UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Departamento de Informática
de Sistemas y Computadores

Novel Cache Hierarchies with Photonic Interconnects for Chip Multiprocessors

A thesis submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy
(Computer Engineering)

Author

José Puche Lara

Advisors

Prof. Julio Sahuquillo Borrás
Prof. Salvador V. Petit Martí

December 2020

Doctoral Committee

- Prof. José Ángel Gregorio Monasterio
  Universidad de Cantabria

- Prof. Antonio Robles Martínez
  Universidad Politécnica de Valencia

- Prof. Carlos Reaño González
  Queen's University of Belfast

Agradecimientos

La presentación de la tesis doctoral pone el punto final a una etapa tan importante en la vida como es la formación académica. Ha sido un camino largo y complicado y, consecuentemente, su última etapa tampoco ha sido fácil. Desde que, hace ya más de cinco años, Julio nos abordara al bueno de José Duro y a mí, y nos picara el gusanillo de la investigación (*els millors congressos del món*), han sido muchas las reuniones y las horas delante de la pantalla.

Sin embargo, no he estado solo. Se dice que la sección de agradecimientos de la tesis es la más importante porque al final es la única que se lee la gran mayoría de gente que, por azar o por castigo, acaba teniendo en sus manos tu tesis doctoral, exceptuando a los revisores. Personalmente, sin embargo, la considero importante por otro motivo: esta tesis doctoral también recoge toda la ayuda y el apoyo de quienes me han acompañado durante todos estos años, y para ellos es esta sección.

Me gustaría empezar agradeciendo su labor y su trato, como no puede ser de otra manera, a mis directores Julio y Salva. Gracias por depositar su confianza en mí para ser su alumno de doctorado; gracias por su paciencia y ayuda, por lo que me han enseñado tanto a nivel académico como a nivel personal; y gracias por haber conseguido, en definitiva, ayudarme a ser doctor.

Un escalón por debajo, pero igualmente de capital importancia para la elaboración de esta tesis, se encuentran mis compañeros del Grupo de Arquitecturas Paralelas. Muchos nombres han pasado por el acogedor laboratorio de la segunda planta, pero quiero hacer especial mención a algunos de ellos: gracias al ya mencionado José Duro, el buen *jebi*, por su elocuencia (qué eufemismo), generosidad y también, por qué no, por esos momentitos que nos ha regalado; gracias a José Rocher, incansable compañero de fatigas y lloros, por su sinceridad y complicidad desinteresadas; gracias a Javier Prades, por los cafés, por la pajita a Marte, por llevarme habitualmente la contraria, por las 13 Copas de Europa y por ayudarme a ver la vida de otro modo; gracias a Francisco Andújar, el doctor de Isso, eminencia albaceteña en historia borbónica y redes de computadores, por su energía contagiosa; y gracias a Leíto Chancay, cuya amistad debería contar como

Core A*, por todo en general y mucho en particular. A todos los demás: Josué, José María, Tomás, Santi, Danilo, Willian, Jaime, Adrián, Andrés, Escamilla, Vicent, Xisco (¡alabado seas!), Carlos, Lucía, Ricardo ... gracias por haber hecho de este viaje uno que mereciera la pena vivir.

Por último, quiero agradecer a mi familia y amigos su apoyo incondicional durante este tiempo. A mis padres, que siempre se las arreglan para ayudarme hasta en los temas más insospechados. A mi pareja, a la que se le ha hecho el doctorado casi más largo que a mí. A mis abuelos, en especial a mi querido abuelo Andrés, al que también se le ha hecho largo mi doctorado y ha preferido ver la presentación desde arriba. A mi tía Dori, por preocuparse tanto por mí. Y a mis amigos, que siempre están ahí, porque sin saberlo ayudan cada día a empezar el siguiente con una sonrisa. Gracias de corazón a todos.

Contents

List of Figures

List of Tables

Abbreviations and Acronyms

ANTT	**A**verage **N**ormalized **T**urnaround **T**ime
CMP	**C**hip **M**ulticore **P**rocessor
CPI	**C**ycles **P**er **I**nstruction
DRAM	**D**ynamic **R**andom-**A**ccess **M**emory
IABW	**I**deal **A**verage **B**and**W**idth
IPC	**I**nstructions **P**er **C**ycle
LLC	**L**ast-**L**evel **C**ache
L1	First-level (cache)
L2	Second-level (cache)
L3	Third-level (cache)
MPKI	**M**isses **P**er **K**ilo-**I**nstruction
NUMA	**N**on-**U**niform **M**emory **A**ccess
SMT	**S**imultaneous **M**ulti**T**hreading **M**emory
ST	**S**ingle-**T**hreaded
PA	**P**rocess **A**llocation
PS	**P**rocess **S**election
OATR	**O**nline **A**verage **T**ransaction **R**ate
OS	**O**perating **S**ystem
QoS	**Q**uality **o**f **S**ervice
ROB	**R**e**O**rder **B**uffer
STP	**S**ystem **T**hroughput
TLB	**T**ransaction **L**ookaside **B**uffer
UMA	**U**niform **M**emory **A**ccess

Abstract

Current multicores face the challenge of sharing resources among the different processor cores. Two main shared resources act as major performance bottlenecks in current designs: the off-chip main memory bandwidth and the last level cache. Additionally, as the core count grows, the network on-chip is also becoming a potential performance bottleneck, since traditional designs may find scalability issues in the near future.

Memory hierarchies communicated through fast interconnects are implemented in almost every current design as they reduce the number of off-chip accesses and the overall latency, respectively. Main memory, caches, and interconnection resources, together with other widely-used techniques like prefetching, help alleviate the huge memory access latencies and limit the impact of the core-memory speed gap. However, sharing these resources brings several concerns, being one of the most challenging the management of the inter-application interference. Since almost every running application needs to access to main memory, all of them are exposed to interference from other co-runners in their way to the memory controller. For this reason, making an efficient use of the available cache space, together with achieving fast and scalable interconnects, is critical to sustain the performance in current and future designs. This dissertation analyzes and addresses the most important shortcomings of two major shared resources: the Last Level Cache (LLC) and the Network on Chip (NoC).

First, we study the scalability of both electrical and optical NoCs for future multicores and many-cores. To perform this study, we model optical interconnects in a cycle-accurate multicore simulation framework. A proper model is required; otherwise, important performance deviations may be observed otherwise in the evaluation results. The study reveals that, as the core count grows, the effect of distance on the end-to-end latency can negatively impact on the processor performance. In contrast, the study also shows that silicon nanophotonics are a viable solution to solve the mentioned latency problems.

This dissertation is also motivated by important design concerns related to current memory hierarchies, like the oversizing of private cache space, data replication overheads, and

lack of flexibility regarding sharing of cache structures. These issues, which can be overcome in high performance processors by virtue of huge LLCs, can compromise performance in low power processors. To address these issues we propose a more efficient cache hierarchy organization that leverages optical interconnects. The proposed architecture is conceived as an optically interconnected two-level cache hierarchy composed of multiple cache modules that can be dynamically turned on and off independently. Experimental results show that, compared to conventional designs, static energy consumption is improved by up to 60% while achieving similar performance results.

Finally, we extend the proposal to support both sequential and parallel applications. This extension is required since the proposal adapts to the dynamic cache space needs of the running applications, and multithreaded applications's behaviors widely differ from those of single threaded programs. In addition, coherence management is also addressed, which is challenging since each cache module can be assigned to any core at a given time in the proposed approach. For parallel applications, the evaluation shows that the proposal achieves up to 78% static energy savings.

In summary, this thesis tackles major challenges originated by the sharing of on-chip caches and communication resources in current multicores, and proposes new cache hierarchy organizations leveraging optical interconnects to address them. The proposed organizations reduce both static and dynamic energy consumption compared to conventional approaches while achieving similar performance; which results in better energy efficiency.

Resumen

Los procesadores multinúcleo actuales cuentan con recursos compartidos entre los diferentes núcleos. Dos de estos recursos compartidos, la cache de último nivel y el ancho de banda de memoria principal, pueden convertirse en cuellos de botella para el rendimiento. Además, con el crecimiento del número de núcleos que implementan los diseños más recientes, la red dentro del chip también se convierte en un cuello de botella que puede afectar negativamente al rendimiento, ya que las redes tradicionales pueden encontrar limitaciones a su escalabilidad en el futuro cercano.

Prácticamente la totalidad de los diseños actuales implementan jerarquías de memoria que se comunican mediante rápidas redes de interconexión. Esta organización es eficaz dado que permite reducir el número de accesos que se realizan a memoria principal y la latencia media de acceso a memoria. Las caches, la red de interconexión y la memoria principal, conjuntamente con otras técnicas conocidas como la prebúsqueda, permiten reducir las enormes latencias de acceso a memoria principal, limitando así el impacto negativo ocasionado por la diferencia de rendimiento existente entre los núcleos de cómputo y la memoria. Sin embargo, compartir los recursos mencionados es fuente de diferentes problemas y retos, siendo uno de los principales el manejo de la interferencia entre aplicaciones. Hacer un uso eficiente de la jerarquía de memoria y las caches, así como contar con una red de interconexión apropiada, es necesario para sostener el crecimiento del rendimiento en los disenõs tanto actuales como futuros. Esta tesis analiza y estudia los principales problemas e inconvenientes observados en estos dos recursos: la cache de último nivel y la red dentro del chip.

En primer lugar, se estudia la escalabilidad de las tradicionales redes dentro del chip con topología de malla, así como ésta puede verse comprometida en próximos disenõs que cuenten con mayor número de núcleos. Los resultados de este estudio muestran que, a mayor número de núcleos, el impacto negativo de la distancia entre núcleos en la latencia puede afectar seriamente al rendimiento del procesador. Como solución a este problema, en esta tesis proponemos una de red de interconexión óptica modelada en un entorno de

simulación detallado, que supone una solución viable a los problemas de escalabilidad observados en los diseños tradicionales.

A continuación, esta tesis dedica un esfuerzo importante a identificar y proponer soluciones a los principales problemas de diseño de las jerarquías de memoria actuales como son, por ejemplo, el sobredimensionado del espacio de cache privado, la existencia de réplicas de datos y rigidez e incapacidad de adaptación de las estructuras de cache. Aunque bien conocidos, estos problemas y sus efectos adversos en el rendimiento pueden ser evitados en procesadores de alto rendimiento gracias a la enorme capacidad de la cache de último nivel que este tipo de procesadores típicamente implementan. Sin embargo, en procesadores de bajo consumo, no existe la posibilidad de contar con tales capacidades y hacer un uso eficiente del espacio disponible es crítico para mantener el rendimiento. Como solución a estos problemas en procesadores de bajo consumo, proponemos una novedosa organización de jerarquía de dos niveles cache que utiliza una red de interconexión óptica. Los resultados obtenidos muestran que, comparado con diseños convencionales, el consumo de energía estática en la arquitectura propuesta es un 60 % menor, pese a que los resultados de rendimiento presentan valores similares.

Por último, hemos extendido la arquitectura propuesta para dar soporte tanto a aplicaciones paralelas como secuenciales. Esta extensión no es trivial, ya que la propuesta adapta dinámicamente el espacio de cache requerido por las aplicaciones en ejecución, y es bien conocido que aplicaciones paralelas y secuenciales presentan coomportamientos muy diferentes. Además, el manejo de la coherencia de memoria se vuelve más complejo, ya que los módulos de cache pueden ser asignados y desasignados a cualquier núcleo en cualquier momento de la ejecución. Los resultados obtenidos con la esta nueva arquitectura muestran un ahorro de hasta el 78 % de energía estática en la ejecución de aplicaciones paralelas.

En resumen, esta tesis estudia y propone soluciones a los problemas derivados de la compartición de caches y recursos de comunicación dentro del chip en los procesadores multinúcleo actuales. Las dos organizaciones de cache propuestas reducen el consumo de energía tanto estática como dinámica en comparación a otras arquitecturas convencionales, a la vez que consiguen un rendimiento similar.

Resum

Els processadors multinucli actuals compten amb recursos compartits entre els diferents nuclis. Dos d'aquests recursos compartits, la memòria d'últim nivell i l'ample de banda de memòria principal, poden convertir-se en colls d'ampolla per al rendiment. A més, amb el creixement del nombre de nuclis que implementen els dissenys més recents, la xarxa dins del xip també es converteix en un coll d'ampolla que pot afectar negativament el rendiment, ja que les xarxes tradicionals poden trobar limitacions a la seva escalabilitat en el futur proper.

Pràcticament la totalitat dels dissenys actuals implementen jerarquies de memòria que es comuniquen mitjançant ràpides xarxes d'interconnexió. Aquesta organització és eficaç atès que permet reduir el nombre d'accessos que es realitzen a memòria principal i la latència mitjana d'accés a memòria. Les caches, la xarxa d'interconnexió i la memòria principal, conjuntament amb altres tècniques conegudes com la prebúsqueda, permeten reduir les enormes latències d'accés a memòria principal, limitant així l'impacte negatiu ocasionat per la diferència de rendiment existent entre els nuclis de còmput i la memòria. No obstant això, compartir els recursos esmentats és font de diversos problemes i reptes, sent un dels principals la gestió de la interferència entre aplicacions. Fer un ús eficient de la jerarquia de memòria i les caches, així com comptar amb una xarxa d'interconnexió apropiada, és necessari per sostenir el creixement del rendiment en els dissenys tant actuals com futurs. Aquesta tesi analitza i estudia els principals problemes i inconvenients observats en aquests dos recursos: la memòria cache d'últim nivell i la xarxa dins del xip.

En primer lloc, s'estudia l'escalabilitat de les xarxes tradicionals dins del xip amb topologia de malla, així com aquesta es pot veure compromesa en propers dissenys que compten amb major nombre de nuclis. Els resultats d'aquest estudi mostren que, a major nombre de nuclis, l'impacte negatiu de la distància entre nuclis en la latència pot afectar seriosament al rendiment del processador. Com a solució a aquest problema, en aquesta tesi proposem una xarxa d'interconnexió òptica modelada en un entorn de

simulació detallat, que suposa una solució viable als problemes d'escalabilitat observats en els dissenys tradicionals.

A continuació, aquesta tesi dedica un esforç important a identificar i proposar solucions als principals problemes de disseny de les jerarquies de memòria actuals com són, per exemple, el sobredimensionat de l'espai de memòria cache privat, l'existència de rèpliques de dades i la rigidesa i incapacitat d'adaptació de les estructures de memòria cache. Encara que ben coneguts, aquests problemes i els seus efectes adversos en el rendiment poden ser evitats en processadors d'alt rendiment gràcies a l'enorme capacitat de la memòria cache d'últim nivell que aquest tipus de processadors típicament implementen. No obstant això, en processadors de baix consum, no hi ha la possibilitat de comptar amb aquestes capacitats, i fer un ús eficient de l'espai disponible es torna crític per mantenir el rendiment. Com a solució a aquests problemes en processadors de baix consum, proposem una nova organització de jerarquia de dos nivells de memòria cache que utilitza una xarxa d'interconnexió òptica. Els resultats obtinguts mostren que, comparat amb dissenys convencionals, el consum d'energia estàtica en l'arquitectura proposada és un 60% menor, malgrat que els resultats de rendiment presenten valors similars.

Per últim, hem estès l'arquitectura proposada per donar suport tant a aplicacions paral·leles com seqüencials. Aquesta extensió no ès trivial, ja que la proposta adapta dinàmicament l'espai de memòria cache requerit per les aplicacions en execució, i ès ben conegut que aplicacions paral·leles i seqüencials presenten comportaments molt diversos. A més, la gestió de la coherència de memòria es torna més complex, ja que els mòduls de memòria cache poden ser assignats i desassignats a qualsevol nucli en qualsevol moment de l'execució. Els resultats obtinguts amb aquesta nova arquitectura mostren un estalvi de fins al 78 % d'energia estàtica en l'execució d'aplicacions paral·leles.

En resum, aquesta tesi estudia i proposa solucions als problemes derivats de la compartició de caches i recursos de comunicació dins del xip en els processadors multinucli actuals. Les dues organitzacions de memòria cache proposades redueixen el consum de energia tant estàtica com dinàmica en comparació a altres arquitectùres convencionals, alhora que aconsegueixen un rendiment similar.

Chapter 1

Introduction

This chapter introduces some concepts to help understand the manuscript and presents the motivation for the work developed in the proposed approaches. First, background on Chip Multiprocessors (CMP) is introduced, focusing on major components of the architecture like the cache hierarchy or the network-on-chip. Next, we discuss major challenges that arise in these research topics and how they are addressed in current processors. After that, we present insights about where performance problems arise in multicores due to shared resources, especially the network on chip (NoC) and the shared caches. Then, we summarize the way this dissertation addresses these shortcomings by proposing a novel cache organization. Finally, we present the thesis outline.

1.1 Background: Shared Resources in Chip Multiprocessors

Since published by Gordon Moore in 1965, the performance improvement of microprocessors has roughly followed what is known as the Moore's Law [1], which states that the amount of transistors in a microprocessor doubles every two years. Following this rule, the performance of microprocessors has steadily increased throughout the past decades, thanks to the continuous reductions in the integration scale and the design of more and more complex hardware structures and architectures. Although this technology scaling trend has resulted in the achievement of huge computing power, the bigger complexities and the large amounts of transistors bring new concerns regarding power consumption.

As the number of transistors grows, the electrical power they consume generates heat that far exceeds the cooling capacities of current processors. This fact is known as the *power wall*, which refers to the performance scaling limitations caused by power consumption constraints in terms of heat and cost. There are two possible solutions to address the power wall problem: i) designing inexpensive cooling techniques and technologies; and ii) reducing the processor complexity and, by doing so, cutting down the power consumption.

In order to keep the power consumption in acceptable levels while sustaining the performance gains brought by technology scaling, processor designers and manufacturers moved to multicore approaches following the second solution mentioned above. The multicore paradigm initially pursued to deploy multiple but less complex *cores* (i.e. computing units provided with a given amount of private memory) in a processor die as opposite to increasing the complexity of a single-core monolithic processor. This approach allows reducing the overall energy consumption while achieving higher aggregated performance than monolithic processors. Currently, multicores can be found not only in the high-performance computing market but also in almost any other segment of the market, since these processors have become ubiquitous in almost any commercial electronic device.

Multicore processors, however, are not exempt from other shortcomings and design issues. First, single-threaded applications need to be adapted to effectively exploit the

parallelism provided by multicore architectures. Second, multicore architects reduce cost by implementing resources (e.g. the main memory bandwidth) which are shared among the processor cores. This fact, which represents an advantage from the cost perspective, becomes a shortcoming since inter-application interference is introduced at the shared resources, making the execution time of individual applications unpredictable.

This dissertation is aimed at achieving an efficient use of shared resources in a chip multiprocessor. Taking into account the different nature of each resource, this aim is challenging, since it has to be addressed from different directions. For this reason, we address this problem with a two-step strategy. First, we explore the use of a novel technology to address scalability issues of current and future electrical NoCs. Then, the results of this study are used to build, on the top of that, a holistic solution for the cache hierarchy. Finally, after these two steps, we refine and adapt the proposed solution to efficiently work with both single-threaded and multithreaded or parallel applications.

Below, chip multiprocessor architectures are further studied, including the specific challenges that still remain for each of these shared resources.

1.1.1 Chip Multiprocessor

For the past decades, microprocessor manufacturers exploited the reductions on the integration scale to achieve more computing power. At the same time, hardware complexity increased as new components such as non blocking caches, support for out-of-order execution, or speculative loads were introduced in the new designs. Together, technology developments and refined hardware structures have sustained microprocessors performance, continuously increasing the maximum frequency (f) that the chips are able to achieve.

In conjunction with Moore's law, Dennard scaling [2] has been the other major pillar for the performance-driven computing industry. Formulated by Robert H. Dennard in 1974, the MOSFETs scaling rule states that as transistors get smaller, their power density remains constant. The method proposed by Dennard and his team allowed scaling chip dimensions, voltage, current and operating frequency while keeping a steady device power density. Following this rule, if there was a reduction in a transistor's linear size by 2, the power it used fell by 4 (with voltage and current both halving).

Unfortunately, technological advances have exposed new findings that slow down the traditional scaling path predicted by these rules. On the one hand, as shrinking transistors becomes more and more challenging, Moore's law is expected to extend the duration between technology nodes far from that predicted by the initial rate. On the other hand, Dennard scaling has already been broken down. With current transistor sizes, the ability to drop the voltage and the current that they need to operate reliably has reached an end. This is because the original Dennard scaling rule accounted only for transistors size, voltage, current and frequency, that is, the dynamic power of the processor. However, as transistor dimensions shrinks, a new source of power dissipation appears in the form of leakage currents. The explanation to this fact is as follows: the Dennard scaling rule requires an equal scaling of the threshold voltage to scale down the supply voltage; threshold voltage, however, is physically limited and scaling it down dramatically increases static power; therefore, failing to scale down threshold voltage translates to failing to scale down supply voltage, which breaks Dennard scaling. In all projections, device power densities are expected to keep growing in the foreseeable future.

As a consequence of the breaking down of Dennard scaling, the operating frequencies of new devices stopped growing. Given this situation, chip manufacturers and designers adopted Chip Multiprocessors, also known as multicore processors. A CMP is typically a single processor that contains multiple cores, computing units that are provided with small private caches. The cores are connected through an interconnection network on-chip, which also links the cores to other shared resources like the last level cache (LLC) or the memory controllers. Multicore designs that implement a significant amount of cores (e.g. 32 cores) typically follow a mesh topology since it eases the layout of wires and facilitates the manufacture process.

The adoption of the multicore paradigm helped both performance and power consumption: the former, due to the exploitation of parallelism (i.e. tasks can be distributed among the available cores); and the latter, because of the lower complexity of the cores, which reduces their switched capacitance and, ultimately, the dynamic power consumption. These advantages, however, come at the expense of new challenges that need to be addressed by computer architects and software developers. For instance, in order to

exploit parallelism, applications first need to be adapted to divide their workload and therefore be benefited from the multiple cores.

Multicores also pose the challenge of sharing resources. Future multicores are expected to increase the amount of cores, hindering the management of the shared processor resources. Key shared resources in a typical multicore are the on-chip last level cache cache and the main memory bandwidth. When the cores are running different applications concurrently, these applications introduce interference at the shared resources, thrashing the cache or stressing the main memory bandwidth.

Another shared resource that is gaining importance as the core count increases is the on-chip network. Technology advancements currently enable integrating hundreds of cores in a single chip, which compromises the scalability of existent NoC approaches. Additionally, CMPs need from fast communication among cores, caches, memory controllers and other resources; otherwise, the performance may be significantly compromised. Therefore, future multicore generations have to face the challenges of fast communications and low power consumption, so existent interconnection approaches and their scalability need to be revisited.

In order to sustain the performance while keeping power consumption in acceptable levels, multicore processors need to address the aforementioned issues, and make an efficient use of the available shared resources. With this objective, this dissertation analyzes and addresses specific problems derived from current NoCs and cache hierarchies. Below, these two major components of chip multiprocessors are further discussed and explained.

1.1.2 Cache Hierarchy

Processors have implemented cache memories for more than four decades, becoming a key component since the design or the first pipelined processors. Cache memories exploit spatial and temporal localities of the data referenced by the running applications, which significantly boosts the system performance. Due to the increasing disparity between the speed of the cache and the main memory, new cache levels were deployed in the called cache hierarchy. The cache hierarchy includes huge last level caches with the aim of reducing the number of off-chip main memory accesses, therefore this organization helps mitigate the consumed main memory bandwidth.

Most existing multicores implement a three level cache hierarchy, where the two first levels are private to each core and the third level, referred to as the LLC, is shared by all the cores. The higher hierarchy cache levels (L1 cache and L2 cache) are small (e.g. from 32 KB up to 512 KB) and fast, while the LLC size is much bigger, especially in high-performance processors. For instance, Intel Xeon Skylake processors implement an LLC of up to 38.5 MB at its maximum configuration (28 cores), and IBM POWER8 devotes to it 96 MB (12-chiplet processor) [3, 4].

It can be noticed that, to reduce the number of main memory accesses, current LLCs are on the order of tens of MB and this capacity keeps increasing. LLC misses are specially harmful for the processor performance, since accessing main memory is costly and can take hundreds of processor cycles. These huge latencies block the reorder buffer of the cores, and ultimately prevent them issuing new instructions. Additionally, since a key goal of LLCs is to avoid capacity misses, high associativities need to be used to reduce the number of conflict misses; for this reason, the associativity degree of the LLCs is usually higher than 16 ways.

Despite its huge capacity, the fact that the LLC is a shared resource rises important shortcomings that need to be addressed. First, requests from the co-running applications compete among them for the LLC, introducing interference in this resource. When distinct applications access the LLC, they can interfere each other and evict cache blocks that are being used by the co-runner, impairing the cache performance. Additionally, not all applications make an efficient use of the LLC, since some workloads (e.g. streaming applications) may have no locality. Other applications, like those belonging to the cloud segment, present different needs such as providing isolation or guaranteeing quality of service (QoS).

Efficiently managing the LLC is, therefore, an important concern that needs to be addressed for improving performance and fairness. Targeting this objective, many solutions have been proposed in the past to efficiently share the cache, being the most popular approach the cache partitioning policies (see Section 2.2.1). Approaches adopting this solution assign specific cache ways to the applications running in the processor cores according to different criteria. Proceeding in this way applications that make a less efficient use of the cache receive a reduced number of ways, which limits their adverse impact on

the performance. Way partitioning can be performed by software (i.e. page coloring), by extending the hardware capabilities or by modifying the replacement algorithm.

Although effective for reducing the interference, existing cache partitioning solutions do not completely solve the problem. Way partitioning solutions do not reduce the on-chip latency, since they mostly work at the LLC. Moreover, this kind of approaches are not as effective for processors that implement low-capacity caches, such as those belonging to the low-power and embedded segments of the market, as it is for high-performance processors. This means that, in order to improve the cache performance in multicores, a holistic approach on the topic needs to be taken, addressing both on-chip latencies and isolation capabilities.

1.1.3 Network On-Chip

Multicores require from efficient on-chip interconnection networks to provide fast communication among cores, caches and memory controllers. Simple and effective electrical networks following a mesh topology are the mainstream design choice for many chip manufacturers, although some processors also implement ring-based topologies (e.g. Intel Skylake [3]). However, its effectiveness drops as the core count increases, and the scalability constraints of conventional electrical NoC need to be revisited.

Regarding scalability issues, the continuous reductions on the integration scale currently enable integrating hundreds of cores on a single chip [5–7]. Conventional electrical networks may suffer performance degradation when dealing with such a high number of cores. These networks often present a mesh topology where memory controllers are usually placed at the corners and edges of the processor chip. Since the NoC must interconnect all the processor tiles (e.g. core plus L2 cache), the higher the number of cores, the higher the average distance from the nodes to the memory controller, and hence to the main memory. In such a scheme, when a node requests a cache block, the incurred latencies and energy consumption could be unacceptable depending on the distance to the memory controller. Current multicore architectures incorporate several MCs to alleviate this shortcoming but, unfortunately, the number of memory controllers can not scale linearly with the number of nodes [8].

The scalability concerns are directly related with power consumption. As discussed before, in order to prevent heating problems, multicore designers are forced to keep a limited power budget. NoC designers are not exempt from this directive, since as the number of network nodes increases, so it does the number of network switches and the overall cost, in terms of power, of sending data through the network. It is expected, therefore, that high-performance CMPs exceeding a high number of cores (e.g. 100 cores) suffer from the power bottlenecks of current electrical interconnects [9].

For the last decade, the aforementioned concerns led to new and alternative designs to conventional NoCs. In this context, one of the most promising solutions to the scalability issues of existing approaches is photonics interconnects. Below, this novel technology is briefly presented and discussed.

1.1.3.1 Photonics Interconnects

The need of a faster on-chip multicore communication technology has led to lay out the use of CMOS-compatible photonic interconnects as a possible solution. Nanophotonics technology has experienced a vertiginous development during the last decade and this trend is expected to continue in the next years. Because of its high bandwidth and energy consumption, which scarcely varies with the communication distance, CMOS compatible photonics interconnects is the most promising technology to satisfy future CMPs' bandwidth demands [10].

Manycore and multicore architectures can leverage the capacities provided by nanophotonics to reduce their network latency and, as a result, their average memory access time. Additionally, wavelength multiplexing eases the implementation of multicast and broadcast operations, which are widely used in CMPs (e.g. the coherence protocol performs multicast transmissions when invalidates incoherent copies in multiple nodes).

However, new technologies such as photonics interconnects bring the cost of facing also new challenges. Since this technology is still maturing, it is difficult to find up-to-date models in CMP simulation frameworks, which are mainly developed by computer architects whose focus is on computing and communication aspects. This situation leads to incomplete models whose results could present important deviations and generate an

expectancy on this technology that still does not correspond with its current development state. Therefore, the first challenge is achieving detailed and accurate simulation environments to develop reliable CMP simulators considering photonics interconnects.

Other major challenges still remain for this technology, especially for silicon photonics. On the one hand, current losses observed in silicon photonics devices can be too prohibitive from a power consumption perspective. The higher the loss introduced by these devices, the higher the output power that the laser source needs to drive all receivers at satisfaction bit error rates. On the other hand, ring heater power is yet to be improved. Ring heater power is needed since it mitigates temperature variations and post-manufacturing geometric mismatches of microring resonators. A prohibitive heating power means that only a limited amount of microrings can be equipped in the Optical-Network on Chip (ONoC) [11]; otherwise, the static power consumption would be out of budget.

However the challenges, the potential benefits that nanophotonics technology can bring to already existing and future CMP architectures are still worth it. Current efforts of computer architects shall focus on achieving accurate models and designing new interconnection approaches so, together with technology scaling, on-chip communications could overcome scalability issues in the near future.

1.2 Main Contributions of the Thesis

The three main contributions of this thesis are described below:

- **Accurately modeling an ONoC in a CMP simulation framework.** Photonic interconnects are a promising solution for the so-called communication bottleneck in current CMP architectures. This technology presents an inherent low-latency and power consumption almost independent of communication distance, which are key features to improve the performance in future Networks on Chip generations. Nevertheless, since nanophotonics technology is still growing and therefore it is still immature, current detailed system simulators lack to provide complete and detailed models of photonic components. Consequently, in case that such optical models are assumed, the obtained results may introduce important

deviations. This dissertation summarizes all the components that conform a fully operative optical NoC and presents their current state-of-the-art. Additionally, realistic ring-based ONoCs featuring token arbitration techniques are evaluated and compared against their electrical counterparts.

- **Increasing energy efficiency in low-power CMPs.** Conventional cache hierarchy approaches of current multicores incur area and energy wasting due to several well-known design issues: oversizing the private cache space, replicating data through the inclusive cache levels and the increasing cache associative degree. In spite of these shortcomings, the multi-level hierarchy is the commonly adopted design approach. In this dissertation we claim that a more energy efficient organization is needed to address these design issues. To this end, this thesis proposes Flat On-Chip Storage (FOS), a novel cache organization aimed at addressing energy and area of low-power processors by facing these design issues. FOS combines L2 and L3 cache levels into a single and flattened cache level composed of a pool of private small cache buffers. These buffers are initially powered off to save energy, and buffers are powered on and assigned to cores when the system performance is expected to improve. To provide fast and uniform access from the private L1 caches to the FOS cache buffers, multiple architectural challenges are overcome, including the design of a custom optical network-on-chip.

- **Increasing energy efficiency in low-power CMPs for multithreaded applications.** The aforementioned drawbacks associated to conventional cache organizations are especially harmful when running parallel applications. This kind of applications typically employs multiple threads accessing to shared data, and efficiently managing these data in the cache hierarchy is key to achieve high performance. This dissertation extends the baseline FOS architecture to propose FOS-Mt, a cache organization aimed at addressing energy savings in current multicores for multithreaded applications.

1.3 Thesis Outline

This dissertation consists of seven chapters. The remaining chapters are organized as follows:

- Chapter 2 discusses previous work on both photonics interconnects and cache organizations.
- Chapter 3 presents the experimental platforms and common aspects of the evaluation methodology.
- Chapter 4 introduces the accurate modeling of optical devices, and the integration of these models in a state-of-the-art CMP simulation framework.
- Chapter 5 proposes Flat On-chip Storage, an efficient cache organization that reduces energy consumption in CMPs.
- Chapter 6 extends previous Flat On-Chip Storage approach and optimizes the proposal for multithreaded applications.
- Chapter 7 summarizes this thesis, discusses future work and enumerates related publications.

Chapter 2

Related Work

This chapter discusses previous work related to this dissertation, summarizing important state-of-the-art works which paved the way for the approaches proposed in this thesis. First, it is introduced the related work regarding optical interconnects, covering both the latest technological advances and recent optical networks on-chip. Next, we discuss important work dealing with cache sharing and cache partitioning, focusing our efforts on energy efficient caches. Finally, we also discuss related work that addresses energy savings at the cache hierarchy.

2.1 Optical Interconnects

This section discusses the state-of-the-art work focusing on optical devices and optical networks on-chip. First, we summarize some representative works dealing with technological advances in Photonic Integrated Circuits, focusing on the integration of on-chip lasers and improving microrings efficiency. Next, we discuss important proposals regarding optical networks on-chip in CMPs, which have guided us towards the design of the networks proposed in this dissertation.

2.1.1 Optical Devices

Current research efforts focusing on Photonic Integrated Circuits (PICs) concentrate on the realization of reliable hybrid silicon lasers, electro-optical modulators, and detectors, which are the most critical building blocks of optical circuits. The promising research results pave the way for achieving fully on-chip integrated devices, able to overcome the inherent limitations of electronic networks [12].

Laser sources are the most difficult devices to be integrated in the on-chip processor die due to power, area, and optical signal attenuation constraints. Duan et al. have developed hybrid silicon/III-V lasers [13, 14] exhibiting new features and lower power consumption than previous works [15, 16]. However, these advances do not yet achieve the ultra-low power consumption required for on-chip laser integration. Moreover, integrated lasers are only able to provide output signal powers of tens of mW, raising attenuation concerns, although it is expected to accomplish higher figures in the next few years, hence allowing the exploitation of the real potential of on-chip lasers. In fact, some works on optical NoCs assume that on-chip lasers will be integrated in future technologies since they are much more energy-efficient [17].

Also, regarding lasers, other approaches apart from those deriving from technological improvements are possible. In this regard, Joshi et al. [18] propose a multibus network architecture to dynamically distribute the laser power across multiple buses and even turn the laser sources off at run-time, hence reducing the overall energy consumption. Following a similar approach, in [19] authors propose Ecolaser, an adaptive control mechanism that turns the laser off and saves up to 77% of the laser power. In addition,

other works like [20–23] target the heating problem of current designs and propose thermal management solutions, which is a key design issue to deal with the energy efficiency requirements of existing and future approaches.

The switching capacity (i.e. data routing) of electro-optical modulators represents the key feature for establishing the operation bandwidth of any PIC. High-bandwidth modulation in silicon (achieving up to 30 to 50 Gb/s data rates and working at switching times of several GHz [24–26]) can be realized by free-carrier induced index change and using biased `pn` structures (carrier depletion) [27, 28].

Optical coherent receivers (i.e. detectors), which convert the amplitude, phase, and polarization of an optical field signal into the electrical domain, have already been integrated showing similar performance to those deployed in commercial devices with very high data conversion. Polarization-division-multiplexed quadrature phase-shift keying (PDM-QPSK) has been employed for 100-Gb/s networks, and even higher modulation formats such as 16-ary quadrature amplitude modulation (16-QAM) can be utilized . These modulation formats can achieve a performance in commercial devices up to 224 Gb/s with PDM-16-QAM signals [29]. Depending on the technology node, the output signal of these devices may need to be amplified before being used in the digital circuit. Small input capacitances for transistors would allow obtaining a signal that would not require any amplifier and therefore would reduce the power consumption [30].

Regarding current research on other PIC components, critical issues are to minimize light signal attenuation in the manufacturing process of waveguides [31] and to reduce the width of the light spectrum that resonators can filter. The latter characteristic defines the number of wavelenghts achievable by Dense Wavelength Division Multiplexing (DWDM), which currently ranges from 64 to 160 wavelengths per waveguide.

2.1.2 Optical Networks On-Chip

Based on the aforementioned technological advances regarding the integration of fully optical components, many works have been proposed focusing on optical networks on-chip [30, 32–43]. Next we discuss state-of-the-art research on hybrid photonic-electronic and pure optical NoC implementations.

In this regard, Vantrease et al. explore bandwidth requirements of future manycores and propose Corona [32], which is a 3D-stacked architecture that employs silicon photonics for both on-chip and off-chip communication. Kurian et al. present ATAC [30], a 1K-core system that leverages optical interconnects to implement a global broadcast network designed together with minor modifications in the coherence protocol. LumiNOC [34] follows a more conservative approach and implements a fully-optical network which is partitioned into subnets for the sake of efficiency.

Although fully-optical solutions are theoretically possible, their static power is considered still too high and prevents them to be adopted in the near future. It could be said that most of the ONoCs that can be found in existing works are hybrid, in the sense that they put together electrical and optical components. For instance, in [39], authors propose a hybrid network composed of optical links and a circuit-switched electrical mesh, aimed at improving broadcast operations in current and future manycores. In [33], authors propose Firefly, a hybrid network that performs node clustering and communicates these clusters using optical interconnects. METEOR [42] is also a hybrid network composed of an optical ring and a conventional 2D electrical mesh network, which authors claim to improve efficiency with respect to previous works like [32, 33]. In [35], authors also target efficiency by globally sharing several optical channels across the network. The scalability issues attached to optical crossbars are further studied in [40], where authors propose PROPEL, a hybrid network that implements a scalable optical crossbar and electrical switching for a 256-core CMP.

Following a different approach, in [37] Joshi et al. explore silicon-photonics clos networks (PClos), reducing energy consumption with respect to fully-optical global crossbars. In [38], authors adopt the previous approach and propose a bufferless implementation of PClos that improves scheduling and routing for this kind of networks.

Research on photonic NoCs is closely related to research on DWDM arbitration techniques. Since most DWDM-based communication schemes require wavelength sharing, some works consider the arbitration as an important part of the communication efficiency [32, 35]. In this context, Vantrease et al. [44] also take advantage from silicon photonics to perform arbitration related tasks. They identify latency, average network utilization and fairness as the key features that a suitable arbitration mechanism must address.

Other studies also point out that the cost of photonic resources must be taken into account. In [45], García Guirado et al. analyze the overhead of photonic components employed in some NoC proposals. They note that a responsible management and distribution of these resources can result in significant performance gains while keeping optical-related costs in acceptable levels.

Notice that most of the aforementioned works like [32–35] aim to improve network performance (by increasing network bandwidth) while also reducing energy consumption. Other works do not concentrate on improving the optical networks, but they take advantage of the capabilities that these networks can bring to overcome traditional limitations of current CMPs. In this context, some works like [36, 46, 47] leverage optical interconnects to enhance particular aspects of the architecture. In [46], Bartolini and Grani enhance a traditional NoC with an optical ring to support low-latency coherence control messages, making use of the optical technology capabilities to implement multicast transmissions. In [36], Cianchetti et al. employ an optical crossbar to reduce the number of hops for distant communications when transmitting 64-byte cache lines, which translates into a shorter memory access time. Similarly, in [47], a hybrid NoC that employs optical links for long-distance block transmissions while keeping electrical links for close communications is proposed.

Finally, other works explore the use of silicon photonics in other architectures different than CMPs. For instance, in [48], authors leverage optical interconnects to improve the scalability issues inherent to future GPUs, as the number of compute units grows in each GPU generation. Authors use a hybrid photonic crossbar to improve latency and energy efficiency of communications at the memory hierarchy of an AMD Southern Islands GPU.

2.2 Adaptive Cache Organizations

Cache organizations have been an important research topic when dealing with performance and energy consumption in chip multiprocessors. More specifically, important works [49–52] showed in the past that static and rigid cache hierarchies are inefficient and that applications may suffer from well known issues like unfairness or starvation in such systems. On the other hand, some seminal work [53–55] pointed out that splitting

the cache organization in small and independent cache structures do not only reduce energy consumption but also, when these structures are properly managed, may also enhance the system performance.

In this context, research approaches aimed at improving the utilization and efficiency of the cache hierarchy can be classified following different criteria. This section first presents a subset of previous *cache sharing* and *cache partitioning* approaches, which are typically applied to shared caches. Next, we discuss some important works that propose *cooperative caching* approaches as a way to improve the efficiency of the cache hierarchy. Finally, other approaches are studied in Section 2.2.3.

