UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

TOWARDS CACHE-COHERENT CHIPLET-BASED ARCHITECTURES WITH WIRELESS INTERCONNECTS

NITISH ARYA

Thesis supervisor: SERGI ABADAL CAVALLÉ (Department of Computer Architecture)

Thesis co-supervisor: ABHIJIT DAS

Degree: Master Degree in Innovation and Research in Informatics (High Performance Computing)

Thesis report

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

29/06/2023

Acknowledgement

I would like to express my sincere gratitude and appreciation to all those who have supported me throughout the completion of this master's thesis.First and foremost, I am grateful to my family for their unwavering love, encouragement, and belief in my abilities. Their constant support and motivation have been the driving force behind my accomplishments.

I am deeply grateful to my supervisor, Dr. Sergi Abadal, and co-supervisor, Dr. Abhijit Das, for their invaluable guidance, encouragement, and expertise throughout this research journey. Their insightful feedback and continuous support have been instrumental in shaping this thesis. I would also like to extend my thanks to NaNoNetworking Center in Catalunya at UPC, for providing the resources and support without which this thesis would not have been done.

I would like to express my appreciation to my friends and colleagues who have provided me with assistance, encouragement, and stimulating discussions. Their support has helped me overcome challenges and kept me motivated during the thesis writing process.

In conclusion, I am grateful to everyone who has played a part, big or small, in the successful completion of this master's thesis. Their support and encouragement have been invaluable, and I am truly humbled by their contributions.

Abstract

Cache-coherent chiplet-based architectures have gained significant attention due to their potential for scalability and improved performance in modern computing systems. However, the interconnects in such architectures often results in increased latency and energy consumption. Due to physical limitations such as increasing wire resistance due to thinning of wires adds latency, increasing power consumption of the network and the multicast and broadcast traffic from coherence severely affect the interconnect performance. Researchers have worked to suggest other alternatives for inter-chiplet interconnection such as silicon photonics and 3D ICs but they pose challenges of their own. Further, the need to design a coherence protocol for such alternatives is a complex task. Another alternative on the rise is wireless interconnects which constitutes of placing antennas on the package for inter-chiplet communication. This thesis focuses on exploring the feasibility and advantages of integrating wireless interconnects into cache-coherent chiplet-based architectures. Although there is a stack of layers to integrate wireless on the package, we focus our work on the communication stack which deals with packet latencies and performance. We suggest multi-chiplet architectures where 16 cores are segmented in 4 chiplets and 64 cores are segmented in 4 and 8 chiplets and baseline systems which are monolithic and contains 16 and 64 cores. Through extensive full system simulations of these systems with multiple inter-chiplet latencies we debugged and obtained traffic data. This data is analysed in terms of spatial, temporal and time variance. Our analysis shows the traffic is highly bursty for inter-chiplet communications. We observed that chiplet scaling degrades performance for all applications considered. The inter-chiplet communication through wireless medium for Token passing protocol on the traffic obtained from wired simulations is performed. Hybrid wired and wireless interconnects offer potential performance benefits without modifying the coherence protocol. The findings from

this research will contribute to the design and optimization of cache-coherent chiplet-based architectures, shedding light on the practicality and advantages of utilizing wireless interconnects in future computing systems.

Keywords: Chiplet design, Wireless interconnects, Performance optimization, Latency.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As technology continues to advance, the demand for high-performance computing systems has grown exponentially. Traditional monolithic processors are facing limitations in terms of scalability and power efficiency. In response to these challenges, chiplet-based architectures have emerged as a promising solution [29]. Chiplet-based architectures involve integrating multiple smaller chips, known as chiplets, onto a single package, enabling better scalability and customization.

One critical aspect of chiplet-based architectures is the interconnect network that facilitates communication and data exchange between chiplets. The network-on-chip(NoC) have shown to mitigate the problems because of scalability and high-bandwidth requirement of the many core chips [8]. However, the growth in number of components is accompanied by increased latency and power consumption [30]. Integration of heterogeneous processing elements further adds to interconnection performance due to diverse traffic patterns of different architecture and their interactions [23].

To address these challenges, researchers have explored optimizing interconnects in terms of topologies, physical characteristics and routing. While metallic inter-chip interconnects present scaling problems, recent research have brought to light emerging alternative such as wireless interconnects[34]. By utilizing wireless communication, it is possible to potentially reduce the overhead associated with traditional wired interconnects in terms of latency for distant communication. This approach offers the potential for improved performance, reduced latency, and increased energy efficiency [2][34].

This thesis aims to investigate the feasibility and advantages of integrating wireless interconnects into cache-coherent chiplet-based architectures.

The goal is to characterize wired chiplet based communication traffic and explore the potential performance benefits offered by wireless communication. Through simulations, analysis, and evaluation of wireless protocols for chiplet traffic, the research will contribute to the development of cache-coherent chiplet-based architectures, providing valuable insights into the practicality and advantages of utilizing wireless interconnects in future computing systems.

1.1 Background and motivation

Chiplets

In the past, enhancing the performance of computing systems involved increasing the number of transistors and the frequency of integrated circuits (IC). To fulfill the demands for greater computing power, energy efficiency, and lower costs in diverse applications, researchers have proposed architectural innovation and technology scaling as means to achieve these objectives. Computing systems have evolved from single-core to multi-core, including both homogeneous and heterogeneous multi-core designs. However, traditional approach faces several challenges:

- escalating costs
- a rapid rise in leakage power
- degradation of scalability
- increased complexity in system design, which hampers improvements in computing systems

To address these issues, Chiplet, a small-scale hard IP with high yield and reusability [27] [28], has emerged as a promising solution for computing system architectures [7,8]. By leveraging Chiplet, computing system designs combine the benefits of technology scaling, three-dimensional (3D) integration technology, and novel devices to construct high-performance computing systems, offering several advantages:

- reduced design costs through smaller area requirements and higher yield Figure 1.1

- shortened design cycles through Chiplet reuse
- enhanced system scalability through flexible Chiplet combinations [29]

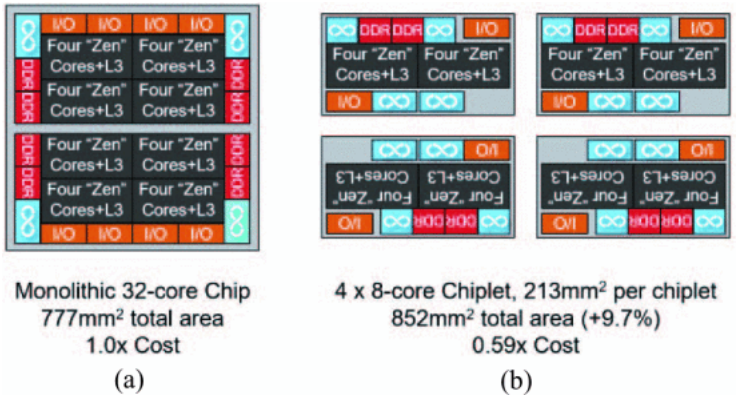


Figure 1.1: Hypothetical monolithic 32-core chip compared to an assembly of four eight-core chiplets. [28]

While the chiplet approach has a lot of advantages, it comes with its own complexities. More engineering, correct partitioning of the die and most importantly a new inter-chiplet communication path are some of the challenges associated with chiplets [28] [30].

Interconnect

Connecting increasing number of cores in a scalable way is achieved through networks-on-chip. In recent years, packet switched NoCs have replaced the traditional bus-based and crossbar-based interconnects providing a scalable communication fabric between multiple cores on a single chip. NoCs are composed of a topology of routers which are connected with each other through point-to-point links. Routers use buffers to store flits which are division of packets and their size is defined by the network link width [10]. The topology defines how routers are connected; a typical NoC architecture is shown in Figure 1.2 with a mesh topology.

Chiplet interconnect is another aspect of interconnecting. It involves heterogeneous integration that integrate multiple homogeneous dies. An important problem in integrating chiplets is to establish efficient communication which determines the performance and functionality of the entire chip. The

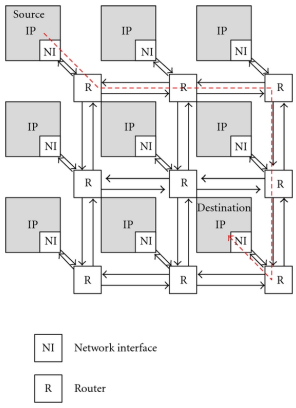


Figure 1.2: Typical NoC architecture in a mesh topology [33]

interconnection Hence designing protocols for this purpose is complex[25]. Figure 1.3 shows inter-chiplet interconnect established through Network-on-Interposer fabric.

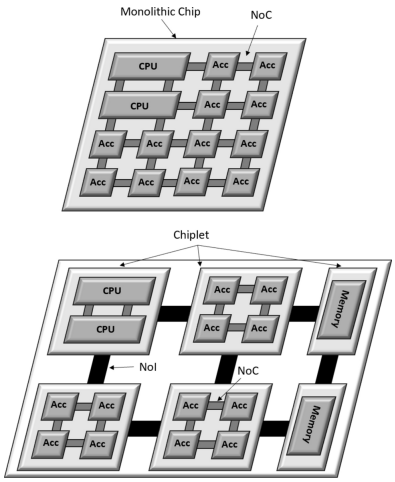


Figure 1.3: Monolithic and interposer based multi-chiplet system [25]

The wired approach of connecting routers in any topology has its challenges. Physical characteristics such as increasing wire resistance due to thinning of wires adds latency, increasing power consumption of the network and the multicast and broadcast traffic from coherence severely affect the interconnect performance. Since better performance achievement is a limi-

tation with the traditional interconnect systems other alternatives are being constantly researched. These include inter-chip photonics [37], vertically integrated 3D ICs [32] or silicon interposers [24] and wireless interconnects [3, 26, 16, 17].

