CIXOB-k: Combined Input-Crosspoint-Output Buffered Packet Switch

Roberto Rojas-Cessa,* *Student Member, IEEE*, Eiji Oki,** *Member, IEEE*, and
H. Jonathan Chao, *Fellow, IEEE*
Department of Electrical Engineering
Polytechnic University
6 Metrotech Center, Brooklyn, New York 11201, USA

Abstract—

We propose a novel architecture, a Combined Input-Crosspoint-Output Buffered (CIXOB-k, where k is the size of the crosspoint buffer) Switch. CIXOB-k architecture provides 100% throughput under uniform and unbalanced traffic. It also provides timing relaxation and scalability. CIXOB-k is based on a switch with Combined Input-Crosspoint Buffering (CIXB-k) and round-robin arbitration. CIXB-k has a better performance than a non-buffered crossbar that uses iSLIP arbitration scheme. CIXOB-k uses a small speedup to provide 100% throughput under unbalanced traffic. We analyze the effect of the crosspoint buffer size and the switch size under uniform and unbalanced traffic for CIXB-k. We also describe solutions for relaxing the crosspoint memory amount and scalability for a CIXOB-k switch with a large number of ports.

Keywords— **Scheduling, round-robin matching, buffered crossbar, unbalanced traffic.**

I. Introduction

The explosion of Internet traffic has led to a greater need for high-speed switches and routers that have over 1-Tbit/s throughput [1]. Crossbar switching fabrics are very popular for switch implementation because of their non-blocking capability, simplicity, and market availability. A switch with a crossbar fabric and queues at the output ports to store those cells that could not be sent to the output lines is called Output Buffered (OB). In an OB switch, all cells coming to an input are forwarded to the destined outputs as they arrive. This architecture is not scalable because the required internal bandwidth or speedup (S) — defined as the number of times that the switch core works faster than the input line rate — is equal to the number of ports ($S = N$); the working speed of the switch core and the output memory make an OB switch implementation infeasible for even a medium-sized switch. However, this architecture has been used as a comparison reference for any switch model because of its desirable characteristics such as high throughput and low delay.

Crossbars could have input queues to store those cells (or packets) that could not go through because of contention for an output; this architecture is known as Input Buffered (IB). An IB architecture is scalable and its implementation does not have the restrictions of the OB model because the core fabric works at the input line rate ($S = 1$). However, IB switches need to resolve input and output contention by means of arbiters at the inputs and outputs. The requirements for such arbiters are (a) low complexity, (b) fast contention resolution and, (c) high efficiency to provide a high performance. Low complexity is needed to make implementation feasible. For a high-capacity switch, a fast resolution is necessary so that the arbiter can select a cell among those eligible in the allotted time.

Head of Line (HOL) blocking is a well-known problem for a crossbar with FIFOs at the inputs [2]. This problem is overcome by using separate queues at the inputs, one for each output. This queuing system is called Virtual Output Queuing (VOQ). For a crossbar with VOQs, maximum matching algorithms have been proposed to achieve 100% throughput. Maximum matching algorithms are efficient but with such a high complexity [3] that implementation is infeasible for high-speed systems. Schemes based on a maximum size or weight matching, like Longest Port Queuing (LPQ), Oldest Cell First (OCF), and Longest Port First (LPF) [4], have been proposed [3].

Maximal matching schemes have been considered as an alternative to maximum matching schemes; iSLIP [5], Dual Round-Robin Matching (DRRM) [6], [7], and Longest Output Occupancy First Algorithm (LOOFA) [8] are examples. To make up for the lack of efficiency that a maximal scheme has (compared to a maximum type), speedup, a number of iterations — the number of times that an algorithm is performed in a single scheduling cycle to obtain a cumulative result–, or a combination of both is used, as in LOOFA. iSLIP is a good example of an iterative matching scheme. Although iSLIP provides 100% throughput for uniform independent traffic, because of the arbitration time and connection state amount of this arbitration scheme, it has been proposed for a small number of ports [9] due to its centralized implementation (i.e., 32 for iSLIP). Transmission of phases — request, grant, and acknowledge – is performed within a cell slot between input and output arbiters. This transmission of information reduces the available time for arbitration because these transmission phases are performed during the cell slot in

This research is supported by NSF Grant 9906673.

*Corresponding author, E-mail:rrojas@kings.poly.edu, phone: (718)-260-3496, fax: (718)-260-3906.

**He is a Research Engineer with NTT Network Service Systems Laboratories, Tokyo, Japan. This work was done while he was a Visiting Scholar at Polytechnic University.

serial with input and output arbitration, even when the transmissions are done within a single chip (so that the off-chip delay is avoided). Another drawback with the proposed single-chip centralized implementation is that the pin count limits the number of ports.

A switch architecture using speedup, such that $1 < S < N$, is called Combined Input and Output Buffered (CIOB), where queues are placed at the inputs and outputs. As the demand for high switching rates increases, this speedup becomes a bottleneck since the available time for arbitration is inversely proportional to the cell slot duration divided by S. The DRRM scheme considers speedup instead of a number of iterations to improve the matching performance. Although the overhead information exchanged between input and output arbitration is reduced in this scheme, the arbitration time becomes insufficient for a switch with a large number of ports and with a high port speed.

For a long time, buffered crossbars have been considered as a solution to improve switching throughput instead of non-buffered crossbars. However, it is known that the number of buffers would grow in the same order as the number of crosspoints ($O(N^2)$, where N is the number of ports), making implementation infeasible for the memory required by a large buffer or a large N.

In pure buffered crossbars — a pure crossbar architecture has only buffering at the crosspoints and none in any other place — a large crosspoint buffer has been utilized to minimize cell loss. The number of ports is limited by the memory amount that can be implemented in a module chip. An example of this architecture was proposed in [11], where a 2×2 switch module with a crosspoint memory of 16 kbytes each was implemented. In this architecture, a large crosspoint buffer is needed to store all those cells that could not be switched to the output port to comply with the required cell loss rate.

To reduce the memory in the crosspoint buffer, input queues are used. FIFO queues have been proposed, where HOL blocking, as in a non-buffered crossbar, remains in this architecture. Examples of these architectures were presented in [12], [13], and [14]. A buffered crossbar with a single-cell buffer was proposed in [12] and [13], together with a FIFO input buffer at the input ports. This architecture provides an improvement over non-buffered crossbars with FIFO input buffers. The well-known limited throughput of a FIFO input-buffered architecture of about 58% was improved to 91% with a priority scheme (also called HOL blocking scheme by the same author). However, the FIFO buffers at the inputs limit the maximum throughput performance in this architecture because the HOL blocking can not be completely eliminated. In [14] a similar architecture with a 4-cell crosspoint buffer is considered. This buffered crossbar, used with 32-cell input FIFOs, achieves an acceptable cell loss (10^{-8}). In this architecture, a flow control mechanism is also used to avoid cell loss at the core. All cell loss occurs at the input FIFO for a very congested output. This study shows that with input FIFOs, a small-sized crosspoint buffer, and a control mechanism, the cell loss rate can be kept small and the HOL blocking diminished to a certain degree.