2.2.1 Cache Partitioning

This category includes works that pursue to improve the use of the shared cache space by partitioning it according to different criteria. Approaches falling in this category [49–51, 56–58] aim to improve the system performance by properly assigning specific cache ways to the different applications running in the processor cores. An example of *cache-way* based partitioning can be found in [49], where a CMP prototype using this technique at the last level cache is presented. Another interesting piece of research belonging to this group is UCP [50] by Qureshi and Patt. This approach distributes the cache ways of the LLC among the co-running applications based on the reduction of cache misses that each one of them is likely to obtain for a given amount of cache resources. The estimates on the reduction of misses are carried out by duplicating tags and sampling a reduced amount of sets. ASM-Cache [56] employs similar techniques to gather useful information to partition the cache. In [51], authors propose distributing the cache ways with the aim of improving system fairness rather than just focusing on performance.

Other works [59–61] that also explore cache partitioning at the cache-way granularity focus on the cache replacement algorithm. Both [59, 60] propose using eviction probabilities to manage the occupation of the LLC, for instance, increasing the probability of evicting cache lines occupied by a given core. In [61], authors also partition the cache but control the size of partitions by properly scaling the futility of their cache lines.

A different approach considered to devise cache partitioning solutions is based on *skew associative* caches [62] or *zcaches* [63]. These approaches [64, 65] decouple the cache

associativity from the number of cache ways by increasing the replacement candidates upon a cache miss, hence they enable to partition the cache using the replacement policy. In [64], up to 90% of the cache is partitioned this way, by soft-pinning a large portion of the cache lines. In [65], authors dynamically partition the cache based on the predicted transient behavior of latency-critical workloads, maintaining their tail latency and ensuring QoS.

The aforementioned works focus on approaches that propose hardware modifications to current designs. However, software-based approaches [66–70] have been also proposed to partition the cache. Most of these approaches are based on page coloring, which is used to control the location of the application data in the cache. In [66], authors adopt a hybrid software and hardware approach, which aims to reduce the cache misses by: i) color mapping at compile time code and data pages; and ii) adding a page remap field to the TLB. A simple and software-only approach is followed in [67], which relies on the OS to allocate the physical page that maps the desired portion of a L2 shared cache. COLORIS [68] analyzes the case of page coloring in over-committed processors (i.e. where there are more executable threads than cores) and propose several policies to reconfigure the assignment of page colors among application threads. Lastly, in [69] authors refine the approach and propose hot-page coloring, which colors only a small set of frequently accessed pages for each process. The reason behind this approach is reducing the adverse effect associated to the lack of versatility of page coloring approaches, which need to reallocate the pages when repartitioning occurs.

Finally, more recent works [71–75] leverage Intel's Cache Allocation Technology, implemented in some current processors (e.g. Intel Xeon E5 v4 family), in order to limit the amount of LLC ways that a hardware thread can occupy. Some of these works [73–75] focus on the cloud field, and employ Intel CAT to maximize the utilization of large-scale data centers and to partition the cache when renting virtual machines. In [73], authors propose Heracles, a feedback-based framework that increases the utilization in large data centers while ensuring that latency-sensitive jobs meet their latency targets. In Dirigent [74], a lightweight performance management runtime system is proposed to accurately control the QoS of latency-critical applications using existing architectural solutions like Dynamic Voltage and Frequency Scaling (DVFS) or the already mentioned

Intel's CAT. In [71], Vicent Selfa et al. propose a family of clustering-based cache partitioning policies to address fairness in systems that feature Intel's CAT. Finally, in [72], Lucía Pons et al. propose a simple and effective approach based on identifying critical and non-critical applications that considerably improve the workloads' turnaround times by featuring Intel's CAT.

2.2.2 Cooperative Caching

The proposals presented above mostly apply to shared caches, but some works concentrate their effort on CMPs implementing private caches, or caches that count with both private and shared levels. In this category, cooperative caching approaches [76–78] allow applications with high cache demands to borrow part of the private cache from other (i.e. neighbor) cores. In [76], Chang et al. propose a centralized directory that manages the private L2 caches in a four-core system, allowing cache-hungry applications to borrow cache ways from neighbor cores. This directory is also in charge of keeping coherence and several counters that indicate the reuse of blocks.

The scalability issues observed in the aforementioned solution are addressed in [77], where a distributed implementation of the centralized directory is proposed. This proposal introduces several improvements to the centralized version: the tag management is optimized, reducing the time needed to look up a given block; size and bottlenecks are also reduced, since distributed directories store the tags in an interleaved manner. However, these improvements require extending the underlying coherence protocol, which entails significant hardware modifications.

Finally, the cooperative-caching approach is further refined in [78]. In this proposal, aimed at working both with private and shared caches, authors propose a hardware solution that dynamically adjusts the cache space available for a given core based on the behavior of the running application. The approach overcomes the strict private/shared categories applied to the cache levels and dynamically adjusts the *private size* on each cache module, leaving the remaining cache ways to be occupied by shared blocks belonging to other executing threads.

Cooperative caching solutions are an interesting approach to address the different cache requirements that the co-running applications can present. However, the performance of

these approaches is limited by the interconnection network, which can increase latency when borrowing cache space from distant cores. Therefore, proposals falling in this category highly restrict the maximum cache space that is available for a single core. This fact limits the potential performance gains, especially in processors with a significant amount of cores.

2.2.3 Other Approaches

This section discusses other approaches that have been proposed dealing with existing cache hierarchies, and some works that combine and improve previous approaches. In this regard, we first focus on approaches targeting Non Uniform Cache Access (NUCA) architectures and on schemes that redefine the complete cache hierarchy.

With the aim of reducing the latency and energy consumption of large caches, some works explored in the past solutions focused on NUCA architectures [79–86]. Conventional NUCA architectures organize the cache space as a set of interconnected banks (i.e. modules), which are typically accessed in an interleaved manner. This *statically* interleaved access to the cache banks brings well-known shortcomings associated to rigid cache hierarchies like multiple tag look-ups, excessive data movement and thrashing.

An interesting piece of research that aims to deal with these issues is NUCA-Substrate [83]. In this work, authors propose a dynamic mapping that keeps the most used blocks in banks close to the running application and the least used blocks are kept farther away, hence improving the average access latency. In NUCA-Substrate, a cache hungry application may occupy a high number of NUCA banks, which are located close to the tile executing the application, while applications that barely performs cache accesses are left with a reduced cache space.

Similarly, works like [84–86] also concentrate on placement/migration policies in NUCA organizations. In [84], a distributed cache design that reacts to different types of cache accesses and places blocks accordingly is proposed. This approach requires cooperation from the OS to avoid modifying the coherence mechanism at the last level cache. In [85], authors propose a NUCA architecture that dynamically controls the amount of cache space that can be shared by continuously estimating the effect of increasing the available cache size on performance. Similar to these approaches, ESP-NUCA [86] dynamically

allocates banks closer to the owner processor while also integrates victims and replicas, reducing average on-chip access latency and maximizing cache usage.

The discussed approaches and prior research addressing cache efficiency problems demonstrate that NUCA approaches help reduce the on-chip access latency and that cache partitioning techniques are able to provide isolation. However, neither NUCA techniques work well with hotspot interferences nor cache partitioning approaches significantly reduce access latency. To address these shortcomings, in [87, 88] Daniel Sanchez et al. propose an evolution of existent NUCA architectures and distributed hierarchies, letting the software define virtual caches over the existent hardware. For this purpose, in [87] it is presented a hybrid software-hardware approach that defines, by software, collections of cache bank partitions that act as virtual caches, and gives the software control over both data placement and capacity allocation. The proposal is further improved in [88], where they introduce Jenga, a reconfigurable cache hierarchy that adapts itself to applications. In Jenga, not only the cache space is distributed, but the whole cache hierarchy is adapted to the applications behavior, for instance, eliminating accesses to unnecessary cache levels.

2.3 Energy Consumption in the Cache Hierarchy

In the last decade, energy consumption has become a major design concern in the cache design since caches occupy a significant percentage of area in the processor die. As a result, many works have been developed aiming to address this issue. A comprehensive survey of some of these architectural techniques to address the energy challenge on caches can be found in [89]. This section discusses the main existing approaches by grouping them in two categories: i) prior solutions to reduce energy consumption in caches; and ii) alternative technologies that can bring important changes to the design of current cache organizations.

2.3.1 Reducing Energy Consumption in Caches

One commonly adopted approach that has been followed to reduce the total energy energy in on-chip caches concentrates on reducing the static energy or leakage of the

cache [90–100]. Targeting this objective, drowsy caches [91] allow putting individual cache ways in a low-power mode. In [92], authors extend this approach and propose a circuit technique that supports a super-drowsy mode in the cache using a single V_{DD}. Following this research line, S. Petit et al. [95] studied how temporal locality impacts on drowsy policies and proposed a new policy that makes use of the reuse information to trade off performance with energy consumption. In [101], Powell et al. propose Gated-V_{DD}, an approach that does not only gate the supply voltage of SRAM-cells but also dynamically reconfigures a resizable cache according to the application needs. DRG-Cache [93] combines an integrated circuit with an architectural level technique that reduces leakage power while it avoids destroying the data on a custom standby mode of operation.

The aforementioned works are considered *state-preserving* techniques, since they save the state of the blocks that occupy the ways which are turned off. In contrast, some other researchers have focused on *state-destroying* leakage control mechanisms [102–108], a more aggressive approach. Kaxiras et al. [102] introduce Cache Decay, an approach that invalidates and turns off individual cache lines when it is estimated that the stored data is not likely to be used. In [103] authors propose a hardware technique similar to Cache Decay that dynamically adapts the cache size to the application load, turning off some cache lines. Other works [106, 108] also explore this research line, and they propose to adapt and reconfigure the cache based on the cache behavior of running applications, achieving significant leakage reductions.

State-destroying approaches generally achieve higher energy savings than the state-preserving ones. This is an expected result since they do not need to refresh the data in low-power or standby operation modes. However, for applications with high locality that make a significant use of the cached data, these approaches become ineffective, because the cost of fetching back an old block is high and incurs a significant penalty. Therefore, there is a trade-off between performance (and indirectly, dynamic power) and leakage power when designing leakage-reduction techniques. Additionally, most of these solutions work at the granularity of cache line, and their implementation complicates the on-chip circuitry and makes them impractical.

Finally, some researchers [109–111] have also studied the effect of temperature on the power consumption and leakage policies. In [111], authors propose a thermal-aware

cache that implements a power-down technique to minimize the power density of the active parts in the cache. Considering the relationship between leakage power and temperature, authors in [110] propose a system-level leakage power model and a voltage scaling technique that adapts V_{DD} to the temperature.

2.3.2 Alternative Technologies

Finally, in this section we discuss some proposals that leverage novel technologies to reduce energy consumption in on-chip caches. An interesting comparison of cache technologies that have been employed for last-level caches can be found in [112].

A low-power alternative cache cell technology is Domain Wall Memory (DWM) [113–116], which inspires the emerging *racetrack memories* (RM). DWM is a spin-based memory technology in which several bits of data are densely packed into the domains of a ferromagnetic wire. The significant density shown by DWM is promising and may lead to important speed and energy advantages with respect to conventional caches. TapeCache [114] introduces the first attempt to use a DWM-based memory as last level cache in a general purpose processor. In [116], Wang et al. extend the approach and propose a ring-chaped racetrack memory to be used as the L2 cache in 4-core CMP.

Magnetic RAM and Spin-Transfer Torque RAM (i.e. MRAM and STT-RAM) [117, 118] are memory technologies that present fast read access, high density and non-volatility. In [117] authors propose a 3D-stacked MRAM-based L2 and reduce the long write latencies to avoid harming performance. Similarly, authors in [118] study STT-RAM memory cells to reduce their high dynamic energy and the slow write latencies.

2.4 Summary

This chapter has summarized current approaches working on traditional cache hierarchies. These approaches have identified a poor use of the cache space and an excessive energy consumption in the cache hierarchy.

Existing approaches have typically addressed these limitations but working on a *rigid hierarchy*. Even those works that could be considered adaptive cache organizations are

strongly limited by the underlying NoC, which reduces the potential benefits of these approaches. Unlike this research, in this thesis we claim that most of these shortcomings come by design from the underlying hierarchy, thus we follow a more risky and disruptive direction by redesigning the cache hierarchy from scratch.

Chapter 3

Experimental Framework

This chapter presents the working environment and most of the tools that have been used throughout this dissertation. We first describe Multi2Sim, the simulation framework that has been employed to implement and evaluate the proposals. Next, the chapter presents some tools that have been helpful for evaluating the proposed solutions, e.g. energy-modeling tools like CACTI 6.5. Finally, we show the benchmark suites that have been chosen to compare and evaluate the studied systems and approaches.

3.1 Simulation Framework

The proposals presented in this thesis (see Chapters 4, 5 and 6) use novel technologies and hardware prototypes that are not implemented in current commercial processors. Due to this reason, they have been modeled and evaluated using simulation software packages, namely Multi2Sim, DRAMSim2 and CACTI. Below, these tools are explained.

3.1.1 Multi2Sim

The Multi2Sim [119, 120] simulation framework has been chosen as the most indicated implementation and evaluation platform in this thesis. This framework is an accurate cycle-by-cycle execution-driven simulator for CPU-GPU heterogeneous computing. Multi2Sim faithfully models superscalar pipelines that support out-of-order execution, a complete memory hierarchy, a CPU-GPU compatible coherence protocol and an interconnection layer. It supports different architectures that range from conventional CPU architectures like Intel x86, MIPS-32 or Arm to more recent GPU architectures like AMD Southern Islands or NVIDIA Fermi. Since this dissertation focuses on hybrid technologies applied to CPU environments, we have significantly extended the Multi2Sim source code for the CPU x86 architecture, including the memory subsystem and the interconnection layer. On the top of these features, Multi2Sim can be easily extended to model new hardware components and to integrate them in existing software packages.

Multi2Sim divides the simulation into four different phases, namely: i) disassembly; ii) emulation; iii) timing/detailed simulation; and iv) visualization. The disassembly stage decodes instructions from a specific ISA into an interpretable representation of the instruction fields. Regarding the emulation stage, Multi2Sim is an *application only* emulator, since it focuses on running the user application and does not emulate in detail the OS and device drivers. Finally, the detailed simulation stage, also referred to as *timing/architectural* simulation, models the behavior and timing of different processor and memory hardware structures like pipeline stages, registers, caches and network buffers.

All the experimental results presented in this thesis have been obtained with version 4.2 of Multi2Sim. To make this dissertation self-contained, in this section we present the

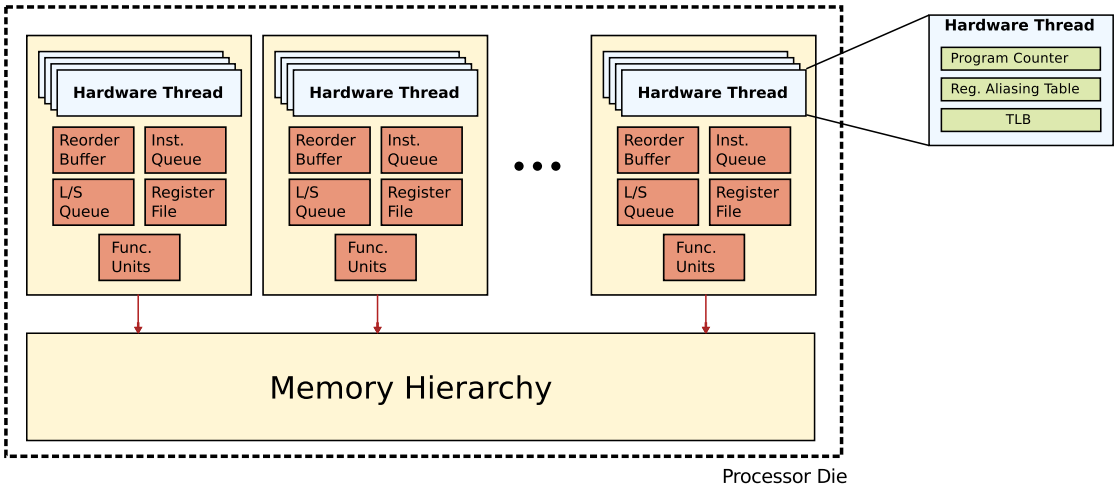


FIGURE 3.1: Multi2Sim model of a CMP implementing N superscalar cores.

most interesting features of the Multi2Sim framework and the main modifications that have been carried out to support and evaluate our proposals. More information about the Multi2Sim framework can be found in `http://www.multi2sim.org/downloads/m2s-guide-4.2.pdf`.

3.1.1.1 Chip Multiprocessor

Multi2Sim models superscalar, multithreaded and multicore CPUs. For multicore architectures, the processor-related hardware structures (i.e. processor pipelines, registers, etc.) are replicated, and they work simultaneously in every execution cycle. Figure 3.1 shows an example of how a CMP that implements any number of superscalar cores is modeled in Multi2Sim.

As shown in the figure, different structures are shared and replicated in each core. In Multi2Sim, each core can run a given number of hardware threads, which share the main core resources like the ROB or the functional units; these resources are replicated for each core. Additionally, each thread counts with its own private resources such as the TLB or the program counter; these resources are replicated for each thread in each core. Each running sequential application or process, referred to as *context* in Multi2Sim, can be mapped to any hardware thread in any core. In addition, software threads spawned by parallel or multithreaded applications are also treated as Multi2Sim contexts. For instance, when running a 4-thread parallel application in a 2-core processor with support

to 2 SMT hardware threads per core (SMT2), a pair of the software threads are mapped to one of the cores and the rest to the other core.

Since this thesis pursues to improve CPU performance and energy efficiency by exploring hybrid technologies like photonics interconnects, the research has mainly focused on the memory subsystem and the interconnection network layers. Therefore, no significant modifications have been implemented at the processor-related structures, which means that the previously exposed processor model is the one that has been used in all the experiments presented in this dissertation.

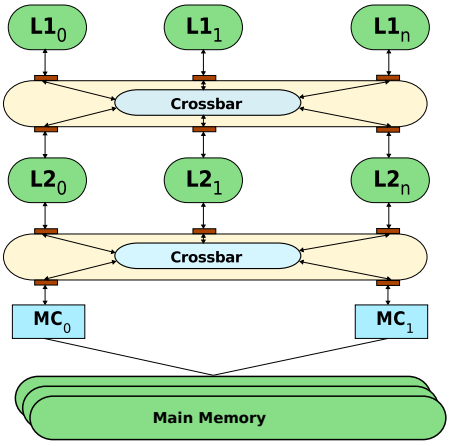


FIGURE 3.2: Model of a 2-level memory hierarchy in Multi2Sim.

3.1.1.2 Cache Hierarchy

Multi2Sim also allows configuring the memory hierarchy in a very flexible manner. Cache levels are recursively defined, without limits regarding the number of levels. Cache modules can be configured with multiple parameters, defining them as private or shared, varying their geometry (e.g. size or associativity), and configuring if the access is address-interleaved among modules and the level of interleaving.

Figure 3.2 presents an example of a cache hierarchy modeled in Multi2Sim. In the example, those accesses that miss in the L1 caches are forwarded to the next lower level in the hierarchy, using an all-to-all crossbar. When an access misses in the LLC, it is forwarded to main memory, from where the block is fetched after a given latency.

In this thesis, major modifications and extensions have been implemented in the memory subsystem level of Multi2Sim to model the thesis' proposals. These modifications are detailed in Chapters 5 and 6, where these proposals are presented.

3.1.1.3 NMOESI coherence protocol

Multi2Sim implements the NMOESI cache coherence protocol, an extended approach of the well-known MOESI protocol [121], typically implemented in a majority of conventional chip multiprocessors. Cache lines managed by conventional MOESI protocol can be in one of five states, (M, O, E, S and I), while the NMOESI approach incorporates an additional non-coherent state (N) used mainly in GPU architectures. Cache lines that are in the N state are considered non-coherent, and they do not generate any coherence traffic like invalidation messages. Thanks to this added state, a given memory hierarchy can be shared by CPU and GPU cores in an heterogeneous system.

Throughout the development of this thesis, the NMOESI protocol implementation in Multi2Sim has been thoroughly studied and optimized. Features like *critical word* support and optical transmissions for coherence-related messages have been implemented [46]. Additionally, the proposal discussed in Chapter 6 incorporates new hardware structures that require some minor modifications to the implementation of the coherence protocol. These slight changes are further explained in the chapter, concretely in Section 6.2.2.

3.1.1.4 Interconnection Layer

The proposals presented in this dissertation leverage photonics interconnects for communication purposes. This novel technology is in an immature state and, while it counts with significant potential in terms of latency and high density, is is not still implemented in commercial processors. For this reason, a simulation environment able to accurately model the latencies achieved by optical interconnects is needed to measure the potential benefits of this on-chip communication technology.

Multi2Sim implements a flexible model of interconnection networks that offers a wide range of possibilities when communicating hardware structures. The model allows defining custom networks by indicating their topology, nodes, switches, links and routing algorithm. Additionally, Multi2Sim also allows communication over bus architectures, fully-connected crossbar switches and virtual channels to solve deadlock situations. These capabilities are enough to define the interconnects in any level of the memory hierarchy, but some important features like *cut-through switching* or *flow-control* are still missed.

The interconnection layer has been significantly modified and extended during the elaboration of this thesis. Optical interconnects are now fully supported in Multi2Sim, including optical arbitration techniques like token-based arbitration. Details regarding the implementation of an accurate model of an optical NoC in Multi2Sim can be found in Chapter 4.

Modifications carried out in the network layer also include the extension of the basic *Store-And-Forward* (SAF) message (i.e. packet) switching mechanism originally implemented in Multi2Sim. In the store-and-forward packet switching, the whole packet is stored in the input buffer before being forwarded to the next node. This requires to wait for the complete packet storage, even when the output channel is free. Past research [122] demonstrated that forwarding (when some conditions are met) the first bytes of the packet to the next node as soon as they are received reduces the delay and improves the average latency. The result of this research is widely known as *cut-through* switching, a method that divides the messages into smaller transmission units that can be forwarded as soon as they are received. Extending this approach, the *wormhole* [123] switching method divides the messages into *flits*, i.e. flow control digits, which are the minimum transmission unit that can be sent in a single network cycle. In order to make the baseline systems evaluated in our proposals more realistic, *virtual cut-through* (VCT) and *wormhole* (WH) switching have been implemented, although VCT has been the most used switching method in our experiments.

Finally, in order for Multi2Sim to account for latencies of optical devices in a cycle-by-cycle basis, the frequency domains of the simulation framework have been adjusted, which allows synchronizing both optical and electrical transmission networks working in the same device at different frequencies.

3.1.2 DRAMSim2

Although Multi2Sim provides realistic models for every on-chip component mentioned before, it neither models the memory controller nor the main memory. In its stand-alone version, Multi2Sim models main memory latencies as a constant and fixed value, which may produce inaccurate results [124], for instance, due to not modeling memory controller contention. This unrealistic behavior is not acceptable as it puts at risk the potential conclusions of this work, where many of the proposals involve the memory subsystem. To overcome this shortcoming, we use Multi2Sim together with DRAMSim2, a dedicated cycle-accurate main memory simulator that includes realistic models for DRAM memory controllers, DRAM memory modules and the buses that these modules use for communication purposes. The modeled memory controller, which is a key component regarding performance due to the effect of contention at its queues, includes mechanisms like the DRAM Command Ordering Scheme or the Row-Buffer Management Policy. Commercial DRAM devices, like the ones implemented in DIMM modules, are modeled in detail, including its internal organization (i.e. multiple banks working in parallel). Additionally, DRAMSim2 allows grouping the DRAM devices into ranks, as it is done in real hardware.

Apart from the internal structure, DRAMSim2 also models the commands to access the DRAM banks. For instance, three of these commands are *Precharge*, *Activate* and *Read/Write*.

We integrated DRAMSim2 with Multi2Sim in the following way. Upon an LLC miss, Multi2Sim triggers the corresponding request to the memory controller, which is then managed by DRAMSim2. When the memory request is resolved, a Multi2Sim event indicating that the block is ready is scheduled, and the block is then forwarded to the LLC.

3.1.3 CACTI 6.5

CACTI [125] is a cache/memory access time, cycle time, power and area model. This analytical tool is useful for studying the latency, power, cycle-time and area trade-offs that are inherent to cache design. CACTI supports the following features:

- Modeling of the cycle time, area, delay and power for caches, including direct mapped, set-associative and fully-associative caches.
- Both multi-ported UCA caches and multi-ported, multi-banked NUCA caches are supported.
- A model of leakage power, considering the operating temperature.
- It supports technology nodes of 90, 65, 45 and 32 nm.

As input, CACTI takes the cache capacity, the cache line size, the cache associativity, the technology generation, the number of ports and the number of independent banks. With this information, CACTI obtains the corresponding cache configuration that minimizes delay or other optimization parameters such as area or power. CACTI models up to eight important components of the cache: deccoder, wordline, bitline, senseamp, comparator, multiplexor, output driver and inter-bank wires. For these components, delay, power and area are modeled, and together they compose the output of the tool.

In this thesis, CACTI 6.5 has been used, which is a specially improved release for large-scale caches and fixes many bugs from prior versions. This software has been employed to obtain the energy results of some of the proposed approaches (see Chapter 6), and also to estimate the access time for tag and data arrays.

3.2 Benchmark Suites

All the proposals discussed in this dissertation have been evaluated using a wide set of benchmarks. Two main suites of benchmarks have been used: SPEC CPU 2006 [126] and SPLASH-3 [127]. The former suite has been selected as the main suite for evaluating sequential applications, while the latter has been chosen when running experiments devised for multithreaded applications. Next we give a brief explanation about the contents of each suite.

The SPEC CPU 2006 benchmark suite contains a set of CPU-intensive benchmarks, which can stress the system's processor, memory subsystem and compiler. Developed by the Standard Performance Evaluation Corporation [128], the suite contains applications belonging to the High Performance Computing field, including both integer and floating

point benchmarks. Recently, the SPEC corporation has released an updated version of this suite, the SPEC CPU 2017 benchmark suite [129]. Although this suite incorporates new features (e.g. measuring power), according to our experiments and to other works like [130], benchmarks belonging to the former SPEC CPU 2006 suite introduce a higher stress on the memory subsystem. Due to this reason, we have selected the original SPEC CPU 2006 suite rather than the new release. For all the experiments, the SPEC CPU 2006 workloads are run using the *data-ref* input set.

Finally, the SPLASH-3 2017 and the ALPBench 2005 benchmark suites have been selected for evaluating parallel applications. The SPLASH-3 suite is the updated release of the original SPLASH-2 benchmark suite [131]. Applications belonging to this set conform a wide variety of complex parallel applications, that have been continuously used throughout the last two decades. This new release updates and fixes some bugs found in the previous SPLASH-2 suite [131]. In particular, the fixed bugs come from data races introduced by some synchronization optimizations. These data races could lead to unexpected behaviors that translate to non-deterministic, incorrect outputs or even performance errors. For these reasons, we have selected the newer SPLASH-3 suite rather than the original release. For all the experiments, the SPLASH-3 workloads are run using the *simmedium* input set. Regarding ALPBench benchmarks, they are executed using their default inputs.

Chapter 4

Accurately Modeling an Optical-NoC in a Detailed Simulation Environment

Manycore and multicore architectures can take advantage from the capabilities provided by silicon photonics technology to reduce their network latency and, as a result, boost performance. However, since on-chip photonics interconnects are still a maturing technology, it is difficult to find up-to-date CMP simulation frameworks that allow its modeling. This fact leads to incomplete models of different on-chip optical networks that lack some components or features, which can present results that differ from those of current prototypes. This chapter presents and discusses an example of an optical ring modeled under our extended framework. It also shows results of different simulation configurations to demonstrate how incomplete models impact on the achieved results.

The chapter is organized as follows. First, some background on optical interconnects, focusing on the behavior of its main components, is presented. Next, it is explained how these components have been modeled in our simulation framework and how they can impact on performance and energy consumption. Finally, the developed model is studied with different configurations, exposing the deviations observed in configurations that follow incomplete models.

4.1 Background on Optical Interconnects

Optical on-chip interconnects bring new opportunities to the design of future manycore and multicore architectures. Already existent implementations and prototypes suggest that different technologies like Free Space optics with Vertical-Cavity Surface-Emitting Lasers (VCSELs) or silicon with ring resonators offer low loss, low latency and high bit density interconnects, able to overcome the main drawbacks of their electrical counterparts [132]. In this chapter, we focus our efforts on modeling and studying ring resonators-based interconnects.

Despite the promising benefits, to fully leverage the potential advantages offered by nanophotonics interconnects, it is critical to count with efficient physical components able to manage the light signal. Promising steps have been accomplished towards achieving the full integration of optical devices, although still some challenges remain specially at system and device level [132]. As a result, advances in silicon nanophotonics currently allow the prototyping and development of functional optical networks in a single chip [133]. This section presents and describes the main silicon photonic devices needed to build an optical interconnect, a working example using these devices, and the main communication schemes followed in the design of current ONoCs.

4.1.1 Silicon Photonics Devices

To be functional, any interconnect requires a power source, a transmission channel and operating devices. In the concrete case of silicon photonics interconnects, these elements correspond to lasers, waveguides and ring resonators.

- **Laser.** A laser source is required to inject light into the transmission channel. Off-chip placed lasers are usually the most adopted solution, since they ease the design, do not suffer signal loss due to the device integration in silicon and also simplifies wavelength locking [15, 16]. Chapter 2 further discusses the different options regarding power sources in on-chip optics, as well as the different solutions available in the literature.

In photonics interconnects, the light signal from the laser is multiplexed into different wavelengths. The number of wavelengths in a waveguide, explained below, can be increased with Dense Wave Division Multiplexing (DWDM) [134].

- **Waveguides.** Waveguides are the transmission medium employed to carry the light signal between the laser and the operating devices. Waveguides employed in prototypes are typically 0.5 μm wide and usually made of crystalline silicon and silicon oxide, which present refractive indexes of 3.4401 and 1.4298 respectively. Then, the speed of propagation of light in silicon is obtained by:

$$c = \frac{3 \times 10^8}{\frac{3.4401 + 1.4298}{2}} = 1.23 \times 10^8 m/s$$

  Therefore, throughout this dissertation, it will be assumed that the speed of propagation of light over the silicon die is $12.3mm/100ps$.

- **Ring resonators.** Microring resonators are ring-shaped waveguides that filter a specific wavelength. Ring resonators are coupled to a waveguide and, when they are on-resonance, they remove the light signal of its resonant wavelength from that waveguide. On the contrary, when resonators are off-resonance, they barely affect the light signal, so the signal can continue its way through the waveguide. A ring resonator can be brought into and out of resonance either by applying an electrical pulse to it or by adjusting its temperature, which modifies the index of refraction of the ring. By default, the wavelength filtered by a resonator is given by its diameter, which usually ranges from 3 to 5μm [135], although resonators can be tuned dynamically to filter different wavelenths.

  Given this behavior, ring resonators can be used as diverters, detectors or injectors, which are key operating devices in photonics interconnects.

  - *Diverter.* A resonator acts as a diverter when it is coupled to a waveguide while it is on-resonance. Then, the resonant wavelength is filtered by the resonator and eventually dissipated by the losses in the ring. Therefore, by bringing the ring resonator into and out of resonance, it is possible to modulate data by encoding 0s and 1s like the absence or the presence of light. For this reason, resonators acting as diverters are also referred to as modulators.

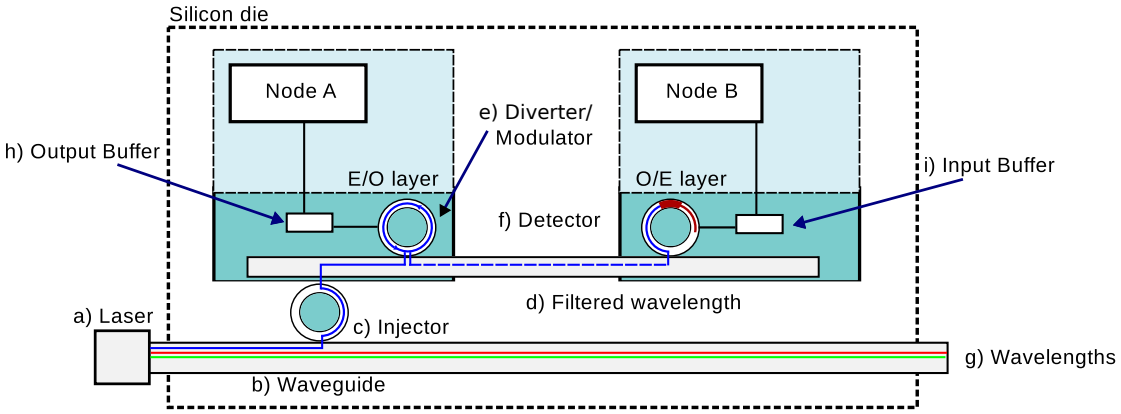


FIGURE 4.1: End-to-end transmission between two network nodes using photonic interconnects.

  - *Detector.* A detector is a ring resonator that has been doped with Germanium (Ge), which makes the light signal to be converted into an electric pulse as it goes through the resonator. Detectors can therefore be used as translators of the light signal to the electrical domain, and can be also referred to as photodetectors in the literature.
  - *Injector.* Injectors are resonators that are coupled between two parallel waveguides, which makes the resonator to inject a given wavelength from one waveguide to the other.

- **Splitters.** These devices, which can be referred to also as *couplers*, split a fraction of the laser power between two different waveguides, This fraction of the light signal is split across all wavelengths, hence the unsplit fraction is unaffected by the device. Splitters provide the capability of distributing power through different paths, allowing more complex designs and topologies.

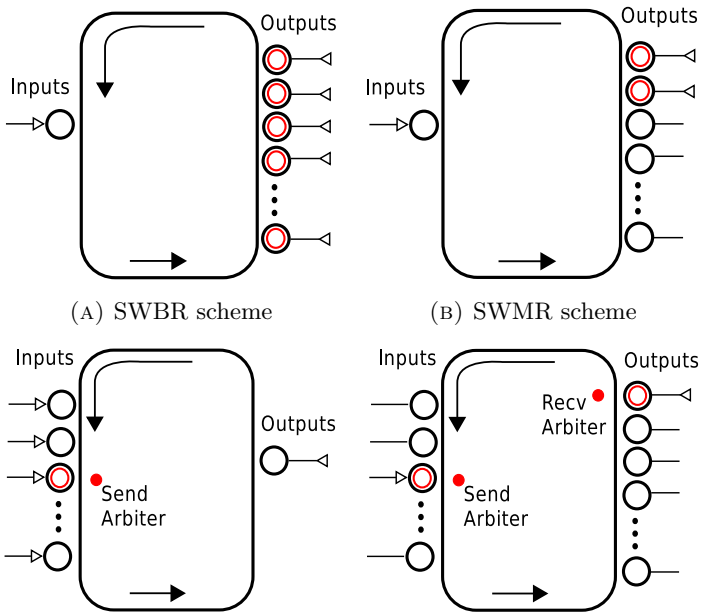
4.1.2 Working Example

Figure 4.1 shows an example of an end-to-end transmission between two network nodes A and B using photonics interconnects. In the example, node A transmits a bit flow to node B following these steps:

- **Step 1:** The laser introduces the light signal, multiplexed in several wavelengths, into the main waveguide (letters b and g in the figure). As observed in the figure, the laser is located outside the silicon die.
- **Step 2:** The wavelength identified by the blue color, henceforth λ_i, is extracted by the injector and introduced into the parallel waveguide, which will be the transmission medium to connect both nodes. Ring resonators corresponding to nodes A and B are placed next to this waveguide.
- **Step 3:** Node A starts the transmission of a bit flow, which is stored in its output buffer (letter h in the figure).
- **Step 4:** Once the light signal corresponding to the wavelength λ_i reaches the diverter of node A (letter e), it is absorbed by the ring and modulated according to the bit flow previously stored in the output buffer. The modulated flow is re-introduced in the waveguide.
- **Step 5:** The light signal eventually reaches node B and passes through its ring resonator. In order to act as a photodetector, this ring has been doped with Germanium, hence converting the light signal to an electrical one. Photodetectors typically require the use of an amplifier to strength the outcoming electrical signal, although for the sake of simplicity, this component has not been included in the figure.
- **Step 6:** Finally, the initial bit flow is recovered in the input buffer of node B and the transmission is completed.

4.1.3 Communication Schemes

As explained above, optical transmissions are carried by a shared transmission medium, that is, the waveguide. In order to communicate several nodes using a single shared channel, different communications schemes are defined to avoid collisions and to guarantee the successful delivery of messages. These schemes define what nodes act as senders (or *writers*) and receivers (or *readers*), and thus, how their resonators are tuned to establish the communication. There are five well-known communication schemes [134]:



(A) SWBR scheme (B) SWMR scheme

(C) MWSR scheme (D) MWMR scheme

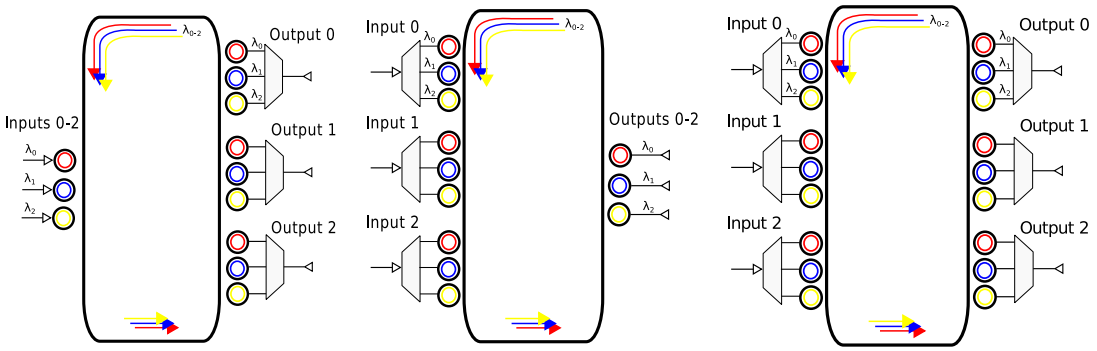
FIGURE 4.2: Communication schemes using a single wavelength for transmissions.

- *Single Writer Single Reader (SWSR)*: SWSR is the basic scheme that is followed by any tranmission between peers. When using this scheme, a single sender and a single receiver have their resonators statically tuned to the same wavelength and the communication is one-way from the writer node to the reader node.

- *Single Writer Broadcast Reader (SWBR)*: The SWBR scheme is used to perform a broadcast among all the receivers in a given wavelength. Broadcast schemes are not desirable in optical interconnects since they involve tuning all the photodetectors to the same wavelength each time that a broadcast transmission is performed. Figure 4.2a presents a simple block diagram of this communication scheme.

- *Single Writer Multiple Reader (SWMR)*: Under this scheme, the sender is in charge of tuning the resonators of the receivers. This scheme is similar to the SWBR scheme, but in SWMR only the nodes interested in receiving the message are tuned by the sender. The block diagram of this communication scheme is presented in Figure 4.2b.

- *Multiple Writer Single Reader (MWSR)*: The MWSR scheme is used when more than one sender have one common destination. This scheme requires using an arbitration technique to guarantee that the shared medium is only used by one of



(A) SWMR (B) MWSR (C) MWMR

FIGURE 4.3: Communication schemes using DWDM.

the senders at a time. Figure 4.2c presents the the block diagram of the MWSR scheme.

- *Multiple Writer Multiple Reader (MWMR)*: The MWMR scheme allows every possible communication between any source node and any destination node. Therefore, arbitration is also needed in order to avoid collisions in the network. This scheme is the one that provides the highest flexibility in the network design, although it implies the use of a high number of optical resources. Its block diagram is presented in Figure 4.2d.

4.1.3.1 Dense Wavelength Division Multiplexing

The Dense Wavelength Division Multiplexing (DWDM) technology allows multiplexing the light signal coming from the laser into a wide range of independent wavelengths. This means that a waveguide carrying a light signal multiplexed into λ_n different wavelengths is able to transmit λ_n different bit flows in parallel. According to the literature, state-of-the-art resonators can filter ranges that are between 64 and 160 wavelengths per waveguide. These numbers, however, are expected to change with the evolution of these devices, since the current ranges of wavelengths achievable by DWDM directly depends on the width of the light spectrum that resonators can filter.

This technology, therefore, is a key component in the design of ONoC, since it allows extending the previous communication schemes to communicate several nodes in parallel by using different wavelengths. Using a subset of wavelengths to communicate a

group of nodes, for instance, can be useful in the design of a network that needs to implement multicast transmissions. Figure 4.3 presents three of the previously explained communication schemes using DWDM to multiplex the light signal into three different wavelengths. A potential application of the MWSR scheme would be communicating the nodes with the memory controller, since only one destination node is reached.

4.2 Modeling the Components of an Optical Network-on-Chip

Due to the possibilities that optical interconnects bring to address communication and design issues in future CMP architectures, a significant amount of theoretical models and simulation frameworks have been developed for this field.

