NOVEL DESIGN FOR AN ASYNCHRONOUS OPTICAL PACKET SWITCH

W. Vanderbauwhede, D. A. Harle
Broadband and Optical Networks group
Institute of Communications and Signal Processing
Department of Electronic and Electrical Engineering
University of Strathclyde, Glasgow
W. Vanderbauwhede@comms.eee.strath.ac.uk, D.Harle@eee.strath.ac.uk

Abstract We present a novel design for an asynchronous optical packet switch. The architecture is GMPLS-compliant, DWDM-capable and fully scalable. The switch uses a novel in-line buffer design, based on parallel recirculating buffers. The buffers solve contention by statistical multiplexing, and can be configured to conserve packet order and prioritize traffic. The control system is based on a direct local lookup of the destination port and wavelength and traffic class using the packet label. Performance modeling indicates that the switch has excellent throughput with low latency and low packet loss.

1. Introduction

1.1 Background

Much current optical network R&D focuses upon the implementation of a dynamically reconfigurable optical transport layer based on fast optical cross-connects (OXC's) coupled with a suitable control and management architecture. Thus, in the near future, an optical transport network capable of supporting large numbers of high capacity circuit-switched optical channels, with bit rates of 10-40 Gb/s will be realized. Although in this future scenario it might seem that bandwidth is not an issue, this is not the case; economics will always demand that network resources are used efficiently. A major advantage of packet switching lies in its bandwidth efficiency and ability to support diverse services. Hence research is now addressing the advent towards bringing the packet switching concept into the optical domain, that is optical packet switching (OPS).

An attractive feature of OPS is that it can appear as a natural evolution of the optical transport network. Designated wavelengths supporting optical packets can be dropped from the OXC, processed within the OPS and then either inserted back into the core or dropped off locally.

The successful development of optical packet switching requires a close integration of advanced component technology with the network layers; an example being the need to control and manage fast switches.

1.2 Focus of the paper

This paper focuses upon the system-level design of an innovative DWDM-capable optical packet switch (OPS). Thus, the actual implementation of the basic building blocks of the design is not directly relevant, what matters is the functionality of the component. Possible implementations will be mentioned to illustrate or to evaluate the feasibility of a given design. However, this by no means implies that the final realization would be restricted to the proposed implementation. To make this point more clear, consider the following example: a key component of the system is the multiplexer. This is a component which combines several streams of packets on a single carrier. The way this component is realized is not actually relevant for the design: in the extreme, it could be an O/E converter followed by an electronic logic OR gate followed by an E/O converter, or it could simply be a passive optical star coupler.

2. Requirements

An optical network with high data transport rates and QoS imposes a number of requirements on the optical packet switching node (Fig. 1):

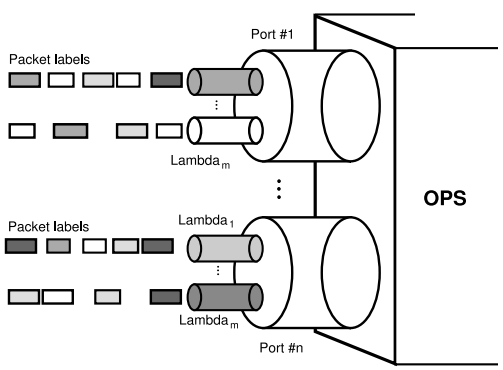

Figure 1. Node layer stack

- A GMPLS-compliant architecture

  To be able to fulfill QoS requirements, the OPS must be GMPLS compliant. Generalized Multi-Protocol Label Switching [1],[2] is an extension or generalization of MPLS [3] that allows a label to be a wavelength, frequency, time slot or position in space. The basic idea behind MPLS is to pre-establish paths along which the data will be forwarded. For an OPS, forwarding of a packet is based on three "labels": the input port of the OPS, the input wavelength, and the packet label. Furthermore, to guarantee a certain QoS level, it must be possible to prioritize the traffic, e.g. according to the DiffServ classes [4]. In general, it is desirable to conserve the packet order.

- DWDM

The OPS node must be suitable for DWDM and scalable. In addition to simple space switching, the node must be able to distinguish between different wavelengths and be able to switch datastreams from one wavelength to another. The number of wavelengths should not be limited by the design, although it may be limited by the state of the art for the technology. Such a scalability requirement has a major impact on the architecture and cannot be over emphasized.

- High data transport rates

  The OPS node design must allow operation at high bitrates (40 Gb/s per datachannel, scalable to 100 Gb/s and higher) under high network loads.

- Asynchronous switching

  The node must be able to handle packets of variable length, with variable inter-arrival times and asynchronous arrivals. To minimize packet losses, there must be contention resolution.

3. Node architecture

3.1 Node layer model

The OPS node can be divided in 4 layers (Fig.2):

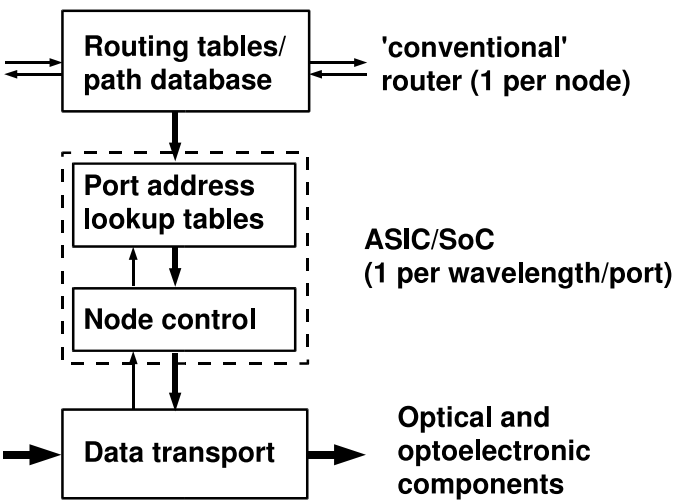

Figure 2. Node layer stack

- The network management layer contains the routing tables and the path database, as well as the management software.
- The interface layer consists of the label lookup tables. These are written by the network management layer and read by the switch control layer. The lookup tables contain the destination port address and the priority for each packet label.

