A High-performance OC-12/OC-48 Queue Design Prototype for Input-buffered ATM Switches, IEEE Infocom '97, Kobe, Japan, April 7-11, 1997, pp 20-28.

A High-performance OC-12/OC-48 Queue Design Prototype for Input-buffered ATM Switches

Haoran Duan; J. W. Lockwood, S. M. Kang, and J. D. Will
University of Illinois at Urbana–Champaign
NSF ERC for Compound Semiconductor Microelectronics*
Department of Electrical and Computer Engineering
and Beckman Institute for Advanced Science and Technology
405 N. Mathews Ave., Urbana, IL 61801, U.S.A.
(217) 244-1565 (Voice), (217) 244-8371 (FAX)
email: hd@ipoint.vlsi.uiuc.edu
WWW: http://ipoint.vlsi.uiuc.edu

Abstract

This paper presents the design and prototype of an intelligent, 3-Dimensional-Queue (3DQ) for high-performance, scalable, input buffered ATM switches. 3DQ uses pointers and linked lists to organize ATM cells into multiple virtual queues according to priority, destination, and virtual connection. It enforces per virtual connection Quality-of-Service (QoS) and eliminates Head-Of-Line (HOL) blocking. Using Field-Programmable-Gate-Array (FPGA) devices, our prototype hardware can process ATM cells at 622 Mb/s (OC-12). Using more aggressive technology (Multi-Chip-Module (MCM) and fast GaAs logic), the same 3DQ design can process cells at 2.5 Gb/s (OC-48). Using 3DQ and Matrix-Unit-Cell-Scheduler (MUCS) as essential components, an input-buffered ATM switch system has been designed, which can achieve near-100% link bandwidth utilization.

1 Introduction

The queuing strategies of ATM switches can be broadly classified as *input-queuing* (IQ), *output-queuing* (OQ), or *shared-memory* (SM). Combining two of the above results in *input-output-queuing* (IOQ), *input-shared-queuing* (ISQ), or *output-shared-queuing* (OSQ) [1] [2] [3]. Of all queuing configurations, the IQ structure requires the least memory bandwidth for buffering ATM cells. Each queue module of an IQ switch only buffers cells at the arrival rate of a single port, rather than at a multiple of the arrival rate as with other structures [4] [5] [6] [7] [8]. In addition, it has been found through simulation that with the same buffer size, an IQ switch has more tolerance for *bursty* traffic [9]. Moreover, for multicast traffic, a burst of n cells that are to be delivered to m output ports only needs n cell buffers for the IQ structure, rather than $m \cdot n$ cell buffers for the OQ structure. Therefore, IQ is well-suited to meet the requirements of current and future ultra-broadband ATM networks.

We have successfully implemented a prototype, 5-port input-buffered ATM switch system using FPGA devices for our ATM network testbed—the iPOINT (Illinois Pulsar-based Optical Interconnect). The system was fully functional and provided an aggregate throughput of $800 Mb/s$ [10] [11].

The First-In-First-Out (FIFO) queue implementation used in the previous testbed suffered from HOL blocking and QoS enforcement. Our second-generation iPOINT input-queuing ATM switch, as illustrated in Figure 1, uses a non-FIFO, 3-Dimensional-Queue (3DQ) for cell buffering. 3DQ avoids HOL blocking by buffering ATM cells in a Random-Access-Memory (RAM) and organizing them into multiple virtual queues. Cells are selected for transmission based on Quality-of-Service (QoS) parameters of their virtual connections and run-time traffic conditions.

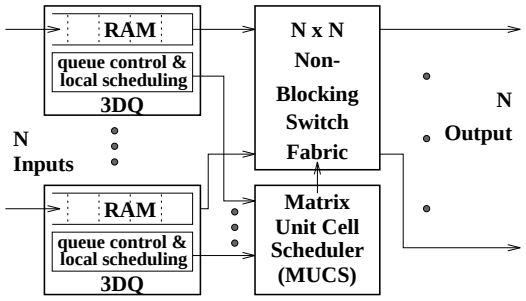

Figure 1: The Second-generation iPOINT Testbed

Using commodity, low-cost cache memory components and moderate performance FPGA devices, our 3DQ system performs ATM cell queuing/processing at 622 Mb/s (OC-12). By re-synthesizing the logic into

*This research has been supported by National Science Foundation Engineering Research Center grant ECD 89-43166, Advance Research Program Agency (ARPA) grant for Center for Optoelectronic Science and Technology (COST) grant MDA 972-94-1-0004.

faster devices, upgrading the speed of the SRAM, and using a Multi-Chip-Module (MCM) for interconnection, we project that 3DQ could operate at a speed of OC-48. Scheduled by novel Matrix-Unit-Cell-Scheduler (MUCS) [12], our second-generation ATM switch can achieve near-100% link bandwidth utilization.

In this paper, we present this 3DQ. After reviewing different input-queuing structures, we discuss the design principles, QoS mechanisms, performance, operation, and implementation of 3DQ. The novel switch level cell scheduler—MUCS, is also briefly reviewed to give a full explanation of the system operation.

2 Existing Input Queues

The major barrier in building a high performance, scalable input-buffered ATM switch has been the lack of efficient cell queuing and scheduling. Multiple designs have been proposed and/or implemented. None, however, provide both per-virtual circuit QoS and eliminate HOL blocking.

2.1 FIFO input queuing

An IQ switch can be built using a single FIFO queue, as illustrated in Figure 2(a). In this scheme, cells can

Figure 2: FIFO Input Queues

only be transmitted from the head of each queue. Cells behind the head of the queue are blocked—regardless of whether or not their destination port is available. For unicast traffic with Poisson arrivals, it has long been known that HOL blocking limits the throughput to 58% [13]. Further, because cells must be transmitted in the same order as they arrive, it is difficult to enforce the QoS.

FIFO queue throughput can be increased by adopting a *window lookahead* based or a *slot reservation* based scheduling algorithm [14] [15] [16], which reduces HOL blocking by checking the first d cells in each FIFO queue module, as shown in Figure 2(b). The throughput improvement is limited, however, because the bursty characteristic of ATM traffic increases the probability that the first d cells in the queue have the same destination address. For bursts longer than the window length, checking only the first d cells does not increase the throughput.