However, as indicated in Section 4.1.1, silicon nanophotonics interconnects present a wide variety of brand new components that behaves completely different from their electrical counterparts. In addition, this technology is still growing and, consequently, new alternatives and solutions that change the existing models are proposed on a regular basis.

Therefore, in order to properly identify the advantages and disadvantages of optical interconnects, it is critical to count with reliable and updated models of their components and behavior. To achieve a reliable model of these components, they need to be studied to identify their critical properties and their impact on performance and energy consumption. This analysis, together with a table summarizing the main features of each optical device, is presented below.

Laser

Since the light signal has to be strong enough to reach all the operating devices involved in a transmission, lasers are critical components for the energy consumed by an optical interconnect. As explained in Chapter 2, on-chip lasers are still under development and current prototypes still suffer from high signal losses and a high dissipation ratio [13, 14]. Off-chip lasers, on the other hand, are currently considered a more suitable solution whose power budget is kept between 1 and 5 Watts, but that still depends on waveguide

characteristics like effective refractive indexes, turns, couples, splits, etc. Therefore, we assume that laser devices are placed off-chip in our simulation environment, and compute its total power according to the waveguide design and the losses introduced by the remaining components. Finally, notice that system performance is not affected by the modeling of laser devices. That is, the execution time is not affected by the laser characteristics, assuming that the laser is powerful enough to support all the system interconnects.

Waveguides

Waveguides are the transmission medium for photonics interconnects. They have an impact on system performance because of two of its main features: i) the achieved light speed over the silicon die; and ii) the optical path length. Regarding the former feature, as stated in Section 4.1.1, the light speed over the silicon die is assumed to be $12.3mm/100ps$, and so it is in our model. Notice that using a different value for this parameter would impact on the communication latency and, as a result, on the system performance. With respect to the optical path length, which depends on the number of interconnected nodes, the chip dimensions and the chosen topology; it is a value that has a direct impact on the required laser wattage and, ultimately, in the consumed energy. For instance, in the baseline system presented in Section 4.3.2, a 576 mm^2 CMP [44] formed by 16 tiles and one memory controller (*i.e.* a network that interconnects up to 17 nodes) requires a 116 mm path length. Therefore, waveguides are key components for optical networks, since they may affect both performance and energy.

Ring Resonators: Modulators and Detectors

Modulators and photodetectors are components that can also affect the communication latency and the system performance, since they are in charge of electrical-to-optical and optical-to-electrical conversions. In our simulation environment, both modulators and receivers are modeled as components that take 1 cycle at a given frequency to transmit 1 bit over a single wavelength. According to the literature (see Chapter 2), state-of-the-art modulators present a switching time by 100 ps. Therefore, we model these devices assuming by default that they modulate the light signal at a frequency of 10 GHz. Similarly, photodetectors present latency ranges varying from tens to hundreds

of picoseconds [17]. So, for the sake of simplicity, we take a conservative approach and assume that the latency of receivers matches the latency of modulators.

Finally, notice that both devices work at wavelength granularity, that is, a single modulator/photodetector only modulates/converts the light signal of a given wavelength. This means that, when the light is multiplexed by using DWDM, as many modulators/photodetectors as multiplexed wavelengths are needed to achieve a successful transmission.

Other components: Injectors and Splitters

Injectors are the physical components that are in charge of conveying the light from the laser to the waveguides and also among waveguides. In our simulation environment, the first case is translated to a small delay at the beginning of the execution, since once the laser is turned-on, it provides the light source until the end of the execution[1]. Regarding the second case, since these devices are very similar to modulators and photodetectors, we assume a conservative latency penalty of 1 cycle (indicated as a parameter in Multi2Sim configuration file) each time the light signal passes through an injector in the optical path. Performance, therefore, is barely affected by injectors and splitters. Energy consumption, however, can be compromised since passing through injectors makes the light signal weaker, introducing losses by 0.1 to 1 decibels. In other words, the higher the number of injectors in the optical path, the higher laser's wattage would be needed to sustain the signal strength.

Unlike injectors, splitters do not present any impact on performance, since they are passive devices that only split or couple the light signal. However, splitting the signal also introduces losses in the optical path, typically ranging from 1 to 4 decibels. This means that increasing the number of splitters throughout the optical path also impacts on the overall consumed energy.

DWDM and Communication Schemes

The DWDM technology has a strong impact on performance since increasing the available number of wavelengths in optical transmissions is the equivalent of increasing their

[1]Due to our methodology of evaluation, this latency penalty is hidden by the *fast forward* stage of the execution.

Element	**Impact on system performance**	**Impact on energy consumption**
Laser	Null	High
Waveguide	High	Medium
Modulator	Medium	Low
Detector	Medium	Medium
Injector	Low	Medium
Splitter	Low	Medium
DWDM	High	High

TABLE 4.1: Summary of the main modeled components and their impact on performance and energy consumption.

bandwidth. That is, for a 64-wavelength multiplexed signal, the sender node can transmit up to 64 bits in parallel per cycle. Therefore, a suitable waveguide model should include the number of multiplexed wavelengths, since this is a critical value for the communication latency and the system performance. Our model supports the DWDM technology, and the available number of wavelengths per waveguide is required to perform any transmission.

Finally, DWDM-based communication schemes (see Section 4.1.3.1) define the possible communications between the interconnected nodes, that is, they only set the limits of the network behavior. This means that their impact on performance is not as high as that of other components like the number of wavelengths. However, notice that, apart from the simple SWSR and SWBR schemes, all the remaining options require arbitration to access the shared wavelengths [44]. As a result, in order to properly model an ONoC it is necessary to define an arbitration mechanism to manage situations where several nodes have to send or receive a message using the same wavelengths simultaneously.

Summary

Table 4.1 summarizes all the aforementioned elements and how much they impact on performance or energy consumption. The table categorizes the impact of these elements into 4 categories: *High*, *Medium*, *Low* and *Null*. As observed, almost all of the studied components present a significant impact on the overall consumed energy, since they introduce losses in the optical path which requires to increase the laser power. Regarding performance, waveguides are the component that present the highest impact on it since they give the achieved light speed over the silicon. On the other hand, the laser is

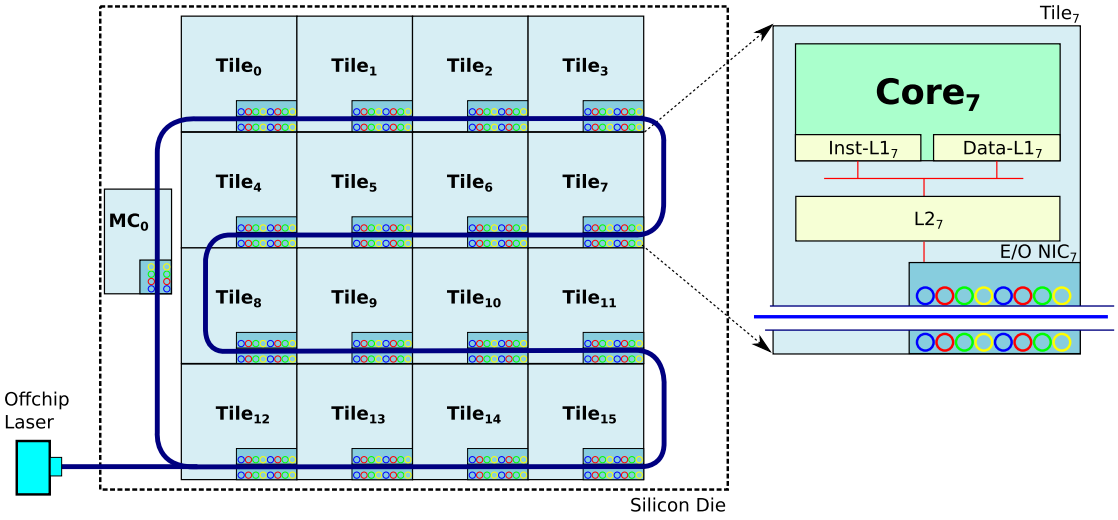


FIGURE 4.4: Block diagram of the evaluated system.

the component whose impact on energy consumption is considered the highest, for it is in charge of carrying the light signal through all the operating devices overcoming the introduced losses. Finally, DWDM has been considered as the most important feature of photonics interconnects, since multiplexing the light signal increases the number of resonators needed to filter each wavelength (increasing the energy consumed) but also increases the aggregated bandwidth, which has a positive impact on performance.

4.3 Studied System: CMP with ONoC

This section presents the baseline system and the different setups proposed to evaluate the modeled optical components. In order to point out the potential deviations regarding performance or energy consumption, we propose a baseline system that includes almost all the aforementioned components, together with a simple optical-based arbitration technique.

Figure 4.4 shows the block diagram corresponding to the devised baseline system. The proposed system is a 16-tiled CMP, where each tile consists of an out-of-order core with private L1 and L2 caches and the network interface to access the communication layer. In this chapter, we employ photonics interconnects to communicate the LLC, that is, the L2 in this example, with the memory controller that is in charge of managing the main memory accesses. Notice that, when executing sequential applications under this

setup, the compute cores only exchange messages with the memory controller and not between them, which simplifies the network design.

A ring-shaped path is chosen as topology for the devised optical network. Ring-based topologies are usually chosen in on-chip designs since they are simple and can be drawn onto the silicon surface without introducing significant signal losses due to bends or crossings. Furthermore, these topologies do not involve using optical switches, which are still immature components in the ONoC field.

The proposed optical ring is composed of two separate waveguides. One of the waveguides is devoted to messages sent from the cores to the memory controller, so it transmits request and writeback messages from the LLC to main memory. The other waveguide operates in the opposite direction, managing the response messages (i.e. acknowledgements and data blocks) sent from the memory controller to the LLC modules. We refer to these waveguides as channels C_0 and C_1 respectively. This two-waveguide design is due to two main reasons: i) on the one hand, it prevents non-critical requests and notifications from delaying the deliveries of data blocks; and ii) on the other hand, it eases the design of the optical communication schemes since it removes the need of arbitration in the memory controller side.

Since channel C_0 is in charge of transmitting request and writeback messages from multiple nodes to a single memory controller, it suits a Multiple Writer Single Reader (MWSR) scheme. This implies that two nodes can not perform a request to the memory controller at the same time and that an arbitration mechanism needs to be devised. We further discuss the chosen arbitration mechanism in Section 4.3.1.

Symmetrically, channel C_1 adopts a Single Writer Multiple Reader scheme, where the memory controller acts as the sender node while the LLC modules do it as receivers. As previously explained, this scheme does not require from any arbitration mechanism to transmit a message, but it requires tuning the destination node's resonators before sending it. To this end, several techniques can be used, like sending an electrical signal to the corresponding node. In this chapter, however, we adopt another existent solution which consists in sending a small optical token to the destination node indicating it to activate its resonators. Since they are of small size, these tuning messages only require one additional wavelength to be transmitted.

4.3.1 Optical Token-based Arbitration

Channel C_0 adopts a MWSR scheme to communicate the compute cores with the memory controller. In a MWSR, any node can send a message to the single destination (that is, any node can write to the channel) but only one node can receive it (that is, read from the channel). To safely write to the common channel, one of the multiple writers needs to acquire rights to transmit on it. In this work, we adopt the Optical Token Channel scheme proposed by Vantrease et al. [44] to perform the arbitration tasks in channel C_0. This proposal is inspired by the 802.5 Token Ring LAN standard [136].

The token channel proposal is a simple mechanism where the nodes that request to send a message exchange an optical token to gain access to the common channel. A node that removes the token has exclusive access to the channel and may write and send a message on it. When the node finishes its transmission, it reinserts the token into the channel allowing other requesters to gain access to the channel. Therefore, in token channel, only one source can use the channel at any one time.

We implement the Optical Token Channel arbitration in our simulation framework, introducing a dedicated threshold to limit the maximum number of messages that can be sent by a node without releasing the token. Notice that, if only one message can be sent upon acquiring the token, even when a node has several pending transmissions, the token has to be released after finishing every single transmission to check if there are other nodes waiting for the token. In our implementation, a node can hold the token up to Thr_N transmissions, reducing the average number of times that the token is released per node. Moreover, the threshold also prevents some nodes from continuously holding the token causing other requesters to suffer long latency penalties or even starvation. When there are no nodes requesting access to the channel, the token is kept travelling in the path waiting for a requester to absorb it.

4.3.2 Experimental Setup

This section presents the baseline system setup together with the absolute latencies modeled for every component, including caches and optical devices. Experiments have been performed using our extended Multi2Sim simulation framework (see Chapter 3),

which also simulates in detail the out-of-order cores and the memory hierarchy. The Multi2Sim network layer has been widely extended to properly model the optical NoC and the components described in previous sections. In this chapter, we focus on studying the potential performance and energy consumption deviations in sequential applications when incomplete models are used. Experiments have been carried out using the SPEC2006 benchmark suite [126].

We show executions for both individual applications and multi-program mixes to explore how the detailed network simulations impacts on the achieve results. Applications are executed for at least 100M instructions after *fastforwarding* the initial 300M instructions. This *fastforward* stage is done to warm-up caches and to avoid introducing performance differences due to this reason, which would affect the main contribution of this chapter. When evaluating the performance of multi-program workloads, all the applications are kept running until the last benchmark finishes the target number of instructions. Otherwise, the fastest benchmarks would be more affected by contention than the slowest ones, invalidating our results.

Table 4.2 summarizes the main baseline system parameters, corresponding to the setup of the processing core, the memory hierarchy, the optical ring and the main memory latency. Regarding the two former components, we model a 16 core CMP working at a frequency of 3GHz, each one provided with two 32KB L1 caches and a 256KB L2 private cache. As mentioned above, the L2 private caches are connected to the memory controller by an optical ring, which is divided into two different channels.

Latencies of the optical ring are given by the optical path length, the conversion times, the arbitration delay and the available number of wavelengths. The roundtrip latency of the optical ring scales with the length of the optical path drawn by the waveguide onto the silicon. In order to accurately estimate this length, we assume that the proposed CMP occupies a 576 mm^2 squared area, which is given by the core count. In this setup, a single tile has length and width proportional to $1/\sqrt{N}$, being N the number of cores. Therefore, for $N = 16$ on a 576 mm^2 die processor, a waveguide that reaches every node and memory controller has 116 mm length (approximately, for the indicated chip dimensions and the example path presented in Figure 4.4).

Processing Core	
Number of cores	16
Frecuency	3GHz
Issuing policy	Out of order
Branch predictor	bimodal/gshare hybrid: gshare with 14-bits global history + 16K 2-bit counters, bimodal with 4K 2-bit counters, and selection with 4K 2-bit counters
Issue/Commit width	4 instructions/cycle
ROB size	256 entries
Memory hierarchy	
L1 Instruccion cache	Private, 32KB, 8 ways, 64Bytes-line, 2 cycles
L1 Data cache	Private, 32KB, 8 ways, 64Bytes-line, 2 cycles
L2	Private, 256KB, 16 ways, 64Bytes-line, 11 cycles
Photonic Ring	
Topology	Ring
Waveguides	2
Wavelengths	64 wavelengths per waveguide
Frequency	10 GHz
Modulator lat	1 network cycle
Photodetector lat	1 network cycle
Arbitration	Token channel
Phit size	64 bits
Roundtrip lat	14 network cycles on idle
Main Memory	
Latency	Fixed latency: 100 cycles

TABLE 4.2: Baseline system setup. Network cycles account for 100 *ps*, while clock cycles account for 333 *ps*

The overall roundtrip latency of the ring also accounts for modulators, waveguide travelling and receivers latencies. First, the waveguide travelling latency is defined by the optical path length ant the light propagation speed over silicon, as previously explained in Section 4.1. Therefore, for a 116 mm path length and assuming the aforementioned 12.3 $mm/100ps$ light propagation speed, the waveguide travelling time on idle is 12 cycles at a 10Ghz frequency. Then, assuming a latency of 1 cycle for the electrical-to-optical and optical-to-electrical conversion times introduced by modulators and detectors, the overall roundtrip latency becomes 14 cycles at 10 GHz, almost 5 processor cycles for a core working a 3 Ghz. Notice that, since arbitration must be performed before each node-to-controller transmission in our system, the arbitration model also takes into account these conversion latencies in order to get reliable results.

Finally, we define the number of wavelengths in which the light can be multiplexed, which depends on the sensitivity degree of the ring resonators (i.e. the optical technology

development). As mentioned above, most of the state-of-the-art works like [32, 33, 35] on optical networks assume 64 wavelengths per waveguide, hence this is the most typically used value. However, other recent works [17] point out that this figure can grow over 160 wavelengths per waveguide with the current technology development. Therefore, to make this study up-to-date and to include both present and future perspectives, in this chapter we evaluate the benefits of silicon photonics interconnects considering both options. Notice that we consider the number of 64 wavelengths as currently the most realistic approach, but we also analyze the results obtained with 160 wavelengths in order to advance technological constraints related with optical networks and the potential speedups.

The last parameter that we show in Table 4.2 is the main memory latency. Although Multi2Sim can be linked to the DRAMSim2 framework to improve the accuracy of the main memory subsystem, in order to isolate the impact on the results of the studied components, in this chapter we set a fixed latency of 100 cycles to access main memory.

4.4 Experimental Results

This section presents the experimental results achieved by the baseline system under different simulation setups. Notice that the main contribution of this chapter lies on pointing out the deviation of performance results that may rise when evaluating incomplete or unrealistic models of ONoCs. More specifically, the proposed experiments analyze the potential deviation caused by the absence of arbitration and by the total number of wavelengths per waveguide. Additionally, experiments have been carried out with both individual and multi-program workloads, in order to cover a wide variety of scenarios. Finally, we also study the impact of arbitration and the number of wavelengths on the overall power needed to sustain the light source.

4.4.1 Benchmark Characterization

We first analyze the behavior of the studied applications, and characterize them based on their individual memory activity. Notice that those benchmark that access more frequently to main memory due to their higher miss ratios are also those that make a

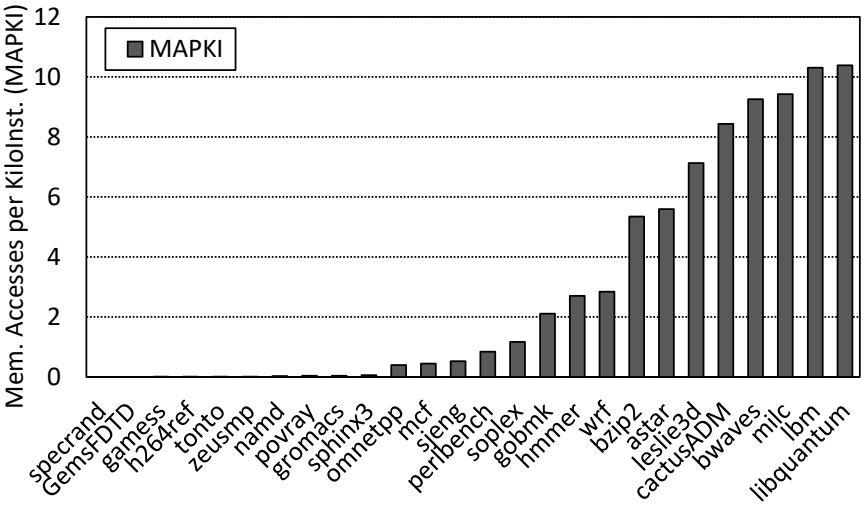


FIGURE 4.5: Memory Accesses Per KiloInstructions (MAPKI) of SPEC2006.

higher use of the underlying interconnection network. Therefore, studying the memory behavior of the individual applications helps understand their performance results both from the memory and the network perspective. In our system, messages can be 8-byte and 72-byte sized: 8-byte messages are devoted to requests and acknowledgements while 72-byte messages are those carrying data, that is, the 64-byte data block together with an 8-byte header.

Figure 4.5 shows the number of Memory Accesses per Kilo Instruction (MAPKI) performed by the studied benchmarks in increasing value order. At first glance, two different groups of applications can be distinguished. Applications on the left side present a low number of memory accesses (i.e. $MAPKI = 2$), hence their performance is not significantly affected by the ONoC latency. These applications store almost their entire working set in the private L1 and L2 caches and, as a result, they scarcely access to main memory. In contrast, applications on the right side incur on a high number of memory accesses, introducing a much higher load and contention on the network layer.

An interesting observation is that, for applications falling in the first *low load* category, arbitration models lacking one or more critical components may achieve accurate results, since performance deviation may be hidden due to the fact that arbitration is scarcely needed. Similarly, other mistakes in the network model can be hidden when the network is saturated because of the contention introduced by *high load* benchmarks. Since we are interested in showing how optical network models lacking arbitration delay can affect

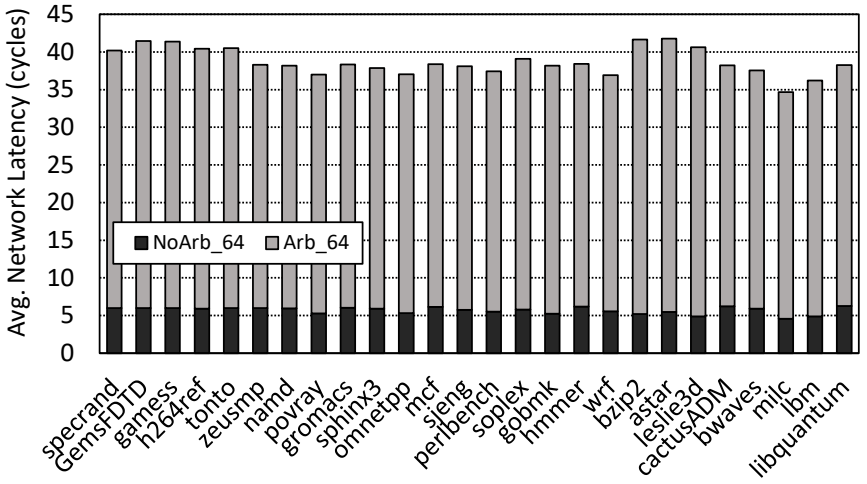


FIGURE 4.6: Network latency of benchmarks executed with and without arbitration delay and 64 wavelengths per waveguide.

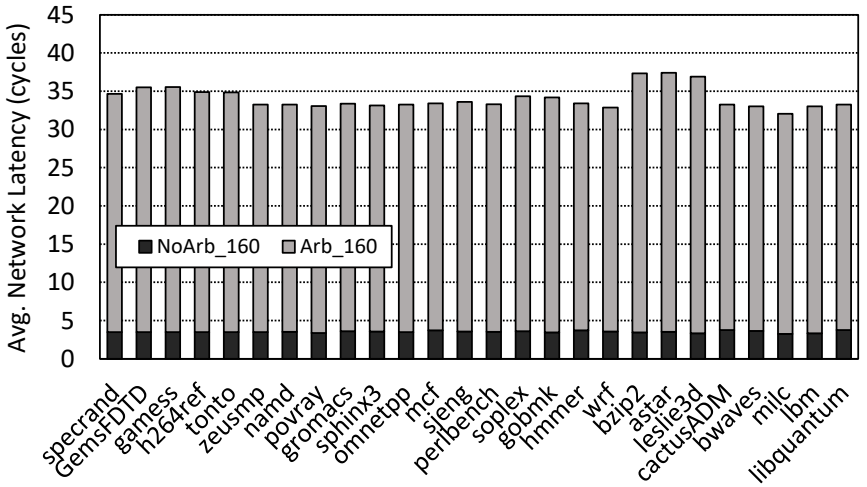


FIGURE 4.7: Network latency of benchmarks executed with and without arbitration delay and 160 wavelengths per waveguide.

the experimental results, henceforth we will differentiate how they impact on these two kinds of applications when discussing the remaining experimental results.

4.4.2 Individual Execution

Different scenarios have been considered to evaluate how the detailed simulation of optical network models lacking one or more components impacts on the achieved performance results. To clearly differentiate the impact of the optical network on the individual application performance, we first run every application in an isolated manner. Nevertheless, since arbitration is a key feature of the optical networks, we take arbitration delay into account even when the application is running alone in the system. This means that, in the experiments performed in this section, messages must wait for a full token roundtrip before being sent to the destiny. By doing this, we emulate the overhead of checking if there are more nodes in the ring also waiting to send a message. Notice that arbitration impact can be critical even when running individual applications because token acquire and release must be performed anyway. Then, as explained in Section 4.3.1, the transmitting node releases the token after sending Thr_N messages, and waits for a roundtrip latency before acquiring the token again. In order to prevent deadlocks, when there are no nodes requesting access to the channel, the token is kept travelling in the path waiting for a requester to absorb it.

Figure 4.6 shows the average network latency of the studied applications with and without arbitration delay and assuming 64 wavelengths per waveguide. The lower frame of the bars refers to the *NoArb_64* scheme which does not model arbitration delays, while the upper frame (*Arb_64*) represents the latency added by arbitration. In the *NoArb_64* approach, messages are not delayed by the mentioned token roundtrip latency, hence they are only delayed when the ring is being occupied by a previous transmission, that is, by contention. As observed, the network latency for this approach is quite homogeneous and close to 5 processor cycles for all the evaluated benchmarks However, this value grows over 35 cycles when arbitration delay is modeled, which means that this introduces a network latency deviation higher than a 6 $\times$ factor on average.

Next, Figure 4.7 shows the latencies corresponding to the same arbitration and no-arbitration approaches but assuming 160 wavelengths per waveguide. Such a number of wavelengths increases the overall network bandwidth, which reduces the cycles needed to send the 72-byte size messages through the ring, although the latency coming from arbitration delays remains constant. Therefore, results are similar to the previously

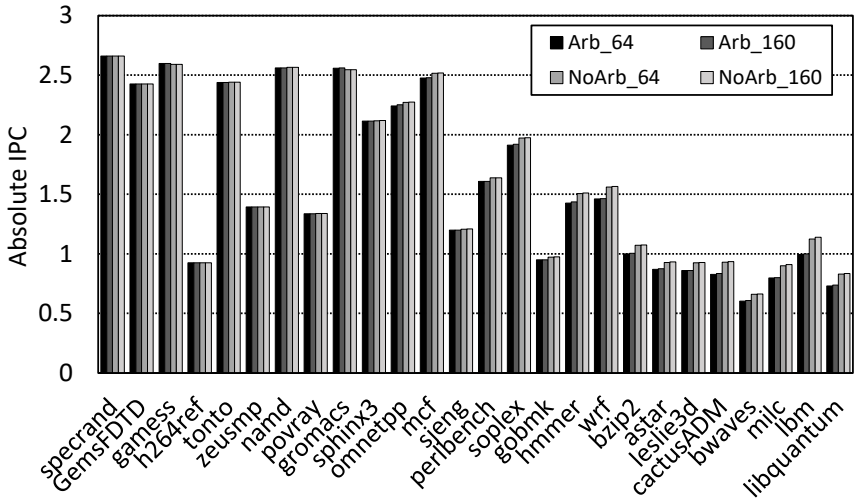


FIGURE 4.8: Absolute IPC of executions with and without arbitration delay in 64 and 160 wavelengths per waveguide configurations.

obtained. The results obtained by these setups show that arbitration delay represents a high percentage of the average network latency, hence significantly increasing the available bandwidth does not translate to relevant improvements on the overall network latency. In this regard, notice that the average network latency for the *NoArb_160* approach is barely 1 cycle less than for the previously shown *NoArb_64*, due to the fact that only big sized messages see their transmission latency reduced.

These results point out the potential latency deviation that experiments carried out with optical models without arbitration can suffer. This deviation is as much as a 10 $\times$ factor in applications like `leslie3d`, `astar` or `bzip2`. However, notice that this network latency deviation may be translated to different system performance deviation depending on the applications behavior and their use of the network.

To clearly expose the deviation in performance derived from the network latency results, Figure 4.8 shows the IPC achieved by the studied approaches. Applications are shown in increasing order of MAPKI, hence the IPC of the benchmarks on the right side of the plot is lower than that of the applications on the left side. Therefore, applications on the right side perform a higher use of the network. As observed, these *high load* applications experiment an artificial increase in their IPCs due to the absence of arbitration delays. On the other hand, *low load* applications like `GemsFDTD` or `gamess` do not present major differences in their results.

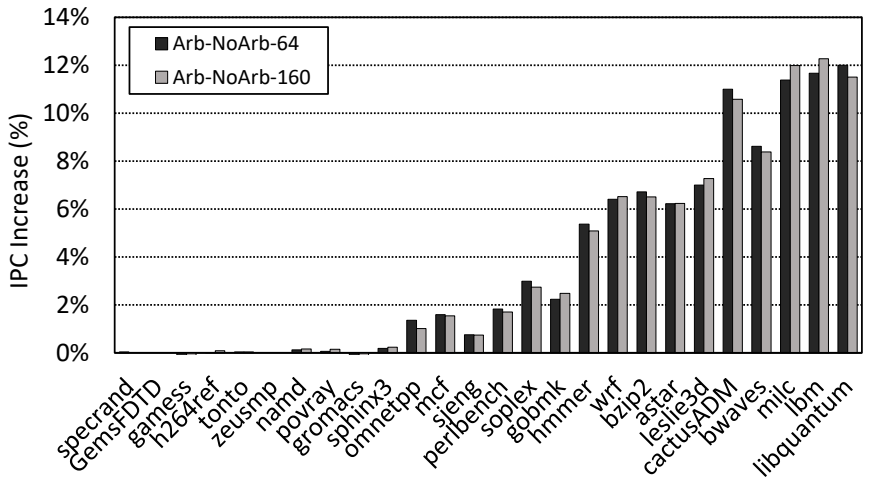


FIGURE 4.9: IPC deviation of executions with and without arbitration delay in 64 and 160 wavelengths per waveguide configurations.

Performance deviations incurred by not modeling arbitration delays in both 64 and 160 wavelengths configurations are shown in Figure 4.9. This figure plots the IPC increase experienced when arbitration overheads are not taken into account; that is, *Arb-NoArb-64* refers to the relation of IPCs of *Arb* and *NoArb* configurations with 64 wavelengths each.

As previously observed in Figure 4.8, applications on the right side show pronounced IPC variations between the *Arb* and *NoArb* approaches while applications on the left side remain with a similar performance. Besides, the highter the MAPKI that the application achieves, the higher deviation its results incur: the 5 applications whose $MAPKI >= 8$ present the highest deviations, farther than 8%. This figure outlines that the absence of a correct arbitration model can suppose an error in the system performance results up to 12%, depending on the number of memory accesses the application performs.

4.4.3 Multiprogram Workloads

The performance deviations shown in the previous section may vary when the resources of the system are shared by several applications. This section presents the results obtained when the applications are executed concurrently with co-runners in other cores of the CMP, with the aim of analyzing the effect of network and memory contention. To study the impact on each application, we replicate the application under study in different cores of the CMP. We refer to these setups as `2astar`, `4astar`, etc. where 2

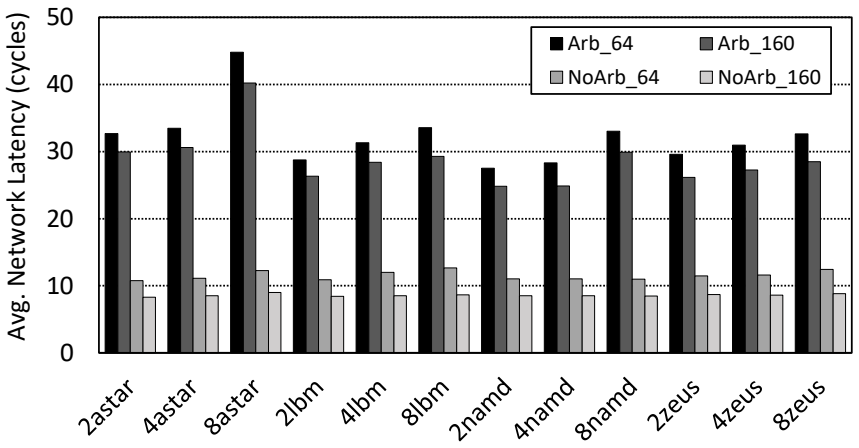


FIGURE 4.10: Network latencies of executions with and without arbitration delay in 64 and 160 wavelengths per waveguide configurations.

and 4 instances of `astar` are co-running. Four different benchmarks have been selected for this study (`namd`, `zeusmp`, `astar` and `lbm`), covering different MAPKI results (from low to high memory stress). The objective behind these experiments is to study the impact of contention on the average network latency and system performance results, since it can hide the deviations shown by previous experiments.

Figure 4.10 plots the average network latency for both 64 and 160 wavelengths per waveguide configurations. As observed, the obtained results present a behavior similar to that of the individual execution. However, because of the effect of contention, the average network latency is increased with the number of corunners. The latency increase is as much as 14 cycles for the pair `2astar-8astar` and 5 cycles for `2lbm-8lbm`, `2namd-8namd`, and `2zeus-8zeus`. An interesting observation is that this network latency increase does not hide the latency deviations between the *Arb* and *NoArb* approaches. In fact, the impact of latency deviation in the average network latency, which varies from a 3× to a 4.5× factor for *Arb* and *NoArb* approaches, is much bigger than the latency introduced by the corunners.

Figure 4.11 shows the deviation in the system performance that rises when arbitration is not considered. It can be seen that the artificial IPC growth due to the lack of arbitration in these executions highly depends on the application. In the case of `namd`, the deviation is almost null since this application does not perform a significant number of memory accesses. On the other hand, `lbm` presents the highest deviation for it is a memory intensive application. In general, as the network contention grows, the IPC deviation

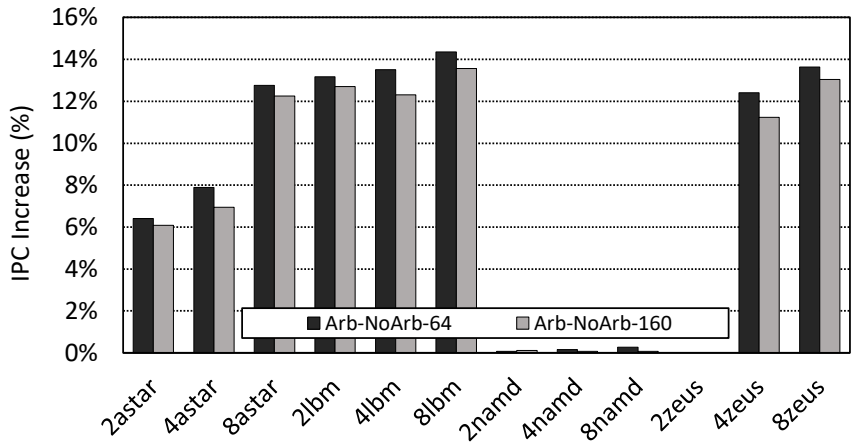


FIGURE 4.11: IPC deviation of executions with and without arbitration delay in 64 and 160 wavelengths per waveguide configurations.

observed in the results is increased. This is an expected result because non-arbitration approaches take less network latency to access memory, so they are less affected by the effect of contention.

Finally, in `zeusmp`, results show that this application is differently affected depending on the contention introduced by the corunners. When this application runs in isolation or with only one corunner, its IPC remains almost constant regardless of the studied approach. However, as the number of corunners is increased, its IPC drops. These results indicate that `zeusmp` suffers from memory contention introduced by other corunners, but the negative impact on performance of this interference can be hidden when arbitration overhead is not modeled.

4.4.4 Power Consumption

This section summarizes the estimated power consumption for the optical components modeled and discussed in Section 4.2. We study every component separately, and take its power consumption from the state-of-the-art studies discussed in Section 2.1.1. To estimate the power consumption, we distinguish between the dynamic energy consumed by modulators and detectors, and the static energy consumed by the tuning of microrings and by the laser. Although in Section 4.2 we assert that off-chip lasers are the more realistic approach nowadays, to perform a self-contained analysis, in this section we include an on-chip laser budget. If an on-chip laser is used, the power model should

Energy_dynamic	
Trasmitters	135 fJ/bit
Receivers	365 fJ/bit
Total	0.42 pJ/bt
Energy_static	
Laser power output	22.5 mW
Microrings Tunning	1.35 mW/ring
Total	22.5 + (1.35 x rings) mW

TABLE 4.3: Energy consumption parameters.

also take into account its static energy consumption. Regarding dynamic energy consumption, the power required by the microring tuning, expressed in femtojoules per bit (i.e. fJ/bit), is shown in Table 4.3. Next we discuss how we estimate the overall power consumption and how it is affected by the arbitration modeling and by the number of wavelengths per waveguide.

First, static energy consumption is defined by the laser and microrings tuning. The energy consumed by the microring tuning directly depends on the number of microrings employed by the optical network. Thus, the higher number of rings, the higher the power required and the energy consumed. The minimum number of microrings needed to achieve the desired communication channels is closely related to the selected communication scheme and the number of wavelengths. Remember that, for the studied approaches, MWSR and SWMR are the communication schemes that are followed by the waveguides Ch_0 and Ch_1 respectively. These schemes imply that, for a given number W of wavelengths, every node must include $2W$ microrings, that is, W rings to send and W to receive. In our system, we set this value to $W = 64$ and $W = 160$.

The energy consumption relative to thermo-optic microring tuning depends on the number of channels that the microring is able to filter and the channel spacing as well. In this chapter, we assume a conservative approach of 50 GHz of channel spacing, which means that the power consumption needed to microring tuning is 1.35 $mW/ring$, as shown in Table 4.3.

Finally, the power consumption needed by the laser source depends mostly on the losses that are introduced by the optical path and that weaken the light signal. In this chapter, we assume an on-chip hybrid silicon laser which presents an injection power of 22.5 mW.

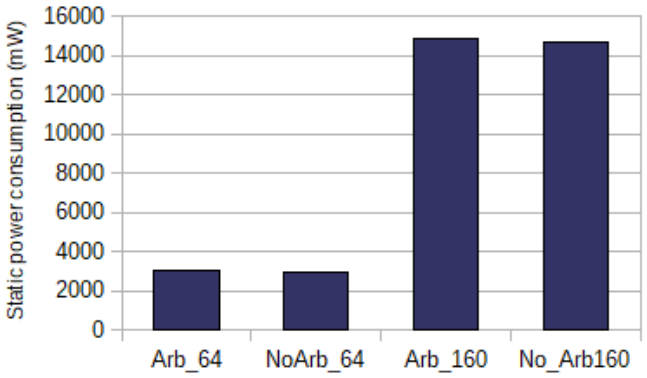


FIGURE 4.12: Static power consumption (mW) required by the four photonic network configurations studied.

Notice, however, that this is the minimum power required to inject light into the waveguide; the final power needed should also take into account the mentioned losses.

Figure 4.12 shows the total amount of static power consumption required by the studied schemes. As observed, the optical rings that employ 160 wavelengths increase their power consumption because of their higher number of microrings. Therefore, future systems with higher number of nodes and wavelengths should face the challenge of reducing the power consumption associated to thermal tuning while keeping bandwidth and sustaining performance.

Regarding the effect of arbitration in the power consumption, due to the use of MWSR, token-arbitration only requires one wavelength to pass the token between senders because there is only one possible destination node. Then, only one resonator must be added to each node to guarantee that the token is properly transmitted. Similarly, the destination selection performed in the SWMR channel Ch_0 for the messages sent by the memory controller also requires only one extra wavelength. These additional wavelengths do not significantly affect the overall power consumption, since they only suppose about 3% and 1.23% of the power needed with 64 and 160 wavelengths respectively.

4.5 Summary

This chapter has addressed the importance of developing and counting with complete and up-to-date models when working with novel technologies like silicon nanophotonics.

Every component that conforms a fully operative optical NoC has been modeled in order to obtain accurate, reliable and representative results both in network and system performance. In this dissertation, we have also discussed about the importance of reviewing current state-of-the-art in optical technology, and pointing out realistic and future parameters that may change for some optical components like waveguides, modulators or photodetectors.

Aiming to quantify the deviation that an incomplete optical model could present in a detailed simulation environment, a realistic proposal composed of two optical rings and a simple arbitration technique have been modeled, evaluated and compared against an incomplete setup with no arbitration. Experimental results outline that the deviation observed in average network latency between the two studied approaches can be as high as 1000%. Moreover, this is translated to a system performance deviation higher than a 10% in some cases, both in individual and multi-program workloads execution.

Finally, according to current state-of-the-art power consumption values, the arbitration approach increases the overall network energy consumption up to 3% with respect to the non-realistic setup. However, results point out that microrings are the components that present the strongest impact on the energy consumed, since they consume energy each time they are tuned at a different wavelength. These results, together with the demonstrated performance deviations, show that current simulation frameworks must be properly extended with complete and accurate models in order to obtain reliable results when researching on such a novel and immature technology.

Chapter 5

FOS: A Low Power Cache Organization for Chip Multiprocessors

Most of the current multicore processors typically implement a cache hierarchy that consists of one or two levels of private caches per core and a large shared last level cache (LLC). Although this is the most commonly adopted approach, it presents important design issues like private space over-sizing or data replication. Moreover, conventional cache organizations also exhibit lack of flexibility, which leads to an inefficient use of the total cache capacity. This chapter focuses on addressing these shortcomings by introducing a more energy efficient organization and leveraging silicon photonics interconnects.

