

# Crosstalk-Preventing Scheduling in AWG-Based Cell Switches

Andrea Bianco

Dipartimento di Elettronica  
Politecnico di Torino, Torino, Italy.  
andrea.bianco@polito.it

David Hay

Dipartimento di Elettronica  
Politecnico di Torino, Torino, Italy.  
hay@tlc.polito.it

Fabio Neri

Dipartimento di Elettronica  
Politecnico di Torino, Torino, Italy.  
fabio.neri@polito.it

**Abstract**—AWG-based optical switching fabrics are affected by coherent crosstalk, that can significantly impair system operation when the same wavelength is used simultaneously on several input ports to forward data to output ports. To permit large port counts in a  $N \times N$  AWG, the scheduling of transmissions across the AWG must therefore prevent switch configurations that generate large crosstalk. We study the properties and the existence conditions of switch configurations able to control coherent crosstalk. Our results show that it is possible to keep an AWG-based switch with large port counts in the feasible operational region without significant performance degradation, provided that a proper scheduling algorithm is used.

## I. INTRODUCTION

Optical packet switching received a lot of attention in the research community because optical technologies promise to overcome intrinsic limitations of current switching architectures [1], [2]. Indeed, the amount of traffic transported by the Internet has been increasing at a pace that is faster than Moore's law, and electronic technologies may not be able to support the realization of large packet switches and IP routers in the near future. Power density and dissipation, in particular, are becoming major bottlenecks [3].

On the other hand, photonic technologies exhibit a number of interesting properties; prime examples are very large data rates, a switching complexity almost independent of the data rate, very large information densities on physical interconnections, no significant constraints on the physical size of the switch and on the length of internal switch interconnections (while electrical backplanes and interconnects have severe distance limitations), and very good scalability of power requirements. Nevertheless, all-optical packet switches are still far from being feasible, due to several limitations such as the lack of optical memories, the very limited data processing capabilities, and the inherent difficulties in realizing functions in the time domain. Therefore, switching architectures in the near future will probably exploit both electronics and photonic technologies [4]: packet processing and storing will likely be realized in electronics, while the packet forwarding from input to output ports will likely rely on an optical switching fabric.

One of the most promising approaches to realize an optical switching fabric is to use a passive wavelength routing device with tunable transmitters and receivers around it. Arrayed Waveguide Gratings (AWGs) [5] have been very successful in the commercial deployment of Wavelength-Division Multiplexing (WDM) transmission systems. AWGs are passive

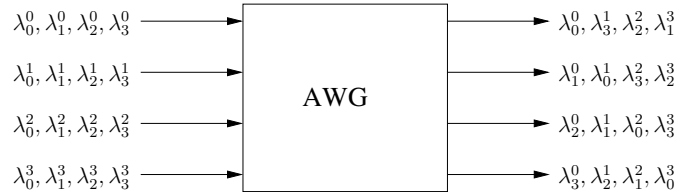


Fig. 1. Considered routing for the AWG device.

devices behaving as multiport interferometers. In the  $N \times N$  configuration, AWGs behave as wavelength routers: The information at an input port is forwarded to an output port that depends on the input wavelength and the input port. More specifically, at each input port, different wavelengths are used to reach different output ports. Overall, an  $N \times N$  AWG can be simultaneously traversed by  $N^2$  information flows (one for each input/output pair, leading to a full mesh connectivity) exploiting space and wavelength separation. The specific wavelengths used to route information through an AWG depend on the design of the device, but are typically on the ITU grid standard bands (with 100 GHz or 50 GHz spacing) for commercial devices.

More precisely, although other wavelength assignments are possible, we assume, with no loss of generality, that i) the  $N \times N$  AWG operates with a set of  $N$  wavelengths  $\Lambda = \{\lambda_0, \lambda_1, \dots, \lambda_{N-1}\}$ , and ii) at input  $i$ , information is delivered to output  $j$  using wavelength  $\lambda_k$ , with  $k = (j - i) \bmod N$  being the wavelength channel numbers. This cyclic behavior is typical of the interferometric nature of the AWG, whose routing behavior is replicated over the wavelength axis with a period called Free Spectral Range (FSR). Our assumptions on the AWG behavior imply that only  $N$  wavelengths are necessary for the  $N^2$  connections over  $N$  input and  $N$  output ports. Fig. 1 depicts the behavior considered in this paper for a  $4 \times 4$  AWG, where superscripts refer to input port indices, and subscripts to wavelength channel numbers.

We consider a very straightforward realization of non-blocking, cross-bar like, optical interconnection among input and output ports, depicted in Fig. 2. Input Queueing (IQ) slotted operation is assumed: fixed-size data units, named cells, reach input ports where they are temporarily buffered waiting for the availability of the output line. Although an asynchronous switch behavior is possible, we disregard this possibility in this paper. To balance electronic and optical

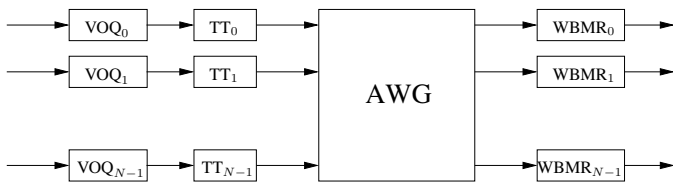


Fig. 2. Simple single-stage AWG-based switch.

