Saturn: A Terabit Packet Switch Using Dual Round-Robin

Jonathan Chao, Polytechnic University

ABSTRACT

Large input-output buffering with a moderate speedup has been widely considered as the most feasible solution for large-capacity switches. We propose a new terabit per second packet switch and call it the Saturn switch. It uses a simple dual round-robin arbitration scheme to schedule packets, and achieves high throughput and low statistical delay bound. It employs a bit-sliced crossbar fabric to switch packets at 10 Gb/s at inputs and outputs, and adopts a novel token-tunneling technique to arbitrate contending packets at high speed (e.g., within 10 ns), thus achieving a switch capacity of more than 1 Tb/s with existing electronic technology.

INTRODUCTION

The virtually unlimited bandwidth of optical fibers has caused tremendous increase in the speed of data transmission over the past decade, and hence stimulated high-demand gigabit per second multimedia services such as distance learning and videoconferencing that will undoubtedly be part of our lives in the new century. The Internet, together with its robust and reliable Internet Protocol (IP), is widely considered the most reachable platform for next-generation information infrastructure. The challenge to the success of the Internet lies in the deployment of terabit per second packet switches to meet the exponential growth of multimedia and Internet traffic while providing quality-of-service (QoS) support.

Fixed-length switching technology is widely accepted to achieve high switching efficiency. Variable-length packets are segmented into fixed-length cells[1] at inputs and are reassembled at outputs. Much research effort, including the Weighted Fair Queuing (WFQ) family (e.g., [1, 2]), has been devoted to packet scheduling at outputs to support fair bandwidth sharing that provides delay bounds for regulated traffic. Switches are said to be ideally output buffered when all cells arriving at inputs would be immediately forwarded to outputs. In other words, the speed of the internal switch fabric and the output port memory would have to be large enough to accept all possible input cells. If the ratio of the internal bandwidth to the input bandwidth is defined as the speedup factor, the purely output buffering has a speedup factor equal to the switch size, say N. This is infeasible for a terabit per second switch. In contrast, purely input buffering has a speedup factor of one. It suffers from head-of-line (HOL) blocking, whose non-work-conserving property causes difficulty in providing bandwidth guarantees. Therefore, a moderate speedup factor between 1 and N is usually adopted in designing large-scale switches. More than one arbitration is allowed in a time slot to increase the overall arbitration efficiency, which can be further improved with iterative matching to schedule cells from inputs to outputs.

Recently, we observed several methods perfectly emulating purely output queuing under a moderate speedup factor (2–4) so that ideal packet scheduling can be realized at outputs (e.g., [3–5]). They consider the states of output packet scheduling the arbitration priority, and iterative stable matching is needed to ensure perfect emulation. While these might be the future choice for perfect scheduling and providing delay bounds, its time complexity of at least $O(N)$ matching iterations is infeasible with existing electronic technology for a terabit per second switch. Together with some sorting time required to emulate the desired fair queuing, the total time budget can be as large as implementing N simple arbitrations. The enormous state maintenance and large amount of state information exchange between inputs and outputs also make it impractical to implement perfect emulation of fair queuing with stable matching.

From a more practical point of view, the arbitration should be separated from the output packet scheduling to keep the implementation and time complexities reasonable. Although perfect emulation of output queuing cannot be realized and no delay bound can be achieved absolutely, delay control is still attainable in the statistical sense: the portion of cells with an undesired delay is bounded by an acceptable probability. Relaxing the delay bound requirement from absolute to statistical should not cause significant performance degradation because, even if the delay bound is absolutely guaranteed, some cells may still be lost due to buffer overflow and other reasons.

[1] *The term cell is adopted from asynchronous transfer mode (ATM) in the sense of fixed length, but the cell used here does not necessarily have the same format and length as the ATM cell.*

0163-6804/00/$10.00 © 2000 IEEE

Mean burst length	Average cell delay									
	POQ ($c = N$)	$c = 2$			$c = 3$			$c = 4$		
		Input	Total	% over POQ	Input	Total	% over POQ	Input	Total	% over POQ
$b = 1$	4.50	0.26	4.62	2.67	0.02	4.50	0.00	0.003	4.50	0.00
$b = 10$	85.2	12.1	96.2	12.8	0.34	85.3	0.14	0.031	85.2	0.00
$b = 50$	442.2	71.8	509.5	15.2	1.84	443.2	0.22	0.16	442.3	0.023

Table 1. *Comparisons of average cell delay under various speedup factors and traffic burstiness.*

To support distributed switching control, we proposed the dual round robin (DRR) arbitration scheme [6] in which input selection and output contention resolution are separately handled by two independent sets of round-robin arbiters. Among the virtual output queues (VOQs) maintained at each input, a cell is selected in a round-robin manner to be the request for output contention resolution. The selected cell keeps contending until winning a token, and then the next cell is selected. Compared with first-in-first-out (FIFO) input queuing, DRR scheduling reduces the destination correlation of the cell arrival sequence for output contention resolution, and thus significantly improves the throughput and delay performance for bursty traffic.

