Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 50 (2006) 2271–2285
Computer Networks
www.elsevier.com/locate/comnet

High-performance switching based on buffered crossbar fabrics

Lotfi Mhamdi [a,*], Mounir Hamdi [b], Christopher Kachris [a], Stephan Wong [a], Stamatis Vassiliadis [a]

[a] *Computer Engineering Laboratory, Delft University of Technology, The Netherlands*
[b] *Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Sai Kung, Kowloon, Hong Kong*

Received 28 July 2004; received in revised form 29 April 2005; accepted 1 September 2005
Available online 29 September 2005

Responsible Editor: G. Parr

Abstract

As buffer-less crossbar scheduling algorithms reach their practical limitations due to higher port numbers and data rates, internally buffered crossbar (IBC) switches have gained a lot of interest recently due to their great potential in solving the complexity and scalability issues faced by their buffer-less predecessors. The IBC switching architecture combined with the virtual output queueing (VOQ) architecture was shown, through distributed scheduling algorithms, to be able to sustain the current and expected increases in Internet throughput rates. Due to the architectural similarity between the input queued (IQ) and IBC switches, all the algorithms proposed for the latter were just a simple mapping of earlier algorithms proposed for the former. In this paper, we propose a set of scheduling schemes that are purely advocated for the VOQ/IBC switch architecture. We first address the issue of the internal buffers importance in the arbitration process. We propose a weighted scheduling algorithm, named *Critical internal Buffer First* (CBF), which takes full advantage of the internal buffer elements and makes its decision exclusively on the internal buffer information. Second, in order to simplify the scheduling scheme and make it practical, we propose a class of scheduling algorithms, named *Current Arrival First–Priority Removal* (CAF–PRMV) that use priority levels instead of weights. We argue that the interaction, through the internal buffer element, between the input and output schedulers is very important in designing such practical and highly scalable schemes for the IBC switching architecture. Our hardware implementation, in reconfigurable logic, shows that our CAF–PRMV class of algorithms can sustain a 10 Gbps line speed for a 32×32 VOQ/IBC switch.

Keywords: Buffered crossbar fabric; Matched scheduling; Priority

1. Introduction

Traditionally, the difference in switches' design was in the way the queuing function is implemented. There has been extensive research work in this area and many switching architectures have been

* Corresponding author. Tel.: +31 15 27 89 656.
E-mail addresses: lotfi@ce.et.tudelft.nl (L. Mhamdi), hamdi@cs.ust.hk (M. Hamdi).

1389-1286/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2005.09.001

proposed, such as output-queued (OQ) and input-queued (IQ) switches. It is well known that OQ switches are highly desirable for their optimal performance and QoS guarantees [1,2]. However, the high internal speedup coupled with the limitation in memory access time prohibits the OQ to scale to even a medium sized switch. On the other hand, IQ switches have gained more interest because of their low cost and high scalability [3]. While the head-of-line (HoL) blocking problem limits the achievable throughput of an IQ switch to approximately 58.6% [4], the well-known VOQ architecture [5,3,6] was proposed instead and has improved the switching performance of FIFO-based IQ by several orders of magnitude, hence making IQ switches even more attractive.

Despite its popularity, the VOQ solution has its own problems [7,3]. A centralized scheduler is required to configure the switch so that, at each time slot, each input sends one cell to at most one output and each output receives one cell from at most one input. In doing so, the scheduler needs to resolve two potential blockings: input blocking and output blocking. These blockings make the task of the scheduler complicated and the packets delay unpredictable. An extensive amount of research work has been done and a plethora of scheduling algorithms has been proposed [5,8,9,3,10–12]. Unfortunately, for high-bandwidth VOQ switches, all these algorithms are either too complex to run at high speed [3] or fail to perform well under practical traffic patterns such as non-uniform. This is mainly attributed to the centralized design of these schedulers and to the nature of the crossbar-based architecture. Therefore, only few of these algorithms were demonstrated to be practical and shown to achieve high throughput under some specific traffic patterns [5,3].

Internally buffered crossbar (IBC) switches have been considered as a viable alternative to buffer-less crossbar switches to improve the switching performance. The presence of internal buffers improves drastically the overall performance of the switch due to the advantages it offers. First, the adoption of internal buffers makes the scheduling totally distributed, hence reducing dramatically the arbitration complexity and making it linear. Second, and most importantly, these internal buffers reduce (or avoid) the output contention. Meaning, they allow the inputs to send cells to an output irrespective of simultaneous cells transfer to the same output. While there have been many architectures for the (IBC) switching architecture [13–15], our focus in this article is on IBC with input VOQs. A buffered crossbar switch combined with input VOQ was first introduced in [16] as depicted in Fig. 1. In the remainder of this article, we refer to this architecture as VOQ/IBC architecture.

In the VOQ/IBC architecture, a scheduling cycle consists of three independent phases: input scheduling, output scheduling and delivery notification. Recently, there have been many scheduling algorithms proposed. These algorithms have been compared to just the conventional buffer-less crossbar-based switches. As expected, they have been shown to exhibit much better performance. However, there is still a lot of room for coming up with better algorithms (both in terms of performance and hardware complexity) for the VOQ/IBC architecture because the proposed algorithms are just simple mapping of earlier algorithms proposed for the buffer-less crossbar switches into the new VOQ/IBC architecture. Moreover, none of the schemes presented in the literature has addressed the issue of the interaction between the internal buffers and the input line cards. In fact, as will be illustrated in this paper, by carefully considering the interaction between the VOQ and the internal queues of the VOQ/IBC architecture, we can design matched pairs of input/output arbitration algorithms that outperform the straightforward algorithms even with less hardware complexity.

In this paper, we propose a set of scheduling algorithms that take full advantage of the VOQ/IBC switching architecture. The CBF scheduling scheme is proposed first. It is based on the *Youngest internal Buffer First* (YBF) at the input side with a

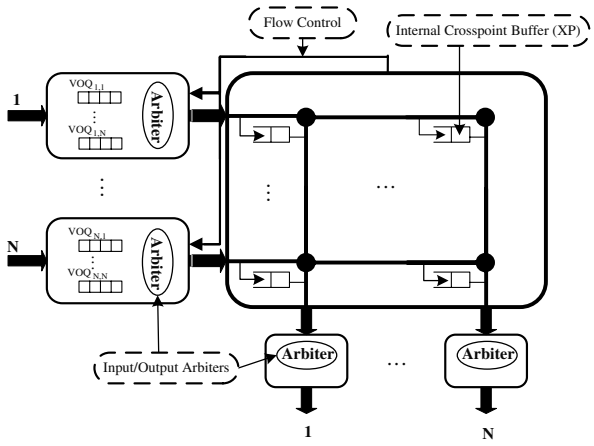

Fig. 1. The VOQ/IBC architecture.

scheme based on the *Oldest internal Buffer First* (OBF) at the output side. We show that the information on the internal buffers is sufficient for the schedulers to make effective decisions. In an attempt to further reduce the scheduling complexity and make it more practical, we subsequently propose a set of practical scheduling algorithms. They are named *Current Arrival First–Priority Removal* (CAF–PRMV). These algorithms are, in fact, an approximation of LIFO (Last-In-First-Out). Their main advantage lies in their stateless information exchange (queue occupancies, HoL waiting time, etc.) making them simple to implement. As will be shown through simulation, they achieve very high throughput under both uniform and non-uniform distributed Bernoulli arrivals. They outperform the existing algorithms under all traffic patterns (uniform Bernoulli, bursty and non-uniform Bernoulli arrivals). In addition to the delay throughput performances, other relevant performance metrics were studied such as input VOQs occupancies. Our scheme exhibited much less queues occupancies among all the schemes presented. An FPGA based hardware implementation was presented and the results showed that our class of algorithms can be implemented in a 32×32 VOQ/IBC switch with each port operating at 10 Gbps (OC − 192).

