BigBus: A Scalable Optical Interconnect

Janibul Bashir, Eldhose Peter and Smruti R. Sarangi, Indian Institute of Technology Delhi

Abstract

This paper presents *BigBus*, a novel design of an on-chip photonic network for a 1024-node system. For such a large on-chip network, performance and power reduction are two mutually conflicting goals. This paper uses a combination of strategies to reduce static power consumption while simultaneously improving both performance as well as the energy-delay^2 product. The crux of the paper is to segment the entire system into smaller clusters of nodes, and adopt a hybrid strategy for each segment that includes conventional laser modulation, as well as a novel technique for sharing power across nodes dynamically. We represent energy internally as tokens, where one token will allow a node to send a message to any other node in its cluster. We allow optical stations to arbitrate for tokens at a global level, and then we predict the number of token equivalents of power that the off-chip laser needs to generate. Using these techniques *BigBus* outperforms other competing proposals. We demonstrate a speedup of 14-34% over state of the art proposals and a 20-61% reduction in ED^2.

1. INTRODUCTION

A thousand node (core + cache bank) multicore chip might not purely be in the domain of mere speculation in the future. Even with the impending death of Moore's law, we would like to opine that creating a 1000-node system, where a node can be a computing or a memory element, is a worthy goal to pursue.

The two major constraints in such networks are: area and power and. We believe that even with current technology, area and power are not very significant impediments. For example, the Xeon Phi 7290 has 72 4-way SMT cores (288 threads) fabricated in a 14nm process. The number of cores is expected to increase in future versions such as Knight's Mill and Knight's Hill. Recently UC Davis released a low power 1000-core research processor [Davis 2016], and many more efforts are there in this direction such as the EU funded PRIME project [EPSRC 2013]. All the efforts in this direction advocate the lean core approach [Kurian et al. 2010; Abellán et al. 2016], where each core is a single or dual issue in-order processor.

We try to size a similar system, with 768 cores, and 256 cache banks. If each core is similar to a dual-issue inorder ARM A7 core, then we can fit 768 such cores in a $130mm^2$ die (duly scaled to the 10 nm process using the widely used scaling rules proposed by Huang et al. [Stan et al. 2011]). In addition, we can fit 64 MB of SRAM memory ($170mm^2$) (computed using Cacti 5.3 [Thoziyoor et al. 2008]). The remaining area can be used for PLLs, power supplies, and the interconnect. After due scaling, the maximum power usage of 768 ARM A7 cores (at a 10 nm process) is within 120W. In

Author's addresses:Janibul Bashir, Eldhose Peter and Smruti R. Sarangi, Computer Science Department, Department of Computer Science and Engineering, IIT Delhi, Hauz Khas, New Delhi -110016. E-mails: {janibbashir,eldhose,srsarangi}@cse.iitd.ac.in

DOI: 0000001.0000001

comparison, Xeon Phi 7290 has a maximum TDP (thermal design power) of 245W. Our experiments indicate that the average power consumption figures of our simulated cores are much lower.

A mere agglomeration of low power cores, and cache banks does not yield a high performing system, unless it is supplemented with a high performance interconnect. We propose one such interconnect in this paper called *BigBus*, which is a high performance optical interconnect. It is based on a 3D integration technique where the different layers are stacked on top of each other. Our design is a two-layer 3D chip, where an optical layer (waveguides + optical stations) is stacked on top of a CMOS based logical layer (cores+cache banks). The different layers are connected together using a popular through-silicon via (TSV) 3D integration technique [Topol et al. 2006; Gu and Xu 2009]. It is one of the most promising techniques adopted in 3D chips as it results in shorter interconnection length resulting in high performing interconnect [Ye et al. 2009; Ye et al. 2013; Pasricha and Bahirat 2011].

Our reasons for choosing photonics based technology are because of its inherent advantages in terms of latency, bandwidth, and possibly power efficiency if designed well (duly justified in Section 5). We need such interconnects for such large systems of cores and caches. Furthermore, photonics technology has already come down to the board level (optical PCI-Express [Rath 2013]), and is expected to come inside the chip very soon.

Sadly, creating an optical network for a 1000-node system is not easy. We know of two similar efforts (ATAC [Kurian et al. 2010] and Electro-Photonic NoC for Kilocore Systems [Abellán et al. 2016]), which we shall show to be extremely profligate in terms of power consumption. Specifically, static power consumption has been shown [Demir and Hardavellas 2014; Zhou and Kodi 2013; Peter et al. 2015; Bashir and Sarangi 2017] to be one of the single largest bottlenecks in the creation of scalable optical networks. There are some standard techniques to reduce power consumption without a significant drop in performance such as wavelength sharing [Zulfiqar et al. 2013] and laser modulation [Peter et al. 2015]. Almost all power saving schemes in optical networks use some combination of these basic approaches. However, like a good recipe, which uses simple ingredients yet an elaborate process, here also the main task is in designing a non-trivial network that has favorable properties. Our results show that any trivial combination of these ideas does not lead to good solutions, and it is necessary to first perform a thorough study of target workloads, and then arrive at a design that is able to deliver high performance and extreme power savings.

The main contributions of this paper are summarized as follows:

— The intelligent placement of cores and cache banks, which reduces the number of hops for most of the messages to two. **Our placement scheme is counter-intuitive; however it turns out to be the best solution for a system with an optical NoC**. We challenge the traditional way of thinking, and show that our solution is superior.
— A novel partial-MWMR scheme where the data waveguides are partially shared between the optical stations. Previous proposals were either assigning a waveguide to a particular optical stations or allowing all the stations to share the waveguide.
— An hierarchical optical NoC that can extend to 1000 node system without facing much power loss.
— An effective NUCA protocol for such a large system.
— A novel predictive technique for laser modulation. It uses the current network statistics like pending events and waiting time.

We motivate our technique in Section 3 by analyzing the behavior of some Parsec benchmarks. We observe that unless some spatial locality is created, we will have too many messages being sent all over the chip. After creating spatial locality, we note the variance in the optical power requirement of different transmitters, and try to intelligently size sub-networks to take the variance in traffic into account. We build on these insights in Section 4.1, and show our results in Section 5, where we demonstrate that our approach is superior to competing approaches by 14-34% in terms of performance and 20-61% in terms of ED^2.

2. BACKGROUND AND RELATED WORK

2.1. Optical Communication Infrastructure

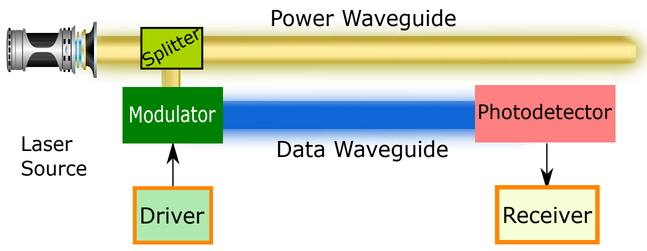

Fig. 1: Optical communication

The basic elements of an on-chip optical communication system are a laser source, driver, modulator, waveguide, detector, and receiver (see Figure 1). The laser source produces the optical signal. Each transmitter has a driver that controls a modulator, which modulates the optical signal to encode the data. The modulated optical signal passes through a waveguide (optical channel) to reach the destination. At the destination node, a photo-detector detects the optical transmission and performs an O/E conversion. There are different types of optical buses. In an MWSR [Vantrease et al. 2008] waveguide, multiple nodes can write to a single waveguide and a single destination reads from that waveguide. A token channel [Vantrease et al. 2008] or a token slot [Vantrease et al. 2009] approach can be used to implement the arbitration at the sender side. The SWMR [Pan et al. 2009] waveguide has a single sender and multiple receivers. The MWMR [Pan et al. 2010] waveguide is a combination of SWMR and MWSR, where multiple nodes can write to a single waveguide and multiple nodes can read from the same waveguide. Whenever there are multiple senders, we need an arbitration mechanism [Vantrease et al. 2009] to grant exclusive access, and whenever there are multiple receivers we need beam splitters [Peter and Sarangi 2014] to divert a portion of the optical power to the receivers. Also note that in an SWMR waveguide [Pan et al. 2009] receivers by default are mostly turned off. Before sending a message, the standard approach is to turn on the receiver by sending a single bit, and then transmit the message. This approach significantly reduces both static as well as dynamic power (known as reservation assisted SWMR, see [Pan et al. 2009]).
Lasers: We propose to use directly modulated lasers (DML) in this work. They are commercially available and it is possible to modulate these lasers in a single cycle (@2.0 GHz) [FAUGERON et al. 2013] without facing issues related to thermal instability [Peter et al. 2015]. In our work, we consider 4 arrays of DML lasers. Each array consists of 32 DML lasers (max. optical power per array: 5.75W [FAUGERON et al. 2013]) that can be separately modulated. Thus, we have 4 fast single cycle lasers with 32 power levels each (also used by Peter et al. [Peter et al. 2015]). We distribute power from lasers to individual optical stations (transceivers) using a special waveguide called a *power waveguide*. As observed by Morris et al. [Morris et al. 2014] using

multiple off-chip laser sources has some benefits. Optical power can enter the chip at multiple places. This reduces the power losses and the routing complexity of the power waveguide. Also, there are limits to the power a single waveguide can carry due to non-linear effects in light transmission. Having more power waveguides circumvents this limit.

Data Network: The traditional approach of using a single serpentine shaped waveguide [Vantrease et al. 2008; Chittamuru et al. 2017] passing through all the stations is not feasible because of a large number of optical stations. The power losses will be prohibitive. In comparison, a mesh based network [Zhou and Kodi 2013] with either electrical routers, or wavelength based routing [Kirman and Martínez 2010] will also become very complex because of the difficulty in quickly setting up and tearing down hundreds of optical paths. Hence, we look at approaches that partition the communication structures into segments similar to Moustafa et al. [Mohamed et al. 2014].

2.1.1. Efficient Power Waveguides. Designing the power waveguides is non-trivial because they typically use cascading beam splitters to distribute power to stations, and each such splitter is associated with a power loss. Cascading them increases the power consumption exponentially [Peter and Sarangi 2015]. We use the optimal (in terms of power loss) chain and tree based designs proposed by Peter et al. [Peter and Sarangi 2015] in our work. Peter et al. propose a single cycle solution to dynamically compute optimal split ratios using a table lookup based approach. Second, based on predicted activity and laser power, we need to dynamically change the split ratios of beam splitters. We use the single cycle splitter based on ring resonators proposed by Peter et al. [Peter et al. 2016]. The original paper had limited split ratios – from 1:1 till 1:2.5. We extend the range to 1:15 by cascading 2-3 such splitters in this work.

2.2. Schemes to Reduce Static Power Consumption

2.2.1. Power Sharing. The basic idea in channel sharing is that multiple nodes can either use the same waveguide to transmit messages or share power between themselves. SUOR [Wu et al. 2014] proposes to use a waveguide simultaneously to transmit multiple messages between sender-receiver pairs when the paths (from sender to receiver) are disjoint. ColdBus [Peter et al. 2015] proposes an *extra* power waveguide, which can supply some power to stations on demand (arbitration via tokens). In contrast, the wavelength stealing scheme [Zulfiqar et al. 2013] steals the wavelengths allotted to other nodes in the system. Collisions are detected using erasure coding. Xu et al. [Xu et al. 2012] propose a method in which two nodes share a waveguide. They save power by reducing the number of ring resonators and receivers. Likewise, Chittamuru et al. [Chittamuru et al. 2017] proposed an MWMR topology in which the nodes dynamically transfer the unused bandwidth to the other nodes in the system. Moreover, the authors propose to monitor the traffic injected into the network and then distribute the bandwidth accordingly. None of these proposals were found to scale to such large systems.

2.2.2. Laser Modulation. Some proposals propose to turn off the laser when not in use [Demir and Hardavellas 2014]. If we turn off the laser, we have to decide when to turn it back on. This cannot be done on demand because it takes time to send a message to the off-chip laser. Hence, we need to predict the network activity before hand. In order to predict and turn the laser on, the standard approach is to split the entire duration of an execution into fixed duration *epochs*. The predictor predicts in which epoch the laser should be turned on. Probe [Zhou and Kodi 2013] proposes to use a predictor based on link and buffer utilization statistics and ColdBus proposes to use a predictor based on the PCs of memory instructions. Similarly, Chen et al. [Chen and Joshi 2013] propose to turn off some portions of the network depending upon the bandwidth requirements. In this proposal, we use a novel predictor based on the history of network traffic and the average waiting time of messages at a station.

