Design and Implementation of An Index-based Parallel Scheduler for Optical Burst Switching Networks

Guiling Wu[1], Tairang Zhan[1], Jianping Chen[1], Xinwan Li[1], Chunming Qiao[2]

[1]*State Key Lab of Advanced Optical Communication Systems and Networks, Shanghai Jiao Tong University, Shanghai 200240, China*

[2]*CSE Department, SUNY Buffalo, New York 14228, USA*

wuguiling@sjtu.edu.cn, Ztr1918294@yahoo.com.cn, jpchen62@sjtu.edu.cn, lixinwan@sjtu.edu.cn, qiao@computer.org

Abstract: A fast and easy to implement index-based parallel burst scheduler for OBS network is proposed. A 16-channel scheduler with scheduling time of 33.3ns per burst request is demonstrated in FPGA.

OCIS codes: (060.2330) Fiber optics communications; (060.4259) Networks, packet-switched

1. Introduction

OBS (Optical Burst Switching) has been proposed as a promising solution to support IP over DWDM [1,2]. Fast and resource efficient burst scheduling is one of the key issues in OBS networks. The LAUC-VF (Latest Available Unused Channel with Void Filling) algorithm has a high bandwidth efficiency [3]. Its practical application, however, is limited since a straightforward implementation of LAUC-VF takes O(n) time to schedule a burst request where n is the number of voids in all data channels [4]. The larger variation in its scheduling time with the load also makes it difficult to configure the offset time and may degrade the network performance dramatically [5]. The SWAP (Slotted Wavelength Assignment Pipeline) scheduler proposed by M. T. Anan et al. achieves fast scheduling at the cost of low utilization of the channel by using a fixed-duration time slot as the minimum unit of channel allocation [6]. The CTBR (Constant Time Burst Resequencing) scheduler proposed in [7], which delays the scheduling of bursts and processes them in the order of the expected burst arrival time, is able to run in O(1) time. The scheduler, however, needs a resequencer at output ports, and a set of FDLs at the input of every OBS core switching node to delay all data bursts.

In this paper, a novel index-based parallel LAUC-VF scheduler is proposed, which for the first time, can find feasible voids on different channels in parallel with O(1) time complexity and achieves the highest possible efficiency. A 16-channel scheduler is implemented in FPGA (Field Programmable Gate Array) with a scheduling time of 33.3ns per burst request. This represents a significant advance in practical realization of high-speed and resource efficient burst scheduling algorithms.

2. Index based parallel scheduling scheme

The channel scheduling process consists of two phases in the proposed scheme. The first phase is to search for feasible voids, which can accommodate the newly arrived burst, on all date channels. The second phase is to select the optimal void among the feasible voids found.

In order to reduce the time to search for the feasible voids, which takes up most of scheduling time of the existing implementations, an index-based searching approach is proposed to find one feasible void on each channel and applied in parallel to each channel. The key idea of the index-based searching approach is to build an index vector of voids for each data channel and search for the feasible voids based on the index vector. Figure 1 illustrates how the searching approach works on one channel. The scheduling time window of the channel is divided into N slots. Each time slot is assigned an index number according to its time sequence. A N-bit index vector and a N-entry void table are built accordingly as shown in Fig.1. Each void on the channel is assigned an index value, which is same as that of the slot where the start time of the void is located. The start time and end time of each void is recorded in the N-entry void table according to its assigned index number. Finally, if and only if a void has an index number of i, the i-th bit in the N-bit index vector is set to 1. For example, in Fig.1 (a), there are two scheduled bursts and three voids. Void_1, void_2 and void_3 are assigned index numbers of 1, 3 and 9, respectively, according to their starting slots. The 1st, 3rd, and 9th bits in the N-bit index vector are set to "1" while other bits are "0", and the start time and the end time of the three voids are recorded in the 1st, 3rd, and 9th entry of the N-entry void table, respectively. Searching for a feasible void on a channel using the index-based data structure includes the following three steps.

Step 1: Determine the index number of the arrival time slot of a new burst, $m = \left[(t_a - t_s)/\tau \right] + 1$. Where, t_a is the arrival time of the burst, t_s is the starting time of the scheduling window, τ is the length of time slot, and $[\cdot]$ is a rounding operation.

