Multiple-Input-Buffer and Shared-Buffer Architectures for Optical Packet- and Burst-Switching Networks

Konstantinos Yiannopoulos, Kyriakos G. Vlachos, *Member, IEEE*, and Emmanouel Varvarigos

***Abstract*—We present an architecture for implementing optical buffers, based on the feed-forward-buffer concept, that can truly emulate input queuing and accommodate asynchronous packet and burst operation. The architecture uses wavelength converters and fixed-length delay lines that are combined to form either a multiple-input buffer or a shared buffer. Both architectures are modular, allowing the expansion of the buffer at a cost that grows logarithmically with the buffer depth, where the cost is measured in terms of the number of switching elements, and wavelength converters are employed. The architectural design also provides a tradeoff between the number of wavelength converters and their tunability. The buffer architectures proposed are complemented with scheduling algorithms that can guarantee lossless communication and are evaluated using physical-layer simulations to obtain their performance in terms of bit-error rate and achievable buffer size.**

***Index Terms*—Input queuing, optical buffers, optical packet and burst switching, programmable delays, wavelength converters.**

I. Introduction

OPTICAL buffering is an important functionality in optical packet- and burst-switched (OPS/OBS) networks. It allows the temporary storage of data packets and bursts to resolve contention at the switch outputs. Various solutions have been proposed up to date including programmable delay lines [1]–[4] and optoelectronic conversion schemes [5]. Electronic buffering has been extensively utilized in currently installed optical networks; however, it is limited by the electronic processing speeds and the relatively slow O/E and E/O conversion times. Moreover, the cost and complexity associated with O/E/O conversions may be alleviated in optical-buffering schemes through component miniaturization and integration [6]; an integrated optical buffer has been recently reported [7]. Optical buffers that are based on programmable delay lines maintain the advantages of lower energy per written/read bit and smaller power dissipation, as compared to electronic buffers [8], and have been extensively used to form feed-forward or recirculating architectures employing, in addition, wavelength conversion to enhance buffering capabilities [9], [10]. Most schemes, however, assume slotted operation [1], which requires complex scheduling algorithms and, thus, are not suitable for asynchronous optical packet and burst switching. In particular, feedback loops theoretically provide infinite storage time, but they suffer from noise accumulation and optical signal-to-noise-ratio (OSNR) degradation. Furthermore, they require that the length of the feedback loop matches exactly the packet/burst duration in order to avoid loss of synchronization and that a large number of them are used to keep the blocking probability small. In contrast, feed-forward delay-line buffers are easier to implement, since the difference in the length of optical paths has to match a segment of the packet/burst duration (timeslot). Moreover, even though feed-forward delay-line architectures allow for short buffering times, recent studies indicate that statistically multiplexed optical networks will require only minimal buffering [11], provided some traffic engineering is performed. For these reasons, feed-forward delay-line buffer architectures seem to be a more practical solution for implementing limited optical buffering in packet/burst-switched networks.

Within this context, we present in this communication an architectural design for optical buffers, based on the feed-forward-buffer concept that can truly emulate input queuing and accommodate asynchronous packet and burst operation. The architectural design uses wavelength converters and fixed-length delay lines to internally route packets and bursts. These are combined together to form either a multiple-input buffer design, where a separate input buffer is employed per input port, or a shared-buffer design, where the same optical buffer is shared by all input ports. The latter significantly decreases the individual number of fiber delays needed. Both schemes are modular and are engineered to allow for the logarithmical increase of the buffer complexity and cost as a function of the buffer size. Moreover, as we show later on, the use of multiple wavelengths to route internally data bursts significantly reduces the number of delay stages needed and, thus, the number of wavelength converters. The architectural designs are complemented with physical-layer simulations to derive their design parameters (achievable buffer size) and illustrate their efficient performance in terms of bit-error rate (BER). In our analysis, we have assumed a wavelength-tuning range of w and investigated the cascadability of s stages.

The rest of this paper is organized as follows. Section II discusses the multiple-input-buffer design that consists of

Manuscript received December 1, 2006; revised February 14, 2007. This work was supported in part by the Operational Program for Educational and Vocational Training (EPEAEK), PYTHAGORAS II Program, and in part by the European Union via the IST/NoE e-Photon/ONe+.

The authors are with the Computer Engineering and Informatics Department and with the Research Academic Computer Technology Institute, University of Patras, 26500 Rio, Greece (e-mail: kvlachos@ceid.upatras.gr).

Digital Object Identifier 10.1109/JLT.2007.896804

parallel timeslot-interchangers (TSIs) and emulates input queuing. The buffer complexity and the number of stages grow optimally (logarithmically) with the buffer depth. Although previously reported work has also achieved logarithmic growth with respect to the number of stages [12], [13], we calculate for the first time to our knowledge the exact dependence of the logarithm base on the tunability of the wavelength converters to further reduce the required number of stages. We show that approximately 50% of the wavelengths available for internal switching contribute to this reduction. We also discuss the implications of deploying the buffer in an OPS/OBS node and provide a scheduling/contention-resolution algorithm to achieve lossless communication. In Section III, we present the shared-buffer design that emulates distributed buffering among all incoming/outgoing links of the optical switch. Similar to the previous buffer architecture, the number of stages grows logarithmically with buffering time (depth), but distributed buffering reduces the number of wavelengths that are available. As a result, more stages are required to construct the buffer. Still, the second architecture achieves lossless performance under less-strict requirements on the traffic conditions. This paper concludes with Section IV, which discusses the realization of the aforementioned architectural designs and investigates their efficient error performance versus the number of delay stages employed.

II. Multiple-Input Buffer Architecture

In the current section, we discuss the architecture of a feed-forward buffer that is capable of storing optical packets and bursts. Storage is accomplished by delaying the packets/bursts, and variable storage times are feasible by introducing programmable-delay elements inside the buffer. To facilitate our analysis, we assume that time is divided in timeframes, and packets/bursts are confined within their limits. We further assume that the timeframe contains T timeslots and that each packet/burst asynchronously occupies a number of consecutive timeslots. Under this scheme, providing that variable storage time for the packets/bursts is readily translated to interchanging timeslots. To this end, the buffer is equivalent, in terms of functionality, to k parallel TSIs: one per input port of the buffer, as shown in Fig. 1(a).

Each TSI constitutes an input buffer of size T and consists of s serially connected programmable-delay stages, as shown in Fig. 1(b). The architecture takes advantage of the wavelength parallelism to minimize the number of stages and, consequently, the hardware cost. A tunable-wavelength converter (TWC), with a tuning range of w wavelengths, is positioned at the input of the each delay stage. The TWC assigns packets/bursts to wavelengths based on the delay line that the packets/bursts must access in the delay bank. Mapping between wavelengths and delay lines is achieved by means of a passive wavelength demultiplexer, while a wavelength multiplexer feeds the delayed bursts to the next stage. The delays $D(i,j)$ that are introduced at stage i are a design parameter of the proposed architecture. These will be calculated in the following section, after introducing the space–time graph of the input-buffer architecture [14].

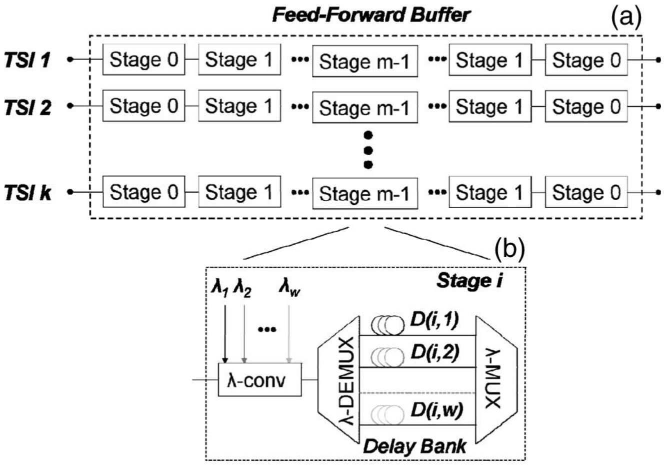

Fig. 1. (a) Multiple-input-buffer architecture. (b) Structure of each stage in the input-buffer. λ-conv is the TWC, and λ-MUX/DEMUX are the wavelength multiplexers and demultiplexers, respectively.