As with maximal matching schemes as in non-buffered crossbars, the HOL blocking problem for FIFO buffers can be overcome in a buffered crossbar with the consideration of VOQs.

100% throughput is obviously achieved for a buffered crossbar with infinite crosspoint buffer sizes [15], [16], and [17]. To our knowledge, no minimum finite memory size has been specifically proven to provide 100% throughput for a buffered crossbar.

[18] proposed a Combined Input-crosspoint buffered (CIXB-1) switch model, where the crosspoint buffer has one-cell size, with VOQs at the inputs and simple round-robin for input and output arbitration. It showed that the combination of input buffers and single-cell crosspoint buffers (i.e., $k = 1$) and a round-robin arbitration scheme provides 100% throughput under uniform traffic. A VOQ structure is provided in the input buffers. In this architecture input and output arbitration are more independent than in a non-buffered crossbar model using a maximal matching arbitration, simplifying arbitration time complexity. Also, in a CIXB-k switch — where k is the crosspoing buffer size —, the arbitration can be performed during a complete cell slot. The arbitration time can be separated from the transmission time, allowing synchronization flexibility and consideration of a large number of ports. The properties of a buffered crossbar allows a simplification of the arbiter design and the adoption of distributed-fashion arbiters. In this switch, fixed-size cells are transmitted through the switching fabric to ease implementability. The impact on the minimum amount of memory at each crosspoint by the effect of the transmission delay between an input card and the crossbar allowing the switch to keep up with the features of CIXB-1. It has been shown that CIXB-1 has a better delay performance for uniform and unbalanced traffic compared to a non-buffered crossbar with iSLIP arbitration. However, CIXB-1 has does not provide 100% throughput for unbalanced traffic [1] and the timing relaxation is limited, the distance between the port cards and the crossbar has to be less than one time slot. It is desirable to provide 100% throughput under unabalanced traffic and relax the timing in a larger proportion. It is important to show the trade off between the value of k and the timing relaxation. When adopting a CIXB-k architecture, a small size module has to be implemented due to the pin count and memory amount limitation. For implementation of a large N switch, several CIXB-k modules have to be placed in a bi-dimensional array. However, since two or more CIXB-k modules are addressed to the same output and since the distributed arbiters within a module are independent, timing relaxation in the outputs of the modules and synchronization may be lost.

In this paper we present a Combined Input-Crosspoint-Output buffered (CIXOB-k, where k is the crosspoint buffer size) switch. This model keeps the advantages

[1] We refer to the model for unbalanced traffic to the one used in [18], also presented here.

of CIXB-k and provides 100% throughput under uniform and unbalanced traffic. We show the effect of CIXB-k when increasing the crosspoint buffer size under unbalanced traffic and observe that with a minimum speedup, 100% throughput can be achieved. CIXOB-k uses a smaller speedup that a non-buffered crossbar with a round-robin arbitration scheme to provide 100% throughput under unbalanced traffic.

Since a large memory amount is limited by the available VLSI technology, we discuss a solution for making a high amount of memory feasible of implementation. Also, we show that CIXOB-k provides scalability for a large number of ports while maintaining the properties of CIXB-k and keeping the memory amount low. Our solution for scalability can be also used for a CIXB-k switch model. CIXOB-k also provides a solution for timing relaxation.

This paper is organized as follows. Section II describes the CIXB-k switch model and its properties. Section III shows the performance of the described architecture. Section IV shows a solution for scalability. Section V describe our conclusions.

II. Switch Model

In this section, we describe the CIXOB-k switch model. We introduce the proposed architecture and the terminology used in the rest of the paper.

A Buffered Crossbar (BX) has N input and output ports. A crosspoint (XP) element in the BX that connects input i, where $0 \leq i \leq N-1$, to output j, where $0 \leq j \leq N-1$, is denoted as $XP_{i,j}$. The XP Buffer (XPB) of $XP_{i,j}$ is denoted as $XPB_{i,j}$.

We consider fixed size packets, named cells. A variable length packet can be segmented into cells for internal switching and re-assembled before it leaves the switch. The transmission time has a fixed length, called cell or time slot.

Our switch model has a structure as shown in Figure 1 and as it is described below:

- *Input Queue.* There are N VOQs at each input port. A VOQ at input i that stores cells for output j is denoted as $VOQ_{i,j}$.
- *Crosspoint Buffer XPB.* Each crosspoint has a one-cell buffer. Only those inputs with a cell in the crosspoint buffer are considered for output arbitration.
- *Output Queue.* There is an output queue at each output port to receive a transmitted cell before the cell leaves the switch.
- *Flow Control.* A flow control mechanism tells input port i which $XPB_{i,j}$ is occupied, so that the $VOQ_{i,j}$ is inhibited (a non-empty VOQ is considered eligible for input arbitration if this VOQ is not inhibited). In this paper, we consider a credit-base flow control mechanism, unless otherwise specified.
- *Speedup* A speedup, $S = \frac{X_r}{I_r}$, is provided for transmission along the buffered crossbar. Is the speed in what the port cards transmit to and receive from the buffered crossbar. X_r is the buffered crossbar line rate and I_r is the input line rate.
- *Round Trip (RT).* We define RT as standing at the input port. RT is the composite time of the transmission cell delay from a port card to the crossbar ($d1$) plus the Output Arbitration time (OA) and the transmission of the flow control information back to the portcard ($d2$), as shown in Figure 2. Cell and data alignments are included in the transmission times. In a general case, as presented in [18]:

$$RT = d1 + OA + d2 \leq k - IA \tag{1}$$

  where IA is the input arbitration time and k is the crosspoint buffer size (i.e., CIXOB-4 means $k = 4$). The constrains for IA and OA are:

$$IA \leq 1 \tag{2}$$

  and

$$OA \leq 1. \tag{3}$$

- *Arbitration.* Round-robin arbitration is used at the input and output ports. An input arbiter selects a VOQ, among the eligible VOQs, to send a cell to the BX. Request eligibility for VOQs is determined by the flow control mechanism. An output arbiter selects a buffered cell among non-empty crosspoint buffers to forward a cell to the output.

A. Properties of CIXOB-k

In this section, we list the properties of CIXOB-k.

Property 1: As originally described in [18], CIXB-1 provides 100% throughput under Bernoulli uniform traffic. This property is disclosed as in [18] in Appendix A. CIXOB-k inherits this property since it represents a CIXB-k when $S = 1$ (or no speedup).

Property 2: CIXOB-k allows timing relaxation for a transmission delay between port cards and BX. It can provide one time slot for input or output arbitration. Timing relaxation can be achieved as long as Eq.(1) is satisfied.

III. Performance Study

To observe the performance of CIXOB-k we simulated CIXB-k. In this section, we present the simulation results of a 32×32 CIXB-k under Bernoulli and bursty (On-off model) traffic with a uniform distribution, and unbalanced traffic. We compare the performance of CIXB-k with an OB switch and a non-buffered crossbar with iSLIP arbitration under uniform traffic. Also, we compare CIXB-k with IB switches with non-buffered crossbar with iSLIP and PIM arbitrations under unbalanced traffic. We also show the effect of different buffer sizes of CIXB-k under unbalanced traffic.