Although the alternatives listed provide a way out, they have specific technology challanges. Integrating inter-chip photonics with existing electronic systems is challenging due to fabrication processes. Also they suffer from signal losses and are sensitive to temperature. Similar fabrication complexity, thermal sensitivity and power overhead issues arise in 3D stacking and RF circuit implementation [1].

Motivation

The motivation behind the thesis lies in addressing the challenges associated with interconnects in chiplet-based architectures. Communication within components is crucial for component scalability while keeping power, area and latency optimal and traditional wired inter-chip interconnects will not keep up with these demands.

Each of the solution discussed in previous subsection for NoC hybridization has its own benefits and challenges. A promising candidate is wireless interconnects due to its several advantages such as transmission distance independence, reconfigurability in design process, logical topology modification without touching the physical topology and improved scalability in terms of latency throughput and energy consumption. A hypothetical representation of a hybrid wireless and wired interconnect is shown in Figure 1.4.

The wireless interconnects build the inquisitiveness to explore the feasibility and potential advantages of incorporating them into cache-coherent chiplet-based architectures. By leveraging wireless communication, it is expected that the overhead associated with wired interconnects, specifically for communications between distant components can be reduced.

1.2 Objectives

Driven by the outcome of research conducted in the paper [17] that latency, and not bandwidth, is the primary performance constraint, through simulations and analysis, the research seeks to characterize the traffic in chiplet-based architectures and study the impact of wireless interconnects to assess

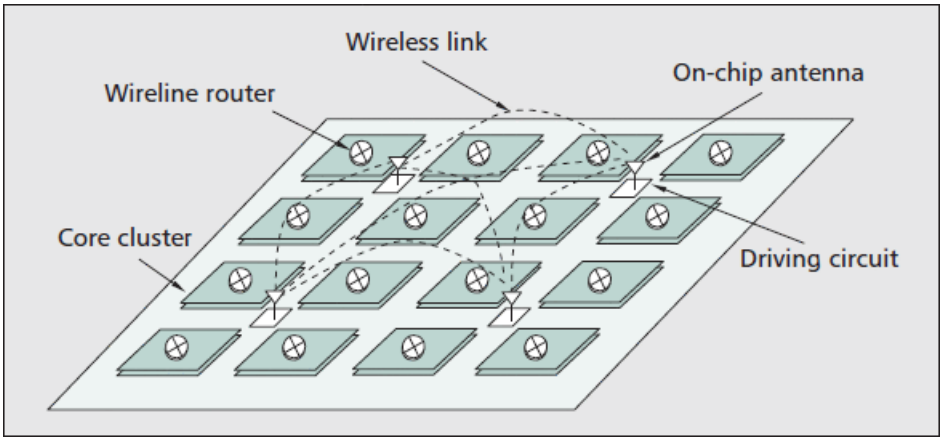


Figure 1.4: Schematic diagram of a hybrid wireless and wired NoC [1]

their potential performance benefits.

The objectives for the thesis are:

1. Characterize Wired Interconnect Traffic: Analyze and characterize the interconnect traffic in chiplet-based architectures with wired interconnects. Study packet distribution patterns, latency, and burstiness to guide the path for both wired and wireless interconnects.

2. Investigate Wireless Interconnect Feasibility: Explore the feasibility of integrating wireless interconnects into cache-coherent chiplet-based architectures and study the potential advantages and challenges of using wireless communication for inter-chiplet communication.

3. Assess Advantages and Trade-offs: Analyze the benefits and trade-offs of using wireless interconnects in cache-coherent chiplet-based architectures. Identify scenarios where wireless interconnects provide performance advantages over wired interconnects and understand the limitations or challenges associated with wireless communication.

By addressing these objectives, the thesis aims to contribute to the understanding and optimization of cache-coherent chiplet-based architectures, and provide insights into the feasibility and advantages of utilizing wireless interconnects to enhance system performance.

1.3 Contribution

The contribution of this work is listed in this section. We design the experiment for conducting traffic analysis of chiplet-based architectures. For this we design configurations for 16 and 64 core systems. The designs are explained thoroughly in chapter 3.

Following this design, we performed full system simulations to obtain traffic and other output. We conduct traffic analysis for the inter-chiplet communication for all the configurations and from various statistical operations obtain and explain the results obtained. These results would help understand the nature of inter-chiplet traffic in terms of temporal variation,spatial variation and type.

The plots in chapter 4 are representative of the traffic flow in all the system configurations designed. Temporal variance is depicted through histograms of the application execution which provides a intuitive visual sense of the variation. Statistically the temporal variation across all applications is shown and inference from the values is explained. Spatial variation is shown through heatmaps which again gives a visual sense of inter-chiplet traffic across all designs for all applications.

Apart from the traffic analysis general performance and latency comparison across designs is also presented. The full system simulation output is presented to help understand the application performance and latency for various inter-chiplet designs. The scaling trend of applications from 16 to 64 cores in monolithic and chiplet based fashion is represented through plots.

1.4 Organization

This section describes the organization of this work in this report. The next chapter 2 presents the review of literature research done for this work. It highlights research which is in line to the work of this study. It also describes the state of the art in chiplet-based interconnects and explains how this works adds to the given field.

chapter 3 is the backbone of this research. In a detailed view it presents the procedure followed for conducting the research. It starts explaining from designing the experiment, the benchmarks and simulators used, the evaluation methodology and statistical parameters considered. With sensible figures, tables and output descriptions it gives a comprehensive overview of

this work.

The evaluations of this work are presented in chapter 4. Characterization of the communication traffic in terms of spatial, temporal and variance profile is presented with plots. The performance evaluation is done for each benchmark in terms of running time inside the simulation. The designed experiments are evaluated in terms of average packet latencies.

Finally the chapter 5 ties the outputs of this work together. It describes the experiment performed in brief. Then it summarizes the main takeaways from the evaluations part. It then explains the feasibility of the topic addressed and provides arguments in its support. It also points the direction for future work and better experimentation design.

Chapter 2

Literature Review

2.1 Related work

[17] is one of the works which lies closely in line with the work of this thesis. Their work is very broad and tries to cover every aspect from wired to wireless including physical layer to network layer, of the wireless interconnect. It presents a hybrid inter-chip system where both CPU and GPU are considered. However it focuses more on the physical layer and traffic characterization is not a part of it.

[9] is an interesting work presenting a write-update coherence protocol for future chiplet based systems. It provides a good solution for interconnecting the chiplets through a new coherence protocol. When compared with the wireless part, our work does not modify the coherence protocol which is an advantage as the incorporation into chiplet-based systems would be relatively easy. [31] presents the statistical model of characterizing the traffic for any communication and is an important source of this work. Several analysis techniques were derived from this work to support our characterizing for the future wireless systems.

2.2 State of the art

Both chiplets and chiplet interconnects are currently emerging areas of research. There is extensive research being done on chiplet-based architectures both in industry and academia. The state of the art in chiplet-based architectures is still evolving[20][4][23], but there are a number of companies that

are developing and using this technology. Some of the leading companies in this space include AMD[28], Google[11], and Nvidia[13].

AMD was one of the first companies to adopt chiplet-based architectures with its Zen 2 architecture[27]. This architecture uses a modular design that allows different chiplets to be combined to create a variety of different processors. AMD's EPYC architecture is another attempt to strength its position in chiplet design approach[28]. Nvidia is also developing chiplet-based architectures for its GPUs. Its Hopper architecture[13] uses a design that combines multiple chiplets with different types of processing elements. This allows Hopper GPUs to offer a significant increase in performance over previous generations.

State-of-the-art in-package memory techniques, such as 2.5D integration (where memory is placed on the same substrate or interposer) [38] or 3D integration (involving vertical stacking of memory over the processing die) [19], have been developed to address the challenges associated with off-chip memory communication. These techniques aim to provide low-latency and high-bandwidth memory access. However, both approaches face certain drawbacks, including high power/thermal density, limited memory size, and reduced reliability and yield caused by factors such as through-silicon vias, thin wafer sizes, and layer misalignment [22].

Silicon photonics is also a rising contender for connecting chiplets together. In [21] the authors present a photonic cache coherence network for chiplet-based manycore systems. [14] also proposes the use of integrated silicon-photonic interconnects and exploit this fabric for scalable uniform memory architecture. Integrating silicon photonics in the package substrate is however a complex problem.

The state of the art in wireless interconnects for chiplet-based architectures is also still evolving. However, there are a number of promising technologies that are being developed[26][17][3]. Within wireless interconnects, several works like graphene antenna design[1], MAC protocols for wireless[15], network-on-chip[34][12] and on the area and scalability of [2] have been done.

The use of chiplet-based architectures and wireless interconnects is still in its early stages, but it has the potential to revolutionize the semiconductor industry. By allowing different chiplets to be combined and interconnected in a variety of ways[16], chiplet-based architectures can offer a significant increase in performance, flexibility, and power efficiency. Wireless interconnects can further improve performance and flexibility by allowing chiplets to be interconnected over a wide range of distances.