2.2 Non-FIFO input queuing

A priority queue, as shown in Figure 3(a), improves QoS enforcement. Higher priority cells are switched before lower priority ones. Sorting cells only by priority, however, does not avoid HOL blocking. As long as only one cell can be considered for transmission from each input module, the throughput is not improved.

Queuing on a per-virtual-circuit (per-VC) basis, as shown in Figure 3(b), can improve the throughput of the switch. Cells of different virtual-connections are less likely to have the same destination address. The scheduler checks up to k active virtual-connections for non-conflicting destination addresses. Using a round-robin service principle, this queue scheme reduces (but does not eliminate) HOL blocking and better shares the bandwidth among virtual circuits.

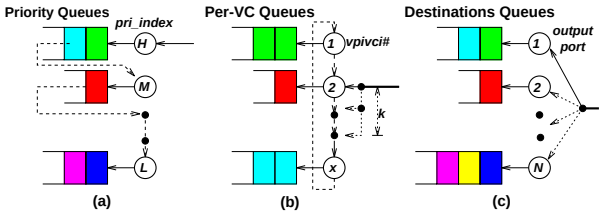

Figure 3: Non-FIFO Input Queues

Queuing by destination port, *i.e.*, an N-queue structure, as shown in Figure 3(c), completely eliminates HOL blocking. It allows scheduling algorithms to evaluate all cell transmission choices [17] [18]. However, N-queue structure lacks QoS enforcement scheme because it does not sort cells by their priority.

3 3-Dimensional-Queue Principle

The iPOINT 3-Dimensional-Queue (3DQ) system organizes incoming ATM cells into multiple queues which form a virtual 3-dimensional space, shown in Figure 4. *Virtual connection*, *priority*, and *destination* are the

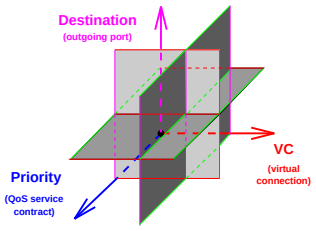

Figure 4: 3-Dimensional Queuing Space

three queuing dimensions. By queuing according to destination, throughput degradation of HOL blocking is eliminated. By queuing according to virtual connection, per-VC bandwidth allocation is enabled. Finally, by queuing according to priority, QoS can be ensured.

3DQ principle has been efficiently implemented using a Random-Access-Memory (RAM). The memory pool is grouped into fixed, ATM-cell-sized units, called *cell room*, and shared by all incoming cells at the input port. Each *cell room* has a corresponding address pointer which links it to the next *cell room* (similar to the queue of the XUNET testbed [19] [20]), and a corresponding *copy index* which indicates how many destination ports remain to be served for this cell. The *freelist* logic tracks all unused *cell rooms*. It assigns an available *cell room* to each incoming cell and collects the *cell room* when the cell has been transmitted. For a multicast cell, the *cell room* is collected only after the cell has been delivered to all multicast destinations.

ATM cells are first sorted by their virtual connections (VCs). Within each VC, cells are transmitted by FIFO-discipline to preserve the circuit's transmission order.

All cells of an active VC are represented by one unique location in the 3D queuing space. Active VCs, represented by the *cell room* address pointers of the cells at the heads of their VC queues, are then clustered into N destination groups. Within each destination group, VCs are further sorted into subgroups, called *service queues*, where VCs have the same priority. Higher priority *service queues* are served ahead of lower priority ones. The internal organization of 3DQ is shown in Figure 5. All pointers, linked lists, and management control logic units are implemented by our FPGA-based hardware. Only a few, hardware-based, table lookup/update operations are needed for queue manipulation.

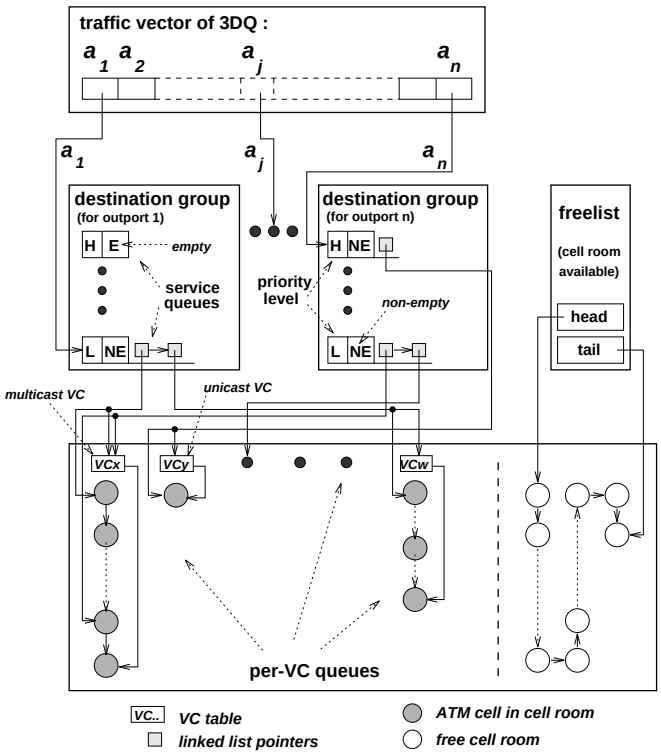

Figure 5: 3-D Queue System Internal Organization

The interface between 3DQ and MUCS—the switch level cell scheduler, is an N-element traffic vector. Each vector element is a nonnegative integer, a_j, where a nonzero value indicates that a cell is ready to be switched to output port j. While $a_j = 0$ indicates an empty destination group, a positive value of a_j indicates the highest priority level among all buffered cells that belongs to the output port j destination group. In each *cell slot*, one cell (which corresponds to one element in the traffic vector) can be chosen for transmission by MUCS [12], as reviewed in Section 10. The scheduling operation is overlapped with the ATM cell transmission of the previously selected cell.

4 Mechanisms for Per-VC QoS

3DQ has multiple built-in mechanisms which enable QoS-oriented cell transmission at per VC level.

