A fast scheduling scheme for hundred-percent-throughput input-buffered ATM switch

A.E. Abdel Naiem [a], M. Fouad [a], K. Hussein[b] and N. Monir[a]
[a] *Electrical Eng. Dept., Faculty of Eng., Zagazig University, Zagazig, Egypt*
[b] *Electronics Research Institute*

The original Parallel Iterative Matching (PIM) algorithm that runs on a Virtual Output Queuing (VOQ) based Asynchronous Transfer Model (ATM) switch performs a number of iterations, during the time slot, so that more inputs are matched in each iteration. To obtain 100% throughput an expected number of $O(\log_2 N)$ iterations is required. Doing such a number of iterations in one time slot may be impossible for large switches operating at high speed. A new scheduling scheme for VOQ-based ATM switches is proposed in the present work to achieve 100% throughput without doing iterations except the first one. The proposed scheme depends on a parallel matching algorithm that runs for only one iteration and then passes the excessive grants from inputs receiving multiple grants (multi-granted inputs) to the requesting inputs that have not received any grant (ungranted inputs). Thus only one iteration is performed and then all the inputs are matched. Naturally, this is accomplished on the expense of adding some complexity to the switch circuitry

يقدم البحث نموذجا جديدا لنظام التوافق في محولة نقل غيرمتزامن ذات طوابير خرج تخيلية. يعتمد النظام الأصلي على القيام بتكرار عدد من المحاولات أثناء نفس الشريحة الزمنية لكي يتم توافق أكبر عدد من المدخلات في كل تكرار. والمشكلة هنا أنة قد يكون من المستحيل عمليا إنجاز العدد اللازم من هذه التكرارات في وقت الشريحة الزمنية الواحدة وخاصة لمحولات النقل ذات الحجم الكبير والتي تعمل بسرعة عالية. يقدم البحث مقترحا جديدا يعتمد على إجراء محاولة توافق واحدة يتم بعدها تحديد عدد الاستجابات الحاصل عليها كل دخل بواحدة فقط وتمرير الاستجابات الزائدة إلى المدخلات الأخرى من خلال دوائر مصممة لذلك حتى يتم الحصول على إنتاجية ١٠٠%.

Keywords: Asynchronous Transfer Mode (ATM), Parallel Iterative Matching (PIM), Virtual Output Queuing (VOQ), Throughput

1. Introduction

In the first few years of deploying Asynchronous Transfer Model (ATM) switches, output-buffered switches including shared-memory switches dominated the market [1]. However, as the demand for large capacity switches increases rapidly either line rates or the switch port number increases, the speed requirement for the memory must increase accordingly. According to Moore's law, memory density doubles every 18 months. But the memory speed increases at a much slower rate. For instance, the memory speed in 2001 is 5 ns for state-of-the-art CMOS static RAM, compared with 6 ns in 1999.

On the other hand, the speed of logic circuits increases at a higher rate than that of memory. This limits the capacity of output-buffered switches.

Therefore, in order to build larger-scale and higher-speed switches, people have focused on input-buffered or combined input-output-buffered switches with advanced scheduling and routing techniques.

Input-buffered switches have two problems [2]: 1) throughput limitation due to the Head-Of-Line (HOL) blocking and 2) the need of arbitrating cells due to output port contention. The first problem can be circumvented by moderately increasing the switch fabric's operation speed or the number of routing paths to each output port i.e., allowing multi-ple cells to arrive at the output port in the same time slot. The present work focuses on resolving the second problem by a novel, fast scheme that will be described in this paper.

Recently, much research has been devoted to devising fast scheduling schemes to arbitrate cells from input ports to output ports. Here are the factors used to compare different scheduling schemes: (1) throughput, (2) delay, (3) fairness for cells, independent of port positions, (4) implementation complexity, and (5) scalability as the line rate or the switch size increases. Furthermore, some scheduling schemes even consider per-flow .Scheduling at the input ports to meet the delay-throughput requirements for each flow, which of course greatly increases implementation complexity and cost. Scheduling cells on a per-flow basis at input ports is much more difficult than at output ports. For example, at an output port, cells or packets can be time-stamped with values based on their allocated bandwidth and transmitted in ascending order of their timestamp values. However, at an input port, scheduling cells must take output port contention into account. This makes the problem so complicated that so far no feasible scheme has been devised.