As these technologies continue to develop, we can expect to see them used in a wider range of applications, from high-performance computing to consumer electronics.

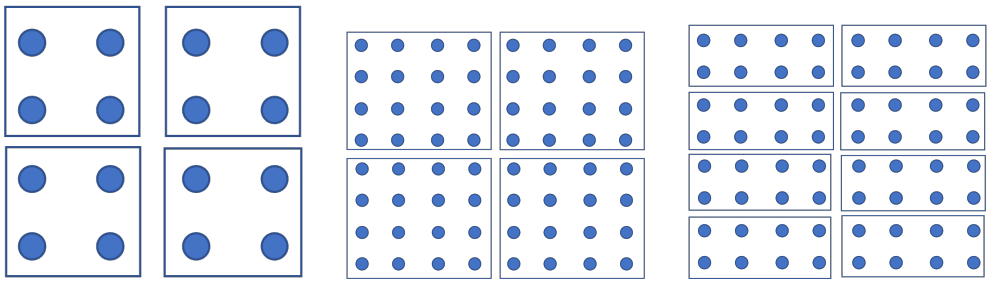
Chapter 3

Methodology

The chapter describes the experimental setup, the process flow and the configuration parameters adopted for this work in detail. The design of experiment describes the terms, models, parameters and their range.

3.1 Design of experiment

We selected two systems with 16 and 64 cores each to simulate many core systems. To model chiplet based architectures, we segmented the systems into chiplets as shown in Figure 3.1. For 16 core systems, each chiplet contains 4 cores. For the 64 core systems, there are two architectures: a 4 chiplet system with 16 cores per chiplet and an 8 chiplet system with 8 cores per chiplet. As a baseline, we have a single chip version of each 16 and 64 cores.



(a) 16 cores with 4 chiplets (b) 64 cores with 4 chiplets (c) 64 cores with 8 chiplets

Figure 3.1: Chiplet based architectures for 16 and 64 core systems

3.1.1 Modeling wired interconnects

All the three architectures are connected in MESH topology and a unit is represented as a tile. Each tile encapsulates a core, private L1 and L2 cache, a network interface and a router. The inter-chiplet communication is studied by choosing a discrete range of latencies for each of the architectures stated above. Two different latencies were chosen as mentioned in Table 3.1:

- inter-chiplet link latency = 10 times intra-chiplet link latency
- inter-chiplet link latency = 100 times intra-chiplet link latency

This allows us to study the trend of traffic when different components are chosen for the package as chiplets and not just processor cores.

Table 3.1: System Configurations

Cores	**Topology**	**Chiplets**	**Latency** (inter-chiplet relative to intra-chiplet)
16-core System	Mesh	None	
	Mesh	4	10
	Mesh	4	100
64-core System	Mesh	None	
	Mesh	4	10
	Mesh	4	100
	Mesh	8	10
	Mesh	8	100

3.1.2 Modeling wireless interconnects

Considering the above organization of chiplets, the inter-chiplet links are to be replaced by a wireless interconnect system. Each chiplet is considered to possess an antenna for communication with the other chiplets. This is true for every architecture containing chiplets in some configuration.

We adopt the MAC protocol for wireless communications in long range for our design. This ensures that the communication is deadlock and collision free. For now, we only implement Token Passing protocol [35].

3.2 Experimental Setup

The important parameter to study is the inter-chiplet traffic of real applications under real hardware. However the time and cost to design and test in reality is not feasible so we resort to architectural simulators. The design of the experiment above results in eight different system configurations which need to be studied. To get the quality of traffic which represents reality we need simulators which are detailed in architectural simulations. Further, the interconnection simulated must be accurate and precise at least at the packet level and should allow to simulate the topologies described in above sections.

To obtain the traffic, the criteria for applications which need to be studied are: common usage, comparable with other research work and small running time but effective benchmarking. Small running time is required as the simulations are quite large and take days to finish but this must not limit the quality of traffic produced. For simulating wireless communication, a protocol oriented and trace compatible with the wired simulator is needed.

To orchestrate the whole experiment a simulation flow is created. Figure 3.2 shows the process flow diagram followed for the experimentation. The orange box with dotted line is the simulation infrastructure, i.e. the simulators used and their compatibility. The incoming arrows in the orange box are the inputs and input parameters to the simulators and the outgoing arrows indicate the output produced. The traces produced by the full system simulation for the wired part are extracted so that only inter-chiplet traffic is left. This is then fed to the simulator running the wireless part. The details for each of the simulators and input parameters are explained in the subsection 3.2.2. The output files, statistics and the Extract block between the two simulators is thoroughly explained in section 3.3.

3.2.1 Benchmarks

For the purpose of this research, a benchmark would have ideally the following characteristics:

- Multi-threaded: to stress the chiplet design and get characterization traffic for a scaling system
- Diverse: applications are diverse in the real world and the benchmark must accommodate for this diversification through its choice of applications

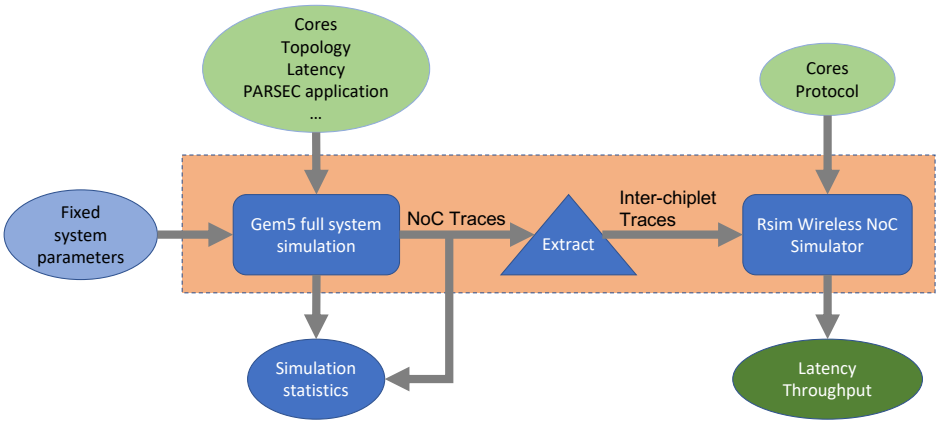


Figure 3.2: Simulation flow

- Supports Research: it should be tunable to the needs of the research and architectural specification
- Communication to computation ratio: our work is focused on the communication part more than the computation part hence a good percentage of communication is required in the applications

Keeping in mind the above points several choices are available. The obvious ones are PARSEC , SPLASH-2 and SPEC CPU2006. SPLASH-2 [36] is a benchmark suite composed of multi-threaded applications but is geared towards HPC[6]. SPEC CPU2006 is also a significant collection of benchmarks but is not intended for studies of parallel architectures[6]. PARSEC was chosen to be our suite of applications and we handpicked the applications which are listed in Table 3.2 along with the time they took for simulation in our work. There are total 13 applications in the PARSEC benchmark suite. However not all of them are considered for full system simulation in this work, the reason being it was either not communication intensive or for 64 core system the simulation could not finish due to memory and timing constraints. However, the selected eight benchmarks tick all the characteristics listed above.

The PARSEC benchmarks offers six input sets for each benchmark: test, simdev, simsmall, simmedium, simlarge and native. For studies pertaining to microarchitectural research simsmall, simmedium and simlarge are suitable. All the benchmarks for each of the system configurations were run with the

Table 3.2: Simulated Benchmarks

Benchmark	Simulation time
blackscholes	small
bodytrack	medium
canneal	large
dedup	large
ferret	medium
fluidanimate	large
freqmine	medium
vips	large

simsmall input set. This size is a good balance between the obtained trace quality and the simulation running time.

`parsecmgmt` - The PARSEC management tool is a good option for building and running applications. Initially when creating the disk-image for gem5, `parsecmgmt` is used for building all the packages. Further, when passing a script to the gem5 running instance it is used to set the test input and specify the benchmark to run after boot process is finished.

3.2.2 Simulators

To evaluate the proposed chiplet based architecture, we use gem5 [7]. gem5 is a simulation infrastructure which allows modeling hardware at the cycle level and has enough fidelity to boot unmodified Linux-based operating systems and run full applications for multiple architectures. It is dynamically configurable through a robust Python based scripting interface. The main feature for this work was to run unmodified applications with cycle-level statistics in full system manner which is supported by gem5. gem5's design is modular which means components such as the memory system, CPU model and coherence protocols can be modeled according to the design.

The interconnection network can also be configured to model a variety of systems due to gem5's modular design. Currently it supports classic caches and the Ruby memory system. The Garnet network model(HeteroGarnet) [5] is present within the Ruby memory system and due to its level of details and flexibility we chose it for modeling our interconnection topologies.

gem5 is used with HeteroGarnet to model all system configurations for

full system simulations. The base system configuration shown in Table 3.3 is same for all the architectures under study.

Table 3.3: Fixed System Parameters

Parameter	Value
ISA	x86
CPU Type	TimingSimpleCPU
Simulation mode	Full System
L1[data/inst.] size, associativity	32kB, 4
L2 size, associativity	256kB, 8
Cache coherency	MESI_Two_Level
Kernel	x86-linux-kernel-4.19.83
OS	Ubuntu 18.04.2 LTS pre-loaded with PARSEC benchmark
Memory	512MB
Clock	1GHz
Interconnection Network	Garnet 3.0(HeteroGarnet)

A command similar to Listing 3.1 is used to execute a particular instance of a benchmark on a specific system configuration. As seen in the command, every parameter is an input to the gem5 build file[line:1] which is specific for x86 ISA. The gem5.opt is built beforehand for specific ISA and Coherence protocol.