The chapter is organized as follows. First, we perform an analysis on the performance sensitivity of the studied applications to the private cache size. Then, the proposed Flat On-Chip Organization (FOS) approach is described and discussed, including both the devised cache organization and the interconnection network. Finally, we present the evaluation setup and the achieved experimental results.

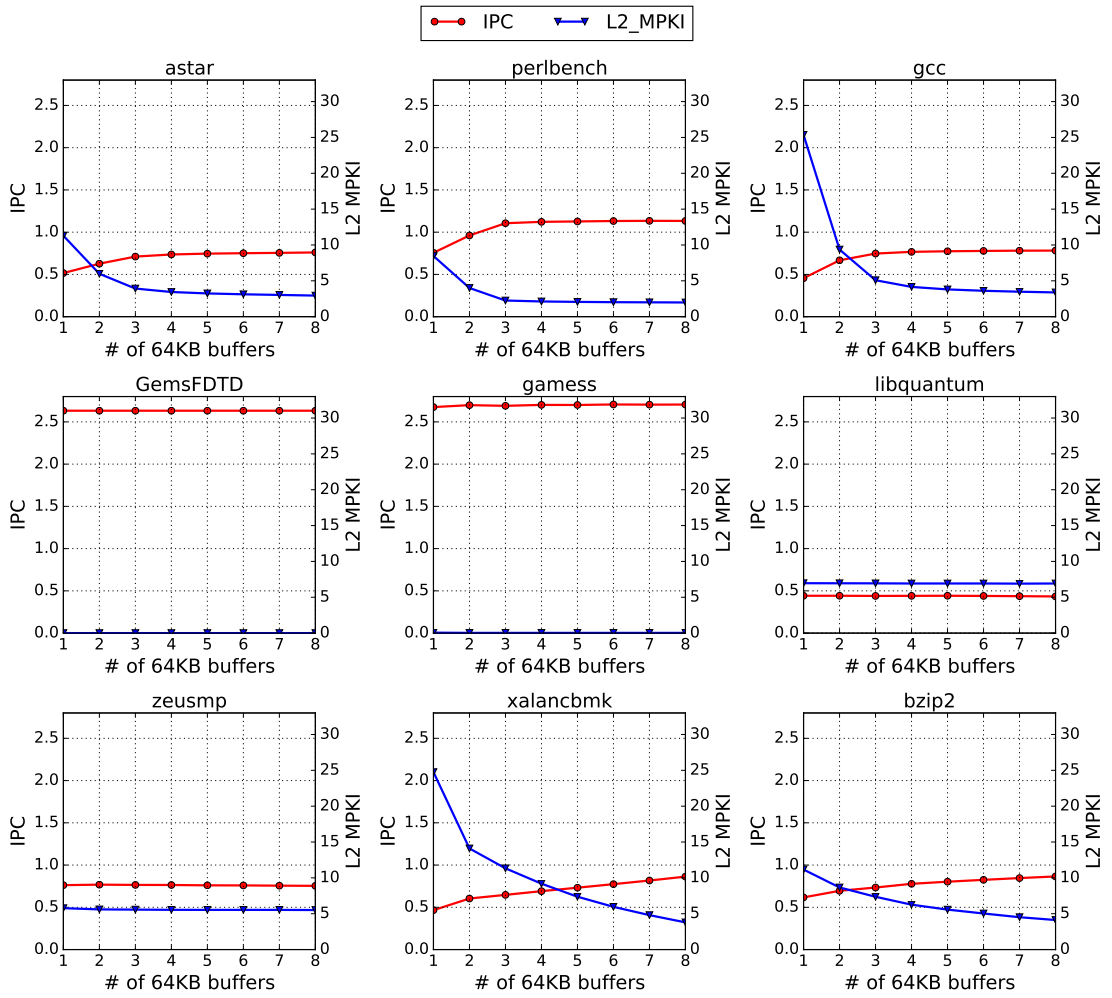


FIGURE 5.1: IPC and L2 MPKI values varying the number of buffers (1 buffer is 64 KB - 8 buffers are 512 KB).

5.1 Cache Demands and Performance Analysis

To guide the design of FOS, this section presents a characterization of a representative subset of the SPEC CPU2006 benchmarks based on their cache demands. Performance is also studied since it can be differently affected by the amount of cache resources, depending on the application behavior. In the following experiments, L2 caches are assumed to be composed of 64 KB independent cache buffers, and the maximum cache space has been fixed to 512 KB (8 cache buffers). The remaining system components are kept like in the baseline system used in the experiments, described in Table 6.4. Notice that this baseline setup slightly differs from that of the previous chapter, as also

does the methodology of evaluation. Therefore, results might present small deviations with respect to those obtained with the CMP studied in Chapter 4.

To characterize the applications, we vary the number of cache buffers from 1 to 8, and measure the performance (i.e. IPC) and the number of Misses Per Kilo Instruction (MPKI) of the L2 cache. The goal of this study is twofold: i) to analyze the relationship between these two metrics and the provided cache space, and ii) to evaluate the potential performance gains of increasing the cache capacity up to the entire 512 KB space for individual applications. Figure 5.1 presents the achieved results. As depicted in the figure, applications can be categorized into three main groups based on how the performance of the studied applications evolves when adding more cache space:

- **Minimum Cache Needs (MCN)**: Applications in this group are characterized by their performance insensitivity to the number of available buffers due to two main reasons. First, some applications, like `libquantum` and `zeusmp`, present an inherent low locality so their MPKI does not change regardless of the amount of available cache buffers. Second, the working set of applications like `GemsFDTD` and `gamess` fits in a single 64 KB buffer, thus their performance scarcely improves or even remains constant when they are provided whit additional buffers.

- **Limited Cache Needs (LCN)**: Applications like `astar`, `gcc` or `perlbench` reach their maximum performance when they are executed with 4 or 5 buffers. In this category, increasing the number of cache buffers over the saturation point does not impact on the application performance.

- **Non-limited Cache Needs (NCN)**: Applications belonging to this group are called *cache-hungry*, since they always improve performance with higher cache capacities. For instance, both in `xalancbmk` and `bzip2` performance keeps growing with the number of cache buffers.

This study shows that a flexible distribution of the cache resources among the competing applications can potentially achieve important benefits. First, it can provide significant energy savings over conventional L2 caches by powering off cache buffers that are not in use, especially for applications falling in the first two categories. Second, in multiprogram workloads, the performance of LCN and NCN applications can be enhanced since LCN

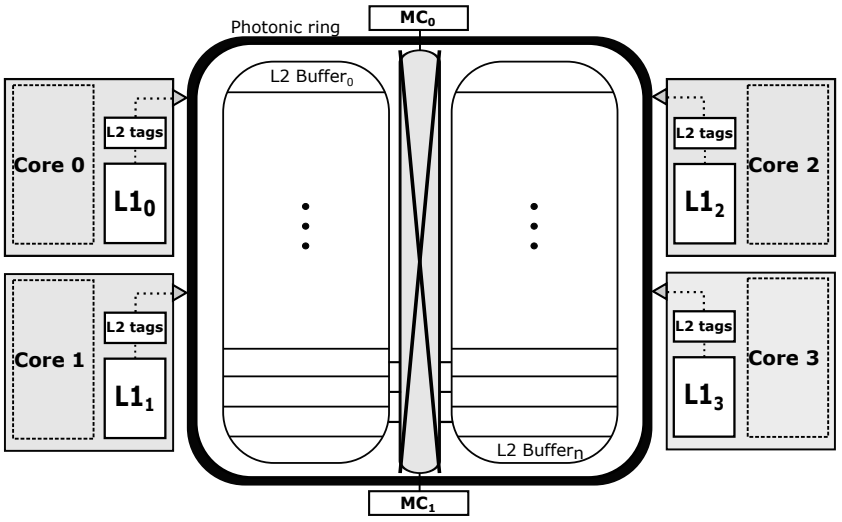


FIGURE 5.2: Schematic of the proposed architecture.

applications benefit from the reduction of interferences and cache trashing while NCN ones can get a high number of buffers. Additionally, applications which, due to the small size of their working set, are classified as MCN can also experience performance improvements. This occurs because providing an isolated cache space can prevent other cache-hungry applications from replacing the blocks exhibiting a high cache-locality of the MCN applications, which is critical for their performance. Finally, low locality MCN applications are expected to present the same performance regardless the cache amount of resources.

5.2 Flat On-chip Storage

The key idea behind the proposal is to have a common pool of cache buffers that replaces all the cache levels (e.g. second and third) except the private L1 caches implemented in the core pipeline. All these buffers should be properly interconnected with the cores to avoid high NoC latency deviations that could negatively affect performance. A high-level block-diagram of this proposal is depicted in Figure 5.2.

FOS buffers are small (i.e. from 32 to 128KB) cache modules that are assigned to specific cores at run-time based on the predicted application requirements; once a buffer is assigned to a core, it is set to private mode for that core. That is, cores allocate cache buffers and use them in a private manner. Having a common pool avoids the

constraint of a fixed-size private cache, which has been identified as a key concern in previous research [50], since it usually incurs area and energy wasting due to over-provisioning the cache space. Previous approaches [102] handle energy savings at a very small cache line granularity (e.g. 64 B), which complicates the layout of wires and increases area overhead, making this approach impractical from an implementation perspective. However, acting on the entire buffer itself as an activation/deactivation unit (e.g. 64 KB) helps the implementation of cache energy saving mechanisms.

A flexible distribution of the cache resources, in which each application is provided with the cache size that it requires for performance, can help improve both energy savings and performance. To provide a good trade-off between energy and performance, however, three main architectural challenges must be addressed:

- When should a buffer be allocated/deallocated to/from a core?
- Which buffer must be accessed by a given core upon an L1 cache miss?
- Which interconnect should be used, considering that buffers are not beside the core?

Different design decisions have been made to deal with these issues. A Buffer Management Mechanism is proposed to detect at run-time when a given application can significantly improve its performance by providing an additional buffer. Next, a new hardware structure, named Private Tag Array, which is in charge of keeping track in each core of the blocks stored in the allocated buffers, is discussed. Finally, we address the interconnection challenge by leveraging silicon nanophotonics, hence we propose using a state-of-the-art optical ring to achieve a rather uniform access time to the cache buffers. These design decisions are further discussed and explained below.

5.2.1 Buffer Management Mechanism

This section presents the Buffer Management Mechanism (BMM), which implements both a *Buffer Allocation Algorithm* and a *Buffer Deallocation Algorithm*, discussed below.

Buffer Allocation Algorithm: This algorithm, summarized in Algorithm 2, assigns buffers to applications based on performance and energy considerations. The algorithm is triggered at fixed-length (X execution cycles) intervals during the application execution.

During each interval, the number of cache misses in the FOS pool is measured to quantify the MPKI of the current interval ($MPKI_{n-1}$). Then, the algorithm predicts if the performance of each application would improve in case the application was provided with one additional buffer. The key challenge is that the algorithm needs to estimate the number of cache misses that each application would have experienced with an additional cache buffer $b+1$, while running with only b buffers. To deal with this issue, we have modeled extra hardware that works similar to the *Auxiliary Tag Directory* (ATD) [50]. To reduce the area of this structure, it only implements a tag array corresponding to 32 sets, which are randomly selected among all the sets in the cache.

Using the collected data, the algorithm estimates, for each interval, how much the MPKI would have improved with an additional buffer. For this purpose, the expected MPKI decrease rate ($MPKI_{dec\_rate}$) of each application with $b+1$ buffers is computed in step 2. This metric alone, however, only considers performance, so it is complemented with the $MPKI_{hist}$ metric, which helps improve energy efficiency by providing the average $MPKI$ during a sliding window composed of the last w intervals. This metric can be used, for instance, to check if adding an extra buffer provides marginal system performance benefits, which might not justify in terms of energy consumption the activation of a new buffer. Lastly, one more metric is computed to obtain the weight of the last interval MPKI in the mentioned sliding window ($MPKI_{weight}$).

Step 3 aims to check the conditions that must be fulfilled for energy efficiency. The $MPKI_{hist}$ must be greater than a minimum threshold (Thr_{window}), and the number of FOS misses ($MPKI_{n-1}$) must be greater than a threshold (Thr_{min}). If none of these conditions is satisfied, Step 4 is skipped and no buffer is activated.

Finally, Step 4 checks the conditions for performance provided that Step 3 conditions have been fulfilled. The $MPKI_{dec\_rate}$ of the application must be higher than the Thr_{dec} threshold, or the weight of the last interval MPKI in the sliding window ($MPKI_{weight}$) must exceed threshold Thr_{weight}. The last condition enables the proposal to react to sharp application phase changes that cannot be detected by $MPKI_{dec\_rate}$.

Algorithm 1 Buffer Management Mechanism.

Algorithm Inputs: n: current interval number; b: buffers currently assigned to the application; w: size of the history window (measured in number of intervals); $\{Thr_x\}$: thresholds.

————-Allocation Algorithm————-

1. **Initialization.** At the beginning of interval n, for each running application, compute its $MPKI_{n-1}$ and $MPKI_{n-1}(b+1)$.
2. **Metric computation.** For each running application, calculate:

$$MPKI_{dec\_rate} = \frac{MPKI_{n-1}(b+1)}{MPKI_{n-1}} - 1,$$

$$MPKI_{hist} = \frac{\sum_{i=n-w}^{n-1} MPKI_i}{w}, \text{ and}$$

$$MPKI_{weight} = \frac{MPKI_{n-1}}{MPKI_{hist}}.$$

3. **Filtration.** Skip step 4) iif:

$$MPKI_{hist} < Thr_{window} \textbf{ or } MPKI_{n-1} < Thr_{min}.$$

4. **Request.** Request a new buffer iif:

$$MPKI_{dec\_rate} > Thr_{dec} \textbf{ or } MPKI_{weight} > Thr_{weight}.$$

———- Deallocation Algorithm ———-

1. **Initialization.** At the beginning of interval n, for each running application, compute its $MPKI_{n-1}$ and $MPKI_{n-1}(b-1)$.
2. **Metric computation.** For each running application, calculate:

$$MPKI_{inc\_rate} = 1 - \frac{MPKI_{n-1}}{MPKI_{n-1}(b-1)},$$

3. **Release.** Release an allocated buffer iif:

$$MPKI_{inc\_rate} < Thr_{inc} \textbf{ and } IddleInts > Thr_{rel}.$$

The proposal requires keeping track of which buffers are assigned to each core. To do so, the *Buffer Allocation Logic* is accessed at the end of each interval through the devised optical network (ONoC). When a buffer is requested, the allocation logic selects, powers on and assigns an empty buffer to the requesting core by sending the corresponding acknowledgment to it. If several requests take place simultaneously, the allocation logic serves them following a simple FIFO order. Note that the allocation logic is simple and only works between intervals upon requests of additional buffers.

Buffer Deallocation Algorithm: The operation of releasing a buffer allows feeding the common pool of free buffers and prevents an application from unnecessarily holding a high number of buffers after a hungry phase. The release mechanism is summarized in the bottom of Algorithm 2. This algorithm aims to estimate whether the performance

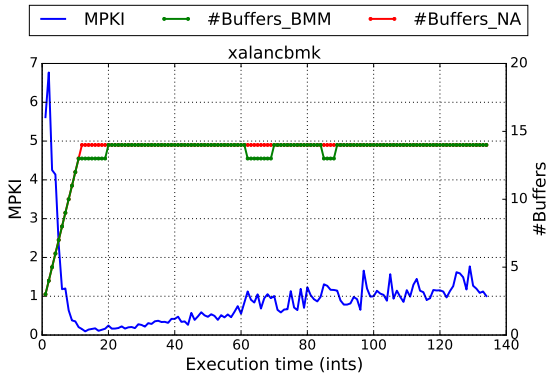
of a given application on a long-enough steady state would remain the same or similar in case that a buffer is deallocated from it. For these estimates, the ATD-based approach is used.

First, on each interval, the algorithm obtains the $MPKI_{n-1}(b-1)$ of the application. This value is used to estimate the number of cache misses that a given application would have experienced during the last interval with one less buffer. Using the value provided by the ATD, we obtain the estimated $MPKI_{inc\_rate}$ metric, which is the ratio that the MPKI is expected to grow in case that one buffer was deallocated from the cache space of the application. The $MPKI_{inc\_rate}$ is compared to the Thr_{inc} threshold and, if it is smaller enough and the application is in a steady state (i.e. it has not asked for any buffer during the last Thr_{rel} intervals), one buffer is released. The Least Recently Used (LRU) policy is employed to choose which buffer is deallocated.

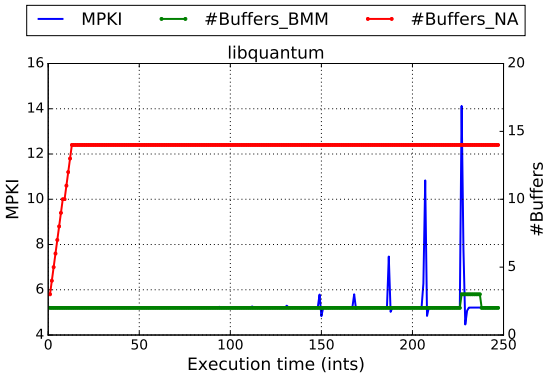
To address performance and energy, FOS tries to reduce the time devoted to release the buffer. To this end, the devised implementation of the draining operation carries out in parallel the two main writeback steps required to safely deallocate and power the buffer off for energy savings: i) writeback those modified blocks in the L1 caches that belong to the target buffer, and ii) writeback those modified blocks in the target buffer. During the draining intervals the dirty blocks are written back to main memory spread over time, while the remaining data are kept accessible to the threads. When the draining phase finishes, the buffer is powered off and becomes available to be assigned.

5.2.2 Buffer Management Mechanism Evaluation

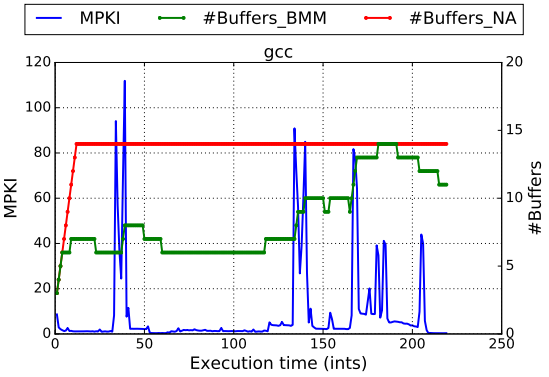
We evaluate the behavior of the allocation/deallocation algorithms of the Buffer Management Mechanism (BMM) across different types of applications. To this end, Figure 5.3 depicts a histogram of the MPKI and the number of buffers allocated to `xalancbmk`, `gcc` and `libquantum` along the execution time. The figure also shows the number of buffers used by FOS under no allocation restrictions (NA). Under this setup, buffers are assigned without restriction as soon as more cache space is required, they are never deallocated and there is no constraint regarding the number of buffers that a single core can allocate.



(A) `xalancbmk` MPKI and buffer allocation histogram.



(B) `libquantum` MPKI and buffer allocation histogram.



(C) `gcc` MPKI and buffer allocation histogram.

FIGURE 5.3: MPKI and allocated buffers for `xalancbmk`, `gcc` and `libquantum` along execution time.

In `xalancbmk`, an application previously identified as NCN, *Non-limited Cache Needs*, it can be observed that BMM allocates buffers from the beginning of the execution until the application gets 14 buffers, which is the maximum number of buffers per core that has been set in these experiments. Notice that the deallocated buffers (intervals 19, 60 and 85) are quickly reallocated by the BMM since MPKI improvements are expected along all the execution time. This means that the devised algorithm correctly identifies the buffer needs for `xalancbmk`.

In `gcc`, an LCN (i.e. *Limited Cache Needs*) application, the algorithm must face different stages during the execution. During the initialization stage (intervals 0 to 40), BMM assigns up to 6-7 buffers to the application. Then, a sharp increase in the MPKI is observed, thus the allocation algorithm increases the number of buffers up to 8. After that, the application returns to a steady state and the deallocation algorithm turns off two buffers, coming back to 6. At the end, the application enters in an irregular stage

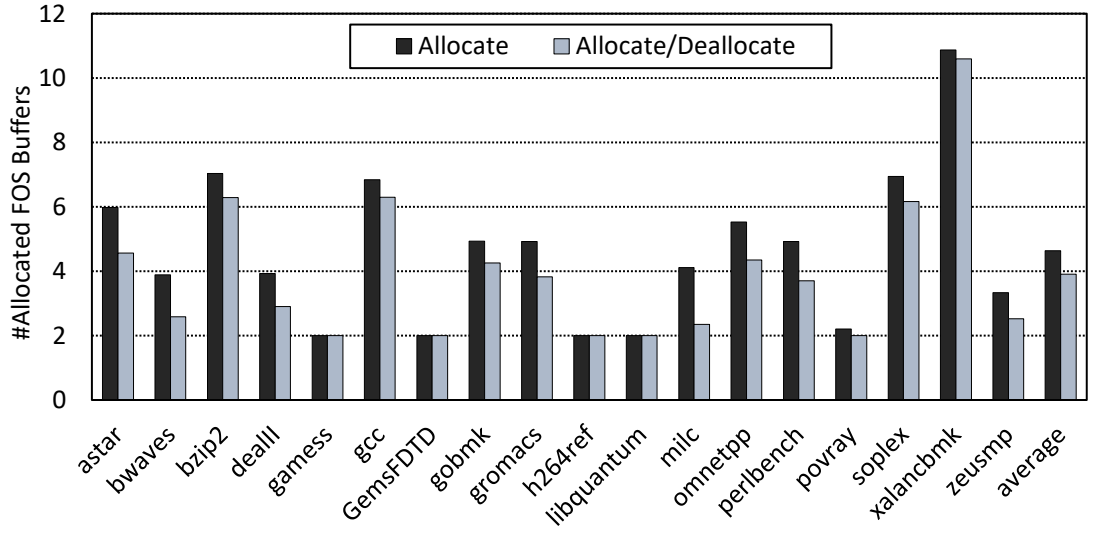


FIGURE 5.4: Average number of buffers allocated with and without activating the deallocation algorithm.

where several MPKI peaks are observed. The final stage is managed by the algorithm by reacting to these sharp increases turning on more buffers, while simultaneously trying to deallocate buffers. This example shows how BMM is able not only to identify buffer needs in different applications but also to react to sharp increases in the MPKI.

Lastly, `libquantum` is studied as an example of a MCN (i.e. *Minimum Cache Needs*) application. As observed, the BMM only allocates two buffers even when the MPKI is higher than that observed in `xalancbmk`. This means that the *ATD-based* algorithm correctly detects that allocating more space for this application is pointless because an MPKI reduction is not expected.

To conclude this study and provide insights on the effect of powering off buffers, Figure 5.4 plots the average number of buffers allocated depending on whether the deallocation algorithm is used or not. As can be seen in the figure, the allocate/deallocate approach reduces in some applications like `astar` and `milc` up to almost 2 buffers the average number of allocated buffers. This directly translates (as it will be shown later in Section 5.5.1) to a reduction of up to 30% in the static energy consumed due to leakage currents. Only those applications that do not allocate any buffers (NCN applications) keep the same amount of cache space under both approaches, which means that energy savings are achieved for any other types of applications. Therefore, activating the deallocation algorithm has a significant effect on the energy consumed by FOS. Moreover,

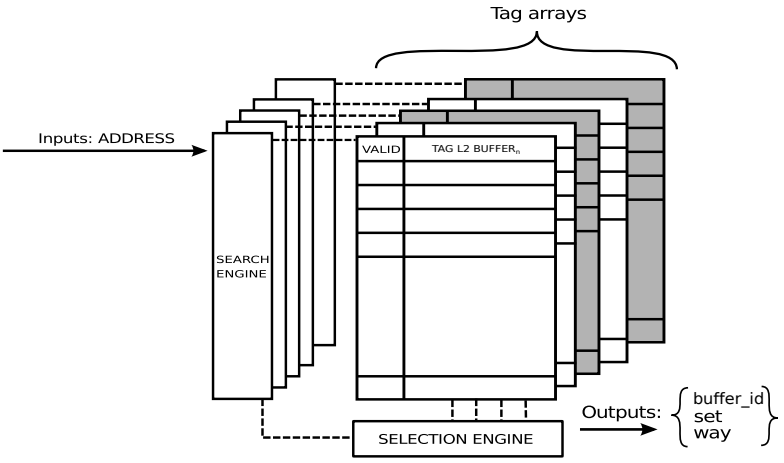


FIGURE 5.5: High-level hardware schematic of a PTA module.

performance is scarcely affected when the deallocation algorithm is active (less than 1% performance degradation in the worst case).

5.2.3 Implementation Issues and Shared Data Support

Private Tag Array: In FOS, an application running in a given core typically has a subset of the FOS buffers assigned. Then, upon an L1 cache miss, the core only needs to check those buffers assigned to it to locate the block in the pool of buffers. In other words, when running sequential applications, it is not strictly necessary to search a block in all the buffers that are powered on. To ease the block location in the allocated buffers, we decoupled the tag arrays of the blocks stored in the buffer pool from their data arrays and moved these tag arrays close to the L1 caches as similarly done in some commercial processors (e.g. IBM POWER5).

The tags of the decoupled buffers are managed on each core by the Private Tag Array (PTA), which corresponds with the *L2 tags* modules shown in Figure 5.2. The PTA block diagram is presented in Figure 5.5. As observed, the PTA is organized in multiple (e.g. $MaxBuffers$) banks, where each bank, when active, keeps the tags associated to one of the assigned buffers. Those banks that are not in use (depicted in darker color in the figure) are switched off. The operation of the PTA is described as follows:

1. The PTA is indexed with the address of the block to be accessed in the FOS pool.

2. The active banks are accessed in parallel by the logic and the results (hit or miss and location information) are notified to the *Selection Logic*.

3. Upon a hit, the Selection Logic provides the buffer, set, and way identifiers where the target block is stored.

4. If the tag is not present in any bank, i.e. on a FOS pool miss, the Selection Logic replaces the LRU line in the LRU buffer. The decision of retrieving the LRU buffer follows a hierarchical LRU approach [137], much simpler than a strict LRU. We assume that LRU control information is stored in the PTA.

Hardware Overhead: The main hardware overheads introduced by FOS are the PTA and the two ATD structures required by the BMM. Below, the overhead associated to these structures is studied. First, a PTA requires as many banks as the maximum number of buffers that a single core can be assigned. Theoretically, since FOS guarantees at least two buffers per core, the maximum number of buffers in a PTA, being n the number of cores, is given by the equation:

$$MaxBuffers = TotalBuffers - 2 * (n - 1)$$

For scalability and hardware simplicity reasons, we limit this value to $MaxBuffers = 12$. Notice that not limiting this value would imply an excesive and unsustainable growth in the PTA size when the core count grows beyond 16 cores. However, limiting the PTA size can not be done at the cost of performance. For this reason, to find the optimal PTA size, we have experimentally explored the effect of this value, and results showed that setting it to 12 buffers allows the system to achieve the maximum performance in almost all the studied cases. Hence, we consider that this is a reasonable value that trades off performance and scalability.

Then, the hardware cost per PTA is obtained by multiplying $MaxBuffers$ by: i) the maximum number of blocks that can be stored in a buffer; and ii) the number of bits in a tag entry (i.e. typically the tag bits plus the valid bit). This is summarized in the following expression:

$$PTA_{cost} = MaxBuffers \times \frac{Size_{buffer}}{Size_{block}} \times Size_{entry}$$

TABLE 5.1: Evolution of size devoted to tag-storing structures in FOS .

	Conventional Cache		**FOS**		
#Cores	**Data Array**	**Tag Array**	**Data Array**	**Tag Array**	**Overhead**
2	1024 KB (95.35%)	50 KB (4.65%)	16 × 64 = 1024 KB (92.6%)	82 KB (7.41%)	32 KB (2.98%)
4	2048 KB (95.35%)	100 KB (4.65%)	32 × 64 = 2048 KB (92.6%)	164 KB (7.41%)	64 KB (2.98%)
6	3072 KB (95.35%)	150 KB (4.65%)	48 × 64 = 3072 KB (92.6%)	246 KB (7.41%)	96 KB (2.98%)
8	4096 KB (95.35%)	200 KB (4.65%)	64 × 64 = 4096 KB (92.6%)	328 KB (7.41%)	128 KB (2.98%)
10	5120 KB (95.35%)	250 KB (4.65%)	80 × 64 = 5120 KB (92.6%)	410 KB (7.41%)	160 KB (2.98%)
12	6144 KB (95.35%)	300 KB (4.65%)	96 × 64 = 6144 KB (92.6%)	492 KB (7.41%)	192 KB (2.98%)

Additionally, the overhead associated to an ATD that leverages *Set Sampling* is up to 32 replicated sets per core [50], which is translated to 960 bytes. Overall, the tag area required for PTAs and ATDs in a 2-core, 16-buffer CMP is 82 KB (39 KB PTA +2 KB ATD on each core), which is 7.4% of the total area (data plus tag array) of a 1 MB shared cache. Compared to a conventional cache (which occupies 4.65% of the total area), this represents a relatively small (i.e. by 2.97%) overhead. To clearly expose this overhead, Table 5.1 lists the different tag capacity of FOS and a conventional cache scaling with the number of cores and buffers. In this table, the tag space of FOS has been computed assuming the aforementioned value $MaxBuffers = 12$ and a 25-bit tag size.

Finally, we discuss the hardware cost associated to the implementation of the Buffer Management Mechanism. First, reduced hardware is needed to compute the four MPKI related metrics (i.e. $MPKI_{n-1}(b)$, $MPKI_{n-1}(b+1)$, $MPKI_{hist}$, $MPKI_{weight}$) used in the algorithm. This hardware compute the metrics with a small set of hardware counters that keeps track of the target event. For instance, two counters are needed to gather the number of committed instructions and the amount of cache misses, which are required to obtain $MPKI_{n-1}(b)$. In addition, five registers are used to store the threshold values. As discussed below, there is no need to use additional logic neither to compute the metrics nor to check the thresholds, since these fast computations can be carried out with the available functional units of the core at the time the core is stalled.

Execution Time Overhead: Note the BMM does not need to act at strictly fixed-length periods of time. Therefore, to mitigate the overhead in time, the logic of the devised approach could be triggered during the core stall cycles (e.g. when the ROB is blocked due to long memory latencies). According to our experiments, the period of time when the processor is blocked, referred to as the core slack time, is on average a

24% of the total time. Then, taking into account the low complexity of the required computations (e.g. just tens of processor cycles every 40 M cycles), there is no appreciable impact on the system performance even if the core is stopped to perform these computations.

Shared Data Support: This work focuses on multiprogram workloads composed of sequential applications. To support the execution of shared memory multithreaded applications, FOS needs to be extended to avoid multiple non-coherent copies of the same block to be present in the buffer pool. To address this issue, FOS can be extended with a small number of distributed directory nodes, similarly to distributed cooperative caches schemes (e.g. [77]). With this structure, after a PTA miss, FOS would be able to notify the accessing core if the block is already present in the buffer pool, so preventing the replication of shared blocks. In addition, the sharer vector and other coherence protocol information could be stored either in the distributed directories or in the buffers. The design of such structures and FOS adjustments for multi-threaded applications can be found in Chapter 6.

5.3 FOS Network-on-Chip

FOS provides a novel and flexible management of a common pool of buffers. This disrupting approach relies on a Network on-Chip (NoC) that must fulfill two main conditions: speed and low latency variability. Speed is needed since most of the accesses that miss in the L1 cache result in (by using the PTA) hits in the FOS pool, thus the network must be fast enough to avoid delaying these hits. With respect to low latency variability we mean that the access to the target buffer must present similar latency regardless of the accessed buffer. Distinct NoCs might be devised fulfilling these conditions, although in this work we focus on *Optical Networks-on-Chip* (ONoCs).

As stated in Chapter 4, current advances in silicon nanophotonics allow the integration of a complete functional optical network on a single chip [138–140]. These networks present several features that meet the aforementioned requirements: first, fast transmissions are possible even with a reduced number of optical resources; second, the impact of distance on the average latency in optical networks is much lower than in traditional

Table 5.2: Optical network-on-chip parameters and latencies.

ONoC Parameters	
Frequency	10 GHz
Wavelengths per channel	32 - 128 - 8 λ
Signal propagation	11.4 ps/mm
Modulation bandwidth	10 Gbps
ONoC Latencies (ps)	
Token transmission	Varying in range [100..500]
Microring tuning delay	400
Data modulation (64b-576b)	100 - 500
Trans. latency	Varying in range [100..900]

electrical networks. Nevertheless, the ONoC must be properly designed in order to avoid a prohibitive increase in the overall network energy consumption.

5.3.1 FOS ONoC

The main features of the devised network are:

- A rather uniform and low transmission latency regardless of the accessed buffer.
- Reduced number of optical resources and network complexity.

Table 5.2 summarizes the modeled ONoC parameters and latencies. Considering the aforementioned network characteristics, a ring topology can be used since it does not involve the use of optical switches and it has an inherent low complexity. Optical rings [45] have been studied in previous research work and have been proved to work with a reduced number of optical components.

With the aim of reducing the data access time, the FOS ring is configured with three separated, spatially multiplexed channels C_0, C_1 and C_2, which interconnect the L1 and the pool of buffers. Channel C_0, provided with 32 wavelengths, is used to send requests and data from L1 caches to the FOS level. Next, the C_1 channel is used to deliver the requested blocks from the FOS buffers to the L1 caches. Incurring a high latency when delivering data blocks to the L1 level can be critical for the core performance so, to provide a reduced latency, this channel is provided with 128 wavelengths. Finally, a third channel is employed to communicate the FOS buffers with the PTA structure

described in Section 5.2.3. FOS buffers use this channel to notify the PTA when writing and replacement operations are finished. Since notification messages are small (4 B), this channel is provided with just 8 wavelengths. Channels use different waveguides, so transmissions can take place in all of them simultaneously, regardless of the tuned wavelengths.

Communication on each channel follows a custom *Multiple Writer Multiple Reader* (MWMR) [134] approach. As previously explained in Section 4.1.3, following a MWMR scheme requires several steps before performing any transmission. First, a sender node must get access to the channel, which is granted by an arbitration logic. In this proposal, we adopt a Token Channel-based arbitration, as similarly done in Section 4.3.1. Token-based arbitration techniques are widely used in optical rings since they guarantee collision-free transmissions and barely introduce extra overhead [44]. The proposed Token Channel-based arbitration requires to acquire 1-bit token before transmitting data, and optical tokens help reduce the token acquiring latency [44]. The implementation of optical tokens only requires an additional wavelength on each channel, and the token transmission latency mainly depends on the silicon lightspeed and the path length. Assuming a signal propagation of light of 11.4 ps/mm [47] in silicon, a 10 GHz ONoC, and a ring length of 44.8 mm, token transmission latencies range from 100 ps to 500 ps, depending on the distance [1]. The ring shape of the proposed ONoC is shorter than that of Chapter 4 because of its oval shape, instead of the much longer *"C"* shape ring path presented before.

After getting ring access, the sender must notify the receiver that a message is going to be sent using a given wavelength λ_i. This action is performed using several extra wavelengths to ask the receiver to turn on its ring resonators in order to read the modulated wavelength. Regarding tuning/detuning latencies, the most recent works like [47] assume up to 400 ps tuning delays.

Once a communication path has been established between two nodes, the data transmission can be carried out. The overall transmission latency (leaving apart communication setup) comes from delays associated to electrical-to-optical conversion, data transmission

[1] Notice that some of the values (i.e. signal propagation of light, ring length) assumed in these experiments slightly differ from those presented in Chapter 4. This is because, despite the previously shown values being theoretically correct, in this chapter we update some of the values that we employed in previous research according to more recent state-of-the-art parameters.

TABLE 5.3: Loss values of the photonic components.

Component	**Value**	**Reference**
Laser Efficiency	5 dB	[47]
Coupler	1 dB	[141]
Waveguide path loss	0.1 dB/mm	[47]
Waveguide bend/cross	0.005/0.5 dB	[47]
Ring drop	1 dB	[141]
By/Through ring loss	0.001/0.1 dB	[141]
Photodetector	0.1 dB	[141]
Receiver sensitivity	-25 dB	[141]

along the waveguide, and optical-to-electrical conversion. As stated in Chapter 4, in our simulation environment, both conversions are assumed to take 1 cycle at the network frequency to transmit 1 bit over a single wavelength. In the studied system, request messages are 8 B long, data messages are 72 B (576 $bits$) long and notification messages are 4 B (32 $bits$) long. Considering the number of wavelengths available on each channel (32, 128, 8) and 10 Gbps conversion speeds, each conversion takes 200 ps, 500 ps and 400 ps, respectively, on each channel. Analogously, the data transmission delay also depends on the length and the width (number of wavelengths) of the optical path. For a 44.8 mm ring and depending on the number of wavelengths, the maximum transmission latencies for 64-, 576- and 32-bit messages are 600, 900 and 900 ps, respectively. These latencies can be lower for path lengths shorter than 44.8 mm.

Overall, for a 2 GHz core clock frequency and assuming no contention, the whole transmission latency (i.e. considering the latencies of tuning resonators, both conversions, and the message size) of any message varies, depending on the distance, between 2 and 4 clock cycles. In short, from an experimental perspective, the NoC latency variability does not exceed two clock cycles regardless of the location of the end-to-end points, hence becoming rather uniform in the studied system. Notice that other alternative networks, like an electrical mesh or a crossbar, might be much slower and present a higher latency variability.

5.3.2 Energy Consumption in the FOS ONoC

To satisfy the latency requirements, apart from including a relatively high number of wavelengths on the C_1 channel, FOS ONoC replicates some optical resources to allow

simultaneous transmissions on channels C_0 and C_2. This section analyzes the energy consumption of these components.

The components that determine the power consumption in an ONoC are mainly the laser and the microring resonators. First, regarding the microrings power consumption, according to previous research [142] we consider 5μW/ring for the thermal tuning of resonators. Although a huge number of microrings performing filtering actions might increase the power loss and the crosstalk noise power, the FOS ONoC does not present this technological constraint because all its rings never work simultaneously. The worst case scenario for FOS occurs when all the three channels are communicating simultaneously and, in this case, only requires a reduced number of tuned microrings.

Second, laser power in ONoCs depends on the total losses that optical devices introduce along the communication path. To estimate the minimum laser power needed to reach all the components in the FOS ONoC, we use the power model provided by Morris et al. in [141], given by the equation:

$$P_{laser} = P_{rx} + C_{loss} + M_s,$$

In this model, P_{laser} is the laser power, P_{rx} is the receiver sensitivity, C_{loss} is the channel loss, and M_s is the system margin. The model is fed with the loss values listed in Table 5.3, and the outcome provides the minimum laser power to carry a signal strong enough to be received by the photodetectors on every node.

The energy per bit for the FOS ONoC has been computed using the average number of transferred bits and execution time across the executed workloads. According to the model, the energy per bit consumed by the ONoC is up to 1.5 pJ/bit. In contrast, the energy dissipation value expected in electrical links is 0.25 pJ/bit (estimated with the ORION 2.0 tool [143]). As expected, the ONoC presents higher energy consumption than conventional electrical links, but it should be noticed that electrical private links only implement point-to-point communications, while the FOS network interconnects every L1 cache to every FOS buffer. Nevertheless, the power savings reached by FOS mainly come from buffer deactivation as it is discussed in Section 5.5.

Table 5.4: Baseline system parameters.

Core	
Number of cores	2 - 4, OoO, 4 issue/commit width
Frequency	2 GHz
ROB size	128 entries
Cache Hierarchy	
L1 Inst-Data cache	Private, 32KB, 8-way, 64Bytes, 2 cycles
L2	Private, 512KB, 16-way, 64Bytes, 8 cycles
Interconnect L1-L2	
Frequency	2 GHz
Bandwidth	64 Bytes/cycle
Main Memory & Memory Controller	
DRAM bus freq.	1066MHz
DRAM device	DDR3 (2133 Mtransfers/cycle) 8 banks
Latency	t_{RP}, t_{RCD}, t_{CL} 13.09ns each

5.4 Experimental Framework and Studied Approaches

5.4.1 Simulation Setup

We have widely extended the code of the Multi2Sim simulation framework [119] to model and evaluate our approach. As stated in Chapter 3, to improve the accuracy of the DRAM memory subsystem, Multi2Sim has been linked to the DRAMSim2 framework [144], which is a hardware-validated DRAM simulator. Also, the CACTI v6.5 [125] tool has been used to estimate the energy consumption and access times of the studied cache structures for a 32nm technology node. Experiments have been carried out using the SPEC CPU2006 benchmark suite. Applications have been executed until they commit at least 500 M instructions after fast-forwarding the initial 300 M instructions. When evaluating mixes composed of multiple applications, they are kept running until the slowest one commits the target number of instructions, but statistics for each individual application are gathered at the time it commits the targeted 500 M instructions. This way prevents some applications to run in isolation during the last part of the execution of the mix, which would result in higher IPC values.

