Managing Laser Power in Silicon-Photonic NoC Through Cache and NoC Reconfiguration

Chao Chen, *Member, IEEE*, José L. Abellán, *Member, IEEE*, and Ajay Joshi, *Member, IEEE*

***Abstract*—In manycore systems, the silicon-photonic link technology is projected to replace electrical link technology for global communication in network-on-chip (NoC) as it can provide as much as an order of magnitude higher bandwidth density and lower data-dependent power. However, a large amount of fixed power is dissipated in the laser sources required to drive these silicon-photonic links, which negates any bandwidth density advantages. This large laser power dissipation depends on the number of on-chip silicon-photonic links, the bandwidth of each link, and the photonic losses along each link. In this paper, we propose to reduce the laser power dissipation at runtime by dynamically activating/deactivating L2 cache banks and switching ON/OFF the corresponding silicon-photonic links in the NoC. This method effectively throttles the total on-chip NoC bandwidth at runtime according to the memory access features of the applications running on the manycore system. Full-system simulation utilizing Princeton application repository for shared-memory computers and Stanford parallel applications for shared-memory-2 parallel benchmarks reveal that our proposed technique achieves on an average 23.8% (peak value 74.3%) savings in laser power, and 9.2% (peak value 26.9%) lower energy-delay product for the whole system at the cost of 0.65% loss (peak value 2.6%) in instructions per cycle on average when compared to the cases where all L2 cache banks are always active.**

***Index Terms*—Laser power management, manycore systems, network-on-chip (NoC) reconfiguration, silicon-photonic NoC.**

I. Introduction

TODAY'S manycore systems have tens of cores on a single chip. In the future, this core count is expected to grow into the hundreds (maybe even thousands). This ever-increasing core count has made it necessary to design a shared network-on-chip (NoC) medium for on-chip communication. The manycore systems that are commercially available today [1]–[3] use a mesh topology. The choice of this low-radix high-diameter mesh topology is driven by the fact that a mesh network requires short interconnects and energy-efficient short interconnects can be easily designed using repeater-inserted electrical link technology. However, the mesh topology, due to its multihop packet transmission, requires the programmer to carefully map an application across several cores to extract maximum performance from a manycore architecture. A single-stage high-radix topology, such as a crossbar, reduces this extra effort required for mapping of threads. However, the implementation of a crossbar using electrical link technology for systems with large core counts is expensive in terms of energy and area.

Silicon-photonic link technology has been proposed as a potential replacement to the electrical link technology in the design of the high-radix NoC of manycore systems [4]–[7]. The silicon-photonic link technology can provide an order of magnitude higher bandwidth density through dense wavelength division multiplexing [4]. Moreover, the data-dependent power in a silicon-photonic link is independent of the link length. Hence, it is possible to design a high-radix topology with long global interconnects using silicon-photonic link technology for the NoC. However, the fixed power dissipated in the laser source that is required to drive the silicon-photonic links can be nontrivial. Depending on the physical layout of the NoC and the materials used for the photonic devices, the fixed power in the laser source can more than offset the bandwidth density and data-dependent power advantages of the silicon-photonic link technology [4], [8], [9].

To enable an energy-efficient design of high-radix topologies using silicon-photonic link technology for NoC, we need to develop techniques for lowering laser power dissipation. The applications running on the manycore systems typically exhibit spatial variations and/or temporal variations in the optimal NoC bandwidth requirements, optimal L1/L2 cache size requirements and optimal core count requirements. This creates an opportunity for proactively reconfiguring the overall system architecture at runtime to minimize the power dissipated in the laser source. In this paper, we propose a run-time methodology to proactively control the size of the shared L2 cache and, in turn, the bandwidth of the NoC (between private L1 and shared L2 cache banks) to manage the power dissipated in the laser source. Our methodology periodically samples the L2 cache replacement rate to make a decision on increasing/decreasing the number of active L2 cache banks while ensuring a minimal change in the system performance. A decision to reduce the L2 cache bank count creates an opportunity to switch OFF the silicon-photonic links associated with the deactivated L2 cache banks and, in turn, save laser power. The key idea here is to provide the minimum-sized L2 cache

Manuscript received August 6, 2014; revised November 11, 2014; accepted January 27, 2015. Date of publication February 10, 2015; date of current version May 20, 2015. This work was supported by the National Science Foundation CAREER Award CCF-1149549. This paper was recommended by Associate Editor L. P. Carloni.

C. Chen is with the Digital Networking Group, Freescale Semiconductor, Inc., Austin, TX 78735 USA (e-mail: chen9810@gmail.com).

J. L. Abellán is with the Department of Computer Science, Catholic University of Murcia, Murcia 30107, Spain.

A. Joshi is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215 USA.

This work was done when C. Chen was a Ph.D. student at Boston University.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2015.2402172

(i.e., the minimum number of L2 cache banks) and the minimum NoC bandwidth (i.e., the minimum number of active silicon-photonic links) required for an application to achieve the maximum possible performance at any given point of time. The main contributions of this paper are as follows.

1) We propose a policy for run-time management of power dissipated in the laser source that drives a crossbar NoC that connects the private L1 caches and the distributed shared L2 cache banks. At runtime, using the L2 cache replacement rate metric, we determine the required change in the L2 cache bank count and hence the number of silicon-photonic links to maximize application performance and minimize laser power, which in turn maximizes the energy efficiency of the overall system.
2) We evaluate our laser power management policy using a 64-core system with silicon-photonic crossbar NoC as the target system. For evaluation, we use the Gem5 [10] simulator and princeton application repository for shared-memory computers (PARSEC) [11] and stanford parallel applications for shared-memory (SPLASH)-2 [12] parallel benchmark suites. In our analysis, we compare two different versions of the target system—a system that uses our laser management policy to reconfigure the L2 cache and the NoC, and a system that keeps the entire L2 cache and the NoC ON all the time. On average, our proposed approach saves laser power by 23.8% (peak value 74.3%) and system power by 9.9% (peak value 30.6%), with instructions per cycle (IPC) degradation of 0.65% (peak value 2.6%), and energy-delay product (EDP) improvement by 9.2% (peak value 26.9%).

The rest of this paper is organized as follows. We describe the related work in Section II followed by the detailed architecture of our target system in Section III. We present an overview of the silicon-photonic link technology in Section IV followed by a detailed discussion of our proposed runtime cache and NoC reconfiguration methodology in Section V. We present the evaluation of our proposed approach for laser power management in Section VI. This is followed by a discussion of the application of our methodology to alternate cache architectures, systems with large core counts, and alternate NoC topologies in Section VII. Finally, Section VIII summarizes the main conclusion of this paper.

II. Related Work

The silicon-photonic link technology has been widely explored to implement the entire spectrum of network topologies—from low-radix high-diameter mesh/torus topologies [7], [13] to medium-radix medium-diameter butterfly/crossbar topologies [4], [6], [14] to high-radix low-diameter bus/crossbar topologies [5], [15]. A general consensus among the various efforts so far is that silicon-photonic networks provide a bandwidth density and data-dependent power advantage for NoC communication. However, the fixed amount of power dissipated in the laser sources that drive the silicon-photonic NoC negates these advantages. Hence, to enable the use of silicon-photonic NoC in future manycore systems, we need to develop techniques to proactively manage laser power.

The standard design-time device-level solutions to reduce optical loss in silicon-photonic devices, and in turn reduce the laser power range from exploring different materials to process flows to device geometries. At the circuit level, novel receiver circuits can be designed to operate with low-sensitivity photodetectors or one can lower the power injected into the silicon-photonic links and any resulting errors can be detected/corrected using coding techniques. At the architecture level, a nanophotonic crossbar architecture that uses optical channel sharing to manage static power dissipation is proposed in [16]. Here, a token-stream mechanism is used for channel arbitration and credit distribution, to enable efficient global sharing of crossbar channels. Zhou and Kodi [17] proposed to use a prediction mechanism to dynamically scale the NoC bandwidth depending on the demands of the overlying application and in turn reduce optical power dissipation. Li *et al*. [18] proposed to divide the photonic NoC into subnets and share photonic channels for sending arbitration and data packets to reduce laser power. Chen and Joshi [19] proposed to use weighted time-division multiplexing to distribute the laser power across multiple photonic links based on the runtime variations in the bandwidth requirements within and across applications to maximize energy efficiency.

We propose to manage the laser power dissipated in the silicon-photonic NoC between the private L1 caches and distributed shared L2 cache banks through NoC and cache reconfiguration. In our approach, we determine the minimum number of L2 cache banks that is required to maximize the application performance. The extra L2 cache banks are deactivated and the silicon-photonic links associated with those L2 cache banks are switched OFF to save laser power. The idea of reconfiguring the cache architecture to reduce cache energy and/or improve performance has been explored in the past [20], [21]. However, it has never been used to manage the power dissipated in a silicon-photonic NoC. The main goal here is to minimize the EDP of the overall manycore system by leveraging the spatial and temporal variations in the behavior of the applications.

