PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DE COMPUTAÇÃO

Design and Exploration of 3D MPSoCs with on-Chip Cache Support

RODRIGO CADORE CATALDO

Submitted in partial fulfilment of the requirements for the degree of Master of Science in Computer Science at Pontifícia Universidade Católica do Rio Grande do Sul.

Advisor: César Augusto Missio Marcon

Co-Advisor: Débora da Silva Motta Matos

Porto Alegre, Brazil

December 2015

Dados Internacionais de Catalogação na Publicação (CIP)

C357d Cataldo, Rodrigo Cadore

Design and exploration of 3D MPSoCs with on-chip cache support / Rodrigo Cadore Cataldo. – 2015.
129 f.

Diss. (Mestrado) – Faculdade de Informática, PUCRS.
Orientador: Prof. Dr. César Augusto Missio Marcon.
Co-orientador: Prof^{a}. Dr^{a}. Débora da Silva Motta Matos.

1. Multiprocessadores. 2. Energia Elétrica – Consumo. 3. Arquitetura de Computador. 4. Informática. I. Marcon, César Augusto Missio. II. Matos, Débora da Silva Motta. III. Título.

CDD 23 ed. 004.35

Ramon Ely CRB 10/2165
Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada *"Design and Exploration of 3D MPSoC with On-Chip Cache Support"* apresentada por Rodrigo Cadore Cataldo como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, aprovada em 4 de março de 2016 pela Comissão Examinadora:

[illegible]

Prof. Dr. César Augusto Missio Marcon– Orientador — PPGCC/PUCRS

[illegible]

Profa. Dra. Débora da Silva Motta Matos – Coorientadora — UERGS

[illegible]

Prof. Dr. Fernando Gehm Moraes – — PPGCC/PUCRS

Martha Johanna Sepúlveda F.

Profa. Dra. Martha Johanna Sepúlveda Florez - — EI SEC TUM - Munich/Alemanha

Homologada em 30/06/2016, conforme Ata No. 013 pela Comissão Coordenadora.

[illegible]

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

PROJETO E EXPLORAÇÃO DE MPSOCS 3D COM SUPORTE A CACHES INTRACHIP

RESUMO

Avanços na tecnologia de fabricação de semicondutores permitiram implementar um sistema computacional completo em um único chip, em inglês de *System-on-Chip* (SoC). SoCs integram múltiplos elementos de processamento (PEs), componentes de memória e dispositivos de entrada/saída. Este trabalho emprega o termo inglês *Multiprocessor System-on-Chip* (MPSoCs) para um SoC que integra múltiplos PEs cooperantes.

À medida que o número de PEs aumenta em um MPSoC, torna-se necessário o uso de arquiteturas que proveem escalabilidade e concorrência da comunicação. A rede intrachip, em inglês *Network-on-Chip* (NoC), que interconecta o sistema através de roteadores distribuídos no chip foi proposta para atender estes requisitos.

O sistema de interconexão também deve prover recursos para atender a comunicação entre PEs e módulos de memória. Infelizmente, trabalhos prévios demonstraram que basear toda a comunicação de memória com uma NoC não é adequado para atender os requisitos de latência. Além disso, muitas propostas baseadas em NoC descartam o suporte à programação do tipo memória compartilhada que permanece um requisito básico de aplicações paralelas.

A principal contribuição deste trabalho é o projeto e exploração experimental de MPSoCs 3D com suporte a caches intrachip que empregam uma matriz de chaveamento com suporte à coerência de cache para comunicação entre PEs e a hierarquia de memória, e uma NoC para a intercomunicação de PEs, devido à sua eficiência em transmitir pequenos pacotes e sua escalabilidade.

Resultados experimentais foram realizados com o simulador Gem5 utilizando o conjunto de instruções da ARM e dois benchmarks: PARSEC e NASA NAS. Os resultados foram organizados em três conjuntos de avaliação:

1. Avaliação da memória principal utilizando memórias emergentes baseadas em tecnologias 3D e duas memórias tradicionais para *desktops*: *Double Data Rate* (DDR) e *Low Power* (LP) DDR. Para a pluralidade das aplicações, memórias emergentes resultaram em um impacto igual ou menor que 10% de acréscimo no tempo de execução provendo significativa redução no consumo de energia, quando comparadas às memórias tipo DDR;
2. Avaliação de caches utilizando cinco arquiteturas de cache e explorando seus efeitos no tempo de execução de aplicações e consumo de energia. Foram exploradas três arquiteturas compartilhadas e duas arquiteturas privadas em caches L2. Para a maioria das aplicações, a tradicional arquitetura compartilhada da L2 mostrou o melhor tempo de execução. Entretanto, para o consumo de energia, as arquiteturas L2 privadas obtiveram os melhores resultados;
3. Avaliação da escalabilidade do sistema proposto. Os experimentos utilizaram vários tamanhos de *clusters* e aplicações baseadas em troca de mensagens.

Palavras Chave: MPSoC baseado em NoC, MPSoC 3D, Hierarquia de memória, Coerência de cache, MPSoC com suporte a caches intrachip.

DESIGN AND EXPLORATION OF 3D MPSOCS WITH ON-CHIP CACHE SUPPORT

ABSTRACT

Advances in semiconductor manufacturing technology have allowed implement the whole computing system into a single chip, which is namely System-on-Chip (SoC). SoCs integrate several processing elements (PE), memory components and I/O devices. This work employs the term Multiprocessor Systems-on-Chip (MPSoCs) to SoCs that integrate several cooperating PEs.

The increasing quantity of PEs in an MPSoC demands the use of architectures that provide scalability and concurrent communication. The Network-on-Chip (NoC) that interconnects the system through distributed routers has come to tackle these requirements.

The interconnection system must also provide resources to fulfil the communication between PEs and memory modules. Unfortunately, previous works have shown that a single packet-based NoC is not well-suited to provide scalability and low latency for cache supported systems. Additionally, many NoC-based designs lack support for a shared-memory programming model that is an essential requirement for most of the parallel applications.

The main contribution of this work is the design and experimental exploration of 3D MPSoCs with on-chip cache support that employ a crossbar-based infrastructure for the cache-coherent memory hierarchy, and a packet-based NoC for inter-processor communication, due to its efficiency in travelling small packets and its benefits to ever-increasing scalability requirements.

Experimental results performed on the Gem5 simulator using the ARM’s ISA and PARSEC and NASA NAS benchmarks were conducted under three evaluations scenarios:

1. Main memory evaluation using emerging 3D memory technologies and two traditional desktop memories: Double Data Rate (DDR) and mobile Low Power (LP) DDR. For the plurality of the applications, the emerging 3D memory technologies had less or equal than 10% of runtime execution increase providing significant energy saving when compared with DDR memories;
2. Cache evaluation using five cache architectures and exploring its effects on execution runtime and energy consumption. Three shared L2 cache designs and two private L2 cache design were explored. For the majority of the applications evaluated, the traditional shared L2 design had the lowest execution runtime. However, the private L2 designs showed the lowest energy consumption;
3. Scalability evaluation of the proposed system. Experiments using various sizes of clusters and applications based on message exchange.

Keywords: NoC-based MPSoC, 3D MPSoC, Memory hierarchy, Cache coherence, MPSoC with on-chip cache support.

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

2D	Two-Dimensional
3D	Three-Dimensional
API	Application Programming Interface
CABA	Cycle Accurate Bit Accurate
CCI	Cache Coherent Interconnect
CMOS	Complementary Metal-Oxide-Semiconductor
CMP	Chip MultiProcessor
DDR	Double Data Rate
DMA	Direct Memory Access
DRAM	Dynamic Random Access Memory
DVFS	Dynamic Voltage and Frequency Scaling
eDRAM	Embedded Dynamic Random Access Memory
FS	Full System
GALS	Globally Asynchronous, Locally Synchronous
Geomean	Geometric Mean
GHB	Global History Buffer
GIC	Generic Interrupt Controller
GNU	GNU’s Not Unix
GPL	General Purpose License
HBM	Hybrid Bandwidth Memory
HMC	Hybrid Memory Cube
HPC	High-Performance Computing
I + D	Instruction plus Data
IC	Integrated Circuit
ISCA	International Symposium on Computer Architecture
IDE	Integrated Development Environment
IPC	Instruction Per Cycle
ISA	Instruction Set Architecture
ISS	Instruction Set Simulator
LLC	Lower-Level Cache
LPDDR	Low Power Double Data Rate
LRU	Least Recently Used
MPP	Massively Parallel Processor
MPSoC	Multiprocessor System-on-Chip
MSHR	Miss Status Holding Register
NAS	Numerical Aerodynamic Simulation
NPB	NAS Parallel Benchmarks
NR	Non-replicated
NUCA	Non-Uniform Cache Architecture
NUMA	Non-Uniform Memory Access
NoC	Network-on-Chip
NORMA	NO Remote Memory Access

NR	Non-Replicated
O3	Out-Of-Order
OS	Operating System
OSCI	Open SystemC Initiative
OSI	Open System Interconnection
OVP	Open Virtual Platforms
PARSEC	Princeton Application Repository for Shared-Memory Computers
PE	Processing Element
PIPT	Physically Index Physically Tagged
ROI	Region of Interest
RTL	Register Transfer Level
SA	Simulated Annealing
SCU	Snoop Control Unit
SE	System-call Emulation
SMP	Symmetric MultiProcessing
SoC	System-on-Chip
SRAM	Static Random Access Memory
SSE	Streaming SIMD Extensions
TDP	Thermal Design Point
TLB	Translation Lookaside Buffer
TLM	Transactional Level Modeling
TLM-DT	TLM with Discrete Time
TSV	Through Silicon Via
UART	Universal Asynchronous Receiver/Transmitter
UMA	Uniform Memory Access
VFP	Vector Floating-Point
VIPT	Virtually Index Physically Tagged
VNC	Virtual Network Computing

CONTENTS

1 INTRODUCTION

The evolution of technologies employed to manufacture Integrated Circuits (ICs) and the ever-increasing ratio of transistors per unit area brought the possibility of integrating billions of transistors into a single chip, which allows building the complete system functionality on a single IC called System-on-Chip (SoC). [INT15a][QUA15] are examples of commercial SoCs.

Recent *terascale* applications demand complex functionalities requiring massive computational resources [ZAR15], boosting the development of the Multiprocessor System-on-Chip (MPSoC). This architecture encompasses multiple and possibly heterogeneous Processing Elements (PEs), a memory hierarchy (i.e., cache layers and main memory) and I/O components. Future MPSoCs will be made up of hundreds of such PEs [FER12][ITR11]. However, the growth of the quantity of PEs into a single chip requires a scalable interconnection to maintain acceptable communication latency and throughput.

The traditional solutions based on buses can only handle a few PEs and cannot scale to higher degrees of parallelism [BEN02]. Two-dimensional (2D) Network-on-chip (NoC) has emerged as a communication architecture that can overcome this problem through a scalable, packet-based network, inspired by years of study in computer networks [AHM10][JIA11]. This network employs a protocol stack similar to the traditional Open System Interconnection (OSI) stack to provide an abstraction of the electrical, logic, and functional proprieties of the interconnect scheme. Thus, the designer can optimize each layer independently. The complexity increase of this network relies on the fact that interconnect technology is an important limiting factor for achieving SoC's operational goals [BEN02].

Chip designers proposed three-dimensional (3D) ICs, where components are distributed through stacked layers, instead of linearly in a single layer to improve data communication and throughput [AHM10][FRE12]. Consequently, reducing some complexity involved in the fabrication of ICs, such as global wires length [FEE09][FIC13]. Figure 1 shows an example of reduced wire length due to the stacking of layers. Besides, 3D topologies may reduce the hop count when compared to their 2D counterpart [MAR14]. According to Feero and Pande [FEE09], there are 40% more hops in a 2D mesh compared to that in a 3D mesh. Conversely, multiple stacked layers of IC exacerbates the need for heat dissipation.

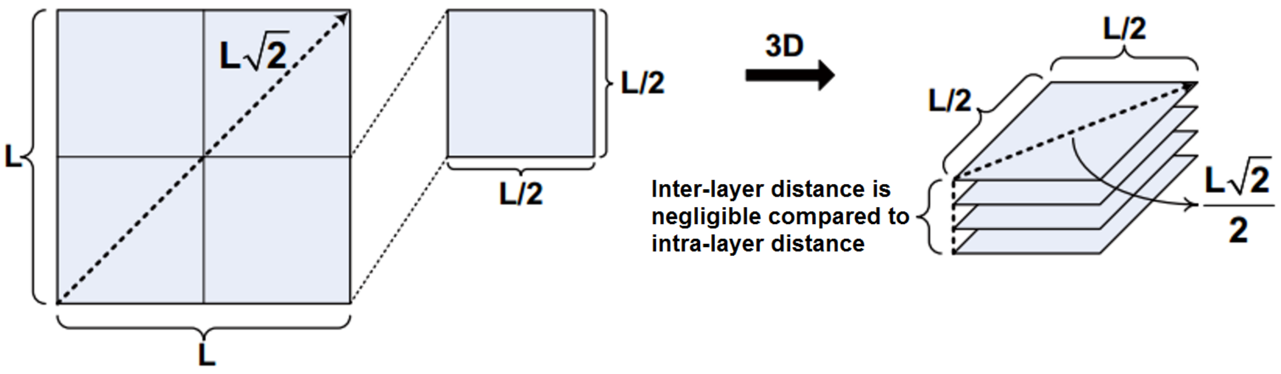

L√2
L
L
L/2
L/2
3D
L/2
L/2
Inter-layer distance is negligible compared to intra-layer distance
L√2/2

Figure 1. Wiring length of simple layer vs a stack of smaller layers [LI06].

1.1 Memory Hierarchy in MPSoCs

One of the most critical components that determine the success of an MPSoC-based architecture is its memory system [KAN05] because many applications spend a significant portion of their cycles in the memory hierarchy. Delays in the memory hierarchy are critical since the processor depends on it to execute instructions. Therefore, the memory hierarchy has a higher impact on the performance of an MPSoC-based system than the inter-core communication.

The technology employed to develop PEs advances faster than that used in the development of the memory elements, which enables PEs to operate at higher frequencies. This disparity allows PEs to consume data at rates not possibly achieved by Dynamic Random Access Memories (DRAMs), creating a performance gap [CHA11]. The microprocessor architectures frequently use a memory hierarchy to tackle this gap. The idea is to provide the illusion of a large and very fast memory, which is done placing faster but smaller memories closer to the PE and slower but larger memories increasingly farther from it. The smaller memories contain a subset of the entire memory, which usually consists of the most access data by the PEs [PAT13].

The smaller memories are called caches. Its presence is based on the principle that a program does not access all of its code or data at once with equal probability [PAT13]. Hence, the set of most referenced information can be stored in a faster memory, maximizing the number of instructions executed by a PE.

Unfortunately, the use of a simple packet-based communication as the foremost network compromises the efficiency of the memory hierarchy. Coskun et al. [COS09] demonstrated that an UltraSPARC T1 Core occupies 10 mm^2 of chip area, and an L2 shared cache consumes 19 mm^2 of chip area, in 90 nm technology. Consequently, shared L2 caches are commonly employed in NoC-based designs since the core area cannot easily absorb L2 cache area. This results that any access to an L2 cache must traverse the NoC twice: one for requesting the content of a given address and another one for the reply.

As shown by previous works [FU14][WA08][YE10], even conservative rates of packets injection (i.e., $\leq 1\%$) result in large 2D NoC latencies due to congestion caused by the intensive memory accesses and inter-core communication. For instance, Ye et al. [YE10] show that two L2 cache blocks distributed into an 8×8 NoC and packets injection rate of only 0.2% results in 1976 cycles of network latency, in average – a prohibitive latency for such small injection rate. These works have also demonstrated that on-chip traffic congestion is primarily caused by the intensive memory access requests and responses. Thus, a better design must be employed to tackle this situation with efficiency [YE10].

1.2 Motivation

3D ICs present a promising technology for cache-enabled architecture schemes in a NoC-based design. This technology enables the efficient manufacture of both logic and memory into a single IC, as they can be produced independently and integrated in a latter process [LOI10]. Furthermore, such fabrication process enables the use of emerging new technologies, such as 3D main memories enabled by Through Silicon Vias (TSVs) connection. The integration of main memory and processor in a single chip package allows new design explorations for energy saving and throughput, as they are not affected by limited I/O pin count.

Secondly, 3D NoCs have emerged to reduce the length and the number of global interconnections that packets must pass through, and consequently, enabling to decrease the network latency and to increase its throughput [FER15]. Nonetheless, NoC architectures hardly fulfill the latency and throughput requirements for memory systems, since memory access rates are normally orders of magnitude higher than message exchange rate, even for I/O bounded systems. Thus, this work proposes a two-layer abstract system that employs disparate communication architectures for the inter-processor and the memory system communication; i.e., a NoC performs the inter-processor communication, whereas a special purpose architecture fulfills the memory communication requirements.

1.3 Contribution

The main contribution of this work is the design and experimental exploration of 3D MPSoCs with on-chip cache support that employs independent infrastructures for inter-processor and memory system communication. We propose the use of packet-based NoC for inter-processor communication due to its efficiency on traveling small packets and its benefits on ever-increasing scalability requirements [BEN02]. For the memory system, we propose the use of a cache coherence hierarchy implemented in a crossbar-based infrastructure.

However, as hardware coherence is costly to maintain and does not scale efficiently [MAT10], our system has a two-layer architecture. The first layer is the interconnection among PEs of a cluster presenting a single coherent address space and a Uniform Memory Access (UMA) model. The second layer interconnects clusters through an NO Remote Memory Access (NORMA) model – i.e., clusters do not share address space. As such, a NoC architecture accomplishes the communication between clusters having the potential to scale to hundreds of PEs.

Additionally, aggregating UMA and NORMA models in the same target architecture enables us to use multiprocessing and multicomputer programming jointly, which enlarges the exploration and implementation spectrum of highly complex applications.

The experimental results were performed on the Gem5 simulator [BIN11] due to its ability to accurately model a coherent memory system, while being several times faster than a hardware-level model. The target architecture is based on the ARM Versatile Express development board [ARM15a].

This work adopts scenarios employing 3D IC to provide fast communication between processors and caches using a cache coherent interconnect. Since TSV are shorter and wider than intra-layer interconnects, they support higher signaling speeds [EBR13]. Besides, TSV-enabled memories are employed to provide even additional bandwidth and energy saving.

The experimental results are organized in four evaluations. First, we analyze the instruction-level parallelism provided by the parallel workloads and the vanilla Linux kernel scheduler. This result is fed to a power estimation tool (McPAT [HP15]) to provide runtime dynamic energy consumption. Then, we analyze the impact on the execution time of the parallel workloads under a diverse set of memory technologies. This set is comprised of conventional desktop memories, mobile low-power memories, and TSV-enabled emerging technologies. We complement this study with five different cache architectures and, again, analyze the impact on execution runtime. Besides, these scenarios provide energy consumption analysis. Finally, the scalability of the system is tested using diverse sized clusters on the UMA model.

The results demonstrate that the impact on execution runtime for the newer TSV-enabled memories are within -10% and 10% for half of the applications employed (4 out of 8), which is an impressive result, considering the focus of today’s memories on energy saving. The remaining two applications suffer to achieve a desirable speed-up and do not perform as well for these types of memories.

The conventional shared L2 design has the best execution runtime of all cache architectures explored in this work. Unfortunately, its design has one of the most demanding power consumption. As such, every cache design is a different optimization point regarding execution runtime and power consumption.

The NoC-based design shows a remarkable performance for appropriate applications and scales well.

1.4 Objectives

This work proposes the design of 3D NoC-based MPSoCs with stacked memory layers, which is satisfied with the following strategic and specific objectives.

Strategic:

1. To explore available simulation tools for memory hierarchy evaluation;
2. To understand and explore cache design for on-chip shared use;
3. To assess the impact of main memory design for MPSoCs systems;
4. To explore 3D MPSoC architectures taking into account a memory-centric design (i.e., the fulfillment of the memory requirements).

Specific:

1. To analyze the instruction-level parallelism of the parallel workloads employed in this work. This evaluation is achieved through a full-system simulator and power estimation tool;
2. To analyze the impact on execution time of a diverse set of memory technologies;
3. To implement and validate 3D MPSoCs with shared/private L2 caches for latency and energy evaluations. These assessments are achieved through a benchmark suite, aiming to substantiate that such design is efficient to fulfill the ever-increasing demand of MPSoCs;
4. To analyze the inter-cluster communication employing a hierarchical packet-based NoC. This interface provides the scalability of the architecture. This evaluation is achieved using a message passing benchmark suited for this type of system.

1.5 Organization

This work is organized as follows. Chapter 2 discusses cache organization commonly found in MPSoC and multicore architectures and details the major memory architecture designs. Chapter 3 presents the related work on 2D and 3D MPSoCs with on-chip cache support. Chapter 4 presents the Gem5 simulator – its features, accuracy and shortcomings – and an overview of alternative modern full system simulators. Chapter 5 discusses the design and exploration of 3D MPSoC in this work. This is complemented with the experimental results presented and discussed in Chapter 6. Finally, Chapter 7 presents conclusions and directions for future works.

2 CACHE HIERARCHY IN MPSOC ARCHITECTURES

The memory hierarchy of the MPSoC architecture follows the same characteristics than the one present in the multicore architecture [ASA09], whose levels represent a tradeoff between latency and cost. The hierarchy can be summarized from the higher level of performance to the lower ones as follows: multiple levels of cache, main memory, and massive storage. Often, the lower levels of this hierarchy are placed outside of chips since they do not have a response time suitable to the processor demand and require large amounts of chip area. Nonetheless, frequently the higher levels are placed on the chip.

The first level of the cache hierarchy (i.e., L1 cache) can be easily integrated into the processor area because it has few kibibytes of memory, which does not happen with lower levels of cache due to their bigger size.

Balasubramonian and Jouppi [BAL11] classify the cache architecture into eight categories derived from concepts that define (i) the relative position of the caches and processors - *centralized* vs. *distributed*; (ii) the exclusive access of information - *private* vs. *shared* - and (iii) the latency of memory accesses - *uniform vs. non-uniform access*. Figure 2 depicts all cache architectures derived from these concepts. These concepts provide tradeoffs regarding wire lengths, throughput, area occupation, and energy consumption and imply there is no optimal cache hierarchy to fit every system.

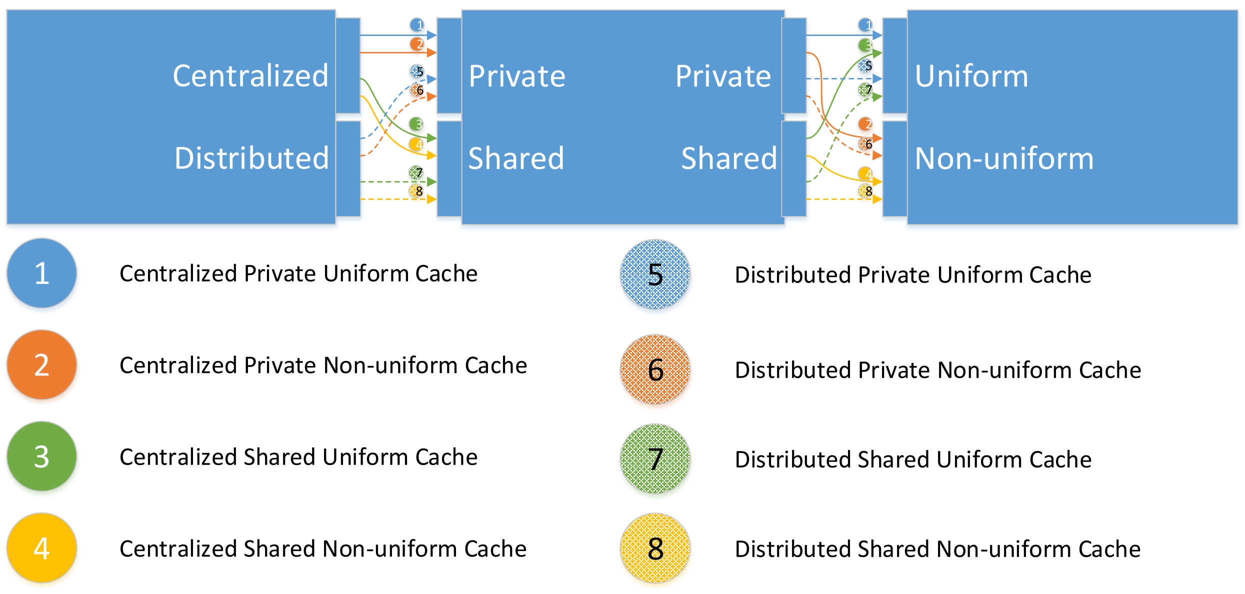

Centralized
Distributed
Private
Shared
Private
Shared
Uniform
Non-uniform
1
Centralized Private Uniform Cache
2
Centralized Private Non-uniform Cache
3
Centralized Shared Uniform Cache
4
Centralized Shared Non-uniform Cache
5
Distributed Private Uniform Cache
6
Distributed Private Non-uniform Cache
7
Distributed Shared Uniform Cache
8
Distributed Shared Non-uniform Cache

Figure 2. Eight categories of cache architectures derived from [BAL11].

Follows the definition of the concepts stipulated by Balasubramonian and Jouppi that will be used throughout this work.

Centralized vs. Distributed: a centralized cache occupies a contiguous area on the chip. Conversely, a distributed cache is physically distributed on a chip area.

Private vs. Shared: a private cache grants exclusive access to a given processor. The shared cache shares access among processors, requiring mechanisms for cache-coherence operation.

Uniform access vs. Non-uniform access: in the uniform memory model, all processors access the caches with the same latency, whereas in a non-uniform memory model, the processors and caches placement implies different latencies.

This classification refers to the physical organization of the cache and, as such, excludes cache coherent protocols that share information between uniform accesses.

The multitude of cache levels is not restricted to a single category. Therefore, to clarify the discussion herein, we will adopt the terminology of Balasubramonian and Jouppi [BAL11]: a cache level closer to the processor is considered an upper-level cache, and, for the opposite scenario, i.e., closer to the main memory, the cache is considered a Lower-Level Cache (LLC).

Four sections divide the remaining chapter. The first discusses memory access models commonly used in processor designs. The second presents commercial chips and correlates them with the cache hierarchy in MPSoCs. Finally, the third and fourth sections discuss recent researchs in cache hierarchy of 2D and 3D chips, respectively.

2.1 Memory Architecture Design

Given the inherent difficulty of writing programs to run efficiently on parallel systems, one feature often found is the ability to address the entire physical memory space as a single entity. Thus, the programmer does not need to concern itself with the data placement, because all variables are accessible at any time to any processor [PAT13]. This type of system is named Symmetric MultiProcessing (SMP). When the physical address is a unique entity, the hardware typically provides cache coherence to give a consistent view of the memory subsystem [MAR12][PAT13].

For single address space, two of the most common architectures are UMA and NUMA (*Non-Uniform Memory Access*) [PAT13]. In the first architecture, all processors access any memory position with the same latency. In the second architecture, processors access the same memory position with different latencies.

The communication rate among processors and memories limits the performance in the UMA architecture. Adding processors to the system beyond some point does not increase performance linearly since they share the same memory bandwidth [GEN12]. Thus, scaling beyond the dozens of processors requires an NUMA architecture [HWA12]. The non-uniformity of access can also be applied to an individual memory unit, whereas the access latency depends on the location requested. Older L2 caches were comprised of a single bank, which were simpler to design and did not provide parallel access to its data content. Nowadays, L2 caches are divided into blocks to allow parallel access and diminish the individual bank latency [ARM11a][OLU07]. Nonetheless, such design can increase the latency penalty due to the interconnect delay as shown in Figure 3.

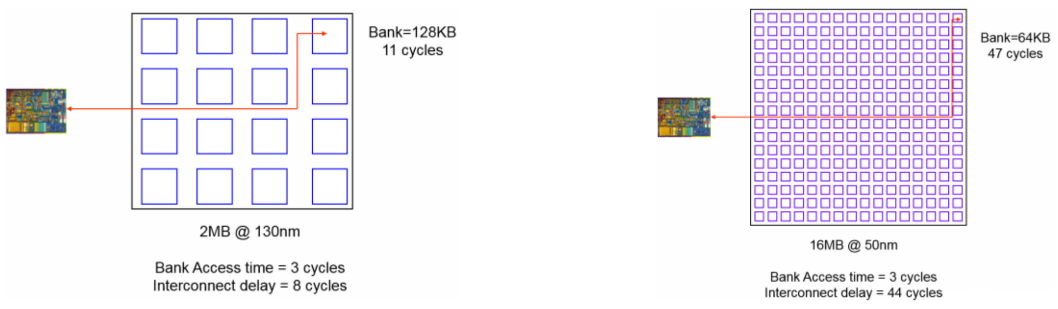

Bank=128KB
11 cycles
2MB @ 130nm
Bank Access time = 3 cycles
Interconnect delay = 8 cycles
Bank=64KB
47 cycles
16MB @ 50nm
Bank Access time = 3 cycles
Interconnect delay = 44 cycles

Figure 3. Effect of increasing data banks on multi-banked L2 caches [LEE15a].

Aiming to minimize this penalty, Kim et al. [KIM02] propose the Non-Uniform Cache Architecture (NUCA), which allows avoiding the high latency access due to interconnect delay if the cache controller accesses each bank at a speed proportional to its distance from the cache controller. This opens new design space for the exploration of cache policies regarding mapping, search algorithm, and data migration [KIM03]. Besides, the designer can exploit the various cache latencies to design memory hierarchy with more than one technology embedded in the same unit. Wu et al. [WU09] study the effect of using disparate memory technologies in both intra- and inter-cache levels. A single cache level partitioned into multiple regions, where each region exploits the advantage of the memory technology employed (e.g., higher density, lower power dissipation) define the intra-cache level. The inter-level cache has the same principle but uses multiple levels of cache to employ more than one technology. Experimental results from their work with a full system simulator showed that the intra-level cache provides 12% of IPC (*Instruction Per Cycle*) improvement over a 3-level cache design implemented with *Static Random Access Memory* (SRAM) under the same area constraint, in average. Moreover, the inter-level cache provides 7% of IPC improvement on the same conditions.

Future MPSoCs, made of hundreds of processing units [FER12][ITR11], hinder the ability of the hardware to provide a coherent view of the entire memory space as proposed by the UMA and NUMA architectures. For large-scale systems, NORMA is attractive due to its ability to decentralize resources and increase reliability [HWA12]. Processor communication is carried out by message passing through the NoC [HWA11], and through network protocols.

2.2 Cache Organization in Modern Commercial and Research Chips

Up to the 2000s most on-chip cache hierarchies were comprised of two levels of cache (L1 and L2). However, in recent years, both the academia and industry employed a large effort to standardize the use of L3 cache on-chip [BAL11].

As mentioned earlier, L1 can be integrated into the core area and, as such is a *private* component. Nevertheless, this clear trend is not present for the extra levels of cache. Thus, to enrich the discussion of cache organization, this section focuses on the LLCs.

2.2.1 Private Cache Architecture

The private LLC organization has a dedicated cache hierarchy for each core. Figure 4 exemplifies a schematic representation of an MPSoC with two levels of cache, where the L1 cache area is divided into instruction cache (I1) and data cache (D1). The communication architecture is used iff a memory request misses in both L1 and L2 cache banks.

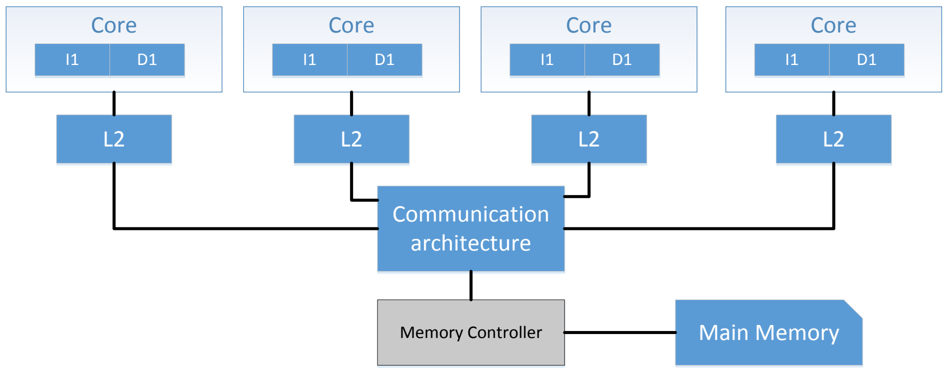


Figure 4. Schematic exemplification of an MPSoC with cache hierarchy composed by private L1 and L2 caches (based on [ASA09]).

The remaining of this section presents and discusses research and commercial chips. Two of the presented chips, Intel's Knights Landing and AppliedMicro X-Gene 3, have not been finalized and may present technical differences from the final product.

A product family from Intel called Xeon Phi is the first chip presented in this category, and the Knights Corner iteration of it is shown in Figure 5. Each core is equipped with a 32KiB L1 instruction and 32KiB L1 data cache, and an individual 512 KiB L2 cache. The entire cache system is kept coherent. This product uses a bidirectional ring for communication. Each direction is comprised of three independent rings for data, address and acknowledgment messages [INT15b]. Local and remote average accesses of the L2 cache are 24 and 250 cycles, respectively [FAN13]. Remote access is achieved through a coherence protocol.

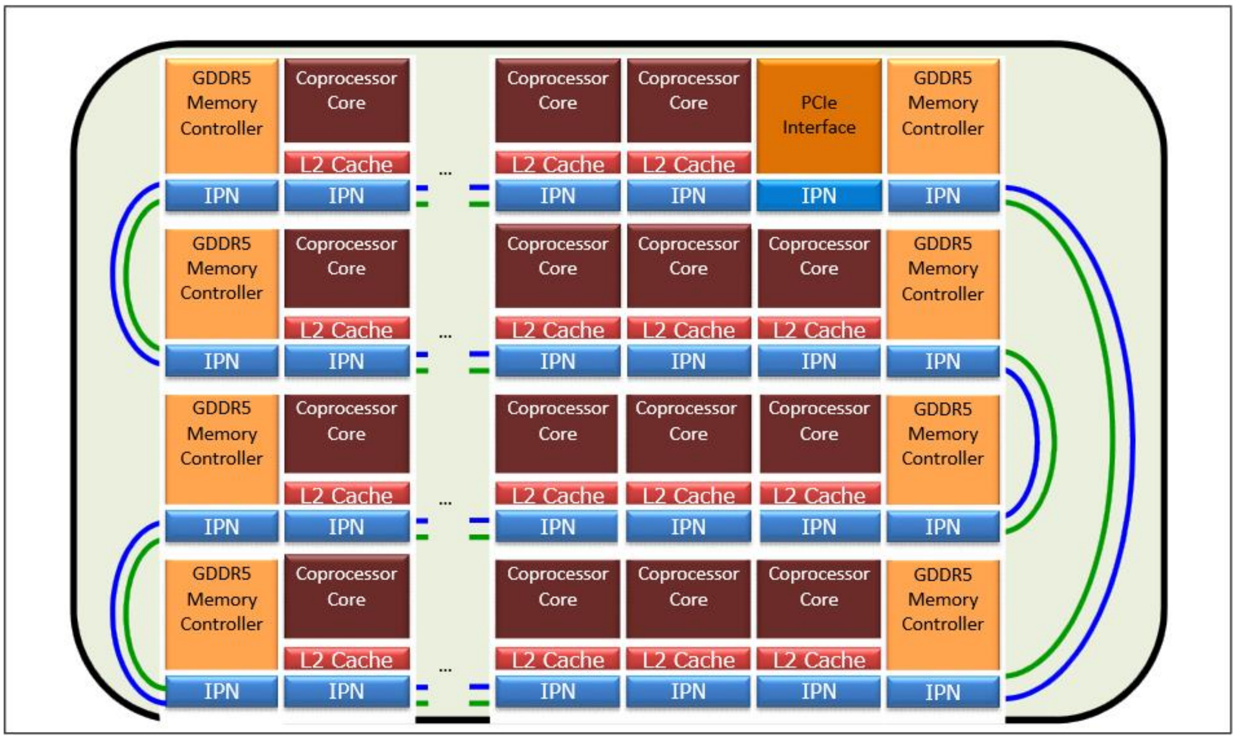


Figure 5. Conceptual diagram of the structure of the Intel Xeon Phi coprocessor [INT14].

The next-generation of the Xeon Phi, called Knights Landing, is an upcoming chip that features stacked memory chips, which are linked by TSVs, to increase greatly the amount of memory bandwidth that feeds the cores [IDG15]. Full details of this platform are still not available, but it is expected to contain at least 60 cores [PLA15][STO15].

Tilera Corporation is a semiconductor company focusing on scalable multicore embedded processor design. Its products range from supporting 4 to 200+ cores [TIL11a]. In the Tilera-Gx architecture, each core has 32KiB L1 instruction and 32KiB L1 data cache, and an individual 256KiB L2 cache. Figure 6 depicts the core unit, memory subsystem, and the communication network. Tilera-Gx uses five independent NoCs, all of them employing mesh topology, whereas three of them are related directly to the memory subsystem. The rationale for independent networks is to allow low latency communication for a scalable architecture [TIL11c]. Local and remote L2 cache latencies are 11 cycles and 41 cycles, respectively. The remote access is achieved through a proprietary and distributed coherence protocol [TIL13].

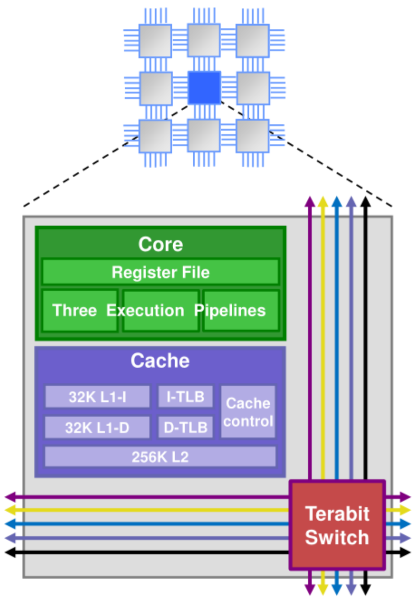


Figure 6. General-purpose core unit of Tile-Gx100 [TIL11b].

2.2.2 SHARED CACHE ARCHITECTURE

The complementary classification to a private architecture is a *shared* one, where, one or more LLC blocks are distributed across the chip to service the whole MPSoC design. Figure 7 depicts a schematic representation of an MPSoC with this organization. As the case of Figure 4, the L1 cache area is divided between instruction cache (I1) and data cache (D1). The NoC is used iff a memory request misses in the L1 cache bank.

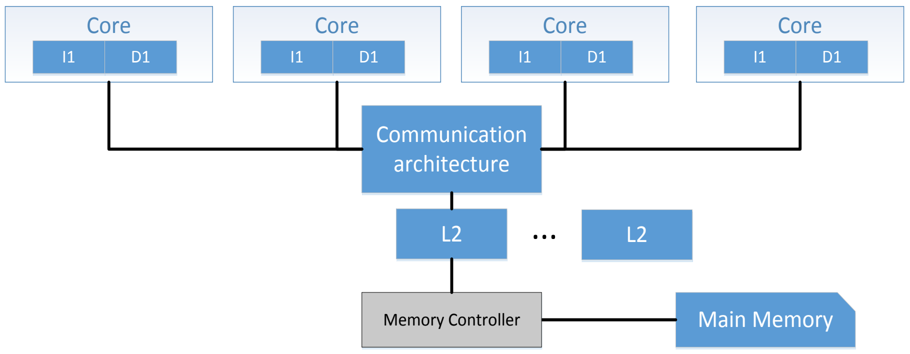


Figure 7. Schematic illustration of an MPSoC with shared L2 caches (based on [ASA09]).

IBM produces the Power8 processor with 12 cores per chip. Every core has access to 64KiB of L1 data and instruction caches, and each core has an individual 512KiB SRAM L2 cache. The cores share a 96MiB eDRAM (*embedded DRAM*) L3 cache and an optional eDRAM L4 cache [IBM14][STU13]. Buses operating at up to 2.4 GHz accomplish the chip interconnection [STA15]. Microarchitecture optimization – pipeline, multiple segments, and distributed arbitration – is applied to handle the propagation delay across the chip. The architecture employs NUCA model from L2 to lower memory levels, which means that not all accesses to L2 and lower levels have the same latency.

Power8 supports up to 192 cores using 16 chips. The off-chip interconnection is a multi-tiered fully connected structure designed to reduce latency, which are logically organized in a mesh-like 4×4 topology. This topology is shown in Figure 8. Each chip connects to its three-abscissa neighbors (called intra-group) and to its three ordinate neighbors (called inter-group) [STA15]. Local and remote L3 access are approximately 27 and 130 cycles, respectively [7ZI15a].

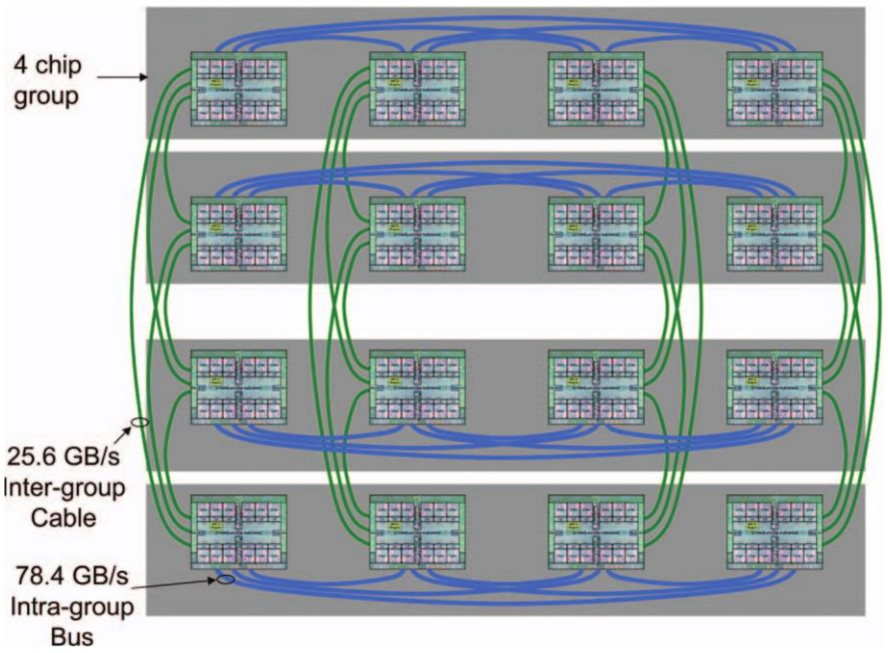


Figure 8. Power8 SMP topology forming up to 4 groups comprised of 4-chip per group [STA15].

Figure 9 shows a product from Fujitsu with a similar architecture called SPARC64-X. This product has four shared L2 caches of 6MiB each (up to 24MiB). Up to four CPU sockets can be connected directly and up to 64 sockets can be connected through a crossbar chip. A single chip encompassing 16 cores implements a high-throughput serial transfer protocol for inter-socket communication. NUMA is employed to maximize memory locality [FUJ15].

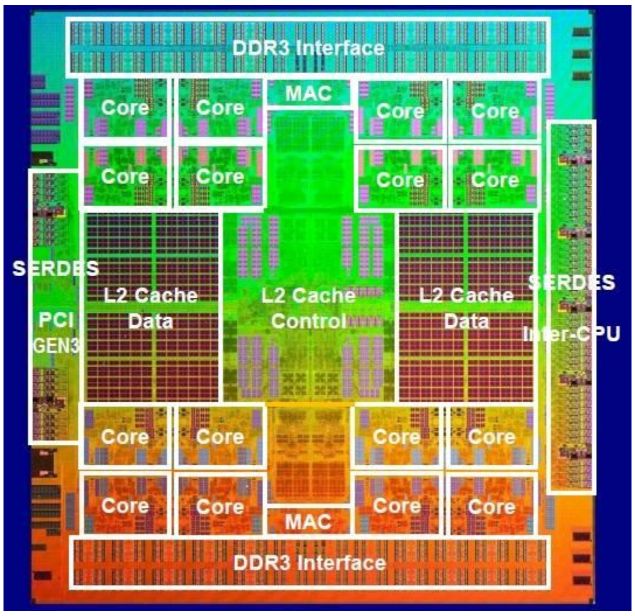
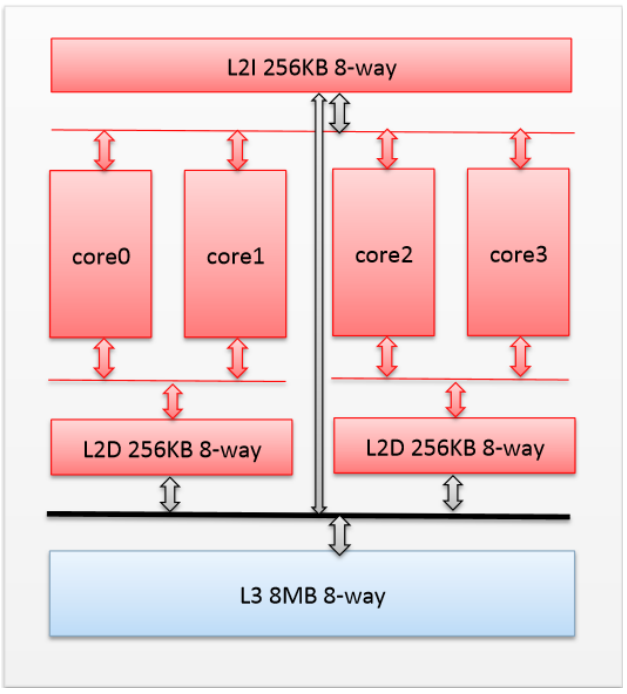
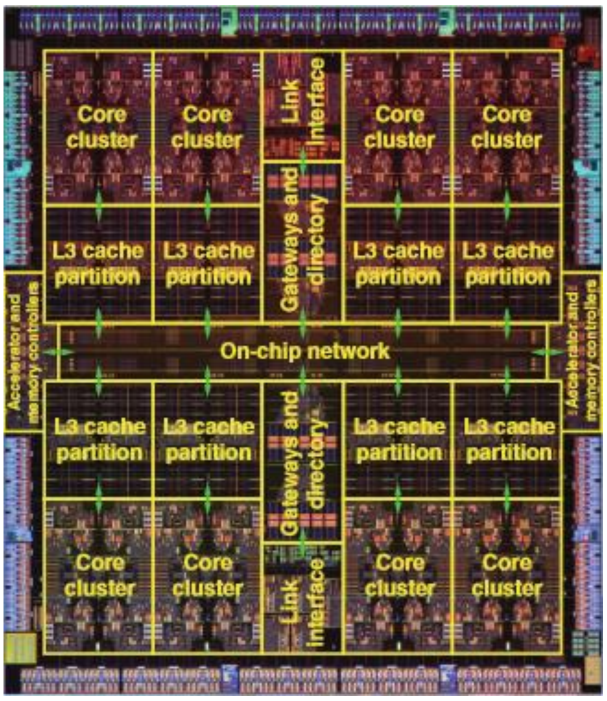


Figure 9. SPARC64-X die containing 16 cores and 4 banks of L2 cache [REG15].

In 2014, Oracle proposed the SPARC M7 chip with 32 cores [SIV14]. The cores are organized in 8-core clusters, where each core accesses 16KiB of instruction and data cache. Pairs of two cores have access to a shared 256KiB L2 cache of data. Finally, each pair of four cores has access to a 256KiB L2 cache of instructions and an 8MiB L3 cache [LI15][SIV14]. Figure 10 shows the cache hierarchy and the chip organization of SPARC M7.





(a) (b)

Figure 10. SPARC M7: (a) schematic of core cluster design, and (b) chip layout [SIV14].

The crossbar-based network used in previous SPARC processors is scrapped, and a hybrid network is employed. This network connects all L3 caches and four memory controllers. Three physical networks compose the logical communication architecture: a requesting network with a 4-ring topology (maximum hop count is 11), a point-to-point response network and a multi-stage mesh of six 10×10 switches [AIN15][LI15].

Figure 11 depicts the organization of two of the three physical NoCs – the omitted request network overlies the data network physically. Some power management units are present to achieve high-performance under acceptable power constraints. Konstadinidis et al. [KON15] discuss in detail the power-related elements of the SPARC architecture.

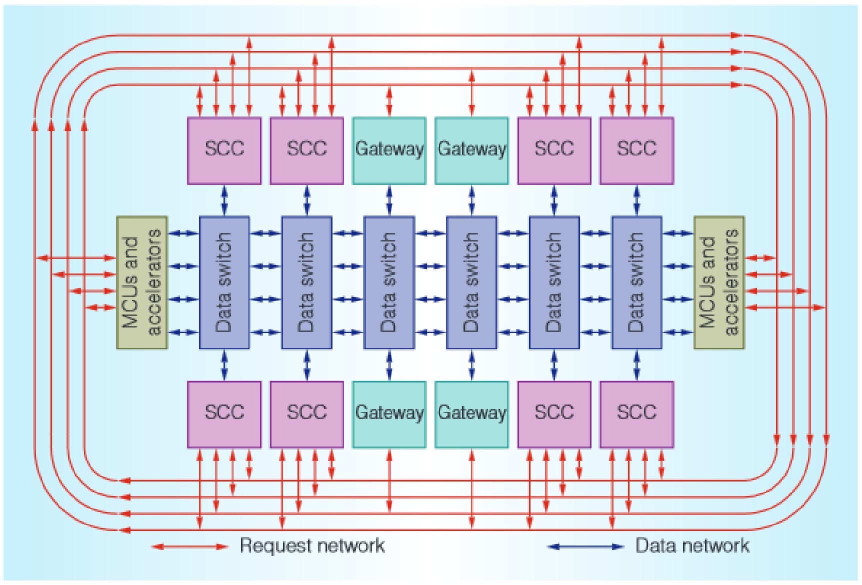


Figure 11. On-chip network request and data topology. The response network is omitted in this representation [AIN15].

P2012 is an area- and power-efficient manycore architecture based on multiple Globally Asynchronous, Locally Synchronous (GALS) STxP70-based processor clusters. Each cluster features up to 16 processors with a multi-banked one cycle access L1 cache and specialized hardware for aggressive power management [MEL12]. CEA leti/list laboratories and STMicroelectronics developed this platform, which was later rebranded to STHORM [TOR14]. Figure 12 depicts a cluster of the platform with two processors and six hardware accelerators. Each tile has a dedicated 1MiB L2 cache. Inter-cluster communication is achieved through an asynchronous NoC, which allows each cluster to operate independently in its own optimized needs.

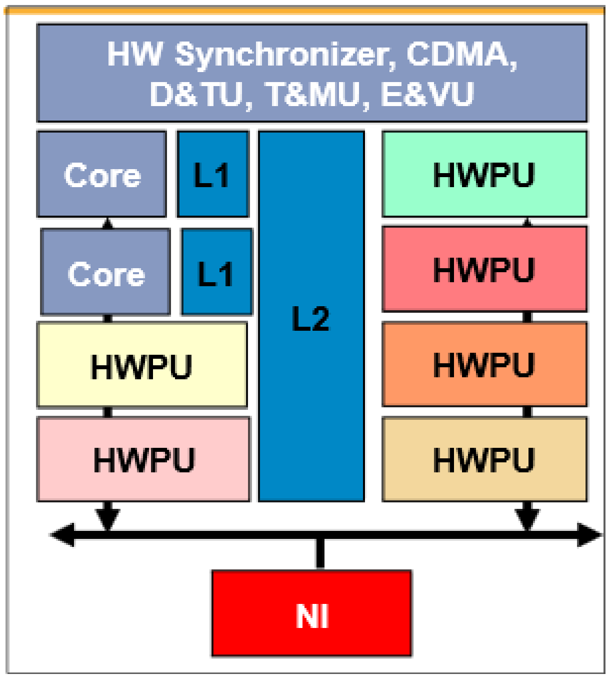


Figure 12. The P2012 cluster [BEN10].

ARM has proposed an architecture called big.LITTLE that pairs a high-performance Out-Of-Order (O3) processor cluster with a low-power in-order processor cluster to deliver optimal performance and energy consumption [ARM13a]. The two processor clusters are architecturally compatible, meaning that they can exchange tasks. Therefore, the high-performance processor cluster can shut down when no application requires such performance, and only the more modest processor cluster can handle the system. Any time that performance requirement is increased, the high-performance processor cluster is initialized and can ‘steal’ tasks from the modest cluster [ARM13b]. Each cluster has its own L2 cache.

Figure 13 shows the Exynos 5 Octa SoC – one of the products that uses the big.LITTLE architecture. For this SoC, the L2 cache has 2MiB and 512KiB of space for, respectively, the high-performance cluster and the low-power processor cluster [SAM15a]. Communication is accomplished through a multilayer bus. As an example of cache latency, Exynos 5250 has L1 and L2 latencies of 4 and 21 cycles, respectively [7ZI15b].

X-Gene is a SoC solution developed by Applied Micro Circuits Corporation. Its first architecture iteration (X-Gene 1) is comprised of eight ARMv8 cores with private L1 caches, four L2 caches shared by each pair of cores, and L3 caches [SIN14]. Interconnection is done through a low-latency NoC (Arteris FlexNoC IP solution) [ART15]. The second iteration of this product has some changes to its core structure but maintains the same cache hierarchy. L2 and L3 cache latencies are 13 and 89 cycles, respectively [7ZI15c].