- The switch control layer consists of the control electronics. This controls the switch and buffers based on the content of the lookup table, the packet length and the state of the switch. The control layer will signal to the network management layer in case of impending buffer overflow or hardware faults.
- The data transport layer is the optical part of the switch. This part is controlled by the switch control layer. The datastreams remain in the optical domain. Only the header is read. No OE/EO conversion is necessary.

3.2 Implications on node design and architecture

The requirements for the node architecture are in part purely physical, i.e. bitrate and number of ports and wavelengths, and partly imposed by the network management layer, i.e. allow wavelength translation, solve contention etc. Such requirements impose significant but straightforward design constraints:

- Scalability requires modular approach
- Contention resolution requires optical buffering
- Variable packet size and spacing requires asynchronous switching

3.3 Modular DWDM optical packet switching node architecture

The design of the packet switching node (OPSN) adopts a five stage architecture which is illustrated in Fig.3. The key components of the design are single-wavelength optical packet switching elements (OPSE). The modular design is essential because monolithic designs in general do not scale well. If the number of ports and wavelengths is small, a monolithic design will be advantageous, but for large numbers of ports and/or wavelengths, the practical implementation becomes increasingly difficult.

The five key stages are as follows:

- per input port: wavelength demultiplexing
- per input port: wavelength translation
- per wavelength: space switching
- per output port: wavelength translation
- per output port: wavelength multiplexing

The first and last stages are passive components, e.g. resp. AWG's and star couplers. The 3 intermediate stages consist of single-wavelength optical packet switches. Every OPS has a limited number of ports (either the equal to the number of input ports of the node, or to the number of wavelengths). In this way, direct address lookup is possible: the packet label is the memory address, the memory content is the destination port address and the packet priority.

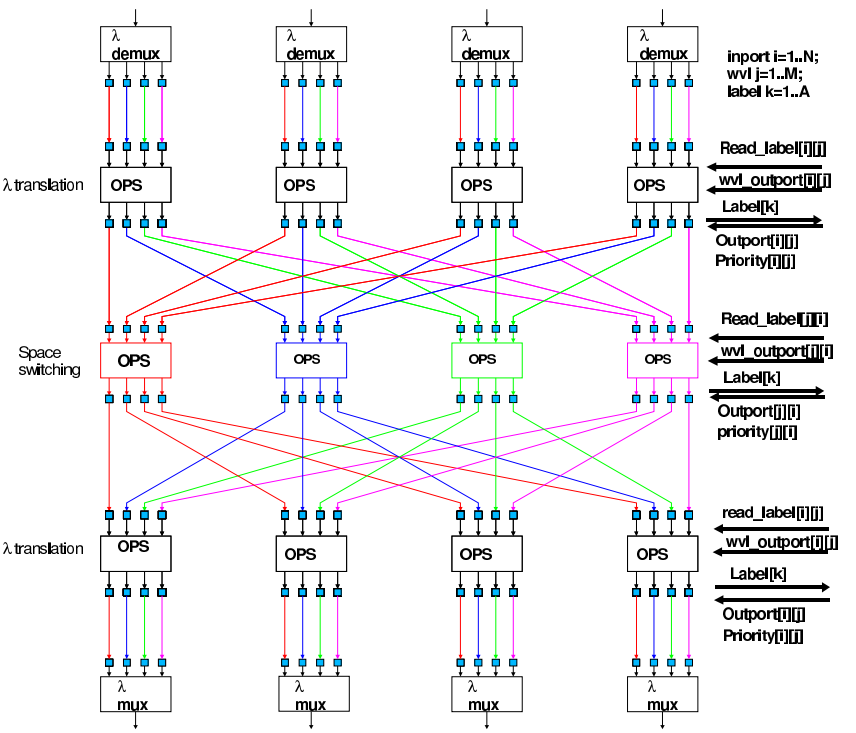

Figure 3. Modular DWDM OPS node architecture

A node of this type is non-blocking [6]. There are many configurations which result in the same final switching arrangement:

- Suppose the OPS node has n input ports and m wavelengths per input port.
- Suppose the switch architecture would be one large OXC, with TWC's that would translate every input wavelength for every input port to a common wavelength.
- The number of possible in-out connections for this "monolithic" OXC would be $(n \cdot m)!$
- The number of possible paths in OPSnet design would be: $m!^n n!^m m!^n$.
- For all practical cases, the latter is a much larger number than the former.

It should be noted that the non-blocking requirement is needed only to allow the node to act as a circuit switch. For a pure OPS, the last switching stage could be omitted.

4. Optical packet switch design

As a result of the modular nature of the DWDM OPS node, the architecture comprises a number of smaller OPS which operate on a single wavelength. The design of this OPS is now discussed in detail.

