A 2-Stage Matching Scheduler for a VOQ Packet Switch Architecture[1]

Ying Jiang
Suntek Technology Co., Ltd.
B4010, No.A38, Xueyuan Road, Haidian District
Beijing, 100083, China

Mounir Hamdi
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

***Abstract*- Virtual Output Queuing (VOQ) is a practical and high-performance packet switch architecture. There are many simple iterative arbitration algorithms proposed for the VOQ architecture. These algorithms either employ a 3-phase or a 2-phase hand-shaking scheme between the switch inputs and outputs. It has been shown that neither scheme can outperform the other in all traffic patterns. As a result, we propose a 2-stage matching algorithm that combines the benefit of both schemes, and also achieves the best desynchronization of arbiter pointers during their scheduling. We will demonstrate that this new algorithm outperforms all the other iterative algorithms considered under various traffic models. We also propose a possible hardware implementation method of the algorithm.**

I. INTRODUCTION

The traditional output queuing (OQ) switch architecture is appreciated for its optimal performance, and is frequently used as a yardstick by which the performance of newly proposed switch architectures is measured. OQ switches can always achieve 100% throughput since up to *N* packets can be transferred to a single output port in a time slot for an *N* x *N* switch. However, the high internal speed up, *N*, required makes it impractical to build these switches for a large number of ports and/or for high line rates. In contrast, input queuing (IQ) switches are designed to operate with a switching fabric running at an internal rate equal to the external link speed. Unfortunately, when using a first-in-first-out (FIFO) queuing discipline at the input queues, due to the head-of-the-line (HoL) blocking problem, they only provide a maximum throughput of 58.6% [1] under uniform traffic and much lower than that for other traffic patterns.

An architecture called virtual output queuing (VOQ) [2] is proposed to solve the HoL problem while achieving the scalability of IQ switches. Rather than maintaining a single FIFO queue for all cells, each input maintains a separate queue for the cells directed to different outputs. In this architecture, the switch performance essentially depends on its scheduling algorithm, that is, the arbitration between the input ports and the output ports. A good scheduling algorithm should achieve high performance, work with very high line rates and/or large number of input/output ports, and be simple to implement.

[1] This research was conducted as part of the MPhil thesis of Miss Jiang at HKUST, and has been supported in part by a grant from the Hong Kong Research Grants Council under the Grant HKUST6202/99E.

Many scheduling algorithms for VOQ switch architectures have appeared in literature. They can be classified into two types: approximating maximum size matching (MSM) and approximating maximum weight matching (MWM). Although the MWM algorithms have been proven to achieve 100% throughput under any admissible traffic patterns [4][5], they are too complex to be implemented in hardware and have a time complexity of ($O(N^3 logN)$). We instead concentrate in this paper on a group of more practical iterative algorithms, including iSlip [6], DRRM [7], FIRM [8], etc., and evaluate their performance. The iterative algorithms can use a 2-phase or a 3-phase hand-shaking scheme. After exploring the difference between the two modes, we propose a 2-stage matching algorithm that combines the benefit of both, and also achieves the best desynchronization of arbiter pointers during their scheduling. This new algorithm outperforms all the other iterative algorithms considered under various traffic models. We also study its stability property and propose a possible hardware implementation of the algorithm.

The rest of the paper is organized as follows. Section II introduces various algorithms in the literature and explains the difference between 2-phase and 3-phase algorithms. Based on a static-pointer scheme, we propose the 2-stage arbitration algorithm in Section III. Section IV details the simulation results. A possible hardware design is given in Section V and the last Section concludes the paper.

II. BACKGROUND KNOWLEDGE

A. Maximum Size Matching

The scheduling problem in an *N x N* crossbar switch can be modeled using a bipartite graph (See Fig. 1), where each part of the graph contains *N* nodes, and one part corresponds to the input ports, while the second part corresponds to the output ports. The requests from the input ports to the corresponding output ports are represented as edges, creating a bipartite graph. The maximum size matching (MSM) for a bipartite graph can be found by solving an equivalent network flow problem [10] and the algorithm is called

0-7803-7400-2/02/$17.00 © 2002 IEEE

maxsize here.

