Pipelined Maximal Size Matching Scheduling Algorithms for CIOQ Switches

Mei Yang and S. Q. Zheng

Department of Computer Science
University of Texas at Dallas, Richardson, TX 75083-0688, USA
{*meiyang, sizheng*}*@utdallas.edu*

Abstract

In this paper, we propose new pipelined request-grant-accept (RGA) and request-grant (RG) maximal size matching (MSM) algorithms to achieve speedup in combined input and output queueing (CIOQ) switches. To achieve a speedup factor S, in the proposed pipelined RGA/RG MSM algorithms, we pipeline operations of finding S matchings in S scheduling cycles based on the observation that all matched inputs/outputs will not be used in later iterations in the same scheduling cycle. We show that our pipelined RGA/RG MSM algorithms reduce the scheduling time constraint by $\frac{SI}{I+S-1}$, where I is the number of iterations allowed in each scheduling cycle. Taking the example of pipelined PIM, we evaluate the performance of the proposed algorithms by simulation. Simulation results have shown that pipelined PIM achieves 100% throughput and the same performance as non-pipelined PIM for CIOQ switches with speedup of 2 under both Bernoulli and bursty arrivals.

1. Introduction

Recently, combined input and output queueing (CIOQ) switches have attracted interest from both academic and industrial communities due to their ability of achieving 100% throughput and emulating output queueing (OQ) switch performance with a relatively small speedup factor S. For a CIOQ switch with speedup of S, the switching matrix and the memory need to run S times faster than the line rate. It has been shown that a CIOQ switch can exactly emulate an OQ switch with speedup of 4 or 2 by employing stable-matching [1] based algorithms, such as the MUCFA algorithm [2] and the CCF algorithm [3]. These results have significant implications: regardless of the switch size, a small constant speedup is sufficient to implement a CIOQ switch with behavior identical to an OQ switch which has a speedup proportional to the switch size. Unfortunately, these algorithms are highly impractical due to their high time complexity.

In [4], Dai and Prabhakar proved that for a CIOQ switch with $S = 2$, any maximal size matching algorithm can achieve 100% throughput for arbitrarily distributed input patterns so long as input arrivals satisfy the strong law of large numbers (SLLN) and no inputs/outputs are oversubscribed. Since almost all real traffic processes satisfy these properties, this result has high practical significance for at least two reasons. First, achieving 100% throughput is a necessary condition for a CIOQ switch to realize OQ-equivalent QoS guarantees with carefully designed queuing discipline at each VOQ and at each output queue. Second, maximal size matching algorithms are easier to implement than maximum size matching or stable matching algorithms.

Well-known maximal size matching algorithms include PIM [5], *i*SLIP [6], etc. All these algorithms are iterative algorithms with each iteration composed of either three steps (request, grant and accept) or two steps (request and grant). It has been shown by either proof or simulations that averagely PIM and *i*SLIP find a maximal size matching in $O(\log N)$ iterations [5, 6]. With the fastest implementation, each iteration can be finished in $O(\log N)$ time [7]. However, these algorithms may not be fast enough for CIOQ switches with $S > 1$ and high line rate since each maximal size matching needs to be found in $1/S$ cell slot if they are found sequentially.

To relax the stringent scheduling time constraint, Smiljanic et al. proposed a pipeline-based scheduling algorithm called round-robin greedy scheduling (RRGS) [8]. In RRGS, each input is associated with a schedule module (SM), and N SMs are interconnected as a ring. Each SM receives allocation requests from an input, allocates an unreserved output, and transfers updated output reservation information to the next adjacent SM. Round-robin selection operations are assigned into different time slots in a simple pre-determined cyclic manner so that RRGS can avoid output contention. However, RRGS suffers from a fairness problem among inputs competing for the same output. Another problem is that, though operated in pipeline fashion, scheduling for a cell slot is performed in N previous cell slots, which is a rather slow process. Other improvements of RRGS, such as COPRS [9] and GPS [10], solve the unfairness problem of RRGS with the tradeoff of performance degradation or more complicated implementation.