complexity, each input port is equipped with a single Tunable Transmitter (TT) and each output port with a single wideband receiver. Thus, at most one cell can be transmitted from each input and to each output in each time slot, using the proper wavelength to reach the chosen output. The adopted switch control (scheduling) algorithm ensures that at most one cell is forwarded to each output at the same time, thus avoiding output contention. In addition, current schedulers typically use Virtual Output Queueing (VOQ) at the inputs to achieve high throughput [6], [7]: Incoming cells at each input port are stored in  $N$  separate FIFO queues according to their destination. Recall that, at each output port, cells are received at a wavelength depending on the transmitting input port. Since at each time slot at most a single cell is received at each output, there is no need for tunable receivers at the outputs, and wideband receivers suffice. Yet, since each output port receives (over time) cells from different inputs, burst-mode operation—hence, Wideband Burst Mode Receivers (WBMR)—is necessary. Note that in this setup, the AWG is largely under-utilized, as only at most  $N$  out of the possible  $N^2$  input/output connections are used at a given time. This under-utilization is a direct consequence of the single transceiver architecture assumed in the paper, a constraint normally introduced to reduce the electronic complexity of input/output line-cards. Different AWG-based optical switching architectures have been studied in the technical literature (e.g. [8]–[10]); although some of these architectures achieve better performance, they require a significantly higher complexity and are not further considered in this paper.

The physical-layer behavior of commercial AWGs is very good, providing a uniform transfer functions, and extinction ratios among adjacent channels in the order of 30–40 dB. These characteristics are largely sufficient for multiplexers and demultiplexers, which are indeed commonly used in commercial WDM systems. However, significant coherent crosstalk figures were reported in  $N \times N$  AWGs with large port counts [11]. In [12], the maximum possible value of  $N$  is shown to be around 16 for currently available devices, if the same wavelength is used at all AWG inputs. Hence, the maximum port count is severely limited. We remark that several proposals in the literature assume very large AWG port counts, even if this turns out to be unfeasible without counter-measuring the above described crosstalk impairments. These large crosstalks can only marginally be reduced by improving the physical layer behavior of the device [13].<sup>1</sup>

<sup>1</sup>Crosstalk prevention was studied also for other architectures, such as Lithium niobate optical switching [14], however these results are not applicable for AWG-based switches.

One possibility to overcome this impairment is to exploit homologous wavelengths in several FSRs (as proposed in some studies, e.g. [15]), but this increases the operational bandwidth of the system (possibly preventing the utilization of optical amplifiers), and the behavior of the device outside the principal FSR often degrades rapidly. The alternative approach pursued in this paper is to prevent coherent crosstalk by controlling AWG-based switches with scheduling decisions that avoid using simultaneously the same wavelengths at too many different inputs.

In the remainder of the paper, the crosstalk-constrained scheduling problem is formulated in Sec. II. Sec. III shows basic properties of the permitted switch configurations. Sec. IV elaborates on the performance obtainable with single-stage AWG switching architectures, while in Sec. V two-stage architectures are considered. Finally, Sec. VI summarizes our findings.

## II. PROBLEM STATEMENT

The considered IQ  $N \times N$  switch handles fixed-size cells that arrive at input ports and leave output ports in a time-slotted manner: All the switch external lines are assumed to be synchronized [16]. Each time slot is comprised of  $S$  *scheduling decisions*, where  $S$  is the switch speedup. At each scheduling decision, at most one cell can be sent from each input port and at most one cell can be sent to each output port. Thus, each scheduling decision is a *permutation* (or a partial-permutation) of port indexes. We denote these permutations by vectors  $\pi = [\pi[0], \pi[1], \dots, \pi[N-1]]$ , where  $\pi[i]$  is the output port index to which input  $i$  forwards a cell. Clearly each output port index can appear at most once in  $\pi$ . If the scheduling decision creates a partial permutation, some entries in  $\pi$  are “don’t care”. We denote by  $I$  the unit permutation:  $I = [0, 1, \dots, N-1]$ .

The traffic to be forwarded by the switch can be described by a traffic matrix  $T = [t_{i,j}]$ , where  $t_{i,j}$  is the number of cells (or, alternatively, the number of cells per time unit, or the number of cells per time frame) that must be forwarded from input  $i$  to output  $j$ . Using a matrix notation, an input/output permutation could also be described by an  $N \times N$  permutation matrix, i.e., a 0-1 matrix  $P = p[i,j]$ , where rows (columns) represents inputs (outputs), and  $p[i,j] = 1$  if and only if input  $i$  is connected to output  $j$ . In a permutation matrix, at most a single “1” is present in each column and in each row.

Since we deal with an AWG passive router, cell forwarding through the switching fabric is done by assigning to each cell a wavelength out of a predetermined set of  $N$  wavelengths  $\Lambda = \{\lambda_0, \dots, \lambda_{N-1}\}$ , according to the following rule:

A cell sent from input port  $i$  with wavelength  $\lambda_k \in \Lambda$  is forwarded to output port  $i + k \pmod{N^2}$ .

Given a permutation  $\pi$ , we call  $\lambda(\pi)$  the *wavelength assignment* of  $\pi$ ; that is, the vector of indices of the wavelengths that are needed at input ports to realize permutation  $\pi$ . Note

<sup>2</sup>In the remainder of the paper the  $\pmod{N}$  operator, denoting the remainder of the division by  $N$ , may be omitted to improve readability.

that, with our wavelength assignment rule, the wavelength used to reach output  $j$  from input  $i$  is  $\lambda_{(j-i) \bmod N}$ . Hence,  $\lambda(\pi) = (\pi - I) \bmod N$ , when  $I$  is the identity permutation. As mentioned in Sec. I, other wavelength assignments are possible, depending on the design of the AWG device. These different wavelength behaviors can in some cases (for example if output  $(i - k) \bmod N$  is reached from input  $i$  using  $\lambda_k$ ) be modeled by relabeling wavelengths and ports in our formalization, so that the properties outlined in the sequel hold for several AWG wavelength assignments.