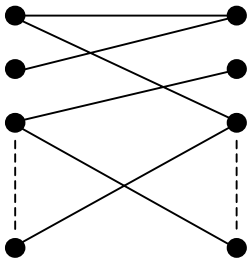

Fig. 1. Bipartite graph.

The most efficient algorithm converges in $O(N^{5/2})$ time [10]. Although it is guaranteed to find a maximum match, it is too complex to implement in hardware and takes a long time to converge. Instead, simple iterative algorithms are often used to approximate MSM.

B. Simple Iterative Algorithms: iSlip & FIRM

There exists a group of algorithms that are easy to implement in hardware. They only take into consideration whether each VOQ is occupied or not, to make the scheduling decision and try to approximate MSM. The calculation of the matching is performed in an iterative fashion, where each additional iteration augments the matching calculated in the previous iteration. During each iteration, the inputs send requests to the outputs, then each output selects one request independently and issues a grant to it, and finally each input chooses one grant to accept. In each of the following iterations, only the unmatched inputs and outputs are considered. We now introduce two well-known iterative scheduling algorithms: iSlip and FIRM.

iSlip was first described in [6]. The main characteristic of iSlip is its simplicity: it is readily implemented in hardware and can operate at high speed. Its performance is also good. For uniform i.i.d. (independently identically distributed) Bernoulli arrivals, iSlip is stable for any admissible load. It means that 100% throughput can be achieved. iSlip works in this way:

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

Step 2. Grant. If an output receives any requests, it chooses the one that appears next in a fixed, round-robin schedule starting from the highest priority element. The output notifies each input whether or not its request was granted. The pointer to the highest priority element of the round-robin schedule is incremented (module *N*) to one location beyond the granted input if and only if the grant is accepted in Step 3.

Step 3. Accept. If an input receives a grant, it accepts the one that appears next in a fixed round-robin schedule starting from the highest priority element. The pointer to the highest priority element of the round-robin schedule is incremented (module *N*) to one location beyond the accepted one.

TABLE I
POINTER UPDATING SCHEMES

		iSlip	FIRM
Input	No grant	unchanged	
	Granted	one location beyond the accepted one	
Output	No request	unchanged	
	Grant accepted	one location beyond the granted one	
	Grant not accepted	unchanged	the granted one

Fcfs In Round Robin Matching (FIRM) was proposed in [8]. This algorithm is almost the same as iSlip. We list in TALBE I, the difference between iSlip and FIRM in updating their pointers, which is the main difference between them. The updating scheme plays an important role in determining the performance of the algorithm. In [8], it is claimed that FIRM provides improvement over iSlip in average delay which reaches approximately 50% at loads above 95%. The reason is that FIRM has a better pointer desynchronization effect than iSlip under high load of uniform traffic.

C. 3-Phase and 2-Phase Algorithms

The above-mentioned two algorithms are 3-phase algorithms: each iteration includes 3 steps: request, grant and accept.

Dual Round Robin Matching (DRRM) scheme [7] is rather a 2-phase algorithm: grant and accept. The secret of removing one step is that each input makes only one grant to the outputs instead of sending out all the requests. Each input chooses one among the non-empty VOQs to send a grant and each output chooses one among the grants received to accept, using the same (pointer-based) scheme in iSlip. The advantage of DRRM is that it requires less control-information transmission time.

Logical Equivalence of Parallel Iterative Matching (LE-PIM) proposed in [3] is similar to DRRM in that it's also a 2-phase algorithm. The difference between LE-PIM and DRRM is mainly in resolving contentions: the former uses random selection, while the latter uses round-robin selection.

