An Efficient Scheduling Algorithm for CIOQ Switches with Space-Division Multiplexing Expansion

Mei Yang and S.Q. Zheng
Department of Computer Science
Univ. of Texas at Dallas
Richardson, TX 75083-0688, USA
{meiyang,sizheng}@utdallas.edu

***Abstract*— Recently, CIOQ switches have attracted interest from both academic and industrial communities due to their ability of achieving 100% throughput and perfectly emulating OQ switch performance with a small speedup factor S. To achieve a speedup factor S, a conventional CIOQ switch requires the switch matrix and the memory to operate S times faster than the line rate. In this paper, we propose to use a CIOQ switch with space-division multiplexing expansion and grouped inputs/outputs (SDMG CIOQ switch for short) to achieve speedup while only requiring the switch matrix and the memory to operate at the line rate. The cell scheduling problem for the SDMG CIOQ switch is abstracted as a maximum bipartite k-matching problem. Using fluid model, we prove that any maximal size k-matching algorithm on an SDMG CIOQ switch with an expansion factor 2 can achieve 100% throughput assuming input arrivals satisfy the strong law of large numbers and no inputs/outpus are oversubscribed. We further propose an efficient and starvation-free maximal size k-matching scheduling algorithm, kFRR, for the SDMG CIOQ switch. Simulation results show that kFRR achieves 100% throughput with an expansion factor 2 under two SLLN traffic models, uniform traffic and polarized traffic, confirming our analysis.**

I. Introduction

Due to their ability of achieving 100% throughput and even emulating output queueing (OQ) switch performance with a small speedup factor, combined input and output queuing (CIOQ) switches attract attentions from both academic and industrial communities. An $N \times N$ CIOQ switch is shown in Figure 1. To remove head-of-line (HOL) blocking [1], each input maintains N virtual output queues (VOQs) with $VOQ_{i,j}$ buffering packets from input i destined for output j. With an internal speedup larger than 1, packets need to be buffered at outputs as well.

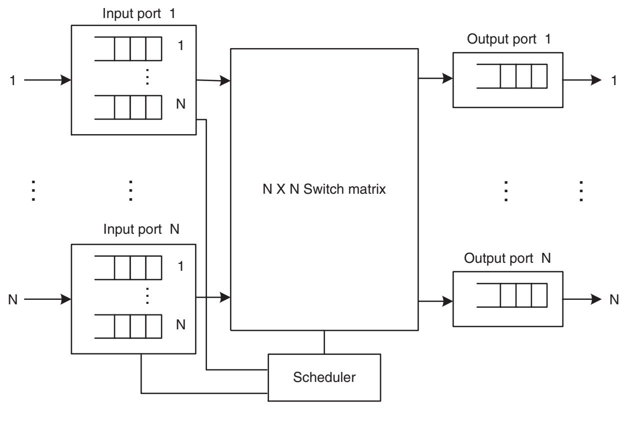

Fig. 1. A CIOQ switch.

In this paper, we assume that CIOQ switches are cell based. In such a switch, variable-length packets are segmented into fixed-size cells upon arrival, transferred through the switch matrix, and then reassembled into packets before they depart. Time is divided into cell slots and one cell slot equals to the transmission time of a cell. In each cell slot, the scheduling algorithm selects a matching between inputs and outputs such that no input (resp. output) may be matched to more than one output (resp. input). Fixed-size cells and slotted time switching make it easier for the scheduler to configure the switch matrix for high throughput [2].

The cell scheduling problem on VOQ-based switches can be modelled as a maximum bipartite matching problem [2]. The most efficient maximum size matching algorithm has a time complexity of $O(N^{2.5})$ [3], [4]. However, maximum size matching algorithms are not practical due to their high time complexity and unfairness [5]. Although maximum weight matching algorithms are proved to achieve 100% throughput for all admissible i.i.d. arrivals [5], they are too complex for high speed implementation due to their high time complexity ($O(N^3 \log N)$ [4]). Most practical scheduling algorithms proposed, such as PIM [6], iSLIP [2], DRR [7], FIRM [8], static round-robin (SRR) [9], and iterative ping-pong arbitration scheme [10], etc., are iterative algorithms that find a maximal size matching to approximate a maximum size matching.

A switch with a speedup of S can remove up to S cells from each input and deliver up to S cells to each output within a cell slot. S is defined as the speedup factor. Hence, an input queueing (IQ) switch has a speedup of 1, an output queueing (OQ) switch has a speedup of N, and a CIOQ switch has a speedup between 1 and N. It has been shown that a CIOQ switch with a speedup of 4 or 2 can exactly emulate an OQ switch by employing some specially designed scheduling algorithms, such as MUCFA algorithm [11], CCF algorithm [12], and JPM algorithm [13]. These results have significant implications: regardless of the switch size, a small constant speedup is sufficient to implement a CIOQ switch with behavior identical to an OQ switch which has a speedup factor proportional to the switch size. Unfortunately, these scheduling algorithms are highly impractical due to their high time complexity. Iterative in nature, these scheduling algorithms solve a stable marriage (matching) problem [14]. To find a stable matching [14], these scheduling algorithms require $O(N^2)$ iterations in the worst case, and N iterations for special cases (such as the acyclic case of [12]).

In [15], Dai and Prabhakar proved that employing any maximal size matching algorithm a CIOQ switch with $S = 2$

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

To achieve speedup for CIOQ switches, in the conventional scheme, it requires the switch matrix and the memory to run S times faster than the line rate. Under current technology, the switch matrix can support up to 2.5Gbps line rate [16]. On the other hand, advances in fiber-optic transmission technologies have greatly pushed the increasing of optical transmission rate. Each individual channel now can operate at OC-192 (10Gbps) or even OC-768 (40Gbps). Although silicon technologies have advanced rapidly, the gap between the data rate that optical transmission technology can deliver and the switching speed that electronic switch matrix can provide is becoming wider and wider [17]. Thus it may not always be feasible to run the switch matrix much faster than the line rate. Memories with sufficient access rate are simply not available for high line rate due to the limitation of current VLSI technology. Even with fast switch matrix and memories, it may not always be possible to run the cell scheduling algorithm fast enough to realize switch speedup greater than 1.