Recall that performance degradation due to the coherent crosstalk arises when several different input ports in an AWG-based switch use simultaneously the same wavelength to send cells to different output ports. The impairments due to coherent crosstalk increase as the number of input ports using the same wavelength increases, up to the point in which the switch operation becomes impossible. We focus on avoiding such effects by restricting the switch scheduler to use only a certain type of permutations:

*Definition 1:* A permutation  $\pi$  is  $k$ -legal if, in the vector  $\lambda(\pi) = \pi - I$ , no index appears more than  $k$  times.

$k$  represents the maximum number of times the same wavelength is used at different input ports in a given scheduling decision. Our goal is to build a switch which can handle the incoming traffic using only  $k$ -legal permutations, with the smallest possible value of  $k$ , to minimize crosstalk.

### III. PROPERTIES OF $k$ -LEGAL PERMUTATIONS

We start by investigating the properties of  $k$ -legal permutations. Definition 1 immediately implies that a permutation  $\pi$  is 1-legal if and only if its wavelength assignment  $\lambda(\pi)$  is also a permutation. Note that any  $k$ -legal permutation is also  $m$ -legal, for any  $m \in \{k, \dots, N\}$ . Furthermore,

*Lemma 3.1:* Let  $\chi \in \{0, \dots, N-1\}^N$ , such that all its elements are  $x$ . If a permutation  $\pi$  is  $k$ -legal, then the permutation  $\pi + \chi$  is also  $k$ -legal.

*Proof:* Assume towards a contradiction that  $\pi + \chi$  is not  $k$ -legal, so in  $\psi = \pi + \chi - I$  there exist  $k+1$  indices  $i_1, \dots, i_{k+1}$  such that  $\psi[i_1] = \dots = \psi[i_{k+1}]$ . This implies that  $\pi[i_1] + \chi[i_1] - i_1 = \dots = \pi[i_{k+1}] + \chi[i_{k+1}] - i_{k+1}$ . Since  $\chi[i_1] = \dots = \chi[i_{k+1}] = x$ , it follows that  $\pi[i_1] - i_1 = \dots = \pi[i_{k+1}] - i_{k+1}$ , implying that  $\pi - I$  has  $k+1$  identical elements. This contradicts the assumption that  $\pi$  is  $k$ -legal and the claim follows. ■

Next, we show how to build 1-legal permutations for odd values of  $N$ .

*Lemma 3.2:* If  $N$  is odd, then there exist a 1-legal permutation  $\pi_{\text{odd}}$  of  $\{0, \dots, N-1\}$ .

*Proof:* Let  $\pi_{\text{odd}}$  be the following permutation:

$$\pi_{\text{odd}}[i] = 2i \bmod N \quad i \in \{0, \dots, N-1\}.$$

If  $i \neq j$ , then  $2i - 2j$  is even and does not equal 0. Since  $N$  is odd, this implies that  $2i - 2j \neq 0 \bmod N$ , implying that  $\pi_{\text{odd}}[i] \neq \pi_{\text{odd}}[j]$ . Hence,  $\pi_{\text{odd}}$  is a permutation.

The wavelength assignment of  $\pi_{\text{odd}}$  is  $\pi_{\text{odd}} - I = 2I - I = I$  which is clearly a permutation. Thus,  $\pi_{\text{odd}}$  is 1-legal. ■

The following lemma deals with *even* values of  $N$ . Recently, this result was independently proven in [17], thus we omit the proof here due to lack of space.

*Lemma 3.3:* If  $N$  is *even* then there is no 1-legal permutation of  $\{0, \dots, N-1\}$ .

We next deal with 2-legal permutations:

*Lemma 3.4:* For every  $N$ , there is a 2-legal permutation of  $\{0, \dots, N-1\}$ .

*Proof:* Since every 1-legal permutation is also a 2-legal permutation, the claim follows immediately by Lemma 3.2 for odd values of  $N$ .

Assume that  $N$  is even and consider the following assignment permutation:

$$\pi_{\text{even}}[i] = \begin{cases} 2i \bmod N & i < N/2 \\ 2i + 1 \bmod N & N/2 \leq i \leq N-1 \end{cases}$$

Clearly,  $\pi_{\text{even}}$  is a permutation: Its first half covers all the even output ports and its second half covers all the odd output-ports. We now compute the wavelength assignment of  $\pi_{\text{even}}$ :

$$\lambda(\pi_{\text{even}})[i] = \begin{cases} i \bmod N & i < N/2 \\ i + 1 \bmod N & N/2 \leq i \leq N-1 \end{cases}$$

Clearly, each input port except input port 0 and  $N-1$  has a different wavelength assignment, while for input ports 0 and  $N-1$  we get  $\lambda(\pi_{\text{even}})[0] = \lambda(\pi_{\text{even}})[N-1] = 0$ , implying that  $\pi_{\text{even}}$  is a 2-legal permutation. Hence, we have a 2-legal permutation with only one wavelength repetition. ■

### IV. SINGLE-STAGE AWG-BASED SWITCHES

We first consider a single-stage AWG switch. We start by investigating the speedup required to realize any adversarial traffic pattern.