A. Uniform traffic

We show the delay performance of CIXB-1 since CIXOB-k inherits its properties. The results of the simulation of CIXB-1, iSLIP with 1 and 4 iterations (1-SLIP and 4-SLIP, respectively) and OB for traffic uniformly

distributed to all output ports with Bernoulli arrivals are shown in Figure 3. Under independent uniform traffic, CIXB-1 has a smaller average delay than 4-SLIP. We can also see that CIXB-1 has comparable average delay performance to OB.

Figure 4 shows that CIXB-1 has almost the same average delay performance as CIXB-k for other k values under Bernoulli and bursty ($l = 10$) uniform traffic with a load of $\rho = 0.9$. Figure 5 shows the tail delay probability of CIXB-k for different k values at different traffic loads. It is seen that the tail delays are almost the same for any crospoint buffer size for Bernoulli uniform traffic for a given traffic load. The buffer size can be kept as small as possible to minimize the amount of memory without having a significant performance degradation.

As CIXOB-k uses the same switch model plus speedup, it also offers 100% throughput under this traffic model. The OQ at CIXOB-k can be used to relax the transmission time between BX and the port card, even in the lack of speedup ($S = 1$).

B. Unbalanced Traffic

In this section, we show the necessary speed up for CIXOB-k to provide 100% throughput under unbalanced traffic by observing the throughput performance of CIXB-k.

B.1 Traffic Model

We use the same model as in [18]. We define non-uniform traffic by using an unbalanced probability w. Let us consider input port s, output port d, and the offered input load for each input port ρ. The traffic load from input port s to output port d, $\rho_{s,d}$ is given by,

$$\rho_{s,d} = \begin{cases} \rho\left(w + \frac{1-w}{N}\right) & \text{if } s = d \\ \rho\frac{1-w}{N} & \text{otherwise.} \end{cases} \quad (4)$$

where N is the switch size. Here, the aggregate offered load that goes to output d from all input ports, ρ_d is given by,

$$\rho_d = \sum_s \rho_{s,d} = \rho\left(w + N \times \frac{1-w}{N}\right) = \rho. \quad (5)$$

When $w = 0$, the offered traffic is uniform. On the other hand, when $w = 1$, the traffic is completely unbalanced. This means that all the traffic of input port s is destined for output port d only, where $s = d$.

B.2 Speedup in CIXOB-k

As it was shown in [18], CIXB-1 with round-robin offers a better throughput than iSLIP under unbalanced traffic. Here, we compare CIXB-k with IB switches with non-buffered crossbars using iSLIP and PIM[5] for one and four iterations. Figure 6 shows the throughput under this traffic model for all these schemes. We can see that the achievable throughput of CIXB-1, is always better than that non-buffered crossbar with iSLIP and PIM. However, this is less than 100% throughput.

In Figure 7, we show the performance of CIXB-k for unbalanced traffic for different k values. Even though, the performance of CIXB-k improves as k is increased for unbalanced traffic, it offers a throughput only close to 100%. However, as the buffer size is increased, implementation cost arises. Figure 8 shows the minimum throughput obtained for different values of k and for different switch sizes (i.e., values of N). These values are obtained from Figure 7.

As seen, CIXB-k with a large k provides a throughput close to 100%. Furthermore, a large memory size is considered impractical for implementation in a single 32×32 switch module. A solution is to use a small k and a small speedup. Figure 9 shows the speedup, obtained from Figure 8, to achieve this objective.

It can be seen that the speedup is very small and the value range is narrow for different k values, so it is practical to consider the small k, according to how much the time for transmission between port cards and BX need to be relaxed.

IV. Scalability of CIXOB-k

In this section we discuss relaxation of memory constrain in CIXOB-k and a way to scale up this switch model for a large number of N.

A. Relaxing the Memory Amount

In CIXOB-k, a major concern is to allocate the memory amount in the crossbar since for a buffered crossbar the amount of memory is in order $O(N^2)$. For $k > 1$, the memory amount is $M = kN^2$. The value of k is mainly selected by the needed time relaxation. In a switch module, the maximum permissible size of k is set by the current VLSI technology limitation. However, if a larger k value is needed, a multiple-plane implementation can be used [6]. Figure 10 shows this implementation. The total amount of memory is divided by the number of planes:

$$M = \frac{kN^2}{m}, \quad (6)$$

where m is the number of planes. In this architecture, a cell is segmented into the number of planes (i.e., m) and each segment is forwarded simultaneously or in the same order.[2] At the inputs and outputs, there is a VOQ for each plane. In this way the memory constraint is relaxed. This technique can also be used by a switch as CIXB-k, where there is no speed up. The major implementation constraint for a CIXOB-k is still the pin count, which constrains the number of ports. A solution for extending the number of ports N is presented below.

B. CIXOB-k with a large N

We can scale up the switch size by using $n \times n$ CIXOB-k switch modules, where $n < N$, and connecting them as a bi-dimensional matrix as it would be done with non-buffered crossbar modules. However, since two or more

[2] Depending on the synchronization between planes.

CIXB-k modules are connected to the same output port card and since the distributed arbiters within a module are independent, timing relaxation can be lost in the outputs of the modules and synchronization among all modules has to be resolved.

In order to maintain the timing relaxed at the output when joining two or more output modules, $h = \frac{N}{n}$ Output Queues $OQ(l)$ are allocated at the output port cards, where $0 \leq l \leq h-1$. A round-robin arbiter selects the next $OQ(l)$ that forwards a cell to the output line each time slot. Figure 11 shows this description.[3] Since all OQs are independent, they are able to receive a cell simultaneously.

When using this scalable implementation, the throughput of the switch is not affected because the OQs do not affect the behavior of BX. This solution can also be used by CIXB-k.

V. Conclusions

We have presented a CIXOB-k switch model that relaxes the timing for arbitration and cell transmission and provides 100% throughput for uniform and unbalanced traffic. We presented the behavior of CIXB-k under unbalanced traffic and compared to a non-buffered switch model with iSLIP and PIM. We showed that CIXB-k offers better performance than IB switches with iSLIP and PIM. By studying the performance of CIXB-k for different values of k, the sufficient speedup to provide 100% throughput under unbalanced traffic can be determined by selecting the value of the crosspoint buffer size. The sufficient speedup for CIXOB-1 is smaller than the one needed by a non-buffered crossbar with a maximal-matching arbitration scheme, as iSLIP. With a relaxed timing with this switch model, a necessary value of k can be selected. The resulting amount of memory needed can be relaxed by using a multiple-plane implementation. We also presented a solution for to scale CIXOB-k for a large number of ports N. The solutions for scaling up memory and switch size can also be applied to CIXB-k.

References

[1] N. Yamanaka, E. Oki, S. Yasukawa, R. Kawano, and K. Okazaki, "OPTIMA: Scalable, Multi-Stage, 640-Gbit/s ATM Switching System Based on Advanced Electronic and Optical WDM Technologies," IEICE Trans. Commun., vol. E83-B, no. 7, pp. 1488-1496, 2000.