Another challenge to building a large-capacity switch lies in the stringent arbitration time constraint to resolve output contention. The architecture described in [7] is an input-buffered crossbar switch with centralized contention resolution, which does not scale well for a large number of switch ports due to the centralized nature of its arbiter. Traditional arbiters handle all inputs together, and the arbitration time is proportional to the number of inputs. As a result, the switch size or capacity is limited given a fixed amount of arbitration time. We propose a novel *token tunneling* arbitration scheme for output contention resolution that is implemented in a distributed manner. This is a variation on the ring reservation method proposed in [8] and provides fairness. The arbitration time of the ring reservation method is proportional to the number of switch ports. With the token tunneling technique, it is possible to reduce the arbitration time to the order of the square root of the port number. The ring reservation method proposed in [8] is implemented using sequential logic, whereas our token tunneling arbitration is implemented with combinational logic that makes it even faster. Our design has delay in the basic arbitration unit comparable to the bidirectional arbiter described in [9], but our overall arbitration delay is much smaller because of the token tunneling method. Furthermore, our design needs only two pins per output port, compared to six in theirs. Crossbar chips are generally pad-limited; therefore, the number of pins required per port determines the number of ports that can be accommodated in a single chip.

The rest of this article is organized as follows. We will summarize the pros and cons of various approaches to delay control with speedup, and highlight why statistical delay control should be considered. We will describe the DRR arbitration scheme and highlight its advantages in performance and implementation complexity. We will demonstrate our design of a terabit per second packet switch using token tunneling and bit-slice techniques, which is called the switch at terabit using dual round-robin (Saturn)[2] switch. We then summarize our work and highlight some future research directions.

Delay Control with Speedup

Using speedup with input-output queuing is widely accepted as the most feasible solution to building large-scale switches. There are two different meanings in the literature when talking about a speedup factor of c. In the first one, the switch fabric runs at c times the speed of input and output ports [10]. A time slot is further divided into c cycles, and cells are transferred from inputs to outputs in every cycle. An input can transmit and an output can accept c cells in a time slot. In the second meaning, an output can still accept c cells in a time slot, but during the same period at most one cell can be transferred from an input [11]. The switch fabric does not need to run c times faster. We call this approach *output expansion* and to the first as *speedup*.

We observed some analytical studies on the switch throughput and cell average delay for output expansion under bursty traffic model [11, 12]. A factor of 2 only achieves 82.8–88.5 percent throughput depending on the degree of input traffic correlation (burstiness) [12]. In the bursty traffic model, each input alternates between active and idle periods of geometrically distributed duration. During an active period, cells destined for the same output arrive continuously in consecutive time slots. The probability that an active or idle period will end at a time slot is respectively fixed.

Under the same traffic model, however, the simulation studies for the first speedup approach implicitly show that a speedup factor of 2 yields 100 percent throughput [10]. Table 1 summarizes the average cell delay with speedup factors from 2 to 4 and under various bursty traffic after 2 billion cells are simulated in a 256 x 256 switch with FIFO inputs and outputs and round-robin arbitration. From the table we see that the delay performance when the speedup factor is 3 or more is nearly indistinguishable from that of purely output queuing. "% over POQ" is the ratio of the input queuing delay of input-output buffered switches to the output queuing delay of purely output queuing (POQ) switches.

We observed three different approaches to achieving delay bound in the input-output buffered switches in recent literature. The most

[2] *Another reason it is called the Saturn switch is that the switch has multiple token rings circulating around the switch fabric, similar to the planet Saturn, which also has multiple rings around it.*

DRRM is adopted in the SATURN switch. An input arbiter at each input selects a non-empty VOQ according to the round-robin service discipline. After the selection, each input port sends one request, if any, to an output arbiter.

Mean burst length	ϵ	$MDB(\epsilon) = \min\{D: Pr[W_{in} > D] \le \epsilon\}$			Mean delay with POQ
		Speedup = 2	Speedup = 3	Speedup = 4	
$b = 1$	10^{-3}	3.7	0.8	0.2	4.5
	10^{-6}	9.0	2.6	1.3	
$b = 10$	10^{-3}	145	18	5	85.2
	10^{-6}	311	66	25	
$b = 50$	10^{-3}	817	95	23	442.2
	10^{-6}	2024	384	133	

Table 2. *Comparing statistical input delay bounds with the average output queuing delay.*

straightforward idea is to incorporate fair queuing into the arbitration [2]. It has been shown that, with a speedup factor of 6, the cell delay can be bounded for each individual flow that is leaky-bucket regulated, and the bound is independent of other flows and switch size [5]. However, such arbitration involves enormous state maintenance and transactions among all flows over the switch, which seems too much to be practical. Besides, a term of the bound obtained is proportional to the ratio of the burstiness and the flow rate [5], and thus could be very large for low-rate flows.

