This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

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

Konstantinos Yiannopoulos, Emmanouel Varvarigos, Kyriakos Vlachos, *member IEEE*
Computer Engineering and Informatics Dept and Research Academic Computer Technology Institute,
University of Patras, Rio 26500, GREECE (skype: kvlachos, email: kvlachos@ceid.upatras.gr)

***Abstract*— In this paper, we present the architectural design of optical burst-buffers that can truly emulate input queuing and accommodate asynchronous burst operation. The architectural design uses wavelength converters and fixed feed-forward delay lines that are combined to form either a multiple-input buffer or a shared buffer. Both schemes are modular, allowing the logarithmic expansion of buffer size with the number of switching elements (wavelength converters).**

Optical burst switching; optical buffers; input queuing; wavelength converters; programmable delay.

I. INTRODUCTION

Optical buffering is an important functionality in optical burst-switched networks. It allows the temporal storing of data bursts to resolve contention for the switch outputs. Various solutions have been proposed up to including programmable delay lines [1] and optoelectronic conversion [2]. Electronic buffering has been extensively utilized in currently installed optical networks at a great cost and complexity and is limited by the electronic processing speeds and the relative slow O/E and E/O conversion times. On the other hand, programmable delay lines have been extensively used to form feed-forward or re-circulating schemes, employing in addition wavelength conversion to enhance buffering capabilities [3], [4]. Most schemes, however, assume slotted operation that requires complex scheduling algorithms and thus are not suitable for asynchronous optical burst switching.

In particular, feedback loops theoretically provide infinite storage time, but they suffer from noise accumulation and OSNR degradation. Further, they require that the length of the feedback loop matches exactly the burst duration in order to avoid loss of synchronization and that a large number of them is 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 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 [5], provided some traffic engineering is performed.

For these reasons, feed-forward delay line buffer architectures are a more practical solution for implementing limited optical buffering in burst switched networks. Within this context, we present in this communication an optical burst buffer architecture that is based on the feed-forward-delay-line concept that can truly emulate input queuing and accommodate asynchronous burst operation. The architectural design uses wavelength converters and fixed length delay lines to internally route data. 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 in the sense that buffer size increases logarithmically with the number of programmable delays. Moreover, as we show later on, the use of multiple wavelengths to route internally data bursts minimizes the number of delay stages needed and, thus, the number of active devices. In our analysis, we have assumed a wavelength tuning range of w. The rest of the paper is organized as follows: in Section II the multiple-input buffer design that consists of parallel Time-Slot-Interchangers (TSIs) and emulates input queuing is presented. In Section III we present the shared buffer design that emulates distributed buffering among all incoming / outgoing links of the optical switch. Finally, Section IV concludes the paper.

The work was supported by the Operational Program for Educational and Vocational Training (EPEAEK), PYTHAGORAS II Program and by EU via the IST/NoE e-Photon/ONe+ project.

II. MULTIPLE-INPUT BUFFER ARCHITECTURE

In the current section we discuss the architecture of a feed-forward buffer that is capable of storing on-the-fly optical bursts that arrive at its inputs. Storage is accomplished by delaying the 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 time frames, and bursts are confined within the limits of the time frame they have arrived. We assume that the time frame contains T timeslots and that each burst asynchronously occupies a number of consecutive timeslots. Under this scheme providing variable storage time for the bursts is readily translated to interchanging timeslots. As a result the buffer is equivalent, in terms of functionality, to k parallel Time-Slot-Interchangers (TSIs), one per input port of the buffer, as 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 that WDM offers, with a goal to minimize the number of serially connected stages, and consequently the hardware cost. A tunable wavelength converter (TWC), which is capable of providing w separate wavelengths at its output, is deployed at the input of the respective delay stage. The TWC assigns the bursts to wavelengths based on the delay line that the bursts must access in the delay bank. Mapping between