#### A. Worst-case traffic

Due to the AWG switch crosstalk impairment, the most difficult traffic to handle is a “generalized diagonal” traffic, in which all cells from input port  $i$  are directed to output port  $i + x \bmod N$  ( $x \in \{0, \dots, N-1\}$ ), with the same value of  $x$  for each input port: All inputs are forced to use the same wavelength to reach the proper output. As a consequence, under generalized diagonal traffic, if we are restricted to  $k$ -legal permutations, at most  $k$  cells can be forwarded in any scheduling decision, implying that the required speedup is  $S = N/k$ . Since  $k$  should be a small constant (less than 16, as shown in [12], but possibly even smaller, in case of all-optical cascades of switching stages) to avoid coherent crosstalk impairments, this implies that single stage AWG switches with large port counts require prohibitive speedups to cope with such an adversarial traffic. In contrast, a crossbar switch, in which there is no restriction on the permutation used, can schedule generalized diagonal traffic with speedup  $S = 1$ .

## B. Uniform traffic

Let us now focus on the classical uniform traffic pattern, where the uniform traffic matrix  $T$  contains all "1"s. It is well known that an  $N \times N$  uniform traffic matrix can be scheduled using a fixed *Time Division Multiplexing (TDM)* approach, in which, during a frame of  $N$  time slots, each input is in turn connected to the different  $N$  outputs. This can be interpreted as the decomposition of matrix  $T$  in a set of  $N$  "covering" permutation matrices. If we ignore the constraint of using only  $k$ -legal permutations, a possible decomposition of matrix  $T$  is achieved through a set of  $N$  generalized diagonal (switching) matrices  $\Pi_{\text{TDM}} = \{\pi_0, \pi_1, \dots, \pi_{N-1}\}$ , where  $\pi_i = I + \chi_i \bmod N$  and  $\chi_i$  is a vector in which all elements are  $i$ . For example, for  $N = 3$ , we have the following decomposition:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Next, we look for a decomposition of a uniform traffic matrix in  $k$ -legal permutations, and we wish to determine the minimum number of  $k$ -legal sub-permutations needed to decompose a uniform traffic matrix.

*Scheduling Uniform Traffic Using 1-legal Permutations:* We consider two cases, depending on the parity of  $N$ .

If  $N$  is odd, we use the following covering sequence of  $N$  1-legal permutations, in which each input-output pair is connected exactly once:  $\Pi = \{\pi_{\text{odd}} + \chi_x \mid 0 \leq x \leq N-1\}$ , where  $\pi_{\text{odd}}$  is the permutation defined in Lemma 3.2 and  $\chi_x$  is the  $N$ -vector whose elements are all equal to  $x$ . Being  $N$  odd, by Lemmas 3.2 and 3.1 all permutations in  $\Pi$  are 1-legal. Furthermore, each input port  $i$  is connected to output port  $j$  if and only if the permutation  $\pi_{\text{odd}} + \chi_x \in \Pi$  with  $x = (j - 2i) \bmod N$  is used.

When  $N$  is even, Lemma 3.3 implies that no 1-legal permutation exists. Thus, we cannot apply the same strategy as with odd values of  $N$ .

A straightforward way to get around this problem is to add another port to the switch making its port count odd. Note that this extra port will not be active in sending/receiving cells. The cost of adding ports relative to the entire switch is called the *spatial speedup* of the switch and in this case it is  $1 + \frac{1}{N}$  (a single additional port should be added for  $N$  existing ports).

Usually in switch design, a spatial speedup corresponds directly to the (time) speedup of the switch (as defined in Sec. II). The rationale behind it is that building a  $s$  times larger switch is logically equivalent (although this may not be technologically true in the optical domain where the complexity of time operation is larger) to building a  $s$  time faster switch by speeding-up the time between the ports. This leads to a possible conclusion that an AWG-based switch with even  $N$  can be realized by a (time) speedup of  $1 + \frac{1}{N}$ . The next theorem shows that this is not the case:

*Theorem 4.1:* An AWG-based switch with even  $N$  using 1-legal permutations requires a speedup  $S > 1 + 1/N$  to schedule a uniform traffic pattern.

*Proof:* Let the *weight* of a sub-permutation be the number of input (output) ports matched in this sub-permutation. Since  $N$  is even, Lemma 3.3 implies that there is no 1-legal permutation of  $\{0, \dots, N-1\}$ . Thus, the maximum weight of a 1-legal partial-permutation of  $\{0, \dots, N-1\}$  is  $N-1$ , i.e., the minimum number of repeated outputs in  $\pi$  is 2. Note also that, since each input port should be connected to each output port, the total weight of all (partial)-permutations in an  $N$  period is  $N^2$ . This implies that at least  $\lceil \frac{N^2}{N-1} \rceil = \lceil N + 1 + \frac{1}{N-1} \rceil = N + 2 > N + 1$  scheduling decisions are required in such a period, translating to a (time) speedup  $S > 1 + 1/N$ . ■

This result highlights an interesting difference between space and time speedup for the considered switch architecture.