The remainder of the paper is organized as follows: Section 2 presents the related work and points out the advantages and limitations of existing schemes. In Section 3, we introduce the CBF scheme along with a its performance study with comparison to the previously existing schemes. Section 4 contains a detailed presentation of our set of practical scheduling schemes, CAR_PRMV. We propose variations of the output scheme, PRMV, based on priority levels and we present an experimental study with comparison to state of the art algorithms. Section 5 gives a possible hardware implementation of our scheme. Finally, Section 6 concludes the paper.

2. Related work

The internally buffered crossbar switching architecture has attracted a lot of attention due to the potential it offers in solving the scheduling complexity faced by the IQ switching architecture. As a result, an important amount of research work has been advocated to this architecture. These research results can be classified into two main topics. The first one is related to proving OQ emulation by a VOQ/IBC switch while the second topic was to propose a wide range of scheduling schemes for the VOQ/IBC. Our current paper falls into the latter topic, that is proposing scheduling algorithms for the VOQ/IBC running at a speed up of one.

Very interesting results have been proposed showing that a VOQ/IBC, running twice as fast as the line rate, can emulate a FIFO OQ switch [17]. Other results [18] proved that a VOQ/IBC with a speed up of two can emulate a broad class of algorithms (i.e., FCFS, Strict Priority, Early Deadline First) operating an OQ switch. A more recent and extended result [19] showed that a VOQ/IBC switch with two times speed up can provide 100% throughput, rate and delay guarantees.

Alongside the work on OQ emulation, there have been many scheduling schemes proposed for the VOQ/IBC architecture. These algorithms can be classified into weight-based schemes [16,20] and Round Robin (RR) based schemes [21,22]. In [16], a scheme based on the Oldest Cell First (OCF) at the input as well as the output scheduling was proposed. While this scheme achieves high throughput under uniform Bernoulli arrivals, the same benefits were not achieved for the non-uniform case. A scheme based on the Longest Queue First (LQF) in the input and a round robin (RR) arbitration at the output was presented in [20] and was proven, through a fluid model, to be stable under uniform input traffic. In addition to its rather complex input scheduling (LQF), this scheme may lead to permanent queue starvation. A set of round robin schemes was proposed [21,22] and they were shown, through simulation, to achieve very good performance under uniform arrivals. These schemes are desirable because of their simplicity in hardware and fairness, however they experience the same problem as in [16] and perform poorly under non-uniform traffic patterns. As mentioned earlier, none of the schemes proposed for the VOQ/IBC has truly taken full advantage of the features the architecture offers. In this paper, we fill this gap and propose scheduling algorithms that are purely dedicated to the VOQ/IBC switching architecture.

3. The critical internal buffer first algorithm (CBF)

The proposed CBF scheme is equivalent to our previously proposed scheme, named the *Most Critical internal Buffer First* (MCBF) [23]. Before presenting the CBF algorithm, we first introduce, for ease of explanation, some notations that will be used in the remainder of this article.

3.1. Notations

We consider the switch model depicted in Fig. 1. Fixed size packets, or cells, are considered. Upon the arrival of variable length packets to the switch, they are segmented into cells for internal processing and re-assembled before leaving the switch. A processing cycle has fixed duration, called time slot (time cell). There are N input line cards; each one maintains N logically separated VOQs. When a packet (cell), destined to output j, $1 \leqslant j \leqslant N$, arrives to the input card i, $1 \leqslant i \leqslant N$, it is held in $\mathrm{VOQ}_{i,j}$. The following notations will be used throughout the paper:

- The internal fabric consists of N^2 buffered crosspoints (XP). A crosspoint $\mathrm{XP}_{i,j}$, holds cells coming from input i and going to output j. If the cell has been transferred from the input $\mathrm{VOQ}_{i,j}$ to the internal buffer $\mathrm{XP}_{i,j}$ at time $t_{\mathrm{arrival\_to\_fabric}}$, its age is noted by: $\mathrm{XPA}_{i,j} = t_{\mathrm{now}} - t_{\mathrm{arrival\_to\_fabric}}$.
- Eligible input VOQ (EVOQ): A $\mathrm{VOQ}_{i,j}$, is said to be eligible (noted $\mathrm{EVOQ}_{i,j}$) for being scheduled in the input scheduling process if it is not empty and the internal buffer $\mathrm{XP}_{i,j}$ is empty (or not ful l).
- The line of crosspoint buffers LXPB_i is the set of the internal buffers $\mathrm{XP}_{i,j}$ that correspond to the same input, i, and holding cells for all outputs. The age of all the HoL cells queued in LXPB_i is equal to the internal buffer line age and defined by: $\mathrm{BLA}_i = \sum_j(\mathrm{XPA}_{i,j})$.
- The column crosspoint buffer CXPB_j is the set of the internal buffers $\mathrm{XP}_{i,j}$ that correspond to the same output, j, and receiving cells from all inputs. NCB_j is the number of cells queued in CXPB_j. The age of all the HoL cells queued in CXPB_j is equal to the internal buffer column age and defined by: $\mathrm{BCA}_j = \sum_i(\mathrm{XPA}_{i,j})$.

3.2. Motivation

It is of note that the scheduling process is, natively, divided into two phases: input scheduling and output scheduling. Of major importance is the matching of the input/output scheduling in pairs so that they complement each other. Therefore, the choice of which output scheduling is best suited to which input scheduling is very important. To this end, the internal buffer element is of key importance in finding matched scheduling because of its shared nature. No output is idle as long as $\mathrm{NCB}_j \geqslant 1$; $1 \leqslant j \leqslant N$. To keep the outputs as busy as possible, a good strategy is to maintain a load balancing among the internal buffers. This is the intuition behind the proposed algorithm that serves the most critical buffer first. It favors the most recently served internal buffer in the input side while the output gives priority to the least recently served internal buffer. Consequently, we have ensured that the two scheduling phases are matched and the second complements the job of the first. More interestingly, we do not need to maintain any information during both scheduling except the information about the internal buffers. In fact, the CBF scheme is a per cell waiting time scheduling scheme that is equivalent to our previously proposed per internal buffer occupancy scheduling scheme, MCBF [23]. The following section describes our proposed scheme.

3.3. CBF specification

The CBF scheduling scheme is based on the Youngest internal Buffer First (YBF) policy at the input side, whereas, the output arbitration is based on the Oldest internal Buffer First (OBF). Throughout the remainder of the paper, we will be using both CBF and YBF–OBF acronyms interchangeably to refer to the same algorithm. Using CBF will refer to the whole scheme (input and output side) while using YBF (respectively OBF) will refer to the input scheduling (respectively output scheduling). The CBF scheme, especially its output scheduling OBF, is equivalent to the buffer-less Oldest Port First (OPF) Algorithm [24]. The output scheduling, OBF, performs its arbitration based on the input waiting time function used by OPF [24], defined as the waiting time sum of all HoL cells at the input and waiting for the same output. Our input scheduling YBF, however, uses an opposite arbitration criterion to the output waiting time function used by OPF [24]. Instead of giving priority to the request with the greatest output waiting function as in OPF, YBF gives priority to the request with the smallest output waiting time instead. In order to achieve fairness, both YBF and OBF maintain each a highest priority pointer to break ties in the presence of conflicts. Each input (respectively output) pointer is initialized to the index of the input (respectively output) it belongs to. The specification of both scheduling algorithms is as follows:

- Input scheduling (YBF)
  - For each input i: Starting from the highest priority pointer's location, select the first EVOQ

corresponding to: $\min_j\{BCA_j\}$ and send its HoL cell to the internal buffer ($XP_{i,j}$). Move the highest priority pointer to the location $(j+1) (\mathrm{mod}\, N)$.