1-4244-0353-7/07/$25.00 ©2007 IEEE

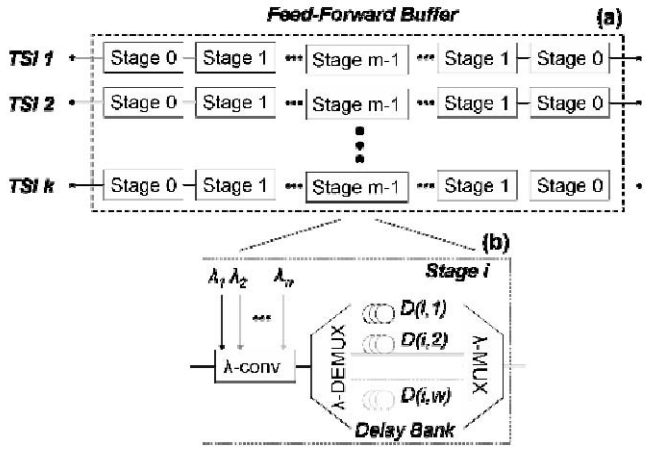

Figure 1. (a) The multiple-input-buffer architecture. (b) The structure of each stage in the input-buffer. λ-conv is the tunable wavelength converter and λ-MUX/DEMUX are the wavelength multi- and demultiplexers.

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 each stage will be derived from the space-time graph of the input-buffer architecture [6] in the subsection that follows.

A. Formation of the space-time graph

The space-time graph that corresponds to the first stage of the TSI (Stage 0) is shown in Fig. 2. As discussed in [6], the space-time graph of the input-buffer consists of nodes located at columns and rows. Columns *i* and *i*+1 represent the inputs and outputs, respectively, of input-buffer stage *i*, while rows account for the timeslots. Nodes that are located at the input of stage *i* connect to nodes at the stage output with time transitions, which are presented as straight lines on the space-time graph. Time transitions inside the input-buffer stage *i* correspond to the burst accessing a delay line, and as a result time transitions may not access output nodes that are located at previous timeslots.

Our goal is to select the time transitions, or equivalently the delay times *D(i,j)*, at each stage so as to construct a $\log_n$-Benes interconnection network on the space-time graph. The purpose of constructing the $\log_n$-Benes network is two-fold: the network requires a minimum number of serially connected stages that equals

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

for a given number *T* of timeslot per time frame, where *n* is the $\log_n$-Benes crossbar size. The network is also re-arrangably non-blocking, therefore the input-buffer is capable of switching without internal collisions. Eq. (1) shows that by constructing the $\log_n$-Benes network, one can achieve a drastic reduction in the number of stages, as compared to previously reported work that was based on $\log_2$-Benes networks [7], [8]. Furthermore, finding a collision free path within the Benes network is a well studied problem [9].

The building blocks of the $\log_n$-Benes network are *n* x *n* crossbar switches, and thus the first step in constructing it is to

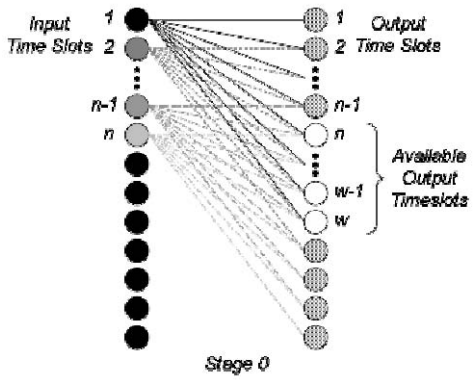

Figure 2. Derivation of the crossbar size on the space-time graph.

determine the size of the crossbars. The crossbars are formed out of time transitions on the space-time graph, as is shown in Fig. 2. Since only downward time transitions are allowed at the space-time graph, the output timeslots that are available to all *n* input timeslots are limited to *w*-*n*+1. Since the inputs of the crossbar equal its outputs, we find that

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

with $\lfloor x \rfloor$ denoting the integer part of *x*. As a result, approximately 50% of the available wavelengths contribute to the formation of the crossbars that comprise the $\log_n$-Benes, even though all available wavelengths are required to route packets inside the buffer.

