OptiShare: A Dynamic Channel Sharing Scheme for Power Efficient On-chip Optical Architectures

Eldhose Peter Smruti R. Sarangi
Department of Computer Science and Engineering
Indian Institute of Technology
New Delhi, India - 110016
Email:{eldhose,srsarangi}@cse.iitd.ac.in

***Abstract*—In this paper we explore a new class of optical networks that try to reduce static power using laser modulation techniques as well as by sharing optical channels. In our proposed scheme, *OptiShare*, a station can only source power from a segmented tree based power waveguide only if it has one or multiple tokens. A station arbitrates for a token before sending a message, and then releases it. At the end of an epoch, we aggregate usage information from across the network and predict the number of units of power (1 unit = 1 token) that should be supplied in the next epoch. Such an approach helps us conserve static power, and share optical power equitably across stations. *OptiShare* is 14% faster than a traditional network that does not share waveguides, and consumes 20-30X less energy than systems that do not use laser modulation.**

I. Introduction

On-chip optical communication is posed to be a disruptive technology in tomorrow's processors. It offers significant advantages as compared to traditional electrical networks in terms of latency as well as bandwidth. However, before its commercial adoption some of its important shortcomings such as high static power consumption need to be addressed. The reasons for the high static power consumption of optical networks can be traced back to the basic nature of optical physics. Since it is not possible to store photons, light sources (unless separately modulated) need to be on all the time. This leads to a wastage of power. As a result it is necessary to devise schemes to reduce the wastage of optical power.

There are two main methods for generating power for on-chip photonics applications. We can either use an off-chip laser, or arrays of on-chip VCSEL or EEL lasers. All types of lasers have limited wall plug efficiencies (20-35%), and thus dissipate most of the consumed energy as heat. On-chip lasers dissipate this energy inside the chip's packaging thus making it difficult to remove the additional heat. In comparison, off-chip lasers dissipate this heat outside the chip's package, and thus are arguably more suitable for large server processors. However, to reduce static power, we need to modulate the laser and generate only as much optical power as required (see [1, 2, 3]).

A standard approach for modulating the laser is [1, 3] to divide time into epochs, predict the power required for the next epoch, and modulate the laser accordingly. This has proved to be a very effective approach for chips with up to 128 cores. However, for larger systems this approach is not suitable because there are a lot of unused cycles in each epoch in which the laser remains on, yet there is no traffic.

We propose a different approach in this paper. We propose to share power between the optical stations (transceivers). If a station needs power to transmit a message, it first needs to get one out of N circulating tokens, source power from one of the power waveguides, and then send messages. The laser's power output is proportional to the number of tokens, and is adjusted every epoch based on the predictions for the next epoch. Additionally, we propose a similar scheme for arbitrating for multiple data waveguides. This scheme allows a station to transmit several messages simultaneously for increasing performance.

II. Related Work and Background

A. Laser Modulation

We can reduce static laser power consumption by either turning off the laser or by reducing the laser output power based on predicted network utilization. For example, the authors of the Ecolaser [2] work propose to turn off the laser when there is no traffic. Ecolaser does not use any prediction mechanisms per se. On similar lines Kurian et al. [4] propose a 1024 core system that uses on-chip Germanium based lasers, which turn ON/OFF within 1ns and thereby reduce static power consumption. In contrast, the approach in the Probe [1] project is to use a prediction mechanism based on network usage to modulate the laser power. ColdBus [3] uses a different prediction based approach based on the addresses of the PCs of memory instructions to predict the activity (proportional to laser power) in the subsequent epoch. The authors obtain better results than Probe for a 64 core system. Note that in these works (Probe and ColdBus), we predict the network usage per optical station. However, in OptiShare, we predict the power output of the laser at the granularity of 16 stations (not on a per station basis).

B. Channel Allocation

One of the most common types of optical buses is the SWMR bus [5], where a transmitting station is connected to all the other receiving stations. To save power receivers can be activated on demand. It is not possible to share the optical channel (bus) between stations with this design. In contrast a

