queue Head−of−Line
Parallel Evaluation with comparator tree
(b) Comparator Tree for Four Queues

**Fig. 6**. RTX Queue Scheduler Architectures

**RTX Queue Scheduler** When the opportunity for a speculative transmission exists, the RTX queue scheduler selects an active queue for speculative transmission. A number of policies are possible, for example, round-robin, random, youngest-cell-first (YCF) and oldest-cell-first (OCF). We are currently experimenting with scheduling policies and their correlation with traffic patterns. A round-robin scheduler is implemented as a priority encoder and is shown in Figure 6 (a). The round-robin priority encoder is work-conserving and arbitrates only between active queues. The OCF and YCF scheduling policies require timestamps with each descriptor entry and are implemented using comparator trees and shown in Figure 6 (b).

## 4. PERFORMANCE EVALUATION

This section evaluates the hardware complexity of the queueing architecture with various RTX queue scheduling policies.

### 4.1. Synthesis Methodology

We developed the designs in a mix of behavioral and structural VHDL. The designs were synthesized and placed & routed using the Quartus 4.2 tool. All designs requested speed based optimization from the Quartus tool. The designs were mapped to Altera Stratix EP1S80 chips, in which each LE (logic element) has a 4-input LUT, register and carry chain, with support for dynamic single bit arithmetic.

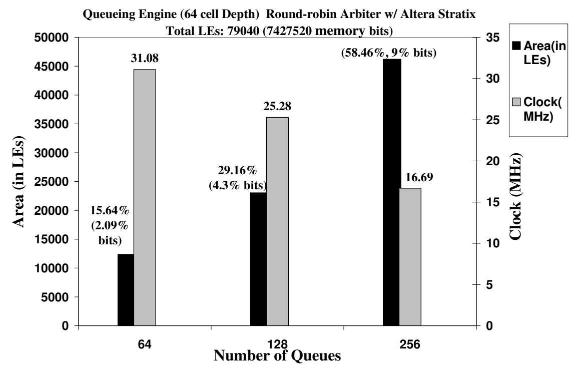


**Fig. 7**. Area-Clock Rate Characteristics (Altera Stratix)

### 4.2. Discussion of Implementation Results

Figure 7 shows area and clock-rate results for the queueing architecture with 64, 128 and 256 ports. The timing macro, serdes interface and RX interface macros were not included in the design for synthesis. The queue depth in each case was held constant at 64 cells, equal to the RTT of the switching system. The numerals in brackets over the area columns are the percentage of memory bits used by each design point. We see a linear increase in area used by each point. The number of memory bits used also increases linearly. There is an almost 50% drop in clock-rate from 64 ports to 256 ports. All designs were synthesized using a work-conserving round-robin priority arbiter. These results are consistent with expectation.

Figure 8 shows area-memory results for a 64 port queueing architecture arrangement. The queue depth was varied from 32 elements to 128 elements. The designs were synthesized using a round-robin priority arbiter. As the queues

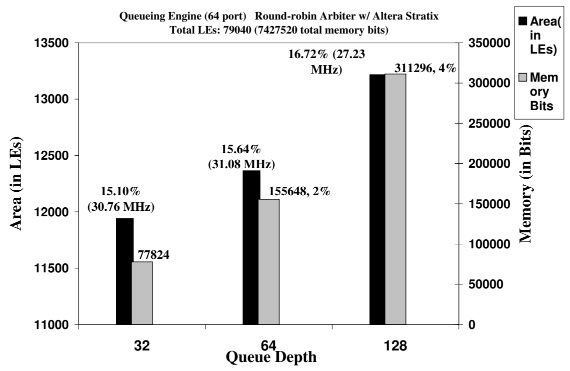


**Fig. 8**. Area-Memory Characteristics (Altera Stratix)

were mapped to embedded RAM, increasing the queue depth affects logic area marginally. The extra logic is in the form of logic elements or wires needed to cover more memory elements. The FIFO queues support a counter which counts upto the size of the FIFO and can be read at every clock cycle. This structure grows in size with queue depth. The clock-rate is also largely unaffected. We see a linear increase in the number of memory bits as queue depth increases. This is because queue elements are mapped to embedded RAM and do not take any logic element resources. This is consistent with expectation. The synthesis tool reported that our priority encoder muxes were not restructured during synthesis.