The second step is to determine the time transitions that construct the $\log_n$ -Benes network, and 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)-(c) where the network of Fig. 3(a) is transformed to a standard representation. The construction of the $\log_n$-Benes network requires that at each stage *i*, the crossbars are formed between timeslots that are located n^i positions apart, as is illustrated in the equivalent network representations of Fig. 3(b)-(c). This corresponds to setting the switch time delays, in timeslots, equal to [7]:

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

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 inactive transitions introduce a constant delay after which the output time frame commences (white squares in Fig. 3(a)). At the output of each stage, the delay equals

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

timeslots and as a result the total delay that the bursts experience when traversing the buffer is

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

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

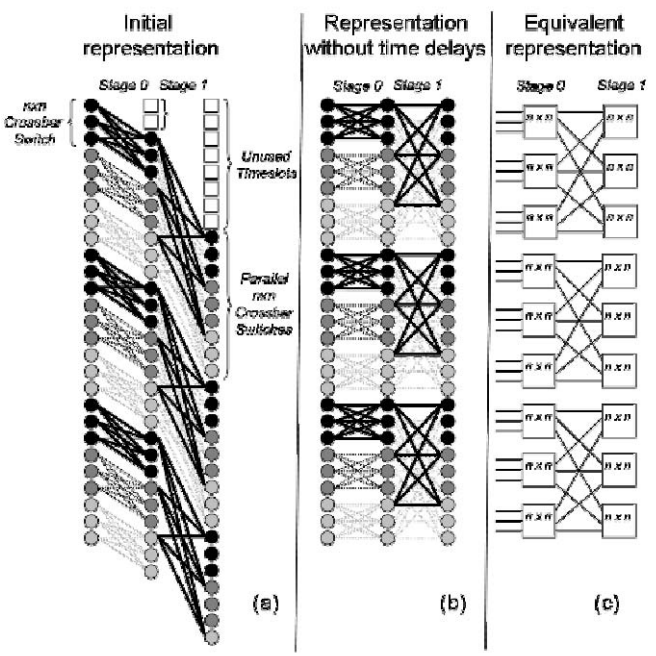

Figure 3. Formation of the $\log_n$-Benes network on the space-time graph.

B. *Asynchronous operation in an OBS node*

The proposed buffer may be cascaded by a space switch to form the switching core of an OBS node [8]. Even though the $\log_n$-Benes interconnection network ensures that there are no collisions inside the TSIs, collisions may still occur at the outputs of the space switch if two or more bursts require accessing the same switch output at the same time. Losses are avoided by ensuring that the respective timeslots do not coincide at the space switch, and this is systematically achieved by means of the scheduling algorithm that we illustrate in Fig. 4. According to the scheduling algorithm, we partition the time frame into k successive sub-frames of equal duration. We then label the sub-frames at a buffer output p with a (p-1)-times cyclic shift of $\{1,2,\ldots,k\}$ and assign input timeslots that head for a switch output q to the respective output sub-frame. Since sub-frame q never occurs simultaneously at two buffer outputs, losses are avoided provided that the total duration of bursts (measured in timeslots) that arrive at buffer input p and are destined for switch output q is limited by

$$d_{pq} \le \frac{T}{k}, \quad 1 \le p,q \le k. \tag{6}$$

After mapping the timeslots at the inputs to timeslots at the outputs of the buffers, it is necessary to determine the delays that have to be introduced to the content of the input timeslots so as to appear at the respective assigned timeslots at the output of the buffer. This is equivalent to routing the timeslots on the $\log_n$-Benes interconnection network of Fig. 3 and is achieved by means of a parallel Benes routing algorithm. The algorithm is an extension of a low complexity $\mathrm{O}\left(\log^2 T\right)$ parallel routing algorithm on a binary Benes network [9], which involves setting the state of the outermost crossbars (at stages 0 and s-1) of the Benes network after solving permutation vector

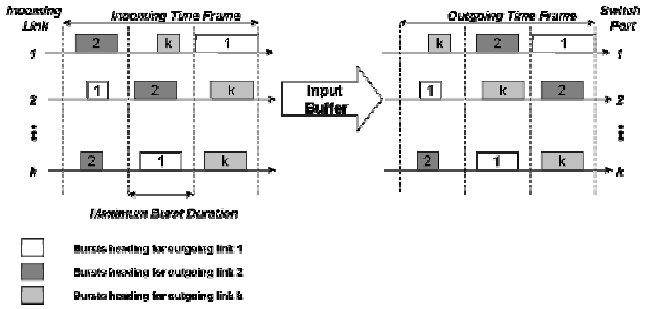

Figure 4. Timeslot assignment at the multiple-input buffer architecture.