III. Target System

We choose a 64-core processor that is manufactured using 22-nm CMOS technology as our target system. However, it should be noted that our proposed technique for laser power management can easily scale with core count. Our target system is a 3-D system, where the logic layer contains the CMOS logic and monolithically integrated photonic devices, while the laser layer contains the laser sources. The cores in the logic layer are configured based on the cores used in Intel's 48-core SCC [1]. The micro-architectural parameters of the target system are listed in Table I. The target system has a total of 4 MB L2 cache. This size was determined by running all our target benchmark applications and determining the minimum L2 cache size that maximizes performance of the applications that we have used for evaluation. Alternate set of applications could indicate the need for larger L2 cache sizes. However,

TABLE I
SIXTY-FOUR CORE TARGET SYSTEM PARAMETERS

Micro-architecture Configuration	
Core Frequency	1.25 GHz @ 0.9 V
Branch Predictor	Tournament predictor
Issue	2-way Out-of-order
Reorder Buffer	128 entries
Functional Units	2 IntALUs, 1 IntMult 1 FPALU, 1 FPMult
Physical Regs	128 Int, 128 FP
Instruction Queue	64 entries
L1 I/D-Cache	Private, 16 KB each @ 2 ns
L2 Cache	8 shared and distributed banks 4-way set-associative 64 B/block, 512 KB/bank @ 6 ns
Cache Coherence	Directory based
Memory	8 memory controllers 8 PIDRAM @ 50 ns

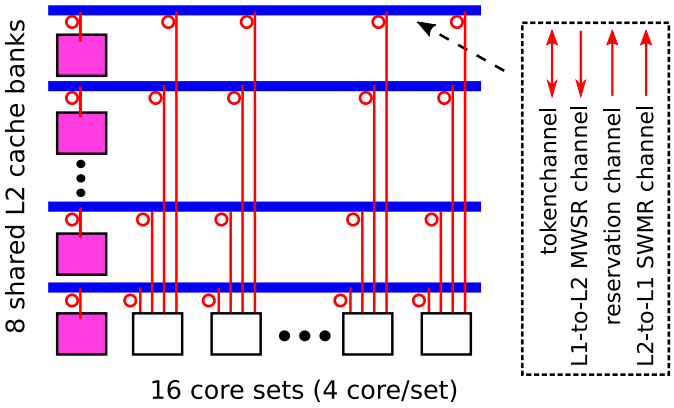


Fig. 1. Logical view of a silicon-photonic crossbar network between 64 cores (each with a private L1 cache) and 8 L2 cache banks. The crossbar uses SWMR for L2-to-L1 communication and MWSR for L1-to-L2 communication. Each ring shown in the figure represents a set of 64 ring modulators and 64 ring filters for data channels with a few extra rings for token and reservation channels.

our proposed technique of laser power management using L2 cache and NoC reconfiguration would still be applicable.

For on-chip communication, we consider a high-radix crossbar NoC architecture (see Fig. 1). The NoC employs the multiple-write-single-read (MWSR) mechanism for L1-to-L2 communication with a dedicated silicon-photonic channel having a width of 512 bits for each L2 cache bank. A token-based protocol is used to arbitrate between the L1 caches for getting access to L1-to-L2 communication channels [5]. Tokens are assigned in a round-robin fashion for fairness. The network employs the single-write-multiple-read (SWMR) mechanism for L2-to-L1 communication with a dedicated silicon-photonic channel having a width of 512 bits for each L2 cache bank. The reservation channel is used to enable the ring filters at targeted destination L1 caches before an L2 cache bank transmits packets onto the L2-to-L1 data channels [16]. Each logical link in Fig. 1 represents one L1-to-L2 MWSR data channel (with corresponding token channel) and one L2-to-L1 SWMR data channel (with corresponding reservation channel). Each ring shown in Fig. 1 represents a set of 64 ring modulators and 64 ring filters for data channels (with few extra rings for

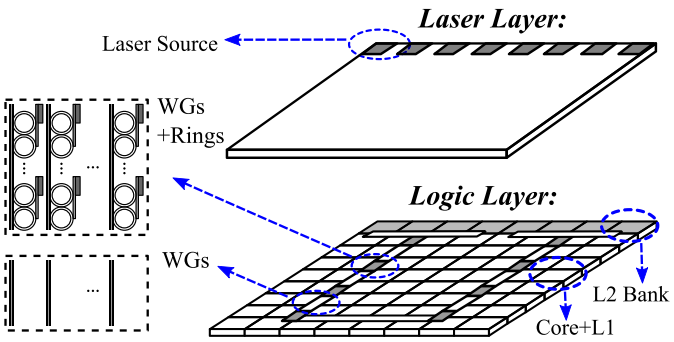


Fig. 2. Physical layout of the silicon-photonic crossbar NoC between 64 cores (each with a private L1 cache) and eight L2 cache banks. The L2 cache banks are located at the ends of the silicon-photonic links for ease of arbitration. The reconfiguration controller (RC) is placed in the center of the L2 caches and the laser sources are placed right above the RC and L2 caches to minimize the reconfiguration overhead.

token and reservations). This approach of using MWSR mechanism for L1-to-L2 communication and SWMR for L2-to-L1 communication where all the communication associated with an L2 cache bank is mapped onto dedicated silicon-photonic links provides the opportunity to easily switch OFF/ON these dedicated silicon-photonic links when the L2 cache bank is deactivated/activated.

Fig. 2 shows the physical layout of the silicon-photonic network between 64 private L1 caches and eight shared and distributed L2 cache banks. Cache lines are interleaved across L2 cache banks to enable the parallel accessibility. We use a modified exclusive shared invalid (MESI) directory-based protocol for maintaining cache coherency between L1 and L2 cache banks. The cache coherency directories are located next to the associated L2 cache banks. Any core-to-core communication is through the directory, hence no forward-type coherency messages are implemented. The L2 cache banks and memory controllers (MCs) are co-located on the edge of the processor; therefore L2 cache banks can access all the MCs through local wiring upon L2 cache misses. In the physical layout, the waveguides are routed in a U-shape layout. Instead of the conventional off-chip laser sources, we opt for on-chip laser sources because they can be quickly switched ON/OFF to correspondingly switch ON/OFF the associated silicon-photonic links (more details in Section IV). The laser sources are placed in a separate layer above the low power density L2 caches to minimize the impact of power dissipated in the logic layer on the laser source temperature. Here, the laser sources are placed along one edge of the chip but other arrangements are also feasible [22].

There is a separate off-chip photonic network that connects MCs to PIDRAM chips [23]. We assume an average time of 50 ns for the round-trip delay between the MCs and PIDRAMs. We ignore the variations in queuing latencies at the inputs of MCs because the high off-chip bandwidth using PIDRAM significantly reduces the number of outstanding memory requests in the queue.

IV. PHOTONIC TECHNOLOGY

Silicon-photonic devices can be interfaced with conventional CMOS devices using monolithic integration or 3-D

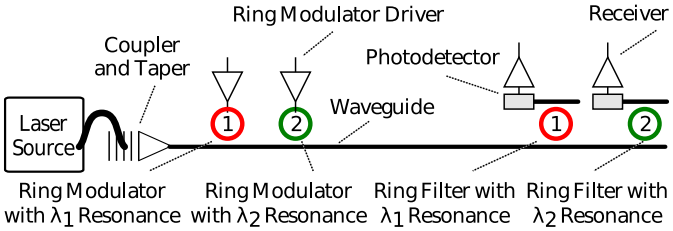


Fig. 3. Photonic link components. Two point-to-point photonic links implemented with WDM.

integration. Monolithic integration (using bulk CMOS process or SOI process) involves using the existing layers in the process technology to design photonic devices. On the other hand, 3-D integration involves integrating an SOI chip of a different process or depositing custom photonic layers (SiN or polycrystalline silicon) above the metal layers and interfacing the CMOS devices and photonic devices through vias. Our proposed technique for using cache and NoC reconfiguration for managing laser power is applicable to silicon-photonic NoC designed using both monolithic integration and 3-D integration. The percent of savings in laser power in the 3-D integration case would be same as the monolithic integration case. However, in case of 3-D integration the parasitics at the interface of the optical layer and logic layer are very high, and, in turn, the E-O-E conversion cost would be very high making it infeasible for silicon-photonic NoC. Hence, we focus on using monolithic integration and minimizing the laser power to make the photonic NoC more energy efficient.

Fig. 3 shows a generic wavelength-division multiplexed (WDM) photonic link used for intrachip communication. Light waves of wavelength λ_1 and λ_2 emitted by a laser source are coupled into the chip using vertical grating couplers. These couplers allow for vertical coupling into planar waveguides. These light waves pass next to a series of ring modulators controlled by modulator drivers based on the data to be transmitted on the link. The modulators convert data from the electrical domain to the photonic domain. The modulated light waves propagate along the waveguide and can pass through zero or more ring filters. At the receiver side, the ring filter "drops" the light wave having the filter's resonant wavelength onto a photodetector. The resulting photodetector current is sensed by an electrical receiver. At this stage, data is converted back into the electrical domain from the photonic domain.

For our analysis, we consider a silicon-photonic link design similar to that proposed in [4]. This link design uses double-ring filters with a 4 THz free-spectral range, which enables up to 128λ modulated at 10 Gb/s on each waveguide (64 λ in each direction, interleaved to alleviate filter roll-off requirements and crosstalk). A nonlinearity limit of 30 mW at 1 dB loss is assumed for the waveguides. In our target system, this limits the number of wavelengths per waveguide to 15 in our proposed layout. The waveguides are single mode and have a pitch of 4 μm to minimize the crosstalk between neighboring waveguides. Modulator ring and filter ring diameters are $\sim$10 μm. The latency of the photonic link is assumed to be 3 cycles (1 cycle in flight and 1 cycle each for electrical-to-optical and optical-to-electrical conversion) + serialization latency + latency due to contention for NoC resources. We use the measurement values of silicon-photonic links designed in 45 nm SoI [24]–[27] and project those values to 22 nm SoI technology (see Table II). As we scale CMOS technology from 45 to 22 nm, the optical losses in photonic devices and the thermal tuning power per K will not change as these values depend on the photonic device designs, which do not scale as we scale CMOS transistors from 45 to 22 nm. For thermal tuning power calculations, we assume the rings have to be tuned over a range of 10 K. We use fixed-voltage technology scaling rules to determine the energy consumed in modulator driver circuit (for E-to-O conversion) and receiver circuit (for O-to-E conversion).

TABLE II
PHOTONIC TECHNOLOGY PARAMETER VALUES BASED ON MEASUREMENTS AND PROJECTIONS

Laser source efficiency	5%
Coupler loss	1 dB
Splitter loss	0.2 dB
Modulator insertion loss	1 dB
Waveguide loss	3 dB/cm
Crossing loss	0.04 dB
Filter through loss	1e-2 dB
Filter drop loss	0.5 dB
Photodetector loss	0.1 dB
Non-linearity loss	1 dB
Modulator Driver circuit energy	0.035 pJ/b
Receiver circuit energy	0.11 pJ/b
Thermal tunning power	16 μW/K
Receiver sensitivity	-14

Over the past few years, several different on-chip laser source technologies have been proposed [28]–[30]. These technologies require further development, but they can simplify packaging and improve the energy-efficiency of photonic NoCs. We employ multiple on-chip single-mode indium phosphide (InP)-based laser sources for powering the waveguides. Integrated multiwavelength lasers [31] or comb lasers [32] could also be utilized in our target system. However, these alternatives have a large footprint or require further technological development. Therefore, we consider a set of single-mode laser sources in this paper. We use dedicated laser sources for each L2 to simplify the laser switch ON/OFF process. Each L2 bank requires 64 wavelengths each for L2-to-L1 communication and L1-to-L2 communication. The 30 mW nonlinearity limit restricts the number of wavelengths per waveguide to 15. Hence, a single waveguide is driven by 15 single band laser source with each source emitting about 2 mW of optical power. In our target system, 1920 single band laser sources (300×50 μm each) occupy approximately 28 mm^2 on the laser layer. The laser source efficiency varies with temperature and output optical power. For our (InP)-based laser sources the efficiency can be as large as 10% if we can maintain the laser temperature at 65 °C. In this paper, we use a projected laser source efficiency of 5%. A more detailed discussion of the laser source technology that we have used can be found in [22]. It should be noted though that our proposed approach for laser power management is also applicable to other on-chip laser source technologies.