[2] M. Karol and M. Hluchyj, "Queuing in High-performance Packet-switching," *IEEE J. Selected Area Communications*, vol. 6, pp. 1587-1597, December 1988.

[3] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100% Throughput in an Input-queued Switch," *IEEE Trans. On Comm.*, vol. 47, no. 8, pp. 1260-1267, August 1999.

[4] A. Mekkittikul and N. McKeown, "A Practical Scheduling Algorithm to Achieve 100% throughput in input-queued switches," *INFOCOM '98*, vol. 2 , pp. 792 -799, March 1998.

[5] N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queue Switches", *IEEE Trans. Networking*, vol. 7, no. 2, pp. 188-200, April 1999.

[6] H. J. Chao, "Saturn: A Terabit Packet Switch Using Dual Round-Robin," *IEEE Comm Magazine*,vol. 38, no. 12, pp. 78-84, December 2000.

[3] Where for the case of multiple-plane implementation, this structure is replicated for each plane.

[7] Y. Li, S. Panwar, and H. J. Chao, "On the Performance of the Dual Round-robin Switch," *to appear in IEEE INFOCOM 2001*, April 2001.

[8] P. Krishna, N. S. Patel, A. Charny, and R. Simcoe, "On the Speedup Required for Work-Conserving Crossbar Switches," *IEEE Selected Areas in Communications*, vol. 27, no. 6, pp. 1052-1066, June 1999.

[9] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz, "Tiny Tera: A Packet Switch Core," *IEEE Micro*, vol. 17, no. 1, pp. 26 -33, Jan.-Feb. 1997.

[10] Texas Instruments, GS40 0.15-μm CMOS, Standard Cell/Gate Array, http://www.ti.com/, version 0.2, May 2000.

[11] S. Nojima, E. Tsutsui, H. Fukuda, and M. Hashimmoto, "Integrated Packet Network Using Bus Matrix," *IEEE JSAC*, vol. SAC-5, no. 8, pp. 1284-1291, October 1987.

[12] A. K. Gupta, L. O. Barbosa, and N. D. Georganas, "16 x 16 Limited Intermediate Buffer Switch Module for ATM Networks," *GLOBECOM '91*, pp. 939-943, December 1991.

[13] A. K. Gupta, L. O. Barbosa, and N. D. Georganas, "Limited Intermediate Buffer Switch Modules and their Interconnection Networks for B-ISDN," *ICC '92*, pp. 1646-1650, June 1992.

[14] Y. Doi and N. Yamanaka, "A High-Speed ATM Switch with Input and Cross-Point Buffers," *IEICE TRANS. COMMUN., VOL. E76, NO.3*, pp. 310-314, March 1993.

[15] F. A. Tobagi, "Fast Packet Switch Architectures For Broadband Integrated Services Digital Networks," *Proceedings of the IEEE*, vol. 78, No. 1, pp. 133-167, January 1990.

[16] H. Ahmadi and W. E. Denzel, "A Survey of Modern High-Performance Switching Techniques," *IEEE JSAC*, vol. 7, no. 7, pp. 1091-1103, September 1989.

[17] R. Y. Awdeh and H. T. Mouftah, "Survey of ATM Switch Architectures," *Computer Networks and ISDN Systems*, vol. 27, pp. 47-93, 1995.

[18] R. Rojas-Cessa, E. Oki, Z. Jing, and H. J. Chao, "CIXB-1: Combined Input-One-Cell-Crosspoint Buffered Switch," *to appear in IEEE HPSR2001 Proc.*, May 2001.

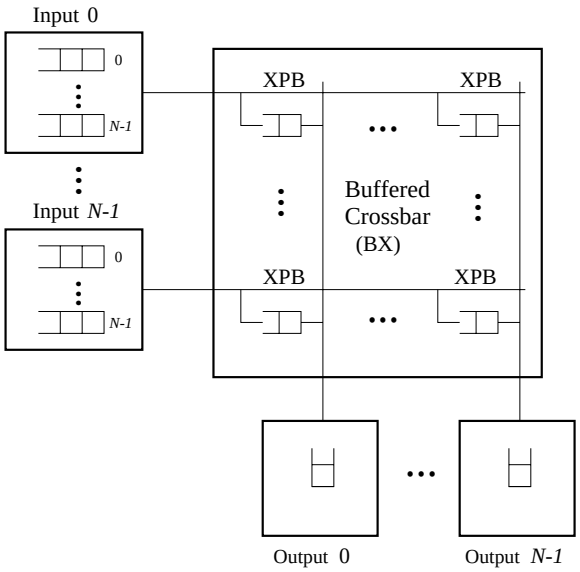
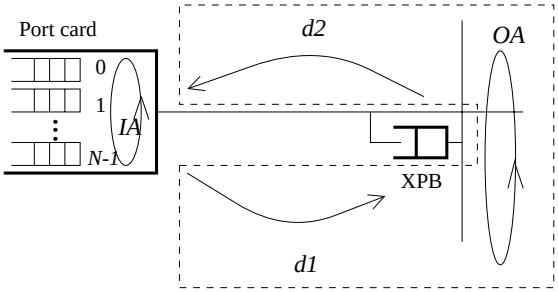


Fig. 1. Memory locations in CIXOB-k

Appendix

I. 100% throughput with CIXB-1

In this section, we show that CIXB-1 provides 100% throughput under uniform traffic. This property is expressed below as a theorem. The switch and traffic model described in Section II-A is considered for this proof. First, we present some definitions that are used in this section.



OA: Output Arbitration time
IA: Input Arbitration time
d1 Transmission delay from portcard to crossbar
d2 Transmission delay from crossbar to portcard