equations. The outermost crossbars are then omitted, and the remaining network is partitioned into a number of Benes sub-networks of reduced size. The algorithm is recursively applied on the resulting sub-networks until the state of all crossbars in the networks is set. An example of the routing algorithm is detailed in the following subsection.

C. *Benes routing in the $\log_n$-Benes network*

A routing example for the $\log_n$-Benes network of Fig. 3 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 consideration which output is assigned to each input timeslot. In the example of Fig. 5 the input and output permutation vectors are

$$\begin{aligned} \pi_{in} &= (00 \quad 01 \quad 02 \quad 10 \quad 11 \quad 12 \quad 20 \quad 21 \quad 22) \\ \pi_{out} &= (02 \quad 21 \quad 22 \quad 20 \quad 00 \quad 11 \quad 01 \quad 10 \quad 12). \end{aligned} \tag{7}$$

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

$$\pi_0 = (0a_8 \quad 0a_7 \quad 0a_6 \quad 1a_5 \quad 1a_4 \quad 1a_3 \quad 2a_2 \quad 2a_1 \quad 2a_0). \tag{8}$$

This corresponds to time transitions for which α_i denote the output nodes that have been accessed. In a similar fashion, the output permutation vector at the input of stage s is

$$\pi_2 = (0b_2 \quad 2b_7 \quad 2b_8 \quad 2b_6 \quad 0b_0 \quad 1b_4 \quad 0b_1 \quad 1b_3 \quad 1b_5) \tag{9}$$

with b_i referring the input nodes that have been accessed by the inverse time transitions. In Eq. (9), b_i are assigned to rows according the output permutation vector, due to the symmetry of the $\log_n$-Benes network. Moreover, the symmetry of the network imposes that α_i and b_i that are located in a common row are equal, and as a result T equations that correlate α_i and b_i are derived. The equations are solved after taking into consideration that α_i (and b_i) satisfy

$$a_{m \cdot n+i} \ne a_{m \cdot n+j}, \; i \ne j, \quad i,j \in \{0,1,\ldots,n-1\} \tag{10}$$

so that no collisions occur inside the crossbars. Following (10), the solution to (8) and (9) is calculated as

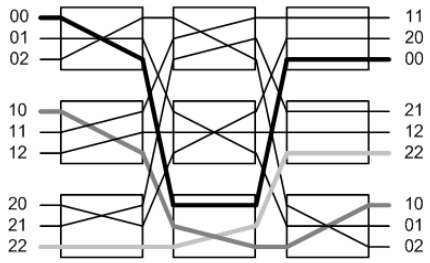

Figure 5. Routing in the $\log_n$-Benes network.

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

After solving the equations for the outermost stages of the $\log_n$-Benes network, we remove the aforementioned stages and divide the remaining network into three (n in general) sub-networks. The permutation vectors of each sub-network are derived from (11) after grouping together the vector elements that have a common least significant symbol

$$\pi_{in}^0 = (0 \quad 1 \quad 2) \rightarrow \pi_{out}^0 = (2 \quad 0 \quad 1)$$
$$\pi_{in}^1 = (0 \quad 1 \quad 2) \rightarrow \pi_{out}^1 = (2 \quad 1 \quad 0) \tag{12}$$
$$\pi_{in}^2 = (0 \quad 1 \quad 2) \rightarrow \pi_{out}^2 = (0 \quad 2 \quad 1).$$