In [11], Oki et al. has proposed PMM, a pipelined maximal size matching scheduling approach to relax scheduling time and improve fairness. It has been shown that, PMM dramatically relaxes the arbitration time constraint for arbitration of a maximal size matching algorithm and achieves 100% throughput under uniform traffic. However, to relax the arbitration time K times, this approach requires K sub-schedulers, each sub-scheduler is equivalent to a scheduler in the non-pipelined version. Also PMM in-

Proceedings of the Eighth IEEE International Symposium on Computers and Communication (ISCC'03)
1530-1346/03 $17.00 © 2003 IEEE

troduces more average cell delay.

Nevertheless, all these pipelined schemes discussed above can not provide speedup in one cell slot. In this paper, we propose new pipelined request-grant-accept (RGA) and request-grant (RG) maximal size matching (MSM) algorithms to achieve speedup of $S > 1$ in CIOQ switches. We assume that each cell slot is composed of S overlapped scheduling cycles. In our pipelined RGA/RG MSM algorithms, we pipeline operations of finding S matchings in S scheduling cycles based on the observation that all matched inputs/outputs will not be considered for matchings in later iterations in the same scheduling cycle. We show that our pipelined RGA/RG MSM algorithms reduce the scheduling time constraint by $\frac{SI}{I+S-1}$ times. Taking the example of pipelined PIM [5], through simulations, we show that it achieves 100% throughput and the same performance as non-pipelined PIM for CIOQ switches with speedup of 2 under both Bernoulli and bursty arrivals. Compared to RRGS and PMM, the advantage of the proposed algorithms is that they do not require extra hardware resources but achieve speedup of $S > 1$.

The rest of the paper is organized as follows. Section 2 reviews existing iterative maximal size matching algorithms. In Section 3, we present our pipelined RGA and RG MSM algorithms and discusses the scheduling time relaxation they can achieve. Section 4 demonstrates simulation results of pipelined PIM under both Bernoulli arrivals and bursty arrivals. Section 5 concludes the paper.

2. Existing Iterative Maximal Size Matching Algorithms

Consider an $N \times N$ crossbar CIOQ switch shown in Figure 1. To remove HOL blocking [12], each input (port) maintains N virtual output queues (VOQs) with $VOQ_{i,j}$ buffering cells from input i destined to output j. Likewise, each output (port) has a queue for holding cells destined to it. Assume that time is slotted into cell slots and one cell slot equals to the transmission time of one cell on the input/output line. To realize a speedup factor S, each cell slot is composed of S scheduling cycles. A scheduling cycle consists of two parts, *matching part* and *switching part*. In the matching part, a scheduling algorithm selects a matching between inputs and outputs such that each input (resp. output) is matched to at most one output (resp. input). In the switching part, input i transfers a cell to output j if they are matched to each other and $VOQ_{i,j}$ is not empty. In our paper, we only consider the matching part.

The cell scheduling problem for VOQ-based switches can be abstracted as a maximum bipartite matching problem on the bipartite graph composed of nodes of inputs and outputs and edges of connection requests from inputs to outputs [6]. However, maximum size matching algorithms are not practical due to their high time complexity ($O(N^{2.5})$ [13]) and unfairness problem. We instead consider a group of practical iterative MSM algorithms. Existing iterative MSM algorithms can be classified into two categories: request-grant-accept (RGA) algorithms and request-grant (RG) algorithms.

In these algorithms, initially, all inputs and outputs are marked "unmatched". In each iteration, a matching between "unmatched" inputs and outputs is found and all matched inputs and outputs in this iteration are marked "matched" so that they will not be considered in the next iteration. In what follows, we outline the operations performed in each iteration of RGA and RG MSM algorithms.

