Methods for fast exploration of manycore architectures based network-on-chip with emerging technologies

Ibrahim Krayem

▶ **To cite this version:**

Ibrahim Krayem. Methods for fast exploration of manycore architectures based network-on-chip with emerging technologies. Performance [cs.PF]. Université de Rennes, 2024. English. NNT : 2024URENS008 . tel-04689936

HAL Id: tel-04689936

https://theses.hal.science/tel-04689936v1

Submitted on 6 Sep 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

COLLEGE MATHS, TELECOMS
DOCTORAL INFORMATIQUE, SIGNAL
BRETAGNE SYSTEMES, ELECTRONIQUE

Université de Rennes

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES

ÉCOLE DOCTORALE N° 601
Mathématiques, Télécommunications, Informatique, Signal, Systèmes, Électronique
Spécialité : *Informatique*

Par

Ibrahim KRAYEM

Methods for fast exploration of manycore architectures based Network-on-Chip with Emerging Technologies

Thèse présentée et soutenue à Lannion, le 29/02/2024
Unité de recherche : IRISA - Équipe Taran

Rapporteurs avant soutenance :

GOGNIAT Guy — Professeur des Universités, Université Bretagne Sud
VERDIER François — Professeur des Universités, Université Côte d'Azur

Composition du Jury :

Président :	Gilles SASSATELLI	Directeur de recherche, LIRMM, CNRS/Université Montpellier
Examinatrice :	Lilia ZAOURAR	Ingénieure de Recherche-PhD, CEA List
Dir. de thèse :	Cédric KILLIAN	Professeur des Universités, Université Jean-Monnet
Co-dir. de thèse :	Daniel CHILLET	Professeur des Universités, Université de Rennes

Invité :

Kevin Martin — Maître de conférences HDR, Université de Bretagne-Sud

"Tout récipient se restreint par son contenu, sauf le récipient de la connaissance, car il s'élargit avec lui."

La Voie de l'Éloquence

Acknowledgement

Tout d'abord, je voudrais remercier mes superviseurs, Daniel Chillet et Cedric Killian, pour leurs conseils, leurs discussions perspicaces et leur précieux mentorat tout au long de mon parcours de doctorat. Vous m'avez montré comment faire de la bonne recherche. Vous m'avez encouragé dans les moments difficiles et je vous en suis très reconnaissant.

Je tiens à exprimer ma profonde gratitude à tous les membres du comité de soutenance pour avoir accepté d'évaluer mon travail de thèse. M. VERDIER François et M. GOGNIAT Guy pour avoir été membre de mon jury de thèse et avoir montré de l'intérêt pour le projet. Mme. Lilia ZAOURAR et M. Gilles SASSATELLI qui ont accepté d'être les examinateurs de mon projet de thèse et ont consacré leur temps à l'évaluation approfondie de mon travail. Votre volonté d'évaluer ce manuscrit ne valide pas seulement des années de travail acharné, mais constitue le dernier pont crucial entre l'étudiant et l'universitaire. Votre expertise et vos critiques perspicaces ont été un atout inestimable, façonnant non seulement ce manuscrit mais aussi mes aspirations futures en tant que chercheur.

En outre, je remercie sincèrement les membres de l'équipes TARAN et Granit : le chef de notre laboratoire Prof. Olivier Sentieys, tous les membres permanents de notre laboratoire qui n'ont jamais hésité à nous fournir l'aide dont nous avions besoin, et qui ont offert beaucoup de conseils et de discussions lors de nos réunions de groupe. Un grand merci à tous les doctorants et chercheurs postdoctoraux de notre équipe. Mais je n'oublie pas non plus de remercier le Dr. Joel Ortiz Sosa pour sa collaboration et l'aide apportée au début de mon projet.

La base sur laquelle ma vie est construite est sans aucun doute ma famille. Mon père Ali et ma mère Jamila, vos sacrifices, votre amour et votre foi inébranlable en moi ont été le vent sous mes ailes. Vous m'avez inculqué les valeurs de travail, d'intégrité et de compassion qui m'ont guidé à chaque étape de ce voyage. Votre foi en moi a souvent dépassé la mienne et je vous en serai éternellement reconnaissante. Je n'oublierais pas non plus mes frères Hussein, Abbass et Mohammed et mes sœurs Fatima et Marwa qui ont toujours cru en moi et m'ont soutenu à chaque étape de ma vie. Votre amour, votre patience et votre foi en mes capacités ont été une source constante de motivation dont je vous suis éternellement reconnaissante.

Je tiens à exprimer ma plus profonde gratitude envers la communauté libanaise, qui s'est révélée être bien plus qu'une communauté : une véritable deuxième famille. Votre soutien gravé dans mon cœur.

À ma fiancée Nour, les mots me manquent pour exprimer toute ma gratitude pour ton amour inconditionnel, ton soutien et ta patience. Ton soutien continu a été une source constante de force, rendant les défis moins ardus et les joies bien plus douces. Pour votre amour, votre soutien, votre patience et pour les innombrables petites choses que vous faites et qui font une grande différence, merci.

Enfin, et ce n'est certainement pas le moins important, je tiens à exprimer ma profonde gratitude envers le Dieu et l'Imam Al Mahdi tout-puissant pour toutes leurs grâces et leur soutien tout au long de ce projet. Je leur offre ce travail en signe de remerciement, et je suis reconnaissant pour tout ce qu'ils ont fait.

Contents

List of Figures

List of Tables

Résumé étendu

Réseaux sur Puce - Network-on-Chip

Le paysage technologique en constante évolution stimule la demande de systèmes embarqués multicœurs à grande échelle et hautement efficaces. Depuis son introduction au début des années 2000, le concept de Réseaux sur Puce (Network-on-Chip - NoC) a subi une transformation notable, s'établissant fermement dans l'architecture de ces systèmes. Actuellement, les NoCs sont incorporés dans des composants avancés tels que le CPU Intel Core i9 [Int], le Système sur puce Snapdragon® 8 Gen 2 de Qualcomm [Yua+23] et le FPGA Stratix d'Altera [Sam+23], témoignant de leur importance et de leur pertinence dans l'industrie des semi-conducteurs.

L'étude des NoCs implique une analyse minutieuse de leurs divers composants, notamment les algorithmes de routage et les critères d'évaluation pertinents. Face à l'évolution rapide de la technologie des semi-conducteurs et aux besoins accrus en traitement de données, les NoCs classiques rencontrent continuellement de nouveaux défis [PJD22]. La recherche et le développement incessants ont conduit à l'apparition de stratégies innovantes pour surmonter ces obstacles, assurant ainsi l'adaptabilité et l'efficacité des NoCs dans le contexte technologique actuel [MSP23].

Évaluation des performances

L'évolution vers des architectures multicœurs, caractérisée par un parallélisme massif sur une seule puce, a marqué un tournant ces dernières années. Ces architectures, intégrant des dizaines, voir des centaines de cœurs hétérogènes, favorisent des capacités de calcul parallèle considérables, pertinentes notamment pour le calcul à haute performance (High-Performance Computing HPC) [Viv21]. Cette augmentation du parallélisme entraîne une augmentation conséquente des échanges de données, mettant en lumière l'importance cruciale du système de communication pour la performance globale.

Les avancées dans l'intégration du silicium ouvrent la voie à des interconnexions innovantes telles que les réseau sans fil sur puce (WiNoCs) [Ort+19]. Les WiNoCs exploitent

la large bande de fréquences du processus CMOS, permettant des communications rapides sur de longues distances sans augmentation significative de la latence, contrairement aux NoCs électriques qui sont limités par les traversées de routeurs. Bien que les WiNoCs puissent fonctionner seuls, la combinaison de WiNoCs et de NoCs électriques dans un système hybride offre une flexibilité, en utilisant des communications sans fil pour les distances longues et des routeurs électriques pour les distances courtes, pour surmonter les limitations de bande passante des WiNoCs.

L'exploration des architectures de systèmes multicœurs représente un défi en raison de la complexité et du temps requis pour l'évaluation des performances, laquelle repose souvent sur des simulations lentes, particulièrement dans le cas de système à large échelle. En alternative, la modélisation mathématique se présente comme un compromis intéressant entre rapidité de calcul et précision. Les modèles analytiques appliqués aux interconnexions sur puce, basés sur la théorie des files d'attente, offrent des indicateurs de performance comme l'utilisation moyenne de la mémoire tampon et la latence des paquets. Toutefois, un défi majeur réside dans le fait que les modèles actuels sont principalement conçus pour des interconnexions homogènes et ne s'adaptent pas bien aux systèmes d'interconnexion hybrides.

Traces de trafic d'application

L'évaluation des performances d'un NoC lors des premières phases de conception architecturale, souvent désignée comme l'exploration de l'espace de conception, est cruciale pour les concepteurs de système sur puce. La simulation de systèmes complets, englobant l'architecture dans son ensemble – coeurs de calcul, mémoires, interconnexions, et l'exécution des applications –, représente la méthode la plus fiable pour évaluer les performances des interconnexions et leur influence sur l'exécution des applications. De plus, cette approche est essentielle pour mesurer précisément la vitesse d'exécution d'une application.

Malgré son importance, la simulation de systèmes complets pour l'évaluation des performances des NoCs s'avère extrêmement chronophage. La complexité de personnaliser des simulateurs tels que le fameux Gem5 représente un premier obstacle majeur pour les concepteurs souhaitant tester des interconnexions sur puce émergentes. De plus, la durée nécessaire à ces simulations limite drastiquement l'exploration efficace de l'espace de conception. Un exemple concret de cette limitation est la simulation de l'exécution d'une

architecture à 64 cœurs traitant une vidéo de 13 secondes avec l'application x264 dans le cadre de la suite PARSEC. Avec un processeur Dual Intel Xeon 4214 et 64 Go de mémoire, une telle simulation requiert environ cinq jours, ce qui restreint considérablement la possibilité d'évaluer divers paramètres en un temps raisonnable.

Les modèles analytiques et les simulateurs NoC représentent des solutions pour l'évaluation des performances des NoCs. Cependant, ils présentent des limites, notamment l'incapacité à évaluer l'accélération des applications, car ils se concentrent sur les traces de communication sans exécuter les applications simultanément avec le NoC. Malgré cela, leur utilisation reste privilégiée dans la conception des NoCs. Historiquement, les concepteurs ont recours à des modèles de trafic génériques, comme les modèles aléatoires ou transposés, pour évaluer le comportement des NoCs. Récemment, l'adoption de modèles de trafic plus réalistes, reflétant le comportement des applications réelles, a gagné en intérêt. Cette approche implique d'abord l'exécution d'une application sur un simulateur de système complet pour enregistrer les traces de communication, qui sont ensuite injectées dans un outil d'analyse de performance NoC. L'injection de ces traces peut se faire de deux manières : i) l'injection paquet par paquet à des moments précis, ou ii) l'utilisation d'un trafic basé sur le taux d'injection de paquets (PIR). La première méthode peut être très chronophage en raison du volume de données à traiter, tandis que la seconde, bien que plus rapide, peut affecter la fidélité des résultats en raison des variations dans le calcul du PIR.

L'utilisation des traces de trafic d'application pour l'analyse de performance des NoCs est un sujet largement abordé dans la littérature. Ces traces sont couramment employées soit pour valider les propositions de recherche, comme dans notre étude, soit pour comparer et caractériser des applications de référence. Des approches variées sont adoptées pour cette analyse. Par exemple, certaines études, comme celles référencées dans [Adusumilli 2023, Chen 2020], utilisent un taux d'injection de paquets (PIR) constant pour chaque cœur durant toute l'application, ce qui peut atténuer les variations et manquer de précision dans l'évaluation des performances de l'interconnexion. D'autre part, d'autres chercheurs [Mandal 2019, Mandal 2021a] ont divisé les traces en fenêtres de temps de taille fixe pour mieux appréhender l'évolution de la latence en fonction des cycles d'horloge. Cependant, la sélection arbitraire de la taille de ces fenêtres soulève des questions quant à l'impact de ce choix sur la précision de l'évaluation des performances.

Les travaux que nous avons réalisés et qui sont présentés dans ce document abordent ces différentes problématiques. Nous proposons d'étendre les modèles analytiques pour

prendre en compte l'hétérogénéité d'un NoC. De plus, nous proposons un modèle exhaustif pour l'analyse des traces d'application et l'automatisation de leur utilisation dans l'évaluation des performances des NoCs, que ce soit via des simulations ou des modèles analytiques.

Le reste du manuscrit est organisé de la façon suivante

Chapitre 1

Ce chapitre dédié aux NoCs traite de leur évolution historique et des principes fondamentaux qui les régissent. Nous entamerons par une analyse approfondie des topologies, des composants et des formats de transfert de données des NoCs, suivie d'un examen des algorithmes de routage et des techniques de contrôle de flux. Par la suite, nous définirons et discuterons les critères d'évaluation qui déterminent les performances des NoCs. Il est également crucial de reconnaître les limites des NoCs électriques traditionnels, tout en examinant les technologies émergentes, en particulier les réseau sans fil sur puce. Finalement, le chapitre conclura avec une explication détaillée des différents modèles de trafic utilisés dans les NoCs, en se concentrant sur les applications synthétiques et les benchmarks.

Chapitre 2

Dans ce chapitre, nous proposons d'étendre les modèles analytiques pour prendre en compte l'hétérogénéité d'un NoC. Nous introduisons un modèle analytique innovant pour les NoCs hybrides, basé sur le modèle de file d'attente $M/G/1$. Ce modèle est conçu pour évaluer la latence moyenne des paquets et le débit du réseau dans unNoC hybride, où les paquets peuvent emprunter des chemins électriques, sans fil, ou mixtes. Le modèle repose sur une distribution de Poisson pour le trafic de communication, avec des routeurs utilisant l'arbitrage round-robin et un accès au canal sans fil via un système de passage de jetons. Il permet une configuration distincte des temps de service et des capacités des canaux électriques et sans fil, reflétant les conceptions matérielles réelles. Ce modèle offre une hétérogénéité qui facilite l'exploration des paradigmes de conception futures, tels que les architectures basées sur les chiplets ou les accélérateurs. Il aborde des caractéristiques clés non couvertes par les modèles conventionnels, telles que i) un mécanisme d'accès au canal sans fil distinct, ii) des largeurs de bande de communication hétérogènes entre les

médias électriques et sans fil, et iii) des différences dans les temps de latence entre les routeurs électriques et sans fil. À notre connaissance, ce modèle est le premier de son genre pour une interconnexion hybride sur puce.