3. MOTIVATION

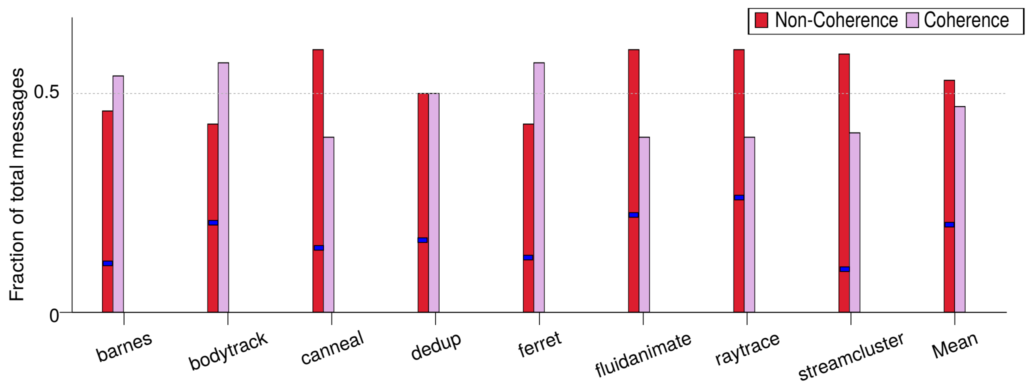

Fig. 2: Categorization of messages in a network

Here we look at some of the salient features of our workloads, and try to derive some insights regarding the design of a 1000+ node optical network.

3.1. Separate Networks

We ran simulations for a system with simple 2-issue in-order cores as described in Section 5.2 by assuming a hypothetical point-to-point network topology. We ran a subset of benchmarks (that scale) from the Parsec benchmark suite [Bienia et al. 2008]. The results of the simulation (see Figure 2) show that on an average more than 50% of the traffic injected into the network is due to non-coherence messages: the communication between L1 and L2 caches, between L2 cache banks, and traffic to/from the memory controllers. The remaining traffic is accounted for by the coherence messages between the private L1 caches. Based on this, let us derive **Insight: Separate-Network**, which is that let us have a separate network for coherence messages, and a separate network for non-coherence messages. Most of the non-coherence messages are messages meant to find out if a certain block is present in a cache bank. These messages should be routed on a separate sub-network as per our observations in Figure 2. Let us thus propose a separate sub-network for exclusively connecting all the L2 cache banks.

3.2. Creating Additional Spatial Locality

Let us once again consider non-coherence messages, where a majority of messages are sent from private L1 caches to L2 cache banks. In a naive scheme, the L2 cache line can be present in any cache bank (anywhere on the chip). Let us divide the network into multiple sub-networks, where each sub-network connects $1/K^{th}$ the number of cores and cache banks. We have empirically determined that $K = 4$ is the best choice. Let us refer to each sub-network as a *quadrant*. It would be best if we find the L2 cache line within the quadrant of the requesting L1 cache. This can explicitly be ensured by using non-uniform cache access (NUCA) techniques. Let us create sets of 4 cache banks called *bank sets*, where each bank set has exactly 1 cache bank in each quadrant. A cache line is allowed to migrate to any other cache bank within its corresponding bank set. In the worst case we need to search for a line in all the 4 cache banks in a bank set.

Now, we can use this mechanism to create spatial locality, by migrating a requested cache line to the quadrant of the requesting core. This can be trivially achieved by swapping two entries between two cache banks belonging to the same bank set. On a subsequent access, the line will be found in the same quadrant. This will reduce traffic, and contention. Moreover, the message will also travel through a shorter waveguide (lesser losses). Figure 2 shows the potential benefit of such a scheme, where the blue markers indicate the fraction of messages that will travel within only one quadrant. We observe that 36% of the messages can be restricted to only one quadrant for non-coherence messages. Thus we have, **Insight: Use-NUCA** use NUCA policies to reduce contention and increase locality.

3.3. Traffic Imbalance

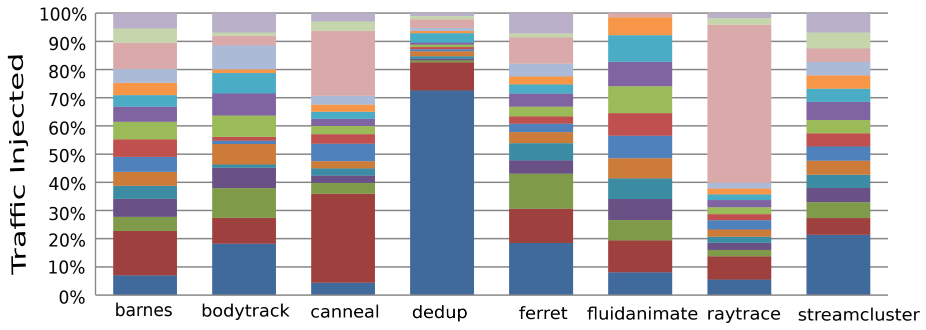

Fig. 3: Traffic injected across 16 stations

In our workloads nodes inject a varying amount of traffic into the network. We plotted the relative traffic injected for a representative group of 16 adjacent stations in Figure 3. Here each color represents the percentage of traffic injected by the station. It is clear that some stations in a network will require much more bandwidth than the others because the ratio of traffic injected between the top 2 most active stations can be as high as 10:1. Thus, it is necessary to have a dynamic load balancing scheme by sharing data and power networks between adjacent nodes. **Insight: Share-Power** share the data and power networks between adjacent nodes if possible.

4. ARCHITECTURE

4.1. Structure

In this paper we propose a system with 768 cores and 256 cache banks (see Figure 4(a)). Each core in *BigBus* is a dual-issue in-order RISC processor similar to the cores used by competing work (ATAC [Kurian et al. 2010]). Arranging the cores, cache banks, and optical links is a non-trivial problem primarily because of the scale of the system. Related work that considers a smaller number of cores and cache banks did not give due consideration to this problem because with fewer stations, locations of stations are not all that important. We can either use a simple full SWMR based crossbar or a mesh based network with simple path setup algorithms. However, with 1024 communicating nodes, the arrangement of stations, and their layout become important problems. We cannot afford long waveguides because of their high optical losses. Let us use some of the insights derived from Section 3 to create our network.

We place some cache banks close to some of the cores to decrease the access latency. These cores and cache banks can be connected together by a common waveguide in

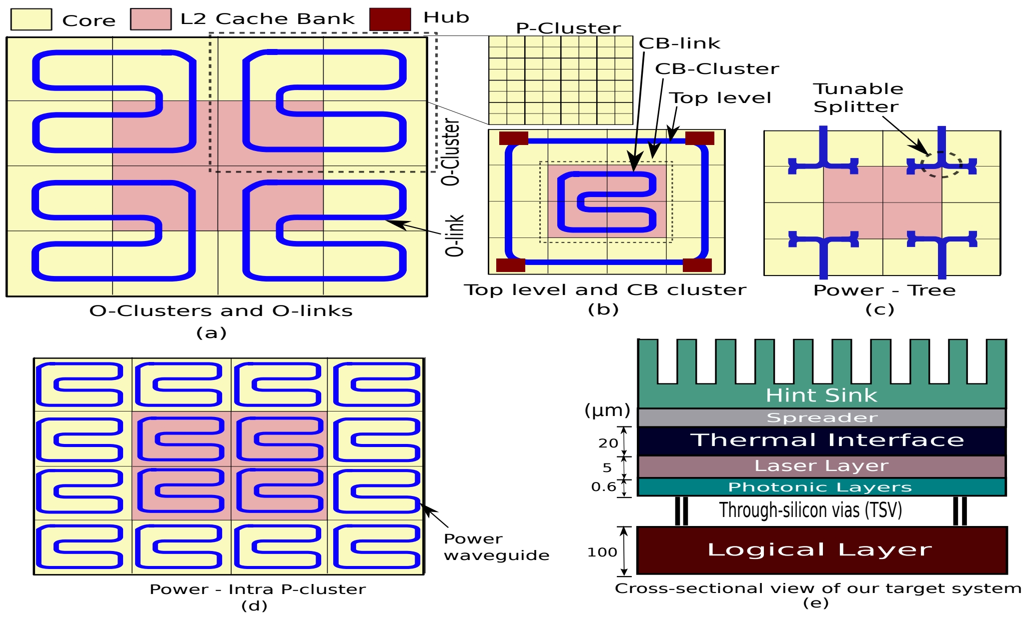

Fig. 4: Architecture

order to limit the length of data waveguides. We have adopted this insight in *BigBus*, where we have placed the cache banks at the center and the cores at the periphery of the chip. We recognize square blocks of 4 cores or 4 cache banks, and refer to them as a *cluster*. Intra-cluster communication is electrical, and inter-cluster communication is optical. This cluster size was found to be optimal in terms of both performance and power (determined empirically). Each cluster has an optical station. We then proceed to build larger clusters: 16 clusters (arranged as a 4×4 square) form a *P-Cluster* (see Figure 4(a)), and each *O-Cluster* contains 4 *P-Clusters*. A *P-Cluster*, thus, contains 64 cores/cache banks, and an *O-Cluster* contains 192 cores and 64 cache banks (3 *P-Clusters* of cores and one *P-Cluster* of cache banks). We thus have 4 *O-Clusters* in our system. Note that our design is not very specific to the idiosyncrasies of the target system. For a different system, we can **apply the same principles to create a similar network**.

4.1.1. Data Network. The cache banks and cores in an *O-Cluster* are connected together by a serpentine structured optical link (called *O-Link*) [Vantrease et al. 2008; Pan et al. 2009] (see Figure 4(a)). The serpentine structure is simple, well studied, and does not require explicit path-setup or routing. All these approaches save power. Additionally, we define a separate optical link to connect all the cache banks at the center of the chip called *CB-Link* (**Insight Separate-Network** in Section 3). The set of all cache banks is called the *CB-Cluster* (see Figure 4(b)). Lastly, we define an optical link called the *top level link* that connects all the O-Links together. It is attached to each O-Link via a hub (containing a message queue), which is a dedicated structure that transfers data between O-Links.

4.1.2. Power Delivery Network. We have four off-chip 1550 nm laser sources connected to the chip at 4 separate points – one for each *O-Cluster* (see Figure 4(c)). Each input uses tapered waveguides to minimize the insertion loss [Humphrey 1994]. Subsequently, we use a tree based network to distribute power to the 4 constituent *P-Clusters*. Each

P-Cluster has several serpentine shaped (logical ring) power waveguides (running in parallel) that distribute power to all the constituent stations. Each station then uses tunable ring resonators to source the power and use it for data transmission (see Figure 4(d)).

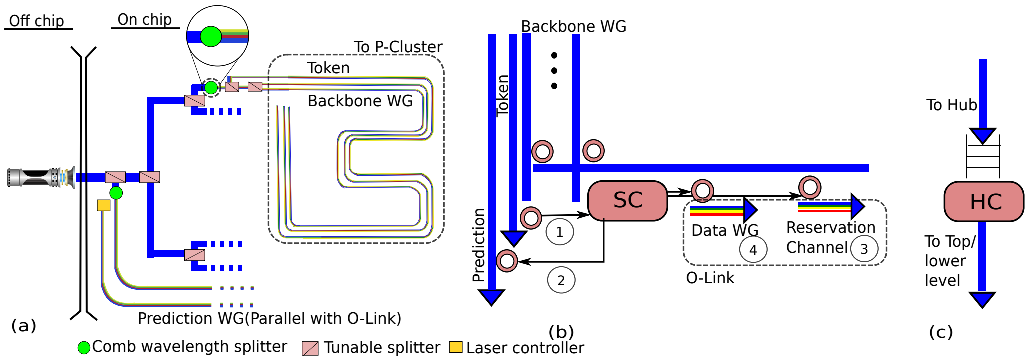

Fig. 5: Optical network

At each *P-Cluster*, we use a comb splitter [Levy et al. 2011] (also see [Peter et al. 2015; Thakkar et al. 2016]), which is used to split a monochromatic optical signal (at 1550 nm) into 64 equally spaced wavelengths (between 1450-1650 nm). The additional area overhead is limited to 0.17% [Levy et al. 2011]. We use DWDM (dense wavelength division multiplexing) technology to transmit all these separate wavelengths on the same waveguide. Subsequently, we use a set of 16 cascading tunable optical splitters (see [Peter et al. 2016]) to split the DWDM signal into 17 parts. Now, each part of the optical signal is assigned to one waveguide: 16 backbone waveguides, and 1 token waveguide. All of these waveguides run parallel to each other and have a serpentine layout as shown in Figure 5(a). A station can source power from the backbone waveguides only if it has a token. Note that each *P-Cluster* need not be assigned the same amount of power (depends on the predicted activity of each station).