s

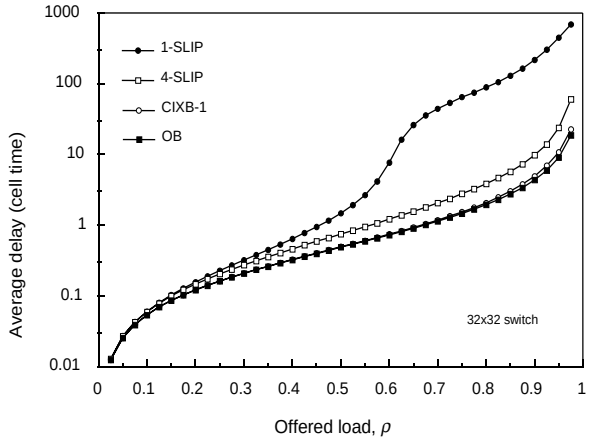


Fig. 3. Average delay of CIXB-1, iSLIP, and OB under Bernoulli uniform traffic

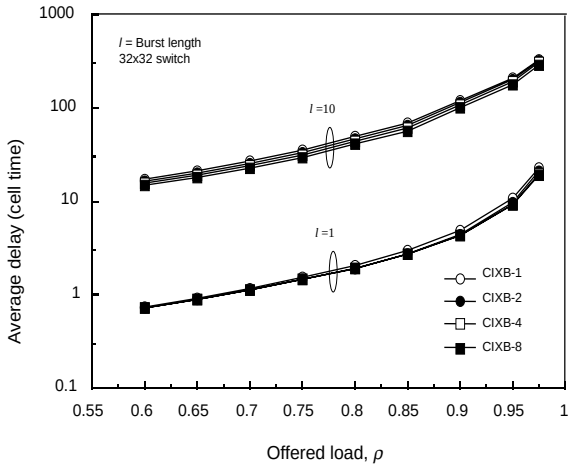


Fig. 4. Average delay of CIXB-k with different k and load ρ under bursty traffic

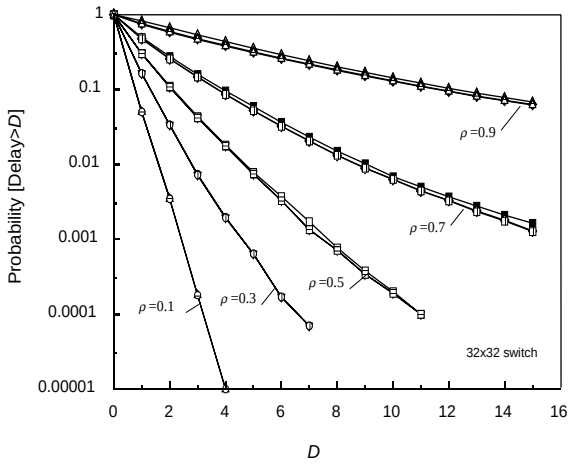


Fig. 5. Tail Delay Distribution for different load for different k

Definition 1: $VOQ_{i,j}$ is active if it has at least one buffered cell.

Definition 2: $VOQ_{i,j}$ is inhibited to send a request to BX when $XP_{i,j}$ is occupied.

Definition 3: Input port i is totally inhibited when all VOQs in it are inhibited.

Definition 4: Output j is active at time slot t if it has a cell to be delivered at time slot t.

Definition 5: Output port j is a non-active output at time slot t if it does not have a cell to be delivered at time slot t.

Let us denote the following:

- $X(t)$ is the number of inhibited input ports.
- $XP_{i,j}(t)$ is the state of crosspoint buffer $XPB_{i,j}$ at the beginning of time slot t. If a previous request that was not granted by the output arbiter remains in the crosspoint, $XP_{i,j}(t) = 1$. Otherwise, $XP_{i,j}(t) = 0$.

Although input and output arbitrations are performed one time slot before cells are transmitted in a practical implementation, we can consider that a request is the same as the transmitted cell.

Using the definitions presented above, 100% throughput is re-phrased as: *100% throughput is achieved when the conditions such that* $X(t) = 0$ *and all VOQs are served with an average rate equal to the average arrival rate.*

Theorem 1: CIXB-1 achieves 100% throughput under uniform traffic.

We prove Theorem 1, by proving Lemmas 1 and 2.

Lemma 1: In CIXB-1, no input is inhibited after N time slots from any initial condition. [4]

$$X(t) = 0. \tag{7}$$

Proof of Lemma 1

We use the following facts:

Fact 1 An input can send at most one request to the BX per time slot.

Fact 2 At output port j, occupied crosspoint $XP_{i,j}$, where $0 \leq i \leq N-1$, is served within N time slots.

[4] This means that the theorem is independent of the initial position of the input/output round-robin pointers, and the initial state of any $XPB_{i,j}(t)$.

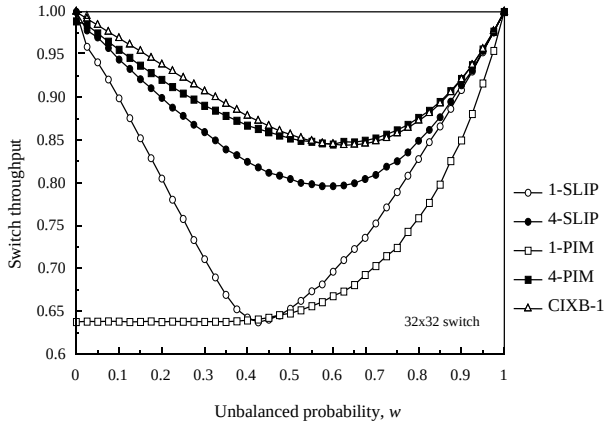


Fig. 6. Throughput of CIXB-1, iSLIP, and PIM with unbalanced traffic

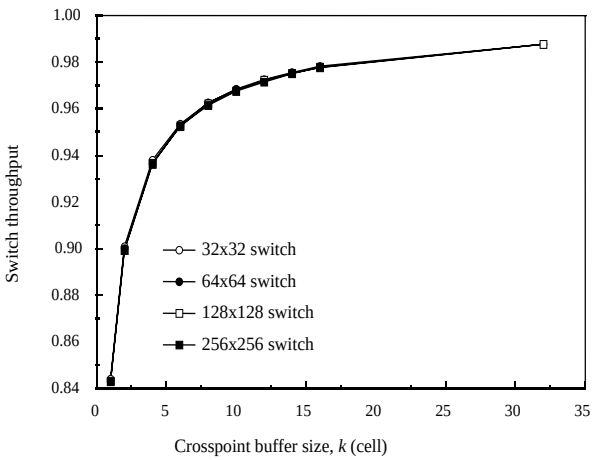


Fig. 8. Minimum Throughput of CIXB-k for different k and N values

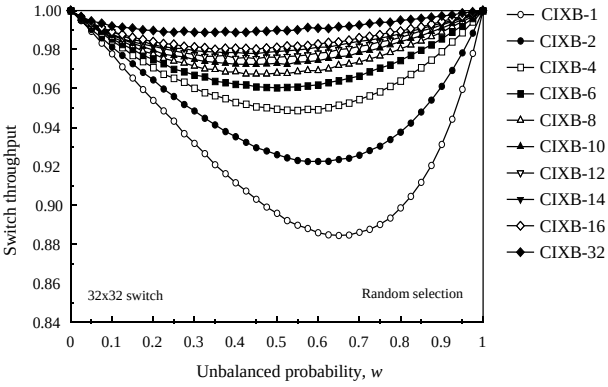


Fig. 7. Maximum Throughput of CIXB-k for different k and N values

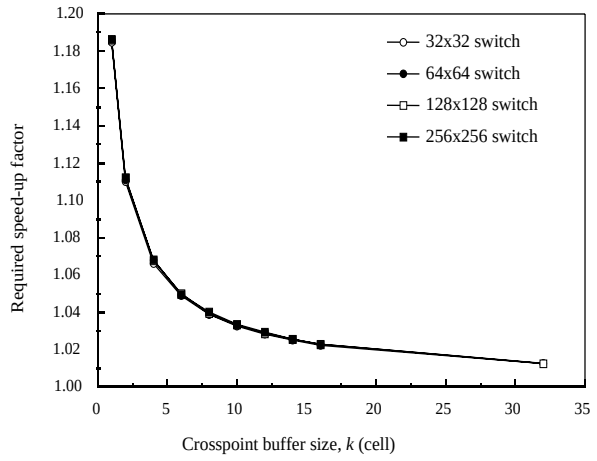


Fig. 9. Speedup to achieve 100% with CIXOB-k under unbalanced traffic

We prove this lemma by contradiction. Let us have the initial time $t = 0$. Assume that input i is inhibited at time $t = N$; all the crosspoints related to input i are in state $XP_{i,j}(N) = 1$, such that $\sum_j XP_{i,j}(N) = N$.