A. Formation of the Space–Time Graph

The space–time graph of the proposed design consists of nodes that are located at columns and rows, as detailed in the study in [14]. Columns i and $i+1$ represent the inputs and outputs, respectively, of input-buffer stage i, while rows account for the timeframe slots. For example, the node that is located at row j and column i on the space–time graph represents the jth timeslot of the input of stage i. From a space–time graph perspective, incoming packets/bursts are viewed upon as occupying a number of consecutive timeslots at the graph input. Packets/bursts are buffered at each stage after accessing the delay line that equals their storage time. This is represented on the space–time graph by solid lines (time-transitions) that connect timeslot nodes at the inputs and outputs of the respective stage. Time transitions connect the input nodes to output nodes at subsequent rows, since packets/bursts may only be delayed at each stage.

Buffering a packet/burst in the proposed design for a given duration corresponds to a path on the space–time graph, with origin of the node representing the input-slot pair on which the packet arrived and destination of the node representing the output-slot pair where the packet/burst leaves. Taking into consideration that multiple bursts/packets arrive at the buffer inputs within a timeframe, it follows that an interconnection pattern which maps input to output timeslots is formed on the space–time graph during each timeframe. Our goal in this section is to engineer the time transitions, or, equivalently, the delay times $D(i,j)$, at each stage so that the interconnection pattern formed contains a $\log_\mathrm{n}$–Benes graph as a subgraph. The $\log_\mathrm{n}$–Benes graph is derived from the $\log_2$–Benes graph after replacing the 2×2 switches with nxn crossbars and is a well-known rearrangeably nonblocking interconnection topology.

The purpose of constructing the $\log_\mathrm{n}$–Benes space–time graph is many-fold: The implementation requires a minimal number of serially connected stages that equals

$$s = 2 \cdot m - 1 = 2 \cdot \lceil \log_n T \rceil - 1 \tag{1}$$

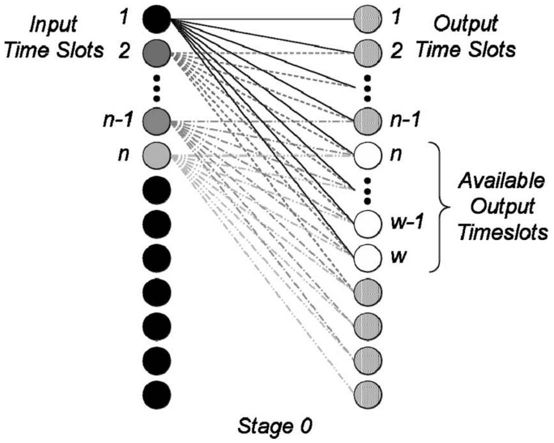

Fig. 2. Derivation of the elementary crossbar size on the space–time graph. The number of timeslots that are located at the next stage of the input-buffer and which may be accessed by a timeslot located at the current stage equals the number of available wavelengths w. The crossbar that connects timeslots between successive stages is formed so that all input timeslots may access all output timeslots.

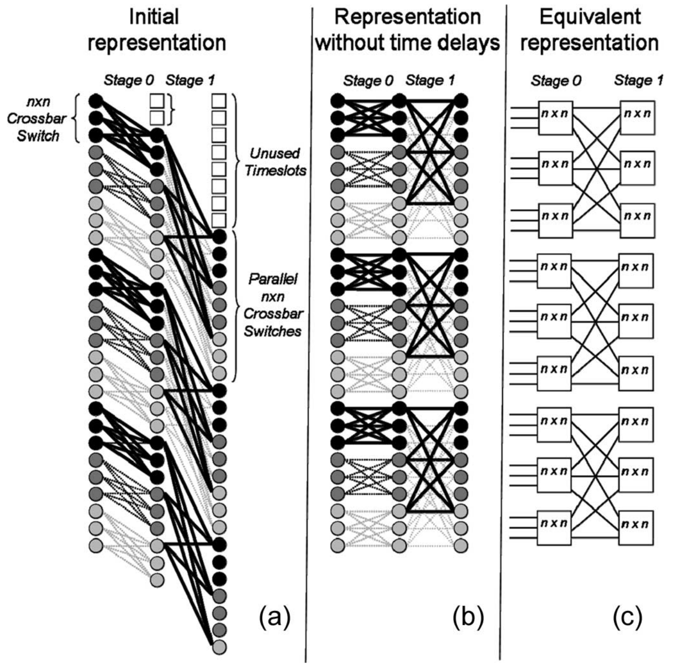

Fig. 3. Formation of the $\log_n$–Benes subgraph on the space–time graph. The nxn virtual switches at stage i are formed out of nodes that lie n^i timeslots apart.

for a given number T of timeslot per timeframe. Equation (1) shows that by implementing the $\log_n$–Benes space–time graph, one can achieve a drastic reduction in the number of stages, compared to previously reported [12], [13]. This is of particular importance when considering the hardware cost of the implementation. Moreover, as we will show later in Section IV, physical-layer impairments, such as timing jitter and the patterning effect, aggravate the optical-signal quality as the number of cascaded stages increases. Furthermore, the Benes space–time graph provides optimal (logarithmic) scalability with respect to the timeframe size T and is rearrangably nonblocking; thus, the proposed design is capable of storing packets/bursts without suffering internal collisions. Finally, finding collision-free paths within the Benes graph is a well-studied problem [15].

The building blocks of the $\log_n$–Benes graph are nxn crossbar switches, and thus, the first step for constructing it is to determine the size of the crossbars. The crossbars are formed out of time transitions on the space–time graph, as shown in Fig. 2, which corresponds to the first stage (stage 0) of the buffer. Input packets/bursts that have arrived within timeslots $\{1,\ldots,n\}$ may all access output timeslots $\{n,\ldots,w\}$, since time transitions to preceding timeslots are not allowed. Thus, the total number of output timeslots that are available to all n input timeslots is limited to $w-n+1$. The crossbar inputs equal the crossbar outputs, and we find that the crossbar size is

$$n = w - n + 1 \Leftrightarrow n = \left\lfloor \frac{w+1}{2} \right\rfloor \tag{2}$$

with $\lfloor x \rfloor$ denoting the integer part of x. Equation (2) shows that approximately 50% of the available wavelengths contribute to the formation of the crossbars that comprise the $\log_n$–Benes.

The second step for constructing the $\log_n$–Benes graph is to determine the time transitions that form the graph's crossbars in the respective stages. The process is shown in Fig. 3(a) for the first and second stage of the input buffer, as well as in Fig. 3(b) and (c), where the network of Fig. 3(a) is transformed to a standard representation. The formation of the $\log_n$–Benes graph crossbars requires that at each stage i, time transitions connect timeslots that are located n^i positions apart. This corresponds to setting the switch time delays, in timeslots, equal to

$$D(i,j) = j \cdot n^i, \quad i = 0,\ldots,m-1, \quad j = 0,\ldots,w-1. \tag{3}$$

The delays account for all time transitions on the space–time graph, even though only n time transitions per timeslot node contribute to the formation of the virtual crossbars. The remaining $w-n$ inactive transitions introduce a constant delay, after which, the output timeframe commences [white squares in Fig. 3(a)]. At the output of each stage, the delay equals

$$\Delta_i = n^i \cdot (n-1), \qquad i = 0,\ldots,m-1 \tag{4}$$

timeslots, and as a result, the minimum total delay that the packets/bursts experience when traversing the buffer is

$$\begin{aligned}\Delta &= \sum_{i=0}^{m-1} n^i \cdot (n-1) + \sum_{i=0}^{m-2} n^i \cdot (n-1)\\ &= n^m + n^{m-1} - 2\\ &= T + \frac{T}{n} - 2\end{aligned} \tag{5}$$

timeslots. Equation (5) may be viewed upon as constant storage latency introduced by the buffer.