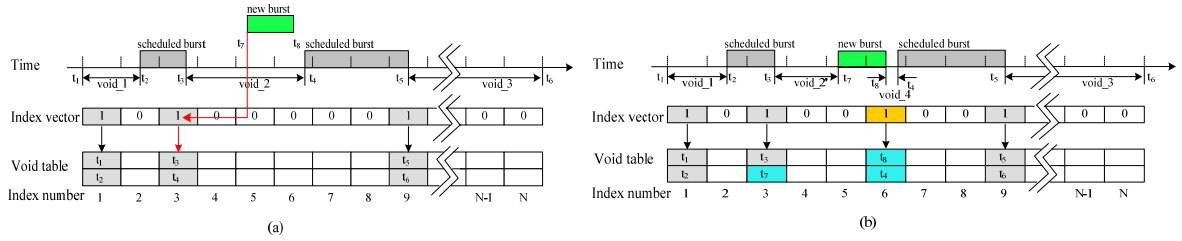

Fig. 1. Index-based data structure for one channel

Step2: Find a candidate void whose starting time is just before the arrival time of the new bursts.

In this step, we first generate a *N*-bit mask code M[*N*: 1] corresponding to time slots from *N* to 1, where the *i*-th least significant bit *i* of the mask is set to 1 if and only if $i <= m$. For example, in the Fig.1(a), the *N*-bit mask code for the new burst whose *m* is 5 and accordingly, M[*N*:1] is 00…000011111. Then, we perform a bit-wise "and" operation between the *N*-bit mask code and the *N*-bit index vector V[*N*:1] (=00...100000101 in this case) to obtain 00...000000101. Since the highest index number of "1" bit is 3, the candidate void starts in the third slot.

Step 3: Extract the start time and the end time of the candidate void from the corresponding *N*-bit void table according to the index number of the candidate void, and compare them with those of the new burst to determine whether the candidate void found in step 2 is a feasible void.

In Fig.1(a), the new burst arriving at t_7 and departing at t_8 can fit in void_2 between t_3 and t_4. Fig.1(b) shows the updated index vector and void table after the burst is scheduled on the channel. We can see that the slot 6 is already partially occupied by a scheduled burst, but it can still be used by the new burst, which distinguishes our approach from other slot-based scheme where a time slot can only be occupied by one burst, even if it is only partially.

In order to find the feasible voids on all channels, the index-based searching process is applied to each channel in parallel. The whole searching process can run in O(1) time since each step in the first phase such as the location and the determination can be completed in a constant time by hardware.

The second phase produces the difference between the start time of each feasible void and the arrival time of a new burst, and adopts a general pairwise comparison approach to select the optimal void according to the selecting criteria of LAUC-VF. Theoretically, pairwise comparison process has a time complexity of O(logC) in the worst case, where C is the number of channels. The comparison operations, however, can be completed quickly in hardware.

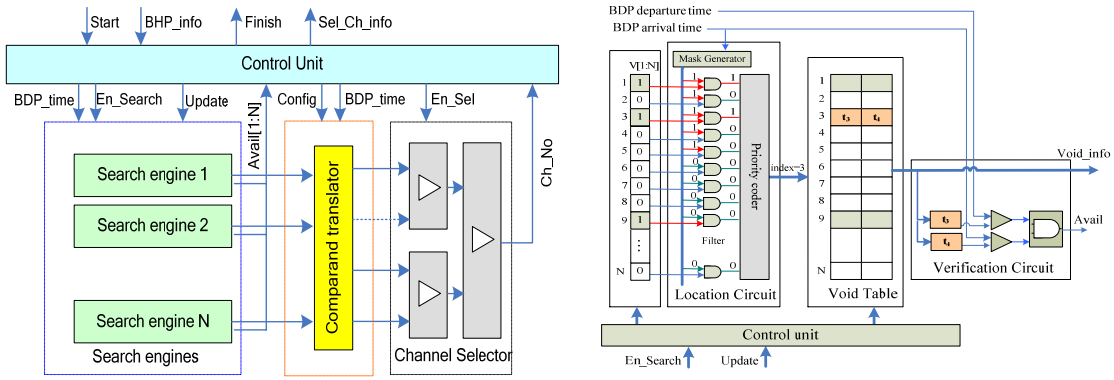

Fig. 2. The hardware architecture of scheduler (left) and the circuit diagram of a searching engine (right)

3. Hardware architecture and implementation

Fig.2(a) shows the hardware architecture of the proposed index-based parallel scheduler, which consists of a control unit, a set of parallel search engines, a comparand translator and a channel selector. The control unit is responsible for the timing control of the whole scheduling process, and the interfaces with external modules. Each searching engine runs the index-based searching procedure to find the feasible void on the corresponding channel for new bursts. The comparand translator is used to output the comparands (the difference between the arrival time of a new burst and the start time of each feasible void for LAUC-VF). The channel selector uses multi-stage comparators to select the optimal void via the pairwise comparison process. Other scheduling schemes with void filling can also be implemented based on this hardware architecture just by adopting the corresponding comparand translators.