The OPS architecture can be illustrated as follows (Fig. 4):

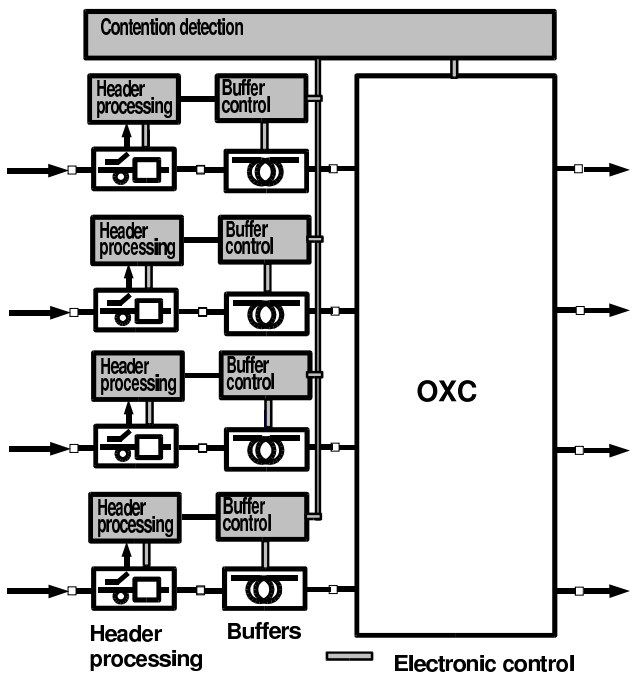

Figure 4. Optical packet switch architecture

- Optical part

  The optical part of the single-wavelength OPS consists of:

  – a space switch matrix (OXC)
  – a module to process the packet header (1 per input port)
  – an optical buffer (1 per input port)

- Electronic part

  The electronic part consists of:

  – a module to process the packet header (1 per input port)
  – a module to control the buffer (1 per input port)
  – a contention detection module (1 per input port)

4.1 Optical part

The optical part of the single-wavelength OPS has the following basic functionality: every packet arriving at any input port must be switched to an output port, which is determined by the packet label. This means a **space switch matrix** is the essential building block for the OPS. As it must allow to switch optical data from all input ports to any output port simultaneously, the switch should be non-blocking.

To be able to determine the destination port address and packet priority, the packet header must be read. Therefore, the optical input signal must be monitored. In case the header processing detects a corrupted header, there must be the possibility to drop the packet. These functions are combined in the **packet header processing module**.

Furthermore, in the case of contention, priority based buffering may be applied. Thus, an OPS needs an **optical packet buffer**. The OPSnet design is based on an in-line packet buffer at every input port. This means every packet goes in the buffer before entering the switch. This design is preferable over the more conventional approach [5, 14] were packets are switched to the buffer in case of contention. With the default buffering scheme, the packet order can be conserved, which is not possible for the conventional approach. Furthermore, the conventional design requires twice as many ports on the space switch because every buffer needs a dedicated port. This is obviously not the case with the in-line buffer design. Lastly, the contention control is much simpler, because in case of contention the packet is just held in the buffer, instead of having to be switched to the buffer.

4.1.1 Packet header processing module. The **packet header processing** module (Fig. 5) monitors the optical signal via a splitter, (e.g. a directional coupler) which splits off a fraction of the optical power for conversion to an electronic signal. The module drops packets with corrupt headers, e.g. via a space switch or with a modulator. Using a modulator has the advantage that the same device could also be used for changing the header. The module also comprises a delay line to align the optical packet timing with the electronic signal timing. This is necessary because the header extraction and serial-to-parallel conversion of the header bits introduces a latency.

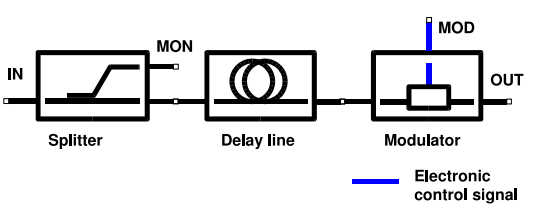

Figure 5. Optical header processing module

4.1.2 Buffer module. The **optical buffer module** is the most complex block in the optical part of the OPSnet design. The design (Fig. 6) is called a parallel recirculating packet buffer. The basic idea is that every packet has its own buffer. All packet buffers can release the packet at multiple points in the cycle. The individual packet buffers are connected in parallel. To switch packets to the buffer, an optical demultiplexer is required, essentially a space switch with a single input port. The outputs of the packet buffer are combined using a passive multiplexer. The buffer control ensures that, at any instant of time, only one packet leaves its buffer bound for its destination port. Multiple simultaneous releases are allowed if different packets are forwarded to different ports.

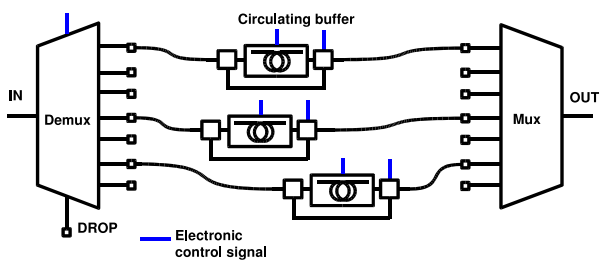

Figure 6. Parallel circulating buffer design

The main reason for this design, as opposed to a "serial" buffer which would contain a number of packets in a single recirculating loop, is to maximize the reinsertion probability of the packets. Packets in a serial buffer take much longer for a round trip. Furthermore, because every packet buffer is independent, it is easier to shift packets in and out of the buffer. Also, this design allows for a much more sophisticated control, as the decision to shift the packet out of the buffer can be based on the packet order and packet priority.

