Optimal Burst Scheduling in Optical Burst Switched Networks

Yuhua Chen, *Member, IEEE*, Jonathan S. Turner, *Fellow, IEEE*, and Pu-Fan Mo

***Abstract*—Optical burst switching (OBS) is an emerging technology that allows variable size data bursts to be transported directly over dense wavelength division multiplexing links. In order to make OBS a viable solution, the burst-scheduling algorithms need to be able to utilize the available wavelengths efficiently, while being able to operate fast enough to keep up with the burst incoming rate. For example, for a 16-port OBS router with 64 wavelengths per link, each operating at 10 Gb/s, we need to process one burst request every 78 ns in order to support an average burst length of 100 kB. When implemented in hardware, the well-known horizon scheduler has O(1) runtime for a practical number of wavelengths. Unfortunately, horizon scheduling cannot utilize the voids created by previously scheduled bursts, resulting in low bandwidth utilization. To date, minimum starting void is the fastest scheduling algorithm that can schedule wavelengths efficiently. However, while its complexity is $O(\log\ m)$, it requires $10\log\ m$ memory accesses to schedule a single burst. This means that it can take up to several microseconds for each burst request, which is still too slow to make it a practical solution for OBS deployment. In this paper, we propose an optimal burst scheduler using constant time burst resequencing (CTBR), which has O(1) runtime. The proposed CTBR scheduler is able to produce optimal burst schedules while having processing speed comparable to the horizon scheduler. The algorithm is well suited to high-performance hardware implementation.**

***Index Terms*—Algorithm, optical burst switching (OBS), optical packet switching, scheduling, wavelength division multiplexing (WDM), wavelength routing.**

I. Introduction

ADVANCES in dense wavelength division multiplexing (DWDM) technology allow tens or hundreds of DWDM channels to be carried over a single optical fiber at 10 Gb/s per channel. This means that a router has to be able to process data rates up to 10 Tb/s (terabits per second) per port, which is beyond the capability of the electronic routers. It is conceivable that optical switching technologies will eventually replace electronic switching technologies, in order to take advantage of the enormous amount of bandwidth made possible by DWDM technologies.

Optical burst switching (OBS) [1]–[5] has emerged as a promising candidate for future all optical Internet. In OBS networks, variable-size optical data bursts can be transported directly over DWDM without converting back to electronic form. A burst header is sent on a separate control channel shortly before the transmission of the data burst. Burst headers set up lightpaths on-the-fly, allowing data bursts to remain in the optical domain and pass through OBS routers without encountering optical–electrical–optical (O/E/O) conversion.

One of the key design issues in OBS networks is WDM channel scheduling. In order to make OBS a practical solution, we need to solve the following two problems at the same time: 1) how to design channel-scheduling algorithms that can utilize the available wavelengths efficiently and 2) how to make the algorithm fast enough so that the scheduler can keep up with the burst incoming rate. For example, for a 16-port OBS router with 64 wavelengths per link, each operating at 10 Gb/s, we need to process a burst request every 78 ns in order to support an average burst size of 100 kB.

Several scheduling algorithms have been proposed for OBS routers [1], [3], [5]–[9]. Horizon scheduling [1], [3] provides fast burst scheduling and can achieve O(1) operation in hardware. However, it can cause excessive burst discard since it cannot utilize the voids created by previously scheduled bursts. Latest available unused channel with void filling (LAUC-VF) [5] can produce efficient channel schedules, but it takes $O(m)$ time to schedule a burst, where m is the number of voids per channel. The minimum starting void (Min-SV) algorithm [6], [7] can produce efficient burst schedules as LAUC-VF. The complexity of Min-SV is $O(\log\ m)$, which is a significant improvement over LAUC-VF. However, Min-SV still requires $10\log(m)$ memory accesses for each burst request. It is not unusual that a system will have to keep track of 100 k to 1 million voids. This means that Min-SV can take up to a few microseconds to schedule a single burst, which is still too slow to meet the stringent burst-scheduling requirement.

In this paper, we propose a novel hardware-based scheduling algorithm that combines the horizon scheduler with O(1) runtime constant time burst resequencing (CTBR). The resulting CTBR scheduler is able to produce optimal burst schedules while being able to operate at speed comparable to the horizon scheduler. The proposed solution runs much faster than Min-SV and is significantly simpler than Min-SV in terms of implementation.

The rest of this paper is organized as follows. Section II gives an overview of optical switching technologies. Section III describes the OBS network architecture. Existing scheduling algorithms are discussed in Section IV. The optimal-burst-scheduling algorithm is proposed in Section V. We conclude this paper in Section VI.

Manuscript received October 14, 2006; revised April 13, 2007.

Y. Chen and P.-F. Mo are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204-4005 USA (e-mail: yuhua.chen@mail.uh.edu; pmo@uh.edu).

J. S. Turner is with the Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130 USA (e-mail: jst@cse.wustl.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JLT.2007.899785

II. Optical Switching Technologies

A. Wavelength Routing

In wavelength routing (optical circuit switching), an optical circuit (lightpath) is set up before data are transmitted. The lightpath will be held for the entire duration of the transmission. The selected wavelength is considered occupied, even if there are no data transmitted over the link. Therefore, it will result in poor wavelength utilization in facing the bursty nature of the Internet traffic.

In addition, since data transmission can only begin after an end-to-end lightpath is set up, wavelength routing encounters at least one round-trip time delay. Although wavelength routing plays a major role in current generation optical networks, it is not considered the most appropriate technology for the emerging optical Internet.

B. Optical Packet Switching

Optical packet switching [10]–[16] allows optical packets from different sources to share the wavelengths, achieving statistical multiplexing performance. However, optical packet switching technology is not mature enough to provide a viable solution. The major barriers are as follows: 1) synchronization and 2) lack of optical random access memory (RAM).

For example, because of the extremely limited capability of optical packet header processing, in optical packet switching, the packet header is usually tapped off and processed electronically at the routers. After the header is processed, it needs to be converted back to optical signal and combined with the optical payload. Synchronizing the header and the payload presents a technical challenge.

Moreover, packet switched networks are store and forward networks, where packets are stored in the switches before forwarding due to output contention. This technique is widely used in electronic routers where electronic RAM is abundant and cheap. Unfortunately, there is no equivalent optical RAM. Therefore, optical packet switching can only utilize fiber delay lines (FDLs) to provide a limited amount of fixed time delay, which degrades the performance of optical packet switching networks.

C. Optical Burst Switching (OBS)

In OBS networks, bursts from different sources can be dynamically multiplexed onto the same wavelength, providing statistical multiplexing performance without encountering the technical barriers faced by optical packet switching.

In OBS networks, burst headers and data bursts are sent on separate DWDM channels. At an OBS router, only the burst headers on the control channel are converted back to electronic signal and processed electronically. Based on the information carried in the burst header, which arrives ahead of its associated data burst, the OBS router dynamically sets up a lightpath right before the arrival of the data burst and tears down the lightpath after the data burst passes through. Data bursts can stay in the optical domain and passes through the OBS router transparently.

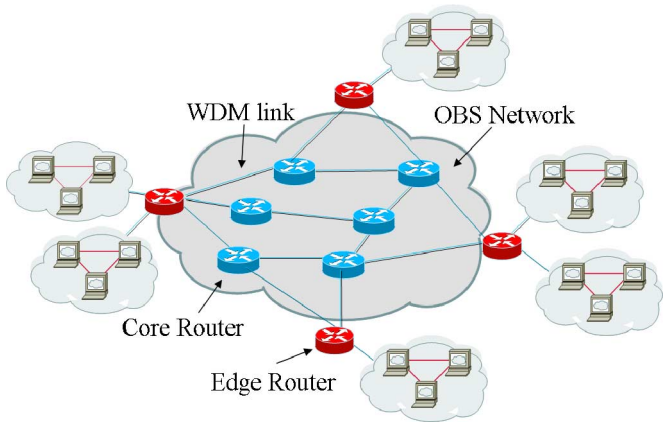

Fig. 1. OBS router architecture.

Due to the separation of burst headers and data bursts, OBS does not require tight synchronization between the header and the data burst. In addition, since a lightpath is set up before the arrival of the data burst, there is no need for optical buffers in OBS networks.

III. OBS Network Architecture

Fig. 1 illustrates the basic concept for an OBS network. The network consists of a set of OBS routers connected by DWDM links. The transmission links in the system carry tens or hundreds of DWDM channels, any one of which can be dynamically assigned to a user data burst. One (or possibly more than one) channel on each link is used as a control channel to control the dynamic assignment of the remaining channels to data bursts.