To relax the stringent timing requirement of the switch matrix and the memory operation time, we propose an efficient scheduling algorithm based on an alternative CIOQ switch architecture to achieve the same performance as a CIOQ switch with speedup but only require the switch matrix and the memory operate at the line rate. We first present a CIOQ switch with space-division multiplexing expansion and grouped input/output ports. For easy reference, we refer to such a switch as an SDMG CIOQ switch. In an SDMG CIOQ switch, the number of connections between each input/output (port) and the switch matrix is increased, while the switch matrix only needs to run as fast as the line rate. To relax the memory access rate, the interface between each VOQ (or output queue) and the switch matrix is expanded to multiple copies to allow more than one cell to be transferred from a VOQ (or into an output queue). The expansion factor of an SDMG CIOQ switch is defined as the ratio of the number of connections between an input/output port and the switch matrix and the number of lines associated to an input/output port.

We then focus our study on efficient scheduling algorithms for the SDMG CIOQ switch. We model the cell scheduling problem on the SDMG CIOQ switch as a maximum bipartite k-matching problem. Using fluid model, we prove that any maximal size k-matching algorithm for an SDMG CIOQ switch with expansion factor 2 can achieve 100% throughput assuming input arrivals satisfy SLLN and no inputs/outpus are oversubscribed. We further develop a k-connection FIRM-based round-robin (kFRR) algorithm to find maximal size k-matchings on SDMG CIOQ switches. Through simulations, we show that the kFRR algorithm achieves 100% throughput under two SLLN traffic models: uniform traffic (both Bernoulli arrivals and bursty arrivals) and polarized traffic. This confirms our analysis based on fluid model. And the performance of kFRR with expansion factor 2 is generally better than the performance of FIRM with speedup factor 2. We further show that the kFRR algorithm can be easily implemented in hardware using programmable k-selectors [18].

The remainder of this paper is organized as follows. Section II presents the SDMG CIOQ switch architecture, describes the graph model of the cell scheduling problem for the SDMG CIOQ switch and gives an analysis of the expansion factor that is sufficient for an SDMG CIOQ switch employing any maximal size k-matching scheduling algorithm to achieve 100% throughput assuming input arrivals satisfy SLLN and no inputs/outputs are oversubscribed. In section III, we propose the kFRR scheduling algorithm and discuss its properties. Section IV presents simulation results of kFRR. In Section V, we discuss a hardware implementation scheme of kFRR. Section VI summaries the paper.

II. SDMG CIOQ Switches

In this paper, we assume that the SDMG CIOQ switch we discuss is cell based. In such a switch, variable-length packets are segmented into fixed-size cells as they arrive and reassembled into packets before they depart. Time is divided into cell slots and one cell slot equals to the transmission time of a cell. In addition, we assume that cells arrive at the switch at the beginning of a cell slot and cells depart from the switch prior to the end of a cell slot.

A. Switch Architecture

In the CIOQ switch shown in Figure 1, one input (resp. output) line connects to one input (resp. output) port, and there is one connection between an input (resp. output) port and the switch matrix. To achieve a speedup of S for a CIOQ switch, conventionally it requires the switch matrix and the memory of the CIOQ switch to run S times faster than the line rate.

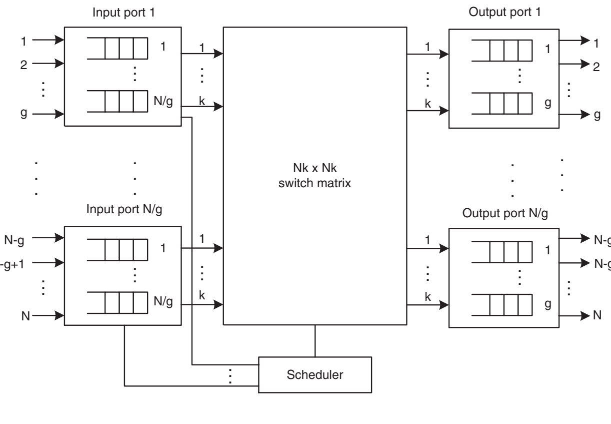

Fig. 2. An SDMG CIOQ switch.

To achieve the speedup required for a CIOQ switch, we consider an alternative CIOQ switch architecture with more

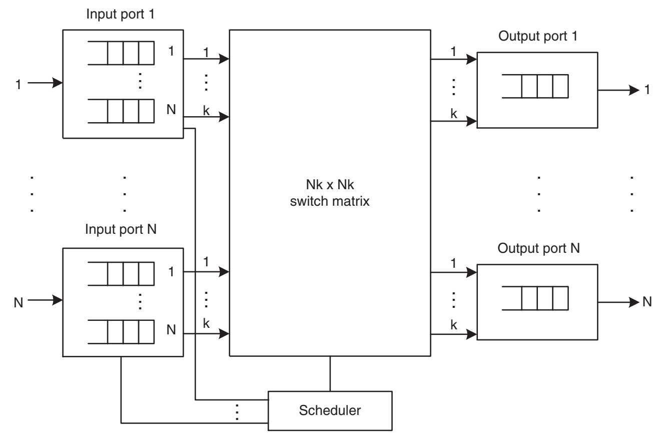

Fig. 3. An SDMG CIOQ switch with $g = 1$.

connections between each input/output port and the switch matrix. We generalize this CIOQ switch architecture by grouping multiple lines into one port. The purpose of introducing grouped input/output ports is to achieve better buffer utilization [19], improve scheduling performance [20], and balance switch I/O loads. We name such a CIOQ switch as a *CIOQ switch with space-division multiplexing expansion and grouped input/output ports* (SDMG CIOQ switch for short). Figure 2 shows an $N \times N$ SDMG CIOQ switch, where N is the number of input/output lines. Figure 3 shows a special SDMG CIOQ with $g = 1$. The characteristics of the SDMG CIOQ switch are listed as follows.