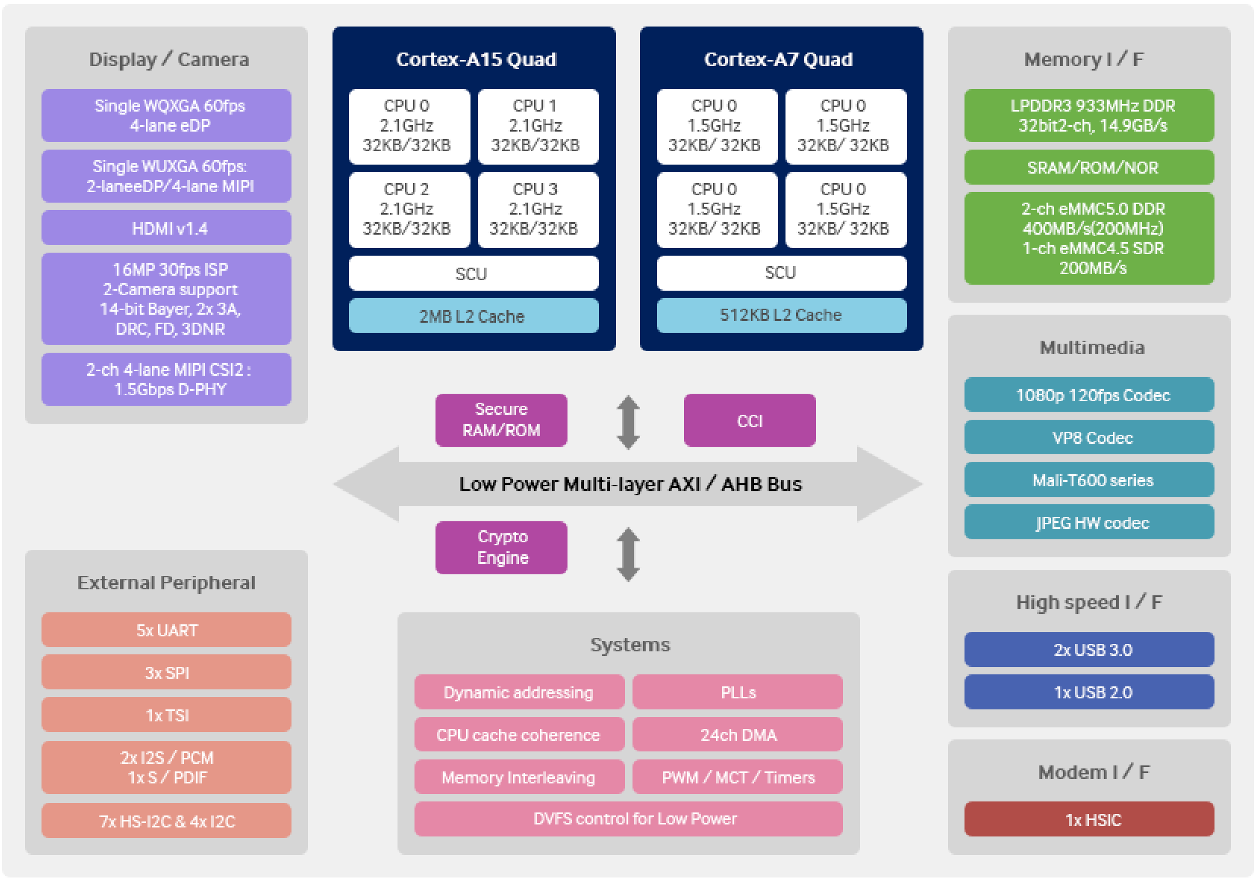


Figure 13. Architecture of the Exynos 5 Octa SoC [SAM15a].

The third iteration of X-Gene is intended for mass production in 2017. It aggressively increase the number of cores (up to 64) and proposes a new socket interconnect technology called X-Tend. This technology aims to connect seamlessly multiple X-Gene SoCs of any iteration [FOR15][SIN14]. From the currently available information, there is no intention to enhance the current cache hierarchy employed in its first and second iteration.

2.2.3 Summary of Private and Shared Cache Architecture

Table 1 summarizes the architecture characteristics of all chips presented in this section and Table 2 summarizes additional technological features of the same set.

Table 1 shows that there is no one-size-fits-all scenario when it comes to multicore/manycore architectures with on-chip cache support. Designs that use private LLC caches tend to stick with uniform access, which is expected since private caches are predominately smaller than shared cache and size is a significant factor in determining the access latency. Besides, private caches are inherently distributed. In the shared LLC domain, we see a trend of using a more diverse set of cache organizations. Power8 is an example that uses an NUCA organization to provide scalability in its huge 96MiB L3 cache.

The interconnect fabric of chips is comprised of a diverse of architectures. Again, there is no optimal solution for this scenario. From the eight chips presented three chips employ packet-based NoC, one chip employs a ring topology, one chip employs crossbars, one chip employs buses, and two chips employ a mixed topology. The number of supported cores varies from eight to a hundred; trending to turn to scalable networks (such as packet-based) appears as the number of cores surpass a few dozen (Tilera-GX 100 and X-Gene 3).

Table 1. Classification, interconnect fabric, number of cores, and cache latencies of chips.

CHIP	CLASSIFICATION (LLC)	INTERCONNECT FABRIC	NUMBER OF CORES	L2 LATENCY	L3 LATENCY
XEON PHI	Private Uniform access Distributed	Three independent bidirectional rings	Up to 61	24 cycles (local) and 250 cycles (remote)	Not present
TILERA-GX 100	Private Uniform access Distributed	Five independent 2D NoC	100	11 cycles (local) and 41 cycles (remote)	Not present
POWER8	Shared Non-uniform access Distributed	Eight on-chip buses	12 per chip	12 cycles	27 cycles (local) and 130 cycles (remote)
SPARC 64X	Shared Uniform Access Distributed	Crossbar based	16 per chip	-	Not present
SPARC M7	Shared Uniform Access Distributed	Point-to-Point and ring topologies	32	-	-
P2012	Shared Uniform Access Centralized	Asynchronous NoC (inter-cluster) and buses (intra-cluster)	Up to 16 per cluster	-	Not present
EXYNOS 5	Shared Uniform Access Distributed	Multi-layer bus	Up to 8	21 cycles	Not present
X-GENE	Shared Uniform Access Centralized	2D NoC	8 cores (gen1&2) Up to 64 cores (gen3)	13 cycles	89 cycles

Xeon Phi is the only chip where cache latency is publicly available. No additional information was found for the SPARC 64x, SPARC M7 and P2012 chips. The remaining chips had their latency experimentally deduced. Applications such as LMbench [MCV15] can deduce cache latency, albeit limited to data latency. The use of this kind of application is common practice in microarchitecture exploration [RUG08]. From the set of information available, there is no clear relation between the classification and latency of the LLC, implying that LLC caches can be tuned to the system demand.

Table 2 presents the process employed in the device fabrication and the existence of 3D chip or prototypes. Only two chips were fabricated with more advanced technologies than 32nm. Both Xeon Phi's Knights Landing and X-Gene 3 are in development and intend to use sub 20nm process. In regards to 3D integration, since 2012 the P2012 development team has been prototyping 3D versions of their product exploiting the Wide I/O DRAM standard [DUR15]. Xeon Phi's Knights Landing also use stacked memory for its system; however, in this case, a proprietary version based on HMC is employed [PLA15].

Table 2. Technology and 3D chips presented.

CHIP	TECHNOLOGY	3D CHIPS/PROTOTYPE
XEON PHI	22nm (Knights Corner) 14nm (Knights Landing)	No (Knights Corner) Yes (Knights Landing)
TILERA-GX 100	40nm	No
POWER8	22nm	No
SPARC 64X	28nm	No
SPARC M7	20nm	No
P2012	28nm	Yes
EXYNOS 5	28nm	No
X-GENE	40nm (gen1) 28nm (gen2) 16nm (gen3)	No

2.2.4 Discussion about the Best Cache Model

The consequences of committing to a given cache architecture are a topic of research that has been long being investigated. This work will also examine these effects on some different schemes using highly-parallel applications. In the following paragraphs, previous results from other authors will be discussed.

Asaduzzaman, Sibai, and Rani [ASA09] show experimental results of *shared* and *distributed* architectures in multicore systems. Distributed L2 caches alleviate traffic congestion in the interconnect fabric since each core has its L2 cache. However, this architecture has the initial cost of feeding data to the distributed caches. Moreover, the area consumption of L2 caches is not negligible [COS09]. In juxtaposition, shared L2 caches increase traffic congestion in the interconnect fabric since each memory request for an L2 cache must traverse it. The initial cost of feeding data for a shared L2 cache can be mitigated if more than one core shares the same data. Furthermore, shared L2 caches are better adjusted to area restriction due to their reduced quantity when compared to the previous architecture. Consequently, they conclude that the impact of adding cores to the system favors the *distributed* architecture. However, the three applications used for testing on their work are limited to the same characteristics: multimedia application profile and little code size that fits entirely in the L2 cache (1 to 2.5KiB).

Yun and Valsan [YUN15] study the effect of using the page-coloring technique on commercial off-the-shelf multicore processors featuring shared LLC. Page coloring can partition the LLC space among the cores, effectively creating isolation on a shared cache. They evaluate the system with two ARM processors (an in-order and an O3), and one Intel O3 processor with corresponding 512KiB, 2MiB, and 8MiB LLC shared caches, respectively.

The caches analyzed in their work employ a Miss Status Holding Register (MSHR) to track the status of ongoing memory requests. This structure allows the cache to be non-blocking – i.e., service additional requests even when a request results in a cache miss. The MSHR effectively determines the maximum number of outstanding memory requests and is commonly shared among a single LLC. Because of this, even if the cache space is partitioned among cores using software-based techniques, accessing the cache partition does not guarantee interference freedom.

Experimental results were conducted using shared LLC evenly partitioned among four cores. This work uses a benchmark to determine access latencies that resemble the technique adopted by LMbench, which is based on a pointer-chasing application. Figure 14 shows the results obtained running platforms with up to four cores for each processor. One core runs the latency benchmark and the other cores, if present, run a bandwidth benchmark to stress the cache and DRAM subsystems.

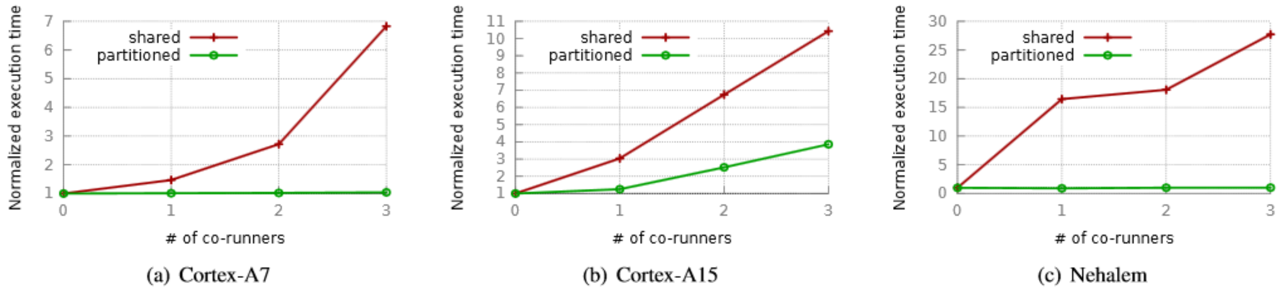


Figure 14. Normalized execution time for three system configurations [YUN15].

The bandwidth benchmark has no data dependency and, therefore, can generate multiple outstanding memory requests on an O3 core. When the cache partitioning technique is not applied, the response times of the latency benchmark are increased substantially in all platforms. This latency increase is attributed to cache-lines of the latency benchmark being evicted by the co-runners tasks. However, in the cache-partitioning scenario, this technique is shown to be effective in preventing such cache line evictions hence providing isolation. The authors also show that this isolation is not always achieved in O3 cores due to the limiting factor of a shared MSHR. One example is the execution of the bandwidth benchmark in Cortex-A15 as shown in Figure 15(b). Cortex-A7 and Cortex-A15 are in-order and O3 cores, respectively.

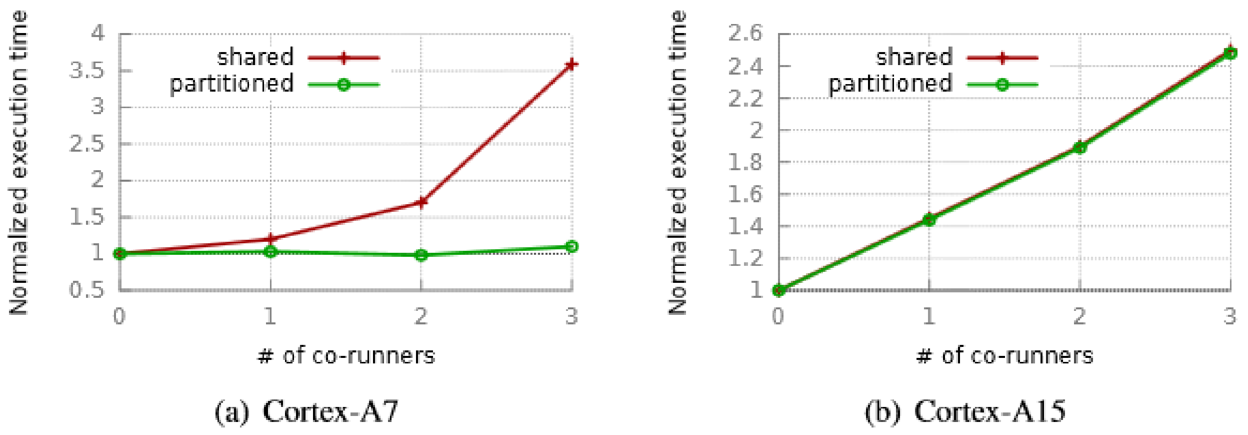


Figure 15. Effectiveness of cache isolation in (a) in-order cores and (b) O3 cores [YUN15].

The study of Yun and Valsan show that software-defined architectures are limited in its ability to provide isolation in a shared cache design, reinforcing the need to choose wisely the intended cache organization in an architecture proposal. Unfortunately, Yun and Valsan employ single-threaded applications and do not analyze parallel applications that are the core beneficiaries of an MPSoC architecture.

Garg et al. [GAR05] identify the design of memory hierarchy in a multicore architecture to be a critical component of the system since it must meet the capacity (regarding bandwidth and low latency) and coordination requirements of multiple threads of control. In their work, the authors investigate the design of the L1 cache for multithreaded/parallel workloads.

The base processor design of their work is the clustered multithreaded architecture. In this design, all major back-end structures of the processor (functional units, register files, issue queues, load-store queue, and the L1 data cache) are partitioned into multiple clusters. The front-end and the L1 instruction cache is shared by all clusters. Figure 16 shows the progression of the logical design of single multithreaded to *Chip MultiProcessor* (CMP)[1] and, finally, to clustered multithreaded architecture, where clusters are statically assigned to threads. There are two major cross-cluster communication networks based on bidirectional rings.

The motivation of the work of Garg et al. is summarized in Figure 17(a) and (b), for 4 and 8 thread executions, respectively, which is the speedup of a centralized shared cache in a clustered multithreaded architecture. Four cases are depicted: centralized cache; centralized cache with increased associativity (16-way); 16-way centralized cache assuming

[1] Although some authors use the CMP terminology, instead of MPSoC, we emphasize that their architecture are the same.

zero port contention and; 16-way centralized cache assuming zero port and interconnect contention. This experiment shows that there is a lot of potential lost due to contention at both the port and interconnect levels. The goal is to validate that a decentralized cache in this architecture is a better solution.

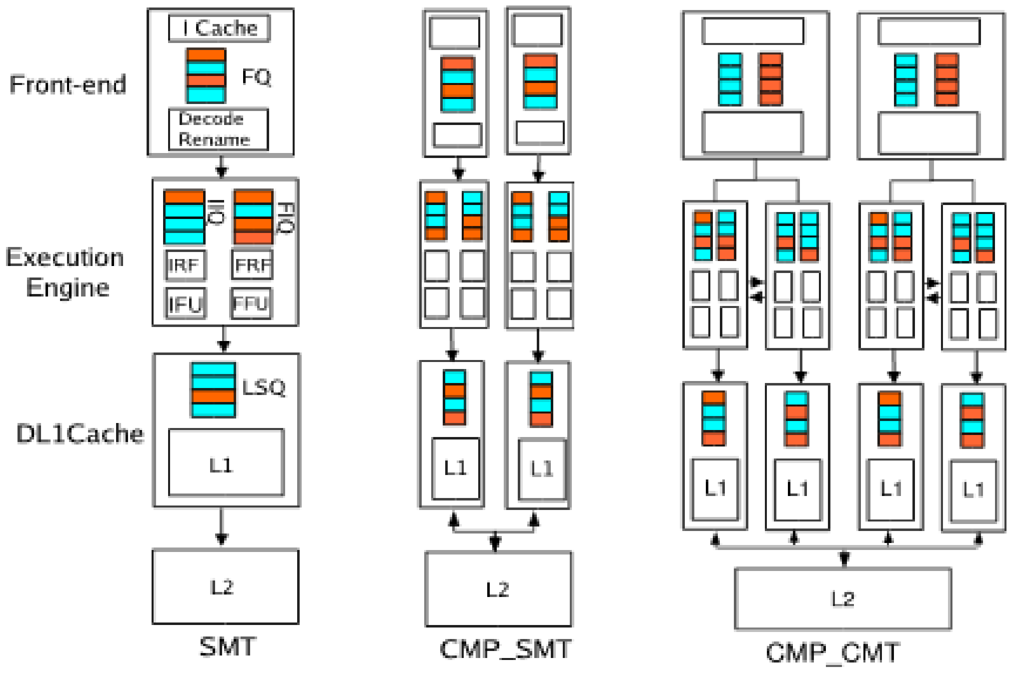
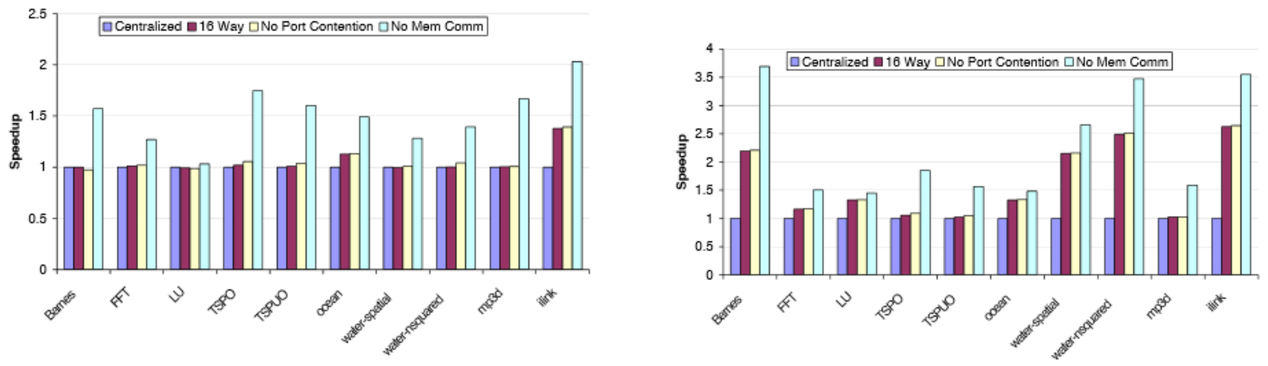


Figure 16. From single multithreaded to CMP and clustered multithreaded architecture (based on [ELM05]).



(a) Number of Threads = 4. (b) Number of Threads = 8.

Figure 17. Performance of the centralized cache [GAR05].

Garg et al. [GAR05] evaluate five cache architectures whose cache banks are statically assigned to each thread of execution. The classification of cache architecture is as following:

Coherent Cache: fully coherent cache, where each L1 data cache partition is directly accessible only by the thread running on a local cluster. Only a single copy of data is allowed inside a cache partition set. A snoopy-based MESI coherence protocol is assumed.

Distributed Cache: The processor can access directly (although with non-uniform latencies) all L1 data banks, even those in remote clusters. The latency of accessing remote L1 is smaller than going through the L1-L2 interconnect and accessing the L2 cache.

Non-Replicated (NR) Cache with First Touch Policy: In the NR cache, as in the distributed cache, data is not replicated in multiple partition-sets. Instead of it, data is placed in the partition-set of the first requester.

NR+Migration cache: Extend the NR cache to allow migration of data. The cache maintains threshold values to determine if a line is to be migrated or not. Their work uses two 3-bit counter to determine the number and the source of accesses.

Selectively-replicated Cache: This cache combines the benefit of both NR and fully coherent cache. Dynamically, threshold values are employed to determine if access will be done remotely (as done in the NR cache) or locally through the copying mechanisms of the coherence protocol. These threshold values are architecture dependent.

Experimental results were conducted using a modified version of Simplescalar-3.0 for the ALPHA instruction set. The clustered multi-threaded architecture comprises 8 clusters. The size of L1 cache in the base design is 128KiB, so each partition of the distributed cache is 16KiB. For the L1-L2 interconnect, a split-transaction bus is assumed. For applications, SPLASH-2 benchmark along some locally available parallel applications are used.

Figure 18 shows the normalized execution time of the previously described cache architectures in some benchmark applications. The execution time is normalized to a single-thread centralized cache architecture. Average time spent in synchronization with each cache organization is also shown in black at the bottom of each bar. The primary disadvantage of a shared cache, either centralized or distributed, is the increase of inter-thread conflicts due to the limited associativity and porting of a typical L1 cache. This effect is mitigated by providing a private cache to each thread, as is the case of all other architectures in this figure.

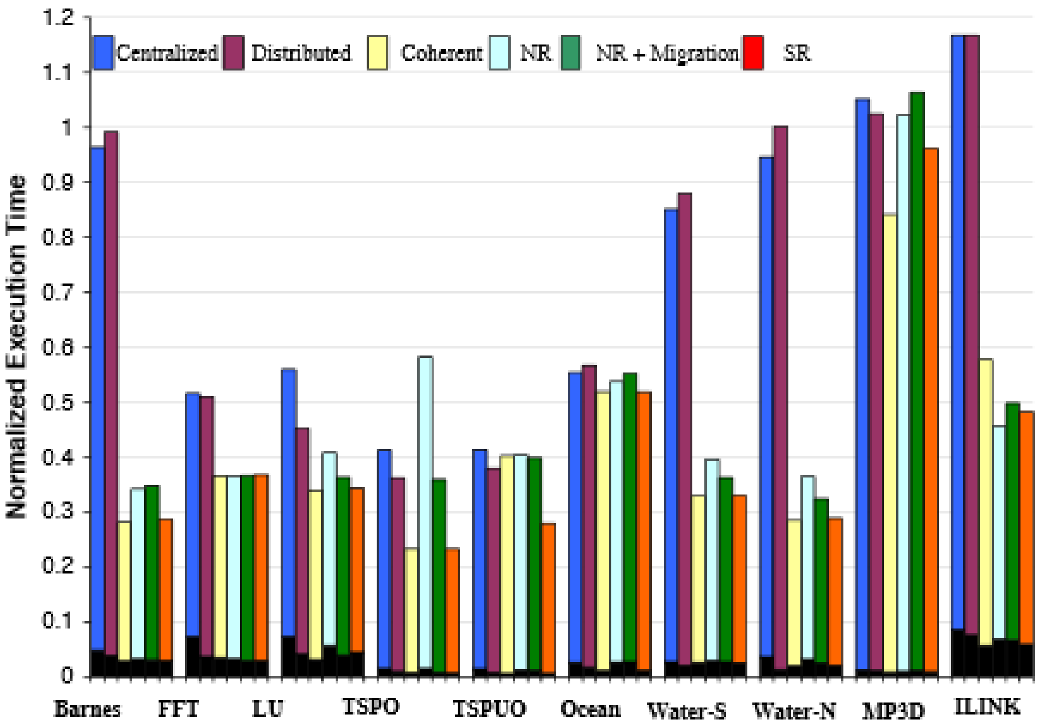


Figure 18. Normalized execution time of six cache architectures [GAR05].

Figure 19 complements the results of the last figure by providing a detailed analysis of the overall number of operations on each cache organization. Analyzing the TSP application, we can see that the number of remote direct read hits when using the NR organization is as high as 40% of the total operations, which contributes to its higher execution time when compared to the coherent cache organization.

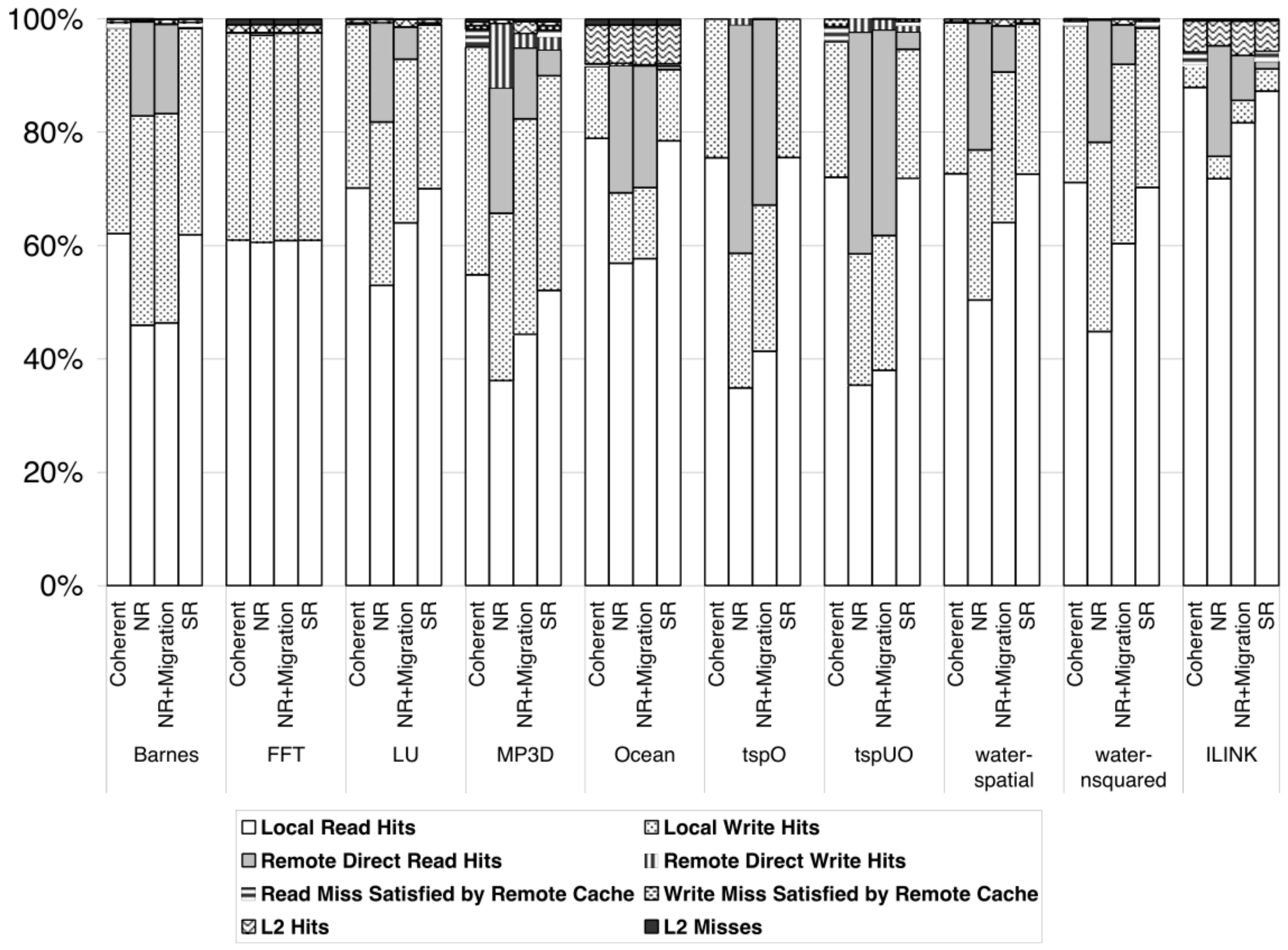
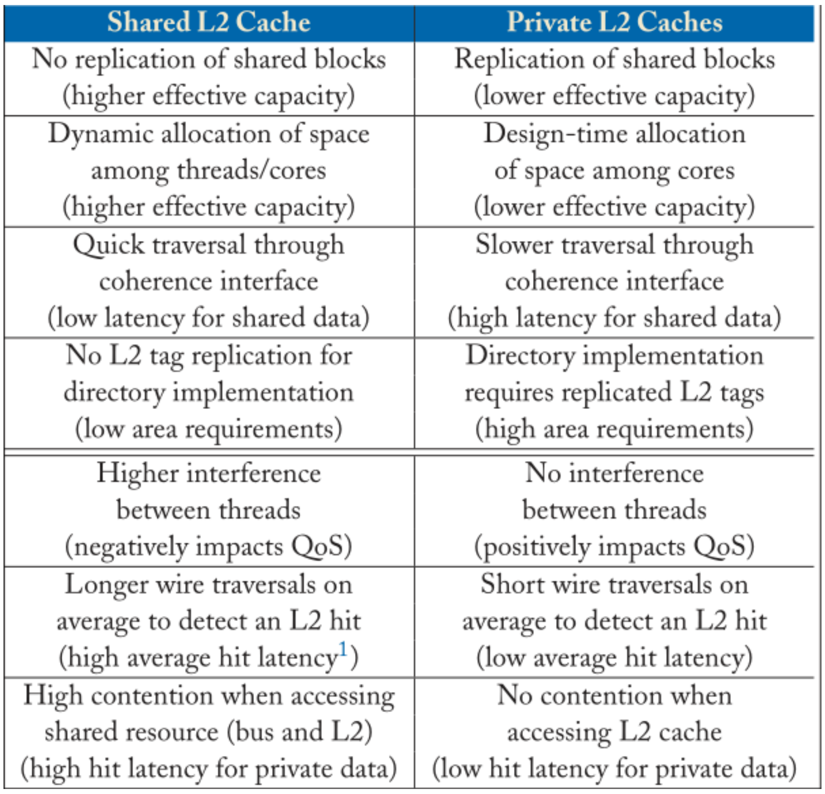


Figure 19. Overall operations in four cache organizations [GAR05].

In summary, Garg et al. designed and evaluated some options for L1 cache organizations. For the workload analyzed, mainly composed of parallel applications, the presence of at least a private partition in this cache showed significant improvement in performance when compared to a pure shared L1. Techniques such as selectively-replication showed interesting results but requires modifications on some of the applications to provide the optimal performance.

This section presented some of the work in the research of cache organizations. Balasubramonian and Jouppi [BAL11] provides a summary of advantages and disadvantages of shared and private L2 caches in Table 3. In the succeeding chapter, we will explore additional cache organizations proposed for scalable manycore/MPSoC systems. These propositions will employ new technological advances to enhance the overall performance of the system. Two technological advances will frequently appear: 3D integration and new emerging non-volatile memory technologies.

Table 3. Advantages and disadvantages of shared and private L2 caches [BAL11].



Shared L2 Cache	Private L2 Caches
No replication of shared blocks (higher effective capacity)	Replication of shared blocks (lower effective capacity)
Dynamic allocation of space among threads/cores (higher effective capacity)	Design-time allocation of space among cores (lower effective capacity)
Quick traversal through coherence interface (low latency for shared data)	Slower traversal through coherence interface (high latency for shared data)
No L2 tag replication for directory implementation (low area requirements)	Directory implementation requires replicated L2 tags (high area requirements)
Higher interference between threads (negatively impacts QoS)	No interference between threads (positively impacts QoS)
Longer wire traversals on average to detect an L2 hit (high average hit latency[1])	Short wire traversals on average to detect an L2 hit (low average hit latency)
High contention when accessing shared resource (bus and L2) (high hit latency for private data)	No contention when accessing L2 cache (low hit latency for private data)

3 RELATED WORK

This chapter presents state-of-the-art work related to the memory hierarchy of MPSoC architectures. Section 3.1 describes 2D MPSoC architectures using primarily NoC-based communications and Section 3.2 describes 3D MPSoC architectures employing diverse communication paradigms due to advantages of 3D integration.

3.1 Cache Organization in 2D MPSoC Architectures

For 2D MPSoC, L1 cache is predominantly integrated into the core area. As observed in Section 2.2 there is not still a dominant architecture paradigm for L2 caches [SAB10]. This section presents state-of-the-art 2D MPSoC architectures with on-chip cache support.

3.1.1 Ye et al.: Regional Cache Organization for NoC based Many-core Processors

Ye et al. [YE10] study the effect of using eight designs of L2 caches in a NoC-based MPSoC. Figure 20 shows two of eight L2 caches studied. Their work clearly showed that the dominant factor of L2 access latency was the packet-based NoC latency. Even reduced injection rates of packets (≤ 1%) resulted in significant network congestion on mesh topologies such as 8×8. Besides, both Ye et al. [YE10] and Wang et al. [WA08] studies conclude that on-chip traffic congestion is predominantly caused by the intensive memory access of requests and responses.

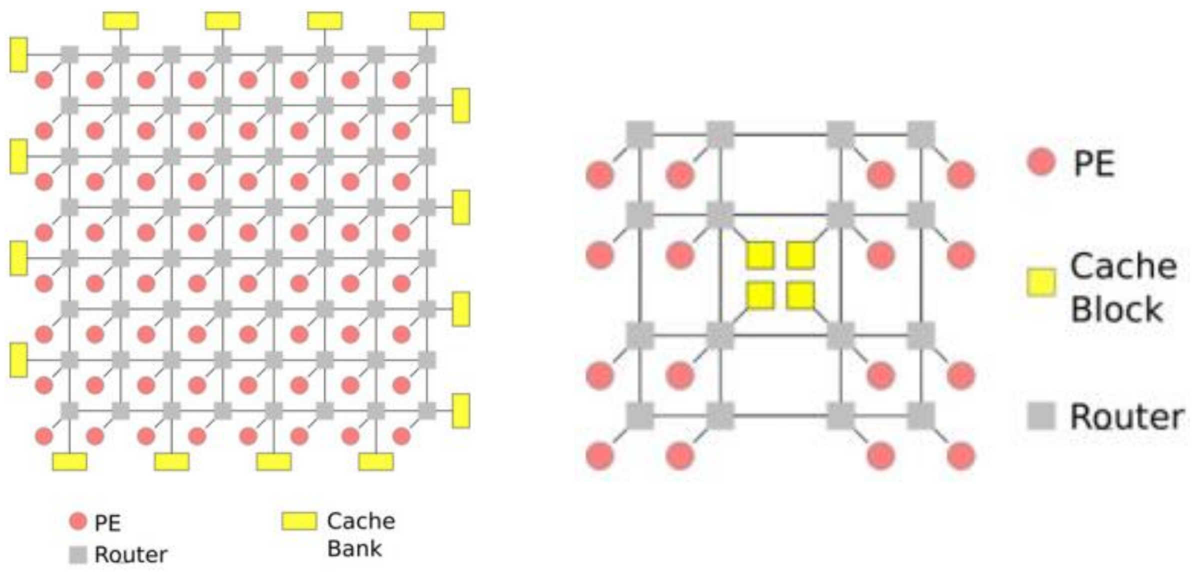


Figure 20. Two of eight L2 cache designs studied on a 2D MPSoC [YE10].

Figure 21 shows average delays of four cache organizations connected to 64 PEs. The L1 cache miss rates of the PEs are measured using the SPLASH-2 benchmark and fed to a NoC simulator. The range of miss rates encountered were of 1% up to 9%. The sizes of all caches are assumed as follows: 32KiB for L1 instruction cache, 64KiB for L1 data cache, and 64KiB for L2 cache.

It is clear from Figure 21 that a naive approach to the cache organization in MPSoC designs can lead to thousands of cycles of delay. This delay is far worse than a single access to the main memory – nullifying the benefits of using a memory hierarchy.

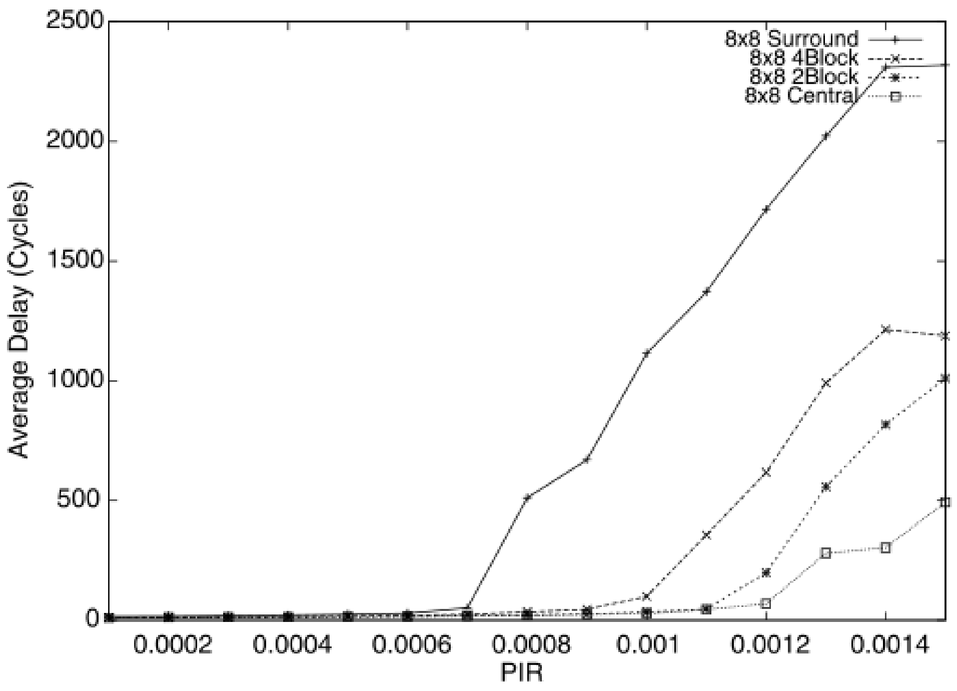


Figure 21. Average delay of four L2 cache designs on the 8×8 2D MPSoC [YE10]. PIR is the packet injection rate expressed in packets per cycle.

3.1.2 Cho and Jin: Managing Distributed, Shared L2 Caches through OS-Level Page Allocation

Cho and Jin [CHO06] present a mixture of shared and distributed L2 cache architectures to achieve optimal performance under diverse workloads. In their work, each core has an L2 slice controlled by the operating system (OS). An L2 slice is a smaller set of a full L2 cache. The OS controls where a cache line will be placed, locally or remotely, resulting in different access times. In essence, this means that the OS can choose to use any cache architecture according to its policies. For single-threaded applications, this scheme can provide additional cache space through other idling cores – in essence, creating a new remote LLC. For parallel applications (SPLASH-2), performance did not increase as expected when compared to pure private or shared caches. The authors hypothesize that SPLASH-2 is optimized to maximize data locality to small caches. As discussed in Section 2.2.4, another important factor showed by Yun and Valsan [YUN15] is that partitioning a cache space into private slices does not guarantee isolation since all slices share the MSHR structure. Figure 22 displays the cache organization in Cho and Jin's work.

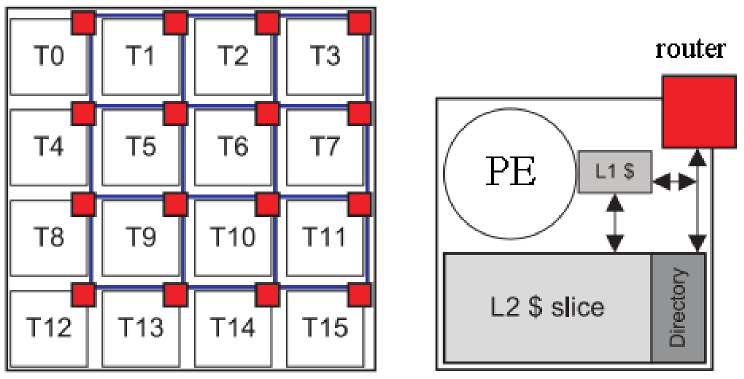


Figure 22. Sixteen core tiles (left) and the design of a single core (right) (based on [CHO06]).

3.1.3 Shirifi et al.: Addressing End-to-End Memory Access Latency in NoC-Based Multicores

Sharifi et al. [SHA12] acknowledge the high latency of memory accesses in NoC-based multicore and proposes two network prioritization schemes that can cooperatively improve performance by reducing end-to-end memory access latencies. Figure 23(a) depicts the five steps that comprise the end-to-end communication. Hits in the L1 cache do not use the NoC; however, hits in the L2 cache use the path-1 and path-5 of the network. In the case of a miss event in L2, the request must reach the closest memory controller and reach back, as a response, all the way back to the L1 (all paths of Figure 23(a)). Figure 23(b) shows that traversing the NoC can reach approximately the same delay of the main memory latency. This measurement was done through a simulation of a 4×8 multicore with 4 memory controllers (placed on each corner), and each core running applications from the SPEC2006 benchmark. The results of one of the cores running the milc application from SPEC2006 are plotted. The average latency was approximately 400 cycles.

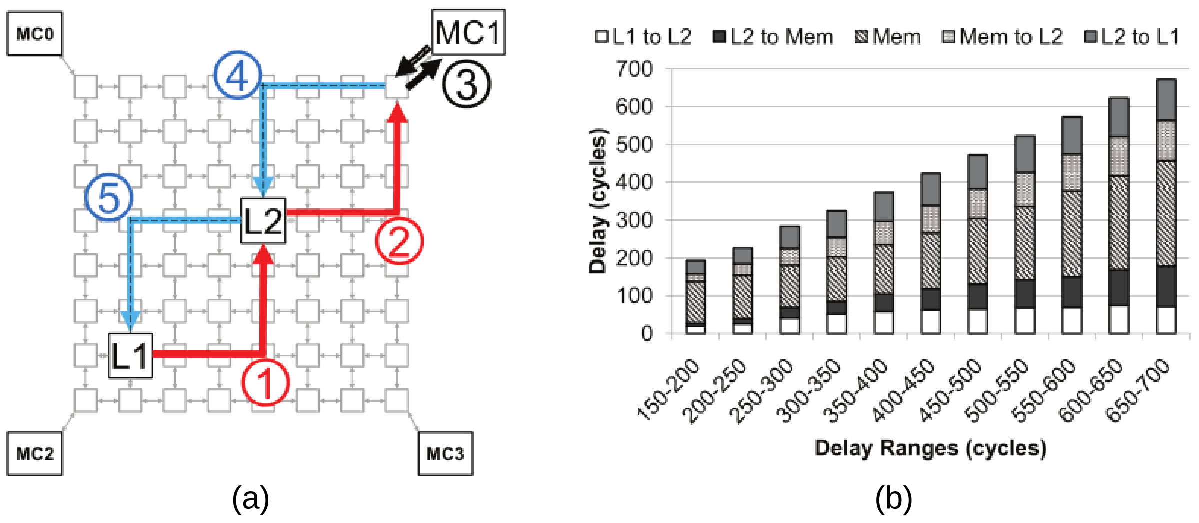
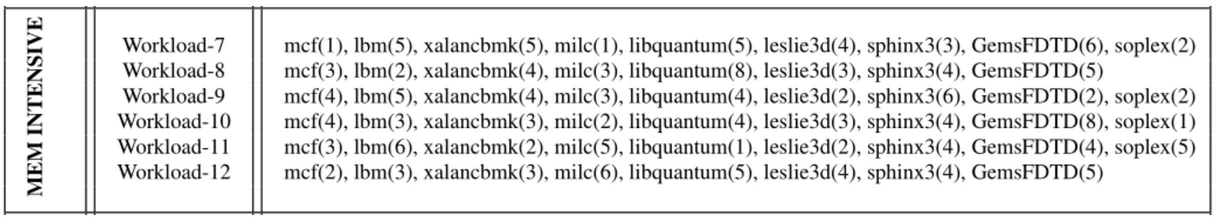


Figure 23. Detailed flow (a) and delay (b) of memory requests and responses [SHA12].

The first network prioritization scheme proposed by Sharifi et al. tries to expedite late messages on their returning path by giving them a higher priority in the on-chip routers. This procedure aims to reduce the memory delay variance experienced by an application. For this, packets are annotated with an age field (the so-far delay) and compared to a predefined threshold value. Every 1ms, the core periodically determine and update the threshold values. If it is larger than the threshold value, this message is considered late and will have a higher priority on its return path. In each memory controller, a mapping of each core/application with a threshold value is created.

The second network prioritization scheme is to increase the utilization of idling memory banks in the system. To do this requests destined for the idle banks are given higher priorities in the network to reach their target faster. However, given the distributed nature of a NoC, there is no global information available to routers to help them decide whether a given packet is destined for an idle bank or not. To address this situation each router estimate the pressure imposed by requests recording local packets passing through it. Requests addressed to banks that are not recorded locally are given higher priority in the on-chip network. A threshold value is used to clear old requests. The higher priority of packets is achieved through a specialized pipeline in the routers. The standard routing procedure used in Sharifi et al. work is comprised of five stages. However, when a higher priority packet is received a two-stage pipeline is employed.

Figure 24 summarizes the benchmark applications; where the numbers in parenthesis represent the amount of copies for each application in the workload. Generally, both prioritization schemes improve the performance of the system. However, this is not true for the w-9 workload under scheme-1. The authors hypothesize that by giving a higher priority to some of the messages in the network, they hurt other messages, and this can offset the benefits of this scheme. Their work presents further results regarding some threshold values and workloads.



MEM INTENSIVE	Workload-7	mcf(1), lbm(5), xalancbmk(5), milc(1), libquantum(5), leslie3d(4), sphinx3(3), GemsFDTD(6), soplex(2)
	Workload-8	mcf(3), lbm(2), xalancbmk(4), milc(3), libquantum(8), leslie3d(3), sphinx3(4), GemsFDTD(5)
	Workload-9	mcf(4), lbm(5), xalancbmk(4), milc(3), libquantum(4), leslie3d(2), sphinx3(6), GemsFDTD(2), soplex(2)
	Workload-10	mcf(4), lbm(3), xalancbmk(3), milc(2), libquantum(4), leslie3d(3), sphinx3(4), GemsFDTD(8), soplex(1)
	Workload-11	mcf(3), lbm(6), xalancbmk(2), milc(5), libquantum(1), leslie3d(2), sphinx3(4), GemsFDTD(4), soplex(5)
	Workload-12	mcf(2), lbm(3), xalancbmk(3), milc(6), libquantum(5), leslie3d(4), sphinx3(4), GemsFDTD(5)

Figure 24. Workload employed by Sharifi et al. [SHA12].

Figure 25 depicts results obtained through a simulation of 32 cores interconnected in a 4×8 2D mesh.

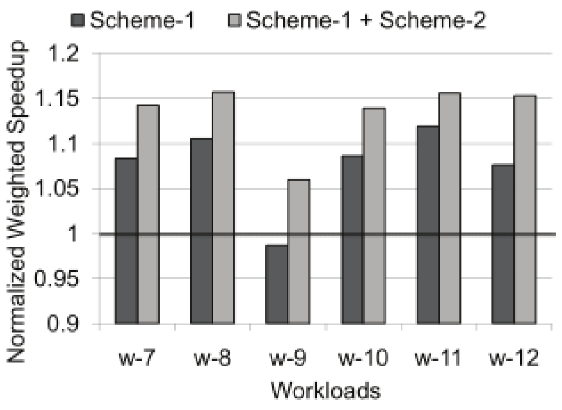


Figure 25. Normalized speedup for memory intensive workloads [SHA12].

3.2 Cache Organization in 3D MPSoC Architectures

3D MPSoCs introduced new potential architectures that are not commonly found on their 2D counterpart. The ability to stack multiple dies with diverse fabrication processes enabled the exploration of such architectures. Examples of such explorations are the use of emerging memory technologies in lower cache levels and even the presence of main memory in stacked dies. These emerging technologies present much higher density when compared to traditional technologies with the cost of higher access latency. This section presents a non-exhaustive list of current research effort on 3D MPSoC with cache hierarchy support bringing design possibilities, benefits, and drawbacks.

Zhang et al. [ZHA14] summarize current research effort in the design of 3D CMP, with tiers[2] dedicated for memory hierarchy. The survey focuses on two categories of architectures for 3D CMPs: stacking cache-only and stacking main memory. The stacking cache focuses on the use of hybrid cache architectures that combine the benefits of the fast

[2] In the 3D technology, a tier consists of a single 2D layer, and two or more tiers are stacked and connected to perform a 3D system.

accesses of SRAM with emerging memory technologies that allow scalable space under chip area and power constraints. The stacking main memory use TSVs to increase the bandwidth and mitigate the Memory Wall limitations [WUL95].

3.2.1 Loi and Benini: An Efficient Distributed Memory Interface for Many-Core Platform with 3D Stacked DRAM

Figure 26 shows a 2D NoC with a stacked memory layer ([LOI10]). Each processor has fast access to a stack of memory banks on its top and remote slower access to memory stacks of other processors. Access to remote memory stacks is done through a NoC.

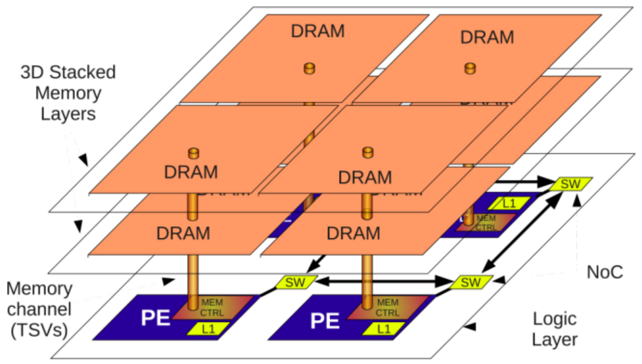


Figure 26. 3D hardware architecture [LOI10].

Figure 27 shows the architecture employed in the Loi and Benini [LOI10] work. The green square depicts a local access to memory, and the red lines depict an access to a remote memory. The 3D DRAM module is the module stacked over the rest of the system.

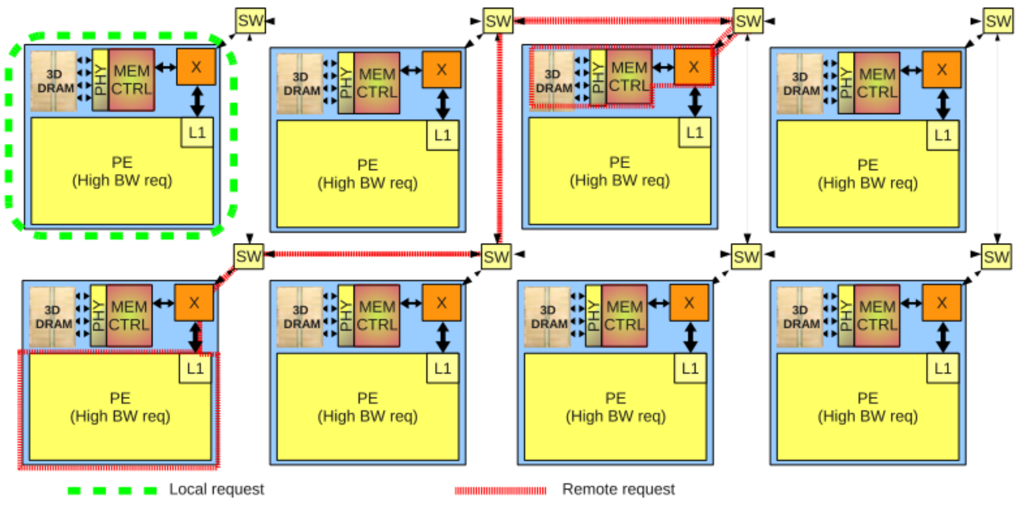


Figure 27. Illustration of the architecture proposed by [LOI10].

The main contribution of their paper is the development of a 3D-DRAM controller responsible for administering the two types of memory accesses described above. In addition, the data bus used in their work was enhanced from the traditional bi-directional bus to two independent buses for read and write. Additionally, the width of the bus is increased to a range of 32 bits up to half of the row size. Therefore, access to main memory is faster than its traditional counterpart. Further, the power dissipation is reduced by stacking memory die on top of the processor instead of using off-chip pins (24 µW vs. 30-40 µW).

Cadence SoC Encounter and Synopsys Design Compiler were used to obtain a synthesized platform. Experimental results using 256-bit bus width show peaks of 4.53 GB/s for local memory access, and 850 MB/s for remote access through the NoC. The bandwidth improvement ranges from 1.44x to 7.40x when compared with the JEDEC DDR (*Double Data Rate*) standard.

3.2.2 Fu, Liu, and Chen: Direct Distributed Memory Access for CMPs

Fu, Liu, and Chen [FU14] propose a distributed memory with direct access to local and remote cache banks due to previous unsatisfactory results achieved with packet-based communication. The local cores access remote memory through remote-to-local virtualization without any network protocol translation. Every core has an auxiliary memory controller to access the local and remote memories. This controller is divided into two circuits for managing the core communication and handling the actual memory bank. Interconnection for memory accesses between cores is done using multiple buses. Two unidirectional buses are employed for each combination of column and row of a mesh topology. For an 8×8 topology, 16 buses are used. Each core in this architecture accesses a private 16KiB L1 cache and a 64KiB L2 cache. Figure 28 depicts the architecture of the system.