The second idea is to emulate an input-output buffered switch with speedup as a purely output queued switch by specially scheduling cells from inputs to outputs such that the cell departure sequence from each output is identical to the emulated purely output queued switch [3]. The emulated switch of the desired service discipline (e.g., WFQ) is required to run in the background for calculating the departure timestamp (or the *time-to-live* value) and then setting a priority value for each cell. Based on the priority values, an iterative stable matching algorithm is used to schedule cells from inputs to outputs in every arbitration cycle. The time complexity of the iterations, to the best of our knowledge, is $O(N)$ [4], and it is unclear if an algorithm with such high time complexity can be implemented in real time at high speed.

In the third approach, a quasi-static scheduling algorithm is adopted [13]. Part of the path scheduling between inputs and outputs is predetermined (e.g., for the group of QoS-assured traffic), so delay control at the input stage is separated from that at the output stage. The predetermined scheduling will be adjusted when there is a big change in traffic statistics. However, the deduction of bandwidth efficiency due to the static nature of scheduling is unavoidable.

All the above approaches seek an absolute delay bound for each flow. However, the delay requirement in the network is usually expressed in percentile. Let W be the delay of a cell. Then the statistical delay requirement is expressed in terms of a tail probability as follows:

$$\Pr[W > D] \le \epsilon,$$

where D is the acceptable delay (we call it a *statistical delay bound*), and ϵ is the desired level in percentile. For instance, ϵ could be as small as the cell loss rate (e.g., 10^{-6}). If $\epsilon = 0$, D becomes an absolute delay bound.

With a moderate speedup factor (e.g., 2–4), less backlog is accumulated at inputs than at outputs. Table 2 summarizes the statistical input delay bounds comparable with the average delay under purely output queuing, where the minimum delay bound (MDB) is defined as $MDB(\epsilon) = \min\{D: \Pr[W_{in} > D] \le \epsilon\}$.

A Dual Round-Robin Arbitration Scheme

In the DRRM scheme we proposed previously [6], each input port has N VOQs. DRRM is adopted in the Saturn switch. An input arbiter at each input selects a nonempty VOQ according to the round-robin service discipline. After the selection, each input port sends one request, if any, to an output arbiter. An output arbiter at each output receives up to N requests, chooses one of them based on the round-robin service discipline, and sends a grant to the winner input port. Because of the two independent round-robin arbiters, we call the arbitration scheme dual round-robin (DRR) arbitration.

DRR arbitration has four steps in a cycle:

- Each input arbiter performs request selection.
- The input arbiters send requests to the output arbiters.
- Each output arbiter performs grant arbitration.
- The output arbiters send grant signals to input arbiters.

Figure 1 shows an example of the DRR arbitration algorithm. In a request phase, each input chooses a VOQ and sends a request to an output arbiter. Assume input 1 has cells destined for both outputs 1 and 2. Since its round-robin pointer, r_1, is pointing to 1, input arbiter 1 sends a request to output 1 and updates its pointer to 2. Let us consider output 3 in the grant phase. Since its round-robin pointer, g_3, is pointing to 3, output arbiter 3 grants input 3 and updates its pointer to 4.

With iSLIP [7], each VOQ in the input buffer can send a request to an output arbiter. In other words, each input can send up to N requests to N arbiters, one for each. After grant arbitration, an input may receive multiple grants, and another round of arbitration is needed to guarantee that at most one cell is selected in each input port. A cycle of iSLIP arbitration consists of five steps:

- Input ports send multiple requests to the output arbiters.
- The output arbiters perform grant arbitration.
- The output arbiters send grants to input arbiters.

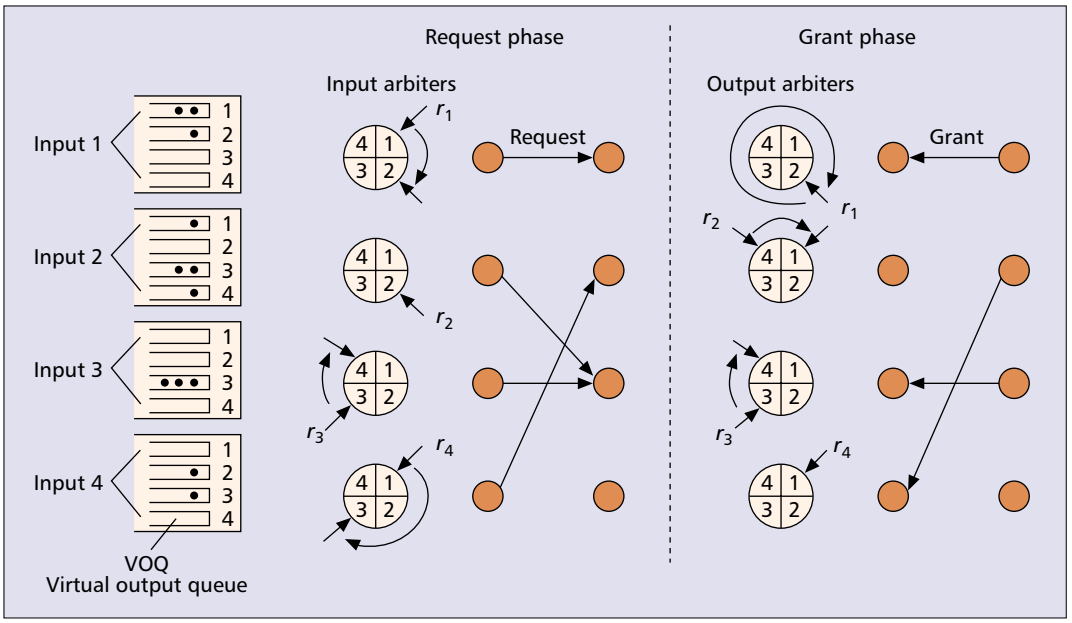

Figure 1. *An example of the dual round-robin scheduling algorithm.*

- The input arbiters perform another arbitration to solve the problem of possible multiple grants.
- The input arbiters send accept signals to output arbiters.