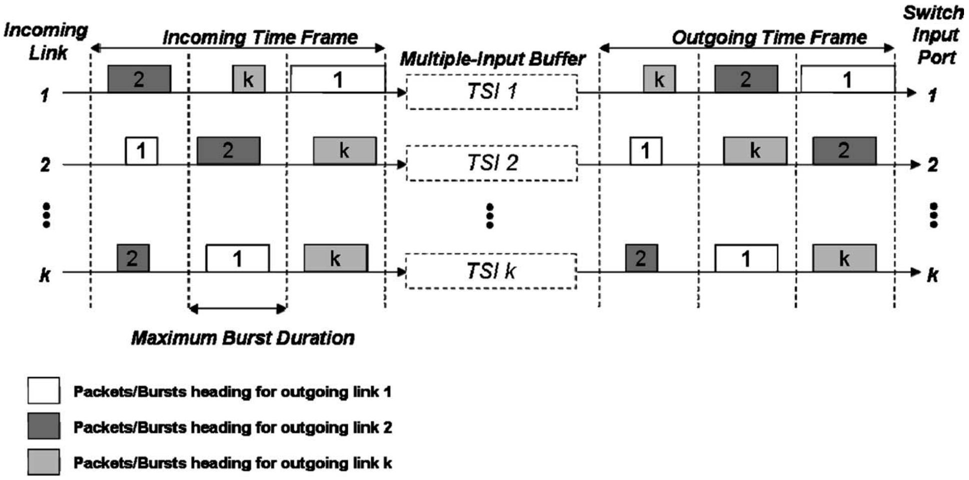

Fig. 4. Scheduling algorithm for the multiple-input-buffer architecture.

B. Asynchronous Operation in a Lossless OPS/OBS Node

The multiple-input buffer design provides lossless storage for packets/bursts that arrive at the incoming links of an OPS/OBS node within a timeframe T. However, the parallel TSIs do not spatially interconnect, and thus, the design is not capable of switching packets/bursts between input and output links of the OPS/OBS node. This shortcoming is addressed by deploying a space switch between the buffer outputs and the node outgoing links. The $\log_{\mathrm{n}}$–Benes subgraph of the time–space graph that represents the buffer ensures that there are no collisions inside the buffer, but still, collisions may occur at the outputs of the space switch if two or more packets/bursts simultaneously require accessing the same output. A scheduling algorithm is, therefore, required at the node control plane to arbitrate potential packet/burst collisions. We illustrate a suitable scheduling algorithm in Fig. 4. According to the algorithm, the output frame is partitioned into k successive subframes of equal duration. Packets/bursts arriving over incoming link p and heading for outgoing link q are scheduled in the output subframe (see Fig. 4)

$$S = \begin{cases} q - p + 1, & q \geq p \\ k + p - q + 1, & q < p. \end{cases} \tag{6}$$

Equation (6) dictates that packets/bursts that head for a particular outgoing link are scheduled to different subframes, depending on the link on which they arrive. As such, packet/burst collisions are avoided, provided that the total duration of packets/bursts d_{pq} (measured in timeslots) that arrive over link p and heading for outgoing link q does not exceed the subframe

$$d_{pq} \leq \frac{T}{k}, \qquad 1 \leq p, \qquad q \leq k. \tag{7}$$

Equations (6) and (7) provide an algorithm for scheduling packets/bursts without losses. Following the discussion of Section II-A, packets/bursts are scheduled after being converted to the appropriate internal wavelength and accessing the respective delay line at each programmable-delay stage. As a result, scheduling requires that the state of wavelength converters have to be set prior to sending the packets/bursts to the buffer. From a space–time graph perspective, (6) defines the timeslot patterns at the input and the output stages of the $\log_{\mathrm{n}}$–Benes network for one timeframe. Thus, setting the internal wavelengths is equivalent to calculating the state of the crossbars in all intermediate stages of the $\log_{\mathrm{n}}$–Benes graph so that the input timeslot pattern is routed to the respective output.

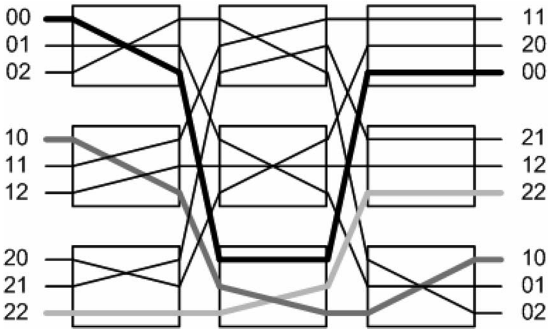

Fig. 5. Routing in the $\log_{\mathrm{n}}$–Benes network.

To perform routing in a $\log_{\mathrm{n}}$–Benes graph, we propose to use a modified parallel-routing algorithm that extends the parallel-routing algorithm on a binary Benes graph [15]. The algorithm involves setting the state of the outermost crossbars (at stages 0 and $s-1$) of the Benes graph, given the respective timeslot patterns. The outermost crossbars are then omitted, and the remaining network is partitioned into multiple Benes graphs of reduced size. The algorithm is recursively applied on the resulting graphs until the state of all crossbars is set. An example of the routing algorithm is detailed in the following section.

C. Benes Routing in the $\log_{\mathrm{n}}$–Benes Network

A routing example for the $\log_{\mathrm{n}}$–Benes network is illustrated in Fig. 5 for $n = 3$ and $T = 9$. The timeslots at the input of the buffer (stage 0) are assigned successive n-ary values, and the respective input permutation vector is formed. The permutation vector that corresponds to the output of the buffer (stage $s-1$) is formed in a similar fashion, after taking into account the results of the scheduling algorithm. In the example of Fig. 5, the input and output permutation vectors are

$$\begin{aligned} \pi_{\text{in}} &= (00 \quad 01 \quad 02 \quad 10 \quad 11 \quad 12 \quad 20 \quad 21 \quad 22) \\ \pi_{\text{out}} &= (02 \quad 21 \quad 22 \quad 20 \quad 00 \quad 11 \quad 01 \quad 10 \quad 12). \end{aligned} \tag{8}$$

We first focus on the input permutation vector. After exiting stage 0 on the space–time graph, the input permutation vector becomes

$$\pi_0 = (0a_8 \quad 0a_7 \quad 0a_6 \quad 1a_5 \quad 1a_4 \quad 1a_3 \quad 2a_2 \quad 2a_1 \quad 2a_0). \tag{9}$$

Equation (9) corresponds to time transitions with a_i denoting the output nodes that have been accessed. In a similar fashion, the output permutation vector at the input of stage s is

$$\pi_2 = (0b_2 \quad 2b_7 \quad 2b_8 \quad 2b_6 \quad 0b_0 \quad 1b_4 \quad 0b_1 \quad 1b_3 \quad 1b_5) \tag{10}$$

with b_i referring to the input nodes that have been accessed by the inverse time transitions. In (10), b_i are assigned to rows according the output permutation vector, due to the symmetry of the $\log_n$–Benes graph [15]. Moreover, the symmetry of the network implies that a_i and b_i that are located in a common row are equal, and as a result, T equations that correlate a_i and b_i are derived. The equations are solved after taking into consideration that a_i (and b_i) satisfy

$$a_{m\cdot n+i} \neq a_{m\cdot n+j}, i \neq j, \qquad i,j \in \{0,1,\ldots,n-1\} \tag{11}$$

so that no collisions occur inside the crossbars. The solution to the equations for the example of Fig. 5 is evaluated as

$$\begin{aligned} \pi_0 &= (02 \quad 01 \quad 00 \quad 12 \quad 10 \quad 11 \quad 21 \quad 20 \quad 22) \\ \pi_2 &= (02 \quad 21 \quad 20 \quad 22 \quad 00 \quad 11 \quad 01 \quad 10 \quad 12). \end{aligned} \tag{12}$$