*Scheduling Uniform Traffic Using 2-legal Permutations:* When  $N$  is odd, we already found in the previous section a scheduling for uniform traffic providing 100% throughput with no speedup, i.e. relying on 1-legal permutations. When  $N$  is even, and we permit 2-legal permutations, we can use the set  $\Pi = \{\pi_{\text{even}} + \chi_x \mid 0 \leq x \leq N-1\}$ , where  $\pi_{\text{even}}$  is the permutation defined in Lemma 3.4 and  $\chi_x$  is an  $N$ -vector whose all elements are  $x$ . The correctness of the construction is identical to the one described for 1-legal permutations.

In summary, in a single-stage AWG-based switch, uniform traffic can be scheduled using 1-legal permutations with no speedup when  $N$  is odd. When  $N$  is even, either a time speedup larger than  $1 + 1/N$  or a spatial speedup of  $1 + 1/N$  are required for 1-legal permutations. Alternatively, 2-legal permutations with no time or space speedup can be used. The worst case traffic scenario implies that a time speedup of  $N/k$  is required to schedule all admissible traffic patterns under the  $k$ -legal constraint.

## V. TWO-STAGE SOLUTION

In this section, we consider an AWG-based two-stage switch architecture. We adapt the two-stage Load Balanced Switch, introduced in [18] and briefly recalled below, and schedule the AWGs switching stages to operate with  $k$ -legal permutations.

### A. The Two-Stage Load Balanced Switch

The Load-Balanced Switch (Fig. 3) consists of two switching stages: The first stage performs load balancing of the incoming traffic, while the second stage performs the actual switching of cells to their destination. The basic idea is to transform any generic traffic pattern at the switch input into a uniform traffic at the output of the first stage, hence at the input of the second stage. To achieve this, cells arriving at one input in the first stage are forwarded in turn to all outputs of the first switching stage, regardless of the final output port. This permits to evenly distribute the considered input load to all first stage outputs. Since this traffic "spreading" operation is performed at all inputs, all first-stage outputs receive, on average, the same amount of traffic. The traffic at the input of the second stage is therefore uniform: Same load at all ports, and equal probability for any input-output pair. This

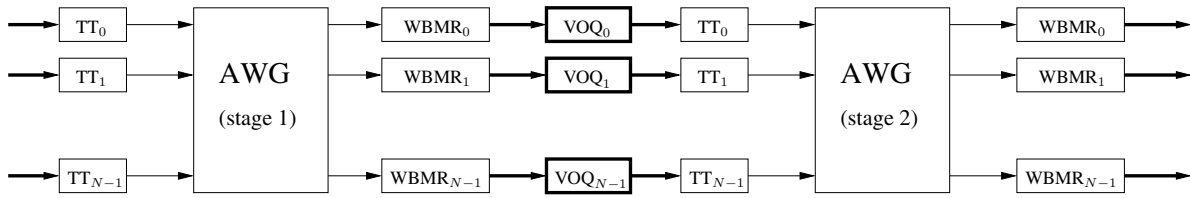


Fig. 3. A two-stage AWG-based switch. Electronic paths and components are highlighted with thick lines.

uniform traffic can be easily forwarded in a Load-Balanced switch by a fixed TDM switch schedule in the second stage, providing 100% throughput if the traffic is stationary and weakly mixing<sup>3</sup>, excellent delay performance and efficient buffer usage.

It is important to notice that no cell buffering is required at inputs, as arriving cells are immediately forwarded. VOQ buffering is instead required between the two stages (cells destined for different output ports are stored in separate FIFO buffers), in which cell queues may build up in case of congestion.

The load-balancing operation with VOQ buffering between the two switching stages has the drawback of out-of-sequence cell delivery. To avoid this, either resequences modules must be introduced at the outputs of the second stage, or more complex queuing structures and policies must be used between the two stages [19]–[21]. Both solutions must be implemented in the electronic domain, and increase complexity. Furthermore, the second solution has scalability problems, as its implementation complexity grows polynomially with  $N$ .

The scheduling in the two stages can be fully distributed, i.e., based on local decisions at each input port, without any coordination among different ports, apart from a switch-wide slot synchronization, provided that traffic is weakly mixing, to avoid adversarial patterns that would impair the load-balancing effect when the first stage is operated in fixed TDM. It can be easily understood that, for both switch stages, the scheduling translates into a periodic sequence of  $N$  permutations, such that each input-output pair is connected exactly once in each period. This is equivalent to scheduling uniform traffic matrices in both stages; hence, the scheduling for the two-stage  $N \times N$  load-balanced switch must cyclically run over a set of  $N$  covering permutations for a uniform (i.e., comprising all “1”) traffic matrix. While these  $N$  covering permutations can be found in several ways, we are interested, for the AWG-based switch, in a set of  $N$   $k$ -legal permutations, which can be obtained as described in Sec. IV-B.

Note that the two  $N \times N$  switching stages can be interpreted as a two-fold speedup realized in the space domain: Up to  $2N$  cells are simultaneously switched in every time slot.

*Implications to AWG-based Switches:* The results from the previous sections imply that any traffic pattern in the two-stage

<sup>3</sup>A stochastic sequence  $\{a(t), t \geq 1\}$  is weakly mixing if for all  $A, B \in \mathbb{R}^{(N \times N)^\infty}$ ,  $\lim_{t \rightarrow \infty} \frac{1}{t} \sum_{s=0}^{t-1} |\Pr(\theta_s a \in A, a \in B) - \Pr(a \in A) \Pr(a \in B)| = 0$ , where  $\theta_s a$  is the sequence  $a$  shifted by  $s$  time-slots:  $\theta_s a = \{a(t+s), t \geq 1\}$ .