Listing 3.1: A typical gem5 run command

```
1  ./build/X86/gem5.opt \
2  -d $GEM5_OUTPUT_DIR \
3  --debug-flags=$DEBUG_FLAG \
4  --debug-file=$DEBUG_FILE_PATH \
5  configs/example/fs.py \
6  --checkpoint-dir $CHECKPOINT_DIR_PATH \
7  -r 1 \
8  --restore-with-cpu TimingSimpleCPU \
9  --disk-image=/path/to/disk-image/x86-parsec \
10 --kernel=/path/to/kernel/x86-linux-kernel-4.19.83 \
11 --cpu-type TimingSimpleCPU \
12 --num-cpus=${NUM_CPUS} \
```

```
13 --caches \
14 --l1d_size='32kB' \
15 --l1i_size='32kB' \
16 --l1d_assoc=4 \
17 --num-l2caches=${NUM_CPUS} \
18 --l2_size='256kB' \
19 --l2_assoc=8 \
20 --num-dirs=${NUM_CPUS} \
21 --mesh-rows=${MESH_ROWS} \
22 --ruby \
23 --network=garnet \
24 --topology=$TOPOLOGY \
25 --script=$RUN_SCRIPT_PATH
```

HeteroGarnet is attached to gem5 simulation with the `--network=garnet` option along with `--ruby`. The desired topology is provided with `--topology` keeping the `--mesh-rows` value comptabile with the topology. More details on the use of HeteroGarnet and obtaining traces is presented in section 3.3.

The benchmark to run is given to `--script`(bash in our work) to gem5.opt. The script contains `parsecmgmt` with application name and the input size as shown in Listing 3.2[line:4].

Listing 3.2: Script passed to run on guest OS

```
1 #!/bin/bash
2 cd /home/gem5/parsec-benchmark
3 source env.sh
4 parsecmgmt -a run -p bodytrack -c gcc-hooks -i simsmall
     -n 16
5 sleep 5
6 m5 exit
```

RSim

RSim is an in-house wireless network simulator designed for studying wireless interconnect traffic. It accepts the trace in the format `cycle, source_node`. It takes as an input the number of nodes and the Medium Access Protocol(MAC) protocol to use for transmission of packets. Other inputs are the buffer size of each node, transmission rate of each node and maximum number of packets.

Each node acts as an antenna transmitting packets. The latency of a packet is calculated as the difference in time of arrival and time of injection. Every node stores the statistics of each packet it transmitted and the whole system statistics are calculated accordingly. The transmission occurs according to the MAC protocol.

To give RSim the trace generated by gem5 as input, it has to be filtered so that RSim accepts the format of each packet. We slightly modified the Ring-Token(Token) protocol implemented in RSim to also take into account the number of flits in a packet. Hence, for packets containing more flits each node takes more time to transmit.

3.3 Characterization and Analysis

Statistical analysis is important for any research dealing with data. It helps analyse the data, test the hypothesis on which the research started, evaluate the performance and to design the experiments. To perform these statistical analysis, we first need obtain and clean the data so that the tests can be performed. Furthermore, a single dataset may be filtered and used for multiple analysis evaluating entirely different parameters. The following sections describe the process by first defining what kind of data we need, obtaining it and the procedures involved, gathering it together and filtering it for different purpose.

3.3.1 Obtaining Traces

The base of our study has been to improve the performance of many core systems. The research focuses primarily on the interconnection network and the simulation infrastructure setup also indicates the same. The level of depth in this study has been at the system level, i.e. we do not dive into the physical layer of interconnect communication stack. This suggests that we should collect data at this level through our simulators.

The interconnection network acts as the backbone of the system connecting all the memory components together and all the communication happens through it. Hence it is an ideal choice to study and obtain the required traces from this network. Diving further into the implementation details we can see in Figure 3.3 the life cycle of a message which is generated from one of the source cache controllers. At this point we have enough level of detail

to capture the trace.

The source cache controller generates a message and designates one or more cache controllers as the recipients. This message is subsequently placed into message queues. Each cache controller typically possesses multiple outgoing and incoming message buffers, which are utilized for handling various types of messages. A Network Interface(NIC) is attached to each controller which wakes up and checks these buffers to convert it to flits and transmit them. We place our debug filter right in the NIC before conversion to flits. This is done because once the message is *'flitisized'*, the broadcast packets are converted to unicasts which will not be helpful as we are considering to transmit them through wireless links which are an excellent solution for broadcasting.

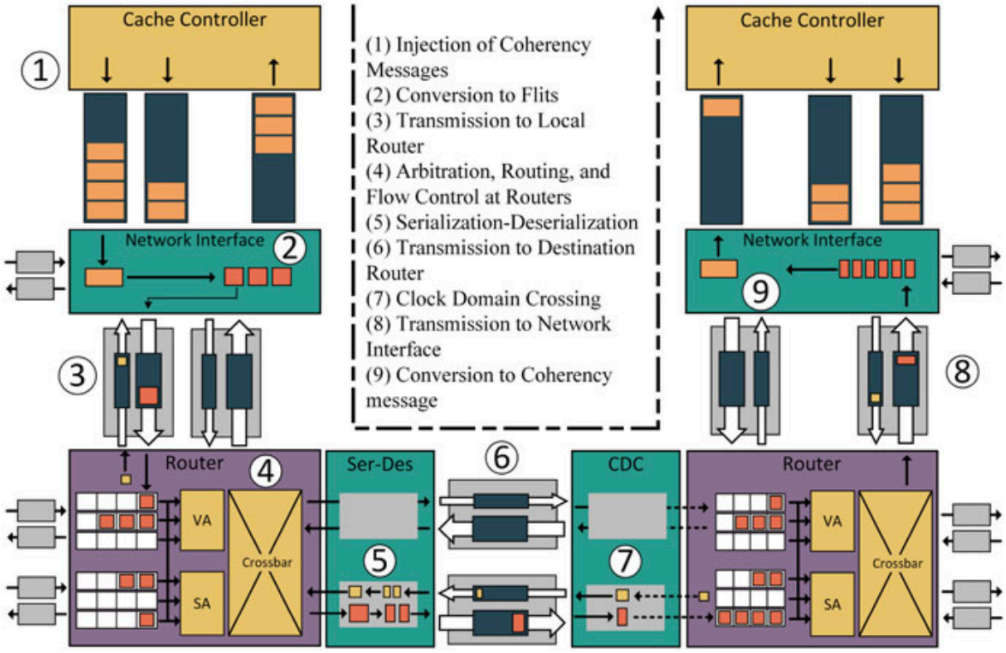


Figure 3.3: Life cycle of a Coherence Message [18]

Debug Trace

The code in Listing 3.3 is placed in `NetworkInterface.cc` file of the `Garnet` module under `Ruby` in the `gem5 source files`. The code is a `DPRINTF`

statement, meaning it is used for debugging purposes in gem5. When the Network Interface connected to a cache controller wakes up and deques the message from message buffer, we print that message into the debug trace using the `DPRINTF` statement. The first argument to this statement is the debug flag which can be activated by passing it to the gem5 run command. The following arguments define the desired output format. The format we chose is shown in Figure 3.4. It contains five fields indicating the cycle number, the router which is sending this message, the destination router, the number of flits this message will be divided into and the type of message respectively. As shown in Figure 3.2, when this trace is passed through the extractor only three fields are kept: the cycle number, the source and the number of flits.

The size of the trace file for a given application in a given configuration is several Gigabytes. There were 64 of these trace files resulting in more than 100 Gigabybtes of data which had to be processed for obtaining meaningful results. Apart from this output, gem5 generates `stats.txt` for each run which is a great source of obtaining statistics. The next section describes this helpful feature of gem5.

Listing 3.3: Debug traces at message level in `NetworkInterface.cc`

```
DPRINTF(MESSAGE, "%ld %d %d %d %s\n ",
ticksToCycles(curTick()),
route.src_router,
route.dest_router,
num_flits,
MessageSizeType_to_string(net_msg_ptr->
   getMessageSize()));
```

Cycle	Src Router	Dest Router	Num. Flits	Message Type
16030487788	1	13	1	Control

Figure 3.4: Trace output format from gem5

Default gem5 output

Each gem5 run by default produces `stats.txt` containing most of the system parameter values and various statistics. A sample is shown in Listing 3.4. A component wise statistic is printed in each line such as in line 6 where the ruby component lists the input of the network module which outputs the average packet latency of the whole network. The important thing is that garnet which is defined under the Ruby also dumps its statistics into `stats.txt`. All the necessary statistics like this are obtained from this file and are processed to produce the results. Also a distribution of number of packets from each source router to each destination router is produced which complements our trace obtained from the debug flag and is filtered and passed to Rsim for further experimentation.

Listing 3.4: A sample `stats.txt` generated by a completed gem5 run

```
---------- Begin Simulation Statistics ----------
simSeconds                                   9.789171
              # Number of seconds simulated (Second)
simTicks                                  9789170927751
             # Number of ticks simulated (Tick)
.
.
system.ruby.network.average_packet_latency    22.178867
...network.average_packet_network_latency     20.290069
.
.
...network.ctrl_traffic_distribution.n0.n15   255202
...network.data_traffic_distribution.n6.n13   613599
.
.
---------- End Simulation Statistics   ----------
```

3.3.2 Evaluation metrics

The traffic obtained from gem5 for different applications on different architectures holds critical information about inter-chiplet traffic which is obtained from the analysis we perform. We perform both charazterization and performance metrics for our data. This sections describes the various tests performed.