There are two ways to send data through an OBS network. The first option is through electronic edge routers. Edge routers provide legacy interfaces (e.g., IP, Gigabit Ethernet, SONET) and burst assembly/disassembly functionality. The second option is to have end systems interface directly with OBS routers through network interface cards. The format of data carried in bursts is not constrained by the OBS systems. Data bursts may be IP packets, Ethernet packets, or raw bit streams.

An OBS network works as follows. Shortly before the transmission of a data burst on a data channel, a burst header, which we call burst header cell (BHC), is sent on the control channel, specifying the channel on which the burst is being transmitted and the destination of the burst. The BHC also carries an offset field and a length field. The offset field defines the time between the transmission of the first bit of the BHC and the first bit of the data burst. The length field specifies the time duration of the burst. The offset and the length fields are used by the OBS routers to schedule the setup and release of optical data paths. Fig. 2 shows an example of BHCs sharing the same control channel, while the corresponding data bursts are sent on separate data channels.

An OBS router, on receiving a BHC, selects an idle channel on the outgoing link leading toward the desired destination. Shortly before the arrival of the data burst, the OBS router establishes a lightpath between the incoming channel, on which the burst is arriving, and the outgoing channel selected to carry

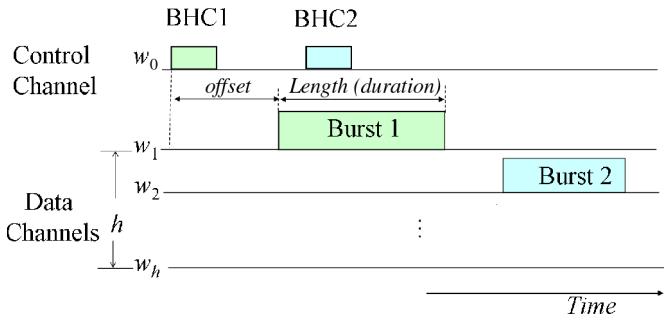

Fig. 2. Bursts and BHCs.

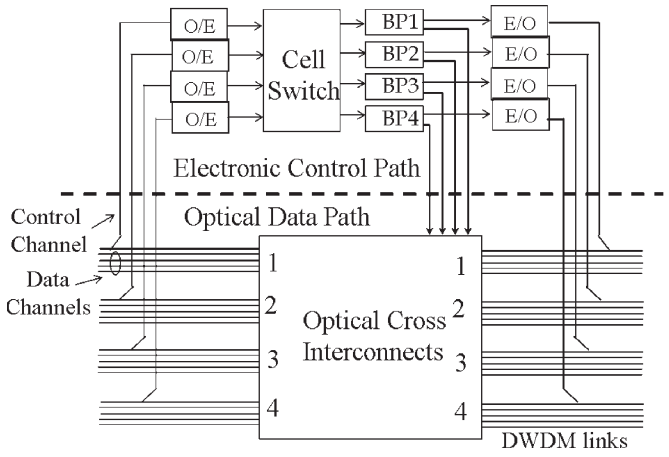

Fig. 3. OBS router architecture.

the burst. The data burst can stay in the optical domain and flow through the OBS router to the proper outgoing channel.

The OBS router forwards the BHC on the control channel of the outgoing link after modifying the channel field to specify the selected outgoing channel. By modifying the offset field appropriately, the OBS routers can account for variable delays, which the BHC experiences within the control subsystem. This process is repeated at every OBS router along the path to the destination.

Fig. 3 shows the key components of an OBS router. The architecture consists of two parts: an optical datapath and an electronic control path. The datapath has optical interconnects with wavelength conversion capability. The control path includes O/E/O conversion, a cell switch, and a set of burst processors (BPs). Each BP is responsible for making channel-scheduling decisions for a single outgoing link. The cell switch routes the BHCs received on the control channels of the incoming DWDM links to the corresponding BP according to the destination of the data burst. The BP selects an outgoing channel for the burst and configures the optical switching matrix such that the bursts arriving on incoming data channels can pass through to the desired outgoing channels directly without buffering.

We use the term BHCs to denote burst headers for convenience purposes. In practice, burst headers can take any format that is supported by the electronic control path in an OBS router. For example, the burst header can be an IP packet, in which case, the cell switch in Fig. 3 will be replaced with an IP router.

IV. Background on Scheduling Algorithms

Switching matrix scheduling algorithms have been extensively studied for electronic switches [17]–[19]. However, they are similar to the burst-scheduling algorithms in the OBS routers only in a sense that the goals of the both types of scheduling algorithms are to obtain the switching matrix configurations such that input traffic can be efficiently sent to the desired outputs. There are two fundamental differences between the switching algorithms designed for electronic switches and the ones needed for OBS routers.

First, all scheduling algorithms designed for electronic switches rely on RAMs to buffer data waiting to be scheduled. One property of electronic RAMs is that once the data are stored in the RAM, it can stay there until it is retrieved. Unfortunately, RAM is not available in the optical domain. Although FDLs can provide a limited time delay, the amount of delay is determined by the length of the FDL. In OBS systems that do not have FDLs, data have to be discarded if they cannot be forwarded to the desired output at the time of arrival.

Second, most of the switching matrix scheduling algorithms developed for electronic switches can only handle a small number of switching ports (32 for example). However, for an OBS router, each port can carry tens or hundreds of channels. Letting d be the number of ports in an OBS router and h be the number of DWDM channels per port, the effective size of the switching matrix is $dh \times dh$. For example, if $d = 32$ and $h = 64$, the effective size of the switching matrix is 2048×2048, which exceeds the capability of almost all scheduling algorithms proposed for electronic switches.

Therefore, existing scheduling algorithms designed for electronic switches cannot be applied to OBS routers. New scheduling algorithms have to be invented in order to handle the large number of DWDM channels in OBS networks.

To date several algorithms have been proposed to solve the wavelength scheduling problem in the OBS networks The major results concerning practical implementations are summarized below.

A. Horizon Scheduling

Horizon scheduling is a practical scheduling algorithm proposed for OBS networks [1], [3]. The horizon for a channel is defined as the latest time at which the channel is currently scheduled to be in use. Given this information, the horizon scheduler simply selects the channel with the latest horizon from a set of channels whose scheduling horizons are smaller than the burst's arrival time. Once a channel has been selected, the scheduler updates the scheduling horizon to be equal to the time when the burst is due to be completed (determined by the offset and length fields in the BHC). If no channels have horizons that are smaller than the arrival time of the burst, then the burst is discarded.

Theoretically, horizon scheduling takes $O(\log h)$ time to schedule a burst, where h is the number of DWDM channels per link. However, when implemented in hardware, a horizon scheduler can find a burst schedule in O(1) time for practical values of h. The hardware implementation of horizon scheduling with O(1) runtime is explained in detail in

Section V-F. A pipelined horizon scheduler design that can schedule a burst every two clock cycles, regardless of the number of channels per link can be found in [20].

However, since horizon scheduling only keeps track of a single state for each channel, it cannot utilize the voids created by previously scheduled bursts. Therefore, horizon scheduling can cause excessive burst discards when the variation of the offset between the BHCs and the bursts is large.

B. Latest Available Unused Channel With Void Filling (LAUC-VF)

LAUC-VF was proposed in [5]. LAUC-VF keeps track of all voids on the channels and tries to schedule a burst in one of the voids whenever possible. If more than one void can fit a burst, the one with the latest beginning time is assigned to the burst. Since LAUC-VF can use the voids created by previously scheduled bursts, link utilization of LAUC-VF is higher than that of horizon scheduling.

However, LAUC-VF takes much longer to schedule a burst compared to horizon scheduling. The complexity of LAUC-VF is O(m), where m is the number of voids. It is common that a system needs to keep track of 100 k to 1 million voids. In general, LAUC-VF is too slow to be practical.

C. Minimum Starting Void (Min-SV)

Min-SV [6], [7] uses a geometric approach and organizes the voids into a balanced binary search tree. Min-SV algorithm finds a void that minimizes the distance between the starting time of the void and the starting time of the burst. The Min-SV algorithm takes O($\log\ m$) time to finish, which is a significant improvement over LAUC-VF. To date, it is the fastest scheduling algorithm that can produce an efficient burst schedule.

However, in order to schedule a burst, Min-SV needs to performance $10 \log\ m$ memory accesses for each burst-scheduling request, which means that it can take up to a few microseconds to schedule a single burst. Therefore, Min-SV is still too slow to provide a practical solution to the problem.