Figure 4(e) shows the cross-sectional 3D view of our final design. It is clear from the figure that our architecture is a multi-level design in which different layers are stacked on top of each other using a popular through-silicon via (TSV) 3-D integration approach [Topol et al. 2006; Gu and Xu 2009]. It is the most popular design choice and has been adopted in various 3D network topologies in order to increase the network performance [Ye et al. 2009; Ye et al. 2013].

4.2. Channel Sharing and Data Transmission

We assume that power is shared between the nodes in a *P-Cluster* and not across nodes in different *P-Cluster*s (**Insight Share-Power** in Section 3). For each *P-Cluster* we circulate a certain number of tokens (1 token = power of 1 unicast) on the token waveguide corresponding to that *P-Cluster*. We use a maximum of 16 tokens on the token waveguide. A station can grab tokens from only the token waveguide corresponding to its *P-Cluster*.

Now, to transmit a message, a station needs to grab a token designated for its *P-Cluster* from the token waveguide. After a station obtains a token from the token waveguide, it can access one of the backbone waveguides, draw power from it, and use it to transmit a message along a data waveguide. The details are as follows.

In the token waveguide, we pass different numbers of logical tokens for different *P-Clusters*. Let us assume that for a given *P-Cluster*, we pass N logical tokens. Here, each token corresponds to 1 wavelength (Assumption: wavelengths are numbered from 1 to N). The token waveguide is similar to the data waveguide and it makes a single pass through all the optical stations. If a station intends to send a message, it first tries to grab a token. It selects a number at random between 1 and N (let's say i), and checks if the i^{th} token is available with the help of a ring resonator. If it is able to detect a signal (meaning: token is available), it tries to absorb the optical power in the i^{th} power waveguide. If the i^{th} wavelength is not available, it tries to check for the ((i + 1)$\%N)^{th}$ wavelength and so on.

After N unsuccessful tries, we give up and wait for the next epoch. This approach guarantees mutual exclusion, yet is not necessarily fair. Nevertheless, we found this solution to be fast, and approximately fair. Hence, we use this scheme and did not choose more sophisticated yet slower and elaborate optical arbitration schemes [Vantrease et al. 2009] that guarantee fairness.

Once a station gets the i^{th} token, it is free to source power from the i^{th} backbone waveguide. Figure 5(b) shows this process. We use double pumped signaling (modulation at both edges of the clock) for transmission (similar to Corona [Vantrease et al. 2008]), which means that we can transmit 2 messages per clock cycle. For the first half cycle, we transmit a reservation packet that turns the receiver on (1 bit for each receiver). After that we transmit one 64-bit flit per half-cycle. After the transmission, we stop sourcing power from the backbone waveguide and return the token (stop diverting the corresponding wavelength).

4.2.1. Data Transmission. To pass a message from one node to another, we use the data waveguides. Data waveguides in our system consist of 4 O-Links, a CB-Link and a top level link. A message can be internal to an *O-Cluster* or the *CB-Cluster* or might span across these clusters. In the first two cases, the message passes through the corresponding intra-cluster waveguide to reach the destination. In the third case, the message requires a minimum of 3 hops. First it travels to the hub, then it moves to the hub of the destination *O-Cluster* via the top level link, and then it travels within the destination's *O-Cluster*. We have a dedicated queue at each hub that buffers packets.

Let us first consider a naive scheme called *BigBus No Share (BigBus_NS)*. Here we use SWMR [Vantrease et al. 2008] waveguides, where every node has its own dedicated data waveguide to send data to any station within its *O-Cluster*. This means there are 64 parallel waveguides in the bundle of O-Link and CB-Link waveguides (corresponding to the 64 optical stations in an *O-Cluster*). We first grab token i to source power from the backbone power waveguide i, and then transmit a message using a station's dedicated data waveguide. Let us now consider a more sophisticated scheme.

4.2.2. Partially Shared Scheme. Here, we have 64 parallel waveguides divided into four groups ($G_1 \ldots G_4$) of 16 (one group per *P-Cluster*). Stations in *P-Cluster* i can transmit messages on any waveguide in group G_i. However, they cannot transmit messages on waveguides in other groups. Each such waveguide is a p-MWMR (partial MWMR) waveguide. This means that only stations in its corresponding group can transmit messages (16 such stations) but all the stations (64) in its *O-Cluster* can receive messages. To access the shared data waveguide we use the result of the arbitration for the power waveguide and no extra arbitration is required. If we get access to the i^{th} backbone waveguide in the bundle of power waveguides, we infer a permission to access the i^{th} data waveguide as well. As per **Insight Share-Power**, we try to leverage unused power as follows. If a station has multiple messages waiting in its queue, then it tries to grab multiple tokens (1 token corresponding to each waveguide). The advantage of this scheme is that we can send more messages in parallel, and improve performance.

Also if a token is available it means that some laser is generating power. Instead of wasting it, it is better if stations within a *P-Cluster* can use it. We call this configuration as *BigBus_PS*.

Finally, we consider a *fully shared* scheme in which any node in an O-Cluster can arbitrate for any token. When we go for a fully shared data bus, the complexity of token arbitration becomes high. The power waveguide and the token waveguide need to pass through 64 stations (parallel to the data waveguide). The number of stations waiting for the token increases. At the same time, the number of transmitters and receivers required at a station increases from 16 to 64, which makes the fully shared scheme practically infeasible.

Note that in all these schemes, multiple receivers are capable of receiving data from a same waveguide. However, we are using the reservation based approach proposed by Pan et al. [Pan et al. 2009] in which receivers, by default, are mostly turned off. Before sending a message, the standard approach is to turn on the receivers by sending a reservation flit, and then transmit the message.

4.3. Activity Prediction

The number of tokens that needs to be transmitted on the token waveguide needs to be accurately determined for each *P-Cluster*. We do not want to send a lot of tokens (power wastage), neither too few tokens (too much of contention). As a result, we divide time into fixed size durations called *epochs* and devise a prediction mechanism to predict the number of tokens that we require in the subsequent epoch. We then modulate the off-chip laser to produce just enough power in the next epoch such that we can transmit power equivalent to N tokens, where N is the predicted number of tokens.

The novel prediction scheme is designed to consider the current trends in traffic along with historical values. Activity prediction is done in two stages. In the first stage, each station decides whether to increase or decrease tokens based on a function that has two inputs: wait time ($\mathcal{T}$) and the number of pending events ($\mathcal{N}$) at that station. The output of this function, $\mathcal{F}$, is 0, 1, 2, or 3. Here we give priority to the number of pending events over the wait time. The rules are shown below in decreasing order of precedence. $T_p = 8$, and $T_w = epoch\_size/2$. We have given priority to the number of pending events to improve the performance as we noticed that in the case of a sudden increase in traffic, the number of pending events can be higher.

$$\mathcal{F}(\mathcal{T},\mathcal{N}) = \begin{cases} 3 & \mathcal{N} \geq T_p \\ 2 & \mathcal{T} \geq T_w \| T_p/2 \geq \mathcal{N} < T_p \\ 1 & T_w/2 \leq \mathcal{T} < T_w \| \mathcal{N} < T_p/2 \\ 0 & \mathcal{T} < T_w/2 \end{cases}$$

We assume a dedicated prediction waveguide that runs through all the stations of an *O-Cluster*. It is powered by the off-chip laser. We divert 1 unit of power (corresponding to 1 unicast) to it 9 cycles before the end of the epoch. The comb splitter [Levy et al. 2011] associated with it produces 64 different wavelengths (1 for each station). We use an active-low based signaling strategy, and assign a wavelength to each station in the *O-Cluster*. In two half-cycles, all the stations can encode (using ring resonators) two bits and send their information back to a dedicated structure called the (laser controller) $LCntrlr$, which collates all the predictions.

The $LCntrlr$'s first task is to add all the 2-bit values. To add 64 2-bit numbers, we first add sets of 4 2-bit numbers (16 such sets) in parallel using a lookup table (42 ps, and $0.2nJ$ energy consumption, computed using Cacti 5.3 [Thoziyoor et al. 2008] and scaled using the results in [Huang et al. 2011]). We now have 16 partial sums, where each sum is at the most 12. Now, we use another lookup table to again add two partial sums at a time to generate 8 partial sums (another 42 ps and $0.2nJ$ energy). We now have 8 partial sums each being 5 bits. We next use a 3-level tree of adders to add them. The final output is 8 bits. Using numbers from [Herr et al. 2013], we observe that we can finish all the additions in 450 ps, consuming 1.5μW of power . The total energy consumed is less than 2μW and the total time is 534 ps, which is well within 1 clock cycle (@1.5 GHz). Let the computed sum be $\mathcal{S}$. We consider the top three MSB bits of $\mathcal{S}$ and map the values represented by the three bits ([0..7]) to the set [-3..4]. We consider the top three MSB bits to make the hardware simpler. Let the mapped value be V.

Let us now take historical information into account. Let us maintain a 10-bit shift register, where the i^{th} bit contains the MSB of the number of tokens supplied in the i^{th} previous epoch. We use the 10 bit value in this shift register to access a history table with 1024 entries (all entries initialized to 0). Each entry contains a predicted value (P) for the number of tokens. For the current epoch, let the predicted value be P_c. We set it to $P_c + V$, in the history table, and use the value ($P_c + V$) as the number of predicted tokens in the next epoch. This shift register can be accessed in a single cycle (31 ps) and consumes 120pJ of energy (calculated using Cacti 5.3).

4.4. Network Reconfiguration

Actions			**Cycles**
Prediction Phase			
Calculate the function $\mathcal{F}$ at stations			1
Send prediction to the laser controller*			1
Collate recommendations and make prediction			2
Calculate the split ratios and power			2
Parallel Activity 1		**Parallel Activity 2**	
Send to the off chip laser (E/0+tran+O/E)	2		
		Send split ratios to the splitters	1
Reconfiguration Phase*			
Compute laser array config.	1	Table access in *P-Cluster*	1
Laser retuning	1	Reconfigure splitters	1
Total			10
* Network inactive			

Table I: Prediction and reconfiguration

We use the technique proposed by Peter et al. [Peter and Sarangi 2015] for the power distribution tree in an *O-Cluster*, which has four leaf nodes (1 in each *P-Cluster*). We quantize the laser power using 5 bits (32 levels). Thus, at each leaf node we have 1024 combinations of power for both the branches. As suggested by Peter et al. we keep a 10-bit table in hardware, where for each combination of power in the branches, we store the input power, and the split ratios of the splitters. In less than 2 cycles [Peter and Sarangi 2015], we can access this table and compute the power requirement for the entire *O-Cluster* and the split ratios (6-bits) of all the splitters.

We have 16 splitters at each *P-Cluster*. We need to compute their split ratios. The input to each *P-Cluster* in the reconfiguration stage is the number of backbone waveguides that need to be activated (carry power). This is a number, κ, between 0 and 16. We separately save the split ratios for each splitter when $\kappa = 16$. For the rest, we envision a 16 entry table, where each row saves the split ratios of the 16 splitters (each 6 bits). The size of this table is: $16 \times 16 \times 6/8$ bytes = 192 bytes. This can be replicated for each *P-Cluster*. Table I shows the sequence of actions for predicting and reconfiguring the network. We require a total of 10 cycles for prediction and reconfiguration. Out of these in 3 cycles, the network cannot be used for data transmission.

4.5. NUCA Protocol

As per **Insight Use-NUCA** (Section 3), we divide the address space of L2 blocks into 64 sets by considering the 6 least significant bits (LSB) of the L2 block address. We assume that each of the 64 cache banks in an *O-Cluster* is mapped to a distinct bank set. Also note that in each bank set we have 4 banks: one in each *O-Cluster*. Now, when there is an L1 miss, we first extract the 6 LSB bits from the L2 block address, and access the corresponding bank in the same *O-Cluster*. This bank is called the *home bank* for the *O-Cluster*. If there is a miss in the home bank, we access the rest of the three banks in its set. We use the CB-Link for sending these messages. If there is a miss in the rest of the three banks, we send the request to main memory. Otherwise, if there is a hit in any of the banks, we migrate the block to the home bank.