Chapitre 3

Ce chapitre se consacre à l'élaboration d'un modèle exhaustif pour l'analyse des traces d'application et l'automatisation de leur usage dans l'évaluation des performances des NoCs, que ce soit via des simulations ou des modèles analytiques. L'intégration des fenêtres d'analyse est mise en avant, soulignant leur rôle crucial dans la capture de la dynamique temporelle de la latence. Le cadre développé ne se limite pas à recommander l'utilisation de fenêtres d'analyse ; il fournit également des indications sur la taille optimale de ces fenêtres et expose les bénéfices de leur utilisation. En offrant une perspective détaillée sur l'évaluation des applications réelles et en explorant l'impact des fenêtres d'analyse sur les résultats, ce modèle permet d'améliorer la robustesse des études de recherche et de réaliser des comparaisons plus précises entre différentes architectures. Ainsi, ce chapitre enrichit la compréhension de l'évaluation des performances des NoCs et promeut une approche plus approfondie et complète de l'analyse des applications de référence dans le domaine des architectures informatiques.

Introduction

Context and Motivations

In the early days of the electronics industry, devices were mainly based on electron tubes. These tubes control the flow of electrons between two metal electrodes in a vacuum or gas-sealed container [Oka94]. In 1925, Lilienfeld [Edg30] introduced the concept of current control by a perpendicular electric field, later known as Metal-Semiconductor Field-Effect Transistor (MESFET). Despite the difficulties encountered in the initial manufacture of the MESFET, progress was made with Schottky's theory of rectification behavior in metal-semiconductor contacts and the creation of the first bipolar junction transistor in 1947 by Bardeen, Brattain and Shockley [Ort09]. More progress was then made in the 1960s with the development of the Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) and the establishment of silicon as the reference semiconductor, thus culminating in the successful manufacture of the MESFET [Mea66], marking the transition to the era of nanoscale electronics.

Moore's Law, a key concept in semiconductor technology, has its origins in an observation made in 1965 by Gordon Moore, co-founder of Intel. Figure 1 illustrates the 122 years of Moore's law, which postulates that the number of transistors on a microelectronic chip will double approximately every two years, increasing computing power. The graph shows the evolution of different computing technologies, from mechanical systems to integrated circuits. It starts with mechanical devices used in the early 20th century, followed by relays, vacuum tubes, transistors, and finally integrated circuits dominating today. An exponential increase in the computing power can be observed over time, particularly from the 1960s onward with the advent of transistors. This trend highlights the rapid and impressive progress of semiconductor technology, leading to major advances in computing and digital technologies.

Figure 2 additionally illustrates the evolution of the node sizes of semiconductor technology over several decades, from 3µm to $3nm$. A constant and significant reduction in node size is recognized, reflecting the technological advances in miniaturization. The graph further shows that, as early as the 1970s, the nodes were relatively large, measur-

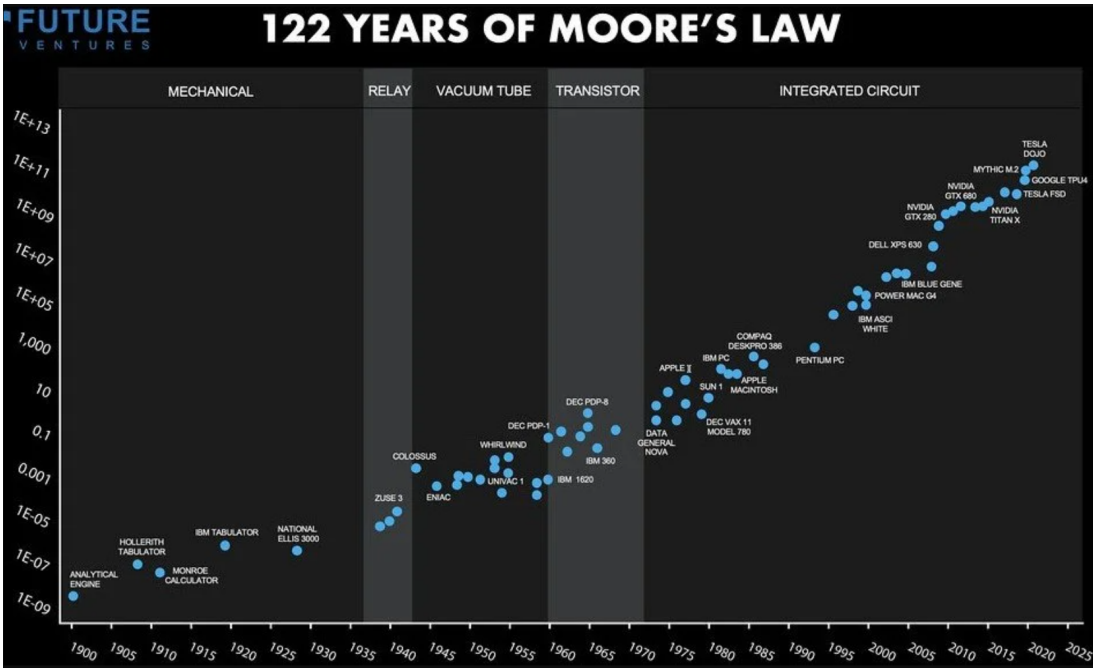


Figure 1: 122 years of Moore's law [AI].

ing micrometers. However, as technology progressed, they gradually decreased, reaching sub-micrometer sizes such as $90nm$, $65nm$, and down to recent nanometer sizes such as $7nm$ and $5nm$. The images inserted in the graph show the density of transistors or structures at each stage, highlighting the massive increase in density as technology advances. This continuous reduction in node size is a testament to the ingenuity of microelectronics engineering, enabling increased performance and greater chip energy efficiency.

For instance, in the beginning of 2023, apple announced [App] the M2, M2 pro and M2 Max (Figure 3) next generation System-on-Chip (SoC): the M2 Max builds upon the capabilities of the M2 Pro, featuring up to a 12-core CPU and a remarkable 19-core GPU. In addition, the M2 Max boasts a GPU with up to 38 cores. This cutting-edge chip is manufactured using second-generation 5 nm process technology. When it comes to the transistor count, the M2 Pro is packed with 40 billion transistors, while the M2 Max contains an astounding 67 billion.

Today, other giants such as Huawei are not to be outdone, regularly announcing significant technological advances in this field. The Kirin 9000S processor is used in the latest Mate 60 Pro phone [HUA]. The Kirin 9000 is an advanced 5nm chipset, featuring a CPU with 1x Cortex-A77 at 3.13 GHz, 3x Cortex-A77 at 2.54 GHz, and 4x Cortex-A55

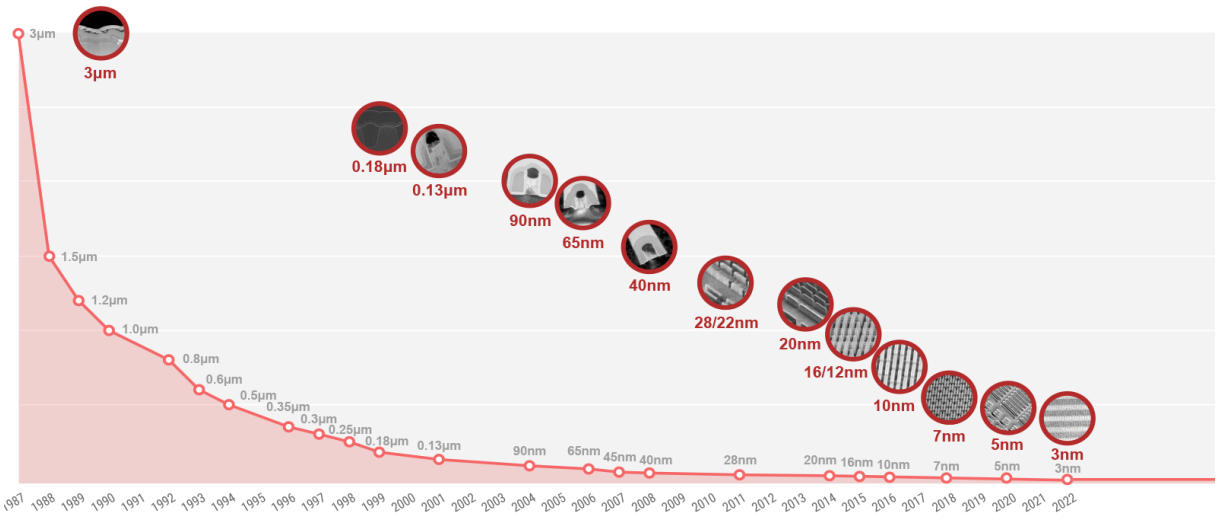


Figure 2: The evolution of the node sizes of semiconductor technology over several decades [TSM].

at 2.05 GHz. It features a 24-core Mali-G78 GPU with Kirin Gaming+ 3.0, Huawei Da Vinci 2.0 AI architecture with 2x Ascend Lite + 1x Ascend Tiny, supports 5G SA&NSA, Sub-6G&mmWave, has a four-channel Kirin 6.0 ISP, 8 MB system cache and supports LPDDR5/4X memory.

These innovations are not simply technological triumphs. They also embody a global struggle for supremacy in the semiconductor sector. It is not just a competition between companies but also between nations. Chips and Integrated Circuit (IC) are at the heart of most modern technologies, from smartphones and autonomous cars [GGM23] to communications infrastructures and supercomputers. Whoever leads in this field holds considerable influence, both economically and strategically.

The frenetic pace of this "chip war" illustrates just how crucial these innovations are to the future of technology and, by extension, to our way of life.

The work presented in this manuscript takes place in the technological domain, particularly concentrating on interconnects, which are crucial components in the architecture of modern computing systems. Our focus is on interconnection and chip performance. Initially, the most common approach to interconnecting the Intellectual Property (IP) such as Central Processing Unit (CPU) cores, memory, and Input/Output (I/O) devices was a shared bus [Ack+00]. This means that all IP had to communicate with each other over the same pathway (bus). Figure 4 illustrates an example of the general architecture of a

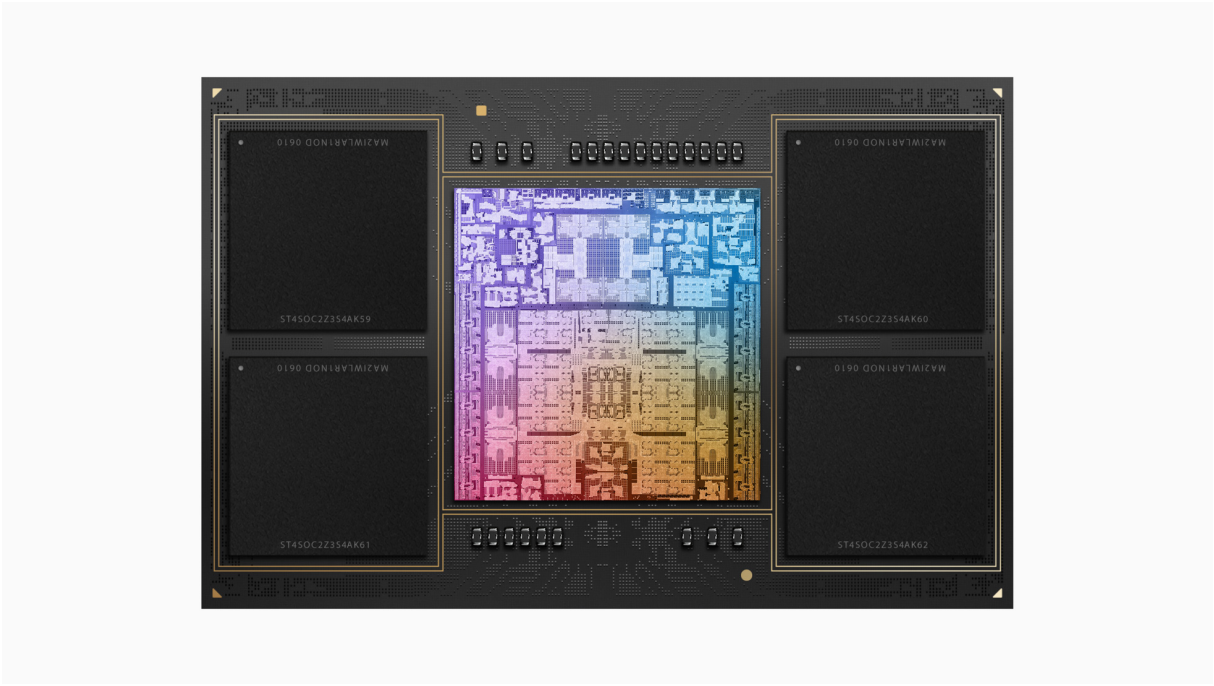


Figure 3: M2 Pro features 40 billion transistors, 200GB/s of unified memory bandwidth, and up to 32GB of fast, low-latency unified memory.

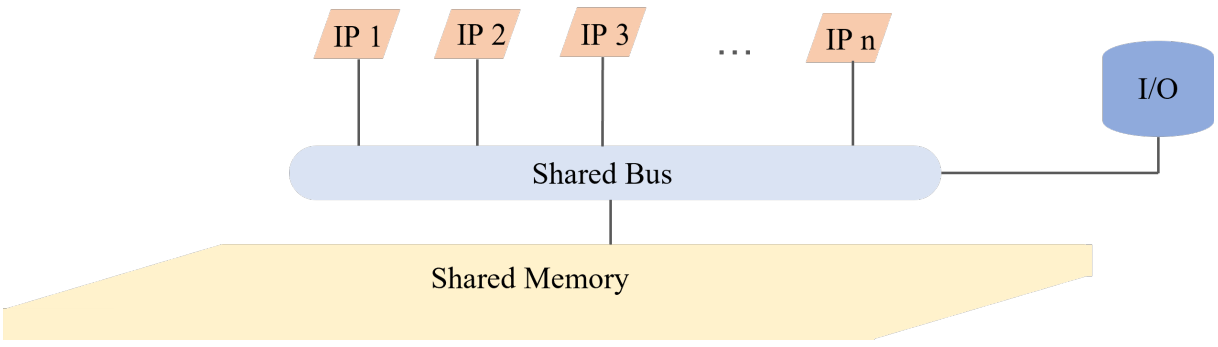


Figure 4: General Architecture for an MPSoC using a shared-memory bus-based system.