4.1.3 Space switch module (OXC). The function of the **space switch module** is to switch optical signals from any given input port to any given output port. The switch must be non-blocking, i.e. it must always be possible to configure all paths from input port to output port at the same time. Essentially this means that, if the switch has N input ports, it must have at least $N!$ possible states. For convenience, we assume in this design that $N = 2^n$. With this assumption, exactly n bits are needed to control the switch.

For the functional design, the most important design parameter for the space switch is the switching speed, as this determines the minimum gap between 2 packets (no data arrive at a given input port while this port is being switched from one output to another). The minimum gap width, combined with the maximum packet length the buffers can handle, determine the maximum possible throughput.

A possible implementation of the buffer and space switch design, based on a tunable wavelength converter/wavelength router technology [5],[7] is being considered [8].

4.2 Electronic part

The electronic part of the OPS (schematically represented in Fig. 7) controls the space switch matrix and the optical buffers using the information in the packet header. The control system consists of two modules for every input port; a module to process the packet header and a module to control the optical buffer, plus a common module for contention detection. The control system reads the packet header and performs a lookup of the destination port address and the packet priority. The packets are then switched to the buffer, the buffer control ensures every packet has its own circulating buffer with the correct length. The contention detection module indicates whether the destination port is free for a given packet. If not, the packet remains in the buffer. The system is completely asynchronous. Although the design is not technology-specific, the only possible implementation would be a deep-submicron CMOS or SiGe ASIC [9, 12, 13], as other implementations would be far too slow. In particular the destination port address lookup must be as fast as possible, and thus requires the use of embedded SRAM.

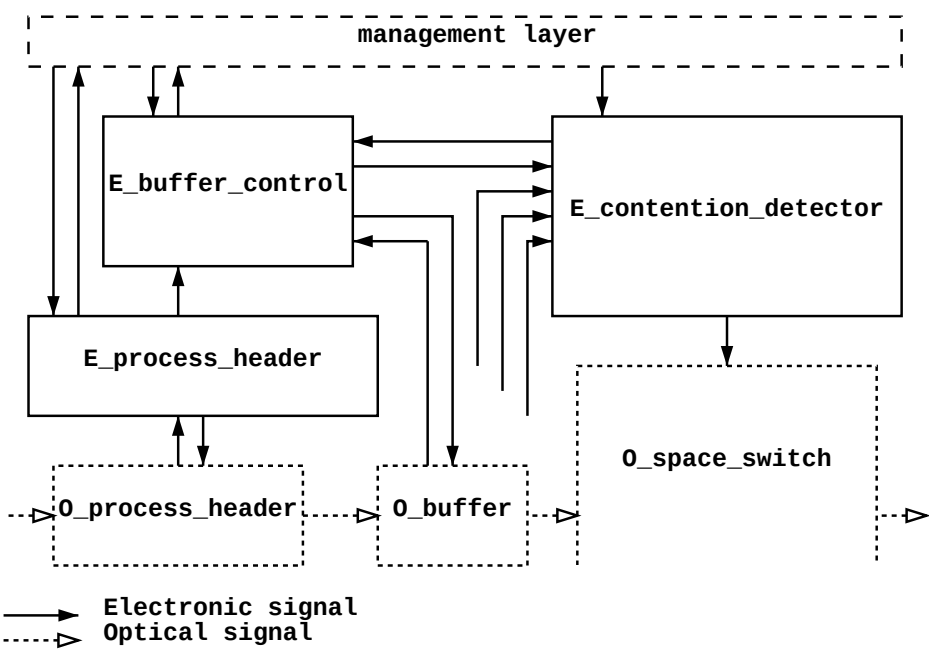

Figure 7. Electronic control circuit

4.2.1 Header processing module. The header processing module (Fig. 8) consists of 2 submodules: a module which reads the packet header data and an asynchronous RAM in which the destination port addresses and priorities are stored. The output signals of the module are the destination port address,

the packet priority and the packet length. These are passed on to the buffer control modules as soon as the destination port address has been read.

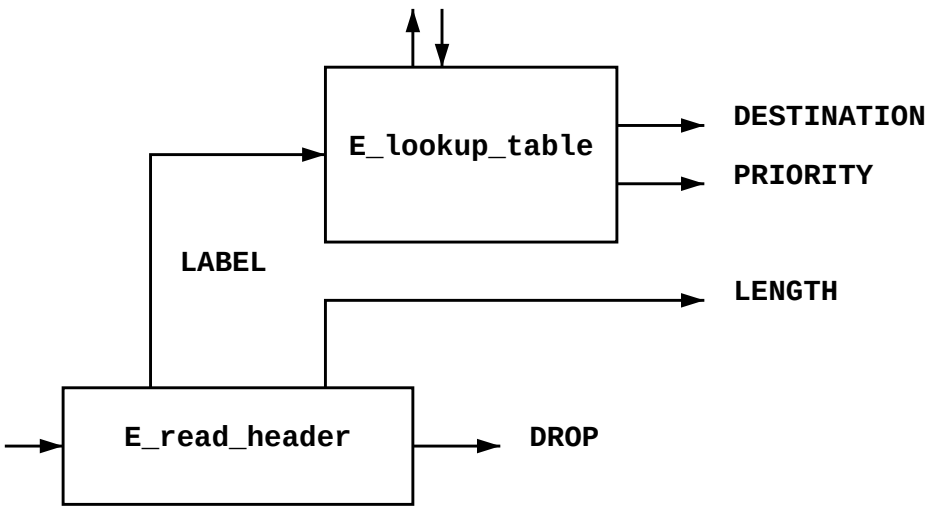

Figure 8. Electronic header processing

The asynchronous memory has an output signal which indicates whether it can be written or not. Obviously, when the memory is being read, it should not be rewritten. But this is only for a very short time after the label is clocked into the memory address bus and before the word is clocked out. Essentially, the lookup table can always be rewritten except when the packet header is being processed. This means the OPS can be reconfigured on-the-fly and there is no risk of losing packets due to the reconfiguration; a major advantage of the OPS over the circuit switch approach.