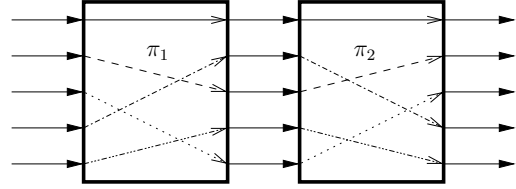


Fig. 5. Illustration of a 1-legal decomposition of the identity permutation in a  $5 \times 5$  switch using permutations  $\pi_1 = [0, 2, 4, 1, 3]$  and  $\pi_2 = [0, 3, 1, 4, 2]$ ; both  $\pi_1$  and  $\pi_2$  are 1-legal permutations.

AWG-based load-balanced switch can be scheduled with no speedup in each stage when the number of ports is odd. A spatial speedup equal to  $1 + 1/N$  is needed in the case of even number of ports. If considering the two-stage architecture as a switch spatial speedup, a generic traffic matrix can be scheduled with a speedup of  $2(2 + 2/N)$  for AWGs with odd (even) number of ports. Since the spatial speedup avoids any speed increase in components and transmission lines, this architecture is well suited for the optical domain, keeping the electronic speed in the feasible domain of today technology.

#### B. Avoiding buffers in the middle-stage

A promising approach to circumvent the need of buffering in the middle stage and resequencing at the egress is to control the AWG in both stages simultaneously, so that their combination will produce the desired permutation. In this setting, suppose we have an *oracle crossbar scheduler* that produces a sequence of permutations; our goal is to realize each of these permutations  $\pi$  using two  $k$ -legal permutations  $\pi_1$  and  $\pi_2$  such that  $\pi = \pi_1 \circ \pi_2$  where  $\circ$  denotes function composition. We call the pair of permutations  $\langle \pi_1, \pi_2 \rangle$  a  *$k$ -legal decomposition of  $\pi$* . Since the oracle may produce any permutation of  $\{0, \dots, N-1\}$ , our algorithm must be able to decompose all these permutations. Fig. 5 depicts a 1-legal decomposition of the identity permutation in a  $5 \times 5$  switch.

The resulting architecture is depicted in Fig. 4, where Tunable Wavelength Converters (TWC) (or equivalently, WBM followed, with no cell buffering stage, by a TT) are needed between the two stages to create the proper permutation. With this solution, VOQ buffering and O/E/O conversions are no longer required between the two stages, and cell out-of-sequence delivery is eliminated. VOQs are however needed in front of the first stage, similarly to the classical IQ switch architecture depicted in Fig. 2.

Besides eliminating the need for buffering in the middle stage, our decomposition approach has another significant advantage over the two-stage load-balanced switch approach,

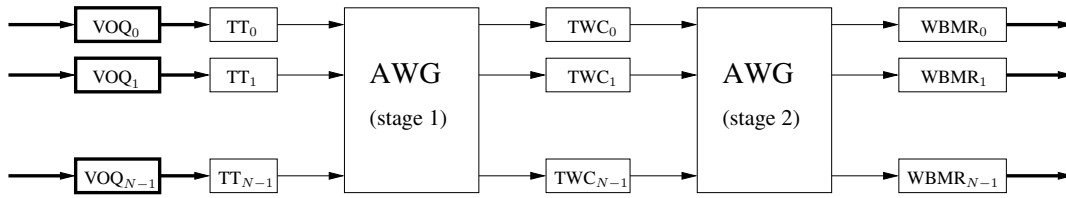


Fig. 4. A two-stage AWG-based switch with no internal buffering. Electronic paths and components are highlighted with thick lines.

as it can be adapted to work using *any* input-queued scheduling algorithm.

It is important to notice that while the Load Balanced switch provides 100% throughput on wide range of traffic patterns, there are still pathological traffic patterns that make its throughput arbitrarily small [22, Chapter 1.3.3]. Moreover, Load Balanced switches require a full switch reconfigurations at each scheduling decision; these reconfigurations may become infeasible as the line rates grow. Lastly, the two-stage Load Balanced switch is only aiming at providing 100% throughput; however, there is no bound or discussion on other important performance measures such as latency, smoothness (delay jitter) or fairness. These measures are crucial to provide the stringent QoS demands required by contemporary applications, and therefore a thorough research was done in the last decade to devise scheduling algorithms which perform better under these metrics (see, for example, [16] for a comprehensive survey).

Our decomposition algorithms offer a modular black-box approach in which any existing (or future) scheduling algorithm can be *converted* to be a crosstalk-preventing algorithm. In our approach, we model the switching algorithm as an oracle whose output is its scheduling decisions (that is, a sequence of permutations). Our algorithms get as an input these permutations (one by one) and produce the necessary  $k$ -legal permutations needed for crosstalk-preventing scheduling.

For example, to reduce the reconfigurations of the AWG devices one can use a scheduling algorithm which takes into account the reconfiguration delays and aims at minimizing the total delay (e.g., [23]–[25]). Since the change in permutations under such schedulers is not frequent, the number of needed decompositions decreases accordingly, thus facilitating the computation demands of our decomposition algorithms.

We now discuss for which values of  $k$  a  $k$ -legal decomposition exists for all permutations.

*Impossibility of Using 1-legal Permutations:* Recall that by Theorem 3.3, no 1-legal permutation exists for even  $N$ ; this implies that no 1-legal decomposition exists. However, we are able to prove, by a counter example, that no 1-legal decomposition algorithm exists for any  $N$ , regardless its parity.