Multiprocessor System-on-Chip (MPSoC) using a shared memory bus system. Although this approach was simple, it quickly became a bottleneck as the number of cores increased. The shared bus had limited bandwidth, and when multiple cores tried to communicate simultaneously, they had to wait their turn, reducing the overall system performance.

To get around this problem, the importance of on-chip networks came into picture. Initially electrically based, then followed by the emerging of new technologies such as photonics and wireless communication as well as the hybrid communication that combines both electrical with either wireless or photonic communications. These emerging technologies were first considered because of limitations in terms of latency and energy consumption. This thus implies the major issue of performance analysis, which forms a crucial step in the early stages of system-on-chip design.

This evaluation, which involves several metrics such as latency, throughput and energy consumption, can be carried out using different approaches:

- Full system simulator; such as sniper and Gem5, that cover interconnections, cores and memories.
- NoC simulator: like noxim and booksim, are cycle-accurate interconnection network simulator.
- Analytical model: based on queuing theory with which several metrics can be considered such as latency, throughput and saturation.

It is worth noting that simulators are very slow, which is considered as a limitation for designers. On the other hand, analytical models are much faster while still maintaining good precision. Moreover, in terms of complexity, simulators are much more complicated than analytical models.

Contributions and Organization of the Thesis

This doctoral research makes a significant contribution to the understanding and improvement of NoCs through two major advances, each detailed in its own chapter.

The first contribution, outlined in Chapter 2, "AMHNOC: Analytical Model for Hybrid NoC", presents a novel analytical model for evaluating the performance of hybrid NoCs. We have introduced new evaluation metrics and tools, including simulators and analytical models, to finely analyze hybrid interconnects. In-depth parameter exploration with synthetic traffic was carried out to understand the impacts of traffic patterns, architecture size, packet size, wireless data rate, and number of antennas on network throughput and execution time. This exhaustive analysis has enabled us to identify optimal configurations for different types of reference applications, minimizing relative error and maximizing overall performance.

The second contribution, detailed in Chapter 3, "Application trace windowing for fast NoC performance analysis", proposes a methodological framework for accelerating NoC performance analysis by exploiting application trace windowing. We have developed a window splitting and merging technique to improve the accuracy of performance evaluation while reducing computational complexity. Experimental results validated the effectiveness of this method with random traffic and a combination of precise and Poisson traffic, confirming the significant reduction in errors and improvement in execution time.

Together, these contributions broaden the horizon of possibilities in NoC design and optimization, providing system designers with analytical and practical tools to meet the challenges of today's and tomorrow's high-performance computing systems.

Chapter 1

State of the art

1.1 Introduction

As the technological landscape evolves, the demand for efficient, multicore embedded systems on a large scale becomes increasingly critical.

As the technological landscape continues to develop, the need for efficient, multicore embedded systems on a large scale systems becomes increasingly important. Initially introduced in the early 2000s, the concept of NoC has undergone a remarkable evolution and has become strongly based on the architecture of these systems. Today, it is integrated into processors and SoC designs such as the Intel Core i9 CPU [Int], Qualcomm's Snapdragon® 8 Gen 2 SoC [Yua+23], and Altera's Stratix FPGA [Sam+23].

An in-depth study of NoCs requires an extensive exploration of its multiple components, including the routing algorithms used and the relevant evaluation metrics. In parallel with the constant evolution of semiconductor technologies and the increasing demands of data processing, traditional NoCs are regularly subjected to new challenges [PJD22]. Ongoing research and development has enabled the emergence of novel approaches to address these issues, ensuring the adaptability and efficiency of NoCs [MSP23].

In this chapter, the historical evolution of NoCs, highlighting their essential principles will be illustrated (Section 1.2). Starting first with a detailed analysis of topologies, components, and data transfer formats followed by a study of routing algorithms and flow control methods. The evaluation criteria defining the performance of NoCs will be then detailed. Furthermore,despite that traditional electrical NoCs offer many advantages, it is essential to highlight their limitations.

Additionally, the present emerging technologies, in particular Wireless Network-on-Chip (WiNoC) will be presented in Section 1.3. After in Section 1.4, a detailed explanation of the traffic patterns used considering its two types i.e., synthetic and benchmark applications, will be provided.

1.2 Historical Evolution and Background of Network on Chip Architectures

Communications saturation in bus-based architectures results from the integration of numerous logic blocks, known as IP, communicating with each other on a single chip as transistors become miniaturized. This led to the recognition of NoC as a remedy for the limitations of MPSoC [BD02]. To overcome the limitations posed by shared bus-based architectures, NoCs offer a scalable and robust solution for interconnecting numerous cores and other components within a single chip, making it an essential feature of the modern SoC landscape.

1.2.1 Evolution of microprocessors

The evolution of microprocessors has been characterized by a continuous increase in computing power, miniaturization of components and improved energy efficiency. Over the last 50 years, the evolution of microprocessors has significantly changed the landscape of digital technology. The constant focus on miniaturization and increased functionality has led to the development of increasingly complex systems. Figure 1.1 [Rup22] shows several key trends in microprocessor development over the past five decades. Since the 1970s, there has been an exponential growth in the number of transistors, in accordance to Moore's law, which predicts a doubling in the number of transistors every two years. This is due to the reduction of the transistor size to just a few nanometers [Tau23]. At the same time, the performance of single thread increased sharply from the 1970s and up until the 2000s before stabilizing. This indicates that the performance gains per thread were reaching their limits. Additionally, the rapid increase in the microprocessor's frequency, until the early 2000s, before slowing down, is probably due to the thermal and power consumption constraints. We also note that microprocessor power consumption rose until the early 2000s before stagnating, reflecting a growing interest in energy efficiency in chip design. Following this stagnation, the industry moved towards a significant increase in the number of logic cores, marking a shift towards multi-core architectures to counter the stabilization of single-threaded performance. In this context, NoC have been crucial to guarantee optimal communication between the various components and cores, reflecting the need to adapt to the growing complexity of SoC.

The progression of SoC and NoC reflects the constant pursuit for higher performance

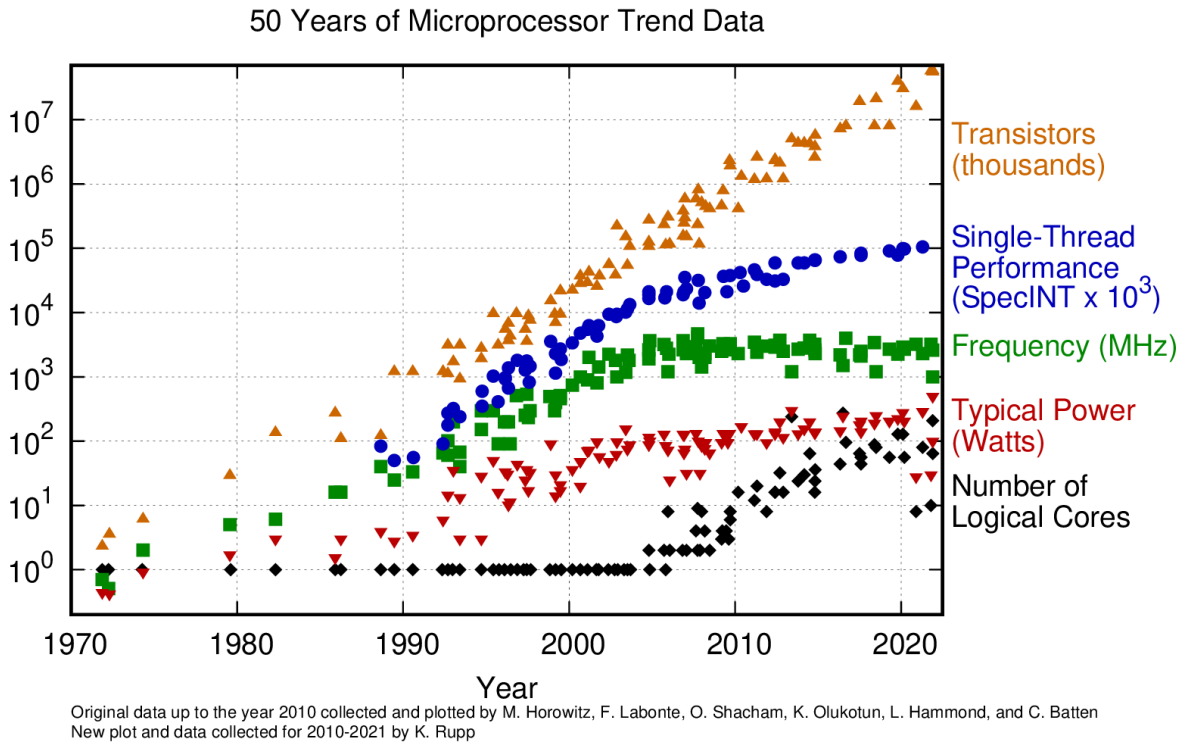


Figure 1.1: 50 years of Microprocessor Trend Data [Rup22].

in the microprocessor field. While early generations of microprocessors concentrated on increasing the number of transistors and clock frequency, physical constrains and heat dissipation problems led to the adoption of the multi-core approach. NoCs emerged as an elegant solution to the growing complexity of intra-chip communication, offering superior scalability and energy efficiency compared to traditional bus and point-to-point architectures. This transformation has been accompanied by innovations like Surface Wave Interconnect (SWI), which hold the potential to further revolutionize interconnects by using surface waves for ultra-fast, low-power wireless communication, thus enabling even denser, more integrated systems. These technologies are reshaping the SoC design paradigms, opening the door for a new era of compact, potent electronic devices capable of satisfying the ever-growing processing and connectivity demands of the digital age.

1.2.2 Topologies

1.2.2.1 2D Topologies

The study of network topologies within NoCs is an essential aspect, as it affects communication efficiency and the optimization of the overall SoC performance. Careful topology selection has a direct influence on the scalability, latency, and power consumption, which is necessary to meet the needs of today's sophisticated applications. Generally, the topology of NoC determines how to manage the physical layout and the network connections. Two categories of topologies have been defined, direct and indirect topologies [CML12]. In indirect topology, not all routers are associated with an IP, in which some of them are only responsible for forwarding packets within the network. By specializing routers in an indirect topology, more advanced NoCs can be constructed, allowing for precise control over traffic management to reduce bottlenecks and enhance overall system performance. In contrast, in direct topology (such as mesh, ring, binary tree, etc.), every router is connected to an IP. Such a pair (IP, router) is called a node. Figure 1.2 illustrates some examples of direct topologies:

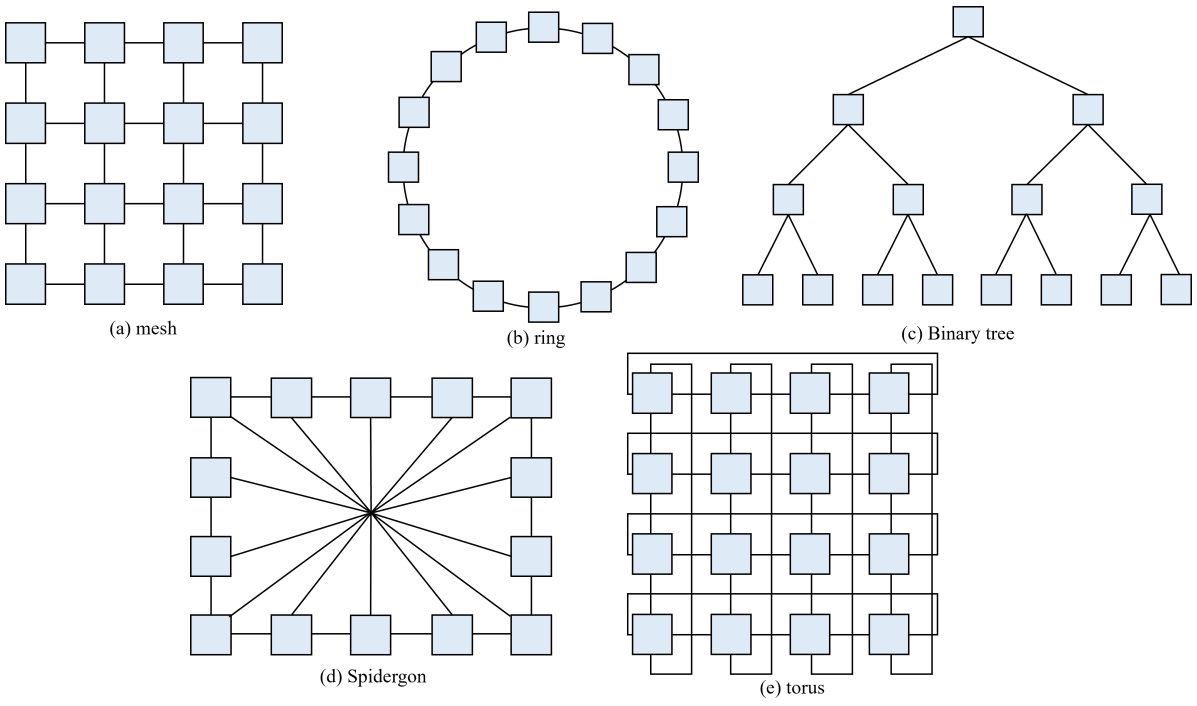


Figure 1.2: Different NoC direct topologies.

- **Mesh** [Kum+02]: The mesh topology is defined by the dimensions n and m ($n \times$

m), which determine the number of nodes in each row and column, respectively. Figure 1.2-a shows an example of a 4×4 mesh with $n = 4$ and $m = 4$ and 16 cores in total. Furthermore, all IP are connected to a router, which in turn is connected to 2, 3, or 4 adjacent routers. Each router is equipped with a maximum of five ports (for Electrical Network-on-Chips (ENoCs)), one connected to an IP and the rest linked to up to four neighboring routers if they exist. The communication pathways used to establish a connection between two neighboring routers, or between a single router and an IP, comprise two unidirectional links that run in opposite directions. The mesh topology offers a simple, regular and scalable structure with redundant communication paths. However, it suffers from potentially higher communication latency between remote nodes and increased routing complexity with network size.