5.4.2 Studied Approaches

The FOS implementation mainly modifies the cache hierarchy and the NoC. In order to study where the achieved benefits come from, the proposal has been compared with four different systems modifying these components; that is, different L2 cache organizations and L1-L2 interconnection networks have been considered. All the four combinations that have been considered will be referred to as a tuple X-Y, where X indicates the L2 cache organization (e.g. shared or private) and Y the underlying NoC technology (e.g. electrical or optical). For instance, *Shared-OPT* refers to a system with an L2 shared cache with optical NoC. Below, the four baseline schemes are discussed.

- **Private-ELC:** This scheme presents 512 KB L2 private caches, connected to the corresponding cores through electrical links. This configuration is used to study the performance constraints associated to fixed-size private caches. Table 6.4 summarizes the main parameters of this baseline system.
- **Shared-ELC:** Unlike the previous one, this scheme presents a unified shared L2 cache connected to the L1 cache level with electrical links. This scheme is aimed at comparing FOS against a common shared space, which exhibits inter-application interference.
- **Shared-OPT:** This configuration replicates the same cache hierarchy as the *Shared-ELC* approach, but electrical links are replaced by an optical ring. The parameters and latency values used in this ring match those of FOS 's ONoC (see Section 5.3.1). This scheme, together with *Shared-ELC*, contribute to discern whether the performance enhancements come from reducing the network latency or from a better performance of the shared organization.
- **NUCA-OPT:** A NUCA-based approach is introduced to replicate the FOS system without the SMM algorithm. For this purpose, this scheme is configured as a pool of n cache buffers of k size, where n and k match the values selected for the FOS setup. This scheme presents the same optical network as FOS. The motivation behind this scheme is twofold: i) exposing clearly the energy efficiency benefits in terms of cache management brought by the SMM algorithm, and ii) comparing FOS against a scheme with faster cache modules and equal network latencies.

TABLE 5.5: Composition of the evaluated 2- and 4-benchmark mixes. Legend: M: Minimum Slice Needs, L: Limited Slice Needs, N: Non-limited Slice Needs.

Mix	4-benchmark mixes	2-benchmark mixes
Mix0	astar (L), dealII (L), milc (M), soplex (L)	astar (L), bzip (N)
Mix1	GemsFDTD (M), soplex (L), xalancbmk (N), h264ref (M)	astar (L), gcc (L)
Mix2	xalancbmk (N), gromacs (L), omnetpp (L), zeusmp (M)	astar (L), xalancbmk (N)
Mix3	libquantum (M), omnetpp (L), milc (M), xalancbmk (M)	gobmk (L), gromacs (L)
Mix4	perlbench (L), povray (M), gamess (M), gromacs (L)	gobmk (L), h264ref (M)
Mix5	omnetpp (L), astar (L), libquantum (M), milc (M)	milc (M), h264ref (M)
Mix6	gamess (M), perlbench (L), omnetpp (L), xalancbmk (N)	omnetpp (L), perlbench (L)
Mix7	gcc (L), libquantum (M), milc (M), zeusmp (M)	povray (M), libquantum (M)
Mix8	astar (L), gcc (L), dealII (L), zeusmp (M)	soplex (L), zeusmp (M)

Finally, **FOS** has been configured in the same way as *NUCA-OPT*, that is, as a pool of n buffers of k size. Experimental results consider k equal to 64 KB and n equal to 8 times the number of cores. In this way, the compared schemes have the same storage capacity, i.e. 1024 KB and 2048 KB for 2- and 4-core systems respectively.

The reason why we select 64 KB sized buffers for FOS is twofold. First, reducing the buffer size below 64 KB increases the demands of networks resources, hence increasing the energy consumption and the hardware complexity. Second, the larger the granularity, the lower the number of buffers (i.e. the pool size) that BMM has available to distribute, hence the lower the flexibility of the allocation and the deallocation algorithms which can also affect the performance. Taking these considerations into account, we experimentally concluded that 64 KB is the most suitable buffer size for evaluation purposes.

Regarding the BMM algorithm setup, a wide set of experiments has been performed to tune the BMM parameters. To illustrate the potential of FOS, this work presents the results with the parameter values that showed the best energy efficiency on average through most of the experiments; that is, Thr_{min}=0.2, Thr_{window}=0.8, Thr_{dec}=0.25, Thr_{weight}=1.5, Thr_{rel}=25. Finally, w and the interval length have been set to 10 intervals and 40k cycles, respectively.

5.4.3 Design of Multiprogram Workloads

To evaluate our approach, experiments have been launched for applications running alone and for multiprogram workloads in conventional 2-core and 4-core multicores. Experiments with more cores are not required since, as explained in Section 5.5.3, our

approach assigns the same cache space to each application regardless of the number of co-runners. In other words, since applications obtain always the same amount of cache as long as there are enough buffers for every running application, the benefits of FOS are expected to be similar when scaling the number of cores. The multiprogram mixes were randomly generated varying the ratio of applications belonging to each one of the three categories identified in Section 5.1. The list of studied multiprogram workloads for 2-core and 4-core experiments is presented in Table 5.5.

5.5 Experimental Results

This section analyzes the impact of the devised FOS's cache management and buffer distribution on energy and performance of multiprogram workloads. By design, FOS assigns buffers exclusively as private to cores, which means that, provided that there is enough cache space, FOS will assign the same cache space to each application as it assigns in individual execution. In other words, FOS behaves similar for each application in multiprogram workloads as it behaves in individual execution in terms of performance and energy. This behavior is a key issue in the design of FOS, for it is aimed at scaling with the number of cores or concurrently running applications.

5.5.1 Energy Consumption of Multiprogram Workloads

This section evaluates the energy consumption of the 2 and 4-benchmark multiprogram workloads across all the studied approaches. Regarding 2-benchmark mixes, Figure 5.6 plots the consumed energy normalized to the *Shared-ELC* approach and broken down into: i) static energy consumption due to cache leakage, ii) dynamic energy consumption due to cache lookups, and iii) network energy consumption. We show the latter energy component in order to highlight the energy consumption differences between both electrical and optical approaches, since in this work FOS is implemented with a custom optical ring. Notice that energy savings directly coming from the proposal, however, are found in the cache's (static and dynamic) energy values, and these savings are independent of the underlying network supporting FOS. In other words, in case that FOS was implemented over a high-performance electrical network, the energy savings achieved by the cache management would remain.

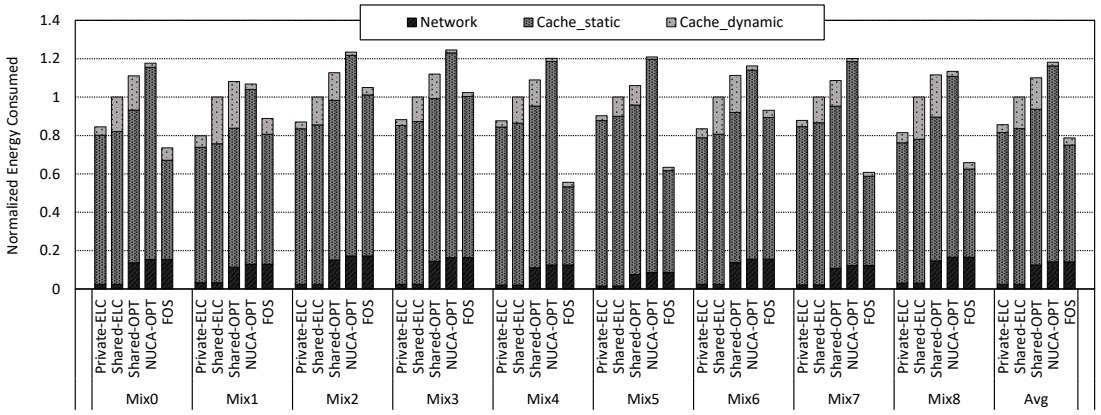


FIGURE 5.6: Normalized energy consumption of 2-benchmark mixes.

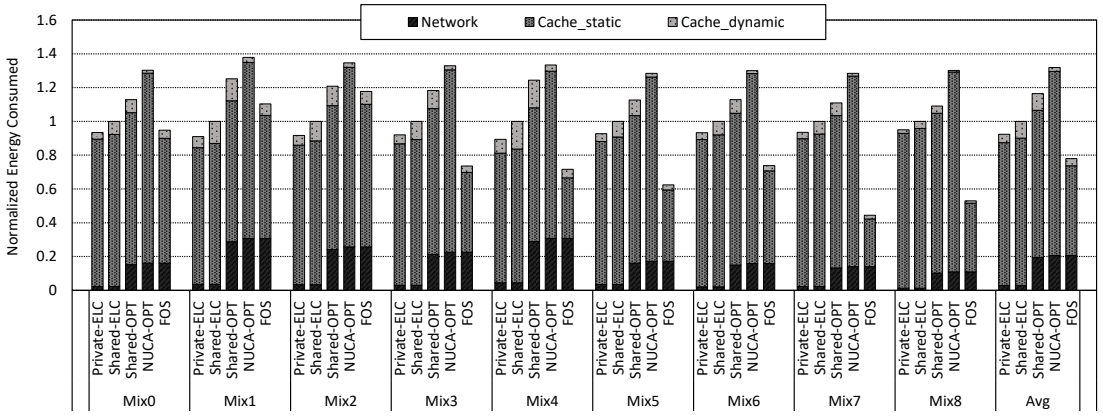


FIGURE 5.7: Normalized energy consumption of 4-benchmark mixes.

As observed, the static energy (i.e. leakage currents) consumption of caches is the component that clearly dominates the overall energy consumption across all the conventional cache organizations. Notice that this energy component is largely reduced by FOS thanks to the number of buffers that are turned off dynamically at runtime. These savings are especially significant in `Mix4` or `Mix7`, where FOS improves leakage currents by 60% to 70% over conventional organizations. Nevertheless, when executing cache-hungry mixes like `Mix2` the whole pool of buffer needs to be activated for performance, which means that FOS is correctly working. Of course, performance improvement is achieved at the cost of energy, which in this case matches the one consumed by *Shared-ELC* and *Shared-OPT* approaches.

Dynamic energy consumption (shown in the upper component of each bar) represents

the smallest energy component. *NUCA-OPT* presents the best results, thanks to the reduced complexity of their cache modules. FOS implements cache modules with the same size, although its dynamic energy consumption is slightly higher than in *NUCA-OPT* because of the PTA overhead. Since tag lookups in the PTA structure are performed only for the active buffers, however, the corresponding overhead reduces the dynamic energy consumption on average by a 4× factor over *Shared-ELC* and *Shared-OPT* approaches. Moreover, when executing mixes with a low number of active buffers like `Mix4` and `Mix7`, FOS achieves a dynamic energy consumption similar to that of *NUCA-OPT*.

Regarding the network energy, it can be observed that, on average, the energy consumption of optical interconnects is by 10× higher than that of electrical links. This overhead translates to an overall energy consumption increase by 20% from *Shared-ELC* to *Shared-OPT*. In the *NUCA-OPT*, the optical network overhead, together with the higher leakage values of 64 KB modules, leads to an energy consumption by 35% higher than the *Shared-ELC* baseline. On the contrary, and despite experiencing also this network overhead, FOS presents an energy consumption by 23% on average lower than *Shared-ELC*, improving energy consumption over the electrically-connected approaches in seven out of nine mixes.

In summary, it can be observed that the energy savings provided by FOS coming from cache management nicely compensate the energy expenses coming from the optical network. These results show the high potential of the proposal, which is one of the main aims of this work, and let us to conclude that FOS would provide higher energy savings while sustaining the performance if it was implemented with a low power NoC that was able to fulfill the latency requirements.

Lastly, Figure 5.7 plots the energy results for the 4-benchmark multiprogram workloads. As observed, FOS provides cache energy savings across most of the mixes. These savings come from both dynamic and static energy consumptions. Dynamic energy consumption is highly improved, in percentage, compared to *Shared-ELC* and *Shared-OPT* approaches. Static energy consumed by the cache modules is also significantly reduced in a wide set of mixes, being in some cases less than half the energy consumed by conventional approaches. Again, in cache intensive workloads like `Mix1`, FOS keeps an energy consumption similar to that exhibited by the *Shared-ELC* system, since as mentioned above, FOS activates buffers for performance.

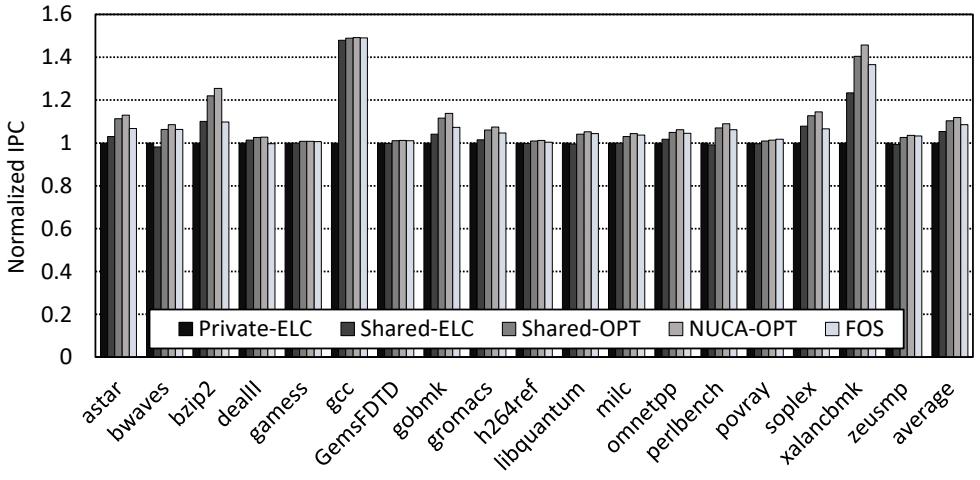


FIGURE 5.8: Normalized performance of the individual applications.

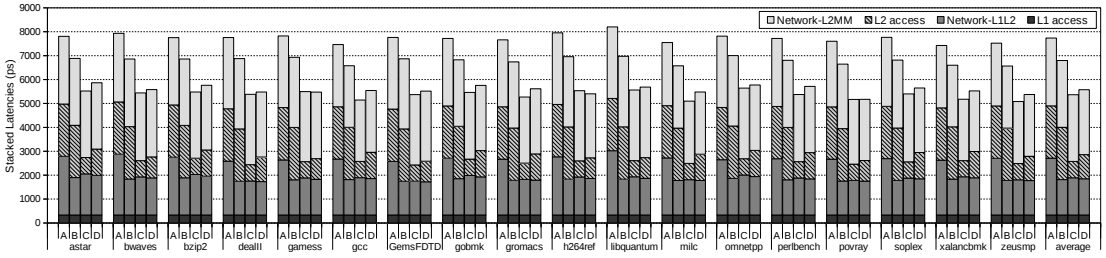


FIGURE 5.9: Memory subsystem latencies broken down in four main categories. Legend: A: Shared-ELC, B: Shared-OPT, C: NUCA-OPT, D: FOS.

5.5.2 Performance Evaluation of Individual Applications

This section evaluates the FOS's performance considering each benchmark running alone in one of the cores of a 2-core system, while the other core is left empty. First, we analyze and compare the normalized performance (in terms of IPC) of FOS and the four studied schemes: *Private-ELC*, *Shared-ELC*, *Shared-OPT* and *NUCA-OPT*. Figure 5.8 shows the results normalized with respect to *Private-ELC*. As observed, in spite of the fact that the main goal of FOS is not related to achieve better performance, our approach improves performance, on average, over *Private-ELC* by 8% (i.e. by 3% higher than that of *Shared-ELC*). The schemes with optical network (i.e. *Shared-OPT* and *NUCA-OPT*) achieve the highest speedups, improving performance by 10% and 12% over the *Private-ELC* scheme.

To provide further insights on where performance improvements come from, we analyzed

the memory subsystem latency, breaking this latency down into five main components: i) L1 cache access time, ii) L1 to L2 cache NoC latency, iii) L2 cache access time, iv) L2 cache to memory controller NoC latency, and v) memory controller latency to handle the main memory access. Figure 5.9 plots the results for the first four components, since the main memory subsystem is the same across all the studied approaches. Results are shown for the *Shared-ELC*, *Shared-OPT*, *NUCA-OPT* and *FOS* systems, which are referred to as *A*, *B*, *C* and *D*, respectively, in the figure. As observed, optical interconnects provide a 1000 *ps* faster L1-L2 network latency, on average, than the electrical links employed by the *Shared-ELC* scheme. This latency reduction explains, at a first glance, why all the optical approaches outperform both *Private-ELC* and *Shared-ELC* systems in individual execution.

Latency differences across the studied systems can also rise, however, because of the different access times of the L2 cache modules. *NUCA-OPT* system divides its cache space in 64 *KB* modules, while *Shared-ELC* and *Shared-OPT* implement a single 1024 *KB* module, whose access time is 3× higher[2]. On the other hand, the access time presented by FOS depends on the number of buffers actually being allocated. For instance, in `xalancbmk`, a cache intensive application, the FOS memory subsystem latency is up to 1100 *ps*; however, in `h264ref`, a compute intensive application, this time drops down to 700 *ps*. Notice that the L2 latency of FOS is slightly higher than that of the *NUCA-OPT* system. This small increase is due to the sequential access to the tag and data arrays that FOS performs.

In summary, optical NoC and small L2 cache buffers help improve performance for individual execution. In this regard, both the optical NUCA and FOS present similar performance.

5.5.3 Performance and Cache Space Management Evaluation for Multiprogram Workloads

In addition to the performance in isolated execution, this section presents results for 2- and 4-program mixes. Figure 5.10 shows the system performance (i.e. IPC harmonic

[2]According to the cache model results obtained with the CACTI 6.5 tool using a 32 *nm* technology node.

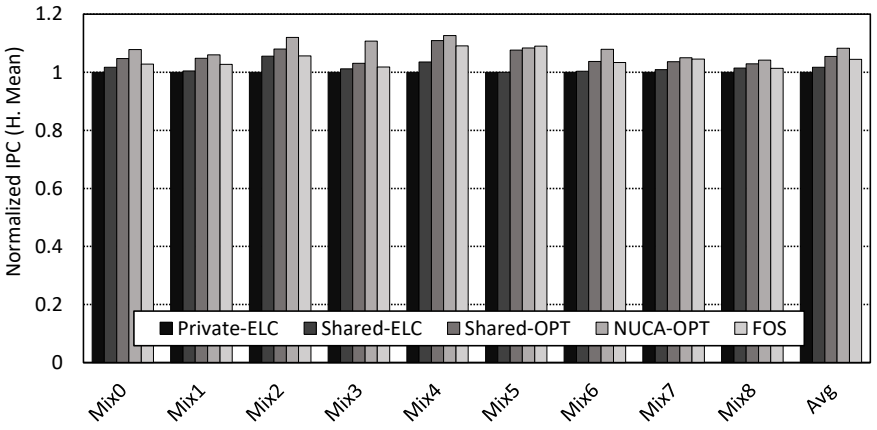


FIGURE 5.10: Normalized Performance of 2-bench mixes.

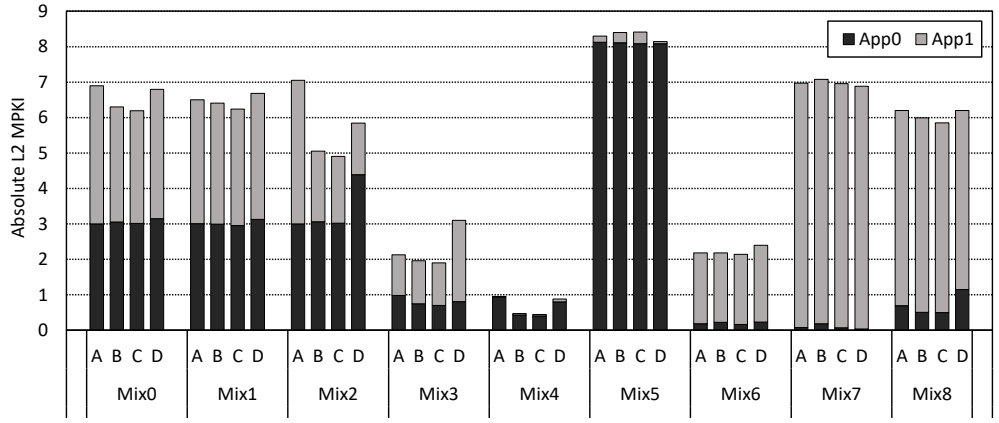


FIGURE 5.11: MPKIs of 2-benchmark mixes for: A: Private, B: Shared, C: NUCA, D: FOS.

mean) across the 2-program mixes. As expected, the *NUCA-OPT* approach is on average slightly the best performing approach, closely followed by *Shared-OPT* and FOS. Results, however, are not homogeneous across all the studied workloads since FOS has to face different scenarios depending on the cache behavior of each mix.

To provide deeper insights and identify these behaviors, we measured the MPKI of FOS on the allocated cache buffers, and the MPKI of the L2 cache on the remaining schemes. Figure 5.11 presents the results for *Private*, *Shared*, *NUCA* and *FOS* systems, which are referred to as *A*, *B*, *C* and *D* respectively. The legends *App0* and *App1* refer to the order of the application in the mix as stated in Table 5.5. Since FOS trades off performance against cache space, we should consider MPKI jointly with the number of

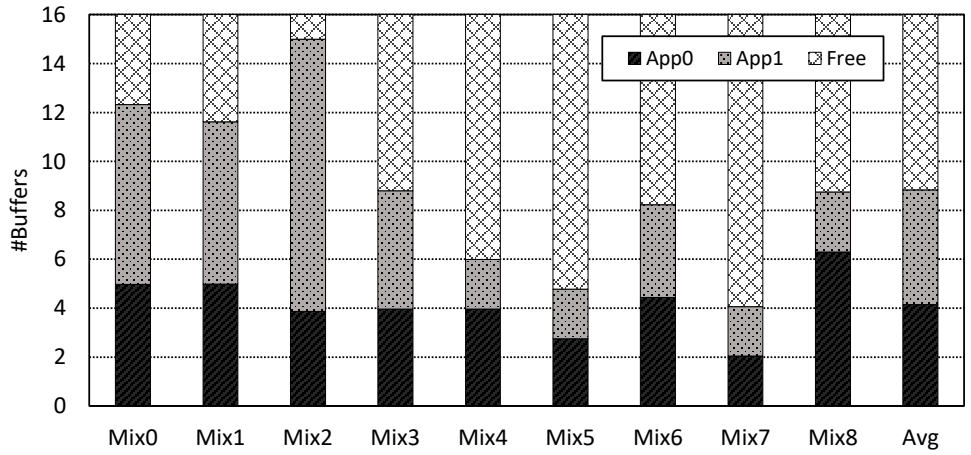


FIGURE 5.12: Average number of buffers assigned to each application in FOS.

allocated buffers in the analysis. Figure 5.12 shows the average number of buffers that the SMM algorithm allocates to each application.

First, we analyze the cache behavior of each mix according to the categories previously identified in Section 5.1. Applications with *Minimum Cache Needs* (MCN) like `milc` or `libquantum` (i.e. *App0* in `Mix5` and *App1* in `Mix7`, see Table 5.5) do not present major differences in their MPKIs among the studied approaches due to their inherent low locality. The BMM algorithm realizes this fact and only allocates 3 and 2 buffers to them on each mix. Secondly, applications with *Limited Cache Needs* (LCN), like `astar` in mixes 0 and 1 (*App0*) or `gcc` in `Mix1` (*App1*), are provided with 5 buffers each and present MPKI values similar as the ones shown by conventional configurations. These values, however, are slightly higher than those with shared L2 caches since FOS has to progressively activate buffers as they are predicted to be needed. Finally, applications included in the *Non-limited Cache Needs* (NCN), like `xalancbmk` in `Mix2` (*App1*) or `bzip2` in `Mix0` (*App1*), are provided with 10 and 7 buffers on average and clearly present better MPKI values than the *Private* cache configuration.

These MPKI results match the cache sensitivity study carried out in Section 5.1. When executing multiprogram workloads, however, MPKI can increase due to the inter-application interference. Nevertheless, since FOS assigns buffers as private to cores, the inter-application interference at the shared space is avoided. For instance, in `Mix5`, `h264ref` (*App1*) reduces its MPKI compared to both *Shared* and *NUCA* systems even though

in this mix there are only 4.5 active buffers on average during execution. In short, reducing the inter-application interference in the shared space allows FOS to improve performance over the *NUCA* approach in some mixes like `Mix5`. Notice that MCN applications with small working sets (like `h264ref`) are very sensitive to block replacements, hence preventing other applications from accessing their cache buffers translates to relevant performance gains. This is the reason why `h264ref` does not present significant performance improvements in individual execution, but it does in `Mix5`.

Although the BMM algorithm covers a wide set of cache demanding scenarios, there are two *corner cases* where the BMM algorithm does not achieve the best behavior. Next, we discuss these cases and how they could be addressed. These cases appear due to two different situations: i) buffers are assigned to applications in FIFO order, which means that an application with high cache needs may obtain a big amount of buffers in the earlier stages of its execution while reducing the available space for the co-running application, and ii) irregular and fast-changing cache demands are hard to be identified with the experimental BMM thresholds. An example of the first case can be seen in `Mix2`, where `xalancbmk` (*App0*) slightly reduces the available space for `astar` from 5 buffers to 4 and, due to this reason, the MPKI of the latter is higher in this mix than `Mix0` and `Mix1`. This behavior, however, can be prevented in FOS by adjusting the corresponding threshold to limit the maximum number of buffers that an application can be assigned. An example of the second case is `gromacs` (*App1* in `Mix3`). It can be seen that, when executed in the FOS system, this application almost doubles the MPKI of the *Private* system, in spite of having by 45% of the cache space still available. A more aggressive setup of the BMM algorithm would limit this behavior but, in this work, only the setup that provides the highest energy efficiency is shown (see Section 5.5.1).

To conclude the analysis of 2-benchmark mixes, results have shown that, on average, FOS assigns only 55% of the total cache space that the other evaluated approaches, with none or negligible performance losses. This cache space is dynamically distributed according to application needs, which allows some applications to work with a smaller cache space.

Figure 5.13 presents the performance results for the 4-benchmark mixes. These mixes have been evaluated with 2 MB cache storage capacity for all the evaluated cache approaches. Results show that, on average, FOS improves performance by 4% over the

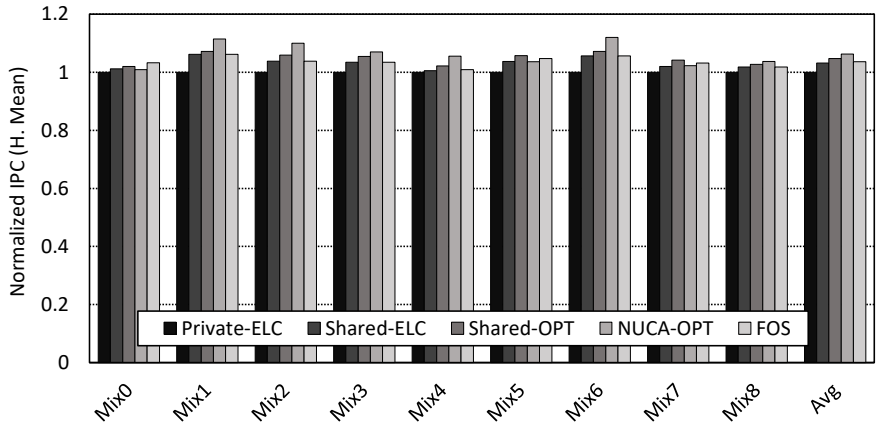


FIGURE 5.13: Normalized Performance of 4-bench mixes.

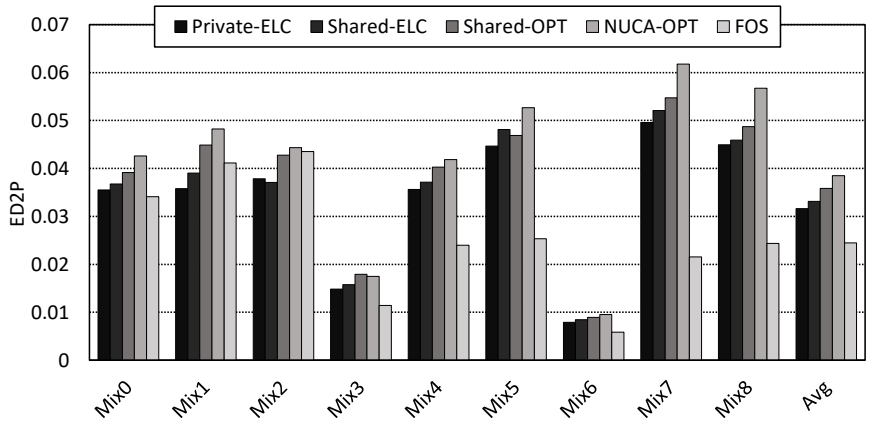


FIGURE 5.14: Energy Delay Squared Product of 2-benchmark mixes.

Private-ELC system, while *Shared-ELC*, *Shared-OPT* and *NUCA-OPT* improve by 4%, 5% and 6%, respectively. Performance differences are lower than in 2-benchmark mixes between electrical and optical approaches, mainly because optical rings present slightly higher contention. Performance, however, could be enhanced with an improved optical NoC and with more aggressive thresholds. The performance improvements achieved by FOS are especially significant in mixes 0, 5 and 7 because in these mixes FOS prevents `milc` from interfering the other applications in the shared space. Moreover, despite turning off part of the cache space and making an efficient use of the cache space, FOS just presents by 2% performance degradation with respect to *NUCA-OPT*.

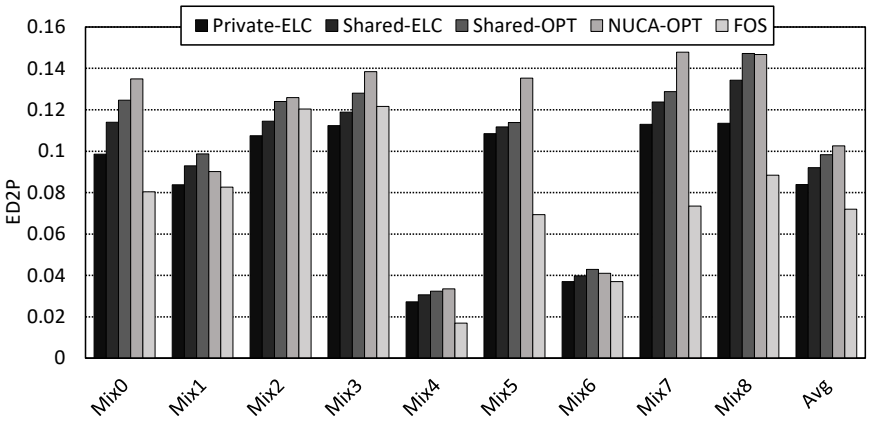


FIGURE 5.15: Energy Delay Squared Product of 4-benchmark mixes.

5.5.4 Energy Efficiency

To summarize the previously exposed performance and energy results and conclude the experimental evaluation, in this section it is discussed the overall energy efficiency of FOS with respect to conventional approaches. To this end, Figures 5.14 and 5.15 show the energy delay squared product (ED2P) for 2- and 4-benchmark mixes, respectively. It can be seen that, on average, FOS is the approach presenting the best results, reducing the ED2P in seven out of nine mixes both in 2- and 4-benchmark multiprogram workloads. Moreover, in some cases this reduction is over 60%. An interesting observation is that, according to the performance results presented below, the best performing approach (i.e. *NUCA-OPT*) shows the worst ED2P values in both figures, while the worst performing approach (i.e. *Private-ELC*) presents the second lowest ED2P values. This means that in the former case, *NUCA-OPT* increases performance at the expense of energy, mainly due to the optical NoC; while in the latter it occurs the opposite, that is, energy savings are achieved at the expense of performance.

FOS addresses these trade-offs showing that, in spite of presenting an energy consumption even lower than that of *Private-ELC* in most mixes, it achieves similar performance to *NUCA-OPT*, which translates to better ED2P results.

Finally, Table 5.6 summarizes a quantitative comparison of the energy, performance and energy efficiency values corresponding to all the studied approaches. Results show that FOS is the most energy-efficient approach, improving by 16.55% the energy efficiency with respect to *Private-ELC*. *NUCA-OPT* is shown as the least efficient scheme, since in

TABLE 5.6: Energy efficiency, performance and energy savings achieved by each architecture normalized to the Private-ELC approach.

Architecture	**Energy efficiency**	**Performance**	**Energy savings**
Shared-ELC	−8.91%	3.33%	−14.42%
Shared-OPT	−14.8%	4.87%	−22.29%
NUCA-OPT	−18.24%	7.03%	−27.61%
FOS	16.55%	3.79%	8.45%

spite of being the best performing approach, it prohibitively increases energy consumption by 27.61% over *Private-ELC*. Similarly, *Shared-ELC* and *Shared-OPT* configurations achieve marginal performance gains but increase the energy consumed, which also makes *Private-ELC* a more efficient organization than these approaches. This means that, among all the studied approaches, FOS is the only one that improves performance over the private baseline while also reducing the energy consumed.

5.6 Summary

In this chapter, FOS has been presented as a novel cache organization and management approach to trade off performance for energy consumption. The FOS architecture replaces conventional low level (e.g. L2 and L3) caches with a single level consisting of a pool of cache buffers, and introduces a novel management approach especially suited for low power processors. Buffers are much larger (i.e. by a thousand times larger) than individual cache lines, which eases the implementation of practical energy-aware approaches working at this granularity. For the sake of exploring the impact of FOS on performance, it has been implemented and evaluated with an underlying Optical Network on Chip, since it provides a rather fast and uniform access latency to the buffers. Other networks (not necessarily optical) might be used, whenever latency requirements are satisfied.

Experimental results have shown that FOS reduces dynamic energy consumption on average by a factor of up to 4× over a shared cache organization with the same storage capacity. Energy savings on static energy consumption are also achieved, reaching a reduction by up to 60% of the total static energy over conventional approaches. As a result, FOS is the least energy consuming approach among all the studied cache

organizations. Overall energy results also include the overhead introduced by the devised ONoC, which means that energy benefits can be even higher.

Moreover, FOS energy savings do not come at the expense of performance, since moderate performance improvements come from the network and the cache side. That allows FOS to achieve a similar performance to an optically connected NUCA cache, in spite of using, on average, 50% of the total cache space. In summary, we have shown that FOS is the most energy efficient approach, as it does not only achieves the highest energy savings but also sustains performance.

Chapter 6

FOS-Mt: An efficient Flat Storage Organization for Multithreaded Workloads

Chapter 5 has presented and evaluated the FOS architecture. The achieved results show that there is significant potential redesigning existent cache organizations when dealing with energy efficiency in current processors. However, although the presented FOS architecture makes important progress towards this objective, it still lacks of several features like coherence management or providing support for multithreaded applications. Therefore, the FOS architecture needs to be extended to support any kind of applications. This chapter describes the required features for this purpose, and evaluates the approach with multi-threaded workloads.

The chapter is organized as follows. First, we introduce FOS-Mt and its new features with respect to the FOS architecture. Then, the details of the FOS-Mt architecture are further explained, specially focusing on coherence management and the new buffer allocation algorithm. Finally, we present the evaluation setup, which has been extended with respect to the previous chapter, and the experimental results.

6.1 Background: the FOS architecture

The architecture presented in this chapter builds on the FOS architecture previously discussed in Chapter 5. The FOS architecture is the result of a research on the lack of flexibility and several issues regarding energy inefficiency in current cache organizations. The proposed solution consists in having a common pool of cache buffers that are dynamically assigned to cores at runtime. Buffers are initially powered off for energy savings, and they are powered on and allocated to a single core based on the cache needs of the running application. This approach has been demonstrated to be effective since it reduces both static and dynamic energy consumption at the same time it sustains performance.

However, this approach still presents some limitations that need to be addressed in order for it to support multithreaded applications. First, multithreaded workloads presents different needs and behaviors with respect to sequential applications. For this kind of workloads, the management of shared data blocks in the cache is critical in order to sustain performance. Due to this fact, another issue that needs to be addressed in the multithreaded design of FOS is the allocation algorithm that is in charge of assigning buffers to cores. Finally, since new hardware structures would be needed to support cache coherence management, the underlying network must be also adapted and modified with respect to that employed in FOS.

6.2 Flat On-chip Storage for Multithreaded Applications

This section presents the proposed architecture, its main components, the implementation details, and a working example.

6.2.1 FOS-Mt Architecture Overview

Similarly to its predecessor FOS, the FOS-Mt architecture redesigns the cache hierarchy to address energy and enable practical implementation of energy aware mechanisms at a coarse grain granularity (e.g. 128KB buffer size). Having a single buffer pool, which replaces L2 and L3 caches, shrinks the area which is a key concern in low power and

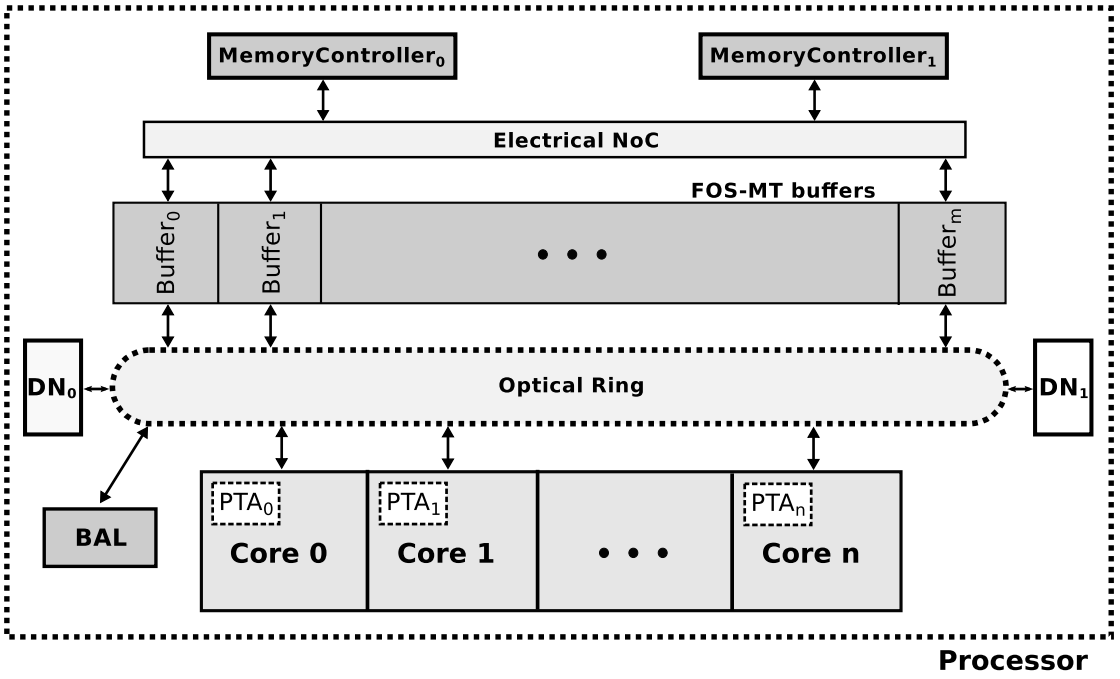


FIGURE 6.1: Block diagram of FOS-Mt.

embedded designs. Neither low power nor embedded processors are allowed to build huge last level caches, hence managing the limited cache space in an efficient way trading off energy and performance becomes a major design concern.

Figure 6.1 presents a block diagram of the updated FOS-Mt approach. As observed, the pool of buffers remains, together with the PTAs, the special *Buffer Allocation Logic* (labeled as BAL) node and the interconnection network (labeled as optical ring) which connect all the structures. Additionally, new directory structures that help manage coherence (labeled as DN, Directory Nodes) are also shown in the diagram. Below these components are discussed.

The off-core buffers are still organized as a pool of elements that can be individually assigned to any core. One of the main issues of such an organization is that it implies that the access latency will vary depending on the target buffer that a given core is accessing to, which may affect performance. The FOS architecture addresses this issue by implementing an on-chip optical network that provides a rather homogeneous access time (see Section 5.3 for further details on the FOS network). This approach remains in FOS-Mt, although the optical network has been adapted to introduce the

new hardware structures (e.g. the directory nodes). Section 6.3.1 presents the details of the implemented network.

Unlike NUCA-based conventional approaches where the cache modules are accessed by address interleaving, FOS-Mt dynamically distributes and assigns to the cores the available *cache buffers*. Therefore, in our approach, address interleaving is not a valid way to access the target buffer. To overcome this shortcoming while providing a flexible buffer management the FOS architectures decouple the off-core buffer tag arrays from their corresponding data arrays. Decoupling tags from data has been also done in modern processors like the IBM POWER5 [145].