V. Optimal Wavelength Scheduling

A. Constant Time Burst Resequencing (CTBR)

In this section, we propose an optimal wavelength scheduler that can produce optimal burst schedules in O(1) runtime.

We use the idea that rather than processing bursts as soon as BHCs arrive, one can delay the scheduling of the bursts and, then, process them on the order of the expected burst arrival time [3]. In other words, BHCs are processed on the order of burst arrival times, not on the order of the arrival times of the BHCs. This can be achieved by passing BHCs through a burst resequencing buffer and holding them there for Δ time units before the expected burst arrival time. For example, if a burst is expected to arrive at time t, the BHC stays in the resequencer until time $t - \Delta$. Once BHCs are resequenced, it is processed by a horizon scheduler.

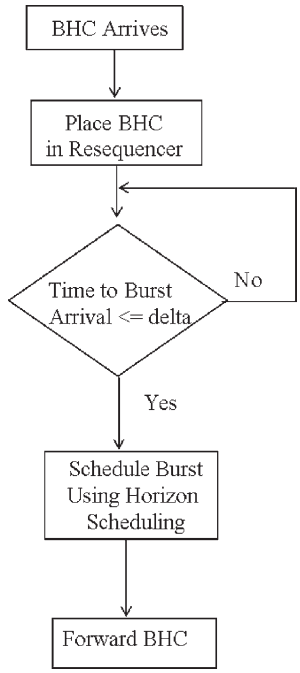

Fig. 4. Burst resequencing flow chart.

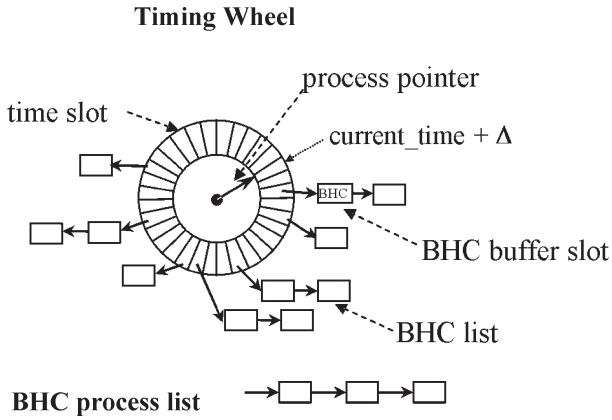

Fig. 5. CTBR data structure.

We need two components to achieve optimal wavelength scheduling in O(1) time, namely, the constant time burst resequencer and the horizon scheduler. The burst resequencer sorts the BHCs on the order of the burst arrival times and forwards the BHCs to the scheduler when the time difference between the current time and the burst start time is equal to or less than Δ time units. The horizon scheduler then chooses an available wavelength to carry the burst based on its schedule. Such scheduler is denoted as CTBR scheduler. The flow chart of the operations is shown in Fig. 4.

Data structure described in [21] can be adapted to sort BHCs in O(1) time. The data structure, which is shown in Fig. 5, consists of a timing wheel and a BHC process list. The timing wheel has an ordered set of time slots. If the time unit is δ, time slot i represents the time between $(i-1)\delta$ and $i\delta$. When a BHC arrives, it is placed in the time slot that corresponds to the arrival time of the burst. If multiple BHCs go to the same time slot, they are linked together. The process pointer points to the time slot corresponding to Δ time units ahead of the current

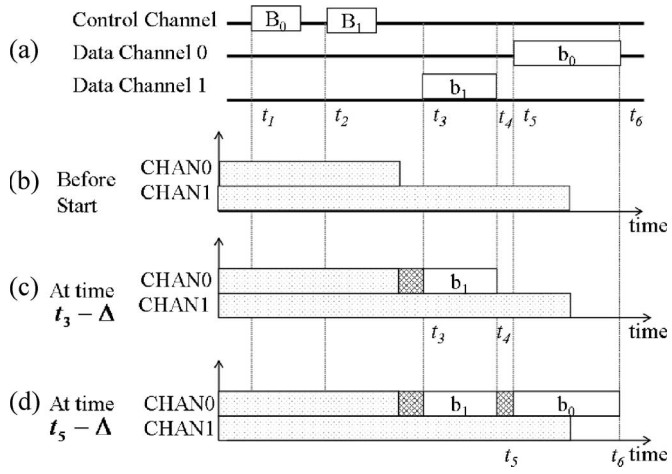

Fig. 6. Scheduling results of horizon scheduler.

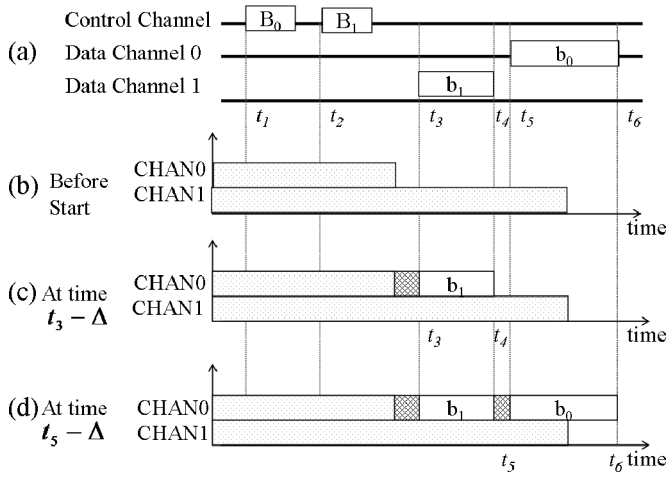

Fig. 7. Scheduling results of CTBR scheduler.

time and advances by one time slot every time unit. Once the process pointer gets to a time slot, the list of BHCs in that time slot is appended to the BHC process list. The horizon scheduler then processes BHCs from the head of the BHC process list.

B. Examples

In the following example, we compare the scheduling results of a basic horizon scheduler and a CTBR scheduler.

Fig. 6 illustrates an example where horizon scheduling fails to use channels efficiently. In Fig. 6, two BHCs B_0 and B_1 are sent on the control channel, while their corresponding bursts b_0 and b_1 are sent on data channel 0 and 1, respectively. Fig. 6(a) shows a case where the offset between B_0 and b_0 is so large that b_0 actually arrives later than burst b_1. Fig. 6(b) shows the channel status before t_1. At time t_1, b_0 is scheduled onto the channel. Channel 0 is the only channel that can accommodate the burst. The new horizon is equal to the end time of b_0. However, because of the large offset between B_0 and b_0, the large gap (void) between the previous horizon and the arrival of b_0 is wasted. Fig. 6(c) shows the gap that results from the scheduling event of b_0. At time t_2, b_1 is to be scheduled. However, the horizons of both channels are larger than the arrival time of b_1. Although there is actually no burst being sent on channel 0 for the duration of b_1, that space is marked unavailable by the horizon scheduler. Therefore, b_1 has to be discarded. Fig. 6(d) shows that only one burst is scheduled successfully.

Fig. 7 illustrates the operation of the CTBR scheduler. The resequencing buffer is not shown. Bursts are scheduled on channels at Δ time units before the burst arrival.

Fig. 7(a) is the same as Fig. 6(a). Fig. 7(b) is the channel status before start. Because b_1 arrives before b_0, BHC B_1 is processed ahead of B_0 by a CTBR scheduler. At $t_3 - \Delta$, BHC B_1 is pulled out from the resequencing buffer. Channel 0 is selected for b_1 and the channel horizon is updated as shown in Fig. 7(c). At time $t_5 - \Delta$, BHC B_0 is processed, and b_0 is scheduled on channel 0. Fig. 7(d) shows the scheduling result obtained by a CTBR scheduler. Compared to the results from a horizon scheduler in Fig. 6, the CTBR scheduler is able to schedule both bursts successfully on the link. The small gaps between successive bursts are not utilized because no bursts are present in the system.

C. Algorithm Description

Define the horizon of channel i to be the earliest time after which there is no scheduled use of channel i. Define the horizon list $\boldsymbol{H}$ to be a channel list sorted based on horizons. Let $H(i)$.chan be the channel listed in the ith entry of the horizon list. Let $H(i)$.horizon be the horizon of the channel listed in the ith entry of the horizon list. Assume that each DWDM link has h data channels.

Let function $\mathbf{Enqueue}$(bhc) be the operation to place a BHC in the resequencing buffer based on the burst arrival time.

Let function $\mathbf{Dequeue}(\Delta)$ dequeue be the BHC if the time difference between the burst arrival time and the current time is equal to or less than Δ.

The algorithm can be implemented using the following procedures.