The main characteristics of the workloads traditionally exhibited by multithreaded applications in single-chip multiprocessors are heterogeneity, variability, spatial hotspot behavior, and temporal bursty behavior. In more detail:

- **Temporal bursty behavior:** With regards to the trace analysis, we first note that communication data traversing an interconnection network exhibits temporal variance [31] . In other words, the level of burstiness or packets per unit time is not constant and varies across applications. We thus parameterize this level of burstiness using a single parameter, the Hurst exponent H. Self-similarity is exhibited where $0.5 < H \leq 1$. There are a number of tests to measure H as self-similarity manifests itself in a number of ways. Here we use time-domain analysis based on the re-scaled adjusted range statistic, known as the R/S statistic. To obtain H, one plots $log_{10}\frac{R(s)}{S(n)}$ versus $log_{10}n$. This is called an R/S *pox plot*, where the slope of the R/S line is H. This slope is calculated using an inverse-variance-weighted least-squares curve fit.
- **Spatial behavior:** To inspect the spatial distribution of packets in a trace we plot heatmaps representing the chiplets and their traffic generation. This will help us identify the hotspot chiplet in the traffic and whether it is a source or destination. Also combined with variability and burstiness, this helps in designing a protocol for wireless communication or selecting one.
- **Variability:** The existence of multiple applications means that there might be changes in terms of communication pattern from one application to another. Furthermore, the particular chiplet combination in a heterogeneous architecture influences such variability as different applications may require to use a particular accelerator chiplet intensely while others may not. This can be seen from a combination of the heatmaps and plotting number of messages per unit cycles.
- **Average Latency:** Packets crossing the inter-chiplet boundary are the ones which are supposed to be the major component of average latency of communication. We describe this trend through plots of all the benchmarks across different architectures. This will help identify which applications scale with the chiplet architecture and which perform worst with more chiplets.
- **Execution Time:** In conjunction with latency, the performance will

depend finally on the execution time of the application. We plot the various execution times of Region of Interest of applications across the architectures considered. Baseline for the execution time is the monolithic 16 core system and all the times are normalized to it.

3.4 Conclusion

In this chapter we defined our design for conducting the experiment. The experiment is setup with two systems of 16 and 64 cores. For each of the system we divided them into chiplets of reasonable size, i.e. 4 chiplets containing 4 cores each for the 16 core system and 4 and 8 chiplets containing 16 and 8 cores each for the 64 core system. Our key component is the interconnection network through which all the communication occurs. This configuration is first studied for the wired part with range of topologies for each of the system and then through the wireless part for obtaining similar output statistics for characterization and performance evaluation.

Detailing further, we defined the simulation flow from the gem5 full system simulation to Rsim wireless Network on chip simulator. We setup the simulation pipeline with fixed and variable parameters with trace propagation and filtering. Each part of the flow has been detailed to the system level describing significant details in form of code statements to trace formats. The benchmark applications and the criteria for choice is defined. Keeping all the variables in constraint with time, resource and complexity this chapter presents the essence of the entire research conducted. The following chapter brings to light the results, outputs, comparisons and outcomes of the designed experiment.

Chapter 4

Evaluation

The evaluation of inter-chiplet traffic plays a crucial role in understanding the communication dynamics within many core systems. In this chapter, we present a comprehensive analysis of inter-chiplet traffic for both 16-core and 64-core systems, considering various topologies and latencies. The goal is to gain insights into the communication patterns and performance implications of different interconnect configurations.

Throughout the sections we have collectively characterize and evaluate all the system configurations by juxtaposing them in different combinations. Given the significant impact of network traffic on network performance, the presence of comprehensive traffic models becomes crucial for a profound exploration of the extensive design landscape encompassing network architectures, protocols, and implementations [31]. Hence, characterizing the traffic's spatial and temporal profile becomes much important.

The time of execution for the type of application also becomes necessary to evaluate. It helps understand the scaling of architecture and its implication on the application's execution time. The speedup or slowdown can be inferred and can be linked to the latency of the interconnection network which can help us understand the potential impact of incorporating wireless interconnection networks. The same is true for the packet latencies of the interconnection network.

Driven by these concepts our evaluations for characterization concentrate on the following remarks:

- spatial and temporal profile of traffic using statistical parameters
- comparing inter vs intra chiplet traffic

- investigating the *type* of messages
- comparing execution times
- comparing average latencies

The parameter that applies to both the wired and wireless approach is mainly the average latencies. We assess the performance of wireless interconnections against wired counterparts, considering factors such as latency and throughput. By contrasting the two approaches, we aim to highlight the trade-offs and potential benefits of wireless interconnects in the context of inter-chiplet communication within many core chiplet based systems.

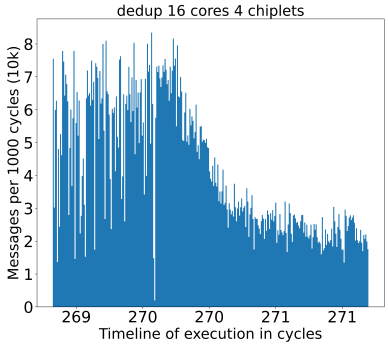
Through this comprehensive evaluation, we expect to gain valuable insights into the characteristics of inter-chiplet traffic in both 16-core and 64-core systems. The findings will not only contribute to the understanding of communication dynamics but also provide a basis for optimizing interconnect designs and exploring alternative interconnection technologies for future chiplet based architectures.

4.1 Characterization

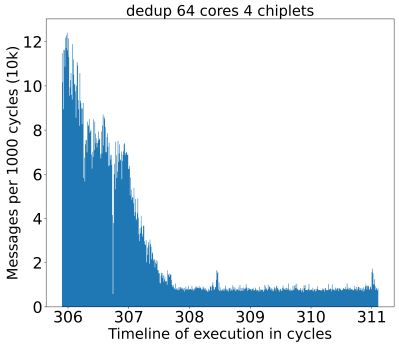
4.1.1 Inter chiplet traffic

Communication data traversing the interconnection network exhibits multiple type of variances. One such variation is with respect to time. To model such variation we can consider the number of packets traversing through the network in equally divided time intervals. Time in its best form is represented in cycles for the interconnection network and intervals could be of n cycles where n is chosen suitably so that it can represent the traffic count in visually comprehensible manner. This variation is shown in Figure 4.1. We can definitely see that in both the applications the traffic is highly bursty. Looking at different chiplet configurations of Figure 4.1, we see that application dedup's variance changes with number of cores and chiplets. Also the maximum number messages transferred grow by 50% per thousand cycles from 16 to 64 cores. These bursts are not evenly spread across the timeline indicating heterogeneity.

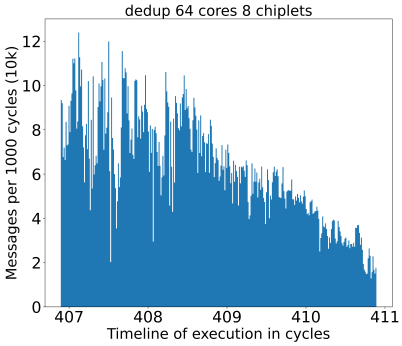
The Figure 4.2 shows the percentage of each type of message crossing the chiplet boundary for 8 chiplet system. 60-70% of the messages are Control and Response Data and this pattern is similar for other configurations.