Besides simplicity in implementation, let's also consider how a 2-phase arbitration scheme differs from a 3-phase arbitration scheme in terms of performance. We first show some simulation results. The simulation results are gathered from a 32*x*32 switch. Delay is only considers the period of time a cell spends waiting in a VOQ before being transmitted. Each point in the figures runs for 500,000 time slots, and the statistics are gathered from the $50{,}000^{th}$ time slot. The average delay is got from all the cells leaving the output links during this period of time. Relative average delay is the average delay divided by the average delay of an OQ switch under the same traffic. By normalized load, we mean the percentage of time slots that have cells coming in, averaged over all the inputs.

We evaluated 4 algorithms: iSlip, DRRM (which can be considered as a 2-phase iSlip), FIRM, and 2-FIRM (2-phase FIRM). TABLE II lists the pointer updating schemes of DRRM and 2-FIRM.

Fig. 2 shows the results under uniform traffic. Under this traffic model, the packets are Bernoulli arrivals, i.i.d., and the destinations are uniformly distributed over all the outputs. We can see that the performance of iSlip and DRRM is almost the same, and so for FIRM and 2-FIRM. The performance of FIRM and 2-FIRM is better than that of iSlip and DRRM because of better desynchronization of the pointers. It's understandable since 2-phase and 3-phase are symmetric, only exchanging the functions of the input and output schedulers. Under uniform traffic, the traffic is also symmetric in average, resulting in similar performance. How about non-uniform traffic?

TABLE II
POINTER UPDATING SCHEMES

		DRRM	2-FIRM
Input	No traffic	unchanged	
	Grant accepted	one location beyond the granted one	
	Grant not accepted	unchanged	the granted one
Output	No grant	unchanged	
	Granted	one location beyond the accepted one	

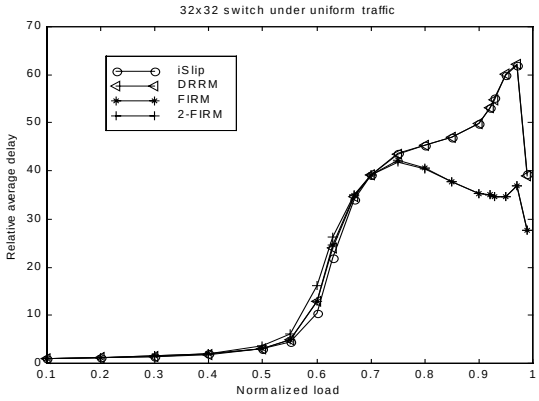

Fig. 2. Relative average delay as a function of normalized load for a 32x32 switch under uniform traffic.

Hotspot is a special non-uniform traffic model with an example pattern as follows (for a 4 x 4 switch):

$$\begin{pmatrix} 2x & x & x & x \\ 2x & x & x & x \\ 2x & x & x & x \\ 2x & x & x & x \end{pmatrix}$$

One of the outputs gets twice as many packets as the others and is called the "hotspot" (output 0 here). Fig. 3 shows the simulation results under hotspot traffic. We can see large difference between 2-phase and 3-phase algorithms. FIRM algorithms (FIRM and 2-FIRM) show no obvious advantage over iSlip algorithms (iSlip and DRRM). We will explain why 2-phase algorithms outperform 3-phase ones in this case. Under hotspot traffic, the loads on different inputs are the same, so it's better to give each input the same chance to be served. One of the outputs (let's assume output 0) is more heavily loaded than the others, so it's better to give it more service. If a 3-phase scheme is used, output 0 only sends one grant, and if this grant is not accepted, it will not be served. For a 2-phase algorithm, since there is more traffic towards output 0, each input has a larger chance of granting output 0, and as long as there is one input granting output 0, output 0 will be served. So we can see that a 2-phase algorithm gives the hot-spotted port more service if it is on the output side,

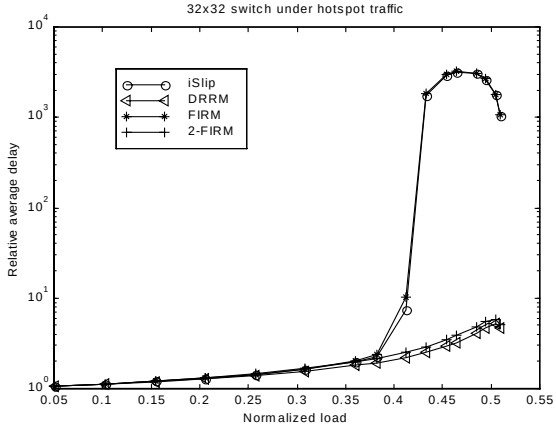

Fig. 3. Relative average delay as a function of normalized load for a 32x32 switch under hotspot traffic.