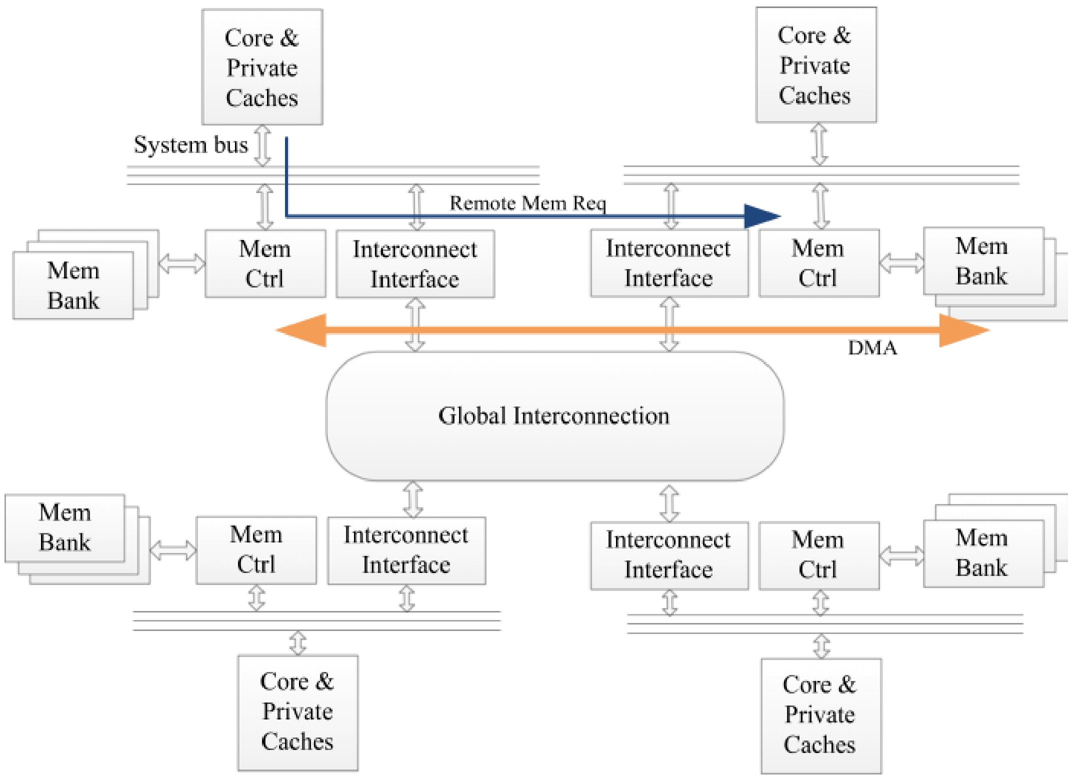


Figure 28. Distributed memory with direct access model [FU14].

Simulation results with the PARSEC benchmark show that direct memory outperforms packet-based access regarding both memory access latency and IPC by 17.8% and 16.6%, in average, respectively. The rationale for this is twofold: the lack of network protocol translation overhead in the direct memory access and the reduced contention for using multiple independent connections. The Gem5 simulator was employed to trace the memory accesses of both PARSEC and SPEC2006 benchmarks. Subsequently, they were fed to an in-house C++ cycle-accurate simulator responsible for simulating all implementation details of both direct and packet-based communication. Figure 29 depicts the average access latency breakdown for both architectures: directed memory access model (herein called DDMA) and; packet-based memory access (herein called PDMA).

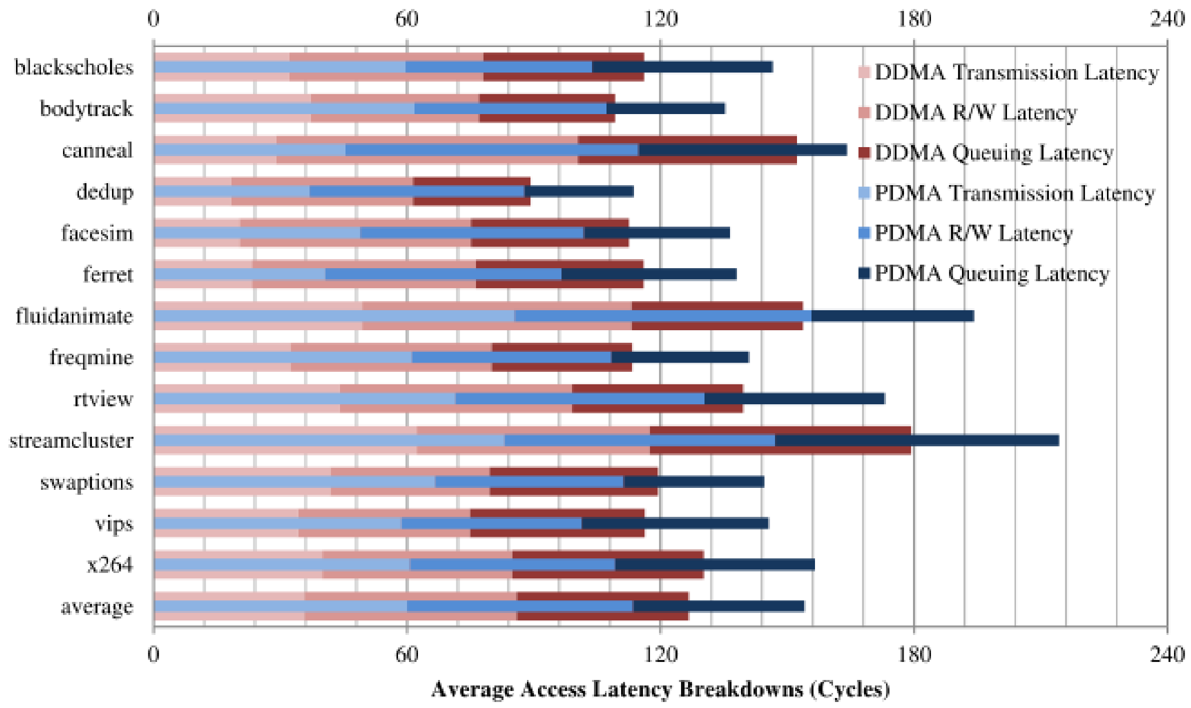


Figure 29. Average access latency breakdown for the PARSEC benchmark suite [FU14].

3.2.3 Woo et al.: An Optimized 3D-Stacked Memory Architecture by Exploiting Excessive, High-Density TSV Bandwidth

Woo et al. [WOO10] propose to redesign the traditional L2 and DRAM interface to exploit the advantages of TSV integration. The idea is to leverage the TSV bandwidth to hide latency behind enormous data transfers. To tackle the Memory Wall problem, the authors prefetch entire memory pages (4KiB) instead of a normal 64-byte cache line. Traditionally, this type of massive transfer is avoided since it results in a trailing-edge effect on the memory bus and degrades the system’s performance. In their work, the authors exploit the TSV connections to avoid the trailing-edge effect. Initially, Woo et al. intended to increase the cache line size to the prefetch size. However, they found out that the access latency increases almost linearly with the line size – assuming an H-tree subarray cache design as depicted in Figure 30.

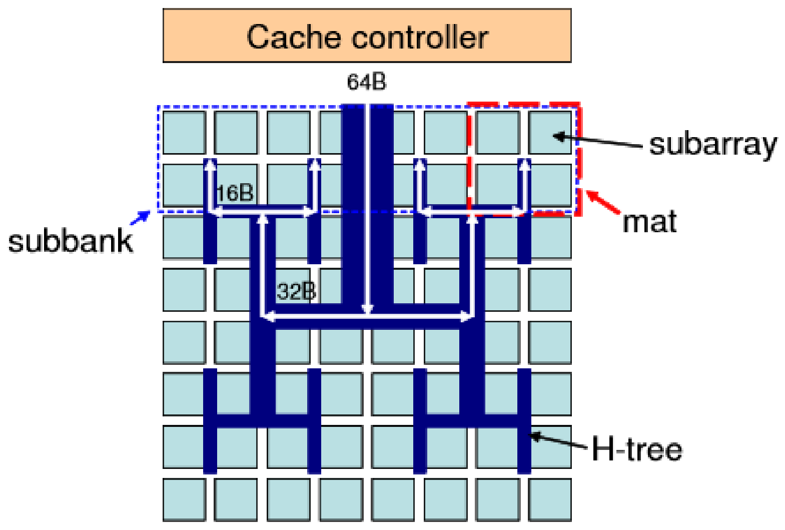


Figure 30. Cache subarray design [WOO10].

To overcome this limitation a new layer is added comprised of an L2 cache subdivided into sixty-four sub-banks. In this design, a read or write operation uses the conventional 64-byte line while a fill or write-back operation uses a new TSV bus to write 4KiB pages. Figure 31 shows this design, where 32K TSVs are needed to provide a high-performance bus.

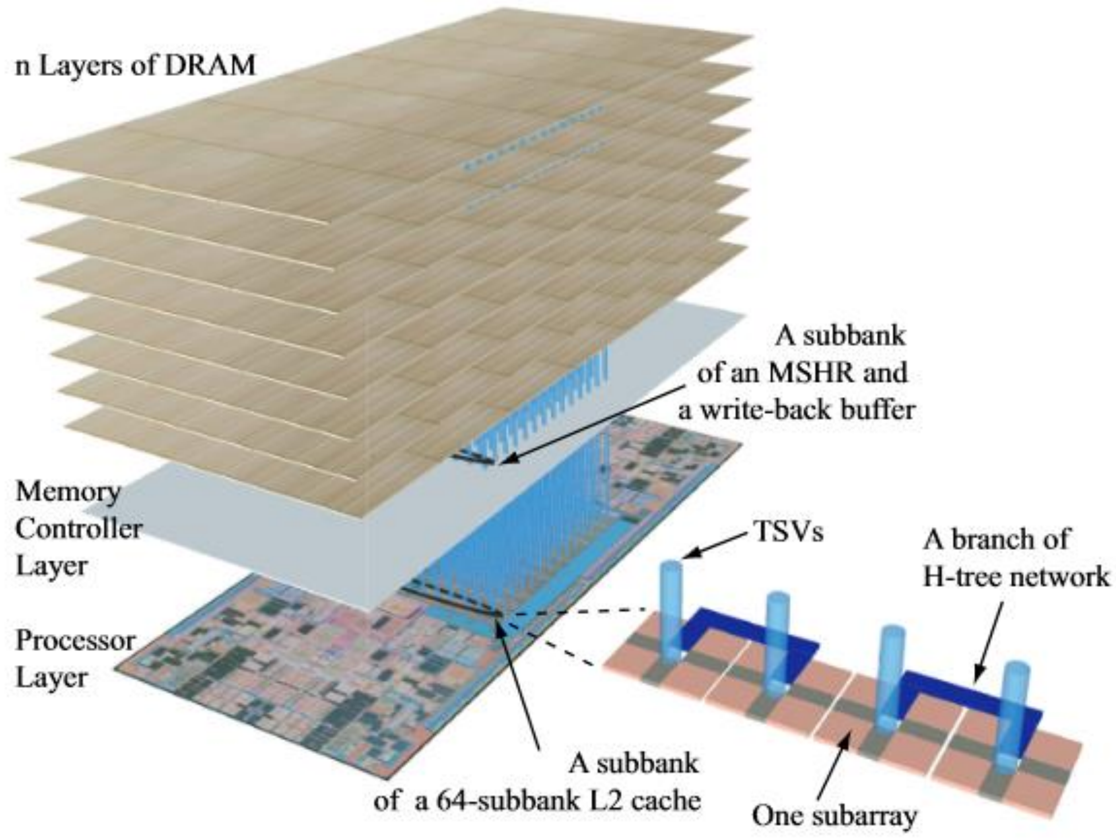

n Layers of DRAM
A subbank of an MSHR and a write-back buffer
Memory Controller Layer
TSVs
A branch of H-tree network
Processor Layer
A subbank of a 64-subbank L2 cache
One subarray

Figure 31. SMART-3D design: 64 bytes-wide bus with TSVs directly on top of each L2 sub-bank [WOO10].

Experimental results were conducted using the SESC simulator. Evaluation of single- and multi-threaded applications were conducted using the SPEC2006, NuMineBench, and other benchmarks. The results were compared to other types of architectures including a traditional 3D mesh NoC-based with and without Global History Buffer (GHB) prefetcher. Figure 32 depicts one of the results obtained with the SPEC2006 benchmark. The SMART-3D (2MB) outperforms all other models except in the benchmarks 436 / 456 and 436 / 483. The authors assert that they could improve its execution by enhancing page-level locality in the 436 benchmark. It is also important to note that, in most cases, the SMART-3D 1MB can outperform a 3D-GHB 4MB. The performance shown here scales to an 8-core system too, but with a less significant degree due to the increased cache contention.

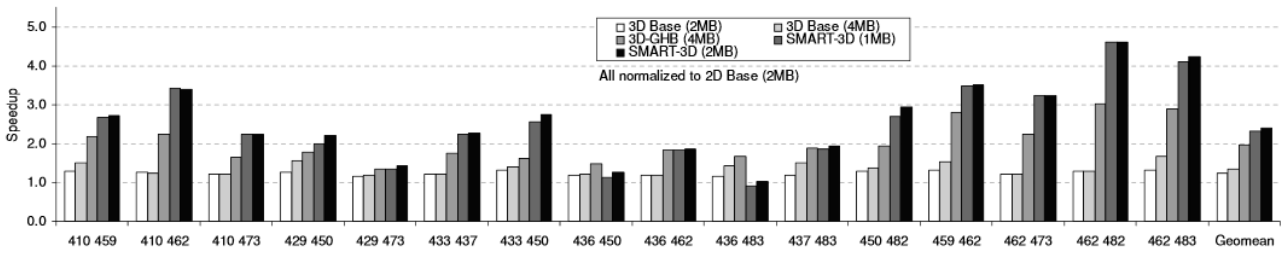


Figure 32. 2-Core system performance improvement under different architectures [WOO10].

Figure 33 shows the energy consumption comparison between the 3D-Base and 3D-SMART architecture obtained through CACTI and Synopsys Raphael. 3D-SMART can save huge amounts of energy when an application has enough spatial locality, such as 437 and 450 because prefetching entire memory pages reduce the row buffer misses. On the other hand, applications that hop between different pages will consume more energy in this architecture.

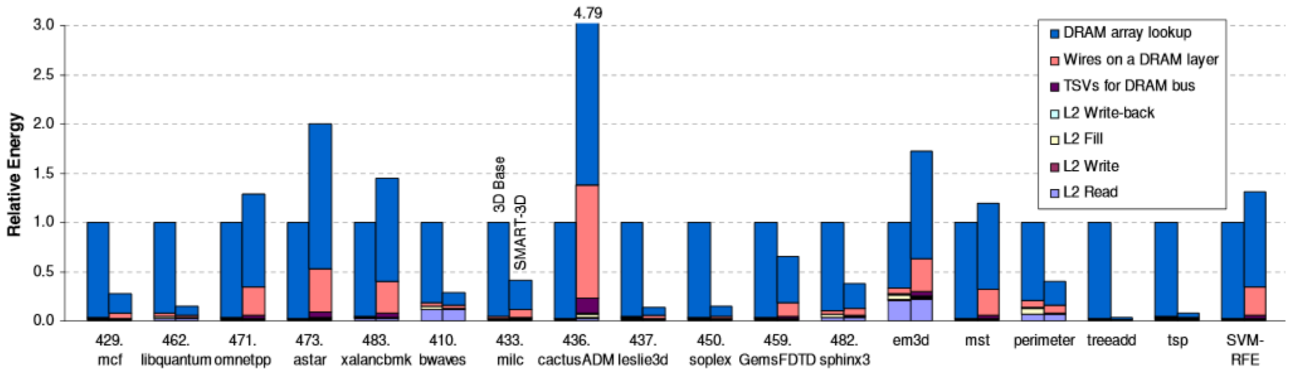


Figure 33. Dynamic Energy Consumption of 3D-Base and 3D-SMART [WOO10].

3.2.4 Kim et al.: Design and Analysis of 3D-MAPS (3D Massively Parallel Processor with Stacked Memory)

Kim et al. [KIM13a] describe a Massively Parallel Processor (MPP) with stacked memory called 3D-MAPS, which consists of a 64-core tier and a 64-memory block tier. Each core communicates with its dedicated 4 KB SRAM block. Intercommunication is achieved through a 2D 8×8 mesh. This chip was built with a two-tier 3D stacking technology using approximately 50K TSV and 50K face-to-face bond pads. The estimated fabrication cost of 3D-MAPS compared to a theoretical 2D-MAPS (each memory block placed right beside its corresponding core) is approximately half the cost.

The number of inter-tier connections for core-memory communication far exceeds the traditional number of such designs. It even exceeds the Wide I/O single data rate standard that is intended for 3D designs (approximately 7400 vs. 800 connections). The intention is to allow the system to read/write to memory every clock cycle. Therefore, fully exploiting the wide memory bandwidth.

The design and analysis of this chip were conducted using commercial tools from Cadence, Synopsys and Mentor Graphics, as well as in-house tools for handling 3D technology characteristics. The maximum frequency achieved to the cores was 277MHz, which results in a theoretical maximum memory bandwidth of 70.912GB/s. The highest memory bandwidth observed in simulations was 90% of this value (median filter benchmark) and the minimum value observed was 13% (string search benchmark). Current Intel Core i7 clocking in 1333MHz has approximately 64GB/s as its maximum memory bandwidth. For seven benchmarks, the peak power consumption of 3D-MAPS ranges from 3.5W to 4W.

Kim et al. are currently working on a second version of this chip, called 3D-MAPS v2, which will be comprised of five tiers: two logic chips and three DRAM chips. Further, they intend to double the core count for this version [GEO15].

3.2.5 Fick et al.: Centip3De: A Cluster-Based NTC Architecture With 64 ARM Cortex-M3 Cores in 3D Stacked 130 nm CMOS

Fick et al. [FIC13] present a large-scale 3D CMP with a cluster-based near-threshold computing architecture called Centip3De. A 3D stacking technology is used in conjunction with 130 nm CMOS (*Complementary Metal-Oxide-Semiconductor*). The 64 cores are organized into 4-core clusters, and their aggregate cache is combined with a single shared 4×-larger cache. This larger cache has an increased voltage and frequency to assist all four cores. The use of an increased frequency, when compared to the cores, allows the cache to maintain single-cycle latency, even when the access is shared. In the 130nm design, the cores operate between 10 and 80MHz.

Eight buses of 128 bytes assist the communication of all 16 clusters. The buses are split into two columns that span the cache and core layers of the chip. By building the bus architecture vertically, the authors reduced by 50% the required routing resources when compared to a single-layer floorplan.

The system architecture was described and validated using the Gem5 simulator, and the experimental results were conducted using the SPLASH-2 benchmark. Figure 34 depicts the results obtained with a power budget of 250 mW and a configuration described in the format 4-core/3-core/2-core/1-core clusters. As core count diminishes, the frequency and voltage of each core increases and this allows the single-threaded (herein called S/T) performance to improve. The future version of this chip intends to employ a seven-layer chip: two of cores, two of caches and three of DRAMs.

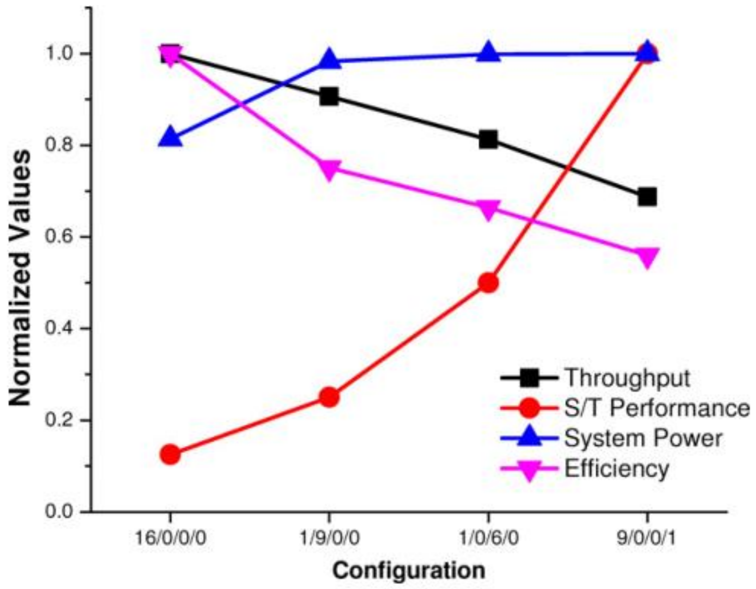


Figure 34. Range of system configurations under a 250 mW power budget [FIC13].

3.2.6 Guithmuller, Miro-Panades, and Greiner: Adaptive Stackable 3D Cache Architecture for Manycores

Guthmuller, Miro-Panades, and Greiner [GUT12] present a modular and scalable manycore architecture with multiple stacked cache tiles, as shown in Figure 35. In this architecture, each cache controller is responsible for a given segment of the main memory. The OS can configure the controller to allocate private sections within the cache for any given application. At assembly time, the architecture can incorporate multiple stacks of identical cache tiers. Besides, at runtime, the cache quantity allocated to a memory segment can be tuned. As the time for the processor to fetch data varies according to memory addresses, this mapping defines an NUMA architecture. The tier encompasses cores with L1 and L2 caches interconnected by a local bus, and the additional cache tier serves as an L3 cache.

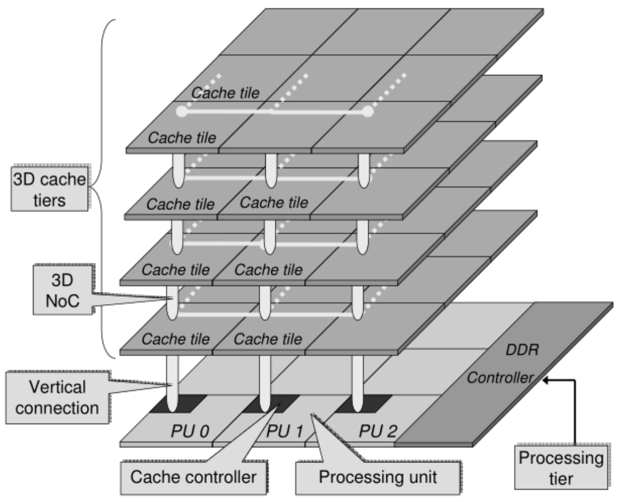


Figure 35. Manycore architecture proposed by Guthmutter, Miro-Panades and Greiner [GUT12].

Table 4 describes the test cases employed in the SoCLib platform to perform experimental analysis of the proposed architecture. Two memory configurations were experimented: an SRAM cache and a mixed SRAM/eDRAM cache. Each memory tier includes 16MiB of cache. The total area of the cache block is reduced by 47% with eDRAM/SRAM (compared to pure SRAM) since eDRAM presents higher density degree.

Table 4. Six test cases scenarios [GUT12].

Test case	App 1	App 2	L3 cache
1	FFT 2^{18}	FFT 2^{14}	1 tier
2	FFT 2^{20}	FFT 2^{18}	4 tiers
3	FFT 2^{20}	LU 512	1 tier
4	FFT 2^{18}	LU 512	1 tier
5	FFT 2^{18}	LU 1024	1 tier
6	FFT 2^{20}	LU 512	4 tiers

The first applications are always the most memory intensive of the set. Figure 36(a) shows the normalized execution time for all test cases on cache configurations (lower is better), and Figure 36(b) shows the FFT 2^{20} case where adding new cache tiers enhanced the performance application significantly.

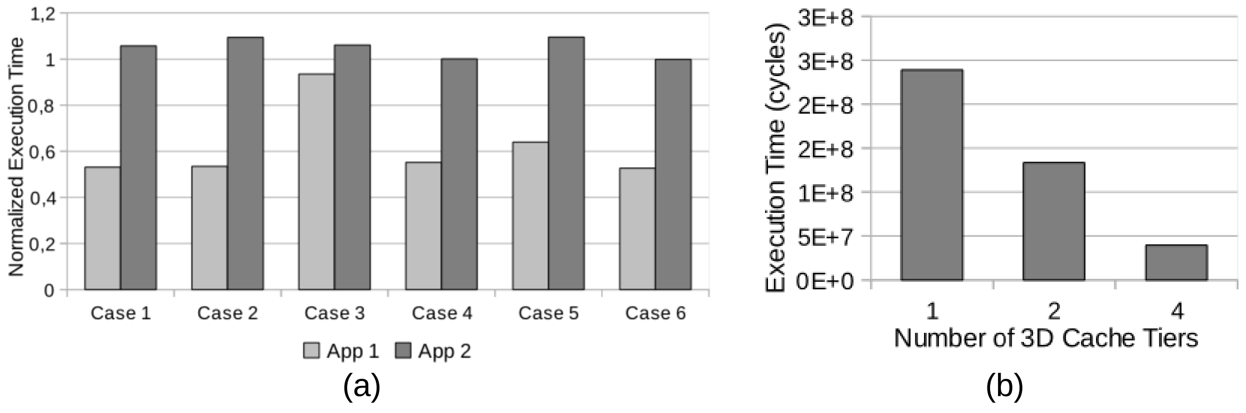


Figure 36. (a) Normalized execution time for all test cases, and (b) execution time for the FFT 2^{20} benchmark [GUT12].

Figure 37 shows that the silicon area overhead due to die stacking (mainly, TSV area) can be as small as 10% of the total die area. The 10 µm TSV diameter parameters is based on the Wide I/O standard.

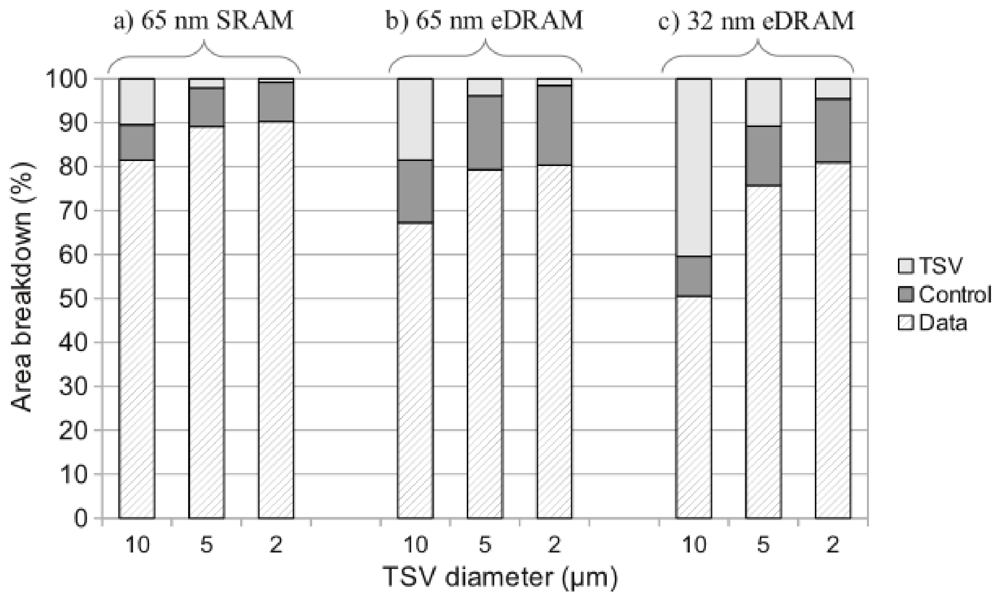


Figure 37. Area breakdown with three TSV diameters [GUT12].

3.2.7 Li et al.: Design and Management of 3D Chip Multiprocessors Using Network-in-Memory

Li et al. [LI06] present a topology design mixing 2D NoC and TDMA bus. The 2D NoC interconnects CPU or caches placed on the same layer. A TDMA bus interconnects the multiple stacks of layers, eliminating the multi-hop penalty inherently present in a packed-based NoC. As the distance between two layers is slight compared to the distance traveled between two NoC routers in 2D, each TDMA bus was implemented as shown in Figure 38. Thus, a single-hop and transaction-less communication is achieved.

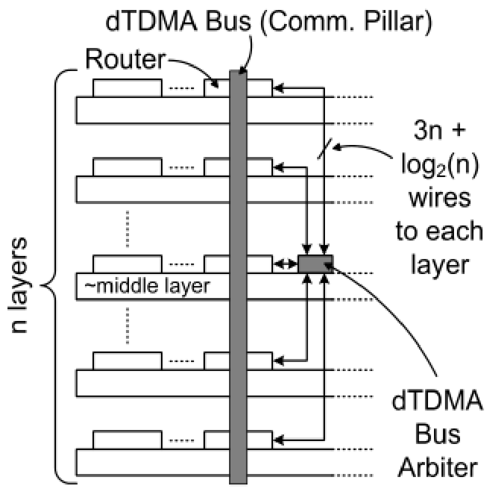
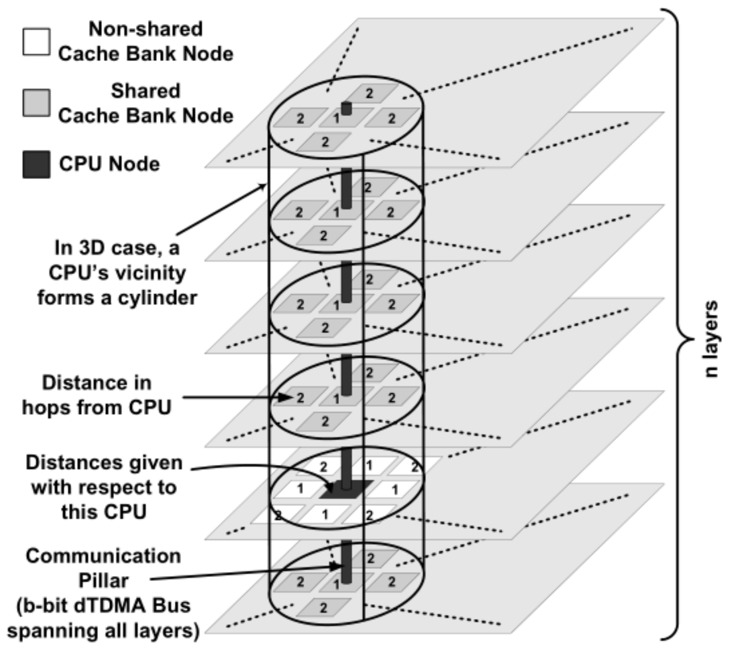
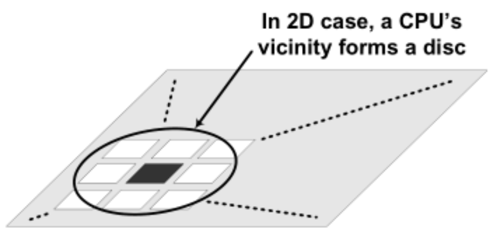


Figure 38. TDMA bus as the communication pillar [LI06].

Li et al. mix caches and processors on the same die allowing the CPU to increase the number of caches accessible under a lower hop count as shown in Figure 39(c). Figure 39(a) and (b) show the 3D and 2D architecture design, respectively.


Non-shared Cache Bank Node
Shared Cache Bank Node
CPU Node
In 3D case, a CPU's vicinity forms a cylinder
Distance in hops from CPU
Distances given with respect to this CPU
Communication Pillar (b-bit dTDMA Bus spanning all layers)
n layers

(a)



(b)

2D Case:

For each CPU:

No. of cache banks within 1 hop:	4
No. of cache banks within 2 hops:	4

3D Case:

Assuming n layers, then for each CPU on a Communication Pillar:

No. of cache banks within 1 hop:
4 + (n – [no. of CPUs on pillar])

No. of cache banks within 2 hops:
4 + 4*(n – [no. of CPUs on pillar])

(c)

Figure 39. (a) 3D architecture design; (b) 2D architecture design and; (c) number of hops for both architectures [LI06].

The processors have private 64KiB L1 caches and share a large 16MiB (256×64KiB) L2 cache. The L2 cache uses an NUCA architecture and can migrate most-accessed lines of a given core to a closer block to him. This technique is called dynamic NUCA. Results conducted using the Simics simulator demonstrate that a 3D L2 memory architecture generates much better results than the conventional 2D design. The average L2 hit latency is summarized in Figure 40 under four different schemes: a 2D architecture with dynamic NUCA; a 2D architecture with static NUCA; a 3D architecture with dynamic NUCA and; a 3D architecture with static NUCA. Static NUCA cannot migrate lines due to the mapping of lines and blocks been defined at project time.

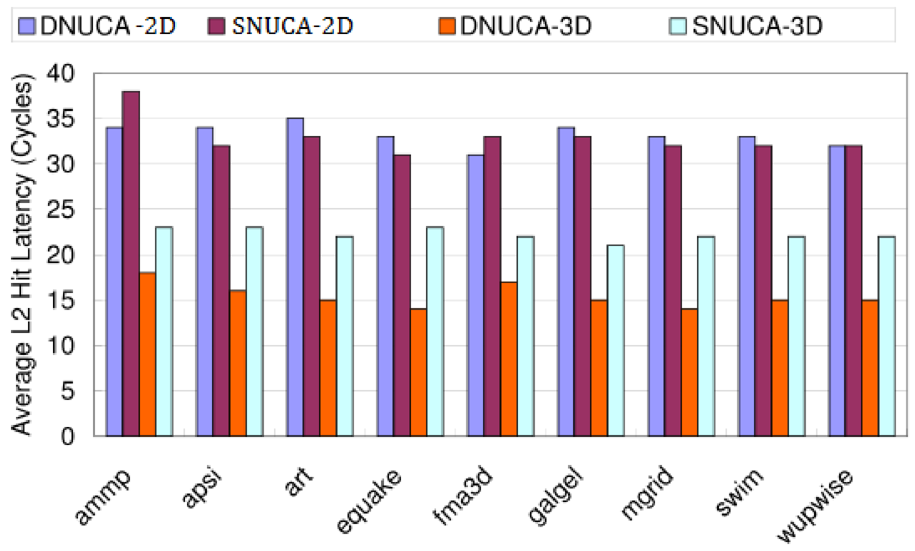
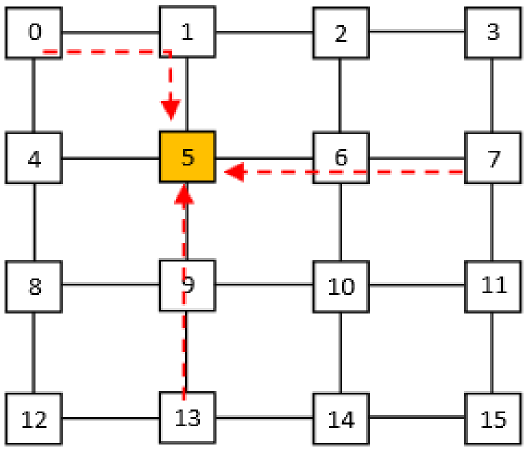


Figure 40. Average L2 hit latency under four architectural schemes (based on [LI06]).

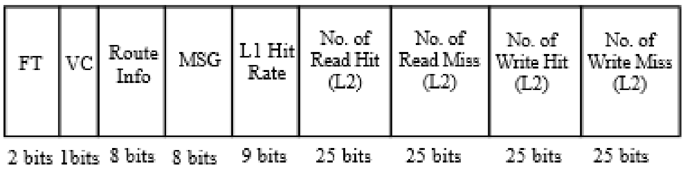
3.2.8 Niknam et al.: Energy Efficient 3D Hybrid Processor-Memory Architecture for the Dark Silicon Age

Niknam et al. [NIK15] tackle the energy consumption in the cache hierarchy by exploiting non-volatile memory in shared distributed LLCs to decrease the leakage power consumption. Their project intends to mitigate the dark silicon phenomenon [ESM11]. The system comprises 16 cores with private L1 caches, the proposed last level cache, and a 3D mesh network interconnects all cores. The LLC structure consists of 16 tiles where each of them includes 1MiB SRAM and 4MiB STT-RAM. As the STT-RAM density is roughly four times of the SRAM, the cache block occupies approximately the same area. In addition, STT-RAM has near zero leakage power.

A central core collects information from all other cores to estimate the system performance. Figure 41(a) depicts the central core for a 4×4 topology, and Figure 41(b) illustrates the information received every 1ms. Note that a 128-bit width is used to achieve this communication in a single flit. With this information, when the number of LLC accesses increases beyond a threshold value, the system workload is estimated to be partially heavy. In this case, one of the tiles using STT-RAM-based LLC is selected to swap for the SRAM-based LLC. In the opposite scenario, i.e., the number of LLC accesses is below another threshold value, one of the tiles using SRAM-based LLC is selected to swap for the STT-RAM-based LLC. The arbitration for tiles is done according to the number of writes access to the LLC. The write parameter is used because it is the slowest access latency in non-volatile memory and, therefore, affects the most the system performance.



(a)



FT	VC	Route Info	MSG	L1 Hit Rate	No. of Read Hit (L2)	No. of Read Miss (L2)	No. of Write Hit (L2)	No. of Write Miss (L2)
2 bits	1bits	8 bits	8 bits	9 bits	25 bits	25 bits	25 bits	25 bits

(b)

Figure 41. (a) Three communications targeting the central core, and (b) the control flit structure [NIK15].

Experimental results were conducted using the Gem5 simulator, McPAT, SystemC-based NoC simulator, and Noxim. The proposed system was compared to a baseline SRAM cache. Figure 42(a) shows the percentage use of SRAM and STT-RAM under different PARSEC benchmarks. Figure 42(b) displays the normalized L2 miss rate of both the baseline and the proposed system. It shows that the miss rate decreases in most cases due to the bigger capacity of the STT-RAM cache. In other instances, this does not happen because of the application characteristics and because swapping between caches results in a burst of misses due to the cold-start effect.

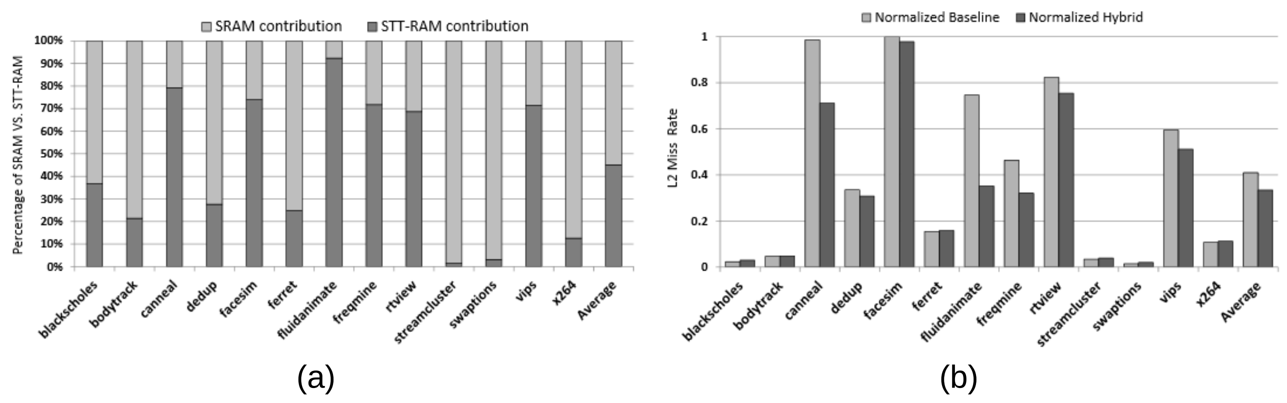


Figure 42. (a) Percentage of SRAM vs STT-RAM use and (b) L2 Miss rate [NIK15].

3.2.9 Schoenberger and Hofmann: Analysis of Asymmetric 3D DRAM Architecture in Combination with L2 Cache Size Reduction

Schoenberger and Hofmann [SCH15a] introduce a 3D DRAM architecture with a latency-optimized layer that partially adopts the cache functionality. Following the locality principle of caches, the DRAM optimizations do not affect significantly on applications with a limited data set. However, given a certain data set and beyond it, DRAM optimization can influence 25% or more of the execution time. Figure 43 depicts the relative runtime overhead of caches and DRAM.

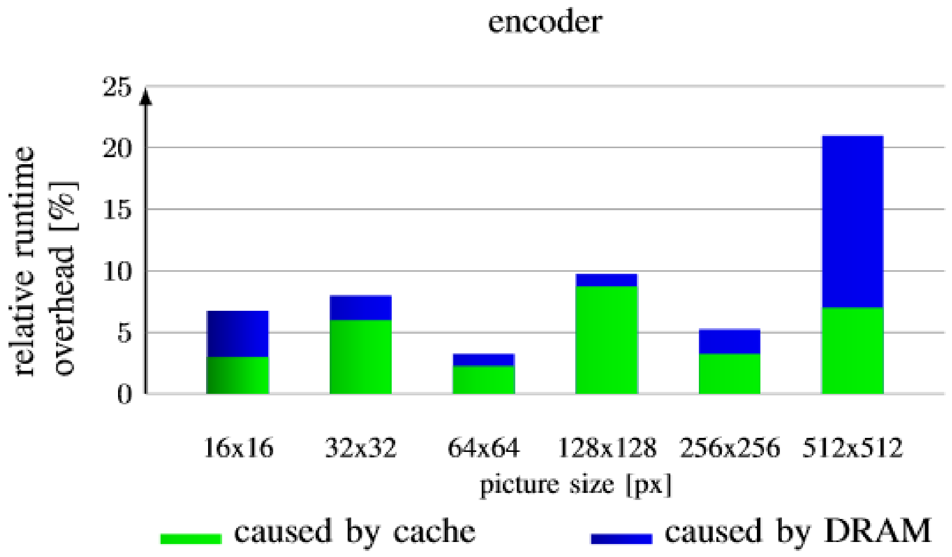


Figure 43. The effects of cache and DRAM in a JPEG2000 encoder [SCH15a].

The authors note that quadruplicating the input data amount leads to an exponential rise of absolute execution values. Because of this behavior, the authors propose a fast DRAM layer that acts as a hybrid between an L3 cache and DRAM. Figure 44 shows this organization in a TSV-connected 3D system.

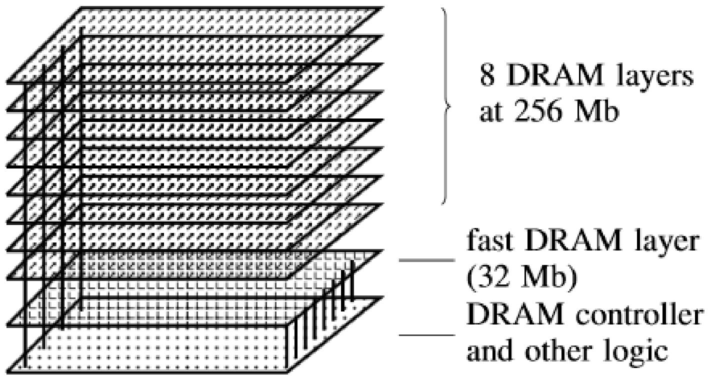


Figure 44. Fast DRAM layer and 8 DRAM layers [SCH15a].

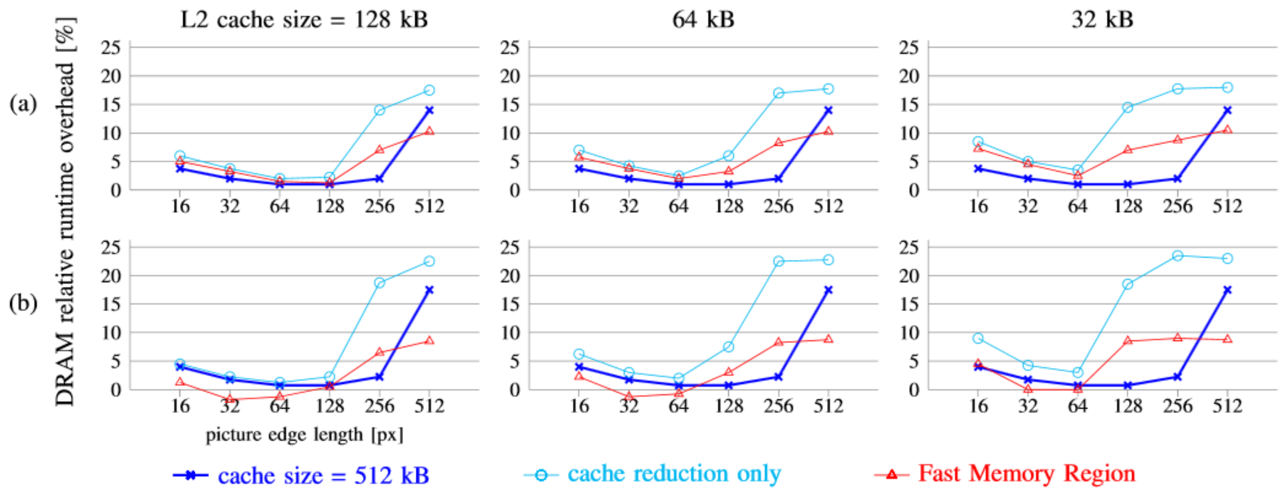


Figure 45. Simulation results of the effect of L2 cache size reduction in combination with fast DRAM layer, (a) encoder (b) decoder [SCH15a].

For simulation results, the authors decreased the timing delays of the fast DRAM layer enabling to investigate the reduction of the L2 cache size. Figure 45(a) and (b) summarize the results obtained in three scenarios of an encoder and decoder processes, respectively: (i) a fixed L2 cache with 512KiB size; (ii) a variable (128KiB – 32KiB) L2 cache size; and (iii) a fast DRAM layer in combination with the same variable L2 cache. All results are about the baseline presented in Figure 43. Hence, the access time decreasing causes the negative values in the decoder process.

The results show that the fast DRAM layer does not compensate the reduction of the L2 cache size. Conversely, this layer can reduce the rise of the DRAM share at runtime significantly. The reduction factor is up to three times for the smallest analyzed cache size and the encoding process.

3.2.10 Wu et al.: Hybrid Cache Architecture with Disparate Memory Technologies

Wu et al. [WU09] describe two types of hybrid cache architectures: (i) inter-cache, where the levels of the cache hierarchy can be made of disparate memory technology, and (ii) intra-cache, where a single level of a cache is divided into multiple segments, each one containing a different memory technology. The latter uses a fast segment for the most accessed addresses and a slow segment for the remaining. This fast segment uses SRAM memory technology because it presents the best latency of the four technologies analyzed. For the slow segment, three types of memory technologies are evaluated: eDRAM (volatile), MRAM (non-volatile) and PCRAM (non-volatile). Figure 46 depicts the normalized IPC and power of the four L2 memory technologies under the same area constraint.

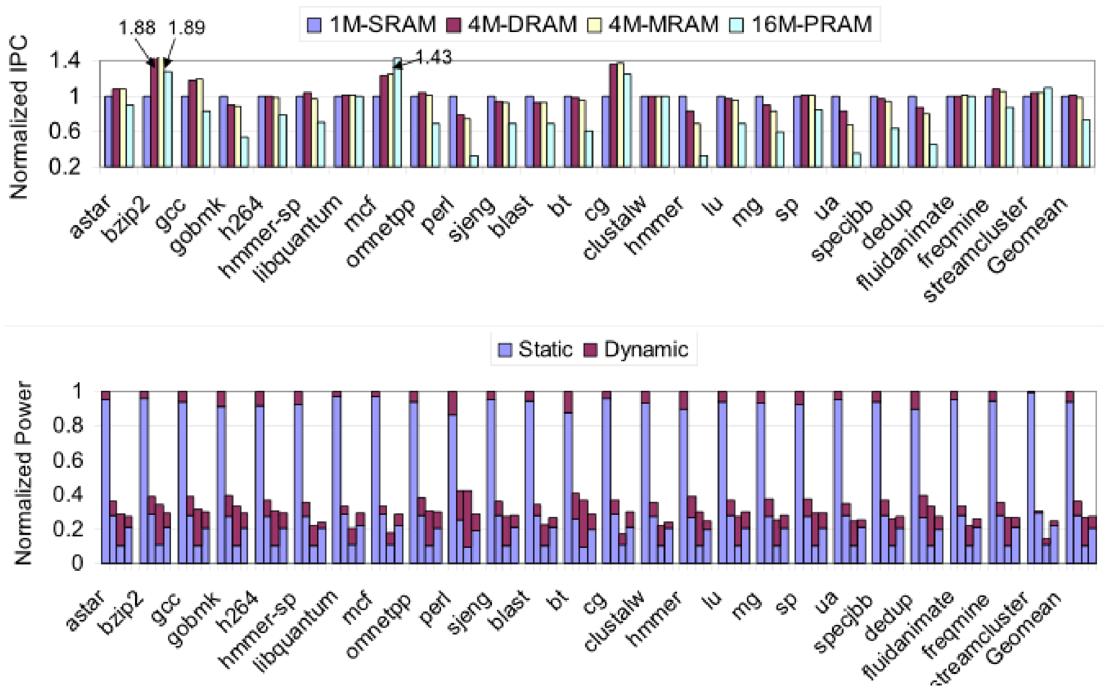


Figure 46. Performance (top) and power (bottom) comparison of the four L2 memory technologies under the same area constraint [WU09].

The authors simulate a myriad of benchmarks, in order to show that an inter-cache hybrid architecture design can provide 7% of IPC improvement over a baseline 3-level SRAM cache, and an intra-cache hybrid design can provide 12% of IPC improvement over the same baseline. Figure 47 shows the normalized IPC of the SRAM-MRAM intra-cache hybrid (RHCA) against the baseline 3-level SRAM cache, best inter-cache hybrid results (LHCA), and a dynamic NUCA (DNUCA). Those results were obtained using Mambo, a PowerPC-based simulator [BOH04].

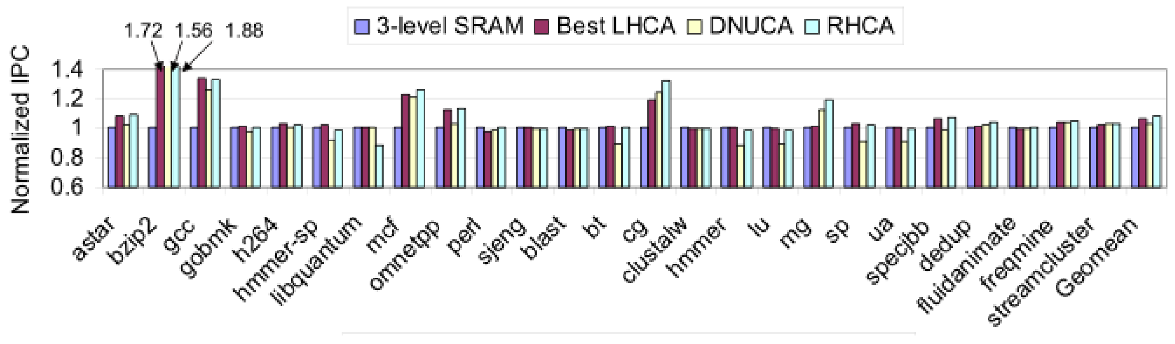


Figure 47. Performance of SRAM-MRAM (herein called RHCA) against baseline SRAM cache, best inter level hybrid cache (herein called LHCA), and dynamic NUCA [WU09].

3.2.11 Summary of Related Work on 3D MPSoCs

Table 5 shows a comparison of the reviewed works in memory organizations of 3D MPSoCs considering six topics. The first topic is the memory technology employed. In this regard, researchers are exploring some types of memories that are still not widely used in the industry. Our summary of commercial chips (Table 1) is comprised entirely of SRAM technology (for cache) and DRAM (for main memory). Niknam et al. [NIK15] and Wu et al. [WU09] explores the use of STT-RAM and MRAM/PRAM, respectively. The second topic is the use of L2 and/or L3 caches. The majority of works employ L2 caches, and a few are exploring L3 caches.

Table 5. Related work summary.

Work	Memory technology	L2/L3 present	Memory layer intended for	Requirements	Traffic	Simulator
[LOI10]	DRAM	No/No	Main memory	latency, area	JEDEC standard	-
[FU14]	DRAM	Yes/No	Main memory	latency, throughput	PARSEC/SPEC2006 benchmarks	Gem5 + In-house
[WOO10]	DRAM	Yes/No	Main Memory + Cache Memory	Throughput, latency	Various benchmarks	SESC
[KIM13a]	SRAM	No/No	Cache memory	latency, throughput	Eight benchmarks	-
[FIC13]	SRAM	No/No	Cache memory	throughput, energy consumption	SPLASH-2 benchmark	Gem5
[GUT12]	SRAM + SRAM/eDRAM	Yes/Yes	Cache memory	scalability	SPLASH-2 benchmark	SoCLib
[LI06]	Not specified	Yes/No	Cache memory + Processor	latency, energy consumption, area	SPEC OMP benchmark	Simics
[NIK15]	SRAM + STT-RAM	Yes/No	Cache Memory	Throughput, energy consumption	PARSEC benchmark	Gem5 + others
[SCH15a]	DRAM	Yes/No	Main Memory + Hybrid Memory	Latency	JPEG2000	3DMemory
[WU09]	SRAM + (eDRAM/MRAM/PRAM)	Yes/Yes	Cache memory	latency, scalability	Various benchmarks	Mambo
This work	**SRAM/DRAM**	**Yes/No**	**Main Memory + Cache Memory**	**Latency, energy consumption, scalability**	**PARSEC + NASA NAS benchmarks**	**Gem5**

As all work summarized here are for 3D integration, the intention of using additional layers is shown in the third topic. Three basic scenarios are explored: (i) layer for cache memory; (ii) layer for main memory and; (iii) layer for processors. Li et al. [LI06] is the only work explored that uses multiple layers for processor distribution, and they do this to decrease the distance of cache memories to the processors. Unfortunately, they did not specify the cache technology employed. Schoenberger and Hofmann [SCH15a] design a specific memory layer that is between a cache and main memory. This memory was named hybrid memory. The other works use cache memories, main memories or the combination of two employing TSV interconnection.