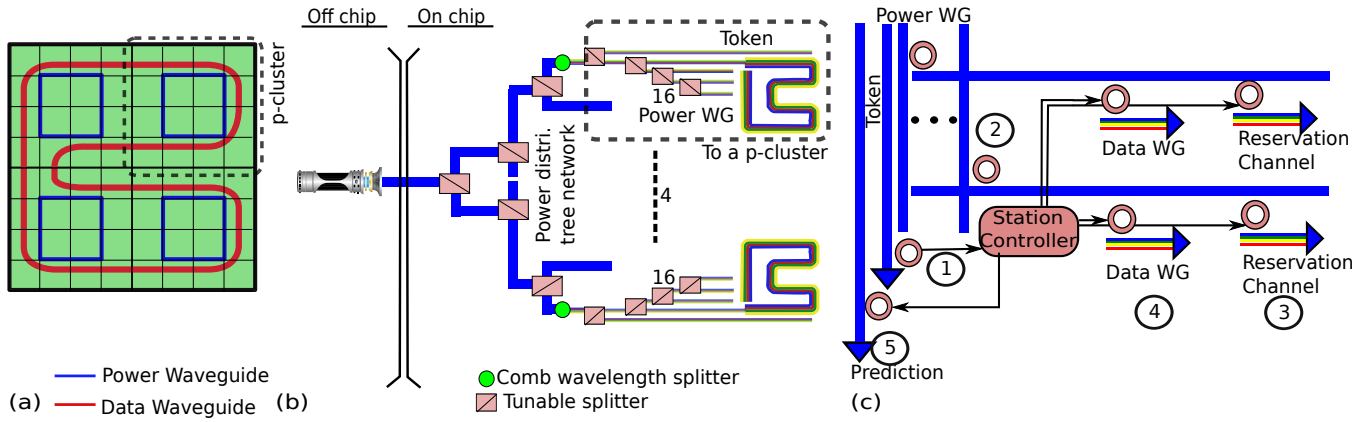

Fig. 1: (a)Architecture (b)Power distribution WG (c)At the time of sending message, station controller(SC) collects token, draw input power, modulate it, send the reservation and data signals through the data WG corresponding to the received token. At the end of each epoch, SC sends prediction signal to laser controller.

MWSR (multiple writer single reader) [6] bus has one waveguide per receiver and senders need to arbitrate to get access. A MWMR (multi writer multi reader) waveguide is a hybrid where multiple writers and readers are physically connected to the same optical waveguide. However, if designed properly a MWMR bus can help us reduce laser power by allowing us to effectively share optical power between transmitting stations. Additionally, we reduce the number of ring resonators with this design, which helps us reduce trimming power and losses associated with light passing through non-resonant resonators.

Several practical implementations of MWMR buses are as follows. In the channel borrowing [7] scheme, a few (typically 2) nodes share a channel. Here, we split the communication into two parts: upstream and downstream. The i^{th} node shares the channel with the $(N-i-1)^{th}$ node, where N is the total number of nodes. SUOR [8] and OrNOC [9] suggest similar schemes using channel segmentation where multiple transactions can happen through the same waveguide simultaneously(if the paths of communication do not intersect). They rely on arbitration. In comparison, an approach by Zulfiqar et al. [10] suggests wavelength stealing where a node can use channels belonging to other nodes. Message collisions are detected using erasure coding.

III. Design of OptiShare

In this paper we implement *OptiShare* in a system of 256 cores. Each optical station is connected to 4 cores using electrical links to form a node/cluster. As shown in Figure 1(a), we divide the entire system of 64 nodes into 4 equal parts of 16 nodes (referred to as *p-clusters*). The entire system of 64 nodes($256\ cores/4$) is connected via a data network(shown as a serpentine ring) for passing messages between nodes. The data network is a bundle of 64 partially shared reservation assisted [5] (receiver is activated on demand) MWMR waveguides.

A. Power Distribution

Figure 1(b) shows the power distribution network with a single off-chip laser source. The optical signal is distributed to all the p-clusters via a two level tree network. The tree has 4 leaf nodes (each connected to a p-cluster). In every p-cluster, we have 16 power distribution waveguides. Each such waveguide carries enough power for a single unicast at a time. Thus, the optical power at the leaf nodes of the tree can take values in the range of 0-16 units(since there are 16 power waveguides). Based on traffic predictions, we decide how many power waveguides should be activated at a time.

The junctions in the tree are made of single cycle tunable splitters [11]. These splitters help us split the optical signal between two branches of the tree based on the traffic requirements of nodes in those branches. As the requirements of the nodes change, the split ratios of each of these splitters change. We propose to change the split ratios at the beginning of each epoch depending on the predictions made in the last epoch.

For the power distribution tree, we use the method proposed by Peter et al. [12] to configure the split ratios. At each internal node we assume that the power can take a 5-bit value (up till 32 tokens) in either branch. Thus, there are a total of $32 \times 32 = 1024$ combinations (10 bits). We store a 10-bit table in hardware, where each entry stores the input power (measured in terms of tokens) to the splitter and the split ratio (6-bit number). We need to access this table thrice (once for each internal node). It will take 2 cycles [12]. The input power to the tree will be the same as the laser output power entering the chip. After factoring in the coupling loss, we can compute the laser's output power (computed using a table, see [12]).

B. Data Network

The data network has 64 parallel waveguides clustered into groups of 16. Each such group is assigned to a p-cluster. Additionally, each p-cluster has a token waveguide. We transmit 16 wavelengths on this waveguide. A token uniquely corresponds to a distinct wavelength. Each data waveguide is a MWMR waveguide. Stations in only one p-cluster can write on it, and all the stations can read from it (see Figure 1(c)).

16 stations in each p-cluster need to arbitrate for tokens. We use a simple two pass arbitration scheme (similar to [3, 13]) where the token waveguides makes two passes around the

stations. In the second pass, a station checks for the existence of the token, and then tries to divert the corresponding wavelength in the first pass of the waveguide. The arbitration is not necessarily fair, but is very fast and effective in practice. (1. i^{th} implies i^{th} waveguide, 2. Multiple messages at a time)

C. Prediction

Each station in a p-cluster sends the top two MSB (most significant) bits of the wait time (measured in cycles), and the top two MSB bits of the number of pending requests to a dedicated circuit. This dedicated circuit adds the values to produce two numbers: $\mathcal{T}$ (total wait time) and $\mathcal{N}$ (number of pending requests). For adding 16 2-bit numbers, we use four 256 entry tables to give us 4 partial sums (each 4 bits). We add these numbers with a tree of adders. It is possible to finish these operations in 1 cycle (500 ps). We also maintain a history register that contains 10 bits. The i^{th} bit contains the MSB of the number of tokens in the i^{th} previous epoch. We use these 10 bits to access a pattern table that saves a predicted value P for the number of tokens. The predicted value (N) of the number of tokens for the next epoch is given by this formula:

$$N = \begin{cases} P-1 & (\mathcal{T}+\mathcal{N}) < T_w/2 \\ P & T_w/2 \leq (\mathcal{T}+\mathcal{N}) < T_w \\ P+2 & (\mathcal{T}+\mathcal{N}) \geq T_w \end{cases}$$

We subsequently, update the pattern table with the predicted value (N), and add the number of tokens for all the four p-clusters. We empirically found $T_w = 16$ to be the best choice.

IV. Evaluation

To evaluate *OptiShare*, we use benchmarks from the Parsec [14] suite. We use Tejas [15], a cycle accurate architectural simulator written in Java to simulate our benchmarks. It has been thoroughly verified against native hardware and has been shown to be more accurate than competing simulators. The parameters for the optical network are similar to those used by the authors of the ColdBus [3] paper. A few of the simulation parameters are shown in Table I.

We compare *OptiShare* with 3 other configurations. The *swmrSave* configuration is similar to *OptiShare*, but it uses SWMR channels instead of the partially shared MWMR channels that we use. A node can send messages through a dedicated waveguide that it owns. Hence, a node cannot send multiple messages at the same time. However, it can access the token and power waveguides as *OptiShare*. The *noShare* configuration augments *swmrSave*, and here the laser is kept ON all the time. We do not modulate the laser. We thus do not have the notion of tokens here. The *fullShare* configuration extends *noShare* by making converting all SWMR waveguides to MWMR.

A. Analysis of Contention

Table II provides details about the contention at stations and about our prediction scheme. All the three columns show the values for the *swmrSave* configuration. The first column shows the average number of cycles that each request waits in the queue at each optical station. Benchmarks in which requests wait longer at each station due to contention are expected to benefit more by using the *OptiShare* scheme, because we can use multiple waveguides to transmit more than one messages simultaneously. *facesim*, *fluidanimate*, *raytrace* and *streamcluster* show the highest values of contention – ranging from 7-9, while *vips* has the lowest amount of contention (0.3). The second column shows the number of requests that get shifted to the next epoch because laser power is not available in the current epoch. The mean value is just 0.01 (per station). This means that our predictor works well in predicting future requests. The third column shows the average number of requests that had to wait till the next epoch because the laser was completely off. The mean value is 0.05 (per station) per epoch.

Parameters	
Cores	256(2 GHz)
Retire, Issue W	4, 4
ROB, IW, LSQ	186,54,64
Int alu, mul, div Latency	2, 1, 1 1, 3, 21
Float alu, mul, div Latency	2, 1, 1 3, 5, 24
Int, Float Reg	160,144
Branch Predictor	TAGE
L1 Cache	32 kB, WB, 2 cycles, 64 bytes, Assoc 4
Directory	16384,MOESI, 4bank
Shared L2	8*256kB, WB, 8 cycles, 64 bytes, Assoc 8
Main memory	250 cycle, 4 controllers

TABLE I: Parameters

benchmarks	# wait cycles due to station busy	# requests shifted to next epoch due to traffic	# requests delayed due to laser off
blackscholes	3.75	0.02	0.0001
dedup	0.72	0.02	0
facesim	7.07	0.02	0
fluidanimate	9.22	0.01	0.0002
freqmine	0.09	0	0.4445
raytrace	9.48	0.01	0.0001
streamcluster	8.04	0.01	0.0004
vips	0.3	0	0
x264	1.9	0.01	0
Mean	4.51	0.01	0.0495

TABLE II: Statistics of *swmrSave*

B. Performance

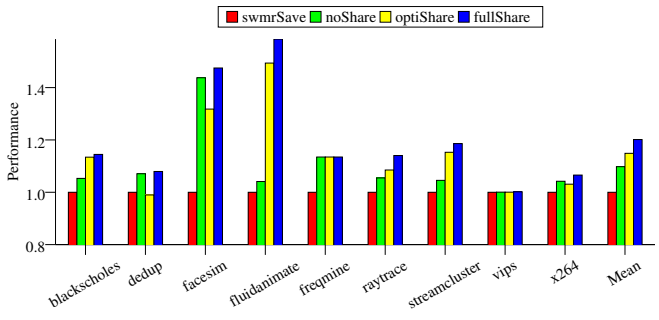

Fig. 2: Performance

In Figure 2, we show the performance comparison (in terms of simulated execution time) of the four configurations. On an average *fullShare* is the best, but it is also very power hungry(see Section IV-C). It is 20% better than *swmrSave*. We can see that the mean performance of *OptiShare* is 14% better compared to *swmrSave*. This is because a node can send multiple messages at a time. The *noShare* configuration performs 9% better than *swmrSave* because the laser was available to send data all the time in *noShare*. We do not have to wait till next epoch. In *dedup* and *facesim*, there is an anomaly because the prediction accuracies were modest. This is also the reason why we can see in Figure 3 that *OptiShare* has more average wait cycles per request in these two benchmarks. In *fluidanimate*, we demonstrate a good performance improvement in *OptiShare* and *fullShare* because

the number of messages per station was high, and this in turn created contention in the queues at the stations (second column of Table II), and *OptiShare* could tackle this issue by sharing waveguides. In *vips*, we do not see an appreciable difference in performance between different configurations because message transmission is infrequent. This transmission pattern also reduced the accuracy of our predictor.

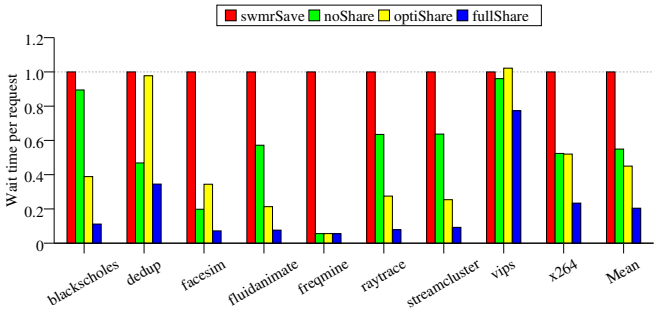

Fig. 3: Number of wait cycles per request

C. Energy consumption

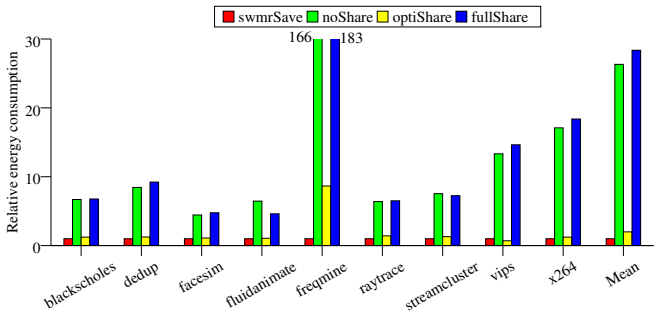

Fig. 4: Relative energy consumption

In Figure 4 we show the relative optical energy requirements of the four configurations. The average laser energy consumption of *noShare* and *fullShare* is 26 and 28 times that of *swmrSave*. This is because, in *fullShare* and *noShare*, we do not turn off the laser based on network traffic. But in the other two configurations, we turn off or reduce the output power of the laser whenever the traffic is low. Compared to *noShare*, the laser power consumption in *fullShare* is high, because of the loss in resonators and the power required to implement the token arbitration scheme. The average power consumption of *OptiShare* is twice that of *swmrSave* primarily because, in *OptiShare*, the station that has a high network activity can absorb many tokens and this can cause increased waiting time for the other stations in the same cluster. Then all the stations ask for more power in the next epoch. This increases the number of tokens used by the *OptiShare* scheme. To increase the rate of message transmission during periods of high contention, we increase the number of tokens by 2 at a time, but we reduce tokens 1 at a time in the interest of performance. *OptiShare* consumes more energy as compared to *noShare* because of this reason.

We observe a huge difference (166 and 183X) in laser power for *noShare* and *fullShare* in the case of *freqmine*, because the network is rarely used in these benchmarks and we could turn off the laser most of the time. In *vips*, *OptiShare* used a lesser number of tokens than *swmrSave* and is thus more power efficient.

V. Conclusion

In this paper we propose a novel optical communication architecture, *OptiShare*, which includes:

- A segmented, reconfigurable power distribution network that distributes the optical power to nodes based on predicted traffic.
- An intelligent, optimized and dynamic channel sharing scheme to reduce static power and the number of waveguides.

The current approach is a hybrid between traditional wavelength sharing and laser modulation schemes. As compared to a traditional network with SWMR waveguides *OptiShare* is 14% faster, yet consumes 1.8 times more energy. It is 20-30X more power efficient than networks that do not use laser modulation at all.

References

[1] L. Zhou and A. K. Kodi, "Probe: Prediction-based optical bandwidth scaling for energy-efficient nocs," in *NOCS*, 2013.

[2] Y. Demir and N. Hardavellas, "Ecolaser: an adaptive laser control for energy-efficient on-chip photonic interconnects," in *ISLPED*, 2014.

[3] E. Peter, A. Thomas, A. Dhawan, and S. R. Sarangi, "Coldbus: A near-optimal power efficient optical bus," in *HiPC*, 2015.

[4] G. Kurian, C. Sun, C.-H. O. Chen, J. E. Miller, J. Michel, L. Wei, D. Antoniadis, L.-S. Peh, L. Kimerling, V. Stojanovic *et al.*, "Cross-layer energy and performance evaluation of a nanophotonic manycore processor system using real application workloads," in *IPDPS*, 2012.

[5] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: illuminating future network-on-chip with nanophotonics," in *ACM SIGARCH Computer Architecture News*. ACM, 2009.

[6] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System Implications of Emerging Nanophotonic Technology," in *ISCA*, 2008.

[7] Y. Xu, J. Yang, and R. Melhem, "Channel borrowing: an energy-efficient nanophotonic crossbar architecture with light-weight arbitration," in *ICS*, 2012.

[8] X. Wu, J. Xu, Y. Ye, Z. Wang, M. Nikdast, and X. Wang, "Suor: Sectioned undirectional optical ring for chip multiprocessor," *J. Emerg. Technol. Comput. Syst.*, vol. 10, no. 4, pp. 29:1–29:25, Jun. 2014.

[9] S. Le Beux, J. Trajkovic, I. O'Connor, G. Nicolescu, G. Bois, and P. Paulin, "Optical ring network-on-chip (ornoc): Architecture and design methodology," in *DATE*, 2011.

[10] A. Zulfiqar, P. Koka, H. Schwetman, M. Lipasti, X. Zheng, and A. Krishnamoorthy, "Wavelength stealing: an opportunistic approach to channel sharing in multi-chip photonic interconnects," in *MICRO*, 2013.

[11] E. Peter, A. Thomas, A. Dhawan, and S. R. Sarangi, "Active microring based tunable optical power splitters," *Optics Communications*, 2016.

[12] E. Peter and S. R. Sarangi, "Optimal power efficient photonic swmr buses," in *Silicon Photonics (with HiPEAC)*, 2015.

[13] Y. Pan, J. Kim, and G. Memik, "Flexishare: Channel sharing for an energy-efficient nanophotonic crossbar," in *HPCA*, 2010.

[14] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: characterization and architectural implications," in *PACT*, 2008.

[15] S. R. Sarangi, K. Rajshekar, K. Prathmesh, G. Seep, and P. Eldhose, "Tejas: A java based versatile micro-architectural simulator,," in *PATMOS*, 2015.