Initialize (current_time) {
  for $(i = 0; i < h; i++)$
    $H(i)$.chan $= i$;
    $H(i)$.horizon $=$ current_time; }
At BHC Arrival,
  $\mathbf{Enqueue}$(bhc);
Channel_Scheduling {
  While $((\text{bhc} = \mathbf{Dequeue}(\Delta))! = \mathbf{NULL})$ {
    Selected_Entry $= \mathbf{Channel\_Select}$(bhc);
    if (Selected_Entry! $= -1$)
      Selected_Channel $=$ Selected_Entry.chan;
      $\mathbf{Channel\_Update}$(Selected_Entry, bhc);
    else Discard bhc; }
$\mathbf{Channel\_Select}$(bhc) {
  if $(H(0)$.horizon $>$ bhc.arrival_time)
    return -1;
else
    $i = h - 1$;
      while $((H(i)$.horizon $>$ bhc.arrival_time)
        and $(i \geq 0))$
          $i = i - 1$;
      return i; }

Channel Update(Selected_entry, bhc) {
H(Selected_entry).horizon = bhc.arrival_time + bhc.length;
temp_entry = H(Selected_entry);
j = Selected_entry + 1;
while (($j < h$)
and (temp_entry.horizon > $H(j)$.horizon)) {
$H(j-1) = H(j)$;
$j = j + 1$; }
$H(j-1)$ = temp_entry;
return 1; }

D. Algorithm Analysis

In this section, we first analyze the properties of the burst sequence and show how horizon scheduling can produce optimal wavelength schedules under some burst sequence conditions. We then show that CTBR scheduler is able to produce optimal burst schedules.

Let $b_1, \ldots, b_n$ be a sequence of bursts, where b_i is characterized by a triple (r_i, t_i, l_i); r_i is the time at which the BHC is received, t_i is the arrival time of the burst, and l_i is the length (time duration) of the burst. The value of t_i in the triple can be computed by adding the offset field to the BHC arrival time. For convenience, assume that for $i < j$, $r_i < r_j$; that is, the bursts are listed in the order in which the BHCs arrive.

Define the width $W(B)$ of a burst sequence B to be the size of the largest subset of bursts which all overlap in time with one another (that is, the earliest burst ending time in the set is later than the latest burst starting time in the set).

Theorem 1: A sequence of bursts $B = \{b_1 = (r_1, t_1, l_1), \ldots, b_n = (r_n, t_n, l_n)\}$ can be scheduled without delaying or dropping if and only if the number of channels on the link is at least equal to $W(B)$.

Proof: First, we prove that the sequence B cannot be scheduled if the number of channels on the link is less than $W(B)$ using contradiction.

Assume that a sequence of bursts B with width $W(B)$ can be scheduled on the link with less than $W(B)$ channels.

It is trivial to show that for any two bursts that are scheduled on the same channel, the end time of the first burst has to be earlier than the start time of the second burst.

Since the width of the sequence B is $W(B)$, there exist $W(B)$ bursts in B which overlap in time with each other. Because there are less than $W(B)$ channels on the link, there must be at least two bursts in the set that share the same channel. Call these two bursts b_1 and b_2. Without loss of generality, let the start time of b_1 be smaller than the start time of b_2. Since b_1 and b_2 are scheduled on the same channel, the end time of b_1 has to be earlier than the start time of b_2, which contradicts the definition of width $W(B)$.

Second, we prove that if the width of the sequence B is $W(B)$, all bursts in the sequence can be scheduled on the link using $W(B)$ channels.

Assume that the sequence B has to use more than $W(B)$ channels. The reason to schedule a burst on a new channel is that the burst duration overlaps with bursts on scheduled channels. When a burst has to use the $(W(B)+1)$th channel, the duration of the burst overlaps with bursts on all $W(B)$ channels. This contradicts the definition of width $W(B)$. ■

Based on Theorem 1, we would like to have a link scheduling algorithm that would schedule a sequence of bursts without delaying or dropping, so long as $W(B)$ is no larger than the number of available channels.

The following two theorems show that a simple horizon scheduler can produce optimal wavelength schedules if certain burst sequence conditions are met.

Theorem 2: If the bursts arrive in the same order as the BHCs, a horizon scheduler can schedule a burst sequence $B = \{b_1 = (r_1, t_1, l_1), \ldots, b_n = (r_n, t_n, l_n)\}$ using no more than $W(B)$ channels.

Proof: Assume that the horizon scheduler schedules the sequence B using more than $W(B)$ channels.

The horizon scheduler keeps track of the ending time of the last burst scheduled on each channel. It only assigns a burst to a new channel if the ending times of the last burst on all scheduled channels are later than the start time of the burst to be scheduled. Therefore, when the horizon schedule assigns a burst b to the $(W(B)+1)$th channel, the end times of the last burst on all $W(B)$ channels are later than the start time of b.

Now, we prove that the end time of b is later than the start times of the last burst on all $W(B)$ channels.

Since the bursts arrive in the same order as the BHCs, the burst sequence B has the property that $t_1 \leq t_2 \leq \cdots \leq t_n$. Therefore, the start times of bursts that have been scheduled on the channels are earlier than the start time of the burst to be scheduled. The start time of b is later than the start times of the last burst on all $W(B)$ channels. Therefore, the end time of b is later than the start times of the last burst on all $W(B)$ channels.

Therefore, burst b overlaps with $W(B)$ bursts, which contradicts the fact that the width of the burst sequence B is $W(B)$. ■

Theorem 2 shows that the horizon scheduler can get optimal performance if the bursts arrive in the same order as their BHCs. This condition can be relaxed as follows. We can allow some misordering of bursts and still achieve this level of performance.

Theorem 3: The horizon scheduler uses at most $W(B)$ channels to schedule a burst sequence $B = \{b_1 = (r_1, t_1, l_1), \ldots, b_n = (r_n, t_n, l_n)\}$ if for all $i < j$, $t_i < t_j + l_j$.

Proof: This can be proved using induction.

When $j = 1$, it is the first burst to be scheduled, the horizon scheduler uses one channel to schedule the burst.

Assume for $j < k$, the horizon scheduler uses at most $W(B)$ channels to schedule a burst sequence $B = \{b_1 = (r_1, t_1, l_1), \ldots, b_{k-1} = (r_{k-1}, t_{k-1}, l_{k-1})\}$ if for all $i < j$, $t_i < t_j + l_j$.

When b_k is scheduled, there are two cases.

Case 1) There exist at least one channel whose horizon (the latest end time of the bursts scheduled on the channel) is earlier than the start time of b_k.

Then, b_k is scheduled on the channel with the latest horizon that is earlier than the start time of b_k. Because, before b_k is scheduled, the horizon

scheduler uses no more than $W(B)$ channels, the new schedule that includes b_k uses no more than $W(B)$ channels.

Case 2) There is no channel whose horizon is earlier than the start time of b_k. This means that there are at least $W(B)$ bursts whose end times are later than the start time of b_k.

Because for all $i < k$, $t_i < t_k + l_k$, the end time of b_k is later than the start times of these $W(B)$ bursts. Therefore, b_k overlaps with at least $W(B)$ bursts, which contradicts the fact that the burst sequence has a width of $W(B)$. Therefore, this case does not exist.

Therefore, b_k can be scheduled using no more than $W(B)$ channels. By induction, the horizon scheduler uses at most $W(B)$ channels to schedule a burst sequence $B = \{b_1 = (r_1, t_1, l_1), \ldots, b_n = (r_n, t_n, l_n)\}$ if for all $i < j$, $t_i < t_j + l_j$. ■

Based on Theorem 3, we get optimal performance if no burst b_i precedes another burst b_j $(i < j)$ by more than the length of b_j. Although the theorems are trivial to prove, the implications from these theorems are nontrivial. The theorems show that if bursts are scheduled on the order of their arrival sequence, a simple horizon scheduler can produce optimal channel schedules. Since CTBR schedule resequences BHCs according to burst arrival times before making scheduling decisions, it can produce optimal burst schedules.

Define the burst span to be the time between the arrival of the BHC and the end of the corresponding burst.

Theorem 4: Let Δ be target time for the resequencing buffer i. If a set of bursts with burst span $> \Delta$ can all be scheduled on a single channel, then the CTBR scheduler can schedule them using no more than one channel.

Proof: Let $b_1, \ldots, b_n$ be a sequence of bursts, where b_i is characterized by a triple (r_i, t_i, l_i), where r_i is the time at which the BHC arrives, t_i is the arrival time of the burst, and l_i is the length (time duration) of the burst. Assume that $r_1 < r_2 < \cdots < r_n$.