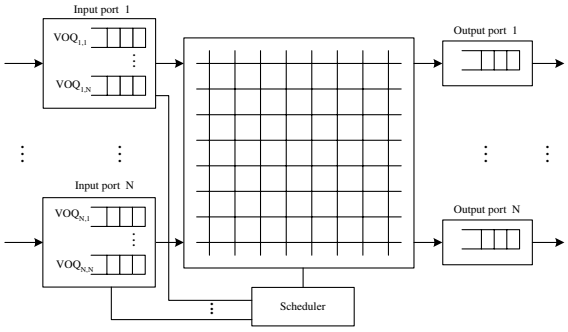

Figure 1. A CIOQ switch.

One iteration of RGA MSM algorithm:

Step 1: **Request**. Each "unmatched" input sends a request to every "unmatched" output for which it has a queued cell.

Step 2: **Grant**. If an "unmatched" output receives any requests, it selects one request to grant, and notifies each requesting input whether or not its request is granted.

Step 3: **Accept**. If an "unmatched" input receives any grants, it selects one grant to accept, and marks itself and its matched output "matched".

One iteration of RG MSM algorithm:

Step 1: **Request**. Each "unmatched" input selects one request to an "unmatched" output, and sends the request to this "unmatched" output.

Step 2: **Grant**. If an "unmatched" output receives any requests, it selects one request to grant. The output notifies each requesting input whether or not its request is granted. It marks itself and the matched input "matched".

These algorithms terminate when a non-profitable iteration is encountered, which indicates that a maximal size matching has been found. In the worst case, $O(N)$ iterations are needed. In average, it has shown that either by proof or simulations that $O(\log N)$ iterations are adequate [5, 6]. Several iterative RGA and RG MSM algorithms have been proposed, such as PIM [5], *i*SLIP [6], DRRM [15], FIRM [16]. The differences between these algorithms lie in the selection (arbitration) schemes used in Step 2 and 3 of RGA algorithms and Step 1 and 2 of RG algorithms. These algorithms can be implemented by the scheduler architecture proposed by Mckeown [6]. The scheduler consists of $2N$ arbiters, each one is associated with an input/output. With the fastest arbiter design, each iteration can be finished in $O(\log N)$ time [7].

However, MSM algorithms mentioned above may not be fast enough to be used for CIOQ switches with $S > 1$. Let T_{cs} be the time for one cell slot, and T_{msm} be the total time for finding a

maximal size matching. In order to achieve speedup of S, S maximal size matchings must be found during one cell slot. If maximal size matchings are found sequentially, it requires $T_{msm} \leq T_{cs}/S$. This is very difficult, if possible, to realize for a CIOQ switch with high line rate and $S > 1$.

3. Pipelined RGA/RG MSM Algorithms

To speed up the process of finding S maximal size matchings, we propose a pipelined framework for a class of accelerated RGA and RG algorithms, which are called *pipelined RGA* and *pipelined RG* MSM algorithms. In each cell slot, these algorithms find a set of S matchings in a pipelined fashion. We associate a label l with each input and each output, $1 \leq l \leq S$, indicating the scheduling cycle that the input/output is involved. Initially, $l = 1$ for all inputs I_i's and outputs O_j's, $1 \leq i, j \leq N$. During the execution of pipelined RGA/RG MSM algorithms, when a matching between input I_i and output O_j is found, the l labels of input I_i and output O_j are updated one beyond the current value. All matched pairs of inputs and outputs with the same l-label values form a matching M_l, $1 \leq l \leq S$.

We associate two counters with each $VOQ_{i,j}$: $L_{i,j}$, the length of $VOQ_{i,j}$ (i.e. the number of cells queued in $VOQ_{i,j}$), and $G_{i,j}(l)$, the number of grants associated with $VOQ_{i,j}$ in the l-th scheduling cycle, $1 \leq l \leq S$. In each cell slot, $L(_{i,j})$'s are initialized before scheduling, $G_{i,j}(l)$'s are initialized to be 0 before scheduling and updated after each iteration. We also assume one scheduling cycle consists of fixed number of iterations, represented by I. The pipelined RGA/RG MSM algorithms are sketched below, where k represents the pipelined iteration.