Similar to iSLIP, the DRR scheme also has the desynchronization effect, and thus achieves 100 percent throughput under random and uniform traffic. The input arbiters granted in different time slots have different pointer values, and each requests a different output, resulting in desynchronization. However, the DRR scheme requires less time to do arbitration and is easier to implement because less information exchange is needed between input and output arbiters.

Consider the fully loaded situation in which every VOQ always has cells. Figure 2 shows the HOL cells chosen from each input port in different time slots. In time slot 1, each input port chooses a cell destined for output A. Among those three cells, only one (the first in this example) is granted and the other two have to wait at HOL. The round-robin pointer of the first input advances to point to output B in time slot 2, and a cell destined for B is chosen and then granted because of no contenders. The other two inputs have their HOL cells unchanged, both destined for output A. Only one of them (the one from the second input) is granted; the other has to wait until the third time slot. At that time, the round-robin pointers among the three inputs have been desynchronized and point to C, B, and A, respectively. As a result, all three cells chosen are granted.

Figure 3 shows the tail probability under FIFO+RR (FIFO for input selection and RR for round-robin arbitration), DRR, and iSLIP arbitration schemes. The switch size is 256, and the average burst length is 10. DRR's and iSLIP's performances are comparable at speedup of 2, while all three schemes have almost the same performance as speedup $c \geq 3$.

Note that, in [15], Logical Equivalence of

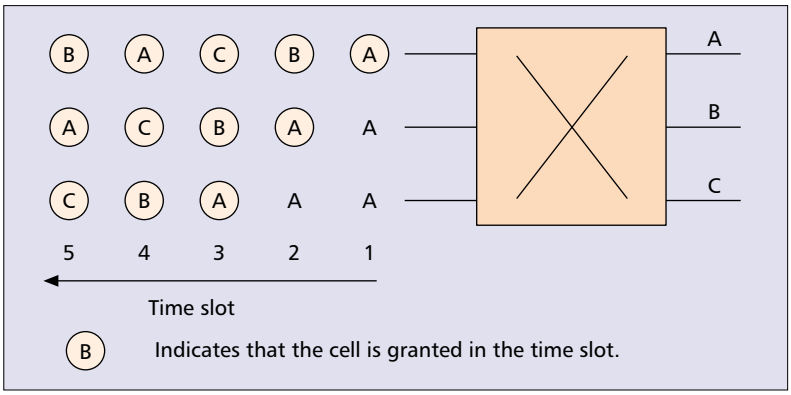

Figure 2. *The desynchronization effect of DRRM under the fully loaded situation. Only HOL cells at each input are shown for illustration.*

Parallel Iterative Matching (LE-PIM) was proposed for the modeling of a PIM algorithm. LE-PIM is similar to DRR in that at each iteration, an input sends the request (if any) to an output only. The difference between LE-PIM and DRR is mainly in resolving contentions: the former uses random selection, while the latter uses round-robin selection.

Switch Architecture with Token Tunneling

Figure 4 shows the Saturn switch, consisting of input port controllers (IPCs), output port controllers (OPCs), and multiple switch planes. Each IPC and OPC has buffers to temporarily store cells. The former holds cells that lost contention to other inputs. Because of internal speedup, multiple cells can arrive, in each cell time slot, at an IPC, but only one can depart to the output link; a buffer is thus required. Each switch plane is a crossbar structure, where N is the switch size and n is the number of ports in

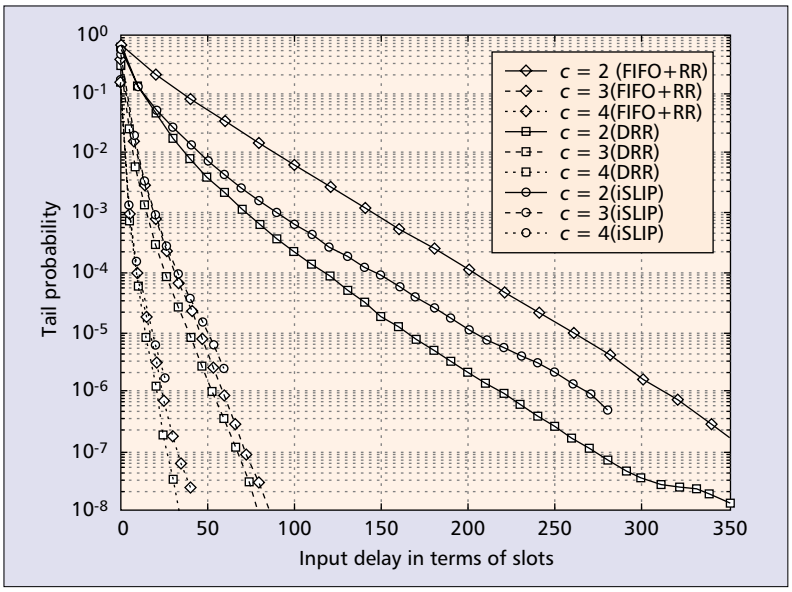

Figure 3. *Comparison on tail probability of input delay under three arbitration schemes.*