- Output scheduling (OBF)
  - For each output j: Starting from the highest priority pointer's location, select the first $XP_{i,j}$ corresponding to: $\max_i\{BLA_i\}$ and send its HoL cell to the output. Move the highest priority pointer to the location $(i+1) (\mathrm{mod}\ N)$.

3.4. Performance study

This section presents a delay performance study for the algorithms presented above using a 32×32 buffered crossbar switch. The same experiments' settings will be used throughout the remainder of this article. Delay is measured as the period of time a cell is kept waiting in an input/internal/output buffer before being scheduled. Each point in the resulting figures is obtained for 500,000 time slots (cell time), and the statistics are gathered from the (50,000)th time slot. Average delay is calculated from all the cells output during this period of time. Normalized load means the percentage of time slots, which have cells coming in, averaged over all inputs. The CBF algorithm is compared to the LQF–RR and OCF–OCF algorithms respectively under bursty uniform traffic and unbalanced traffic. The unbalanced traffic used in this paper is the same as the one used in [21].

A stability performance study was carried out along with the delay study. As it was presented in [25], the input queues occupancies can serve to prove the stability of the scheduling algorithm. That is, if under a service policy (scheduling algorithm) X, one can show that $E(\|L(n)\|) < \infty$[1], he can conclude that X is stable. $\|L(n)\|$ is the $l-$ two norm vector representing the occupancy of the VOQs a time n and defined as follows:

$$\|L_n\| = \Big(VOQ_{1,1}(n)^2 + \cdots + VOQ_{1,N}(n)^2 + \cdots + VOQ_{N,1}(n)^2 + \cdots + VOQ_{N,N}(n)^2\Big)^{1/2}$$

In this section, we just present, through simulation, the occupancy of the VOQs under the above-mentioned scheduling algorithms. This study can be used to find a practical upper bound on the input buffer space that a scheduling algorithm needs to prevent congestion.

Fig. 2 depicts the delay performance of each of the algorithms with burst lengths, l, equal to 10, 50 and 100 respectively. YBF–OBF shows the best performance amongst all the other algorithms. At 99% load and burst length of 10, YBF–OBF has an average queuing delay less than 80% that of LQF–RR. The same advantage is seen with burst length equal to 50 and 100 respectively. YBF–OBF has the best resistance to the burstiness effect. As the burst length increases, the growth of the average delay of YBF–OBF is slower than the rest of the other simulated algorithms. With a burst length of 50% and at 99% load YBF–OBF has an average delay of 2524, then LQF–RR with an average delay of 3014 and finally OCF–OCF with an average delay of 3311.

As for the stability performance, illustrated in Fig. 3, YBF–OBF has surprisingly the best performance amongst all despite the fact that it maintains no state information about the input VOQs when it does its arbitration. YBF–OBF has always the smallest norm of the VOQs occupancy vector, $\|L(n)\|$, irrespective of the burst length or the traffic load. We can see that, for a burst length of 10 for example, YBF–OBF has the smallest queues' norm, about 1670. However, LQF–RR has a vector with norm equals 1937 and OCF–OCF equals 1946.

Fig. 4 depicts the delay performance under the unbalanced traffic model. As shown below, YBF–OBF no longer has the smallest delay. This result

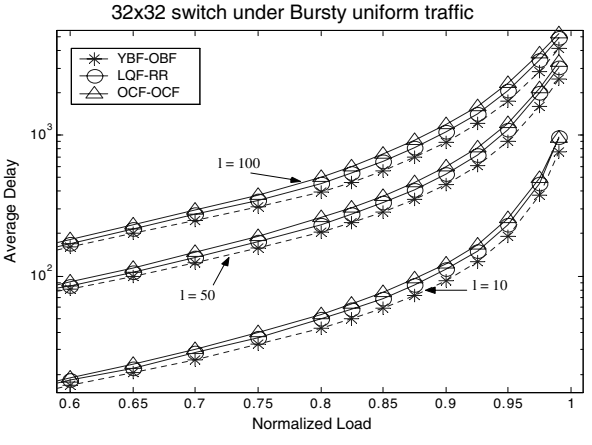

Fig. 2. Delay performance under Bursty uniform traffic, and different burst lengths.

[1] The expected value of the $l-$ two norm vector, $L(n)$, representing the input VOQs occupancies is finite.

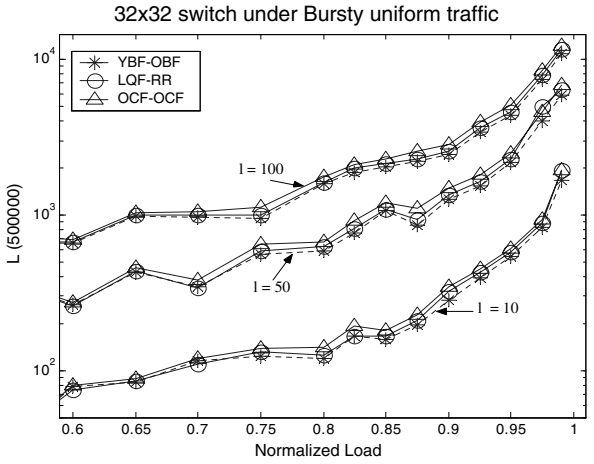

Fig. 3. Stability performance under Bursty uniform traffic, and different burst lengths.

is somehow expected because YBF–OBF is a stateless scheme that uses no state information at all about the input VOQs during the arbitration process. Moreover, since the internal buffer size is only one cell-length, this element of information seems to be not very efficient so that CBF can make effective decision.

It is to note that the YBF–OBF is expected to have much better performance if we increase the internal buffer size. The reason is, as mentioned earlier, as the internal buffer size increases, YBF–OBF will emulate more the (OPF) algorithm [24]. To see this effect, we simulated the YBF–OBF with the rest of the algorithms under non-uniform traffic (unbalanced traffic) with a 32×32 switch and an 8-internally buffered crossbar fabric (internal buffer size $= 8$ cells).

As illustrated in Fig. 5, the delay performance of YBF–OBF is the best as expected because it is getting closer to the buffer-less OPF as the internal buffer size increases. The average queuing delay of CBF has improved 374 times. However both LQF–RR and OCF–OCF have an improvement factor of less than 2. At 99% load, YBF–OBF has an average queuing delay of 34 time slots, then LQF–RR with an average delay of 57, thereafter OCF–OCF with a delay equals to 71.

As shown above, we can see that the CBF scheme is better than the state-of-the art scheduling schemes, such as LQF–RR and OCF–OCF. It takes full advantage of the internal buffers elements when making its arbitration. One drawback with the CBF scheme appears to be its hardware implementation. If we consider the input schedulers to take place at the ingress part for example, this means that high information exchange between the line cards and the fabric will be dedicated only for the sake of scheduling and therefore result in wasted internal bandwidth. While, the hardware implementation problem seems to be overcome by careful design concerning especially, the input arbiters, the CBF scheme is a weighted scheme and the time stamping mechanism performed by CBF might be considered as too costly.