5. EVALUATION

Parameter	**Value**	**Parameter**	**Value**
Cores	768	Technology	10 nm
Frequency	1.0 GHz		
Processor Core			
Pipeline	Dual-issue In-order	IW size	54
iTLB	128 entry	dTLB	128 entry
Private L1 i-cache, d-cache			
Write-mode	Write-back	Block size	64 bytes
Associativity	4	Size	32 kB
Latency	2 cycles	MSHRs	32
Directory	fully mapped, 16-banked, distributed MOESI, 16384 entries, 8-way		
Shared L2			
Write-mode	Write-back	Block size	64 bytes
Associativity	4	# banks	256
Latency (per bank)	8 cycles	Bank size	256 KB
Main Memory			
Latency	150 cycles	Mem. controllers	32
Queue Sizes			
Station Queue	16	Hub Queue	200
Electrical NoC			
Topology	2-D Mesh	Routing Alg.	X-Y
Flit size	256-bit	Link Traversal	1 cycle
Routing delay (w/wo bypassing)	2/3 cycles	# Virt. channels	4
		Buffers/port	8
Auxiliary structures (size in number of entries)			
RCB	128	VB	20
MQ	16		

Table II: Simulation parameters (also see [Kurian et al. 2010])

5.1. Experimental Setup

To evaluate the performance of the proposed system we ran simulations using the cycle architectural simulator Tejas [Sarangi et al. 2015], which has extensive support for modeling optical interconnects. Furthermore, it uses the Orion 2 [Kahng et al. 2011] and McPat tools [Li et al. 2009] for computing the power consumption. All the simulation tools – Orion, McPat, and Tejas – have been thoroughly validated against native hardware. Unlike other works that either do not perform thermal simulation

Optical Parameters	
Wavelength (λ)	$1.55\mu m$
Waveguide Width (W_g)	$0.5\mu m$
Waveguide Thickness	$0.2\mu m$
Refractive Index of $SiO_2(n_r)$	1.46
Refractive Index of Si (n_c)	3.45
Input Driver Power	76 μW
Output Driver Power	166 μW
Insertion Coupling Loss	50%
Output Coupling Loss	13%
Photodetector quantum efficiency	0.8 A/W
Photodetector minimum power	36 μW
Combined transmitter and receiver delay	180-270 ps
Optical propagation delay	7 ps/mm
Electrical propagation delay	35 ps/mm
Bending Loss	1 dB
Waveguide Loss	0.5 dB/cm
Coupler Loss	1 dB
Photodetector	0.1 dB
Wall Plug Efficiency	20 %
Splitter Loss	0.36 dB
Micro-heater power	1 μW/°C

Table III: Optical Parameters [Pan et al. 2010; Reed 2008; O'Connor 2004; Peter et al. 2017]

or use mean values for microring trimming power, we perform detailed analyses using a thermal simulator, HotSpot [Zhang et al. 2015], and then derive an accurate estimate of the trimming power. We use benchmarks from the Parsec and Splash2 benchmark suites [Bienia et al. 2008; Woo et al. 1995] for simulations. Out of the 12 benchmarks from the Parsec suite, *blackscholes* and *facesim* could at the most run on 128 cores. Hence, we do not consider them. *vips* and *x264* have fairly long sequential sections, and thus we did not find them suitable. *ferret* and *fluidanimate* can only run on systems where the number of cores is a power of 2. Hence, we used 512 cores for these benchmarks. For the rest of benchmarks we could run them on 768 cores. Out of the 12 benchmarks in the Splash2 benchmark suite, *fmm* and *barnes* run on 768 cores while *lu-contiguous*, *ocean-conti* and *fft* run on 512 cores. We do not use the rest of the Splash2 benchmarks as they run on a fewer number of cores. These are standard workloads used for evaluating such large systems [Kurian et al. 2010]. **Finally, note that all the performance numbers in this paper use the simulated execution time of the entire benchmark as the basis.**

We compare *BigBus* with three other state of the art photonics based multicore architectures namely Probe [Zhou and Kodi 2013], *ColdBus* [Peter et al. 2015] and ATAC [Kurian et al. 2010]. For a fair comparison, we have made some modifications to these architectures. Probe is a 64 core architecture that mainly focuses on the prediction mechanism, power distribution using multiple lasers and laser modulation. Here each station predicts whether it will be active or not in the next epoch, and the laser allocates power based on this prediction. We use the segmented power distribution (*P-Cluster based*) scheme for Probe, because a single power waveguide will not scale for a thousand node system. While simulating Probe, we use its activity prediction and laser modulation schemes. We call this configuration *mProbe*. ATAC is a 1024-core nanophotonics based system, which uses an unmodulated laser, and a three layer hierarchical network. To simulate ATAC, we use their power network, which does not use any laser modulation techniques. Here also we use our segmented power distribution scheme because of scalability issues. We call this configuration *mATAC*. We do not show the results for [Abellán et al. 2016] as it uses a similar laser modulation scheme as *mATAC*, and thus has similar power consumption numbers. Like Probe, *ColdBus* also assigns optical power on a per station basis with an additional *extra* waveguide for providing contingency power. We simulate *ColdBus* [Peter et al. 2015] with the segmented power delivery network.

Note that the main difficulty while comparing with other such systems is that we cannot just replace one network by another. The network is co-designed with LLC cache access protocols, cache coherence protocols, and the optical components. As a result some modifications were required to ensure a level playing field. Note that in all the architectures we are using 768 cores and 256 cache banks.

The hardware costs of all the configurations are almost the same with some minor differences in terms of number of ring resonators used in each scheme. Table IV summarizes the optical component requirements of the *BigBus_PS* scheme. It should be noted that the *BigBus_PS* requires the highest number of ring resonators as it draws optical power from multiple waveguides, whereas *mATAC* uses the least number of ring resonators, as it does not use any laser modulation technique. The number of ring resonators required in *BigBus_PS* scheme is nearly 13% more than those used in other schemes.

Optical system	# of waveguides	# of ring resonators
Power Delivery	16×16	≈260K
Data Network	64×5	≈1600K
Arbitration	16	≈4K
Prediction	16	≈1K
Total	608	≈1800K

Table IV: Optical Components

Section 5.2 explains the target system, and Table III shows the parameters of the optical network. The power analysis is done in Section 5.3. We compare our optical network with a state of the art electrical network (electrical noc parameters are given in Table II) in Section 5.4, and find the electrical network to be 53% slower. In Section 5.11 we perform detailed thermal simulations of our design, and calculate the micro-ring trimming power that is required to make all the rings operate at a pre-specified maximum temperature. The trimming power was found to be in the vicinity of 5-6W, which is very reasonable.

5.2. Target System

Table II shows the architectural parameters for the target system. The system is a 768 core chip fabricated at 10nm technology. It is based on a tile based architecture containing 768 cores and 256 cache banks with 4 nodes on each tile (node↔core/cache bank). The granularity of 4 was found to be optimal in terms of both performance and power (determined empirically). The chip operates at 1GHz frequency and has an area of $400mm^2$. Each core is a dual issue in-order core with a 32KB I/D private L1 cache. The chip has a shared L2 cache in the form of 256 cache banks with a capacity of 256KB each. The overall size of the L2 cache is 64MB, which is reasonable at 10nm technology given that a 100 core chip, TILE-Mx, already has an on-chip cache of size 40MB. Based on experimental results, we did not find the need to have an L3 cache with private L2 caches.

In our topology, each cluster has an optical station which is connected to both the power and data waveguides. It filters out the power from power waveguides, modulates the light and sends the information through the data waveguides. All these objectives are achieved using a set of ring resonators. In addition, we use a double pumping strategy in which the data is transmitted at both edges of the clock (similar to Corona [Vantrease et al. 2008]). Thus, on a single waveguide we can transmit 128 bits of data per clock. In our design, we assume a cache line of size 64 bytes and there is effectively 1 waveguide per station. Thus, we can transfer the entire cache line in 5 flits (first flit being the reservation flit). Thus, the total bandwidth of the system is 16 GBPS. For an electrical network, we assume a mesh topology with a router traversal time of 2/3 cycles [Balfour and Dally 2014]. Routing related decisions are taken in the first cycle and the switch traversal occurs in the second cycle (See Table II for the electrical NoC parameters).

5.3. Power Model

In optical interconnects, the total power required to send a message is the sum of power consumed by electrical components such as drivers, and receivers, and power consumed by optical components. The power consumed by optical components also includes the optical losses that occur during the propagation of light through the optical waveguides such as distance dependent power loss, bending loss, coupling loss, waveguide crossing loss, and the power loss that occurs while travelling across the ring resonators. However, in electrical interconnects the messages have to travel from one hop to another and hence consumes power in several different ways such as power consumed in traversing the crossbars in the electrical routers, power consumed in electric buffers and the power losses that occur while traversing the metallic interconnects.

For electrical interconnects, we assume that the state-of-art electrical links at 10nm technology consume $13pJ$ of energy for transmitting 128-bit message [Joshi et al. 2009; Pan et al. 2009; Batten et al. 2008] and $16pJ$ of energy is required to traverse through the crossbar of an electrical router [Joshi et al. 2009; Pan et al. 2009]. Thus, the total energy required to transmit a single flit (128 bits) in an electrical interconnect is $29pJ/flit/hop$. However, for optical interconnects, we adopted the analytical model proposed by Joshi et al. [Joshi et al. 2009] to calculate the total energy consumed by each message. The various optical losses that occur during the course of data transmission are modelled according to the values given in Table III. Based on these parameters, we analytically calculated the power consumption in an optical network by first finding the minimum power required by the receiver to detect the message. We then calculated the minimum power required at the modulator by visiting the nodes in reverse from the receiver to the modulator, including all the losses that occur in between the sender and a receiver. The minimum power required is calculated using the $P_d * 10^{A/10}$ equation, where P_d is the minimum power required at the photodetector and A is the attenuation occurred (incorporating all the optical losses). Using the analytic model, we found that the energy required to send a single bit in our proposed optical network is $1.3pJ$.

5.4. Optical vs Electrical Networks

Using optical networks to replace electrical networks for a system is still a debatable point. There are arguments that say that kilo-core systems with electrical interconnects will not be feasible because of a large number of hops and the resultant NoC complexity [Sikder et al. 2015]. We carried out simulations in order to compare the performance of electrical and optical networks for kilo-core systems. We use a mesh based electrical network proposed by Kumar et al. [Kumar et al. 2002], which we treat as a baseline, and expanded it to 768 cores. Among kilo-core networks, we found this network to deliver the highest performance. The results for the simulated execution time are shown in Figure 6. We compare the scheme by Kumar et al. with *BigBus_PS*.

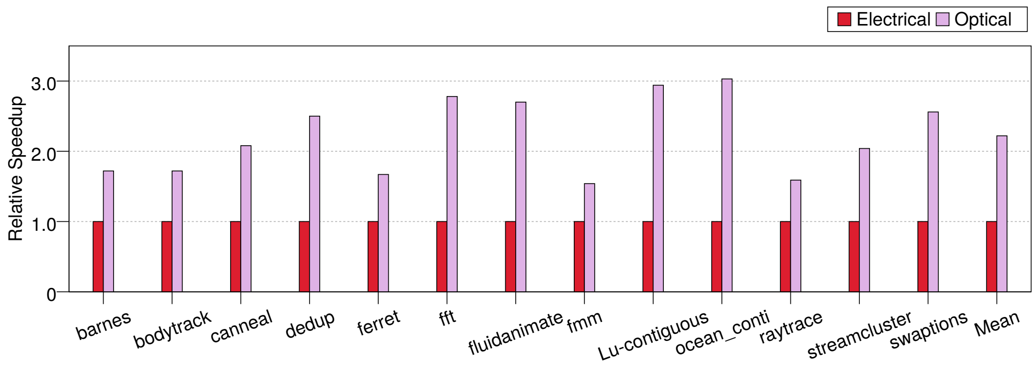

Fig. 6: Optical vs Electrical Networks

We find *BigBus_PS* to be 53% faster, and thus we do not consider electrical networks henceforth.