- It has N/g grouped input ports and N/g grouped output ports; each grouped port is associated with g lines. g is called the *group factor*.
- Each input port maintains N/g VOQs, denoted as $VOQ_{i,j}$, where i is the input port number, j is the output port number, $1 \leq i, j \leq N/g$.
- Each output port maintains g output queues, each associated with an output line.
- It has an $Nk/g \times Nk/g$ switch matrix with k connections to each port. We assume that the switch matrix is non-blocking or rearrangeable non-blocking. k is called the *port connection factor*.
- Cells belonging to each VOQ of an input port will be transferred through the switch matrix in order.
- A cell in an input port can be switched to its destination output port through any of the k connections to the switch matrix and any of the k connections between the switch matrix and the destination output port.

We define $P = k/g$ as the *expansion factor* of an SDMG CIOQ switch. To relax the memory access rate, the interface between each VOQ (or output queue) and the switch matrix is expanded to multiple copies to allow more than one cell to be transferred from a VOQ (or into an output queue). Clearly, additional interconnection schemes are needed inside input and output ports to implement parallel memory access. This issue is not going to be addressed in this paper. In the rest of this paper, we will use input (resp. output) and input (resp. output) port interchangeably.

We would like to point out that Obara et al. proposed a similar switch architecture to enhance the scheduling performance for an ATM switch [20]. Our purpose of using the SDMG CIOQ switch architecture is to achieve speedup but only require the switch matrix and the memory to operate as fast as the line rate.

B. Graph Model for The Cell Scheduling Problem

For an SDMG CIOQ switch, in each cell slot, the scheduling algorithm needs to determine a conflict-free switch matrix setting for switching cells from input ports to output ports. The cell scheduling problem on the SDMG CIOQ switch can be modelled as a maximum k-matching problem on bipartite graph $G = (V, E)$, where $V = V_1 \cup V_2$, $V_1 = \{\text{input ports}\}$, $V_2 = \{\text{output ports}\}$, $| V_1 |=| V_2 |= N/g$, $E = \{$connection requests from input ports to output ports$\}$, and let $M =| E |$.

Note that G may not be a simple graph since there may be more than one edge between one pair of nodes. A *k-matching* is a subset of edges $\mathcal{K} \subseteq E$ such that no node of G is incident with more than k edges in $\mathcal{K}$, where $k \geq 1$. A match is an edge $(i, j) \in \mathcal{K}$. A matching is a special case of k-matching with $k = 1$. A *maximum size k-matching* is one with the maximum number of edges, while a *maximal size k-matching* is one that is not contained in any other k-matchings. A *perfect k-matching* $\mathcal{K}$ is one that each node of G is incident with k edges in $\mathcal{K}$. Figure 4 compares a maximum size 2-matching and a maximal size 2-matching for a 4×4 SDMG CIOQ switch with $g = 1$ and $k = 2$. With the maximum size 2-matching shown in Figure 4(b), $VOQ_{1,1}$, $VOQ_{1,3}$, $VOQ_{2,2}$, $VOQ_{2,4}$ and $VOQ_{3,2}$ will be served.

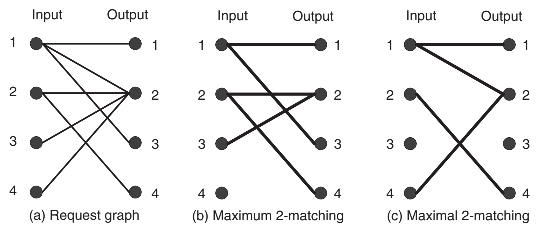

Fig. 4. A maximum and maximal size 2-matching of a 4×4 SDMG CIOQ switch.

As a special case of the bipartite b-matching problem [21], the maximum bipartite k-matching problem can be transformed to a maximum-flow problem in $O(M)$ time. Since the transformed flow network is a unit network [4], we can use Dinic's algorithm to solve the corresponding maximum-flow problem in $O(\sqrt{N}M)$ time [4]. However, this algorithm is too complex to be implemented at high speed. Another noticeable problem with maximum size k-matching algorithm is that it may cause unfairness. For example, in Figure 4, if $VOQ_{1,1}$, $VOQ_{1,3}$, $VOQ_{2,2}$, $VOQ_{2,4}$ and $VOQ_{3,2}$ continue having requests and other VOQs continue having no requests in successive cell slots, then $VOQ_{1,2}$ may get starved since edge $(1, 2)$ does not belong to any maximum size 2-matchings.

For practical use, we desire scheduling algorithms to be fast, starvation-free, easy to implement and of high throughput [2]. A maximal size k-matching algorithm is a better option than a maximum size k-matching algorithm since it is easier to implement and possible to avoid unfairness. In the next section, we will propose a practical and starvation-free maximal size k-matching algorithm, kFRR, for SDMG CIOQ switches.

C. Analysis of Maximal k-Matching Algorithms

An interesting question is what expansion factor, P, is sufficient for an SDMG CIOQ switch employing a maximal size k-matching algorithm to achieve 100% throughput assuming input arrivals satisfy SLLN and no inputs or outputs are oversubscribed?

Let an $N/g \times N/g$ matrix $Z(n)$ be the request matrix at cell slot n, where $Z_{i,j}(n)$ denotes the number of cells in $VOQ_{i,j}$ at the beginning of cell slot n. A maximal size k-matching algorithm determines a matrix $\pi(n)$ in cell slot n, where $\pi(n)_{i,j}$ indicating how many cells can be transferred from input i to output j during cell slot n. We have the following equations.

$$\forall i, j, \pi(n)_{i,j} \leq k,$$

$$\forall i, \sum_{j'=1}^{N/g} \pi(n)_{i,j'} \leq k,$$

$$\forall j, \sum_{i'=1}^{N/g} \pi(n)_{i',j} \leq k,$$

$$\sum_{j'=1}^{N/g} \pi(n)_{i,j'} + \sum_{i'=1}^{N/g} \pi(n)_{i',j} \geq k, \text{ if } Z_{i,j}(n) \geq k. \tag{1}$$

Equation (1) comes from the fact that for a maximal size k-matching algorithm, if input i has at least k cells destined for output j in cell slot n, then at least one of the following holds: (i) input i has k matches to some outputs, (ii) output j has k matches to some inputs.