In our target system, the laser sources are placed on the laser source layer on top of the logic layer. Compared with off-chip laser sources, the on-chip laser sources are located closer to the logic layer and have faster switch ON/OFF time. Since our power savings technique relies on switching ON/OFF the silicon-photonic links associated with activated/deactivated L2 cache banks, the on-chip laser ON/OFF characteristics are critical for maximizing the benefits of our proposed technique. When the on-chip laser supply current is turned ON, both the carrier density and photon density in the on-chip laser active medium reach a steady-state condition rapidly (on the order of ns [33]). The on-chip laser dissipates power in the form of heat when turned ON, therefore some temperature stabilization time is required. However, considering all of these effects, on-chip semiconductor diode lasers have demonstrated stable pulses with pulse widths (time to switch ON/OFF the laser current) of 35 ns [34]. This switch ON/OFF time is suitable for our approach as we reconfigure the L2 cache every ten million instruction (which is approximately 500 μs for our target set of benchmarks running on the target system described earlier). And even if more precision were required to ensure power and lasing wavelength stability, a simple wavelength locking circuit could be employed [35]. In this paper, we explore the application of our power management technique considering on-chip laser source with switch ON/OFF times of 100 ns.

V. Runtime Reconfiguration

In this section, we describe our runtime cache and NoC reconfiguration technique that can reduce laser power while sustaining performance. The key idea is to track the dynamic changes in the size of an application's working set during different phases of application execution and reconfigure the L2 cache bank count and NoC at runtime. Essentially, if the entire working set can fit into a smaller L2 cache, we can save laser power by deactivating some L2 cache banks and their associated silicon-photonic links. On the other hand, if the working set requires a larger L2 capacity, we need to activate more L2 cache banks and their corresponding silicon-photonic links to avoid excessive L2 cache misses and sustain application performance. It should be noted that we are not proposing a new L2 cache reconfiguration technique. Several techniques have already been proposed for L2 cache reconfiguration [20], [21], [36], [37]. However, the use of L2 cache reconfiguration to manage laser power in silicon-photonic NoC has never been explored before.

A. RC

We propose to use a RC that is responsible for tracking the variations in the application working set size and making the decisions on the required changes in the L2 cache bank count. As we will see, the decision on changing the L2 cache capacity relies on the L2 cache replacement rate that the RC must collect from the current set of L2 banks. The RC is located on the edge of the chip where the L2 cache banks and laser sources are also located (see Fig. 2). For each L2 bank, two single-bit point-to-point electrical channels are used: one to send data from the L2 bank to the RC and the other to send data from the RC to the L2 bank. The single-bit channel can transmit the cache replacement rate of the L2 bank (a 32-bit floating point value assuming the single-precision IEEE 754 standard) toward the RC in 32 clock cycles (very small compared to the large sampling period). The RC employs another pair of unidirectional single-bit point-to-point electrical links per laser source for communication. These electrical channels used for communication between RC and L2, and between RC and laser sources have minimal energy and area overhead due to their very low bandwidth and proximity to the RC.

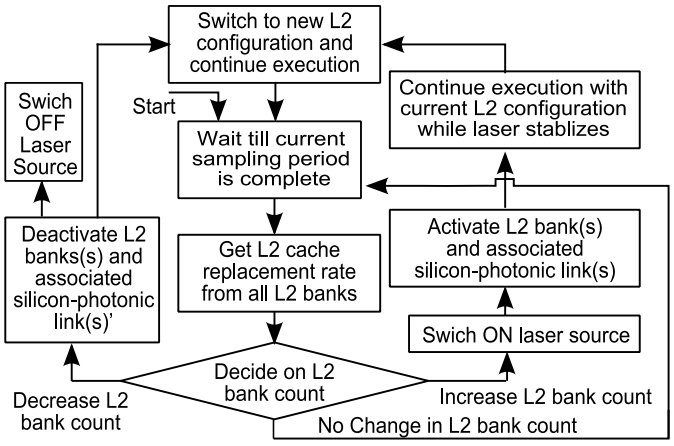


Fig. 4. Flowchart for activating/deactivating L2 cache banks and the associated silicon-photonic links at runtime.

Fig. 4 shows the flow chart of the reconfiguration process performed by the RC. After each sampling period, to collect the L2 cache replacement rate, the RC sends a 1-bit signal toward every active L2 cache bank utilizing each banks' associated electrical link. After one clock cycle, the L2 cache bank starts reporting back the L2 cache replacement rate. The RC collects all replacement rates in the next 32 clock cycles (32-bit floating point values transmitted through 1-bit electrical links), and decides whether to increase/decrease the number of L2 cache banks and the number of active silicon-photonic links. In our system, the RC increases/decreases the number of L2 cache banks in powers of 2 as it is shown in Fig. 5. Alternate granularities could be used in our approach for increasing/decreasing the L2 cache size and hence the laser power. However, that complicates the process of determining the home bank (HB) for each cache line. So for simplicity, here, we, change the L2 cache bank count in powers of 2.

When the RC decides to activate one or more L2 cache banks, it needs to switch ON the laser sources that power the silicon-photonic links associated with those L2 cache banks. For that, the RC sends 1-bit signals toward both the laser sources and the L2 cache banks (the logic that activates/deactivates an L2 bank is always active) utilizing the corresponding electrical links. Each laser source needs to stabilize before powering the silicon-photonic links and as discussed in Section IV, we expect that the entire switch ON time of associated laser sources (including the stabilization delay) can be less than 100 ns (125 clock cycles in our target processor frequency). The laser sources notify the completion of their stabilization by sending back a 1-bit signal toward the RC.

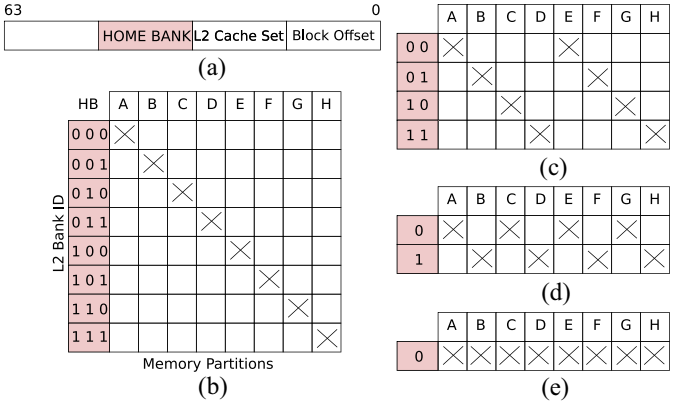


Fig. 5. Memory block mapping. (a) Format of memory block address. Bits in the HB field are used to determine the L2 cache bank to which the address should be mapped. (b)–(e) Matrices showing the mapping between the memory partitions and the active L2 cache banks. Crosses in the matrix's cells indicate the current mapping. From (b)–(e) L2 cache bank count decreases in power of 2. From (e)–(b) L2 cache bank count increases in power of 2.

Similarly, the new set of L2 cache banks send a 1-bit signal to the RC once they are powered up.

Once the L2 banks and associated silicon-photonic links are activated, the RC can start the process of mapping memory blocks to the new L2 bank count. To do that, the RC communicates the decision of activating one or more L2 cache banks to the current set of active L2 banks utilizing point-to-point electrical links. In particular, the RC transmits a short message (a 1-byte message) through the associated electrical links toward the current set of active L2 banks (the message takes eight clock cycles in flight). This message contains the number of bits that should be used to determine the new mapping of memory blocks [i.e., the size of the HB field (see Fig. 5)] to the new set of L2 banks (e.g., the RC sends a value 3 if the new L2 bank count is 8). Based on this value, the current set of active L2 banks are able to determine which cached blocks have to be flushed to memory (by comparing the new HB field against their cached memory blocks' tag field), as these blocks will need to be cached by the new set of L2 banks that are being activated during the reconfiguration process. We estimate that the process of flushing cached entries from L2 can take a large number of clock cycles (up to 18 000 cycles if all L2 bank' entries have to be flushed). As we will explain in Section V-C, during this long period, the application execution does not stop and L1 cache controllers can address the L1 cache misses in the higher levels of the memory hierarchy (L2 cache banks and main memory) without any memory consistency issues, thereby leading to marginal performance losses. Once the L2 banks have finished the flushing process, they send a 1-bit signal toward the RC to notify the completion of the reconfiguration process.

In the case when the RC decides to deactivate some L2 cache banks, it needs to keep the associated laser sources ON until those L2 cache banks flush their contents to the main memory and are completely deactivated. For that, the RC sends the same short message as before through the associated electrical links toward all current set of L2 banks. By reading this message that contains the size of the new HB field, the L2 banks will know which ones have to be deactivated (by comparing the size of the HB field against their L2 bank identifier—e.g., if the size of the HB field is 2 bits, so there will be four L2 banks after reconfiguration and all L2 banks with greater or equal L2 identifier must be deactivated). The banks that are going to be deactivated can start with the deactivation process by flushing all their memory blocks to memory. As we will detail in Section V-C, this process, like activating L2 banks, can be performed without compromising memory consistency and stopping the application execution. After deactivating those L2 cache banks, their associated L2 bank controllers send a 1-bit signal through their electrical links toward the RC, who can then start switching OFF the corresponding laser sources. After the laser source is switched OFF, its controller notifies the RC by sending a 1-bit signal through its associated electrical links.