Recall that the VOQ/IBC has key advantages that can serve to ensure that the scheduling algorithm can be simple and efficient at the same time. Our goal is to design scheduling schemes with the least hardware requirement while maintaining good

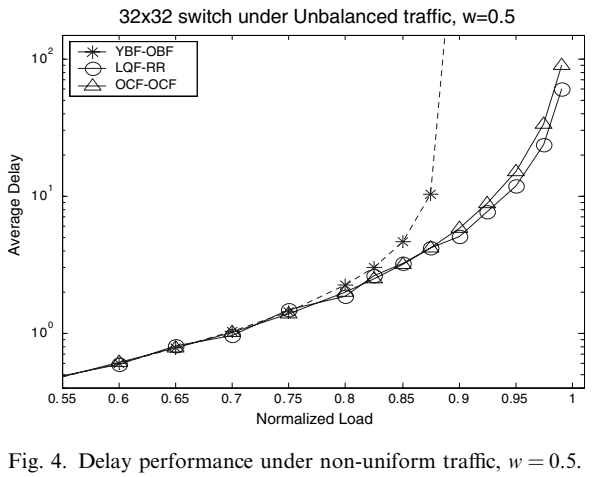

Fig. 4. Delay performance under non-uniform traffic, $w = 0.5$.

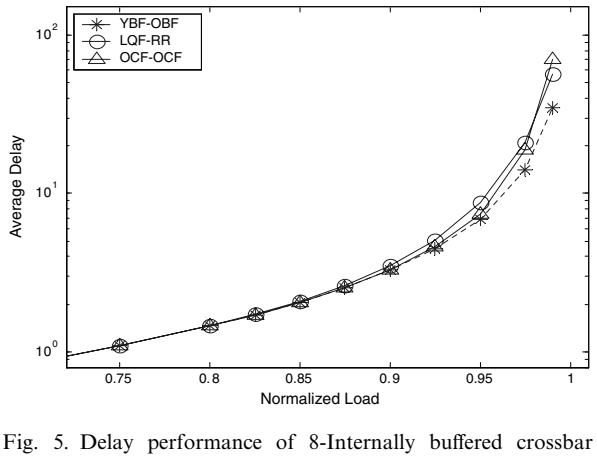

Fig. 5. Delay performance of 8-Internally buffered crossbar switch under Bernoulli non-uniform traffic, $w = 0.5$.

performance. To this end, in the following section, we propose the CAF_PRMV group of new scheduling algorithms. They meet both our simple hardware requirement and satisfy our goal to achieve good performance.

4. The current arrival first–priority removal algorithms

In this section, we propose our group of practical algorithms: *Current Arrival First–Priority Removal* (CAF_PRMV). The input scheduling gives priority to the newly arriving packets while the output scheduling completes this task by serving the recently arrived packets to the internal buffer. The intuition behind this is to overcome the lack of performance under the non-uniform traffic without using any weight functions or state information. The existing algorithms either perform poorly under non-uniform traffic or require sorting. The idea of serving the newly arriving cells favors the input that has more often cells coming in and did not punish the uniformly arriving cells, hence tackle the non-uniform traffic while being stateless. From above, we knew that an input (respectively output) scheduling scheme could not perform well so long as it is not matched with the appropriate output (respectively input) scheduling scheme. To this end, CAF_PRMV was designed to be a matched pair of input/output scheduling. That is the output scheduling, PRMV, is complementary to the input scheduling, CAF. To better understand this, some different schemes are presented and will be analyzed. The input scheduling, CAF, will remain the same, the changes are done at the output side.

4.1. CAF specification

At each time slot, the Current Arrival First (CAF) algorithm checks whether there is a new cell arriving at the input port. To accomplish this task in the output scheduling, CAF assigns a priority level to each cell leaving the input port. This level will decide the priority (urgency) of that cell in the output scheduling phase. The use of priority levels is an efficient choice. First, the output-scheduling phase will be much simpler and faster than sorting for example. Second and most importantly, the adoption of priority levels makes the implementation easy and the hardware requirement simple. The specification of the input scheduling CAF is as follows:

For each input i:

- If there is a currently arriving packet, P, to an eligible $VOQ_{i,j}$.
- Then send its HoL packet to $XP_{i,j}$ with two priority bits[2] as follows:
  - If $VOQ_{i,j}$ contains other packet(s) than P.
    - Then set the two priority bits to 11.
    - Else set the two bits to 10.
- Else based on the highest priority pointer location, serve the next eligible $VOQ_{i,j}$ corresponding to $\min_j\{NCB_j\}$. The highest priority pointer is incremented (mod N) to one location beyond the selected input $VOQ_{i,j}$.
  - If $VOQ_{i,j}$ contains other packet(s) than P.
    - Then set the two priority bits to 01.
    - Else set these bits to 00.

We can see that the two priority bits create four priority levels for a packet. Thus, when a packet, P, leaves the input card, it has along with it its priority level for being scheduled in the output scheduling phase. According to these priority levels, a packet could be new with a non-empty corresponding input $VOQ_{i,j}$ (priority 11), or it could be not new (old) with non-empty corresponding input $VOQ_{i,j}$ (priority 01), or it could be new but with empty corresponding input $VOQ_{i,j}$ (priority 10), or it could be not new with empty corresponding input $VOQ_{i,j}$ (priority 00). These priority levels are summarized in the following table.

From Table 1, many combinations could be envisioned ($4! = 24$). However, to perfectly accomplish the task of the input scheduling, CAF, many combinations should be eliminated. For example, priority level P4 can only be the lowest level. This is because any output scheduling which gives priority to an old internally buffered cell with empty corresponding input VOQ cannot help CAF doing its job. Since the input VOQ is empty, there should be no rush in getting out that cell (queue stable and not congested). It is more urgent to send out any other packet with non-empty input VOQ. On the other hand, priority level number one (P1) can only be the highest priority among the four priority levels. This is because any output scheduling scheme which favors any priority level to the first one, leads to a decrease in the performance of CAF, and therefore

[2] The first bit means that the packet, P, is new, while the second bit indicates the occupancy of the $VOQ_{i,j}$ holding P.

Table 1
Priority level of a packet

Priority level	New packet	Non-empty VOQ
P1	1	1
P2	0	1
P3	1	0
P4	0	0

will not be appropriate. An internally queued packet that comes from a busy input VOQ (having currently new packet coming in) should be sent out urgently. This avoids the VOQ from being congested or unstable. Doing so, there are only $(2! = 2)$ different combinations left (depending on P2 and P3 ranking of Table 1) and are as follows:

- $C1 = (\mathrm{P1}, \mathrm{P2}, \mathrm{P3}, \mathrm{P4})$: using this combination means that packets are served based on the priority order[3] P1, P2, P3, and P4.
- $C2 = (\mathrm{P1}, \mathrm{P3}, \mathrm{P2}, \mathrm{P4})$: using this combination means that packets are served based on the priority order P1, P3, P2, and P4.

The output scheduling PRMV will then be consisting of two different schemes depending on the combination used. These two schemes are called PRMV1 and PRMV2, respectively.

4.2. Specification of PRMV1

The specification of the output scheduling PRMV1 is as follows:

For each output j:

- Starting from the highest priority pointer's location, select the next non-empty internal buffer $\mathrm{XP}_{i,j}$ giving preference based on $C1$. The highest priority pointer is incremented $(\mathrm{mod}\, N)$ to one location beyond the selected internal buffer $(\mathrm{XP}_{i,j})$.