*Theorem 5.1:* For any  $N$ , the following permutation  $\pi = [0, 1, 2, \dots, N-3, N-1, N-2]$  has no 1-legal decomposition.

*Proof:* Assume that there is a 1-legal decomposition of  $\pi$  into two 1-legal permutations  $\pi_1$  and  $\pi_2$ . Let  $\lambda_1 = \lambda(\pi_1)$  and  $\lambda_2 = \lambda(\pi_2)$  the wavelength assignments of  $\pi_1$  and  $\pi_2$ , respectively; since  $\pi_1$  and  $\pi_2$  are 1-legal,  $\lambda_1$  and  $\lambda_2$  are

permutations. Since for every  $i \leq N-3$ ,  $\pi[i] = i$ , then  $\lambda_2[\pi_1[i]] = -\lambda_1[i]$ . Thus, in the composite permutation  $\pi[i] = \pi_2[\pi_1[i]] = \pi_1[i] - \lambda_1[i] = i + \lambda_1[i] - \lambda_1[i] = i$ , as required.

Since  $\lambda_2$  is a permutation, the remaining elements  $\lambda_2[\pi_1[N-2]]$  and  $\lambda_2[\pi_1[N-1]]$  must use the remaining values  $-\lambda_1[N-2]$  and  $-\lambda_1[N-1]$ . If  $\lambda_2[\pi_1[N-2]] = -\lambda_1[N-2]$ , this results in  $\pi[N-2] = \pi_2[\pi_1[N-2]] = N-2 \neq N-1$ . Thus, we should have  $\lambda_2[\pi_1[N-2]] = -\lambda_1[N-1]$  and  $\lambda_2[\pi_1[N-1]] = -\lambda_1[N-2]$ , implying that  $\pi[N-2] = \pi_2[\pi_1[N-2]] = N-2 + \lambda_1[N-2] - \lambda_1[N-1]$ , which in turn implies that  $\lambda_1[N-2] - \lambda_1[N-1] = 1$  since  $\pi[N-2] = N-1$ . This yields that  $\pi_1[N-2] = N-2 + \lambda_1[N-2] = N-1 + \lambda_1[N-1] = \pi_1[N-1]$ , which contradicts that  $\pi_1$  is a permutation, and the claim follows. ■

*Decomposition Using 2-legal Permutations:* We continue by investigating 2-legal decomposition algorithms. First, for  $N = 3$ , Theorem 5.1 and the fact that each permutation of  $\{0, 1, 2\}$  is either 1-legal or 3-legal immediately implies the following:

*Corollary 5.2:* There is no 2-legal decomposition algorithm for  $N = 3$ .

For  $N = 4, \dots, 12$ , we verified by exhaustive search that any permutation  $\pi$  of  $\{0, \dots, N-1\}$  can be 2-legally decomposed. Hence, we can state that:

*Theorem 5.3:* There exists a 2-legal decomposition for all permutations with  $N \in \{4, \dots, 12\}$ .

Furthermore, it is important to notice that, given the permutation  $\pi$ , by choosing a 2-legal first permutation  $\pi_1$ , one only needs to verify that  $\pi_2 = \pi_1^{-1} \circ \pi$  is 2-legal to decide whether the decomposition is legal. Our experiments, reported in Tab. I, show that, to decompose any permutation  $\pi$ , it is sufficient to choose  $\pi_1$  from a small set  $\Pi_1$  of 2-legal permutations. Since  $\Pi_1$  can be pre-computed and programmed directly to the scheduler of the AWG, it implies that it is feasible to implement this scheduler for these values of  $N$ . Tab. I shows the size of  $\Pi_1$  for different values of  $N$ .

We were not able to find an algorithm to compute a 2-legal decomposition, nor we were able to prove that 2-legal decompositions exist for every  $N$ . We thus leave these two issues as open research problems and formulate the following conjecture, partially supported by our exhaustive searches for small values of  $N$ :

*Conjecture 5.1:* There exists a 2-legal decomposition for all permutations with  $N \geq 4$ .

TABLE I  
NUMBER OF FIRST-STAGE PERMUTATIONS NEEDED TO PROVIDE A  
2-LEGAL DECOMPOSITION.

Number of ports $N$	Number of permutations	Number of 2-legal permutations	Size of $\Pi_1$ (upper bound)
4	24	20	2
5	120	65	4
6	720	396	8
7	5040	2338	12
8	40320	16912	29
9	362880	132759	53
10	3628800	1183200	107
11	39916800	11531641	237
12	479001600	123019776	543

## VI. CONCLUSIONS

In this paper we studied ways to prevent coherent crosstalk impairments in AWG-based optical switching fabrics. The notion of  $k$ -legal permutations was introduced, in which each wavelength is re-used at most  $k$  times. We first found properties of 1-legal permutations, showing that a difference exist between odd and even values of the number of input and output ports  $N$ . We then showed that uniform traffic patterns can be scheduled in input-queued cell switches using 1-legal permutations with no speedup for odd  $N$  and with a small speedup with even  $N$ . General traffic patterns can be instead scheduled with 1-legal permutations using two-stage load-balancing switches using the same small speedup, VOQs between the two switching stages, and cell resequencing at outputs. 2-legal permutations were observed to permit to avoid intermediate VOQs and resequencing problems for small values of  $N$ . We left as an open research question to prove that this holds for all values of  $N$ .