each crosspoint chip (XPC). Implementation of a crossbar switch fabric with a large number of ports within a single chip would be an ideal solution in designing a high-capacity crossbar switch. However, this is prevented by pin count and power consumption limitations of each chip. By using the bit-slice technique with multiple switch planes (e.g., 4 bits in each plane), the switch operation speed is reduced and can be implemented with low-cost complementary metal oxide semiconductor (CMOS) technology.

As shown in Fig. 4, a switch plane consists of a matrix interconnection of XPCs. Each XPC consists of a matrix interconnection of crosspoint units (XPUs). The switch handles multicasting by sending multicast patterns (MPs) and cells in parallel to the switch fabric. The MP is a bit map with each bit corresponding to each output port. If a bit at the *i*th position is set to 1, the cell is destined to output port *i*. When more than one bit of the MP are set to 1, the cell is multicast to multiple output ports. Cells are first stored in the VOQs at each IPC and transmitted to the switch fabric when they are granted through the arbitration cycle. There are $N + 1$ VOQs, one for each output port and one for multicasting cells. The switch fabric operates at a higher rate than the line rate to reduce HOL blocking and improve delay/throughput performance, as shown earlier.

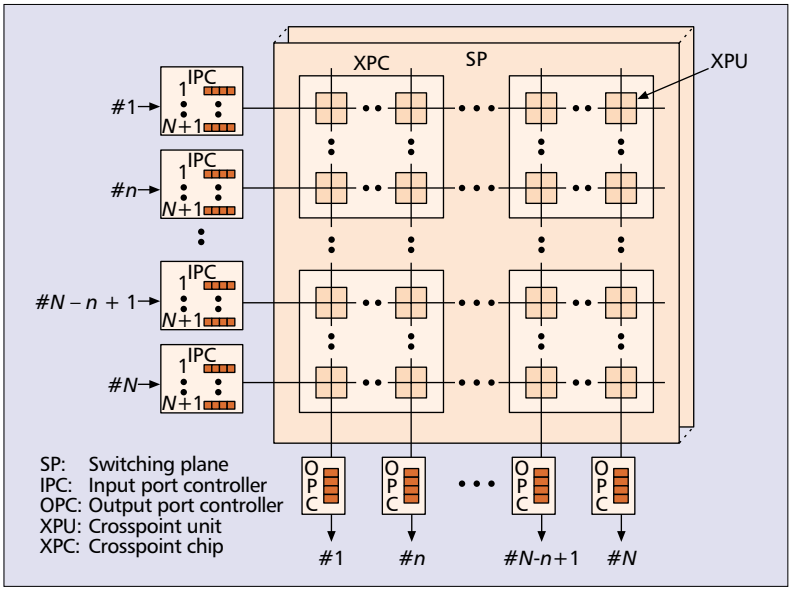

Figure 4. *Saturn switch architecture with multiple switch planes.*

At the beginning of an arbitration cycle, the MP is first loaded into the corresponding XPU row by the IPC, and each XPU holds an MP bit (i.e., the request bit). When there are more than one MP bits in a column set to 1, it means there are more than one cell contending for the same output port. Only one of them can be served in every arbitration cycle. The arbitration is done by all XPUs in the same column in a distributed and highly parallel fashion, thus achieving high speed and scalability. Once winner cells are determined, the XPCs acknowledge the winning IPCs through the handshaking signals between the XPCs and IPCs.

Token Ring Arbitration

XPUs solve the contention among inputs in a distributed manner. Inputs are served in a round-robin (RR) fashion, and each XPU column has its RR pointer. Token-in (*ti*) and token-out (*to*) interconnections in an XPU column form a ring, as shown in Fig. 5. The XPU pointed to by a column's RR pointer generates a *token* (shown by a black circle) at the beginning of each arbitration cycle and sends it down the ring. The XPU that has an MP bit of 1 and is closest to the generated token grabs the token, removes it from the ring, resets its MP bit to 0, and becomes the contention winner XPU of that arbitration cycle. The winner XPU then routes a cell and generates the token in the next arbitration cycle. The XPU that generates the token can win the contention only if all other MP bits in its column are 0 and its MP bit is 1. The position of the RR pointer remains the same if there are no requests in the arbitration cycle. Figure 5 shows an example of passing a token in a ring for arbitration. The XPU in row two is pointed by the RR pointer at the beginning of arbitration cycle *i*, and thus generates a *token*. The XPU in row three is transparent to the token since its MP bit is 0. After passing through the XPU in row three, the token is grabbed by the XPU in row four since its MP bit is 1. That XPU removes the *token* from the ring and resets its MP bit to 0. It then routes the winner cell in the next arbitration cycle, achieving the parallelism of cell transmission and arbitration. The XPU in the fourth row generates the token at the beginning of arbitration cycle $(i + 1)$ since it won the contention in arbitration cycle *i*. The XPU in the fifth row grabs the token and becomes the new contention winner.