- **Ring** [CGL11]: As represented in Figure 1.2-b, all nodes are connected in a ring with each node maintaining connections with two neighboring nodes, regardless of the size of the ring. The greatest distance between any pair of adjacent nodes increases proportionally to the number of nodes. Although ring topology offers predictable latency with a simple structure, it remains sensitive to failures with a limited ability to handle heavy traffic loads.

- **Binary tree** [DYN03]: It comprises a specific arrangement of routers or nodes that are connected to one another in a structure similar to that of a tree, with the exception of the root node. Each node is connected to one parent and zero, one, or two child nodes. We refer to a tree as balanced when each leaf node is equally spaced from the root, indicating that all tree branches are of equal lengths. As shown in Figure 1.2-c, each parent node is connected to two child nodes, with a total of 15 nodes in the architecture. The binary tree topology provides a structured hierarchy and modular expansion. Nevertheless, it suffers from an increased potential latency for inter-tree communications and susceptibility to failures in high-level nodes.

- **Spidergon** [Cop+04]: In a Spidergon, the nodes are connected in a ring with each node being connected to the node on the opposite side of the ring, as illustrated in Figure 1.2-d. Spidergon topology affords a better latency and power consumption than other topologies for medium-sized networks, however, the increasing complexity and potentially higher interconnection costs for very large networks are still considered as disadvantages for such topology.

- **Torus** [Pan+05]: As presented in Figure 1.2-e, a $n \times m$ torus configuration is derived from a mesh topology $n \times m$ by incorporating a wraparound channel on every row and column. The inclusion of these wraparound channels helps to decrease both the diameter and the average distance across the router. As a result, each router is equipped with 5 ports, linked to 5 routers. The advantage of such topology includes their flexibility and redundancy with multiple paths between nodes, facilitating fault tolerance and better performance. On the other hand, their drawbacks include the higher interconnection complexity and potentially increased routing cost.

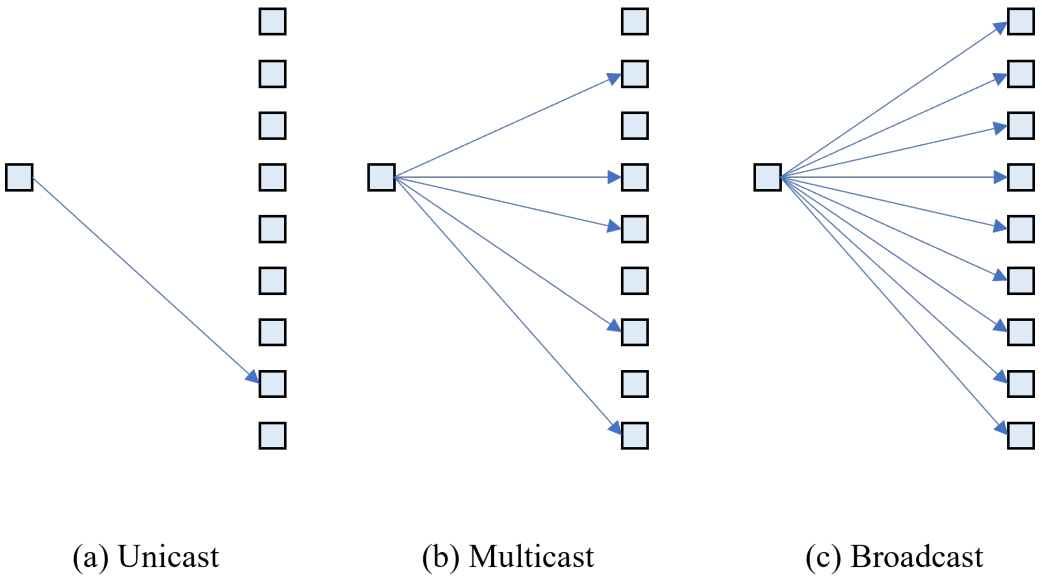


Figure 1.3: Unicast, multicast and broadcast communication.

One of the most commonly used topologies for NoCs is the mesh topology that is adopted in this thesis (Figure 1.2-a). Whatever the topology used, it is necessary to improve the performance of NoC by proposing interconnection architectures that facilitate collective communications. For example, **unicast** communication enables the packet transmission from a one source to one destination. Additionally, the **multicast** communication, enables transmission from one source to M destinations and is ideal for applications such as cache synchronization or instruction distribution. **Broadcast** communication, on the other hand, enables transmission from one source to all destinations, and is crucial for operations such as service announcements or global configuration updates.

1.2.2.2 3D Topology

Having explored two-dimensional NoC topologies, it is important to note that another dimension can be added, for even greater efficiency, resulting in the revolutionary emergence of what are known as 3D NoCs [PF07]. As shown in Figure 1.4, the 3D topology enables connections not only horizontally and vertically (intra-layer links) on a single plane but also across different layers of a stacked chip (inter-layer links). By stacking the integrated circuits within multiple layers, 3D NoCs exploit vertically to increase the connection density and decrease the distance that should be traversed by the signal, thus, reducing the latency and potentially increasing the bandwidth, thereby, the overall system performance is improved [AAK10; BG20]. Therefore, 3D NoCs mark a substantial advancement in the field of electronic systems design. Additionally, this 3D architecture also offers a better thermal management and reduced power consumption, paving the way for more compact designs, which is particularly beneficial for mobile devices for instance as well as for applications that require a high computing power in a small space.

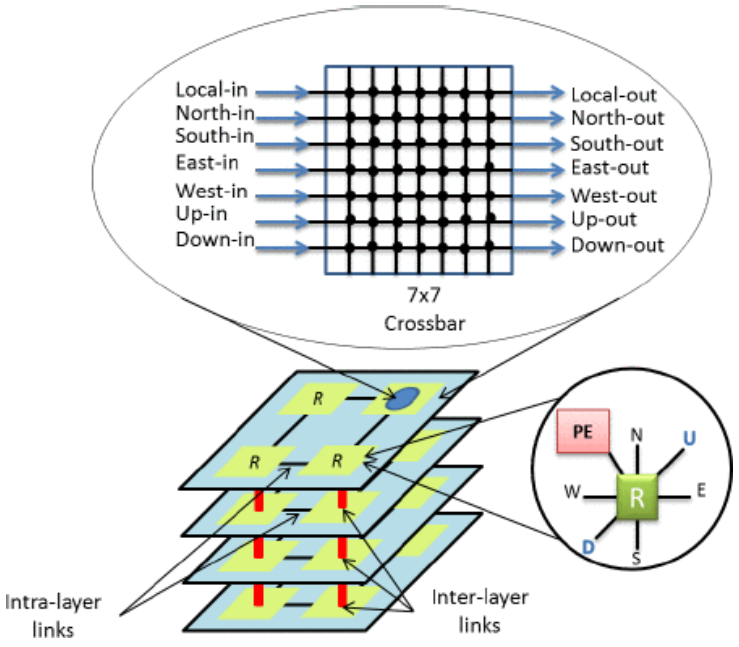


Figure 1.4: 3D OASIS-NoC 2×2×4 mesh topology [AAK10].

1.2.3 Data transfer format

In NoCs, a message, which is a complete unit of information, is transmitted from one point to another. It includes complex data types such as files, images or texts and often divided into several smaller packets to facilitate transmission [ZWG13]. Each packet contains a part of the message, as well as some additional information, such as the source and destination addresses and error-checking information. A packet is further divided into smaller units called flits (Flow Control Digit). The flit is the basic transfer unit in a NoC and corresponds to the width of the interconnection between the routers. A packet typically consists of a head flit, which often contains routing information, as well as information about the number of flits included in the packet, and a body flit carrying the actual packet data. Including a tail flit adds an additional flit to the packet to terminate it. It should be noted that the number of flits can vary within a packet, including in rare cases flits of uniform size.

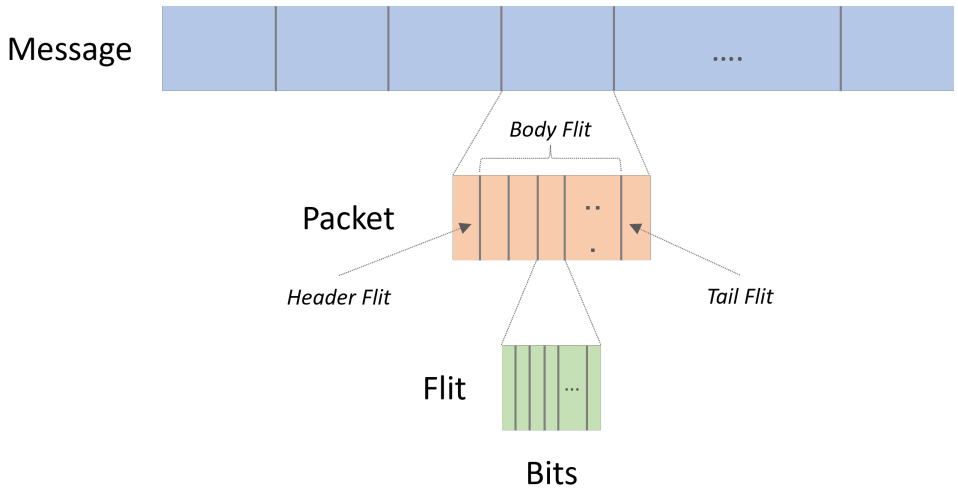


Figure 1.5: Data transmission hierarchical: Bits, Flits, Packets and Messages.

This process of dividing messages into packets and flits enables more efficient use of the network. It also allows better control of traffic, as packets can be routed individually, and transmission can be controlled at flit level [Lee+13].

1.2.4 Components

After discussing how messages are structured and organized efficiently, it is important to identify the individual components of a NoC to better understand how these messages are processed and routed. The components of a NoC are the fundamental elements that define its efficiency and reliability. In order to ensure optimal communication between the cores of a SoC and that the network can effectively handle the high processing demands of contemporary IT systems, careful design of these components is imperative.

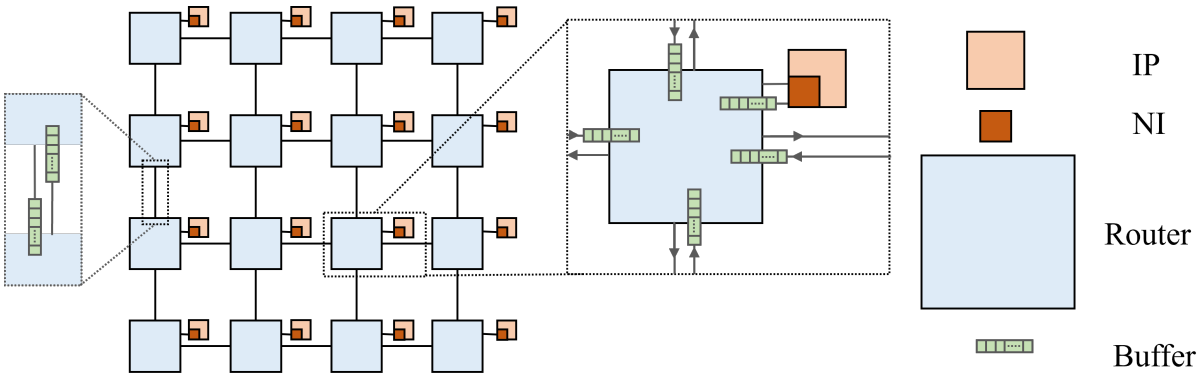


Figure 1.6: An example of a NoC-based many-core on-chip system.

Figure 1.6 shows a general example of a NoC-based many-core on-chip system. Generally, routers, links and Network Interface (NI) are the three essential components that form NoCs [DYN03], especially in a basic electrical NoC. The **Router** is an important component of a NoC that is responsible of routing data packets. Typically, it comprises input and output ports, **buffers** for holding incoming packets, and a switching unit. The role of the switching unit is to transmit packets according to the implemented routing algorithms (discussed in Section 1.2.5).

Initially, there is only one channel per port with virtual channels which are defined to share physical router resources and increase throughput, allowing logical separation of data streams to reduce congestion and improve overall performance [BV15]. The **NI** on the other hand, establish logical connections between the network and the IP that is interconnected with the router through electrical links. The actual set of routing paths is decided by the routing algorithm and will be detailed in the following section.

1.2.5 Routing algorithm

The path a packet takes to reach its destination is determined by the routing algorithm which indicates the next channel to be used at each intermediate node. The channel though can be chosen from several options. Nonetheless, the packet will be blocked and will not progress if all candidate channels are full.

Moreover, the routing algorithm can also impact the connectivity, which indicates the ability to route packets from any source node to any destination node. Similarly, adaptivity, which is the ability to select a different path for routing packets when there is contention or faulty components, is also a key feature determined by the routing algorithm used [Mer+22].

Furthermore, the routing algorithm has a vital role in ensuring deadlock and livelock freedom, i.e., guaranteeing that packets will not end up in an indefinite blockage or constantly drift within the network without reaching their destination.

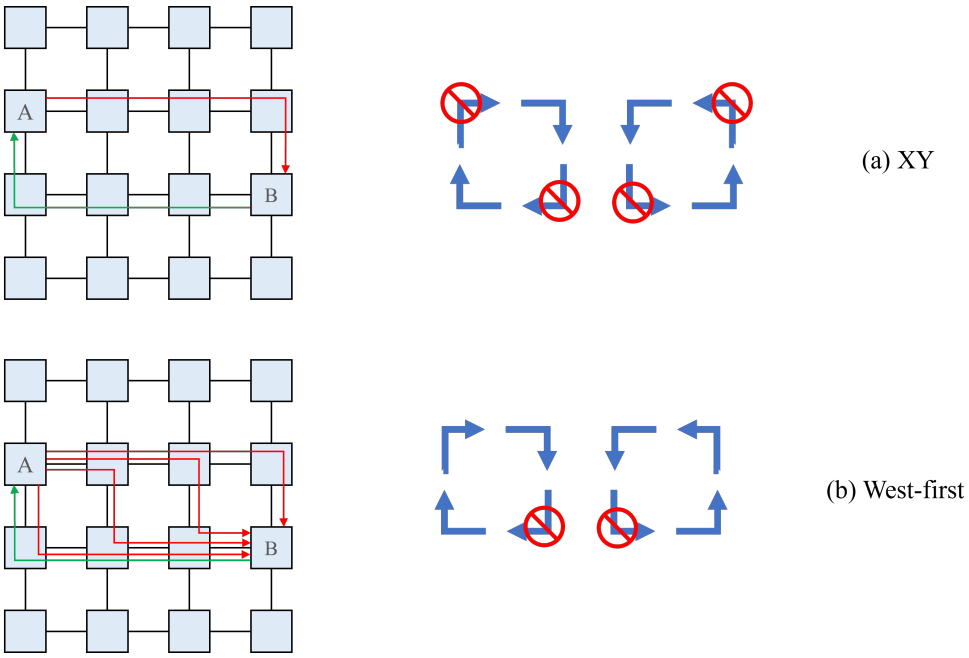


Figure 1.7: Example of routing algorithms: (a) XY and (b) West-first routing algorithm.