The fourth and fifth topics define the requirements explored for the system and which benchmark suite is used for this, respectively. The majority of works uses either latency or throughput evaluations to estimate the performance of the system. Some of the works also explore the energy consumption in the cache hierarchy. For the benchmark suite, all works use established benchmark suites except Schoenberger and Hofmann [SCH15a] that employ the JPEG2000 encoder/decoder.

The final topic summarizes how the system is evaluated, with most of the work employing simulators. Loi and Benini [LOI10] synthesized its system with Synopsys and Cadence tools and also provided an analytical evaluation of its system. Kim et al. [KIM13a] system was built with GlobalFoundries and Tezzaron IC technologies.

This work explores 3D MPSoCs on the following topics. The cache and main memory employ the conventional SRAM and DRAM technology, respectively. The cache hierarchy is explored up to the L2 cache. For 3D stacking, TSV-enabled emerging main memories and five cache organizations are evaluated regarding performance (specifically the comparison of execution time) and energy consumption. The scalability of our system is analyzed using a packet-based NoC. For the main memory and cache hierarchy, we employ the PARSEC benchmark. For scalability evaluation, we employ the NASA NAS benchmark as it supports the message passing communication model. Wu et al. [WU09] also employed this benchmark for evaluation. Finally, Gem5 is used to provide the execution of all applications and McPAT provide energy analysis of the cache hierarchy.

4 GEM5: FULL SYSTEM SIMULATOR

A full system simulator is a fast architecture simulator capable of executing software stacks from real systems (user and kernel code) without any modification [ENG10]. Such tool can create virtual platform designs that can gather experimental data with workloads compatible with the running software. A key feature of such simulator is the flexibility to explore architectural designs without the inherent hardware cost of doing so manually.

The simulation of computer architectures requires tremendous computational effort since it comprises any number of processors, memories, and I/O devices. Thus, it is necessary low-level descriptions to achieve accurate hardware-level simulation, such as Register Transfer Level (RTL), and a detailed hardware simulation model, which increases the time for design exploration making prohibitive the entire system simulation [BUT12][GUT14]. Therefore, simulators often use models of higher abstraction level that exchange precision for efficiency. Many simulators allow the designer to choose the degree of precision desired. Hence, they can operate in two execution modes [BIN11][IBM07]: *simple (atomic)* and *cycle-based*.

Simple (atomic) mode only captures the program execution without regard to the timing accuracy. Resource contention is normally ignored, and fixed latency is used instead. The operating system can be abstracted and, therefore, the simulator emulates each system call. Memory accesses are assumed synchronous and instantaneous. This model is intended for rapid software development/debugging due to its fast execution time.

Cycle-based mode enables to capture all the functionality of the atomic mode together with the accurate timing information that supports resource contention through arbiters, queues, and interrupts. This mode is intended for architecture exploration and platform design as it gathers information data with a greater level of fidelity.

This chapter describes the Gem5 simulator – its structure, flexibility and known limitations. Afterward, an overview of modern full system simulators and its distinctions is discussed.

4.1 Introduction to Gem5

Gem5 is a full system simulator that employs a flexible and highly modular discrete event model, which is the result of the combined effort of a myriad of industrial institutions and academic such as AMD, ARM, University of Michigan, University of Texas and others. Currently, Gem5 supports six commercial Instruction Set Architectures (ISAs)[3] (i.e., Alpha, ARM, MIPS, POWER, SPARC, and x86) and boots the Linux Kernel on at least three of them (ARM, Alpha, and x86) [BIN11]. Gem5 uses a BSD-like license that allows commercial and academic use and distribution of source code and binary formats [BIN11].

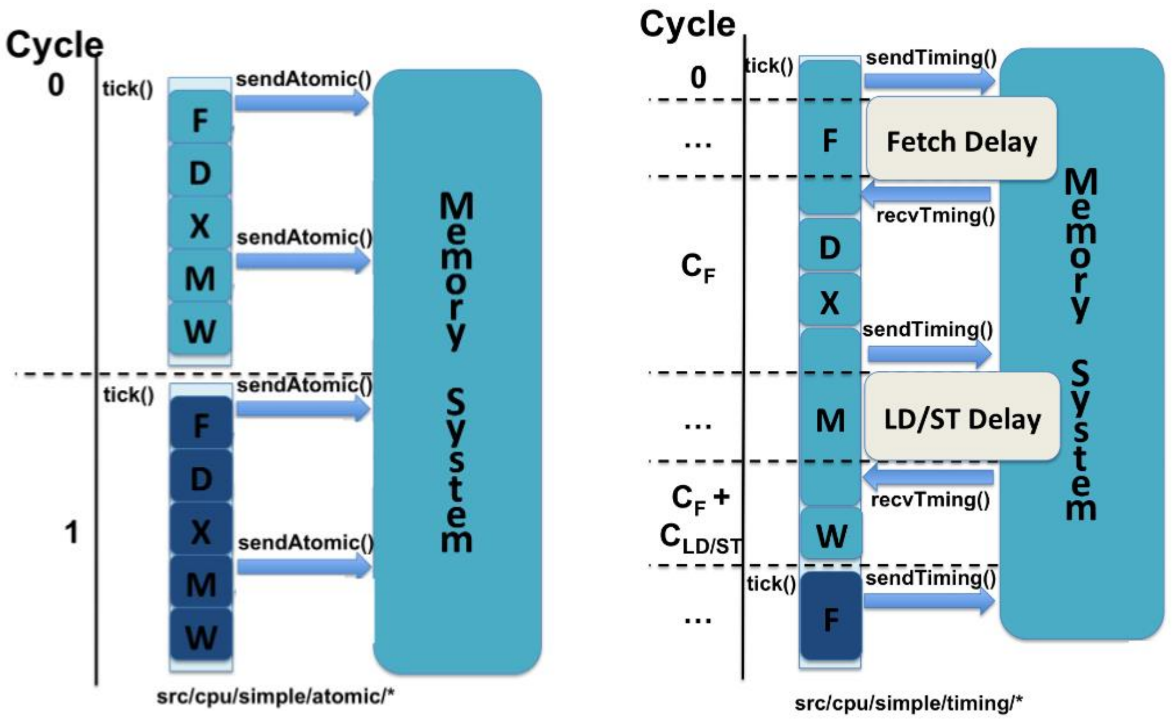
Gem5 aims to be a community tool focused on object-oriented design for architecture modeling [BIN11]. Utilizing standard and message buffer interfaces, Gem5 follows a Transactional Level Modeling (TLM)-like semantic, which enables ample support for community-based changes on the simulator[4].

Gem5 supports *System-call Emulation* (SE) and *Full-System* (FS) modes, which are the simple and cycle-based modes described previously, respectively. The SE mode

[3] The current state of those ISAs is maintained at the Gem5 site [GEM15a].
[4] [GEM15b] is the site where changes are proposed and reviewed by the community.

handles the most commonly used system calls. Whenever the program requests a system call, Gem5 traps and emulates the expected result. In this mode, no effort is made to model devices and other OS services. The FS mode models a bare-metal environment suitable for running an OS. Because of the complexity of FS mode, not all ISAs present in Gem5 are capable of running it. Currently, Alpha, ARM, SPARC, and x86 ISAs are supported [BIN11][GEM15a].

FS mode supports four different CPU models: *AtomicSimple, TimingSimple, In-Order*, and *O3*. *AtomicSimple* and *TimingSimple* are non-pipelined CPU models that conduct the basic cycle of an instruction (fetch, decode, and execute) and commits it on every cycle of execution. The *AtomicSimple* model is a single IPC CPU, which executes all memory accesses instantaneously. The *TimingSimple* model enhances the execution with the timing of memory accesses. Figure 48 shows the differences of execution cycle between these two models. *In-Order* and *O3* are “execute-in-execute” CPU models that emphasize instruction timing and simulation accuracy. “Execute-in-execute” means that instructions are executed only in the pipeline execution stage. While *In-Order* is restricted to execute instructions in the order that they are received, *O3* models the instruction execution according to the order defined by the CPU dispatcher. Both models have parameterizable resources like the number of pipeline stages, load/store queue and reorder buffer (*O3* model only). The challenge of using these two last models is the simulation time required. They are roughly an order-of-magnitude slower than the simpler models [SAI12]. Additionally, not all ISAs support those detailed models [END14].



(a) (b)

Figure 48. Schematic of (a) AtomicSimple and (b) TimingSimple CPU models [SAI12].

Gem5 covers *Classic* and *Ruby* memory modes. The *Classic* mode was inherited from the M5 simulator [BIN06] while *Ruby* was inherited from the Gems framework [MAR05]. The Classic mode is faster, providing ease configuration through python scripts. Currently, this mode maintains memory coherence through an MOESI-like snooping protocol [SUH06]. Changing this protocol requires an overhaul of the entire cache system. In contrast, *Ruby* sacrifices simulation speed providing a flexible infrastructure to simulate a wide variety of memory systems. In particular, *Ruby* provides a specific language, where one can define more easily cache coherence protocols. Moreover, *Ruby* supports many interconnection topologies such as a crossbar, mesh, and point-to-point. Unfortunately, *Ruby* is limited to Alpha and x86 ISAs [GEM15c].

Devices in Gem5 are built on a base class called *io_device*. The device must define three fundamental functions from this class: getAddressRange, read, and write. The getAddressRange function makes the device returns its address range that must be provided for the core simulation engine. The read and write operations are performed in their respective functions so that the device can interact with the remainder of the system. Many devices are already implemented in the Gem5 framework. Examples of such devices are Network Interface Controllers, Hard disk controller, Direct Memory Access (DMA) engines and Universal Asynchronous Receiver/Transmitter (UART) [BIN11][GEM15d].

4.2 The Accuracy of the Gem5 Simulator

Accuracy is one of the fundamental aspects presented in Gem5. This attribute is intended to be balanced concerning the simulation speed desired by the user [BIN11]. Therefore, the user has some control of the accuracy achieved by this simulator. Since academia develops much of the source code, the focus of development is many times to study new concepts instead of replicating existing hardware modules. For instance, Wiener [WIE12] incorporates the ARM CoreLink CCI (*Cache Coherent Interconnect*) into the Gem5 simulator. However, this is not a simple port – while CCI is aimed at single-hop interconnections, the proposed Gem5 counterpart supports multi-hop interconnections with some simplifications into the underlying coherence protocol. One example of this is the snoop hit scenario. When a snoop hit occurs, the memory controller does not handle the request; instead, it destroys the snoop hit immediately. This means a speculative fetch to the primary memory is “magically” avoided [WIE12]. Nonetheless, Gem5 still aims to model state-of-the-art systems accurately. Recently, studies were conducted to evaluate this crucial goal of Gem5.

Butko et al. [BUT12] were one of the first published papers that discuss Gem5 accuracy concerning performance estimation. The authors used the Snowball SKY-S9500-ULP-C01 development kit as the reference hardware model. This development kit comprises a dual-core ARM Cortex-A9 processor. Experimental results showed that the mismatch between the real hardware and the simulation system ranges from 1.39% to 17.94%. The benchmarks employed selected applications of the SPLASH-2 and APLBench suite. The primary reason for the discrepancy encountered in their work is the abstraction used in the model of the external DDR memory latency.

Endo, Couroussé, and Charles [END14] propose an In-Order CPU model for the ARM ISA based on the *O3* model of the same ISA. With both models, the timing accuracy of Gem5 is evaluated with real hardware comparing the execution time of 10 benchmarks of PARSEC 3.0. The Cortex-A9 model (*O3* CPU model) estimates the execution time with an absolute error of only 7.4% (ranging from 1% to 17%), in average. The In-Order model (Cortex-A8) estimates the execution time with an absolute error of 8% (ranging from 2% to 16%), in average. The authors conclude that, even considering the generic nature of Gem5, the magnitude of error encountered can be regarded as good for an architecture simulator.

Gutierrez et al. [GUT14] investigate the source of discrepancies in latency estimation between the Gem5’s ARM ISA and the execution of a real hardware platform (ARM Versatile Express TC2 development board). Only the *O3* CPU model was tested in this work. Using the PARSEC benchmark, it was observed an average of 11% and -12% of runtime deviation, for single and dual-core systems, respectively. The work also shows that when measuring multi-threaded benchmarks, Gem5's scaling is accurate to within 1%, in average. Changes on the Gem5 were proposed and submitted to the Gem5 source code to reduce the overall inaccuracies encountered.

All the works cited here agree that the discrepancies are within acceptable range since Gem5 supports many ISAs and does not have a commercial nature. Nonetheless, it is important to understand these deficiencies and mend them whenever it is possible.

4.3 Simulation Design and Flow

The core of Gem5 simulator is an event-driven engine, which tightly combines C++ and Python programming languages. Every component in the simulation is represented simultaneously as a C++ object and as a Python object [WIE12] to enable simple composition of any system. The designer only needs to recompile the platform if he changes the behavior of some components or if he wants to increase/decrease the level of verbosity/optimization of the simulator. Otherwise, the platform can be modified just by editing Python scripts. Figure 49 shows the initialization process of a "Hello, World" example running in SE mode.

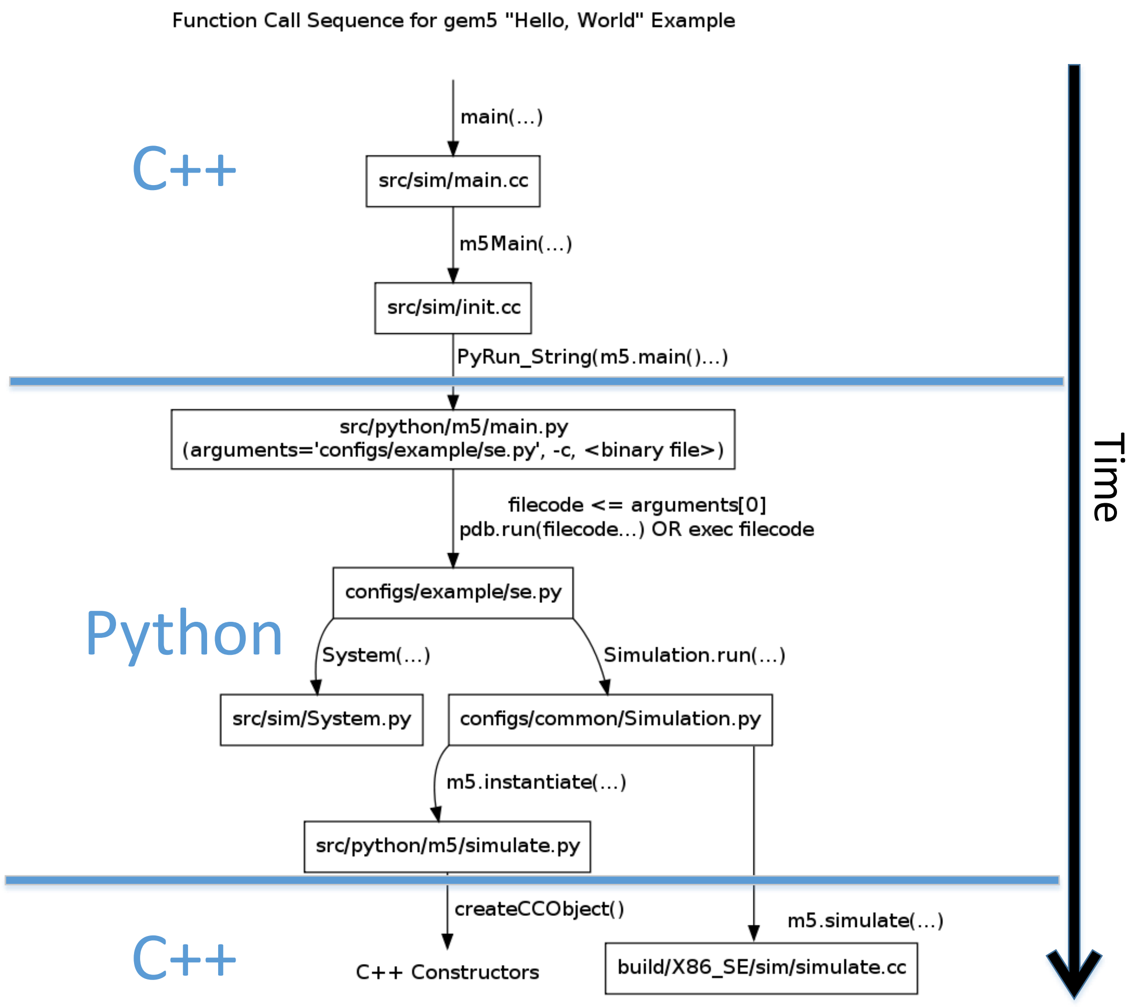

Function Call Sequence for gem5 "Hello, World" Example
C++
main(...)
src/sim/main.cc
m5Main(...)
src/sim/init.cc
PyRun_String(m5.main()...)
src/python/m5/main.py
(arguments='configs/example/se.py', -c, <binary file>)
filecode <= arguments[0]
pdb.run(filecode...) OR exec filecode
Python
configs/example/se.py
System(...)
Simulation.run(...)
src/sim/System.py
configs/common/Simulation.py
m5.instantiate(...)
src/python/m5/simulate.py
C++
createCCObject()
C++ Constructors
m5.simulate(...)
build/X86_SE/sim/simulate.cc
Time

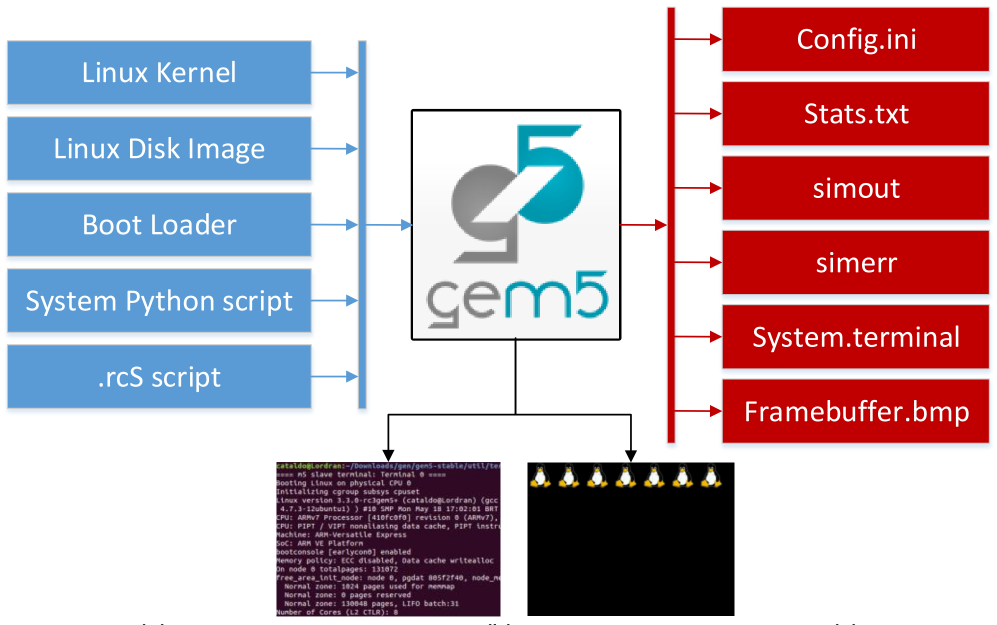
Figure 49. Initialization of Gem5 (based on [GEM15e]).

The first code (C++) is the main function, where the designer can define some variables (e.g., debug flags), invoke the python debugger, enable remote gdb debugging and so on. Then the designer invokes a Python script that describes all objects to be instantiated and the way they are interconnected – in other words, the architecture is defined here; however, he can also define variables in this script, changing the predefined architecture. An example of Gem5 invocation of an ARM platform in FS mode is: *./build/ARM/gem5.opt –debug-flags=Ethernet,IdeDisk configs/example/fs.py –mem-size=512 –caches –num-cpus=4.*

Interaction with the system is twofold: telnet connection and Virtual Network Computing (VNC) session. For the telnet connection, Gem5 provides a specialized program called m5term. For VNC session, the user must use an external application. Telnet is limited to keyboard interactions only, whereas VNC enables additional mouse inputs [WIE12].

Gem5 supports checkpoints allowing the designer to start execution at his desired region of interest, which enables fast-forward large workloads that can take many hours just to initialize. This procedure mitigates some of the slower execution of Gem5 Instruction Set Simulator (ISS) when compared to binary translation-based simulators [BIN11].

Another interesting feature to mitigate the slower execution of detailed CPU models is a program called M5ops. This program enables special instructions on the executing simulation to trigger simulator events. Two functions are especially useful: dumpstats and switchcpu. Dumpstats clears all the simulation statistics until its call, which is useful for cleaning the warming up process of the system. Switchcpu causes the simulation to quit with an event of type “switch cpu”. Then the designer can check for this kind of event and change the CPU model during the system execution. Therefore, the system can execute faster until the region of interest and only changes its model for a slower one. Gebhart et al. [GEB09]demonstrate the compilation and the execution of the PARSEC benchmark using such features on Gem5.



```
cataldo@Lordran:~/Downloads/gem/gem5-stable/util/te
==== m5 slave terminal: Terminal 0 ====
Booting Linux on physical CPU 0
Initializing cgroup subsys cpuset
Linux version 3.3.0-rc3gem5+ (cataldo@Lordran) (gcc
 4.7.3-12ubuntu1) ) #10 SMP Mon May 18 17:02:01 BRT
CPU: ARMv7 Processor [410fc0f0] revision 0 (ARMv7),
CPU: PIPT / VIPT nonaliasing data cache, PIPT instru
Machine: ARM-Versatile Express
SoC: ARM VE Platform
bootconsole [earlycon0] enabled
Memory policy: ECC disabled, Data cache writealloc
On node 0 totalpages: 131072
free_area_init_node: node 0, pgdat 805f2f40, node_me
  Normal zone: 1024 pages used for memmap
  Normal zone: 0 pages reserved
  Normal zone: 130048 pages, LIFO batch:31
Number of Cores (L2 CTLR): 8
```

Figure 50. Gem5 simulation (a) inputs, (b) runtime interfaces and (c) outputs (based on [WIE12]).

Gem5 generates a couple of files after the end of the simulation. Figure 50 depicts simulation input, runtime interfaces, and output. Simulation is ended either by the user or by choosing a maxtick parameter. In FS mode, the following outputs are produced:

Simout and simerr - The standard output and error stream generated by the simulated OS.

System.terminal - The output of the simulated system's terminal.

Framebuffer.bmp - The latest contents of the simulated system's display.

Config.ini - A key output of the Gem5 simulator. This file shows all components instantiated, its interconnections and its respective parameters. This allows the validation of the system simulated with the one intended by the user.

Stats.txt - The second key output of the Gem5 simulator. This file aggregates all statistics generated by every component in the system. The overall extent of this file is limited to the implementation of the system's components. Fortunately, the components already implemented in Gem5 have a good amount of statistics gathering.

4.4 Overview of Full System Simulators

This section presents an overview of some relevant full system simulators that are employed in the design space exploration of MPSoC platforms. The criterion for choosing these simulators was established in accordance with published works in areas related to design exploration of MPSoC and/or cache.

4.4.1 SoCLib

SoCLib [SOC15a] is an open platform for virtual prototyping of MPSoCs described in SystemC language, providing high-level of abstraction while maintaining accurate transaction-level results. In this platform, processors are described using ISS. Currently, SoCLib is maintained at Lip6 laboratory in France. SoCLib is licensed under GNU's Not Unix (*GNU*) General Purpose License (*GPL*) version 2.

Hardware is implemented using one of two types: TLM with Distributed Time (TLM-DT) or Cycle Accurate Bit Accurate (CABA). TLM-DT is a model compliant to TLM2.0 Open SystemC Initiative (OSCI) standard [SOC15b]. However, TLM-DT and TLM2.0 differ in the timing representation because an absolute time is used instead of the global simulation time provided by the SystemC core. Besides, all messages are annotated with timing information, since synchronization between timed processes is no longer centralized [TEC15a]. CABA aims to model hardware at the cycle accurate level. As stated in [SOC15c]: "*The idea is to force the 'event driven' SystemC simulation engine to run as a cycle-based simulator.*" To force this behavior, a hardware description is taken and converted to three types of function: (i) **Transition function**, which is responsible for the computation of the next value of registers. It takes as input the current values of the registers and input signals; (ii) **Moore generation function**, which is responsible for the computation of output signals that only depend on the internal registers; and (iii) **Mealy generation function**, which is responsible for the computation of output signals that depend on the internal registers and the values of the input signals.

Figure 51 shows an example of a simple hardware and its representation using a graph, which is the input to the CABA conversion algorithm that produces the description of Figure 52. Fraboulet, Risset, and Scherrer [FRA04] define this algorithm and an extended version to avoid unnecessary duplicated code. As expected, this higher-level of detail compared to TLM-DT results in a significant increase of simulation time – CABA is approximately ten times slower than TLM-DT [POU09].

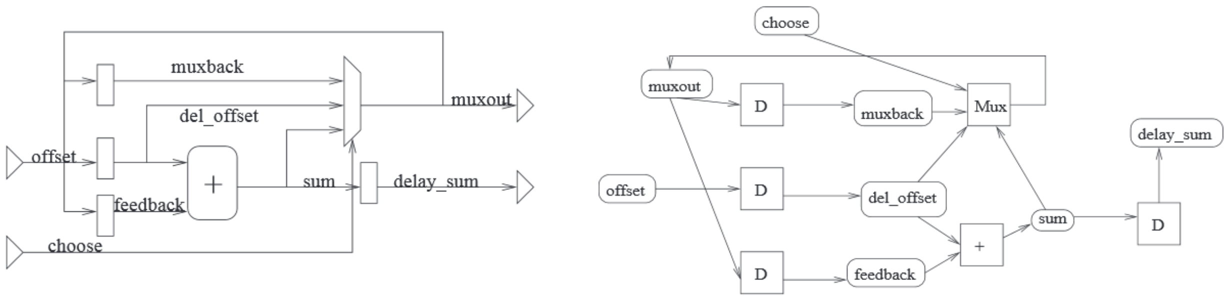


Figure 51. Example of a simple hardware (left) and its representation as a graph (right) [FRA04].

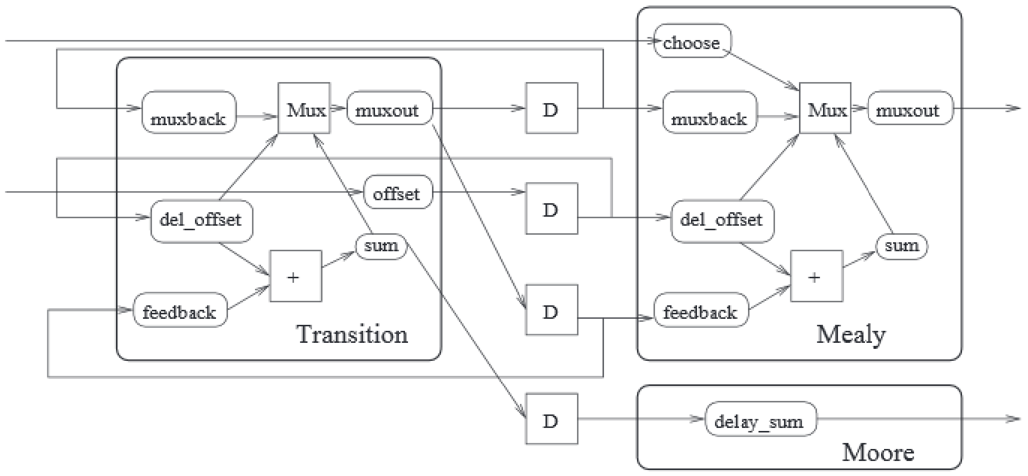


Figure 52. CABA representation of the previous hardware description in Figure 51 (this is not the final optimized version) [FRA04].

The most recent SoCLib version supports four OS's [SOC15a]. We chose to test MutekH OS, since it is maintained by the same laboratory as SoCLib, and we found out many errors in SoCLib during simulation. Observing the repository activity, we can hypothesize that although SoCLib continues to be updated, the platform used for MutekH did not receive the same maintenance. There are a two-year gap and approximately 300 commits between updates in those two cases[5].

4.4.2 RABBITS

Rabbits is a system simulator that relies on QEMU for software execution and SystemC for hardware modeling [RAB15]. Architecture support is limited to the ARM family. QEMU is employed for its binary translation technique, as it improves the simulation time required for hardware/software evaluation [GLI09]. Figure 53 shows an example of a platform in this simulator, whereas the processing units are executed in the QEMU framework and its communication with the outside system is wrapped in SystemC. Rabbits is licensed under GNU GPL version 3.

[5] Revision 2624 (SoCLib) and Revision 2325 (MutekH platform examples).

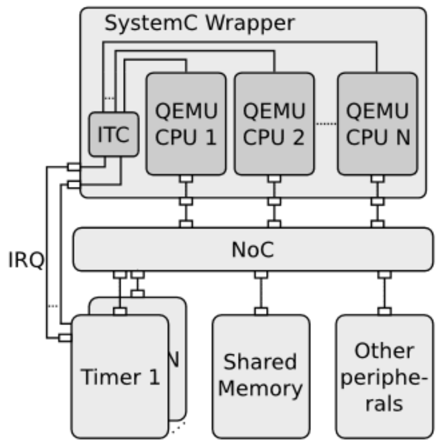


Figure 53. Example of a simulation platform in Rabbits [GLI09].

QEMU lacks any implementation of cache models [GLI09][MAH13]. Recently, there have been extensions for cache support in QEMU – Mahmoodi et al. [MAH13] proposed an extension for the MIPS architecture and Dung, Taniguchi, and Tomiyama [DUN14] for the ARM architecture. However, Rabbits was designed earlier than this and rolls out its implementation of cache in SystemC.

The major drawback of Rabbits is its lack of proper documentation. This is even alluded at its main page [RAB15]. Rabbits offers seven platforms in its repository. However, without documentation, it is rather difficult to determine their characteristics. Because of this limitation, Rabbits was rejected for this work.

4.4.3 SIMICS

Simics is a full system simulator intended to produce an executable specification for hardware designers and a fast virtual platform for software developers [WIN10] that uses ISS for processor execution [MAG02]. Contrasting to the previous simulators described in this section, Simics is a commercial product and has a closed license [GUT14], and accesses to its source code is handled on a case-by-case basis [WIN15a].

Hardware designers write TLM-like code using a proprietary language called Simics DML. This language uses a C-like programming syntax. The hardware is not analyzed to determine its delay; instead, the designer issues latency at a later stage [WIN10]. Figure 54 shows an example of a hardware interface with interrupt support.

```
bank regs {
    parameter register_size = 4;
    register version       @ 0x00 "Device version register";
    register control       @ 0x04 "Device control register";
    register status        @ 0x08 "Device status register";
    register reset         @ 0x0c is (write_only) "Reset register (write only)";
    register irq_num       @ 0x10 is (read_only) "IRQ assigned to device";
    register rule_set      @ 0x14 "Rule set (bit encoded)";
    register line_length   @ 0x18 "Line length (in bits)";
    register start_compute @ 0x1c "Start computation";
    register input[32]     @ 0x20 + 4*$i is (write_only) "Input buffer";
    register output[32]    @ 0xa0 + 4*$i is (read_only)  "Output buffer";
}
```

Figure 54. Example of a DML device programming interface [WIN10].

For software developers, Simics offers a customized version of the widely-used Eclipse Integrated Development Environment (IDE). Also, virtual platforms are executed using the following OS's: vanilla Linux, Wind River Linux, VxWorks, and others [WIN15b].

Due to the lack of a cache and a memory latency models [WIN10], we decided to work on a different full system simulator.

4.4.4 OVP

Open Virtual Platforms (OVP) is a system simulator that uses dynamic binary translation to cope with system design complexity while still maintaining high simulation speed [REK13]. OVP has a dual license model [IMP13]: (i) its main simulation core and some processor models are proprietary, (ii) while other processor models and Application Programming Interfaces (APIs) are open sourced via a modified Apache 2.0 license. OVP comprises three crucial components:

- **OVPsim** is the simulation engine responsible for translating the target architecture binary code to x86 host instructions. It can be wrapped and called from other simulators environments like, for instance, SystemC [IMP15a]. While OVPsim is freely available (for non-commercial usage), CpuManager is the commercial alternative provided by Imperas. Some functionalities are exclusive to the commercial tool [IMP14].
- **Library of processor models** contains open-source and pre-compiled models of processing units. These models support various I/O components (e.g., UART and DMA), memory systems and OS's (Linux, Android, and µcLinux are supported) [IMP15b].
- **OVP APIs** are four interfaces for the C language. These interfaces are responsible for instantiation of full systems, creating new processing unit models and creating new peripheral models [IMP15c].

OVP is a software virtual platform that does not model hardware in a latency-aware manner [IMP11]. Consequently, it is designated as an instruction accurate simulator. Therefore, for a given processing unit, OVP guarantees that registers hold the correct values at the end of each instruction. However, there is no concern for pipeline progression, out-of-order execution or delays in the memory system [AGR09].

Some of the processor models support L1 cache [IMP15d]. However, the designer cannot customize the cache behavior. For this, the designer must roll out his cache model implementation at the cost of simulation performance, as OVP is unable to optimize this scenario [IMP15d]. One of the creators of OVP, James Kenney, states that: "(…) *in terms of performance, on my 3Ghz PC, I expect to see several hundred MIPS simulation speeds for simulations without caches* (…) *and 10-20 MIPS when I have full MMCs, although this is of course highly dependent on the complexity of the MMC model* (…)" [IMP15e].

A challenge of using OVP for cache evaluation is its lack of proper latency model since it assumes a 'perfect' memory model – where there is no latency penalty [IMP15f][IMP15g]. Therefore, OVP was rejected for this work since it is intended for software virtual platform, whereas this work aims to explore software/hardware virtual platforms. Nevertheless, restructuring OVP is undesirable since there are alternative simulators better suited for this scenario.

4.4.5 Taxonomy of Full System Simulators

Table 6 depicts the key characteristics of the full system simulators discussed in this work. The items below describe the characteristics analyzed here.

ISA(s) supported by the simulator. This does not differentiate if the ISA is limited to atomic mode only or supports both modes of execution (atomic and cycle-based).

Processor emulation technique employed by the simulator. Binary translation is faster than ISS; however, it requires a complex implementation code [BIN11].

Primary **License** of the simulator. Note that some parts of the simulator (primarily external programs) may employ a separate license.

Accuracy employed by the simulator. Functional accuracy is limited to the behavior of the system while cycle-accurate aims to detail the timing behavior as well.

Operating systems supported by the simulator. They can be restricted to a limited number of the overall ISA(s) held by the simulator. Again, this is not differentiated in this table.

The presence of a **cache model** presumes the support for storage, coherence protocol, and timing behavior.

Table 6. Taxonomy of full system simulators.

Simulator	ISA(s) supported	Processor emulation	License	Accuracy	Operating systems	Cache model
SoCLib	SPARC, Nios II, POWER, MIPS, ARM, and others	ISS	GNU GPL v2	Cycle-accurate	DNA/OS, MutekH, NetBSD and others	Yes
Rabbits	ARM	Binary translation	GNU GPL v3	Cycle-accurate	Linux and DNA/OS	No
Simics	x86, ARM, M68k, MIPS, POWER, SPARC, Alpha	ISS	Closed	Functional	Linux, NetBSD, Solaris, Windows, and others	No
OVP	ARM, MIPS, x86	Binary translation	Dual license	Functional	Android, Linux, and others	No
Gem5	POWER, ARM, MIPS, Alpha, SPARC, x86	ISS	BSD	Cycle-accurate	Android, FreeBSD, Linux, Solaris	Yes

5 DESIGN AND EXPLORATION OF 3D MPSOC ARCHITECTURE

This chapter explores 3D MPSoCs with on-chip cache support targeting some specific architectures, in order to evaluate tradeoffs in energy consumption and latency minimization of the cache hierarchy. The experimental analysis were conducted using the Gem5 full system simulator.

Energy saving has become one of the most important design challenges as technology has advanced [RET11][TRA10]. Several previous works [BEN13][GOR07] have shown that more than 50% of energy consumption in processor-based architectures can be consumed by the cache subsystem. Aiming to reduce energy consumption while maintaining acceptable performance requires careful design.

At some level of abstraction, we roughly define the energy consumption of a cache as a composition of two parcels, which are the *Standby* energy and the *Access* energy. Figure 55 exemplifies these parcels of energy consumption, taking into account 45 ns of memory operation and two memory accesses.

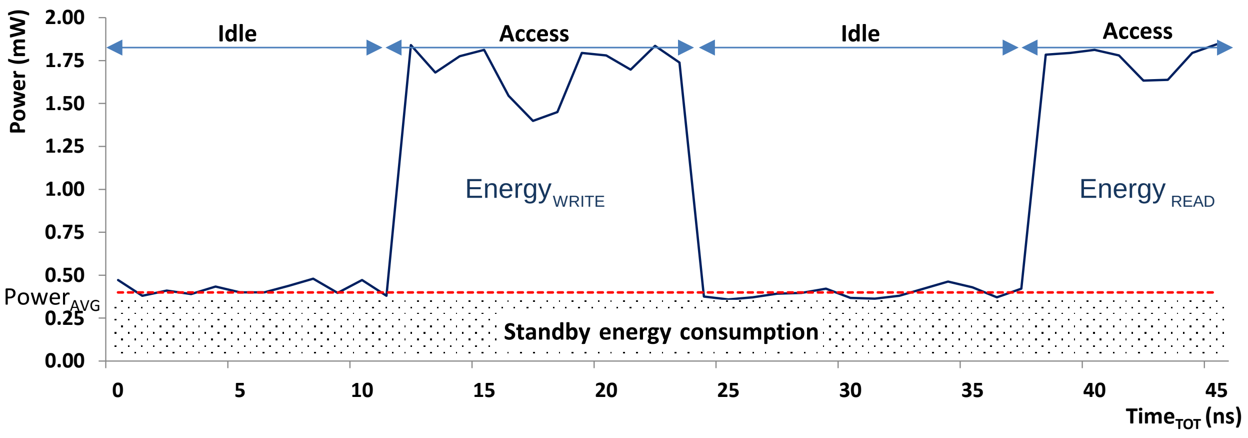


Figure 55. Exemplification of the energy consumption of a memory cache.

The *Standby* energy is the average energy consumed by a cache without any information access (i.e., reading or writing accesses), which can be modeled with the average power dissipation ($Power_{AVG}$) and the total operation time ($time_{TOT}$). Additionally, $Power_{AVG}$ is composed by the static power dissipated in subthreshold and gate leakages, and the average dynamic power dissipated during cache maintenance operations; i.e., operations that are transparent to any application execution (e.g., memory refresh).

The *Access* energy is composed by all additional energy consumed during a read ($Energy_{read}$) or write ($Energy_{write}$) operation, which takes into account the quantity of reads (n_{read}) and writes (n_{write}). Both, $Energy_{read}$ and $Energy_{write}$ are average values.

Composing all parcels, the total energy consumed by a given cache level may be modeled by Equation 1.

$$Energy_{Total} = n_{read} \times Energy_{read} + n_{write} \times Energy_{write} + Time_{TOT} \times Power_{AVG} \qquad (1)$$

It is important to emphasize that our model regards the energy consumption produced by the application execution. Consequently, we employed the term Access energy, instead of dynamic energy, because some memories consume lots of dynamic energy, even without accesses required by the application (e.g., dynamic RAM). Any memory refresh is abstracted by the application execution, and thus, the energy consumed in a given memory refresh is computed as a Standby energy consumption.

Latency minimization is a common requirement in applications targeting on-chip architectures [BJE06][SAB10]. Unfortunately, such minimization can induce an increase in other requirements such as area and energy consumption minimization [BEN13]. Therefore, a multi-constraint oriented approach is recommended.

Each one of these requirements is strongly related to the principles of locality, which justify the use of caches [PAT13]. These principles are (i) **temporal locality** - If a particular memory location is referenced, then it is likely that the same place will be referenced again in the near future memory access; and (ii) **spatial locality** - If a particular memory location is referenced, also adjacent memory locations tend to be referenced. These principles can be easily associated with two basic elements of programming languages: loop statements and sequential execution [LUT13]. Therefore, due to the principle of locality as new levels of caches are added, the following statements are made:

1. The importance of saving access and standby energies are inversely and directly proportional to the cache level, respectively.
2. The importance of latency reduction is inversely proportional to the cache level.

The requirements discussed here are outlined in Table 7 considering three levels of cache. The qualitative importance applied to each row is about the cache subsystem only.

Table 7. Qualitative comparison of the importance of some memory requirements for the evaluated cache level. The colors indicate the importance of each level against the specified criteria. Blue, yellow and red means low, intermediate and high importance, respectively.

Cache level	Access energy saving	Standby energy saving	Latency reduction
L1	High	Low	High
L2	Intermediate	Intermediate	Intermediate
L3	Low	High	Low

This chapter is organized as follows. Next section discusses the Versatile Express architecture that is the baseline for this work. Then, the architecture exploration that will be conducted over the baseline architecture. During this discussion, we also detail the limitations found in the Gem5 framework to work with MPSoCs. Finally, we present and discuss the parallel workloads employed in this work – PARSEC and NASA NAS benchmarks.

5.1 Architecture Baseline

The starting point of our architectural exploration is the ARM system modeled by the Gem5 simulator. The ARM ISA was chosen for three reasons: (i) it is one of the ISAs supported by the Gem5 simulator; (ii) it is widely employed in embedded systems such as mobile cellphone, automobile vehicles, and developments kits [ARM15e][NVI15][SAM15a]; (iii) ARM is committed to helping the development of the Gem5 simulator and provides tool support for analyzing the ARM system behavior [ARM15f].

The ARM system of Gem5 is based on the ARM Versatile Express development board [GUT14]. Table 8 details the most relevant characteristics of the processor, cache, and memory subsystems, respectively.

Table 8. Characterization of the Versatile Express development board (based on [ARM11a][ARM12a][ARM12b][ARM15a]).

Processor subsystem	Big.LITTLE architecture	
	2× Cortex-A15	3× Cortex-A7
Core Type	Out-of-Order	In-Order
Speed	1Ghz	800Mhz
Pipeline	15 stages (integer)	8 stages (integer)
Extensions	VFP & NEON	VFP & NEON
Cache Subsystem	Cortex-A15	Cortex-A7
L1 cache	Private L1 cache (32KiB instruction and 32KiB data)	
L1 I/D associativity	2-way	
L1 I-cache block size	64 Bytes	32 Bytes
L1 D-cache block size	64 Bytes	
L1 I/D replacement policy	LRU (*Least Recently Used*)	Pseudo random
L1 I-cache addressing	PIPT	VIPT
L1 D-cache addressing	PIPT	
L1 coherence protocol	MESI	MOESI
L2 cache	Shared L2 cache (1MiB)	Shared L2 Cache (512KiB)
L2 associativity	16-way	8-way
L2 block size	64 Bytes	
L2 replacement policy	Pseudo random	
L2 addressing	PIPT	
L2 coherence protocol	MOESI	
Interconnection	Internal CoreLink CCI-400	
Memory Subsystem		
DRAM type	DDR2 x32	
DRAM frequency	400MHz	
Memory size	2GB	
Memory Interfaces	1	
System bus frequency	500Mhz	
Components		
NOR Flash, UART, SD Card Controller, 10/100 Ethernet, HDLCD and others		

This board uses the big.LITTLE architecture, which means that it has a high-performance cluster of O3 processing units and a low-power cluster of in-order processing units. Both have the same extensions: ARM Vector Floating-Point (VFP) and a general purpose engine called NEON for accelerating multimedia and signal processing algorithms [ARM15b]. On the cache subsystem, 2-way set associative is used for L1 caches and 16-way, and 8-way are used for L2 caches of the high-performance and low-power clusters, respectively. Only the L1 instruction cache of the Cortex-A7 core uses Virtually Index Physically Tagged (VIPT), while all the other caches use Physically Index Physically Tagged (PIPT). For the high-performance cluster, the L1 cache employs the MESI coherence protocol, while the L2 cache employs the MOESI protocol. A dedicated hardware called *Snoop Control Unit* (SCU) coordinates the translation between the two protocols [ARM11a]. For the low-performance cluster, all caches employ the MOESI protocol. Finally, the memory subsystem is comprised of a 2GB 32-bit DDR2 clocked at 400 MHz.

Table 9 shows how Gem5 models the architecture of ARM Versatile Express. Internally, this architecture is called *VExpress_EMM*. Note that this table is limited to the default parameters and to the specific version of Gem5[6]. Basically, all parameters shown in Table 9 can be changed using Python scripts – the exceptions are: adding/removing pipeline

[6] In this work, Gem5's version is identified by the following id: 5fe05690d03d (Mercurial repository identification). Revision 10923 from the Gem5 stable repository. Date of commit: August/2015.

stages, addressing policy employed in caches, coherence protocols employed in caches and the CCI. For these, Gem5's capabilities must be expanded.

Table 9. Default parameters for the modelling of ARM Versatile Express on Gem5 (based on [END14][GEM15f][GUT14][SAI12]).

Processor subsystem	Cortex-A9 based (ARMv7-A profile)
Core Type	Out-of-Order
Speed	500Mhz
Pipeline	7 stages (integer)
Extensions	VFP & NEON
Cache Subsystem[7]	Cortex-A9 based
L1 cache	Private L1 cache (32KiB instruction and 32KiB data)
L1 I/D associativity	2-way
L1 I-cache block size	64 Bytes
L1 D-cache block size	64 Bytes
L1 I/D replacement policy	LRU (*Least Recently Used*)
L1 I-cache addressing	PIPT
L1 D-cache addressing	PIPT
L1 coherence protocol	MOESI
L2 cache	Shared L2 cache (1MiB)
L2 associativity	16-way
L2 block size	64 Bytes
L2 replacement policy	Pseudo random
L2 addressing	PIPT
L2 coherence protocol	MOESI
Interconnection	Internal CoreLink CCI-400 based
Memory Subsystem	
DRAM type	DDR3 x64
DRAM frequency	800MHz
Memory size	2GB
Memory Interfaces	1
System bus frequency	1Ghz
Components	
	Hard Disk, UART, 10/100 Ethernet, HDLCD and others

In comparison to the Versatile Express board, there are discrepancies in all subsystems analyzed. In the processor subsystem, the speeds of cores are lower. However, this can easily be changed. In the cache subsystem, all caches use PIPT for addressing, as does Cortex-A15 (Cortex-A9 uses VIPT and PIPT for L1 and L2 cache addressing, respectively [ARM10]). Again, all caches use MOESI for cache coherence protocol. Our hypothesis is that it happens because there is no SCU implementation on Gem5. Therefore, there is no hardware to coordinate between two coherence protocols.

An expanded CoreLink CCI structure that supports multi-hops networks performs the interconnection architecture. The default option of the memory subsystem is the 64-bit DDR3. Nevertheless, there is a model for 32-bit DDR2 that resembles the Versatile Express main memory. Finally, some components of the system are missing (SD Card Controller) and others have a broader range of operation (Hard Disk Controller).

Figure 56 illustrates the integer pipeline stages of both models Cortex-A7 and Gem5 O3 core. Gem5 model is based on the Alpha 21264 pipeline [GEM15f], which is customizable, whereas the user can define the width of each stage depicted in this figure.

[7] Default parameters for the Classic memory system.

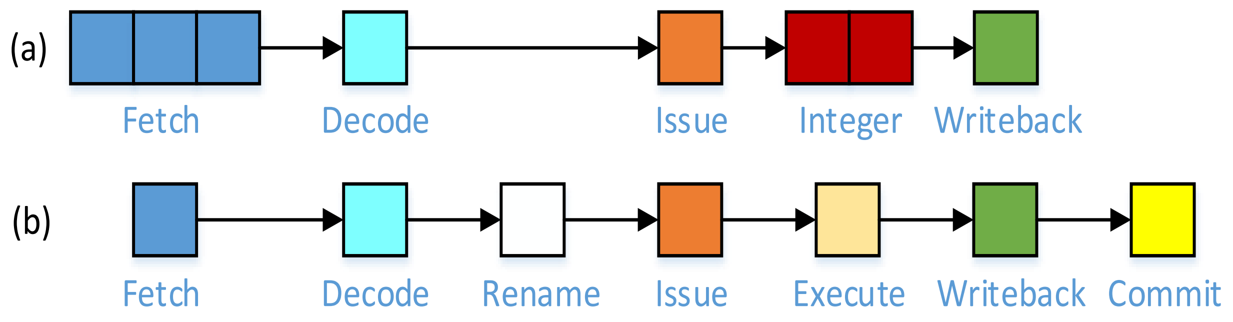


Figure 56. Pipeline stages of (a) Cortex-A7 and (b) Gem5 O3 CPU model (based on [ARM11b][GEM15f]).

5.2 Architectural Exploration

The main contribution of this work is the exploration of a diverse set of MPSoC architectures based on the ARM ISA. These designs aim to balance scalability, throughput, latency and energy consumption. The basis of this study is twofold: to separate the memory from the communication systems and to use 3D IC to tackle the constraints described above.

We propose to disconnect the memory and communication system since they have different requirements. When a packet-based communication system is expanded to provide access to the memory system, it can be easily overburdened and unable to sustain an acceptable performance [FU14][WA08][YE10]. Thus, the use of two independent system is attractive due to its scalability.

The interprocessor communication system uses a packet-based NoC, which is capable of providing an efficient on-chip communication when compared to traditional solutions as shared bus [BEN02]. Physically, distributing router units reduces the wire delays and the capacitance of the interconnection. Architecturally, decentralizing the interconnect fabric enables reliable systems building through independent operations.

Specifically, we employ the hierarchical NoC proposed by Matos et al. [MAT11]. This architecture manages intra-cluster communication by a low-latency circuit-switched crossbar and manages inter-cluster communication by a high-bandwidth packet-based NoC. By doing this, the intra-cluster communication avoids the inherent multi-hop penalty of packet-based NoC and additional buffers that consume significant power. Figure 57(a) depicts the communication system connecting eight cores.

The Gem5's CCI-like structure, which is heavily based on the CoreLink CCI provided by ARM, grants the resources for memory system interconnection. The CCI is a low-cost and low-power crossbar-based communication architecture that provides support for cache coherence protocols enabling to connect caches, I/O devices, and GPUs. All read and write data channels are fixed 64-bit width. Moreover, CCI supports more than one memory controller, which allows parallelism for accessing the main memory [ARM12c][ARM15g]. Figure 57(b) depicts the Gem5's CCI connecting the eight cores into a single coherent memory space. The General Interrupt Controller (GIC) is responsible for distributing interrupts for all cores. Since there is no core distinction for accessing the main memory this denote a UMA system. Currently, ARM does not employ NUMA in its interconnect family products [ENT15][LIN15].

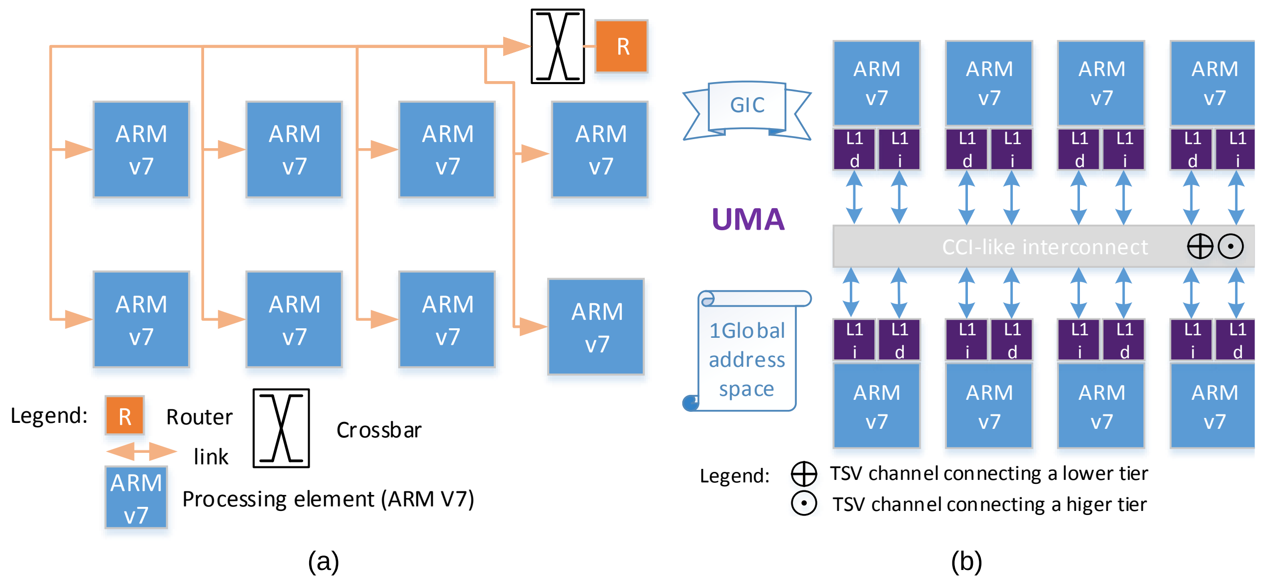


Figure 57. Schematic representation of (a) on-chip communication system performed with a crossbar connected to a NoC through a router port; (b) the same processors with L1 caches of data and code and the Gem5's CCI memory interconnection.