Independently of the initial state at $t = 0$,[5] these N requests could have been issued by input i only during the previous $N - 1$ time slots ($1 \leq t \leq N - 1$) due to Fact 2. However, according to Fact 1, during the previous $N-1$ time slots ($1 \leq t \leq N-1$) an input could sent at most $N-1$ requests. This contradicts the assumption of $\sum_j XP_{i,j}(N) = N$. Therefore, $\sum_j XP_{i,j}(N) \leq N-1$. In the same way, $\sum_j XP_{i,j}(t) \leq N - 1$ for any $t > N$.

After N time slots from the initial time, there is at least one $XP_{i,j}$ such that $XP_{i,j}(t) = 0$. Therefore, $X(t) = 0$ is kept.

[5] If a request was issued before time $t = 0$, by Fact 2, that request has been served by the output arbiter by time $t = N - 2$, so only requests issued at time $t = 0$ or later can be considered.

□

Lemma 2: After N time slots of the initial condition, each input port sends a cell from a different VOQ at each time slot in an N-slot cycle.

Proof of Lemma 2:

We can re-phrase Lemma 2: *$VOQ_{i,j}$ has issued one and only one request in an N-slot time cycle after N time slots of the initial condition (such that a different $VOQ_{i,j}$ sends a cell every time slot).*

We consider the following facts from CIXB-1:

Fact 3 An output port forwards at most one cell (or grant a single request) each time slot.

Fact 4 At input i, any available crosspoint $XP_{i,j}$, where $0 \leq j \leq N - 1$, receives a request within N time slots.

Definition 6: In input i that uses round-robin arbitration, there are two VOQs, $VOQ_{i,j}$ and $VOQ_{i,j'}$ where

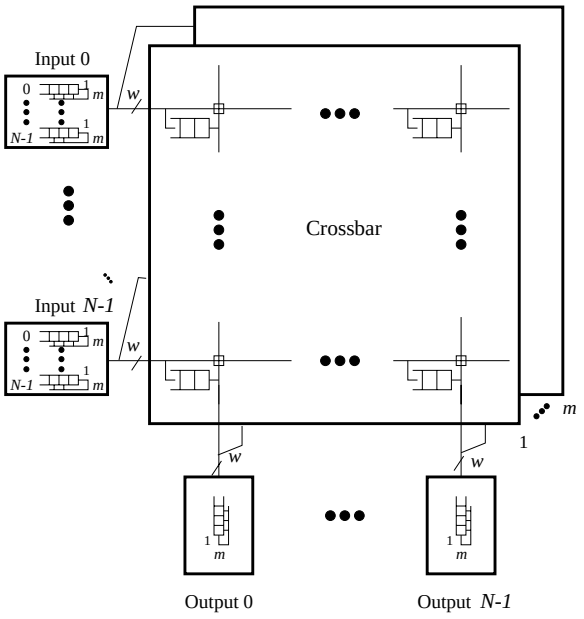


Fig. 10. The crosspoint memory is distributed through all planes

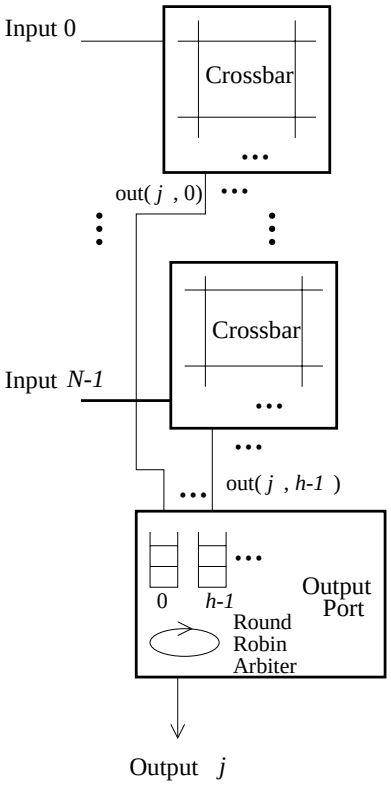


Fig. 11. Example of implementation of buffers used for scalability of CIXB-k.

$j \neq j'$, such that $VOQ_{i,j}$ is called VOQ^c if this VOQ issues two requests within N time slots, and $VOQ_{i,j'}$ is called VOQ^e if this VOQ does not issue a request within N time slots. These terms are assigned to the VOQs at the time slot that any of these conditions becomes true [6].

Definition 7: There are two possible relationships between VOQ^c and VOQ^e determined by the position in which they issue (or do not) a request at time t: $VOQ^c < VOQ^e$ (referred to also as $c < e$) or $VOQ^c > VOQ^e$ (or $c > e$). $c < e$ means that at time $t-1$, VOQ^c is expected to issue a request before VOQ^e, and VOQ^c issues a request at time t. In a similar way, $c > e$ means that at time $t-1$, VOQ^c is expected to issue a request after VOQ^e, and VOQ^e issues a request at time t.

We show some examples of this definition, but first, we explain the symbols used in the figures. These symbols are introduced in Figure 12. The arbitration status of the VOQs and the occupancy of XP of input i are shown in this figure. A VOQ is represented as a square and the VOQs in an input are represented as a column of squares, where each column represents a time slot. The status of a VOQ is as shown in the list of symbols. In the lower part of the figure, we show the notation to describe the XP status in the BX. The matrix of circles represents the status of the XPs in the BX. In this matrix, a row represents an input, a column represents an output. For simplicity, we assume that the VOQs in study belong to input 0 in the BX, without losing generality. However, this input can be any other.

Examples of the relationship between c and e are shown in Figures 13, 14, 15, and 16. Let us look at the example in Figure 13 (1). Let us take the present time as $t = 5$. At this time, VOQ_0 is denoted as VOQ^c and VOQ_2 is denoted as VOQ^e because VOQ_0 has issued two requests and VOQ_2 has issued no request in $3 \leq t \leq 5$. Since two VOQs are already identified VOQ^c and VOQ^e, time $t = 4$ is observed to determine the relationship between c and e. At $t = 4$, after VOQ_1 is granted, VOQ^c (VOQ_0) is expected to be granted after VOQ^e (VOQ_2). However, VOQ^c is granted first. In this way, $VOQ^c > VOQ^e$.

[6] Note that both conditions actually become true at the same time.

Now, with these definitions, we continue with the proof of Lemma 2. We prove this lemma by contradiction.

Let us take the initial time $t = 0$. By Lemma 1, input i has at least one $XP_{i,j}$ available at time N.