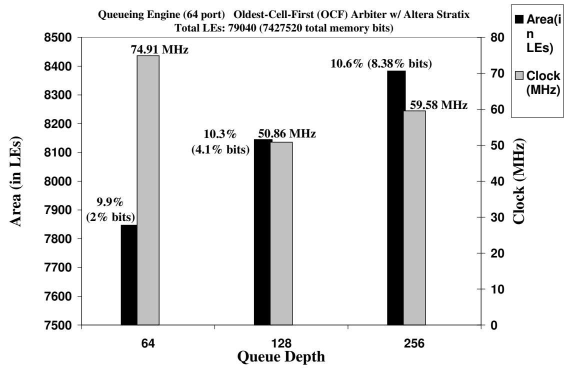


**Fig. 9**. Area-Clock Rate Characteristics (Altera Stratix)

Figure 9 shows area-clock rate results from synthesis of a 64 port queue architecture using an oldest-cell-first (OCF)

RTX queue scheduling policy. The queue depth is varied from 64 elements to 256 elements. The OCF scheduling policy is implemented in structural VHDL using a comparator tree. There are six levels in this tree. Each comparator compares two 19 bit values. A six bit port id and a 13 bit timestamp are used for comparison. Priority encoders use FPGA MUX structures that can span multiple logic elements and do not map very efficiently. In the case of a comparator tree, 19 bit buses map easily to the rich interconnect wiring of the FPGA. Additionally, comparators can be realized very easily in LUTs. Thus, comparator trees map well to the Altera Stratix FPGA structure. By comparing corresponding datapoints in Figure 8 and Figure 9, we see that designs with the OCF scheduler achieve twice the clock-rate of a round-robin priority-arbiter based design. This is can be directly attributed to the reduced hardware complexity of the OCF scheduling policy. This is because other hardware elements between the OCF and round-robin designs are the same.

## 4.3. Improving Round-Robin Scheduled Queue Architectures

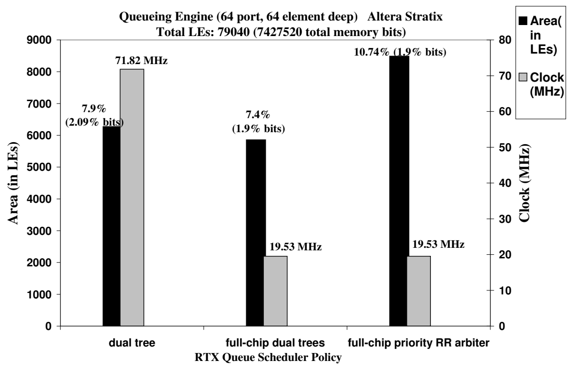


**Fig. 10**. Area-Clock Rate Characteristics (Altera Stratix)

An insight from evaluation experience with OCF scheduling policy designs is that realizing round-robin arbiters using comparator trees might result in better clock-rate performance. Round-robin arbitration for a set of 64 queues can be expressed as dual comparator trees in the following manner. $i$ is the current queue position of the RTX queue scheduler. For a non-existent active queue set (no cells in any queues), the $min$ function returns a zero.