thus decreasing the non-uniformity. We can also conclude that in the general case, especially when the traffic model changes from time to time, there will be temporary non-uniformity on the input or output side. When it is on the input side, it's better to use a 3-phase scheme, else to use a 2-phase scheme.

III. THE 2-STAGE ALGORITHM

In this Section, we propose a new algorithm: A 2-stage matching algorithm. We will also introduce a new pointer-updating scheme which achieves the best pointer desynchronization. It's based on the *Static Round Robin* (SRR) arbitration algorithms [12] that we previously proposed, and is here applied to the 2-stage scheme.

A. A Fully Desynchronized Pointer-Updating Scheme

From Section II.B, we know that keeping pointers desynchronized is important. Because of the better desynchronization of FIRM than iSlip, FIRM performs better than iSlip under uniform traffic. [12] proposes a way to achieve the best pointer desynchroniztion effect. It is achieved by forcing the pointers to be always different. That is, the pointers at the output/input side are set totally different

at the beginning and advance synchronously at each time slot. A possible configuration could be (for a 4 x 4 switch):

TABLE III
A POINTER CONFIGURATION TO FORCE THE POINTERS TO BE DESYNCHRONIZED

	Input/ Output 0	Input/ Output 1	Input/ Output 2	Input/ Output 3
Pointers at $4i^{th}$ time slot	3	2	1	0
Pointers at $(4i+1)^{th}$ time slot	0	3	2	1
Pointers at $(4i+2)^{th}$ time slot	1	0	3	2
Pointers at $(4i+3)^{th}$ time slot	2	1	0	3

The algorithms using this scheme and the common 3-stage model are called *Static Round Robin* (SRR) in [12]. There are 3 algorithms: single SRR (SSRR), double SRR (DSRR) and rotating DSRR (RDSRR). SSRR means only the output pointers are forced to be desynchronized; DSRR means that both the input and output pointers are desynchronized. DSRR has better performance than SSRR and both are much better than iSlip and FIRM. However, both have the problem of unfairness among inputs under some traffic patterns, so a rotating pointer scheme is used for DSRR to solve it, and thus RDSRR.

In this paper, we only want to use the pointer-updating scheme and try to desynchronize the pointers at both sides. So, the pointer updating in our algorithm will be the same as DSRR. Pay attention when both-side pointers are desynchronized, the choice of the pattern is important. The inputs and outputs are required to have a "mutual selection" relation. For example, if we configure both the output and input pointers as in TABLE III, at the $4i^{th}$ time slot, output 0 selects input 3 as its highest priority input and at the same time input 3 also selects output 0 as its highest priority output. The same relationship exists for all pairs of input and output and always exists at any time slot.

B. The 2-Stage Matching Algorithm

Remember we have discussed in section II.C that using 3-phase or 2-phase (that is, let the output or the input side make the main decision) helps reducing non-uniformity on the input or the output side. This is the basic idea of our algorithm.

The 2-stage algorithm works in the following way:

1. The pointers at both sides are maintained in the way mentioned in section III.A. That is, they are always kept desynchronized.

2. In each iteration, there are 3 steps:

Step 1. Each input sends a request to every output for which it has a queued cell.

Step 2. Each input selects one VOQ to send a grant that appears next from its highest priority output. Each output selects one request received in Step 1 to grant that appears next from its highest priority input. OutputCount = number of outputs receiving grants from inputs. InputCount = number of inputs receiving grants from outputs.

Step 3. If OutputCount ≥ InputCount, each output selects one among the grants received in Step 2 which appears next from its highest priority input and sends an accept. Else, each input selects one among the grants received in Step 2 which appears next from its highest priority output and sends an accept.

