

Matrix Unit Cell Scheduler (MUCS) for Input-Buffered ATM Switches

Haoran Duan, John W. Lockwood, and Sung Mo Kang

University of Illinois at Urbana–Champaign
Department of Electrical and Computer Engineering
Center for Compound Semiconductor Microelectronics*
and Beckman Institute for Advanced Science and Technology

405 N. Mathews Ave., Urbana, IL 61801
(217) 244-1565 (Voice) (217) 244-8371 (FAX)
email: `hd@ipoint.vlsi.uiuc.edu`
WWW: `http://ipoint.vlsi.uiuc.edu`

September 17, 1997

*This research has been supported by National Science Foundation Engineering Research Center grant ECD 89-43166, Defense Advanced Research Projects Agency (DARPA) grant for Center for Optoelectronic Science and Technology (COST) grant MDA 972-94-1-0004, and AT&T foundation.

ABSTRACT

This paper presents a novel matrix unit cell scheduler (MUCS) for input-buffered ATM switches. The MUCS concept originates from a heuristic strategy that leads to an optimal solution for cell scheduling. Numerical analysis indicates that input-buffered ATM switches scheduled by MUCS can utilize nearly 100% of the available link bandwidth. A transistor-level MUCS circuit has been designed and verified using HSPICE. The circuit features a regular structure, minimal interconnects, and a low transistor count. HSPICE simulation indicates that using $2\ \mu\text{m}$ CMOS technology, the MUCS circuit can operate at clock frequency of 100 MHz.

1 Introduction

The input-queuing (IQ) architecture is attractive for building ultra-high speed ATM switches [1] [2]. By queuing incoming traffic according to the output destination at each input port (N -queue) and scheduling the transmission of queued cells effectively, an IQ-ATM switch can avoid HOL blocking and achieve 100% throughput [3]. The lack of an efficient cell scheduler, however, has been one of the major barriers for building a high-performance, scalable IQ-ATM switch. Such a scheduler must resolve output contention swiftly, provide high throughput, satisfy QoS requirements, and most importantly, be implementable in a low-cost integrated circuit.

Existing scheduling algorithms are complicated and costly to implement in hardware. PIM-based (parallel interactive matching) algorithms [4], such as a IRRM (iterative round-robin matching) or a SLIP-IRRM (IRRM with slip) scheduler [5], require $2N$ round-robin arbitrators with $O(N^2)$ complexity of interconnects [6]. Neural-network-based solutions require $O(n^2)$ neurons and $O(n^3)$ interconnects [7].

In this paper, we present a novel matrix unit cell scheduler (MUCS). The MUCS enables ATM switches to utilize nearly 100% of the available link bandwidth and requires significantly less hardware to implement. Its transistor-level circuit implementation features a uniform structure, has a very low interconnect and transistor count, and achieves a near-optimal cell scheduling within a linear processing time [8].

2 The MUCS Algorithm

MUCS resolves output contention by computing a traffic matrix, \mathbf{A} . Initially, each row of \mathbf{A} summarizes the status of queued cells at each input port of the switch through an N -element traffic vector. Each element, a_{ij} of \mathbf{A} , is a nonnegative integer. Whereas $a_{ij} = 0$ indicates that no cell is ready to be switched to output port j , from input port i , a positive value indicates the highest QoS index among all buffered cells to be switched to output port j [9] [10].

The MUCS algorithm selects an optimal set of entries as winning cells from matrix \mathbf{A} according to the weight w_{ij} of a_{ij} . It calculates the heaviest entries under the constraint that no more than one cell is transmitted from an input port and no more than one cell is delivered to an output port simultaneously. When implemented in a MUCS module, the selection process can be performed fully in parallel by the hardware.

Letting N be the size of the switch and M the size of the reduced traffic matrix \mathbf{A}' of \mathbf{A} at each iteration, the algorithm is described as follows:

1. Initially, $M = N$, and $\mathbf{A}' = \mathbf{A}$.
2. For each iteration, the entry weight w_{ij} is assigned to every entry of \mathbf{A}' . The value of w_{ij} is determined by:

$$w_{ij} = \begin{cases} \frac{a_{ij}}{wr_i} + \frac{a_{ij}}{wc_j}, & \text{if } a_{ij} \neq 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

where

$$wr_i = \sum_{j=1}^M [\text{number of } a_{ij} > 0] \quad (2)$$

$$wc_j = \sum_{i=1}^M [\text{number of } a_{ij} > 0] \quad (3)$$

3. The entry with the heaviest weight is selected as the winner. Multiple entries can be selected as winners simultaneously if they all have the same weight and do not reside in the same row or column. Once the winners have been selected, a new reduced matrix, \mathbf{A}' , is formed as all elements of current \mathbf{A}' , excluding the rows and columns with winning cells. The algorithm terminates when all entries have been treated.
4. If a tie occurs, winners are selected randomly.