After solving the equations for the outermost stages of the $\log_n$–Benes graph, we remove the aforementioned stages and divide the remaining graph into three (n in general) subgraphs. The permutation vectors of each subgraph are derived from (12) after grouping together the vector elements that correspond to the same subgraph or, equivalently, have a common least significant symbol

$$\begin{aligned} \pi_{\text{in}}^0 &= (0 \quad 1 \quad 2) \rightarrow \pi_{\text{out}}^0 = (2 \quad 0 \quad 1) \\ \pi_{\text{in}}^1 &= (0 \quad 1 \quad 2) \rightarrow \pi_{\text{out}}^1 = (2 \quad 1 \quad 0) \\ \pi_{\text{in}}^2 &= (0 \quad 1 \quad 2) \rightarrow \pi_{\text{out}}^2 = (0 \quad 2 \quad 1). \end{aligned} \tag{13}$$

No further permutation vectors have to be evaluated for the specific example, since the permutation vectors of (12) and (13) suffice to define the state of the crossbars at all stages.

The state of the crossbars at each stage l is set after taking into account the least significant symbols of the input and output permutation vectors that correspond to stage l. We assume that the aforementioned symbols form the reduced permutations vectors ρ_l^{in} and ρ_l^{out}; for instance, these are

$$\begin{aligned} \rho_0^{\text{in}} &= (0 \quad 1 \quad 2 \quad 0 \quad 1 \quad 2 \quad 0 \quad 1 \quad 2) \\ \rho_0^{\text{out}} &= (2 \quad 1 \quad 0 \quad 2 \quad 0 \quad 1 \quad 1 \quad 0 \quad 2) \end{aligned} \tag{14}$$

for stage 0 in our example. It is straightforward to verify from Fig. 3(a) that routing inside the $\log_n$–Benes graph corresponds to setting the delays at each stage equal to

$$d_l = \Delta_l + \left(\rho_l^{\text{out}} - \rho_l^{\text{in}}\right) \cdot n^l, \qquad 0 \leq l < s \tag{15}$$

where Δ_i is given by (4). This is equivalent to setting the wavelengths of the respective wavelength converters equal to

$$w_l = n + \rho_l^{\text{out}} - \rho_l^{\text{in}}, \qquad 0 \leq l < s. \tag{16}$$

An advantage of the parallel-Benes-routing algorithm relies on its low complexity, which leads to small execution times when the algorithm is implemented in hardware. An initial implementation of the algorithm on Xilinx Virtex-II FPGAs for $w = 3$ and $T = 16$ achieved execution time equal to 100 ns. This result is comparable to the duration of a frame of length $T = 10$ slots, where each slot carries a single asynchronous transfer mode cell at 40 Gb/s (10.6 ns).

III. Shared-Buffer Architecture

In the previous section, we discussed a buffering architecture that involves deploying one buffer per input. The architecture is optimal as far as the number of delay stages is concerned, but it requires the data traffic that arrives at the buffer to be equally distributed among its inputs, according to (6). In the current section, we discuss a shared-buffer architecture that requires less strict traffic conditions for lossless operation. Shared buffering is achieved by dedicating a number of wavelengths to spatially interconnect the parallel TSIs of the multiple-input-buffer design. The new design is shown in Fig. 6(a) and consists of serially interconnected programmable-delay stages in which the delay bank is accessible to all input ports, as detailed in Fig. 6(b). At each delay stage, k parallel wavelength converters assign the incoming packets/bursts to wavelengths that correspond to a pair of delay lines and output ports. The delay lines and output ports are accessed by the packets/bursts through all-passive space switches. Similar to Section II, our goal is to engineer the delays $D(i,j)$ that must be introduced at each stage so that the resulting space–time graph contains a Benes interconnection network as a subgraph.

A. Formation of the Space–Time Graph

The space–time graph for the first stage (stage 0) of the shared-buffer architecture is illustrated in Fig. 7. In contrast to the space–time graph of the multiple-input-buffer architecture, each timeslot node in the space–time graph of the current architecture includes k separate space nodes that correspond to the delay-stage inputs and outputs, as illustrated at the inset

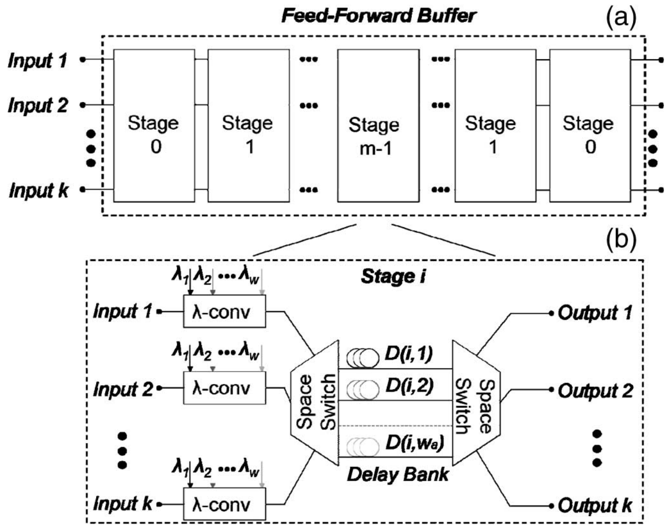

Fig. 6. (a) Shared-buffer architecture. (b) At the respective stages, each wavelength is assigned to a pair of delays and output ports. The normalized wavelength tunability w_a is defined as the ratio of the available wavelengths w to the number of input/output ports k.

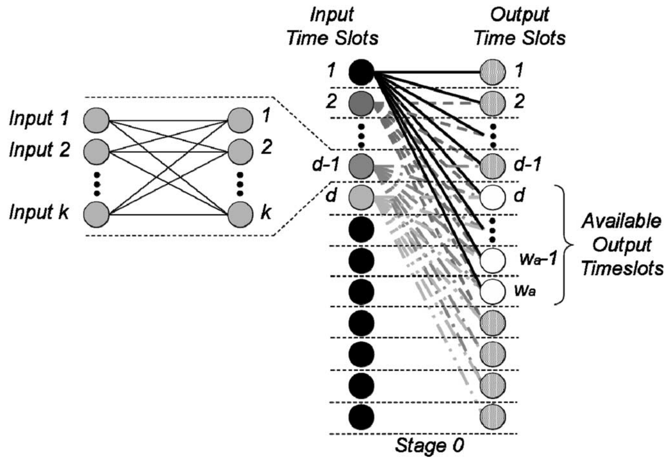

Fig. 7. Derivation of the elementary crossbar on the space–time graph. Each node of the space–time graph representing timeslots is expanded to k separate nodes that correspond to the input/output ports. The number of timeslots that are located at the next stage of the buffer and may be accessed by a timeslot located at the current stage equals the number of available wavelengths w divided by the number of ports k. The virtual switch that connects timeslots between successive stages is formed so that all input timeslots may access all output timeslots.

of Fig. 7. All transitions between the input and output space nodes of a delay stage are valid within a timeslot, since all stage outputs may be accessed by any stage input in Fig. 6(b). Time transitions are limited to nodes located at following rows, following the discussion of Section II-A.