In simple words, this algorithm decides in each time slot whether to use a 2-phase or a 3-phase scheme based on which one can make more matches.

IV. SIMULATION RESULTS

We evaluated the performance of our new algorithm by comparing it with the algorithms mentioned previously. Besides uniform and hotspot, we also considered bursty and unbalanced traffic models. For bursty traffic, busy and idle periods appear alternatively; in a busy period, there is a cell arriving in each time slot; in an idle period, there is no cell arriving in any time slot. The average loads for the inputs are the same, and the destinations are uniformly distributed burst by burst over all outputs. The traffic matrix of unbalanced traffic is as follows (for a 4 x 4 switch):

$$\begin{pmatrix} x & x & 0 & 0 \\ 0 & x & x & 0 \\ 0 & 0 & x & x \\ x & 0 & 0 & x \end{pmatrix}$$

The traffic is concentrated on two "diagonals" and each flow is Bernoulli.

Fig. 4 shows the results under uniform traffic. We can see that the 2-stage is better than iSlip, FIRM and DSRR. Since the traffic is uniform, the benefit brought by considering both 2-phase and 3-phase is not too much, but it still outperforms DSRR.

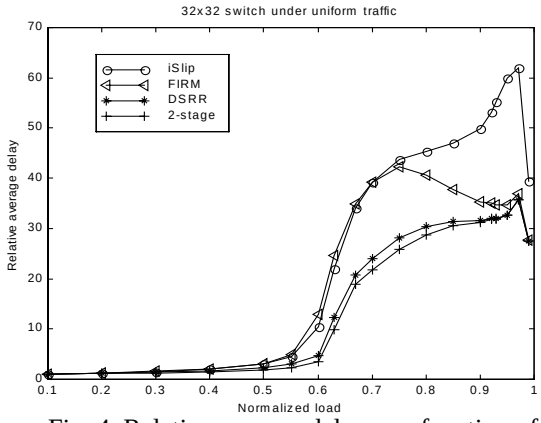

Fig. 4. Relative average delay as a function of normalized load for a 32x32 switch under uniform traffic.

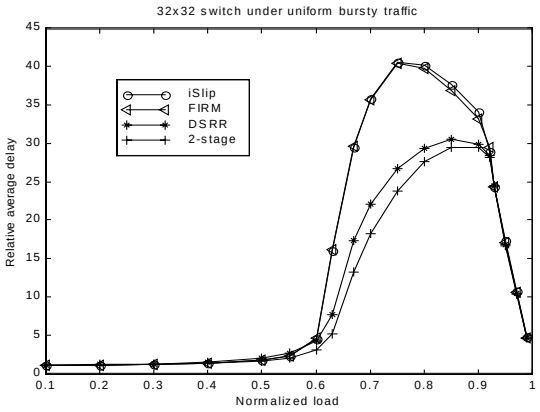

Fig. 5. Average delay as a function of normalized load for a 32x32 switch under bursty traffic.

Fig.5 shows the results under bursty traffic. The case is similar to the uniform case. 2-stage shows more advantage, since there is more temporary non-uniformity among inputs or outputs under bursty traffic.

Under hotspot traffic (Fig. 6), we show here some 2-phase algorithms for comparison. DRRM and 2-FIRM have been mentioned. 2-DSRR is the 2-phase version of DSRR. We know that when the non-uniformity is on the output side, 2-phase algorithms perform much better than their 3-phase counterparts. We show here 2-stage is even obviously better than all the 2-phase algorithms considered.

Under unbalanced traffic (Fig. 7), DSRR is unstable under high load (>0.8). 2-stage is much better than iSlip and FIRM, especially under high load. For example, the relative average delay of 2-stage is kept under 6 while that of FIRM can be more than 100.

From the above results, we can see that our new algorithm, 2-stage matching, is generally much better than both 2-phase and 3-phase algorithms, no matter what kind of traffic model is used. In fact, the traffic models we have considered cover most cases, even some extreme cases.