2. VOQ-based ATM switches

To alleviate the HOL blocking in input-buffered switches is for every input to provide a single and separate FIFO for each output. Such a FIFO is called a Virtual Output Queue (VOQ), as shown in fig. 1. For example, VOQ(*i,j*) stores cells arriving at input(*i*) and destined for output(*j*). Each input sends a request to every output for which it has a queued cell. If an output receives multiple requests, it grants one by randomly selecting a request over all requests. Each request has equal probability

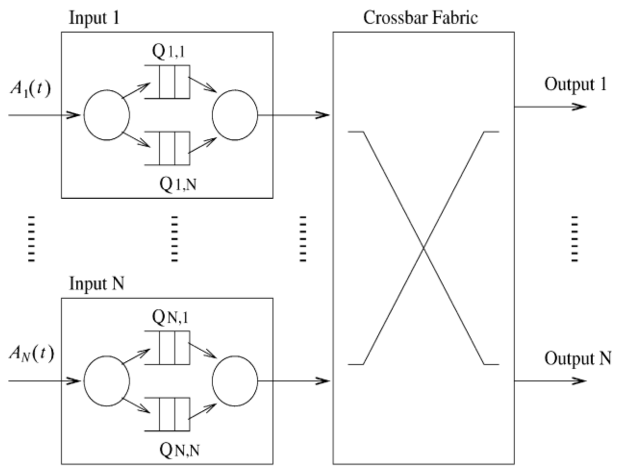


Fig. 1. Virtual Output Queue (VOQ) at the input port [1].

to be granted. If an input receives multiple grants, it accepts one by randomly selecting an output among them. Thus, with virtual output queuing, an input may have cells granted access by more than one output. Since each input can transfer only one cell in a time slot, the others have to wait, and their corresponding outputs will be idle, which limits the throughput. In more intelligent schemes, matching methods can be applied to have the optimal scheduling.

3. Original parallel iterative matching

The inefficiency of the VOQ-based switch that limits the throughput as discussed above can be alleviated if the request-grant-accept algorithm runs iteratively. This method is known as Parallel Iterative Matching (PIM) [1], [3-5].

The PIM scheme uses random selection to solve the contention in inputs and outputs. Input cells are first queued in VOQs. Each iteration consists of three steps. All inputs and outputs are initially unmatched, and only those inputs and outputs that are not matched at the end of an iteration will be eligible to participate in the next matching iteration. The three steps in each iteration operate in parallel on each input and output as follows:

1. Each unmatched input sends a request to every output for which it has a queued cell.
2. If an unmatched output receives multiple requests, it grants one by randomly selecting a request over all requests. Each request has equal probability to be granted.
3. If an input receives multiple grants, it accepts one by randomly selecting an output among them.

It is shown later in the present work that, on average, 63% of the remaining grants will be matched in each iteration. Thus, the algorithm converges at $O(\log_2 N)$ iterations. Because of the random selection, it is not necessary to store the port number granted in the previous iteration. An example of parallel iterative matching is shown in fig. 2, where $L(x, y, z)$ means that there are z cells on the VOQ(x, y).

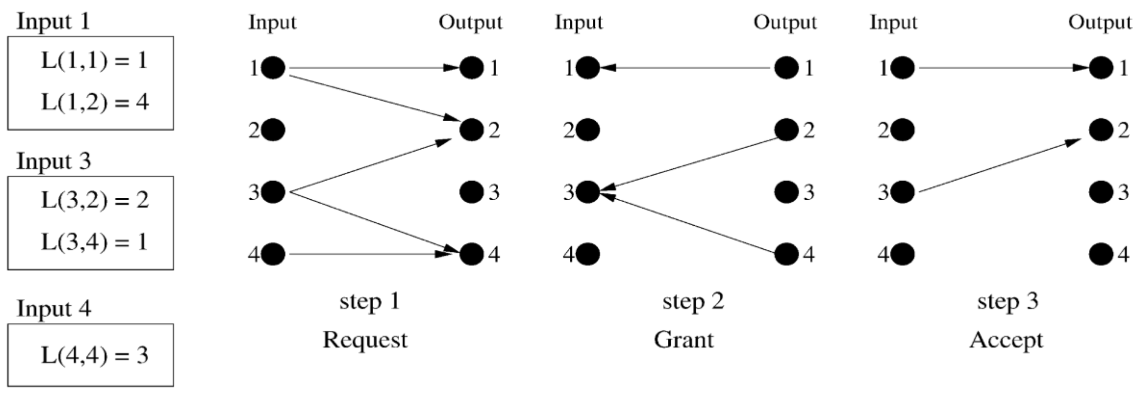


Fig. 2. Resolving contention problems in parallel iterative matching VOQ-based switches [1].