It is worth noting that, the RC does not need to explicitly send the reconfiguration decision toward the L1 caches for them to know the new mapping of memory blocks to the L2 banks (i.e., the new size for the HB field). We propose a few simple modifications to the cache coherence protocol (detailed in Section V-C), which allow the current set of L2 banks to send the new HB field size as a response to L1 petitions using a negative acknowledgement (NACK) message.

B. Reconfiguration Decision Process

To make reconfiguration decisions, ideally the system should test all choices for L2 cache bank count and find the optimal L2 cache bank count that reduces power while maximizing IPC. However, this method of choosing L2 cache bank count is not feasible at runtime. Moreover, IPC cannot be used for making the reconfiguration decision because the variations in absolute value of IPC do not necessarily indicate the need for L2 cache reconfigurations. We propose to use L2 cache replacement rate (# cache replacements/# clock cycles in the sampling period) as a metric to determine the need for increasing/decreasing L2 cache bank count. The absolute value of L2 cache replacement rate indicates if the current L2 cache size is sufficient to store the application's entire working set and if an increase/decrease in L2 cache bank count can improve/hurt the system performance.

The RC uses a dual-threshold approach to make reconfiguration decisions. It compares the L2 cache replacement rate with two thresholds: T_{high} and T_{low}. When the replacement rate is higher than T_{high}, it decides to increase the L2 cache bank count, and when the replacement rate is lower than T_{low}, it decides to decrease the L2 cache bank count. When the replacement rate is between T_{low} and T_{high}, the RC makes a decision to maintain the current L2 cache bank count.

To determine the values for T_{high} and T_{low}, we evaluated the impact of various T_{high} and T_{low} values on system IPC and fluctuations in the L2 cache bank count, respectively (see Fig. 6). As mentioned earlier, when L2 cache replacement rate is greater than T_{high}, the system increases L2 cache bank count. Hence, a large value for T_{high} provides large savings in the laser power as the large T_{high} value reduces the probability of increasing L2 cache bank count and hence the probability of

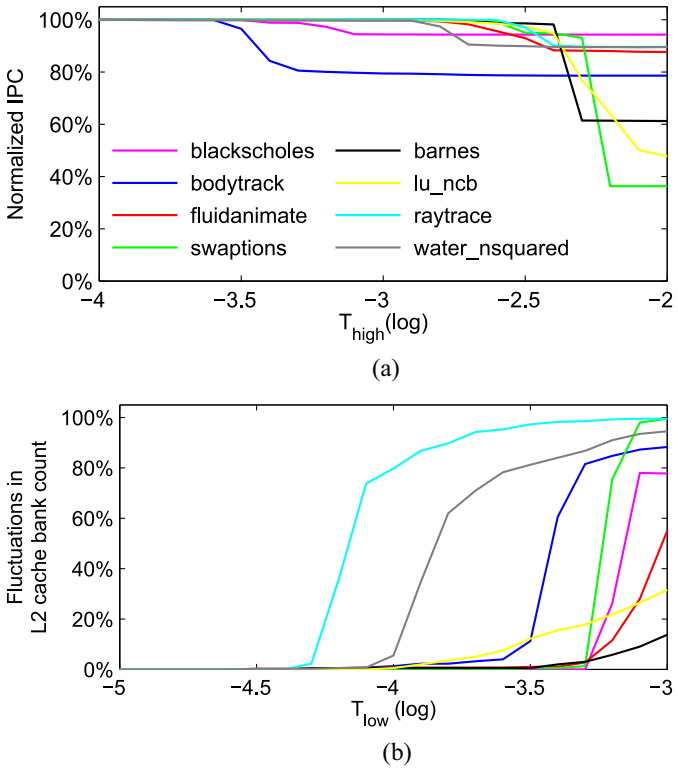


Fig. 6. Calculation of two thresholds. (a) IPC degradation when using different T_{high} values for each benchmark. (b) Number of fluctuations in L2 cache bank count for using various T_{low} for each benchmark when $T_{\text{high}} = 10^{-3}$.

switching ON the associated silicon-photonic links. However, a downside using a large T_{high} value is that it can cause performance degradation as the system may not have sufficient number of L2 cache banks to maintain the entire working set of the application on the processor chip. In Fig. 6(a), we show the normalized performance of our target benchmarks as we increase T_{high} value. We can see these benchmarks start showing degradation in performance for T_{high} values larger than $10^{-3.5}$. For the remaining analysis, we choose $T_{\text{high}} = 10^{-3}$ where the average performance degradation is $<10\%$ across all benchmarks.

The system decreases L2 cache bank count when L2 cache replacement rate becomes lower than T_{low}. A large value for T_{low} will lead to higher laser power savings as it would increase the probability of deactivating L2 cache banks and switching OFF the associated silicon-photonic links. However, a large T_{low} value could lead to an incorrect cache reconfiguration decision (an unnecessary deactivation of L2 cache banks that leads to loss of performance and increase in L2 replacement rate), which will need to be reversed at the end of the very next sampling period. This can happen especially when values of T_{low} and T_{high} are very close to each other. Assume that the system increases the L2 cache bank count after observing a L2 cache replacement rate higher than T_{high}. There is a high probability that the L2 cache replacement rate would drop below T_{low} and the system will need to decrease the L2 cache bank count in the next sampling period. This decrease in the L2 cache bank count would increase the value of the L2 cache replacement rate above T_{high}, which would again indicate the need for increasing the L2 cache bank count. Essentially, we might end up in a situation where we need to change the L2 cache bank count after every sampling period. Hence, we need to choose the value of T_{low} such that it can avoid the need for reconfiguring the L2 cache at the end of every sampling period.

Our simulation-based analysis revealed that each application has a different optimal T_{low} value [see Fig. 6(b)]. In the figure, the fluctuation is defined as the percentage of reconfiguration decisions that are reversed at the end of the very next sampling period. We can choose a T_{low} below $10^{-4.5}$ for all the benchmarks to minimize the fluctuations. However, such a low T_{low} would reduce the probability of the system reducing the L2 cache bank count to save laser power. We propose to use a simple learning process in the RC to find the best T_{low} for each benchmark. We choose $T_{\text{low}} = T_{\text{high}}$ when the application starts execution. Whenever RC observes a "fluctuation," it decreases the value of T_{low}. Over time, the RC converges to the optimal T_{low} for an application. The optimal T_{low} values for each application could potentially be stored and can be reused when the application is executed again.

C. Memory Consistency During the Reconfiguration Process

The manycore system needs to maintain cache coherence and consistency while activating/deactivating L2 cache banks. The reconfiguration process only affects the memory blocks that are associated with the L2 cache banks that are being activated/deactivated. Memory operations to other memory blocks can be performed during the reconfiguration. Moreover, we allow that L1 cache misses to memory blocks that are affected by the reconfiguration process can be also resolved without any memory inconsistency, while minimizing any performance degradation in the system. We explain how this process is performed assuming a cache hierarchy in which the L1 and L2 caches are inclusive, and use a write-back writing policy and a directory-based coherence protocol. Our mechanism can easily be extended to other cache hierarchies as it is described in Section VII.

Fig. 7(a) shows an example in which there are 4 L1 caches (*L1$-a–L1$-d*) and 2 active L2 cache banks (*L2$-a* and *L2$-b*), and here *L2$-b* needs to be deactivated. Since the cache hierarchy is inclusive, the L1 caches store a subset of all memory blocks in L2 caches. In this way, all L2 memory blocks in *L2$-b* must be checked and flushed along with the copies stored at L1 level. To implement the flush operations, we use the same process that is required to flush L2 memory blocks when an L2 replacement is needed or when a memory invalidation message is received from main memory. Essentially, coherence-related invalidation messages are sent from L2 cache banks to the corresponding L1 caches [see 1.Inv in Fig. 7(a)]. Since, the caches use write-back policy, memory blocks at L1 level can hold the latest copy of the memory block [dirty blocks—see owner(L1$-c) and the corresponding block at L1$-c]. Hence, the flush message results in a write-back data transfer from L1 to L2 cache (for clarity, it is not shown in the figure). Cache blocks that are clean do not require a write-back data transfer but an acknowledgment message to L2 is required that confirms the invalidation of the memory block at L1 level (for clarity,

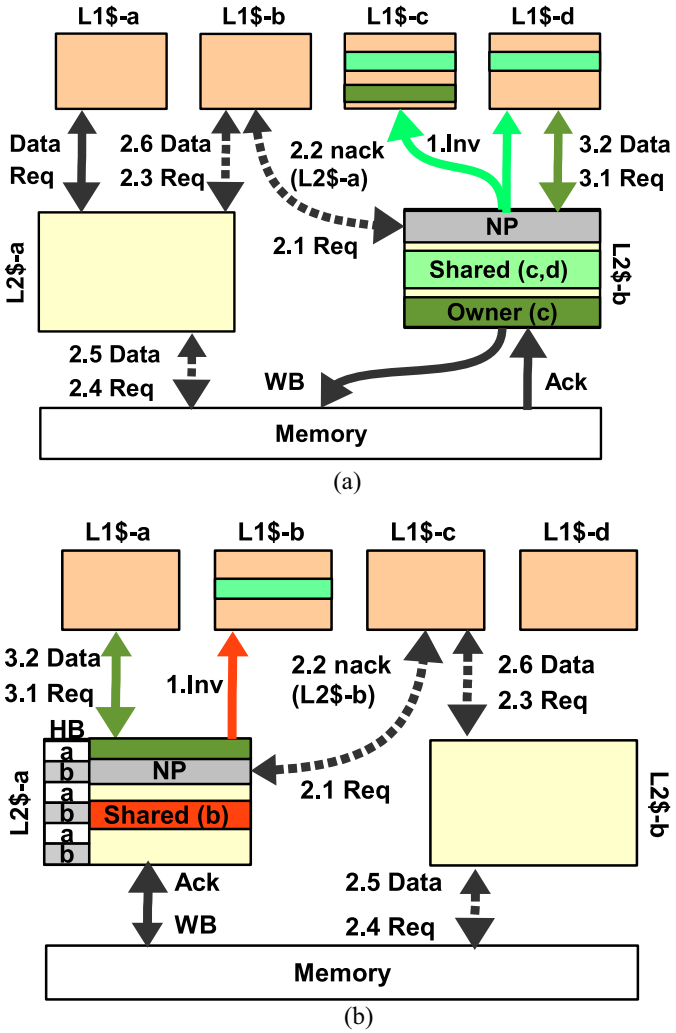


Fig. 7. Memory consistency in the reconfiguration process. (a) Deactivation of *L2$-b*. 1.Inv: message sent to all L1 caches that have a copy of the memory block to be flushed. 2.1–2.2 messages: L1 request using the old HB mapping for a memory block that is NP. The old home L2 bank returns NACK indicating the petitions should go to L2 bank (*L2$-a*). 2.3–2.6 messages: the previous L1 request is forwarded to the correct L2 bank that fetches the data from memory and sends it back to the L1 petitioner. 3.1–3.2 messages: L1 petition for a memory block that has not been flushed yet by the old HB. This request can be processed as it has not been flushed. (b) Activation of *L2$-b*. 1.Inv: message sent to the L1 cache that has a copy of the memory block to be flushed. 2.1–2.6 messages: same steps as described for the deactivation process for a memory block that is NP in the old HB (*L2$-a*), and the L1 petition is forwarded to the new L2 bank. 3.1–3.2 messages: same message as those described for deactivation process.