Consider the fluid model of the SDMG CIOQ switch shown in Figure 2 with port connection factor k, operating under a maximal size k-matching algorithm. We follow all the definitions of fluid model and SLLN used in [15]. We define $A_{i,j}(n)$ as the number of cells that have arrived at $VOQ_{i,j}$ up to cell slot n. We assume that the arrival processes $\{A_{i,j}(\cdot), i, j = 1, ..., N/g\}$ satisfy a strong law of large numbers (SLLN) with probability of one,

$$\lim_{n\to\infty} \frac{A_{i,j}(n)}{n} = \lambda_{i,j}, i, j = 1, ..., N/g, \tag{2}$$

where $\lambda_{i,j}$ is called the arrival rate at $VOQ_{i,j}$. We also assume that no inputs or outputs are oversubscribed, i.e.,

$$\forall i, j, \sum_{j'=1}^{N/g} \lambda_{i,j'} \leq g, \sum_{i'=1}^{N/g} \lambda_{i',j} \leq g. \tag{3}$$

Let (D, T, Z) be a fluid model solution with $Z(0) = 0$. Let $L_i(t) = \sum_{j'} Z_{i,j'}(t)$ denote the total amount of fluid queued at input i at time t and $M_j(t) = \sum_{i'} Z_{i',j}(t)$ be the total amount of fluid destined for output j and queued at some inputs at time t. Define $C_{i,j}(t) = L_i(t) + M_j(t)$. In addition to the fluid model equations (5)-(7) in [15], we have the following lemma.

Lemma 1: For an SDMG CIOQ switch with expansion factor $P = k/g$ operating under a maximal size k-matching algorithm, each fluid limit must satisfy the following equation:

$$\dot{C}_{i,j}(t) \leq \sum_{j'=1}^{N/g} \lambda_{i,j'} + \sum_{i'=1}^{N/g} \lambda_{i',j} - k, \text{ whenever } Z_{i,j}(t) > 0, \tag{4}$$

where $\dot{C}_{i,j}(t)$ represents the differential of $C_{i,j}(t)$.

Proof: Proving Equation (4) is equivalent to showing that, if $Z_{i,j}(n) \geq k$, then

$$C_{i,j}(n+1) - C_{i,j}(n) \leq \sum_{j'=1}^{N/g} (A_{i,j'}(n+1) - A_{i,j'}(n)) + \sum_{i'=1}^{N/g} (A_{i',j}(n+1) - A_{i',j}(n)) - k. \tag{5}$$

Let $V_{i,j}$ denote the set of all VOQs holding cells at input i or destined for output j. Then $C_{i,j}(n+1) - C_{i,j}(n)$ is the difference of the number of arrivals to $V_{i,j}$ at cell slot $n+1$ and the number of departures from $V_{i,j}$ at cell slot n. The number of arrivals to $V_{i,j}$ at cell slot $n+1$ equals to

$$\sum_{j'=1}^{N/g} (A_{i,j'}(n+1) - A_{i,j'}(n)) + \sum_{i'=1}^{N/g} (A_{i',j}(n+1) - A_{i',j}(n)).$$

Since $Z_{i,j}(n) \geq k$ and the switch employs a maximal size k-matching algorithm, from Equation (1), we have the following equation:

$$\sum_{j'=1}^{N/g} \pi(n)_{i,j'} + \sum_{i'=1}^{N/g} \pi(n)_{i',j} \geq k.$$

That is to say, at least k cells are removed from those VOQ's that are in the set $V_{i,j}$. Thus, we get the bound on the right side of Equation (5). ■

Then we have the following theorem.

Theorem 1: For an SDMG CIOQ switch shown in Figure 2, any maximal size k-matching algorithm with $k = 2g$, i.e., $P = k/g = 2$, can achieve 100% throughput assuming input arrivals satisfy SLLN and no inputs or outputs are oversubscribed.

Proof: From Lemma 1 and Equation (3),

$$\dot{C}_{i,j}(t) \leq \sum_{j'=1}^{N/g} \lambda_{i,j'} + \sum_{i'=1}^{N/g} \lambda_{i',j} - k \leq g + g - 2g = 0.$$

Refer to the proof of Theorem 2 of [15] for the rest of the proof. We omit the details in this paper. ■

III. The kFRR Scheduling Algorithm

Iterative maximal size matching scheduling algorithms proposed for input-buffered switches include PIM [6], RRM, iSLIP [2], DRR [7], FIRM [8], static round-robin (SRR) [9], and iterative ping-pong arbitration scheme [10], etc. Among these algorithms, round-robin based algorithms, such as iSLIP, are more attractive than others because of their fairness and implementation simplicity. FIRM, which uses round-robin arbitration scheme, improves iSLIP by reducing the service guarantee time from $(N-1)^2 + N^2$ cell slots to N^2 cell slots. It is starvation-free and easy to implement at high speed [8]. In the following, we generalize the idea of FIRM for the SDMG CIOQ switch and present the k-connection FIRM-based round-robin (kFRR) scheduling algorithm. As FIRM, kFRR is an iterative algorithm. It also uses round-robin arbitration scheme to schedule active inputs and outputs.

We use I_i and O_j, $1 \leq i, j \leq N/g$, to denote the inputs and outputs respectively. For I_i, let a_i be its accept pointer indicating the accept starting position in the circular round-robin priority queue, where $1 \leq a_i \leq N/g$, and $C(I_i)$ denote the number of available connections at I_i. For output O_j, let g_j be its grant pointer indicating the grant starting position in the circular round-robin priority queue, and $C(O_j)$ be the number of available connections at O_j. Prior to the first iteration of kFRR in any cell slot, we set $C(I_i) = C(O_j) = k$, $1 \leq i, j \leq N/g$.

In each cell slot, kFRR iteratively finds a k-matching. It terminates after a fixed number of iterations or until a maximal size k-matching is found. Each iteration of kFRR consists of the following three steps.