4.1 Fair bandwidth sharing

3DQ ensures fair bandwidth sharing among VCs of the same priority by enforcing a Service-Queue-Reentry (SQR) mechanism. Every active VC has only one entry in the corresponding service queue. Once a cell in the VC at the head of a service queue has been transmitted, the VC is removed from the service queue. If the VC has additional cells in the queue, the VC reenters a service queue by appending the updated *cell tag* which points to the new head cell in that VC queue. The service queue being appended is not necessarily the same since the priority level of the VC queue may have been modified. If the VC queue is empty, it becomes inactive until the next arrival of a cell which belongs to that virtual circuit. When a cell arrives to an empty VC queue, the VC is appended to a service queue.

4.2 VC table

For each active virtual circuit, 3DQ maintains a *VC table*, which is essential to per-VC QoS. *VC table* stores QoS parameters, traffic statistics, output destination port(s), outgoing VPI/VCI values, and pointers to the head and tail *cell rooms* of a virtual queue. The corresponding *VC table* is initialized when a VC is established and invalidated when the VC is torn down. Moreover, the corresponding *VC table* is read and updated whenever an ATM cell arrives or is transmitted. Each *VC table* entry stores following parameters:

- *status* indicates whether or not the VC has cells to transmit;
- *desti-vector*, a bit-mapped vector that defines the outgoing destination port(s);
- *para-mcast*, a bit-mapped vector internally used for multicast operation;
- *outgoing-vc* determines the outgoing VPI/VCI;
- *ptr-vcq*, a pointer to the tail *cell room* of the VC queue;
- *qos-para* stores parameters for QoS enforcement;
- *statistics* tracks the number of cells received, transmitted, dropped, and corrupt;
- *upc-rm* stores parameters for traffic policing and Resource Management (RM) cell processing;

4.3 QoS parameters

Per-VC QoS is enforced both locally (at each input module) and globally (at the switch system level) by using the QoS parameters of *VC tables*. These parameters consist of the following:

- *priority* implies the order of transmission relative to other VC;
- *mx-qlgth* indicates the maximum queue length allowed, it sets an upper limit on the memory space that can be used for a single virtual circuit;
- *q-lgth* indicates number of cells of this VC currently queued;
- *q-threshold* is a threshold queue length for priority adjustment for the virtual circuit;

All QoS parameters in an active *VC table* can be modified at run time either by 3DQ logic units or by *control cells*.

4.4 Run time priority adjustment

The *priority* parameter consists of three components: *pri-original*, *pri-local*, and *pri-global*. The priority parameter set during call setup is stored as the *pri-original*. The *pri-local* is controlled by 3DQ hardware. An incoming cell may increase the *pri-local* value, and a transmitted cell may decrease the *pri-local* value. The decision is made by comparing the *q-lgth* with the *q-threshold* (which is also run-time adjustable). The updated *priority* index applies to the VC queue in the next service queue appending operation. The *pri-global* is determined and dynamically adjusted by the switch controller which has the knowledge of allocated bandwidth.

While the *pri-local* provides a mechanism for QoS control at the cell level, the *pri-global* enables the balance of QoS for the switch as a whole. Combining the hardware and software schemes, a hard bound on the cell latency can be made on per VC basis. The algorithm that controls the *q-threshold* and *pri-global* is currently under development.

4.5 Switch controller and control cells

ATM switch controller is responsible for processing signaling messages, establishing/removing virtual circuits, and managing QoS operations at entire switch level. It is a software entity running on a host computer connected to one port of the switch, and communicates with 3DQ through *control cells* which is transmitted on a reserved virtual circuit. *Control cells* are used to initialize a VC table at call setup, to updated certain values of a VC table for dynamic QoS control, to report current traffic statistics to the switch, or to acknowledge a previous issued control cell.

5 3DQ Logic Operation

Figure 6 is a simplified block diagram of 3DQ. It consists of *Cell Memory* to store incoming ATM cells, *Control Memory* to store VC tables, pointers to virtual queues, and logic modules to process ATM cells and manipulate pointers. The 3DQ operation is periodic in the interval of *cell slot*, the time needed to transmit one ATM cell. During each *cell slot*, there are two separate processes that operate in parallel — the reception process (RX) and the transmission process (TX).

5.1 Operation of RX process

The RX process activates when a cell arrives. Besides handling cell buffering operations, it processes the ATM cell header locally, which prevents a processing bottleneck at a centralized switch. As shown in the left-hand portion of Figure 6, the header CRC test is first performed. If the CRC checksum indicates that the header is corrupt, the cell is dropped immediately. For a valid cell, its payload is stored to an available *cell room* of the *Cell Memory*. The cell header is sent to *address compression* logic to find the Internal-Virtual-Circuit-Index (IVCI)—the memory address to access the *VC table*.

After accessing the *VC table*, the RX process performs the following operations in parallel: it updates the VPI/VCI field and reassembles the cell header; appends the incoming cell to the VC queue; re-evaluates the priority of the virtual circuits; and increments traffic statistics.

If the VC queue was empty when the cell arrived, additional operations for service queue(s) are invoked. The RX process creates a new *cell tag* which consists of a *cell room* pointer, an IVCI, and a service queue pointer; appends the *cell tag* to the service queue that is specified by the *desti-vector* and the *qos-para* parameters of the *VC table*; and then updates the traffic vector. Moreover, if the incoming cell belongs to a multicast VC, multicast operations are performed, as discussed in Section 6.

5.2 Operation of TX process

At each *cell slot*, one cell from the N destination groups can be chosen for transmission by the switch cell scheduler. When a cell is selected, the TX process is invoked. The *cell tag* is derived by reading the head of the selected service queue. From the *cell tag*, the *cell room* address and IVCI are retrieved. The TX process then reads the cell from the *Cell Memory*, sends it to the switch fabric, and finally updates the VC queue, the service queue, and the traffic vector. Simultaneously, the priority of the virtual connection is re-evaluated, the *VC table* is updated, and the *cell room* is recycled or its *copy index* is decremented.

As the RX and TX processes both have memory accesses and manipulate queue pointers, their operations are statically scheduled in such a way as to avoid resource contention. Moreover, with careful arrangement of the *VC table* and the *cell tag* formats, only a few memory accesses are necessary to maintain linked lists and process the cell header. As a result, 3DQ operations fit into one *cell slot* for a unicast cell, and a few *cell slots* for a multicast cell.