The time to schedule burst b_i is $T_i = \max(r_i, t_i - \Delta)$. Let us sort the set of bursts in ascending order of T_i and assign new sequence numbers to the bursts. The resulting new burst sequence $B' = \{b'_1 = (T'_i, r'_1, t'_1, l'_1), \ldots, b'_n = (T'_n, r'_n, t'_n, l'_n)\}$, where $T'_1 < T'_2 < \cdots < T'_n$.

In order to prove that the set of bursts can be scheduled on a single channel by the scheduler, we first prove that after burst b'_i is scheduled at time T'_i, no bursts scheduled later than b'_i have a burst finishing time before t'_i, that is, to prove $t'_j + l'_j > t'_i$ for $j = i + 1, \ldots, n$.

Based on the definition of the burst span and the condition in the statement, we have the condition $t'_k - r'_k + l'_k > \Delta$, for $k = 1, \ldots n$. Because $T'_i < T'_j$ for $j = i + 1, \ldots, n$, we have $\max(r'_i, t'_i - \Delta) < \max(r'_j, t'_j - \Delta)$. This breaks down into four cases.

Case 1) $r'_i < r'_j$, when $r'_i > t'_i - \Delta$, and $r'_j > t'_j - \Delta$.
Because $t'_j - r'_j + l'_j > \Delta$, we get $t'_j + l'_j > \Delta + r'_j$. Since $r'_i < r'_j$, we can obtain $t'_j + l'_j > \Delta + r'_j > \Delta + r'_i$. From $r'_i > t'_i - \Delta$; $\Delta + r'_i > t'_i$. Therefore, $t'_j + l'_j > t'_i$.

Case 2) $r'_i < t'_j - \Delta$, when $r'_i > t'_i - \Delta$ and $r'_j < t'_j - \Delta$.
Because $r'_i > t'_i - \Delta$, $t'_i < \Delta + r'_i$. Since $r'_i < t'_j - \Delta$, we get $\Delta + r'_i < t'_j$. Therefore, $t'_i < t'_j$ and $t'_i < t'_j + l'_j$.

Case 3) $t'_i - \Delta < r'_j$, when $r'_i < t'_i - \Delta$ and $r'_j > t'_j - \Delta$.
Since $t'_j - r'_j + l'_j > \Delta$, $t'_j + l'_j > \Delta + r'_j$. From $t'_i - \Delta < r'_j$, we get $t'_i < \Delta + r'_j$. Therefore, $t'_j + l'_j > \Delta + r'_j > t'_i$. We have $t'_j + l'_j > t'_i$.

Case 4) $t'_i - \Delta < t'_j - \Delta$, when $r'_i < t'_i - \Delta$ and $r'_j < t'_j - \Delta$.
Since $t'_i - \Delta < t'_j - \Delta$, $t'_i < t'_j$. Therefore, $t'_i < t'_j + l'_j$.

In all four cases, $t'_j + l'_j > t'_i$. This means that no unscheduled burst will be scheduled in the gap before b'_i by an ideal scheduler. Because all bursts can be scheduled on a single channel by an ideal scheduler, all unscheduled bursts must have burst arrival time later than the burst finishing time of b'_i, which is $t'_i + l'_i < t'_j$ for $j = i + 1, \ldots, n$. As a result, the scheduler is able to schedule burst b'_{i+1} on the same channel as b'_i. This is true for all $i = 1, \ldots, n$. Therefore, all bursts in the set can be scheduled on a single channel. ■

Theorem 5: If a set of bursts with burst span $\geq \Delta$ can all be scheduled on r channels, then the CTBR scheduler can schedule them using no more than r channels.

Proof: In the proof of Theorem 4, we have already proved that after burst b'_i is scheduled at time T'_i, no bursts scheduled later than b'_i have a burst finishing time before t'_i, that is, $t'_j + l'_j > t'_i$ for $j = i + 1, \ldots, n$. No unscheduled bursts will use the unused space before b'_i, share the same channel as b'_i, or can be scheduled on another channel within r channels that produce smaller unused space on the channel. If $t'_{i+1} < t'_i + l'_i$, there must exist another channel within r channels that can accommodate b'_{i+1}. This is true for all $i = 1, \ldots, n$. Therefore, the set of bursts can be scheduled on r channels by the CTBR scheduler. ■

Therefore, the CTBR scheduler is able to produce optimal wavelength schedules.

We have also performed a simulation study on the CTBR scheduler. Fig. 8 shows the system performance under the

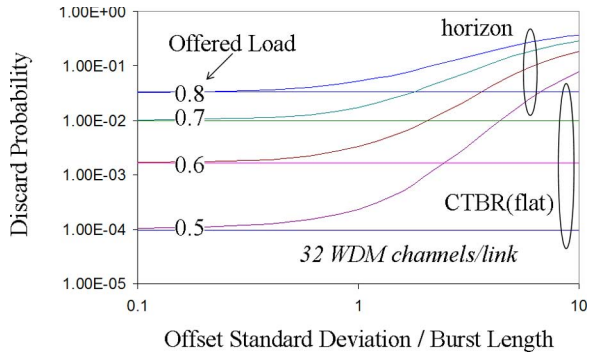

Fig. 8. Performance of CTBR scheduler.

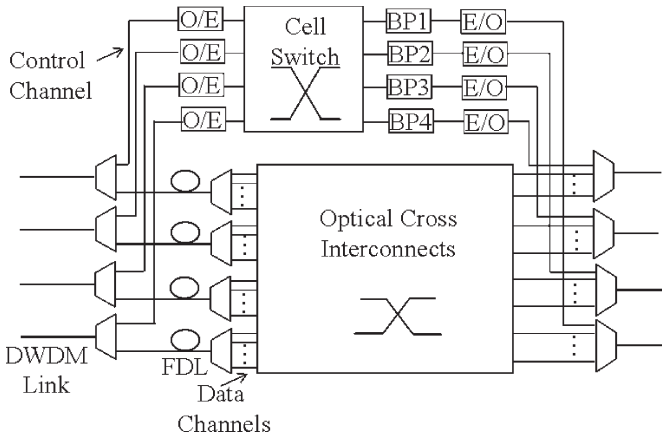

Fig. 9. OBS router architecture with input FDLs.

influence of the variability of the offset. The x-axis is the ratio of the standard deviation of the offset to the average burst length. The burst length is exponentially distributed. The offset has a lognormal distribution with an average of 100 μs. An additional fixed 10-μs offset is added to data bursts at inputs. The value of Δ is 10 μs. Note that based on Theorem 2, the results produced by horizon scheduling are optimal if there is no variability in offset. The burst discard probability of horizon scheduling increases when the ratio increases. The curves for the CTBR scheduler remain flat. Therefore, the CTBR scheduler yields optimal performance, regardless of the variation of the offset values.

E. Discussions

In OBS networks, burst headers have to precede data bursts by an offset time in order to set up optical paths in the OBS core router before bursts arrive at the optical switching fabric. The commonly adopted approach is to set the offset between the burst header and its corresponding burst to be the product of the predetermined hop counts and header processing time at each OBS node. However, such approach cannot be used with the optimal burst-scheduling scheme proposed in this paper since the variable offsets will be equalized by the first router.

Instead, OBS routers that implement CTBR will use the router architecture described in [1] to compensate for the burst header processing time and queueing delays at each router node. A modified OBS router architecture is shown in Fig. 9. As shown in the figure, a set of FDLs are installed at the input of the data channels. Once data bursts reach the OBS router, they enter the FDLs, while burst headers are processed electronically. The CTBR algorithm proposed in this paper is implemented in the BP. In order to prevent the offset from growing as the hop count increases, the BP may hold the burst headers for an extra time if the headers are processed early.

The above described architecture has many advantages. For example, bursts can be launched at the ingress edge router with minimal or zero offset between burst headers and data bursts. Since each router nodes provides its own offset to data bursts, such architecture not only supports CTBR scheduling but also makes network routing independent of the offset, which provides intrinsic support to contention resolution schemes such as deflection routing [22]. Moreover, this architecture facilitates seamless integration of heterogeneous OBS networks without forcing service providers to release sensitive information such as header processing speed.

Although the addition of FDLs in OBS routers may increase the system cost, note that we only need one FDL per router port. In an OBS router, the number of DWDM channels usually has a much higher degree than the number of ports. For example, each router port can accommodate hundreds of DWDM channels (i.e., 256), while the number of ports in an OBS router is usually between four and 32. The length of the FDL is proportional to the burst header processing time and queueing delay in a single router node. Typical values range from a few microseconds to a few tens of microseconds. Therefore, the increase in system cost is minimal.