3 Heuristic Strategy in MUCS

The MUCS algorithm originates from a heuristic strategy that yields a most satisfactory solution when conflicts of interest exist. In the ATM scheduling problem, the best interest of an input port is defined as the transmission of a cell and the best interest of an output port is defined as the reception of a cell. Optimal scheduling decision should enable the transmission of a maximum number of queued cells.

The heuristic strategy behind the MUCS algorithm is most easily described when a_{ij} is discretized to a binary. In this condition, Eq. (1) is reduced to

$$w_{ij} = \begin{cases} (wr_i)^{-1} + (wc_j)^{-1}, & \text{if } a_{ij} \neq 0 \\ 0, & \text{else} \end{cases} \quad (4)$$

The heaviest element in the matrix corresponds to a buffered cell that has the least contending candidates in the same row and the same column. 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.

When a_{ij} is not discretized and Eq. (1) is applied, the solution MUCS found will sacrifice the overall throughput to satisfy certain QoS requirements. For example, if the cell priority is the major factor that determines a_{ij} , the overall throughput is traded for less delay of higher priority cells. Many variations and extensions of the MUCS algorithm can be made by modifying the function that generates a_{ij} .

4 The MUCS Implementation

A novel, mixed digital-analog core has been designed to implement the computation of the MUCS algorithm. A block diagram of the MUCS core is given in Figure 1. The circuit has a regular structure which resembles a symmetrical matrix. Its major building blocks consist of a constant DC current source, a process unit, and a line control unit. For an $N \times N$ switch, N^2 process units are needed, each of which corresponds to an element of the traffic matrix. Each process unit consists of circuit switches, two current dividers (one for the column and another for the row), and capacitors. Signals from line control units open and close these switches for charging and discharging the capacitors. The value of a_{ij} is represented by values of capacitors. When fixed capacitors are used, the

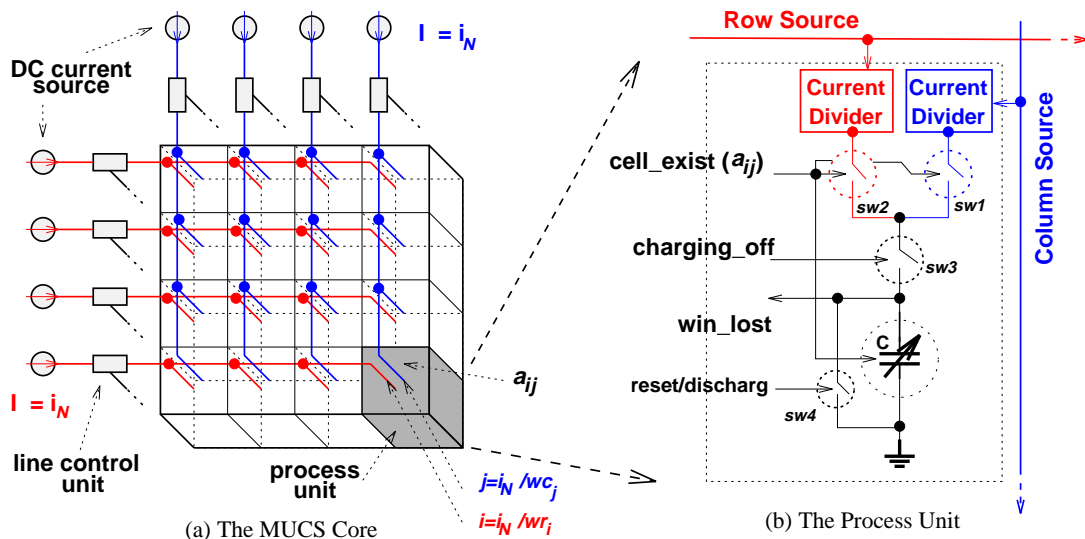


Figure 1: Building Blocks of MUCS Core

traffic matrix \mathbf{A} becomes a binary matrix after loaded into the MUCS circuit and the solution is optimized for maximum throughput. When variable capacitors are employed, different values of a_{ij} (QoS factors) can be included in the scheduling.

The MUCS core computes the solution in N steps. Each step consists of a charging phase followed by a discharging phase. In the charging phase, $sw3$ is closed and $sw4$ is open for each participating process unit. The setting is opposite for the discharging phase.