The aforementioned directory structures store the decoupled tags and the coherence information, and manage the accesses to the off-core data buffers. FOS-Mt introduces two types of hardware structures (see Figure 6.1) called *Directory Nodes* (DNs) and *Private Tag Arrays* (PTAs). The PTAs were also implemented in the original FOS architecture; they replicate, in each core, a subset of the tags of the locally accessed blocks; while the DNs, located in the off-core area, store the decoupled tags and coherence information of the active buffers.

Finally, a *Buffer Allocation Logic* (BAL) is needed to keep track of the buffers that are being assigned to the execution cores. This logic is connected to the optical ring through a special node, labeled in Figure 6.1 as BAL. Notice that, in the layout of a physical implementation, this node can be located elsewhere provided that it is connected to the execution cores.

Below, the implementation of these structures and their operation are discussed. To help the understanding of these structures, Section 6.2.2 presents a simple working example with 2 PTAs associated to the two cores, 2 DNs and 4 off-chip buffers.

6.2.2 FOS-Mt Architecture: Detailed Implementation

This section discusses the details of the FOS-Mt architecture, including key design decisions as explained below. To help understand these decisions, Table 6.1 summarizes them together with the rationale (advantage) that led us to take that decision as well as the downside or drawback associated to such design decision.

TABLE 6.1: FOS-Mt design considerations.

Design decision	**Advantage**	**Potential Disadvantage**
Decoupled tags & Distributed DNs	Prevents centralized block lookup bottlenecks and helps the system to scale	ONoC complexity must be addressed (see Section 6.3)
Per-core (Private) PTAs	Block location information are placed close to the core to speedup the access to the block and reduce network contention	PTA entries must be invalidated on block replacements in the buffers
Keeping coherence information in DNs	Access time to coherence information is reduced	None
Supporting invalidations of PTA entries	Improve memory access time on L1 cache misses	PTA invalidations must be supported

Decoupled Tags and Distributed Directory Nodes

FOS-Mt needs to find out the location of data blocks in the buffer pool. Unlike conventional architectures, this is achieved by decoupling the tag arrays from the data buffers and distributing them among several Directory Nodes.

Distributing this information among several DNs balances the number of accesses of the cores among the DNs and prevents the potential bottleneck of having a centralized directory structure. Additionally, decoupling and distributing tags also helps the system to scale. A closely related implementation can be found in some commercial designs like the Niagara II [146].

To find the location of a data block, first, the target DN is selected in an interleaved way, that is, using the least significant bits of the block address. Then, the DN is looked up to obtain the target buffer, the *buffer set* and the *buffer way* where the requested block is located. With this information, the subsequent access to the data array can be performed as fast as in a direct-mapped cache. To simplify the hardware implementation, each DN is organized including as many *Buffer Tag Arrays* (BTAs) as off-core buffers, and BTAs present the same set-associative organization as the buffers.

Compared to a conventional cache organization, this approach requires an additional NoC hop since it has to access to the DN, which might slightly affect the average memory access time. To address this potential adverse impact on performance a hardware structure, tha PTA, has been devised (see Section 5.2.3).

Per Core (Private) Tag Array

The PTAs are implemented as cache-like structures that keep a copy of the DNs' tag and location information for a *subset* of the blocks locally accessed by the core. Having the tags of the requested blocks closer to the core reduces the number of accesses to the DNs, thus reducing both network contention and improving the average memory access time. The PTA consists of two main types of substructures: several BTAs and a single *Shared Tags Table* (STT) structure.

The BTAs of the PTA of a given core cache both the tags and the locations of the blocks stored in the data buffers assigned to that core. Each active (powered on) BTA is mapped to a given buffer assigned to the core. Due to hardware complexity and scalability reasons, FOS-Mt limits the number of BTAs per core, thus constraining the number of data buffers that can be assigned to a given core. To quantify the impact of this design issue, an analysis of the performance sensitivity to the PTA size (i.e. the number of BTAs implemented in each core) is presented in Section 6.5.1.

The STT keeps track of tags and locations of shared blocks (i.e. accessed by several cores) that reside in buffers not assigned to the local core. This structure acts as a small local cache (e.g. 1K entries in our experimental setup) that enables a fast location of the shared blocks without accessing to the DNs.

A PTA access can result in a hit or a miss. On a PTA miss, the corresponding DN must be accessed in order to find the location of the target data block. This location is cached in the PTA either in a BTA or in the STT, depending on whether the block is located in one of the assigned buffers or not. On a PTA hit, the cached location is used to directly get the target block without a DN lookup.

The PTA entry (i.e. the stored information) of a given block can be made invalid due to different reasons. For instance, if a block is replaced from a buffer, then the cached PTA information for that block must be invalidated to prevent future accesses to that block take the data from its previous location. In addition, PTA information can be invalidated due to coherence management.

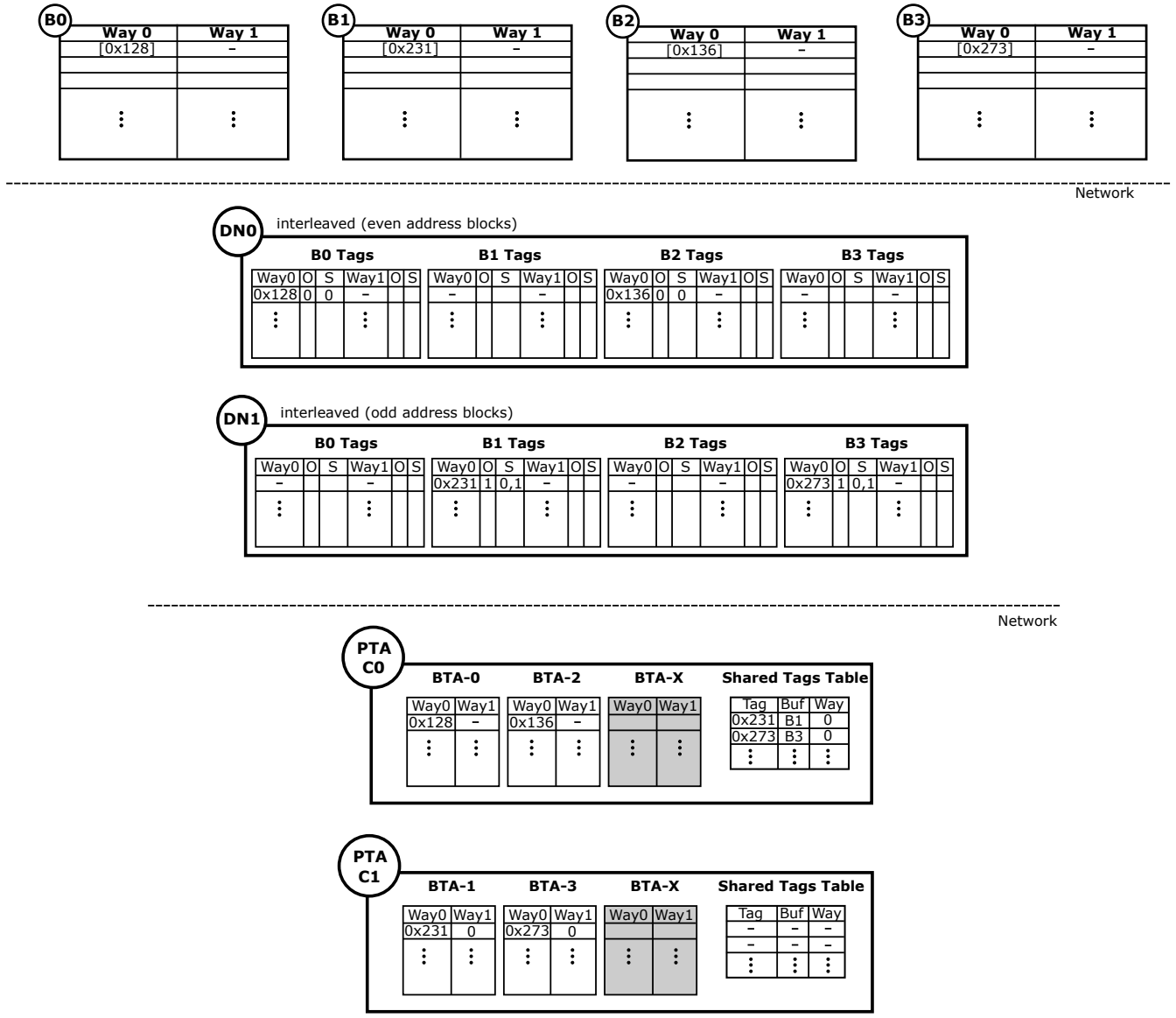


FIGURE 6.2: Example implementation of a 2core-4buffers FOS-Mt.

Coherence Management: The Role of DNs and PTAs

FOS-Mt coherence management in L1 caches is based on conventional directory-based MOESI with minor modifications to accommodate FOS-Mt design decisions.

The first design decision that must be taken related to coherence management is where coherence information (i.e. the MOESI directory) is placed, since this decision may affect the system performance. Coherence information can be stored at the off-core buffers, or decoupled of such buffers and moved to the DNs. In the former case, an additional

hop through the target DN is necessary to access to it when the PTA contains invalid location information. Therefore, in FOS-Mt, coherence information is stored at the DNs. Note that this design decision does not modify the underlying MOESI protocol.

The second design decision related to coherence management that may affect the system performance is related to the PTA. In a conventional MOESI implementation, coherence information should be accessed either i) on an L1 cache miss, to identify a remote L1 cache that contains the last updated version of the target block; or ii) on a write to a non-modified block, to identify and invalidate copies of the block in remote L1 caches. Both conditions could be applied directly to FOS-Mt without modifying the conventional MOESI implementation. Notice, however, that in FOS-Mt, the L1 cache miss management can be optimized when the target block has not been modified by any remote L1 cache. In this case, the contents of the target block can be directly retrieved from the data buffer pointed by the PTA, so avoiding the DN access.

This optimization, however, introduces a corner case when a block is modified in a remote L1 cache and, therefore, the copy of the block in the data buffers becomes stale. This means that the PTA will point to stale data after the block is written. We solve this problem by invalidating the corresponding PTA entries. These invalidations are carried out alongside with the coherence actions performed in the DN and remote cores when the block is modified. After the modification, a PTA miss due to a remote invalidation will access to the DN coherence information where a core with an updated copy of the block is obtained.

Putting It All Together: PTAs & DNs Operation

The mentioned components work together as follows. Upon an L1 cache miss, the PTA is looked-up to check whether the tag of the missing block is in any of its structures. On a hit in the PTA (either in any BTA or in the STT), the PTA logic indicates the target buffer, set and way identifiers of the target block, and the missing block is fetched from that location to the L1 cache.

On a PTA miss, the corresponding DN is accessed to check the buffer tags are not present in another PTA. On a hit in the DN, the block is fetched from the corresponding data buffer or L1 cache, according to the coherence information stored in the DN. As explained

above, coherence information in the DN must be properly updated in the following cases: i) on a PTA miss; ii) on a write to a non-modified copy of a shared block; and iii) when a block is evicted from the L1 and a writeback to a data buffer needs to be performed.

In addition to provide a copy of a block to the requesting core, the PTA of the core and the corresponding DN location information must be updated. To perform this operation, on a PTA miss, a message is sent to the DN indicating a buffer assigned to the core and a way in this buffer to perform the replacement. Then, the DN checks the location of the block, and in case the block is not present in the data buffers, it will be brought from main memory to that location, updating the BTAs in the PTA and the DN accordingly. If the block is already in the data buffers, it can be in a data buffer assigned or not to the requesting core. In the first case, the BTAs are already updated, while in the second case the location information is obtained from the DN and cached in the STT.

Finally, notice that whenever a block is replaced from the data buffers, the location information in DNs and PTAs must be properly updated. This implies i) updating the BTA of the buffer storing the block in the PTA of the core that has this buffer assigned, ii) updating the BTA in the corresponding DN, and iii) invalidating the STT entry in the PTAs of the other cores sharing the block. These actions do not introduce additional overhead because data buffer replacements already require accessing the DN and invalidating block copies in the cores.

To illustrate the operation of the proposed approach, Figure 6.2 shows a FOS-Mt example that involves all the aforementioned structures. In this example, there are two cores, 2 DNs and the maximum number of buffers that can be allocated to each core is limited to 3. As can be seen in the PTAs of core C0 (PTA-C0) and core C1 (PTA-C1), PTA-C0 has allocated buffers B0 and B2, while PTA-C1 has been provided with buffers B1 and B3. The remaining BTAs ($BTA-X$ in the figure) in each PTA are powered off since there are no more buffers available.

As observed, B0 and B2 BTAs in PTA-C0 store the tags 128 and 136 respectively and, by block address interleaving, these tags are kept in the directory node DN0. Accordingly, the data arrays associated to these tags are stored in buffers B0 and B2. In addition, core C0 is the owner of both blocks. There are no sharers recorded for these blocks, which means that so far these blocks have only been accessed by core C0.

On the other hand, PTA-C1 keeps the tags 231 in B1 and 273 in B3. Since these are odd tags, in this case DN1 is the directory node in charge of keeping the references to these blocks. In this example, unlike the blocks fetched by core C0, the blocks 136 and 273 are shared, as indicated by the coherence information in the BTAs B1 and B3 of DN1. This means that these blocks have been first accessed by Core 1 and afterwards by core C0, and so they are recorded in the Shared Tags Table of PTA-C0. Thanks to this structure, PTA-C0 can access to shared blocks without asking DN1 for the buffer where they are stored, avoiding the associated latency penalty. In case the location of a shared block were not found at the STT either because it has been replaced or it is the first access of the sharer, the location can still be obtained from the corresponding DN.

6.2.3 Off-Core Buffer Management

The management of the off-core cache buffers covers both the assignment and the release of buffers. The former task is carried out by the Buffer Allocation Algorithm (BAA), which assigns buffers to threads only when that assignment is expected to provide a good trade-off between energy and performance.

The latter operation is carried out by a simple Release Mechanism that deallocates a buffer from a core based on time and activity criteria. Design decisions have been made conservative to keep energy low, and only one buffer at a time (i.e. at each interval) can be assigned to or unassigned from each core.

The key challenge for the Buffer Allocation Algorithm is to estimate the performance that a thread would have experienced if it had had an additional buffer. For this purpose, when the algorithm is applied, and before making the decision on whether a buffer is assigned or not to a thread, some performance estimates needs to be done. That is, FOS-Mt needs to estimate the performance that a thread would have experienced with one additional buffer $b+1$ while being executed with b buffers, which is challenging. Notice that this issue was already solved during the design of FOS, by implementing additional *shadow tags* in the BTAs of the PTA. These additional tags are used for estimating the number of cache misses with one additional buffer and they are replicated only in 32 sets of each BTA to limit the overhead.

Buffer Allocation Algorithm

The devised algorithm is applied at the beginning of each execution interval to each application thread, after measuring the performance of the assigned buffers during the expiring interval. Execution intervals are assumed to be fixed-length ($8M$ processor cycles) for evaluation purposes. Algorithm 2 summarizes the three main steps of the devised algorithm.

In the first step, *Buffer Performance Measurement*, FOS-Mt computes the performance of the assigned buffers (b) in the expiring $n-1$ interval. Performance is represented in terms of BTAs-Misses Per Kilo-Instruction ($BTAs\_MPKI_{n-1}(b)$). In addition, the performance with an additional buffer $b+1$ is estimated ($BTAs\_MPKI_{n-1}(b+1)$) with the shadow tags. Finally, the amount of hits in the BTAs ($BTAs_{hits}$) and in the STT and DN (STT/DN_{hits}) are also accounted.

Once these values have been computed, the BAA moves to its second step, *Filtration*, where the relevance of the previously gathered values is studied. First, the global $MPKI$ of a given thread must surpass a given threshold Thr_{min}, since reducing low $MPKI$ values by adding one extra buffer is a waste of energy. Also, to guarantee that the metrics are computed over a significant number of hits, the collected $BTAs_{hits}$ must be higher than the threshold Thr_{BH}. If any of these conditions is not met, the algorithm finalizes without assigning any additional buffer to the thread for the incoming interval.

In the last step, *Buffer Allocation and Performance Analysis*, the algorithm computes two metrics and makes the decision on whether a new buffer should be allocated to the thread. On the one hand, the algorithm computes the improvement of the $BTAs\_MPKI$ rate ($MPKI_{dec\_rate}$), which quantifies the rate in which the $BTAs\_MPKI$ of the thread is expected to decrease by allocating one extra buffer. On the other hand, we compute the BTAs hits to total hits rate($BTAs_{weight}$), which indicates how much use that thread is making of its assigned buffers. These two metrics together indicate how much impact allocating a new buffer for the thread can have on performance.

To make the decision on assigning an extra buffer, the previous metrics must be compared with experimental thresholds. During the experiments, we found that static thresholds do not perform well enough since applications and even threads belonging to the same application can present different needs. To deal with this issue, we introduce a constant K to scale the thresholds. This way, K is obtained by categorizing at execution-time

Algorithm 2 Buffer Allocation Algorithm.

Inputs:
n: incoming interval number;
b: amount of buffers currently assigned to the thread;
K: MPKI category for this thread;
$\{Thr_x\}$: algorithm thresholds.

1. **Buffer Performance Measurement.** For each running thread:

$$BTAs\_MPKI_{n-1}(b),$$
$$BTAs\_MPKI_{n-1}(b+1),$$
$$BTAs_{hits} \text{ and } STT/DN_{hits}.$$

2. **Filtration.** For each running thread:

$$\textbf{if } (\ MPKI_{n-1} < Thr_{min}) \textbf{ or } (\ BTAs_{hits} < Thr_{PH}\ )$$

Skip step 3 and finalize.

3. **Buffer Allocation and Performance Analysis.** For each running thread:

$$MPKI_{dec\_rate} = \frac{BTAs\_MPKI_{n-1}(b+1)}{BTAs\_MPKI_{n-1}(b)} - 1,$$

$$BTAs_{weight} = \frac{BTAs_{hits}}{BTAs_{hits}+STT/DN_{hits}}.$$

$$\textbf{if } (\ BTAs - MPKI_{dec\_rate} > K \times Thr_{MPKI}\ ) \textbf{ and } (\ BTAs_{weight} > K \times Thr_{weight}\ )$$

Request a new buffer.

each thread ranging from 1 to 4 based on its historic $MPKI$ value (i.e. considering the total number of off-core misses), and thresholds are scaled on each interval by multiplying its base value by this constant. Finally, if both performance metrics surpass their corresponding threshold, the algorithm considers that energy is traded off by performance and a new buffer is assigned to the thread.

Buffer Allocation Algorithm Working Example

This section presents through a working example how the devised Buffer Allocation Algorithm works. To illustrate the algorithm's behavior, Figure 6.3 shows the number of buffers assigned by BAA, dynamically at run-time, to each of the 8 threads of `barnes` (each plot represents a different thread), one of the benchmarks used in this paper and presented in Section 6.4. Additionally, the figure also plots both BTAs and STT/DN Hits per Kilo-Instruction (referred to as BTAs-HPKI and STT/DN-HPKI) and the total MPKI (i.e. considering the total number of off-core misses of each thread). The experiment has been run using the FOS-Mt setup explained in Section 6.4.

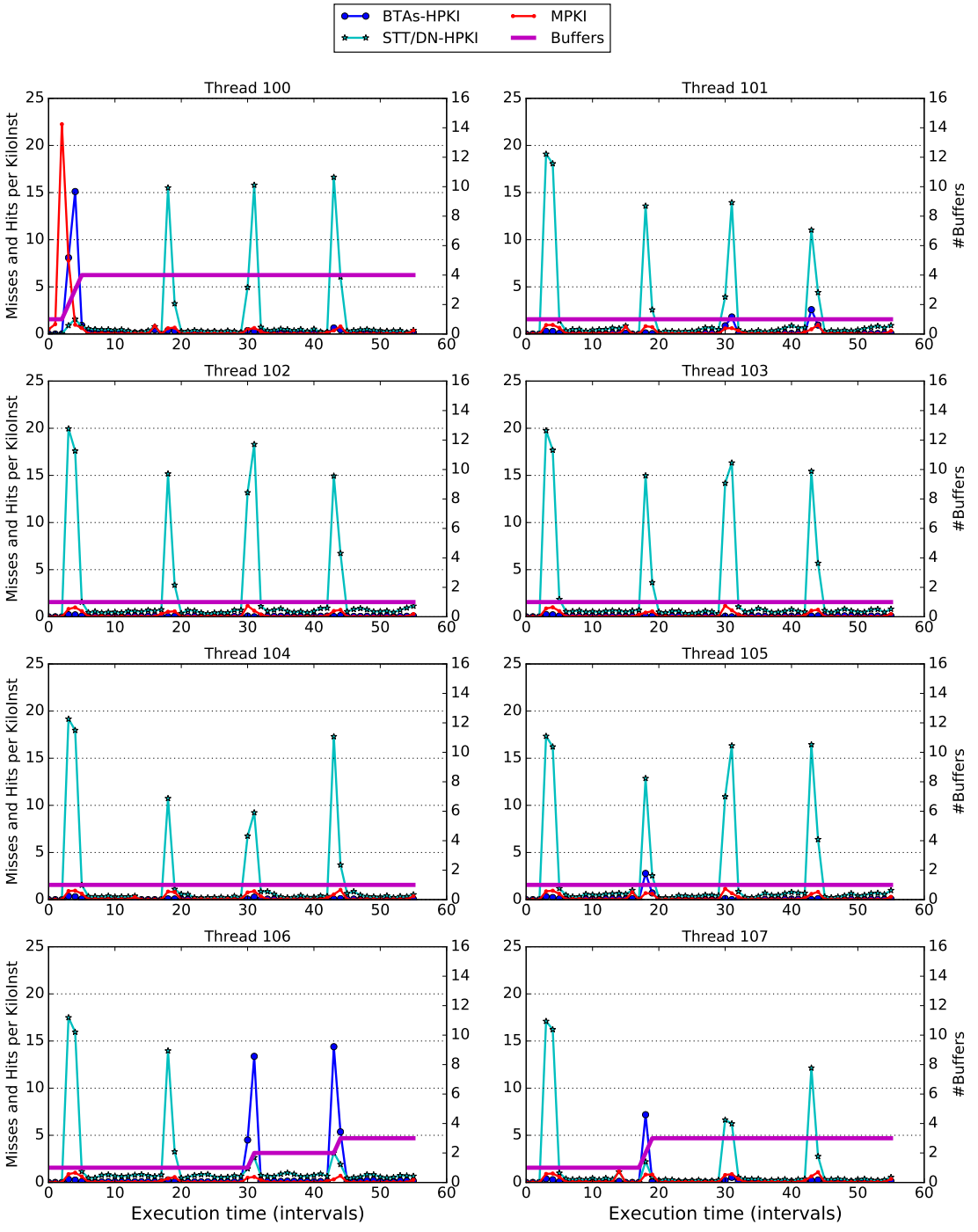


FIGURE 6.3: Number of allocated buffers during execution time for the 8 `barnes` threads.

At the beginning of the execution, the *Buffer Allocation Logic* assigns one buffer to each thread. Then, starting from the second interval, BAA is applied to each thread at the beginning of each execution interval. As explained above, two key conditions must be fulfilled for requesting a new buffer: the $BTAs\_MPKI$ is expected to improve, and the BTAs hits to total hits rate($BTAs\_weight$) is over a given threshold. In the figure, the

second condition can be appreciated in the plots of threads 100, 106 and 107, which are those experiencing a higher amount of BTAs hits, and therefore are awarded with extra buffers during their execution.

Notice that, between intervals 4 and 19, the total number of buffers allocated for all the threads is 12 (the initial 8 buffers plus the additional 4 assigned to thread 100). However, those threads that do not allocate extra buffers in those intervals keep their MPKI in a similar value than that of thread 100.

This is because these threads do not require any extra buffers for their private blocks and are accessing shared blocks previously fetched by Thread 100 to its assigned buffers during intervals 3 to 7. Moreover, the MPKI is not only kept low among the different threads under FOs-Mt, but also the overall MPKI is very close to that of the same application executed on a much larger (i.e. up to 4MB) shared conventional L2 cache organization, as it will be shown later in Section 6.5.

This example shows how the BAA properly addresses when a given thread requires additional buffers. Furthermore, the devised approach is also able to identify which threads do not need any additional cache space when they are accessing to shared data, and this is directly translated to a much more efficient use of the cache.

Buffer Release

FOS-Mt implements a simple release mechanism lying on time and thread activity criteria. A release mechanism is necessary as it allows feeding the buffer pool and prevents threads from unnecessarily holding the maximum number of buffers after a hungry phase. The proposed mechanism consists of two different conditions: i) first, the executing thread must neither have requested nor deallocated any other buffer in the last $Thr_{release}$ intervals; and ii) the thread's $BTAs_{weight}$ must be lower than $K \times Thr_{weight}$. By meeting these two conditions, we ensure that FOS-Mt only releases a buffer after a long-enough steady state and only in case that low performance losses can be introduced.

The LRU policy is used to choose which buffer is deallocated. The details about the deallocation process are already explained in Section 5.2.1.

Experimental results show that the proposed solution offers a reasonable performance, since the draining phase of the Buffer Release process represents on average a small fraction (i.e. around 1%) of the overall 8M-cycle interval length. Remember that this length determines the amount of cycles at which the buffer allocation and release algorithms are triggered. In summary, since the draining phase takes less than than 1% of the overall interval length, we can conclude that the impact on both energy and performance of the devised draining approach can be considered negligible.

Implementation Details in FOS architectures

The FOS architectures are complex organizations whose implementation is not trivial. In this section, we summarize some details to provide insights regarding the implementation of some of the features of these architectures.

First, the initial number of buffers turned-on for each core can be set by the hardware threshold Thr_{low}, which also indicates the minimum number of buffers that a single core can have assigned. To simplify the evaluation, all the experiments in this chapter assume 1 as the value of this threshold.

Secondly, as already explained in Section 5.2.3, it is not required to execute the BAA at fixed time intervals. We leverage the core stall cycles, when the ROB is blocked, to execute the BAA.

In addition, although not depicted in Figure 6.1 for simplicity reasons, FOS-Mt also counts with the *Buffer Allocation Logic* (BAL), located in a special node of the on-chip network. This small logic keeps track of which buffers are being assigned to each core, and manages the buffer allocation/deallocation requests. Its functionality is already described in Section 5.2.1.

Finally, we discuss the case of a context switch in FOS-Mt. In other words, what would occur in case a context switch event is triggered by the OS, so that the thread stops its execution and is evicted from the core. On such a case, one possible option would be to deallocate all the buffers previously assigned to that core but only one buffer and marked to be powered-off after a certain number of intervals (e.g. one OS quantum). However, if the evicted thread returns to resume its execution on the same core before the interval expires, the deallocated buffers are restored and assigned to the thread again. The key

idea behind this design is to prevent the thread from losing all its data in case it is evicted for a short time, e.g. a hardware interruption. Notice that this rationale also applies when running sequential workloads.

Multiprogram Multithreaded Workloads

The implementation of FOS-Mt allows the concurrent execution of multiple multithreaded workloads. When running such multiprogram workloads, FOS-Mt allocates banks according to the cache space requirements of the different threads, regardless of the application they belong to. However, regarding the management of shared blocks, notice that threads belonging to a given application would not access to banks allocated by the threads from other co-running application. Therefore, provided that there is enough cache space, the benefits of FOS-Mt with respect to conventional approaches when running multiprogram workloads composed of multithreaded applications would remain.

6.3 FOS-Mt Optical Network-on-Chip

One of the main lessons learnt during the design of the FOS architecture is that the underlying network that interconnects the different components of the architecture plays an important role sustaining performance, specially when accessing to the farthest cache buffers. Preliminary studies show that the NoC supporting FOS must i) be fast enough to avoid delaying hits in the buffer pool, and ii) provide low latency variability in order to guarantee that all the pool buffers present similar access time. To satisfy these requirements, the FOS architecture employs an Optical-NoC, since optical transmissions are fast even in relatively simple designs and the impact of distance in the network latency is much lower compared to their electrical counterparts.

As its predecessor, the FOS-Mt architecture takes the same solution and implements an ONoC to communicate the cache buffers, the compute nodes and the dedicated hardware structures. Although the main idea remains, due to the higher complexity of FOS-Mt with respect to FOS, some modifications have been added to the network design. This section presents the FOS-Mt customized ONoC, and discusses its new features.

6.3.1 FOS ONoC

The ONoC implemented in FOS-Mt keeps the ring-based topology already used in the FOS architecture. Despite having more components to be connected, an optical ring still suffices to meet the latency requirements, even using a reduced number of optical resources.

With the aim of preventing non-critical requests and notifications from delaying the access to the buffer pool, the network also keeps its multi-channel organization. In FOS-Mt, we propose a ring with three separated channels C_0, C_1 and C_2, which interconnects the L1 caches, the buffer pool and the FOS-Mt hardware structures. Communications on each channel take place on different waveguides, and each of them implements a different set of wavelengths working together so transmissions can take place in the three channels simultaneously. Note that wavelengths on each channel are used to carry distinct bits of the same message to the destination; in other words, increasing the number of wavelengths of a given channel directly translates to higher available bandwidth for the packets, and therefore reducing the packet latency. This means that, in FOS-Mt's ONoC, a single channel can not carry bits of different messages in parallel, but it is possible to transmit faster a message on each channel. Although the design is quite similar to that of FOS, the channels have been slightly modified to support the communication with the DNs and the coherence traffic. Channel C_0 performs transmissions from the L1 caches and PTAs to the pool as well as between PTAs and DNs. Since its traffic is mainly composed of requests and L1 writebacks, its bandwidth requirements are relatively low. Thus, this channel is provided with just 32 wavelengths. Channel C_1 delivers the requested blocks from the pool to the L1 caches. Incurring a high latency when delivering these blocks can be critical for performance; therefore, this channel is configured with 128 wavelengths. Finally, channel C_2 is employed to notify the PTAs and the DNs when writing and replacement operations are finished in the buffer pool. Since notification messages are small (4B), this channel is provided with just 8 wavelengths.

Sender nodes get channel access by arbitration. Optical token-based arbitration (see Section 4.3.1) is also used in FOS-Mt, since it guarantees collision-free transmissions and barely introduces extra overhead. Moreover, this arbitration mechanism has been already demonstrated to efficiently work in FOS.

As already explained in Chapter 4, token transmission latency mainly depends on the silicon lightspeed and the path length. The FOS-Mt ring keeps the same size as its predecessor, that is, 44.8 mm; therefore, token transmission latencies are also the same: from 100 to 500 ps, depending on the distance. A reduced nubmer of extra several wavelengths are still needed, in order for the sender to notify the receiver when a message is going to be transmitted and tune its resonators. Tuning operations are consistently assumed to take 400 ps delay.

Since most of the parameters are kept equal with respect to the ring implemented in FOS, it is expected that the overall transmission latency (conversions plus data transmission) will be similar. First, remember that electrical-to-optical and optical-to-electrical conversions introduce a delay by 9.5 ps/bit and the latter by 4.0 ps/bit, according to the literature (see Section 4.1.1). Next, data transmission is computed exactly as it was originally done in FOS. In the FOS systems, request and acknowledgment messages are 8B (64 bits) long, data messages are 72B (576 bits) long and notification messages are 4B (32 bits) long. Considering the number of wavelengths available on each channel (32, 128, 8) and 10 Gbps conversion speeds, each conversion requires 200 ps, 500 ps and 400 ps, respectively. Analogously, data transmission delay also depends on the length and the width of the channel. Since FOS-Mt also implements a 44.8 mm long ring, the head of the message takes 500 ps to reach the farthest node. After that, the remaining transmission latency depends on the message size and the channel bandwidth (i.e. the number of wavelengths). For instance, a 576-bit writeback message sent from an L1 node to the farthest buffer in the pool over channel C_0 takes 1400 ps $(500 + 900)$, while a 576-bit data message sent from the buffer to the same L1 node takes only 1000 ps $(500 + 500)$ because it is transmitted over the widest channel C_1.

Table 6.2 summarizes the discussed ONoC parameters and latencies. With these parameters and latencies, the whole transmission delay of any message assuming no contention varies between 2 and 4 core cycles for a 2 GHz core clock frequency, which is fast and uniform enough to accomplish the aforementioned FOS-Mt requirements. Note that the highest latency variation introduced because of distance is 500 ps, which means that the distance between transmitting nodes presents a limited impact on latency, which is one of the FOS-Mt NoC main requirements.

Table 6.2: Optical network-on-chip parameters and latencies.

Parameters	
Frequency	10 GHz
Wavelengths per channel	32 - 128 - 8 λ
Signal propagation	11.4 ps/mm
Modulation bandwidth	10 Gbps
Latencies (ps)	
Token transmission	Varying in range [100..500]
Microring tuning delay	400
Data modulation (64b-576b)	100 - 500
Trans. latency	Varying in range [100..900]

Table 6.3: Loss values of photonic components.

Component	**Value**	**Reference**
Laser Efficiency	5 dB	[47]
Coupler	1 dB	[141]
Waveguide path loss	0.1 dB/mm	[47]
Waveguide bend/cross	0.005/0.5 dB	[47]
Ring drop	1 dB	[141]
By/Through ring loss	0.001/0.1 dB	[141]
Photodetector	0.1 dB	[141]
Receiver sensitivity	-25 dB	[141]

6.3.2 Energy model

To complete the ONoC analysis, in this section we apply the energy model already presented before in Section 5.5.1. Notice that, although most parts of the optical network remain identical, the number of components has been increased, since there are more nodes, structures and buffers to interconnect.

The power consumption of the laser is computed using the model of Morris et al. that we presented in Chapter 5. The equation is as follows:

$$P_{laser} = P_{rx} + C_{loss} + M_s,$$

where P_{laser} is the laser power, P_{rx} is the receiver sensitivity, C_{loss} is the channel loss, and M_s is the system margin. The model is fed with the loss values listed in Table 6.3, and its outcome provides the minimum laser power to carry a signal strong enough to be received by the photodetectors in each node. These values are the same than those

employed in FOS, since they do not depend on the ONoC design but on the technology state-of-the-art.

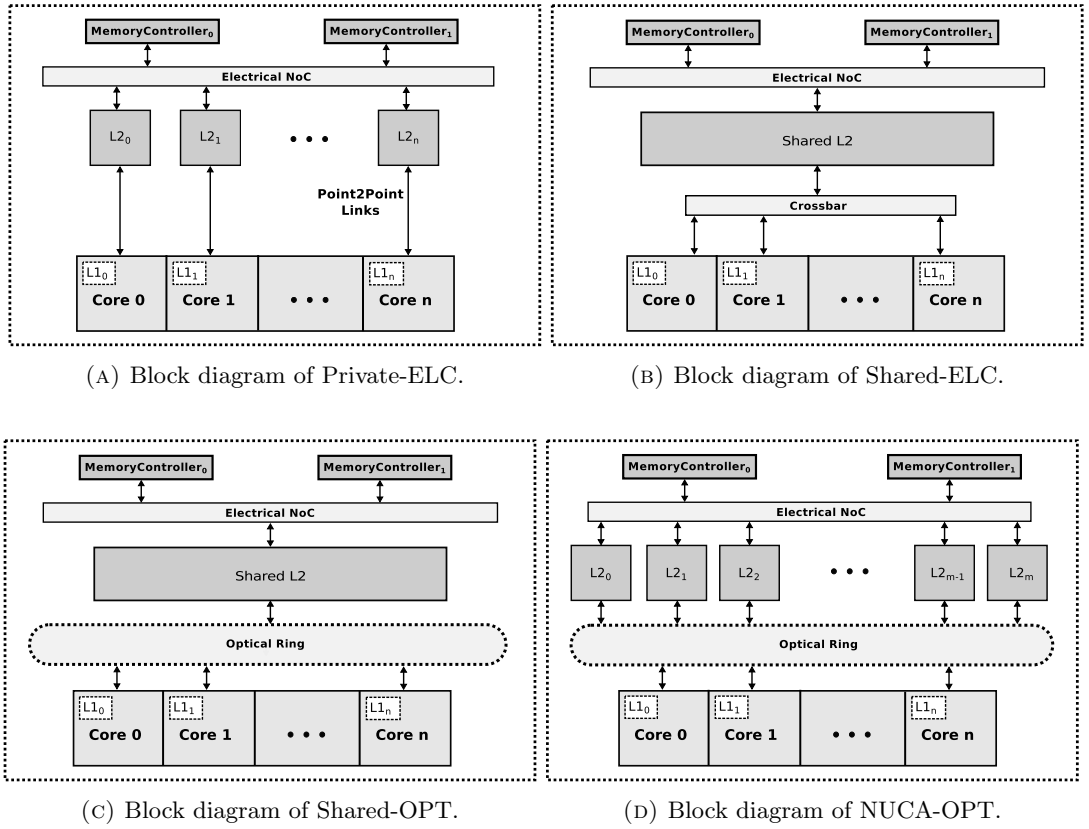
The FOS-Mt ONoC keeps the advantages of the previous design regarding energy efficiency. Aimed at reducing the power loss and the crosstalk noise power, the FOS-Mt ONoC supports up to 3 transmissions that can be performed in parallel, each of them involving a number of resonators equal to the 2× number of wavelengths allocated for the corresponding channel (32 for C_0, 128 for C_1 and 8 for C_2). This means that, at any time there can be at most 336 active resonators from a total amount of 6720.

The total energy per bit for the devised ONoC has been computed using the average number of transferred bits and execution time across the executed workloads. According to the model, the energy per bit consumed by the ONoC is up to 1.5 pJ/bit. In contrast, the energy dissipation value expected in electrical links is 0.25 pJ/bit (estimated with ORION 2.0 [143]). Further details on the energy results achieved by FOS-Mt are shown in Section 6.5.3.

6.4 Experimental Framework

We have widely extended the Multi2Sim simulation framework [119] code to model and evaluate our proposal. As stated in Chapter 3 and similarly done in Chapter 5, Multi2Sim has been linked to the DRAMSim2 framework [144], a hardware-validated DRAM simulator. Also, the CACTI v6.5 [125] tool has been used to estimate the energy consumption and access latency of the studied cache structures for a 32 nm technology node. Experiments have been carried out using a subset of benchmarks from the SPLASH3 2017 and the ALPBench 2005 suites [127, 147].

The evaluation methodology is as follows. Applications are run until the end of their execution, but statistics are gathered only on those intervals where multiple threads are running simultaneously. This methodology prevents the results from being affected by the initialization stage of the applications, where some applications typically execute only one thread and allocate a reduced amount of cache banks. Otherwise, the presented results would show lower average energy consumption.



(A) Block diagram of Private-ELC.

(B) Block diagram of Shared-ELC.

(C) Block diagram of Shared-OPT.

(D) Block diagram of NUCA-OPT.

FIGURE 6.4: Block diagrams of all the conventional baseline approaches.

6.4.1 Studied Approaches

FOS-Mt has been configured with a pool of n buffers of k size, and each buffer counts with 8 ways. In this work, experimental results consider k equal to 128KB and n equal to 4 times the number of cores; that is, the total L2 cache capacity for an 8-core multiprocessor is 4MB. Notice that, in order to keep a reasonable number of maximum buffers, the size of the buffers has increased with respect to the previous chapter; however, other combinations are also possible. To interconnect these 32 buffers with the memory controllers, FOS-Mt implements an electrical mesh that concentrates 4 buffers per node. Regarding the buffer allocation algorithm setup, a wide set of experiments has been performed to tune its parameters. These parameters could be farther refined with more experiments, however, this work presents the results with the parameter values that showed the best energy efficiency through most of the experiments: Thr_{min}=0.2, Thr_{MPKI}=0.05, Thr_{weight}=0.125.

To study where the achieved benefits come from, the proposal has been first compared

TABLE 6.4: Baseline system parameters.

Core	
Number of cores	8, OoO, 4 issue/commit width
Frecuency	2 GHz
ROB size	128 entries
Cache Hierarchy	
L1 Inst-Data cache	Private, 64KB, 8-way, 64Bytes, 1 cycles
L2	Private, 512KB, 16-way, 64Bytes, 8 cycles
Interconnect L1-L2	
Frequency	2 GHz
Bandwidth	64 Bytes/cycle
Main Memory & Memory Controller	
DRAM bus freq.	1600MHz
DRAM device	DDR4 (3200 Mtransfers/cycle) 8 banks
Latency	t_{RP}, t_{RCD}, t_{CL} 13.75ns each

with four different systems varying the cache hierarchy and the NoC, which are the components modified by our proposal. That is, different L2 cache organizations and L1-L2 interconnection networks have been considered. The remaining components of the architecture (e.g. memory controllers, L2 to main memory interconnection, etc.) are the same across all the studied systems, including FOS-Mt.