Figure 58 shows the entire target architecture, which comprises the junction of both message communication and memory architectures. The CCI is located on a second tier presented in Figure 60.

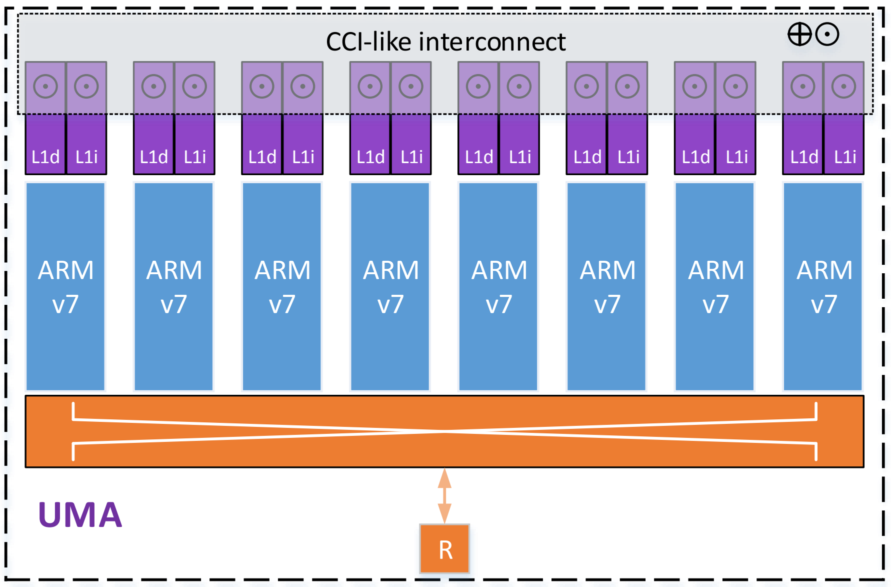


Figure 58. A cluster of eight processors with message communication and memory architectures.

To scale this architecture to hundreds of cores, we propose to use more than one global address space, as maintaining coherence in an UMA architecture model is costly and impractical. This limitation is known as the Coherence Wall [HUA12][MAT10]. Therefore, we propose to use a hierarchy of tiers, with different models for each hierarchy level.

The first tier comprises clusters with fully coherent UMA architecture. In each cluster, tasks are intended to be mapped according to their communication, processing, and memory requirements (e.g., highly communicating task are mapped to the same cluster).

The second tier is a NORMA architecture that binds multiple clusters of the first tier. In this case, tasks are intended to be mapped in different clusters that, for instance, have a sporadic communication traffic between themselves. Consequently, the one responsible for mapping tasks must take these considerations in its policy. This work employs manual mapping arranged by the developer – he chooses the cluster for application tasks execution and, if he wishes so, the particular core within the cluster (through either the sched_setaffinity system call [DIE15a] or the taskset application [DIE15b]).

Figure 59 depicts the memory and communication architecture in the first tier. Clockless repeated wires as proposed by Krishna et al. [KRI13] can be employed to maintain single-cycle hops and reduce wire delays.

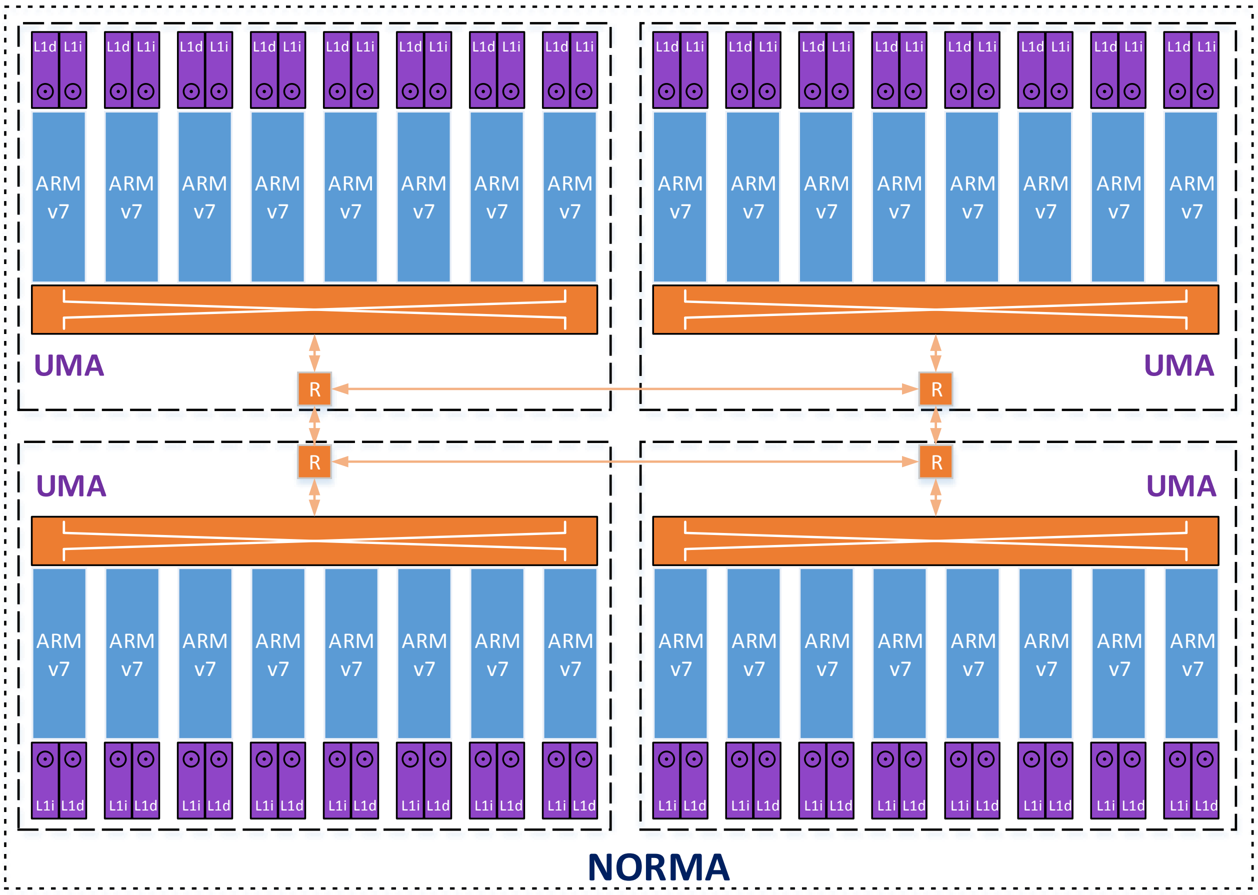


Figure 59. First tier of the system (message communication and memory architectures).

Aiming to achieve an efficient layout that approximates all the levels of the memory hierarchy to the processors, we employ 3D IC architectural model. The cache levels of the memory hierarchy can be shared or privately-accessed across the number of processors attributed to each cluster. More than one configuration is performed to explore effective throughput and energy consumption. Figure 60 depicts an example of a two die 3D system interconnected through TSVs.

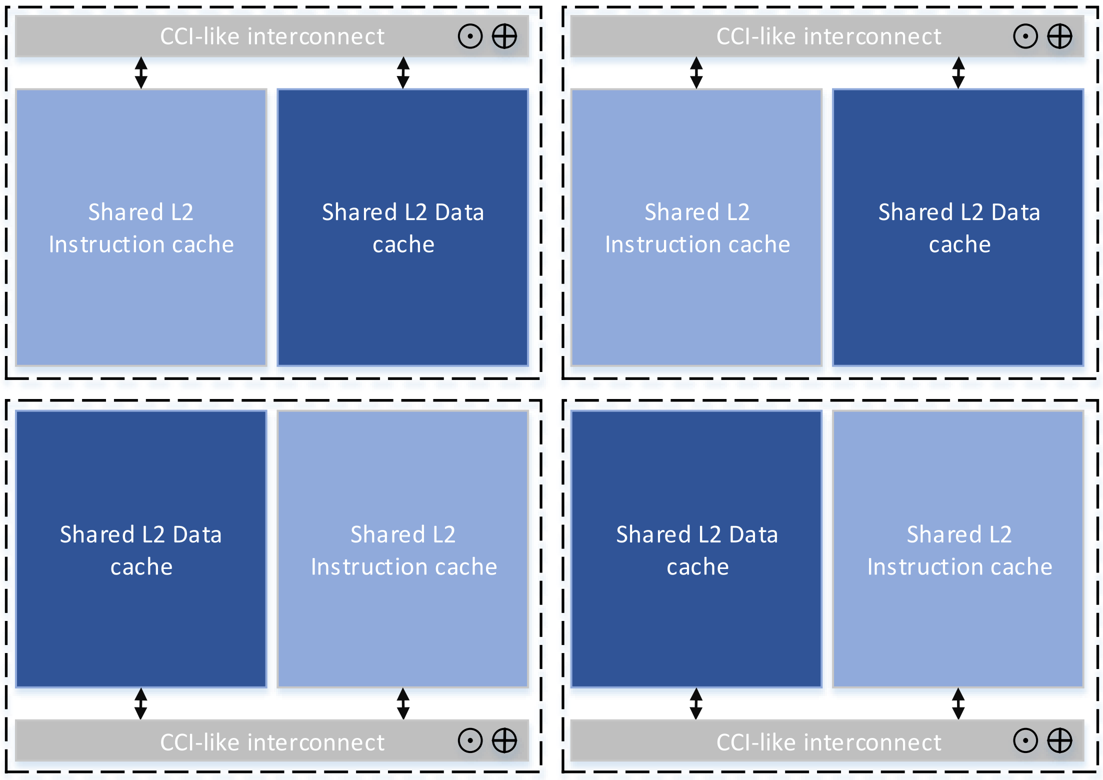


Figure 60. Second tier of the system (Cache L2 and CCI).

5.3 Limitations of the Gem5's ARM ISA

Gem5 does not impose any inherent limit of how many cores it can process [GEM15g]. However, Gem5 simulate with some nuances or restriction according to the type of target processor. For instance, GEM5's documentation for the ARM platform provides examples limited to two cores. Also, systems with more than four cores present some complications [GEM15h][GEM15i]. For the Linux kernel, the platform can inform the number of cores available in many different ways. For the particular version used in this work (3.3.0-rc3), the kernel consults the L2 Control Register for this. Unfortunately, this register reserves only two of its thirty-two bits to inform the cores count [ARM15c], which only allows identifying four cores. We modified the Linux kernel expanding this field to three bits, using one bit of reserved space, increasing the previous limitation to eight cores. Note that, core identification can also be done using the SCU control register. However, this was not implemented because there is no SCU implementation on Gem5 and this register also has the same core count limitation as the L2 register [ARM15d].

Furthermore, the GICv2 for ARM is implemented in Gem5. Regrettably, this version is restricted to service eight cores [ARM11c]. The newer version 3 of this same module is capable of running 128+ cores. Linux kernel support was added just recently (version 4 and onward) [LWN15].

Further difficulties were encountered for executing eight cores. One of the first code that every cores executes (*arch/arm/kernel/smp.c:secondary_start_kernel*) is responsible for incrementing the initial mm_struct (variable init_mm) used by the kernel. An additional increment is done on the scheduler (*kernel/sched/core.c:sched_init*). Therefore, after the system has booted, init_mm must be equal to $n + 1$, whereas n is equal to the core count. If this is not met, the system will later fail with a kernel panic. Unfortunately, only the simplest model of CPU achieves this. We do not know the source of this discrepancy and remedy this by using the switchcpu method described in Section 4.3.

The ARM ISA on Gem5 is limited to the classic memory system. In this system, only the CCI-like interconnect is supported for on-chip interconnection. This structure uses a three-layered bus for requests, responses, and snoop message handling [WIE12]. Off-chip interconnection is based on this model; however, it uses a simplified two-layered bus (lacks snoop messages) [GEM15k]. As such, there is no support for packet-based communications. We developed an abstract model for our inter-cluster communication, which will be discussed in Section 6.4.

Besides, the ARM ISA on Gem5 is limited to a classical L2 shared design. However, our intention is to evaluate five designs: two shared, two private, and one paired design based on the SPARC M7 chip (Figure 10). Therefore, we extended Gem5 to support all these designs.

Native Gem5 does not have full support for Dynamic Voltage and Frequency Scaling (DVFS). Researchers have been extending Gem5 to support this feature [HAR14][SPI13], but this requires modifications of Gem5 code or full-blown dedicated extensions of Gem5 infrastructure. Due to time constraints, we did not incorporate DVFS in our analysis.

5.4 PARSEC – Benchmark Suite for Multiprocessing

Benchmarking is the quantitative foundation for computer architecture research [BIE08]. Without a program selection that provides a representative load of the target application space, performance results can be skewed and invalidate conclusions drawn from it. A well-known fact of multiprocessing is the disruptive change of programming models for programs to benefit from their full potential. The use of older High-Performance Computing (HPC) workloads does not fit this scenario since it is based on smaller suites and sequential applications. This shortcoming is the target intended to be answered by the Princeton Application Repository for Shared-Memory Computers (PARSEC) suite [BIE08].

The first version of PARSEC was created by Intel and Princeton University [BAO15]. The latest version available of PARSEC is 3.0 [PRI15]. It is a highly used benchmark, for instance, PARSEC was employed for benchmarking in more than 55 papers in International Symposium on Computer Architecture (ISCA) from 2010 to 2014 [SOU15a].

The five objectives proposed by PARSEC are described as following:

Multi-threaded Applications - Shared-memory multiprocessor is one of the most employed architecture today for high-performance systems. The trend for future architectures is to deliver performance improvements through increasing core counts on multiprocessing. Therefore, applications that require processing power must use a parallel model of execution.

Emerging Workloads - The increase of processing power enables new classes of applications whose computational requirements were beyond the capabilities of earlier generations of processors. Hence, the benchmark suite should represent this trend.

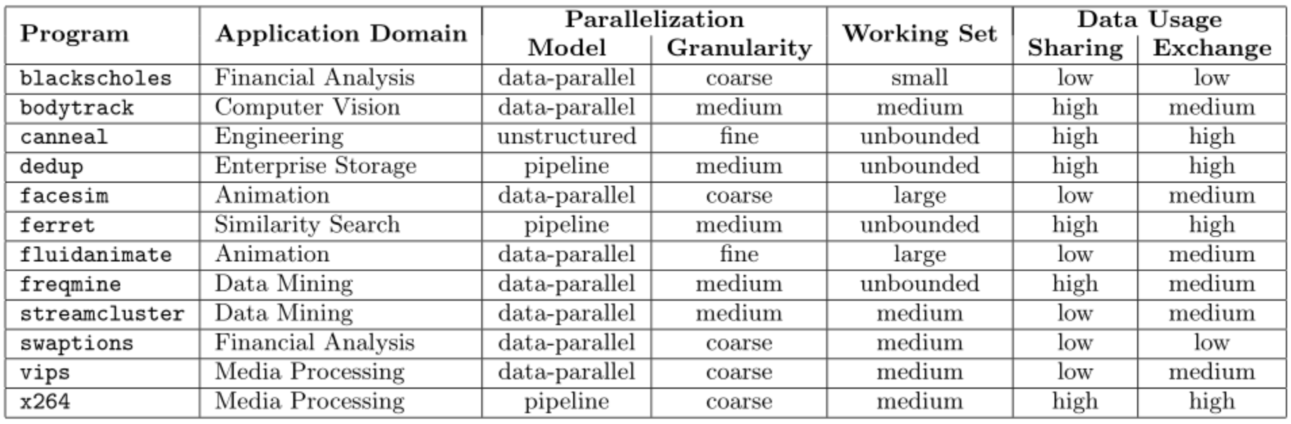
Diverse - A benchmark suite must be broad in its representative load of applications, which includes both interactive applications like computer games, offline applications like data mining, and programs with different parallelization models. While a real representative suite is impossible to create for all cases, reasonable effort should be applied to maximize the diversity of the program selection.

Employ State-of-Art Techniques - A benchmark suite must be up-to-date with current practice in parallel application techniques.

Support Research - A benchmark suite intended for research has additional requirements that goes beyond the ones used for benchmarking real machines alone. Representative input sets with different proprieties should be provided.

PARSEC fulfills the objectives by providing a rich, parallelized, state-of-the-art applications with diverse areas of research. The areas contemplated are computer vision, media processing, computational finance, enterprise servers, and animation physics. Table 10 summarizes the key characteristics of PARSEC benchmarks.

Table 10. Qualitative summary of key characteristics of PARSEC benchmarks [BIE08].



Program	Application Domain	Parallelization		Working Set	Data Usage	
		Model	Granularity		Sharing	Exchange
`blackscholes`	Financial Analysis	data-parallel	coarse	small	low	low
`bodytrack`	Computer Vision	data-parallel	medium	medium	high	medium
`canneal`	Engineering	unstructured	fine	unbounded	high	high
`dedup`	Enterprise Storage	pipeline	medium	unbounded	high	high
`facesim`	Animation	data-parallel	coarse	large	low	medium
`ferret`	Similarity Search	pipeline	medium	unbounded	high	high
`fluidanimate`	Animation	data-parallel	fine	large	low	medium
`freqmine`	Data Mining	data-parallel	medium	unbounded	high	medium
`streamcluster`	Data Mining	data-parallel	medium	medium	low	medium
`swaptions`	Financial Analysis	data-parallel	coarse	medium	low	low
`vips`	Media Processing	data-parallel	coarse	medium	low	medium
`x264`	Media Processing	pipeline	coarse	medium	high	high

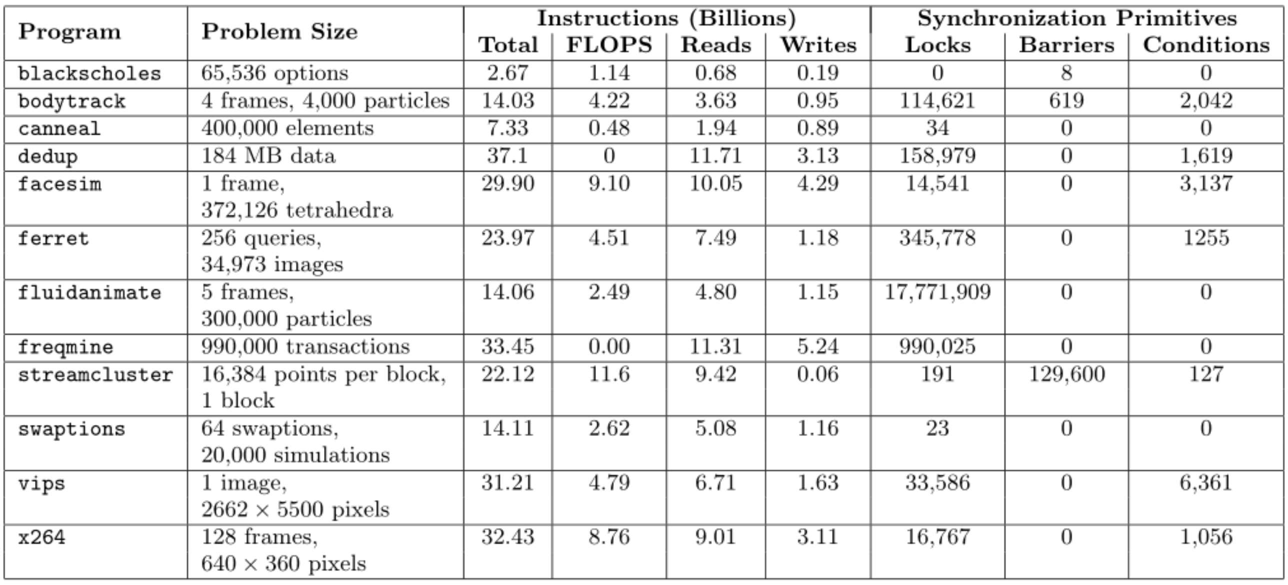
PARSEC provides three categories of input sets for each benchmark. The *test* and *simdev* are tiny input sets intended for testing and development, and should not be used for scientific studies. *Simsmall*, *simmedium*, and *simlarge* are intended for simulators and vary progressively in size and follows a trend that larger inputs contain bigger working sets and more parallelism They approximately represent the runtime execution of 1, 5, and 15 seconds [BAO15], respectively. Finally, the *native* input set is the most interesting one because it resemble real program inputs. However, its runtime execution is about 15 minutes, which is prohibitive for a full system simulator. Section 5.4.2 will show the wall-clock time of the *simmedium* and *simlarge* on the Gem5 simulator.

The remaining sections are organized as follows. First, benchmark applications used in this work will be briefly presented. Then, the Gem5 integration with PARSEC will be discussed.

5.4.1 PARSEC BENCHMARK

This work chose the following applications present on the PARSEC Benchmark: Blackscholes, Bodytrack, Canneal, Dedup, Fluidanimate, Swaptions, Vips, and x264 because they give a broad range of behaviors to analyze the impact of microarchitecture changes. From the data presented in Table 10, we can see that all parallelization model/granularity and working set are achieved. Table 11 summarizes the finer details of instructions and synchronization primitives employed on all benchmark applications under an 8-core system with the input set *simlarge*.

Table 11. Breakdown of finer details of the benchmark applications for input set *simlarge* on a system with 8 cores [BIE08].



Program	Problem Size	Instructions (Billions)				Synchronization Primitives		
		Total	FLOPS	Reads	Writes	Locks	Barriers	Conditions
blackscholes	65,536 options	2.67	1.14	0.68	0.19	0	8	0
bodytrack	4 frames, 4,000 particles	14.03	4.22	3.63	0.95	114,621	619	2,042
canneal	400,000 elements	7.33	0.48	1.94	0.89	34	0	0
dedup	184 MB data	37.1	0	11.71	3.13	158,979	0	1,619
facesim	1 frame, 372,126 tetrahedra	29.90	9.10	10.05	4.29	14,541	0	3,137
ferret	256 queries, 34,973 images	23.97	4.51	7.49	1.18	345,778	0	1255
fluidanimate	5 frames, 300,000 particles	14.06	2.49	4.80	1.15	17,771,909	0	0
freqmine	990,000 transactions	33.45	0.00	11.31	5.24	990,025	0	0
streamcluster	16,384 points per block, 1 block	22.12	11.6	9.42	0.06	191	129,600	127
swaptions	64 swaptions, 20,000 simulations	14.11	2.62	5.08	1.16	23	0	0
vips	1 image, 2662 × 5500 pixels	31.21	4.79	6.71	1.63	33,586	0	6,361
x264	128 frames, 640 × 360 pixels	32.43	8.76	9.01	3.11	16,767	0	1,056

The application descriptions are based on the characterization presented in [BIE08][BAO15].

Blackscholes is an Intel RMS application that calculates the prices for a portfolio of European options analytically with the Black-Scholes partial differential equation. It has been shown that there is no closed-form expression for the Black-Scholes equation and, hence, it must be computed numerically. This program is limited by the amount of floating-point calculations a processor can perform. Input sets are synthetic based on replication of a 1000-real options. This application is the simplest of all PARSEC benchmarks and has negligible communication.

Bodytrack computer vision is also an Intel RMS application that tracks a 3D pose of a market-less human body with multiple cameras through an image sequence. Bodytrack employs an annealed particle filter to track the pose using edges and the foreground silhouette as image features. Computing vision is a significant step to the machines interact with the environment dynamically. Input sets are from a video feed from 4 cameras. The communication in this benchmark presents a bigger impact than in the case of Blackscholes. Bodytrack suffers to achieve expected speed-up with eight cores or more, which is inherent from its sequential sections and redundant computations. This application has the least theoretical speed-up for all benchmarks.

Canneal is the first application developed by Princeton University, which uses cache-aware Simulated Annealing (SA) to minimize the routing cost of a chip design. SA is a common method to approximate the global optimum in a large search space. Canneal uses a very aggressive synchronization strategy that is based on data race recovery instead of avoidance. The cases where data is misread due to synchronization issue are accepted as swaps that increase the routing cost momentarily but may result in a global decrease. The swap operations employ lock-free synchronization that are implemented with atomic instructions. Input sets are from a synthetic netlist. Canneal has the workload with the most critical memory responses.

Dedup is an application developed by Princeton University that achieves high compression ratios of a data stream combining global and local aspects. Such compression is called ‘deduplication’. Also, this benchmark uses a pipelined programming model to parallelize the compression where five pipelines stages are employed, the intermediate three of which are parallel. Dedup is intended for enterprise storage servers, where each input is an archive that contains a selection of files.

Fluidanimate is an Intel RMS application that uses an extension of a previous method called Smoothed Particle Hydrodynamics to simulate an incompressible fluid for interactive animation purposes. Its output can be visualized by detecting and rendering the surface of the fluid. The input set comprises of many particles and frames. Fluidanimate uses by far the largest number of synchronization locks primitives. Also, this is the only application that has over 5% of parallelization overhead when comparing a parallel version to a serial version, which limits its achievable speedup in multiprocessor architectures.

Swaptions is an Intel RMS application that uses the Heath-Jarrow-Morton framework to price a portfolio, which describes how interest rates evolve for risk management and asset liability management for a class of models. Swaptions employs Monte Carlo simulation to compute the prices. The input set includes some swaptions and simulation runs. This application has few communications between cores.

Vips is based on the VARASI Image Processing System, which was developed through several projects funded by the European Union. The application includes fundamental image operations such as affine transformation and convolution that uses the VARASI system to construct, transparently at runtime, multi-threaded image processing pipelines. The input set is an uncompressed image to be transformed into another image.

X264 is an H.264/AVC video encoder application based on the ITU-T H.264 standard, which is now also part of ISO/IEC MPEG-4. H.264 describes the lossy compression of a video stream, which improves over previous video encoding standards by employing new features that achieve a higher output quality with a lower bit-rate at the expense of a significantly increased encoding and decoding time. The parallel algorithm of x264 uses the pipeline model with one stage per input video frame. X264 processes some pipeline stages equal to the number of encoder threads in parallel, resulting in a sliding window which moves from the beginning of the pipeline to the end. The input set is a video with a given resolution and a given number of frames. This benchmark is very communication intensive.

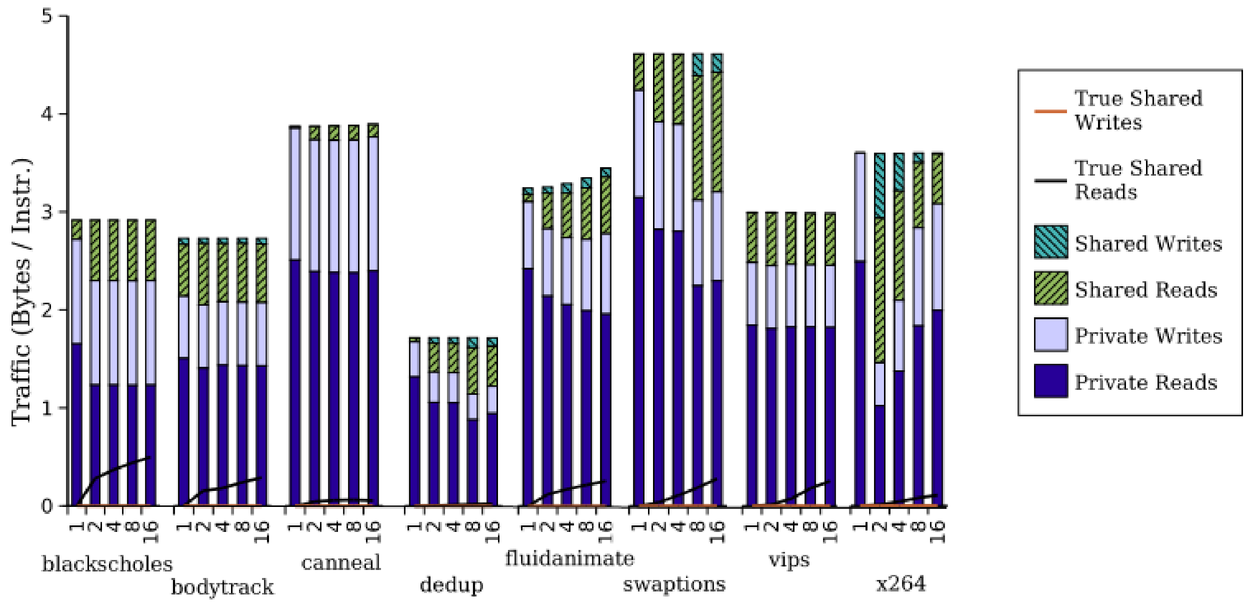


Figure 61. Cache traffic in bytes per instruction for one to sixteen cores (based on [BIE08]).

Bienia et al. [BIE08] used CMP$im with the Pin simulator to obtain experimental results for the PARSEC benchmark. The baseline cache configuration was a shared 4-way associative cache of 4MiB capacity with 64-byte lines. Figure 61 shows the cache traffic of shared writes and reads, private writes and reads, and true shared writes and reads. Simulation is done using one, two, four, eight, and sixteen cores. An access is a true access if the last reference to a given cache line came from another thread. As can be inferred from this figure, all programs exhibit very few true shared writes.

Execution of a PARSEC application can be achieved in two ways: full execution or partial execution. The full execution is the measurement of time elapsed from the start to the end of execution of all instructions, which includes the serial portions of the application and thread instantiation. Partial execution is achieved through measurement in specific places already provided by PARSEC. These places are called Region Of Interest (ROI) and delimitate the parallel portion of the applications [BAO15]. When these places are reached, a user-defined function is called so that the designer can set up the measurement environment. ROI was created for mitigating the skew of running inputs smaller than native since they provide a smaller fraction of parallel code versus serial code. Hence, Bienia et al. [BIE08] state that it is safe to assume that the serial initialization and shutdown phases are negligible in the real inputs, and this allows one to completely ignore them for experiments.

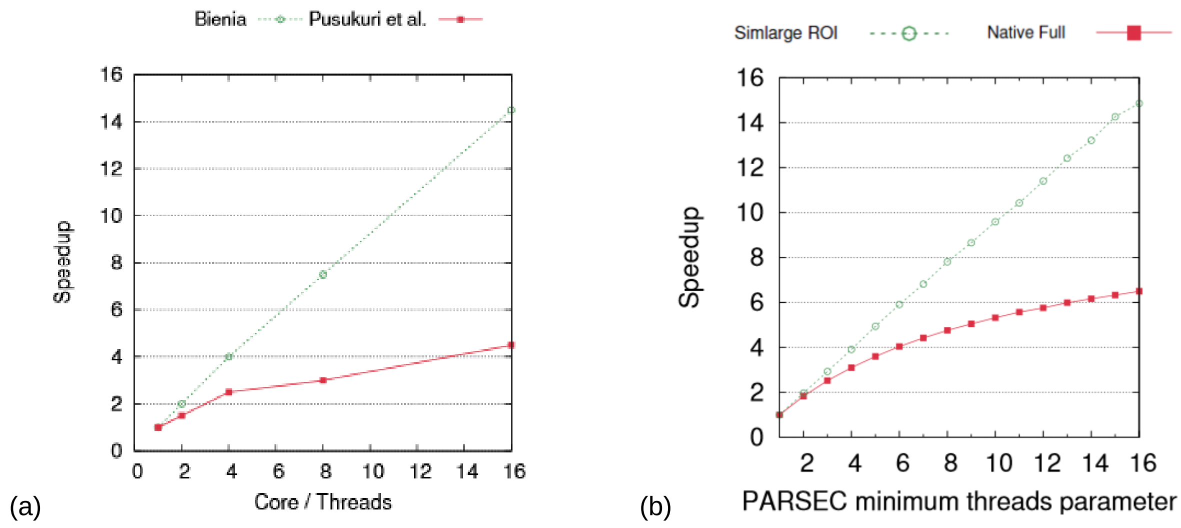


Figure 62. Scalability of blackscholes (a) as reported by Pusukuri, Gupta, and Bhuyan [PUS11]; (b) measured by Shoutern and Renau [SOU15a][SOU15b].

However, Southern and Renau [SOU15a] shown that the use of ROI can mask significant limitation of the parallelization of some applications. Their work is based and motivated by previous findings by Pusukuri, Gupta, and Bhuyan [PUS11], which are shown in Figure 62(a). Figure 62(b) shows the measured results of their work and pinpoints the source of discrepancy between ROI and full simulation. Thus, Southern and Renau extend the work on [PUS11] to measure the runtime scalability of the four main PARSEC input sets (i.e., *simsmall*, *simmedium*, *simlarge*, and *native*) and compare the scalability of the ROI with that of the whole application.Experimental results were conducted on the 13 benchmarks available in PARSEC using three different real multicore systems. Evaluation of execution is done through the wall-clock runtime. The systems tested are: a single CPU with 4 cores, 2 threads per core; a dual socket system with 8 cores per socket, 2 threads per core and; a quad socket system with 12 cores per socket, 1 thread per core.

Figure 63 is a collection of some of the results presented in [SOU15a] work. M32 and M48 are the dual- and quad-socket machines, respectively.

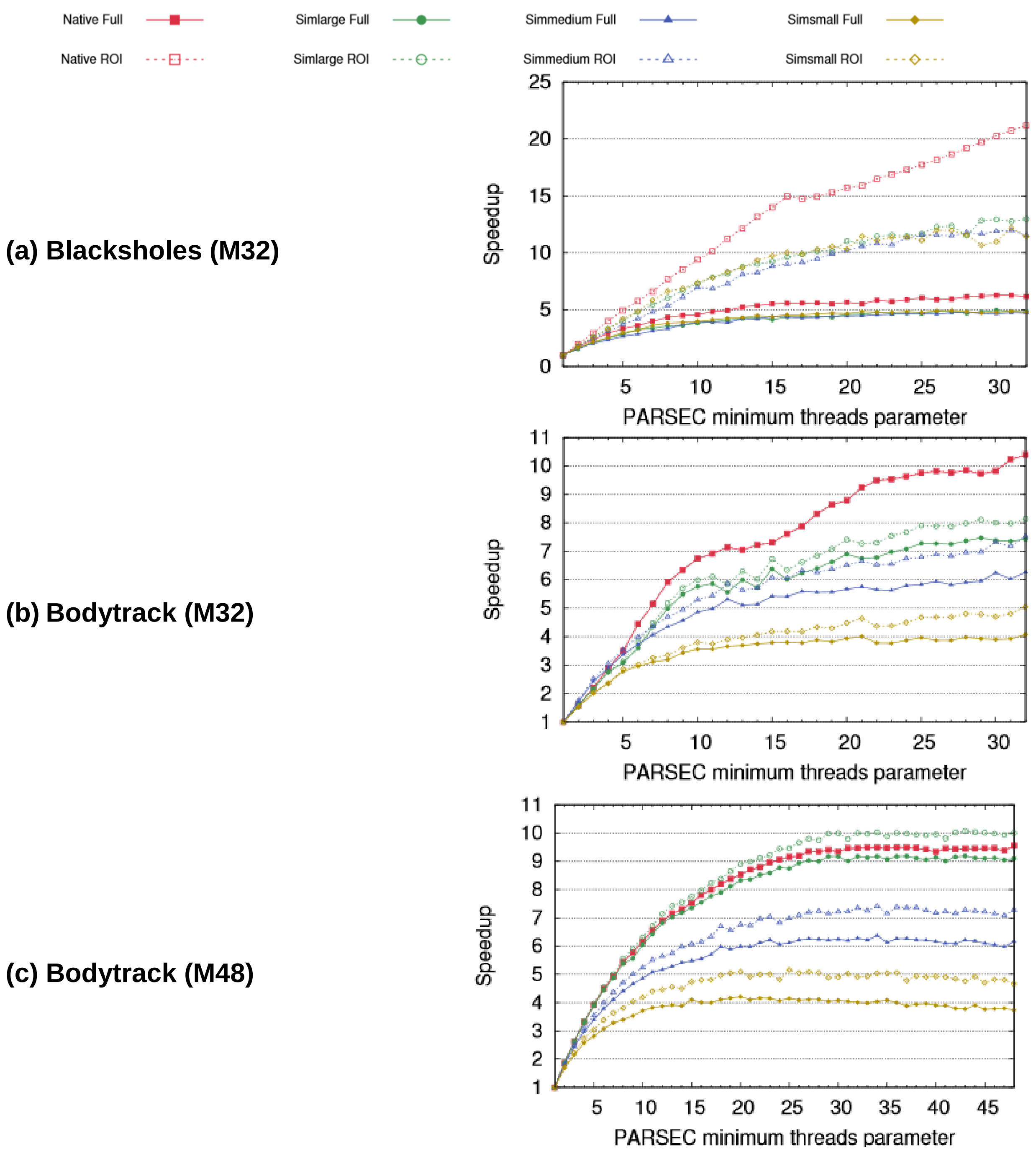


Figure 63. Speedup of bodytrack and blackscholes for all input sets for both ROI and full execution (based on [SOU15a]).

These results show that the relation between simulation and native input sets, and also ROI and full execution, is a complex matter. Figure 63(a) shows a case where all ROI executions extrapolate the real achievable speedup greatly. Figure 63(b) shows a case where full/ROI native run has the best speedup while the *simlarge* input full/ROI run has a lower speedup. Meanwhile, Figure 63(c) shows similar results for native and *simlarge* inputs using the same benchmark but with a different machine configuration.

Finally, Figure 64 shows that there is no single consistent trend on input sets and full execution versus ROI. Some benchmarks show a consistent relation between *native* and *simlarge* inputs (Blackscholes, Bodytrack), full execution and ROI (Swaptions, Vips), while others do not show it at all (Canneal and Raytrace).

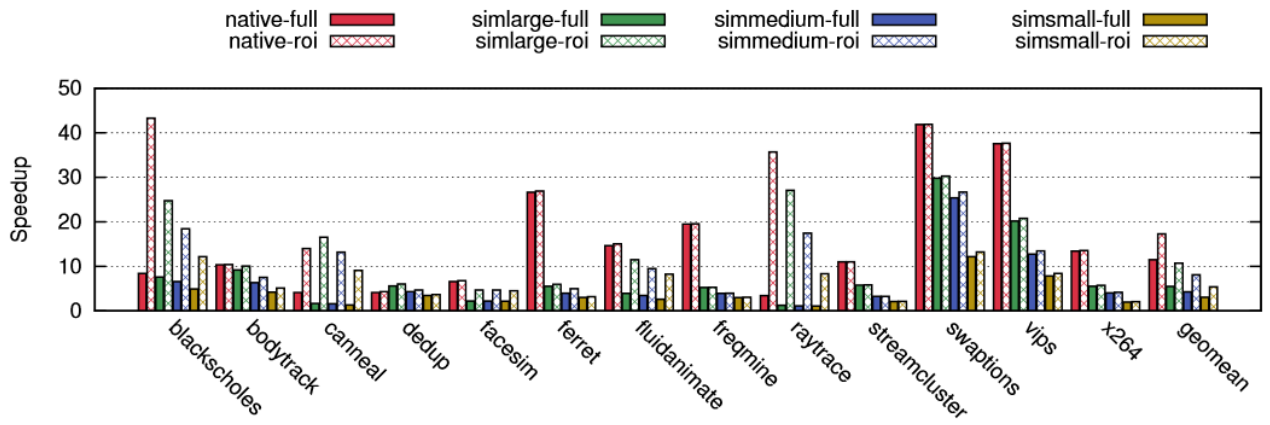


Figure 64. Maximum speedup observed for all machines tested for each benchmark, region, and input set combination [SOU15a].

Table 12 shows the speedup variation comparing the *native* set with full simulation model with *simlarge* set and both full and ROI simulation models. The table is presented in percentage variation having as reference the native set model. This table allows to observe the accuracy of each simulation model.

Table 12. The speedup of simlarge full and ROI models normalized according to the native full model. Positive and negative values mean speedup increase or reduction, respectively.

Speedup variation (%)	Application													
Simulation model	Blackscholes	Bodytrack	Canneal	Dedup	Facesim	Ferret	Fluidanimate	Freqmine	Raytrace	Streamcluster	Swaptions	Vips	x264	geomean
Simlarge-full	-11.1	-10.0	-50.0	50.0	-57.1	-70.6	-73.3	-68.4	-50.0	-45.5	-28.6	-47.4	-57.1	-58.3
Simlarge-ROI	177.8	0.0	325.0	75.0	-28.6	-64.7	-20.0	-68.4	575.0	-45.5	-28.6	-44.7	-57.1	-8.3

Our work employs *simlarge* set with full simulation to model all the benchmark applications. As a consequence, Blackscholes and Bodytrack simulation are the ones that present the most consistent results when compared with their *native* counterpart, while Fluidanimate is the opposite. In the next subsection, we will describe the integration of the PARSEC benchmark with Gem5 and explain why the *native* set was not employed.

5.4.2 Integration with Gem5's ARM ISA

The first step to integrate the PARSEC benchmark suite into Gem5 is to compile it. The management program used by parsec, parsecmgmt, is tailored to compile applications for the host machine. In other words, cross-compiling is not natively supported. Also, ARM is not officially supported by PARSEC. Fortunately, other researchers have already tackle this situation.

A QEMU machine was built to emulate an ARM Versatile Express board and to compile PARSEC natively to solve these issues. The machine followed the instructions underlined in this link [UBU15a]. However, it is not enough, so a patch [GEM15l] is applied to make PARSEC compatible with the ARM ISA. This patch adds atomic instructions for this platform. Additionally, we had to decrease the optimization level of the compiler (O3 to O2) for the x264 application due to corruption on the clean-up phase of this application. This behavior was also encountered by other PARSEC users [PAR15a]. Ultimately, all benchmarks can be compiled except raytrace since it uses Intel's *Streaming SIMD Extensions* (SSE) [GEM15l].

The architecture exploration on GEM5 can use *timing* or *O3* CPU models. The following tables will show the difference in wall-clock time for simulating PARSEC benchmarks. A single core[8] of Intel Xeon E5-2660 clocked at 2.60 GHz is employed for simulation.

Table 13 shows the wall-clock time for a number of benchmarks using the *timing* CPU model. They range from 30 minutes up to 29 hours and 30 minutes. All simulations were done using eight cores with the same cache and memory configurations. In our experiments, changing the main memory configuration did not result in a significant change of wall-clock time. However, the number of CPU cores impact the simulation time. A simulation of 8 timing CPU cores is 2.2 times slower than the same simulation with only one core, in average.

Table 13. Wall-clock time for the timing CPU model of a broad range of PARSEC applications.

	Blackscholes	Bodytrack	Dedup	Swaptions	Vips	x264
simmedium	0.5 hour	1 hour	2 hours	2 hours	3.5 hours	10 hours
simlarge	2 hours	4 hours	10 hours	11 hours	9.5 hours	29.5 hours

Early results demonstrated that the *timing* CPU model did not sufficiently model the cores to capture nuances of changes in the main memory or cache organizations regarding simulated execution time. Hence, we were forced to use the more detailed *O3* CPU model. As previously stated by the authors of Gem5 [SAI12], *O3* is an order-of-magnitude slower than the *timing* and *atomic* CPU models. This can be seen in the experimental results of Table 14, where all applications more than doubled its required simulation time. Also, two additional applications were computed. For this scenario, execution of a single core was not conducted.

Table 14. Extended range of PARSEC applications under eight O3 CPU cores. Simulation time is in wall-clock time.

	Blacksc.	Bodytrack	Canneal	Dedup	Fluid.	Swaptions	Vips	x264
simmedium	2 hours	4 hours	8 hours	8.5 hours	6 hours	11 hours	13 hours	33.5 hours
simlarge	7.5 hours	13 hours	16.5 hours	54 hours	15 hours	42 hours	35 hours	108 hours

Table 14 shows why we did not use the *native* set for simulation. As the intended execution time for *simlarge* is 15 seconds and for *native* is 15 minutes, execution of a single application on Gem5 would easily surpass a month.

5.5 NASA NAS Parallel Benchmark

The Numerical Aerodynamic Simulation (NAS) Parallel Benchmarks (NPB) is another benchmark for testing the capabilities of parallel computers. Its main difference from PARSEC is the ability to select some parallelization techniques providing shared memory and message passing programming models freely. The two main frameworks are OpenMP – for shared memory – and MPI – for message passing – and some additional frameworks [NAS15a]. Aftosmis et al. [AFT06] tested both versions and found out that OpenMP

[8] Gem5 is a single-threaded application.

demonstrated better performance on a small number of CPUs, but the MPI implementation produced significantly better scaling.

The applications of these benchmarks are derived from computational fluid dynamics used for aerophysics applications [SAI96]. These applications are distinguished from traditional scientific applications due to its large memory requirements [WAH98]. Although the NPB suite is rooted in the problems of computational fluid dynamics, they are valuable in the evaluation of parallel computing since they are rigorous and as close to real applications as can be expected from a benchmark suite [ALM04].

Four categories of increasingly input sizes are currently available. The first category has one size (S) and is intended for quick test purposes. The second category also has one size (W) and mimics the input set expected for a 90’s workstation. The third category is comprised of three sizes (A, B, and C) and is considered the standard input set. From one class to the next the input size roughly increases four-fold. Finally, the last category is also comprised of three sizes (D, E, and F) and is considered the large input set. From one class to the next the input size roughly increases sixteen-fold [NAS15a]. Table 15 shows the parameter values of the CG application for all input sets (expect class F). The CG application will be described in the following section.

Table 15. Input set parameters for the CG application (based on [NAS15b]).

Benchmark	Parameter	Class S	Class W	Class A	Class B	Class C	Class D	Class E
CG	no. of rows	1400	7000	14000	75000	150000	1500000	9000000
	no. of nonzeros	7	8	11	13	15	21	26
	no. of iterations	15	15	15	75	75	100	100
	eigenvalue shift	10	12	20	60	110	500	1500

5.5.1 NAS Parallel Benchmark Applications

From the eleven applications provided by the benchmark suite, the following were chosen: MG, CG, FT, IS, and LU. As the objective of using such suite is testing scalability on message passing systems, applications that do not have this profile were excluded. For instance, the EP benchmark provides an estimate of the upper achievable limits for floating-point performance and requires almost no communication [SAI96]. Using such benchmark would be redundant since we already employ Blackscholes from PARSEC to do this.

The application descriptions are based on characterization presented in [BAI94][BAI95][SAI96]. Most benchmarks from NAS are based on iteration steps involving mathematical equations. Hence, this allows the application to self-validate each iteration [BAI95].

MG is a simplified MultiGrid kernel, which solves the 3D Poisson partial difference equation. This problem is simplified in the sense that it uses constants rather than variable coefficients as in a more realistic application. This benchmark is a good test of both short and long distance highly structured communication.

CG is a Conjugate Gradient method used to compute an approximation to the smallest eigenvalue of a large, sparse, symmetric positive definitive matrix. Complementary to MG, it tests irregular long distance communication, employing unstructured matrix-vector multiplication.

FT is another 3D partial differential equation solver using Fast Fourier Transform. This is a test of long distance communication performance.

IS is a benchmark that moves significant data and executes integer computation to achieve sorting operations in particle method codes. This benchmark must sort some generated values in parallel. This problem is unique in NPB that floating-point arithmetic is not involved.

LU is a benchmark that solves a synthetic system of nonlinear partial differential equations using a symmetric successive over-relaxation solver. LU is very sensitive to the small message communication performance of MPI. It is the only benchmark from version 2.0 that sends large numbers of very small messages (40 bytes).

As shown by these descriptions, the NAS benchmark suite focus on a class of computational fluid dynamics applications that solve partial differential equations. Waheed and Yan [WAH98] did the workload characterization of this benchmark to generalize the results of various measurements of its runtime behavior.

The measure of computation versus communication of class A of the NPB benchmarks is the first contribution of Waheed and Yan's work. Figure 65 shows the alternative nature of their relation. Computation in this context is the execution of instructions and memory fetches. Communication is the execution of system calls for invoking a communicative service and network resources for actual data transfer.

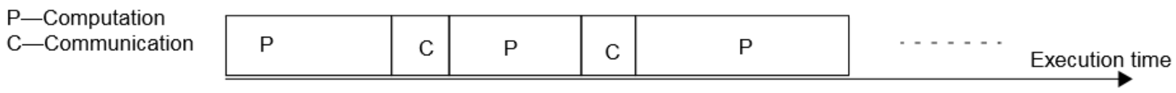
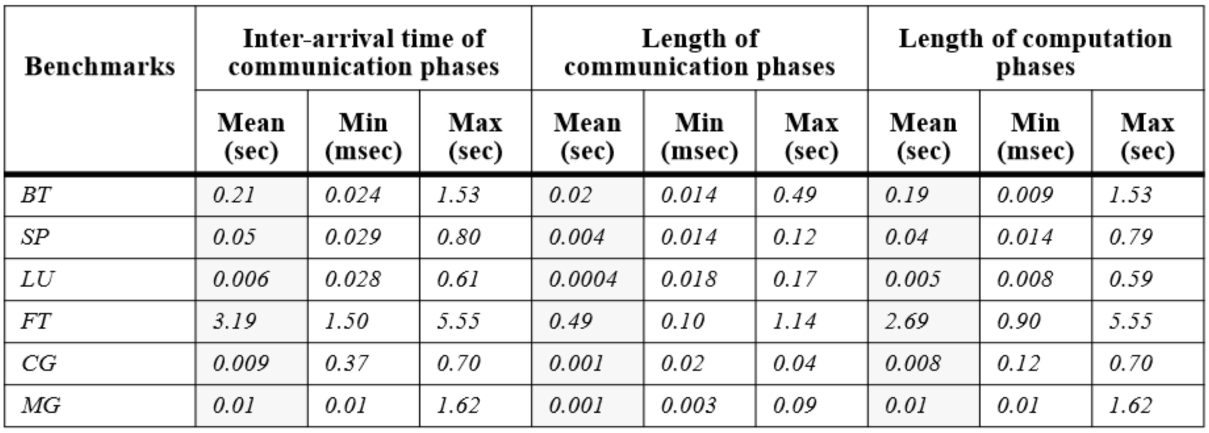


Figure 65. Alternating computation and communication phases in an application [WAH98].

Table 16 shows the results obtained in an NUMA architecture called Origin2000. Each node of the Origin2000 is comprised of an MIPS processor with two levels of separate data and instruction caches for each processor. On-chip cycle counter were employed to determine the depicted characteristics.

Table 16. Statistics of communication and computation phases of NPB on Origin2000 [WAH98].



Benchmarks	Inter-arrival time of communication phases			Length of communication phases			Length of computation phases		
	Mean (sec)	Min (msec)	Max (sec)	Mean (sec)	Min (msec)	Max (sec)	Mean (sec)	Min (msec)	Max (sec)
BT	*0.21*	*0.024*	*1.53*	*0.02*	*0.014*	*0.49*	*0.19*	*0.009*	*1.53*
SP	*0.05*	*0.029*	*0.80*	*0.004*	*0.014*	*0.12*	*0.04*	*0.014*	*0.79*
LU	*0.006*	*0.028*	*0.61*	*0.0004*	*0.018*	*0.17*	*0.005*	*0.008*	*0.59*
FT	*3.19*	*1.50*	*5.55*	*0.49*	*0.10*	*1.14*	*2.69*	*0.90*	*5.55*
CG	*0.009*	*0.37*	*0.70*	*0.001*	*0.02*	*0.04*	*0.008*	*0.12*	*0.70*
MG	*0.01*	*0.01*	*1.62*	*0.001*	*0.003*	*0.09*	*0.01*	*0.01*	*1.62*

This figure shows that FT has a distinct mean value for all analyzed characteristics. This is justified, in part, by the use of long distance communications, as stated earlier. The fact that the computational phase is also particularly long points to the fact that FT uses few but large communications, while all the other benchmarks scatter the communication throughout their execution [WAH98].

Waheed and Yun also observed that 80% of execution time is spent in accessing memory when executing either floating-point as well as integer operations. Based on this measurement, the authors observed that almost every floating point instruction in the benchmarks analyzed require at least one memory access. This observation is consistent with the common belief that computational dynamic fluid solvers are often memory bound applications [WAH98]. Almojel [ALM04] also found that memory operations dominated the execution of the benchmarks EP, MG, CG, FT, and BU under the class A input set.

For the memory subsystem of Origin2000, Waheed and Yun found that L1 cache miss rate falls in the range of 11% to 27%, and L2 cache miss rate falls in 4% to 22%. Translation Lookaside Buffer (TLB) misses are insignificant due to fine-grained data distribution on the benchmarks. This analysis is for the following benchmarks: BT, SP, LU, FT, CG, and MG [WAH98].

5.5.2 INTEGRATION WITH GEM5'S ARM ISA

As NPB does not use special architectural instructions, it was easily compiled for ARM. The intended framework for communication chosen for this work is MPI since its compatible with message passing systems. For MPI to execute in a cluster of processors, some auxiliary tools are needed. In this work, the mpich2 (an application that implements the MPI standard) is supported with a password-less private-key authentication SSH and appropriate user for it, which is done according to the guidelines of [UBU15b].

Table 17 depicts the simulation time required to execute the five NASA NAS applications selected and employing a system with 4 clusters of 8-core each (32 cores total). LU used a smaller input set (W) since its simulation time is demanding.

Table 17. Wall-clock time for the O3 CPU model of NASA NAS applications.