4.3. Specification of PRMV2

The specification of the output scheduling PRMV2 is as follows:

For each output j:

- Starting from the highest priority pointer's location, select the next non-empty internal buffer $\mathrm{XP}_{i,j}$ giving preference based on $C2$. The highest priority pointer is incremented $(\mathrm{mod}\, N)$ to one location beyond the selected internal buffer $(\mathrm{XP}_{i,j})$.

4.4. PRMV variations

In this section, two other versions of PRMV are investigated. Recall that the input scheduling and the output scheduling are performed independently. As shown before, when a packet, P, is scheduled at the input, it takes along with it two priority bits which will decide its priority for being scheduled in the output scheduling phase. However, the priority bit relative to the state of $\mathrm{VOQ}_{i,j}$ that used to hold P might not be accurate. To see this, let us consider the following example: suppose that the current time slot is T_{now}. Suppose a packet P has entered the switch at time T_{past} and was scheduled in the input scheduling during the same time slot, T_{past}—since CAF favors newly arriving cells—and its $\mathrm{VOQ}_{i,j}$ was empty at time T_{past}. Therefore, its two priority bits are set to 10 (1: new packet and 0: empty VOQ). Thus, during each time slot T, $T_{\mathrm{past}} \leqslant T \leqslant T_{\mathrm{now}}$, so long as P is still in the internal buffer, $\mathrm{XP}_{i,j}$, it is considered as having an empty corresponding input $\mathrm{VOQ}_{i,j}$. However, $\mathrm{VOQ}_{i,j}$ might receive new cells during the time interval $[T_{\mathrm{past}} + 1, T_{\mathrm{now}}]$. With this situation happening, P is treated unfairly among other packets since its priority levels do not match the reality. To avoid this problem, an alternative to PRMV was proposed. The idea is that the bit that records the state of a $\mathrm{VOQ}_{i,j}$ is set at the moment of the output scheduling and not during the input scheduling. Proceeding this way solves the inaccuracy problem. One way to do this is by checking the corresponding input $\mathrm{VOQ}_{i,j}$ of every internally buffered packet that is considered for an output scheduling. Therefore, we have the following two new versions of PRMV, called Pr_Check1, Pr_Check2 one for each priority scheme.

4.4.1. Specification of Pr_Check1

The specification of the output scheduling Pr_Check1 is as follows:

For each output j:

- At each time slot, T, Do:
- Starting from the highest priority pointer's location, select the next non-empty internal buffer

[3] The highest priority level is inversely proportional to the priority level index that is, P1 is the highest and P4 is the lowest.

$XP_{i,j}$ giving preference based on $C1$. The highest priority pointer is incremented (mod N) to one location beyond the selected internal buffer ($XP_{i,j}$).

4.4.2. Specification of Pr_Check2

The specification of the output scheduling Pr_Check2 is as follows:

For each output j:

- At each time slot, T, Do:
- Starting from the highest priority pointer's location, select the next non-empty internal buffer $XP_{i,j}$ giving preference based on $C2$. The highest priority pointer is incremented (mod N) to one location beyond the selected internal buffer ($XP_{i,j}$).

While Pr_Check seems to be better than PRMV, this solution is not a practical solution. The reason is that the output scheduler needs to perform a checking cycle during each time slot. This is undesirable due to the amount of information exchanged. Moreover, Pr_Check performs slightly better than PRMV and the difference is only seen under light load. However, under heavy load, their performances are very close, due to steady state of the VOQs. This means that under heavy load, almost all the VOQs have more than one packet during each input scheduling cycle and therefore the unfairness problem is almost self-removed.

4.5. Performance study

The simulation results contained in this section are obtained based on the same settings as in Section 3.4. Fig. 6 shows the performance evaluation the RR–RR, LQF–RR, and OCF–OCF along with our group of proposed algorithms. All our proposed algorithms have shorter queuing delay than all existing schemes. CAF–Pr_Check1 has the best performance among all, with very small difference when compared to CAF_Prmv1.

As for the performance under bursty traffic, Fig. 7, our group of proposed algorithms has the best performance among all the algorithms tested. The largest delay among our algorithms was always under 800. However, the delay for LQF–RR and OCF–OCF is more than 900 each, and RR–RR is 888.

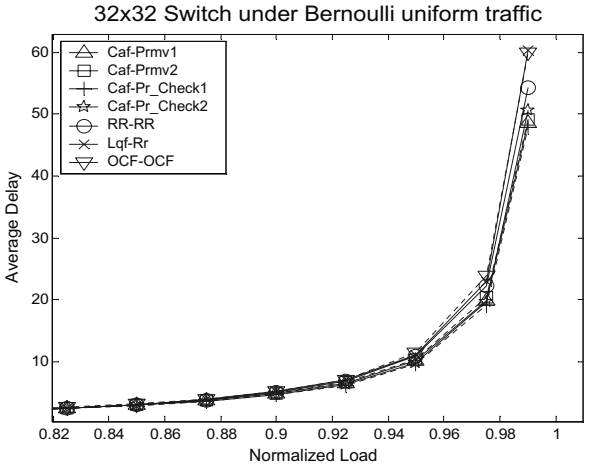

Fig. 6. Average delay under Bernoulli I.I.D. uniform traffic.

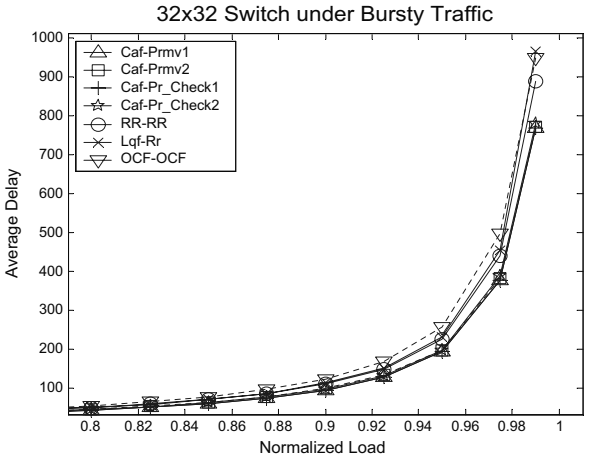

Fig. 7. Delay performance under Bursty uniform traffic.

Fig. 8 shows the stability performance of the input VOQs under bursty traffic. CAF–Pr_Check2 has the worst performance among all. The reason of this low performance is due to the highest priority given to the new arriving packet irrespective of the state of its VOQ. However, in the event of no arrivals, CAF serves packets based on the minimum occupied internal buffer.

In Fig. 9, the unbalanced coefficient, w, is fixed to 0.5. The output scheduling algorithms PRMV2 and Pr_Chech2 perform less than the others because they give priority to the newly coming packet irrespective of whether its VOQ is empty or not, and this is not appropriate to the input scheduling CAF. This is because, in many cases, CAF serves

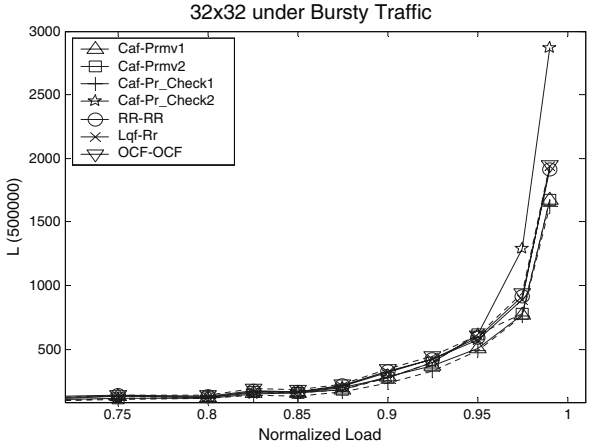

Fig. 8. VOQs occupancies under Bursty uniform traffic.