However, implementing a random function at high speed may be too expensive [3]. One drawback of the PIM algorithm is that it should be executed to produce results within the time it takes to transmit one ATM cell. For example, at 1-Gb/s link speed, this time is less than 0.5 μs. In particular, for large switch sizes and using a large number of PIM iterations, it is difficult to produce results within this short time even when using expensive state of art microprocessors or special –purpose hardware.

4. Evaluation of throughput of VOQ-based ATM switches

In the following, the performance of the VOQ-based switch is investigated using both analytical and computer simulation. The results concerning the throughput are obtained analytically and compared with the simulation results.

4.1. Evaluation of VOQ-based switch throughput analytically

It is assumed that all the VOQs are saturated i.e. each virtual output queue has at least one cell in every time slot. Thus each input sends N requests (a request for each output). Each request has an equal probability to be granted by the corresponding output. The probability that an input is granted by a definite output is $(1/N)$. Let $N_g(i)$ denote the number of grants received by input i. The probability that an input receives m grants can be expressed as

$$\Pr\left[N_g(i) = m\right] = \frac{1}{N^m}\left(1-\frac{1}{N}\right)^{N-m} \frac{N!}{(N-m)!\,m!} \;. \qquad (1)$$

The probability that an input is granted by one or more outputs

$$\Pr\left[N_g(i) \geq 1\right] = \sum_{m=1}^{N} \Pr\left[N_g(i) = m\right]$$
$$= \sum_{m=1}^{N} \frac{1}{N^m}\left(1-\frac{1}{N}\right)^{N-m} \frac{N!}{(N-m)!\,m!} \;. \qquad (2)$$

Adding the term corresponding to $(m=0)$ to the series in (2) and then subtracting its value, which is obtained by setting $m=0$ in (1), eq. (2) can be rewritten as follows

$$\Pr\left[N_g(i) \geq 1\right] = \left[\sum_{m=0}^{N} \frac{1}{N^m}\left(1-\frac{1}{N}\right)^{N-m} \frac{N!}{(N-m)!\,m!}\right] - \left(1-\frac{1}{N}\right)^{N} . \qquad (3)$$

It can be shown that the series summation between the square brackets in (3) is equal to one. Thus, (3) is reduced to

$$\Pr\left[N_g(i) \geq 1\right] = 1 - \left(1-\frac{1}{N}\right)^{N} . \qquad (4)$$

The expression in (4) can be obtained in a simpler way by considering the probability that an input is not granted by any output. This probability can be obtained by setting $m = 0$ in (1);

$$\Pr\left[N_g(i) = 0\right] = \left(1 - \frac{1}{N}\right)^N . \tag{5}$$

Thus the probability that an input is granted (by at least one output) can be expressed as

$$\Pr\left[N_g(i) \neq 0\right] = 1 - \left(1 - \frac{1}{N}\right)^N . \tag{6}$$

It is clear that the expression (6) is the same as (4). This probability actually represents the expected value of the ratio of the number of matched inputs to the total number of inputs in a time slot.

The number of granted inputs *Gi* in each time slot is a random variable whose expected value can be expressed as

$$Gi = \Pr\left[N_g(i) \geq 1\right].\, N = N\left[1 - \left(1 - \frac{1}{N}\right)^N\right]. \tag{7}$$

The number of granted inputs in each time slot is equal to the number of the outputs that are utilized in this time slot. Hence, the throughput of the switch can be evaluated as follows.

$$\begin{aligned} Throughput &= \frac{Number\ of\ utilized\ outputs}{Total\ number\ of\ outputs} \\ &= \frac{Gi}{N} = \frac{\Pr\left[N_g(i) \geq 1\right].\, N}{N} \\ &= 1 - \left(1 - \frac{1}{N}\right)^N . \end{aligned} \tag{8}$$

Fig. 3 shows plots for the throughput of input buffered switches with VOQs, where formula (8) was used for obtaining the throughput of switches of different sizes.