6 Multicast

The 3DQ supports multicast. An ATM cell can be delivered to any permutation of output ports which are specified by the *desti-vector* (a bit-mapped field) in its *VC table* entry. For a n-port switch, a n-bit *desti-vector* uniquely represents all permutations of output ports. The multicast mechanism of the 3DQ design provides efficient memory usage and prevents a congested destination port from blocking other less-loaded ports.

Cells of a virtual circuit with a multicast factor of m are stored only once in the *Cell Memory*. They are virtually duplicated by transmitting the cell m times, to different destination ports. Cell duplication is tracked by each *cell room's copy index*. When a *cell room* is assigned to a cell with a multicast factor of m, its *copy index* is set to m (set to 1 for a unicast cell). Each time a cell is transmitted, the *copy index* of its *cell room* is decremented. A *cell room* is recycled to *freelist* logic only if its updated *copy index* equals to zero. Thus ensures all the multicast destinations have been served.

To understand the multicast operations of the 3DQ, it is instructive to review Figure 5. Each *destination group* maintains *service queues*. Each *service queue entry* holds a *cell tag*, which in turn points to a *cell room* address within an active (non-empty) virtual circuit. For unicast virtual circuits, the *service queue entry* points to the cell at the head of the virtual circuit. For a multicast virtual circuit, multiple *service queue entries* can exist which point to the same VC queue, but not necessarily all to the head of the queue.

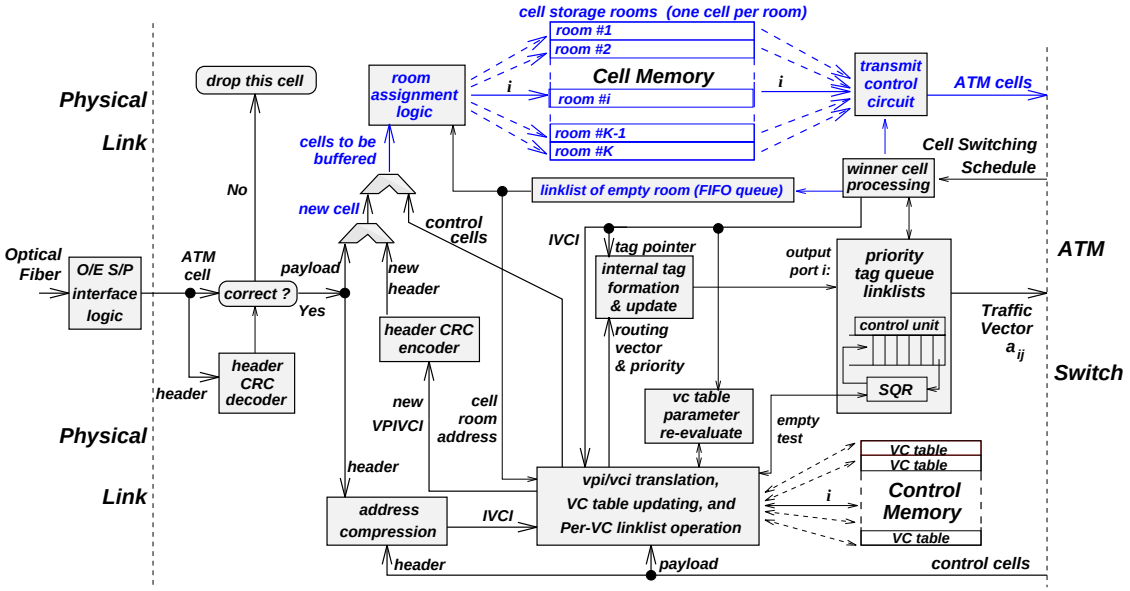

Figure 6: Simplified Diagram of 3DQ

3DQ treats the multicast connection as m virtual streams merged into one VC-queue. An example is given in Figure 7, where $m = 3$. As shown in Figure 7(a), a single VC queue is pointed to by three service queues, each from a different destination group (ports i, j, and k). Since each group is scheduled independently, the 3 virtual streams can have different queue lengths which fluctuate due to the traffic load. Stream i has length 2 while stream k has length 4. When the

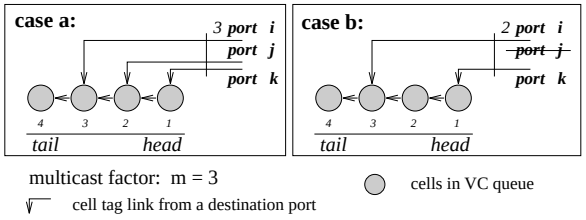

Figure 7: An Example for Multicast Streams

last buffered cell of a particular stream is transmitted, the corresponding pointer is removed form the service queue. As shown in Figure 7(b), all cells to port j have been transmitted. The *para-mcast* field of the *VC table* tracks which stream(s) have finished transmitting cells. It is a bit-mapped vector which is initially set to be the value of *desti-vector*. A bit in the *para-mcast* field is marked to indicate which corresponding streams are no longer in the service queue.

A multicast cell can take more than one *cell slot* during the RX process. The exact number of *cell slots* required depends on how many *cell tags* need to be appended to the service queues. This information is retrieved from the *para-mcast* field. Extra *cell slots* are needed if more than one service queue needs to be appended. The 3DQ design can append up to four service queues for each additional *cell slot*.

The use of more than one *cell slot* in RX for multicast cell processing is acceptable because the offered load at the input of a port with multicast is less than unity on average. Moreover, since any multicast cell with a factor of m needs m *cell slots* for multicast transmission, but only takes $1 + (m-1)/4$ *cell slots* for processing, multicast traffic actually generates less processing load than unicast traffic for our 3DQ design.

7 Simulated QoS Performance of 3DQ

Computer simulations have been performed to study how 3DQ buffers and transmits traffic of different virtual circuits according to their QoS contract and the traffic load at the input port. Incoming traffic is generated using a modified version of the lagged Fibonacci random number generator [21] together with a two-state Markov model [22]. A finite buffer size of 256 *KB* was used for this simulation. Fifteen different virtual connections transporting traffic of five different QoS classes are studied. Their burst characteristics, delay sensitivity, drop sensitivity, mean bandwidths, and buffering space allocations are summarized in Table 1. The total bandwidth of all traffic sources is 69.12 *Mb/s*.