Step 1: **Request**. $\forall I_i, 1 \leq i \leq N/g$, if I_i has available connections and unresolved requests (requests to outputs with available connections), it sends all unresolved requests to their corresponding O_j's.

Step 2: **Grant**. $\forall O_j, 1 \leq j \leq N/g$, if O_j has available connections and receives requests from any inputs, it grants $\min\{C(O_j)$, number of requests to $O_j\}$ requests, starting from g_j. These grants are sent to their corresponding I_i's. g_j is updated to the first input that receives O_j's grant but does not accept it in the Accept phase or the first input that does not receive O_j's grant if all O_j's grants are accepted in the first iteration, starting from g_j in a circular manner if and only if in the *the first iteration*. $C(O_j)$ is updated to the number of available connections at O_j.

Step 1: **Accept**. $\forall I_i, 1 \leq i \leq N/g$, if I_i has available connections and receives any grants, it accepts $\min\{C(I_i)$, number of grants to $I_i\}$ grants starting from a_i. a_i is updated to the next position to the last output whose grant is accepted by I_i in a circular manner. $C(I_i)$ is updated to the number of available connections at I_i.

Figure 5 shows how kFRR adapts to a time-division multiplexing for a 4×4 SDMG CIOQ switch with $k = 2$ under saturated load. Saturated load means at some cell slot, $\forall 1 \leq i, j \leq 4$, $VOQ_{i,j} > 0$, and input arrivals are maintained in such a manner that $VOQ_{i,j} > 0$ in the following cell slots. At the start of cell slot 0, assume $\forall 1 \leq j \leq 4, g_j = 1, \forall 1 \leq i \leq 4$, $a_i = 1$. Then after the scheduling, the grant and accept pointers are updated as $g_1 = 3, g_2 = 3, g_3 = 1, g_4 = 1$, and $a_1 = 3, a_2 = 3, a_3 = 1, a_4 = 1$. Due to the desynchronization of grant pointers, a perfect 2-matching is achieved at cell slot 1 and thereafter.

kFRR has the following properties.

Property 1: At each output, due to the property of round-robin, the lowest priority element is set as the input before the first input that receives its grant but does not accept it in the first iteration or the input before the first input that does not receive O_j's grant if all O_j's grants are accepted in the first iteration.

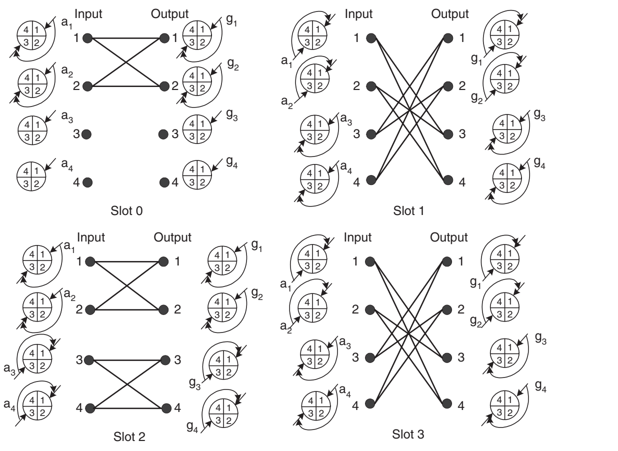

Fig. 5. Illustration of desynchronization effect of grant pointers of kFRR for a 4×4 SDMG CIOQ switch under saturated load.

Property 2: Under saturated load, all VOQs with a common output have the same throughput. The grant pointer moves to each requesting input in a fixed order (every $\frac{N}{kg}$ cell slots), thus providing each with the same throughput.

Property 3: No connection is starved. This property comes from the following theorem.

Theorem 2: kFRR serves an existing connection request within no more than $(\frac{N}{gk})^2$ cell slots.

Proof: The worse case service scenario of kFRR is the situation where a request from I_i to O_j has to wait all other $N/g - k$ inputs to be served by O_j, i.e., for some n, $Z_{m,j}(n) > 0$ for all I_m's and $g_j = ((i+1) \bmod N/g)$, where $m \neq i$. The delay between posting a request and serving the request consists of the delay for the request to be granted and the delay for the grant to be accepted. The delay for the request from I_i to O_j to be granted is $(\frac{N}{gk} - 1)\frac{N}{gk}$ since it takes $\frac{N}{gk} - 1$ cell slots for O_j to grant requests from other $N/g - k$ inputs and it takes at most $\frac{N}{gk}$ cell slots for each grant to be accepted. After the grant to I_i is issued, it also takes $\frac{N}{gk}$ cell slots to get it accepted. Thus totally it takes $(\frac{N}{gk} - 1)\frac{N}{gk} + \frac{N}{gk} = (\frac{N}{gk})^2$ cell slots to serve an existing connection request. ■

Property 4: kFRR finds a maximal size k-matching in at most $N/g - k + 1$ iterations, i.e. kFRR converges in at most $N/g - k + 1$ iterations. The reason is as follows. The size of a maximal size k-matching is at most Nk/g. If finding a maximal k-matching takes more than 1 iteration, the first iteration finds at least k^2 matches, the last iteration finds at least 1 match, and other iterations find at least k matches. Thus, the total number of iterations needed is at most $\lfloor \frac{Nk/g - k^2 - 1}{k} \rfloor + 2$, which is given by $N/g - k + 1$.

Figure 6 shows an example of how many iterations needed for kFRR to converge for an 8×8 SDMG CIOQ switch with $k = 2$ under saturated load. In cell slot 0, kFRR takes 4 iterations to converge. It takes 3 and 2 iterations for kFRR to converge in cell slot 1 and 2 respectively. After cell slot 3, all grant pointers have become totally desynchronized and kFRR converges in a single iteration.