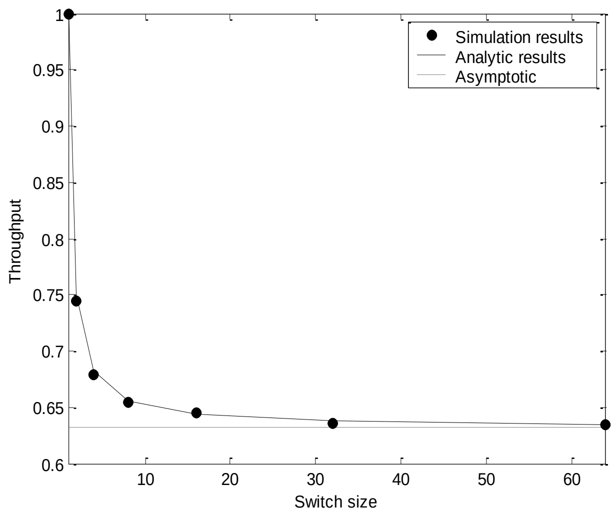


Fig. 3. Throughput of VOQ-based switches of different sizes.

4.2. Evaluation of the throughput of VOQ-based switch using simulation

Fig. 3 shows a plot for the throughput of VOQ-based switches of different sizes obtained using both the analytic formula (8) and the computer simulation. The results of both methods coincide for different switch sizes as shown in the figure. It is clear that the throughput approaches 63% for large switch size.

5. Evaluation of throughput of PIM-based ATM switches using simulation

Based on simulation results, and using PIM, the number of unmatched inputs after each iteration, are shown in fig. 4 for switches of different sizes.

5.1. Parallel matching scheme with output-grant passing

Let us consider a VOQ-based ATM switch subjected to PIM scheduling algorithm and consider that the number of iterations is limited to only one. Due to the random selection of a single request to be granted by an output, an input may receive grants from more than one output. However, the granted input has to select only one grant. As there is no other iterations, an accepted output can receive a cell during the present time slot, whereas the other (unaccepted) outputs remain idle. This limits the throughput of the large switches to 63% as discussed before. To improve the throughput, the original PIM algorithm performs extra iterations (during the same time slot) so that more inputs are matched in each iteration. To obtain 100% throughput, the iterations should be repeated until all the inputs are matched. It has been shown that, to get 100% throughput an expected number of $O(\log_2 N)$ iterations is required. If the number of iterations truncated before matching all the inputs, the obtained throughput depends on the number of iterations as shown in fig. 4. However, in the worst case, the number of iterations required to obtain 100% throughput may be exactly *N*. Doing such a large number of iterations in one time slot may be impossible for large switches operating at high speed.

A new scheme is proposed in the present work to achieve 100% throughput of a VOQ-based ATM switch without doing iterations except the first one. The proposed scheme depends on a parallel matching algorithm that runs for only one iteration and then passes the excessive grants from inputs receiving multiple grants (multi-granted inputs) to the requesting inputs that have not received any grant (ungranted inputs). Thus only one iteration is performed and then all the inputs

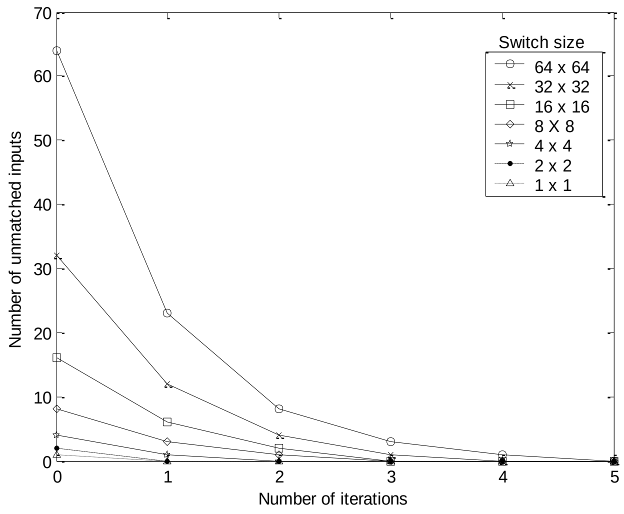


Fig. 4. Number of unmatched inputs after each iteration in PIM-based switch obtained using computer simulation.

are matched .Naturally, this is accomplished on the expense of adding some complexity to the switch circuitry. As the grant passing process is not iterative, it takes considerally less time that required for achieving the original iterations. The remaining of the paper is dedicated for explaining the circuit designed to pass grants from multi-granted inputs to ungranted ones at a high speed.