Pipelined RGA MSM algorithm:
for $k = 1$ **to** $S + I - 1$ **do**

Step 1: **Request**. Each l-labelled input I_i, $l \leq S$, sends a request to each l-labelled output O_j such that $L_{i,j} - \sum_{g=1}^{l-1} G_{i,j}(g) > 0$.

Step 2: **Grant**. Upon receiving any requests, each l-labelled output selects one request to grant, and notifies each l-labelled input whether or not its request is granted.

Step 3: **Accept**. Upon receiving any grants, each l-labelled input selects one grant to accept, and increments the labels of itself and its matched output by 1. If $k - l = I - 1$, increment labels of all unmatched l-labelled inputs and outputs by 1.

Pipelined RG MSM algorithm:
for $k = 1$ **to** $S + I - 1$ **do**

Step 1: **Request**. Each l-labelled input I_i, $l \leq S$, selects one request to an l-labelled output j such that $L_{i,j} - \sum_{g=1}^{l-1} G_{i,j}(g) > 0$.

Step 2: **Grant**. Upon receiving any requests, each l-labelled output selects one request to grant, and increments the labels of itself and its matched input by 1. If $k - l = I - 1$, increment labels of all unmatched l-labelled inputs and outputs by 1.

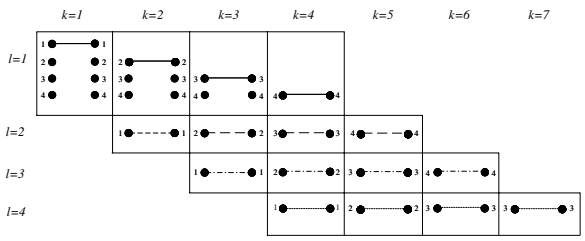

Figure 2. Illustration of a pipelined scheduling algorithm.

Figure 2 illustrates how the pipelined scheduling algorithm works using an example of a 4×4 CIOQ switch assuming $I = 4$ and $S = 4$. In the figure, each row shows the matching process of scheduling cycle $1 \leq l \leq 4$; each column shows the matching process viewed in each pipelined iteration $1 \leq k \leq 7$; each node represents an input/output and each edge connects two nodes that are matched in the current pipelined iteration. For $k = 1$, all inputs and outputs are labelled 1 and join the matching process of $l = 1$. We assume that only I_1 and O_1 are matched, then they will update their labels to 2 and join the matching process of $l = 2$ in $k = 2$ while all the remaining inputs and outputs continue their matching process of $l = 1$ in $k = 2$. For $k = 2$, assume I_1 and O_1 are matched for $l = 2$ and I_2 and O_2 are matched for $l = 1$, then I_1 and O_1 will update their labels to 3 and join the matching process of $l = 3$ in $k = 3$, and I_2 and O_2 will update their labels to 2 and join the matching process of $l = 2$ in $k = 3$, while other inputs and outputs continue their matching process of $l = 1$ in $k = 3$. Similar process continues for $k \geq 3$. We observe that all input and output pairs matched in $l \leq k \leq l + I - 1$ but for scheduling cycle l form matching M_l (represented by edges of the same type). In addition, inputs and outputs involved in matching processes for different scheduling cycles but in the same pipelined iteration do not conflict. Therefore the iterations of finding M_l and the iterations of finding $M_{l'}$ ($l \neq l'$) can overlap, and totally 7 pipelined iterations are needed to find 4 maximal size matchings.

In general, the number of pipelined iterations needed to find S maximal size matchings in pipelined RGA/RG MSM algorithms is given by $S + I - 1$, instead of $S \cdot I$ if a non-pipelined RGA/RG MSM algorithm is executed S times. It is important to point out that, our pipelined RGA/RG MSM algorithms do not require any additional hardware. The reason is that all arbiters associated with inputs/outputs of M_l are immediately released for use in finding M_{l+1}. Thus all arbiters are fully utilized.