A typical arbitration cycle $(i + 1)$ consists of input arbitration, MP loading, output arbitration, and handshaking, as discussed earlier. The input arbitration selects a cell to transmit among the $N + 1$ VOQs. The MP loading takes n bit times to load n MP bits to each XPC. The output arbitration grants the winner cell among N competing inputs. Handshaking is used to signal the IPC whether it is granted so that its RR

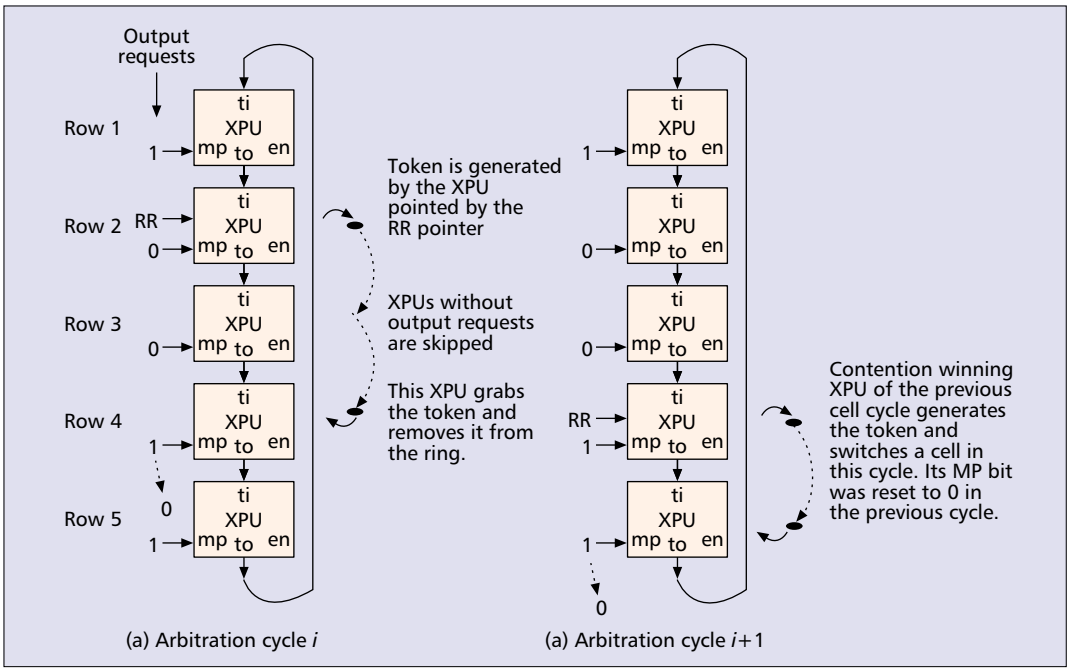

Figure 5. *An example of the token-ring arbitration.*

pointer among the $N + 1$ VOQs can move to the next one. It also tells the IPC to load a new MP if all MP bits in the row are zero. Note that arbitration cycle $(i + 1)$ is overlapped with cell transmission cycle i for which the cell wins the contention in arbitration cycle i.

The *Token Tunneling* Method

Here we propose a novel token tunneling mechanism to speed up the arbitration process. As shown in Fig. 6a, when all of the MP bits stored in a column of XPUs in an XPC are 0, the whole column in the XPU can be skipped by tunneling the token from the input of the XPC to its output directly. Arbitration time thus becomes proportional to the number of ports of an XPC, instead of the number of ports of the entire switch fabric.

The worst case time complexity of the basic token tunneling method is $D_t = 4n + 2(N/n - 2)$ gates delay. This occurs when there is only one MP bit with a value of 1 in an XPU column and it is at the farthest position from the RR pointer. For example, the MP bit is at the bottommost XPU, while the RR pointer points to the topmost one. As mentioned earlier, each XPU contributes two gates delay for arbitration. The token ripples through all the XPUs in the XPC where the token is generated and all the XPUs in the XPC where it is terminated, contributing the $4n$ gates delay. There are a total N/n XPCs in each column, and at most $(N/n - 2)$ XPCs will be tunneled through, contributing the $2(N/n - 2)$ gates delay.

By tunneling through smaller XPU groups of size g and having a *hierarchy* of these groups as shown in Fig. 6b, it is possible to further reduce the worst case arbitration delay to $D_t = 4g + 5d + 2(N/n - 2)$ gates delay, where $\lceil d = log_2(n/g) \rceil$. The hierarchical method basically decreases the time spent in the XPC where the token is generated and in the XPC where the token is terminated. For $N = 256$, $n = 16$, and $g = 2$, the basic token tunneling method requires 92 gates delay, whereas the hierarchical method requires only 51 gates delay.