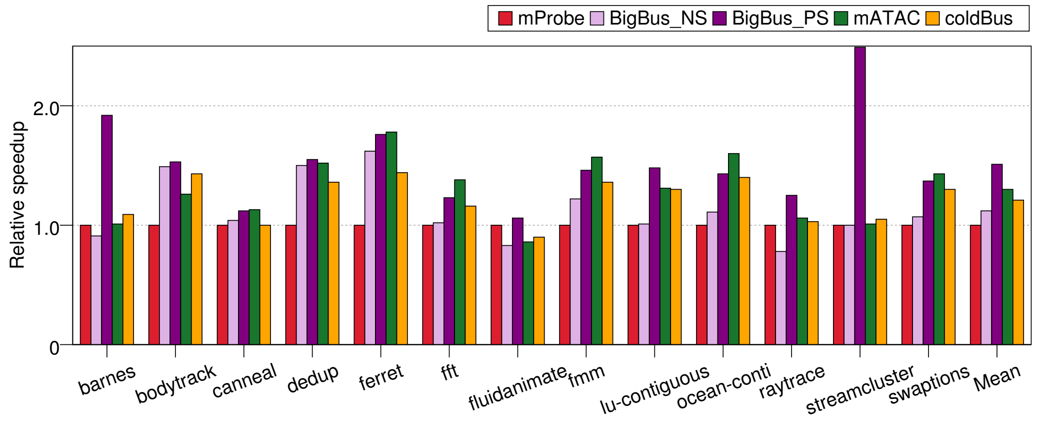

Fig. 7: Performance comparison

5.5. Performance (Basis: Simulated Execution Time)

In Figure 7 we compare the performance of different configurations. *BigBus_NS* is the variant of *BigBus* with the *No Share* scheme while *BigBus_PS* is the variant with the *Partial Share* scheme. In almost every benchmark *BigBus_PS* and mATAC perform better as compared to *BigBus_NS*. Here the performance improvement of *BigBus_PS* is due to the ability to send multiple messages at the same time while in the case of *mATAC*, this is because of the availability of optical power all the time. In *mProbe*, if we allocate a single unit of optical power to a node, it can be used by that specific station. But in the case of *BigBus*, this is shared among multiple nodes. A misprediction in the case of *mProbe* forces the station to wait till the next epoch to get the optical power, while in *BigBus*, a station can share optical power among all the nodes in a *P-Cluster*. This gives *BigBus* better performance. Since *ColdBus* has a backup plan in case of mispredictions, it performs better than *mProbe*. *ColdBus* performs better than *BigBus_NS* because *BigBus_NS* does not guarantee a specific amount of power

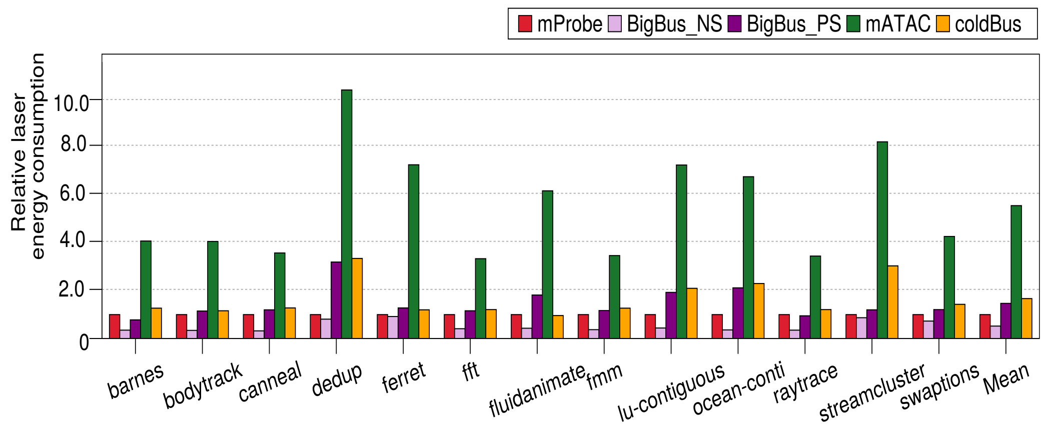

Fig. 8: Laser energy consumption

to each station; each station needs to always arbitrate for optical power. We observe that *BigBus_PS* performs better than all the other configurations because it has the capability to send multiple messages in a single cycle, provided that enough optical power is available. Compared to *mProbe*, *BigBus_NS*, *ColdBus*, and *BigBus_PS* perform 11%, 18%, and 34% better respectively. *BigBus_PS* is the best configuration and is 14% faster than *mATAC*.

Benchmarks such as *barnes*,*fmm*,*ferret*, and *streamcluster* show a high speedup. The reason for this is the very low message wait times of *BigBus_PS* as compared to mProbe. *BigBus_PS* shows a good improvement for *streamcluster* because the traffic is very bursty in nature, and thus sending messages in parallel proves to be very beneficial. Other benchmarks such as *barnes*, *streamcluster*, and *raytrace* show inferior performance with *BigBus_NS* because of the high wait times as compared to *mProbe*.

5.6. Energy consumption

Optical energy consumption primarily depends on the duration for which the laser is turned on. It also depends on the effectiveness of the prediction mechanism and bandwidth sharing. On the flip side, keeping the laser turned off for a longer time can decrease performance. Figure 8 shows the relative laser energy consumption. *BigBus_NS* consumes 48% less laser energy as compared to *mProbe* and 65% less energy in comparison to *BigBus_PS*. Because of the lack of laser modulation, *mATAC* is the most energy consuming configuration. It consumes 10 times more energy as compared to *BigBus_NS*. Compared to *ColdBus*, *BigBus_PS* consumes 12% less laser energy.

In all the benchmarks, the energy consumption of *BigBus_NS* is lower compared to *mProbe* because we share power among multiple nodes more effectively. In the case of *mProbe*, if we assign a certain amount of optical power to a station, it can be used only by that station. But in the case of *BigBus*, the power can be shared among multiple nodes.

It should be noted that the laser power consumed includes the laser power for backbone (for sending messages), token and the prediction waveguides. Figure 9 shows the average optical energy consumption breakdown in a *BigBus* configuration. The token waveguide power consumption is less than 2% of the total power consumed and is roughly the same for every benchmark relative to the total optical power consumed. The reason is that the power consumed is directly proportional to the number of optical tokens used. In addition, for the prediction waveguide, we divert 1 unit of power (corresponding to 1 unicast) to it 9 cycles before the end of the epoch. As a result, the laser power consumed for the prediction waveguide is directly proportional to the number of epochs taken by the simulation to complete. The average amount of laser power consumed by the prediction waveguide in every epoch is less than 0.8% of the total optical power.

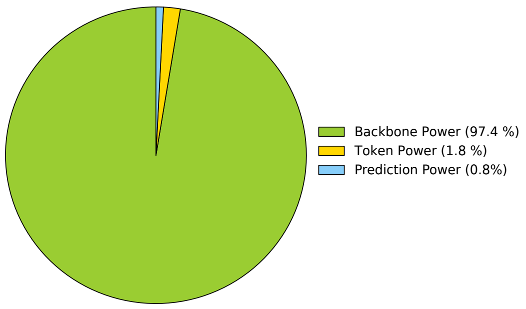

Fig. 9: Optical energy consumption breakdown

5.7. ED^2 Comparison

Figure 10 shows the comparison of different configurations in terms of the energy-delay^2 product (ED^2). Note that here the energy refers to the energy of the full system. The delay is the total time taken by the simulation to complete (simulated execution time). This includes the effect of network congestion, and all other sources of network induced delay. The values are normalized to those of *mProbe*. *BigBus_PS* is the best configuration with a reduction of 61% as compared to *mProbe*, *ColdBus* is the second best configuration (41% reduction), then *mATAC* with 37% reduction and finally *BigBus_NS* has a reduction of 13%.

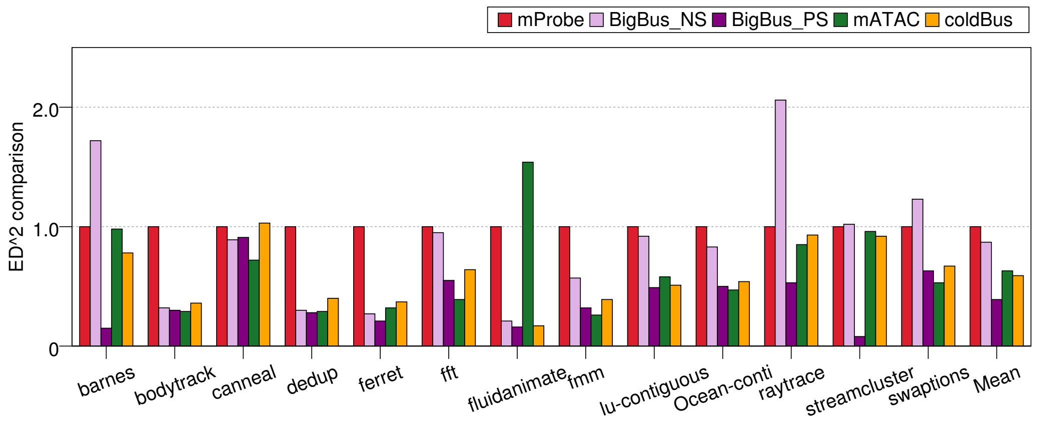

Fig. 10: ED^2 comparison

Let us now look at some benchmark specific trends. In *barnes*, *streamcluster* and *raytrace*, the performance of *BigBus_NS* is inferior to other configurations, and this caused a large increase in the corresponding ED^2 metric. In *bodytrack*, *fmm* and *dedup*, the speedups of *BigBus_PS* and *mATAC* are high compared to *mProbe*, which resulted in a large gain in ED^2 numbers. For a similar reason *BigBus_PS* shows a large improvement in ED^2 in *streamcluster*. The high power consumption and lower performance of *mATAC* in *streamcluster* accounts for its high ED^2 value. Nevertheless, since *mATAC* is faster, in 6 out of 13 benchmarks, the ED^2 values of *mATAC* are lower than *BigBus_PS*. Nevertheless, on an average *BigBus_PS* is much better than *mATAC*.

5.8. Contention

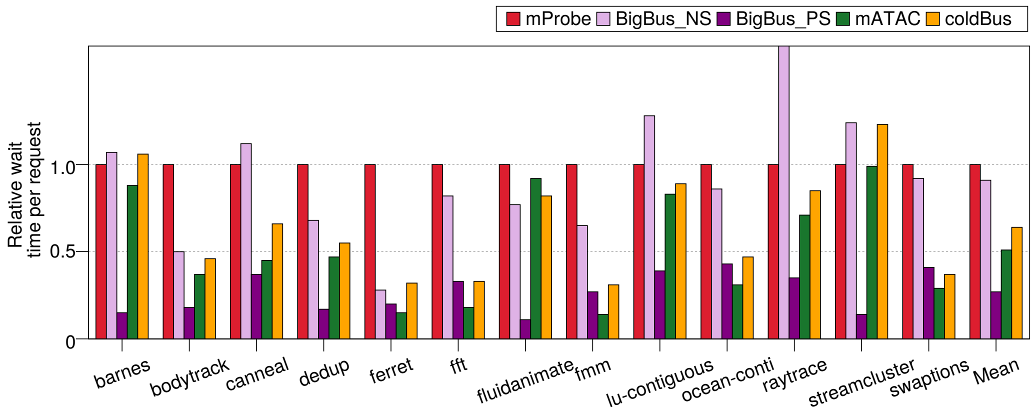

Fig. 11: Average wait time per request

5.8.1. Wait Time at Queues. In *BigBus*, we have two kinds of queues: one is at each station to hold the messages from cores/cache banks and the other is at each hub connecting the O-Link and the top level link. Due to the presence of many requests at a time at a station, there can be many messages in the queue, waiting to be sent in the coming cycles. Also requests at the head of a queue at a station need to wait till they grab a token. In Figure 11, we show the average wait time of a message at a station before it gets transmitted. The values are normalized to the *mProbe* configuration. The average wait time per request is the least for *BigBus_PS*. Specifically, the lower wait time in the case of *barnes*, *ferret*, *fluidanimate* and *streamcluster* is the reason for the high performance of *BigBus_PS*.

If we consider the exact numbers then the comparison is as follows. In the case of *No Share*, *streamcluster* has the maximum number (115) of wait cycles at a queue while *dedup* and *ferret* have the least ($\approx$25). This explains the roughly similar performance between the NS and PS schemes for these two benchmarks. If we compare the waiting times between *No Share* and *Partial Share*, the *Partial Share* configuration always outperforms *No Share* because of the sharing mechanism. In *Partial Share*, the network traffic was efficiently handled by the sharing mechanism so that the messages could be sent within roughly 10-15 cycles. The *Partial Share* scheme decreases the wait time by roughly 71% as compared to the *No Share* scheme, and this is the main reason for performance improvement. Note that the terms, message and request, are used interchangeably in this subsection.

The queue occupancy at a station can increase due to different reasons: (1) the laser is off due to misprediction (2) the laser power is not available due to high traffic (not able to acquire a token even if there are tokens), and (3) a station is busy in sending some other messages (only in the case of NS) of some other nodes in the same cluster. Please refer to Table V.

We encounter requests waiting because of reason (1) very infrequently. Column (A) shows the fraction of requests in the queue because of reason (2). We observe that for NS, this reason accounts for only 1.7% of total requests, whereas it accounts for roughly 99% of requests in the case of PS. The remaining requests in the case of PS cannot be transmitted because of reason (1) (laser power not available). Column (B) shows the fraction of messages for NS that cannot be transmitted because of reason (3) (station is busy in sending other messages). This reason accounts for 98% of all requests.

A high value in column (A) denotes that the activity in *P-Clusters* was high and we fell short of tokens. A high value in column (B) means that (only in the case of NS) there were too many requests at a station. Let us consider the case of *streamcluster*, we can see that almost 99.98% of requests are there in the queue because a lot of nodes within a station tried to simultaneously send messages. This is because of the bursty nature of the benchmark. It also means that only a few clusters were very active, and the rest were inactive; otherwise, we would have fallen short of tokens (because the value in column (A) is very low: 0.02).