Depending on their level of adaptability, routing algorithms can be classified into different categories. For example, deterministic routing, such as XY routing [CGP12], as shown in Figure 1.7 (a), always follows a predefined path and is totally deterministic when the number of turns is restricted to four, but for one communication, just one turn, offering a single possible path for a packet from source A to destination B and vice versa.

Other methods, such as "West first" routing (Figure 1.7 (b)), allows all possible shortest paths when a packet travels east (from source A to destination B), but only allows one path when it travels west (from source B to destination A), requiring the west direction to be taken first if necessary. Other variants of this approach include the "North first", "South first", and "East first" methods [TSJ].

For adaptive routing methods, designed to avoid congested network paths, additional knowledge of the network is required. These routing algorithms naturally require more work to develop and therefore cost more in terms of space, cost, and power. Consequently, it is necessary to consider the appropriate Quality of Service Quality-of-Service (QoS) parameter before resorting to these techniques [AS09].

1.2.6 Flow control methods and arbitration

The routing of packets is allowed and stopped by the flow control mechanism. When a packet is blocked, buffer space is needed to store it. Additionally, packet transmission is halted by the flow control mechanism when there is no more buffer space available. It resumes when the packet is routed and there is space in the buffer. Furthermore, the packet may be dropped or rerouted through another channel if there is no flow control and no additional buffer space available.

Furthermore, flow control regulates the sequence of data from its source to its destination in the path. Its primary function is resource (like channels, bandwidth, and buffers) allocation and conflicts resolution for these resources. By minimizing waiting times for shared resources, effective flow control increases network packet speed and decreases problems such as deadlocks. Wormhole flow control is the most widely used, as it provides better buffer utilization and lower latency. In terms of virtual channels, using the same physical channel allows the use of idle bandwidth (the available transmission capacity on the network that is not actively carrying packets or data transfers) [MMH21].

In the literature, different flow control methods have been proposed and are summarized in Table 1.1 along with their advantages. These methods include Clumsy FC, Prediction Based FC, Distributed Buffer FC, Improved FC, Injection Level FC, Pile Level FC, QHT FC, and Fault Tolerant FC. Several criteria are examined, such as area reduction, cost reduction, design complexity, increased packet injection rate, latency reduction, throughput increase, power consumption reduction, congestion control, fault tolerance and buffer utilization. For each criterion, some techniques show benefits, indicated by a "✓", while others do not, as shown by a "-". Therefore, the table provides a valuable

Table 1.1: Advantages of Different Flow Control Techniques [MMH21].

	Clumsy FC	**Prediction Based FC**	**Distributed Flit Buffer FC**	**Improved FC**	**Injection Level FC**	**Flit Level FC**	**QLT FC**	**Fault Tolerant FC**
Reduced Area	-	-	✓	-	-	-	✓	✓
Reduced Cost	-	-	-	-	✓	-	✓	-
Reduced Design Complexity	-	-	✓	-	-	-	-	-
Reduced Latency	-	✓	✓	-	✓	✓	-	-
Increased Throughput	✓	-	-	-	-	✓	-	-
Reduced Power	-	-	-	-	✓	-	✓	✓
Congestion Control	-	✓	-	-	✓	-	-	-
Fault Tolerance	-	-	✓	✓	-	-	✓	✓
Better Buffer Utilization	-	✓	-	-	-	✓	✓	-

overview of the relative strengths of each technique with respect to the various criteria evaluated.

Arbitration, on the other hand, determines the access priority when multiple cores attempt to simultaneously access a shared resource router in the NoC. There exist many different arbitration processes [AS09]. For example, **Round Robin** arbitration offers a high level of fairness between cores by equally treating each port of entry. This ensures a fair distribution of the schedules. In this way, each input port is offered the same opportunity to access the output port, thus solving the starvation problem [LJL13]. **First Come, First Serve (FCFS)** is another arbitration in which the first packet that arrives is the first to be processed. This is the simplest of the scheduling algorithms. Additionally, the **Fixed Priority** arbitration assigns priorities to each packet. Packets with higher priority are processed before those of lower priority [Jai+15]. This arbitration technique is simple to use, but the critical path delay increases with the number of inputs. It also unfairly favors a single input port, leading to problems of low utilization for the other ports [Bec+12]. Other arbitration techniques that include, for instance, the **Dynamic Adaptive Arbiter**, where the buffer status can dynamically change the priority of the input port, and the **Ring Arbiters**, where the signal is generated by a token, are discussed and detailed in [Jia+13].

1.2.7 Pros and Cons of classical Electrical NoCs

The integration of thousands of heterogeneous cores and huge parallel computation capabilities suitable for High Performance Computing (HPC) [Viv21] are allowed. These parallelism capabilities obviously generate an enormous amount of data exchanges, making the communication medium a key element in overall system performance.

The evolution from dedicated wires and buses to NoC architectures in multi-core systems has been mainly driven by the need for efficient and scalable communication schemes. One of the key advantages of conventional electrical NoCs is their predictability. They can be designed to guarantee certain levels of performance, particularly when using deterministic routing algorithms. In addition, they offer a modular design, enabling designers to reproduce tiles (consisting of a processing element and a router) on the chip, thus simplifying design scalability. This modularity also facilitates the streamlining of the design and verification processes. Moreover, electrical NoCs effectively manage the complexity of on-chip communications, enabling power, latency, and bandwidth to be optimized according to the performance requirements of the applications and systems. Furthermore, the adaptability of its design allows different topologies to be implemented according to the particular requirements of SoC. In addition, their cost-effectiveness and capacity to accommodate intricate communication protocols render them a dependable and reasonably priced option for the vast majority of modern IT applications.

On the other hand, although transistors have become smaller, links remain the same size, resulting in dis-proportionality between the cost of communications and that of computations. In addition, in an extended many-core system, long-distance communications require the passage through many nodes. This leads to a significant increase in the dynamic power consumption. Moreover, as interconnections get closer together in increasingly dense architectures, electromagnetic interference and cross-talk become significant problems. Due to these constraints, different alternatives have been explored by researchers, which has led to the development of emerging technologies which hold the potential to get around some of these issues and open the door for a new era of intra-chip communications. Theses technologies include for example optical or millimeter-wave-based NoCs and will be further detailed in the following section.

1.3 Emerging Technologies

As the electronics industry continues to push the boundaries of miniaturization and efficiency, the design and innovation of on-chip interconnects becomes essential. In addition to the 3D technology discussed in Section 1.2, work has been done to implement emerging technologies in NoCs such as Optical Network-on-Chip (ONoC), WiNoC, RF-Interconnect and other alternatives (Figure 1.8). Table 1.3 summarizes a comparison among the WiNoC, 3D-NoC and photonic NoC [WJ14]. These innovations have led to the implementation of hybrid NoC technologies that use wireless and waveguide technologies with traditional ENoC. In our work, we have focused primarily on these emerging technologies and hybrid NoCs. These technological advances promise to bring significant improvements in speed, efficiency and flexibility for future systems.

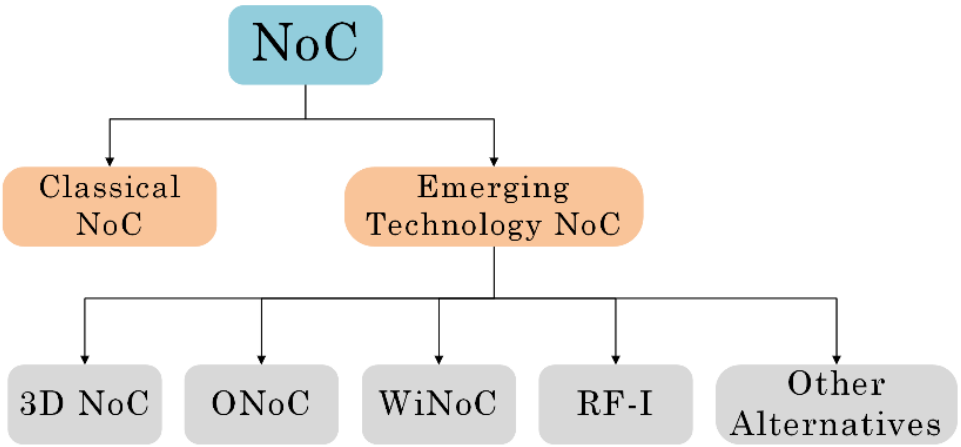


Figure 1.8: Technology based classification NoC [BBB17].

Since many years ago, traditional NoCs have served as the basis of intra-chip communications, allowing for effective integration and communication between the many components on a chip. Like any system, they are not without limitations. It is important to evaluate the advantages and disadvantages of traditional electrical NoCs given the emergence of new technologies and the increasing expectations for performance and efficiency. Likewise, it is crucial to look at emerging technologies aiming to overcome the challenges posed by traditional solutions.

1.3.1 Optical Networks on Chip

Integration of optical links into circuits based on Silicon Photonics (SiP) is considered as a major solution to address the bandwidth and energy limitations of electrical interconnections, giving rise to Optical Networks on Chip (ONoCs) [WNL17]. However, the development of SiP devices and the management of on-chip optical data transmission present many challenges and requiring a thorough understanding of the traffic requirements and constraints associated with integrating both electronic and photonic devices. In this regard, our Lab has carried out research pioneering the Architectural exploration of network interface for energy efficient 3D optical network-on-chip [Pha18; Luo18].

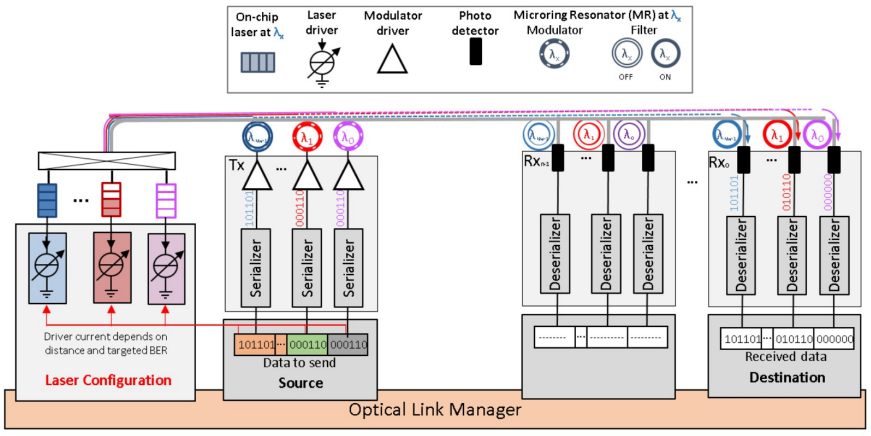


Figure 1.9: Nanophotonic interconnect NoC Optical (ONoC) [Lee22].

Figure 1.9 details an ONoC architecture , displaying photonic devices such as waveguides, multi-wavelength lasers, and microring resonators (MRRs) used as modulators and photo detectors. Light signals generated by the laser sources are modulated using wavelength-specific microring modulators with on-off keying (OOK) modulation, thus, converting parallel bit electrical signals into serialized sequences. These signals, possibly modulated into multiple wavelengths to accelerate transfer, traverse a shared waveguide with wavelength-division-multiplexing (WDM), facilitating parallel signal transmission. At the destination, the signals are filtered by microring filters, directing specific wavelengths to photo-detectors. These photo-detectors can convert them back into electrical signals for the respective chip nodes. Each ONoC node communicates with many others via the Signal Writer and Multiple Reader (SWMR) model, using WDM-enabled waveguides.

1.3.2 RF-Interconnect NoC

Transmission delays, power-hungry routers, and the less importance of scalability, all make switched networks-on-chip less attractive. An alternative is to use high-quality transmission lines. For medium-scale multi-core processors (CMPs), the interconnection infrastructure can eliminate the need for packet switching and offer energy and performance advantages over conventional mesh interconnections. A possible NoC strategy is represented by RF-Interconnects, which use radio frequency signals to connect SoC cores. With greater bandwidth and less power consumption than electrical NoCs, this new technology seeks to overcome their drawbacks without the physical limitations of wired interconnects.

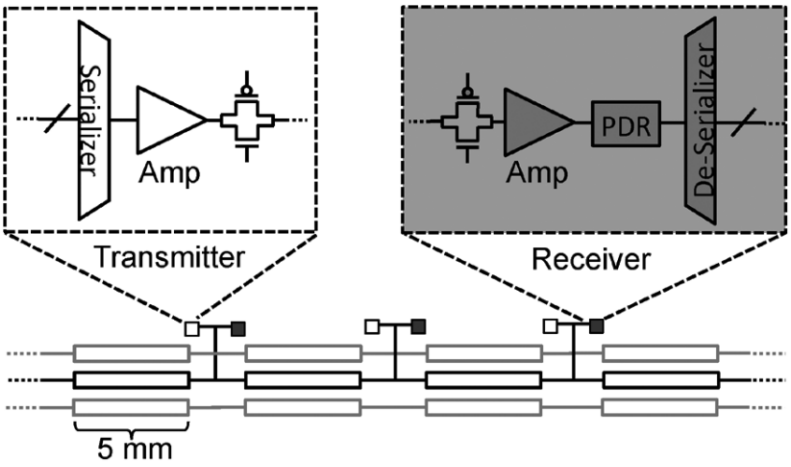


Figure 1.10: General schematic representation for the transmission line link interconnect. On the left, we have the Transmitter block, containing a Serializer followed by an amplifier (Amp). On the right, the Receiver block also contains an amplifier, followed by a Phase and data recovery (PDR) and a Deserializer. The two blocks are connected by a series of lines that represent the transmission line itself, with a measurement 5 mm. [Car+12].