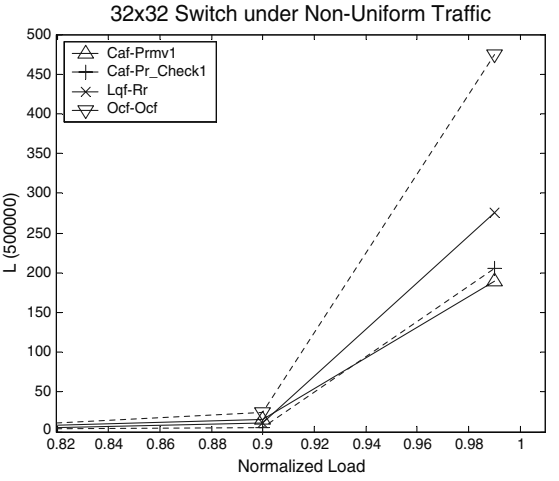

Fig. 10. VOQs occupancies under unbalanced traffic ($w = 0.5$).

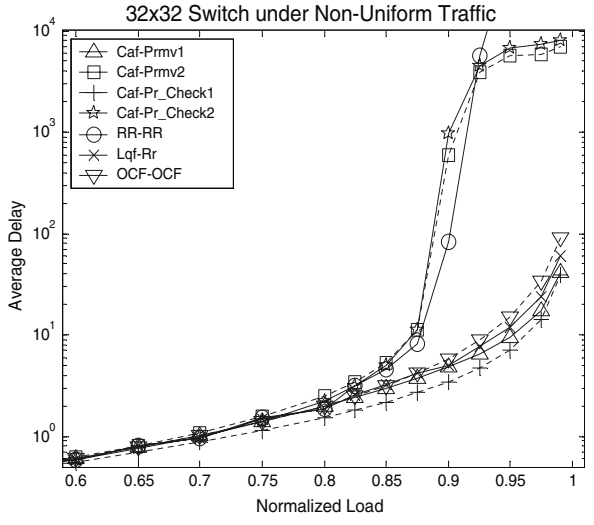

Fig. 9. Delay performance under unbalanced traffic ($w = 0.5$).

packets based on the minimum occupied internal buffer and not on the newly coming packet. Among all, CAF–Pr_Check1 has the best performance, thereafter CAF–PRMV1 and then LQF–RR and OCF–OCF.

As for the VOQs occupancies, we omitted the results of CAF_Prmv2, CAF_Check2, and RR–RR as they are having infinite queues occupancies respectively. However, as depicted in Fig. 10, CAF–PRMV1 and CAF–Pr_Check1 have the minimum queues occupancies among all the algorithms. Under a load less than 80%, all the schemes have the same performance. CAF_Check1 maintains the smallest queues occupancies under light load (less than 90%), but as the load gets higher, CAF_PRmv1, improves better than CAF_Check1. This is because, as the load increases, the inaccuracy problem, caused by the bit that records the VOQs status, is self resolved. The reason is because, under high load every VOQ will have more than one cell and therefore it will not make any difference whether or not a VOQs status checking phase is performed.

To summarize, we note the following points. First, the information about the arrival and the state of the input VOQs is shown to be even more effective than using weighted requests as with LQF–RR or OCF–OCF. Second, while CAF_Prmv1 and CAF_Check1 demonstrated the best overall performances among all the schemes presented, we argue that CAF_Prmv1 is a better choice than CAF_Check1. As mentioned earlier, the checking operation that CAF_Check1 performs each scheduling phase was shown to be of no significant improvement given the high internal bandwidth it consumes. In fact, CAF_Prmv1 maintained a steady good performance. This is in turn because of its efficient input scheduling decision and its good output priority order of scheduling. The input scheduling, CAF, bases its decision on cells arrival, VOQs states and the internal columns of crosspoint buffers occupation. In the event of no arrivals to an eligible input VOQ, CAF serves packets based on the minimum occupied column of internal buffers.

5. Hardware implementation

This section describes the hardware implementation of the CAF_PRMV class of algorithms for a 32×32 buffered crossbar switch. The design was implemented in reconfigurable logic that is a low-cost solution for rapid prototyping. The target device was the Xilinx Virtex II Pro XC2VP20 and we used the Xilinx ISE Platform 6.3 design flow. The input and output arbiters are depicted in Figs. 11 and 12 respectively. Both input and output arbiter use a programmable protocol encoder (PPE). The PPE module is used to select a queue from a number of eligible queues, based on an index pointer. Our current design is based on the PPE proposed by [26].

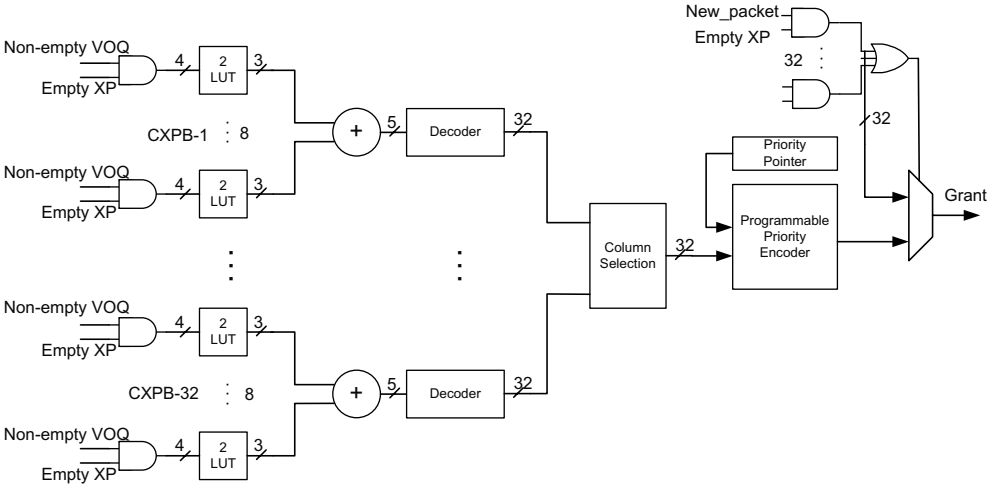

Fig. 11. Input arbiter.

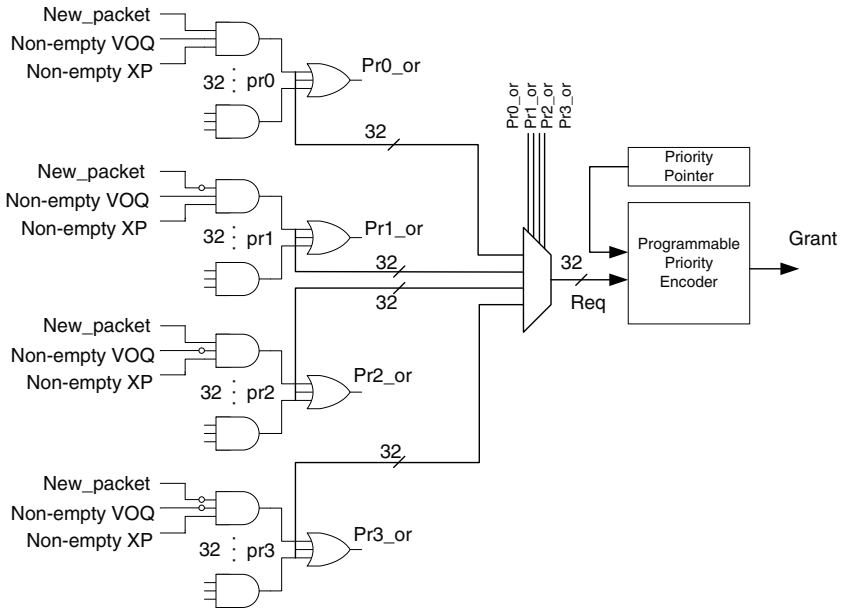

Fig. 12. Output arbiter.