it is not shown in the figure). It is worth noting that these actions are exactly the same as when conventional invalidation messages are employed. Hence, L1 cache controllers are not modified to deal with our flush messages. Once a particular L2 memory block has been flushed from L1 caches (either a write-back data transfer from the L1 with the dirty copy of the block, or a set of acknowledgments from each clean copy in L1 caches have been received at the L2 cache bank), the L2 cache bank will either transmit a write-back message to memory [WB in Fig. 7(a)] in case of dirty blocks, or it will simply remove the associated L2 cache bank entry in case of a clean block. Then, the coherence status of these L2 cache entries change to not-present (NP).

To avoid performance loss during the flushing of memory blocks from the L2 banks that are being deactivated during the reconfiguration process, we allow their corresponding L2 bank controllers to continue processing L1 petitions as follows: all L1 petitions that go to memory blocks that are stored in the L2 bank (they are not in NP state) are processed using the MESI protocol [see 3.1 Req and 3.2 Data in Fig. 7(a)]. However, L1 petitions to memory blocks that are not cached in the L2 bank because either they have been flushed during the reconfiguration process, or they have never been referenced, or they have been recently replaced (they are in NP state—see 2.1 Req) are not forwarded toward main memory to get the requested memory block. In this case, and only during the reconfiguration process, the L2 bank controller returns a NACK message for each of these L1 petitions [2.2 NACK in Fig. 7(a) for the case where L2$-b is being deactivated]. The NACK message contains the number of bits that should be used by the L1 controller to determine the HB field from the memory block address (see Fig. 5). In that way, the L1 controllers know the new L2 bank that will be responsible for these L1 petitions and send petitions to it (see messages from 2.3 to 2.6). In short, all L1 petitions are served during the reconfiguration process. Note that, our proposal requires minimal modifications to the L2 cache controller to deal with the NP state during the reconfiguration process.

Fig. 7(b) shows the L2 cache bank activation process. In this example, there are four L1 caches (*L1$-a–L1$-d*) and one active L2 cache bank (*L2$-a*), and *L2$-b* bank needs to be activated. Note that, as we mentioned in Section V-A, the first step of the reconfiguration process is that the RC sends the new size of the HB field to the current set of L2 banks. Second, the RC starts the activation of the new L2 banks and the corresponding laser sources and silicon-photonic links. Once the new set of L2 banks along with the associated silicon-photonic links are powered up the RC begins the new mapping of memory blocks to the new L2 bank configuration.

To apply the new mapping of memory blocks to the L2 banks, we need to first identify the L2 cache banks that are currently active and are serving the L1 cache requests (prior to reconfiguration) that would be served by *L2$-b* after reconfiguration. We then need to flush these relevant memory blocks from these L2 cache banks back to memory. To accomplish that, the current L2 cache banks use the new size of the HB field (which they know by this stage). They compare the new HB field against their cached memory blocks' tag field, and determine which cached blocks to flush. The process of flushing L2 memory blocks would be similar to that explained in the L2 cache bank deactivation process [1. Inv, WB and Ack in Fig. 7(a)].

To minimize performance degradation, we allow the application execution to continue during the reconfiguration process while maintaining memory consistency. First, all L1 requests that go to memory blocks that are not affected by the reconfiguration process (cached blocks with correct HB bits) can be processed as in the MESI protocol [2.1 and 2.2 in Fig. 7(b)]. Second, the remaining set of L1 petitions that correspond to the memory blocks who will belong to the newly activated L2 banks will go first to the current home L2 bank

(the L1 petitioners do not know the new size for the HB field). These L2 banks send NACK messages that contain the size of the new HB field to the L1 petitioners [2.2 NACK in Fig. 7(b)]. To keep memory consistency, the NACK message will only be sent from the old L2 banks to the L1 petitioners after the requested memory block has been completely flushed from the old L2 bank (it gets an NP state in the old L2 bank). In this way, our mechanism avoids stopping the system during the reconfiguration process [see messages from 2.3 to 2.6 in Fig. 7(b)] and guarantees that there will be a unique copy of the memory block cached at L2 level, thereby avoiding the memory inconsistency.

Note that the RC does not need to communicate the reconfiguration decision to L1 caches, as the active L2 banks will send this information to the L1 caches through the NACK messages explained above. For instance, if an L1 cache does not know the new set of L2 active banks, and sends it petitions to an incorrect L2 bank, the L2 bank will return the NACK message that stores the number of bits required to identify the correct mapping of memory blocks to L2 banks (the new HB field). Moreover, if the L1 cache controller tries to send a message to a previously deactivated L2 bank, it will not be able to establish the communication with this L2 bank because the corresponding silicon-photonic channel is deactivated. So, the L1 cache controller will then update the size of the HB field by removing one bit to the HB field (e.g., from 3 bits that identifies 8 L2 banks to 2 bits that refers to 4 L2 banks) and try to send a message to an L2 bank based on the new mapping. The L1 cache controller will repeat the previous process until it gets an active silicon-photonic channel (corresponding to an active L2 bank). If after sending the L1 petition, this L1 petition reaches an incorrect L2 bank, that L2 bank will return a NACK message (containing the number of bits to be used for determining the new mapping) to the L1 petitioner so that the L1 petition will then send the petition to the correct L2 bank. The modifications to both the L1 and L2 cache controllers to deal with this new scenario (not implemented in a regular MESI protocol) results in minimal overhead compared to the larger laser power savings.

VI. Evaluation

In this section, we quantify the power savings obtained by using our proposed cache and NoC reconfiguration technique.

A. Simulation Methodology

We use the Gem5 full-system simulator [10] to evaluate our proposed idea of using L2 cache and NoC reconfiguration for laser power savings on the 64-core target system described in Section III. We enable the Ruby memory system for an accurate modeling of the shared multibank L2 cache hierarchy. We run PARSEC [11] and SPLASH-2 [12] benchmarks in their parallel regions with the large input set, and check for the need for L2 cache reconfiguration after every ten million instructions (committed by all 64 cores), which is approximately 500 μs for our target system. Each benchmark executes a total of four billion instructions resulting in 400 reconfiguration sampling periods. After every sampling period, we check the replacement rate using performance counters to analyze the need for L2 cache reconfiguration.

We use McPAT [38] and Cacti 5.3 [39] to calculate the core power and cache power, respectively. The McPAT output is calibrated using the Intel SCC [1] published power values and is then scaled to 22 nm technology [40]. We use the photonic technology described in Section IV to calculate laser power, Tx/Rx power and thermal tuning power in the NoC. We calculate EDP [total system power * (application execution cycles/system frequency)^2] of the entire system to evaluate the overall impact of our proposed technique on the balance of system performance and power.

B. Reconfiguration Opportunities

Fig. 8(a) and (b) shows the IPC trace and L2 cache replacement rate trace, respectively, for a 64-core system with different number of L2 cache banks when running `blackscholes`, `bodytrack`, `fluidanimate`, `swaptions`, `barnes`, `lu_ncb`, `raytrace`, and `water_nsquared` benchmarks. The replacement rate can be used as a metric for a fairly accurate prediction of whether we need to increase or decrease the number of L2 cache banks. A high L2 cache replacement rate generally indicates the need for a larger cache capacity, while a low L2 cache replacement rate indicates that cache capacity is larger than what is required. Fig. 8(c) compares the trace of the ideal L2 cache bank count determined using offline analysis and the trace of the L2 cache bank count chosen by the RC at runtime. The offline analysis chooses the minimal L2 cache bank count required to maximize performance after each sampling period. RC determines the L2 cache bank count based on the L2 cache replacement rate during the sampling period and compares it with two thresholds T_{high} and T_{low} to determine the L2 cache bank count for the next period. Fig. 8(c) shows that the optimal L2 cache bank count varies both across applications and within applications over time and the L2 cache bank count chosen by the RC closely matches with the L2 cache bank count chosen by the offline analysis. Our proposed reconfiguration policy ensures that the L2 cache bank count tracks the changes in L2 cache replacement rate, and therefore harnesses any opportunity of saving laser power by reducing the L2 cache bank count. The optimal L2 cache bank count for `blackscholes` is lower than that for `barnes`. This indicates more savings in laser power when running `blackscholes`. In case of the `barnes` application, the optimal L2 cache bank count varies as the application goes through different execution phases, thus providing various levels of laser power savings.