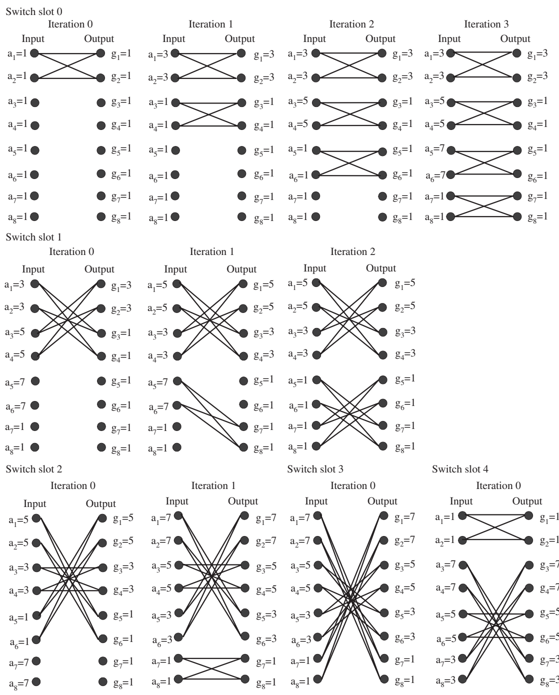

Fig. 6. Example of the number of iterations for kFRR to converge for an 8×8 SDMG CIOQ switch under saturated load.

IV. Performance Evaluation

In this section, we evaluate the performance of kFRR on SDMG CIOQ switches in terms of average transit time, measured by number of cell slots. The transit time is defined as the cell's waiting time in VOQs at input ports plus the transmission time through the switch matrix.

A. Traffic Models

Two traffic models are used in our simulations: uniform traffic and polarized traffic. For uniform traffic, we consider both Bernoulli arrivals and bursty arrivals. Polarized traffic is a non-uniform, locally unbalanced but globally balanced traffic pattern. It is defined as follows [22]. Let $d_{i,j}$ be the proportion of traffic received by $VOQ_{i,j}$. q is defined as the polarization factor with

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

such that,

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

where $q \geq 1.00$. Polarized traffic with $q = 1.00$ is unform traffic. One can verify that both uniform traffic and polarized traffic satisfy SLLN condition and no inputs/outputs are oversubscribed. Simulations have been done for the kFRR algorithm for SDMG CIOQ switch sizes of 4×4, 8×8, 16×16, and 32×32 with different group factors (g), different port connection factors (k), different polarization factors (q) and different number of iterations. Without loss of generality, in our simulations, all pointers in kFRR are initialized randomly.

B. With Bernoulli Arrivals

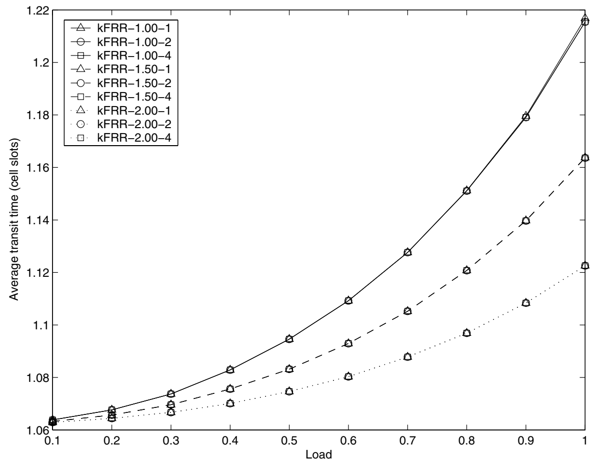

Fig. 7. Average transit time vs. load of kFRR with $g = 1$, $k = 2$ and different number of iterations under Bernoulli arrivals.

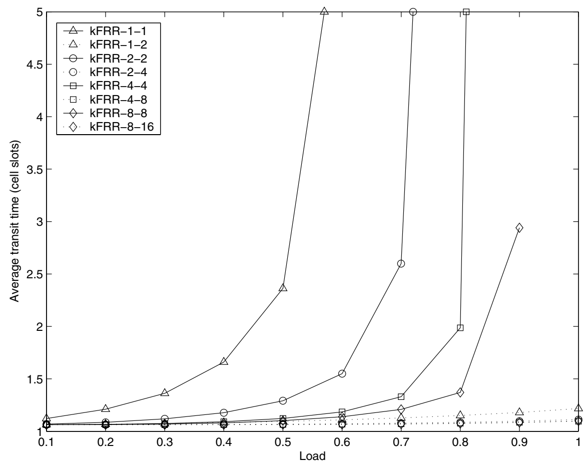

Fig. 8. Average transit time vs. load of one-iteration kFRR with different group factors and different port connection factors under uniform Bernoulli arrivals.

Figure 7 shows the performance of kFRR with 1, 2, and 4 iterations, $g = 1$, $k = 2$ and q = 1.00, 1.50, and 2.00, for a 32×32 SDMG CIOQ switch under Bernoulli arrivals. kFRR achieves 100% throughput with all polarization factors. The performance of kFRR improves when the polarization factor increases. We observe that the difference in the number of iterations does not affect much of the performance of kFRR under Bernoulli arrivals.

Figure 8 compares the performance of one-iteration kFRR with $k = g$ (solid line) and $k = 2g$ (dotted line) for $g = 1, 2,$ and 4, for a 32×32 SDMG CIOQ switch under uniform Bernoulli arrivals. Clearly, kFRR with $k = 2g$ improves the performance of kFRR with $k = g$ dramatically. And the larger the group factor, the better performance kFRR can achieve.

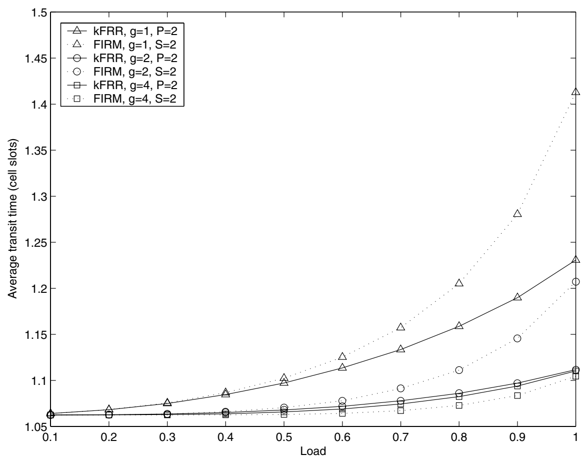

Fig. 9. Average transit time vs. load of one-iteration kFRR with $P = 2$ and one-iteration FIRM with $S = 2$ under uniform Bernoulli arrivals.