The output arbiter is depicted in Fig. 12. Four different 32-bit vectors are created based on the priority encoding. The elements of each vector are or-ed in order to find out if there is at least one vector with highest priority (depending on PRMV1 or PRMV2 scheduling policy). This 4-bit vector is used in the priority multiplexer that selects which vector will proceed to the programmable priority encoder. The figure shows that the basic PPE can easily be extended without much overhead in order to be consistent with the proposed algorithm.

The input arbiter, Fig. 11, is more complex than the output arbiter. When a new packet has just arrived to a VOQ, then this queue is selected to be executed. Otherwise, the arbiter must select the eligible $VOQ_{i,j}$ corresponding to the minimum occupied internal buffer column $CXPB_j$. In addition, in case there are two columns with the same occupancy, the queue is selected based on the index pointer. The minimum function is relatively expensive in terms of delay. To map the minimum function and the index scheme efficiently in reconfigurable logic in only one cycle, the following scheme is used. We select 4 bits from the 32-bit column vector (each bit shows whether the buffer is occupied) as input to a Look-Up-Table (LUT) with 3-bits output. The LUT encodes the number of ones in the 4-bit vector. If there is only one "1" then the output is "01", if there are two "1"s then the output is "10" and if three "1"s then the output is "11" or if four "1"s then the output is "100". Then, these 3-bit vectors are added to determine the number of "1"s in the 32-bit vector. The sum of ones is used as an input to a decoder. In case there are zero "1" then the output of the decoder is "10000". If there is only one "1", the output is "01000", etc.

The column selection is a circuit similar to the output arbiter. If there is at least one vector with "10000", this means that there is at least one column with all the buffers being empty. Hence, the first bit of each input is selected as input to the PPE. This circuit is better illustrated in Fig. 13. In this case there are inputs with 3 number of "1", which means that the 4th column is going to proceed to the PPE. The area and delay results are shown in Tables 2 and 3 respectively.

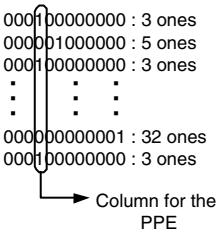

Fig. 13. Column selection.

Table 2
Input arbiter

Input arbiter	Area (slices)	Delay (ns)
1's counters (32)	128	9.9
Column selection	16	9
PPE	84	13.2
Total	228	32.1

Table 3
Output arbiter

Output arbiter	Area (slices)	Delay (ns)
Priority selection	113	4
PPE	84	13.2
Total	197	17.2

The flow of cells across the buffered crossbar and scheduling phases is divided into 4 stages as depicted in Fig. 14. We use Rocket I/O transceivers as in [27] to move cells across the VOQ/IBC switch. Note that a VOQ becomes eligible as soon as it receives the first block of bytes (8 bytes using Rocket I/O transceivers) of a cell. Likewise, the status of an internal crosspoint buffer, XP , becomes "empty" as soon as the first block of bytes of a cell is being transferred to the output port. If the external line speed is 10 Gbps (OC − 192 line rate) and using the Rocket I/O transceivers[4], then each cell needs 44.8 ns[5] to move from one stage to the next. Hence, each stage of the pipeline must be less than 44.8 ns to complete. Our algorithm meets this timing requirement and can even allow time for additional packet processing tasks such as QoS related information and flow control.

[4] Note that our target FPGA device contains only 24 Rocket I/Os. In our case, we can make use of 8 standard I/O pins for the design of a 32×32 switch.

[5] Normally, at 10 Gbps line rate, an ATM cell would need 42.4 ns to be scheduled. However, the rocket I/O transfers the cell in blocks of 8 bytes at a time, rounding the last 5 bytes of each ATM cell to 8 bytes.

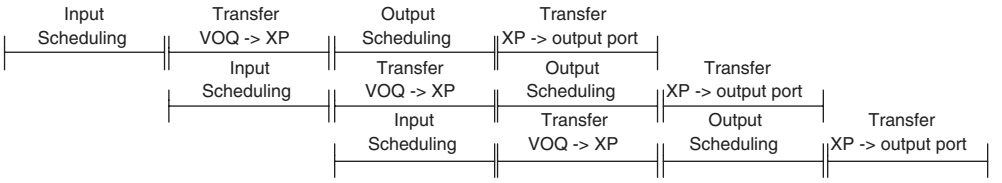

Fig. 14. Flow of cells.

6. Conclusion

The internally buffered crossbar switch architecture (VOQ/IBC) has been shown to be an attractive option to overcome the challenges met by the IQ switch architecture. However, the scheduling schemes presented for the VOQ/IBC switch have been just a simple mapping of earlier algorithms, proposed for the bufferless IQ switch. As a result, they did not benefit from any of the advantages that the VOQ/IBC architecture offers. In this paper, we proposed a set of scheduling schemes that are purely advocated for the VOQ/IBC switch architecture. The CBF we proposed bases its decision exclusively on the internal buffers information. It was shown to exhibit very good performance and outperforms all the previously presented schemes. In an attempt to further reduce the scheduling complexity and simplify the hardware implementation, we proposed the CAF_PRMV class of practical algorithms. We addressed the issue of the interaction between the input line cards and the internal buffers as well as the matching process between the input scheduler and the output scheduler. The simulation results showed that our newly proposed algorithms outperform state-of-the-art algorithms in this area. In particular, the CAF_PRMV1 algorithm performs very well under all traffic patterns, and was shown to be the best among all others. Its main advantage lies in the fact that it is totally stateless. Hence, it requires simple hardware while running at very high speed. The hardware implementation showed that our CAF–PRMV class of algorithms can sustain a 10 Gbps line speed for a 32×32 VOQ/IBC switch.

Acknowledgement

This work was partly supported by the Hong Kong RGC grant (HKUST6260/04E).

References

[1] 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 17 (06) (1999) 1030–1039.

[2] W.J. Dally, P. Carvey, L. Dennison, The avici terabit switch/router, in: Proceedings of Hot Interconnects 6, Aug. 1998, pp. 41–49.

[3] N. McKeown, Scheduling algorithms for input-queued cell switches, Ph.D. thesis, University of California, Berkeley, May 1995.

[4] M. Karol, M. Hluchyj, S. Morgan, Input versus output queuing on a space-division packet switch, IEEE Transactions on Communications 35 (09) (1987) 1337–1356.

[5] T. Anderson, S. Owicki, J. Saxe, C. Thacker, High speed switch scheduling for local area networks, ACM Transactions on Computer Systems (1993) 319–352.

[6] H.C. Chi, Y. Tamir, Symmetric crossbar arbiters for VLSI communication switches, IEEE Transactions on Parallel and Distributed Systems 04 (01) (1993) 13–27.

[7] S. Keshav, R. Sharma, Issues and trends in router design, IEEE Communications Magazine 36 (09) (1998) 144–151.

[8] H.C. Chi, Y. Tamir, Starvation prevention for arbiters of crossbars with multi-queue input buffers, IEEE ICC (June) (1992) 1646–1650.