No further permutation vectors have to be evaluated for the specific example, since the permutation vectors of (11) and (12) suffice to define the state of the crossbars at all stages. The state of a crossbar at stage l is set after isolating the permutation vector elements that correspond to the specific crossbar and taking into account their least significant symbols. We assume that the aforementioned symbols form the reduced permutations vectors ρ_l^{in} and ρ_l^{out} at the input and output of each stage; for instance these are

$$\rho_0^{in} = (0 \quad 1 \quad 2 \quad 0 \quad 1 \quad 2 \quad 0 \quad 1 \quad 2)$$
$$\rho_0^{out} = (2 \quad 1 \quad 0 \quad 2 \quad 0 \quad 1 \quad 1 \quad 0 \quad 2) \tag{13}$$

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

$$d_l = \Delta_l + \left(\rho_l^{out} - \rho_l^{in}\right) \cdot n^l, \quad 0 \le l < s \tag{14}$$

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

$$w_l = n + \rho_l^{out} - \rho_l^{in}, \quad 0 \le l < s. \tag{15}$$

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, but it requires the 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 than (6) for lossless operation. The proposed design, which is shown in Fig. 6(a), consists of serially connected delay stages that are accessible by all input ports, as detailed in Fig. 6(b). At each delay stage, k parallel wavelength converters assign the incoming bursts to wavelengths that correspond to a pair of delay lines and output ports. The delay lines and output ports are accessed by the bursts through all-passive space switches. Similar to Section II, our goal is to define the delays $D(i,j)$ that are introduced at each stage so that an optimal interconnection network is constructed on the space-time graph.

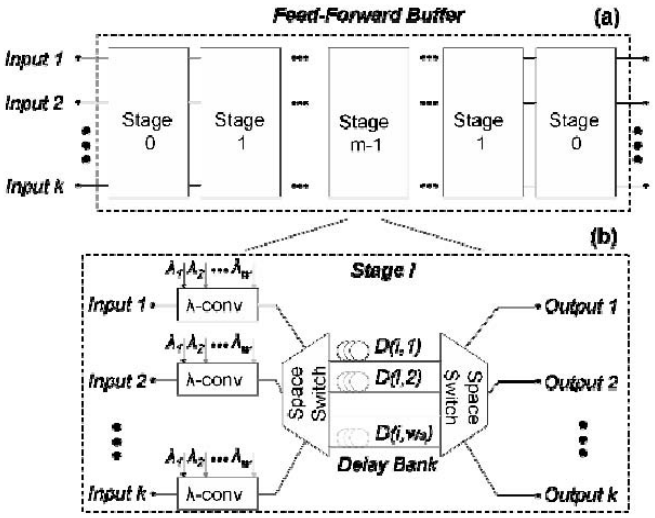

Figure 6. (a) The shared buffer architecture. (b) At the respective stages, each wavelength is assigned to a pair of delays and output ports.

A. Formation of the space-time graph

The space-time graph of the shared buffer architecture is illustrated in Fig. 7. In contrast to the space-time graph we discussed in Section II, 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 of Fig. 7. All transitions between the input and output space nodes of a delay stage are valid within a timeslot, since all outputs of stage may be accessed by any stage input in Fig. 6(b). Time transitions are limited only to nodes that correspond to the current timeslot or future ones. For the rest of this section we consider only time transitions and timeslot nodes on the space-time graph, so as to simplify the illustration of our analysis. However, during the time transitions between input and output timeslot nodes, we assume that all k space input and output nodes that lie within the respective timeslots are connected with space transitions. This is equivalent to constructing the interconnection network on the time transitions of the space-time graph, and afterwards expanding the crossbars and connections of the resulting network by a factor of k.

Within this context, our goal is to construct a $\log_d$-Benes interconnection network on the time transitions of the space-time graph that corresponds to the shared buffer architecture. The procedure is quite similar to that described in the previous section: we first determine the size d x d of the elementary crossbar and based upon this, we form the $\log_d$-Benes network on the space-time graph. The number w_a of timeslots on the space-time graph that are fully accessed at the output of the current stage equals the wavelength tunability w normalized by