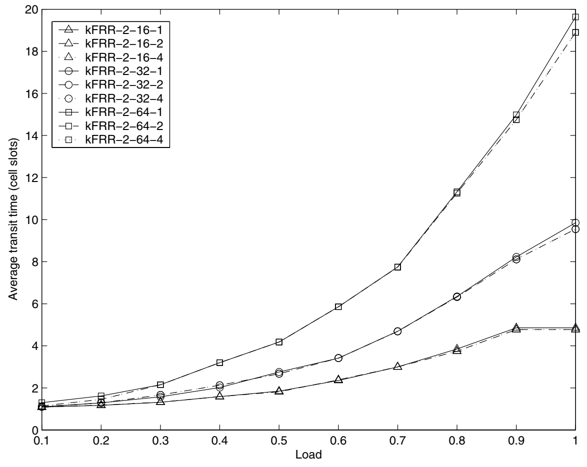

Fig. 10. Average transit time vs. load of kFRR with $k = 2$, $g = 1$ and different number of iterations under bursty arrivals.

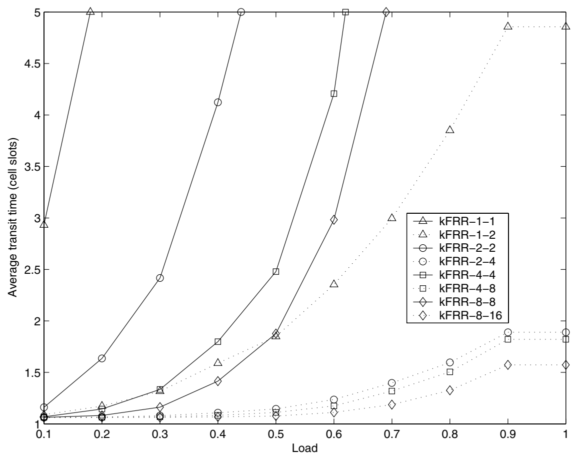

Fig. 11. Average transit time vs. load of one-iteration kFRR with different group factors and different port connection factors under bursty arrivals.

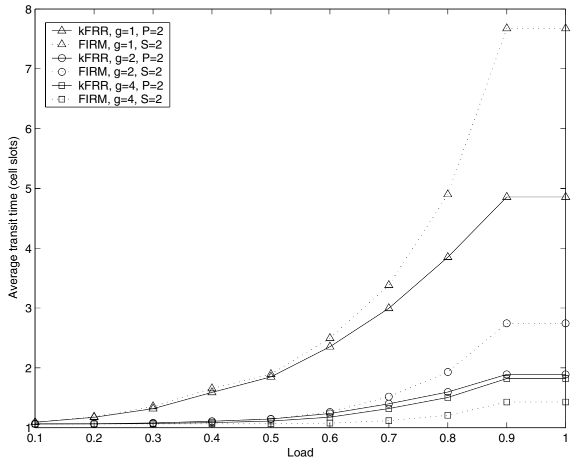

Fig. 12. Average transit time vs. load of one-iteration kFRR with $P = 2$ and one-iteration FIRM with $S = 2$ under bursty arrivals.

Figure 9 compares the performance of one-iteration kFRR with $P = 2$ (solid line) and one-iteration FIRM with $S = 2$ (dotted line) for $g = 1, 2$, and 4, for a 32×32 SDMG CIOQ switch under uniform Bernoulli arrivals. As we can see, under uniform Bernoulli arrivals, the performance of kFRR with expansion factor 2 is better than FIRM with speedup factor 2 when $g = 1$ and 2.

C. With Bursty Arrivals

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

Figure 10 illustrates the performance of kFRR with 1, 2, and 4 iterations, $g = 1$, $k = 2$, for a 32×32 SDMG CIOQ switch under bursty arrivals with $E(B) = 16$, 32, and 64 respectively. kFRR achieves 100% throughput with all average burst length settings. As we can see, the increased number of iterations leads to lower average transit time while the increased average burst length increases the average transit time.

Figure 11 compares the performance of one iteration kFRR with $k = g$ (solid line) and $k = 2g$ (dotted line) for $g = 1, 2$, and 4, for a 32×32 SDMG CIOQ switch under bursty arrivals with $E(B) = 16$. As shown in Figure 11, kFRR with $k = 2g$ improves the performance of kFRR with $k = g$ dramatically. And the performance of kFRR improves with the group factor increasing.

Figure 12 compares the performance of one iteration kFRR with $P = 2$ (solid line) and one-iteration FIRM with $S = 2$ (dotted line) for $g = 1, 2$, and 4, for a 32×32 SDMG CIOQ switch under bursty arrivals with $E(B) = 16$. As we can see, under bursty arrivals, the performance of kFRR with expansion factor 2 is better than the performance of FIRM with speedup factor 2 when $g = 1$ and 2.

V. Hardware Implementation of kFRR Algorithm

An important property of an efficient scheduling algorithm is simple to implement. In this section, we show that kFRR is easy to be implemented in hardware. Figure 13 shows a possible design of a kFRR scheduler for an $N \times N$ SDMG CIOQ switch. It consists of $2N/g$ port arbitration components, a state update logic and a state memory. Each port arbitration component is responsible for making k selections out of Nk/g requests in a round-robin manner. We use a programmable k-selector [18] to construct a port arbitration component. The timing performance of such a design is independent of k and much better than the design using programmable priority encoders [23], [18].

For an $N \times N$ SDMG CIOQ switch, at the start of each cell slot, the scheduler receives an $(N/g \times \log k)$-bit request vector from each input port. Taking the example of one-iteration kFRR scheduler, it works as follows:

Step 1: Each grant arbitration component selects up to k unresolved requests. The grants are sent to N/g accept arbitration components.

Step 2: Each accept arbitration component selects up to k grants and send them to the decision register, and the state memory and update logic, where the grant pointers are updated.