Benchmarks	Frac. reqs. due to reason (2) (%) (A)		Frac. reqs. due to Reason (3) for NS (%) (B)	Average # of requests in the system per epoch (C)			Average # of tokens in the system per epoch (D)		# of queue overflows per epoch in a hub with queue size(E)
	NS	PS		NS	PS	mProbe	NS	PS	
barnes	0.4	99.86	99.59	536.8	1156.54	609.57	25.59	119.39	3.68
bodytrack	0.65	95.89	99.31	421.25	468.89	282.57	27.41	93.05	0.68
canneal	1.95	99.85	98.04	647.63	850.98	703.91	28.84	107.4	2.11
dedup	1.01	97.75	98.97	135.37	149.36	96.21	21.42	86.57	0.34
ferret	2.01	99.82	98.11	194.43	223.94	128.83	19.38	78.12	0.02
fft	3.81	99.92	96.18	851.05	1082.74	491.38	32.95	110.05	3.31
fluidanimate	0.11	99.77	99.88	573.19	940.26	487.05	55.94	109.93	1.98
fmm	4.5	99.9	95.49	744.91	947.83	635.11	29.06	110.31	2.88
lu-contiguous	1.16	98.91	97.03	653.18	859.47	585.4	31.83	114.21	1.21
ocean-conti	2.89	99.19	96.13	732.14	877.38	603.18	26.45	103.65	2.36
raytrace	1.35	99.8	98.64	641.11	1069.23	834.61	26.6	112.77	3.58
streamcluster	0.02	99.94	99.98	375.73	880.05	380.97	30.89	102.28	2.48
swaptions	2.73	99.64	97.16	447.38	628.42	338.7	28.17	107.59	3.07
Mean	1.74	99.25	98.04	534.94	779.62	475.19	29.58	104.26	2.13

Table V: Analysis of contention

5.8.2. Number of requests. The number of requests per epoch can increase till roughly 800. We show the mean of these distributions in column (C) (note that it is across the entire system). The average number of requests vary between 300-1100 for the entire system (in the case of PS). We do not count intra-cluster messages using the electrical network. The number of requests is high in the case of PS, because the number of epochs are fewer (PS runs faster).

In column (D), we show the average number of tokens generated per epoch. The number of tokens is directly proportional to the optical energy consumption. We observe that the number of tokens in PS is roughly 3.5 times more than in NS. It is true that the wait times in PS are lower. However, we have given a priority to the length of the queue while predicting the number of tokens. As shown in column (A), in PS the activity is higher and as a result more requests are generated, and almost all these requests wait in the queue for other requests to complete. They thus end up increasing the occupancy's of the queues, and this leads to more tokens being generated.

5.8.3. Contention at Hubs. We have a queue at each of the 4 hubs. In column (E) of Table V, we show the average number of overflows in a hub's queue per epoch using a queue of size 100. We can see an average of 2.5 overflow in every epoch. If we increase the size of the queue to 200, the average number of overflows per epoch in a hub reduces to nearly half.

5.9. Accuracy of Prediction

In *BigBus*, the optical power is predicted every epoch. It is not possible to directly ascertain the accuracy of prediction, because of the following reasons. Optical power is not a binary value, rather it can take 32 values. Second, since the power is shared across stations in a *P-Cluster*, even if we predict a lower amount of power, it might not make a big difference. There will be a delay in sending messages; however, this delay can get subsequently masked. It is thus impossible to exactly assess the impact of provisioning lesser or higher power. Instead, we can use the "increase in queue occupancy due to unavailability of laser power" as an indirect measure. This will quantify the contention in queues due to lack of power. The average value of this parameter is less than 0.01 per hundred requests. This shows that the accuracy of our prediction mechanism is very high.

Along with queue occupancy, we have shown the average wait time per request for different configurations in Figure 11. This also acts as an indirect measure of the prediction accuracy. As the wait time increases, we can assume that the prediction is wrong most of the time because the unavailability of optical power results in increased wait time. In all the configurations, *BigBus_PS* has the least average wait time per request, which clearly shows that the accuracy of our prediction mechanism is very high.

5.10. Different Epoch Sizes

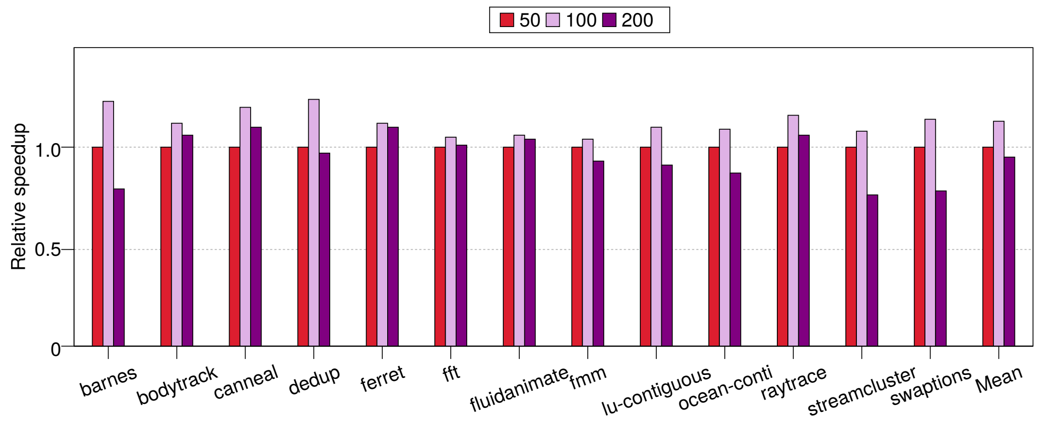

Fig. 12: Performance comparison using different epoch sizes

To evaluate the effect of different epoch sizes, we compare the speedup of *BigBus_PS* with different epoch sizes: 50, 100 and 200 (see Figure 12). In every epoch, we have a fixed duration in which the network is kept inactive to reconfigure the laser power and split ratios of the tunable splitters. Hence, the effective epoch size reduces, and the performance can degrade due to this inactive period. For large epoch sizes, the impact of a misprediction is high. We will either end up wasting a lot of laser power, or lose performance because we do not have enough tokens. For a small epoch size, the prediction accuracy suffers because we do not have sufficient information to make an accurate prediction. Also, we lose a larger proportion of cycles in reconfiguring the network. From the figure, we can conclude that an epoch size of 100 is the best option (11% better than the nearest competitor, epoch size of 50), and thus we choose it.

5.11. Thermal Simulation

Microrings are extremely sensitive to temperature fluctuations. The drift in the resonant wavelength is 0.09nm/°C. A 20°C change can shift the resonant frequency by 1.8 nm. If we assume 64 equispaced wavelengths between 1450 nm and 1650 nm, then the separation between two wavelengths is 3.125 nm. Thus, large temperature changes can make us read the wrong wavelength, also the power coupled into the photodetector decreases significantly even with small changes in temperature. The standard approach to solve these problems is to bring all the ring resonators to a given maximum temperature, τ. All the ring resonators can be designed to operate at τ.

To realize this aim, it is necessary to have micro-heaters that heat each ring resonator such that its temperature reaches τ °C. Micro-heaters are resistive elements and when current is passed through them they generate heat, which can be used to tune the microring resonators [Padmaraju and Bergman 2014]. We use the scheme proposed by Hasitha et al. [Jayatilleka et al. 2015]. It describes the placement and operation of micro-heaters for tuning micro-ring resonators. If the temperature of a ring resonator is $T°C$, we need to increase its temperature by $\tau - T\ °C$. To do this, we first got detailed power traces for all our simulations from the Tejas simulator. We simulated all the standard components of power consumption such as core power, cache power, and NoC power. We also simulated the optical power: input laser power, trimming power, and power consumption in the transmitter/receiver circuits. Subsequently, we used these power values to compute the steady state temperature using a thermal simulator, HotSpot. The temperature leakage convergence loop was taken into account. We set the maximum temperature(τ) at $85°C$, and computed the trimming power (1 $\mu W/°C$ [Pan et al. 2010; Nitta et al. 2011]). Note that there is a feedback loop here because the trimming power determines the total power, which determines the chip temperature and leakage power. We iterate till we get the converged steady state values.

The mean trimming power is roughly 5.8W for our benchmarks, and the mean optical power varies from 30-45W. The total trimming power is thus not prohibitive.

6. CONCLUSION

In this paper, we proposed a novel optical network, *BigBus*, for the 1000-core era. We opted for a novel hybrid design that uses both laser modulation and power sharing across stations. The former approach is very effective in reducing static optical power, and the latter approach is effective in making the best utilization of the power that is available. To take both of these design decisions into account we proposed to use tokens for distributing both power and access to data waveguides. By using the currency of tokens, we could also simplify our design, and propose a predictor that predicted the number of tokens that we need to generate per epoch.

It is true that using tokens has a flip side, which is that it increases the latency of each transmission because a station needs to first arbitrate for a token. However, we compensated for this shortcoming by allowing a station to grab multiple tokens at once and thus were able to increase performance by 26% (*BigBus_PS* vs *BigBus_NS*) over a baseline design that does not have this facility. We also showed that our best design, *BigBus_PS*, outperforms state of the art proposals such as *mProbe*, *mATAC*, and ColdBus by 34%, 14%, and 20% respectively. Finally, we demonstrated that this design decision (in *BigBus_PS*) makes sense in a system with 1000 cores because any other scheme that just uses prediction such as *mProbe* has a much higher ED^2 (61%).

REFERENCES

José L. Abellán, Chao Chen, and Ajay Joshi. 2016. Electro-Photonic NoC Designs for Kilocore Systems. *J. Emerg. Technol. Comput. Syst.* 13, 2 (Nov. 2016), 24:1–24:25. DOI:https://doi.org/10.1145/2967614

James Balfour and William J. Dally. 2014. Design Tradeoffs for Tiled CMP On-chip Networks. In *ACM International Conference on Supercomputing 25th Anniversary Volume*. ACM.

J. Bashir and S. R. Sarangi. 2017. NUPLet: A Photonic Based Multi-Chip NUCA Architecture. In *2017 IEEE International Conference on Computer Design (ICCD)*.

Christopher Batten, Ajay Joshi, Jason Orcutt, Anatoly Khilo, Benjamin Moss, Charles Holzwarth, Milos Popovic, Hanqing Li, Henry I Smith, Judy Hoyt, et al. 2008. Building manycore processor-to-dram networks with monolithic silicon photonics. In *High Performance Interconnects (HOTI)*. IEEE, 21–30.

C. Bienia, S. Kumar, J. P. Singh, and K. Li. 2008. The PARSEC benchmark suite: characterization and architectural implications. In *PACT*.

Z. Cao, R. Proietti, and S. J. B. Yoo. 2014. Scalable and high performance HPC architecture with optical interconnects. In *2014 IEEE Photonics Conference*.

Chao Chen and Akanksha Joshi. 2013. Runtime management of laser power in silicon-photonic multibus noc architecture. *Selected Topics in Quantum Electronics, IEEE Journal of* 19, 2 (2013), 3700713–3700713.

Sai Vineel Reddy Chittamuru, Srinivas Desai, and Sudeep Pasricha. 2017. SWIFTNoC: A Reconfigurable Silicon-Photonic Network with Multicast-Enabled Channel Sharing for Multicore Architectures. *J. Emerg. Technol. Comput. Syst.* (June 2017).

UC Davis. 2016. Worlds First 1,000-Processor Chip. (2016). https://www.ucdavis.edu/news/worlds-first-1000-processor-chip/.

Yigit Demir and Nikos Hardavellas. 2014. EcoLaser: an adaptive laser control for energy-efficient on-chip photonic interconnects. In *ISLPED*.

EPSRC. 2013. PRiME: Power-efficient, Reliable, Many-core Embedded systems. (2013). http://www.prime-project.org/.

M. FAUGERON, M. Chtioui, A Enard, O. Parillaud, F. Lelarge, M. Achouche, J. Jacquet, A Marceaux, and F. van Dijk. 2013. High Optical Power, High Gain and High Dynamic Range Directly Modulated Optical Link. *Lightwave Technology, Journal of* 31, 8 (April 2013), 1227–1233.

H. Gu and J. Xu. 2009. Design of 3D Optical Network on Chip. In *2009 Symposium on Photonics and Optoelectronics*.

Anna Y Herr, Quentin P Herr, Oliver T Oberg, Ofer Naaman, John X Przybysz, Pavel Borodulin, and Steven B Shauck. 2013. An 8-bit carry look-ahead adder with 150 ps latency and sub-microwatt power dissipation at 10 GHz. *Journal of Applied Physics* 113, 3 (2013), 033911.