Terabit Switch Design

For a 256 x 256 Saturn switch with the incoming aggregated bandwidth of 5 Gb/s and internal speedup of 2, the line bandwidth of the switch fabric is 10 Gb/s. The total switch capacity is 5 Gb/s x 256, or 1.28 Tb/s. The cell length can be

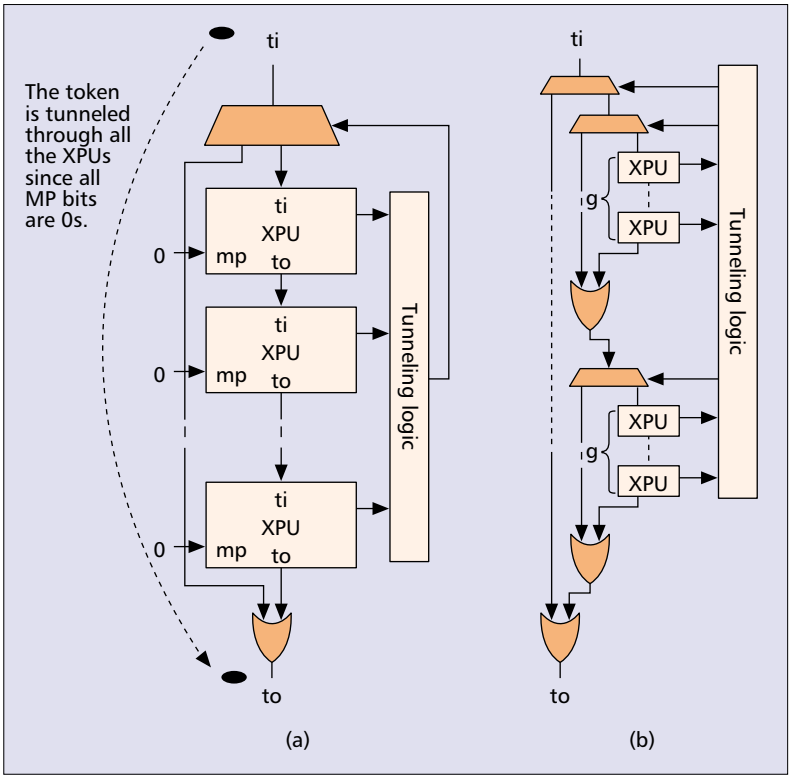

Figure 6. *The token tunneling scheme.*

When all of the MP bits stored in a column of XPUs in an XPC are 0, the whole column in the XPU can be skipped by tunneling the token from the input of the XPC to its output directly.

chosen to be 64 bytes to accommodate the smallest IP packet size (40 bytes). The switch fabric has four switch planes, as shown in Fig. 4. Assuming each XPC accommodates 16 ports, a switch plane has $(256/16)^2$, or 256 XPCs. In other words, the entire switch fabric with four switch planes needs 1024 XPCs. Obviously, if more ports can be accommodated in an XPC (e.g., 32), the total number of XPCs in each plan is reduced to 64. However, the XPC's pin count is proportionally increased, which may be prohibited due to high packaging cost and power consumption.

By choosing the data bus of the XPC to be 4 bits wide (i.e., $k = 4$), the data bus of the switch fabric is 4 x 4, or 16, bits wide. Thus, the operation speed of the data bus is 10 Gb/s/16, or 625 Mb/s, and the duration of each cell is 512 bits/16 or 32 bits with a bit time of 1.6 ns (1/625 Mb/s). Let us assume the time spent for input and output arbitration is identical since both are performing the same arbitration scheme with almost the same number of input requests ($N + 1$ vs. N). It takes 16-bit time to load a 256-bit MP, while it only takes one bit time to send the handshaking signals to the IPC. So there are about 15 bit times, or 15 x 1.6 = 24 ns, to do two arbitrations, or 12 ns for each arbitration. As discussed previously, it takes 92 gates delay for the basic token tunneling scheme with $N = 256$ and $n = 16$. This should be achievable using state-of-the-art 0.25 μm CMOS technology with gate delay less than 100 ps and clocking at 625 Mb/s. The total signal pin count excluding power pins of the XPC is 160.

Conclusions

In this article we study statistical delay bounds of input-output buffered switches under distributed arbitration control. We show that, with a sufficiently large speedup factor, the probability that a cell delay is unacceptably large can be arbitralely small. For instance, a speedup of two is sufficient for random traffic to have less than 10^{-8} probability of waiting for more than 20 slots at inputs, while a speedup of three is needed for bursty traffic with burst length 10 to have a comparable probability of waiting for more than 90 slots. Our simple dual round-robin arbitration scheme can further improve the performance of bursty traffic by reducing the destination correlation of head-of-line cells. While a statistical delay bound is provided at inputs for all sessions of cells, flexible weighted fair queuing should be supported at outputs to achieve different delay bounds for different sessions.