$1 \leq i \leq 64,\ 0 \leq top\_set \leq 64,\ 0 \leq bottom\_set \leq 64$
$0 \leq I \leq 64$

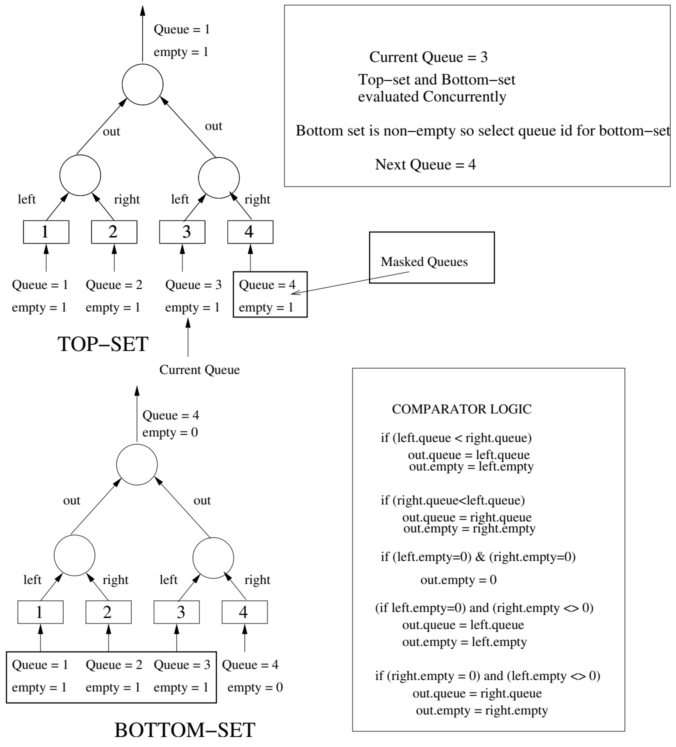


**Fig. 11**. Dual Comparator Tree Round-Robin Arbiter

$$\begin{aligned} bottom\_set &= min(N),\ i < N \leq 64 \\ top\_set &= min(N),\ 1 \leq N \leq i \end{aligned}$$

$if\ (bottom\_set = 0)\ then\ (I = top\_set)\ else\ (I = bottom\_set)$
$if\ (I \neq 0)\ then\ (i = I)\ else\ (i\ is\ unchanged)$

The intuition for the above expression is that a priority arbiter must determine the first active queue that succeeds the current position. This is performed by examining the queues succeeding the current queue position first. If there is no active queue, the preceding queues are examined. Thus, a priority arbitration operation can be expressed as two comparator trees - one for the preceding queues (top_set) and the other for the succeeding queues (bottom_set). The hardware implementation evaluates the *top_set* and *bottom_set* concurrently and chooses the appropriate result. The queueing architecture comparators propagate 6 bit queue ids and 1 bit empty flags forward. For a comparison between two inactive queues, the comparator operation yields the numerically lower queue id and an empty flag set to one (empty queue). If this empty value appears at the root of the tree, the other comparator tree root value is checked. If both trees yield a root value with empty flag set, then the current queue id is left unchanged. The top_set and bottom_set trees are both dimensioned for 64 elements at the leaf nodes (queues). The top_set comparator tree simply loads leaf nodes (queues) succeeding the current queue position with empty flag set (inactive). This masks those queues from the comparator tree action. Similarly, the bottom_set tree loads values preceding and including the current position with empty flag set to one, effectively masking those queues out. Queue ids and empty flags succeeding the current scheduler position are loaded without any change. This method allows a 64 queue tree structure that can be used by both the top_set and bottom_set structures for comparison. This is shown in Figure 11 for the case of four queues.

Figure 10 shows area-clock rate results from synthesis of a dual-tree implementation of a round-robin priority arbiter. The queueing architecture has 64 ports and is 64 elements deep. The dual-tree architecture shows dramatically higher performance (over 2x) over the corresponding round-robin design. An interesting observation is that the dual-tree design shows an equivalent area as that of the corresponding OCF design. The clock rate is a little bit lower. The OCF design uses one 19 bit input comparator tree, while the dual-tree architecture uses two trees with 7 bit comparator inputs. The hardware complexity is comparable.