To refer to a given configuration, we use the identifier of the L2 cache organization combined with that of the L1-L2 interconnection network. For instance, *Shared-OPT* refers to a system with an L2 shared cache with optical NoC. To expose the differences among the four conventional systems that are compared to FOS-Mt, Figure 6.4 presents a block diagrams of these approaches. Additionally, with the aim of comparing FOS-Mt against other existing approaches, we have extended the simulation framework to implement the widely known *cache decay* [102] mechanism that powers off the cache lines that have not been accessed for a long-enough number of cycles. Below, the main characteristics of these five schemes are summarized.

- **Private-ELC:** In order to study the performance constraints associated to fixed-size private caches, we implement a scheme that presents 512KB L2 private caches (i.e the overall L2 cache capacity is 4MB), connected to the corresponding cores through point to point electrical links. Table 6.4 summarizes the main parameters of this baseline system, whose block diagram is presented in Figure 6.4a.

- **Shared-ELC:** To compare FOS-Mt to a scheme with a a common shared space that exhibits inter-application and inter-thread interference, we implement an approach with a 4MB shared and monolithic L2 cache connected to the first level cache (i.e. L1 cache) by an electrical crossbar, as can be seen in Figure 6.4b. For the energy study, this cache organization has been modeled in CACTI with 8 UCA sub-banks.

- **Shared-OPT:** To discern whether the performance enhancements come either from reducing the network latency or from improved performance of the shared organization, this configuration presents the same cache hierarchy as *Shared-ELC*, but the electrical crossbar is replaced by an optical ring whose technological parameters match those of the proposed ONoC (see Section 6.3.1). This means that the latencies associated to the optical components of this network (e.g. waveguides, photodetectors, etc.) are the same as those of FOS-Mt's ONoC. However, since this approach does not implement neither DNs nor BAL nodes, its ring implements only 2 channels of 32 and 128 wavelengths respectively. The block diagram of this approach is presented in Figure 6.4c.

- **NUCA-OPT:** A NUCA-based approach is introduced to replicate the FOS-Mt system without the FOS-Mt buffer allocation algorithm. That is, in this approach all the cache banks are active during the execution, and they are statically accessed to cores by address interleaving. The motivation behind this scheme is twofold: i) exposing clearly the energy efficiency benefits, in terms of cache management, brought by FOS-Mt; and ii) compare the benefits of FOS-Mt against those of a scheme with small and fast cache modules and similar network latencies. This scheme is configured as a pool of n NUCA modules of k size, where n and k match the values selected for the buffers in the FOS-Mt setup. Regarding the interconnection network, *NUCA-OPT* implements an optical ring with 2 channels similar to that of *Shared-OPT*, but interconnecting more network nodes (all the cores and NUCA modules). The block diagram of this approach is exposed in Figure 6.4d.

- **Decay-OPT:** Finally, a shared cache organization implementing a cache decay technique is modeled to compare our proposal against an existing energy-aware approach. To this end, a cache decay mechanism has been implemented in our

simulation framework. The decay technique uses two hardware counters to measure the number of cycles that a cache line has experienced without being accessed. When this number exceeds a given amount of cycles, the line is powered off and the block occupying it is evicted from the cache. In order to make a fair comparison between *Decay-OPT* and our proposal, we apply the decay mechanism to the L2 cache (*Shared-OPT* block diagram) with an optical interconnect.

6.5 Experimental Results

6.5.1 Impact of the PTA Size on Performance

As explained in Section 6.2, the number of Buffer Tags Arrays (BTAs) in the PTA limits the amount of buffers assigned to a given core. On the other hand, including a high number of BTAs per PTA increases access latency and area overhead. In other words, there is a trade-off between performance and PTA complexity. To study this trade-off, in this section we range the number of BTAs per PTA from 1 to 12 and calculate the impact on access latency, hardware complexity and performance.

Table 6.5 shows the access latency (both in nanoseconds and processor cycles) and the storage capacity of the BTAs for all the studied setups. In order to lookup a matching tag, all the BTAs in a PTA are accessed in parallel. Therefore, as a conventional cache, the access time depends on the accessed number of ways. That is, the number of BTAs per PTA multiplied by the number of ways per BTA (which is equal to 8, i.e. the number of ways in a buffer). Regarding the storage capacity, it is calculated as:

$$Capacity = BTAs\_per\_PTA \times tags\_per\_BTA \times bits\_per\_tag. \quad (6.1)$$

Table 6.5: Access time and capacity of the Buffer Tag Arrays.

#BTAs	T (ns)	T (cc)	Capacity (KB)
1	0.21	1	6.75
2	0.27	1	13.5
4	0.33	1	27
6	0.39	1	40.5
8	0.47	1	54
12	0.59	2	81

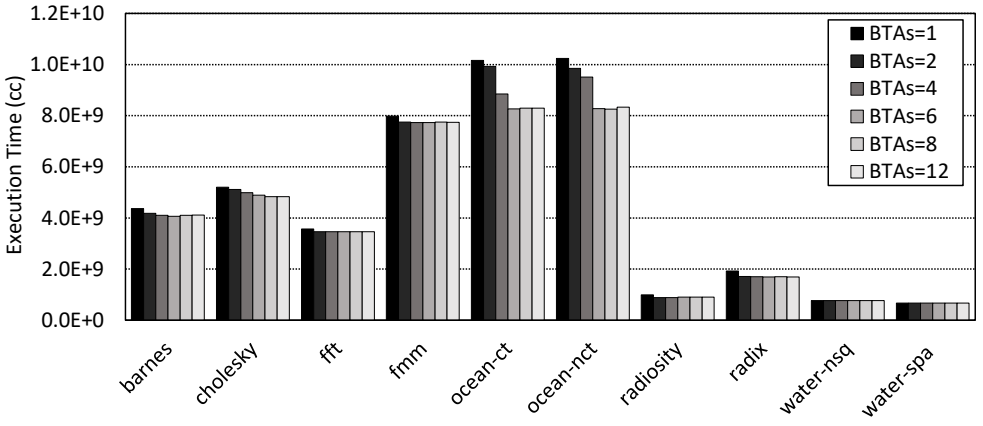


FIGURE 6.5: Execution Time.

For 128KB buffers composed of 64B lines, the number of tags per BTA is equal to 2048 (128KB/64B). Assuming 27-bit tags, this gives a BTA capacity of 6.75KB. Note that the total capacity of the PTA is the sum of that of the BTAs, the Shared Tags Table (STT) and the shadow tags needed by the Buffer Allocation Algorithm. Regarding the STT, we have checked different sizes and organizations, and found that a single STT with 1024-entries and 8-way associative is enough to avoid harming performance. Note that this is in line with other works [148] showing that in parallel workloads, most of the storage space is used by private blocks and, in comparison, the space needed for shared blocks is relatively small. Taking into account that a STT stores, for each entry, the buffer and way identifiers, the capacity of such a STT is equal to 4.375KB. With respect to the shadow tags, we applied a *Set Sampling* of 32 sets per BTA, which gives an additional overhead of 864B per BTA.

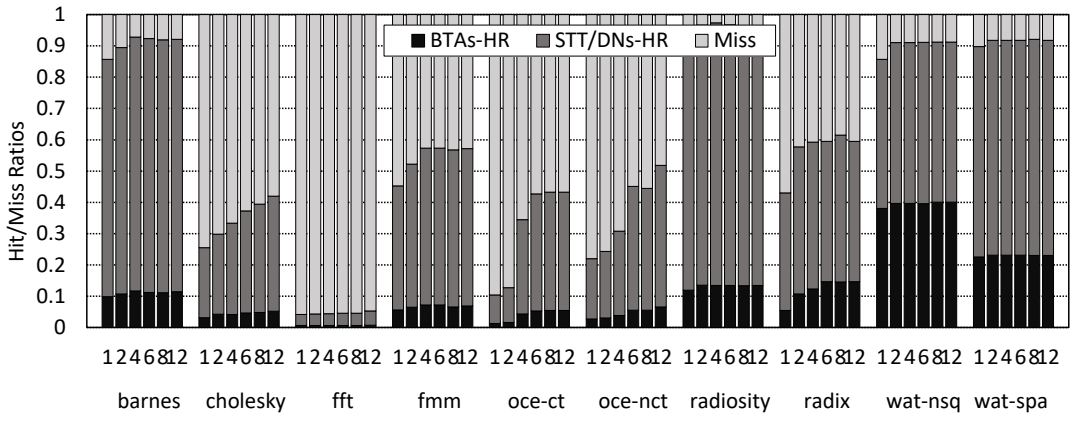


FIGURE 6.6: Distribution of access types.The number in the X axis below each bar indicates the number of BTAs.

To evaluate the sensitivity of performance to the number of BTAs, Figures 6.5 and 6.6 plot the impact on the execution time and on the distribution of the result of access (i.e. private hit, shared hit, and miss), respectively. Regarding the result of the access (Figure 6.6), it can be observed that in 6 out of 10 applications the hit rates drop significantly when the number of BTAs is less than 6. In the remaining applications, the miss rate is unaffected because the working set, and specially the shared data, fit in the buffers assigned to each thread. Next, as shown in Figure 6.5, performance is also affected, specially in `cholesky`, `ocean-ct` and `ocean-nct`. On the other hand, providing the PTAs with more than 6 BTAs does not translate to performance gains for any studied application. To sum up, we can conclude that these results show that 6 BTAs provide the best tradeoff between hardware complexity and performance. Therefore, from now on we will employ this setup in the remaining experiments.

6.5.2 Impact of the Optical Ring in FOS-Mt

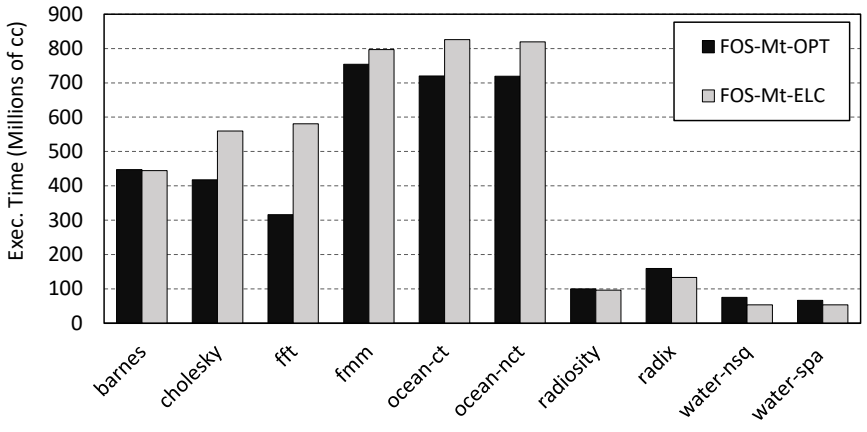


FIGURE 6.7: Execution time across the studied workloads for FOS-Mt with electrical crossbar and optical ring interconnects.

Regarding interconnects, as stated in Section 6.3, our proposal only requires a fast-enough network able to provide rather uniform access latency between the computing cores and the buffer pool. In this section, we evaluate FOS-Mt using both an optical ring (labeled as *FOS-Mt-OPT*) and an ideal electrical crossbar (*FOS-Mt-ELC*) to interconnect the L1 caches and the data buffers. We perform these experiments to investigate how the proposal would behave with a different network. Other approaches, like an

electrical mesh, could be also studied, although these solutions would strongly depend on the topology and would require a dedicated design.

Figure 6.7 shows the execution time across the studied workloads for FOS-Mt with electrical crossbar and optical ring interconnects As observed, in those workloads that introduce low traffic into the network like `water-spa`, `water-nsq` or `radiosity`, FOS-Mt-ELC is slightly the best performing approach. This is due to the small overhead introduced by the optical ring, which needs to arbitrate, tune the microrings and grant exclusive access to the ring before starting any transmission. This penalty is not suffered by the electrical crossbar approach, which under low-traffic and no contention offers a similar latency to that of the optical ring. When running more memory intensive applications like `ocean-ct`, `ocean-nct` and `ftt`, however, FOS-Mt-ELC becomes clearly the worst approach. The electrical crossbar presents worse performance in terms of latency than the optical ring. Moreover, it does not considers differentiated channels as the optical NoC, so transmissions in the critical path are more affected by contention. Therefore, we can conclude that the optical ring is more suitable to manage NoC contention than the electrical crossbar.

In summary, these results show that the devised ONoC presents the best performance on average across the studied applications, since it not only performs much better in memory intensive workloads than the studied electrical crossbar, but it also sustains the performance in compute intensive (and low NoC utilization) applications. Because of this reason, the following sections will only present results for the FOS-Mt-OPT approach.

6.5.3 Energy Evaluation

This section evaluates FOS-Mt's energy consumption and compares it with the approaches explained in Section 6.4. To this end, Figure 6.8 plots the energy consumption of the studied approaches normalized to the *Private-ELC* one. In the figures, the energy consumption is broken down into three main components: i) the energy consumption coming from the network side, that is, from electrical links and from optical rings; ii) the static energy consumed in the cache modules due to leakage currents; and iii) the dynamic energy consumed on every access to the cache modules. In the case of FOS-Mt,

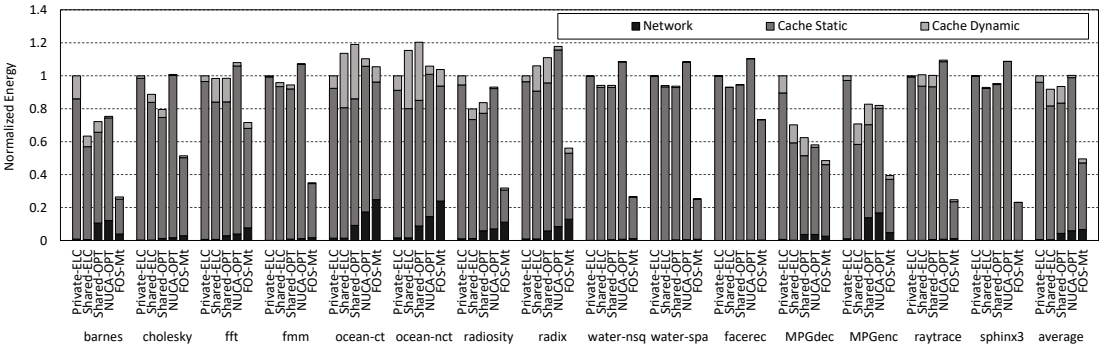


FIGURE 6.8: Normalized energy consumed by conventional approaches and FOS-Mt with respect to *Private-ELC*.

the latter value also considers the dynamic energy consumed in the PTA and the DN lookups.

Results presented in Figure 6.8 show that the static energy consumption of cache modules is clearly the component that dominates the overall energy consumption across all the studied cache organizations. However, in the case of FOS-Mt thanks to the number of cache buffers that it does not activate at run-time, these static energy consumption values are drastically reduced in 10 out of 15 applications.

To provide further insights on the energy savings of FOS-Mt, Figure 6.9 presents the average amount of cache space allocated by each application during its execution time. The figure shows that, on average, FOS-Mt allocates by 49% of the total cache space, varying from 1MB in the case of `water-spa` to almost 4MB (that is, the whole pool) in the cases of `ocean-ct` and `ocean-nct`. As a result, static energy savings are specially significant in applications with a low average number of active buffers like `fmm`, `radiosity`, `water-spansq`, `raytrace` or `sphinx3`, where FOS-Mt improves static energy consumption by up to 75% over the *Private-ELC* baseline. On the other hand, when running applications that need to turn on the whole cache space, FOS-Mt's static energy values match those obtained by other shared approaches. This behavior means that FOS-Mt is properly working, since it is correctly trading off performance and energy when needed.

The static energy differences among the remaining approaches are due to two main reasons. First, it can be observed that the 32 cache modules implemented in *NUCA-OPT* consume a higher amount of energy than the monolithic 4MB cache module implemented

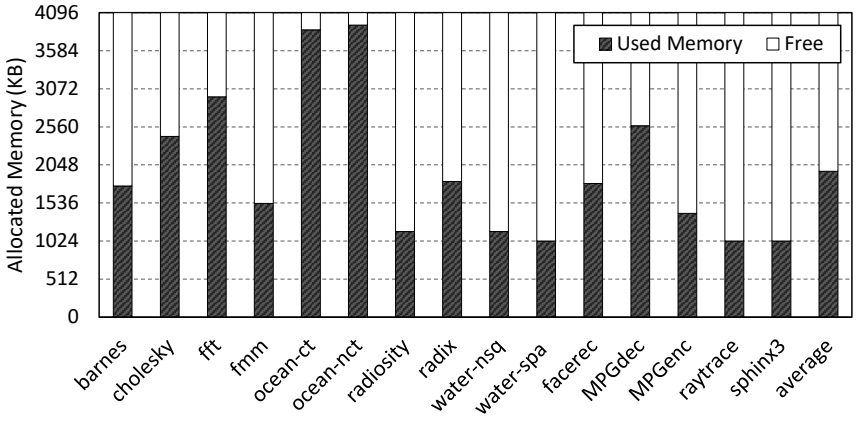


FIGURE 6.9: Average amount of cache space (KB) allocated during execution in FOS-Mt.

in *Shared-OPT/ELC*. The second reason is the execution time achieved by the studied approaches. For instance, in the case of `barnes`, *Private-ELC* presents the highest static energy consumption since the execution time is by 40% higher than that of the other approaches (see Section 6.5.4). The impact of the execution time on the overall static energy consumed can be also observed in `MPGdec` and `MPGenc`.

Regarding dynamic energy consumption, it can be observed that *NUCA-OPT* is the approach that presents the best results, closely followed by FOS-Mt. This is due to the low complexity of the 128KB cache modules employed in both cache organizations, which is directly translated to a lower energy spent on each access. Consequently, the large monolithic modules implemented in the *Shared-ELC* and *Shared-OPT* approaches present the highest energy consumption values of this component. Regarding FOS-Mt, notice that, despite implementing cache modules of the same size as those of *NUCA-OPT*, the PTA and the Directory Nodes introduce a relatively low overhead, which represents on average by about 4% of the total energy consumption. Moreover, this energy overhead is only suffered in the access to the tag array, thus the overall dynamic energy consumption in FOS-Mt is only by 45% higher than that of *NUCA-OPT* on average. Only in some applications like `ocean-ct` and `ocean-nct`, where FOS-Mt allocates a high amount of buffers, its dynamic energy consumption can be by 100% higher than that of *NUCA-OPT*. On the other hand, in applications that allocate a reduced amount of buffers like `raytrace`, the dynamic energy consumption of FOS-Mt matches that of *NUCA-OPT*.

Regarding the energy consumption coming from the network side, it is plotted at the bottom of each bar. It can be seen that, on average, the network energy consumption of the optical approaches is from 8 to 12 times higher than that of *Private-ELC* and *Shared-ELC*. This means that all the optical approaches experience an energy consumption increase by up to 10% with respect to the electrically connected configurations.

When comparing the optical approaches, it can be observed that the optical network of *Shared-OPT* presents a slightly lower energy consumption than the network of *NUCA-OPT*. This is due to the different number of nodes implemented on each network. *NUCA-OPT* implements 32 cache modules, and the amount of optical microrings needed to interconnect them is higher than in *Shared-OPT*, where only one cache module is connected to the network. FOS-Mt is the system that presents the highest network energy consumption since its ONoC implements an additional channel for control messages, as explained in Section 6.3. This channel, which is not needed in *NUCA-OPT* and *Shared-OPT*, requires additional microrings that increase the network energy consumption. In some cases, like `barnes` or `MPGenc`, however, the contention may rise in the optical channels due to the access to the L2 tags, so increasing the network latency. As a result, the energy consumed by the optical approaches may not be determined only by the number of optical resources. Notice that this effect does not happen in FOS-Mt since the buffer tags are located close to the cores.

In spite of being the optical approach with the highest network energy consumption, overall, FOS-Mt provides the highest energy savings due to its highest cache efficiency. These savings cannot be achieved with a NUCA because it does not allocate buffers to applications according to their needs as FOS-Mt does. Notice that this fact remains even for an electrically connected NUCA, since it would only save up to by 10% compared with *NUCA-OPT*.

Recall that FOS-Mt needs to fully interconnect every cache buffer with every L1 cache as well as providing a low and rather uniform access latency to the buffer pool; thus, from a performance perspective, a common electrical mesh is not suitable for FOS-Mt. Anyway, in case that FOS-Mt was implemented over a different kind of network (e.g. a high-performance electrical network) the energy savings achieved by the Buffer Allocation Algorithm would remain.

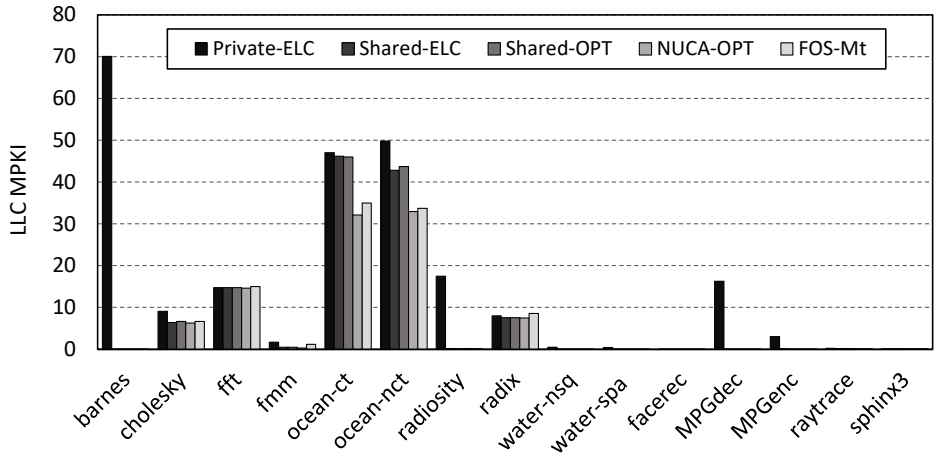


FIGURE 6.10: L2 Misses Per Kilo-Instruction.

6.5.4 Performance analysis

Figure 6.10 plots the L2 Misses Per Kilo Instruction (MPKI) of the studied approaches. Overall, the shared approaches are those that achieve the lowest MPKI values, which is explained by the fact that private caches cannot store enough data for some applications and because of the ping-pong effect of shared blocks in this kind of cache organizations. In general, *NUCA-OPT* is the best performing system for all applications, closely followed by FOS-Mt. Moreover, for `ocean-ct` and `ocean-nct`, *NUCA-OPT* and FOS-Mt reduce the MPKI compared to other shared setups, thanks to the reduction of conflict misses with respect to the monolithic shared approaches. This reduction is mainly due to the distribution of cache accesses among different cache buffers.

FOS-Mt's MPKI is only slightly higher than that of other shared approaches in `radix` and `fmm`. This occurs because these applications experience sudden sharp increases of their MPKIs during execution to which the Buffer Allocation Algorithm is unable to react.

Finally, Figure 6.11 presents the speedup that FOS-Mt and the other shared hierarchies achieve over *Private-ELC*. On average, and despite the fact that FOS-Mt is not an approach aimed at performance, all the shared approaches and FOS-Mt present similar performance values. Moreover, in the case of `barnes`, FOS-Mt even presents the best performance among the studied approaches, achieving a speedup by 36% over the baseline. The only application where FOS-Mt suffers a perceptible, although slight,

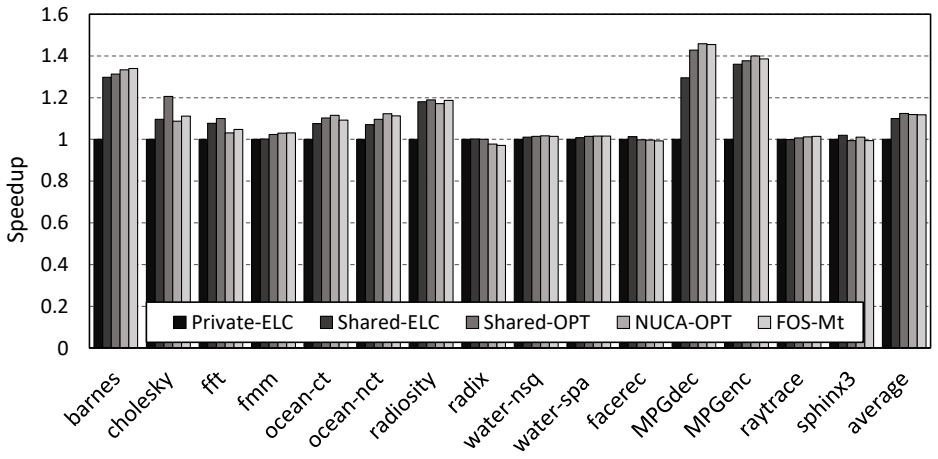


FIGURE 6.11: Speedup with respect to *Private-ELC*.

performance drop is `radix`. As seen in Figure 6.10, this corresponds to a small increase in the L2 MPKI.

6.5.5 Energy Efficiency

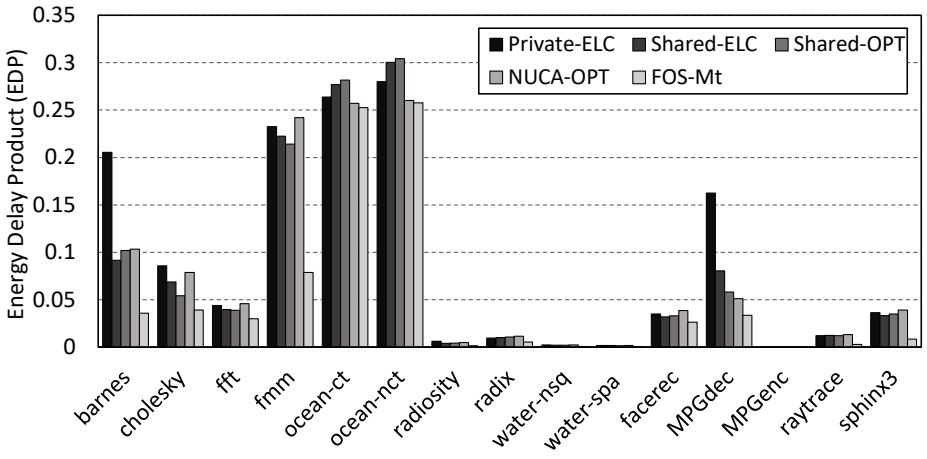


FIGURE 6.12: Energy Delay Product (EDP) of the studied approaches.

To study the energy efficiency of the studied approaches, Figure 6.12 presents the Energy Delay Product (EDP) across the executed workloads. As observed, FOS-Mt shows the lowest (i.e. the best) EDP values thanks to the achieved energy savings. Accordingly, in those applications where the Buffer Allocation Algorithm powers on the whole buffer pool (e.g. `ocean-ct` and `ocean-nct`), the EDP of FOS-Mt is almost the same as that of *NUCA-OPT*. Also, in some cases like `fft` or `cholesky`, it can be observed that the higher execution time of FOS-Mt translates to an EDP similar to that of *Shared-OPT*.

The discussed results prove the energy efficiency of FOS-Mt, whose energy savings not only are able to contain the overheads incurred by the implemented hardware structures and the optical network, but also reduce the overall energy consumption by up to 77% in the best case (i.e. `sphinx3`). Moreover, these energy savings do not come at the expense of performance, since the execution time is not higher than by 2% on average compared to the best performing approach. These results show that -Mt behaves as the most energy efficient organization among the studied approaches since it achieves the best EDP.

6.5.6 Comparison against Cache Decay Approaches

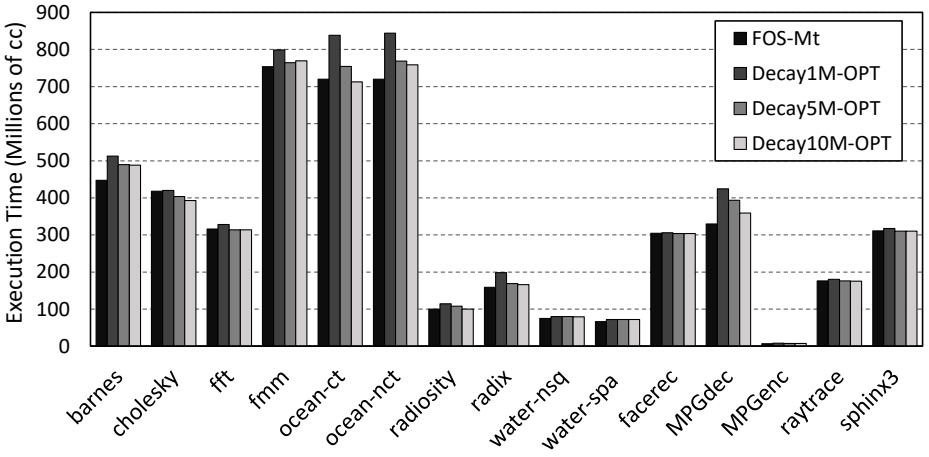


FIGURE 6.13: Execution time of FOS-Mt and various decay approaches.

Previous sections have discussed the performance and energy of FOS-Mt with respect to typical conventional cache hierarchies. Nevertheless, none of these approaches is specifically designed to save energy, as it is our proposal. In this section, FOS-Mt is compared against a well-known energy-saving technique like cache decay. Since the *life time* of a cache block stored in the L2 or a lower cache level may present huge variations among different applications, and even within blocks of the same application [149, 150], we study three cache decay configurations varying the decay interval time. Three main interval lengths have been explored: 1M-cycle, 5M-cycle and 10M-cycle. This number is used to refer to a given approach. For instance, in *Decay1M-OPT* a cache line that has not been accessed for the last million of processor cycles is turned-off.

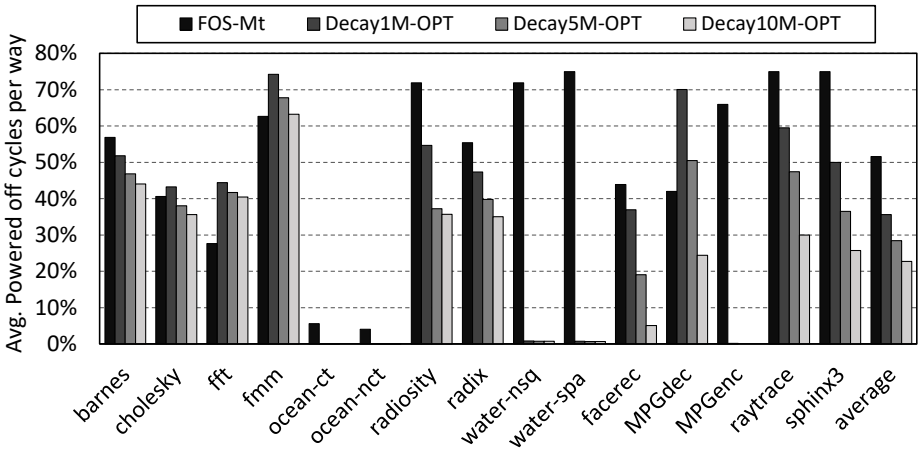


FIGURE 6.14: Average percentage of cycles powered off per way during the execution.

Figure 6.13 shows the execution time (in millions of clock cycles) achieved by the studied decay approaches and FOS-Mt. As expected, reducing the decay interval turns into an increase in the execution time. Therefore, the longer the decay interval, the better the performance; hence the best performing decay approach is *Decay10M-OPT*. Compared to this approach, FOS-Mt achieves better or similar performance in all the applications except `cholesky`.

Although increasing the decay interval helps improve performance, longer intervals imply a reduction in the average amount of time that cache lines are powered off through the execution. To quantify this adverse effect, Figure 6.14 presents the average percentage of time that a line is powered-off. On average, this percentage is as much as 51% in the case of FOS-Mt, while in the case of *Decay1M-OPT* (i.e. the less consuming decay approach) this percentage drops to 37%.

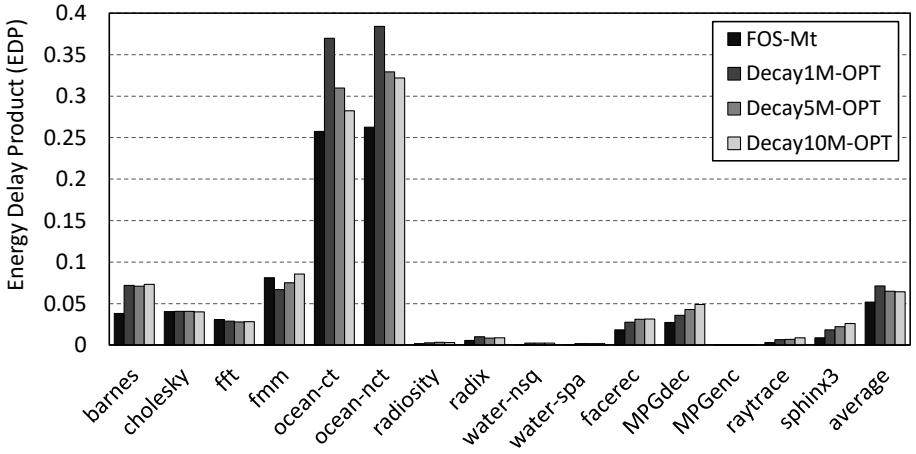


FIGURE 6.15: Energy Delay Product of FOS-Mt and Decay configurations.

Finally, Figure 6.15 plots the EDP of all the studied decay approaches together with FOS-Mt. As observed, FOS-Mt achieves the best EDP values in 13 out of 15 applications, with the only exceptions of `fft` and `fmm`. Regarding the decay variants, it can be appreciated that both *Decay5M-OPT* and *Decay10M-OPT* offer on average roughly the same EDP, while *Decay1M-OPT* is the approach that presents the worst EDP value. These results are explained due to the poorer performance of *Decay1M-OPT* with respect to the remaining approaches. Some applications show a very low EDP mainly due to their short execution time compared to the longer ones.

In conclusion, these results prove that FOS-Mt not only achieves higher energy savings on average than the studied decay approaches, but also presents the lowest execution time in most of the executed workloads. Moreover, the cache decay approach imposes many constraints from an implementation point of view, reducing the effective storage area mainly due to the additional circuitry required to cut the power supply to all the cache lines. However, working at a coarser granularity as FOS-Mt does, eases the implementation.

6.6 Summary

This chapter has discussed FOS-Mt, a novel cache organization for low-power multiprocessors that trades-off performance with energy consumption. FOS-Mt extends the previous FOS architecture, and incorporates new features like support of multithreaded applications and coherence management. As its predecessor, the proposed architecture conforms a single, sliced and flattened cache space that replaces conventional low level caches (e.g. L2 and beyond). FOS-Mt's cache space is divided into relatively small buffers (e.g. 128KB), which are dynamically allocated by the execution cores at run time. These buffers are much larger than individual cache lines, which eases the implementation of practical energy aware approaches that work at this granularity.

Experimental results have shown that FOS-Mt reduces by up to 78% the static energy consumption due to leakage currents in the cache modules. Thanks to the small size of the FOS-Mt's buffers, dynamic energy savings are also achieved, since FOS-Mt reduces this component by a factor of 4× with respect to a shared cache of the same size. Finally, when compared with other conventional approaches like NUCA, shared or private

caches of the same size, FOS-Mt presents similar performance, only suffering a slight performance degradation of by 1%. Although the widely known *Cache Decay* approach imposes serious implementation constraints from a practical perspective, we have also compared FOS-Mt against this approach from an energy efficiency perspective. Experimental results show that FOS-Mt improves EDP on average by 19.3% over the best decay interval across the studied workloads, being in some cases up to 48%.

Chapter 7

Conclusions

This thesis has focused on improving resource sharing in current chip multiprocessors. Two major system components, and their impact on performance and energy consumption, have been thoroughly studied: the cache hierarchy and the network on-chip. In this regard, this dissertation has proposed a novel cache organization aimed at reducing energy consumption in low-power CMPs, and explored silicon photonics as a viable alternative to face communication challenges in current and future multicores.

This chapter summarizes the main contributions of the dissertation, discusses future work directions, and enumerates the scientific publications related with the proposed approaches.

7.1 Contributions

Chip multiprocessors have demonstrated to be an effective solution to the power consumption wall of previous monolithic single-threaded processors. Although multicores efficiently exploit thread level parallelism to achieve better performance, the fact of sharing resources among the co-running threads (or applications) makes to rise contention at the shared resources. The most critical shared resources in current multicores are the main memory bandwidth, the LLC, and the network on-chip. To improve the efficiency of current and future multicores, this dissertation has addressed specific issues that impact on two of these resources: the LLC and the NoC.

Regarding the network on-chip, Chapter 4 proposes silicon photonics as a viable candidate to replace or complement existing interconnects in multicores. To support this claim, in this chapter we quantified the deviation that wrong optical models could present when integrated in detailed CMP simulation environments, and concluded that employing accurate models when simulating this technology is of paramount importance to obtain representative results. The analysis presented in Chapter 4 can be divided in two parts: i) description; and ii) evaluation. First, to provide the required background on the matter, the chapter described each component needed to fully implement an ONoC and integrate it in a silicon die. In particular, it was explained how each of these components is modeled, as well as the impact of each of them on the transmission latency. Then, for experimental evaluation, two approaches were modeled and implemented in a detailed simulation framework: the former, a realistic optical NoC composed of two rings and featuring a state-of-the-art arbitration technique; the latter, the same optical NoC with no arbitration. In order to extend the validness of the study for future scenarios, experiments were carried out using 64 and 160 maximum wavelengths. Experimental results show that the variation in average network latency between the two approaches can be as much as three orders of magnitude, presenting IPC deviations higher than 10% in some cases. Additionally, a state-of-the-art power consumption model stated that the realistic approach increases network energy consumption by 3% in the best case.

Regarding the LLC, Chapter 5 presented FOS, a novel cache organization and management approach to trade-off performance for energy consumption. FOS replaces conventional low level (e.g., L2 and L3) caches with a single level consisting of a pool of cache

buffers, and introduces a novel management approach especially suited for low-power processors. These buffers are much larger (i.e. $64-128\ KB$) than individual cache lines, which eases the implementation of energy-aware techniques working at this granularity. FOS has been implemented and evaluated with an underlying ONoC, since it provides rather fast and uniform access latency to the buffers. Experimental results showed that FOS reduces both dynamic and static energy consumption. Compared to a conventional shared cache with the same storage capacity, FOS reduces the static energy consumption by up to 60%. Dynamic energy consumption is also significantly reduced, by a up to 4× factor over a conventional shared cache of the same capacity. Moreover, FOS energy savings do not come at the expense of performance, since moderate performance improvements come from the network and the cache side. That allows FOS to achieve similar performance to an optically connected NUCA cache, despite using on average only 50% of the cache space.

Motivated by the promising results achieved in Chapter 5, in Chapter 6 we extended the approach to support multithreaded applications. This approach entails multiple challenges, such as designing new buffer allocation policies or efficiently managing the cache coherence. In order to address these challenges, new hardware structures, namely *Private Tag Array* and *Directory Nodes*, have been implemented in the simulation framework. These structures keep track of the location and state of the data blocks, easing the coherence management and reducing the average latency. Additionally, a new *Buffer Allocation Algorithm* has been devised aimed at exploiting the locality of the shared data in the active buffers. Putting all these new features together, experimental results showed that FOS-Mt reduces by up to 78% the static energy consumption due to leakage currents in the cache modules and by up to a 4× factor the dynamic energy consumption. Compared to existing conventional approaches like NUCA architectures, shared or private caches of the same size, FOS-Mt presents similar performance, only suffering minor performance degradation (up to 1% on average). The proposal has been also compared to the widely known Cache Decay approach, a work of reference in the field of cache energy savings. In this study, FOS-Mt improves EDP on average by 19.3% over the best decay interval across the studied workloads, being in some cases up to 48%.

7.2 Future Directions

The proposals summarized in this dissertation explore the use of optical interconnects as a useful tool for computer architects. Future directions of this idea are directly related to extending the approach to other shared resources or other architectures. An interesting idea might be using optical interconnects together with prefetching, managing the available wavelengths according to the prefetching policy. On the other hand, GPU architectures are an interesting field of study for optical interconnects due to the communication bottlenecks they are exposed to. Finally, following a different direction, we also plan to evaluate our FOS proposals without leveraging optical interconnects but other existing low-latency electrical networks.

7.3 Publications

The following papers related with this dissertation were submitted and accepted for publication in different international journals and conferences with peer review.

Journals:

- J. Puche, S. Petit, M. E. Gómez, and J. Sahuquillo. FOS: a low-power cache organization for multicores. *The Journal of Supercomputing* (JS), volume 75, issue 10, pages 6542-6573, 2019.
- J. Puche, S. Petit, M. E. Gómez, and J. Sahuquillo. An efficient cache flat storage organization for multithreaded workloads for low power processors. *Future Generations Computer Systems* (FGCS), volume 110, pages 1037-1054, 2020.

Conferences:

- J. Puche, S. Lechago, S. Petit, M. E. Gómez, and J. Sahuquillo. Accurately Modeling a Photonic NoC in a Detailed CMP Simulation Framework. In *Proceedings of the International Conference on High Performance Computing Simulation* (HPCS), pages 387-394, Innsbruk, Austria, 2016.