For on-chip connectivity, particularly for moderately-sized CMPs, a correctly designed communication link based on transmission line can achieve extremely low latency and power consumption [Car+11]. Furthermore, in radio-frequency (RF) and microwave circuits, transmission lines are a typical component. The general architecture of a single transmission link with transmission circuits is shown in Figure 1.10 (it is encircled by neighboring links). The transmission circuit can typically be as basic as totally digital inverter-chain circuits, and as it becomes more complex, it enables faster data rates at typically lower per-bit energy costs. In [Ham+19], a novel radio frequency (RF) channel

access method for intra-chip communications is presented, overcoming the undesirable effects of multiple connections in conventional methods. Based on active access through associated transistors in a distributed topology, this method offers improved properties in terms of reflection and transmission coefficients along the transmission line. This approach is validated by an in-depth comparison with existing basic access methods (e.g. direct and capacitive). Measurements on a test circuit using 0.25 μm SiGe BiCMOS technology confirm the good performance of the proposed access.

1.3.3 Surface Wave Interconnect

SWI NoC [Kar+12] is an advanced hybrid interconnect architecture that combines an on-chip network based on metal interconnects and Zenneck's SWI technology. Preliminary results show that this hybrid architecture significantly reduces power consumption and improves performance compared to a traditional NoC, while offering a low additional cost in terms of hardware and surface area.

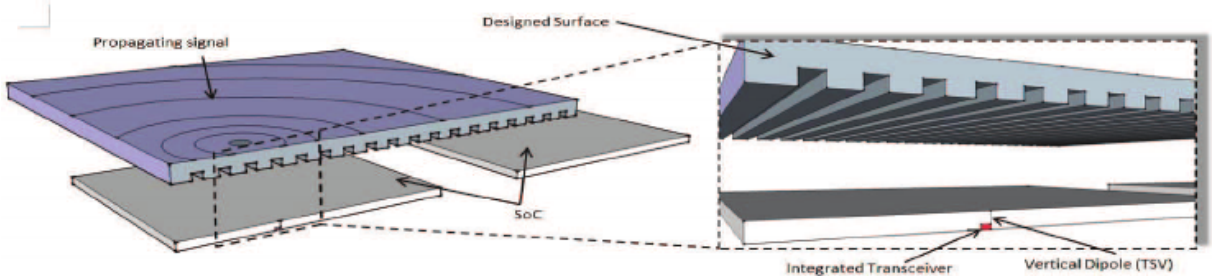


Figure 1.11: Surface wave implementation [Kar+12].

Figure 1.11 shows a schematic illustration of a SWI communication system applied to a SoC. It shows an SoC as the basis for the technology, topped by a surface specially designed for surface wave propagation. The waves generated are intended to transmit information efficiently through the SoC without the use of wired connections. An integrated transceiver is located at one end, responsible for converting electrical signals into surface waves and vice versa. It is also possible to make out a vertical dipole (TVD), which acts as an antenna for receiving or transmitting waves.

Even though SWI offers promising advantages, it also suffers from different technical challenges and limitations. First, it can be difficult and expensive to manufacture surfaces that are intended to efficiently guide surface waves. In addition, there are challenges associated with interference management and signal loss, particularly in congested environ-

ments where multiple signals may cross paths. Another crucial element is the requirement for precise synchronization between transmitters and receivers to preserve data integrity. Hence, continued advancements in materials, design, and modulation techniques are necessary to overcome these obstacles and make SWI a viable solution for interconnecting the upcoming generations of SoC.

1.3.4 Wireless Networks on Chip

A significant work has been done to implement Wireless Networks on Chip WiNoC to traditional NoCs. As our first contribution (detailed in Chapter 2 of this manuscript) relies on the integration of WiNoCs in combination with ENoCs, the different key components of WiNoCs will be presented along with describing their architectures, the channel access methods and protocols used to manage communications between nodes in a network.

1.3.4.1 Key components

Utilizing the very wide frequency band offered by the Complementary metal oxide semiconductor (CMOS) process is the key benefit of WiNoCs [Ort+19]. On-chip wireless signal speed propagation allows long distance communications without an increase in latency compared to electrical NoCs that suffer from more router crossings.

WiNoC essentially integrates wireless intra-chip links within an ENoC. As depicted in Figure 1.12, when a packet reaches a wireless interface, it is serialized, modulated, and then transmitted by the antenna in a specific pattern. Upon arrival, it undergo demodulation and deserialization. On the other hand, the core is also connected to an electrical router allowing traditional electrical communications. By bypassing intermediate router steps, WiNoC drastically reduces the latency of long-distance and broadcast/multi-cast transmissions.

1.3.4.2 Architectures

In purely wireless architectures (Figure 1.13), wired connections are replaced with wireless interconnections, aiming for scalability, cost-effectiveness, and flexibility. In contrast to entirely wired NoC designs, all node connections in this architecture are wireless. However, the constraints of channel range and energy consumption hinder the feasibility of entirely wireless NoCs, particularly when anticipating chips with hundreds to thousands of cores [Zha+11; ZW08].

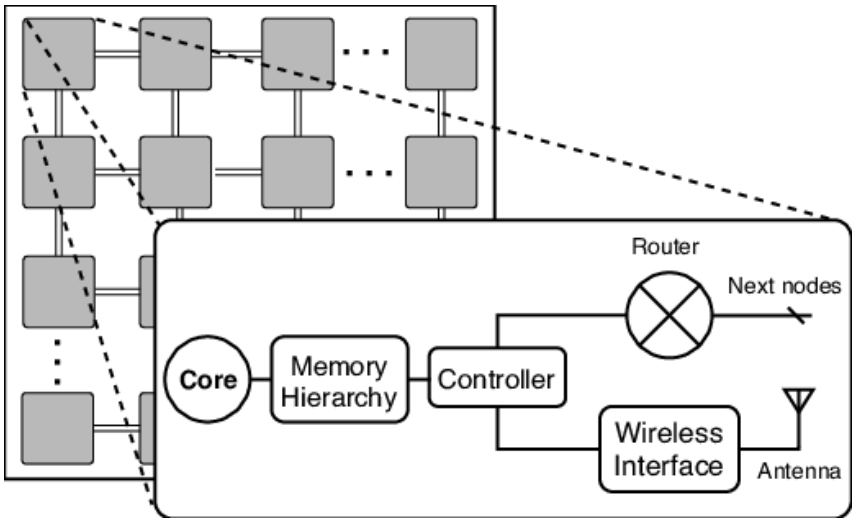


Figure 1.12: Sketch of a Wireless NoC architecture [Tim+18].

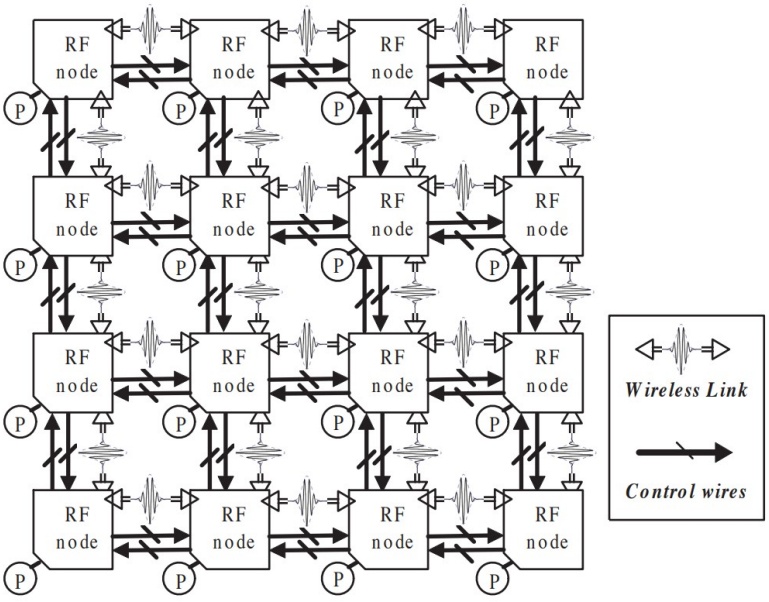


Figure 1.13: A pure multi-channel wireless-based NoC with a mesh topology [Zha+11].

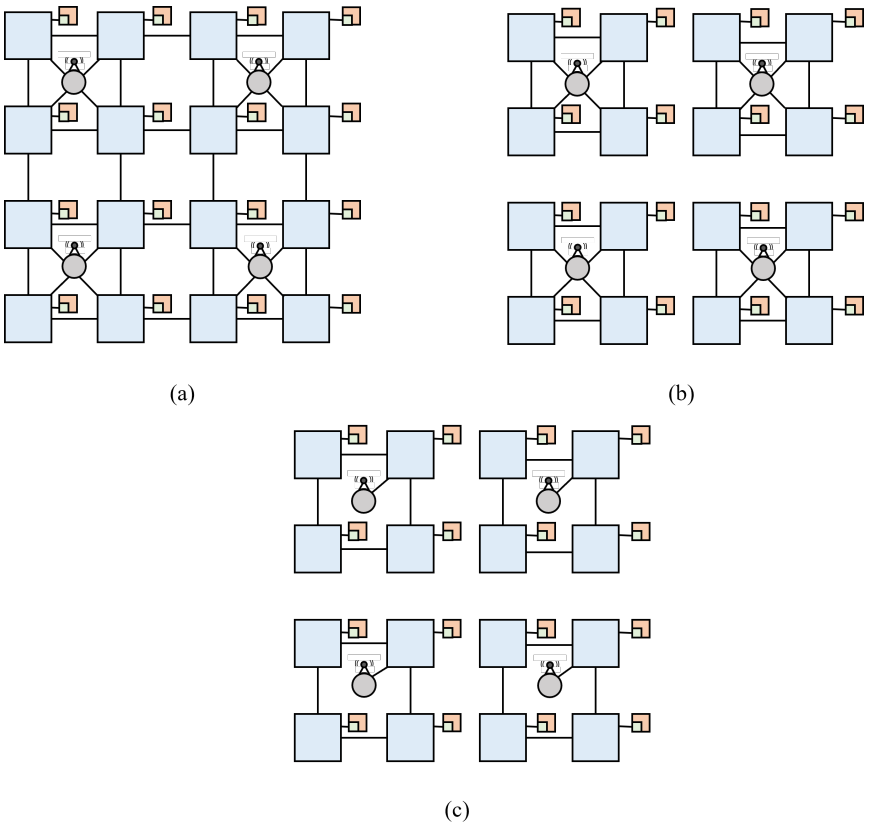


Figure 1.14: Example of Hybrid NoC architecture.

Several wireless and hybrid architectures have been proposed in the literature in which three different architectures are highlighted in Figure 1.14. Figure 1.14-a showcases a many-core that is fully connected with a classic 2-D mesh NoC, while the wireless routers are shared by several cores and accessible by using the electrical routers [OSR19]. Figure 1.14-b represents a many-core that is divided into clusters of cores where all the routers within the same cluster are connected to one antenna. In such example, intra-cluster communications are conducted with electrical NoCs that are composed of electrical routers, whereas inter-cluster communications are only possible through wireless routers integrating an antenna [GD21]. A similar architecture than that shown in Figure 1.14-b is depicted in Figure 1.14-c that differs only with the number of routers connected to an antenna within a cluster where only one router is connected to the antenna within each cluster [Fra+21]. Several other architectures exist, such as architectures with a heterogeneous placement of wireless routers within a classic electrical NoC [Le+17].

The number of routers connected to a single antenna (wireless interface) is also an im-

portant aspect of WiNoC. By increasing the number of connections, the communication load becomes more intensive thus, impacting the system's performance. An overloaded antenna can lead to longer transmission delays as each router must wait for its turn. Additionally, increased communication traffic raises the chance of interference and collisions reducing the throughput. Examples of 16 connections to a single antenna [Deb+13], and 4 connections to a single antenna [Cat+17] can be found in the literature without any real motivation for their choice.

There are also different architectures, including those with one dedicated wireless router per core [Fra+21], or those with a heterogeneous placement of wireless routers within a classic electrical NoC [Le+17].

1.3.4.3 Channel access methods and protocols

In this section, several channel access protocols used to manage communications between nodes in a network are presented. For instance, the use of the token-passing channel access method is widely studied in the literature [MIG15; Deb+13]. The Token Passing protocol involves the transmission of a 'token' from one node to another as shown in Figure 1.15. The node must hold the token in order to transmit the data, thus ensuring that only one node is transmitting data at a time and for a specific period of time avoiding collisions [KPJ08]. However, the passage of tokens is limited by the fact that the tokens must follow a predefined order [Fas+21]. Additionally, this method is limited by its reduced scalability, increased latency, and complexity of implementation, making it less suited to the fast, efficient communication requirements of modern embedded systems.

Furthermore, the Code Division Multiple Access (CDMA) method employs distinct spreading codes to facilitate simultaneous channel access for multiple nodes while ensuring interference-free transmission [Vid+12]. CDMA can improve the performance and reduce the energy dissipation [Vij+14]. However, the performance gets worse as the number of cores increases in the network. Additionally, this method is limited by its complex management of interference and code assignment, and by the challenges of dense integration and signal processing on a single chip.

Moreover, the Time Division Multiple Access (TDMA) assigns each node a specific time slot for data transmission, ensuring collision-free transmissions by permitting only one node to transmit within its designated slot [Pan+09]. Nevertheless, it is limited by its temporal rigidity, which can lead to bandwidth under-utilization and its increased latency, especially in systems with variable traffic loads and heterogeneous communication

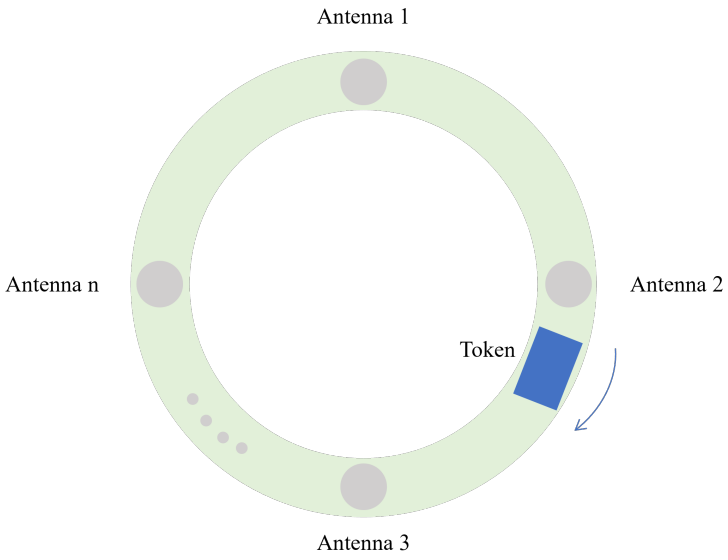


Figure 1.15: Token.

requirements.