For the rest of this section, we only consider time transitions and timeslot nodes that form the Benes subgraph on the space–time graph to simplify the illustration of our analysis. However, the time transitions between input and output timeslot nodes include all k possible space transitions between the corresponding input and output space nodes. This approach results in constructing the Benes interconnection network on the time transitions of the space–time graph and, afterward, expanding the crossbars and connections of the resulting network by a factor of k. Within this context, our goal is to engineer a $\log_{\rm d}$–Benes interconnection network on the time transitions of the space–time graph that corresponds to the shared-buffer architecture. The procedure is illustrated in Figs. 7 and 8. We first determine the size d of the elementary crossbars that comprise the Benes graph, and then, we define the delays that are required to form the $\log_{\rm d}$–Benes graph on the space–time graph. The crossbar size is determined after calculating the number w_a of timeslots that are fully accessed at the output of the stage 0 on the space–time graph. This is equal to the wavelength tunability w normalized by the number of ports k

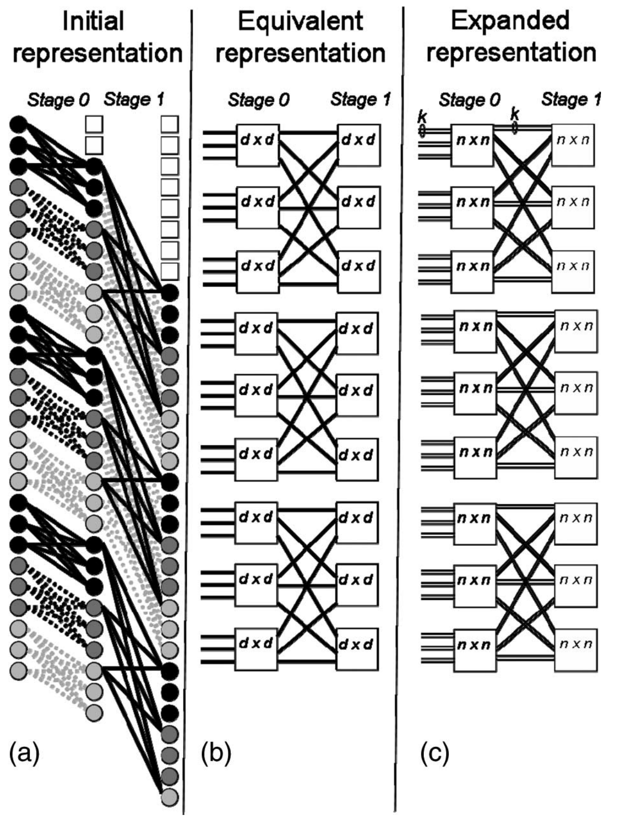

Fig. 8. (a), (b) Formation of the $\log_{\rm d}$–Benes subgraph on the time transitions of the space–time graph. The dxd virtual switches at stage i are formed out of nodes that lie d^i timeslots apart. (c) Expanded interconnection network.

$$w_a = \left\lfloor \frac{w}{k} \right\rfloor . \tag{17}$$

The size of the crossbars is calculated after combining (2) and (17)

$$d = \left\lfloor \frac{w_a + 1}{2} \right\rfloor = \left\lfloor \frac{\left\lfloor \frac{w}{k} \right\rfloor + 1}{2} \right\rfloor . \tag{18}$$

The $\log_{\rm d}$–Benes graph is formed as in Fig. 8(a) and (b) by setting the delays equal to

$$D(i,j) = j \cdot d^i, \quad i = 0, \ldots, m-1, \quad j = 0, \ldots, w_a - 1. \tag{19}$$

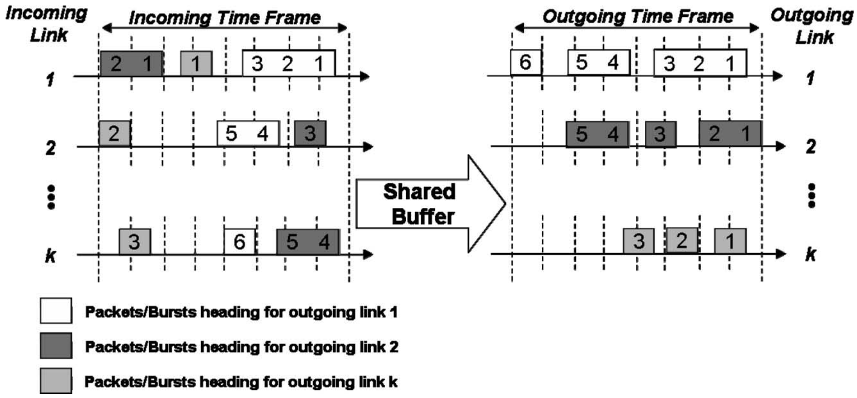
Incoming Link
Incoming Time Frame
Outgoing Time Frame
Outgoing Link
1
2
k
Shared Buffer
2 1
1
3 2 1
2
5 4
3
3
6
5 4
6
5 4
3 2 1
5 4
3
2 1
3
2
1
Packets/Bursts heading for outgoing link 1
Packets/Bursts heading for outgoing link 2
Packets/Bursts heading for outgoing link k

Fig. 9. Scheduling algorithm for the shared-buffer architecture.

Similar to the multiple-input-buffer design, only d time transitions are utilized per timeslot, and as a result, the bursts experience a minimum storage latency equal to

$$\begin{aligned}\Delta &= \sum_{i=0}^{m-1} d^i \cdot (d-1) + \sum_{i=0}^{m-2} d^i \cdot (d-1)\\ &= d^m + d^{m-1} - 2\\ &= T + \frac{T}{d} - 2 \end{aligned} \tag{20}$$

timeslots.

The fully expanded network that includes space transitions is detailed in Fig. 8(c) after taking into consideration that each time transition includes k space transitions and that the crossbar size is $n = k \cdot d$ nodes in total. The fully expanded network is rearrangably nonblocking, and thus, buffering without internal collisions can be achieved. Moreover, the number of delay stages that are required to implement the shared-buffer architecture is given by (see Fig. 8)

$$s = 2 \cdot m - 1 = 2 \cdot \lceil \log_d T \rceil - 1. \tag{21}$$

Equation (21) shows that the shared-buffer design is not optimal as compared to the multiple-input-buffer design, and this is because we have constructed a $\log_\mathrm{d}$–Benes graph on the time transitions instead of $\log_\mathrm{n}$–Benes graphs on both time and space transitions. As a result, the scalability of the shared buffer with respect to the timeframe T is suboptimal. Moreover, the shared-buffer design does not scale with the number of input/output ports, and additional ports may only be accommodated if redundant wavelengths are provisioned. However, the drawbacks of the shared-buffer architecture are balanced by the fact that it requires less-strict-traffic conditions to achieve lossless operation, as we will show in the following section.

B. Asynchronous Operation in a Lossless OPS/OBS Node

The shared-buffering architecture may be deployed as-is in an OPS/OBS node, since buffering and switching are performed independently. Moreover, the rearrangable nonblocking property of the expanded interconnection graph of Fig. 8(c) ensures that no packet/burst collisions take place inside the shared buffer, provided that the total traffic that arrives at all buffer inputs and heads for a specific buffer output does not exceed T timeslots within a timeframe. This is a looser traffic condition than (6), since incoming traffic does not have to be equally distributed among all buffer inputs.

Packets/bursts that arrive within the same timeframe are placed on a common outgoing frame, which starts after Δ timeslots following the end the incoming frame. In the shared-buffer architecture, output timeslots are occupied by incoming packets/bursts according to a modification of the packing rule that is detailed in the study in [13]. The modified packing rule is illustrated in Fig. 9. Packets/bursts that head for a common outgoing link and, thus, the respective timeslots they occupy, are logically grouped together, and the timeslots that belong to the same group are given ranks. A rank of a timeslot equals r^q, if it is the rth timeslot that has arrived at incoming link p and heads for outgoing link q. A timeslot at the incoming timeframe with rank r^q will be mapped at the output timeframe to timeslot

$$y^q = \sum_{l=1}^{p-1} n_{l,q} + r^q - 1, \qquad y \in \{0, \ldots, T-1\} \tag{22}$$

where $n_{l,q}$ is the total duration of packets/bursts (in timeslots) between incoming link l and outgoing link q.