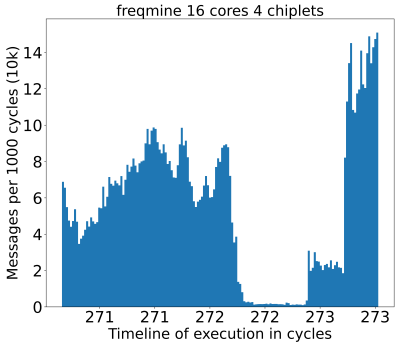
(a) 16 cores with 4 chiplets



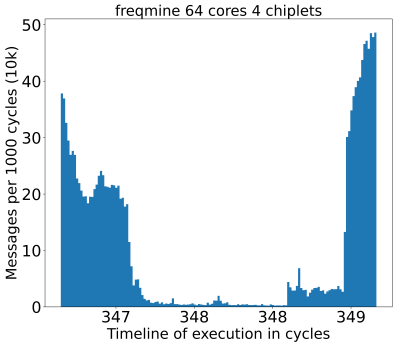
(b) 64 cores with 4 chiplets



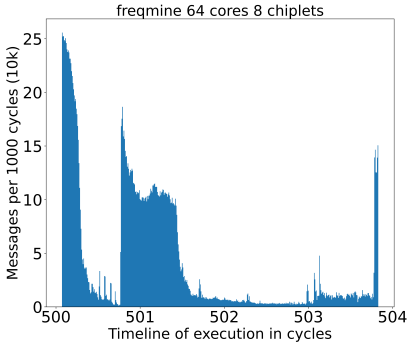
(c) 64 cores with 8 chiplets



(d) 16 cores 4 chiplets



(e) 64 cores with 4 chiplets



(f) 64 cores with 8 chiplets

Figure 4.1: Traffic in intervals of 1000 cycles for freqmine and dedup

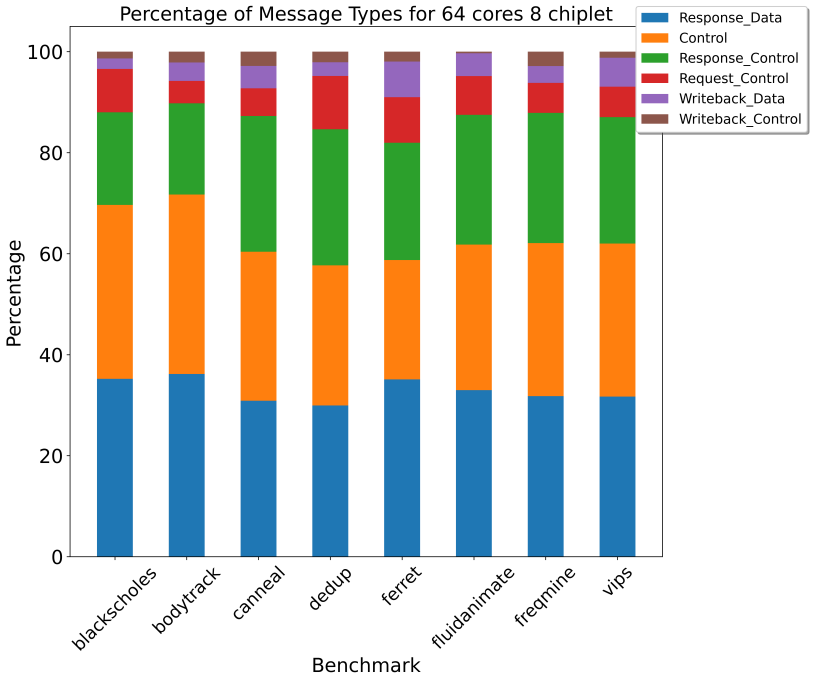


Figure 4.2: Message type characterization for inter-chiplet traffic

4.1.2 Spatial and temporal profile

Interesting results are obtained when spatial analysis is conducted for the communication. The plots below show the heatmaps for all the three configurations and all the applications. In Figure 4.3 we can see the there are a lot of messages from chiplet 1 to 3 in all the applications except canneal. There is variation amond the inter-chiplet transfers but general hotspot is chiplet 1 as a source for all applications. The pattern changes for the 64 core system with same chiplets Figure 4.4 with chiplet 4 acting as a hotspot in half of the applications, with chiplet 1 still the hotspot for the other half. We can also conclude that the number of messages reduced when the base system was scaled to 64 cores meaning less inter-chiplet traffic on average. The distribution is unequal among chiplets except for vips where it is of the same range. The dynamics change pretty abruptly for the 8 chiplet configuration Figure 4.5. All the chiplets transmit roughly equal amount of data but the destination chiplet 5 receives most of the messages for all applications.

Figure 4.6

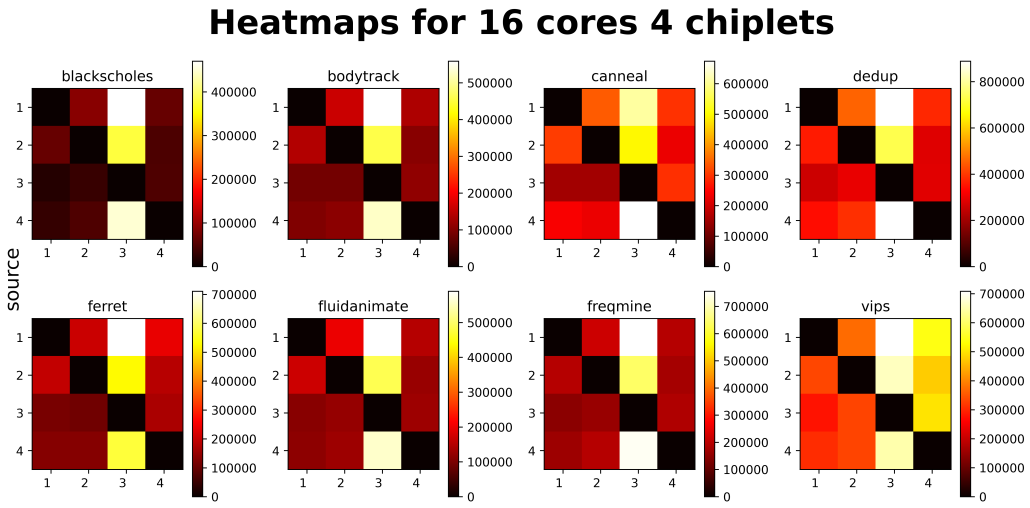


Figure 4.3: 16-4 chiplets

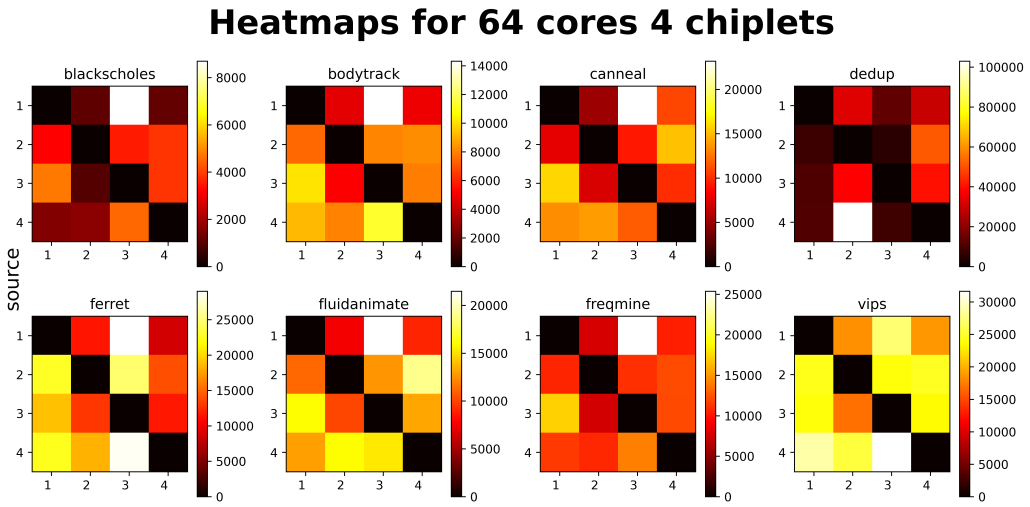


Figure 4.4: 64-4 chiplets

Figure 4.6 shows the hurst exponents of all applications for all chiplet architectures. First observation is that the communication traffic is highly self-similar as $H > 0.8$ for all applications. It means that the traffic shows bursts of communication followed by relatively long silences. 'bodytrack' and 'fluidanimate' show varying burstiness across different chiplet configurations

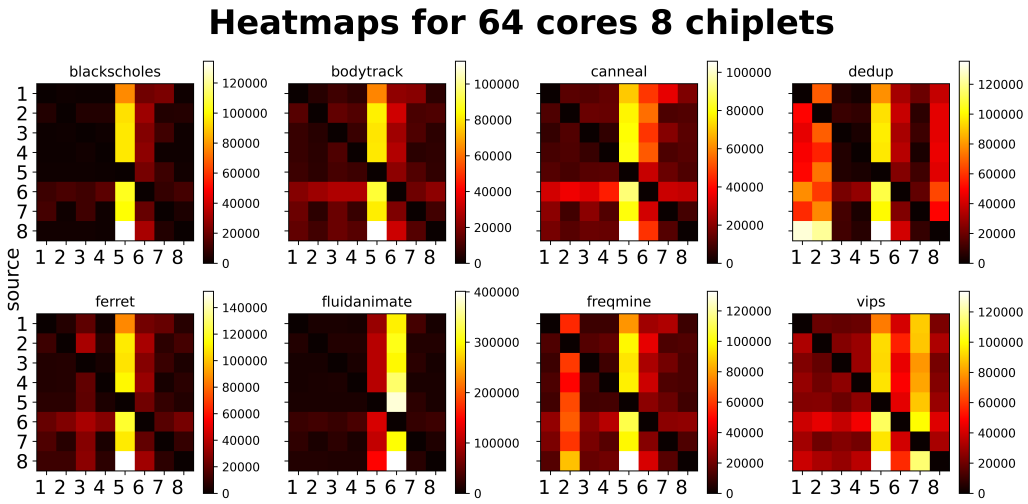


Figure 4.5: 64-8 chiplets

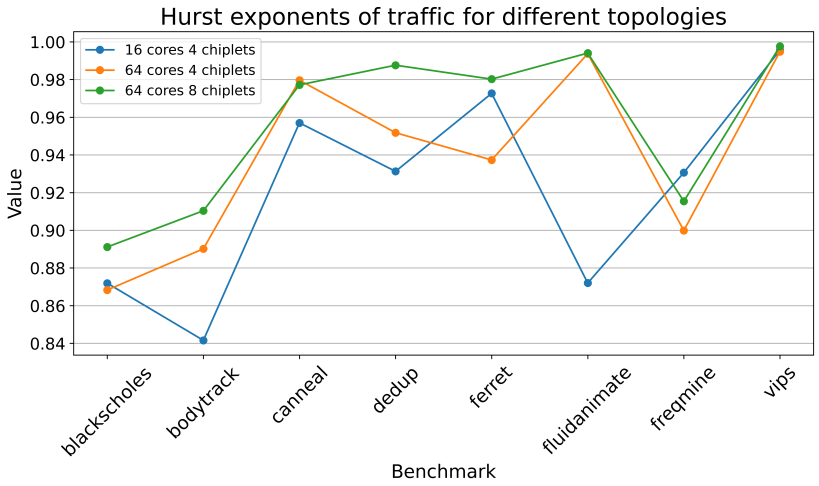


Figure 4.6: Hurst exponent for all applications accross multiple systems

as they become more bursty in 64 core system. Generally the 64 core system with 8 chiplets has the most bursty traffic for all applications except for freqmine meaning the chiplets communicate burstily for short time followed by relatively long silences.