We assume the switch system is symmetrically loaded then vary the transmission bandwidth available for the fifteen VCs. While a transmission bandwidth of 622 *Mb/s* emulates a nearly unloaded switch, a transmission bandwidth of 69.12 *Mb/s* emulates a fully-saturated switch. Our simulations, in fact, even consider extreme cases where the transmission bandwidth to the switch is less than the incoming bandwidth of the data to 3DQ. Simulation results are shown in Figure 8. As indicated, 3DQ clearly distinguishes traffic of different characteristics. On all plots, performance curves cluster into five groups for the five traffic classes. Within each cluster,

Class	A	B	C	D	E
Example	rt-video	CBR	teleconference	ABR	UBR
Bursty	√		√	√	√
Delay sensitive	√	√	√		
Drop sensitive	√	√		√	
Per-VC Bandwidth	6 *Mb/s*	1.54 *Mb/s*	0.5 *Mb/s*	10 *Mb/s*	5 *Mb/s*
Average Burst Length	10	1	5	10	7
Maximum Buffer Size (cells)	512	64	200	512	256
No. of VCs	3	3	3	3	3
Per-class Aggregate Bandwidth	18 *Mb/s*	4.62 *Mb/s*	1.5 *Mb/s*	30 *Mb/s*	15 *Mb/s*

Table 1: Traffic Type

the difference in latency and cell loss is negligible, which indicates the fair treatment of virtual circuits within a traffic class.

Figure 8(a) shows the average cell latency over the entire range of feasible traffic loads. The order of the

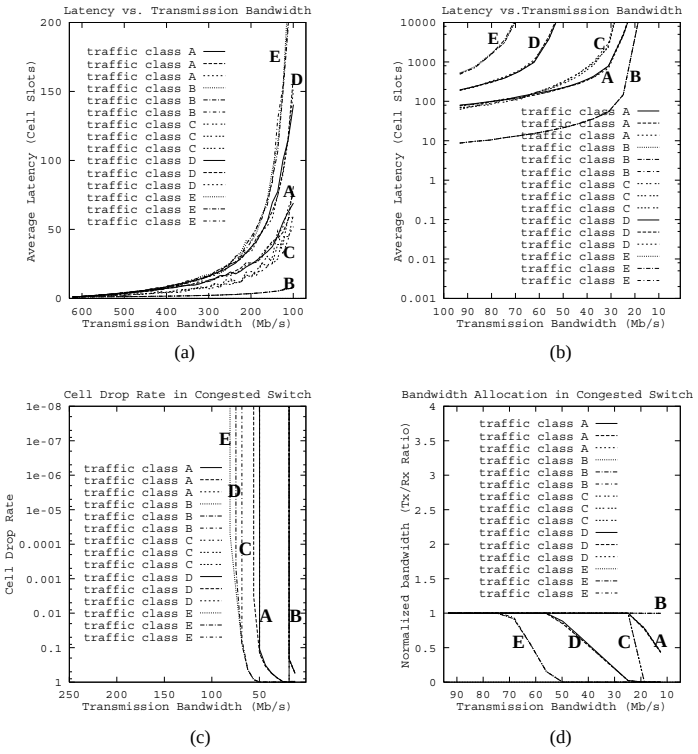

Figure 8: Delay, Drop Rate of VCs vs. Load

curves from the bottom to the top of Figure 8(a) is: class B, C, A, D, and E (classes are summarized in Table 1). Class B traffic always has the minimum latency, and its latency remains relatively constant even as the switch is fully loaded. Class A traffic has longer average delay than class C because of larger burst length of that traffic class.

Figure 8(b) shows the graceful degradation of latency for the traffic classes as 3DQ is pushed beyond saturation. Class A traffic has less average delay than class C during heavy congestion, as it is drop-sensitive.

Figure 8(c) shows the cell loss rate as a function of available bandwidth. The order of cell loss rate, from the least to most, is: class B, A, D, C, and E. We observe no cell loss in any of the fifteen virtual circuits for the entire duration of the 100 million *cell slot* simulation when the transmission bandwidth was larger than 80 *Mb/s*.

Figure 8(d) shows the normalized transmission bandwidth of the virtual circuits. The order from most to least is: class E, D, C, A, and B. As indicated, when the switch operates below the saturation level, the bandwidth requirements of all traffic classes can be satisfied. When the switch is pushed beyond saturation, the bandwidth of lower priority virtual circuits are reduced to ensure the QoS of higher priority ones. For example, when the total transmission bandwidth falls below 50 *Mb/s*, all class E traffic is dropped.

8 Prototyped 3DQ System

3DQ system has been prototyped on a printed circuit board (PCB) called illinois-input-Queue (iiQueue), as shown in Figure 9. The PCB has been fabricated, it measures 8"x14", and has four signal layers. Designed

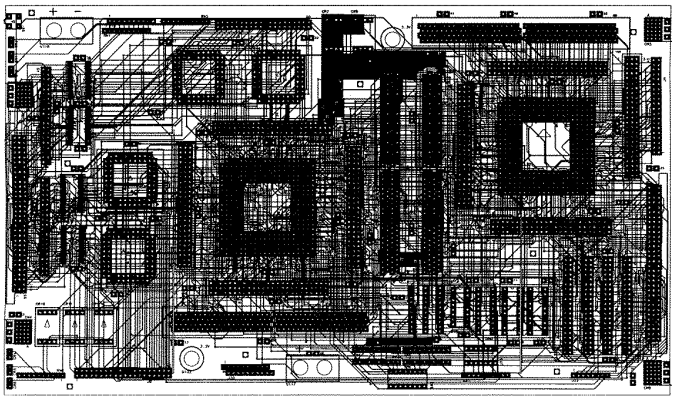

Figure 9: iiQueue PCB Layout Diagram

to operate at 25 *MHz*, iiQueue can process data rates of up to 622 *Mb/s* (OC-12).

8.1 Functional and memory partitions

A block diagram of iiQueue is shown in Figure 10. The major components on iiQueue PCB are RAM modules and FPGA devices. The FIFO devices on the ingress and egress data paths are used only for clock isolation, not cell queuing.