To meet the stringent arbitration time constraint of a terabit per second switch, we proposed the token tunneling method which reduces the arbitration time by a factor of the square root of the switch size. With state-of-the-art 0.25 μm CMOS technology, the arbitration time can be as small as 10 ns for a 256 x 256 Tb/s switch. This scheme can easily be extended to handle multiple-priority requests.

Finally, the Saturn switch uses a bit-sliced crossbar fabric to switch packets at high speed (e.g., 10 Gb/s at each input and output). It also adopts a novel token tunneling technique to arbitrate contending packets at high speed (e.g., within 10 ns), thus achieving a switch capacity of more than 1 Tb/s by existing electronic technology.

Acknowledgment

The author would like to thank Dr. C. H. Lam and Dr. J. S. Park for providing input for the performance study of the Saturn Switch.

References

[1] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks The Multiple Node Case," *IEEE/ACM Trans. Net.*,vol. 2, no. 2, Apr. 1994, pp. 137–50.

[2] D. Stephens and H. Zhang, "Implementing Distributed Packet Fair Queuing in a Scalable Switch Architecture," *Proc. IEEE INFOCOM '98*.

[3] B. Prabhakar and N. McKeown, "On the Speedup Required for Combined Input and Output Queued Switching," Computer Lab, Stanford Univ., tech. rep. CSL-TR-97-738.

[4] S. T. Chuang *et al.*, "Matching Output Queuing with a Combined Input/Output-Queued Switch," *IEEE JSAC*, vol. 17, no. 6, June 1999.

[5] A. Charny *et al.*, "Algorithm for Providing Bandwidth and Delay Guarantees in Input-Buffered Crossbars with Speedup," *Proc. IEEE IEQoS '98*, May 1998, pp. 235–44.

[6] H. J. Chao and J. S. Park, "Centralized Contention Resolution Schemes for a Large-Capacity Optical ATM Switch," *Proc. IEEE ATM Wksp.*, Fairfax, VA, May 1998.

[7] N. McKeown *et al.*, "The Tiny Tera: A Packet Switch Core," *Hot Interconnects V*, Stanford Univer., Aug. 1996.

[8] B. Bingham and H. Busse, "Reservation-Based Contention Resolution Mechanism for Batcher-Banyan Packet Switches," *Elect. Lett.*, vol. 24, no. 13, June 1988, pp. 772–73.

[9] K. Genda *et al.*, "A 160 Gb/s ATM Switching System Using an Internal Speed-up Crossbar Switch," *Proc. GLOBECOM '94*, Nov. 1994, pp. 123–33.

[10] S. C. Liew, "Performance of Various Input-buffered and output-buffered ATM Switch Design Principles Under Bursty Traffic: Simulation Study," *IEEE Trans. Commun.*, vol. 42, no. 2–4, Feb–Apr. 1994, pp. 1371–79.

[11] Y. Oie *et al.*, "Effect of Speedup in Nonblocking Packet Switch," *Proc. ICC '89*, June 1989, pp. 410-14.

[12] S. Q. Li, "Performance of a Nonblocking Space-Division Packet Switch with Correlated Input Traffic," *IEEE Trans. Commun.*, vol. 40, no. 1, Jan. 1992, pp. 97–107.

[13] T. T. Lee and C. H. Lam, "Path Switching – A Quasi-static Routing Scheme for Large-scale ATM Packet Switches," *IEEE JSAC*, vol, 15, no. 5, June 1997, pp. 914–24.

[14] N. McKeown, "Scheduling Algorithms for Input-queued Cell Switches," Ph.D. thesis, UC Berkeley, May 1995.

[15] G. Nong, J. K. Muppala, and M. Hamdi, "Analysis of Nonblocking ATM Switches with Multiple Input Queues," *IEEE/ACM Trans. Net.*, vol. 7, no. 1, Feb. 1999, pp. 60–74.

Biography

H. Jonathan Chao [S'82-M'85-SM'95] (chao@poly.edu) is professor of electrical engineering at Polytechnic University, New York, which he joined in January 1992. His research interests include terabit IP routers, optical packet switches, and quality of service control in IP/ATM networks. He holds 17 patents and has published over 100 journal and conference papers in the above areas. From 1985 to 1991, he was a member of technical staff at Bellcore, New Jersey, where he conducted research in the area of SONET/ATM broadband networks. He was involved in architecture designs and ASIC implementations, such as the first SONET-like Framer chip, ATM Layer chip, and Sequencer chip (the first chip handling packet scheduling). He received Bellcore Excellence Award in 1987. He served as a Guest Editor for IEEE Journal on Selected Areas in Communications (JSAC) with special topics on "Advances in ATM Switching Systems for B-ISDN" (June 1997) and "Next Generation IP Switches and Routers'' (June 1999). He has served as an Editor for IEEE/ACM Transactions on Networking for the past four years. He received his B.S.E.E. and M.S.E.E. degrees from National Chiao Tung University, Taiwan, in 1977 and 1980, respectively, and his Ph.D. in electrical engineering from The Ohio State University, in 1985.