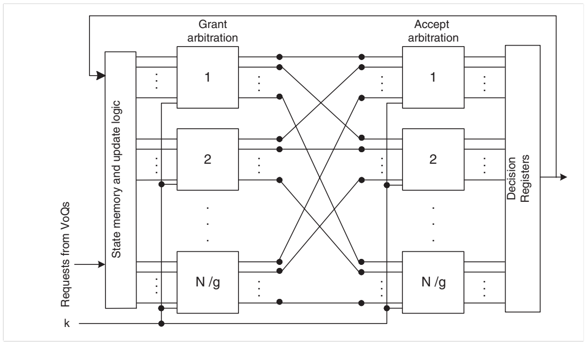

Fig. 13. Block diagram of a one-iteration kFRR scheduler for an $N \times N$ SDMG CIOQ switch.

VI. Summary

The major contributions of this paper include: (1) We introduced the SDMG CIOQ switch, which combines space-division multiplexing expansion and grouped inputs/outputs to improve switching performance. (2) We modelled the cell scheduling problem on the SDMG CIOQ switch as a maximum bipartite k-matching problem. (3) Using fluid model, we proved that any maximal size k-matching algorithm for the SDMG CIOQ switch with an expansion factor 2 can achieve 100% throughput so long as input arrivals satisfy SLLN and no inputs/outputs are oversubscribed. (4) We proposed an efficient and starvation-free distributed scheduling algorithm for the SDMG CIOQ switch, kFRR, for finding maximal size k-matchings. (5) Through simulations, we showed that kFRR with an expansion factor 2 achieves 100% throughput for two SLLN traffic arrivals: uniform traffic and polarized traffic. (6) An efficient hardware implementation scheme of kFRR based on our programmable k-selectors [18] was proposed. In conclusion, the SDMG CIOQ switch is a promising alternative to the CIOQ switch with speedup and kFRR is an efficient scheduling algorithm for the SDMG CIOQ switch.

References

[1] M. J. Karol, M. G. Hluchyj and S. P. Morgan, "Input vs. output queueing on a space-division packet switch", *IEEE Transaction on Communications*, Vol. 35, No. 12, pp. 1347-1356, 1987.
[2] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches", *IEEE/ACM Transactions on Networking*, Vol. 7, No. 2, pp. 188-201, April 1999.
[3] J. E. Hopcroft and R. M. Karp, "An $n^{2.5}$ algorithm for maximum matching in bipartite graphs", *Soc. Ind. Appl. Math. J.*, vol. 2, pp. 225-231, 1973.
[4] R. E. Tarjan, *Data Structures and Network Algorithms*, Bell laboratories, 1983.
[5] N. Mckeown, A. Mekkittikul, V. Anantharam, J. Walrand., "Achieveing 100% throughput in an input-queued switch", *IEEE Transactions on Communications*, Vol. 47, No. 8, pp. 1260-1267, August 1999 .
[6] T. Anderson, S. Owicki, J. Saxie, and C. Thacker, "High speed switch scheduling for local area networks", *ACM Trans. Comput. Syst.*, vol. 11, no. 4, pp. 319-352, Nov. 1993.
[7] J. Chao, "Saturn: a terabit packet switch using dual round-robin", *IEEE Communications Magazine*, Dec. 2000.
[8] D. N. Serpanos and P. I. Antoniadis, "FIRM: A class of distributed scheduling algorithms for high-speed ATM switches with multiple input queues", *Proc. of IEEE Infocom2000*, pp. 548-555, 2000.
[9] Y. Jiang and M. Hamdi, "A fully desynchronized round-robin matching scheduler for a VOQ packet switch architecture", *2001 IEEE workshop on high performance switching and routing*, pp. 407-412, June 2001.
[10] H. J. Chao, C. H. Lam, and X. L. Guo, "A fast arbitration scheme for terabit packet switches", *Globecom'99*, pp. 1236-1243, 1999.
[11] B. Prabhakar, N. Mckeown, "On the speedup required for combined input and output queued switching", *Automatica*, Vol. 35, 1999.
[12] S. T. Chuang, A. Goel, N. Mckeown, B. Prabhakar, "Matching output queueing with a combined input output queued switch", *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 6, pp. 1030-1039, June 1999.
[13] I. Stoica and H. Zhang, "Exact emulation of an output queueing switch by a combined input output queueing switch", *Proc. 6th IEEE/IFIP IWQoS'98*, Napa Valley, CA, pp. 218-224, May 1998.
[14] D. Gale, and L. S. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly*, vol. 69, pp. 9-15, 1962.
[15] J. Dai and B. Prabhakar, "The throughput of data switches with and without speedup", *Proc. of IEEE Infocom2000*, pp. 556-564, May, 2000.
[16] Vitesse, Switch fabric products [Online], available at http://www.vitesse.com/products, 2002.
[17] C. Minkenberg, "On packet switch design", Ph.D. dissertation, Eindhoven University of Technology, 2001.
[18] S. Q. Zheng, M. Yang and F. Masetti, "Hardware switch scheduling for high speed, high capacity IP routers", submitted for publication.
[19] A. Pattavina, "Multichannel bandwidth allocation in broadband packet switch", *IEEE Journal on Selected Aread in Communications*, vol. 6, no. 9, pp. 1489-1499, Dec. 1988.
[20] H. Obara, S. Okamoto and Y. Hamazumi, "Input and output queueing ATM switch architecture with spatial and temporal slot reservation control", *Electronics Letters*, Vol. 28, No. 1, pp. 22-24, Jan. 1992.
[21] W. J. Cook,W. R. Pulleyblank, A. S., and W. H. Cunningham, *Combinatorial Optimization*, Wiley John & Sons Inc., Nov. 1997.
[22] J. Blanton, H. Badt, G. Damm, and P. Golla, "Impact of polarized traffic on scheduling algorithms for high speed optical switches", ITCom2001, Denver, August 2001.
[23] P. Gupta, N. Mckeown, "Designing and implementing a fast crossbar scheduler", *IEEE Microelectronics*, Vol. 19, pp. 20-29, No. 1, 1999.