The control logic is implemented as two 25K-gate FPGA devices as *Ingress Cell Processor* (ICP) and *Physical Interface / Egress Cell Processor* (PIF/ECP). ICP implements the control logic of 3DQ and handles ATM cell processing functions. ECP logic completes the header VPI/VCI translation for multicast connections, detects *control cells* from egress traffic and inserts them into the ingress data path. PIF logic adapts the incoming/outgoing traffic to the internal data format of the iPOINT queue/switch system.

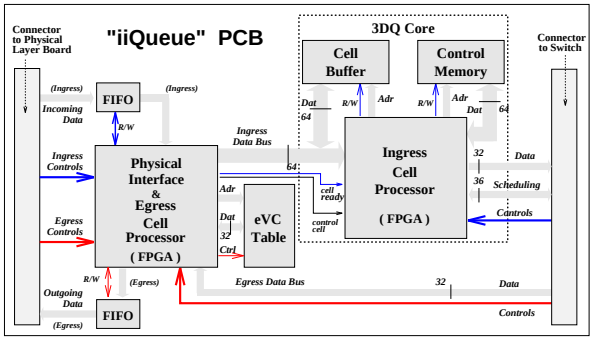

Figure 10: Logic Diagram of iiQueue

Standard secondary-cache memory modules are used to store data and control information of 3DQ. Separate memory buses of *Cell Buffer* and *Control Memory* allow the overlapping of the cell buffering and transmission operations with 3DQ linked-list and QoS operations. There are three separate components: 64-bit wide *Cell Buffer* which holds incoming ATM cells; 64-bit wide *Control Memory* which stores *VC tables* and address pointers that link ATM cells into a virtual 3-dimensional queuing space; and 32-bit wide *eVC Table* which stores multicast VPI/VCI translation table.

8.2 Data flows on iiQueue PCB

Separate ingress and egress data buses are used for incoming and outgoing traffic. Incoming ATM cells are first processed by PIF and converted into 64-bit long words, then queued, processed, and transmitted by the *3DQ Core*.

Outgoing ATM cells arrive ECP from the 32-bit egress data bus connected to switch fabric. ECP provides a transparent path for unicast cells and completes header translation for multicast cells. PIF then converts ATM cells back to the proper data width for the physical interface (*32, 16, or 8* bits).

When a *control cell* is detected, ECP signals PIF to hold incoming traffic in the FIFO for one *cell slot*, then bridges the separate ingress / egress data buses and delivers the *control cell* to the *3DQ Core*. Operating with a 25 *MHz* clock, 3DQ processes traffic at 800 *Mb/s* internally. The extra processing speed of 3DQ creates "holes" in the data flow which enables the insertion of *control cells* from the egress path.

9 Scaling iiQueue to Support OC-48

The 3DQ design can be used to implement an OC-48 queuing module capable of operating at 2.5 *Gb/s*. For the iiQueue prototype, OC-12 performance (622 *Mb/s*) is obtained using moderate-performance FPGA logic and standard PCB technology. An OC-48 module can be implemented by mapping the same logic design into non-FPGA devices (*eg.* CMOS or GaAs gate arrays) and routing the circuit on a Multi Chip Module (MCM). The logic, bus widths, and circuit partitioning of 3DQ and iiQueue remain unchanged.

For the OC-48 module, faster circuit technology is needed to operate the logic at 100 *MHz*. For the prototype iiQueue module, the maximum logic delay is limited to 40 *ns* (1/25 *MHz*). This propagation delay is the sum of the FPGA's block-to-block routing delays between the Control Logic Blocks (CLBs) and by the delay of the CLBs themselves. While the CLB delays are relatively low (3 *ns*), the delays introduced by the pass transistors to interconnect the blocks are significant. By re-synthesizing the logic of the two FPGA devices into GaAs gate arrays, a speedup of four can be obtained. The density of each FPGA corresponds to approximately 25,000 equivalent gates. For GaAs gate arrays, densities of 100,000 equivalent gates are currently available [23] [24].

To interconnect the devices of Figure 10 for the OC-48 iiQueue, the PCB of Figure 9 can be replaced with a MCM. The lower capacitance of the MCM's chip-to-chip interconnects allows for lower signal propagation delays and reasonable power consumption [25] [26]. Similar MCMs have been implemented to interconnect a RISC processor with its secondary cache controller and SRAM memory devices [27] [28].

10 3DQ/MUCS-based Switches

An efficient ATM cell scheduler is necessary to take full advantage of the 3DQ's cell processing and buffering capabilities. It must be able to find a high-throughput, contention-free cell transmission schedule within one *cell slot* time period, which is 680 *ns* for a switch with ports operating at OC-12 and is 170 *ns* at OC-48. Existing scheduling schemes, such as Parallel-Interactive-Match (PIM) based approaches [18] [29], can be used for switch level scheduling. However, those schemes cannot operate at the speeds needed for this switch.

A novel, high performance cell scheduler—Matrix Unit Cell Scheduler (MUCS), has been designed. HSPICE simulations indicate the circuit can response within 100 *ns* using rather modest 2 μm CMOS technology [30]. Integrating 3DQs with MUCS allows the switch system to utilize nearly 100% of the available bandwidth [12]. This section briefly reviews MUCS and presents the throughput simulation of 3DQ/MUCS switches.

10.1 MUCS scheduler

MUCS has been design as a mixed analog-digital circuit which examines the inputs from 3DQ modules and

selects which cell to transmit within one *cell slot*. Winning cells are selected according to the "weight" of elements of a traffic matrix (a_{ij}). This traffic matrix is initially formed by merging the traffic vectors (a_j) from all 3DQ modules, and is reduced when a winner is selected in a matrix iteration.

The weight, w_{ij}, of a_{ij} is defined in Eq. (1), where i, j are indices of input port and output port, respectively. A linear number of matrix iterations solves the complete scheduling problem.

$$\begin{aligned} wr_i &= \sum_{j=1}^{M} a_{ij}; \\ wc_j &= \sum_{i=1}^{M} a_{ij}; \\ w_{ij} &= \begin{cases} \frac{a_{ij}}{wr_i} + \frac{a_{ij}}{wc_j}, & \text{if } a_{ij} \neq 0 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

The above selection rules of MUCS originate from a heuristic strategy that leads to a "socially optimal" solution when conflicts of interest (contention) occur. The heaviest element in the matrix corresponds to a buffered cell at an input port with the least flexibility (this is obvious when a_{ij} is a binary). By choosing the heaviest element(s) first, each iteration of MUCS renders the remaining elements with the maximum number of scheduling opportunities. This leads to a transmission schedule that maximizes the aggregate throughput.