[9] D.N. Serpanos, P.I. Antoniadis, FIRM: a class of distributed scheduling algorithms for high-speed ATM switches with input queues, IEEE Infocom (March) (2000).

[10] A. Mekkittikul, N. McKeown, A starvation-free algorithm for achieving 100% throughput in an input-queued switch, ICCCN (Oct.) (1996) 226–231.

[11] N. McKeown, iSLIP scheduling algorithm for input-queued switches, IEEE Transactions On Networking 07 (02) (1999) 188–201.

[12] Y. Jiang, M. Hamdi, A fully desyncronized round-robin matching scheduler for a VOQ packet switch architecture, in: IEEE Workshop on High Performance Switching and Routing, 2001, pp. 407–411.

[13] A.K. Gupta, L.O. Barbosa, N.D. Gorganas, 16×16 Limited intermediate buffer switch module for ATM networks for B-ISDN, IEEE Globecom (Dec.) (1991) 939–943.

[14] A.K. Gupta, L.O. Barbosa, N.D. Gorganas, Limited intermediate buffer switch modules and their interconnection for B-ISDN, IEEE ICC (June) (1992) 1646–1650.

[15] S. Nojima, E. Tsutsui, H. Fukuda, M. Hashimmoto, Integrated packet network using bus matrix, IEEE Transactions on Communications 05 (08) (1987) 1284–1291.

[16] M. Nabeshima, Performance evaluation of combined input- and crosspoint-queued switch, IEICE Transaction on Communications B83-B (3) (2000).

[17] L. Mhamdi, M. Hamdi, Output queued switch emulation by a one-cell-internally buffered crossbar switch, in: EEE Global Telecommunications Conference, GLOBECOM'03, Dec. 2003.

[18] B. Magill, C. Rohrs, R. Stevenson, Output-queued switch emulation by fabrics with limited memory, IEEE Journal on Selected Areas in Communications (May) (2003) 606–615.
[19] S. Chuang, S. Iyer, N. McKeown, Practical algorithms for performance guarantees in buffered crossbars, IEEE Infocom (March) (2005).
[20] T. Javadi, R. Magill, T. Hrabik, A high-throughput algorithm for buffered crossbar switch fabric, IEEE ICC (June) (2001) 1581–1591.
[21] R. Rojas-Cessa, E. Oki, Z. Jing, H.J. Chao, CIXB-1: Combined input one-cell-crosspoint buffered switch, in: Proceedings of the 2001 IEEE WHPSR, 2001, pp. 324–329.
[22] K. Yoshigoe, K.J. Christensen, A parallel-polled virtual output queued switch with a buffered crossbar, in: IEEE Workshop on High Performance Switching and Routing, 2001, pp. 271–275.
[23] L. Mhamdi, M. Hamdi, MCBF: A high-performance scheduling algorithm for buffered crossbar switches, IEEE Communications Letters 07 (09) (2003) 451–453.
[24] A. Mekkittikul, Scheduling non-uniform traffic in high speed packet switches and routers, Ph.D. thesis, Stanford University, Nov. 1998.
[25] N. McKeown, A. Mekkittikul, V. Anantharam, J. Walrand, Achieving 100% throughput in input-queued switch, IEEE Transactions on Communications 47 (08) (1999).
[26] P. Gupta, N. McKeown, Design and implementation of a fast crossbar scheduler, IEEE Micro 19 (01) (1999).
[27] K. Yoshigoe, A. Jacob, K.J. Christensen, The RR/RR CICQ switch: hardware design for 10 Gbps link speed, in: IEEE International Performance, Computing, and Communications Conference, April 2003, pp. 481–485.

Lotfi Mhamdi (S'02) received the B.S. degree in Information Systems Management Technology from South University, Tunisia, in 2000 and the MPhil Degree in computer science from the Hong Kong University of Science and Technology (HKUST), Clear Water Bay, Hong Kong, in 2002. He is currently working towards the Ph.D. in Computer Engineering in the Delft University of Technology, The Netherlands. His research work spans the area of high-speed networks including the design, analysis, scheduling, and management of high-speed switches and Internet routers. He is a student member of IEEE.

Mounir Hamdi received the B.S. degree in Computer Engineering (with distinction) from the University of Louisiana in 1985, and the MS and the Ph.D. degrees in Electrical Engineering from the University of Pittsburgh in 1987 and 1991, respectively. He has been a faculty member in the Department of Computer Science at the Hong Kong University of Science and Technology (HKUST) since 1991, where he is now Full Professor of Computer Science, Director of the Computer Engineering Program and Director of the Computer Engineering and Networking Lab. In 1999–2000 he held visiting professor positions at Stanford University, USA, and the Swiss Federal Institute of Technology, Lausanne, Switzerland. His general areas of research are in high-speed wired/wireless networking. Currently, he is working on the design, analysis, scheduling, and management of high-performance Internet switches/routers, algorithm/architecture co-design, wavelength division multiplexing (WDM) networks/switches, and high-speed wireless networks. In particular, he is leading a research team at HKUST that is designing one the highest capacity chip sets for Terabit switches/routers in the world.

Dr. Hamdi received the best paper award at the International Conference on Information and Networking in 1998 out of 152 papers. He also supervised the best Ph.D. paper award amongst all universities in Hong Kong. He received the best 10 lecturers award (through university-wide student voting for all university faculty held once a year), the distinguished engineering teaching appreciation award from HKUST, and various grants targeted towards the improvement of teaching methodologies, delivery and technology. He is a member of IEEE and ACM.

Christopher Kachris received the diploma and the M.Sc. in Computer Engineering from the Technical University of Crete, Greece in 2001 and 2003 respectively. In 2003 he joined Ellemedia Technologies in Athens, Greece designing network processors. Currently, he is working towards his Ph.D. in Computer Engineering in the Delft University of Technology, The Netherlands. His research interests include computer architecture, network and media processors and reconfigurable logic.

Stephan Wong was born in Paramaribo, Suriname in 1973. He obtained his Ph.D. in the Electrical Engineering department of the Delft University of Technology (TU Delft), The Netherlands. He is currently working as an assistant professor at the Computer Engineering Laboratory at the Delft University of Technology (TU Delft), The Netherlands. He has considerable experience in the design of embedded media processors. He has worked also on microcoded FPGA complex instruction engines and the modeling of parallel processor communication networks. His research interests include embedded systems, multimedia processors, complex instruction set architectures, reconfigurable and parallel processing, microcoded machines, and network processors.

Stamatis Vassiliadis (M'86–SM'92–F'97) was born in Manolates, Samos, Greece, in 1951. He is currently a Chair Professor in the Electrical Engineering, Mathematics, and Computer Science (EEMCS) department of Delft University of Technology (TU Delft), The Netherlands. He previously served in the Electrical and Computer Engineering faculties of Cornell University, Ithaca, NY and the State University of New

York (S.U.N.Y.), Binghamton, NY. For a decade, he worked with IBM, where he was involved in a number of advanced research and development projects. He received numerous awards for his work, including 24 publication awards, 15 invention awards, and an outstanding innovation award for engineering/scientific hardware design. His 72 USA patents rank him as the top all time IBM inventor.

Dr. Vassiliadis received an honorable mention Best Paper award at the ACM/IEEE MICRO25 in 1992 and Best Paper awards in the IEEE CAS (1998, 2001), IEEE ICCD (2001), PDCS (2002) and the best poster award in the IEEE NANO (2005). He is an IEEE and ACM fellow and a member of the Dutch Academy of Science.