Fig. 5 shows a schematic diagram for the circuit that automatically passes grants from one input to another. The following signals are included in the circuit, and they are, all, set to "Low" logic level at the beginning of a time slot:

1. *Request* (*i*, *j*): A storage cell (latch) that is set to "High" once the input(*i*) sends a request to the output (*j*) and is automatically cleared at the end of the time slot.
2. *Grant* (*i*, *j*): A storage cell (latch) that is set to "High" once the output(*j*) sends a grant to the input(*i*) and is automatically cleared at the end of the time slot.
3. *MultiGrant* (*i*): If the input (*i*) has received more than one grant, this signal is set to "High". This signal can be implement as :

$$MultiGrant\,(i) = F\,(i,\,0)\prod_{k=1}^{N} \quad F\,(i,\,k)$$

Where

$$F(i,\,0) = \sum_{j=1}^{N} Grant(i,j)$$

$$F(i,\,k) = \overline{Grant(i,k)} + \sum_{\substack{j=1,\\ j\neq k}}^{N} Grant(i,j) \qquad ,k\neq 0$$

4. T_1, T_2, ... T_N: *N* Successive clock pulses generated by a ring counter, which is triggered by a clock signal of a periodic time of $1/N$th the remaining part of the time slot after performing the single iteration. Each of these pulses is present on a separate line. These pulses divide the time slot into *N* equal sub-intervals.
5. *And* (*i,j*): A signal resulting from a logic "And" operation of the signals Grant (*i,j*), Request (*i*+1, *j*) and MultiGrant (*i*). If this signal is "High", it means that there is an excessive grant at input (*i*) and it is to be passed to input (*i*+1). It should be noted that

since an output grants only one input, only one "And" gate in each column of fig. 5 can have its output "High" whereas the others (in the same column) are "Low".

Under the assumption of fully saturated VOQs (each VOQ has at least one cell at the beginning of each time slot), the existence of the circuit described above ensures that all the switch inputs will be granted at the end of the time slot.

In any sub-interval T_i, a grant received by a VOQ(n,m) can be passed to VOQ(n+1,m) provided that input (n) is multi-granted and that input (n+1) has already sent a request to output(m). However, if input (n+1) is multi-granted, or if it becomes multigranted due to being newly passed a grant from VOQ(n,m), the output of And (n+1,m) becomes "High", which enables passing this grant from VOQ(n+1,m) to V(n+2,m), and so on. In this way the operation of passing excessive grants between successive inputs continuous until it reaches one of the ungranted inputs at which it stops. The excessive grants may be one or more. The maximum number of excessive grants is N-1.Using the above procedure, all the excessive grants are passed one in each sub-interval.

In the following, the operation of such a circuit is further explained by taking a switch of size 4×4 as an example. A ring counter is used to generate four successive pulses (each of duration of one quarter the time slot) to divide the time slot into four equal subintervals (T_1, T_2, T_3, T_4). Let us consider the occurrence of the following sequence of events:

1. At the beginning of the time slot each input has requests to all the outputs.
2. After the granting phase, let the inputs be granted as follows:

- The first input has grants from the third and the fourth outputs.
- The second input has a grant from the first output.
- The third input is not granted.
- The fourth input has a grant from the second output.