Figure 10 also shows the final area results from incorporating the queueing architecture in the OSMOSIS control channel chip. We did not include the RX interface for synthesis. The OSMOSIS control channel chip uses a 19.53 MHz clock domain for the queueing architecture. The synthesis tool compacts area as it is able to achieve the 19.53 MHz clock frequency with ease. This is observed for the dual-tree queue scheduler and also for the round-robin priority arbiter. For the clock-domain of 19.53 MHz, a dual-tree implementation for round-robin arbitration saves about 3% of chip area than the priority round-robin arbiter. The dual-tree round-robin implementation outperforms the priority arbiter round robin implementation by over a factor of 2 when synthesized for speed.

**Performance Comparison.** Although the hardware architecture in [9] uses a different architecture than the dual-tree scheduling algorithm in this paper, we compare our results with the results in [9]. The scheduling algorithm in [9] for service-tag based scheduling disciplines achieves a clock rate of 72.97 MHz with Xilinx Virtex II and 32 queues. The service-tag is 16 bits wide. The OCF architecture with 19-bit service-tags for OCF arbitration in this paper achieves 74.91 MHz for 64 queues. Note that OCF arbitration and service-tag arbitration are essentially the same operation. Each requires a winner queue to be determined.

## 5. CONCLUSION

New queueing structures are needed to support the emerging class of speculative optical switching fabrics. This paper considered various queueing architecture alternatives and their implementation complexity. The queueing architecture developed in this paper integrates regular request/grant switching protocols and speculative protocols in the same queueing structure. The integrated queue is mapped to Altera Stratix embedded RAM and scales well. Speculative queue scheduling with round-robin priority arbiters and OCF (Oldest-Cell-First) scheduling is considered, and their impact on area and speed is evaluated. Round-robin priority arbiters scale poorly because of their use of multiplexer chains. A dual comparator-tree scheduling architecture is proposed that preserves the round-robin priority arbitration action. This queue scheduling structure outperforms the multiplexer-chain based priority arbiter by over a factor of two in speed. Finally, FPGA sizing and speed results for the queueing engine incorporated in the OSMOSIS control channel chip are presented. The control channel chip and link are functional and are being integrated into the OSMOSIS research prototype.

## 6. REFERENCES

[1] D. Patterson, "Latency Lags Bandwidth,," *Communications of the ACM*, vol. 47, no. 10, pp. 71–75, Oct. 2004.

[2] C. Minkenberg, F. Abel, P. Muller, R. Krishnamurthy, M. Gusat, and B. R. Hemenway, "Control path implementation for a low-latency optical hpc switch," in *HOTI '05: Proceedings of the 13th Symposium on High Performance Interconnects*, 2005, pp. 29–35.

[3] R. Luijten, C. Minkenberg, R. Hemenway, M. Sauer, and R. Grzybowski, "Viable opto-electronic hpc interconnect fabrics," in *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, 2005, p. 18.

[4] I. Iliadis and C. Minkenberg, "Performance of a speculative transmission scheme for arbitration latency reduction," IBM Research, http://domino.research.ibm.com/library/cyberdig.nsf, Tech. Rep. RZ3650, 2006.

[5] C. Minkenberg, I. Iliadis, and F. Abel, "Low-latency pipelined crossbar arbitration," in *IEEE Globecom 2004*, 2004.

[6] L.-S. Peh and B. Dally, "A delay model and speculative architecture for pipelined routers," in *In 7th High-Performance Computer Architecture (HPCA '01)*, 2001, p. 255.

[7] E. S. Shin, I. Vincent J. Mooney, and G. F. Riley, "Round-robin arbiter design and generation," in *ISSS '02: Proceedings of the 15th international symposium on System Synthesis*, 2002.

[8] P. Gupta and N. McKeown, "Designing and implementing a fast crossbar scheduler," *IEEE Micro*, vol. 19, no. 1, 1999.

[9] R. Krishnamurthy and S. Yalamanchili, "Sharestreams: A scalable architecture and hardware support for high-speed qos packet schedulers," in *In Field-Programmable Custom Computing Machines (FCCM)*, April 2004, pp. 115–124.