The module to read the packet header comprises essentially a shift register and some logic to detect the start of the packet and check the integrity. The shift register handles the serial-to-parallel conversion. This conversion could be part of the control logic integrated circuit if the electronics were fast enough. Alternatively, it could be integrated with the InP receiver [9], or it could be implemented optically [10, 11].

The packet start is defined as a series of high bits after a series of low bits at least as long as the packet header. The detection is done on the last bits of the shift register, so that all header bits are shifted in as soon as the signature has been detected correctly. This signal is then used to derive a number of clock pulses to clock the various parts of the header. The checksum and parity bits of the label field and length field are compared to the checksum and parity as calculated from the actual label and length bits. If these do not match, the packet is dropped. This is done by raising a drop flag which controls a modulator or switch in the optical packet processing modules. Otherwise the label field is clocked into the RAM and the length field is clocked into a latch.

4.2.2 Buffer control module. The buffer control module (Fig. 9) is, not surprisingly, the most complex module in the control system. It controls the state of the input demultiplexer, which determines to which buffer a packet is switched; it controls the state of the individual buffer, to determine whether a packet has to remain in the buffer or can be switched to the output multiplexer. The output multiplexer is a passive device (without electronic controls).

The input demultiplexer control consists of a FIFO register which stores the addresses of the empty buffers. When a packet arrives, it is switched to the first empty buffer. This address is taken off the stack. When a buffer gets emptied, the freed address is pushed on the stack. If the stack is empty, which means that all buffers are filled, the packets are switched to a drop port.

To avoid loss of low-loss-class traffic due to buffer overflow, a flag is implemented to indicate that the buffer is nearly full. When this flag is raised, best-effort class packets will be dropped to keep the buffer from overflowing.

Once a packet is switched into a circulating buffer, it requests at every exit to leave the buffer, which triggers a chain of decisions to determine whether the request can be accepted or not.

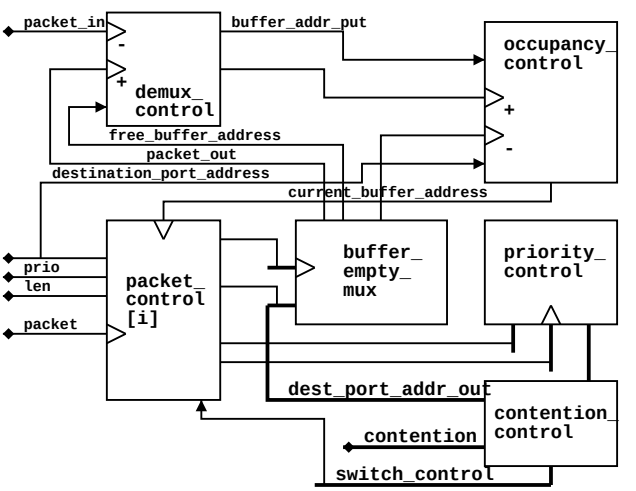

Figure 9. Electronic buffer control

One of the key tasks of the buffer control module is to ensure that the packet order can be conserved, which is implemented as follows: when a packet bound for a certain destination port is switched to a given buffer, the address of this buffer is pushed onto a FIFO which belongs to the destination port address. Only the first buffer on the stack can be emptied. When this happens, all items in the stack shift one place. This ensures that all packets for a given destination port remain in the correct order.

Another task of the buffer control module is to prioritize the packets: packets with a higher priority (minimal-delay class) should leave the buffer earlier. This is achieved by a combinatorial logic circuit which checks the priority of each

packet that can leave the buffer. Only the packet with the highest priority is allowed to leave.

When a packet is occupying the output multiplexer, all other packets bound for the same destination port are denied access.

When a packet is cleared for both conditions (order, priority), the contention check is done. Contention detection is handled by a separate module which receives the destination port address and the packet enable signals from each input port, and indicates back if contention occurs.

4.2.3 Contention detection module. The contention detection module is the only global module in the design. It checks if a destination port is occupied. For each input port, the module receives an enable signal and a destination port address. If a destination port is occupied, and an input port tries to send a packet to it, the module signals back that the port is occupied.

5. Design methodology

For this design, following methodology was adopted: The design is done at bit level, and is purely functional. This means no physical-layer-specific models were developed. Basic components like switches, delay lines, (de)multiplexers are considered ideal.

First, the system architecture was developed and the main modules defined. Next, the building blocks for every module were designed and verified. The design stops at an abstraction level above the physical device. This means the actual implementation of a tunable wavelength converter or wavelength router is not relevant, only its functionality. Once the complete node design was finished and verified, it was ported to a higher abstraction level ("packet" level). This approach avoids the pitfalls of designing an system at a high abstraction level and then being confronted with incompatibilities in the implementation.

The first phase of the design of the DWDM OPS is implemented in the Verilog Hardware Description Language (IEEE-1364). As the name says, this language is intended to model hardware systems (in particular, but by no means exclusively, electronic integrated circuits).