The proposed CTBR scheduler can be used as the underlying channel scheduler to support various high-level algorithms to achieve additional features such as quality of service (QoS). However, since the CTBR scheduler removes the variability in offsets, the offset-based priority scheme proposed in [23] cannot be supported. In the offset-based priority, bursts with higher priority are assigned a larger offset time. As a result, high-priority bursts are scheduled ahead of the low-priority bursts and have better chance of reserving a wavelength successfully. The main contribution of the offset-based priority lies in the fact that it is the first QoS scheme proposed for OBS networks. Unfortunately, it is well known that the offset-based priority has undesirable end-to-end delay for the high-priority bursts and it favors bursts with shorter lengths [24].

Since horizon algorithm [1], [3] and LAUC-VF [5] were conceived, performance of both algorithms has been under intense study. It is clear that LAUC-VF can produce more efficient channel schedules in general. However, as mentioned earlier, the algorithm complexity of LAUC-VF is high. It has been shown in [25] that in the worst case, LAUC-VF-based algorithms take considerably longer time to execute compared to the horizon algorithm. This is highly undesirable, despite better channel utilization of LAUC-VF-based algorithms. The ultra high-speed requirement of OBS networks makes LAUC-VF less practical in terms of OBS deployment.

As a result, considerable effort has been spent to show that the horizon algorithm may be efficient under certain conditions [25], [26]. The driving force of such effort is due to the simplicity of horizon algorithm and its ability to operate at very high speed. Nevertheless, it is well known that the horizon algorithm can be very inefficient in general operational conditions, particularly in a multihop environment where the offset between the burst header and the burst is proportional to the number of remaining hops that the burst will traverse. Therefore, the OBS research community is facing a dilemma in choosing scheduling algorithms with higher channel efficiency versus faster processing speed. The CTBR algorithm proposed in this paper provides a solution that can achieve optimal channel efficiency while being able to operate at the speed comparable to the horizon algorithm. In the next section, we discuss hardware implementation details of the CTBR scheduler.

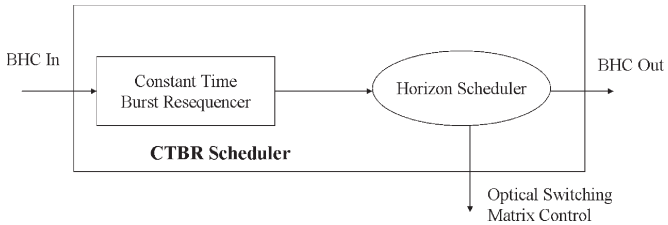

Fig. 10. Block diagram of CTBR scheduler.

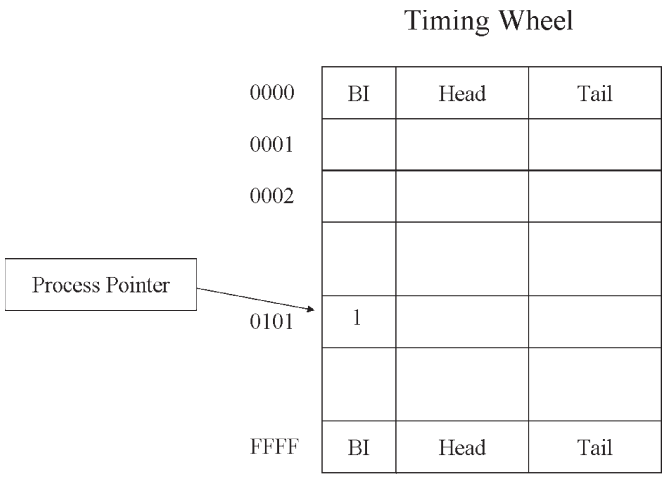

Fig. 11. Structure of timing wheel memory.

F. Hardware Implementation

The proposed CTBR scheduler can achieve O(1) runtime when implemented in hardware. This section details the hardware implementation of the CTBR scheduler.

Fig. 10 shows the block diagram of the CTBR scheduler. The arriving BHC is first placed in the constant time burst resequencer. The BHC stays in the resequencer until Δ time before the burst arrival. The BHC is then removed from the resequencer and sent to the horizon scheduler. The horizon scheduler assigns a wavelength to the burst according to its channel status and then forwards the BHC after modifying the offset field and the wavelength on which the burst will be sent.

The hardware implementation of the constant time burst resequencer, which is shown in Fig. 5, is explained as follows. The timing wheel is implemented as a circular list whose entries correspond to time slots. Each time slot contains a head and a tail pointer to the BHC list in that time slot. It also contains a busy/idle bit to indicate whether there is any BHC list in the time slot. The last entry in the timing wheel is conceptually adjacent to the first entry. The memory structure of the timing wheel is shown in Fig. 11.

Fig. 12 shows the structure of the BHC memory. BHCs are stored in BHC buffer slots. Each BHC buffer slot has space to store a BHC (length, offset, wavelength fields, etc.) plus a next pointer. The free space list is a linked list that contains all unused BHC buffer slots.

When a BHC arrives, the time difference between the burst arrival time and the current time determines the time slot to which to append the BHC. This time difference directly translates to the memory location in the timing wheel. A free BHC buffer slot is obtained from the free space list, and the BHC is written into the BHC buffer slot. If the time slot in the timing wheel is previously empty, both the head and tail pointers in the time slot point to the BHC buffer slot where the BHC is stored. The next pointer in the BHC buffer slot is set to NULL. If the time slot is not previously empty, the BHC is inserted from the head of the list. To do this, the head pointer stored in the selected time slot is written as the next pointer field in the BHC buffer slot. The BHC buffer slot assigned to the BHC becomes the new head pointer in that time slot.

The BHC process list contains a list of BHCs whose associated burst arrival times are equal to or less than Δ time units from the current time. The process pointer always points to the time slot that corresponds to current time plus Δ time units and advances by one time slot every time unit. When the process pointer gets to a time slot, the BHC list in the time slot is removed from the timing wheel and is appended to the BHC process list, if any. This can be done by pointing the last BHC in the BHC process list to the head of the BHC list to be moved. The head and tail pointers in that timing wheel slot are set to NULL. The horizon scheduler always processes BHCs from the head of the BHC process list.

With the timing wheel structure described above, resequencing BHCs based on the burst arrival times can be done with constant number of memory operations, where the constant is small. The number of time slots needed in the timing wheel depends on the granularity of the time unit and the latest future time to be supported in the system. Note that the purpose of resequencing the BHCs on the order of the burst arrival times is to avoid the inefficiency of the basic horizon scheduler in dealing with bursts with large offset variations. Based on Theorem 3, the horizon scheduler produces optimal schedules as long as no bursts precede another burst by more than the length of the burst. This means that the scheduler can produce optimal schedules as long as the time unit for the resequencer is less than half of the minimum burst duration to handle the worst-case arrival timing.

The electronic RAM requirements for the timing wheel and the BHC memory are calculated as follows. Supposing the time unit is 1 μs, we need 1 k time slots in the timing wheel to represent a maximum time period of 1 ms. Supposing the head and tail pointers are 17 bits each, which supports 100 k BHCs, we need 34 kB of memory for the timing wheel. Supposing each BHC is 40 B, to support 100 k BHCs in the system, we need 4 MB of memory to implement BHC buffer slots.

The details of hardware implementation of the horizon scheduler are explained as follows. When the BHC arrives at the channel scheduler, the control information such as the offset, the length, the input link, and the wavelength on which the burst is arriving are extracted. The projected burst arrival time is calculated. The horizon list is used to decide which channel is the best selection for the incoming burst. If there is a channel that can accommodate the incoming burst, the selected channel is added to the proper field in the burst request. If there is no channel available, the burst request is marked "discarded."