The modified packing rule determines the storage time of each packet/burst inside the buffer. Moreover, each packet/burst requests that it be switched to an outgoing node link. Following the discussion of Section III-A, both storing and switching of packets/bursts are implemented by properly assigning wavelengths inside the programmable-delay stages. From a space–time graph perspective, storing and switching form the input/output permutation patterns of the Benes network, while wavelength assignment is equivalent to setting the state of the network crossbars. To this end, we developed a parallel-Benes-routing algorithm suitable for the expanded $\log_\mathrm{d}$–Benes graph of Fig. 8. An indicative example of the algorithm is detailed in the following section.

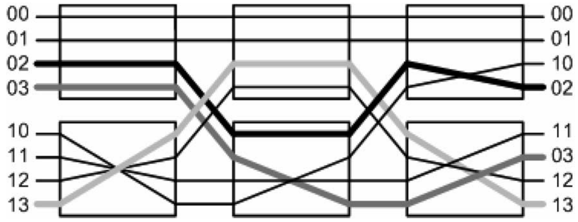

Fig. 10. Routing in the expanded $\log_{\mathrm{d}}$–Benes graph.

C. Benes Routing for the $\log_{\mathrm{d}}$–Benes Network

An example for routing in the $\log_{\mathrm{d}}$–Benes graph is detailed in Fig. 10 for $n = 4$ $(k = 2, d = 2)$ and $T = 4$. The input and output permutation vectors are given by

$$\begin{aligned}\pi_{\mathrm{in}} &= (00 \quad 01 \quad 02 \quad 03 \quad 10 \quad 11 \quad 12 \quad 13)\\ \pi_{\mathrm{out}} &= (00 \quad 01 \quad 03 \quad 11 \quad 02 \quad 10 \quad 12 \quad 13).\end{aligned} \tag{23}$$

The permutation vectors at the output of stage 0 and the input of stage $s-1$ are calculated as

$$\begin{aligned}\pi_0 &= (0a_0 \quad 0a_1 \quad 0a_2 \quad 0a_3 \quad 1a_4 \quad 1a_5 \quad 1a_6 \quad 1a_7)\\ \pi_2 &= (0b_0 \quad 0b_1 \quad 0b_3 \quad 1b_5 \quad 0b_2 \quad 1b_4 \quad 1b_6 \quad 1b_7).\end{aligned} \tag{24}$$

Equation (24) is solved for a_i and b_i using the symmetry properties of the expanded $\log_{\mathrm{n}}$–Benes graph, after taking into account (11). The solution is facilitated, however, by the fact that crossbars are connected with groups of k-parallel lines in the expanded network. As a result, a_i (and b_i) that correspond to the same k parallel-line group may be interchanged, since they originate from and head for the same crossbar. Thus

$$a_{m\cdot n+i} \leftrightarrow a_{m\cdot n+j}, \qquad i \neq j, \qquad i, j \in \{0, 1, \ldots, k-1\}. \tag{25}$$

A solution for the outermost stages in our example is

$$\begin{aligned}\pi_0 &= (00 \quad 01 \quad 02 \quad 03 \quad 13 \quad 12 \quad 11 \quad 10)\\ \pi_2 &= (00 \quad 01 \quad 02 \quad 13 \quad 03 \quad 12 \quad 11 \quad 10).\end{aligned} \tag{26}$$

We then omit the outermost crossbars and divide the resulting graph into two (d in general) subgraphs. The permutation vectors for the subgraphs are formed after grouping together the vector elements that correspond to the same crossbar. This is equivalent to grouping vector elements which have the least significant symbols z that satisfy

$$z \in [i \cdot k, \quad (i+1) \cdot k - 1], \qquad 0 \leq i < d. \tag{27}$$

In the example of Fig. 10, we find that

$$\begin{aligned}\pi_{\mathrm{in}}^0 &= (00 \quad 01 \quad 11 \quad 10) \rightarrow \pi_{\mathrm{out}}^0 = (00 \quad 01 \quad 11 \quad 10)\\ \pi_{\mathrm{in}}^1 &= (02 \quad 03 \quad 13 \quad 12) \rightarrow \pi_{\mathrm{out}}^1 = (02 \quad 13 \quad 03 \quad 12).\end{aligned} \tag{28}$$

Equation (28) is readily solved after renumbering the vector elements with respect to the column they occupy

$$\begin{aligned}\pi_{\mathrm{in}}^0 &= (0 \quad 1 \quad 3 \quad 2) \rightarrow \pi_{\mathrm{out}}^0 = (0 \quad 1 \quad 3 \quad 2)\\ \pi_{\mathrm{in}}^1 &= (0 \quad 1 \quad 3 \quad 2) \rightarrow \pi_{\mathrm{out}}^1 = (0 \quad 3 \quad 1 \quad 2).\end{aligned} \tag{29}$$

Equations (26) and (29) suffice to calculate the permutation vectors ρ_l^{in} and ρ_l^{out} at all network stages and define the crossbars' state. The respective wavelengths are assigned according to (16).

IV. Semiconductor-Optical-Amplifier-Based Mach–Zehnder Interferometer (SOA-MZI)-Based Implementation

In the current section, we discuss the realization of the proposed buffer architectures with SOA-MZI TWCs. The prime target is to investigate the BER degradation that is imposed by each SOA-MZI TWC stage, with a goal to determine the maximum number of stages s that may be cascaded. Even though SOA-MZI TWCs are not format transparent, like the four-wave-mixing-based TWCs [16], they are well-suited candidates for the proposed multistage designs, since they require low switching energies and provide high output powers. Therefore, it is possible to directly cascade delay stages without deploying inline booster amplifiers that aggravate the OSNR and limit the TWCs cacadability.

We have used a common simulation setup for both buffers architectures, as shown in Fig. 11, since both architectures involve the cascaded operation of programmable-delay stages. The fiber delay lines were not taken into account during the simulations, since their lengths are application specific and are determined by the available wavelengths and the time-frame size, according to (3) and (19). Moreover, the delay line losses are negligible in comparison with the Mux/Demux losses for most practical cases.

The cascadability of SOA-MZI TWCs has been investigated at 10 and 40 Gb/s using the VPI Transmission Maker simulation software. The simulation parameters are summarized in Table I, and the simulation setup is shown in Fig. 11 for both rates. A symmetric MZI topology has been considered for all TWCs, with two identical SOAs residing at the MZI arms. The SOAs of the setup have been modeled using the built-in VPI TLLM model [17] with parameters that have been experimentally verified [18]. The SOA-generated amplified spontaneous emission has also been taken into account in the TLLM parameters. A 3-dB optical coupler at the input of the MZIs splits the incoming signal into two components that induce gain and phase changes in the respective SOAs. The SOA gain and phase changes are imparted on two replicas of a continuous-wave (CW) signal that operates at a wavelength. The CW signal replicas interfere in a 3-dB coupler at the output of the MZI, and the MZI switch state is set to ON or OFF, depending on the whether the CW signal replicas have experienced different gains and phases or not. Differential operation of the switch is accomplished by introducing attenuation and temporal delay in the lower MZI arm input-signal component. Under this scheme,