In addition, other related papers have been published in international summer schools and domestic conferences:

- J. Puche, S. Petit, M. E. Gómez, and J. Sahuquillo. Flat On-chip Storage: A Novel Cache Organization for Low Power Processors. In *Proceedings of the 14th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems* (ACACES), Fiuggi, Italy, 2018.

- J. Puche, J. Duro, S. Petit, M. E. Gómez, and J. Sahuquillo. Estudio de exploración sobre la aplicación de la tecnología fotónica en NoCs. In *Actas de las XXVI Jornadas de Paralelismo* (JP), pages 361-369, Córdoba, Spain, 2015.

- J, Duro, J. Puche, S. Petit, M. E. Gómez, and J. Sahuquillo. Explotación de la Localidad de Banco en Cargas Multiprogramadas. In *Actas de las XXVI Jornadas de Paralelismo* (JP), pages 305-312, Córdoba, Spain, 2015.

- J. Puche, S. Lechago, S. Petit, M. E. Gómez, and J. Sahuquillo. Modelado realista de una NoC fotónica en un entorno de simulación CMP detallado. In *Actas de las XXVII Jornadas de Paralelismo* (JP), pages 415-420, Salamanca, Spain, 2016.

- J. Puche, S. Petit, M. E. Gómez, and J. Sahuquillo. FOS: Una nueva organización de cache con interconexión fotónica. In *Actas de las XXVIII Jornadas de Paralelismo* (JP), pages 283-292, Málaga, Spain, 2017.

- J. Puche, S. Petit, M. E. Gómez, and J. Sahuquillo. P-FOS: Una Organización de Cache Eficiente para Procesadores Multinúcleo. In *Actas de las XXIX Jornadas de Paralelismo* (JP), pages 135-141, Teruel, Spain, 2018.

- J. Puche, S. Petit, M. E. Gómez, and J. Sahuquillo. FOS-Mt: Una Organización Eficiente de Cache para Aplicaciones Paralelas en Procesadores de Bajo Consumo. In *Actas de las XXX Jornadas de Paralelismo* (JP), pages 196-202, Badajoz, Spain, 2019.

All publications listed above are exclusively related with this thesis. The Ph.D. candidate has contributed to the design, implementation and evaluation of the proposals, which includes the discussion of early designs, the implementation of the devised algorithms, the preparation of the experimental framework, the execution of the experiments,

the processing and analysis of the results, the writing of the paper drafts for publication, and the presentation of the papers in the national and international conferences. Throughout these iterative processes, the co-authors have strongly supported the candidate, providing experienced advice to improve the work and make it evolve into this dissertation.

In addition, the author of this dissertation has also participated in the development of research work analyzing the impact of resource sharing in the major components of cloud systems. This work, therefore, can be considered indirectly related with the work developed in this thesis. Below, the papers summarizing the main results of the developed work are listed.

- L. Pons, J. Feliu, J. Puche, C. Huang, S. Petit, J. Pons, M. E. Gómez, and J. Sahuquillo. Understanding Cloud Workloads Performance in a Production like Environment. *Journal of Parallel and Distributed Computing* (JPDC), submitted to Journal of Parallel and Distributed Computing.

- L. Pons, J. Feliu, J. Puche, C. Huang, S. Petit, J. Pons, M. E. Gómez, and J. Sahuquillo. Cloud White: A Production System Approach for Estimating QoS Degradation in the Public Cloud. *Future Generations Computer Systems* (FGCS), submitted to Future Generations Computer Systems.

References

[1] G. E. Moore. Cramming more components onto integrated circuits. *IEEE Solid-State Circuits Society Newsletter*, 1965.

[2] Robert H. Dennard, Fritz H. Gaensslen, Hwa nien Yu, V. Leo Rideout, Ernest Bassous, Andre, and R. Leblanc. Design of ion-implanted mosfets with very small physical dimensions. *IEEE J. Solid-State Circuits*, 1974.

[3] J. Doweck, W. Kao, A. K. Lu, J. Mandelblat, A. Rahatekar, L. Rappoport, E. Rotem, A. Yasin, and A. Yoaz. Inside 6th-generation intel core: New microarchitecture code-named skylake. *IEEE Micro*, 37(2):52–62, 2017.

[4] B. Sinharoy, J. A. Van Norstrand, R. J. Eickemeyer, H. Q. Le, J. Leenstra, D. Q. Nguyen, B. Konigsburg, K. Ward, M. D. Brown, J. E. Moreira, D. Levitan, S. Tung, D. Hrusecky, J. W. Bishop, M. Gschwind, M. Boersma, M. Kroener, M. Kaltenbach, T. Karkhanis, and K. M. Fernsler. Ibm power8 processor core microarchitecture. *IBM Journal of Research and Development*, 2015.

[5] S. Y. Borkar, P. Dubey, K. C. Kahn, D. J. Kuck, H. Mulder, S. S. Pawlowski, and J. R. Rattner. Platform 2015: Intel processor and platform evolution for the next decade. *Platform 2015*, April 2005.

[6] A. Sodani, R. Gramunt, J. Corbal, H. S. Kim, K. Vinod, S. Chinthamani, S. Hutsell, R. Agarwal, and Y. C. Liu. Knights landing: Second-generation intel xeon phi product. *IEEE Micro*, 2016.

[7] Yatin Hoskote, Sriram R. Vangal, Arvind Singh, Nitin Borkar, and Shekhar Borkar. A 5-ghz mesh interconnect for a teraflops processor. *IEEE Micro*. doi: 10.1109/MM.2007.77.

[8] Sriram R. Vangal, Jason Howard, Gregory Ruhl, Saurabh Dighe, Howard Wilson, James Tschanz, David Finan, Arvind Singh, Tiju Jacob, Nitin Borkar, and Shekhar Borkar. An 80-tile sub-100-w teraflops processor in 65-nm cmos, 2010.

[9] M. A. Anders. High-performance energy-efficient noc fabrics: Evolution and future challenges. In *Procs. of the 8th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS 2014.

[10] D.A.B. Miller. Device requirements for optical interconnects to silicon chips. *Procs. of the IEEE*, 2009. doi: 10.1109/JPROC.2009.2014298.

[11] Hasitha Jayatilleka, Hossam Shoman, Lukas Chrostowski, and Sudip Shekhar. Photoconductive heaters enable control of large-scale silicon photonic ring resonator circuits. *Optica*, 2019. doi: 10.1364/OPTICA.6.000084.

[12] Harish Subbaraman, Xiaochuan Xu, Amir Hosseini, Xingyu Zhang, Yang Zhang, David Kwong, and Ray T. Chen. Recent advances in silicon-based passive and active optical interconnects. *Opt. Express*, 2015. doi: 10.1364/OE.23.002487.

[13] G. H. Duan, C. Jany, A. L. Liepvre, A. Accard, M. Lamponi, D. Make, P. Kaspar, G. Levaufre, N. Girard, F. Lelarge, J. M. Fedeli, A. Descos, B. Ben Bakir, S. Messaoudene, D. Bordel, S. Menezo, G. de Valicourt, S. Keyvaninia, G. Roelkens, D. Van Thourhout, D. J. Thomson, F. Y. Gardes, and G. T. Reed. Hybrid iii–v on silicon lasers for photonic integrated circuits on silicon. *IEEE Journal of Selected Topics in Quantum Electronics*, 2014. doi: 10.1109/JSTQE.2013.2296752.

[14] G. H. Duan, X. Pommarede, G. Levaufre, A. Shen, D. Carrara, N. Girard, A. Gallet, D. Make, G. Glastre, J. Decobert, F. Lelarge, R. Brenot, S. Olivier, C. Jany, S. Malhouitre, and B. Charbonnier. Hybrid ili-v/silicon photonic integrated circuits for optical communication applications. In *IEEE 13th International Conference on Group IV Photonics (GFP)*, 2016. doi: 10.1109/GROUP4.2016.7739136.

[15] Guang-Hua Duan, Jean-Marc Fedeli, Shahram Keyvaninia, and Dave Thomson. 10 gb/s integrated tunable hybrid iii-v/si laser and silicon mach-zehnder modulator. In *European Conference and Exhibition on Optical Communication*, 2012. doi: 10.1364/ECEOC.2012.Tu.4.E.2.

[16] G. H. Duan, C. Jany, A. Le Liepvre, M. Lamponi, A. Accard, F. Poingt, D. Make, F. Lelarge, S. Messaoudene, D. Bordel, J. M. Fedeli, S. Keyvaninia, G. Roelkens, D. Van Thourhout, D. J. Thomson, F. Y. Gardes, and G. T. Reed. Integrated hybrid iii x2013;v/si laser and transmitter. In *International Conference on Indium Phosphide and Related Materials*, IPRM 2012. doi: 10.1109/ICIPRM.2012.6403306.

[17] Yu Li, Yu Zhang, Lei Zhang, and Andrew W. Poon. Silicon and hybrid silicon photonic devices for intra-datacenter applications: state of the art and perspectives. *Photon. Res.*, 2015. doi: 10.1364/PRJ.3.000B10.

[18] C. Chen and A. Joshi. Runtime management of laser power in silicon-photonic multibus noc architecture. *IEEE Journal of Selected Topics in Quantum Electronics*, pages 3700713–3700713, 2013. doi: 10.1109/JSTQE.2012.2228170.

[19] Y. Demir and N. Hardavellas. Ecolaser: An adaptive laser control for energy-efficient on-chip photonic interconnects. In *2014 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2014. doi: 10.1145/2627369.2627620.

[20] Yigit Demir and Nikos Hardavellas. Parka: Thermally insulated nanophotonic interconnects. In *Procs. of the 9th International Symposium on Networks-on-Chip*, NOCS 2015, . doi: 10.1145/2786572.2786597.

[21] Kishore Padmaraju, Johnnie Chan, Long Chen, Michal Lipson, and Keren Bergman. Thermal stabilization of a microring modulator using feedback control. *Opt. Express*, 2012. doi: 10.1364/OE.20.027999.

[22] T. Zhang, J. L. AbellÃ¡n, A. Joshi, and A. K. Coskun. Thermal management of manycore systems with silicon-photonic networks. In *Design Automation Test in Europe Conference Exhibition*, DATE 2014. doi: 10.7873/DATE.2014.320.

[23] H. Li, A. Fourmigue, S. Le Beux, X. Letartre, I. O'Connor, and G. Nicolescu. Thermal aware design method for vcsel-based on-chip optical interconnect. In *Design Automation Test in Europe Conference Exhibition*, DATE 2015.

[24] Ansheng Liu, Ling Liao, Doron Rubin, Hat Nguyen, Berkehan Ciftcioglu, Yoel Chetrit, Nahum Izhaky, and Mario Paniccia. High-speed optical modulation based

on carrier depletion in a silicon waveguide. *Opt. Express*, 2007. doi: 10.1364/OE.15.000660.

[25] D. J. Thomson, F. Y. Gardes, Y. Hu, G. Mashanovich, M. Fournier, P. Grosse, J-M. Fedeli, and G. T. Reed. High contrast 40gbit/s optical modulation in silicon. *Opt. Express*, 2011. doi: 10.1364/OE.19.011507.

[26] Po Dong, Long Chen, Chongjin Xie, Lawrence L. Buhl, and Young-Kai Chen. 50-gb/s silicon quadrature phase-shift keying modulator. *Opt. Express*, 2012. doi: 10.1364/OE.20.021181.

[27] R. Soref and B. Bennett. Electrooptical effects in silicon. *IEEE Journal of Quantum Electronics*, 1987. doi: 10.1109/JQE.1987.1073206.

[28] Graham Reed, Goran Mashanovich, Frederic Gardes, Milos Nedeljkovic, Yaxi Hu, David Thomson, Ke Li, Peter Wilson, Sheng-Wen Chen, and Shawn Hsu. Recent breakthroughs in carrier depletion based silicon optical modulators. *Nanophotonics*, 2014. doi: 10.1515/nanoph-2013-0016.

[29] Po Dong, Sethumadhavan Chandrasekhar, , Xiang Liu, Lawrence Lee Buhl, Ricardo Aroca, Yves Baeyens, and Young-Kay Chen. 224-gb/s pdm-16-qam modulator and receiver based on silicon photonic integrated circuits. Optical Society of America, 2013. doi: 10.1364/NFOEC.2013.PDP5C.6.

[30] George Kurian, Jason E. Miller, James Psota, Jonathan Eastep, Jifeng Liu, Jurgen Michel, Lionel C. Kimerling, and Anant Agarwal. Atac: A 1000-core cache-coherent processor with on-chip optical network. In *Procs. of the 19th International Conference on Parallel Architectures and Compilation Techniques*, PACT 2010. doi: 10.1145/1854273.1854332.

[31] Yi Xu, Jun Yang, and R. Melhem. Tolerating process variations in nanophotonic on-chip networks. In *Procs. of the 39th Annual International Symposium on Computer Architecture*, ISCA 2012. doi: 10.1109/ISCA.2012.6237013.

[32] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N.P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R.G. Beausoleil, and J.H. Ahn. Corona: System implications of emerging nanophotonic technology. In *Procs. of the*

35th International Symposium on Computer Architecture, ISCA 2008, . doi: 10.1109/ISCA.2008.35.

[33] Yan Pan, Prabhat Kumar, John Kim, Gokhan Memik, Yu Zhang, and Alok Choudhary. Firefly: Illuminating future network-on-chip with nanophotonics. *SIGARCH Comp. Arch. News*, 2009. doi: 10.1145/1555815.1555808.

[34] Cheng Li, M. Browning, P.V. Gratz, and S. Palermo. Luminoc: A power-efficient high-performance photonic network-on-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2014. doi: 10.1109/TCAD.2014.2320510.

[35] Yan Pan, J. Kim, and G. Memik. Flexishare: Channel sharing for an energy-efficient nanophotonic crossbar. In *Procs. of the IEEE 16th International Symposium High Performance Computer Architecture*, HPCA 2010. doi: 10.1109/HPCA.2010.5416626.

[36] Mark J. Cianchetti, Joseph C. Kerekes, and David H. Albonesi. Phastlane: A rapid transit optical routing network. In *Procs. of the 36th Annual International Symposium on Computer Architecture*, ISCA 2009. doi: 10.1145/1555754.1555809.

[37] A. Joshi, C. Batten, Y. Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic. Silicon-photonic clos networks for global on-chip communication. In *Third International Symposium on Networks-on-Chips*, NOCS 2009. doi: 10.1109/NOCS.2009.5071460.

[38] Yu-Hsiang Kao and H. Jonathan Chao. BLOCON: A bufferless photonic clos network-on-chip architecture. In *Fifth ACM/IEEE International Symposium on Networks-on-Chip*, NOCS 2011. doi: 10.1145/1999946.1999960.

[39] Zheng Li, Moustafa Mohamed, Xi Chen, Hongyu Zhou, Alan Rolf Mickelson, Li Shang, and Manish Vachharajani. Iris: A hybrid nanophotonic network design for high-performance and low-power on-chip communication. *Journal on Emerging Technologies in Computing Systems (JETC)*, 2011. doi: 10.1145/1970406.1970410.

[40] R. Morris and A. K. Kodi. Exploring the design of 64- and 256-core power efficient nanophotonic interconnect. *IEEE Journal of Selected Topics in Quantum Electronics*, 2010. doi: 10.1109/JSTQE.2009.2038075.

[41] S. Pasricha and S. Bahirat. Opal: A multi-layer hybrid photonic noc for 3d ics. In *16th Asia and South Pacific Design Automation Conference*, ASP-DAC 2011. doi: 10.1109/ASPDAC.2011.5722211.

[42] Shirish Bahirat and Sudeep Pasricha. Meteor: Hybrid photonic ring-mesh network-on-chip for multicore architectures. *ACM Trans. Embed. Comput. Syst.*, 2014. doi: 10.1145/2567940.

[43] S. Pasricha and N. Dutt. Orb: An on-chip optical ring bus communication architecture for multi-processor systems-on-chip. In *13th Asia and South Pacific Design Automation Conference*, ASP-DAC 2008. doi: 10.1109/ASPDAC.2008.4484059.

[44] D. Vantrease, N. Binkert, R. Schreiber, and M.H. Lipasti. Light speed arbitration and flow control for nanophotonic interconnects. In *Procs of the 42nd Annual IEEE/ACM International Symposium Microarchitecture*, MICRO 2009, .

[45] Antonio García, Ricardo Fernández, Jose M. García, and Sandro Bartolini. Managing resources dynamically in hybrid photonic-electronic networks-on-chip. *Concurrency and Computation: Practice and Experience*, 2014. doi: 10.1002/cpe.3332.

[46] S. Bartolini and P. Grani. A simple on-chip optical interconnection for improving performance of coherency traffic in cmps. In *15th Euromicro Conference on Digital System Design*, 2012. doi: 10.1109/DSD.2012.13.

[47] Sebastian Werner, Javier Navaridas, and Mikel Lujan. Designing low-power, low-latency networks-on-chip by optimally combining electrical and optical links. In *Procs of the IEEE Int. Symp. of High Performance Computer Architecture*, HPCA 2017. doi: 10.1109/HPCA.2017.23.

[48] Amir Kavyan Kavyan Ziabari, Jose L. Abellán, Rafael Ubal, Chao Chen, Ajay Joshi, and David Kaeli. Leveraging silicon-photonic noc for designing scalable gpus. In *Procs. of the 29th ACM on International Conference on Supercomputing*, ICS 2015. Association for Computing Machinery (ACM). doi: 10.1145/2751205.2751229.

[49] Henry Cook, Miquel Moreto, Sarah Bird, Khanh Dao, David A. Patterson, and Krste Asanovic. A hardware evaluation of cache partitioning to improve utilization and energy-efficiency while preserving responsiveness. In *Procs. of the 40th Annual*

International Symposium on Computer Architecture, ISCA 2013. Association for Computing Machinery. doi: 10.1145/2485922.2485949.

[50] M.K. Qureshi and Y.N. Patt. Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches. In *Procs. of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 2006. doi: 10.1109/MICRO.2006.49.

[51] Seongbeom Kim, Dhruba Chandra, and D. Solihin. Fair cache sharing and partitioning in a chip multiprocessor architecture. In *Procs. of the 13th International Conference on Parallel Architecture and Compilation Techniques*, PACT 2004, . doi: 10.1109/PACT.2004.1342546.

[52] Shekhar Srikantaiah, Mahmut Kandemir, and Mary Jane Irwin. Adaptive set pinning: Managing shared caches in chip multiprocessors. In *Procs. of the 13th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2008. doi: 10.1145/1346281.1346299.

[53] Julio Sahuquillo and Ana Pont. Splitting the data cache: a survey. *IEEE Concurrency*, 2000. doi: 10.1109/4434.865890.

[54] Jude A. Rivers, Edward S. Tam, Gary S. Tyson, Edward S. Davidson, and Matthew K. Farrens. Utilizing reuse information in data cache management. In *Procs. of the 12th international conference on Supercomputing*, ICS 1998. doi: 10.1145/277830.277941.

[55] Julio Sahuquillo and Ana Pont. The filter cache: A run-time cache management approach1. In *Procs of the 25th EUROMICRO Conference, Informatics: Theory and Practice for the New Millenium*, EUROMICRO 1999.

[56] Lavanya Subramanian, Vivek Seshadri, Arnab Ghosh, Samira Khan, and Onur Mutlu. The application slowdown model: Quantifying and controlling the impact of inter-application interference at shared caches and main memory. In *Procs. of the 48th International Symposium on Microarchitecture*, MICRO 2015. doi: 10.1145/2830772.2830803.

[57] S. Srikantaiah, M. Kandemir, and Q. Wang. Sharp control: Controlled shared cache management in chip multiprocessors. In *42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 2009. doi: 10.1145/1669112.1669177.

[58] Karthik T. Sundararajan, Vasileios Porpodas, Timothy Jones, Nigel Topham, and Bjorn Franke. Cooperative partitioning: Energy-efficient cache partitioning for high-performance cmps. HPCA 2012. doi: 10.1109/HPCA.2012.6169036.

[59] R Manikantan, Kaushik Rajan, and R Govindarajan. Probabilistic shared cache management (prism). *SIGARCH Comput. Archit. News*, 2012. doi: 10.1145/2366231.2337208.

[60] S. Khan, A. R. Alameldeen, C. Wilkerson, O. Mutlu, and D. A. Jimenez. Improving cache performance using read-write partitioning. In *IEEE 20th International Symposium on High Performance Computer Architecture*, HPCA 2014. doi: 10.1109/HPCA.2014.6835954.

[61] R. Wang and L. Chen. Futility scaling: High-associativity cache partitioning. In *47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 2014. doi: 10.1109/MICRO.2014.46.

[62] André Seznec. A case for two-way skewed-associative caches. In *Procs. of the 20th Annual International Symposium on Computer Architecture*, ISCA â93. doi: 10.1145/165123.165152.

[63] D. Sanchez and C. Kozyrakis. The zcache: Decoupling ways and associativity. In *43rd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 2010, . doi: 10.1109/MICRO.2010.20.

[64] D. Sanchez and C. Kozyrakis. Vantage: Scalable and efficient fine-grain cache partitioning. In *38th Annual International Symposium on Computer Architecture*, ISCA 2011, .

[65] Harshad Kasture and Daniel Sanchez. Ubik: Efficient cache sharing with strict qos for latency-critical workloads. In *Procs. of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2014. doi: 10.1145/2541940.2541944.

[66] Timothy Sherwood, Brad Calder, and Joel Emer. Reducing cache misses using hardware and software page placement. In *Procs. of the 13th International Conference on Supercomputing*, ICS â99. doi: 10.1145/305138.305189.

[67] David Tam, Reza Azimi, Livio Soares, and Michael Stumm. Managing shared l2 caches on multicore systems in software. 2007.

[68] Ying Ye, Richard West, Zhuoqun Cheng, and Ye Li. Coloris: A dynamic cache partitioning system using page coloring. *2014 23rd International Conference on Parallel Architecture and Compilation (PACT)*, pages 381–392, 2014.

[69] Xiao Zhang, Sandhya Dwarkadas, and Kai Shen. Towards practical page coloring-based multicore cache management. In *Procs. of the 4th ACM European Conference on Computer Systems*, EuroSys â09. doi: 10.1145/1519065.1519076.

[70] L. Liu, Y. Li, C. Ding, H. Yang, and C. Wu. Rethinking memory management in modern operating system: Horizontal, vertical or random? *IEEE Transactions on Computers*, 2016. doi: 10.1109/TC.2015.2462813.

[71] Vicent Selfa, Julio Sahuquillo, Lieven Eeckhout, Salvador Petit, and Maráa Engracia Gómez. Application clustering policies to address system fairness with intel's cache allocation technology. In *26th International Conference on Parallel Architectures and Compilation Techniques*, PACT 2017. doi: 10.1109/PACT.2017.19.

[72] Lucia Pons, Vicent Selfa, Julio Sahuquillo, Salvador Petit, and Julio Pons. Improving system turnaround time with intel cat by identifying llc critical applications.

[73] David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. Heracles: Improving resource efficiency at scale. In *Procs. of the 42nd Annual International Symposium on Computer Architecture*, ISCA 2015. doi: 10.1145/2749469.2749475.

[74] Haishan Zhu and Mattan Erez. Dirigent: Enforcing qos for latency-critical tasks on shared multicore systems. In *Procs. of the 21st International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2016. doi: 10.1145/2872362.2872394.

[75] Ginseng: a market-driven llc allocation. In Ajay Gulati and Hakim Weatherspoon, editors, *USENIX Annual Technical Conference*, 2016.

[76] Jichuan Chang and Gurindar S. Sohi. Cooperative caching for chip multiprocessors. In *Procs. 33rd Annual Int. Symp. on Computer Arch.*, ISCA 2006. doi: 10.1109/ISCA.2006.17.

[77] Enric Herrero, José González, and Ramon Canal. Distributed cooperative caching. In *Procs. of the 17th International Conference on Parallel Architectures and Compilation Techniques*, PACT 2008, . doi: 10.1145/1454115.1454136.

[78] Enric Herrero, José González, and Ramon Canal. Elastic cooperative caching: An autonomous dynamically adaptive memory hierarchy for chip multiprocessors. In *Procs. of the 37th Annual International Symposium on Computer Architecture*, ISCA 2010, . doi: 10.1145/1815961.1816018.

[79] M. Awasthi, K. Sudan, R. Balasubramonian, and J. Carter. Dynamic hardware-assisted software-controlled page placement to manage capacity allocation and sharing within large caches. In *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, 2009. doi: 10.1109/HPCA.2009.4798260.

[80] Bradford M. Beckmann, Michael R. Marty, and David A. Wood. Asr: Adaptive selective replication for cmp caches. In *Procs. of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 2006. doi: 10.1109/MICRO.2006.10.

[81] S. Cho and L. Jin. Managing distributed, shared l2 caches through os-level page allocation. In *Procs. of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 2006. doi: 10.1109/MICRO.2006.31.

[82] Zeshan Chishti, Michael D. Powell, and T. N. Vijaykumar. Optimizing replication, communication, and capacity allocation in cmps. *SIGARCH Comput. Archit. News*, 2005. doi: 10.1145/1080695.1070001.

[83] Jaehyuk Huh, Changkyu Kim, Hazim Shafi, Lixin Zhang, Doug Burger, and Stephen W. Keckler. A nuca substrate for flexible cmp cache sharing. In *Procs. of the 19th Annual International Conference on Supercomputing*, ICS 2005, 2005. doi: 10.1145/1088149.1088154.

[84] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki. Reactive nuca: Near-optimal block placement and replication in distributed caches. *SIGARCH Comput. Archit. News*, 2009. doi: 10.1145/1555815.1555779.

[85] H. Dybdahl and P. Stenstrom. An adaptive shared/private nuca cache partitioning scheme for chip multiprocessors. In *Procs. of the IEEE 13th International Symposium on High Performance Computer Architecture*, HPCA 2007. doi: 10.1109/HPCA.2007.346180.

[86] J. Merino, V. Puente, and J. A. Gregorio. Esp-nuca: A low-cost adaptive non-uniform cache architecture. In *Procs. of the Sixteenth International Symposium on High-Performance Computer Architecture*, HPCA 2010. doi: 10.1109/HPCA.2010.5416641.

[87] Nathan Beckmann and Daniel Sanchez. Jigsaw: Scalable software-defined caches. In *Procs. of the 22Nd International Conference on Parallel Architectures and Compilation Techniques*, PACT 2013.

[88] Po-An Tsai, Nathan Beckmann, and Daniel Sanchez. Jenga: Software-defined cache hierarchies. *SIGARCH Comput. Archit. News*, 2017. doi: 10.1145/3140659.3080214.

[89] Sparsh Mittal. A survey of architectural techniques for improving cache power efficiency. *Sustainable Computing: Informatics and Systems*, 2014. doi: 10.1016/j.suscom.2013.11.001.

[90] R. Ubal, J. Sahuquillo, S. Petit, H. Hassan, and P. Lopez. Leakage current reduction in data caches on embedded systems. In *Procs. of the 2007 International Conference on Intelligent Pervasive Computing*, IPC 2007. doi: 10.1109/IPC.2007.95.

[91] K. Flautner, Nam Sung Kim, S. Martin, D. Blaauw, and T. Mudge. Drowsy caches: simple techniques for reducing leakage power. In *Procs. 29th Annual International Symposium on Computer Architecture*, ISCA 2002. doi: 10.1109/ISCA.2002.1003572.

[92] Nam Sung Kim, Krisztián Flautner, David Blaauw, and Trevor Mudge. Single-vdd and single-vt super-drowsy techniques for low-leakage high-performance instruction caches. In *Procs. of the 2004 International Symposium on Low Power Electronics and Design*, ISLPEDâ04, . doi: 10.1145/1013235.1013254.

[93] A. Agarwal, H. Li, and K. Roy. Drg-cache: A data retention gated-ground cache for low power. In *Procs. of Design Automation Conference*, DAC 2002.

[94] H. Hanson, M.S. Hrishikesh, V. Agarwal, S.W. Keckler, and D. Burger. Static energy reduction techniques for microprocessor caches. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2003. doi: 10.1109/TVLSI.2003.812370.

[95] Salvador Petit, Julio Sahuquillo, Jose M. Such, and David R. Kaeli. Exploiting temporal locality in drowsy cache policies. In *Procs. of the 2nd Conference on Computing Frontiers*, CF 2005.

[96] W. Zhang, M. Karakoy, M. Kandemir, and G. Chen. A compiler approach for reducing data cache energy. 2003.

[97] Nasir Mohyuddin, Rashed Bhatti, and Michel Dubois. Controlling leakage power with the replacement policy in slumberous caches. In *Procs. of the 2nd Conference on Computing Frontiers*, CF â05. doi: 10.1145/1062261.1062290.

[98] J.S. Hu, A. Nadgir, N. Vijaykrishnan, M.J. Irwin, and M. Kandemir. Exploiting program hotspots and code sequentiality for instruction cache leakage management. 2003.

[99] S. W. Chung and K. Skadron. On-demand solution to minimize i-cache leakage energy with maintaining performance. *IEEE Transactions on Computers*, 2008. doi: 10.1109/TC.2007.70770.

[100] Yulai Zhao, Xianfeng Li, Dong Tong, and Xu Cheng. Reuse distance based cache leakage control.

[101] Michael Powell, Se-Hyun Yang, Babak Falsafi, Kaushik Roy, and T. N. Vijaykumar. Gated-vdd: A circuit technique to reduce leakage in deep-submicron cache memories. In *Procs. of the 2000 International Symposium on Low Power Electronics and Design*, ISLPED 2000. doi: 10.1145/344166.344526.

[102] Stefanos Kaxiras, Zhigang Hu, and Margaret Martonosi. Cache decay: Exploiting generational behavior to reduce cache leakage power. In *Procs. of the 28th Annual International Symposium on Computer Architecture*, ISCA 2001. doi: 10.1109/ISCA.2001.937453.

[103] Jaume Abella, Antonio González, Xavier Vera, and Michael F. P. OâBoyle. Iatac: A smart predictor to turn-off l2 cache lines. *ACM Trans. Archit. Code Optim.*, 2005. doi: 10.1145/1061267.1061271.

[104] S. Yang, M. D. Powell, B. Falsafi, K. Roy, and T. N. Vijaykumar. An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance i-caches. In *Procs. of the 7th International Symposium on High-Performance Computer Architecture*, HPCA 2001. doi: 10.1109/HPCA.2001.903259.

[105] Kiyofumi Tanaka and Takahiro Kawahara. Leakage energy reduction in cache memory by data compression. *SIGARCH Comput. Archit. News*, 2007. doi: 10.1145/1360464.1360472.

[106] D. Benitez, J. C. Moure, D. Rexachs, and E. Luque. A reconfigurable cache memory with heterogeneous banks. In *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, DATE 2010. doi: 10.1109/DATE.2010.5456936.

[107] Hyunhee Kim and Jihong Kim. A leakage-aware l2 cache management technique for producerâconsumer sharing in low-power chip multiprocessors. *Journal of Parallel and Distributed Computing*, 2011. doi: 10.1016/j.jpdc.2011.08.006.

[108] K. T. Sundararajan, T. M. Jones, and N. Topham. Smart cache: A self adaptive cache architecture for energy efficiency. In *Procs. of the International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, SAMOS 2011. doi: 10.1109/SAMOS.2011.6045443.

[109] S. Kaxiras, P. Xekalakis, and G. Keramidas. A simple mechanism to adapt leakage-control policies to temperature. 2005.

[110] Lei He, Weiping Liao, and M. R. Stan. System level leakage reduction considering the interdependence of temperature and leakage. In *Procs. of the 41st Design Automation Conference*, DAC 2004.

[111] J.C. Ku, S. Ozdemir, G. Memik, and Y. Ismail. Thermal management of on-chip caches through power density minimization. In *Procs. of the 38th Annual International Symposium on Microarchitecture*, MICRO 2005. doi: 10.1109/MICRO.2005.36.

[112] M. Chang, P. Rosenfeld, S. Lu, and B. Jacob. Technology comparison for large last-level caches (l3cs): Low-leakage sram, low write-energy stt-ram, and refresh-optimized edram. In *Procs. of the 19th International Symposium on High Performance Computer Architecture*, HPCA 2013. doi: 10.1109/HPCA.2013.6522314.

[113] Stuart Parkin, Masamitsu Hayashi, and Luc Thomas. Magnetic domain-wall racetrack memory. *Science (New York, N.Y.)*, 2008. doi: 10.1126/science.1145799.

[114] Rangharajan Venkatesan, Vivek Kozhikkottu, Charles Augustine, Arijit Raychowdhury, Kaushik Roy, and Anand Raghunathan. Tapecache: A high density, energy efficient cache based on domain wall memory. In *Procs. of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED 2012. doi: 10.1145/2333660.2333707.

[115] R. Venkatesan, V. J. Kozhikkottu, M. Sharad, C. Augustine, A. Raychowdhury, K. Roy, and A. Raghunathan. Cache design with domain wall memory. *IEEE Transactions on Computers*, 2016. doi: 10.1109/TC.2015.2506581.

[116] G. Wang, Y. Zhang, B. Zhang, B. Wu, J. Nan, X. Zhang, Z. Zhang, J. Klein, D. Ravelosona, Z. Wang, Y. Zhang, and W. Zhao. Ultra-dense ring-shaped racetrack memory cache design. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2019. doi: 10.1109/TCSI.2018.2866932.

[117] G. Sun, X. Dong, Y. Xie, J. Li, and Y. Chen. A novel architecture of the 3d stacked mram l2 cache for cmps. In *Procs. of the 15th International Symposium on High Performance Computer Architecture*, HPCA 2009. doi: 10.1109/HPCA.2009.4798259.

[118] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan. Relaxing non-volatility for fast and energy-efficient stt-ram caches. In *Procs. of the 17th International Symposium on High Performance Computer Architecture*, HPCA 2011. doi: 10.1109/HPCA.2011.5749716.

[119] R. Ubal, J. Sahuquillo, S. Petit, and P. Lopez. Multi2sim: A simulation framework to evaluate multicore-multithreaded processors. In *Int. Symp. on Computer Architecture and High Performance Computing.*, SBAC-PAD 2007, . doi: 10.1109/SBAC-PAD.2007.17.

[120] Rafael Ubal, Byunghyun Jang, Perhaad Mistry, Dana Schaa, and David Kaeli. Multi2sim: A simulation framework for cpu-gpu computing. PACT 2012, . doi: 10.1145/2370816.2370865.

[121] P. Sweazey and A. J. Smith. A class of compatible cache consistency protocols and their support by the ieee futurebus. *SIGARCH Comput. Archit. News*, 1986. doi: 10.1145/17356.17404.

[122] Parviz Kermani and Leonard Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 1979. doi: 10.1016/0376-5075(79)90032-1.

[123] William Dally and Charles Seitz. The torus routing chip. *Distributed Computing*, 1986. doi: 10.1007/BF01660031.

[124] B. Jacob, D. Wang, and Ng. Spencer. *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann, 2007. ISBN 9780123797513.

[125] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman P. Jouppi. Cacti 6.0: A tool to model large caches. In *HP Laboratories*, 2009.

[126] Rafael Rico and Virginia Escuder. Spec cpuint2006 characterization. Technical report, Universidad de Alcalá, 2009.

[127] C. Sakalis, C. Leonardsson, S. Kaxiras, and A. Ros. Splash-3: A properly synchronized benchmark suite for contemporary research. In *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 101–111, April 2016. doi: 10.1109/ISPASS.2016.7482078.

[128] *Standard Performance Evaluation Corporation Website*, 2006. URL `http://www.spec.org/`.

[129] Ankur Limaye and Tosiron Adegbija. A workload characterization of the spec cpu2017 benchmark suite. ISPASS'18. doi: 10.1109/ISPASS.2018.00028.

[130] Agustín Navarro Torres, Jesús Alastruey Beneda, Iba nez Marín, and Vinals Yúfera. Memory hierarchy characterization of spec cpu2006 and spec cpu2017 on the intel xeon skylake-sp. *PLOS ONE*, 2019. doi: 10.1371/journal.pone.0220135.

[131] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The splash-2 programs: characterization and methodological considerations. In *Procs. of the 22nd Annual International Symposium on Computer Architecture*, ISCA 1995, 1995. doi: 10.1109/ISCA.1995.524546.

[132] M. Dana Vantrease. *Optical Tokens in Manycore Processors*. PhD thesis, University of Wisconsin-Madison, 2010.

[133] S. Abadal, A. Cabellos-Aparicio, Jose A. Lazaro, E. Alarcon, and J. Sole-Pareta. Graphene-enabled hybrid architectures for multiprocessors: Bridging nanophotonics and nanoscale wireless communication. In *Transparent Optical Networks (ICTON), 2012 14th International Conference on*, 2012. doi: 10.1109/ICTON.2012.6254490.

[134] Bergman K. Carloni, L. P. Bibermani, A. Chan, and Hendry G. *Photonic Network-on-Chip Design*. 2014. ISBN 978-1-4419-9335-9.

[135] Qianfan Xu, David Fattal, and Raymond G. Beausoleil. Silicon microring resonators with 1.5-μm radius. *Opt. Express*, 2008. doi: 10.1364/OE.16.004309.

[136] Local Area Networks: Token Ring Access Method and Physical Layer Specifications, Std 802.5. Technical report, ANSI/IEEE, 1989.

[137] Jean-Loup Baer, Douglas Low, Patrick Crowley, and Neal Sidhwaney. Memory hierarchy design for a multiprocessor look-up engine. In *12th International Conference on Parallel Architectures and Compilation Techniques (PACT 2003)*, 2003.

[138] A. Shacham, K. Bergman, and L.P. Carloni. On the design of a photonic network-on-chip. In *First International Symposium on Networks-on-Chips*, NOCS 2007.

[139] Guoqing Chen, Hui Chen, Mikhail Haurylau, Nicholas Nelson, Philippe M. Fauchet, Eby G. Friedman, and David Albonesi. Predictions of cmos compatible on-chip optical interconnect. In *Procs. of Int. Workshop on System Level Interconnect Prediction*, SLIP 2005.

[140] Jun Pang, Christopher Dwyer, and Alvin R. Lebeck. Exploiting emerging technologies for nanoscale photonic networks-on-chip. In *Procs. of 6th Int. Workshop on NoC Architectures*, NoCArc 2013.

[141] R. Morris, A. K. Kodi, and A. Louri. Dynamic reconfiguration of 3d photonic networks-on-chip for maximizing performance and improving fault tolerance. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 282–293. doi: 10.1109/MICRO.2012.34.

[142] Yigit Demir and Nikos Hardavellas. Parka: Thermally insulated nanophotonic interconnects. NOCS 2015, . doi: 10.1145/2786572.2786597.

[143] Andrew B. Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *DATE*, pages 423–428. European Design and Automation Association, 2009. ISBN 978-3-9810801-5-5.

[144] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. Dramsim2: A cycle accurate memory system simulator. *IEEE Comput. Archit. Lett.*, pages 16–19. ISSN 1556-6056. doi: 10.1109/L-CA.2011.4.

[145] B. Sinharoy, R. N. Kalla, J. M. Tendler, R. J. Eickemeyer, and J. B. Joyner. Power5 system microarchitecture. *IBM Journal of Research and Development*, 2005. doi: 10.1147/rd.494.0505.

[146] Manish Shah, J. Barreh, J. Brooks, R. Golla, G. Grohoski, N. Gura, R. Hetherington, P. Jordan, M. Luttrell, C. Olson, Bikram Saha, D. Sheahan, L. Spracklen, and A. Wynn. Ultrasparc t2: A highly-treaded, power-efficient, sparc soc. In *2007 IEEE Asian Solid-State Circuits Conference*, 2007. doi: 10.1109/ASSCC.2007.4425786.

[147] Man-Lap Li, R. Sasanka, S. V. Adve, Yen-Kuang Chen, and E. Debes. The alpbench benchmark suite for complex multimedia applications. In *Procs. of the IEEE International Workload Characterization Symposium, 2005*, IIWC'05, 2005.

[148] J. J. Valls, M. E. Gomez, A. Ros, and J. Sahuquillo. A directory cache with dynamic private-shared partitioning. In *Procs. of the IEEE 23rd International*

Conference on High Performance Computing, HiPC 2016. doi: 10.1109/HiPC.2016.051.

[149] Alejandro Valero, Salvador Petit, Julio Sahuquillo, David R. Kaeli, and José Duato. A reuse-based refresh policy for energy-aware edram caches. *Microprocessors and Microsystems - Embedded Hardware Design*, 2015. doi: 10.1016/j.micpro.2014.12.001.

[150] Alejandro Valero, Julio Sahuquillo, Salvador Petit, Pedro López, and José Duato. Combining recency of information with selective random and a victim cache in last-level caches. *ACM Trans. Archit. Code Optim.*, 2012. doi: 10.1145/2355585.2355589.