In the following, we will study how much the scheduling time relaxation the proposed algorithms can achieve. Assume that L is the size of a cell, C is the line rate, then one cell slot $T_{cs} = L/C$. Since only two steps of RGA/GA MSM algorithms (Step 2 and 3 of RGA and Step 1 and 2 of RG) involve selection, we may use the scheduling time of each step to approximate the time for each iteration. Thus, the allowable scheduling time per step of a non-pipelined RGA/RG MSM algorithm is $T_{step} = \frac{T_{cs}}{2SI}$. For pipelined RGA/RG MSM algorithms, the allowable scheduling time per step is given by $T'_{step} = \frac{T_{cs}}{2(I+S-1)}$. The scheduling time

S	N=16	N=32	N=64	N=128
1	1	1	1	1
2	1.6	1.67	1.7	1.75
3	2	2.14	2.25	2.33
4	2.29	2.5	2.67	2.8

Table 1. Scheduling time relaxation factor pipelined RGA/RG MSM algorithms can achieve.

relaxation factor F is defined as

$$F = \frac{T'_{step}}{T_{step}} = \frac{SI}{I+S-1}. \quad (1)$$

For an $N \times N$ CIOQ switch, we set $I = \log_2 N$ for most RGA/RG MSM algorithms. The scheduling time relaxation factor is thus given by $F = \frac{S \log_2 N}{\log_2 N + S - 1}$. Table 1 shows the scheduling time relaxation factor vs. various S and N. The larger S or N is, the larger the scheduling time relaxation factor is. For example, assume $N = 16$, $L = 64 \times 8$ bits, $C = 10Gbps$, $I = \log_2 N = 4$, $S = 2$, we have $T_{step} = 3.2ns$ while $T'_{step} = 5.12ns$.

Combining our pipelined RGA/RG MSM algorithms with the pipelined technique proposed in [7], named *pipelined step* here, we can improve the scheduling time relaxation factor further. As illustrated in Figure 3, a combined pipelined RGA MSM algorithm works as follows, in the iterations of each scheduling cycle, the **Grant** step of the next iteration is done parallelly with the **Accept** step of the previous iteration. Similarly, one can derive how a combined pipelined RG scheduling algorithm works. The scheduling time of one step is thus relaxed to $T''_{step} = \frac{T_{cs}}{(2I+S-1)}$. The scheduling time relaxation factor of combined pipelined RGA/RG MSM algorithms is given by,

$$F' = \frac{T''_{step}}{T_{step}} = \frac{2SI}{(2I+S-1)}, \quad (2)$$

which is larger than F defined in Equation (1) for all $S > 1$. The pipelined step approach is an implementation technique, it has no impact on the performance of the scheduling algorithm. Thus we could implement a pipelined RGA/RG MSM algorithm combined with the pipelined step technique to achieve better scheduling time relaxation factor.

4. Performance Evaluation

The proposed pipelined MSM scheduling approach is applicable to all RGA and RG MSM algorithms, such as PIM [5], *i*SLIP [6], DRR [15], FIRM [16], etc. In the following, we will evaluate the performance of pipelined scheduling approach taking the example of pipelined PIM with fixed I for all scheduling cycles.

Performance evaluation of pipelined PIM is based on simulations of both Bernoulli traffic and bursty traffic. For Bernoulli traffic, we will consider both uniform arrivals and polarized arrivals. Polarized traffic is such a non-uniform traffic pattern that locally unbalanced but globally balanced. It is defined as follows [17].

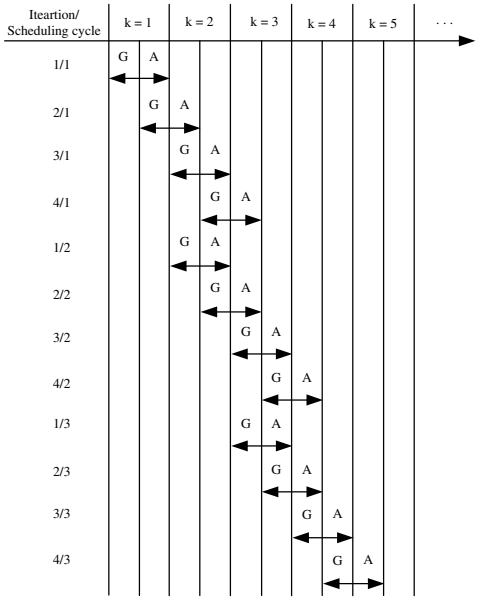

Figure 3. Timing diagram of a combined pipelined RGA scheduling algorithm.