Wei Huang, K. Rajamani, M.R. Stan, and K. Skadron. 2011. Scaling with Design Constraints: Predicting the Future of Big Chips. *Micro, IEEE* 31, 4 (2011), 16 –29.

M. J. Humphrey. 1994. *Calculation of coupling between tapered fiber modes and whispering-gallery modes of a spherical microlaser*. Ph.D. Dissertation. University of Maryland, College Park, Maryland.

ARM Inc. 2007. ARM Unveils Cortex-A9 Processors For Scalable Performance and Low-Power Designs. (2007). https://www.arm.com/about/newsroom/18688.php.

Hasitha Jayatilleka, Kyle Murray, Miguel Ángel Guillén-Torres, Michael Caverley, Ricky Hu, Nicolas AF Jaeger, Lukas Chrostowski, and Sudip Shekhar. 2015. Wavelength tuning and stabilization of microring-based filters using silicon in-resonator photoconductive heaters. *Optics Express* 23, 19 (2015), 25084–25097.

Ajay Joshi, Christopher Batten, Yong-Jin Kwon, Scott Beamer, Imran Shamim, Krste Asanovic, and Vladimir Stojanovic. 2009. Silicon-photonic clos networks for global on-chip communication. In *NoCS*.

Andrew B Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. 2011. Orion 2.0: A power-area simulator for interconnection networks. Institute of Electrical and Electronics Engineers.

Nevin Kirman and José F. Martínez. 2010. A power-efficient All-optical On-chip Interconnect Using Wavelength-based Oblivious Routing. In *ASPLOS*.

S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. 2002. A network on chip architecture and design methodology. In *VLSI*. 105–112.

George Kurian, Jason E Miller, James Psota, Jonathan Eastep, Jifeng Liu, Jurgen Michel, Lionel C Kimerling, and Anant Agarwal. 2010. ATAC: a 1000-core cache-coherent processor with on-chip optical network. In *PACT*.

Jacob S Levy, Yoshitomo Okawachi, Michal Lipson, Alexander L Gaeta, and Kasturi Saha. 2011. High-performance silicon-based multiple wavelength source. In *CLEO: Science and Innovations*. Optical Society of America, CMAA7.

Sheng Li, Jung Ho Ahn, Richard D Strong, Jay B Brockman, Dean M Tullsen, and Norman P Jouppi. 2009. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *MICRO*.

Moustafa Mohamed, Zheng Li, Xi Chen, and Alan Mickelson. 2014. HERMES: A Hierarchical Broadcast-Based Silicon Photonic Interconnect for Scalable Many-Core Systems. *arXiv preprint arXiv:1401.4629* (2014).

Randy Morris, Evan Jolley, and Avinash Karanth Kodi. 2014. Extending the performance and energy-efficiency of shared memory multicores with nanophotonic technology. *Parallel and Distributed Systems, IEEE Transactions on* 25, 1 (2014), 83–92.

C. Nitta, M. Farrens, and V. Akella. 2011. Addressing system-level trimming issues in on-chip nanophotonic networks. In *HPCA*. 122–131.

Ian O'Connor. 2004. Optical solutions for system-level interconnect. In *Proceedings of the 2004 international workshop on System level interconnect prediction*. ACM, 79–88.

Kishore Padmaraju and Keren Bergman. 2014. Resolving the thermal challenges for silicon microring resonator devices. *Nanophotonics* 3, 4-5 (2014), 269–281.

Yan Pan, John Kim, and Gokhan Memik. 2010. Flexishare: Channel sharing for an energy-efficient nanophotonic crossbar. In *HPCA*.

Yan Pan, Prabhat Kumar, John Kim, Gokhan Memik, Yu Zhang, and Alok Choudhary. 2009. Firefly: illuminating future network-on-chip with nanophotonics. In *ACM SIGARCH Computer Architecture News*. ACM.

Sudeep Pasricha and Shirish Bahirat. 2011. OPAL: A multi-layer hybrid photonic NoC for 3D ICs. In *Design Automation Conference (ASP-DAC), 2011 16th Asia and South Pacific*. IEEE, 345–350.

Eldhose Peter, Anuj Arora, Janibul Bashir, Akriti Bagaria, and Smruti R. Sarangi. 2017. Optical Overlay NUCA: A High-Speed Substrate for Shared L2 Caches. *J. Emerg. Technol. Comput. Syst.* (May 2017).

Eldhose Peter and Smruti R Sarangi. 2014. OptiKit: An Open Source Kit for Simulation of On-Chip Optical Components. (2014).

Eldhose Peter and Smruti R Sarangi. 2015. Optimal Power Efficient Photonic SWMR Buses. In *Silicon Photonics (with HiPEAC)*.

Eldhose Peter, Arun Thomas, Anuj Dhawan, and Smruti R Sarangi. 2015. ColdBus: A Near-Optimal Power Efficient Optical Bus. In *HiPC*.

Eldhose Peter, Arun Thomas, Anuj Dhawan, and Smruti R Sarangi. 2016. Active microring based tunable optical power splitters. *Optics Communications* (2016).

R. Proietti, Zheng Cao, Yuliang Li, and S. J. B. Yoo. 2014. Scalable and distributed optical interconnect architecture based on AWGR for HPC and data centers. In *OFC 2014*.

John Rath. 2013. Fujitsu Lights up PCI Express with Intel Silicon Photonics. (2013). http://www.datacenterknowledge.com/archives/2013/11/07/fujitsu-lights-pci-express-intel-silicon-photonics.

Graham T. Reed. 2008. *Silicon Photonics: The State of the Art*. John Wiley & Sons.

S. R. Sarangi, Kalayappan Rajshekar, Kallurkar Prathmesh, Goel Seep, and Peter Eldhose. 2015. Tejas: A Java based Versatile Micro-architectural Simulator,. In *PATMOS*.

Md Ashif I Sikder, Avinash K Kodi, Matthew Kennedy, Savas Kaya, and Ahmed Louri. 2015. OWN: Optical and Wireless Network-on-Chip for Kilo-core Architectures. In *High-Performance Interconnects (HOTI)*. 44–51.

Mircea R. Stan, Kevin Skadron, Wei Huang, and Karthick Rajamani. 2011. Scaling with Design Constraints: Predicting the Future of Big Chips. *IEEE Micro* 31 (2011).

I. G. Thakkar, S. V. R. Chittamuru, and S. Pasricha. 2016. Run-time laser power management in photonic NoCs with on-chip semiconductor optical amplifiers. In *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*.

S Thoziyoor, N Muralimanohar, JH Ahn, and NP Jouppi. 2008. Cacti 5.3. *HP Laboratories, Palo Alto, CA* (2008).

A. W. Topol, D. C. L. Tulipe, L. Shi, D. J. Frank, K. Bernstein, S. E. Steen, A. Kumar, G. U. Singco, A. M. Young, K. W. Guarini, and M. Ieong. 2006. Three-dimensional integrated circuits. *IBM Journal of Research and Development* (July 2006).

Dana Vantrease, Nathan Binkert, Robert Schreiber, and Mikko H Lipasti. 2009. Light speed arbitration and flow control for nanophotonic interconnects. In *Microarchitecture, 2009. MICRO-42*. IEEE.

Dana Vantrease, Robert Schreiber, Matteo Monchiero, Moray McLaren, Norman P. Jouppi, Marco Fiorentino, Al Davis, Nathan Binkert, Raymond G. Beausoleil, and Jung Ho Ahn. 2008. Corona: System Implications of Emerging Nanophotonic Technology. In *ISCA*.

Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. 1995. The SPLASH-2 programs: characterization and methodological considerations. *SIGARCH Comput. Archit. News* 23 (May 1995), 24–36.

Xiaowen Wu, Jiang Xu, Yaoyao Ye, Zhehui Wang, Mahdi Nikdast, and Xuan Wang. 2014. SUOR: Sectioned Undirectional Optical Ring for Chip Multiprocessor. *J. Emerg. Technol. Comput. Syst.* 10, 4, Article 29 (June 2014), 25 pages.

Yi Xu, Jun Yang, and Rami Melhem. 2012. Channel borrowing: an energy-efficient nanophotonic crossbar architecture with light-weight arbitration. In *ICS*.

Y. Ye, L. Duan, J. Xu, J. Ouyang, M. K. Hung, and Y. Xie. 2009. 3D optical networks-on-chip (NoC) for multiprocessor systems-on-chip (MPSoC). In *2009 IEEE International Conference on 3D System Integration*.

Y. Ye, J. Xu, B. Huang, X. Wu, W. Zhang, X. Wang, M. Nikdast, Z. Wang, W. Liu, and Z. Wang. 2013. 3-D Mesh-Based Optical Network-on-Chip for Multiprocessor System-on-Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (April 2013).

Y. Yin, R. Proietti, C. J. Nitta, V. Akella, C. Mineo, S. J. B. Yoo, and K. Wen. 2013. AWGR-based all-to-all optical interconnects using limited number of wavelengths. In *2013 Optical Interconnects Conference*.

R. Zhang, M. R. Stan, and K. Skadron. 2015. Hotspot 6.0: Validation, acceleration and extension. (2015).

Li Zhou and Avinash Karanth Kodi. 2013. Probe: Prediction-based optical bandwidth scaling for energy-efficient nocs. In *NOCS*.

Arslan Zulfiqar, Pranay Koka, Herb Schwetman, Mikko Lipasti, Xuezhe Zheng, and Ashok Krishnamoorthy. 2013. Wavelength stealing: an opportunistic approach to channel sharing in multi-chip photonic interconnects. In *MICRO*.

APPENDIX

A. SYSTEM FEASIBILITY AND THERMAL SIMULATION SETUP

Feasibility of the System : Let us illustrate a reference design with the ARM Cortex A9 processor [Inc 2007], whose area is $1.5mm^2$ at 65nm technology. Note that ARM Cortex A9 has 2-issue OOO cores and we use dual issue inorder cores. Hence, our cores are expected to be smaller. We only provide an upper bound in this section. Now, using the scaling factors provided by Stan et al. [Stan et al. 2011], we compute the size of the core to be $0.19mm^2$ at 10nm technology. With such a small core, it is possible to have 768 cores occupying $146mm^2$. We can also integrate 256 cache banks of capacity 32MB on an area less than $158mm^2$ (calculated using Cacti 6.0 and scaled using [Stan et al. 2011]). Thus, our 1024 nodes (768 cores + 256 caches) require $304mm^2$ at 10nm technology. If we budget an additional 25% (results from Intel's SCC processor) of area for interconnects, and memory controllers, our total chip area comes to $405mm^2$, which is the size of a standard die for high end processors. Even if we consider power, the design is feasible (please see the simulations in Section 5.7). **Note that our approach is not specific to the reference design. It can be used for any large system of cores and caches.**

A.0.1. Thermal Simulation. BigBus has an optical layer and a logical layer connected together using through-silicon vias. The chip temperature depends on the power consumed by cores, cache banks, and other logic layer components. Moreover, the power consumption and the power losses associated with the optical layer is also responsible for the on-chip temperature variation. All the power related values are provided by the Tejas simulator, which includes the Orion and McPAT tools to provide the power numbers associated with the logic layer. The analytical model proposed by Joshi et al. [Joshi et al. 2009] and the laser activity statistics provided by Tejas are used to calculate the power associated with the optical layer. The power profile thus obtained is provided as input to the thermal simulator, along with the floorplan of the chip. Since our chip has a 3D structure, we use the 3D extension of HotSpot [Zhang et al. 2015] to simulate temperature. In our simulations, we set the ambient temperature as $35°C$, and use the default package parameters of HotSpot. For the optical components, we aggregate all the optical components on a single layer and the laser sources are modelled separately by incorporating a layer of lasers. The 3D cross-sectional view along with the dimensions of each component used in our system are shown in Figure 4. In our thermal simulations, the chip reaches to a peak temperature of $82°C$.

B. SCALABILITY ANALYSIS

Our paper proposes a scalable optical interconnect for a 1000-node system. Here scalable refers to the fact that the proposed network will work efficiently for such a large system without incurring a large loss in performance, and without consuming a lot of power. This is because of the fact that for such systems, the current optical NoCs [Chittamuru et al. 2017; Pan et al. 2010] will fail to scale. The reason being the large size and number of waveguides when scaled to a 1000-node system. These long waveguides will have a large number of cascaded splitters which will increase the power consumption exponentially [Peter and Sarangi 2015]. Moreover, we are still years away from a 1000-node system, thus our results will continue to hold for at least next decade. Even beyond, we can add more P-clusters without increasing the length of the waveguides.