Initial assumption. *Assume that $VOQ_{i,j}$ is the first VOQ that has issued more than one request within an N-slot cycle ($N \leq t < 2N$) such that it issues the second request at time $t = 2N-1$ and the first request was issued at time $N \leq t \leq 2N-2$.*

$VOQ_{i,j}$ is denoted as VOQ^c. Therefore, there exists a $VOQ_{i,j'}$ (where $j' \neq j$) that is the first time that a VOQ (or the first VOQ) has not issued any request within this N-slot period. $VOQ_{i,j'}$ is denoted as VOQ^e. Then, there are $N-2$ VOQs that have issued one request during $N \leq t < 2N-1$. If there is a VOQ^c such that it issues a second request at time $t = 2N-1$, there also is a VOQ^e such that either of the two cases is true: (1) $VOQ^c > VOQ^e$ or (2) $VOQ^c < VOQ^e$, where c and e are related according to Definition 7.

We analyze the two possible cases:

(1) $VOQ^c > VOQ^e$. This means that VOQ^e has $XP_{i,j'}(t) = 1$ at time $t = 2N-1$. The other VOQs

have issued requests during times $N \leq t < 2N$ while $XP_{i,j'}$ has been occupied for N time slots. By Fact 2, $XP_{i,j'}$ cannot stay N or more time slots occupied; by Fact 1, $XP_{i,j'}$ should have received a request within time $N \leq t < 2N$. This contradicts the initial assumption that $VOQ_{i,j'}$ has not issued a request during N time slots after N time slots from the initial time. Figure 13 (1), shows an example of $c > e$ for a 3×3 switch. It is shown that $XP_{0,2}$ contradicts the initial assumption by violation of Fact 2. Figure 14 shows two examples of an input in a 4×4 switch, where it can be seen that VOQ_3 and VOQ_0 violate Fact 2 in examples (a) and (b), respectively.

(2) $VOQ^c < VOQ^e$. This means that the crosspoint $XP_{i,j}$ related to VOQ^c has been granted (by the output arbiter) more than once within N time slots, and this can happen only if at least another input has not sent any request competing for output j (or c), in N or more time slots; in other words, at least one crosspoint $XP_{i',j}$, where $i' \neq i$, at output j has not received a request for N or more time slots. By Fact 4, this condition contradicts the initial assumption. Figure 13 (2) shows an example of this case for a 3×3 switch. In the last three time slots ($3 \leq t \leq 5$), the status of the VOQs is presented as the starting point of the situation. Time slots before $t = 3$ are easily deducted by facts 1, 2, 3, and the initial assumption. In this figure, parts (i) and (ii) show all the possible combinations for times slots $1 \leq t \leq 3$ of this example. During time slots 1, 2, and 3, there is a $XP_{i',j} = XP_{i',1}$ that does not receive any requests no matter what happens in any other time slots. The input i' depends on the input granted at time $t = 1$. However, that does not affect this result. In this example, all the possible combinations where at least one crosspoint had violated Fact 4 are presented. Those available crosspoints that have not received a request for a time slot are marked with a square. If an XP is in a square for N time slots or more in a row, that XP violates Fact 4. In this figure, crosspoints $XP_{1,1}$ or $XP_{2,1}$ are the ones that show this violation. Figures 15 and 16 show examples of this case for a 4×4 switch. Note all the combinations of this case are shown in these examples. In any of these examples, there is always a XP related to output j that violates Fact 4, such that the result contradicts the initial assumption.

Since cases (1) and (2) contradict the initial assumption, Lemma 2 is proven. Then, $VOQ_{i,j}$ issues one request and only one within N time slots such that a different VOQ issues a request each time slot after N time slots from the initial time.

□

Since Lemmas 1 and 2 are proved to be valid and each output port that uses round-robin arbitration is work-conserving, Theorem 1 is proved.

□

Backlogged traffic at the VOQs. This assumption is based on the fact that during the initial time, while CIXB-1 does not achieve 100% throughput, accumulation of cells at the VOQs occurs. By Lemma 2, the backlogged traffic remains.

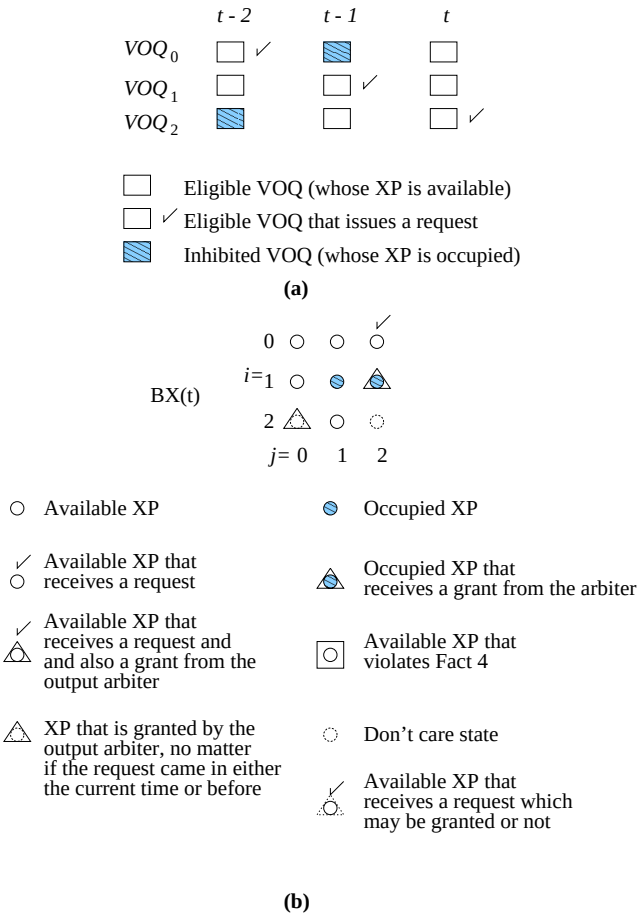


Fig. 12. Nomenclature to be used in examples the proof of Lemma 2.

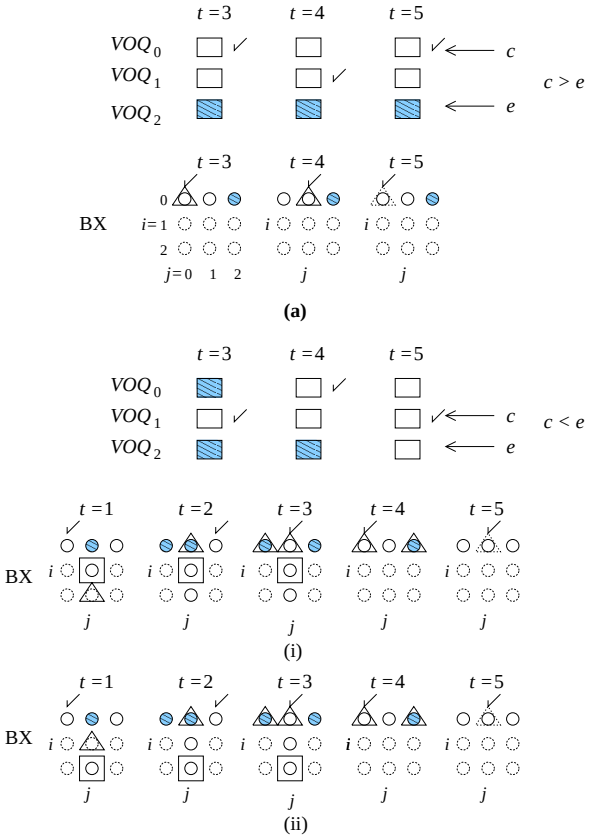


Fig. 13. Example of cases (1) and (2) in proof of Lemma 2 with $N = 3$.