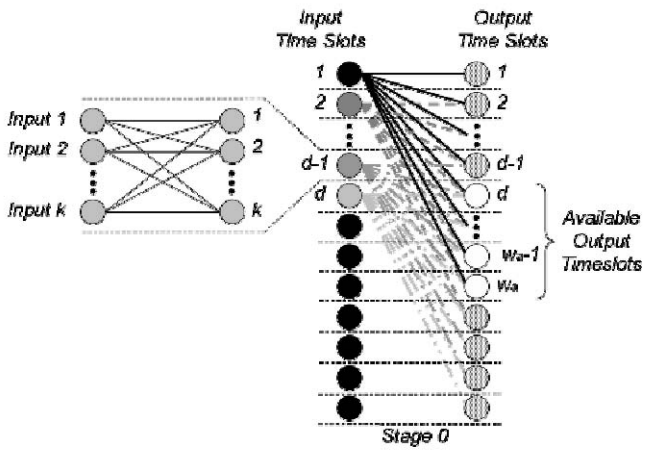

Figure 7. Derivation of the elementary crossbar on the space-time graph. Each node of the space-time graph representing timeslots is expanded to k separate nodes that corresponds to the input/output ports.

the number of ports k

$$w_a = \left\lfloor \frac{w}{k} \right\rfloor \tag{16}$$

and as a result, the size of the crossbars is calculated as

$$d = \left\lfloor \frac{w_a + 1}{2} \right\rfloor = \left\lfloor \frac{\left\lfloor \frac{w}{k} \right\rfloor + 1}{2} \right\rfloor. \tag{17}$$

The $\log_d$-Benes network is formed as in Fig. 8(a)-(c), by setting the delays equal to

$$D(i, j) = j \cdot d^i, \quad i = 0, \ldots, m-1, \quad j = 0, \ldots, w_a - 1. \tag{18}$$

Similar to the multiple-input buffer design, only d time transitions are utilized per timeslot, and as a result the bursts experience a storage latency that is calculated in timeslots as

$$\Delta = \sum_{i=0}^{m-1} d^i \cdot (d-1) + \sum_{i=0}^{m-2} d^i \cdot (d-1) = T + \frac{T}{d} - 2. \tag{19}$$

The fully expanded network that includes space transitions is derived in Fig. 8(c) after taking into consideration that the size of the crossbars becomes $n = k \cdot d$ and that each time transition corresponds to k space transitions. The fully expanded network is re-arrangably non-blocking and thus the shared buffer architecture performs buffering without internal losses. The number of delay stages that are required is given by

$$s = 2 \cdot m - 1 = 2 \cdot \lceil \log_d T \rceil - 1. \tag{20}$$

The number of delayed stages is therefore not optimal, and this is because we have constructed a $\log_d$-Benes network on the time transitions instead of a $\log_n$-Benes networks on both time and space transitions. However, this drawback of the shared buffer architecture is balanced by the fact that it requires less strict traffic conditions to achieve lossless operation, as we show in the following subsection.

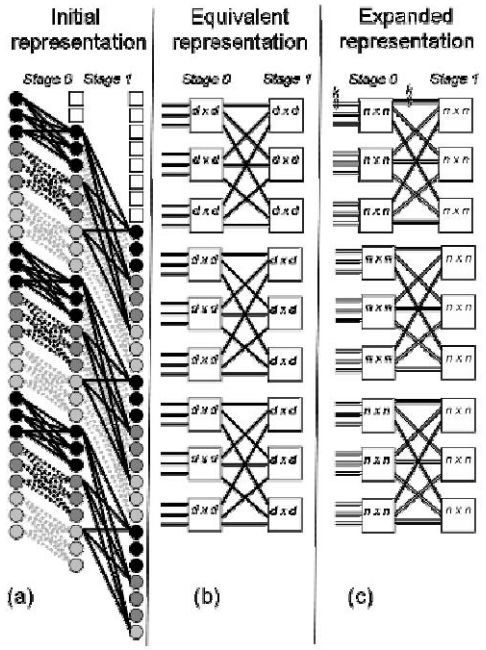

Figure 8. Formation of the $\log_d$-Benes network on the time transitions of the space-time graph.