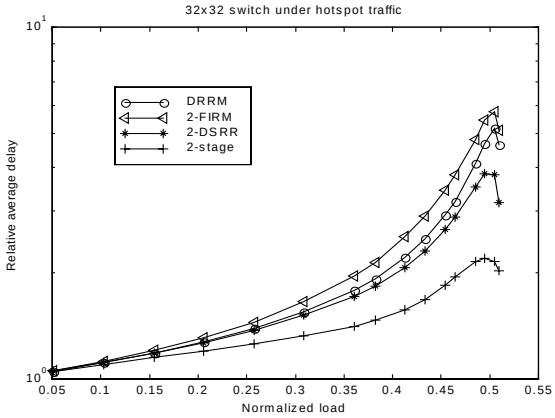

Fig. 6. Relative average delay as a function of normalized load for a 32x32 switch under hotspot traffic.

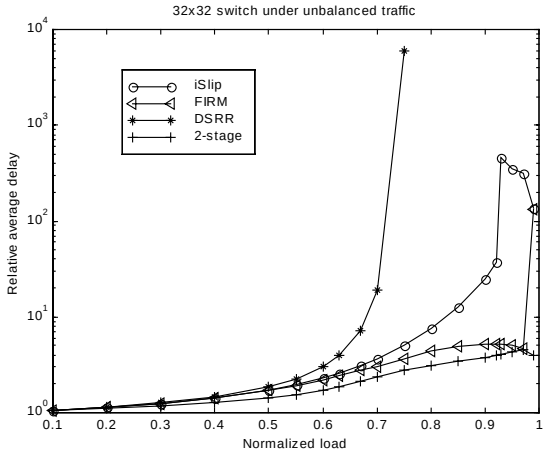

Fig. 7. Average delay as a function of normalized load for a 32x32 switch under unbalanced traffic.

V. HARDWARE IMPLEMENTATION

We have shown that the 2-stage matching has very good performance. In this Section, we will show it is also easy to be implemented in hardware.

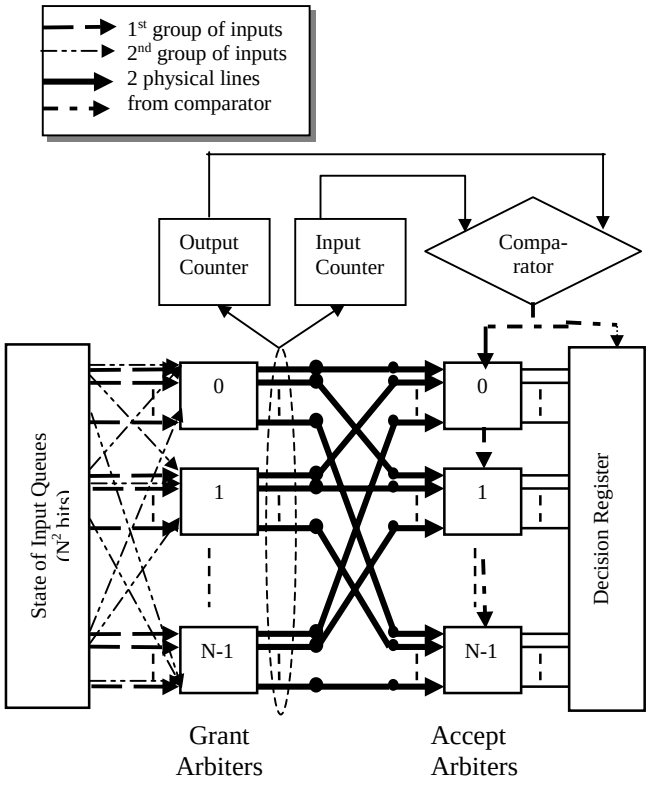

Fig. 8. Hardware implementation of 2-stage matching arbitration algorithm.