Furthermore, the Code Sense Multiple Access (CSMA) protocol [CL14] channel control mechanism was presented as an alternative to token-passing flow control in WiNoC applications that lack wired links. This mechanism detects the absence of other traffic before transmitting on a shared channel. The limitations of this mechanism are presented in the delay encountered when determining the status of the shared transmission channel and postponing communications until the channel is available.

Finally, random access protocols, such as ALOHA and Slotted ALOHA, allow each node to send data at any time. In the event of a collision, the node waits a random period of time before attempting to send the data again. In both cases, if two messages are sent at the same time, they collide on the communication channel, and neither is received correctly. After a transmission failure, the nodes wait for a random delay, hoping that the next attempt will be collision-free. Random access protocols in NoCs can lead to frequent conflicts and collisions under heavy load, resulting in reduced efficiency and problems of reliability and predictability for on-chip communication.

Table 1.2: Summary comparison of the key features for current and emerging on-chip interconnects [Kar+16].

Features	**Metal wire**	**Transmission lines (RF-I)**	**Wireless interconnect (WiNoC)**	**Optical interconnect**	**Surface wave interconnect (SWI)**
Power	Dynamic power that is proportional to the wire capacitance and voltage.	Power consumption is relatively tolerable.	High free space power dissipation.	High power consumption.	Power consumption is relatively tolerable.
Signal Decay	Limited by latency, which increases exponentially without repeaters.	Low signal decay and dissipation.	High decay, inversely proportional to distance.	Very low signal decay and dissipation	Low signal decay and dissipation inversely proportional to square root of distance.
Reliability	Possible cross-talk exists.	Cross-talk exist (capacitor and inductor coupling).	Noise coupling to the antenna and possibility of multipath interference.	High signal integrity.	Less subject to noise coupling.
Fan-out	Needs extra power for multi-drop bus (stubs) and lowers propagation velocity.	Stubs cause impedance discontinuity, which will lead to signal reflection.	Limited by transmission signal propagation cover area only	Require optical splitters and combiners that decay the optical signal (3dB per splitter).	Limited by transmission signal propagation cover area only.
Bandwidth	Limited by interconnect delay; thus, bit rate is dependent on distance.	Limited process technology transistor cut-off frequency, which is currently 100 to 200 Gbps.	Limited process technology transistor cut-off frequency, which is currently 100 to 200 Gbps	Very large bandwidth with multi-wavelength capability up to 500 Gbs.	Limited process technology transistor cut-off frequency, which is currently 100 to 200 Gbps.
Complexity	Need repeaters for cross-chip communication that consume transistors, via and restrict floor planning. However, it still is the cheapest and simplest interconnect.	Medium complexity required: (1) integrated transceiver, (2) wide thick wires and spacing (12–45), m n (3) may require shielding wires and plans to overcome coupling, (4)matching circuits in case of forking path	Medium complexity required: (1) integrated transceiver, (2) integrated antenna or cluster of antennae based on the required bandwidth and the operational frequency.	High complexity and some devices are not CMOS compatible, required: (1) laser source, (2) photo detectors, (3) modulators and filters, (4) waveguide, (5) laser-waveguide couplers in case of off die laser source, (6) nanoscale mirrors, (7) splitters/combiners.	Medium complexity, required: (1) integrated transceiver (2)integrated designed surface (3) integrated transducer.

1.3.5 Overall comparison

Table 1.2 represents a detailed comparison of the key features of current and emerging on-chip interconnects [Kar+16]. Five types of interconnects are evaluated: Metal wire, Transmission line (TTL/RF), Wireless interconnect (WiNoC), Optical interconnect and Surface wave (SW). The characteristics evaluated in this table include power consumption, signal degradation, reliability, fan-out, bandwidth and complexity. Regarding first the power consumption, Metal wire exhibits dynamic power consumption, while WiNoC and SW are more tolerant. As for signal degradation, most interconnects show varying levels of delay and dissipation. Regarding reliability, concerns such as cross-talk and coupling are highlighted. In terms of fan-out, the optical interconnect requires specific optical components to transmit and detect the signal. Bandwidth on the other hand, is limited for most interconnects, with the exception of optical interconnect, which offers high bandwidth. Finally, in terms of complexity, optical interconnection is identified as the most complex, requiring many specific components. According to the various criteria evaluated, this table provides a valuable overview of the relative advantages and disadvantages of each interconnection technique.

Table 1.3 summarizes the main differences between emerging 3D NoC, ONoC and WiNoC. In terms of hardware requirements, 3D-NoC uses multi-layer stacks with vertical connections, while photonic NoC relies on silicon photonic components. WiNoC on the other hand uses on-chip antennas. Regarding the communication bandwidth, 3D-NoC offers higher connectivity with fewer hops thanks to shorter paths, photonic NoC utilizes high-speed optical links, while WiNoC offers high bandwidth via wireless channels with regard to energy consumption, the three NoCs show different advantages. 3D-NoC, for example, uses a shorter path, photonic NoC consumes less energy for optical transmission, while WiNoC has a low-power in wireless communication. In terms of reliability, 3D-NoC presents challenges such as TSV interconnect failure, while photonic NoC faces cross-talk issues in photonic waveguides. WiNoCs on the contrary, is generally considered less reliable due to the challenges of on-chip antenna integration.

A landscape of innovative solutions is developed by comparing the various emerging NoC technologies all together, each with its unique advantages catered to distinct applications and design challenges. These technologies promise to significantly improve the performance and efficiency of systems-on-chip, thus opening up exciting possibilities for the future of intra-chip communications. The choice between these technologies will ultimately depend on the particular needs of each application and the trade-offs between

Table 1.3: Comparisons among the WiNoC, 3D-NoC and photonic NoC [WJ14].

	3D-NoC	Photonic NoC	WiNoC
Hardware requirement	Multiple-layers stacks	Silicon photonic components	On-chip antennas
Communication bandwidth	Higher connectivity (vertical links)	High-speed optical links	High wireless channels
Power consumption	Shorter path (less hops)	Low power in optical transmission	Low power in wireless
Reliability	Through silicon via (TSV) failure	Crosstalk in photonic Waveguides	Less reliable in wireless
Challenge	Thermal issue due to high-power density	Integration of on-chip photonic components	Integration of on-chip antennas

cost, performance, and implementation complexity. As these technologies mature, they will shape the development of the up-coming electronic devices, leading to significant improvements in electronics and computing.

1.4 NoC Traffic patterns

NoC traffic can be typically classified as synthetic or Benchmark applications. The latter are traces from real-life application workloads.

1.4.1 Synthetic Traffic

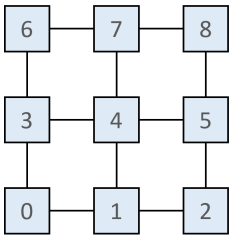
A synthetic traffic is an experimental traffic used to evaluate the communication architecture and attempts to reproduce real-world application traffic. There are two types of synthetic traffic: regular (uniform) and irregular (non-uniform).

An example of a regular traffic is the random traffic pattern (or also called uniform traffic pattern), which is the most common traffic pattern used in network evaluation. This is related to the fact that it represents the worst-case test that enables the evaluation of NoC performance under highly stressful conditions with non-predictive load. This can help in identifying bottlenecks and network performance limits. In a random traffic, each source has an equal probability of sending data to each destination:

$$p_{sd} = \begin{cases} \frac{1}{N_c - 1} & i \neq j \\ 0 & i = j \end{cases} \tag{1.1}$$

The message distribution is presented using a traffic matrix. P, denoted by the symbols

p_{sd}, indicates the percentage of traffic from node s to node d, as represented in Figure 1.16, which shows an example of the matrix for a mesh topology with the number of cores N_c = 9 cores.



$$P = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} & p_{04} & p_{05} & p_{06} & p_{07} & p_{08} \\ p_{10} & p_{11} & p_{12} & p_{13} & p_{14} & p_{15} & p_{16} & p_{17} & p_{18} \\ p_{20} & p_{21} & p_{22} & p_{23} & p_{24} & p_{25} & p_{26} & p_{27} & p_{28} \\ p_{30} & p_{31} & p_{32} & p_{33} & p_{34} & p_{35} & p_{36} & p_{37} & p_{38} \\ p_{40} & p_{41} & p_{42} & p_{43} & p_{44} & p_{45} & p_{46} & p_{47} & p_{48} \\ p_{50} & p_{51} & p_{52} & p_{53} & p_{54} & p_{55} & p_{56} & p_{57} & p_{58} \\ p_{60} & p_{61} & p_{62} & p_{63} & p_{64} & p_{65} & p_{66} & p_{67} & p_{68} \\ p_{70} & p_{71} & p_{72} & p_{73} & p_{74} & p_{75} & p_{76} & p_{77} & p_{78} \\ p_{80} & p_{81} & p_{82} & p_{83} & p_{84} & p_{85} & p_{86} & p_{87} & p_{88} \end{bmatrix}$$

(a) (b)

Figure 1.16: Example of N_c = 9 cores based on (a) network mesh topology with its respective (b) frequency matrix of communication between cores.

On the other hand,an example of an irregular traffic pattern is the well known transpose traffic. Transpose traffic is a distinct traffic pattern within a network which is characterized by the transposition of communication pairs. Given an example of a network composed of nodes labeled with coordinates (i,j). In the transpose traffic pattern, the node situated at coordinates (i,j) communicates with the node at coordinates (j,i). This operation creates a bottleneck along the diagonal bisection of the network, as all packets must traverse it. Incorporating transposed traffic with appropriate NoC routing algorithms can lead to a significant imbalance in the network load. This is because counter-clockwise links around the center of the mesh are used, while clockwise links remain unused, creating an uneven distribution of traffic. This form of traffic pattern is notably prevalent in applications that require matrix transpose operations. There are also many other irregular traffic patterns such as shuffle, bit reverse, bit rotation, etc. that are detailed in [DT04; TSJ]

1.4.2 Benchmark applications

1.4.2.1 Benchmark suites

Real workload benchmarks offer a realistic traffic distribution. Today, the Stanford Parallel Applications for Shared Memory (SPLASH-2) [Woo+95], Princeton Applica-

tion Repository for Shared-Memory Computers (PARSEC) [Bie+08], Multi-Constraint System-Level (MCSL) [Liu+11] and Standard Performance Evaluation Corporation (SPEC) [SPE] benchmark suites are commonly used to study the performance of CMPs. However, the collection of programs in the SPLASH-2 benchmark suite is strongly oriented towards HPC and graphics programs and does not include certain parallel models used in other application areas. Unlike SPLASH-2, the collection of programs in the PARSEC benchmark suite already includes some programs that reflect current computing problems. An in-depth comparison between the SPLASH-2 and PARSEC benchmarks was carried out by Bienia et al. [BKL08] taking into account different configurations for the simulations. The PARSEC benchmark contains thirteen applications from different domains and therefore with different characteristics. Table 1.4 summarizes the different applications with application domains and problem size [DM14]. Although these benchmarks are not new, they are still used in various recent works [AT23; Man+23; Mat21].

Table 1.4: Overview of PARSEC workloads [DM14].

	Benchmark	**Application Domain**	**Parallelism Model**	**Working Set**	**Communication**
1	blackscholes	Computational finance application	data-parallel	small	low
2	bodytrack	Computer vision application	pipeline	medium	medium
3	canneal	Electronic Design Automation (EDA) kernel	data-parallel	huge	high
4	dedup	Enterprise storage kernel	pipeline	huge	high
5	facesim	Computer animation application	data-parallel	large	medium
6	ferret	Similarity Search application	pipeline	huge	high
7	fluidanimate	Computer animation application	data-parallel	large	medium
8	freqmine	Data mining application	data-parallel	huge	medium
9	raytrace	Computer animation application	data-parallel	medium	high
10	streamcluster	Machine learning application	data-parallel	medium	medium
11	swaptions	Computational finance application	data-parallel	medium	low
12	vips	Media application	data-parallel	medium	medium
13	x264	Media application	pipeline	medium	high

For example, **blackscholes** is an application that forms part of the Intel RMS (Rigorous Mathematical Simulation) benchmark suite. This application utilizes the Black-Scholes Partial Differential Equation, which is widely acknowledged for its role in financial mathematics for option pricing [BS73]. The application specifically calculates the prices of a portfolio of European options. In an effort to minimize communication and synchronization overhead, blackscholes employs coarse-grained parallelism, which breaks down the program into sizable tasks. Interestingly, this application exhibits a low packet injection rate per processor, and thus network congestion is typically low. Despite this, it is observed that, regardless of NoC having 64 or 256 cores, each processor generates a message, indicative of the effective utilization of the application of all available cores. In particular, the first core, which is primarily designated to distribute tasks to other cores within the network, processes the bulk of the generated messages.

Dedup is an another application, and it is a kernel derived from the PARSEC suite. Dedup is designed to decrease the size of data streams by employing global and local compression techniques [SW00; QD02]. It operates using medium-grained parallelism, which is a compromise between fine- and coarse-grained parallelism. Consequently, dedup's application is parallel in two dimensions. This characteristic not only yields optimal performance but also gives rise to a high packet injection rate. One of the hallmarks of dedup is its effective task distribution, as it disperses the tasks nearly uniformly across all cores.

The **x264** benchmark is a performance testing application that evaluates the speed and efficiency of a computer when running the x264 video codec [MV23]. The x264 codec is widely used for video compression and is renowned for its ability to deliver high video quality at relatively low bit rates. The x264 Benchmark application measures the time it takes a computer to encode a video using this codec, providing an indicator of the system performance in video processing. The test results can be used to compare performance between different computer systems or to assess the impact of hardware upgrades on video encoding performance. In addition, the x264 Benchmark is often used by computer enthusiasts and professionals alike to optimize system settings for the best possible performance when encoding videos.