Furthermore, we were able to formally prove that a 2-stage load-balanced switch can be configured with pairs of 4-legal permutations (or 3-legal permutations, in case  $N$  is a prime number) with no buffering between the two stages. These results, which are accompanied by quadratic decomposition algorithms, are not reported here due to lack of space.

*Acknowledgments:* The work described in this paper was carried out with the support of the BONE-project (“Building the Future Optical Network in Europe”), a Network of Excellence funded by the European Commission through the 7th ICT-Framework Programme.

## REFERENCES

- [1] R. Ramaswami and K. Sivarajan, *Optical networks: A practical perspective*. Morgan Kaufmann Publishers Inc., 1998.
- [2] J. Gripp, M. Dulek, J. Simsarian, A. Bhardwaj, P. Bernasconi, O. Laznicka, and M. Zirngibl, “Optical switch fabrics for ultra-high-capacity IP routers,” *J. Lightw. Technol.*, vol. 21, no. 11, pp. 2839–2850, 2003.
- [3] N. McKeown, “Optics inside routers,” in *ECOC*, 2003.
- [4] R. Tucker, “The role of optics and electronics in high-capacity routers,” *J. Lightw. Technol.*, vol. 24, no. 12, pp. 4655–4673, 2009.
- [5] C. Dragone, “An  $n \times n$  optical multiplexer using a planar arrangement of two star couplers,” *IEEE Photon. Technol. Lett.*, vol. 3, no. 9, pp. 812–815, 1991.
- [6] Y. Tamir and G. Frazier, “High-performance multi-queue buffers for VLSI communications switches,” in *Proc. of the 15 Ann. Symp. on Comp. Arch.*, 1988, pp. 343–354.

- [7] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, “Achieving 100% throughput in an input-queued switch,” *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1260–1267, 1999.
- [8] N. Caponio, A. Hill, F. Neri, and R. Sabella, “Single layer optical platform based on WDM/TDM multiple access for large scale ‘switchless’ networks,” *European Trans. on Telecommunications (ETT)*, vol. 11, no. 1, pp. 73–82, 2000.
- [9] A. Bhardwaj, J. Gripp, J. Simsarian, and M. Zirngibl, “Demonstration of stable wavelength switching on a fast tunable laser transmitter,” *IEEE Photon. Technol. Lett.*, vol. 15, no. 7, pp. 1014–1016, 2003.
- [10] M. Maier and M. Reisslein, “AWG-based metro WDM networking,” *IEEE Commun. Mag.*, vol. 42, no. 11, pp. S19–S26, 2004.
- [11] H. Takahashi, K. Oda, and H. Toba, “Impact of crosstalk in an arrayed-waveguide multiplexer on  $n \times n$  optical interconnection,” *J. Lightw. Technol.*, vol. 14, no. 6, pp. 1097–110, 1996.
- [12] R. Gaudino, G. G. Castillo, F. Neri, and J. Finochietto, “Simple optical fabrics for scalable terabit packet switches,” in *IEEE ICC*, May 2008, pp. 5331–5337.
- [13] V. Mikhailov, C. Doerr, and P. Bayvel, “Ultra low coherent crosstalk, high port-count free-space wavelength router,” in *Optical Fiber Communications Conference (OFC)*, vol. 1, March 2003, pp. 257–258.
- [14] D. Hunter and D. Smith, “New architectures for optical TDM switching,” *J. Lightw. Technol.*, vol. 11, no. 3, pp. 495–511, 1993.
- [15] M. Maier, M. Reisslein, and A. Wolisz, “High-performance switchless WDM network using multiple free spectral ranges of an arrayed-waveguide grating,” in *SPIE Terabit optical networking: architecture, control, and management issues*, vol. 4213, 2000, pp. 101–112.
- [16] H. Chao and B. Liu, *High Performance Switches and Routers*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2007.
- [17] M. Rodelgo-Lacruz, C. López-Bravo, F. J. González-Castano, and H. J. Chao, “Practical scalability of wavelength routing switches,” UVIGO GTI Technical Report TR08-GTI-080801, Tech. Rep., 2008.
- [18] C. Chang, D. Lee, and Y. Jou, “Load balanced Birkhoff-von Neumann switches, part I: One-stage buffering,” *Computer Communications*, vol. 25, no. 6, pp. 611–622, 2002.
- [19] I. Keslassy and N. McKeown, “Maintaining packet order in two-stage switches,” in *IEEE INFOCOM*, 2002, pp. 1032–1041.
- [20] C. Chang, D. Lee, and Y. Shih, “Mailbox switch: A scalable two-stage switch architecture for conflict resolution of ordered packets,” in *IEEE INFOCOM*, 2004, pp. 1995–2006.
- [21] H. Lee, “A two-stage switch with load balancing scheme maintaining packet dequence,” *IEEE Commun. Lett.*, vol. 10, no. 4, 2006.
- [22] I. Keslassy, “Load balanced router,” Ph.D. dissertation, Stanford University, June 2004.
- [23] X. Li and M. Hamdi, “On scheduling optical packet switches with reconfiguration delay,” *IEEE J. Sel. Areas Commun.*, vol. 21, no. 7, pp. 1156–1164, 2003.
- [24] V. Alaria, A. Bianco, P. Giaccone, E. Leonardi, and F. Neri, “Design of switches with reconfiguration latency,” in *IEEE ICC*, vol. 6, June 2006, pp. 2599–2605.
- [25] B. Towles and W. Dally, “Guaranteed scheduling for switches with configuration overhead,” *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 835–847, 2003.