Fig. 8 shows a possible implementation design of the 2-stage matching algorithm. It includes *N* grant arbiters and *N* accept arbiters. Each grant arbiter has two groups of inputs. For arbiter *i*, the first group of inputs is the requests from input *i*; the second is the requests towards output *i*. There are two round-robin schedulers inside an arbiter, outputting two sets of results: grants to outputs and to inputs. Each dark line in the figure corresponds to 2 physical lines: one is an input grant; the other is an output grant. The two sets will be inputted to the output and input counters respectively. The two counters will calculate the number of outputs and inputs receiving grants respectively. The comparator outputs the information of which number is larger, which controls the operation of the accept arbiters. Each accept arbiter can choose to make the input or the output accept the decision under the control of the comparator. If the comparator shows that OutputCount is larger, each arbiter will take the input grants (grants from the input side) and accept one, and vice versa. Each arbiter only maintains one register to hold the pointer as the input and output with the same number shares a pointer.

For each grant arbiter, the input and output grants work in parallel. For each accept arbiter, it will only choose one task to do: input or output accept. These two stages are of the same time complexity as the corresponding parts of the iSlip design. Output/input counter and comparator only add little to the total time.

VII. Conclusion

In this paper, we have introduced various iterative algorithms proposed in the literature for the VOQ packet switch architecture. They are divided into two groups: 3-phase arbitration algorithms and 2-phase arbitration algorithms. 3-phase algorithms include iSlip, FIRM, etc. 2-phase algorithms include DRRM, LE-PIM, etc. By symmetry, each 3-phase algorithm has its 2-phase counterpart and vice versa. 3-phase algorithms perform better when the main non-uniformity is on the input side and 2-phase algorithms perform better when it is on the output side. In this paper, we have proposed a 2-stage matching algorithm which decides in each iteration which hand-shaking mode to use according to which can make more matches. The pointer-updating scheme is based the fully-desynchronized SRR arbitration algorithms that we proposed previously. This scheme achieves the best pointer desynchronization and helps to improve the performance. We also propose a practical hardware design of low complexity.

References

[1] M.J. Karol, M. G. Hluchyj, and S.P. Morgan, "Input Versus Output Queuing on a Space-Division Packet Switch," *IEEE Transactions on Communications*, 35:1347-56, 1987.

[2] T. Anderson, S. Owicki, J. Saxe, and C. Thacker, "High Speed Switch Scheduling for Local Area Networks, " *ACM Trans. Comput. Syst*., pp. 319-52, Nov. 1993.

[3] 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.

[4] N. Mckeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-Queued Switch," *IEEE Transactions on Communications*, 47: 1260-67, Aug. 1999.

[5] A. Mekkittikul, and N. Mckeown, "A Practical Scheduling Algorithm to Achieve 100% Throughput in Input-Queued Switches," *IEEE INFOCOM 98*, San Francisco, April 1998.

[6] N. McKeown, "Scheduling Cells in an Input-Queued Switch," PhD thesis, University of California at Berkeley, May 1995.

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

[8] D. N. Serpanos, and P. I. Antoniadis, "FIRM: A Class of Distributed Scheduling Algorithms for High-speed ATM Switches with Multiple Input Queues," in *Proc. INFOCOM*, 2000.

[9] R. E. Tarjan, "Data Structures and Network Algorithms," *Society for Industrial and Applied Mathematics*, Pennsylvania, Nov. 1983.

[10] J. E. Hopcroft, R. M. Karp, "An $n^{5/2}$ Algorithm for Maximum Matching in Bipartite Graphs," *Society for Industrial and Applied Mathematics Journal of Computation*, vol.2 pp.225-31, 1973.

[11] N. McKeown, M. Izzard, A. Mekkittikul, and M. Horowitz, "The Tiny Tera: A Small High-Bandwidth Packet Switch Core," *IEEE Micro Magazine*, vol.17, No. 1, pp. 26-33, Jan-Feb, 1997.

[12] Y. Jiang, and M Hamdi, "A Fully Desynchronized Round-Robin Matching Scheduler for a VOQ Packet Switch Architecture," *Proc. IEEE Workshop on High Performance Switching and Routing*, 2001.