We define $q \geq 1.00$ as the polarization factor and $d_{i,j}$ as the proportion of the traffic received by input port i with

$$d_{i,j} = \frac{q^{(i+j) \bmod N} \cdot (q-1)}{q^N - 1}$$

such that,

$$\forall i \in [1..N], \sum_{j=1}^{N} d_{i,j} = 1 \text{ and}$$

$$\forall j \in [1..N], \sum_{i=1}^{N} d_{i,j} = 1.$$

Polarized traffic with $q = 1.00$ is unform traffic. It is easy to verify that both uniform traffic and polarized traffic satisfy SLLN condition and no input/output is oversubscribed. According to [4], any maximal size matching scheduling algorithms with speedup 2 can achieve 100% throughput for any non oversubscribed SLLN traffic. Can pipelined PIM with $S = 2$ also achieve 100% throughput under such kind of traffic?

In the following, we present the performance of pipelined PIM with $S = 2$ in terms of average cell delay on a 16×16 switch. The cell delay is measured by the cell's waiting time at input queues plus the transmission time through the switching matrix. In the following figures, we use "pp" to represent pipelined PIM and "sp" to represent non-pipelined PIM.

Figure 4 shows the average cell delay of pipelined PIM with $S = 2$ under Bernoulli arrivals with the number of iterations allowed ranging from 1 to 4 and polarization factor varying from 1.00 to 2.00. As we can see from the figure, the average cell delay of pipelined PIM is improved when the number of iterations allowed in each scheduling cycle increases. With only one iteration,

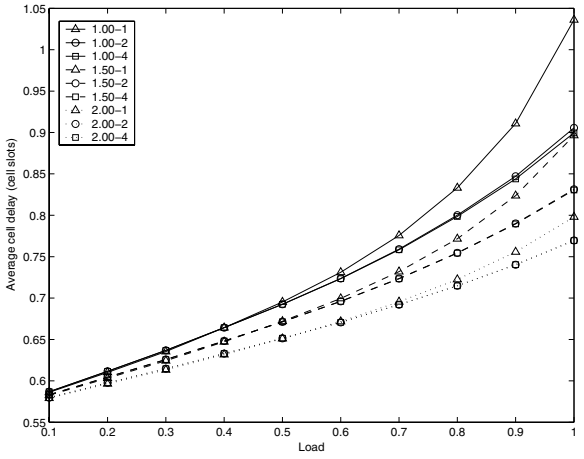

Figure 4. Delay performance of pipelined PIM with $S = 2$ under Bernoulli arrivals with different q's.

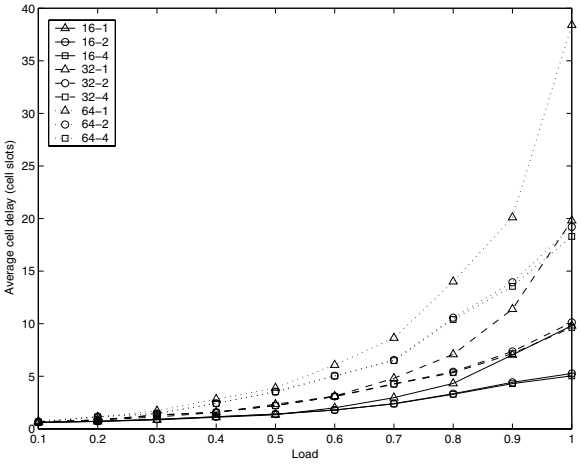

Figure 6. Delay performance of pipelined PIM with $S = 2$ under uniform bursty arrivals.