Fig.2(b) shows the circuit diagram of a search engine. The circuit is composed of an array of flip-flops to maintain the index vector, a location circuit to find the candidate void, an on-chip RAM to store the *N*-entry void table and a verification circuit. The location circuit contains a mask generator, an array of "and" gates and a priority coder. The mask generator generates the *N*-bit mask code according to the arrival time of the new bursts, M[*N*: 1]. The array of "and" gates is used to complete the "and" operation between the mask code and the index vector. The output of a "and" gate is valid if and only if the corresponding bit in the index vector is "1" and its index number is less than *m* (the first and the third output are valid in Fig. 1(b)). The priority coder selects the index number of the candidate void by giving the larger index number a higher priority. The contents of corresponding entry in the void table are fed to the verification circuit according to the index number of the candidate void. The verification circuit checks whether the candidate void is a feasible void, and sets the corresponding avail signal accordingly.

Fig.3 shows the circuit simulation result of a 16-channel index-based parallel scheduler, which is synthesized to Xilinx Virtex4 XC4VFX20 FPGA with 150 MHz of clock frequency. 18 scheduling processes are shown in the figure. Every scheduling process takes 5 cycles (4 cycles to obtain the optimal void, and one cycle to update the data structure), which is about 33.3ns. The signal captured using the ChipScope (Xilinx, Inc.) when the scheduler runs in the FPGA is also shown in Fig.3.

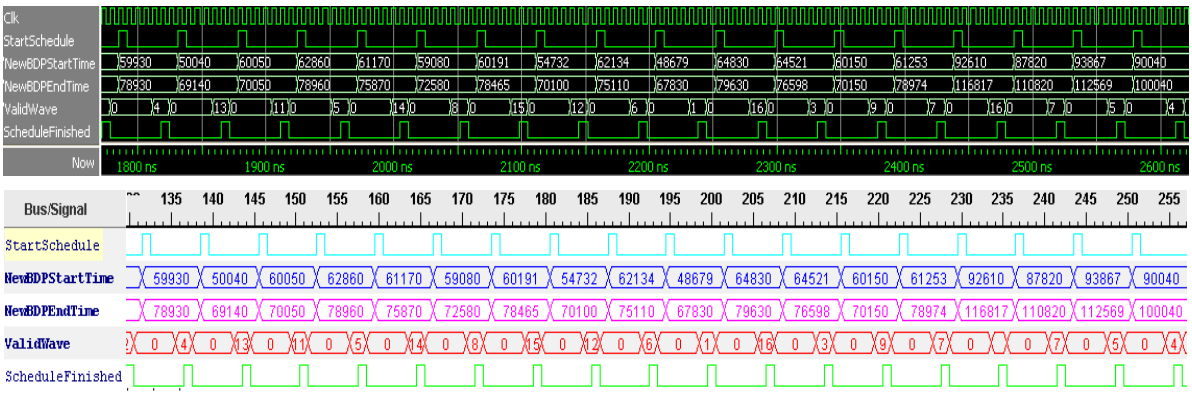

Fig. 3. The circuit simulation of scheduling processes (up) and the signal captured using ChipScope (down)

4. Conclusion

In this paper, we have proposed a novel index-based parallel scheduler for OBS networks to achieve a high channel utilization and high processing speed at the same time. The proposed scheduler can implement the scheduling algorithms with void filling with the highest possible utilization. In particular, The feasible void on one channel for a request can be found by an index-based searching procedure with O(1) time complexity. A 16-channel index-based parallel scheduler, which is able to complete one scheduling procedure every 33.3ns, has been demonstrated in FPGA.

5. Acknowledgement

This work was supported in part by 973 program (2011CB301700), National Science Foundation of China (61071011, 6087701), 863 Project (2007AA01Z275), and the STCSM Project (10DJ1400402, 09JC1408100).

4. References

[1] C. Qiao and M. Yoo, "Optical burst switching (OBS)-A new paradigm for an Optical Internet," Journal of High Speed Networks 8, 69-84 (1999).

[2] J. S. Turner, "Terabit burst switching," Journal of High Speed Networks 8, 3-16 (1999).

[3] Y. Xiong, M. Vandenhoute, and H. C. Cankaya, "Control architecture in optical burst-switched WDM networks," IEEE Journal on Selected Areas in Communications 18, 1838-1851 (2000).

[4] J. Xu, C. Qiao, J. Li and G. Xu, "Efficient Channel Scheduling Algorithms in Optical Burst Switched Networks," IEEE Infocom 2003, San Francisco, April, 2003.

[5] Wei Dai, Guiling Wu, et al., "Offset Time Configuration in Optical Burst Switching Ring Network," JLT 27(19), 4269-4279 (2009).

[6] M. T. Anan and G. M. Chaudhry, "A real-time hardware-based scheduler for next-generation optical burst switches," IEEE ICC'07, Glasgow, June, 2007.

[7] Y. Chen, J. S. Turner, and P. F. Mo, "Optimal burst scheduling in optical burst switched networks," JLT 25, 1883-1894 (2007).