	CG	FT	IS	LU	MG
W	-	-	-	89.8 hours	-
A	20.5 hours	83.3 hours	21.4 hours	-	27.9 hours

6 EXPERIMENTAL RESULTS

This chapter describes and discusses the experimental results, which are divided into four groups: workload distribution, main memory, cache, and scalability. The workload evaluation depicts how the Linux kernel distributes the parallel applications among all available cores. For main memory and cache evaluation, shared memory is employed for inter-core communication and the PARSEC benchmark defines the applications domain. Finally, for scalability evaluation message passing is employed for inter-core communication and the NASA NAS benchmark characterizes the applications domain.

The PARSEC benchmark uses a user-defined value that determines the minimum number of threads spawn by each benchmark. For this work, this value was fixed to the number of cores present in a single cluster. However, it does not mean that each core is running a single thread as PARSEC can spawn additional threads. The real number of threads is depicted in Table 18. As can be seen from this table, the x264 benchmark has the greatest number of threads. This justifies its discrepancy of simulation time when compared to other benchmarks shown in Table 14 – more than double the time of Dedup, the second most demanding benchmark for simulation time. According to the PARSEC user forum [PAR15b], x264 tries to limit the pool of threads by the user-defined minimum number of threads. In other words, the values shown in Table 18 are for the total number of threads created over the entire execution time.

Table 18. Number of threads spawn by each benchmark, where *n* is the user-defined minimum threads parameter (based on [SOU15a]).

Benchmark	Threads	*simlarge* $(\boldsymbol{n} = \mathbf{8})$
Blackscholes	$1 + n$	9
Bodytrack	$2 + n$	10
Canneal	$1 + n$	9
Dedup	$3 + 3n$	27
Fluidanimate	$1 + n$	9
Swaptions	$1 + n$	9
Vips	$3 + n$	11
x264	$1 + 2 \times frames$	257

The NASA NAS Parallel benchmark also uses a user-defined number – however, for this benchmark, this number is for defining core units. For this work, this value was fixed to the number of cores present in the system.

For all evaluations, a baseline cache was set up as shown in Table 19, which contains an example of the baseline organization in an architecture with four CPUs. The cache and scalability evaluations define additional configurations that will be discussed in their respective sections. The baseline configuration has two L1 caches for each core, one for the instruction cache and another for the data cache. In addition, an L2 cache is shared by all cores and uses data prefetching to reduce execution time. A strided prefetcher is employed that monitors the core's address pattern to detect and prefetch constant stride array references originating, for instance, from looping structures [VAN00]. This configuration is based on the Versatile Express board presented in Table 8 from Section 5.1.

Table 19. Baseline cache configuration.

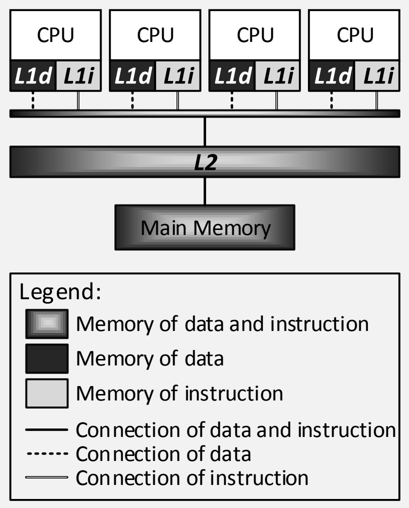
Cache baseline	*Parameter*	*Values*	*Example of baseline organization with 4 CPUs*
Private L1 I+D (Instruction and data)	Size	32 KiB + 32 KiB	 CPU L1d L1i L2 Main Memory Legend: Memory of data and instruction Memory of data Memory of instruction Connection of data and instruction Connection of data Connection of instruction
	Hit Latency	1 ns + 2 ns	
	Associativity	2-way + 2-way	
	MSHR queue size	4 + 6	
Shared L2	Size	1 MiB	
	Hit Latency	12 ns	
	Associativity	16-way	
	MSHR queue size	16	

Figure 66 depicts that all experimental results require 6622.6125 hours (corresponding to 276 days, approximately) of Gem5 simulating. Note that, excluding the scalability, every experiment can be contained in a single core. Hence, more than one simulation can be performed simultaneously. The experiments of scalability employ 32 cores while the others employ 8 cores. The additional runs were needed to explore particular cases for the experimental evaluations, and they will be discussed in the following sections. Besides, we executed two times every experiment to check for improper behavior.

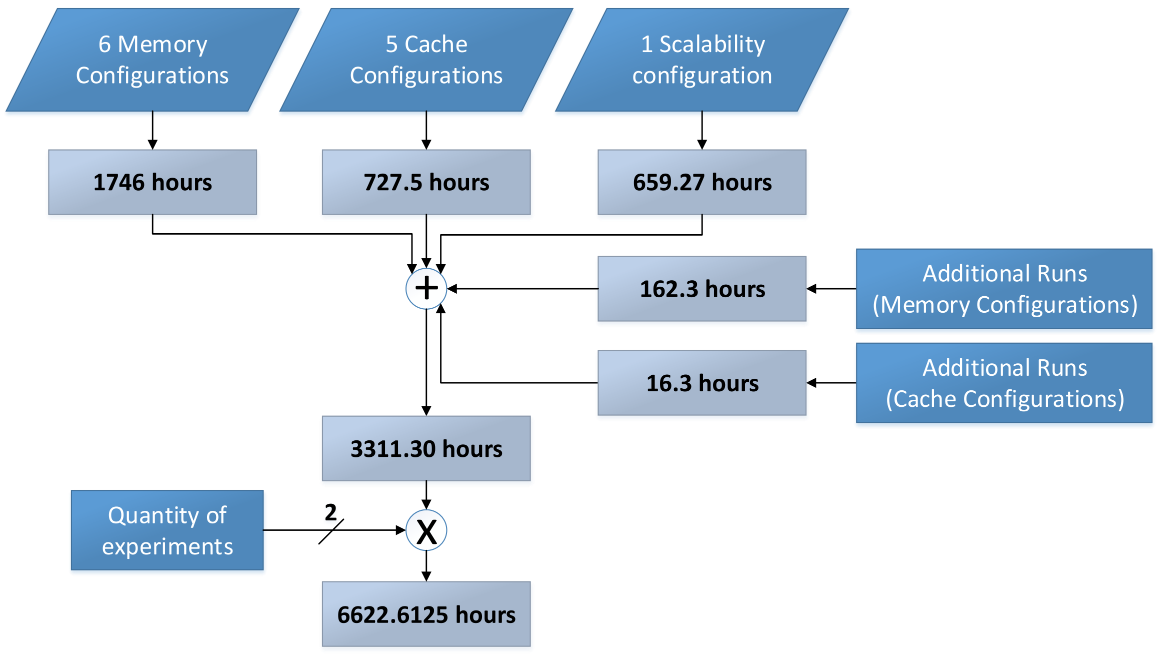


Figure 66. Total simulation time for experimental evaluations.

The rest of this chapter is organized as follows. Section 6.1 shows the distribution of our parallel workload over the available cores. Section 6.2 evaluates six technologies for main memory and its effect on the system. Section 6.3 evaluates five cache architectures and its effects. Finally, Section 6.4 shows the behavior of our system using different sized UMA clusters.

6.1 Workload Evaluation

This evaluation intends to determine how the workload is dynamically distributed over the available cores of the system. Southern and Renau [SOU15a] already analyzed the speedup of the PARSEC workload for real machines, and we discussed its results in Section 5.4.1. Here we are interested to see the effects of using the vanilla Linux kernel 3.03 scheduler on the instruction distribution and the power dissipation of the cores. Table 20 summarize the results obtained running all eight PARSEC's benchmark for instruction count, while Table 21, Table 22, Table 23, and Table 24 do the same for five NASA NAS's benchmark for 32 cores.

Table 20 shows that Dedup and Swaptions present an excellent workload distribution over all cores evenly. The maximum deviation for more or less instruction is within the range of 11% and 0.004%, respectively. Vips and x264 still manage to distribute its workload, but the maximum deviation is within the range of 24% and 40%, respectively. All other benchmarks showed a master-slave like structure where a single core dominates the instruction count. Bodytrack, Fluidanimate, and Blackscholes have a single core with up to 78%, 161% and 171% increase of the least executed core in each benchmark, respectively. Canneal has a massive disparity with the core 3, which executes 320% more instructions than the core 0, the least executed core. Southern and Renau corroborate the limited scalability of Blackscholes, Bodytrack, and Canneal (Figure 64). Blackscholes also showed the least amount of instructions and corroborated its design as being the simplest of the PARSEC suite [BIE08]. The instruction count is one of the limiting factors for the effective speedup of a parallel application and a crucial factor for the power dissipation that will be analyzed shortly.

Table 20. Total instruction count per million (× 10^6) and per core count for each PARSEC application.

Application	Total of instructions	Core 0	Core 1	Core 2	Core 3	Core 4	Core 5	Core 6	Core 7
Blackscholes	7.403	761 **(10%)**	767 **(10%)**	761 **(10%)**	2.061 **(28%)**	762 **(10%)**	759 **(10%)**	763 **(10%)**	767 **(10%)**
Bodytrack	14.084	1.773 **(13%)**	1.312 **(9%)**	2.142 **(15%)**	2.207 **(16%)**	1.331 **(9%)**	2.083 **(15%)**	1.998 **(14%)**	1.233 **(9%)**
Canneal	12.204	299 **(2%)**	309 **(3%)**	309 **(3%)**	10.015 **(82%)**	312 **(3%)**	319 **(3%)**	312 **(3%)**	325 **(3%)**
Dedup	63.333	8.068 **(13%)**	7.536 **(12%)**	8.019 **(13%)**	8.381 **(13%)**	7.802 **(12%)**	7.532 **(12%)**	8.216 **(13%)**	7.775 **(12%)**
Fluidanimate	13.356	1.535 **(11%)**	1.626 **(12%)**	1.316 **(10%)**	3.322 **(25%)**	1.421 **(11%)**	1.513 **(11%)**	1.270 **(10%)**	1.350 **(10%)**
Swaptions	44.259	5.524 **(12%)**	5.542 **(13%)**	5.520 **(12%)**	5.525 **(12%)**	5.546 **(13%)**	5.523 **(12%)**	5.534 **(13%)**	5.540 **(13%)**
Vips	36.721	4.010 **(11%)**	4.365 **(12%)**	4.769 **(13%)**	4.994 **(14%)**	4.596 **(13%)**	4.643 **(13%)**	4.625 **(13%)**	4.715 **(13%)**
X264	145.215	15.657 **(11%)**	14.924 **(10%)**	19.217 **(13%)**	16.005 **(11%)**	19.815 **(14%)**	20.975 **(14%)**	19.000 **(13%)**	19.620 **(14%)**

Table 21, Table 22, Table 23, and Table 24 show the instruction count for all five applications of the NASA NAS benchmark. All applications show a remarkable distribution of instructions on all cores. Deviations are within one percentage point for either more or less instructions. FT and LU have three to four times the amount of instructions compared to the other applications, approximately. This corroborate the simulation time required for each application, as shown in Table 17 from Section 5.5.2.

Table 21. Total instruction count per million (× 10^6) and per core count (0 – 7) for each NASA NAS application.

Application	Total of instructions	Core 0	Core 1	Core 2	Core 3	Core 4	Core 5	Core 6	Core 7
CG	49.382	1.478 **(3%)**	1.585 **(3%)**	1.593 **(3%)**	1.502 **(3%)**	1.534 **(3%)**	1.699 **(3%)**	1.579 **(3%)**	1.578 **(3%)**
FT	177.709	4.423 **(2%)**	5.769 **(3%)**	5.901 **(3%)**	5.861 **(3%)**	5.770 **(3%)**	5.787 **(3%)**	5.724 **(3%)**	5.664 **(3%)**
IS	44.110	925 **(2%)**	1.465 **(3%)**	1.483 **(3%)**	1.186 **(3%)**	1.383 **(3%)**	1.517 **(3%)**	1.414 **(3%)**	1.485 **(3%)**
LU	180.219	4.754 **(3%)**	5.708 **(3%)**	5.799 **(3%)**	5.649 **(3%)**	5.647 **(3%)**	5.593 **(3%)**	7.254 **(4%)**	6.303 **(3%)**
MG	58.810	1.486 **(3%)**	1.889 **(3%)**	1.553 **(3%)**	1.687 **(3%)**	1.876 **(3%)**	1.944 **(3%)**	1.896 **(3%)**	1.857 **(3%)**

Table 22. Total instruction count per million (× 10^6) and per core count (8 – 15) for each NASA NAS application.

Application	Total of instructions	Core 8	Core 9	Core 10	Core 11	Core 12	Core 13	Core 14	Core 15
CG	49.382	1.392 **(3%)**	1.623 **(3%)**	1.518 **(3%)**	1.538 **(3%)**	1.699 **(3%)**	1.484 **(3%)**	1.580 **(3%)**	1.400 **(3%)**
FT	177.709	4.423 **(2%)**	5.769 **(3%)**	5.901 **(3%)**	5.814 **(3%)**	5.858 **(3%)**	5.793 **(3%)**	5.583 **(3%)**	5.806 **(3%)**
IS	44.110	991 **(2%)**	1.587 **(4%)**	1.460 **(3%)**	1.455 **(3%)**	1.462 **(3%)**	1.454 **(3%)**	1.448 **(3%)**	1.477 **(3%)**
LU	180.219	3.881 **(2%)**	5.419 **(3%)**	5.259 **(3%)**	5.196 **(3%)**	5.271 **(3%)**	6.393 **(4%)**	5.183 **(3%)**	5.218 **(3%)**
MG	58.810	1.644 **(3%)**	1.786 **(3%)**	1.809 **(3%)**	1.902 **(3%)**	1.991 **(3%)**	1.849 **(3%)**	1.888 **(3%)**	1.912 **(3%)**

Table 23. Total instruction count per million (× 10^6) and per core count (16 – 23) for each NASA NAS application.

Application	Total of instructions	Core 16	Core 17	Core 18	Core 19	Core 20	Core 21	Core 22	Core 23
CG	49.382	1.326 **(3%)**	1.601 **(3%)**	1.708 **(3%)**	1.594 **(3%)**	1.706 **(3%)**	1.621 **(3%)**	1.605 **(3%)**	1.490 **(3%)**
FT	177.709	4.554 **(3%)**	5.920 **(3%)**	5.852 **(3%)**	5.842 **(3%)**	5.698 **(3%)**	5.725 **(3%)**	5.556 **(3%)**	5.786 **(3%)**
IS	44.110	1.011 **(2%)**	1.531 **(3%)**	1.511 **(3%)**	1507 **(3%)**	1.229 **(3%)**	1.521 **(3%)**	1.634 **(4%)**	1.415 **(3%)**
LU	180.219	4.132 **(2%)**	4.819 **(3%)**	5.516 **(3%)**	5.730 **(3%)**	5.204 **(3%)**	5.224 **(3%)**	5.171 **(3%)**	5.245 **(3%)**
MG	58.810	1.708 **(3%)**	1.945 **(3%)**	1.987 **(3%)**	1.925 **(3%)**	1.954 **(3%)**	1.918 **(3%)**	1.584 **(3%)**	1.903 **(3%)**

Table 24. Total instruction count per million (× 10^6) and per core count (24 – 31) for each NASA NAS application.

Application	Total of instructions	Core 24	Core 25	Core 26	Core 27	Core 28	Core 29	Core 30	Core 31
CG	49.382	1.382 **(3%)**	1.604 **(3%)**	1.604 **(3%)**	1.219 **(2%)**	1.568 **(3%)**	1.484 **(3%)**	1.508 **(3%)**	1.582 **(3%)**
FT	177.709	4.417 **(2%)**	5.606 **(3%)**	5.459 **(3%)**	5.481 **(3%)**	5.309 **(3%)**	5.595 **(3%)**	5.581 **(3%)**	5.504 **(3%)**
IS	44.110	877 **(2%)**	1.471 **(3%)**	1.432 **(3%)**	1.166 **(3%)**	1.362 **(3%)**	1.468 **(3%)**	1.498 **(3%)**	1.281 **(3%)**
LU	180.219	5.737 **(3%)**	7.093 **(4%)**	6.087 **(3%)**	6.223 **(3%)**	6.270 **(3%)**	6.319 **(4%)**	6.603 **(4%)**	6.322 **(4%)**
MG	58.810	1.576 **(3%)**	1.940 **(3%)**	1.875 **(3%)**	1.949 **(3%)**	1.901 **(3%)**	1.935 **(3%)**	1.861 **(3%)**	1.879 **(3%)**

Table 25 displays the active power dissipation of all cores running the PARSEC's benchmark. The output produced by Gem5 (stats.txt) is fed through McPAT [HP15] using a conversion tool [RIC15] to generate these results. McPAT uses some internal elements of the core to produce its overall power dissipation, such as L1 caches, branch predictor, pipeline units and memory management unit.

The previously identified benchmarks that expressed a master-like structure (Blackscholes, Bodytrack, Canneal, and Fluidanimate) also show the predicted master core in regards to consuming the largest active power. Dedup, Swaptions, Vips, and x264 show a proper distribution of consumption over the cores. This table also depicts how demanding a benchmark is to the system. x264 and Swaptions, for instance, have approximately the same distribution of power dissipation, but x264 has a larger overall dynamic power value than Swaptions because of its application nature (Table 18).

Table 25. Total and per core active power dissipation for each PARSEC application.

	Total	*Core 0*	*Core 1*	*Core 2*	*Core 3*	*Core 4*	*Core 5*	*Core 6*	*Core 7*
Blackscholes	12.5 W	1.37 W	1.38 W	1.37 W	2.87 W	1.37 W	1.37 W	1.37 W	1.38 W
Bodytrack	17.8 W	2.17 W	1.72 W	2.63 W	2.71 W	1.73 W	2.57 W	2.48 W	1.77 W
Canneal	15.1 W	0.74 W	0.74 W	0.75 W	9.85 W	0.76 W	0.77 W	0.76 W	0.77 W
Dedup	12.3 W	1.54 W	1.47 W	1.55 W	1.62 W	1.54 W	1.50 W	1.58 W	1.52 W
Fluidanimate	12.6 W	1.48 W	1.55 W	1.30 W	2.85 W	1.37 W	1.45 W	1.28 W	1.34 W
Swaptions	15.1 W	1.89 W	1.90 W	1.89 W	1.89 W	1.90 W	1.89 W	1.89 W	1.90 W
Vips	11.5 W	1.30 W	1.39 W	1.48 W	1.53 W	1.45 W	1.45 W	1.45 W	1.46 W
X264	20.1 W	2.19 W	2.12 W	2.65 W	2.26 W	2.73 W	2.87 W	2.63 W	2.70 W

Legend:	0W – 1W	1W – 1.3W	1.31W – 1.6W	1.61W – 2W	2.1 – 2.5W	> 2.5W

Canneal is a particular case of a single core reaching up to 9.85 W. Note that Cortex-A9, an in-order core, has a *Thermal Design Point* (TDP) of 2.5 W [PIN14], and Cortex-A15, an O3 core, has a TDP of 4W [PUR15]. These two cores are the basis of the core design employed on this work. The Exynos 5 achieves 8W TDP using both Cortex-A15 and a GPU, each using approximately half of the total TDP [PUR15]. In this work, we do not have a GPU model and let the CPU consume a higher TDP value.

6.2 Main Memory Evaluation

The main memory evaluation comprises six memory technologies: (i) DDR3, which is widely employed on desktops; (ii) Low Power Double Data Rate (LPDDR3), which is a typical mobile memory, and some emerging memory technologies; (iii) Wide I/O version 1; (iv) Wide I/O version 2; (v) High Bandwidth Memory (HBM); and (vi) Hybrid Memory Cube (HMC). Table 26 summarizes the differences of these memory technologies.

Table 26. Main memory models for evaluation.

Memory	*Clock (Real)*	*Bus Width*	*Memory Channels*
DDR3	800 MHz	64 bits	2
LPDDR3	800 MHz	32 bits	2
Wide I/O – v1	200 MHz	128 bits	4
Wide I/O – v2	266 MHz	64 bits	8
HBM	500 MHz	128 bits	8
HMC	1250 MHz	32 bits	16

The DDR3 model is based on a standard Micron chip with two memory channels and eight internal banks per rank [MIC15a]. The LPDDR3 model is also from a standard Micron chip with two memory channels and eight internal banks per rank. The LPDDR3 features lower operating voltage and power saving techniques such as temperature-compensated self-refresh [MIC15b] that allows fewer refresh rates for the memory when the internal temperature is low.

The remaining four memory types are for 3D TSV-integrated chips. TSV is at the center of one of the most significant changes to the memory interface and, consequently, to the famous Memory Wall [WUL95]. The Memory Wall is the observation of the increasingly processor/memory performance gap in at least the last 20 years. On top of that, the trend of placing more and more cores on a single chip exacerbates this gap. One way to diminish this is to increase memory bandwidth through wider interfaces. However, until now this has been very challenging because wider interfaces mean more off-chip pins and such pins are very expensive [FU14][ROG09][PAD11]. TSV eliminates such limitation by stacking dies and incorporating main memory into the chip. Hence, no extra off-chip pins are necessary.

Wide I/O is a standard memory interface that maximizes the memory bandwidth at the lowest possible power dissipation. The key is to stack multiple memory channels on top of the system and interconnect them through TSV [CAD15a]. Recently, JEDEC[9] published the second standard of Wide I/O that presents significant improvements [JED15a]. At half of the power dissipation of LPDDR3, Wide I/O can maintain the same memory bandwidth. Increasing the memory frequency, Wide I/O effectively provides more than double the baseline LPDDR bandwidth [GRE12][VIV11]. ST Ericsson confirmed 50% power reduction in the memory interconnections when Wide I/O was compared to LPDDR2 with the same bandwidth [KIM13b]. Samsung develops this technology since 2011 [SAM15b]. The first Wide I/O version has four channels of 128 bus width. The second version doubled the number of channels and halved their width. Both versions can operate at 200 and 266 MHz.

AMD and Hynix produce HBM memories, which is a technology that also exploits an enormous number of signals available with die-stacking technology to provide very high memory bandwidth. Each HBM stack accommodates eight independent memory channels, and each channel follows the traditional DDR interface with power saving techniques from LPDDR [CON14]. HBM achieves better power saving using a substantially smaller form factor and lower operating voltages than traditional DDR [TEC15b]. Figure 67 depicts how stacked HBM uses 19× less surface area than GDDR5 for the same memory amount. HBM is also a JEDEC standard [JED15b].

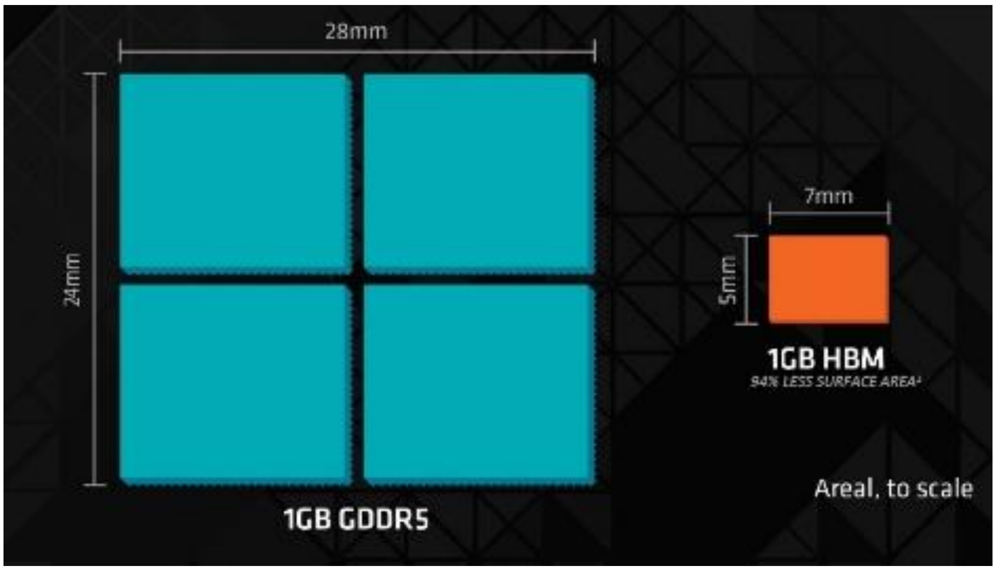


Figure 67. Surface Area of 1GB GDDR5 and HBM technologies [WCC15].

[9] JEDEC is a global leader in developing open standards for the microelectronics industry, with more than 3,000 volunteers representing nearly 300 member companies (source: www.jedec.org/about-jedec).

HMC is another memory solution that relies on TSV. While Wide I/O aims at the mobile low-power market, HMC aims at the high-performance server market. HMC achieves up to 15 times the bandwidth with 70% less energy consumption when compared to traditional DDR3 technology [CAD15b][MIC15c]. In this memory, four to eight stacks of DRAM are on top of a single logic chip responsible for data access [ALT14]. The HMC Consortium develops the HMC interface specification and promotes integration into a wide variety of systems [HMC15]. HMC utilizes closed-page policy and its DRAM devices are redesigned to have short rows (256 bytes instead of 8-16KiB in a typical DDR3 device) for high-performance computing and server workloads that exhibit little locality [AZA14].

Azarkhish [GEM15m] is working on a highly accurate model of HMC for Gem5. However, this work does not use this model as it is not fully developed. As such, we employed HMC as a DRAM model with lesser details of the memory controller intercommunication and latency. Azarkhish et al. [AZA14] also note that PARSEC, and others current multi-threaded workloads, cannot easily utilize the vast bandwidth potential provided by HMC, mainly due to the overheads of the cache coherence mechanisms. We will also observe this effect, where more than half the L2 miss latency in the Dedup benchmark is due to on-chip congestion (Figure 71).

The Gem5's memory controller is an event-based model tailored to capture the most important DRAM timing constraints for current and emerging DRAM interfaces. Hansson et al. [HAN14] compare this model with the cycle-based DRAMSim2 and shows that the impact of the event-based model is minimal to the system-level effects of DRAM while allowing much faster simulation speeds – 7× in average.

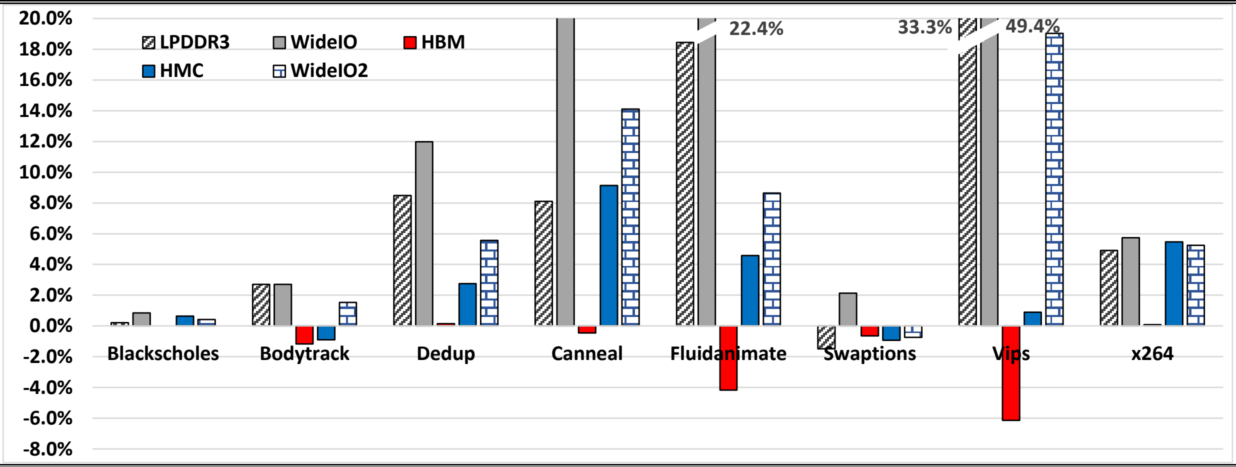
Table 27 illustrates the execution time of the applications of PARSEC benchmark according to six memory technologies. This experiment aims to explore how the memory technology affects each application execution time.

Table 27. Execution time (in seconds) of the applications of PARSEC Benchmark versus six memory technologies.

Execution time (in seconds)		Application							
		Blackscholes	Bodytrack	Dedup	Canneal	Fluidanimate	Swaptions	Vips	x264
Memory technology	DDR3	0.47	0.55	2.74	7.95	0.96	1.08	1.01	3.40
	LPDDR3	0.47	0.57	2.97	8.59	1.14	1.07	1.35	3.56
	WideIO	0.47	0.57	3.07	9.55	1.18	1.10	1.51	3.59
	HBM	0.47	0.55	2.75	7.91	0.92	1.07	0.95	3.40
	HMC	0.47	0.55	2.82	8.68	1.00	1.07	1.02	3.58
	WideIO2	0.47	0.56	2.89	9.07	1.04	1.07	1.20	3.58

Figure 68(a) and (b) illustrate the set of results of Table 27, which is normalized according to the execution time of the DDR3 module. Both Wide I/O versions show the least optimized execution time due to its overall low frequency. However, the results show that the execution time is still competitive even when compared to desktop DDR3 – the execution time for five benchmarks were lower than 20%. Combining these results with the fact that Wide I/O consumes approximately half the power of LPDDR3, this is a very interesting memory for MPSoC systems. Wide I/O achieves this performance by providing double (version 1) and quadruple (version 2) memory channels available to DDR3 and LPDDR3. The benchmarks that are impacted significantly (≥ 10%) are Canneal, Dedup, Fluidanimate, and Vips.

a)



(b)

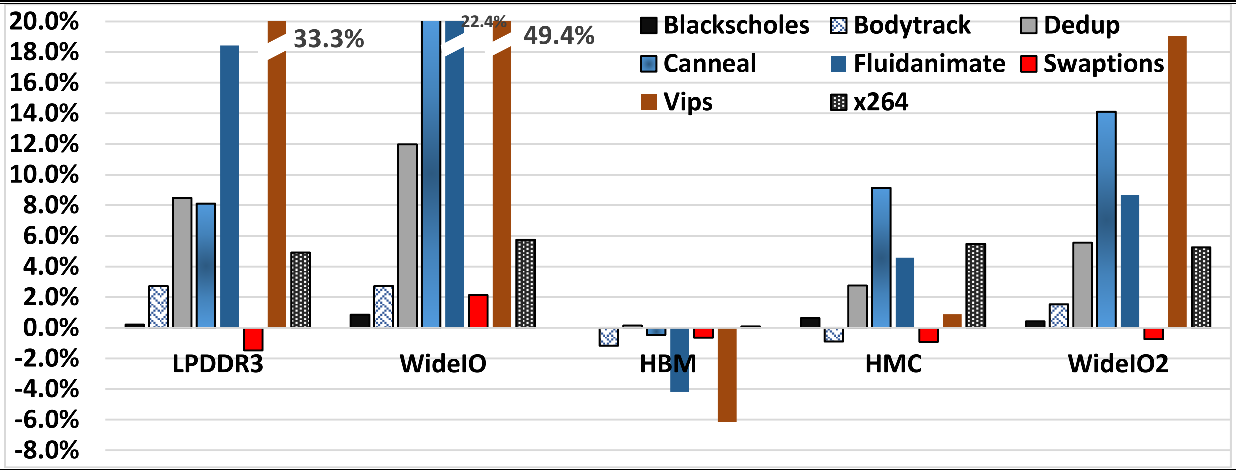


Figure 68. The execution time of eight applications versus six memory technologies. All values are relatively normalized according to the DDR3; i.e., for all applications, the execution time of DDR3 is 0% and the remaining values are perceptual deviations of this reference. Figures (a) and (b) group the values according to the application and the memory technology, respectively.

Canneal centralizes its execution in a single core (Table 20) and has little benefits from additional memory channels. Besides, due to lower DDR frequency when compared to desktop DDR3, Wide I/O suffers to provide comparable performance to sequential programs. Dedup also suffers from this lower frequency as its speedup is very limited as shown by Southern and Renau (Figure 64).

Figure 69 shows the discrepancy of achievable speedup for two extreme cases: Canneal and Swaptions, and an intermediate case: Vips. The speedup is calculated by comparing our results from Figure 68 with the sequential implementation of the same benchmark. On one hand, Canneal has the worst speedup of our set of PARSEC benchmarks (Figure 64) and achieves only up to 1.38 with our 8-core system. On the other hand, Swaptions achieves near perfect speedup with our 8-core system with all memory technologies.

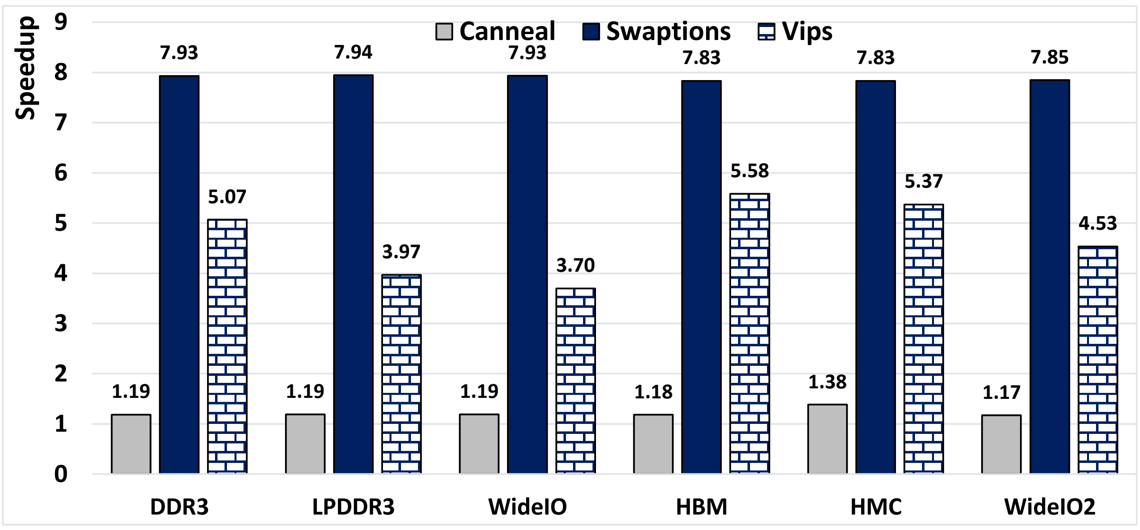


Figure 69. Speedup of some PARSEC applications (higher is better).

Figure 69 depicted the Vips speedup on our platform, which is far less than the speedup reported by Southern and Renau (Figure 64). Bienia et al. [BIE08] showed that in an 8-core organization, Vips has one of the highest L2 miss rates and follows Canneal closely on this regard, as shown in Figure 70. Considering that Southern and Renau used a bigger L2 cache (and even L3 cache), and the fact that the work set of Vips may be larger than the L2 explains the results found in Figure 68. In this regard, Vips follows the behavior of Canneal: low-power and lower frequencies memory technologies (LPDDR, Wide I/O version 1 and 2) suffers to achieve acceptable performance. LPDDR3 and Wide I/O version 1 have the worst performance for this benchmark: 33.3% and 49.4% additional execution time, respectively.

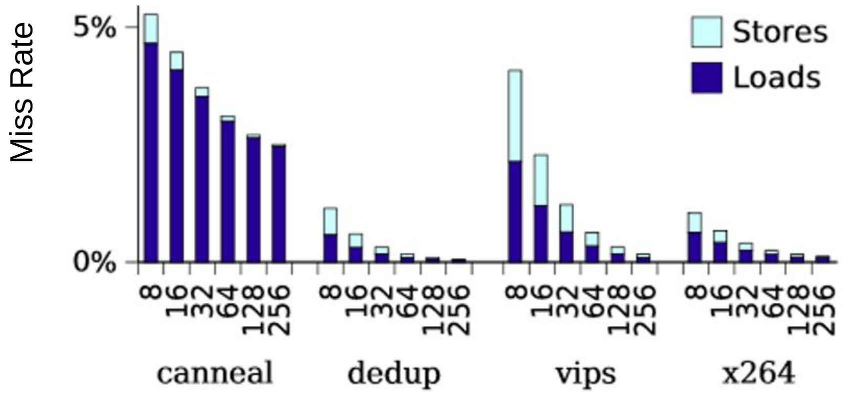


Figure 70. Miss rate as a function line size. Data executed on an 8-core system with 4MB (based on [BIE08]).

The execution time variance of the Fluidanimate is affected significantly by low frequency memories. The use of different memory technologies in this case produces execution time variances of -4.2% up to 22.4%. For this application, low-power memory technologies with lower frequencies (Wide I/O version 1 and 2) and fewer memory channels (LPDDR) increase the execution time, while more memory channels (HBM) decrease the execution time.

LPDDR is the traditional memory technology employed in low-power embedded systems such as mobile cellphones. It uses power saving techniques to reduce its overall power consumption – for instance, a desktop DDR3-1333 consumes 39 pJ/bit [ORC13] while LPDDR3 consumes 9.2 pJ/bit [ROS12]. Although our work uses the same baseline clock for both, LPDDR3 has a number of its timing parameters (tCL, tRCD, and tRAS) approximately 15% slower than its DDR3 counterpart, therefore its bank operations require more time. Additionally, 1.2V and 1.5V are provided to LPDDR3 and DDR3, respectively [MIC15a][MIC15b]. The results show that the effects of low-power for six appliations are restrained – the execution time increase is less than 10% of the normalized runtime. Once again, Vips and Fluidanimate are the most sensible benchmark due to its application nature. Swaptions is the only application that has better execution time on this memory than on DDR3. Considering that LPDDR3 is slower, this can only be attributed to a difference of temporal distribution of memory requests due to the time required to process them. In other words, for this particular case, the slower operation of LPPDR3 resulted on a better distribution of memory requests.

HMC technology serializes the IO pins of the parallel communication requiring a serial and high-speed transceiver technology. As such, HMC uses only 10% of the pin counts of DDR3 (power and ground not counted) and consumes 16 pJ/bit [SCH15b]. The Gem5 model is based on the works of Azarkhish et al. [AZA14] and uses a 32-bit bus width per memory channel. The results show that HMC performs significantly worse than DDR3 even considering its higher frequency. This can be attributed to two HMC characteristics: (i) HMC uses closed-page policy to handle memory pages recently accessed. As such, it punishes applications that have good spatial/temporal locality because they will need additional latency to open an already accessed page. For this exact reason, DDR memories have traditionally employed open-page policies, and applications, such as those present in the PARSEC suite, are developed to exploit this. (ii) The narrow bus width increases the congestion of memory requests from the application. We will show that for the Dedup application more than half of the response time of this memory is attributed to congestion on the external memory bus.

HBM continues the trend of using parallel communication and doubling the number of data and address pins employing a 1024-bit bus width across eight memory channels [KIM14]. HBM consumes approximately 6 to 7 pJ/bit [CON14] and presents a counterpoint to the performance of the HMC for the analyzed workload. HBM has the best overall performance for our workset, albeit a restricted improvement from DDR3. For Canneal and Vips, two applications that share a profile of low speedup and high L2 miss rates, HBM reduced the execution time by 0.5% and 6.1% compared to DDR3, respectively. In addition, Fluidanimate also had a decrease of execution time of 4.2%. Only two applications (Bodytrack and x264) had an increase of execution time – in both cases of the lowest possible increment: 0.1%.

Figure 71 depicts the average latency of the DRAM memory controller and L2 miss handler for the Dedup application, which has 6% up to 10% L2 miss rates across the analyzed memory technologies. The average latency of the DRAM memory controller is the arithmetic mean of all memory channels. This latency is further broken down into the following components: queue, bus, bank and the remaining components (others). The queue latency is the experienced delay for servicing each DRAM burst. As the device bus width determines the DRAM burst, increasing the device bus width results in lower queue delay, as there are fewer DRAM bursts to be executed. The bus latency is the time required to expedite a memory packet, which does not include any outside contention. Finally, the bank latency is the time spent to execute all operations related to DRAM banking. The ‘others’ describe latencies that are outside of the DRAM controller – mainly on-chip congestion. The absence of the ‘other’ in this figure does not mean that is not present – it means that it did not contribute significantly to the overall average latency. The error bars in Figure 71 represent the lowest and highest average value encountered for L2 miss rates.

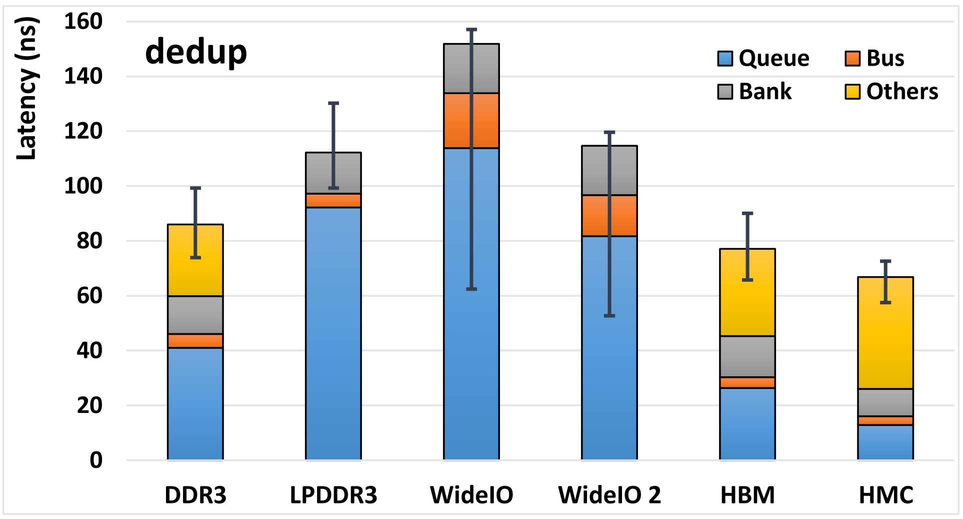


Figure 71. Average of DRAM and L2 latency for the Dedup application. Error bars represent the lowest and highest average values encountered.

From this set of results, we can see that queue delay is responsible for at least half of the DRAM latency in all cases (excluding external congestion). Three complementary ways to decrease this delay are achieved increasing the (i) operational frequency, (ii) memory channels, and (iii) bus width. The memory technologies analyzed in this work employ a selected number of these ways to achieve a balance between performance and energy consumption.

DDR3 and LPDDR3 show the effect of using 64- and 32-bit bus width, respectively. LPDDR3 has approximately 15% slower bank operations than DDR3. Overall, the LPDDR3 has higher latency penalty in all cases. HBM and HMC show the effects of increasing operational frequency and using a large number of memory channels. Conversely, HMC uses a low-bit bus width, which results in higher on-chip congestion due to longer transmission of data from DRAM to L2. HBM and HMC have lower average latency when compared to DDR3, even though both consume less energy. However, they require 3D chips. Wide I/O version 1 shows the worst average latency and version 2 shows a very similar performance when compared to LPDDR3. We highlight that it is possible to avoid bank operations if a read operation finds the data in the write buffer because Gem5's memory controller buffers the write operations and drains them at a later point [HAN14]. Some user-defined variables control the threshold to start writes to the memory.

In conclusion, we analyzed the runtime execution of six memory technologies through a broad range of applications from the PARSEC benchmark intended for financial analysis, engineering, computer vision, storage processing, and media processing. We demonstrated that this set of applications ranged from -6.1% to 49.4% of the performance of a standard desktop DDR3. Canneal has the worst speedup of all the set of applications and is further complicated by the fact that the *simlarge* input set has even lower speedup than the achievable speedup of the *native* input set (Figure 64). For our evaluation, this benchmark was used to represent applications that are mostly non-parallelizable. Vips is another application that suffers to achieve sound speedup in our system. Moreover, not all memory technologies studied here are intended to give better performance when compared to DDR3. Clearly, Wide I/O is designed for ultralow-power bandwidth as it operates at lower frequencies. In this aspect, the performance achieved by it is impressive.

It is important to consider that this range of performance variance may be more or less effective depending on the type of applications running on the system. For applications that execute intensively for only a few seconds, this variance can be negligible. However, for real-time applications with strict deadlines or even applications such as media encoding, this variance is critical. Therefore, the system nature defines the impact of adopting these new types of memories. For instance, consider the following experiment: how many minutes does it take to encode a 2 hour 640×380 video using the x264 encoder under the set of memories analyzed here. Assuming the convetional 24 frames per second used in movies this would result in 172800 frames. Figure 72 depicts the speed (frames per second) of the x264 encoding process using the *simlarge* input set. In this case, extrapolating this information for the aforementioned scenario, we have the following execution times: 76.3 minutes for DDR3; 80.1 minutes for LPDDR3; 80.7 minutes for Wide I/O 1; 76.4 minutes for HBM; 80.5 minutes for HMC; and 80.2 minutes for Wide I/O 2. This experiment shows that DDR3 followed by HBM enable the faster execution for this type of application.

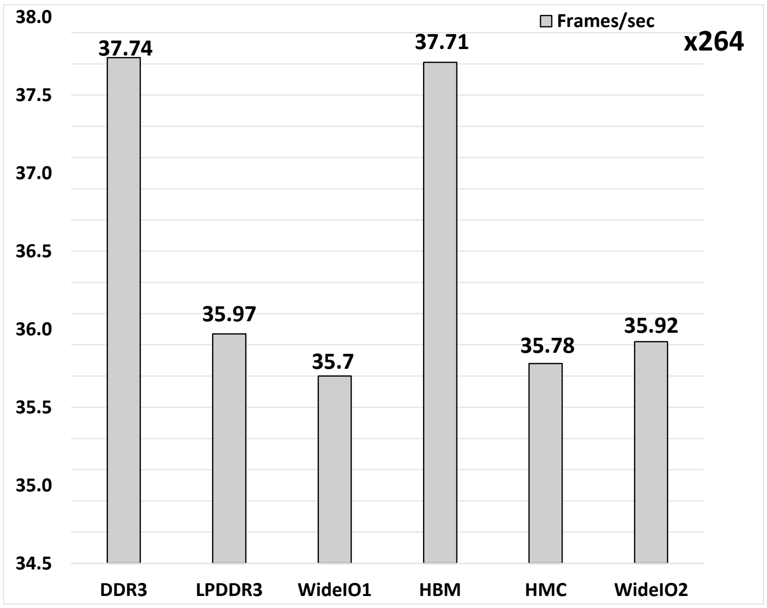


Figure 72. The x264 application encoding 128 frames of a 640×380 video employing six memory technologies under the *simlarge* input set.

6.3 Cache Evaluation

This evaluation comprises a diverse set of L2 cache configurations intended to provide distinct points in the Pareto optimality of cache resources. Two effects are analyzed in this section: execution runtime and energy consumption. As a rule of thumb, as the cache size raises, the execution runtime decreases whereas the energy consumption increases – the former and the latter being a benefit and a drawback, respectively. Besides, as shown in Section 2.1, raising the cache size has other adverse effects on the cache design.

Table 28 defines four organizations of L2 cache restricted to a total of 1MB that differ from the baseline (also described in Table 19) in the way they are distributed and allocated to the system. The baseline (**Shared L2**) is a centralized cache that mixes both instructions and data. The cache entitled **Shared L2 I+D** halves the memory space for instructions and data, dividing the connected components and possibly decreasing congestion. Specifically, L1 instruction cache and Instruction TLB are connected to the Shared L2 I and the corresponding L1 data cache and data TLB are connected to the Shared L2 D. Two cache organizations are defined on the same principles of the shared organization discussed here but are privately accessed by a single core and distributed across the chip, which are entitled as **Private L2** and **Private L2 I+D**. Finally, a paired cache organization in a similar fashion as the proposed L2 cache from SPARC M7 (Figure 10) is experimented upon. Here, we employ a simplified version that pairs two cores only (Core 0 and Core 1, Core 2 and Core 3, and so on). This last cache organization is entitled of **Paired Shared L2**.

The cache designs also differ in the hit latency and MSHR queue size. This affects, respectively, the response time and the depth of the outstanding misses provided by the L2 cache. Conservative values are used in these designs as researchers have shown that more aggressive values are possible [WOO10].

Table 28. Cache parameters for baseline and four L2 cache organizations.

	Shared L2 (baseline)	*Shared L2 I+D (2 caches/cluster)*	*Private L2 (8 caches/cluster)*	*Private L2 I+D (16 caches/cluster)*	*Paired Shared L2 (4 caches/cluster)*
Size	1 MiB	512 KiB+512 KiB	128 KiB	64 KiB+64 KiB	256 KiB
Hit latency	12 ns	10 ns+10 ns	8 ns	4 ns+4 ns	10 ns
Associativity	16-way	16-way+16-way	16-way	16-way+16-way	16-way
MSHR queue size	16	14+14	12	8+8	14

Aiming to clarify the memory hierarchy organizations, Figure 73 shows a simplified example of 4 processor interconnected according to the 4 additional cache organizations.

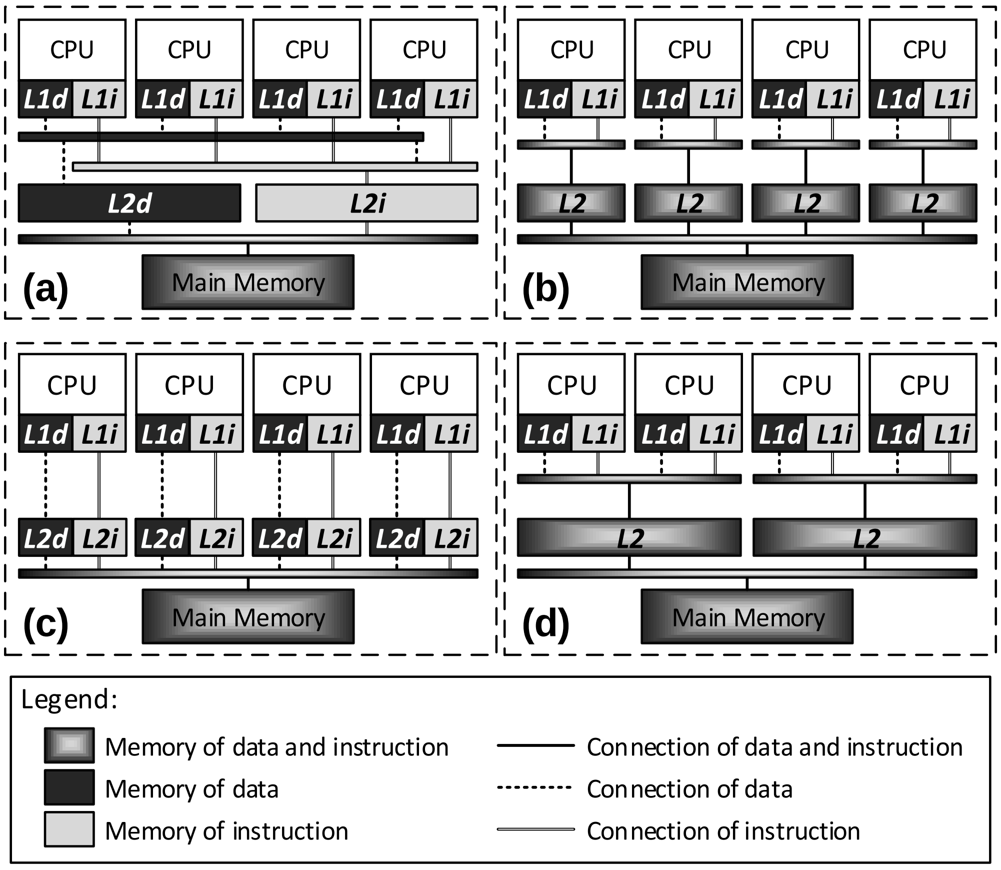


Figure 73. Example of the cache organizations containing four CPUs: (a) Shared L2 I+D; (b) Private L2; (c) Private L2 I+D; and (d) Paired Shared L2.

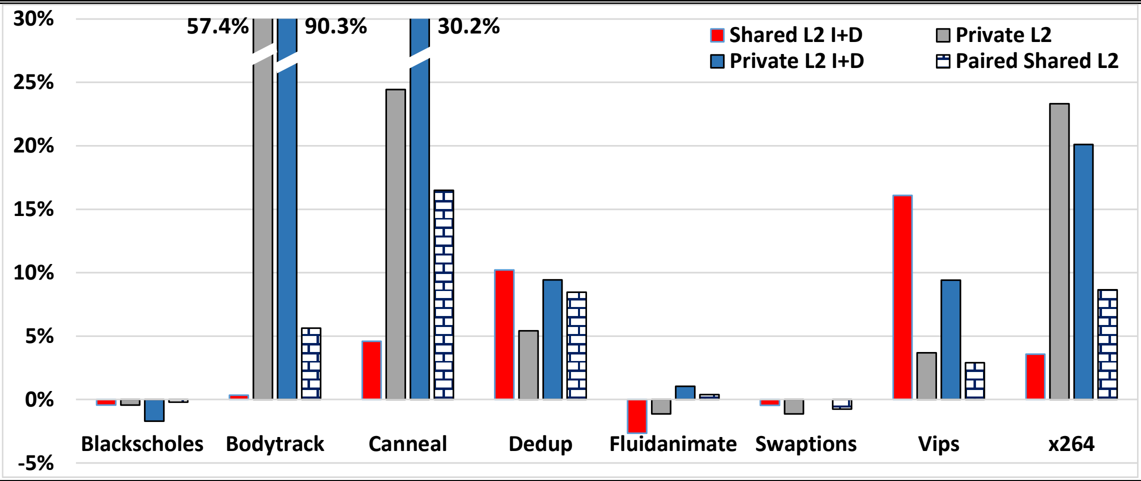
We repeat the same applications used in Section 6.2 for the execution time evaluation of L2 cache organization. The main memory remained unchanged to not influence the application execution. The Wide I/O version 2 shown in Table 26 is employed running at 266 MHz with 8 memory channels. All memory organizations share the same L1 caches described in Table 19. Table 29 shows the execution time of eight applications of PARSEC Benchmark according to a baseline and four other cache organizations. This experiment aims to explore how the cache organization affects each application execution time.