4.2 Performance

4.2.1 Speedup or slowdown

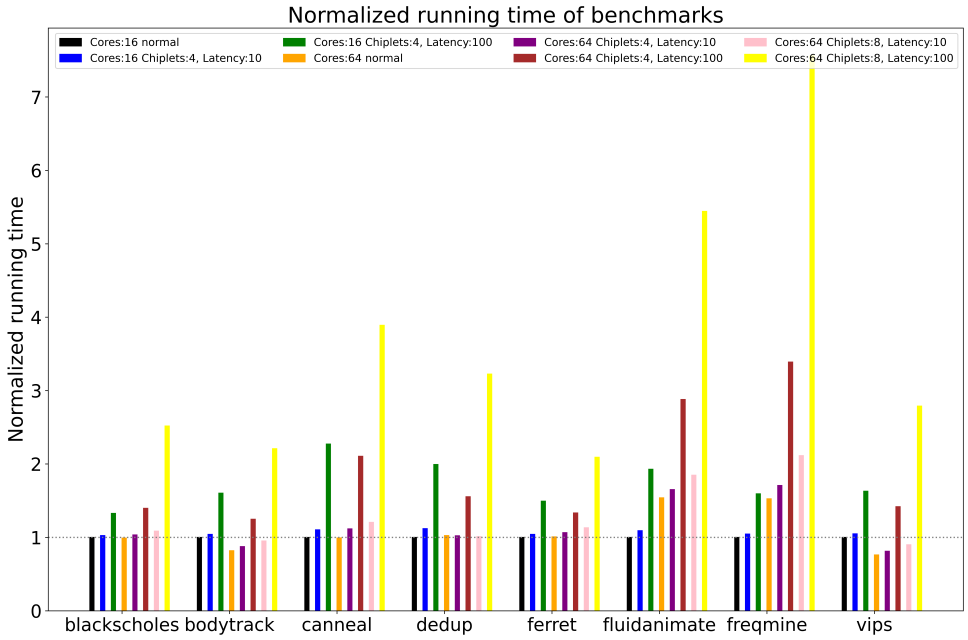


Figure 4.7: Slowdown of applications accross multiple systems

The execution time of applications on many core systems has been researched extensively. Here we present it for our chiplet based configurations in Figure 4.7. The runtime of each application is normalized to its baseline configuration, which is monolithic or normal 16 cores. General trend shows that applications run at least 2 times slower(the yellow bar) on an 8 chiplet system with the slowest being 7 times slower than the base. Further, chiplets indicate poor scaling on the same base system with 64 cores. It is slightly seen when 16 cores are segmented in 4 chiplets but evident in 64 cores. Applicatons blackschole, fluidanimate, freqmine lose performance when scaling from 16 to 64 cores under 4 chiplets(considering 100 latency) where all other applications ran faster on 64 core system with 4 chiplets. They however also ran slow on baseline 64 but so did canneal, dedup and ferret.

4.2.2 Latency

Figure 4.8 shows average packet latencies for application across all configurations (Notice the y-scale is logarithmic). One can definitely see a common trend: the latencies increase when the chiplets increase(purple and pink bars) under the 64 core system. Also the jump in network latency for a chiplet is not quantitatively progressed into average packet latencies, i.e when base network latency is increased 10 times, the packet latency does not increase 10 times. However, the jump is substantial and impacts the performance also. Segmentation of chiplets on the same system also adds to the average latencies, almost twice if chiplets are doubled. Figure 4.8

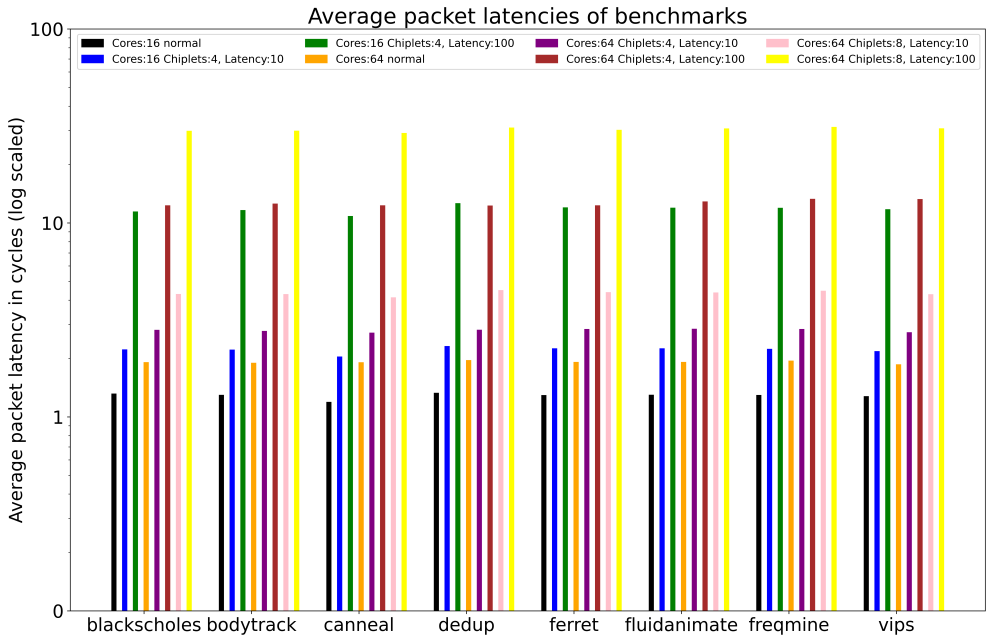


Figure 4.8: Average packet latencies of applications accross multiple systems

Average packet latencies for TOKEN protocol obtained with RSim are listen in Table 4.1. Such high latency values are because of incompatibilty of the trace file output and the RSim input parameters. Due to the design of RSim the node injection rate and packet distribution to nodes was for synthetic traffic generated by the values of Hurst exponent and coefficient of variation of the source nodes.

Table 4.1: Average packet latencies for inter-chiplet data(cores)

Benchmark	Cycles per Packet(16)	(64)
blackscholes	32.8307	224.296
bodytrack	587.901	170409
canneal	8.74244e+06	1.77433e+07
dedup	1016.42	44710.3
ferret	749.175	28900.1
fluidanimate	324020	229.056
freqmine	65.9824	8.05431e+06
vips	234.271	2.95392e+07

Chapter 5

Conclusion

We conducted our researching concerning different chiplet based designs considering different inter-chiplet latencies. Specifically 16 core system with 4 chiplets and 64 core system with 4 and 8 chiplets with 10 and 100 times inter-chiplet link latency for all configurations. We performed full system simulations with 8 applications from the parsec benchmark of varying size and communication to computation ratio. The trace output from all the simulations is used for obtaining multiple plots and graphs which cover spatial, temporal and burstiness of the communication traffic. The same trace is then filtered and sent to RSim for performing simulations for the wireless part.

Consolidating the evaluations and inferences from the above experiment, we found that around 70% of the traffic for an application is inter-chiplet and the rest is within the chiplet. This is found true for all the applications across all the configurations. Further we found that inter-chiplet traffic is highly self similar as the Hurst exponent is higher than 0.8 for all applications. The spatial profile of the traffic remains dynamic with change in chiplet configuration. The emergence of hotspots is application specific.

While looking at performance results we see the relative slowdown of application to the 16 core baseline. We find that chiplets do not scale well for benchmarks, especially on the 64 core system. The performance gain obtained by some application when running on base 64 cores than 16 cores is lost when the system is segmented in chiplets. The trends in average packet latency are not similar. There is no gain in average latency when the chiplets are scaled on the 64 core system. The increase in latency is not exactly the same when the inter-chiplet latency is increased 10 and 100 times. However,

they are significantly increased to hamper the performance in a substantial way.

The substantial increase in latency when going from base to chiplet architectures, even considering 10 times inter-chip latency with respect to the base link latency is evident in our work. The path for chiplets and their integration while keeping up the performance is tough. This paves the way for new advancements in interconnect technology. The average packet latency increase in intra-chiplet wireless interconnects is significantly less than the wired inter-chiplet part. Further, integrating wireless with the current wired interconnects will not require a change in the cache coherence protocol. This is a huge benefit as designing new coherence protocols and debugging them is a task of extreme complexity.

This work is a source of characterization of communication traffic for researchers looking for other interconnect technologies to create a hybrid interconnect for connecting chiplets. Experimentation with other MAC protocols will further decrease the average packet latency as token induces waiting periods in certain situations. Wireless interconnects are a viable solution to pave the path for a hybrid inter-chip interconnect.

Bibliography

[1] Sergi Abadal et al. "Graphene-enabled wireless communication for massive multicore architectures". In: *IEEE Communications Magazine* 51.11 (2013), pp. 137–143. DOI: 10.1109/MCOM.2013.6658665.