The OPS design has a large number of parameters (number of ports, number of wavelengths, buffer depth, switching time, unit packet length, bitrate etc). To be able to create a parametrized model of the OPS, an object-oriented code generator has been developed (in Perl 5). The use of a code generator makes the design very flexible and less error prone. Every module is represented by an object, which generates the Verilog code. A new object is created with a number of attributes, e.g. the delay, the number of wavelengths etc. Every instance of this object has its own attributes, namely the instance name and the names of the nets to which the module is connected. An essential feature of the code generator is that it allows to use objects within objects. This way the complete hierarchical Verilog module structure is reflected in the object structure. The use of a code generator allows the co-design and cos-simulation

of electronic and optical modules. As a result, functional verification on the complete hardware design of the OPS is possible.

The finished design was ported from Verilog to C++, at a higher abstraction level. The granularity in the C++ simulator is at packet level rather than at bit level. The simulator sends streams of traffic with different distributions to an OPS with different buffer designs and variable number of ports, wavelengths and buffer depth. A number of flags can be set to change the switch behaviour, e.g. conservation of packet order.

6. Node dimensioning

6.1 Component count

One of the key reasons for adopting a modular OPSN architecture is the manufacturability of the design. The number of components for a modular OPSN compared to a monolithic approach are presented in Table. 10.

Monolithic design				
Number of OPSN wavelengths	Number of OPSN ports	Number of OPSE modules	Number of ports on the OPSE module	Number of buffer modules
4	4	1	16	16
8	8	1	64	64
16	16	1	**256**	256
32	32	1	**1024**	1024
64	64	1	**4096**	4096
128	128	1	**16384**	16384

Modular design				
Number of OPSN wavelengths	Number of OPSN ports	Number of OPSE modules	Number of ports on the OPSE module	Number of buffer modules
4	4	12	4	32
8	8	24	8	128
16	16	48	16	512
32	32	96	32	2048
64	64	192	64	8192
128	128	384	128	32768

Figure 10. Packet loss vs. data load

From this table, it is clear that the monolithic approach results in a very large number of ports per module. Consider a possible implementation of the OXC using tunable wavelength converters (TWC) and arrayed waveguide multiplexers (AWG). Then the number of different wavelengths required for a 64x64 OPSN would be 4096; the AWG needs to have 4096 channels. This kind of device would be very hard to fabricate. In contrast, the maximum number of wavelengths for the modular approach is 64 per OPSE. TWC's and AWG's with this number of channels are comparatively very easy to fabricate [15],[16].

6.2 Simulation methodology

All simulations were performed on a single-wavelength 2x2 OPS (Fig. 11). The simulator allows any number of ports and wavelengths, but the CPU time

rises more than linear with both, and the results are only weakly dependent on the number of ports. Two streams of traffic are generated, the destination port of each packet is chosen at random with equal probability.

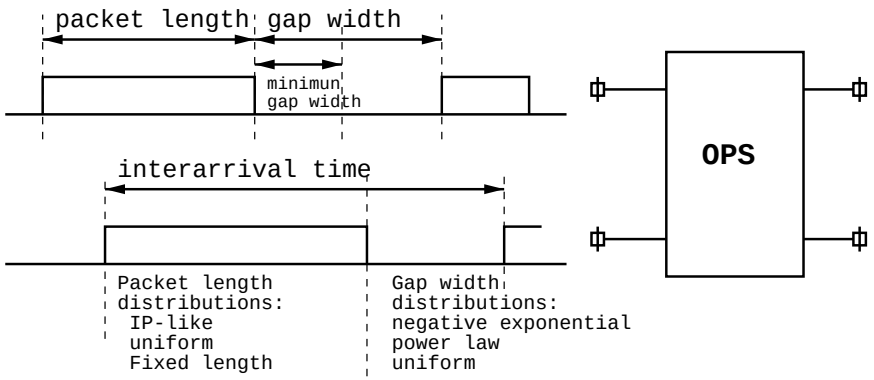

Figure 11. Packet loss vs. buffer depth

The traffic was modeled using a 2-state model: the packet length distribution can be uniform, fixed length or IP-like. The gaps between the packets are modeled using three types of distributions: uniform, negative exponential ("Poisson") and power law ("Pareto"). All three distributions have the same minimum and mean packet length. Simulations were undertaken to determine the packet loss versus buffer depth for varying network load.

6.3 Simulation results

6.3.1 Packet loss versus buffer depth. The graph in Fig. 12 shows how many packets get dropped due to buffer overflow for a varying number of buffers. The number of packets is 10^6, the packet length distribution is a typical IP-over-Ethernet distribution[17](50% 40-byte packets, 30% 552-byte,20% 1500-byte packets). The gap width distribution is a negative exponential ("Poisson") distribution with a minimum width of 50 bytes and a mean of 100 bytes.

Simulations were done with the OPS configured to conserve packet order (Ordered: yes) and not. The error flags represent 95% confidence intervals. Because the number of packets is finite, a packet loss rate of less than 10^{-6} results in 0. This leads to a lopsided distribution toward lower loss, hence a larger confidence interval for values lower than the mean.

These simulations show that a moderate buffer depth (certainly below 100) is sufficient to obtain low packet loss. It also shows that the requirement of conservation of packet order does not dramatically change the number of buffers needed (although, for the same number of buffers, it leads to losses several orders of magnitude higher).

If we take the case of a modular OPSN with 16 wavelength channels per port, a total of $16 \times 48 = 768$ recirculating buffer units per port is required for a packet loss rate $< 10^{-6}$at a combined bitrate of 1.6 Tb/s.