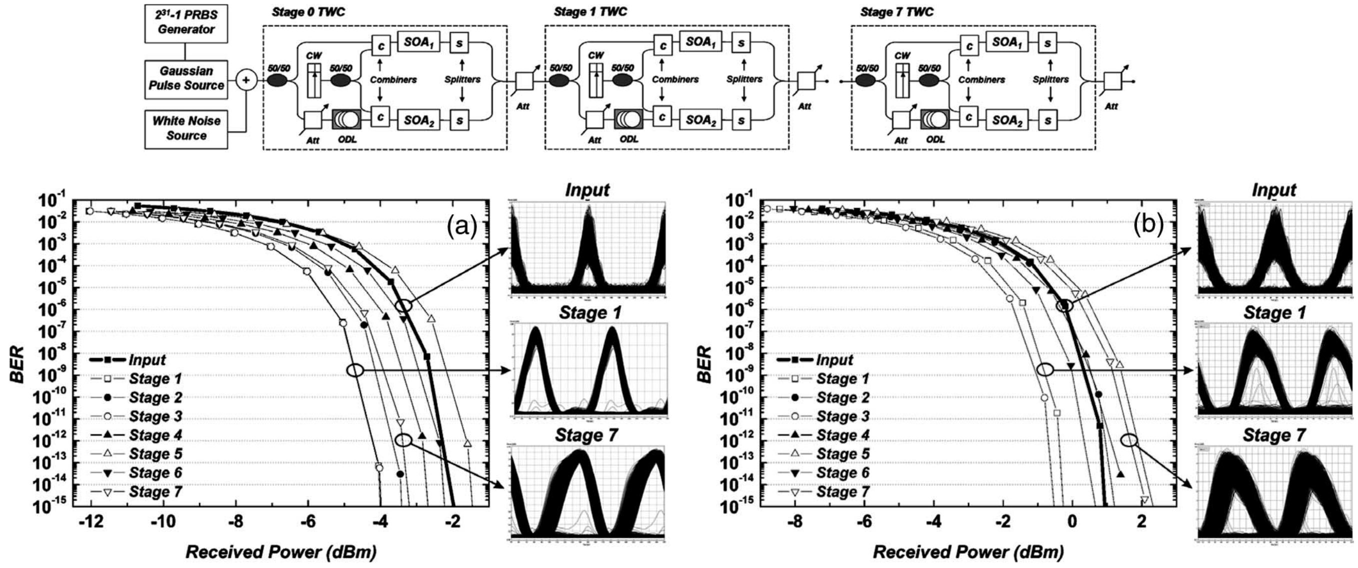

Fig. 11. (Top) Simulation setup of the SOA-MZI TWC. Att: Optical Attenuator. ODL: Optical Delay Line. CW: Continuous Wave. (Bottom) Simulated BER versus receiver power for the cascaded SOA-MZI wavelength converters operating at (a) 10 and (b) 40 Gb/s. The eye diagrams of the signals at the input of the buffer and at buffer stages 1 and 7 are shown as insets.

TABLE I
SIMULATION PARAMETERS

	10 Gbps	40 Gbps
Signal Power	-1.7 dBm	2 dBm
Signal Pulse-width	20 ps	8 ps
Continuous Wave Power	0 dBm	3 dBm
SOA Current	300 mA	500 mA
SOA Recovery Time (10%-90%)	21 ps	12 ps
SOA Gain	14 dB	16 dB
SOA Gain Peak Wavelength	1560 nm	1560 nm
Channel Spacing	100 GHz	400 GHz
Lower MZI Arm Delay	5 ps	4 ps
Lower MZI Arm Attenuation	5 dB	2 dB
Inline Attenuation	6.5 dB	1.6 dB

the upper arm component switches the MZI ON, and the lower arm component switches the MZI OFF [19]. A copropagating input/CW configuration with ideal additional combiners and splitters has been considered for the MZIs to avoid the excessive gain overshooting and slow recovery in the SOAs [20]. Inline optical attenuators of constant attenuation are deployed at the output of the TWCs so that all stages exhibit unity gain, and the limited input dynamic range of the MZI TWCs does not affect the setup performance. In an actual implementation, the inline optical attenuators would account for the losses of the multiplexers, demultiplexers, and delay lines that construct the programmable delay stages of both buffer designs. The attenuators are replaced by optical amplifiers when the buffer stages exhibit excessive loss, using the integrated TWC topology that has been demonstrated in the study in [21].

The simulation setup has been tested with a $2^{31} - 1$ pseudorandom bit sequence encoded on Gaussian optical pulses. The input signal was corrupted by additive white Gaussian noise before entering the first stage of the setup. The simulation results are presented in Fig. 11(a) and (b) for line rates of 10 and 40 Gb/s, respectively. Fig. 11(a) illustrates the BER performance versus the received power for up to seven cascaded TWCs. A negative power penalty of up to 2 dB has been observed for all stages due to the regenerative properties of the MZI TWC [22]. Higher received powers are required to achieve error-free operation of the simulated setup, as compared to previously reported experimental results, owing to the high level of the input noise spectral density. The corresponding eye diagrams reveal that signal quality degrades with the number of cascaded stages, since amplitude variations in the incoming pulse train result in timing variations at the output of the TWC, thus causing accumulation of timing jitter. The same number of stages has also been achieved at 40 Gb/s at the expense of a 1.5-dB power penalty, as shown in Fig. 11(b). Still, the signal quality is worse, as compared with that of the 10-Gb/s operation, and it can be verified from the eye diagrams at 40 Gb/s. Signal quality is degraded due to timing jitter that is accumulated with the number of stages and the patterning effect that is introduced by the limited gain-recovery time of the SOA. Despite the fact that the SOAs are driven at higher currents to provide for a faster gain-recovery time, the large pulsewidth of the 40-Gb/s signals does not allow for a larger gain-recovery time span. Shorter pulsewidths, however, would require a smaller delay between the signal components that enter the MZI; thus, the TWCs would exhibit loss, and amplification would be necessary between stages. The signal-degradation effects of wavelength crosstalk between the CW and input signals have not been taken into account, since very low crosstalk values have been demonstrated in similar SOA-MZI setups [20]. Moreover, the extinction ratio of the SOA-MZI has been considered ideal [23]. We believe that both assumptions are valid, since our buffer designs require only a limited number of cascaded stages.

Following the analysis of the simulation results, we may conclude that the maximum number of delay stages for the aforementioned buffer designs that yield an efficient BER of less than 10^{-9} is seven. For more stages, the simulations revealed that the accumulated timing jitter and pattern effect degrade error performance to less than 10^{-7}. Still, seven stages are well capable of providing storage times of practical use when accompanied by broad wavelength-tuning range w, according to (1) and (21). For instance, for five available wavelengths, the seven-stage multiple-input buffer is capable of storing 81 packets, with sufficient buffering capability (according to [11]). Equal storage time may be achieved for a two-port shared-buffer architecture, at the expense of 11 wavelengths. The tuning range of the SOA-MZI TWCs is determined by the channel spacing and the spectral width of the SOA gain peak, which is typically about 25 nm. Wide TWCs operating at 40 Gb/s have been recently demonstrated [21].

The simulation results of the current section have been recently utilized toward the experimental demonstration of a 10-Gb/s multiple-input buffer [24]. The experimental demonstration of the proposed buffer architecture has been performed for three wavelengths $(w = 3)$ and three stages $(s = 3)$, and the buffer has achieved error-free buffering of four optical packets at 10 Gb/s.

V. Conclusion

We have presented the architectural design of two optical burst buffers using wavelength converters and fixed-length delay lines that are combined to form either a multiple-input buffer or a shared buffer. Both schemes are modular, allowing the expansion of the buffer at a cost that grows logarithmically with the buffer size, where the cost is measured in terms of the number of switching elements (wavelength converters) required, while wavelength parallelism is used to significantly reduce the number of delay stages. Furthermore, we have also proposed architecture-suited algorithms for providing contention resolution within the buffering time, as well as algorithms for scheduling the internal wavelengths, again for contention resolution. The architectural study was complemented with physical-layer simulation of cascaded wavelength converters at 10 and 40 Gb/s to investigate the efficient error performance of the proposed buffer designs. It was found that error-free operation at power penalties below 0 and 1.5 dB, respectively, can be achieved for up to seven cascaded wavelength converters.

References

[1] D. K. Hunter, M. C. Chia, and I. Andonovic, "Buffering in optical packet switches," *J. Lightw. Technol.*, vol. 16, no. 12, pp. 2081–2094, Dec. 1998.