Let us define i_N as the value of the constant DC current supplied, and use wr_i and wc_j (refer to Eqs. (2) and (3)) as the number of active entries in each row and column in the reduced traffic matrix. Then, for the charging phase, if $a_{ij} > 0$, the two current divider blocks of a process unit drain i_N/wr_i and i_N/wc_j currents from the column and row current sources, respectively. The combined current charges the capacitor C . The heavier the weight of an entry, the more the total current drained.

The winner is indicated by the voltage level on a corresponding capacitor in the array. When the voltage of the winning capacitor reaches a pre-defined threshold, the line control logic is triggered to open $sw3$ to terminate the charging process and eliminate other elements of the same row and column. The discharge process erases the residue charge on the capacitor(s) of the process units which belong to the unresolved rows and columns. This, in effect, reduces the dimensions of the matrix for the next charging process. The MUCS core, therefore, absorbs the computation complexity of the traffic matrix entry weight assignment and the heaviest weight element selection by mapping

them as the capacitor charging / discharging procedure of the circuit and executing fully in parallel through hardware.

A transistor-level circuit has been designed that implements the MUCS core. The circuit has a regular structure, minimal interconnects, and a low transistor count. Each processing unit requires only 29 transistors and 2 capacitors. Besides the charging current inputs, each process unit has only three common wired signals for interconnect between process units [11]. HSPICE was used to verify the design. Simulation results indicate that with 2 μm CMOS technology, the MUCS circuit can operate at clock frequency of 100 MHz clock. Using larger current sources, reducing the size of capacitor, or reducing the transistor feature size can make the MUCS circuit operate even faster.

5 Numerical Analysis of MUCS Performance

MUCS achieves a near-optimal throughput when it is optimized for selecting the maximum number of cells. In this case, all entries of the matrix \mathbf{A} are binary element.

5.1 Selecting cells from random traffic matrices

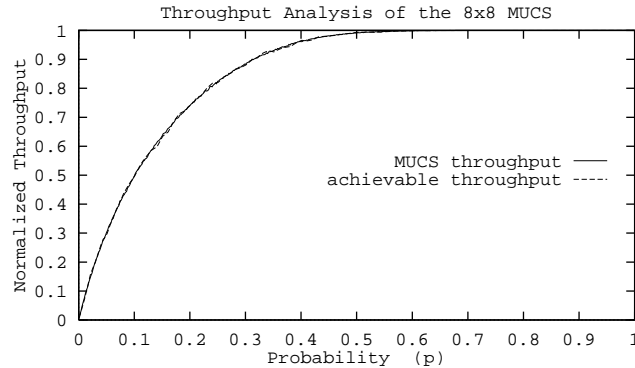


Figure 2: 8x8 Switch Throughput vs. ρ

Simulation was performed to find the normalized average number of cells selected by MUCS from random binary traffic matrices. The results were compared to the maximum number of cells that could be switched. The parameter ρ is defined as the probability of $a_{ij} = 1$.

Figure 2 compares the normalized throughput of the optimal solution and the solution found by MUCS, in a switches of size 8 with ρ varying from 0 to 1. As indicated, the two