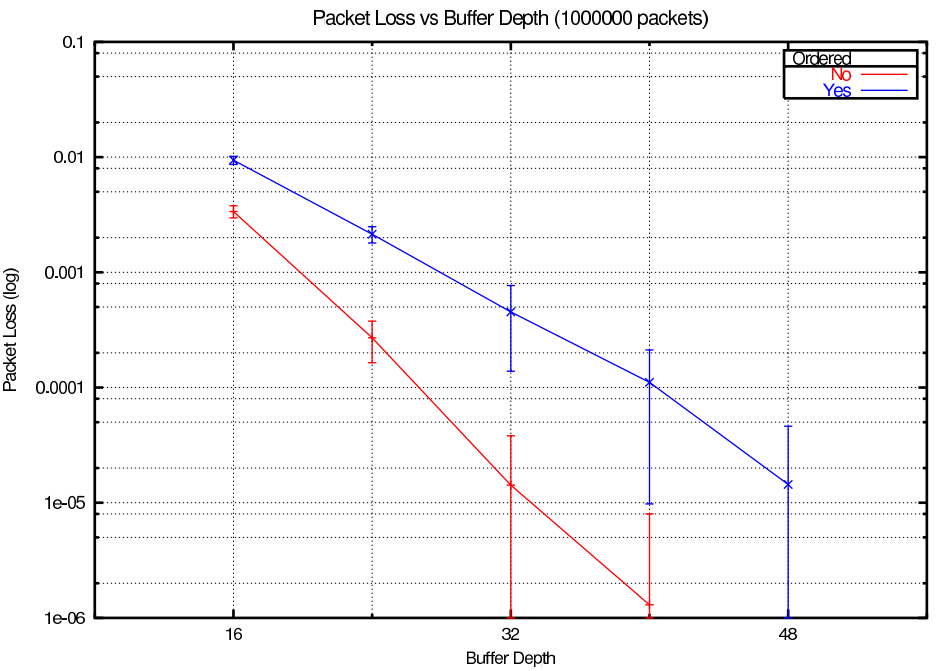

Figure 12. Packet loss vs. buffer depth

6.3.2 Influence of the network load. The next graphs (Fig. **??**) shows the influence of the network load on the packet loss. The network load can be modified either by modifying the average packet length or the average gap width. The simulations shows the effect of both changes. The packet length distribution was here a simplified IP-like distribution with only 2 packet lengths: 50 bytes and 1500 bytes. The ratio of these packets was changed to obtain various average packet lengths, while the mean gap width was fixed at 100 bytes.
For the gap width variation, the distribution is a Pareto distribution with a fixed minimum gap width of 50 bytes, the Pareto index was varied to obtained different average gap width values. An IP-like distribution with mean of 500 bytes was used for the packet length.

It is very clear that changing the average gap width has a dramatic effect: The packet loss changes over about 3.5 orders of magnitude at a buffer depth of 32 for a change in mean gap width from 75 to 125 bytes, which corresponds to a change in load from 87% to 82%. So a 5% change in the load reduces the packet loss with almost 4 orders of magnitude.

Changing the average packet length has only a weak effect: The packet loss changes over less than 1 order of magnitude at a buffer depth of 32 for a change in mean packet length from 166 bytes to 340 bytes, which corresponds to a change in load from 62% to 77%. So a 15% increase in the load increases the packet loss with less than 1 order of magnitude. As can be seen from the graph,