C. Reconfiguration Benefits

Fig. 9 shows the impact of reconfiguration on system performance and power dissipation (using conservative silicon-photonic link design). Here, we compare the performance and power dissipation of a 64-core system that uses a fixed number of 1, 2, 4, and 8 banks with a 64-core system that uses our proposed cache and NoC reconfiguration policy. We did not consider cases with 3, 5, 6, and 7 active L2 cache

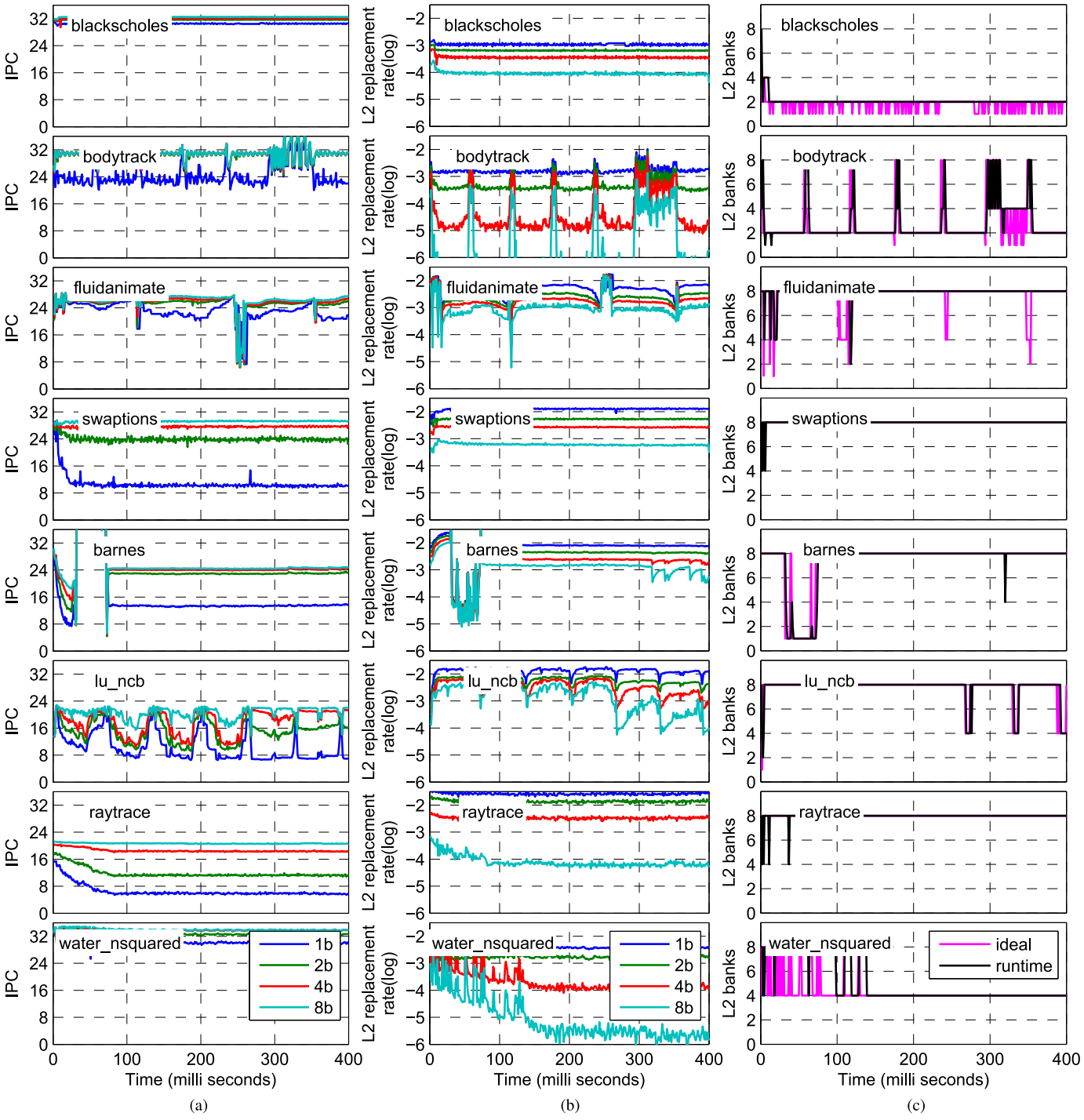


Fig. 8. IPC, L2 replacement rate, and L2 cache bank count tracing for selected benchmarks. (a) IPC of each benchmark with a fixed L2 cache bank count: 1, 2, 4, and 8 banks. (b) Measured L2 replacement rate with a fixed L2 cache bank count. (c) Optimal L2 cache bank count determined offline and the L2 cache bank count chosen by the RC at runtime.

banks as the mapping of the cache address becomes complicated for these bank counts and the overhead of L2 cache reconfiguration process increases. For this comparison, we assume the waveguide loss is 3 dB/cm. By deactivating redundant L2 cache banks and their associated silicon-photonic links, on average the runtime reconfiguration saves laser power by 23.8% (peak value 74.3%), cache power by 22.7% (peak value 72.9%), and system power by 9.9% (peak value 30.6%), compared to the case where all 8 L2 cache banks are ON all the time while having an average IPC degradation of 0.65% (peak value 2.6%). It should be noted that for the `fluidanimate`, `swaptions`, `barnes`, `lu_ncb`, and `raytrace` benchmarks, the laser power when using reconfiguration is comparable to the case when using all eight banks because these benchmarks require all eight banks to be active to maximize performance (seen from Fig. 8).

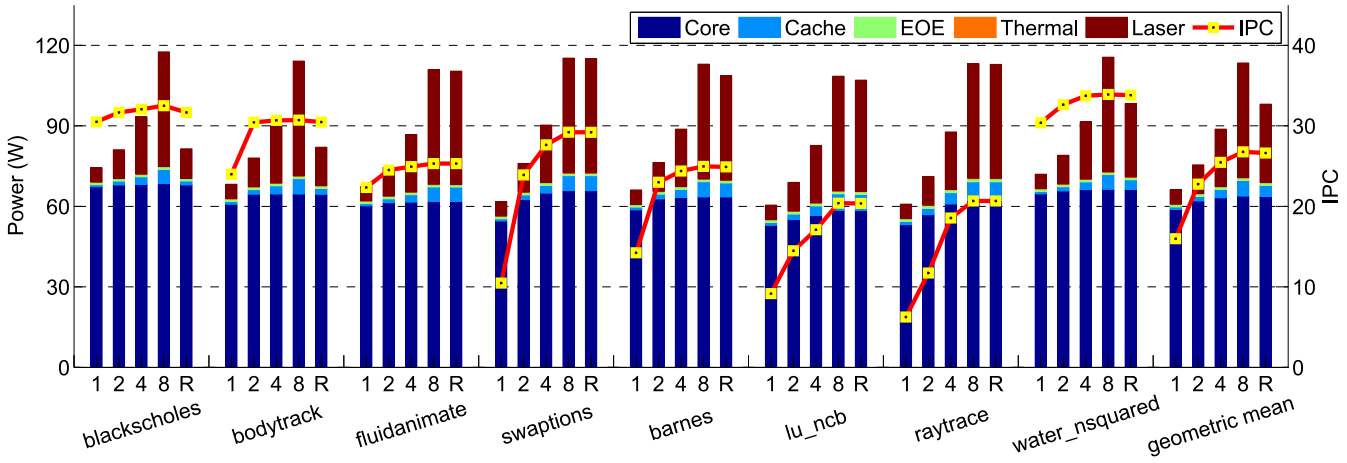


Fig. 9. Impact of runtime reconfiguration on system performance and power. We compare a system with runtime reconfiguration to systems with fixed number of L2 cache banks. Here, "1," "2," "4," and "8" represents the simulations with fixed 1, 2, 4, and 8 banks, respectively and "R" represents the simulations with runtime reconfigurable L2 cache bank count. We show the power breakdown and IPC of 8 benchmarks and their geometric means.

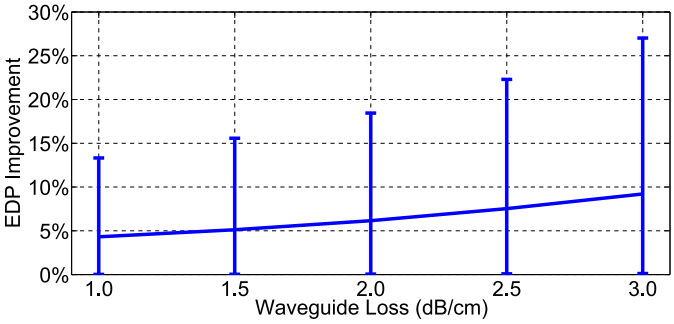


Fig. 10. Impact of waveguide loss on the system EDP improvement (averaged across all benchmarks). The waveguide loss varies in a range of 1–3 dB/cm, while fixing the losses in the remaining optical device losses to the values shown in Table II.

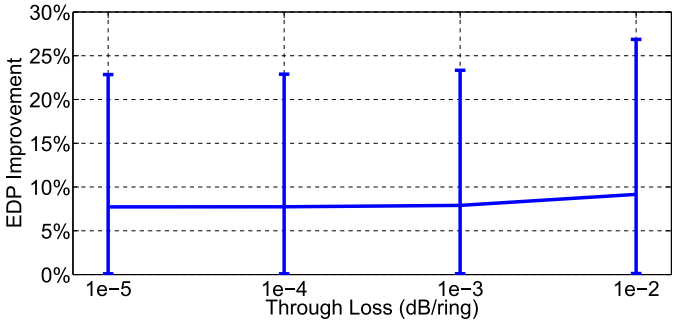


Fig. 11. Impact of ring through loss on the system EDP improvement (averaged across all benchmarks). The ring through loss varies in a range of 1e-5–1e-2 dB/ring, while fixing the losses in the remaining optical device losses to the values shown in Table II.

Fig. 10 shows EDP improvement after applying the runtime reconfiguration. The baseline used for comparison is a system with all eight L2 cache banks active all the time. The waveguide loss in the future silicon-photonic links is projected to be in a range of 1–3 dB/cm and is expected to be the dominant loss component. Here, the ring through loss is fixed at 1e-2 dB/ring. If we assume a conservative waveguide loss of 3 dB/cm, the runtime reconfiguration reduces entire system power dissipation by 9.9% (peak value 30.6%) and improves the entire system EDP by 9.2% on average (peak value 26.9%). If we assume a more aggressive waveguide loss of 1 dB/cm, the reconfiguration saves system power by 5.2% (peak value 16.0%) and improves system EDP by 4.3% on average (peak value 13.2%).