[2] I. Chlamtac *et al.*, "CORD: Contention resolution by delay lines," *IEEE J. Sel. Areas Commun.*, vol. 14, no. 5, pp. 1014–1029, Jun. 1996.

[3] C. Guillemot *et al.*, "Transparent optical packet switching: The European ACTS KEOPS project approach," *J. Lightw. Technol.*, vol. 16, no. 12, pp. 2117–2134, Dec. 1998.

[4] R. L. Cruz and J.-T. Tsai, "COD: Alternative architectures for high-speed packet switching," *IEEE/ACM Trans. Netw.*, vol. 4, no. 1, pp. 11–21, Feb. 1996.

[5] S. Bjørnstad, N. Stol, and D. R. Hjelme, "An optical packet switch design with shared electronic buffering and low bit rate add/drop inputs," in *Proc. Int. Conf. Transparent Opt. Netw.*, 2002, pp. 69–72.

[6] R. V. Caenegem *et al.*, "From IP over WDM to all-optical packet switching: Economical view," *J. Lightw. Technol.*, vol. 24, no. 4, pp. 1638–1645, Apr. 2006.

[7] E. F. Burmeister and J. E. Bowers, "Integrated gate matrix switch for optical packet buffering," *IEEE Photon. Technol. Lett.*, vol. 18, no. 1, pp. 103–105, Jan. 2006.

[8] R. S. Tucker, "The role of optics and electronics in high-capacity routers," *J. Lightw. Technol.*, vol. 24, no. 12, pp. 4655–4673, Dec. 2006.

[9] M. C. Chia *et al.*, "Packet loss and delay performance of feedback and feed-forward arrayed-waveguide gratings-based optical packet switches with WDM inputs-outputs," *J. Lightw. Technol.*, vol. 19, no. 9, pp. 1241–1254, Sep. 2001.

[10] C. M. Gauger, "Dimensioning of FDL buffers for optical burst switching nodes," in *Proc. 6th IFIP Work. Conf. ONDM*, Torino, Italy, Feb. 2002, pp. 117–132.

[11] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with very small buffers," in *Proc. IEEE INFOCOM*, 2006, pp. 1–11.

[12] D. K. Hunter, W. D. Cornwell, T. H. Gilfedder, A. Franzen, and I. Andonovic, "SLOB: A switch with large optical buffers for packet switching," *J. Lightw. Technol.*, vol. 16, no. 10, pp. 1725–1736, Oct. 1998.

[13] E. A. Varvarigos, "The 'packing' and the 'scheduling' packet switch architectures for almost all-optical lossless networks," *J. Lightw. Technol.*, vol. 16, no. 10, pp. 1757–1767, Oct. 1998.

[14] D. K. Hunter and D. G. Smith, "New architectures for optical TDM switching," *J. Lightw. Technol.*, vol. 11, no. 3, pp. 495–511, Mar. 1993.

[15] T. T. Lee and S. Y. Liew, "Parallel routing algorithms in Benes–Clos networks," *IEEE Trans. Commun.*, vol. 50, no. 11, pp. 1841–1847, Nov. 2002.

[16] N. Chi and S. Yu, "Optical subcarrier labeling transparent to the payload format using carrier suppression technique," *IEEE Photon. Technol. Lett.*, vol. 18, no. 8, pp. 971–973, Apr. 15, 2006.

[17] VPI ComponentMaker, *Active Photonics User's Manual*. Ch. 5: Overview of the TLLM.

[18] E. Kehayas *et al.*, "ARTEMIS: 40-Gb/s all-optical self-routing node and network architecture employing asynchronous bit and packet-level optical signal processing," *J. Lightw. Technol.*, vol. 24, no. 8, pp. 2967–2977, Aug. 2006.

[19] K. Tajima, "All-optical switch with switch-off time unrestricted by carrier lifetime," *Jpn. J. Appl. Phys.*, vol. 32, no. 12A, pp. L1746–L1749, Dec. 1, 1993.

[20] J. Leuthold *et al.*, "All-optical Mach–Zehnder interferometer wavelength converters and switches with integrated data- and control-signal separation scheme," *J. Lightw. Technol.*, vol. 17, no. 6, pp. 1056–1066, Jun. 1999.

[21] V. Lal *et al.*, "Performance optimization of an InP-based widely tunable all-optical wavelength converter operating at 40 Gb/s," *IEEE Photon. Technol. Lett.*, vol. 18, no. 4, pp. 577–579, Feb. 15, 2006.

[22] J. Mork, F. Ohman, and S. Bischoff, "Analytical expression for the bit error rate of cascaded all-optical regenerators," *IEEE Photon. Technol. Lett.*, vol. 15, no. 10, pp. 1479–1481, Oct. 2003.

[23] J. Leuthold *et al.*, "All-optical space switches with gain and principally ideal extinction ratios," *IEEE J. Quantum Electron.*, vol. 34, no. 4, pp. 622–633, Apr. 1998.

[24] O. Zouraraki *et al.*, "Optical packet buffering in all-optical time-slot-interchanger architecture," *IEEE Photon. Technol. Lett*. submitted for publication.

Konstantinos Yiannopoulos was born in Tripoli, Arcadia, Greece, in December 1977. He received the Ph.D. and Diploma degrees in electrical and computer engineering from the National Technical University of Athens, Athens, Greece, in 2004 and 2000, respectively.

From 1999 to 2004, he was with the Photonics Communications Research Laboratory, National Technical University of Athens. He is currently a Post Doctoral Researcher with the Computer Engineering and Informatics Department, University of Patras, Rio, Greece. His research related experience includes high-speed all-optical logic, optical-signal processing for packet- and burst-switched networks, and high-rate optical sources. He is the Author or Coauthor of more than ten papers in IEEE journals and sponsored conferences.

Dr. Yiannopoulos is a member of the IEEE Lasers and Electro-Optics Society (LEOS) and the IEEE Communications Society. He was one of the recipients of the IEEE LEOS Graduate Student Fellowship Award in 2004.

Kyriakos G. Vlachos (S’00–M’02) received the Dipl.-Ing. degree in electrical and computer engineering and the Ph.D. degree in electrical and computer engineering from the National Technical University of Athens (NTUA), Athens, Greece, in 1998 and 2001, respectively.

From 1997 to 2001, he was a Senior Research Associate with the Photonics Communications Research Laboratory (ICCS/NTUA). In April 2001, he was with Bell Laboratories, Lucent Technologies, working on behalf of the Applied Photonics Group. Since 2005, he has been a Faculty Member with the Computer Engineering and Informatics Department, University of Patras, Rio, Greece. His research interests are in the areas of high-speed protocols and technologies for broadband high-speed networks, optical packet/burst switching, and grid networks. He has participated in various research projects funded by the European Commission (IST-STOLAS, IST-PRO3, ESPRIT-DOALL, e-photon/ONe+, and IST-PHOSPHOROUS). He is the (Co)Author of more than 70 journal and conference publications and is the holder of five patents.

Prof. Vlachos is a member of the Technical Chamber of Greece.

Emmanouel Varvarigos was born in Athens, Greece, in 1965. He received the Diploma in electrical and computer engineering from the National Technical University of Athens in 1988 and the M.S. and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, in 1990 and 1992, respectively.

He has held faculty positions at the University of California, Santa Barbara, from 1992 to 1998, as an Assistant Professor and, later, an Associate Professor, and Delft University of Technology, Delft, The Netherlands, from 1998 to 2000, as an Associate Professor. In 2000, he became a Professor with the Department of Computer Engineering and Informatics, University of Patras, Rio, Greece, where he is currently heading the Communication Networks Laboratory. He is also the Director of the Network Technologies Sector, Research Academic Computer Technology Institute, which, through its involvement in pioneering research and development projects, has a major role in the development of network technologies and telematic services in Greece.