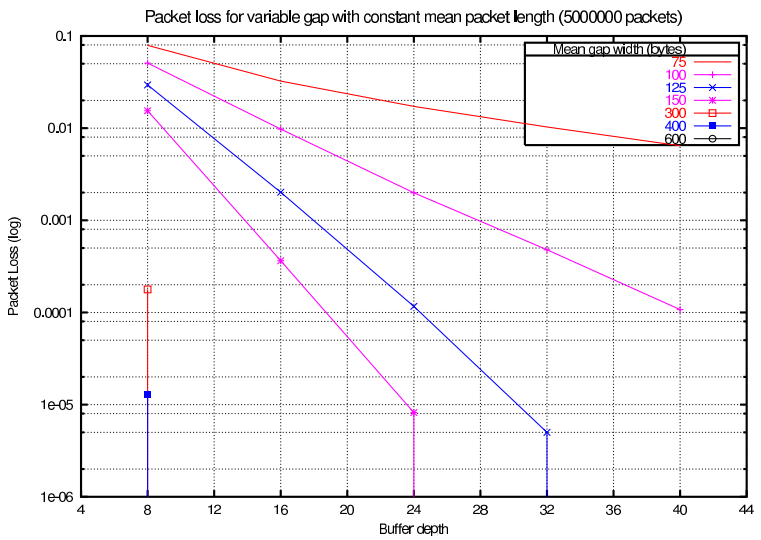

Figure 13. Packet loss vs. buffer depth for varying average gap width

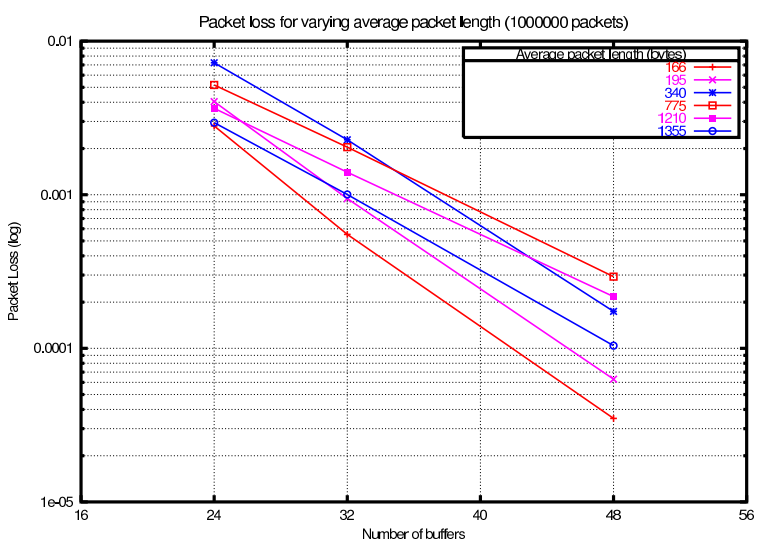

Figure 14. Packet loss vs. buffer depth for varying average packet length

this is the worst case: the packet losses for average packet lengths of 195 and 1355 are equal.

From these results it is clear that high network load with low packet loss can easily be achieved. The load can be increased by increasing the average packet length (e.g. by aggregating shorter packets into longer ones) whilst keeping the average gap width at a safe value.

7. Summary and conclusion

In this paper, a new generic architecture and design for an asynchronous optical packet switch has been presented. The architecture is GMPLS-compliant, DWDM-capable and fully scalable to. The optical packet switch employs a new buffer design called in-line parallel recirculating buffer. This type of buffer has a very high egress probability, which makes is very well suited to statistical multiplexing for contention resolution. The buffer can be configured to conserve the packet order and prioritize the traffic. Calculations show the advantages of the modular architecture for manufacturability and scalability. Initial simulations show that it is possible to obtain low packet losses ($< 10^{-6}$) with high load (> 80%) for low buffer depths (about 50 buffers). This architecture is very attractive because it is generic, i.e. not dependent on a particular optical or electronic technology. This makes it very easy to scale the design to higher bitrates and larger numbers of ports and wavelengths.

Acknowledgments

This research was carried out in the framework of the EPSRC OSI project OPSnet.

References

[1] A. Banerjee, J. Drake, J. Lang, B.Turner, D. Awduche, L. Berger, K. Kompella, Y. Rekhter, “Generalized Multiprotocol Label Switching: An Overview of Signalling Enhancements and Recovery Techniques”, IEEE Comms. Mag., Jul 2001, p144-151

[2] A. Banerjee, J. Drake, J. Lang, B.Turner, K. Kompella, Y. Rekhter, “Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements”, IEEE Comms. Mag., Jan 2001, p144-150

[3] F. Le Faucheur et al., “MPLS Support of Differentiated Services”, RFC 3270, May 2002, http://www.ietf.org/rfc/rfc3270.txt

[4] K. Nichols et al., “Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification”, RFC 3086, Apr 2001, http://www.ietf.org/rfc/rfc3086.txt

[5] D. K. Hunter, M. H. M. Nizam, K. M. Guild, J. D. Bainbridge, M. C. Chia, A. Tzanakaki, M. F. C. Stephens, R. V. Penty, M. J. O’Mahony, I. Andonovic, I. H. White: “WASPNET - a Wavelength Switched Packet Network”, IEEE Communications Magazine, March 1999, pp120-129

[6] M. Collier, T. Curran, "The strictly nonblocking condition in three-stage networks," Proc. ITC-14, 1994.

[7] D. Klonidis, G. Zervas, C. T. Poloti, D. Simeonidou, M. O’Mahoney: “Fast control circuit for a GCSR tunable laser for applications in optical packet switching”, PHOTON02, Sept 2002

[8] W. Vanderbauwhede, D. Harle, OPSnet Project Progress Report 2002 (internal report)

[9] T. Masuda, K. Ohhata 1 , E. Ohue, K. Oda, M. Tanabe 1 , H. Shimamoto 1 , T. Onai, K. Washio, “40Gb/s Analog IC Chipset for Optical Receiver using SiGe HBTs”, ISSCC 1998

[10] K. Chan, F. Tong, C.K. Chan, L.K.Chen, W.Hung, “An All-Optical Packet Header Recognition Scheme for Self-Routing Packet Networks”, OFC 2002, p284-285

[11] K. Ema, J. Ishi, H. Kunugita, T. Kondo, Japan Science and Tech. Corp., Japan; T. Hiramatsu, T. Takahashi, Y. Kato, T. Ban, “All-optical temporal-spatial conver-sion using a four-wave-mixing process in a self-organized quantum-well material”, CLEO 2000

[12] http://www.tsmc.com

[13] http://eu.st.com/stonline/prodpres/dedicate/asic/asic.htm

[14] K. J. Warbrick, P. R. Roorda, D. Pugh, “Performance and Scaling of a Recirculating Optical Buffer”, LCS 2000

[15] M. Kohtoku, H. Sanjoh,S. Oku ,Y. Kadota,Y. Yoshikuni, “Polarization Independent Semiconductor Arrayed Waveguide Gratings Using a Deep-Ridge Waveguide Structure”, IEICE Trans. Electron. Vol.E81-C No.8 pp.1195-1204 1998/8

[16] N. Kikuchi,”Monolithically integrated 64-ch WDM channel selector”, http://www.phlab.ecl.ntt.co.jp/eng/theme/2002/e2002a_01.html

[17] K. Claffy, G. Miller, K. Thompson, “The nature of the beast: recent traffic measurements from an Internet backbone”,23 April 1998, INET 1998, http://www.caida.org/outreach/papers/1998/Inet98