For completeness, we compare our design with a state-of-art photonic NoC proposed by Chittamuru et al. [Chittamuru et al. 2017], called *SwiftNoC*. The authors proposed an NoC for a 256 core system. The chip has a tiled architecture. Each tile is composed of 4 cores and an optical station (router). The system is divided into four clusters, with each cluster contains 16 tiles (64 cores). The clusters are provided with MWMR waveguides in order to communicate with each other. The authors propose to use a bandwidth exchange mechanism to share the available bandwidth. In addition, they propose to monitor the traffic injected into the network and then accordingly distribute the bandwidth among the clusters. For more details please see [Chittamuru et al. 2017].

To compare *SwiftNoC* with our design, we develop two different variants of this NoC : *SwiftNoC_768* and *mSwiftNoC*. In *SwiftNoC_768*, we extended the *SwiftNoC* to 768 cores. In the final chip, there are 12 clusters, with each cluster containing 16 tiles (64 cores). We use 16 MWMR waveguide groups for each cluster. In *mSwiftNoC*, we use the bandwidth transfer and traffic monitoring mechanism of *SwiftNoC* in each *P-cluster* of the *BigBus* design. In addition, we do not use any kind of laser modulation in both the schemes.

Figure 13 shows the relative performance comparison across different configurations. It is clear that *BigBus_PS* performs better than all the other schemes. The improvement is attributed to its ability to send multiple messages at the same time. Moreover, the implementation of the NUCA scheme further enhances the performance making it the best configuration in terms of performance. The lower performance of *SwiftNoc_768* as compared to *mSwiftNoC* is because of the fact that in the former configuration the messages have to travel large distances, which increases the effective latency.

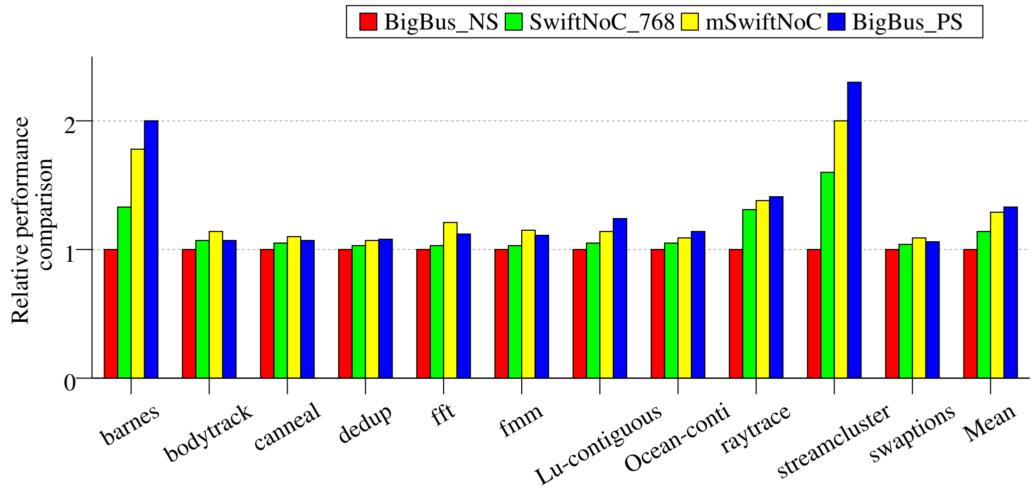

Fig. 13: Performance comparison

Figures 14 and 15 compare the relative laser power consumption and ED^2 values across different configurations. Among these, *SwiftNoC_768* consumes the highest power. The main reason is that in this configuration the messages have to travel large distances, resulting in higher optical losses. In addition, the history based bandwidth relocation mechanism is not as effective as the laser modulation technique used in different variants of *BigBus* (see Section 5.8.1), which further results in higher optical power consumption. In terms of laser power consumption, *BigBus_NS* consumes the least power which is 48% lower than its nearest competitor (*mSwiftNoC*). In terms of

ED^2 values, *BigBus_PS* results in 4%, 23% and 52% reduction as compared to *mSwiftNoC*, *SwiftNoC_768* and *BigBus_NS* respectively.

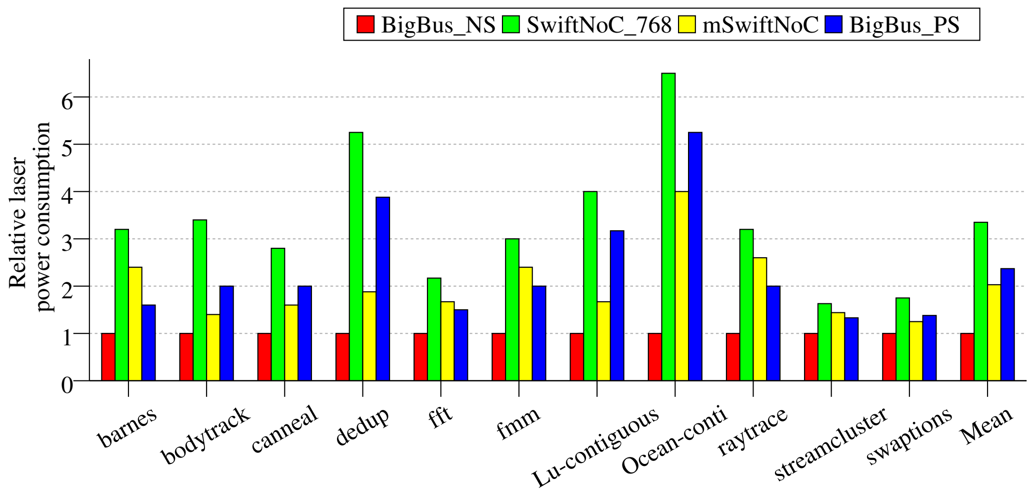

Fig. 14: Laser power consumption

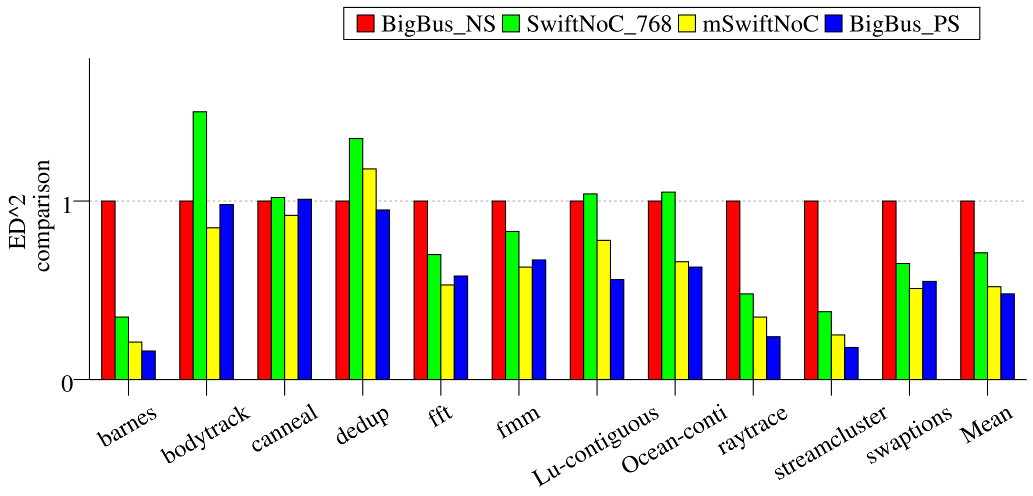

Fig. 15: ED^2 comparison

C. ARCHITECTURES PROPOSED AT UC DAVIS

The researchers at UC Davis have proposed several NoC architectures that provide all-to-all optical interconnections. Most of these research works are scalable and are based on hierarchical designs [Cao et al. 2014; Proietti et al. 2014; Yin et al. 2013]. Much of this can be attributed to the usage of AWGR based routers [Yin et al. 2013] in their design. Scalability is one common objective that both these NoCs and our proposal are trying to achieve. However, our proposal is not limited to just providing a scalable architecture. We also propose a novel prediction scheme, wavelength sharing scheme and a NUCA scheme to not only make the network scalable but at the same time increase the performance of the system and decrease the laser power consumption. Ours is the first architecture, which has deviated from the traditional approach of having cores and cache banks close to each other. It uses a separate cluster containing only the cache banks and leveraging the low latency benefits of optical communication and NUCA scheme to increase the hit rate and decrease the effective access time, resulting in increased performance.

To complete our discussion, we compared the *BigBus_PS* design with one of the scalable optical architectures proposed by Cao et al. [Cao et al. 2014] at UC Davis. The system is a hierarchical design with four different hierarchies: core, node, cabinet, and full system. The full system has $m \times n$ cabinets, with each cabinet containing r nodes and each node has c cores. We use their design to develop a system for 768 cores. We choose $m = n = 2$, $r = 6$, and $c = 32$. We call this architecture as *UCDavisNoC*.

In Figure 16, we compare the performance of *BigBus_PS* against the *UCDavisNoC*. It is clear from the figure that our scheme performs 55% better. The main reason for performance improvement is the novel NUCA scheme, which increases the hit rate in the LLC and hence decreases the execution time. Similarly, in Figure 17, we compare the laser power consumed by two different configurations. Due to an effective laser modulation technique, *BigBus_PS* results in 44% reduction in power as compared to *UCDavisNoC*.

It is clear from the results that even if both the configurations are able to scale to 1000s of nodes but still some extra novel design decisions in our design make it better than the scalable design proposed by the researchers at UC Davis.

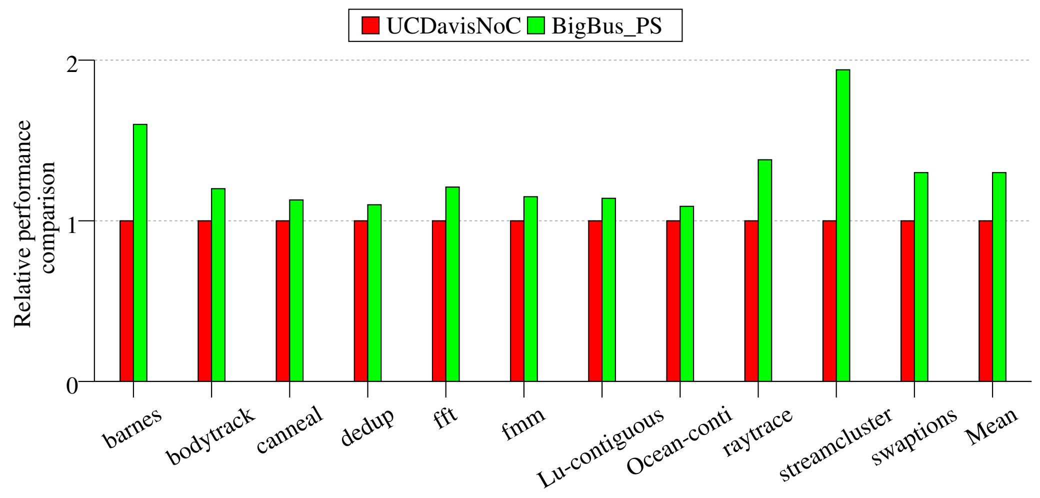

Fig. 16: Performance comparison

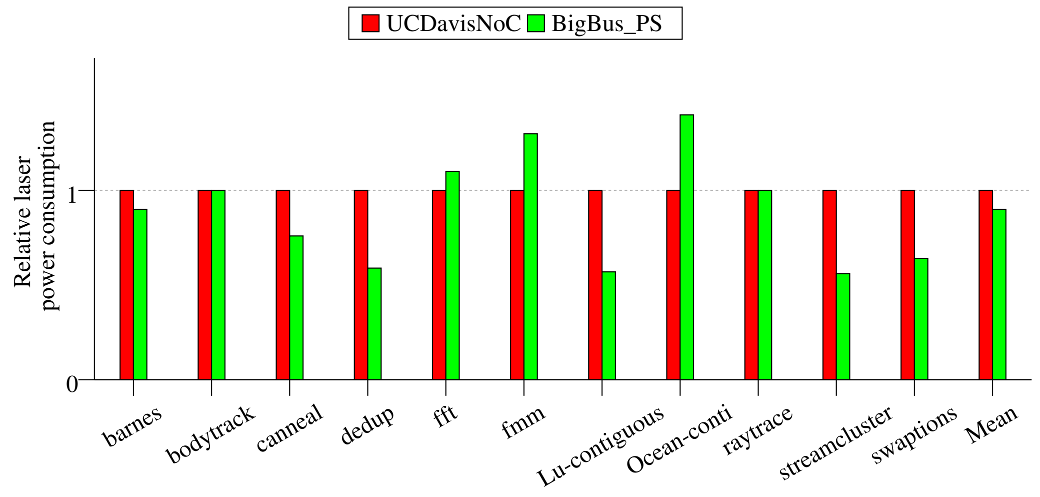

Fig. 17: Relative laser power consumption