The horizon list contains h entries, where h is the number of data channels per link. Each entry has two fields: the channel

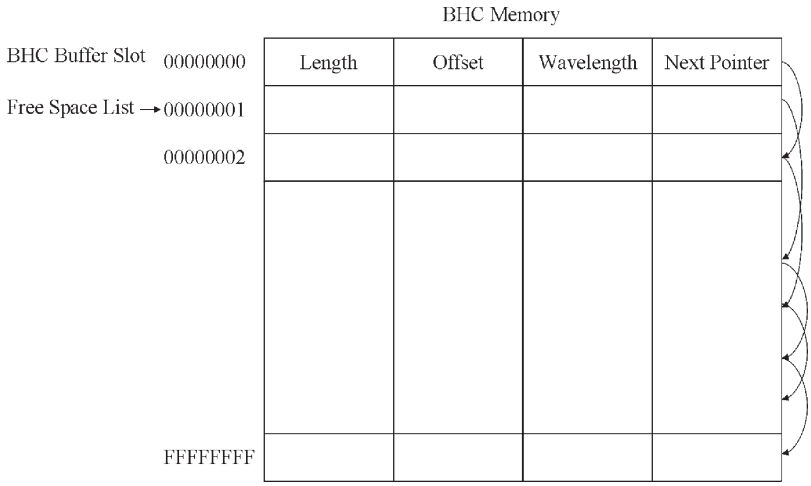

Fig. 12. Structure of BHC memory.

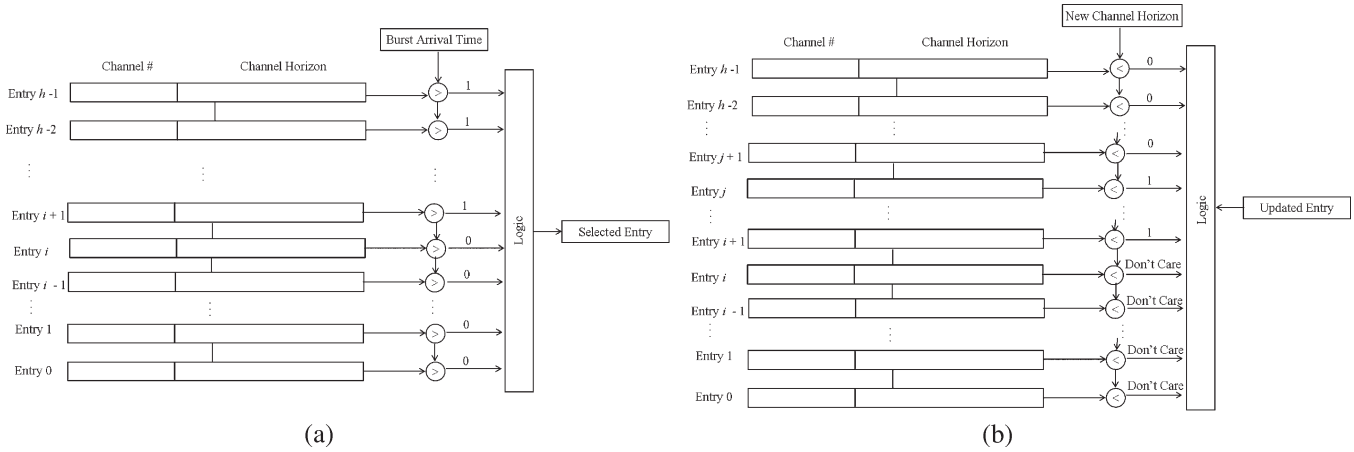

Fig. 13. (a) Horizon scheduling channel selection. (b) Horizon scheduling channel updates.

number and the channel horizon. All entries are sorted in ascending order of the channel horizons. That is, lower numbered entries contain channels with smaller channel horizons. If the current time passes the channel horizon, the channel is marked "unassigned."

There are two operations performed on the channel horizon list: channel selection and channel updates.

During channel selection, the horizon scheduler finds the latest channel horizon that is earlier than the burst arrival time. The logic to do the selection is shown in Fig. 13(a). The channel horizons of entries are compared to the burst arrival time at the same time using parallel hardware comparators. Because the channel horizon list is a sorted list, the result from the comparison has an interesting property. Starting from the lowest entry, results form a set of consecutive "0"s followed by a set of consecutive "1"s, where "0" indicates that the channel horizon is less than or equal to the burst arrival time, and "1" indicates that the channel horizon is larger than the burst arrival time. The entry at the "0" to "1" transition is the entry to be selected. In the example, entry i is the selected entry. The channel number in the selected entry is the channel on which to send the burst. Therefore, the best channel can be found in simply two steps.

After a channel is selected, the new channel horizon is set to the finish time of the new burst. The channel horizon list needs to be updated. The logic for channel updates is similar to that for channel selection. However, only the entries with indexes larger than the selected entry are compared with the new channel horizon. A "1" signal is generated if the channel horizon of the entry is less than the new channel horizon. Otherwise, a "0" signal is generated. Similarly, the comparison generates a set of consecutive "1"s followed by a set of consecutive "0"s starting from the entry above the selected entry. The logic is shown in Fig. 13(b). When the comparison is done, the controller logic generates a set of SHIFT signals for entries with "1"s, a set of STAY signals for entries with "0"s, and an INSERT signal for the entry at the "1" to "0" transition. The selected entry and all entries below it receive a STAY signal as well. All entries that receive SHIFT signals move down by one position. The entry with the INSERT signal loads the selected channel and the new channel horizon. No operation is needed for the entries receiving STAY signals. This can be done as a hardware shift register operation, which can be completed in one clock cycle. In the example, entry j is the new position for the updated entry. After this simple operation, the updated channel horizon list becomes a sorted list again.

As we can see, both components in building a CTBR scheduler can achieve O(1) runtime in hardware. Therefore, the CTBR scheduler can achieve optimal burst scheduling in

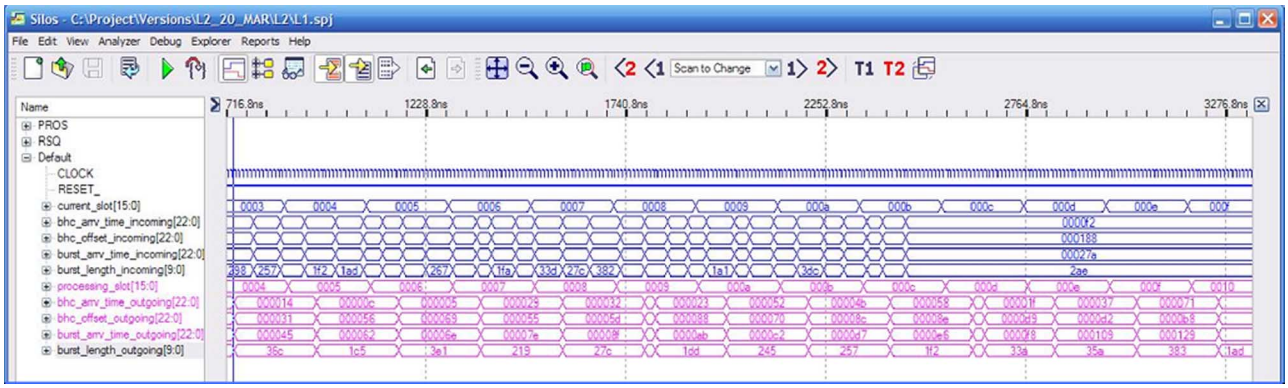

Fig. 14. Verilog HDL circuit simulation for constant time burst resequencer.

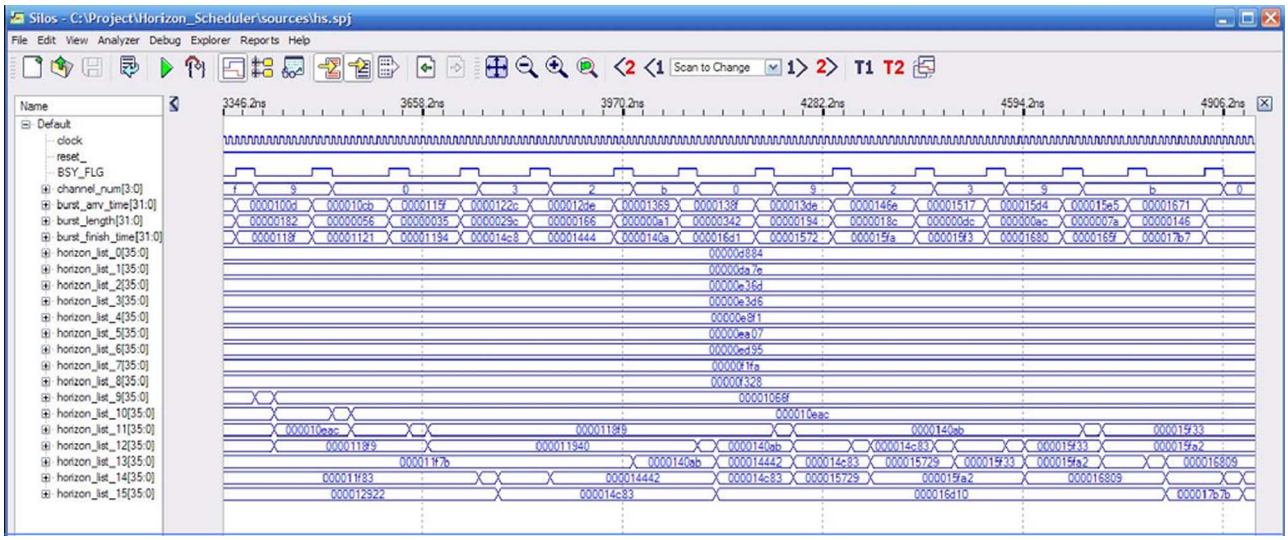

Fig. 15. Verilog HDL circuit simulation for horizon scheduler.