Table 29. Execution time (in seconds) of eight applications of PARSEC Benchmark according to a baseline and four other cache organizations.

Execution time (in seconds)		Application							
		Blackscholes	Bodytrack	Canneal	Dedup	Fluidanimate	Swaptions	Vips	x264
Cache organization	Baseline	0.47	0.57	9.06	2.90	1.06	1.07	1.17	3.58
	Shared L2 I+D	0.47	0.57	9.48	3.19	1.03	1.06	1.36	3.71
	Private L2	0.47	0.89	11.28	3.05	1.05	1.05	1.21	4.42
	Private L2 I+D	0.46	1.08	11.80	3.17	1.07	1.07	1.28	4.31
	Paired Shared L2	0.47	0.60	10.56	3.14	1.06	1.06	1.20	3.90

Figure 74 (a) and (b) illustrate the set of results of Table 29, which is normalized according to the Baseline cache execution time.

(a)



(b)

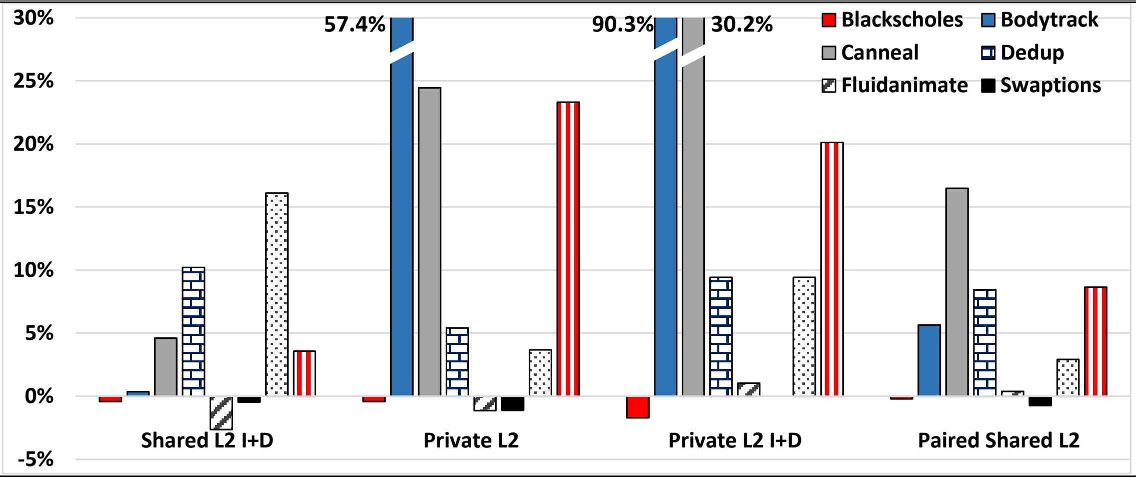


Figure 74. The execution time of eight applications according to four cache organizations (lower is better). All values are relatively normalized according to the baseline cache; i.e., for all applications, the execution time of the baseline cache is 0% and the remaining values are perceptual deviations of this reference. Figures (a) and (b) group the values according to the application and the cache organizations, respectively.

The first significant observation from these results is the fact that no other cache organization executes substantially faster than the baseline cache. Blackscholes, Fluidanimate and Swaptions are the cases where some cache organization executes faster, however, for a limited amount (less than 3%). For the remaining five applications, the new organizations are up to 90% slower, as in the case of Bodytrack.

To better understand how these cache organizations are affecting the performance of the applications, we plot the L2 cache miss rate for all cache organizations for three applications: Blackscholes, Bodytrack, and x264. Using these three applications, we have a representative set to show the effects of cache organization. They represent, respectively: a simple application with scarce communication; an application with high synchronization barriers that limits its achievable speedup and; highly parallel application.

Table 30 depicts the L2 miss rates for the Blackscholes application. The baseline cache has a 4.04% miss rate while the miss rate of the shared L2 I+D is near zero (0.18%) and 10.87% for instructions and data, respectively. This increase of data miss rate, and near zero instruction miss rate implies that the instruction cache size is overestimated. Both private L2 and paired shared L2 have an increase of miss rates, except Core 4. From these results, we can deduce that the thread assigned for Core 4 is pinned to it during the entire execution, which is corroborated by the results from the private L2 I+D cache.

Table 30. L2 miss rates for the Blackscholes application.

	Core 0	Core 1	Core 2	Core 3	Core 4	Core 5	Core 6	Core 7		
Blackscholes	4.04 %									Baseline
	0.18 %								I	Shared L2
	10.87 %								D	
	16.23 %	14.82 %	16.35 %	14.02 %	3.07 %	12.14 %	11.14 %	12.38 %		Private L2
	50.78 %	45.05 %	47.67 %	47.33 %	0.34 %	34.39 %	33.21 %	50.48 %	I	Private L2
	20.08 %	19.51 %	20.49 %	18.07 %	20.49 %	17.57 %	16.71 %	17.95 %	D	
	12.52 %		11.47 %		3.27 %		8.78 %			Paired Shared L2

The private L2 I+D cache also shows a remarkable phenomenon that affects all applications analyzed in this work. In this cache design, we observe instructions miss surpassing the 40% rate. Recalling Table 18, even though we set PARSEC parameters the intent of using an equal number of threads and cores, it internally uses additional threads. In this case, Blackscholes employ one additional thread, which cannot be pinned to a single core and must hop around the available cores and thrashing the cache data with its working set. This thrashing affects very negatively the performance of private L2 caches for parallel applications. In the particular case of Blackscholes, Table 29 and Figure 74 demonstrate that execution time is not affected even with exorbitant L2 miss rates because Blackscholes is the simplest applications of the PARSEC's suite.

Table 31 shows that Bodytrack L2 miss rates are similar to Blackscholes. Once again, the baseline cache has a low L2 miss rate (i.e., 1.03%). The shared L2 data cache has approximately the same miss rate as the baseline while the instruction cache has near zero miss rate (0.15%). Conversely, the private L2 I+D has thrashing of data information instead of instructions as was the case of Blackscholes.

Table 31. L2 miss rates for the Bodytrack application.

	Core 0	Core 1	Core 2	Core 3	Core 4	Core 5	Core 6	Core 7		
Bodytrack	1.03 %									Baseline
	0.15 %								I	Shared L2
	1.39 %								D	
	31.03%	30.85%	30.72%	31.57%	31.50%	31.00%	30.72%	30.88%		Private L2
	2.22%	1.45%	2.22%	1.95%	3.54%	1.99%	1.38%	1.14%	I	Private L2
	61.12%	61.00%	61.00%	61.03%	59.56%	60.11%	60.86%	61.21%	D	
	3.89 %		3.32 %		5.98 %		3.17 %			Paired Shared L2

Bodytrack has a significant amount of data sharing. Luo, Li, and Ding [LUO14] showed that Bodytrack can share up to 10% of the cache size across all cores – in the case studied by them, this means that 10% of data is shared by 8 cores, which is the highest sharing encountered when 7 applications were compared (Canneal, Fluidanimate, Streamcluster, Facesim, Blackscholes, Dedup, and Bodytrack). For privately accessed caches, this phenomenon can degrade performance considerably. The use of an MSI-like coherence protocol (as it is done in our system) can result in two effects [IBM15]: copy & invalidate operation on a store instruction (i.e., one cache has its line invalidated and another gains the exclusive/owned access to that line) and a “ping-pong” effect of two caches writing to the same cache line. Both events result in increased latency to exchange cache lines. An additional execution with increased data size (64KiB to 128KiB) and decreased instruction size (64KiB to 16KiB) was done to assert the effect of data sharing in this application. The average data miss was impacted significantly – falling from 61% to 28%.

The Bodytrack results from Table 29 and Figure 74 depict a different scenario than the one presented in the Blackscholes results. The L2 miss rate increase and the data sharing discussed earlier rises the execution time significantly. The private L2 cache organization showed 58% and 90% raises for the unified and I+D organizations, respectively. The shared L2 I+D shows approximately the same execution time as the baseline cache – corroborated by the miss rates from Table 31. The paired shared L2 organization increases 0.5% the execution time.

Table 32 shows the x264 application analysis. Again, from Table 18, we know that a vast number of threads are created for this application. The baseline cache has 12.75% of miss rate. The Shared L2 I+D has the lowest instruction miss rate of all applications (0.03%) and increases to 25.54% the data miss rate. As in the case of the Blackscholes, yielding up the instruction size for data size would benefit the execution time. The private L2 I+D organization again shows high miss rate for both instruction and data, which are in average 25.04% and 37.69%, respectively. The private L2 organization presents a miss rate between the instruction and data caches of the private L2 I+D organization. Finally, the paired shared L2 cache shows a better miss rate than the shared L2 D (i.e., 21.83%, in average).

Table 32. L2 miss rate for the x264 application.

	Core 0	Core 1	Core 2	Core 3	Core 4	Core 5	Core 6	Core 7		
x264	12.75 %									Baseline
	0.03 %								I	Shared L2
	25.54 %								D	
	33.40%	34.13%	31.60%	33.21%	33.15%	31.74%	33.83%	33.21%		Private L2
	25.23%	24.90%	25.06%	26.00%	25.27%	23.79%	24.95%	25.12%	I	Private L2
	37.47%	38.57%	37.97%	39.38%	36.15%	37.12%	37.09%	37.73%	D	
	22.20 %		21.74 %		22.15 %		21.24 %			Paired Shared L2

The shared L2 I+D and paired shared L2 have an increase of 4% and 9% from the baseline cache for the x264 application, respectively. Private L2 I+D and private L2 have an increase of 23% and 20% from the baseline cache, respectively. These results match the findings of increased L2 miss rate from Table 32.

The remaining applications of Figure 74 had the following results. Canneal and Vips once again had more than 10% of performance variance. For Canneal, private L2 organization had the worst execution time; i.e., 24% and 30% of increase when comparing the unified and I+D schemes with the baseline, respectively. The paired shared L2 organization also surpass the 10% mark, increasing performance by 16% compared to the baseline. For the Vips application, only the shared L2 I+D surpassed the 10% mark – also increasing performance by 16% compared to the baseline. The variance of all other cache organizations is under 10%. Three cache organizations are approximately at the 10% mark for Dedup application – shared L2 I+D (10%), L2 private I+D (9%), and paired shared L2 (8%). The L2 private designs showed the best execution time of the new cache designs, albeit it is still worse compared to the baseline.

All organizations of cache have approximately the same performance for the Fluidanimate and Swaptions applications. This behavior was also observed for the main memory evaluation (Figure 68).

McPAT uses predefined memory models and some statistic files generated by Gem5 simulation to estimate the energy consumption and area occupation of all cache organizations. We employed SRAM cells for all the caches with 32nm technology and conservative interconnect technology projection. McPAT evaluates this type of memory taking into account the dynamic energy consumption and the static energy consumption.

The dynamic energy consumption covers the energy consuming for readings and writings accesses of Equation (1); i.e. $n_{read} \times Energy_{read} + n_{write} \times Energy_{write}$. The static energy consumption comprises two types of static power dissipated due to circuit leakage [LI09]: subthreshold leakage and gate leakage. Subthreshold leakage occurs when a transistor is supposedly turned off but allows a small current to pass through its source and drain. Gate leakage is current that leaks through the gate terminal. The sum of these to static powers composes the $Power_{AVG}$ of Equation (1). Aiming to compare dynamic and static energy parcels, the experiments illustrate both parcels as average power dissipations.

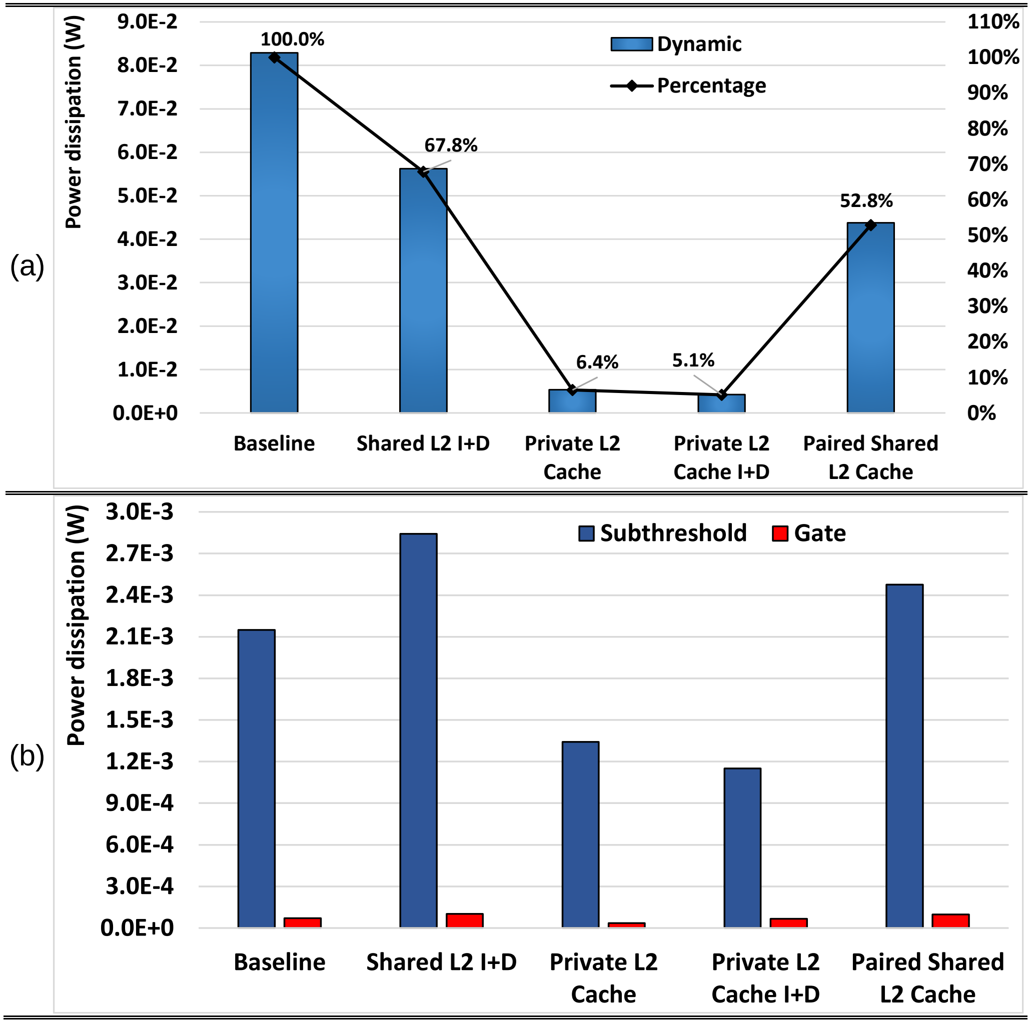


Figure 75. (a) Dynamic and (b) static power dissipation (subthreshold and gate leakage) for all cache organizations evaluated.

Figure 75(a) and (b) illustrate the dynamic and static power dissipation of the baseline and some organizations of cache using the parameters established in Table 28. The comparison is done summarizing all caches of a single cluster. For instance, for the private

L2 cache, the subthreshold and gate leakage of 8 caches are summed up. Note that all caches have the same 1MB size when summed up.

Figure 75(a) depicts that the baseline dissipates more dynamic power than all other cache organizations; e.g. the privative cache organizations dissipate almost 20 times less dynamic power than the baseline. This is an exciting aspect in contraposition of the notorious reduction of execution time when using the baseline cache organization. Besides, Figure 75 (b) shows that only the private cache organization have a lower static energy consumption than the baseline cache. The baseline cache consumes 60% and 100% more subthreshold and gate leakage than the private L2 organization, respectively. On the other hand, this changes to 87% more and 5% less subthreshold and gate leakage than the private L2 I+D organization, respectively. For the remaining two cache organization, shared L2 I+D and paired shared L2, they consume more static energy than the baseline for both cases.

Figure 76 depicts the area consumption of the entire L2 system. The attained values for individual caches are 25.92 mm² for the baseline cache design; 15.35 mm² for one shared L2 I+D; 1.15 mm² for one private L2; 0.73 mm² for one private L2 I+D and; 7.95 mm² for one paired shared cache. These values are within the expected range because considering a rough assumption of halving the cache size results in halving the area consumption, the results have the following disparity: 2.38 mm² (+18%), -2.08mm² (-65%), -0.88 mm² (-55%), 1.47mm² (22%). Clearly, there are a lot of extra factors influencing the attained area, such as floorplanning design, MSHR queue size, writeback buffer size, and so on.

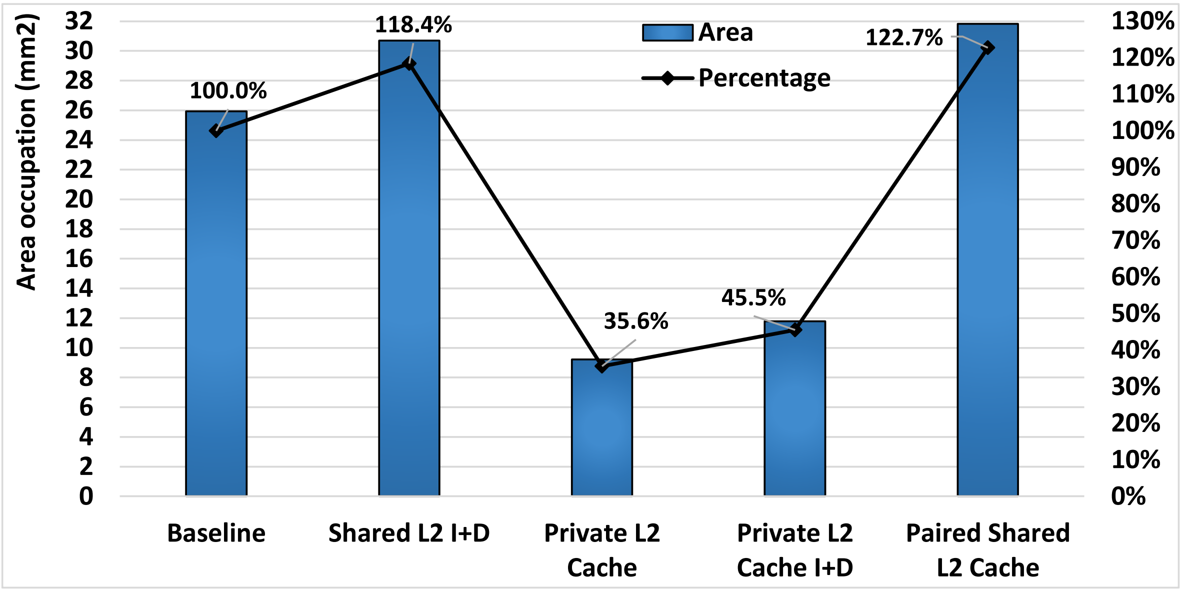


Figure 76. Area occupation (in mm^2) for all cache organizations.

The experiments show that only the private cache organization diminish the area consumption when compared to the baseline. Both private caches consume under 50% of the baseline. The fact that shared L2 I+D cache consumes more area is not surprising since it essentially doubles the amount of logic required to control the same cache size. The paired shared design needs four caches in our cluster, and each one has additional area requirements described earlier. The result is that this cache organization consumes 22% more area than the baseline.

In this section, five cache designs were analyzed. Three of them based on sharing information among cores and two of them privately accessed. All caches have uniform access models. Three types of evaluations were employed: execution runtime, energy consumption, and area consumption. We replay the movie analysis done in Section 6.2 to give a perspective of scale for these results. Figure 77 depicts the frames per seconds for all cache organizations. Feeding the same movie for all cache configurations we have the following results: 80.5 minutes for the baseline design; 83.4 minutes for the shared I+D design; 99.3 minutes for the private design; 96.7 minutes for the private I+D design and; 87.5 minutes for the paired shared design.

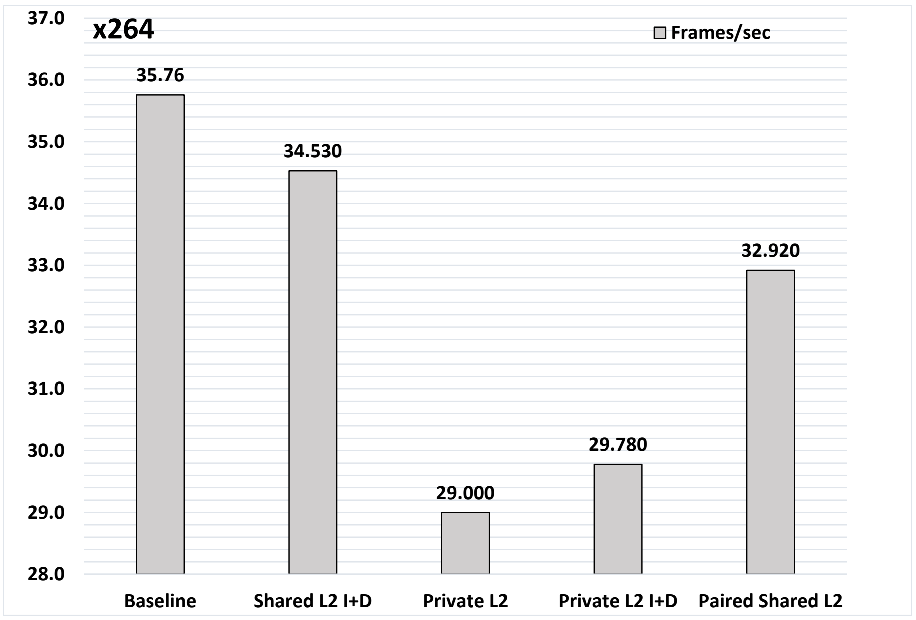


Figure 77. The x264 application encoding 128 frames of a 640×380 video employing five cache organizations under the *simlarge* input set.

6.4 Scalability Evaluation

This section describes the scalability of integrating multiple clusters of cache coherent systems over a NORMA communication model. The NORMA model uses a hierarchal packet-based NoC for interconnection [MAT11]. Each cluster employs a crossbar to interconnect all cores to the NoC router. We implemented some sizes of clusters under the restriction of 32 cores to evaluate the impact of mixing UMA and NORMA memory models.

As stated in Section 5.3, Gem5's ARM ISA does not provide a packet-based NoC. Gem5's NoC requires the use of the *Ruby* memory system that, at the time of writing, is restricted to the ALPHA and x86 ISAs. Therefore, we implemented a NoC model over Gem5, which operates at the same frequency of the cores (1GHz) and can send one flit every cycle. To model a packet-based congestion, we employ a normal probability distribution, which satisfies the following properties [TRI11]: 68% of the observations fall within 1 standard deviation; 95% of the observations fall within 2 standard deviations and; 99.7% of the observations fall within 3 standard deviations. Previous packet-based NoC work were analyzed [CAT14][FER15] for appropriate values of mean and standard deviation. The values are 100 ns and 50 ns for mean and standard deviation, respectively.

We analyzed four sizes of clusters for scalability proposes considering a target MPSoC with 32 CPUs: (i) 4 clusters with 8 CPUs, which was already used in Sections 6.2 and 6.3. (ii) 8 clusters with 4 CPUs; (iii) 16 clusters with 2 CPUs and (iv) 32 clusters containing a single CPU, which models a pure packet-based system. Table 33 depicts the four different sized clusters and its parameters that will be discussed shortly.

Table 33. Four different sized clusters and its cache parameters.

Cluster	*Memory channels*	*Cache*	*Parameter*	*Values*
4×8-core clusters	2 for each cluster (8 total)	Shared L2 (per cluster)	Size	1MiB
			Hit Latency	12 ns
			Associativity	16-way
			MSHR queue size	16
8×4-core clusters	1 for each cluster (8 total)	Shared L2 (per cluster)	Size	512 KiB
			Hit Latency	10 ns
			Associativity	16-way
			MSHR queue size	14
16×2-core clusters	1 for each cluster (16 total)	Shared L2 (per cluster)	Size	256 KiB
			Hit Latency	10 ns
			Associativity	16-way
			MSHR queue size	14
32×1-core clusters	1 for each cluster (32 total)	Shared L2 (per cluster)	Size	128 KiB
			Hit Latency	8 ns
			Associativity	16-way
			MSHR queue size	12

Aiming to clarify the cluster organization, Figure 78 to Figure 81 show the logical organization of the four cluster architectures analyzed in this work. The packet-based NoC is organized as follows for each cluster: 2×2 for 4×8-core clusters; 4×2 for 8×4-core clusters; 4×4 for 16×2-core clusters; and 8×4 for 32×1-core clusters.

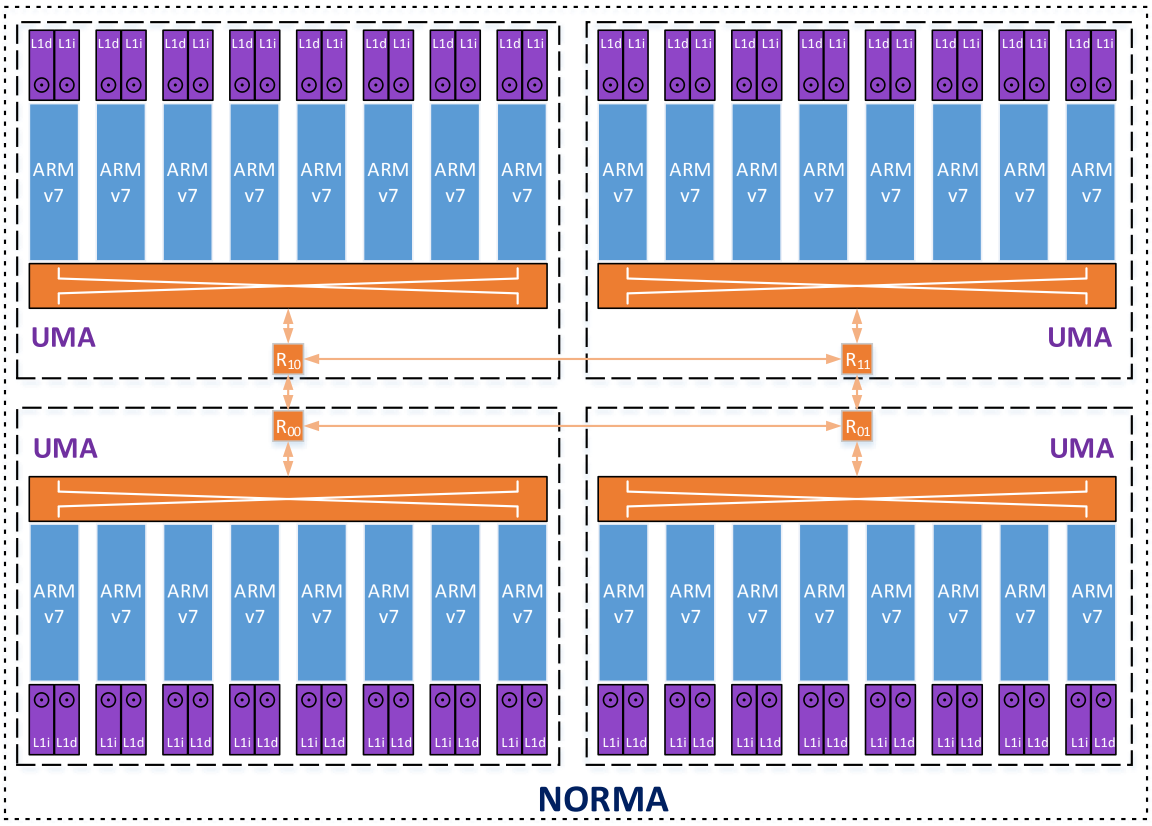


Figure 78. Logical organization of 4×8-core clusters architecture.

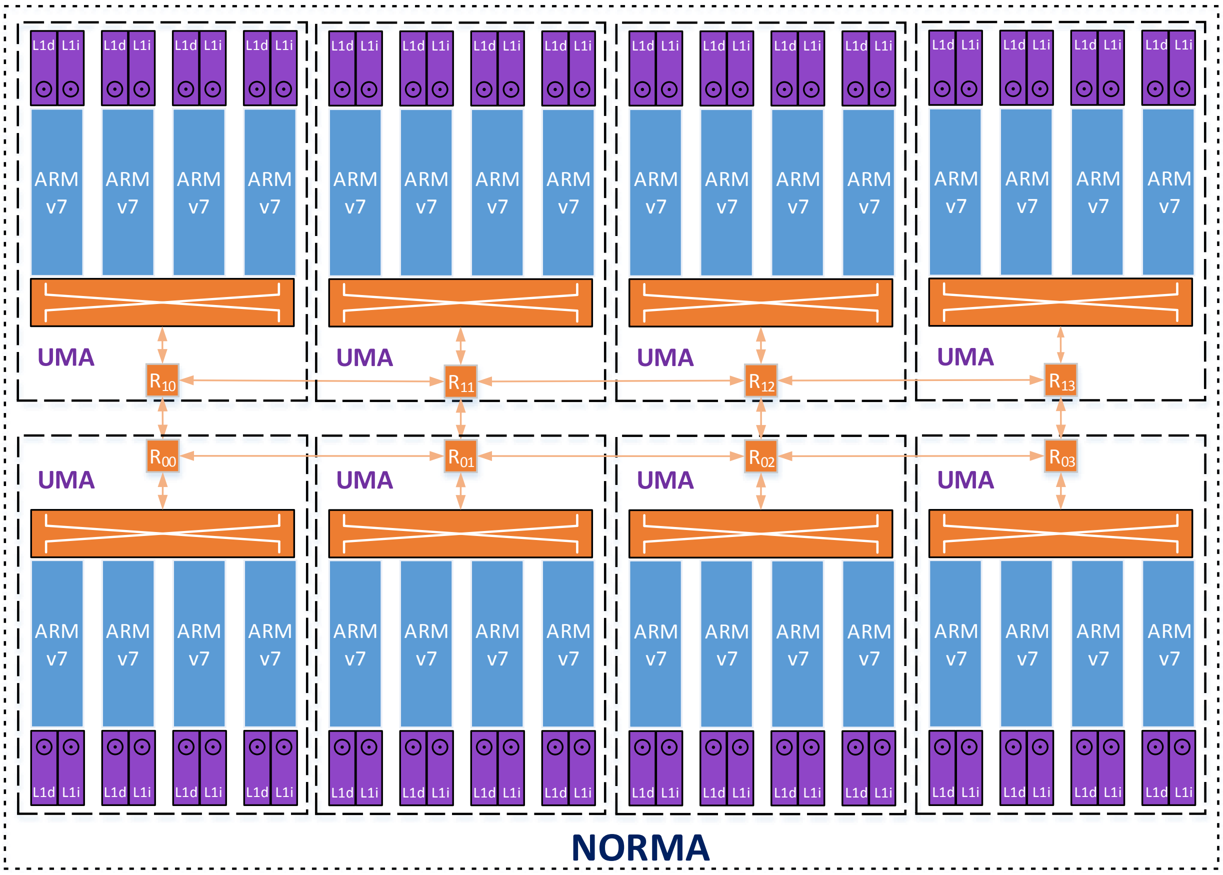


Figure 79. Logical organization of 8×4-core clusters architecture.

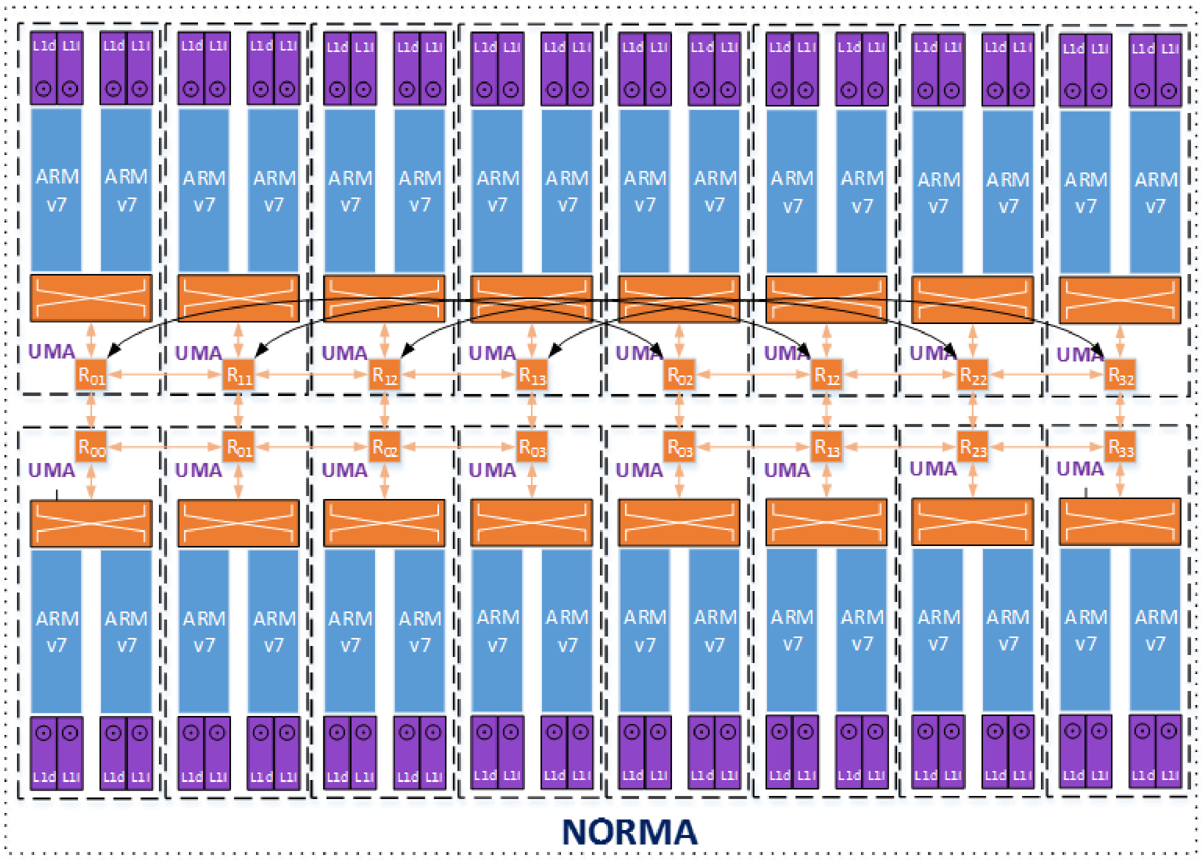


Figure 80. Logical organization of 16×2-core clusters architecture.

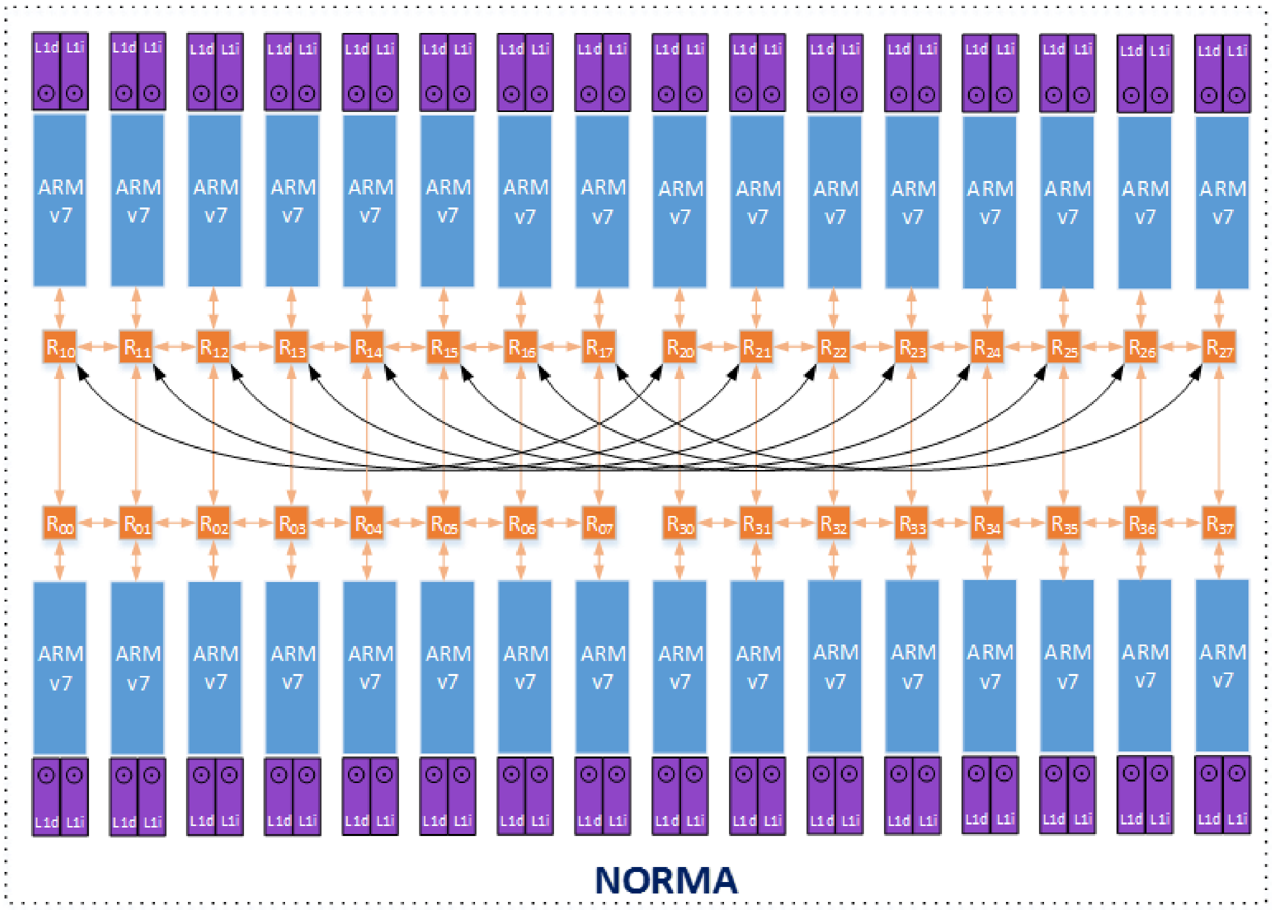

L1d L1i
ARM v7
R10 R11 R12 R13 R14 R15 R16 R17 R20 R21 R22 R23 R24 R25 R26 R27
R00 R01 R02 R03 R04 R05 R06 R07 R30 R31 R32 R33 R34 R35 R36 R37
NORMA

Figure 81. Logical organization of 32×1-core clusters architecture.

All the systems use the shared L2 cache and have the same size restriction – 4MB L2 cache. The cache parameters follow the definitions of the baseline cache and the additional caches discussed in Section 6.3. Essentially, each cluster has less space on its shared L2 caches as additional clusters are added to the system. The 1-core cluster uses the L2 cache as an extension of its L1 cache, albeit with slower access. Note that mpich2 (mpi executable) can switch to the shared memory paradigm for multicore clusters [MPI15]. Hence, the cache coherent interconnect can be used in this case instead of the crossbar.

The Wide I/O version 2 is once again employed for the system's main memory. However, due to limitations of our simulation environment, we could not model the congestion of multiple clusters accessing the same memory controller. To offset this simplification, we limited the availability of memory channels per cluster.

The evaluation workload is the NASA NAS benchmark discussed in Section 5.5. Table 34 illustrates the execution time of five applications of NASA NAS Benchmark according to the cluster organizations. The operation/second/cores is calculated automatically by the NAS benchmark suite as millions of operations per second. Larger values of operations indicate better performance [SMI05].

The IS application has the least overall operations and operations per second as shown in Table 34 and Table 35, respectively. In this regard, IS can be considered the simplest application of the NASA NAS Benchmark. IS achieves the best operations per second on the pure message passing system, allowing the 1-core system to reduce its execution time compared to all other organizations. The 2-core system has 32% fewer operations per seconds. The 4-core system has the worst execution time and operations per second from all the organizations.

Table 34. Execution time (in seconds) of five applications of NASA NAS Benchmark according to four cluster organizations.

Execution time (in seconds)		Application				
		CG	IS	FT	LU	MG
Cluster organization	4×8-core clusters	0.87	0.505	3.12	2.81	1.265
	8×4-core clusters	0.99	0.565	2.85	3.17	1.115
	16×2-core clusters	0.87	0.47	2.11	3.11	0.625
	32×1-core clusters	0.91	0.36	1.23	3.72	0.35

Table 35. Million operation/second/core of five applications of NASA NAS Benchmark according to four cluster organizations.

Million operation/second/core		Application				
		CG	IS	FT	LU	MG
Cluster organization	4×8-core clusters	53.675	5.19	71.44	200.74	96.02
	8×4-core clusters	47.345	4.655	78.24	177.81	109.115
	16×2-core clusters	53.89	5.55	105.78	181.33	194.27
	32×1-core clusters	51.23	7.3	181.03	151.68	343.9

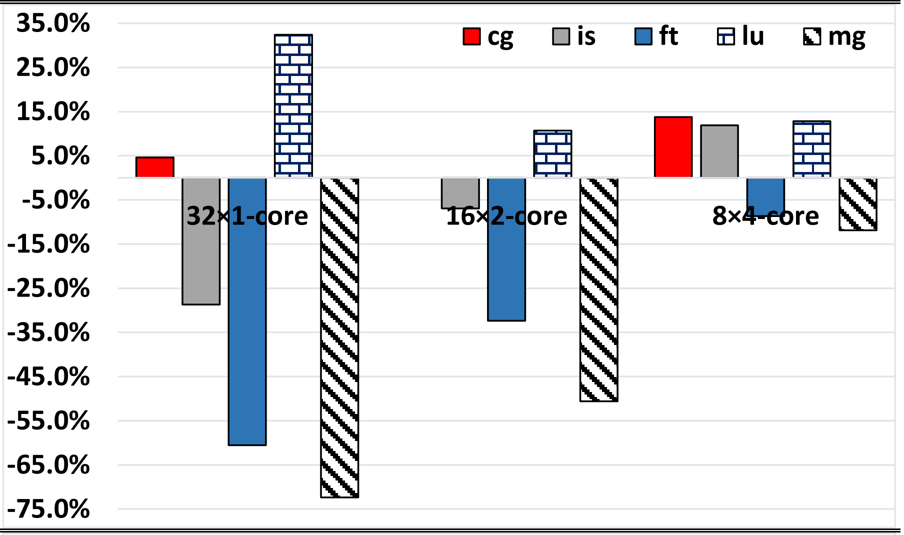
Figure 82 depicts the normalized execution time and million operations/second/cores for all applications. The FT and MG applications also show the best performance on the pure message passing system. FT aims to test long distances and MG intends to test long and short distances, as described in Section 5.5.1. They significantly reduce the execution time from 3.12 to 1.23 seconds and 1.265 to 0.35 seconds, respectively. Their operations/second/cores corroborate this behavior. Table 34 shows that both applications continually decrease execution time as the system reduces the cluster size.

Conversely, the CG application has the best performance on multicore clusters. 2-core and 8-core clusters have approximately the same performance, and they execute 13% more operations/second/cores than the 4-core system. The pure message passing system has a slightly increase of execution time from 0.87 (4×8-core) to 0.91.

The LT application has the worst performance on packet-based system for the set of applications analyzed here. This application employs a vast number of small messages. The 4×8-core clusters show the best execution time of 2.81 seconds. The 16×2-core cluster have a slightly better execution time than 8×4-core clusters at 3.11 and 3.17 seconds. The pure message passing system showed an execution time of 3.72 seconds and a reduction of 25% of operations/second/cores from the cluster with best results.

These results show that a pure NoC-based MPSoC has a remarkable performance for message passing applications. In most cases, the NoC communication exceeds the crossbar-based communication used in intra-clusters due to the parallel nature of the NoC communication provided by multiple independent routers. However, the crossbar-based communication is necessary to provide acceptable performance for shared memory applications, as was the case for Sections 6.2 and 6.3.

(a)



(b)

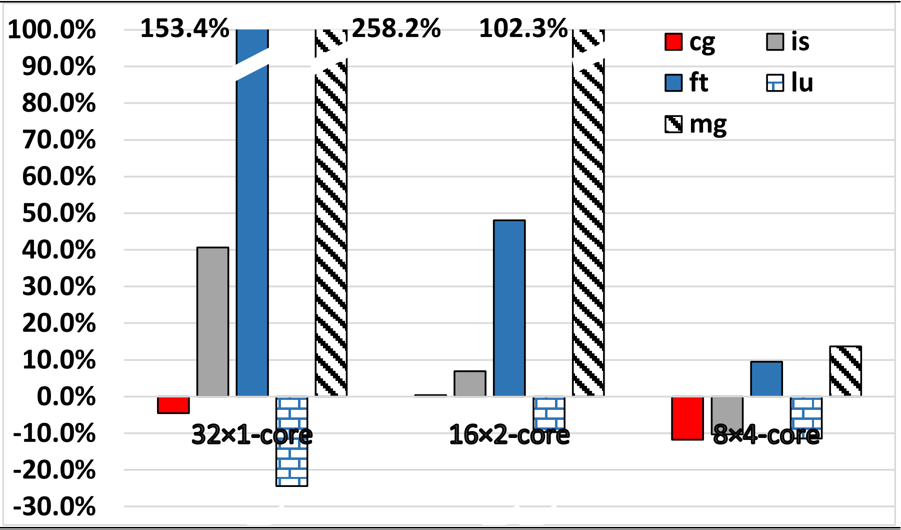


Figure 82. The execution time of five applications according to four cluster organizations. All values are relatively normalized according to the 4×8-core clusters; i.e., for all applications, the execution time of this cluster is 0% and the remaining values are perceptual deviations of this reference. Figures (a) and (b) group the values according to the application and million operations/second/cores, respectively.

7 CONCLUSION AND FUTURE WORK

The ever increasing number of PEs in MPSoC designs requires the study of memory system solutions that preserve the high bandwidth, low latency and low energy consumption expected from such systems. Besides, as the interconnect fabric becomes the predominant factor in delaying information on a chip [BEN02], the exploration of appropriate interconnect for scalability is necessary. Therefore, this work contributes in three topics related to memory organization in a cache-coherent MPSoC: (i) evaluation of main memory technologies; (ii) evaluation of cache architectures; (iii) evaluation of scalability in our proposed system. These three evaluations are based on the execution time of a set of applications executing on our system.

In this work, we are not exploring mapping techniques and, as such, employ the standard Linux kernel scheduler for this task. However, the mapping of tasks is one of the aspects that influence the execution time of applications [MAR11]. Therefore, we provide a workload evaluation that shows how the Linux kernel scheduler maps the set of applications over the available cores in terms of instruction distribution. In this way, this work can be compared to any mapping technique proposal.

The assessment of main memory technologies helps to determine if the intended performance of the newer types of memories applies to real parallel workloads. We compared four types of new memories based on TSV-enabled chips: Wide I/O version 1 and 2, HBM, and HMC. They were compared to the widely adopted desktop DDR3 and LPDDR3 [RAN15]. The execution time was measured for eight applications of PARSEC Benchmark. For four applications (Blackscholes, Bodytrack, Swaptions, and x264), the execution time variance was within the range [-10%, 10%], which is an impressive performance considering that most of the newer types of memories are focused on power saving instead of higher performance. Three of them (Wide I/O version 1 and 2, and HBM) use lower frequencies than DDR3/LPDDR3. For the Canneal and Vips applications, the execution time increased in some cases to 20% and 42%, respectively.

Canneal has the least speedup using parallel computation on the PARSEC Benchmark, which was corroborated by the work of Southern and Renau (Figure 64). Hence, since new memories employ more memory channels instead of higher frequencies to maintain performance, this type of application will behave poorly. Vips also suffered from lower frequencies as the speedup achieved in our system was underwhelming, because we used a smaller L2 cache compared to previous works of Southern and Renau (Figure 64) and Bienia et al. [BIE08]. Therefore, the L2 cache was unable to sustain the same latency for lower frequencies memories (Wide I/O version 1 and 2) and LPDDR3.

The evaluation of cache organizations is done using two aspects: execution time and energy consumption. We analyzed five organizations of caches L2: (i) standard shared L2 cache, which was employed as a baseline for comparative evaluations; (ii) shared L2 I+D cache; (iii) private L2 cache; (iv) private L2 I+D cache; and (v) paired shared cache.

In general, no cache organization had better execution time than the baseline, since our workload comprises parallel applications intended to share data across tasks. The private organizations behaved poorly regarding performance due to their smaller size available and the increase of coherence overhead of multiple L2 caches. The additional shared designs, shared L2 I+D and paired shared, behaved better but increasing up to 16% in the execution time.

Regarding static power consumption, the private cache designs were much better than all shared designs. Again, this is expected since the caches are appropriately resized to have the same size restriction, regardless of the cache design. Hence, private caches are much smaller and consumes less static power. Unfortunately, it degrades the performance of the system both in terms of execution time and coherence overhead. Of all shared designs, the traditional shared L2 cache showed the best static power. Combining the fact that the shared L2 cache also showed the best execution time, for our workload, this was the best cache design.

Yet, the baseline organization had the most consumption of dynamic power by far which was fifteen and nineteen times the consumption of the set of private L2 and private L2 I+D organizations, respectively. A similar trend occurs with the area occupation where they are decreased twenty two and thirty five times compared to the baseline for the private L2 and private L2 I+D organizations, respectively. When the shared organizations were compared to the baseline, they showed a drawback (increase of area occupation) and a benefit (decrease of power dissipation). The set of shared L2 I+D has two thirds of the dynamic power dissipation and six fifths of the area occupation compared to the baseline, approximately. The set of paired shared L2 has half of the power dissipation and six fifths of the area occupation compared to the baseline, approximately.

Therefore, the cache organizations provide compelling tradeoffs for energy consumption, execution time, and area occupation allowing the designer to optimally employ a design that suit his needs. For the lowest energy consumption, we recommend to employ privately small L2 caches. In the exact opposite of this scenario, i.e. highest performance, we recommend to employ large shared L2 caches. The additional shared L2 organization explored in this work are the middle ground of these scenarios.

Finally, the scalability of our proposed system was evaluated. Our system is intended to be comprised of two layers: cache coherent UMA clusters and NORMA multiple connected clusters. Until this evaluation, all tested were conducted using an 8-core cluster. Now, we scale different sized clusters, reaching to the 1-core cluster that represents a full NORMA system. Due to the NORMA communication nature, we had to change the benchmark suite to another compatible with the message passing paradigm. For this reason, we employed the NASA NAS benchmark. Five benchmark applications were employed. They showed that NoC-based MPSoC are capable architecture to provide excellent performance for message passing applications.

To perform such evaluations, we had to extend Gem5's current capability in two significant ways. Firstly, there was no Wide I/O version 2 model. We implemented a new model since it has important distinctions to the first version (discussed in section 6.2). Secondly, the Gem5's *classic* mode provides only the traditional shared L2 cache. We had to implement all the other cache designs. We intent to provide these two contributions to the Gem's community through the patch submission process once this work is finished.

The current work is an initial effort performed by the author to explore memory and cache design for MPSoC systems. The future work in this scope includes a deeper study of power-related metrics to employ in all evaluations and explore the full range of the benchmark suite. Specifically we would like to calibrate our power estimation tool, McPAT, to produce more accurate results. This way, we could use the full range of output information provided by it. Lee et al. [LEE15b] have shown a methodology to train McPAT to produce better results. In their work, McPAT was attuned to the ARM Cortex-A15 architecture using a number of coefficients collected by experimentally running benchmarks on real hardware. Hence, for this methodology, a real ARM system is needed.

We wish to explore the full range of applications provided by the PARSEC benchmark as every one of them represents a unique scenario. Using such approach would allow a more comprehensive comparison with other works. To offset the limitation of using a full-system simulator, we also would like to use Southern and Renau's work [SOU15a] to map each application to its appropriate use – either full simulation or ROI-limited, because, as shown by the authors, ROI can mask the real speedup achieved by the *native* (i.e., real behaviour) input set of PARSEC.

We would also like to extend the study of cache architectures to provide a better representative set. As the paired shared showed interesting results, it would be viable to extend this simpler version to the one used by the SPARC M7 (Figure 10). A deeper study of power saving techniques is needed to provide an in-depth analysis of the full potential for every cache design analyzed in this work. We restricted the power analysis to the capabilities of McPAT.

We limited our 3D study up to 32 cores and only one tier for cache organization. However, this study provides a broad range of interesting possibilities. We originally intended to provide support for L3 caches in an energy-aware architecture. Our model comprises 64 cores that follows the same principles of UMA and NORMA organization of our work but shares an L3 cache. This L3 cache uses page-coloring techniques to identify the source of information. Thus, they provide isolation for every NORMA cluster and full L3 space for a given limited set of cores running. Today, the caches of an idle cluster cannot be used by other clusters. Figure 83 shows the intended architecture.