Similarly, Fig. 11 shows EDP improvement versus the ring through loss. The through loss in the future silicon-photonic links is projected in a range of 1e-5–1e-2 dB/ring. Here, the waveguide loss is fixed at 3 dB/cm. If we assume a conservative ring through loss of 1e-2 dB/ring, the runtime reconfiguration reduces entire system power dissipation by 9.9% (peak value 30.6%) and improves the entire system EDP by 9.2% on average (peak value 26.9%). If we assume a more aggressive ring through loss of 1e-5 dB/ring, the reconfiguration saves system power by 8.6% (peak value 26.4%) and improves system EDP by 7.8% on average (peak value 23%). The reduction of ring through loss has a smaller impact on the system EDP improvement than the reduction of waveguide loss because the ring through loss contributes a smaller amount of losses among all the optical loss components along the silicon-photonic waveguide.

Fig. 12 shows the overhead of the reconfiguration process. The reconfiguration process of L2 cache banks requires several cycles for flushing memory blocks from L2 cache banks and L1 cache (through L2) to the main memory, and fetching these memory blocks back to the appropriate active L2 cache banks. We measured the average reconfiguration overhead for each benchmark. In the worst case (when half the L2 cache blocks are flushed), a maximum of 18 000 cycles are spent in a reconfiguration that accounts for less than 1.782% of the execution time of one sampling interval (in the worst case). Our simulations also show that a maximum of ten reconfigurations were required over the 400 sampling intervals. The total time spent in reconfiguration is less than 0.045% of the entire execution time of a benchmark. Correspondingly, the energy overhead of the RC is minimal and gets amortized across the entire execution time of the benchmark. The

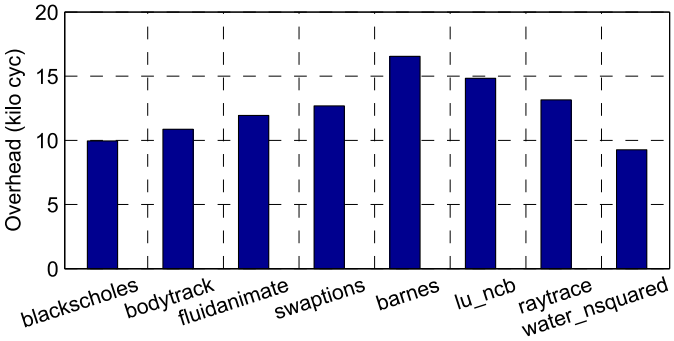


Fig. 12. Average overhead (in terms of number of cycles) per reconfiguration for various benchmarks. The overhead includes the clock cycles spent in flushing memory blocks during reconfiguration, notifying updated configuration.

average execution time of a benchmark is 12 ms, and after each reconfiguration we had an average of 2893 additional DRAM accesses for flushing and refetching 512-bit cache blocks. As a DRAM access costs less than 10 pJ/bit [41], a maximum of ten reconfigurations for each benchmark costs about 150 μJ (10 reconfiguration $\times$ 2893 access/reconfiguration $\times$ 512 bit/access $\times$ 10 pJ/bit), over the 12 ms execution time. It should be noted that a benchmark does not stop execution while the system is being reconfigured. It continues to execute using the older L2 configuration. It switches to the new L2 configuration once the L2 cache banks and the associated silicon-photonic links are activated/deactivated. The RC can be implemented as a combinational logic and requires a single-bit wire for communicating to/from L2 cache banks. Compared to the overall chip area, the area of the RC is minimal.

VII. Application of Reconfiguration Policy to Alternate Manycore Architectures

The proposed methodology for managing laser power through cache reconfiguration was specifically designed assuming an inclusive cache hierarchy with two levels of write-back caches (a private L1 caches and a logically-shared physically-distributed L2 cache), and a sophisticated implementation of a silicon-photonic crossbar. In this section, we discuss how our proposal would operate with other configurations of the cache hierarchy, alternate silicon-photonic NoC architectures, and large core counts.

A. Alternate Cache Configurations

Our reconfiguration system changes on-chip cache capacity based on application demand at runtime. In general, this process can be applied to any cache hierarchy or cache coherence protocol, as long as every reconfiguration step leaves the memory system in a consistent state. Here, we discuss the application of our cache reconfiguration policy to an exclusive cache hierarchy and a write-through writing policy.

An inclusive hierarchy is commonly present in almost all current processor architectures as it eases the implementation of coherency protocols. However, a cache design that enforces inclusion is inherently wasteful of cache space and network bandwidth. The reason for this is that every cache line in lower levels is duplicated in the higher levels, and updates in lower levels trigger many more updates in higher levels, wasting network bandwidth. This negative effect on our reconfiguration process is discussed in Section V-C, where the deactivation of a L2 cache bank requires parsing through all its bank entries to invalidate or flush all L1 cache entries that are copies of the entries in the L2 cache to be deactivated, hence wasting network bandwidth and energy. If we relax the inclusivity property and consider a mutual exclusive hierarchy, the cache implementation would be more complex from memory hierarchy standpoint (cores can communicate directly to the L2 cache banks, and block swapping between the two levels of caches must be made to guarantee exclusivity), but our reconfiguration process would be simpler and more efficient. Since, the memory blocks of the chosen L2 cache bank are NP in other L1 caches, we can deactivate the L2 cache bank without the need for flushing blocks from L1 caches, hence saving network bandwidth. During the process of deactivation, all requests of block swapping and requests from cores to that L2 cache bank would have to be denied using NACK messages. Similar to an inclusive hierarchy, the process to activate a new L2 cache bank would imply checking entries in L2 cache banks, and flushing all the entries that will belong to the new L2 cache bank back to memory.

Another important consideration is the writing policy implemented in caches. It is clear that the use of a write-back policy is efficient from the performance and network bandwidth perspectives, because when a processor core writes a memory block, it is simply marked as "dirty" and future accesses to it do not result in a cache miss. Upon a replacement or coherency-related invalidation, that block is evicted and its content is written back to a higher level of cache or memory. Due to the efficiency of this policy, our reconfiguration system has been designed with this kind of policy. Nonetheless, if we decide to use a write-through policy for L1 caches and L2 cache banks, the design of our reconfiguration process would be more efficient. The reason is that since main memory always holds the most recent value written to every memory block, deactivating an L2 cache bank can be performed without the need for flushing any data from L1 caches. However, in an inclusive hierarchy with write-through caches, future references to blocks in L1 caches belonging to the deactivated L2 cache could still have a hit, violating the inclusivity property. Therefore, the reconfiguration process would also need to invalidate such L1s' blocks consuming extra network bandwidth.

B. Alternate NoC Architectures and Large Core Counts

For evaluation of our reconfiguration policy, we considered a 64-core target system consisting of a silicon-photonic crossbar topology. However, our reconfiguration policy can be easily extended to other NoC topologies. The scope of application of our reconfiguration policy is really only limited by the physical implementation of the NoC. If the NoC employed to interconnect the L1 caches and L2 cache banks uses dedicated point-to-point silicon links, then the deactivation of a L2 cache bank can enable the deactivation of the

corresponding point-to-point silicon-photonic links. However, if the NoC is designed using shared silicon-photonic links, these links cannot be deactivated until all components attached to them have been also deactivated. Given that a logical topology can be mapped to any physical topology (subject to power/performance constraints) [42], in principle our reconfiguration policy can be applied to any logical topology.

Future manycore systems are expected to have thousands of cores on a chip. These thousand-core systems will still have the conventional L1–L2 cache hierarchy and are expected to support data-center style multiprogrammed workloads. The variations in the L2 cache size requirements across the different applications presents an opportunity to apply our cache reconfiguration policy. In addition, these thousand-core systems are expected to use the idea of "dark silicon," where only a fraction of the cores are active at a given point of time to manage power dissipation and to slow down aging. This approach of selectively using cores, and hence selectively activating L1/L2 caches also provides an opportunity to apply our proposed policy to reduce laser power.

VIII. Conclusion

The large laser power dissipated in the silicon-photonic NoC is limiting its widespread adoption in the design of manycore systems. We propose a runtime cache and NoC reconfiguration policy, where we activate/deactivate L2 cache banks depending on the spatial and temporal variations in the application behavior, and then switch ON/OFF the silicon-photonic links associated with these L2 cache banks to dynamically manage the laser power. The key idea is that for a given application at any given point of time, we operate the manycore system using the minimum number of L2 cache banks and silicon-photonic links required to achieve maximum application performance. Our policy is scalable to large core counts and is applicable to alternate cache and NoC architectures. On a 64-core target system with a silicon-photonic crossbar NoC, on average our proposed technique reduces laser power and system power by 23.8% (peak value 74.3%) and 9.9% (peak value 30.6%), respectively, and improves EDP by 9.2% (peak value 26.9%). Our proposed laser management methodology can potentially expedite the process of widespread adoption of silicon-photonic link technology in manycore processors.

Acknowledgment

The authors would like to thank J. Klamkin and M. Popovic for helpful discussions on photonic device designs.

References

[1] J. Howard *et al.*, "A 48-core IA-32 message-passing processor with DVFS in 45nm CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, San Francisco, CA, USA, 2010, pp. 108–109.

[2] S. Bell *et al.*, "Tile64-processor: A 64-core SoC with mesh interconnect," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers (ISSCC)*, San Francisco, CA, USA, 2008, pp. 88–598.

[3] S. Vangal *et al.*, "An 80-tile sub-100-W teraFLOPS processor in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, Jan. 2008.

[4] A. Joshi *et al.*, "Silicon-photonic clos networks for global on-chip communication," in *Proc. 3rd ACM/IEEE Int. Symp. Netw. Chip (NoCS)*, San Diego, CA, USA, 2009, pp. 124–133.

[5] D. Vantrease *et al.*, "Corona: System implications of emerging nanophotonic technology," in *Proc. 35th Int. Symp. Comput. Archit. (ISCA)*, Beijing, China, 2008, pp. 153–164.

[6] Y. Pan *et al.*, "Firefly: Illuminating future network-on-chip with nanophotonics," in *Proc. 36th Annu. Int. Symp. Comput. Archit. (ISCA)*, Austin, TX, USA, 2009, pp. 429–440.

[7] M. J. Cianchetti, J. C. Kerekes, and D. H. Albonesi, "Phastlane: A rapid transit optical routing network," in *Proc. 36th Annu. Int. Symp. Comput. Archit. (ISCA)*, Austin, TX, USA, 2009, pp. 441–450.