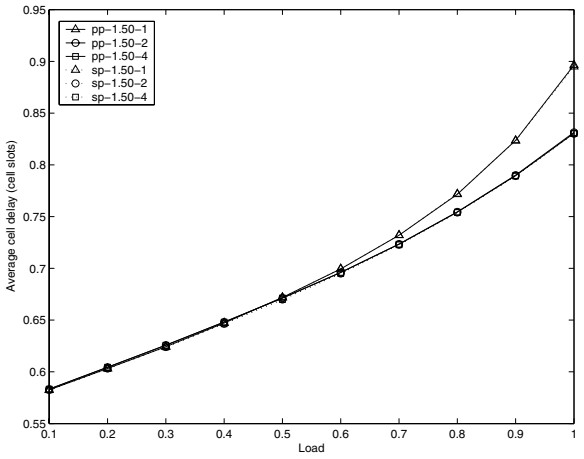

Figure 5. Delay performance of pipelined PIM (represented as pp) with $S = 2$ and non-pipelined PIM (represented as sp) with $S = 2$ under Bernoulli arrivals with $q = 1.50$.

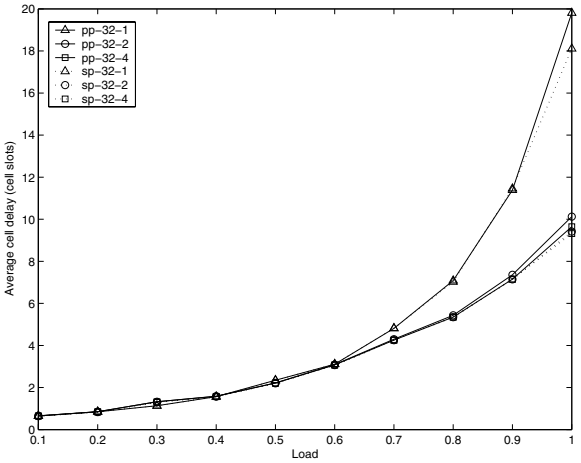

Figure 7. Delay performance of pipelined PIM (represented as pp) with $S = 2$ and non-pipelined PIM (represented as sp) with $S = 2$ under uniform bursty arrivals with $E(B) = 32$.

pipelined PIM with $S = 2$ can achieve 100% throughput for both uniform and polarized Bernoulli traffic.Figure 5 compares the average cell delay of pipelined PIM with $S = 2$ and non-pipelined PIM with $S = 2$ under Bernoulli arrivals with polarization factor set as 1.50. Pipelined PIM performs almost the same as non-pipelined PIM with $S = 2$. Both pipelined PIM with $S = 2$ and non-pipelined PIM with $S = 2$ achieve 100% throughput under both uniform traffic and polarized traffic.

We then study the performance of pipelined PIM under bursty traffic using 2-state modulated Markov-chain sources [6]. Each source alternately generates a burst of full cells (all with the same destination) followed by an idle period of empty cells. The number of cells in each burst or idle period is geometrically distributed. Let $E(B)$ be the average burst length in terms of number of cells, and $E(I)$ be the average idle length in term of number of cells. Then, $E(I) = E(B)(1-\rho)/\rho$, where ρ is the load of each input source. We assume the destination of each burst is uniformly distributed.

Figure 6 illustrates the performance of pipelined PIM under bursty arrival with average burst length ranging between 16, 32, and 64 and the number of iterations allowed set as 1, 2, and 4. As we can see, the increased number of iterations allowed leads to higher throughput and lower average cell delay at high load while the increased average burst length increases the average cell delay. Figure 7 compares the average cell delay of pipelined PIM with $S = 2$ and non-pipelined PIM with $S = 2$ under uniform bursty arrivals and average burst length set as 32. Pipelined PIM performs almost the same as non-pipelined PIM with $S = 2$. Although not shown here, both pipelined PIM and non-pipelined PIM with $S = 2$ achieve 100% throughput with all other average burst length settings.