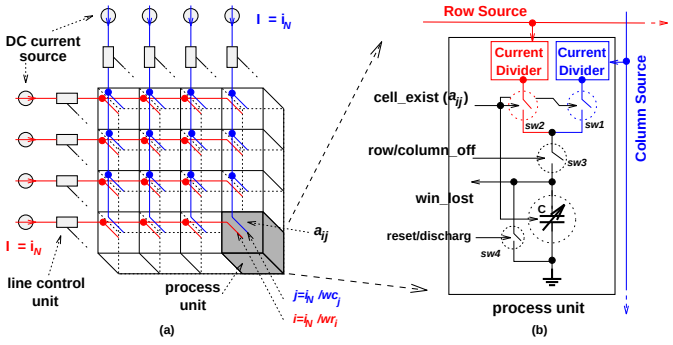

Figure 11: Building Blocks of MUCS Core

Figure 11 illustrates the functional block diagram of the MUCS core. The computation complexity of the entry weight assignment and heaviest weight elements selection are absorbed and mapped to the capacitor charging/discharging procedures, and are executed fully in parallel by the hardware. The transistor level implementation of the MUCS core has been verified by HSPICE. Simulations indicate that the response time of the MUCS is less than 100 *ns* using 2 μm CMOS technology. The circuit has a uniform structure and very low transistor and interconnect count [30].

10.2 Throughput of 3DQ/MUCS switches

Switches of various sizes (N = 5, 8, 16, 32) were simulated to determine the throughput of 3DQ/MUCS switch systems under increasing loads of traffic with uniform random arrivals, Poisson arrivals, and bursty arrival patterns. Single FIFO input-queued switches are simulated for the purpose of comparison.

The simulation results are summarized in Figure 12. For uniform random arrivals and Poisson arrivals (Figure 12(a)(b)), the normalized throughput of 3DQ/MUCS-based switches can approach 100% utilization, while that of a FIFO switch saturates near 58%. For bursty traffic arrivals (Figure 12(c)(d)), the

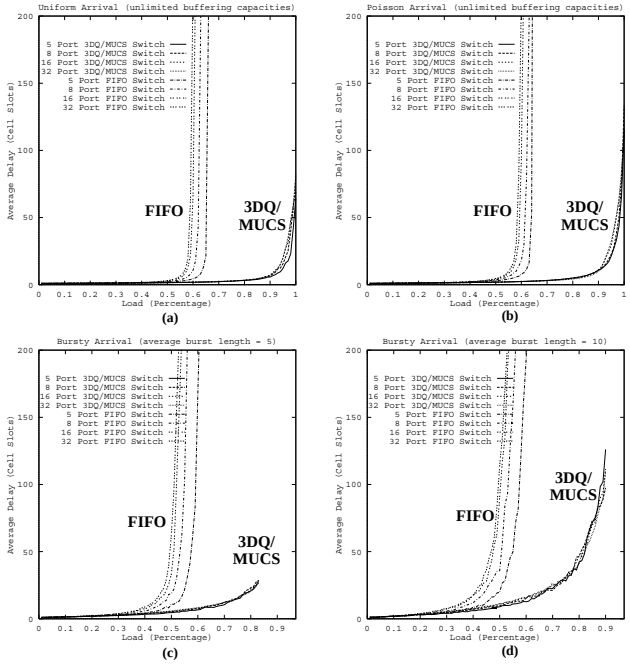

Figure 12: Simulation Results for Unlimited Buffers

3DQ/MUCS-based switches can be loaded to near maximum cell arrival rate for different burst lengths (*e.g.* for L=5, the maximum cell arrival rate is 83.3%), while that of a FIFO switch saturates near 50%. Note that the delay-throughput performance does not degrade as the number of switch ports increases, which indicates the scalability of our design.

11 Conclusion

In this paper, we have presented the principles, design, and implementation of a 3-dimensional queuing system for an input-buffered ATM switch. 3DQ handles cell header translation and cell buffering locally to prevent a processing bottleneck at a centralized switch. 3DQ provides per-VC QoS and eliminates HOL blocking by using RAM-based virtual queues. It combines the individual advantages of per-VC queuing, priority queuing, and N-destination queuing.

We have described the logic operation of 3DQ for cell processing, buffering, transmission, and multicasting. 3DQ provides efficient memory usage for multicast traffic by storing a multicast cell only once in the *Cell Memory*. Moreover, its multicast mechanism prevents a congested destination port from blocking other less-loaded ports.

Computer simulation results verify the QoS capabilities of this design. 3DQ clearly distinguished among the five traffic classes that were simulated. Cell delay

and cell loss of the real-time traffic classes were minimal. The results further show that the latency and cell loss were gracefully degraded as 3DQ was pushed beyond saturation.

A prototype iiQueue module has been designed and fabricated that implements 3DQ with FPGA devices and SRAM modules for ATM queuing at OC-12. By mapping the 3DQ design to faster logic and replacing the PCB with a MCM, a system could be built to support ATM queuing at OC-48. Using MUCS for cell scheduling, a 3DQ-buffered ATM switch can provide per-VC QoS, avoid HOL blocking, and achieve near-100% link bandwidth utilization.

References

[1] A. Pattavina, "Nonblocking Architectures for ATM Switching," *IEEE Communications*, pp. 38–48, Feb. 1993.

[2] C.-Y. Chang, A. J. Paulraj, and T. Kailath, "A Broadband Packet Switch Architecture with Input and Output Queueing," in *GLOBECOM*, pp. 448–452, 1994.

[3] H. J. Chao, J.-S. Park, and B. Choe, "Abacus Switch: A New Scalable Multicast ATM Switch," in *Photonics East: Emerging High-Speed Local-Area Networks*, vol. SPIE Vol. 2608, pp. 230–241, Oct. 1995.