B. Asynchronous operation in an OBS node

The shared buffering architecture may be readily deployed in an OBS node without cascading the space switch that is required by the multiple-input buffer, since buffering and output port assignment are performed independently. Moreover, the re-arrangably, non-blocking property of the expanded interconnection graph of Fig. 8(c) ensures that no burst collisions take place inside the shared buffer, provided that the total traffic that arrives to all buffer inputs and heads for a specific buffer output does not exceed T timeslots within a time frame. This is a looser traffic condition than (6), since incoming traffic does not have to be equally distributed among all buffer inputs.

Bursts that arrive within the same time frame are placed on a common outgoing frame, which commences after Δ timeslots. The timeslots that the bursts occupy at the outgoing time frame are assigned according to a modification of the packing rule [8]. The modified packing rule is illustrated in Fig. 9, according to which, bursts heading 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, if it is the r-th timeslot that has arrived at incoming link p and heads for outgoing link q. A timeslot at the incoming time frame with rank r will be mapped at the output time frame to timeslot

$$y = \sum_{l=1}^{p-1} n_{l,q} + r - 1, \quad y \in \{0, \ldots, T-1\}, \tag{21}$$

where $n_{l,q}$ is the total duration of bursts, in timeslots, between incoming link l and outgoing link q. After timeslot assignment,

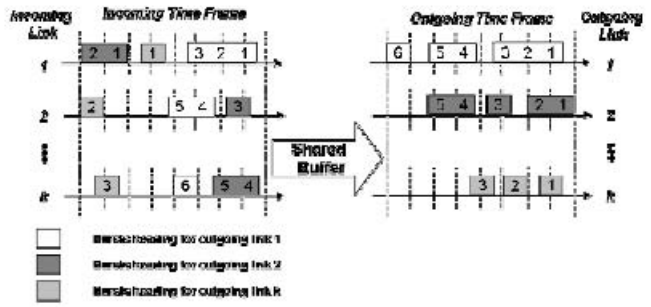

Figure 9. Timeslot assignment in the shared buffer architecture.

a modified parallel Benes routing algorithm sets the delays that timeslots experience inside the buffer stages, similar to Section II. We detail an example in the expanded $\log_d$-Benes network of Fig. 8 in the next subsection.

C. *Benes routing for the $\log_d$-Benes network*

A routing example for $n = 4$ ($k = 2$, $d = 2$) and $T = 4$ in the $\log_d$-Benes network is detailed in Fig. 10. The input and output permutation vectors are given by

$$\begin{aligned} \pi_{in} &= (00 \quad 01 \quad 02 \quad 03 \quad 10 \quad 11 \quad 12 \quad 13) \\ \pi_{out} &= (00 \quad 01 \quad 03 \quad 11 \quad 02 \quad 10 \quad 12 \quad 13). \end{aligned} \tag{22}$$

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

$$\begin{aligned} \pi_0 &= (0a_0 \quad 0a_1 \quad 0a_2 \quad 0a_3 \quad 1a_4 \quad 1a_5 \quad 1a_6 \quad 1a_7) \\ \pi_2 &= (0b_0 \quad 0b_1 \quad 0b_3 \quad 1b_5 \quad 0b_2 \quad 1b_4 \quad 1b_6 \quad 1b_7). \end{aligned} \tag{23}$$

Eq. (23) is solved for a_i and b_i using the symmetry properties of the expanded $\log_d$-Benes network, after taking into account (10). 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

$$a_{m \cdot n+i} \leftrightarrow a_{m \cdot n+j},\ i \neq j, \quad i, j \in \{0,1,\ldots,k-1\}. \tag{24}$$

A solution to our example is

$$\begin{aligned} \pi_0 &= (00 \quad 01 \quad 02 \quad 03 \quad 13 \quad 12 \quad 11 \quad 10) \\ \pi_2 &= (00 \quad 01 \quad 02 \quad 13 \quad 03 \quad 12 \quad 11 \quad 10). \end{aligned} \tag{25}$$

We then omit the outermost crossbars and divide the resulting network into two (d in general) sub-networks. The permutation vectors for the new networks are formed after grouping together the vector elements which have least significant symbols z that are located between

$$z \in \left[i \cdot k, \quad (i+1) \cdot k - 1 \right], \quad 0 \le i < d. \tag{26}$$