5. Concluding Remarks

In this paper, we have proposed new pipelined RGA and RG maximum size matching algorithms to achieve speedup in CIOQ switches without using extra hardware. We derive that our pipelined scheduling approach reduces the scheduling time significantly compared to the non-pipelined approach with the same speedup factor. Combined with the pipelined step technique, we can further improve the scheduling time relaxation factor. Through simulations, we show that pipelined PIM achieves the same performance as non-pipelined PIM under both Bernoulli arrivals (both uniform and polarized) and bursty arrivals. In conclusion, the proposed pipelined RGA/RG MSM algorithms are practical and efficient solutions to achieve speedup on CIOQ switches. We believe that for large S it is possible to obtain faster scheduling algorithm by combining our approach with PMM [11].

References

[1] D. Gale, and L. S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly*, vol. 69, pp. 9-15, 1962.

[2] B. Prabhakar, N. Mckeown, "On the speedup required for combined input and output queued switching", *Automata*, vol. 35, 1999.

[3] S. T. Chuang, A. Goel, N. Mckeown, B. Prabhakar, "Matching output queueing with a combined input output queued switch", *IEEE Journal on Selected Areas in Communications*, vol. 17, No. 6, Jun. 1999, pp. 1030-1039.

[4] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup", *Proc. IEEE Infocom2000*, 2000, pp. 556-564.

[5] T. Anderson, S. Owicki, J. Saxie, and C. Thacker, "High speed switch scheduling for local area networks", *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319-352, Nov. 1993.

[6] N. Mckeown, "The *i*SLIP scheduling algorithm for input-queued switches", *IEEE/ACM Transactions on Networking*, vol. 7., no. 2, pp. 188-201, Apr. 1999.

[7] P. Gupta, N. Mckeown, "Designing and implementing a fast crossbar scheduler", *IEEE Micro. Magazine*, vol. 19, no. 1, pp. 20-28, Jan.-Feb. 1999.

[8] A. Smiljanic, R. Fan and G. Ramamurthy, "RRGS-round-robin greedy scheduling for electronic/optical terabit switches", *Proc. Globecom 1999*, 1999, pp. 1244-1250.

[9] D. Cavendish, "CORPS: A pipelined fair packet scheduler for high speed input queued switches", in *Proc. IEEE Conference on High Perforamnce Switching and Routing*, 2000, pp. 55-64.

[10] A. Motoki, S. Kamiya, R. Ikematsu, and H. Ozaki, "Group-pipeline scheduler for input buffer switch", *Proc. Joint 4th IEEE International Conference on ATM (ICATM 2001) and High Speed Intelligent Internet Symposium*, 2001, pp. 158-162.

[11] E. Oki, R. Rojas-Cessa, and H. J. Chao, "PMM: a pipelined maximal-sized matching scheduling approach for input-buffered switches", *Proc. of Globalcom 2001*, 2001, pp. 35-39.

[12] M. J. Karol, M. G. Hluchyj and S. P. Morgan, "Input vs. output queueing on a space-division packet switch", *IEEE Transaction on Communications*, vol. 35, no. 12, pp. 1347-1356, 1987.

[13] J. E. Hopcroft and R. M. Karp, "An $n^{2.5}$ algorithm for maximum matching in bipartite graphs", *Soc. Ind. Appl. Math. J.*, vol. 2, pp. 225-231, 1973.

[14] N. Mckeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% throughput in an input-queued switch", *IEEE Transactions on Communications*, vol. 47, pp. 1260-1267, 1999.

[15] H. J. Chao and J. S. Park, "Centralized contention resolution schemes for a large-capacity optical ATM switch", in *Proc. IEEE ATM Workshop*, 1998, pp. 11-16.

[16] D. N. Serpanos and P. I. Antoniadis, "FIRM: A class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues", *Proc. of IEEE Infocom 2000*, 2000, pp. 548-555.

[17] J. Blanton, H. Badt, G. Damm, and P. Golla, "Impact of Polarized Traffic on Scheduling Algorithms for High Speed Optical Switches", presented at ITCom2001, Denver, Aug. 2001.