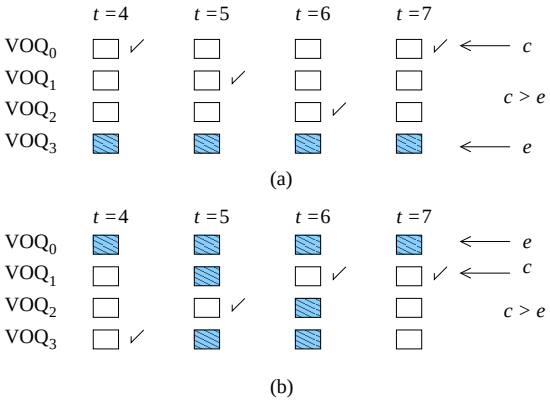


Fig. 14. Examples 1 and 2 of case (1) in proof of Lemma 2 with $N = 4$.

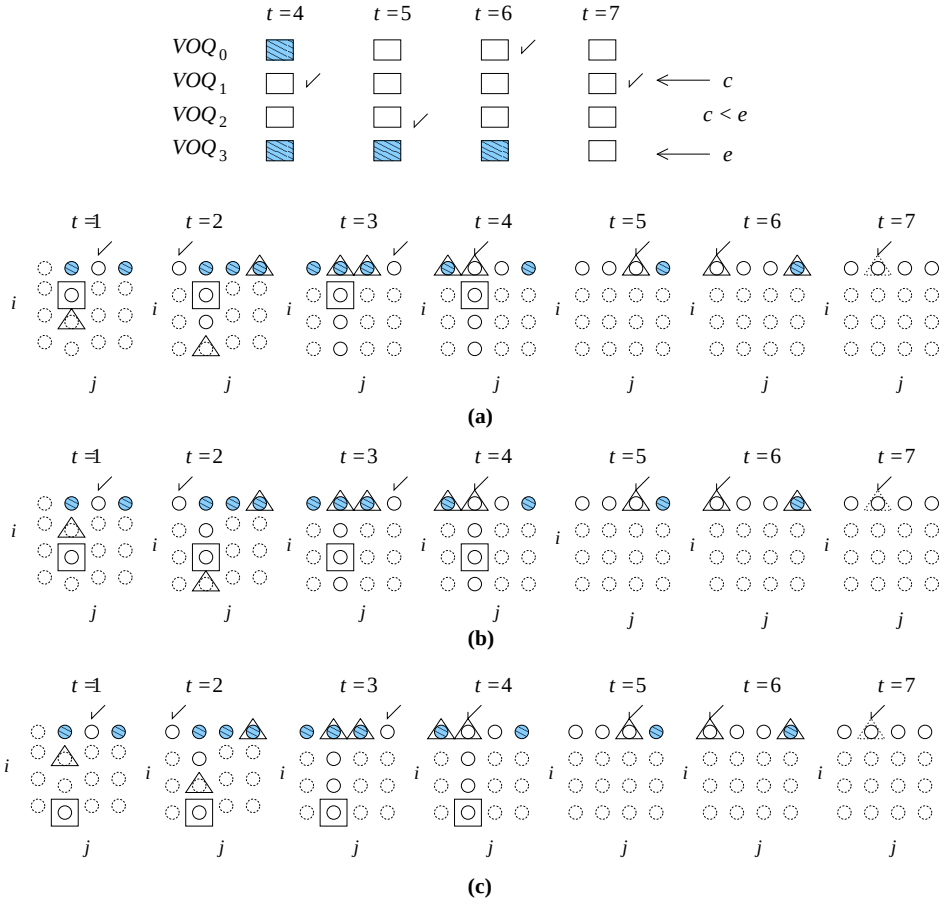


Fig. 15. Example 1 of case (2) in proof of Lemma 2 with $N = 4$.

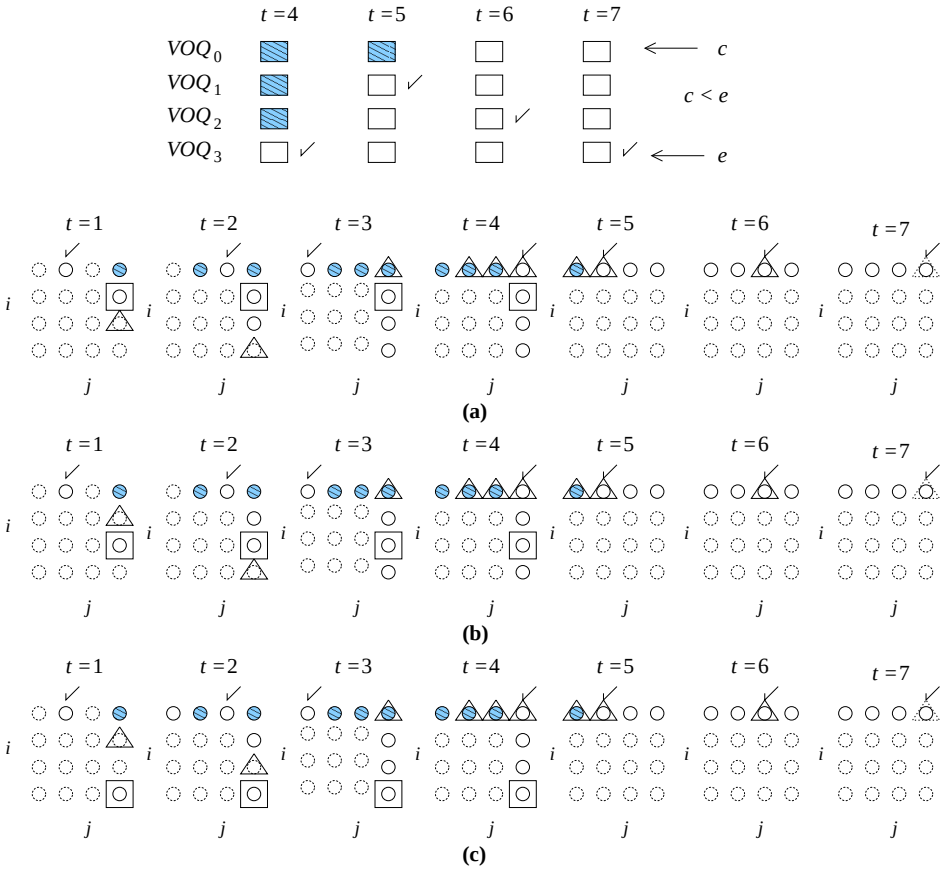


Fig. 16. Example 2 of case (2) in proof of Lemma 2 with $N = 4$.