[4] Y. Yeh, M. G. Hluchyj, and A. S. Acampora, "The Knockout Switch: A Simple, Modular Architecture for High–Performance Packet Switching," *IEEE Journal on Selected Areas in Communications*, vol. SAC-5, pp. 1274–1283, Oct. 1987.

[5] K. Y. Eng, M. J. Karol, and Y. S. Yeh, "A Growable Packet (ATM) Switch Architecture: Design Principle and Applications," *IEEE Transactions on Communications*, vol. 40, pp. 423–430, Feb. 1992.

[6] K. Y. Eng, M. J. Karol, G. J. Cyr, and M. A. Pashan, "A 160-Gb/s ATM Switch Prototype Using The Concentrator-Based Growable Switch Architecture," in *ICC '95*, pp. 550–554, 1995.

[7] H. Yamanaka, H. Saito, H. Yamada, M. Tsuzuki, S. Kohama, H. Ueda, H. Kondoh, Y. Matsuda, and K. Oshima, "622 Mb/s 8x8 Shared Multibuffer ATM Switch with Hierarchical Queueing and Multicast Functions," in *GLOBECOM*, pp. 1488–1495, 1993.

[8] J. Garcia-Haro and A. Jajszczyk, "ATM Shared-Memory Switching Architectures," *IEEE Network Magazine*, pp. 18–26, July 1994.

[9] S. C. Liew, "Performance of input-buffered and output-buffered ATM switches under bursty traffic: Simulation study," in *Globecom '90*, vol. 3, pp. 1919–1925, 1990.

[10] J. W. Lockwood, H. Duan, J. J. Morikuni, S. M. Kang, S. Akkineni, and R. H. Campbell, "Scalable Optoelectronic ATM Networks: The iPOINT Fully Functional Testbed," *IEEE Journal of Lightwave Technology*, pp. 1093–1103, June 1995.

[11] H. Duan, J. W. Lockwood, and S. M. Kang, "FPGA Prototype Queuing Module for High Performance ATM Switching," in *Proceedings of the Seventh Annual IEEE International ASIC Conference*, pp. 429–432, Sept. 1994.

[12] H. Duan, J. W. Lockwood, and S. M. Kang, "An Efficient Input Queuing and Cell Scheduling Scheme for Scalable Ultra-broadband Optoelectronic ATM Switching," in *Emerging High-Speed Local-area Networks and Wide-are Networks of Photonics East Conference, SPIE Proceedings*, vol. 2608, pp. 429–432, Oct. 1995.

[13] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input vs. Output Queueing in Space Division Packet Switching," *IEEE Transactions on Communications*, vol. Com-35, pp. 1347–1356, Dec. 1987.

[14] M. K. Mehmet-Ali, J. F. Hayes, and A. K. Elhakeem, "Traffic Analysis of a Local Area Network with a Star Topology," *IEEE Transactions on Communications*, vol. 36, pp. 703–712, June 1988.

[15] H. Obara and Y. Hamazumi, "Parallel Contention Resolution Control for Input Queueing ATM Switches," *Electronics Letters*, pp. 838–839, Apr. 1992.

[16] H. Matsunago and H. Uematsu, "A 1.5 Gb/s 8x8 Cross-Connect Switch Using a Time Reservation Algorithm," *IEEE Journal on Selected Areas in Communications*, vol. 9, pp. 1308–1317, Oct. 1991.

[17] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High-Speed Switch Scheduling for Local-Area Networks," *ACM Transactions on Computer Systems*, pp. 319–352, Nov. 1993.

[18] N. McKeown, P. Varaiya, and J. Walrand, "Scheduling Cells in an Input-Queued Switch," *Electronics Letters*, pp. 2174–2175, Dec. 1993.

[19] A. G. Fraser, C. R. Kalmanek, A. E. Kaplan, W. T. Marshall, and R. C. Restrick, "Xunet 2: A Nationwide Testbed in High–Speed Networking," in *INFOCOM*, pp. 582–589, 1992.

[20] C. R. Kalmanek, S. P. Morgan, and R. C. Restrick, III, "A high-performance queueing engine for ATM networks," in *International Switching Symposium '92*, Oct. 1992.

[21] W. P. Petersen, "Lagged Fibonacci Random Number Generators." Available via anonymous ftp from netlib.att.com as /netlib/random/zufall.shar.Z, 1994.

[22] M. J. Karol, K. Y. Eng, and H. Obara, "Improving the Performance of Input-Queued ATM Packet Switches," in *INFOCOM*, pp. 110–115, 1992.

[23] Vitesse, *ASIC Products Data Book*, 1994.

[24] T. Tsen, S. Tiku, J. Chun, E. Walton, C. S. Bhasker, J. Penny, R. Tang, K. Schneider, and M. Campise, "0.5 μm AlGaAs/GaAs HMESFET technology for digital VLSI products," in *GaAs IC Symposium*, (San Jose, CA), pp. 193–196, Oct. 1993.

[25] R. C. Frye, T. J. Gabara, K. L. Tai, W. C. Fischer, and S. C. Knauer, "Performance evaluationof MCM chip-to-chip interconnections using custom I/O buffer designs," in *IEEE International ASIC Conference*, (Rochester, NY), pp. 464–467, Sept. 1993.

[26] T. Gabara, W. Fischer, S. Knauer, R. Frye, K. Tai, and M. Lau, "An I/O CMOS buffer set for silicon multi chip module's (MCM)," in *IEEE Multi-Chip Module Conference*, (Santa Cruz, CA), pp. 147–152, Mar. 1993.

[27] R. Goyal, "Designing a 100 MHz SPARC dual processor using MCM-L packaging technology and microwave design techniques," in *IEEE MTT-S International Microwave Symposium*, (San Diego, CA), pp. 1715–1718, May 1994.

[28] T. Williams, N. Patkar, and G. Shen, "SPARC64: A 64-b 64-active-instruction out-of-order-execution mcm processor," *IEEE Journal of Solid-State Circuits*, vol. 30, pp. 1215–1226, Nov. 1995.

[29] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *INFOCOM*, (San Francisco, CA), Mar. 1996.

[30] H. Duan, "Design and Development of Queuing, Cell Processing, and Cell Scheduling Modules for the iPOINT ATM Testbed." Preliminary Examination Proposal, University of Illinois at Urbana–Champaign, 1996.