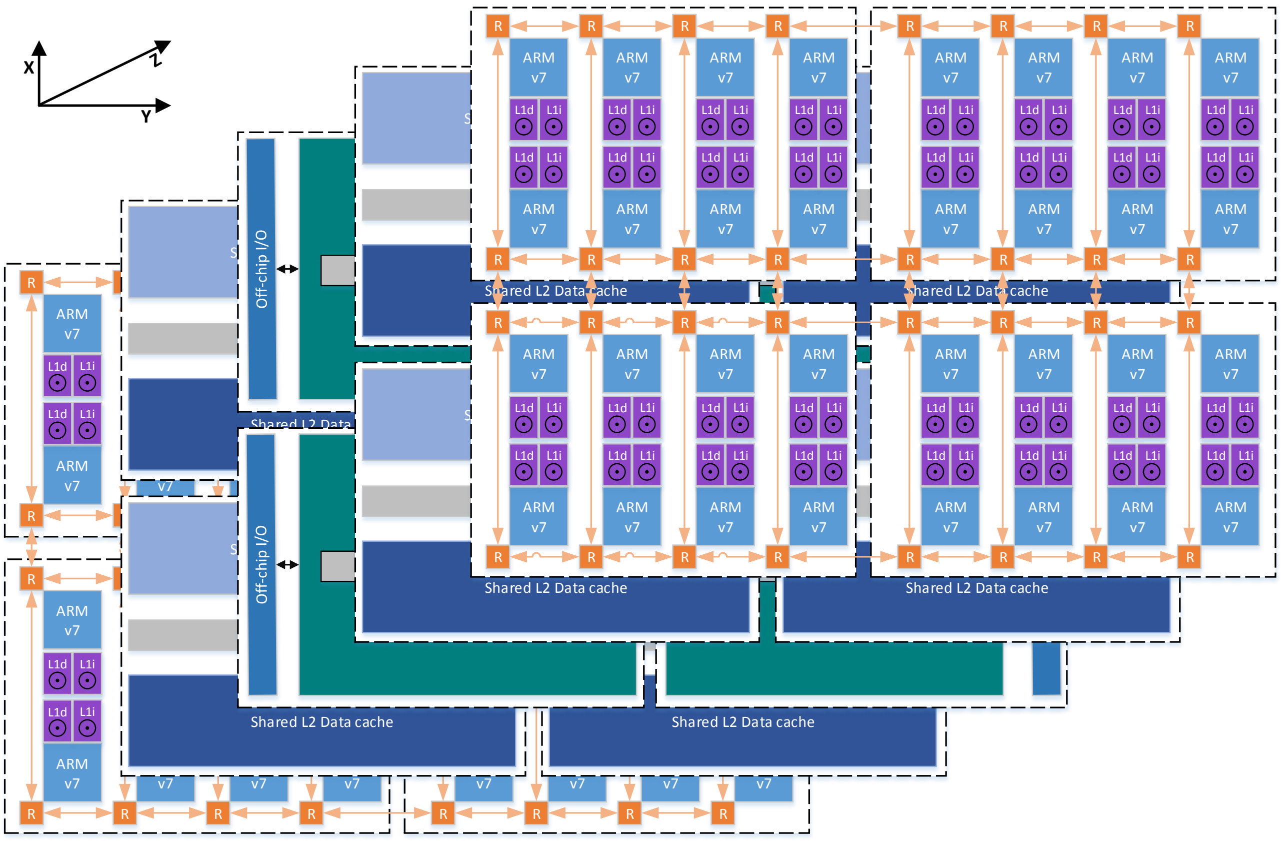


Figure 83. Example of 3D MPSoC architecture exploration with 5 tiers. The external tiers encompasses processing elements and L1 caches. The two inner tiers connected to the external tiers includes L2 caches with separate data and code addressing. Finally, the innermost tier is a L3 cache level, which mixes data and code addressing.

This figure is just one of the possibilities of architectural exploration. Our 3D system can be further extended following three axis of exploration: tiers, UMA, and NORMA. Tiers can be comprised of logic, memory or the combination of the two. UMA determines the number of cores that share a single coherent address space. Finally, NORMA determines the number of UMA clusters that are interconnected through a packet-based NoC. Figure

84 illustrated this idea and shows the explored architectures in this work (M1, M2, M3, M4) and our example of future work in M5 (Figure 83). These values do not depict TSV-connected main memories.

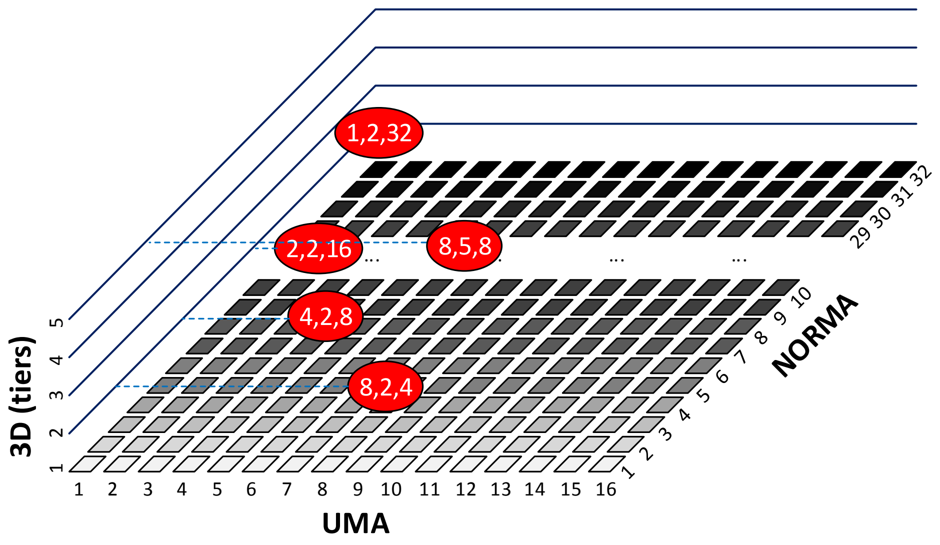


Figure 84. Architectural exploration of 3D system with some UMA, tiers and NORMA elements. The values are encoded as (UMA, 3D - tiers, NORMA).

Finally, we employed a statistical distribution to model a real NoC congestion. It has been a long desire of the community to use R*uby* on the Gem5's ISA [GEM15n][GEM15o][GEM15p]. This would be a remarkable work that would open many design explorations not possible today on the ARM ISA, such as the exploration of different coherence protocols.

8 REFERENCES

[7ZI15a] 7-Zip Team. **7-Zip LZMA Benchmark: Power8**. Available at: www.7-cpu.com/cpu/Power8.html, 2015.

[7ZI15b] 7-Zip Team. **7-Zip LZMA Benchmark: ARM Cortex-A15**. Available at: www.7-cpu.com/cpu/Cortex-A15.html, 2015.

[7ZI15c] 7-Zip Team. **7-Zip LZMA Benchmark: Applied Micro X-Gene**. Available at: www.7-cpu.com/cpu/X-Gene.html, 2015.

[AFT06] M. Aftosmis; M. Berger; R. Biswas; M. Djomehri et al. **A Detailed Performance Characterization of Columbia using Aeronautics Benchmarks and Applications**. In: *Aerospace Sciences Meeting and Exhibit* (AIAA), pp 1-17, 2006.

[AGR09] P. Agrawal. **Hybrid Simulation Framework for Virtual Prototyping Using OVP, SystemC & SCML A Feasibility Study**. Thesis*, Indian Institute of Technology*, 2009, 49p.

[AHM10] A. Ahmed; A. Abdallah; K. Kuroda. **Architecture and Design of Efficient 3D Network-on-Chip (3D NoC) for Custom Multicore SoC**. In: *International Conference on Broadband, Wireless Computing, Communication and Applications (BWCAA)*, pp. 67-73, 2010.

[AIN15] K. Aingaran; S. Jairath; G. Konstadinidis; S. Leung; P. Loewenstein et al. **M7: Oracle's Next-Generation SPARC Processor**. *IEEE Micro*, vol. 35, issue 2, pp 36-45, 2015.

[ALM04] A. Almojel. **Characterization of ILP Distribution for NASA NAS Parallel Benchmarks**. *Journal of King Saud University – Computer and Information Sciences*, vol. 16, pp 45-65, 2004.

[ALT14] Altera Corporation et al. **Hybrid Memory Cube Specification 2.0**. *Technical Specification of Hybrid Memory Cube, Revision 2.0*, 2014, 125p.

[ARM10] ARM Holdings plc. **Cortex-A9**. *Technical Reference Manual, Revision r2p2*, 2010, 230p.

[ARM11a] ARM Holdings plc. **Cortex-A15 MPCore**. *Technical Reference Manual, Revision r2p0*, 2011, 364p.

[ARM11b] ARM Holdings plc. **Big.LITTLE Processing with ARM Cortex-A15 & ARM Cortex-A7**. *White paper*, 2011, 8p.

[ARM11c] ARM Holdings plc. **CoreLink™ GIC-400 Generic Interrupt Controller**. *Technical Reference Manual, Revision r0p0*, 2011, 57p.

[ARM12a] ARM Holdings plc. **CoreTile Express A15x2 A7x3**. *Technical Reference Manual, Revision 0503B*, 2012, 181p.

[ARM12b] ARM Holdings plc. **Cortex-A7 MPCore**. *Technical Reference Manual, Revision r0p3*, 2012, 268p.

[ARM12c] ARM Holdings plc. **CoreLink™ CCI-400 Cache Coherent Interconnect**. *Technical Reference Manual, Revision r1p1*, 2012, 79p.

[ARM13a] ARM Holdings plc. **Introduction to AMBA® 4 ACE™ and big.LITTLE™ Processing Technology**. *White paper*, 2013, 15p.

[ARM13b] ARM Holdings plc. **Big.LITTLE Technology: The Future of Mobile**. *White paper*, 2013, 12p.

[ARM15a] ARM Holdings plc. **Datasheet for CoreTile Express V2P – CA15x2 CA7x3**. Available at: www.arm.com/files/pdf/Datasheet_CoreTileExpress_V2P-CA15x2_CA7x3.pdf, 2015.

[ARM15b] ARM Holdings plc. **NEON**. Available at: www.arm.com/products/processors/ technologies/ neon.php, 2015.

[ARM15c] ARM Holdings plc. **Cortex-A7 MPCore Technical Reference Manual: L2 Control Register**. Available at: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc .ddi0464e/BABBACEE.html, 2015.

[ARM15d] ARM Holdings plc. **Cortex-A9 MPCore Technical Reference Manual: SCU Configuration Register**. Available at: http://infocenter.arm.com/help/index.jsp?topic =/com.arm.doc.ddi0407f/BABEBFBH.html, 2015.

[ARM15e] ARM Holdings plc. **ARM Expects Vehicle Compute Performance to Increase 100x in Next Decade**. Available at: www.arm.com/about/newsroom/arm-expects-vehicle-compute-performance-to-increase-100x-in-next-decade.php, 2015.

[ARM15f] ARM Holdings plc. **Streamline for Gem5 – ARM**. Available at: www.arm.com/ products/tools/streamline-for-gem5.php. 2015.

[ARM15g] ARM Holdings plc. **CoreLink Interconnect**. Available at: www.arm.com/products/ system-ip/interconnect/, 2015.

[ART15] Arteris, Inc. **AppliedMicro Again Chooses Arteris FlexNoC for X-Gene Server on a Chip Products**. Available at: www.arteris.com/press-releases/appliedmicro_amcc _arteris_31_march_2015, 2015.

[ASA09] A. Asaduzzaman; F. Sibai; M. Rani. **Impact of level-2 cache sharing on the performance and power requirements of homogeneous multicore embedded systems**. *Microprocessors and Microsystems*, vol. 33, issue 5-6, pp 388-397, 2009.

[AZA14] E. Azarkhish; D. Rossi; I. Loi; L. Benini. **A Logic-base Interconnect for Supporting Near Memory Computation in the Hybrid Memory Cube**. In: *Workshop on Near-Data Processing (WoNDP, MICRO-47)*, pp 1-6. 2014.

[BAI94] D. Bailey; E. Barszcz; J. Barton; D. Browning; R. Carter et al. **The NAS Parallel Benchmarks**. *Technical Report RNR-94-007*, 1994, 79p.

[BAI95] D. Bailey; T. Harris; W. Saphir; R. Wijngaart; A. Woo et al. **The NAS Parallel Benchmarks 2.0**. *Report NAS-95-020*, 1995, 24p.

[BAL11] R. Balasubramonian; N. Jouppi. **Multi-Core Cache Hierarchies**. *Morgan & Claypool Publishers*, 1st ed., 2011, 154p.

[BAO15] Y. Bao; C. Bienia; K. Li. **The PARSEC Benchmark Suite Tutorial**. *Princeton University*. Available at: http://parsec.cs.princeton.edu/download/tutorial/3.0/parsec-tutorial.pdf, 2015.

[BEN02] L. Benini; G. De Micheli. **Networks on chips: a new SoC paradigm**. *Computer*, vol. 35, issue 1, pp 1-9, 2002.

[BEN10] L. Benini. **Programming Heterogeneous Many-core platforms in Nanometer Technology: the P2012 experience**. *Presentation at ARTIST DESIGN Summer school*, 2010, 71p.

[BEN13] A. Bengueddach; B. Senouci; S. Niar; B. Beldjilali. **Energy Consumption in Reconfigurable MPSoC Architecture: Two-Level Caches Optimization Oriented Approach**. In: *International Design and Test Symposium (IDT)*, pp 1-6, 2013.

[BIE08] C. Bienia; S. Kumar; J. Singh; K. Li. **The PARSEC Benchmark Suite: Characterization and Architectural Implications**. Technical Report TR-811-08, *Princeton University*, 2008, 22p.

[BIN06] N. Binkert; R. Dreslinski; L. Hsu; K. Tim; A. Saidi et al. **The M5 Simulator: Modeling Networked Systems**. *IEEE Micro*, vol. 26, issue 4, pp 52-60, 2006.

[BIN11] N. Binkert; B. Beckmann; G. Black; S. Reinhardt; A. Saidi et al. **The gem5 simulator**. *ACM SIGARCH Computer Architecture News*, vol. 39, issue 2, pp 1-7, 2011.

[BJE06] T. Bjerregaard; S. Mahadevan. **A survey of research and practices of Network-on-chip**. *Journal ACM Computing Surveys (CSUR)*, vol. 38, issue 1, pp 1-51, 2006.

[BOH04] P. Bohrer; M. Elnozahy; A. Gheith; C. Lefurgy; T. Nakra et al. **Mambo: a Full System Simulator for the PowerPC Architecture**. *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, issue 4, pp 8-12, 2004.

[BUT12] A. Butko; R. Garibotti; L. Ost; G. Sassatelli. **Accuracy Evaluation of GEM5 Simulator System.** In: *Reconfigurable Communication-centric System-On-Chip (ReCoSoC)*, pp 1-7, 2012.

[CAD15a] Cadence Design Systems, Inc. **Wide I/O Memory and 3D ICs – A New Dimension for Mobile Devices**. Available at: community.cadence.com/cadence_blogs_8/b/ii/archive/2011/03/28/wide-i-o-memory-and-3d-ics-a-new-dimension-for-mobile-devices, 2015.

[CAD15b] Cadence Design Systems, Inc. **Wide I/O 2, Hybrid Memory Cube (HMC) – Memory Models Advance 3D-IC Standards**. Available at: community.cadence.com/cadence_blogs_8/b/ii/archive/2013/08/06/wide-i-o-2-hybrid-memory-cube-hmc-memory-models-advance-3d-ic-standards, 2015.

[CAT14] R. Cataldo; R. Fernandes; F. Grando; T. Webber; A. Benso et al. **The Impacto f Routing Arbitration Mechanisms on 3D NoC Latency**. In: *International Conference on Electronics, Circuits and Systems (ICECS)*, pp 890-894, 2014.

[CHA11] T. Chaves. **Distributed Memory Organization with Support for Data Migration for NoC-based MPSoCs**. Master's Dissertation, *Pontifícia Universidade Católica do Rio Grande do Sul*, 2011, 90p.

[CHO06] S. Cho; L. Jin. **Managing distributed, shared L2 Caches through OS-Level Page Allocation**. In: *Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp 455-468, 2006.

[CON14] Mike O'Connor. **Highlights of the High-Bandwidth Memory (HBM) Standard**. *The Memory Forum 2014*, available at: www.cs.utah.edu/thememoryforum/mike.pdf, 25p.

[COS09] A. Coskun; J. Ayala; D. Atienza; T. Rosing; Y. Leblebici. **Dynamic Thermal Management in 3D Multicore Architectures**. In: *Design, Automation & Test in Europe (DATE)*, pp. 1410-1415, 2009.

[DIE15a] Die.net. **sched_setaffinity(2) – Linux man page**. *Linux Man pages*. Available at: linux.die.net/man/2/sched_setaffinity, 2015.

[DIE15b] Die.net. **taskset(1) – Linux man page**. *Linux Man pages*. Available at: linux.die.net/man/1/taskset, 2015.

[DUN14] T. Dung; I. Taniguchi; H. Tomiyama. **Cache Simulation for Instruction Set Simulator QEMU**. In: *Dependable, Automatic and Secure Computing (DASC)*, pp 441-446, 2014.

[DUR15] M. Duranton. **Design for Silicon Photonics**. *Presentation*. Available at: www-leti.cea.fr/en/content/download/1683/22763/file/C4, 2015, 19p.

[EBR13] M. Ebrahimi; M. Daneshtalab; P. Liljeberg; J. Plosila; H. Tenhunen. **Cluster-based topologies for 3D Networks-on-Chip using advanced inter-layer bus architecture**. *Journal of Computer and System Sciences*, vol. 79, issue 4, pp 475-491, 2013.

[ELM05] A. El-Moursey; R. Garg; D. Albonesi; S. Dwarkadas. **Partitioning Multi-Threaded Processors with a Large Number of Threads**. In: *International Symposium on Performance Analysis of System and Software (ISPASS)*, pp 112-123, 2005.

[END14] F. Endo; D. Couroussé; H.-P. Charles. **Micro-architectural Simulation of In-order and Out-of-order ARM Microprocessors with gem5**. In: *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pp 266-273, 2014.

[ENG10] J. Engblom; D. Aarno; B. Werner. **Full-System Simulation from Embedded to High-Performance Systems**. *In Processor and System-on-Chip Simulation,* Springer US, 1st ed., 2010, 345p.

[ENT15] EnterpriseTech. **ARM Brings More Cores to the Datacenter War**. Available at: www.enterprisetech.com/2014/10/22/arm-chip-interconnect-spans-lot/, 2015.

[ESM11] H. Esmaeilzadeh; E. Blem; R. Amant; K. Sankaralingam; D. Burger. **Dark Silicon and the End of Multicore Scaling**. In: *International Symposium on Computer Architecture (ISCA)*, pp 365-376, 2011.

[FAN13] J. Fang; A. Varbanescu; H. Sips; L. Zhang; Y. Che; C. Xu. **An Empirical Study of Intel Xeon Phi**. *Cornell University Library, Eprint arXiv:1310.5842,* pp 1-11, Dec. 2013.

[FEE09] B. Feero; P. Pande. **Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation**. *IEEE Transactions on Computers*, vol. 58, issue 1, pp. 32-45, 2009.

[FER12] E. Fernandez-Alonso; D. Castells-Rufas; J. Joven; J. Carrabina. **Survey of NoC and Programming Models Proposals for MPSoC**. *International Journal of Computer Science issues (IJCSI),* vol. 9, issue 2, pp. 22-32, 2012.

[FER15] R. Fernandes; L. Brahm; T. Webber; R. Cataldo; L. Poehls et al. **OcNoC: Efficient One-cycle Router Implementation for 3D Mesh Network-on-Chip**. In: *International Conference on VLSI Design (VLSID)*, pp 105-110, 2015.

[FIC13] D. Fick; R. Dreslinski; B. Giridhar; G. Kim; S. Seo et al. **Centip3De: A Cluster-Based NTC Architecture With 64 ARM Cortex-M3 Cores in 3D Stacked 130 nm CMOS**. *IEEE Journal of Solid-State Circuits*, vol. 48, issue 1, pp. 104-117, 2013.

[FOR15] Forbes.com LLC. **What's The Significance of AppliedMicro's X-Gene 3 and X-Tend Interconnect?** Available at: www.forbes.com/sites/patrickmoorhead/2015/11/27/whats-the-significance-of-applied-micro-circuits-x-gene-3-and-x-tend-interconnect/, 2015.

[FRA04] A. Fraboulet; T. Risset; A. Scherrer. **Cycle Accurate Simulation Model Generation for SoC Prototyping**. Research Report Nº 2004-18, *École Normale Supérieure de* Lyon, 2004, 24p.

[FRE12] V. Fresse; Z. Ge; J. Tan; F. Rousseau. **Case Study: Deployment of the 2D NoC on 3D for the Generation of Large Emulation Platforms**. In: *International Conference on Image Processing Theory, Tools and Application (IPTA)*, pp 435-441, 2012.

[FU14] W. Fu; L. Liu; T. Chen. **Direct distributed memory access for CMPs**. *Journal of Parallel and Distributed Computing*, vol. 74, issue 2, pp. 2109-2122, 2014.

[FUJ15] Fujitsu Limited. **Fujitsu M10 Server Architecture**. *White paper*, 2015, 60p.

[GAR05] R. Garg; A. El-Moursy; S. Dwarkadas; D. Albonesi; J. Rivers et al. **Cache Design Options for a Clustered Multithreaded Architecture.** Technical Report 866, *University of Rochester*, 2005, 21p.

[GEB09] M. Gebhart; J. Hestness; E. Fatehi; P. Gratz; S. Keckler. **Running PARSEC 2.1 on M5"**. Technical Report TR-09-32, *University of Texan*, 2009, 20p.

[GEM15a] Gem5. **Status Matrix**. Available at: www.gem5.org/Status_Matrix, 2015.

[GEM15b] Gem5. **Gem5 Review Board**. Available at: reviews.gem5.org/r/, 2015.

[GEM15c] Gem5. **Gem5 General Memory System**. Available at: www.gem5.org/General_Memory_System, 2015.

[GEM15d] Gem5. **Gem5 Devices**. Available at: www.gem5.org/Devices, 2015.

[GEM15e] Gem5. **Gem5 SimObjects**. Available at: www.gem5.org/SimObjects, 2015.

[GEM15f] Gem5. **O3CPU**. Available at: www.gem5.org/O3CPU, 2015.

[GEM15g] Gem5. **How many CPUs can M5 run?** Available at: www.gem5.org/Frequently_Asked_Questions#How_many_CPUs_can_M5_run.3F, 2015.

[GEM15h] Gem5 Q&A. **Is there a way to model more than 4 CPUs in ARM gem5?** Available at: qa.gem5.org/432/is-there-a-way-to-model-more-than-4-cpus-in-arm-gem5, 2015.

[GEM15i] Gem5 Q&A. **Booting an ARM FS System with more than 4 CPUs**. Available at: qa.gem5.org/89/booting-an-arm-fs-system-with-more-than-4-cpus%C2%A0, 2015.

[GEM15j] Gem5-users. **Internet connection inside of gem5 simulator**. Available at: comments.gmane.org/gmane.comp.emulators.m5.users/13738, 2015.

[GEM15k] Gem5 Doxygen. **Gem5: Noncoherent Bus Class Reference**. Available at: www.gem5.org/docs/html/classNoncoherentBus.html. 2015.

[GEM15l] Gem5. **PARSEC Benchmarks**. Available at: www.gem5.org/PARSEC_benchmarks, 2015.

[GEM15m] Gem5. **azarkhish's review requests**. Available at: http://reviews.gem5.org/users/azarkhish/, 2015.

[GEM15n] Gem5-users. **Using Ruby with ARM FS**. Available at: http://comments.gmane.org/gmane.comp.emulators.m5.users/14858. 2015.

[GEM15o] Gem5 Q&A. **Is ARM FS going to be supported by ruby?** Available at: http://qa.gem5.org/203/is-arm-fs-going-to-be-supported-by-ruby, 2015.

[GEM15p] Gem5-users. **ARM model failed at runtime**. Available at: http://article.gmane.org/gmane.comp.emulators.m5.users/19286, 2015.

[GEN12] J. Gentle; W. Härdle; Y. Mori. **Handbook of Computational Statistics**. *Springer-Verlag Berlin Heidelberg*, 2nd ed., 2012, 1192p.

[GEO15] Georgia Institute of Technology. **The 3D-MAPS Processors**. Available at: www.gtcad.gatech.edu/ 3d-maps/index.html, 2015.

[GLI09] M. Gligor; N. Fournel; F. Pétrot. **Using Binary Translation in Event Driven Simulation for Fast and Flexible MPSoC Simulation**. In: *IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp 71-80, 2009.

[GOR07] A. Gordon-Ross; P. Viana; F. Vahid; W. Najjar; E. Barros. **A One-Shot Configurable-Cache Tuner for Improved Energy and Performance**. In: *Design, Automation & Test in Europe Conference (DATE)*, pp 1-6, 2007.

[GRE12] M. Greenberg. **LPDDR3 and Wide I/O DRAM: Interfaces Changes that give PC-Like Memory Performance to Mobile Devices**. *Presentation at MemCon*, 2012, 37p.

[GUT12] E. Guthmuller; I. Miro-Panades; A. Greiner. **Adaptive Stackable 3D Cache Architecture for Manycores**. In: *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 39-44, 2012.

[GUT14] A. Gutierrez; J. Pusdesris; R. Dreslinski; T Mudge; C. Sudanthi et al. **Source of Error in Full-System Simulator**. In: *International Symposium on Performance Analysis of Systems and Software (ISPASS),* pp 1-10, 2014.

[GUT15] T. Gutierrez. **Dev: add an Ethernet switch model**. *Review Board*, Available at: reviews.gem5.org/r/2305/, 2015.

[HAN14] A. Hansson; N. Agarwal; A. Kolli; T. Wenisch; A. Udipi. **Simulating DRAM controllers for future system architecture exploration**. In: *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp 1-10, 2014.

[HAR14] P. Haririan; A. Garcia-Ortiz. **Non-Intrusive DVFS Emulation in gem5 with Application to Self-Aware Architectures**. In: *International Symposium on Reconfigurable and Communication-Centric Systems-on-chip (ReCoSoC)*, pp 1-7, 2014.

[HMC15] Hybrid Memory Cube Consortium. **Hybrid Memory Cube Consortium – Home**. Available at: www.hybridmemorycube.org/, 2015.

[HP15] Hewlett-Packard Development Company. **HP Labs: McPAT**. Available at: www.hpl.hp.com/ research/mcpat/, 2015.

[HUA12] X. Huang; X. Fan; S. Zhang; Y. Chen. **DLWAP-buffer: A Novel HW/SW Architecture to Alleviate the Cache Coherence on Streaming-like Data in CMP**. In: *International Symposium on Embedded Multicore SoCs (MCSoC)*, pp 23-28, 2012.

[HWA11] K. Hwang. **Advanced Computer Architecture**. *Tata McGraw-Hill Education*, 2nd ed., 2011, 723p.

[HWA12] K. Hawng; J. Dongarra; G. Fox. **Distributed and Cloud Computing: From Parallel Processing to the Internet of Things**. *Morgan Kaufmann*, 1st ed., 2011, 672p.

[IBM07] International Business Machines (IBM) Corporation. **Performance Analysis with the IBM Full-System Simulator**. *SysSimPerfAnalysisGuide*, Version 3, 2007, 42p.

[IBM14] International Business Machines (IBM) Corporation. **IBM POWER8 processor and memory buffer protocols**. *Brochure*, 2014. 8p.

[IBM15] International Business Machines (IBM) Corporation. **Under the Hood: Of POWER7 Processor Caches**. *White Paper*, 2015, 16p.

[IDG15] IDG Consumer & SMB. **Meet Knight's Landing: Intel's most powerful chip ever is overflowing with cutting-edge technologies**. Available at: www.pcworld.com/ article/2366700/intels-most-powerful-chip-ever-packs-emerging-technologies.html, 2015.

[IMP11] Imperas Software Ltd. **Open Virtual Platform (OVP) An introduction and Overview**. Presentation December 2011. Available at: www.ovpworld.org/ presentation.php?slide=OVPINTRO2, 2015.

[IMP13] Imperas Software Ltd. **OVP Overview Datasheet**. Revision 09/2013, 2p.

[IMP14] Imperas Software Ltd. **OVPsim and CpuManager User Guide**. Version 2.3.3, revision 04/2014, 155p.

[IMP15a] Imperas Software Ltd. **Technology OVPsim**. Available at: www.ovpworld.org/ technology_ovpsim, 2015.

[IMP15b] Imperas Software Ltd. **Technology OVP Models**. Available at: www.ovpworld.org/ technology_models, 2015.

[IMP15c] Imperas Software Ltd. **Technology OVP APIs**. Available at: www.ovpworld.org/ technology_apis, 2015.

[IMP15d] Imperas Software Ltd. **OVP Guide to Using Processor Models**. Version 0.5, revision 01/2015, 24p.

[IMP15e] Imperas Software Ltd. **OVP Forums – Cache coherence, cache miss/hit cache levels**. Available at: www.ovpworld.org/forum/viewtopic.php?p=32, 2015.

[IMP15f] Imperas Software Ltd. **OVP Forums – Modelling bus as resource in a system**. Available at: www.ovpworld.org/forum/viewtopic.php?p=562, 2015.

[IMP15g] Imperas Software Ltd. **OVP Forums – Availability of ARM prime-cell library modules**. Available at: www.ovpworld.org/forum/viewtopic.php?p=976, 2015.

[INT14] Intel Corporation. **Intel® Xeon Phi™ Coprocessor Datasheet**. Available at: www.colfax-intl.com/nd/downloads/Xeon-Phi-Coprocessor-Datasheet.pdf, 2015.

[INT15a] Intel Corporation. **PRESS-KIT – Intel® System on Chip (SoC)**. Available at: www.intel.com/pressroom/kits/soc/, 2015.

[INT15b] Intel Corporation. **Intel® Xeon Phi™ Coprocessor – the Architecture**. Available at: software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-codename-knights-corner, 2015.

[ITR11] ITRS. **International Technology Roadmap for Semiconductors: System Drivers**. Available at: www.itrs.net/Links/2011ITRS/2011Chapters/ 2011SysDrivers.pdf, 2015.

[JIA11] W. Jiawen; L. Li; Z. Yuang; P. Hongbing; H. Shuzhuan et al. **A hybrid Hierarchical Architecture for 3D Multi-Cluster NoC**. In: *International Conference on Computer Science & Education (ICCSE)*, pp. 512-516, 2011.

[JED15a] JEDEC. **JEDEC Publishes Wide I/O 2 Mobile DRAM Standard**. Available at: www.jedec.org/news/pressreleases/jedec-publishes-wide-io-2-mobile-dram-standard, 2015.

[JED15b] JEDEC. **JEDEC Standard High Bandwidth Memory (HBM) DRAM**. *JESD235A, Revision of JESD235*, 2015, 172p.

[KAN05] M. Kandemir; N. Dutt. **Memory Systems and Compiler Support for MPSoC Architectures**. *Multiprocessor Systems-on-Chips (Systems on Silicon)*, *Morgan Kauffman*, 1st ed., 2005, 608p.

[KIM02] C. Kim; D. Burger; S. Keckler. **An adaptive, Non-Uniform Cache Structure for Wire-delay Dominated On-Chip Caches**. In: *International Conference on Architecture Support for Programming Languages and Operating Systems (ASPLOS)*, pp 211-222, 2002.

[KIM03] C. Kim; D. Burger; S. Keckler. **Nonuniform Cache Architecture for Wire-delay Dominated On-Chip Caches**. *IEEE Micro*, vol. 23, issue 6, pp 99-107, 2003.

[KIM13a] D. Kim; K. Athikulwongse; M. Healy; M. Hossain; M. Jung et al. **Design and Analysis of 3D-MAPS (3D Massively Parallel Processor with Stacked Memory)**. *IEEE Transactions on Computers*, vol. 64, issue 1, pp 1-14, 2013.

[KIM13b] G. Kimmich. **3D – What's Next**. *Presentation at D43D Workshop*, *2013*. Available at: www.leti-innovationdays.com/presentations/D43DWorkshop/Session3-StateOfTheArts/ D43D13_Session_3_1_GeorgKimmich.pdf, 2015.

[KIM14] J. Kim; Y. Kim. **HBM: Memory Solution for Bandwidth-Hungry Processors**. *Presentation at Hot Chips (HC)*, 2014, 24p.

[KON15] G. Konstadinidis; H. Li; F. Schumacher; V. Krishnaswamy; H. Cho; S. Dash et al. **SPARC M7: A 20nm 32-Core 64MB L3 Cache Processor**. *IEEE Journal of Solid-State Circuits*, vol. 51, issue 1, pp 1-13, 2015.

[KRI13] T. Krishna; C.-H. Chen; S. Park; W.-C. Kwon; S. Subramanian et al. **Single-Cycle Multihop Asynchronous Repeated Traversal: A SMART Future for Reconfigurable On-Chip Networks**. *Computer*, vol. 46, issue 10, pp 48-55, 2013.

[LEE15a] H.-H. Lee. **Non-Uniform Cache Architecture**. *Guest Lecture for ECE4100/6100*, Available at: wiki.cc.gatech.edu/multicore/images/0/03/Nuca.ppt, 2015.

[LEE15b] W. Lee; Y. Kim; J. Ryoo; D. Sunwoo; A. Gerstlauer et al. **PowerTrain: A Learning-based Calibration of McPAT Power Models**. In: *International Symposium on Low Power Electronics and Design (ISLPED)*, pp 189-194, 2015.

[LI06] F. Li; C. Nicopoulos; T. Richardson; Y. Xie; V. Narayanan et al. **Design and Management of 3D Chip Multiprocessors Using Network-in-Memory**. In: *International Symposium on Computer Architecture (ISCA)*, pp. 130-141, 2005.

[LI09] S. Li; J. Ahn; J. Brockman; N. Jouppi. **McPAT 1.0: An Integrated Power, Area, and Timing Modeling Framework for Multicore Architectures**. Technical Report, *HP Labs*, 2009, 40p.

[LI15] P. Li; J. Shin; G. Konstadinidis; F. Schumacher; V. Krishnaswamy et al. **A 20nm 32-Core 64MB L3 Cache SPARC M7 Processor**. In: *IEEE International Solid-State Circuits Conference (ISSCC)*, pp 1-3, 2015.

[LIN15] Linaro. **Non-Uniform Memory Access (NUMA) Support**. Available at: wiki.linaro.org/LEG/Engineering/Kernel/NUMA, 2015.

[LOI10] I. Loi; L. Benini. **An efficient distributed memory interface for Many-Core platform with 3D stacked DRAM**. In: *Design, Automation & Test in Europe (DATE)*, pp. 99-104, 2010.

[LUO14] H. Luo; P. Li; C. Ding. **Parallel Data Sharing in Cache: Theory, Measurement and Analysis**. Technical Report TR-994, *The University of Rochaster*, 2014, 25p.

[LUT13] M. Lutz. **Learning Python**. *O'Reilly Media*, 5th ed., 2013, 1600p.

[LWN15] LWN.net. **arm64: GICv3 support**. Available at: http://lwn.net/Articles/584305/, 2015.

[MAG02] P. Magnusson; M. Christensson; J. Eskilson; D. Forsgren; J. Hållberg et al. **Simics: A Full System Simulation Platform**. *Computer*, vol. 35, issue 2, pp 50-58, 2002.

[MAH13] D. Mahmoodi; R. Stepuch; R. da Silva; A. Tandon; N. Gupta. **Improving CPU performance modelling in QEMU**. Master's Dissertation, *University of* Southampton, 2013, 164p.

[MAR05] M. Martin; D. Sorin; B. Beckmann; M. Marty; M. Xu et al. **Multifacet's General Execution-driven Multiprocessor Simulator (GEMS) Toolset**. *ACM SIGARCH Computer Architecture News*, vol. 33, issue 4, pp 92-99, 2005.

[MAR11] P. Marwedel; J. Teich; G. Kouveli; I. Bacivarov; L. Thiele et al. **Mapping of applications to MPSoCs**. In: *International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp 109-118, 2011.

[MAR12] M. Martin; M. Hill; D. Sorin. **Why On-chip Cache Coherence is Here to Stay**. *Communications of the ACM*, vol. 55, issue 7, pp 78-89, 2012.

[MAR14] C. Marcon; T. Webber; R. Fernandes; R. Cataldo; F. Grando et al. **Tiny – optimised 3D mesh NoC for area and latency minimisation**. *Electronics Letters*, vol. 50, issue 3, pp. 165-166, 2014.

[MAT10] T. Mattson. **The Future of Many Core Computing: a tale of two processors**. *Presentation from Intel Labs*, Available at: openlab-mu-internal.web.cern.ch/openlab-mu-internal/00_News/News_pages/2010/10-08_Intel_Computing_Seminar/SCC-80-core-cern.pdf, 2010.

[MAT11] D. Matos; G. Palermo; V. Zaccaria; C. Reinbrecht; A. Susin et al. **Floorplanning-Aware Design Space Exploration for Application-Specific Hierarchical Networks-on-Chip**. In: *International Workshop on Network on Chip Architectures (NoCArc)*, pp 31-36, 2011.

[MCV15] L. McVoy. **LMbench – Tools for Performance Analysis**. Available at: www.bitmover.com/lmbench/, 2015

[MEL12] D. Melpignano; L. Benini; E. Flamand; B. Jego; T. Lepley et al. **Platform 2012, a many-core computing accelerator for embedded SoCs: performance evaluation of visual analytics applications**. In: *Design Automation Conference (DAC)*, pp 1137-1142, 2012.

[MIC15a] Micron Technology. **TwinDie™ DDR3 SDRAM**. Available at: www.micron.com/~/media/documents/products/data-sheet/dram/mt41j1g4_64m_32m_twindie.pdf, 2015.

[MIC15b] Micron Technology. **Mobile LPDDR3 SDRAM**. Available at: www.micron.com/~/media/documents/products/data-sheet/dram/mobile-dram/low-power-dram/lpddr3/178b_8-16gb_2c0f_mobile_lpddr3.pdf, 2015.

[MIC15c] Micron Technology, Inc. **Hybrid Memory Cube**. Available at: www.micron.com/products/hybrid-memory-cube, 2015.

[MPI15] MPICH. **MPICH Release 3.2**. Available at: www.mpich.org/static/downloads/3.2/mpich-3.2-README.txt, 2015.

[NAS15a] NASA. **NAS Parallel Benchmarks**. Available at: www.nas.nasa.gov/publications/npb.html, 2015.

[NAS15b] NASA. **NPB Problem Sizes**. Available at: www.nas.nasa.gov/publications/npb_problem_sizes.html, 2015.

[NIK15] S. Niknam; A. Asad; M. Fathy; A.-M. Rahmani. **Energy Efficient 3D Hybrid processor-memory architecture for the dark silicon age**. In: *International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, pp 1-8, 2015.

[NVI15] NVIDIA Corporation. **Buy Jetson TK1 DevKit**. Available at: developer.nvidia.com/jetson-tk1, 2015.

[OLU07] K. Olukotun. **Chip Multiprocessor Architecture: Techniques to Improve Throughput and Latency**. *Morgan and Claypool Publishers*, 1st ed., 2007, 154p.

[ORC13] J. Orcutt; R. Ram; V. Stojanovic. **Integration of Silicon Photonics into Electronic Processes**. *Presentation at SPIE 8629, Silicon Photonics VIII,* 2013, 30p.

[PAD11] D. Padua. **Encyclopedia of Parallel Computing**. *Springer US*, 2011, 2211p.

[PAR15a] Parsec users mailing list. **[parsec-user] x264 run error**. Available at: lists.cs.princeton.edu/pipermail/parsec-users/2014-January/001598.html, 2015

[PAR15b] Parsec users mailing list. **[parsec-user] Pthread creation: vips and x264**. Available at: lists.cs.princeton.edu/pipermail/parsec-users/2009-August/000490.html, 2015.

[PAT13] D. Patterson; J. Hennessy. **Computer Organization and Design, Fifth Edition: The Hardware/Software Interface**. *Morgan Kaufmann*, 5th ed., 2013, 800p.

[PIN14] V. Pinto; A. Lorenzon; A. Beck; N. Maillard; P. Navaux. **Energy Efficiency Evaluation of Multi-Level Parallelism on Low Power Processors**. In: *Congresso da Sociedade Brasileira de Computação (CSBC)*, pp 1-12, 2014.

[PLA15] The Platform. **More Knights Landing Xeon Phi Secrets Revealed**. Available at: www.theplatform.net/2015/03/25/more-knights-landing-xeon-phi-secrets-unveiled/, 2015.

[POU09] N. Pouillon; A. Becoulet; A. Mello; F. Pêcheux; A. Greiner. **A Generic Instruction Set Simulator API for Timed and Untimed Simulation and Debug of MP2-SoCs**. Presentation at *Rapid System Prototyping* (RSP), 2009. Available at: www.soclib.fr/trac/dev/attachment/wiki/PapersAndPublications/rsp_09_iss2.pdf, 2015.

[PRI15] Princeton University. **The PARSEC Benchmark Suite**. Available at: parsec.cs.princeton.edu/index.htm, 2015.

[PUR15] Purch. **Determining the TDP of Exynos 5 Dual – The ARM vs x86 Wars Have Begun**. Available at: www.anandtech.com/show/6536/arm-vs-x86-the-real-showdown/13, 2015.

[PUS11] K. Pusukuri; R. Gupta; L. Bhuyan. **Thread Reinforcer: Dynamically Determining Number of Threads via OS Level Monitoring**. In: *IEEE International Symposium on Workload Characterization (IISWC)*, pp 116-125, 2011.

[QUA15] Qualcomm Technologies, Inc. **Snapdragon Mobile Processors and Chipsets**. Available at: www.qualcomm.com/products/snapdragon, 2015.

[RAB15] System Level Synthesis. **RABBITS: an environment for fast and accurate MPSoC simulation**. Available at: tima.imag.fr/sls/research-projects/rabbits/, 2015.

[RAN15] S. Ranganathan; M. Buckley. **Adoption of Low-Power Components in Embedded Applications**. *Axiom*, vol. 3, issue 3, pp 1-7, 2015.

[REG15] The Register. **Fujitsu to embiggen iron bigtime with Sparc64-X**. Available at: www.theregister.co.uk/2012/09/04/fujitsu_sparc64_x_processor/, 2015.

[REK13] W. Rekik; M. Said; N. Amor; M. Abid. **Virtual prototyping of multiprocessors architectures using the Open Virtual Platform**. In: *International Conference on Computer Applications Technology (ICCAT)*, pp 1-6, 2013.

[RET11] S. Rethinagiri; R. Atitallah; J.-L. Dekeyser. **A System Level Power Consumption Estimation for MPSoC**. In: *International Symposium on System on Chip (SoC)*, pp 56-61, 2011.

[RIC15] A. Rice. **SICSA PhD Conference 2011**. Available at: www.cl.cam.ac.uk/~acr31/sicsa/, 2015.

[ROG09] B. Rogers; A. Krishna; G. Bell; K. Vu; X. Jiang et al. **Scaling the Bandwidth Wall: Challenges in and Avenues for CMP Scaling**. In: *International Symposium on Computer Architecture (ISCA)*, pp 371-382, 2009.

[ROS12] F. Ross. **Migrating to LPDDR3: An overview of LPDDR3, commands, operations, and functions**. *Presentation at LPDDR3 Symposium 2012*, 2012, 28p.

[RUG08] J. Ruggiero. **Measuring Cache and Memory Latency and CPU to Memory Bandwidth**. *Intel Corporation, White paper,* 2008, 14p.

[SAB10] M. Sabry; M. Ruggiero; P. Valle. **Performance and Energy Trade-Offs Analysis of L2 on-Chip Cache Architectures for Embedded MPSoCs**. In: *Great Lake Symposium on VLSI (GLSVLSI)*, pp 1-6, 2010.

[SAI12] A. Saidi; A. Hansson. **Simulating Systems not Benchmarks**. Presentation at *European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC).* Available at: gem5.org/dist/tutorials/hipeac2012/gem5_hipeac.pdf, 2012.

[SAI96] S. Saini; D. Bailey. **NAS Parallel Benchmark (Version 1.0) Results 11-96**. *Report NAS-96-18*, 1996, 53p.

[SAM15a] Samsung. **Exynos 5 Octa (5422)**. Available at: www.samsung.com/global/business/semiconductor/product/application/detail?productId=7978&iaId=2341, 2015.

[SAM15b] Samsung. **Samsung Develops Mobile DRAM with Wide I/O Interface**. Available at: www.samsung.com/global/business/semiconductor/news-events/press-releases/detail?newsId=4028, 2015.

[SCH15a] A. Schoenberger; K. Hofmann. **Analysis of Asymmetric 3D DRAM Architecture in Combination with L2 Cache Size Reduction**. In: *International Conference on High Performance Computing & Simulation (HPCS)*, pp 123-128, 2015.

[SCH15b] T. Schmitz. **The Rise of Serial Memory and the Future of DDR**. *White paper: UltraScale Devices*, pp 1-9, 2015.

[SHA12] A. Sharifi; E. Kultursay; M. Kandemir; C. Das. **Addressing End-to-End Memory Access Latency in NoC-based Multicore**. In: *International Symposium on Microarchitecture (MICRO)*, pp 294-304, 2012.

[SIN14] G. Singh; G. Favor; A. Yeung. **AppliedMicro X-Gene 2**. *Presentation at Hot Chips (HC)*, 2014, 24p.

[SIV14] R. Sivaramakrishnan; S. Jairath. **Next Generation SPARC Processor Cache Hierarchy**. *Presentation at Hot Chips (HC)*, 2014, 28p.

[SMI05] B. Smith; B. Bode. **Performance Effects of Node Mappings on the IBM BlueGene/L Machine**. *In Euro-Par 2005 Parallel Processing*, *Springer Berlin Heidelberg*, 1st ed., pp 1005-1013, 2005.

[SOC15a] Laboratoire d'Informatique de Paris 6. **SoCLib**. Available at: www.soclib.fr. 2015.

[SOC15b] Laboratoire d'Informatique de Paris 6. **Writing TLM2.0-compliant timed SystemC simulation models for SoCLib**. Available at: www.soclib.fr/trac/dev/wiki/ WritingRules/Tlmt, 2015.

[SOC15c] Laboratoire d'Informatique de Paris 6. **Writing efficient Cycle-Accurate, Bit-Accurate SystemC simulation models for SoCLib**. Available at: www.soclib.fr/trac/ dev/wiki/WritingRules/Caba, 2015.

[SOU15a] G. Southern; J. Renau. **Deconstructing PARSEC Scalability**. In: *Workshop on Duplicating, Deconstructing and Debunking (ISCAWDDD)*, pp 1-10, 2015.

[SOU15b] G. Southern; J. Renau. **Deconstructing PARSEC Scalability**. *Presentation at Workshop on Duplicating, Deconstructing and Debunking (ISCAWDDD),* 2015, available at: https://users.soe.ucsc.edu/~gsouther/papers/wddd2015_deconstructing _parsec_slides.pdf, 2015.

[SPI13] V. Spiliopoulos. A. Bagdia; A. Hansson; P. Aldworth; S. Kaxiras. **Introducing DVFS-Management in a Full-System Simulator**. In: *International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp 535-345, 2013.

[STA15] W. Starke; J. Stuecheli; D. Daly; J. Dodson; F. Auernhammer et al. **The cache and memory subsystem of the IBM POWER8 processor**. *IBM Journal of Research and Development*, vol. 59, issue 1, pp. 1-13, 2015.

[STO15] Stone Arch Services, Inc. **Intel talks about Knights Landing architecture**. Available at: semiaccurate.com/2015/03/31/intel-talks-knights-landing-architecture/, 2015.

[STU13] J. Stuecheli. **POWER8**. *Presentation at Hot Chips (HC)*, 2013, 20p.

[SUH06] T. Suh; **Integration and Evaluation of Cache Coherence Protocols for Multiprocessor SoCs**. Thesis, *Georgia Institute of Technology*, 2006, 153p.

[TEC15a] Tech Design Forums. **Parallel simulation of SystemC TLM 2.0 compliant MPSoCs**. Available at: www.techdesignforums.com/practice/technique/parallel-simulation-of-systemc-tlm-2-0-compliant-mpsocs/, 2015.

[TEC15b] The Tech Report. **AMD's High-bandwidth memory explained**. Available at: http://techreport.com/review/28294/amd-high-bandwidth-memory-explained, 2015.

[TIL11a] Tilera Corporation. **Tile Architecture Many Core CPU**. *Presentation at Japan Internet Week*, 2011, 9p.

[TIL11b] Tilera Corporation. **Tile-Gx100 ManyCore Processor: Acceleration Interfaces and Architecure**. *Presentation at Hot Chips (HC)*, 2011, 21p.

[TIL11c] Tilera Corporation. **Tile Processor Architecture Overview for the TilePro Series**. *Architecture Overview*, Release 1.2, 2011, 56p.

[TIL13] Tilera Corporation. **Tile-Gx Instruction Set Architecture**. Release 1.2, 2013, 502p.

[TOR14] M. Torquati; K. Bertels; S. Karlsson; F. Pacull. **Smart Multicore Embedded Systems**. *Springer-Verlag New York*, 1st ed., 2014, 175p.

[TRA10] C. Trabelsi; S. Meftali; R. Atitallah; A. Jemai; J. Dekeyser et al. **An MDE Approach for Energy Consumption Estimation in MPSoC Design**. In: *Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, pp 1-6, 2010.

[TRI11] M. Triola. **Elementary Statistics (technology update)**. *Pearson Education*, 11th ed., 2010, 888p.

[UBU15a] Ubuntu Wiki. **QemuARMVexpress**. Available at: https://wiki.ubuntu.com/Kernel/Dev/ QemuARMVexpress. 2015.

[UBU15b] Ubuntu Community Help Wiki. **Setting up an MPICH2 Cluster in Ubuntu**. Available at: https://help.ubuntu.com/community/MpichCluster, 2015.

[VAN00] S. Vanderwiel; D. Lilja. **Data Prefetch Mechanisms**. *ACM Computing Surveys (CSUR)*, vol. 32, issue 2, pp 174-199, 2000.

[VIV11] P. Vivet. **A Three-Layers 3D-IC Stack including Wide-IO and 3D NoC – Practical Design Perspective**. *Presentation at 3D Architectures for Semiconductor Integration and Packaging Conference*, 2011, 24p.

[WA08] X. Wang; G. Gan; J. Manzano; D. Fan; S. Guo. **A Quantitative Study of the On-Chip Network and Memory Hierarchy Design for Many-Core Processor**. In: *IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 689-696, 2008.

[WAH98] A. Waheed; J. Yan; **Workload Characterization of CFD Applications Using Partial Differential Equation Solvers**. *NAS Technical Report NAS-98-011*, 1998, 35p.

[WCC15] WCCF PTE Limited. **AMD Officially Reveals 3D Graphics Memory – HBM Coming With Fiji CPU This Quarter**. Available at: wccftech.com/amd-releases-official-hbm-presentation-3d-stacked-high-bandwidth-memory-cornerstone-gen-gpus-apus/, 2015.

[WIE12] U. Wiener. **Modeling and Analysis of a Cache Coherent Interconnect**. Thesis, *Eindhoven University of Technology*, 2012, 83p.

[WIN10] Wind River Systems, Inc. **System Architecture Exploration Using Wind River Systems**. Revision 08/2010. Available at: www.windriver.com/whitepapers/system-architecture-exploration-using-simics/wr_creating-virtual-platforms_wp.pdf, 2015.

[WIN15a] Wind River Systems, Inc. **Wind River University Program**. Available at: www.windriver.com/universities/, 2015.

[WIN15b] Wind River Systems, Inc. **Wind River Simics for Software Development.** Revision 01/2015. Available at: www.windriver.com/whitepapers/simics-for-software-development/ WP_Simics_Software_Development.pdf, 2015.

[WOO10] D. Woo; N. Seong; D. Lewis; H.-H. Lee. **An Optimized 3D-Stacked Memory Architecture by Exploting Excessive, High-Density TSV Bandwidth**. In: *International Symposium on High Performance Computer Architecture (HPCA)*, pp 1-12, 2010.

[WU09] X. Wu; J. Li; L. Zhang; E. Speight; R. Rajamony et al. **Hybrid Cache Architecture with Disparate Memory Technologies**. In: *International Symposium on Computer Architecture (ISCA)*, pp. 34-45, 2009.

[WUL95] W. Wulf; S. McKee. **Hitting the Memory Wall: Implications of the Obvious**. *ACM Special Interest Group on Computer Architecture (SIGARCH)*, vol. 23, issue 1, pp 20-24, 1995.

[YE10] J. Ye; M. Cao; Z. Qu; T. Chen. **Regional cache organization for NoC based many-core processors**. *IEEE International Conference on Computer and Information Technology*, vol. 79, issue 2, pp. 175-186, 2010.

[YUN15] H. Yun; P. Valsan. **Evaluating the Isolation Effect of Cache Partitioning on COTS Multicore Platforms**. In: *Workshop on Operating Systems Platforms for Embedded Real-time applications (OSPERT)*, pp 1-6, 2015.

[ZAR15] J. Zarrin; R. Aguiar; J. Barraca. **Dynamic, scalable and flexible resource discovery for large-dimension many-core systems**. *Future Generation Computer Systems*, vol. 53, pp 119-129, 2015.

[ZHA14] Y. Zhang; L. Li; Z. Lu; A. Jantsch; M. Gao et al. **A survey of memory architecture for 3D chip multi-processor**. *Microprocessors and Microsystems*, vol. 38, Issue 5, pp. 415-430, 2014.