Thus the first input is multi-granted whereas the third input is ungranted and, hence, it is required to pass a grant from the first input to the third one so as to get all the inputs granted. The circuit shown in fig. 5 performs this task as follows:

1. At the first pulse of the ring counter, none of the gates And (1,1), And (4,2), And (3,3), And (2,4) has its output "High", hence, no action is taken during this subinterval.
2. At the second pulse of the ring counter, none of the gates And (2,1), And (1,2), And (4,3), And (3,4) has its output "High", hence, no action is taken during this subinterval.
3. At the third pulse of the ring counter, the output of And (1,3) gate is "High" and, hence, the flip-flop Grant (1,3) is cleared, the signal MultiGrant (1) becomes "Low", the signal Grant (2,3) becomes "high" set, the signal MultiGrant (2) becomes "High", the output of And (1,3) becomes "Low" and the output of And (2,3) becomes "High". This means that the one of two grants of the first input is passed to the second input, which becomes a multi-granted input.
4. At the fourth pulse of the ring counter, the output of And (2,3) gate is "High" and, hence, the signal Grant (2,3) becomes "low", the signal MultiGrant (2) becomes "Low", the signal Grant (3,3) becomes "high" and the output of And (2,3) becomes "Low".

This means that one of the two grants received by the second input has been passed to the third input and, hence, all the inputs are now granted.

Table 1
The possible passes of grants between inputs during the subintervals specified by the ring counter

Sub-interval	Passing grants sent by output(1)		Passing grants sent by output(2)		Passing grants sent by output(3)		Passing grants sent by output(4)	
	From VOQ	To VOQ	From VOQ	To VOQ	From VOQ	To VOQ	From VOQ	To VOQ
T1	(1,1)	(2,1)	(4,2)	(1,2)	(3,3)	(4,3)	(2,4)	(3,4)
T2	(2,1)	(3,1)	(1,2)	(2,2)	(4,3)	(1,3)	(3,4)	(4,4)
T3	(3,1)	(4,1)	(2,2)	(3,2)	(1,3)	(2,3)	(4,4)	(1,4)
T4	(4,1)	(1,1)	(3,2)	(4,2)	(2,3)	(3,3)	(1,4)	(2,4)

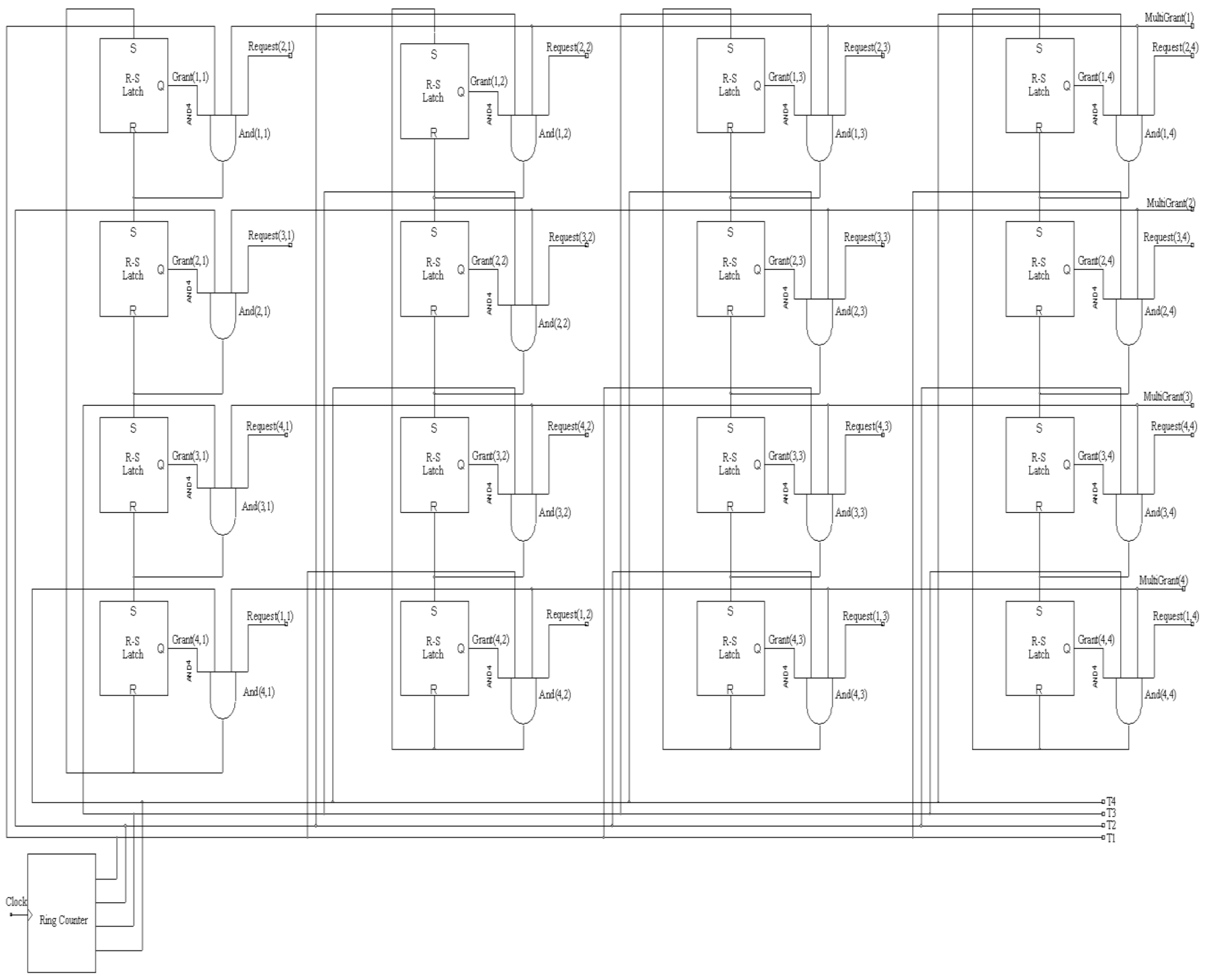
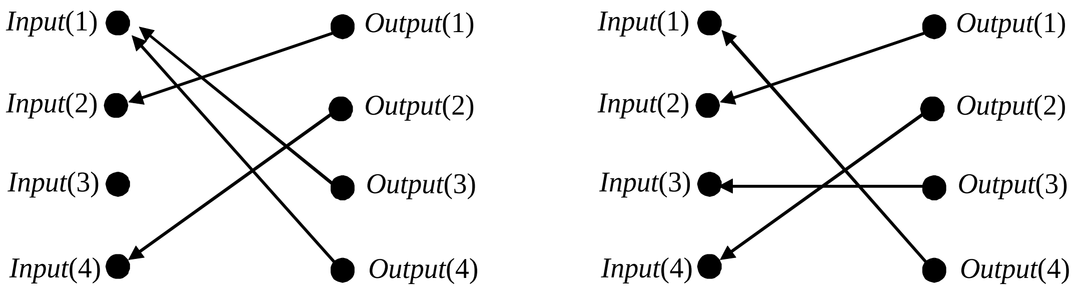