[2] Sergi Abadal et al. "On the Area and Energy Scalability of Wireless Network-on-Chip: A Model-Based Benchmarked Design Space Exploration". In: *IEEE/ACM Trans. Netw.* 23.5 (Oct. 2015), pp. 1501–1513. ISSN: 1063-6692. DOI: 10.1109/TNET.2014.2332271. URL: https://doi-org.recursos.biblioteca.upc.edu/10.1109/TNET.2014.2332271.

[3] Sergi Abadal et al. "OrthoNoC: A Broadcast-Oriented Dual-Plane Wireless Network-on-Chip Architecture". In: *IEEE Transactions on Parallel and Distributed Systems* 29.3 (2018), pp. 628–641. DOI: 10.1109/TPDS.2017.2764901.

[4] AmandaK. *Blue Cheetah Demonstrates Industry Leading Silicon-Proven Die-to-Die Interconnect Solution for Chiplets*. 2023. URL: https://semiwiki.com/forum/index.php?threads/blue-cheetah-demonstrates-industry-leading-silicon-proven-die-to-die-interconnect-solution-for-chiplets.18036/.

[5] Srikant Bharadwaj et al. "Kite: A Family of Heterogeneous Interposer Topologies Enabled via Accurate Interconnect Modeling". In: *2020 57th ACM/IEEE Design Automation Conference (DAC)*. 2020, pp. 1–6. DOI: 10.1109/DAC18072.2020.9218539.

[6] Christian Bienia. "Benchmarking Modern Multiprocessors". PhD thesis. Princeton University, Jan. 2011.

[7] Nathan Binkert et al. "The Gem5 Simulator". In: *SIGARCH Comput. Archit. News* 39.2 (Aug. 2011), pp. 1–7. ISSN: 0163-5964. DOI: 10.1145/2024716.2024718. URL: https://doi-org.recursos.biblioteca.upc.edu/10.1145/2024716.2024718.

[8] Tobias Bjerregaard and Shankar Mahadevan. "A Survey of Research and Practices of Network-on-Chip". In: *ACM Comput. Surv.* 38.1 (June 2006), 1–es. ISSN: 0360-0300. DOI: 10.1145/1132952.1132953. URL: https://doi-org.recursos.biblioteca.upc.edu/10.1145/1132952.1132953.

[9] Grigory Chirkov and David Wentzlaff. "Seizing the Bandwidth Scaling of On-Package Interconnect in a Post-Moore's Law World". In: *Proceedings of the 37th International Conference on Supercomputing*. ICS '23. Orlando, FL, USA: Association for Computing Machinery, 2023, pp. 410–422. DOI: 10.1145/3577193.3593702. URL: https://doi.org/10.1145/3577193.3593702.

[10] W.J. Dally and B. Towles. "Route packets, not wires: on-chip interconnection networks". In: *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*. 2001, pp. 684–689.

[11] Uday Kumar Dasari et al. "Apparatus and mechanism for processing neural network tasks using a single chip package with multiple dies". US 10,936,942 B2. 2021.

[12] Sujay Deb et al. "Enhancing performance of network-on-chip architectures with millimeter-wave wireless interconnects". In: *ASAP 2010-21st IEEE International Conference on Application-specific Systems, Architectures and Processors*. IEEE. 2010, pp. 73–80.

[13] Cliff Edwards. *NVIDIA Opens NVLink for Custom Silicon Integration*. 2022. URL: https://nvidianews.nvidia.com/news/nvidia-opens-nvlink-for-custom-silicon-integration.

[14] Pouya Fotouhi et al. "Enabling scalable chiplet-based uniform memory architectures with silicon photonics". In: *Proceedings of the International Symposium on Memory Systems*. 2019, pp. 222–334.

[15] Antonio Franques et al. "Fuzzy-Token: An Adaptive MAC Protocol for Wireless-Enabled Manycores". In: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2021, pp. 1657–1662. DOI: 10.23919/DATE51398.2021.9473960.

[16] Amlan Ganguly et al. "Interconnects for DNA, Quantum, In-Memory and Optical Computing: Insights from a Panel Discussion". In: *IEEE Micro* (2022).

[17] Amlan Ganguly et al. "The Advances, Challenges and Future Possibilities of MillimeterWave Chip-to-Chip Interconnections for Multi-Chip Systems". In: *Journal of Low Power Electronics and Applications* (2018). DOI: https://doi.org/10.3390/jlpea8010005.

[18] Tushar Krishna. *A Detailed On-Chip Network Model Inside a Full-System Simulator*. 2017. URL: https://pdfs.semanticscholar.org/c1e9/0beac857ce1af1a531b6538804e71efdef05.pdf.

[19] John H. Lau. "Evolution, challenge, and outlook of TSV, 3D IC integration and 3d silicon integration". In: *2011 International Symposium on Advanced Packaging Materials (APM)*. 2011, pp. 462–488. DOI: 10.1109/ISAPM.2011.6105753.

[20] John H. Lau. "Recent Advances and Trends in Heterogeneous Integrations". In: *Journal of Microelectronics and Electronic Packaging* (2019). DOI: https://doi.org/10.4071/imaps.780287.

[21] Chengeng Li et al. "Accelerating Cache Coherence in Manycore Processor through Silicon Photonic Chiplet". In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 2022, pp. 1–9.

[22] Li Li et al. "Reliability Challenges in 2.5D and 3D IC Integration". In: *2017 IEEE 67th Electronic Components and Technology Conference (ECTC)*. 2017, pp. 1504–1509. DOI: 10.1109/ECTC.2017.208.

[23] Tao Li et al. "Chiplet Heterogeneous Integration Technology—Status and Challenges". In: *Electronics* (2020). DOI: https://doi.org/10.3390/electronics9040670.

[24] Gabriel H. Loh et al. "Interconnect-Memory Challenges for Multi-Chip, Silicon Interposer Systems". In: *Proceedings of the 2015 International Symposium on Memory Systems*. MEMSYS '15. Washington DC, DC, USA: Association for Computing Machinery, 2015, pp. 3–10. ISBN: 9781450336048. DOI: 10.1145/2818950.2818951. URL: https://doi.org/10.1145/2818950.2818951.

[25] X. Ma, Y. Wang, and Y. et al. Wang. "Survey on chiplets: interface, interconnect and integration methodology". In: (2022). DOI: https://doi.org/10.1007/s42514-022-00093-0.

[26] Rafael Medina et al. "System-Level Exploration of In-Package Wireless Communication for Multi-Chiplet Platforms". In: *ASPDAC* (2023).

[27] Samuel Naffziger et al. "2.2 AMD Chiplet Architecture for High-Performance Server and Desktop Products". In: *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*. 2020, pp. 44–45. DOI: 10.1109/ISSCC19947.2020.9063103.

[28] Samuel Naffziger et al. "Pioneering Chiplet Technology and Design for the AMD EPYC™ and Ryzen™ Processor Families". In: *ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)* (2021).

[29] Saptadeep Pal et al. "Design Space Exploration for Chiplet-Assembly-Based Processors". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28.4 (2020), pp. 1062–1073. DOI: 10.1109/TVLSI.2020.2968904.

[30] Daniel Sanchez, George Michelogiannakis, and Christos Kozyrakis. "An Analysis of On-Chip Interconnection Networks for Large-Scale Chip Multiprocessors". In: *ACM Trans. Archit. Code Optim.* 7.1 (May 2010). ISSN: 1544-3566. DOI: 10.1145/1736065.1736069. URL: https://doi.org/10.1145/1736065.1736069.

[31] V. Soteriou, Hangsheng Wang, and L. Peh. "A Statistical Traffic Model for On-Chip Interconnection Networks". In: *14th IEEE International Symposium on Modeling, Analysis, and Simulation*. 2006, pp. 104–116. DOI: 10.1109/MASCOTS.2006.9.

[32] A. W. Topol et al. "Three-dimensional integrated circuits". In: *IBM Journal of Research and Development* 50.4.5 (2006), pp. 491–506. DOI: 10.1147/rd.504.0491.

[33] Wen-Chung Tsai et al. "Networks on Chips: Structure and Design Methodologies". In: *JECE* 2012 (Jan. 2012). ISSN: 2090-0147. DOI: 10.1155/2012/509465. URL: https://doi.org/10.1155/2012/509465.

[34] Chifeng Wang, Wen-Hsiang Hu, and Nader Bagherzadeh. "A Wireless Network-on-Chip Design for Multicore Platforms". In: *2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing*. 2011, pp. 409–416. DOI: 10.1109/PDP.2011.37.

[35] Wikipedia contributors. *Token passing — Wikipedia, The Free Encyclopedia*. [Online; accessed 22-June-2023]. 2022. URL: https://en.wikipedia.org/w/index.php?title=Token_passing&oldid=1111727902.

[36] S.C. Woo et al. "The SPLASH-2 programs: characterization and methodological considerations". In: *Proceedings 22nd Annual International Symposium on Computer Architecture*. 1995, pp. 24–36. DOI: 10.1109/ISCA.1995.524546.

[37] Xiaowen Wu et al. "UNION: A Unified Inter/Intrachip Optical Network for Chip Multiprocessors". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22.5 (2014), pp. 1082–1095. DOI: 10.1109/TVLSI.2013.2263397.

[38] Xiaowu Zhang et al. "Heterogeneous 2.5D integration on through silicon interposer". In: 2015. DOI: 10.1109/DAC18072.2020.9218539.