In the example of Fig. 10 we find that

$$\begin{aligned} \pi_{in}^0 &= (00 \quad 01 \quad 11 \quad 10) \rightarrow \pi_{out}^0 = (00 \quad 01 \quad 11 \quad 10) \\ \pi_{in}^1 &= (02 \quad 03 \quad 13 \quad 12) \rightarrow \pi_{out}^1 = (02 \quad 13 \quad 03 \quad 12) \end{aligned} \tag{27}$$

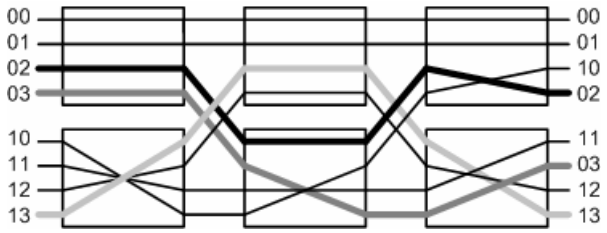

Figure 10. Routing in the expanded $\log_d$-Benes network.

Eq. (27) are readily solved after re-numbering the vector elements with respect to their order

$$\begin{aligned} \pi_{in}^0 &= (0 \quad 1 \quad 3 \quad 2) \rightarrow \pi_{out}^0 = (0 \quad 1 \quad 3 \quad 2) \\ \pi_{in}^1 &= (0 \quad 1 \quad 3 \quad 2) \rightarrow \pi_{out}^1 = (0 \quad 3 \quad 1 \quad 2). \end{aligned} \tag{28}$$

Eq. (25) and (28) define the state of all crossbars in the network. Having determined the state of the crossbars, we calculate the reduced permutation vectors ρ_l^{in} and ρ_l^{out} at each stage and the respective wavelengths are derived from Eq. (15).

IV. CONCLUSION

We have presented the architectural design of two optical burst buffers. Both designs use wavelength converters and fixed delay lines that are combined to form a multiple-input buffer or a shared buffer. The designs are modular, allowing for the logarithmic expansion of buffer size with the number of switching elements (wavelength converters). Wavelength parallelism is used to achieve a significant decrease in the total number of delay stages needed, as compared to previous work. Furthermore, we have also proposed architecture-specific algorithms for providing contention resolution within the buffering time, as well as algorithms for scheduling the internal wavelengths.

REFERENCES

[1] D.K. Hunter et al., "Buffering in optical packet switches," IEEE/OSA J. Lightw. Technol., vol. 16, no. 12, pp. 2081-2094, Dec. 1998.

[2] S. Bjørnstad, N. Stol, D. R. Hjelme, "An Optical Packet Switch Design with Shared Electronic Buffering and Low Bit Rate Add/Drop Inputs," in Proc. ICTON 2002, pp. 69-72.

[3] 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," IEEE/OSA J. Lightwave Technol., vol. 19, no. 9, pp. 1241-1254, Sept. 2001.

[4] C.M. Gauger, "Dimensioning of FDL buffers for optical burst switching nodes," in Proc. ONDM 2002, pp. 202-221.

[5] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Routers with Very Small Buffers," in Proc. IEEE INFOCOM 2006.

[6] D.K. Hunter, and D.G. Smith, "New architectures for optical TDM switching," IEEE/OSA J. Lightwave Technol., vol. 11, no. 3, pp. 495-511, Mar. 1993.

[7] D.K. Hunter et al., "SLOB: A Switch with Large Optical Buffers for Packet Switching," IEEE/OSA J. Lightwave Technol., vol. 16, no. 10, pp. 1725-1736, Oct. 1998.

[8] E. A. Varvarigos, "The 'Packing' and the 'Scheduling' packet switch architectures for almost all-optical lossless networks," IEEE/OSA J. Lightwave Technol., vol. 16, no. 10, pp. 1757–67, Oct. 1998.

[9] T.T. Lee et al., "Parallel Routing Algorithms in Benes-Clos Networks," IEEE Trans. Commun., vol. 50, no. 11, Nov. 2002, pp. 1841–47.