[8] H. Wassel *et al.*, "Opportunities and challenges of using plasmonic components in nanophotonic architectures," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 2, no. 2, pp. 154–168, Jun. 2012.

[9] Y. Pan, J. Kim, and G. Memik, "Featherweight: Low-cost optical arbitration with QoS support," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Porto Alegre, Brazil, 2011, pp. 105–116.

[10] N. Binkert *et al.*, "The M5 Simulator: Modeling networked systems," *IEEE Micro*, vol. 26, no. 4, pp. 52–60, Jul./Aug. 2006.

[11] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite: Characterization and architectural implications," in *Proc. 17th Int. Conf. Parallel Archit. Compil. Tech. (PACT)*, Toronto, ON, Canada, 2008, pp. 72–81.

[12] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proc. 22nd Annu. Int. Symp. Comput. Archit.*, Liguria, Italy, 1995, pp. 24–36.

[13] A. Shacham, K. Bergman, and L. P. Carloni, "On the design of a photonic network-on-chip," in *Proc. 1st Int. Symp. Netw. Chip (NOCS)*, Washington, DC, USA, 2007, pp. 53–64.

[14] H. Gu, J. Xu, and W. Zhang, "A low-power fat tree-based optical network-on-chip for multiprocessor system-on-chip," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, Nice, France, 2009, pp. 3–8.

[15] N. Kirman *et al.*, "Leveraging optical technology in future bus-based chip multiprocessors," in *Proc. 39th Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Orlando, FL, USA, 2006, pp. 492–503.

[16] Y. Pan, J. Kim, and G. Memik, "FlexiShare: Channel sharing for an energy-efficient nanophotonic crossbar," in *Proc. IEEE 16th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Bangalore, India, 2010, pp. 1–12.

[17] L. Zhou and A. Kodi, "PROBE: Prediction-based optical bandwidth scaling for energy-efficient NoCs," in *Proc. 7th IEEE/ACM Int. Symp. Netw. Chip (NoCS)*, Tempe, AZ, USA, 2013, pp. 1–8.

[18] C. Li, M. Browning, P. V. Gratz, and S. Palermo, "LumiNOC: A power-efficient, high-performance, photonic network-on-chip for future parallel architectures," in *Proc. 21st Int. Conf. Parallel Archit. Compil. Tech. (PACT)*, Minneapolis, MN, USA, 2012, pp. 421–422.

[19] C. Chen and A. Joshi, "Runtime management of laser power in silicon-photonic multibus NoC architecture," *IEEE J. Sel. Topics Quantum Electron.*, vol. 19, no. 2, Mar. 2013, Art. ID 3700713.

[20] J. Sim, J. Lee, M. Qureshi, and H. Kim, "FLEXclusion: Balancing cache capacity and on-chip bandwidth via flexible exclusion," in *Proc. 39th Annu. Int. Symp. Comput. Archit. (ISCA)*, Washington, DC, USA, 2012, pp. 321–332.

[21] C.-J. Wu *et al.*, "SHiP: Signature-based hit predictor for high performance caching," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchit.*, Porto Alegre, Brazil, 2011, pp. 430–441.

[22] C. Chen *et al.*, "Sharing and placement of on-chip laser sources in silicon-photonic NoCs," in *Proc. 8th ACM/IEEE Int. Symp. Netw. Chip (NoCS)*, Ferrara, Italy, 2014, pp. 1–8.

[23] S. Beamer *et al.*, "Re-architecting DRAM memory systems with monolithically integrated silicon photonics," in *Proc. 37th Annu. Int. Symp. Comput. Archit. (ISCA)*, Saint-Malo, France, 2010, pp. 129–140.

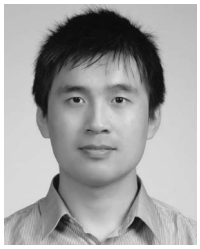
[24] J. S. Orcutt *et al.*, "Open foundry platform for high-performance electronic-photonic integration," *Opt. Exp.*, vol. 20, no. 11, pp. 12222–12232, May 2012.

[25] M. Georgas *et al.*, "A monolithically-integrated optical transmitter and receiver in a zero-change 45nm SOI process," in *Proc. Symp. VLSI Circuits Dig. Tech. Papers*, Honolulu, HI, USA, 2014, pp. 1–2.

[26] Y. Liu, J. M. Shainline, X. Zeng, and M. A. Popović, "Ultra-low-loss CMOS-compatible waveguide crossing arrays based on multimode Bloch waves and imaginary coupling," *Opt. Lett.*, vol. 39, no. 2, pp. 335–338, Jan. 2014.

[27] X. Zheng *et al.*, "2-pj/bit (on-chip) 10-Gb/s digital CMOS silicon photonic link," *IEEE Photon. Technol. Lett.*, vol. 24, no. 14, pp. 1260–1262, Jul. 15, 2012.

[28] R. E. Camacho-Aguilera *et al.*, "An electrically pumped germanium laser," *Opt. Exp.*, vol. 20, no. 10, pp. 11316–11320, May 2012.
[29] L. Liu *et al.*, "A thermally tunable III-V compound semiconductor microdisk laser integrated on silicon-on-insulator circuits," *IEEE Photon. Technol. Lett.*, vol. 22, no. 17, pp. 1270–1272, Sep. 1, 2010.
[30] T. Wang, H. Liu, A. Lee, F. Pozzi, and A. Seeds, "1.3-μm InAs/GaAs quantum-dot lasers monolithically grown on Si substrates," *Opt. Exp.*, vol. 19, no. 12, pp. 11381–11386, Jun. 2011.
[31] K. Lawniczuk *et al.*, "8-channel AWG-based multiwavelength laser fabricated in a multi-project wafer run," in *Proc. 23rd Int. Conf. Indium Phosphide Related Mater. Compound Semiconduct. Week (CSW/IPRM)*, Berlin, Germany, 2011, pp. 1–4.
[32] D. Livshits *et al.*, "Cost-effective WDM optical interconnects enabled by quantum dot comb lasers," *Proc. SPIE*, vol. 7607, Feb. 2010, Art. ID 76070W.
[33] L. A. Coldren, S. W. Corzine, and M. L. Mashanovitch, "Micronwave and optical engineering," in *Diode Laser and Photonic Integrated Circuits*, 2nd ed. Hoboken, NJ, USA: Wiley, 2012.
[34] J. Klamkin *et al.*, "Directly modulated narrowband slab-coupled optical waveguide laser," *Electron. Lett.*, vol. 46, no. 7, pp. 522–523, 2010.
[35] G. Sarlet, G. Morthier, and R. Baets, "Wavelength and mode stabilization of widely tunable SG-DBR and SSG-DBR lasers," *IEEE Photon. Technol. Lett.*, vol. 11, no. 11, pp. 1351–1353, Nov. 1999.
[36] S. Musalappa, "An energy efficient data cache implementing 2-way LRC architecture," M.S. thesis, Dept. Elect. Comput. Eng., Mississippi State Univ., Starkville, MS, USA, 2006.
[37] M. K. Qureshi, A. Jaleel, Y. N. Patt, S. C. Steely, and J. Emer, "Adaptive insertion policies for high performance caching," in *Proc. 34th Annu. Int. Symp. Comput. Archit. (ISCA)*, San Diego, CA, USA, 2007, pp. 381–391.
[38] S. Li *et al.*, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, New York, NY, USA, 2009, pp. 469–480.
[39] S. Thoziyoor, N. Muralimanohar, J.-H. Ahn, and N. P. Jouppi, "CACTI 5.1," HP Labs, Hewlett-Packard Develop. Company, Palo Alto, CA, USA, Tech. Rep. HPL-2008-20, 2008.
[40] J. Meng, C. Chen, A. K. Coskun, and A. Joshi, "Run-time energy management of manycore systems through reconfigurable interconnects," in *Proc. 21st Great Lakes Symp. VLSI (GLSVLSI)*, Lausanne, Swizerland, 2011, pp. 43–48.
[41] T. Vogelsang, "Understanding the energy consumption of dynamic random access memories," in *Proc. 43rd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, Atlanta, GA, USA, 2010, pp. 363–374.
[42] C. Batten, A. Joshi, V. Stojanovć, and K. Asanović, "Designing chip-level nanophotonic interconnection networks," in *Integrated Optical Interconnect Architectures for Embedded Systems*. New York, NY, USA: Springer, 2013, pp. 81–135.



Chao Chen (S'10–M'14) received the B.Eng. degree in electronic engineering from Shanghai Jiao Tong University, Shanghai, China, in 2005, the M.S. degree in electronic and electrical engineering from the Pohang University of Science and Engineering, Pohang, Korea, in 2007, and the Ph.D. degree in computer engineering from Boston University, Boston, MA, USA, in 2014.

He is currently a Performance-Modeling Engineer with the Digital Networking Group, Freescale Semiconductor Inc., Austin, TX, USA. His current research interests include performance modeling, system and network-on-chip architecture, and silicon-photonic technology.

José L. Abellán (S'07–M'08) received the B.S., M.S., and Ph.D. degrees in computer science and engineering from the University of Murcia, Murcia, Spain, in 2007, 2008, and 2012, respectively.

From 2012 to 2014, he was a Post-Doctoral Researcher at Boston University, Boston, MA, USA, under the supervision of Prof. A. Joshi. He is currently an Assistant Professor with the Computer Science Department, Catholic University of Murcia, Murcia. His current research interests include general-purpose computing on graphics processing units architecture, multicore/manycore architectures, networks-on-chip, high performance and green computing, parallel programming, CMOS, and nanophotonic technologies.

Ajay Joshi (S'99–M'07) received the B.Eng. degree in computer engineering from the University of Mumbai, Mumbai, India, in 2001, and the M.S. and Ph.D. degrees in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2003 and 2006, respectively.

From 2006 to 2009, he was a Post-Doctoral Researcher at the Massachusetts Institute of Technology, Cambridge, MA, USA. He is currently an Assistant Professor with the Electrical and Computer Engineering Department, Boston University, Boston, MA, USA. His current research interests include very large scale integration design including circuits and systems for communication and computation and emerging device technologies including silicon photonics and memristors.

Prof. Joshi was the recipient of the National Science Foundation CAREER Award in 2012.