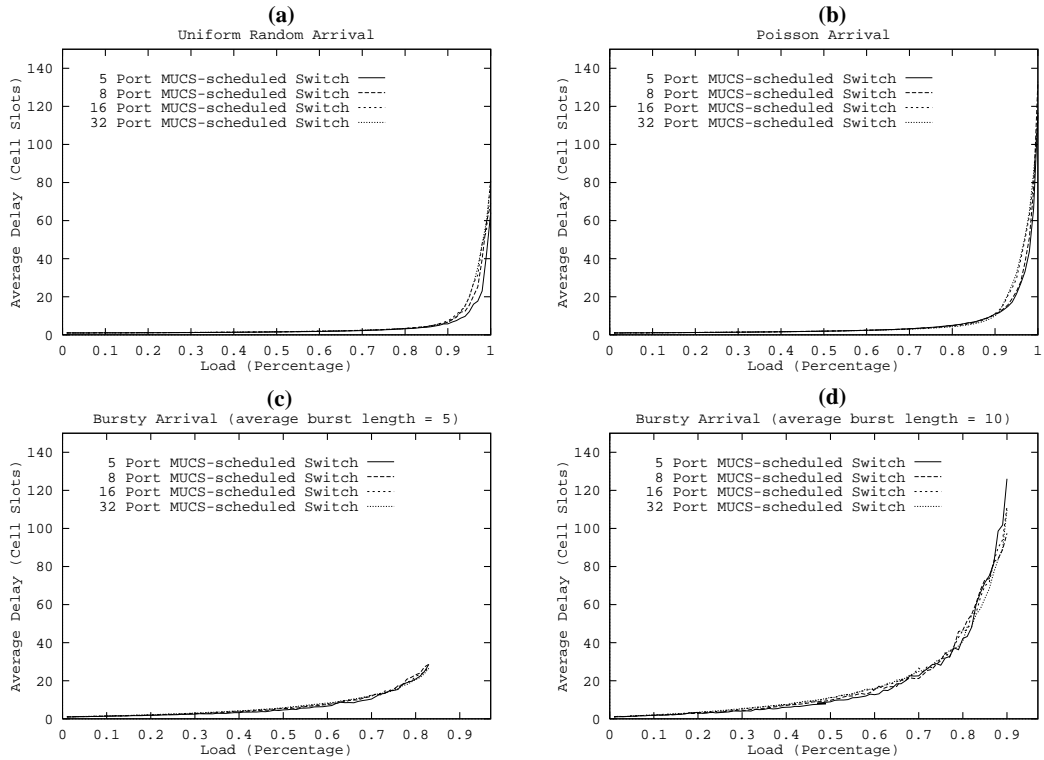


Figure 3: Simulation Results for Unlimited Buffers

plots are nearly overlapped. The heuristic strategy provides a near-optimal solution.

5.2 Scheduling IQ-ATM switches

Switches of various sizes ($N = 5, 8, 16, 32$) were simulated to determine the throughput of IQ-ATM switches scheduled by MUCS. Increasing loads of traffic with uniform random arrivals, Poisson arrivals, and bursty arrival patterns were imposed on the switch.

The simulation results are summarized in Figure 3. For uniform random arrivals and Poisson arrivals (Figure 3(a)(b)), the normalized throughput can approach 100%. For bursty traffic arrivals, the switches can be loaded to near maximum cell arrival rate for different burst lengths. As indicated in Figure 3(c)(d), for traffic with an average burst length of $L=5$ or $L=10$, the MUCS can handle 83.3% or 90.9% link utilization with a mean queue length of less than 40 or 140 cells respectively. Note that the delay-throughput performance does not degrade as the number of switch ports increases, which demonstrates the scalability of MUCS.

6 Conclusion

MUCS provides a highly-effective mechanism for scheduling input-buffered ATM switches. It provides near-optimal throughput and can be implemented with simple hardware. Its performance has been verified through simulation and its hardware design has been verified with HSPICE.

References

- [1] 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*, vol. 13, pp. 1093–1103, June 1995.
- [2] B. Prabhakar, N. McKeown, and R. Ahuja, "Multicast scheduling for input-queued switches," to appear on *IEEE Journal on Selected Areas in Communications*, 1997.
- [3] N. McKeown, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch," in *INFOCOM*, Mar. 1996, pp. 296–302.
- [4] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High-speed switch scheduling for local-area networks," *ACM Transactions on Computer Systems*, vol. 11, pp. 319–352, Nov. 1993.
- [5] N. McKeown, P. Varaiya, and J. Walrand, "Scheduling cells in an input-queued switch," *Electronics Letters*, vol. 29, pp. 2174–2175, Dec. 1993.
- [6] N. Mckeown, M. Izzard, A. Mekittikul, B. Ellersick, and M. Horowitz, "The Tiny Tera: A packet switch core," *IEEE Micro Magazine*, vol. 17, pp. 27–40, Feb. 1997.
- [7] M. K. Mehmet-Ali, M. Youssefi, and H. T. Nguyen, "The performance analysis and implementation of an input access scheme in a high-speed packet switch," *IEEE Transactions on Communications*, vol. 42, pp. 3189–3199, Dec. 1994.
- [8] H. Duan, "Design and development of cell queuing, processing, and scheduling modules for the iPOINT input-buffered ATM testbed." Ph. D. Dissertation, University of Illinois at Urbana–Champaign, 1997.
- [9] H. Duan, J. W. Lockwood, S. M. Kang, and J. D. Will, "A high-performance OC-12/OC-48 queue design prototype for input-buffered ATM switches," in *INFOCOM*, Apr. 1997, pp. 20–28.
- [10] H. Duan, J. W. Lockwood, and S. M. Kang, "A 3-dimensional queue (3DQ) for practical ultra-broadband input-buffered ATM switches," submitted to *IEEE Transactions on VLSI Systems*, 1997.
- [11] H. Duan, J. W. Lockwood, and S. M. Kang, "Scalable broad band input-queued ATM switch including weight driven cell scheduler." Pending US Patent Application 1201.61488, Oct. 1996.