O(1) runtime. The CTBR scheduler has been implemented in hardware using Verilog hardware description language (HDL). The circuit simulation results for the resequencer and a 16-channel horizon scheduler are shown in Figs. 14 and 15, respectively. The circuit has been synthesized using Quartus II to Altera Stratix II EP2S15F672C3 field programmable gate array (FPGA). The clock frequency of the synthesized circuit can achieve 256 MHz for the constant time burst resequencer and 242 MHz for the horizon scheduler.

VI. Conclusion

In this paper, we have proposed an optimal burst scheduler using CTBR, which runs in O(1) time. The proposed CTBR scheduler is able to produce optimal burst schedules while having comparable processing speed as the well-known horizon scheduler. We have demonstrated that the algorithm is well-suited to high-performance hardware implementation.

Acknowledgment

The authors would like to thank L. Wang for completing the hardware implementation of the CTBR scheduler described in this paper. The authors would also like to thank the anonymous reviewers for their constructive comments for improving this paper.

References

[1] J. S. Turner, "Terabit burst switching," *J. High Speed Netw.*, vol. 8, no. 1, pp. 3–16, Mar. 1999.

[2] C. Qiao and M. Yoo, "Optical Burst Switching (OBS)—A new paradigm for an optical Internet," *J. High Speed Netw.*, vol. 8, no. 1, pp. 69–84, Mar. 1999.

[3] Y. Chen and J. S. Turner, "WDM burst switching for petabit capacity routers," in *Proc. IEEE Mil. Commun. Conf.*, 1999, pp. 968–973.

[4] J. Y. Wei and R. McFarland, "Just-in-time signaling for WDM optical burst switching networks," *J. Lightw. Technol.*, vol. 18, no. 12, pp. 2019–2037, Dec. 2000.

[5] Y. Xiong, M. Vandenhoute, and H. C. Cankaya, "Control architecture in optical burst-switched WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 1838–1851, Oct. 2000.

[6] J. Xu, C. Qiao, J. Li, and G. Xu, "Efficient channel scheduling in optical burst switched networks," in *Proc. IEEE INFOCOM*, 2003, vol. 3, pp. 2268–2278.

[7] J. Xu, C. Qiao, J. Li, and G. Xu, "Efficient burst scheduling algorithms in optical burst-switched networks using geometric techniques," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 9, pp. 1796–1881, Nov. 2004.

[8] V. M. Vokkarane and J. P. Jue, "Segmentation-based nonpreemptive channel scheduling algorithms for optical burst-switched networks," *J. Lightw. Technol.*, vol. 23, no. 10, pp. 3125–3137, Oct. 2005.

[9] J. Teng and G. N. Rouskas, "Wavelength selection in OBS networks using traffic engineering and priority-based concepts," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 8, pp. 1658–1669, Aug. 2005.

[10] D. Blumenthal, P. Prucnal, and J. Sauer, "Photonic packet switches: Architectures and experimental implementation," *Proc. IEEE*, vol. 82, no. 11, pp. 1650–1667, Nov. 1994.
[11] P. Gambini *et al.*, "Transparent optical packet switching: Network architecture and demonstrators in the KEOPS project," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 7, pp. 1245–1259, Sep. 1998.
[12] 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.
[13] L. Tamil, F. Masetti, T. McDermott, G. Castanon, A. Ge, and L. Tancevski, "Optical IP routers: Design and performance issues under self-similar traffic," *J. High Speed Netw.*, vol. 8, no. 1, pp. 59–67, Mar. 1999.
[14] B. Wen and K. M. Sivalingam, "Routing, wavelength and time-slot assignment in time division multiplexed wavelength-routed optical WDM networks," in *Proc. INFOCOM*, Jun. 2002, pp. 1442–1450.
[15] S. Yao, S. Dixit, and B. Mukherjee, "Advances in photonic packet switching: An overview," *IEEE Commun. Mag.*, vol. 38, no. 2, pp. 84–94, Feb. 2000.
[16] W. D. Zhong and R. S. Tucker, "Wavelength routing-based photonic packet buffers and their applications in photonic packet switching systems," *J. Lightw. Technol.*, vol. 16, no. 10, pp. 1737–1745, Oct. 1998.
[17] N. McKeown, "iSLIP: A scheduling algorithm for input-queued switches," *IEEE Trans. Netw.*, vol. 7, no. 2, pp. 188–201, Apr. 1999.
[18] M. Ajmone Marsan, A. Bianco, E. Leonardi, and L. Milia, "RPA: A flexible scheduling algorithm for input buffered switches," *IEEE Trans. Commun.*, vol. 47, no. 12, pp. 1921–1933, Dec. 1999.
[19] H. Duan, J. W. Lockwood, S. M. Kang, and J. D. Will, "A high performance OC12/OC48 queue design prototype for input buffered ATM switches," in *Proc. IEEE INFOCOM*, 1997, vol. 1, pp. 20–28.
[20] Y. Chen, J. S. Turner, and Z. Zhai, "Design and implementation of an ultra fast pipelined wavelength scheduler for optical burst switching," in *Proc. IASTED Int. Conf. WOC*, Jul. 2006, pp. 263–270.
[21] M. Henrion, "Resequencing system for a switching node," U.S. Patent 5 127 000, Jun. 30, 1992.
[22] C.-F. Hsu, T.-L. Liu, and N.-F. Huang, "Performance analysis of deflection routing in optical burst-switching networks," in *Proc. IEEE INFOCOM*, 2002, pp. 66–73.
[23] M. Yoo and C. Qiao, "QoS performance of optical burst switching in IP-over-WDM networks," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 10, pp. 2062–2071, Oct. 2000.
[24] Y. Chen, M. Hamdi, and D. H. K. Tsang, "Proportional QoS over OBS Networks," in *Proc. Globecom*, 2001, vol. 3, pp. 1510–1514.
[25] F. Vázquez-Abad, J. White, L. Andrew, and R. Tucker, "Does header length affect performance in optical burst switched networks?" *OSA J. Opt. Netw.*, vol. 3, no. 5, pp. 342–353, May 2004.
[26] L. Andrew, Y. M. Baryshnikov, E. G. Coffman, Jr., S. V. Hanly, and J. White, "The asymptotics of optimal OBS wavelength assignment," in *ACM Sigmetrics: Performance Evaluation Review*, vol. 31. New York: ACM, Sep. 2003, pp. 14–16.

Yuhua Chen (M'04) received the M.S. degree in computer science, the M.S. degree in electrical engineering, and the D.Sc. degree in electrical engineering from Washington University in St. Louis, St. Louis, MO, in 1997, 1998, and 2003, respectively.

She was a Research Associate/Hardware Design Engineer at the Applied Research Laboratory, Washington University in St. Louis, from 1999 to 2004, where she was involved in advanced networking system design and prototyping. She was one of the original developers for optical burst switching (OBS). She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX. Her research interests include OBS, high-performance routers, system prototyping, reconfigurable systems, system-on-chip, and wireless sensor networks.

Jonathan S. Turner (M'77–SM'88–F'90) received the M.S. and Ph.D. degrees in computer science from Northwestern University, Evanston, IL, in 1979 and 1981, respectively.

He holds the Henry Edwin Sever Chair of Engineering at Washington University in St. Louis, St. Louis, MO, where he is Director of the Applied Research Laboratory. His primary research interest is the design and analysis of switching systems with special interest in systems supporting multicast communication. He has been awarded more than 20 patents for his work on switching systems and has many widely cited publications.

Dr. Turner is a member of the Association for Computing Machinery and the Society of Fire Protection Engineers. He received the Koji Kobayashi Computers and Communications Award from the IEEE in 1994 and the IEEE Millennium Medal in 2000.

Pu-Fan Mo received the B.S. degree in computer science and engineering from The Ohio State University, Columbus, in 2003. He is currently working toward the Master's degree with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX.