ColdBus: A Near-Optimal Power Efficient Optical Bus

Eldhose Peter
Dept. of Computer Science and Engg
Indian Institute of Technology
Delhi - 110016
eldhose@cse.iitd.ac.in

Arun Thomas Anuj Dhawan
Dept. of Electrical Engg
Indian Institute of Technology
Delhi - 110016
{*eez138243,adhawan*}*@ee.iitd.ac.in*

Smruti R Sarangi
Dept. of Computer Science and Engg
Indian Institute of Technology
Delhi - 110016
srsarangi@cse.iitd.ac.in

***Abstract*—High static power dissipation is one of the largest hurdles in the widespread adoption of on-chip optical networks. There are numerous proposals in literature for the reduction of static power by predicting the activity in future time intervals, and then powering off the laser, and also by sharing optical bandwidth. We propose a novel scheme called *ColdBus* in this paper that introduces a new method of predicting the optical NoC traffic using program counter addresses; moreover, we then propose novel structures and predictors for handling mispredictions. These units allow optical transmitters to share bandwidth between them while simultaneously minimizing static power. We show that *ColdBus* consumes 4.1X less optical power than other state of the art proposals. It additionally has a speedup of 3.1%, and improves ED^2 by 18% for a suite of Splash, Parsec, and Parboil benchmarks.**

***Keywords*-Optical broadcast networks, Nano photonics, Multi-core communication**

I. Introduction

On-chip optical networking is posed to be a disruptive technology in the future. For the past 10 years, there has been hectic activity in both industry and academia in the area of designing fast and efficient on-chip optical networks. As compared to electrical networks optical networks have some inherent advantages: low latency communication (7ps/mm as compared to 35 ps/mm in electrical networks [1]), and high bandwidth (owing to the fact that 32-64 wavelengths can be multiplexed on the same physical channel [2]). However, before the large scale commercial adoption of optical networks, it is necessary to solve the problem of high static power dissipation [3, 4, 5, 6]. Static power dissipation can increase the total optical NoC power by up to 600%, and can render such networks infeasible.

Unlike electrical networks, static power dissipation is especially concerning in optical networks because of the inherent constraints imposed by physics. The dynamic power dissipation in an electrical network is mostly proportional to the number of messages sent, and static power is typically limited to $\approx 20 - 40\%$ [7]. However, in a naive optical network we need to keep the light source on almost all the time irrespective of network traffic. This causes a lot of static power loss. Researchers have taken this problem into cognizance and have proposed schemes to turn off or modulate [3, 8] the laser depending on expected traffic to save on static power. There are several steps in this process. (1) We need to predict the network utilization per station (optical transmitter/receiver) for the next epoch (interval of time). (2) Subsequently, it is necessary to modulate the optical power output of the laser, and (3) simultaneously we need to reconfigure the optical power distribution network to optimally distribute the power to each station. In this paper, we present a scheme, *ColdBus*, which solves all of these problems differently. For predicting network utilization in a subsequent epoch, prior approaches [3] predict activity based on traffic in the last few epochs. We propose a superior method that is based on predicting d-cache misses. However, it is necessary to handle mispredictions especially when the network traffic is more than what is predicted. We propose a novel technique to handle such cases, and finely balance execution time and laser power.

Subsequently, we present a mechanism to quickly reconfigure the optical network and the laser power based on some recent results [9, 8, 10]. Instead of using standard externally modulated lasers we propose to use directly modulated laser arrays (see Section II-A) that can be modulated much more quickly. Secondly, we use the results in [9] to dynamically reconfigure the optical network. The network proposed by Zhou et al. [3] takes at least 20 ns to reconfigure; however, we take 4.5 ns. Furthermore, our scheme optimally minimizes static power dissipation if we are able to predict network traffic correctly, which is most often the case.

We compare our method to a state of the art method proposed by Zhou et al. [3] that predicts traffic based on prior network activity. We observe a 3.1% improvement in performance, and 4.1X reduction in optical power consumption. In Section V we show detailed results comparing our system with baseline high radix electrical networks. We demonstrate a speedup of 27.5% and a 68% improvement in ED^2.

II. Background

Readers who are unfamiliar with the basics of optical communication, can take a look at the technical report for the OptiKit optical components library [11].

A. Laser Array

Any optical system needs a light source. Using an external array of lasers producing light at 1550 nm is the most com-

mon approach. Alternatively, we can use VCSEL or EEL lasers that are internal to the chip. However, they typically have lower efficiencies and dissipate all the power within the chip. Hence, we have used the former type. Now, there are two kinds of external lasers. The first type called externally modulated lasers use ring resonators. Ring resonators act as optical switches that can completely divert the light traveling on a waveguide (optical channel) or transmit it fully. The second type of lasers are directly modulated lasers (DML lasers), which can be turned on or off by modulating the supply current. DML lasers are commercially available and more than a dozen research prototypes have been fabricated and demonstrated [12, 10, 13].

We use an array of 32 DML lasers as the light source [10]. We assume each laser to have a peak optical power output of 180 mW (representative value). Other than peak optical power, the main concern for architects while using lasers is thermal stability, and switching time. DML lasers can safely operate on the board without thermal stability issues ([12, 10]). Their switching time is given by:

$$\tau_d = \tau_c \ln \frac{I_1}{I_1 - I_{th}} \tag{1}$$

Here, τ_c denotes the carrier lifetime, I_1 denotes the input current of the laser in the *on* state and I_{th} denotes the threshold input current. Using values: $\tau_c = 3ns$, $I_1 = 700mA$ and $I_{th} = 80mA$ [10], we find the switching time of each DML laser to be 0.36 ns. We can create a 32 step laser array by selectively powering up a subset of lasers. For example, if we need to produce 3 W of power, we can just power up 17 lasers.

B. Power Waveguide and Comb Splitters

After the laser power is routed into the chip, it is distributed to the optical stations (8-32) with the help of a structure known as the *power waveguide*. It can either be a regular set of waveguides designed as a ring [2], where each optical station diverts a portion of the power using optical splitters, or it can be designed as a tree [14]. We use a ring based power waveguide, where each station diverts a portion of the optical power depending upon its expected requirements in the next epoch.

To support DWDM (dense wavelength division multiplexing) we need to produce light at multiple wavelengths (typically 64 [2]). This can either be done off-chip with a multi-wavelength source, or can be done on-chip with comb based splitters [15], which can split monochromatic light at 1550 nm to produce light at 64 different equispaced wavelengths. We use the latter option using comb splitters.

C. Optical Network Reconfiguration with Tunable Splitters

We assume an optical network with a ring based SWMR (single writer multiple reader) topology (also see [16]). Here, each transmitting station is connected to all the other receiving stations. In each epoch we need to precisely predict the stations that are likely to transmit in the next epoch, and the number of receivers that each station is likely to transmit to. After making this prediction, there are two important problems to be solved.

Let us assume that we predict that a transmitter will transmit a message to 3 receivers. We need to find the minimum amount of optical power that will be required. Additionally, we need to find the optimal split ratios of the three beam splitters that supply power to the three receivers. A naive solution is to assign split ratios of the form: 1:3, 1:2, and 1:1 to the splitters. Peter et al. [9] show that this approach is sub-optimal because the power losses associated with the beam splitters is a function of the split ratio. They propose an optimal linear time algorithm based on a dynamic programming formulation that calculates the optimal split ratios of the splitters such that the input optical power is minimized. They show that the difference between both the approaches (optimal vs naive) can be as much as 4X for Y-junction based splitters. We use the method proposed by Peter et al. [9] to compute the split ratios of beam splitters in the network. We use their hardware implementation, which can compute both the optimal split ratios and the minimum optical power for our system with 20 stations in less than 500 ps (2 cycles at 2.5 GHz).

Next, we need a fast tunable optical power splitter that can be tuned to divert a fixed portion of power from a waveguide. Zhou et al. [3] have used MMI based splitters that take 6 ns to get reconfigured. However, faster power splitters that can get reconfigured in a single processor cycle are known to exist in literature and have been used by several highly cited papers in the past [16, 17]. We assume one such splitter in our design. It supports 16 different split ratios and can be reconfigured within 400 ps. It is similar to the design proposed by Lazaro et al. [18]. This design has a silicon ring resonator, where a part of the ring is integrated with Graphene. It is very fast (multi-GHz speeds) and requires as little as 45 mV for moving to a new split ratio. A detailed discussion on power splitters is beyond the scope of the paper, and readers can consult references to similar work in the traditional optics literature [18, 19, 20, 21, 22].

III. Related Work

The following references [3, 4, 5, 6] have taken into cognizance the fact that power dissipation in optical networks is an important problem. There are two approaches to reduce static power, which is the dominant component of power consumption in optical NoCs. The first is to modulate laser power based on the amount of current or expected network traffic, and the second is to efficiently share bandwidth between transmitting stations such that the total laser power is minimized.

Pan et al. [16] proposed the Firefly protocol, which is a very effective method to reduce static optical power. They proposed a reservation-assisted SWMR bus. Before transmitting a packet, the transmitter transmits a reservation flit on

the bus. This flit contains the IDs of the receivers that need to be turned on. The receivers turn on, read the contents of the packet, and then turn off. The reservation-assist mechanism is a part of our baseline design. Ecolaser [8] proposes to turn off the laser completely when not in use. Whenever there are some messages to send, the sender invokes the laser source to turn-on. The turned-on laser stays on for a duration known as the stay-on-time. They consider the time to turn on a laser from the off state as 1ns. Since it takes time to compute the laser power, send messages to the laser/splitters, and tune the laser, our experiments indicate that it is better to adopt a predictive approach. The idea is to collect statistics for the current epoch, and predict the behavior of the transmitters and receivers for the next epoch.

Probe [3] is a prediction based optical bandwidth scaling system, which allocates bandwidth to each sender based on its expected network usage. The authors of Probe use link and buffer utilization statistics to predict the activity in the next epoch. ATAC+ [17] also tries to minimize the static power consumption. Here the output power of the laser is changed based on network usage by changing the laser bias current. Morris et al. [23] proposed the *Pulse* architecture, which uses a nanophotonic broadcast-tree network for multicore processors that use the snoopy cache coherence protocol. They predict the sharers of each cache line (on an L2 miss), enable them, and then broadcast a message to the sharers. Chen et al. [24] propose a similar method for varying the laser power based on network activity. Li et al. [25] propose a method (ESPN) that divides an optical network into several sub-networks. Each sub-network has a separate laser source that can be independently switched on/off depending on network traffic. We compare our approach against such approaches that use link/network utilization based prediction. We show that our approach is roughly 4.1X more power efficient in Section V.

The second approach is about optimally utilizing optical power in a constrained system. Pan et al. proposed the Flexishare protocol [4] that restricts the number of optical channels. A sender needs to first acquire a set of channels before transmitting its flits. The main insight is that all the stations are not active at the same time, and thus bandwidth can be reused across stations. Arslan et al. extend this approach to create a generic wavelength stealing architecture [26]. Here, a station can steal some wavelengths belonging to another station. Whenever there is a collision, the destination tries to correctly recover the messages using erasure coding. Le Beux et al. [27] propose a wavelength sharing scheme for a ring based interconnect, where it is possible for different pairs to use the same set of wavelengths. It is possible to do so when there is no intersection in the communication paths.

Our scheme has elements from both these categories. We modulate the laser as well as have mechanisms for sharing bandwidth across stations. We compare our approach against some recent work by Zhou et al. [3] and Chen et al. [24].

IV. Architecture

A. Architecture and Layout

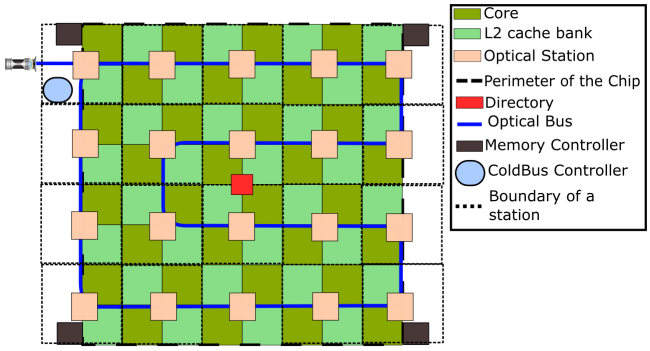

Figure 1. Layout of the chip

The layout of the chip is shown in Figure 1. The main elements are the cores, cache banks, memory controllers and the directory. Cores and cache banks are placed on the chip in a chessboard fashion. The directory is placed at the center and the memory controllers are at the corners of the chip. As shown in the figure (in dotted black line boxes), the elements are divided into groups of four [2, 16]. Within a group, the elements communicate electrically. For communication with an element in a different group, we use the optical NoC. Each group has an optical station. The different stations (20 in number) are connected by a ring-shaped SWMR waveguide.

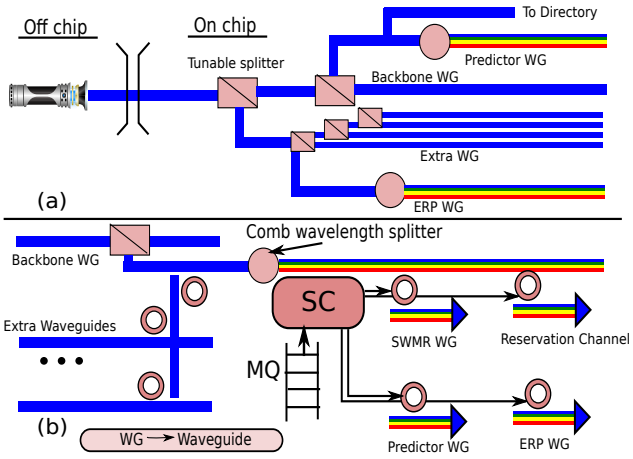

Figure 2. Design of *ColdBus*

1) ColdBus Power Waveguides: Design: The optical communication infrastructure is shown in Figure 2(a). An external InP laser array (with additional InGaAsP layers) acts as the light source. It produces light at a wavelength of 1550 nm. After the light is coupled into the chip, we divide it into two parts using a tunable splitter. The first and the more dominant part flows along the backbone waveguide.

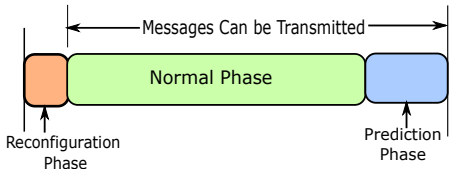

Figure 3. Operations during an Epoch

The second portion flows along the extra waveguide set (4 such waveguides), which are used to provide power just in case a station cannot source enough power from the backbone waveguide. Both the waveguides run in parallel, and connect all the stations in a logical ring configuration. Each station is connected to both the backbone and extra waveguides via a tunable beam splitter and ring resonator respectively. The splitters divert a part of the signal in these waveguides, and supply it to the stations. Subsequently, each station uses a comb splitter [15] to produce signals at 64 different wavelengths. Each transmitting station is connected to the rest of the receiving stations (SWMR $\rightarrow$ single writer multi reader configuration). It modulates 64 wavelengths to encode messages. Note that we replicate the comb splitters in each station because our optical devices to split and divert power from power waveguides operate for only a single wavelength (@ 1550 nm). The area overhead is however minimal (0.21%) (see [15]).

2) Overall Operation: We divide time into epochs. The first part of the epoch is the *reconfiguration* phase (see Figure 3). In this phase, the laser retunes itself to supply the power required for the current epoch. During this phase, no new messages are sent on the optical NoC. This phase is typically very short (1-3 cycles), and thus the loss in IPC is not significant. The next phase is the *normal* phase, when the processor operates normally. The last phase is the *prediction* phase. In addition to normal operation, all the stations predict their activity in the next epoch. They send their predictions to a centralized *ColdBus* controller.

The *ColdBus* controller first computes the split ratios of the splitters on the backbone waveguide for all the stations (other than the station with the directory) such that the input power is minimized (algorithm proposed by Peter et al. [9]). We use a tree-based waveguide with splitters that have fixed split ratios for the directory; hence, its power requirement is known at design time. We sum up the power for the backbone waveguide. Additionally, we have an extra predictor that predicts the number of stations that might simultaneously draw power from the extra waveguides in the next epoch. We then add all components, and predict the output power of the off-chip laser in the next epoch. This information is sent to the laser using a fast optical link. The laser begins retuning itself at the beginning of the next epoch (*reconfiguration* phase).

B. Normal Phase

Each station has a station controller. Whenever, an element (core/ cache bank/ directory/ memory controller) needs to send a message to an element associated with another optical station, it sends it to its station controller (SC). The SC has a queue called the *message* queue (MQ) containing 16 entries. If the MQ is full, then the SC sends a NACK message back. In this case, the sender waits for 5 cycles and retries (with random exponential backoff). Once the station detects the MQ to be non-empty, it tries to send the message over the optical network.

The first step is to detect if the station is turned on (receiving power from the backbone waveguide) in the current epoch. If it is, then we can source power from the backbone waveguide, and use it to send messages to a set of receivers. However, we might have predicted in the previous epoch that a station is turned off. In this event, we cannot source power from the backbone waveguide; rather, we need to source it from one of the 4 extra waveguides. There might be multiple stations that simultaneously need power from the extra waveguides, and thus we need a mechanism to perform arbitration, which is as follows.

We have a separate extra-request-response waveguide (ERP), where we transmit a maximum of 4 wavelengths (tokens) (similar to [4]). It makes two passes through the stations (like a coiled ring). We call the rings as the first-pass and second-pass rings respectively. Stations intending to source power first select a token (wavelength) at random and check the second-pass ring for the presence of the wavelength by trying to absorb it via a ring resonator, and then sending the diverted signal to a photodetector. If no signal is present, then it means that some other station is either trying to check for the token, or is using it. In any case, we back off and try to check another token. However, if we detect a signal, then we grab the token by absorbing the corresponding wavelength from the first-pass ring. If we have grabbed let's say token 3, then we source power from the third extra waveguide.

Once, power is available, stations can now proceed to send data. For all the stations other than the one that contains the directory, we support only unicast messages. In specific, we use the *Reservation Assisted SWMR* (R-SWMR) protocol, which was originally proposed as a part of the Firefly [16] protocol. For each station, the R-SWMR protocol uses an additional reservation waveguide that is used to enable receivers. Receivers are enabled just before they begin receiving flits, and after receiving the packet, they are turned off. To save power, the signalling protocol on the reservation waveguide is *active high*. The power sourced from the extra and backbone waveguides is used to drive the reservation waveguide as shown in Figure 2(b). The reservation waveguide requires 19 wavelengths (one per station). Since we have 64 bits available, we can afford two more bits per station to specify the type of the message

(request, response, control). A station reads all 3 bits (1 enable bit, 2 bits for the type) concurrently. If it needs to be enabled, then it remembers the type of the message.

Let us now consider the timing. Ring resonators, and optical components are known to switch at very high speeds ($>$ 10 GHz [28]). Therefore, researchers have used double pumped signalling (refer to Corona [2]), where data is transferred at both the edges of the clock. We use the same paradigm, and consider each time slot to be half a cycle. Now, for transmitting a memory request to a cache bank, we enable the receiver in the first half of the cycle, and then we send the address (64 bits) in the second half of the cycle. Thus, for transmitting a full cache line (64 bytes), we require 5 cycles. For the directory we use an optimally designed balanced tree for broadcast traffic (see Peter et al. [9]) . All the receivers on this network are always turned on.

C. Prediction Phase

ColdBus utilizes prediction mechanisms to vary the power in the *backbone* and *extra* waveguides. For the backbone waveguide, we use a *link predictor* based on prior work, and a novel address based predictor. We use an extra predictor to predict the number of tokens in the extra waveguide.

1) Link Activity Based Predictor: We use the link/network activity based predictor with a history of 8 bits as used in Probe [3]. This scheme predicts whether a station will transmit in the next epoch based on its behavior in the last 8 epochs. In specific, we use the second predictor in the reference, which is optimized for variable traffic.

2) **Address Based Predictor:** We consider a different method for predicting network traffic. The basic insight is that most of the traffic in a chip with the last level cache as the L2 cache arises because of L1 misses. They can be either L1 cache coherence messages or messages to L2 cache banks. The miss rate in the i-cache is negligible. We mostly have d-cache misses. If we can predict them with high confidence, then we can predict network activity. For our suite of numerical benchmarks most of the L1 misses are accounted for by a disproportionately small number of load/store instructions. Most often these are irregular memory accesses, or are reduce operations that require data from sibling caches. We thus design a predictor (referred to as *address predictor*) that tries to predict an L1 miss based on the PC of the memory instruction. It is specific to each core, and is inspired by the predictor used by Petoumenos et al. [29].

The address predictor uses the last 10 bits of the PC to index an array of 3-bit saturating counters. We set a threshold of 5/8 to infer a 1 (miss). This gives the best results, and the accuracy is above 90%. However, the main problem is that at fetch/decode time we do not know in which epoch the load or store will be sent to the memory system. In complex out-of-order pipelines the interval between decode to execute or retire is highly variable.

We designed an epoch predictor that has two 1024 entry tables: one for stores, and one for loads. We use the last 10 bits of the PC to access each table. It contains the time (measured in epochs) from decode to the time the load/store is sent to the memory system. Each entry in the table contains the length of the intervals from decode to execute for the last 5 epochs(each entry = 4 bits each, 1 unit=1 epoch). We compute a weighted average of the values (the earliest entry has a weight of 1, the next has a weight of 0.5 and so on). The SC collates the predictions from all the elements in the station. Using information theoretic analyses we found out that with a perfect epoch predictor, the address predictor is better than the best possible link based predictor.

3) **Extra Predictor:** Each station sends a recommendation to a global extra predictor. If a station waited for more than 10 cycles to get an extra token, then it recommends increasing the number of extra tokens by 1 in the next epoch. If it got all tokens in less than 3 cycles, it recommends decreasing the number of tokens by 1; otherwise, the prediction is to keep the same number of tokens. The extra predictor uses the most popular recommendation.

4) Calculation of Power for the Next Epoch: At the end of the *normal* phase, the stations send their results to the laser controller. Each station needs to send 3 bits (1 for the backbone WG(on/off), 2 for the extra WG (# tokens)). We use a separate predictor waveguide (PWG), which is shared across all the stations. Each station is assigned 3 wavelengths. 20 stations require 60 wavelengths, which is less than the maximum of 64. At the *ColdBus* controller, we follow the same procedure as discussed in [9]. We compute the total optical power requirement (8 bits), the split ratios of each splitter (6 bits), and then send this information to the off-chip laser as well as to all the tunable beam splitters. Note that both the PWG, and ERP waveguides transmit light all the time, and use active low signalling such that it is not necessary to source power to transmit along these waveguides.

The off-chip laser is connected using a fast optical link (with taper based optical pins). Assuming that it is 3 cms away, it takes 210 ps (7 ps/mm) to reach it. The laser controller subsequently looks up the new laser configuration in a 64 entry (6 bit) lookup table and modulates the laser's power at the epoch boundary. It is clock synchronized with the CPU. Table I summarizes the steps.

D. Reconfiguration Phase

The three important technological requirements for fast reconfiguration at the beginning of an epoch are a fast switching laser, a preferably single cycle tunable splitter, which is used to divert power from the backbone waveguide, and a method to quickly compute optimal split ratios. We have shown in Section II how we use results in prior work to solve all the three problems.

Action	Time (cycles)
Make (address/extra) prediction	1
Collate predictions at each station	1
Send to the *ColdBus* controller	3
Table lookup: opt. config of splitters & laser power	2
Parallel Activity 1	
Broadcast split ratios for next epoch	3
Reconfigure the splitters	1
Parallel Activity 2	
Send to the off-chip laser (E/O + transmission + O/E)	2
Compute new laser configuration	1
Retune the laser	1
Total	11 cycles

Table I
ACTIONS IN THE PREDICTION AND RECONFIGURATION PHASES

Architectural Parameters	
Frequency/ Die Size	2.5 GHz/ 400 mm^2
Cores	32
Technology	22 nm
Flit size	128 bits (Double pumped)
Fetch /Decode /Issue Width	4/4/6
ROB Size /IW Size	168 / 54
Int Reg / Float Reg (arch)	16/16 (160 physical)
L1 size L2 size Directory size	32KB 2 MB (per cluster) 2 MB, fully mapped, MOESI
Associativity	1(L1)/ 8(L2)/ 8(Dir)
Latency	2(L1)/ 22(L2)/ Dir(1/10/30)
Memory Controllers	4
Memory Latency	200 cycles
Electrical NoC	
Router Delay	3 cycles (lookahead routing w/ bypass)
Link Delay/Flit Size	1 cycle/16 bytes
Router details	4 VCs, 8 buffers per port, 5 in/out ports
Topology-1	Conc. Mesh, Dim. ordered X-Y routing
Topology-2	Flattened Butterfly

Table II
PROCESSOR PARAMETERS

Optical Parameters	
Wavelength (λ)	$1.55\mu m$
Width of waveguide (W_g)	$3\mu m$
Slab height	$1\mu m$
Rib height	$3\mu m$
Refractive Index of $SiO_2(n_r)$	1.46
Refractive Index of Si (n_c)	3.45
Insertion Coupling Loss	5 dB
Input Driver Power	61 μW (at 22 nm)
Output Driver Power	134 μW (at 22 nm)
Ring Heating	26 μW per ring
Ring Modulation	500 μW
Splitter Loss	0.1 dB - 0.3 dB
Waveguide Loss	1 dB/cm
Filter Drop	1 dB
Non Linearity	1 dB
Photodetector minimum power	36 μW
Output Coupling Loss	0.6 dB
Photodetector quantum efficiency	0.8 A/W
Combined transmitter and receiver delay	27 ps [1]
Optical propagation delay	7 ps/mm
Electrical propagation delay	20 ps/mm
Laser wall plug efficiency	30%

Table III
OPTICAL SIMULATION PARAMETERS [30, 31, 3, 23]

V. EVALUATION

A. Setup

1) Optical Simulation: We conducted all our optical device simulations using Synopsys RSoft (version 8.0.2.1). Please refer to the technical report on OptiKit [11] for details of the simulated optical components including their dimensions and optical power losses. We use the *FullWave* mode for simulating waveguide bends and the *BeamProp* mode for simulating the rest of the structures. The simulation parameters are given in Table III for the drivers, and modulators. Latency and power numbers are scaled to 22 nm using the formulae proposed by Huang et al. [32]. Note that the term "laser power" in this section takes the laser's wall plug efficiency into account, and the term "optical power" does not.

2) Power and Latency Simulation: In the next step, we create layouts for the optical buses using a set of C programs. We further optimize the layouts to reduce transmission losses. For the cores, and caches, we use a layout similar to that proposed in the following references: [1, 33, 31]. We have 32 out-of-order cores, 32 cache banks, 4 memory controllers, 1 directory (designed as a 4 banked cache). The optical stations are assigned to the elements as shown in Figure 1. We conduct all our architectural simulations with the cycle-accurate Tejas multicore simulator [34]. Tejas is bundled with code from the McPat [35] and Orion [7] simulators that can be used to simulate power and energy consumption for the cores and the NoC. The architectural parameters are shown in Table II. We simulate the entire parallel sections of a suite of Splash2 [36], Parsec [37], and Parboil [38] benchmarks).We compare *ColdBus* against a state of the art architecture that uses link based predictors [3, 23, 39, 24], ring based wave guides, and a broadcast tree for the directory. This architecture does not use the extra waveguide, and extra predictor (because they are our contributions). Note that even for the architecture with the *link* predictor we use the same baseline architecture: Peter et al.'s algorithm for computing optimal configurations, and fast tunable splitters. Prior work has assumed much larger delays: Probe [3] (6ns for tunable splitters, 20ns for reconfiguring the laser), Chen et al. [24] (1 μs). Let us refer to this architecture as *ArchLink*. Lastly, note that we define *performance* as a quantity that is inversely proportional to the execution time.

B. Comparison of Predictors

Table IV shows the statistics for the address predictor (averaged across all the cores). The first point to notice is that the prediction accuracy is very high (93%). For our suite of scientific benchmarks, most of the L1 misses are in fact accounted for by a small set of instructions, and it is easy to predict this set. Note that predicting an L1 miss is not enough; we need to also the predict the epoch at which the miss will happen (columns 3-7 of Table IV). The epoch prediction hit ratio is low for 25 cycle epochs (60%). This

Benchmarks	L1 Miss Pred. Acc.	Epoch prediction hit ratio(%)				Station activity(%)			
		25	**50**	**75**	**100**	**25**	**50**	**75**	**100**
barnes	95.4	38.14	56.44	80	89.1	79.52	90.74	94.64	96.59
blackscholes	97.78	92.45	97.05	98.94	99.06	28.79	44.17	53.86	59.8
fft	94.88	49.56	58.83	85.16	94.58	77.11	89.17	94.96	97.21
fmm	69.38	45.58	74.29	85.92	87.38	6.10	9.02	11.33	13.37
histo	91.72	38.73	66.87	90.3	97.52	80.33	83.21	86.43	87.68
ocean	94.61	65.43	75.84	79.4	85.26	78.76	85.23	87.37	88.74
radiosity	97.77	68.05	88.69	95.39	97.53	30.01	38.46	45.95	55.58
stencil	98	78.67	80.53	88.81	93.24	63.52	68.19	70.12	71.04
water_nsq	98.67	52.58	84.45	87.22	90.48	33.41	44.57	55.94	63.96
water_sp	98.01	72.63	90.23	92.4	97.92	48.57	59.55	72.68	74.82
Mean	93.62	60.18	77.32	88.35	93.21	52.61	61.23	67.32	70.87

Table IV
L1 MISS PREDICTION ACCURACY, EPOCH PREDICTION ACCURACY, STATION ACTIVITY

Benchmarks	**Address**				**Link**			
	25	**50**	**75**	**100**	**25**	**50**	**75**	**100**
Percentage of epochs predicted as active, actually inactive (Mean): $P(On\|Off)$	3.76	3.75	3.86	3.35	8.46	8.05	8.54	7.89
Percentage of epochs predicted as inactive, actually active (Mean) : $P(Off\|On)$	29.25	31.99	32.57	34.68	7.42	7.11	5.19	4.18

Table V
CLASSIFICATION OF THE PREDICTION ERROR)

is because the delay in the instruction window (and also the IPC) is variable and thus accurately predicting the time from decode to execute/commit is difficult. However, the epoch prediction rates are much better for 50, 75, and 100 cycle epochs (75-95%), because at these time scales the variability is much lower. The overall trends are roughly the same for all the benchmarks other than $barnes$, which had very low prediction ratios. We attribute the reason to the fluctuating IPC in $barnes$.

The last 4 columns of Table IV show the station activity for a system with the address predictor. The station activity is measured as the percentage of epochs a station transmitted a message (averaged across all the stations). It increases from 52.61% (25 cycle epoch) to 70.87% (100 cycle epoch). The trends across all the benchmarks are mostly similar other than fmm, $radiosity$ and $water\_nsq$ that have comparatively fewer accesses to the L2 cache, and have fewer coherence messages. The trends mean that as we increase the epoch size, each epoch becomes more sparse, and the amount of laser power that is wasted increases. On the other hand, a smaller epoch size is good but it has a lower epoch prediction hit ratio. Let us now dig deeper into these opposing factors (see Table V). The percentage of time we predict a station to be active, when it is actually inactive ($P(On|Off)$) is much lower for the address predictor ($\approx 3.5\%$) as compared to the link predictor ($\approx 8.5\%$). $P(On|Off)$ has an implication on power (not performance). Let us now consider $P(Off|On)$, which has an impact on both power and performance. Here, the address predictor is far more inaccurate than the link predictor (22-34% as compared to 8.5-4%). It turns out that these epochs, which are correctly predicted by the link predictor are fairly sparse epochs, and keeping the laser on for the entire duration of the epoch is wasteful. The extra waveguide can service the power requirements of these epochs (more details in Section V-D). The other noteworthy trend is that $P(Off|On)$ increases with the epoch size (from 29 to 34%). Let us explain with an example. Assume that out of 10, 50-cycle epochs, epochs 1, 3, 4, 5, and 6 are active. The first epoch is predicted to be inactive. Hence, $P(Off|On) = 1/5$. If we now consider 100-cycle epochs, $P(Off|On) = 1/3$, which is the effect that we see in the second row of Table V.

To summarize, the link predictor keeps the stations on longer. The effect of keeping stations on for let's say 10% more time does not translate to a 10% increase in laser power. The effects are more pernicious. We are increasing the time that stations are concurrently on, and since splitter losses increase exponentially [9] with the number of splitters that are turned on, the increase in power consumption will be super-linear. Our claim is that by avoiding sparse epochs, we can ultimately get a better ED^2.

C. Savings in Laser Power

Figures 4 and 5 show the percentage of time that the laser was active for the address and link predictors respectively across different epoch sizes. The term $A25$ refers to the results with *ColdBus* for a 25-cycle epoch. Likewise, the term $L25$ refers to the results with *ArchLink* for a 25-cycle epoch. Now, for *ColdBus*, the mean laser activity increases from 24% to 38% as we move from 25-cycle epochs to 100-cycle epochs. Similarly, the mean laser activity increases from 63% to 77% for *ArchLink*. For the same benchmark, the laser activity increases with the epoch size primarily because epochs start getting sparser. There are two exceptions, $histo$ and $ocean$, which show slightly reduced activity for 100-cycle epochs in the case of *ColdBus*. This is an artifact of the timing of the message transmission events. There are two noteworthy points: (1) the laser activity for *ColdBus* is at least 2X less than *ArchLink*, and (2) the laser activity for *ColdBus* is also 50-70% of the station activity. This means that 30-50% of the epochs need to rely on power from the extra waveguide. If these epochs are sparse enough, we can expect substantial power savings without a concomitant loss in performance.

Figures 6 and 7 show the mean laser power in Watts (averaged across all the epochs). We observe that the reduction in laser activity directly translates to a reduction in laser power. The laser power for *ColdBus* is between 4-6W, and the equivalent power for *ArchLink* is 12-20W. Out of the different configurations we choose $A25$, and $L50$ as representatives from each set because they have the best performance(and also best ED^2). We compare $A25$, $L50$,

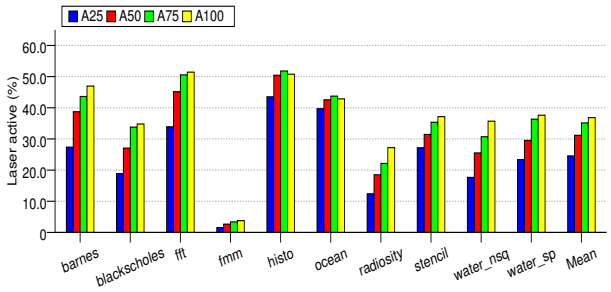

Figure 4. Percentage of the total time the laser was active (*ColdBus*)

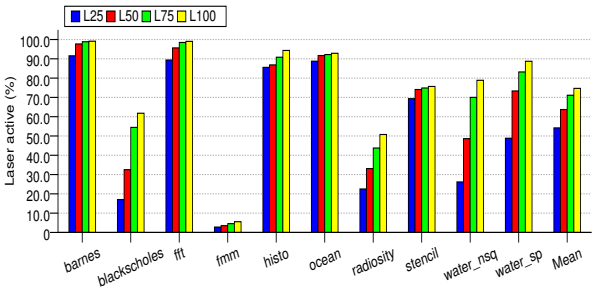

Figure 5. Percentage of the total time the laser was active (*ArchLink*)

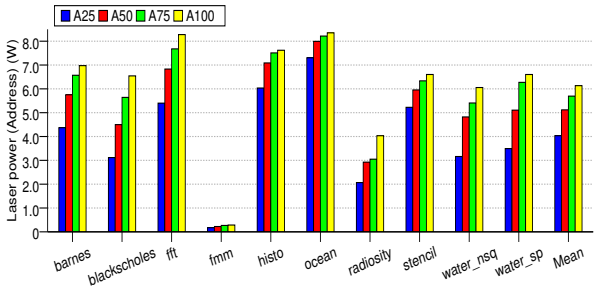

Figure 6. Laser Power (*ColdBus*)

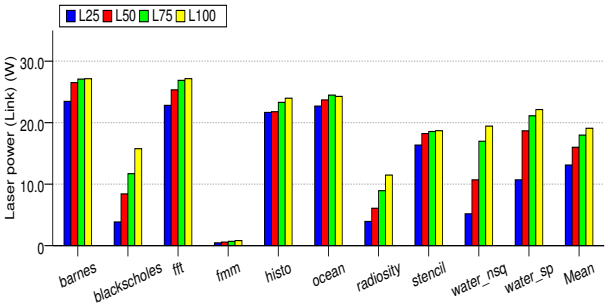

Figure 7. Laser Power (*ArchLink*)

and the *FULLOP* configuration that keeps the laser turned on all the time (similar to Corona [2]) in Figure 8. The power of the *FULLOP* configuration remains constant at 35W. The mean laser power of *ColdBus* is 4.1W, and the mean laser power of *ArchLink* is 16.8W. We thus have a 4.1X improvement in laser power consumption.

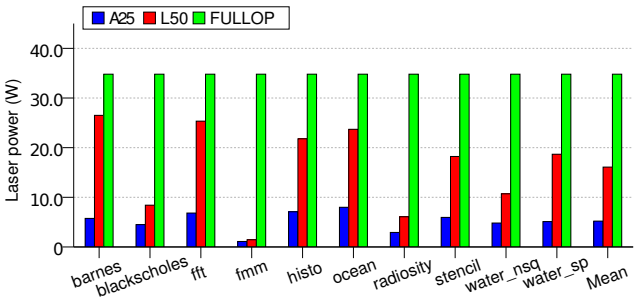

Figure 8. Laser Power

D. Extra Token Usage

Figure 9 shows the number of extra tokens used per 1000 instructions. As we can see, the number of extra tokens required is actually a very small number (most of the time less than 10-20). The only exceptions are fft and fmm running with 25-cycle epochs (15-20 extra tokens). This further means that the usage of the extra waveguide is on an average about 2%. The number of messages sent using extra tokens is also a negligible fraction of the total number of messages sent on the NoC ($\approx$ 1%).Figure 10 shows the percentage of requests that we were able to transmit in the same epoch. Messages can be delayed due to contention in the station, and unavailability of power. We observe that there is a gradual increase in the percentage of successfully transmitted messages (93% to 99%) with epoch size.

E. Performance and ED^2

Let us now compare the performance of the different configurations. Figure 11 shows the relative performance of the different *ColdBus* configurations (results are normalized to A25). Configurations with lower epoch sizes are bedeviled by a low epoch prediction rate, and configurations with higher epoch sizes get disproportionately penalized by an inaccurate prediction. For example, for 100-cycle epochs, the epoch prediction rate is high, and the number of extra tokens used is also less. Thus, we should ideally expect the best performance. However, this seldom turned out to be the case, because of the high penalty associated with having inadequate extra tokens. Our extra predictor reduced the number of extra tokens to very low numbers (most of the time to 0) for *ColdBus* with 100-cycle epochs. In this scenario, if a message needed an extra token, it sometimes needed to contend with many other messages, or waited till the end of the *long* epoch if no tokens were available. Due to the bursty nature of memory accesses, this problem was very

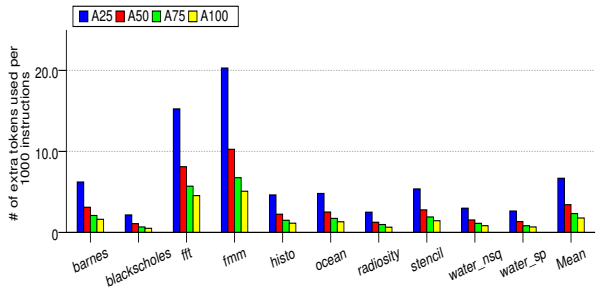

Figure 9. Number of extra tokens used per 1000 instructions (*ColdBus*)

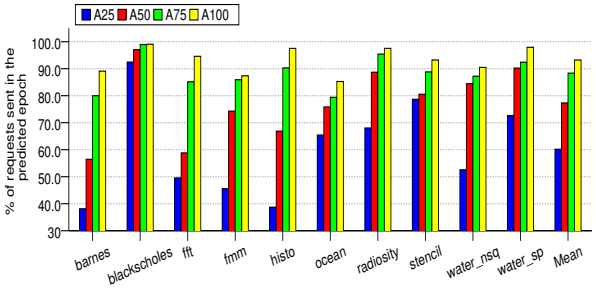

Figure 10. Percentage of requests issued in the same epoch

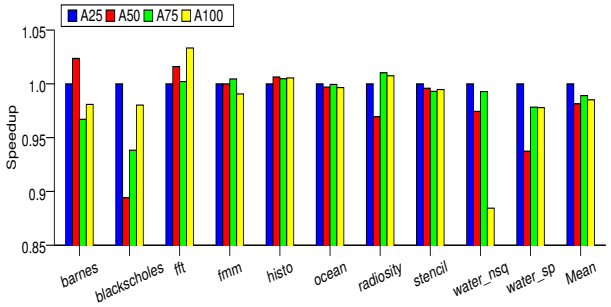

Figure 11. Speedups (*ColdBus*)

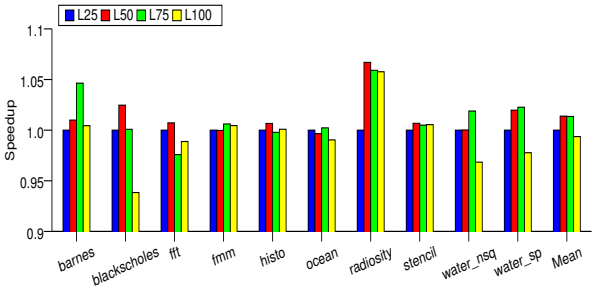

Figure 12. Speedups (*ArchLink*)

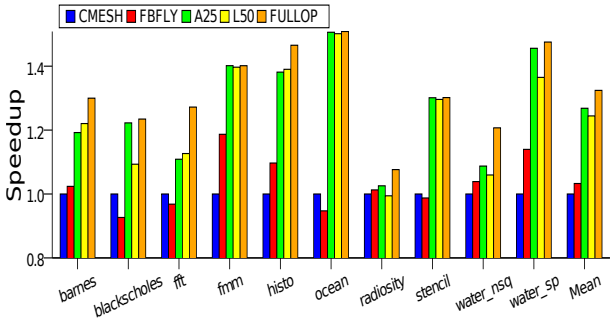

Figure 13. Speedups across configurations

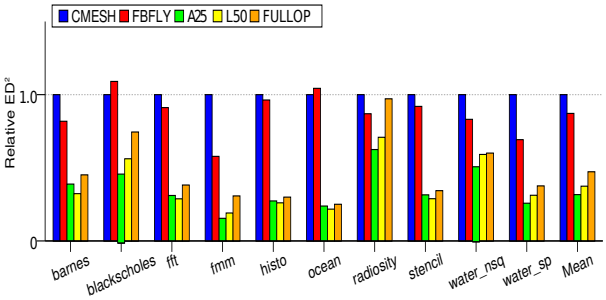

Figure 14. ED^2 (for the entire system)

common, and proved to be the single largest performance limiter for configurations with large epoch sizes. Figure 12 shows the relative performance numbers for *ArchLink*. Here also there is a degradation towards an epoch size of 100 cycles because the power for each epoch is fixed, and there is a performance penalty if we require extra power in the middle of an epoch (see Section V-A2). $A25$ and $L50$ are respectively the best configurations(1-2% better than the rest, also best in terms of ED^2).

In Figure 13 we compare $A25$, $L50$, with *FULLOP* and two electrical NoCS: a concentrated mesh(CMESH) and flattened butterfly (FBFLY) topology. The results are normalized to CMESH. On an average we find FBFLY to be 5% better than CMESH. $A25$ is 27.5% faster than CMESH and $L50$ is 23% faster than CMESH. The best configuration is *FULLOP*, which always has enough power for all the messages. Its mean performance is 32% more than CMESH. We can quickly conclude that with our extra predictor, we in fact gain on IPC as compared to *ArchLink*, and the gains are typically in the range of 1-8% (mean 3.1%). Finally, let us compare the configurations on the basis of the full system ED^2 metric in Figure 14 (normalized to CMESH). FBFLY is 20% better (0.8). The three optical configurations: $A25$, $L50$, are *FULLOP* are significantly better. Their geometric mean ED^2 values are: 0.32 ($A25$), 0.39 ($L50$), and 0.48 (*FULLOP*). Thus, we can conclude that in terms of ED^2 *ColdBus* is 18% better than *ArchLink* and is thus a more power efficient design.

VI. Conclusion

In this paper we demonstrated that predicting network traffic based on program counter addresses is a much better approach for reducing laser power as compared to traditional approaches that base their predictions on past network activity. Dealing with occasional mispredictions is non-trivial and we thus had to design an additional structure called the *extra* predictor and waveguide to take care of such events. In Section V, we showed that *ColdBus* is better than state of the art network activity based approaches in terms of power (4.1X less optical power), performance (3.1% increase), and ED^2 (18% decrease).

References

[1] N. Kirman, M. Kirman, R. K. Dokania, J. F. Martinez, A. B. Apsel, M. A. Watkins, and D. H. Albonesi, "Leveraging Optical Technology in Future Bus-based Chip Multiprocessors," in *MICRO*, 2006.

[2] 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.

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

[4] Y. Pan, J. Kim, and G. Memik, "Flexishare: Channel sharing for an energy-efficient nanophotonic crossbar," in *HPCA*, 2010, pp. 1–12.

[5] Z. Li, M. Mohamed, H. Zhou, L. Shang, A. Mickelson, D. Filipovic, M. Vachharajani, X. Chen, W. Park, and Y. Sun, "Global on-chip coordination at light speed," *Design and Test of Computers*, vol. 27, no. 4, pp. 54–67, July 2010.

[6] M. Mohamed, Z. Li, X. Chen, and A. Mickelson, "Hermes: A hierarchical broadcast-based silicon photonic interconnect for scalable many-core systems," *arXiv preprint arXiv:1401.4629*, 2014.

[7] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: a fast and accurate noc power and area model for early-stage design space exploration," in *DATE*, 2009.

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

[9] E. Peter and S. R. Sarangi, "Optimal power efficient photonic swmr buses," in *Silicon Photonics 2 Workshop (along with HiPEAC)*, 2015.

[10] M. FAUGERON, M. Chtioui, A. Enard, O. Parillaud, F. Lelarge, M. Achouche, J. Jacquet, A. Marceaux, and F. van Dijk, "High optical power, high gain and high dynamic range directly modulated optical link," *Lightwave Technology, Journal of*, vol. 31, no. 8, pp. 1227–1233, April 2013.

[11] E. Peter and S. R. Sarangi. Optikit: An open source kit for simulation of on-chip optical components. [Online]. Available: http://www.cse.iitd.ac.in/%7Esrsarangi/files/papers/optikit.pdf

[12] M. Faugeron, M. Tran, F. Lelarge, M. Chtioui, Y. Robert, E. Vinet, A. Enard, J. Jacquet, and F. Van Dijk, "High-power, low rin 1.55-directly modulated dfb lasers for analog signal transmission," *Photonics Technology Letters*, vol. 24, no. 2, pp. 116–118, 2012.

[13] M. Faugeron, M. Krakowski, Y. Robert, E. Vinet, P. Primiani, J. Le Goëc, O. Parillaud, F. van Dijk, M. Vilera, A. Consoli *et al.*, "Monolithic master oscillator power amplifier at 1.58 μm for lidar measurements," in *International Conference on Space Optics*, vol. 7, 2014, p. 10.

[14] B. Fu, Y. Han, H. Li, and X. Li, "Accelerating lightpath setup via broadcasting in binary-tree waveguide in optical nocs," in *DATE*, 2010.

[15] J. Levy, K. Saha, Y. Okawachi, A. Gaeta, and M. Lipson, "High-performance silicon-based multiple wavelength source," in *Lasers and Electro-Optics*, 2011.

[16] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: Illuminating Future Network-on-Chip with Nanophotonics," in *ISCA*, 2009.

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

[18] J. A. Lazaro, J. Gonzalez, J. A. Altabas, and A. Lerin, "Graphene silicon ring resonators for wavelength routers in photonic network-on-chip," in *ICTON*, 2015.

[19] W. Bogaerts, P. De Heyn, T. Van Vaerenbergh, K. De Vos, S. Kumar Selvaraja, T. Claes, P. Dumon, P. Bienstman, D. Van Thourhout, and R. Baets, "Silicon microring resonators," *Laser & Photonics Reviews*, vol. 6, no. 1, pp. 47–73, 2012.

[20] C. H. Joyner and J. Leuthold, "Multimode interference coupler with tunable power splitting ratios and method of tuning," May 27 2003, US Patent 6,571,038.

[21] J. Leuthold and C. H. Joyner, "Multimode interference couplers with tunable power splitting ratios," *Journal of lightwave technology*, vol. 19, no. 5, p. 700, 2001.

[22] R. Thapliya, T. Kikuchi, and S. Nakamura, "Tunable power splitter based on an electro-optic multimode interference device," *Applied optics*, vol. 46, no. 19, pp. 4155–4161, 2007.

[23] R. Morris, E. Jolley, and A. K. Kodi, "Extending the performance and energy-efficiency of shared memory multicores with nanophotonic technology," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 83–92, 2014.

[24] C. Chen and A. Joshi, "Runtime management of laser power in silicon-photonic multibus noc architecture," *IEEE Journal of Selected Topics in Quantum Electronics*, 2013.

[25] Z. Li and T. Li, "Espn: A case for energy-star photonic on-chip network," in *ISLPED*, 2013.

[26] 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.

[27] 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.

[28] "Electrooptically tunable microring resonators in lithium niobate," *Nature Photonics*, pp. 407 – 410, 2007.

[29] P. Petoumenos, G. Keramidas, and S. Kaxiras, "Instruction-based reuse-distance prediction for effective cache management," in *SAMOS*, 2009.

[30] M. Haurylau, G. Chen, H. Chen, J. Zhang, N. A. Nelson, D. H. Albonesi, E. G. Friedman, and P. M. Fauchet, "On-chip Optical Interconnect Roadmap: Challenges and Critical Directions," *Selected Topics in Quantum Electronics, IEEE Journal of*, 2006.

[31] I. O'Connor, "Optical Solutions for System-Level Interconnect," in *SLIP*, 2004.

[32] W. Huang, K. Rajamani, M. Stan, and K. Skadron, "Scaling with design constraints: Predicting the future of big chips," *Micro, IEEE*, vol. 31, no. 4, pp. 16 –29, 2011.

[33] N. Kirman and J. F. Martínez, "A power-efficient All-optical On-chip Interconnect Using Wavelength-based Oblivious Routing," in *ASPLOS*, 2010.

[34] S. Sarangi, R. Kalayappan, P. Kallurkar, S. Goel, and E. Peter, "Tejas: A java based versatile micro-architectural simulator," in *PATMOS*, 2015.

[35] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *MICRO*, 2009.

[36] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The splash-2 programs: characterization and methodological considerations," *SIGARCH Comput. Archit. News*, vol. 23, pp. 24–36, May 1995.

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

[38] J. A. Stratton, C. Rodrigues, I. J. Sung, N. Obeid, L. W. Chang, N. Anssari, G. D. Liu, and W. W. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," *Center for Reliable and High-Performance Computing*, 2012.

[39] R. W. Morris Jr. and A. K. Kodi, "Power-efficient and high-performance multi-level hybrid nanophotonic interconnect for multicores," in *NOCS*, 2010.