Fig. 5. The implementation of the logic required to automatically pass excessive grants from multi-granted to ungranted inputs after the matching operation for a switch of size 4x4.



(a) At the beginning of the time slot

(b) At the end of the time slot

Fig. 6. The granted inputs.

6. Conclusions

This paper proposes a new scheme to achieve 100% throughput of a VOQ- based ATM switch without doing iterations except the first one. The proposed scheme depends on a parallel matching algorithm that runs for only one iteration and then passes the excessive grants from inputs receiving multiple grants (multi-granted inputs) to the requesting inputs that have not received any grant (ungranted inputs). Thus all the inputs are matched within only one iteration.. Naturally, this is accomplished on the expense of adding some complexity at the hardware. The computer model for simulation has been developed for the traffic, the switch and the proposed schedule. The performance of the proposed scheme has been has been tested through complete simulation and found to give 100% throughput.

References

[1] H. Chao, C. Lam and E. Oki, Broadband Packet Switching Technologies: A Practical Guide to ATM Switches and IP Routers, John Wiley and Sons (2001).

[2] I. Makhamreh, “Throughput Analysis of Input-Buffered ATM Switch”, IEE Proc.-Commun., Vol. 145 (1) (1998).

[3] G. Nong, J. Muppala and M. Hamdi, “Analysis of Nonblocking ATM Switches with Multiple Input Queues”, IEEE/ACM Trans. Network., Vol. 7 (1), pp. 60-74, (1999).

[4] G. Nong and M. Hamdi, “Burst-Based Scheduling Algorithms for Non-Blocking ATM Switches”, IEEE Commun. Lett., Vol. 4 (6), pp. 202-204 (2000).

[5] G. Nong, M. Hamdi and J. Muppala, “Performance Evaluation of Multiple Input-Queued ATM Switches with PIM Scheduling Under Bursty Traffic”, IEEE Commun. Lett., Vol. 4 (6), pp. 202-204, (2001).

Received June 6, 2005
Accepted December 31, 2005