The **raytrace** benchmark application is designed to evaluate the ray tracing performance of a computer system, putting processor and graphics card to the test through a series of complex graphics rendering scenarios. This advanced rendering technique simulates the behavior of light to produce ultra-realistic images, taking into account reflections, refractions, and shadows [Whi05]. Raytrace measures the system response time and the

frame rate per second, providing an overall score that helps users compare performance between different systems or configurations, and identify potential hardware upgrades or optimization needs to improve ray tracing performance.

A **streamcluster** benchmark application is designed to evaluate the performance of a computer system in processing and analyzing real-time data streams [OCa+02]. This application simulates scenarios where large quantities of data are constantly being generated and need to be processed rapidly to extract relevant information, as in the use cases of online stream processing, recommendation systems, or social network analysis. This benchmark measures the speed with which the system can process data, manage flows and perform clustering operations, providing an overview of system performance under high load conditions and in real time. The results can help identify bottlenecks, compare performance between different systems, and guide hardware and software configuration choices to optimize data flow processing performance.

1.4.2.2 Characterization

Benchmark characterization is an essential process for evaluating and comparing the performance of NoCs. Benchmarks are designed to provide a standardized set of workloads that simulate a variety of typical applications and usage scenarios. Effective characterization involves the analysis of several aspects of a benchmark application, such as memory access patterns, CPU utilization, branching behavior, and communication patterns in multi-core systems or NoC. This analysis identifies the performance and resource requirements of applications, facilitating the optimization of processor architectures, operating systems and underlying hardware. Researchers and engineers use this information to improve system design, ensure compatibility with target workloads, and guide the development of new technologies to meet end-user needs. In short, benchmark characterization is fundamental to technological progress in computing, as it establishes a direct link between simulation performance and real-world requirements.

The benchmarks can be simulated using a dedicated simulator to extract communication traces. Communication traces list all injected packets during application execution. Each injected packet has information about the source, destination, and injection time. From the communication traces obtained with the simulator, a traffic characterization can be performed.

Furthermore, the injection times, which determines the packet inter-arrival times, form an important piece of information in performance studies. This involved a better statisti-

cal management of both interconnection simulators and analytical methods as their cycle by cycle management is much slower given the fact that tens or even hundreds of thousands of packets are injected within a definite interval. In general, several distributions can be used including i) General distributions, ii) Geometric distributions and iii) Poisson distributions.

i) The general distribution is the most flexible, as it can model a wide variety of behaviors, but often requires more parameters to be characterized. The general form of this distribution depends on the specific parameters chosen for the model. It can be represented by a probability density function (PDF) or a cumulative distribution function (CDF), with no single standard form. Its parameters can vary according to the model chosen and often include mean and standard deviation along with other model-specific parameters.

ii) The Geometric distribution on the other hand is useful for modeling the waiting time until the first packet arrives after a period of no traffic, which is common in on-chip networks where periods of activity can be intermittent. It is generally defines by:

$$P(X = k) = (1 - p)^{k-1} \times p \tag{1.2}$$

where p is the probability of success for a single trial and k is the number of trials to the first success.

iii) Additionally, the Poisson distribution is often used to model the number of events occurring in a fixed time interval, which can be relevant for predicting the number of packets arriving in a given time interval in an on-chip network. It is defined by:

$$P(X = k) = \frac{e^{-\lambda}\lambda^k}{k!} \tag{1.3}$$

where λ is the average rate of events per cycle and k is the number of events.

In summary, each distribution offers different advantages depending on the specific nature of traffic and analysis needs in on-chip networks.

1.5 Conclusion

In conclusion, this chapter offers a comprehensive analysis of the state-of-the-art in NoC architectures. From the detailed study of the historical evolution of microprocessors and the various aspects of NoCs , such as topologies, data transfer formats, components,

routing algorithms, and flow control methods. These elements are crucial to understanding the operation and performance of NoCs.

However, it is essential to recognize that despite their many advantages, traditional NoCs have limitations, particularly in terms of scalability and energy efficiency. Examination of emerging technologies such as optical, RF-interconnect and wireless NoCs revealed advances and innovations that are redefining the capabilities of systems-on-chip communications. This has led us to examine the emerging technologies. Among these, hybrid NoCs stands out by combining the best features of different approaches and integrating new technologies to improve performance and reduce energy consumption. As the complexity of systems-on-a-chip continues to grow, the exploration and adoption of these emerging technologies become essential to sustain advances in the field.

Finally, the analysis of traffic patterns, both synthetic and real applications, has provided an in-depth characterization of the workloads and performance requirements of today's NoC architectures. Synthetic traffic, which enables the basic performance and robustness of NoC architectures to be assessed, and real applications, which is used to understand the practical and direct impact on real-life performance. This two-sided exploration is crucial to the development of NoCs that are not only theoretically valid, but also efficient and adaptive to the concrete requirements of modern applications.

Chapter 2

AMHNOC : Analytical Model for Hybrid NoC

2.1 Introduction

Several years ago, the rise of many-core architectures became very noticeable, primarily due to the implementation of massive parallelism on a single chip. With the integration of thousands of heterogeneous cores, they allow huge parallel computation capabilities suitable for HPC [Viv21]. Recent advances in silicon integration offer opportunities to use emerging on-chip interconnects such as WiNoCs [Ort+19]. The use of the very wide frequency band offered by the CMOS process is the key benefit of WiNoCs. On-chip wireless signal speed propagation allows long distance communications without an increase in latency, whereas electrical NoCs suffers from more router crossings. WiNoCs can be used alone; therefore, all data exchanges between two nodes are done with wireless communications. However, the wireless bandwidth is generally lower than electrical NoCs, hence they can be combined as a hybrid NoC. In this case, communications can either be handled by wireless links for long-distance communications or by electrical routers for the short ones.

Indeed, the design space for exploring such architectures is complex and time-consuming, as the performance evaluation mostly relies on simulations that are slow processes, particularly for large multicore systems. On the other hand, mathematical modeling offers a good trade-off between computation times and accuracy level. Analytical models for on-chip interconnects are based on the queueing theory and provide performance metrics such as average buffer utilization and average packet latency. However, current analytical models only support homogeneous interconnects and are not adapted for hybrid ones. Section 2.2 details the two methods for analyzing the performance of NoC: simulator and analytical methods based on the queueing theory.

In Section 2.3, we propose Analytical Model for Hybrid NoC (AMHNOC), based

on the $M/G/1$ queueing model, to evaluate the average packet latency and the network throughput of a hybrid NoC, where packets can be routed on electrical, wireless, and mixed paths between cores. This model considers the Poisson distribution for communication traffic. Routers are based on round-robin arbitration, whereas wireless routers are based on token-passing channel access. The service time and channel capacities of electrical and wireless can be set separately to meet real hardware designs [OSR19]. This model provides a degree of heterogeneity useful to explore future design paradigms, such as chiplet or accelerators-based architectures [Viv21]. The main features of hybrid models that are not satisfied by conventional models are: i) wireless channel access differs from classic electrical router and is based on a token passing medium access control, ii) communication bandwidths between electrical and wireless mediums are heterogeneous, and iii) electrical and wireless routers latencies (service time) may differ. The proposed analytical model addresses these limitations, and, to the best of our knowledge, this is the first analytical model for a hybrid on-chip interconnect.

The parameters to be explored in the proposed model are latency, throughput and saturation. For the energy consumption, this will be calculated in a different way, and analytical methods have already been proposed which estimate and calculate energy consumption analytically [Bha22; GNR08], whereas during my work we concentrated on latency, throughput and saturation.

2.2 Performance Evaluation

NoCs have emerged as a promising solution to the challenges of communication between components in today's highly integrated SoC. NoCs play a key role in the overall performance and energy efficiency of SoC, particularly in areas such as embedded systems, data centers, and artificial intelligence. Due to the increasing complexity of NoCs and their importance for system performance, there is a need for advanced performance analysis methods. These methods can be classified into two main categories: analytical methods and simulation.

Analytical methods are approaches based on mathematical models that attempt to characterize the performance of a NoC in terms of parameters such as throughput, latency and energy consumption. These methods are often used in the early phases of design as they can provide quick insight without the need for large-scale simulation. However, due to their simplifications, they may not fully capture the design details of a NoC.

On the other hand, simulators are tools that model the behavior of a NoC in a detailed manner. They can take into account different aspects of the design, such as topology, routing algorithms, and buffer management policies. Although simulators can provide more accurate results, they are often slower and require more computing resources than analytical methods.

In this section, an overview of the state-of-the-art in performance analysis methods for NoC will be illustrated. We will explore the different available techniques and tools, examining their advantages and disadvantages.

2.2.1 Evaluation metrics

In the analysis and evaluation of NoCs, it is important to have reliable and relevant evaluation criteria, commonly known as metrics. These metrics are crucial to quantifying the performance, efficiency and quality of service of a NoC. They not only allow designers to compare different NoC architectures and configurations, but also follow the evolution of performance throughout the development cycle. In this section, we examine the key evaluation metrics used in NoC analysis, such as throughput, latency, energy efficiency, and many others. A solid understanding of these metrics is essential for any NoC design, optimization, and validation effort.

The **latency** is the delay of a packet between the source and the destination. Low latency is often desirable, especially for real-time applications. It is an essential measure of NoC performance, and is usually expressed in terms of clock cycles. In a NoC, the latency can be affected by several factors, including: i) Routing distance where the more nodes a packet has to traverse to reach its destination, the higher the latency, ii) Network utilization rate that indicates that packets may have to wait before being forwarded if the network is heavily used, thus increasing latency, and finally iii) Routing algorithm where different routing algorithms can affect the latency by determining the path packets take through the network.

$$Latency = Q + T + D \tag{2.1}$$

Generally speaking, the latency is the summation of queuing time Q, which is the time spent in the queue before the packet can be processed or transmitted. Service time T generally comprises the time taken to process the packet at the source node, including the time taken to prepare for transmission, which may include header serialization time.

And transmission time D is the time required for the packet to travel through the network to its destination, this time includes the encoding of the entire packet on the transmission medium, often expressed as S/W where S is the packet size and W is the channel bandwidth.

Figure 2.1 shows the evolution of the latency versus packet injection rate. We can clearly see the **Saturation Packet Injection Rate (SPIR)** which is an important metric for evaluating the performance of NoCs. This metric represents the packet injection rate at which the network reaches saturation, i.e., the point at which the increase in traffic load (the packet injection rate) leads to a disproportionate increase in network latency. In other words, when the network is operating below the saturation packet injection rate, it can efficiently handle traffic with minimal increase in latency. However, once this rate is reached, latency begins to increase rapidly as more and more packets are injected into the network. The objective in designing a NoC is generally to maximize the saturation packet injection rate, as this means that the network can handle a greater traffic load before encountering performance problems.

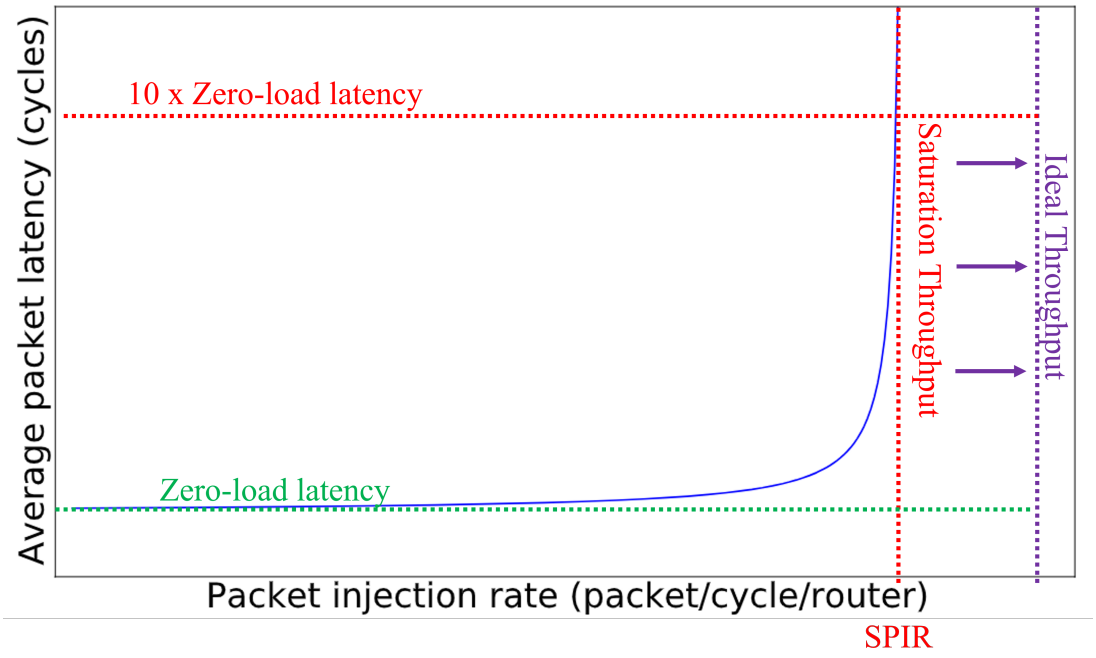


Figure 2.1: Average latency relation to packet injection rate.

Using the SPIR, we can also determine the **throughput** which is the amount of data

that can be transmitted per unit of time. It gives an idea of the network's capacity to manage traffic. In general, the throughput can be quantified by:

$$Th = \begin{cases} PIR & if\ PIR < SPIR \\ SPIR & if\ PIR \geq SPIR \end{cases} \tag{2.2}$$

Another metric used by NoC designer, **energy efficiency**, measures the amount of energy required to transmit a certain amount of data. With the trend towards miniaturisation and portability, energy efficiency has become an increasingly important criterion. Furthermore, **reliability** is also considered a crucial metric in the evaluation of performance of NoC as it represents the ability of the network to transmit data without errors. The latter can occur due to various factors, such as noise or manufacturing faults. The energy in a NoC is the sum of static energy and dynamic energy:

$$E = E_{static} + E_{dynamic} \tag{2.3}$$

Static and dynamic energy represent two essential aspects of energy consumption. Static energy refers to the electrical power dissipated due to current leakage and conduction losses, even when the circuit is idle or operating at minimum frequency. This component is significant in advanced manufacturing technologies, where current leakage becomes more prevalent. Dynamic energy, on the other hand, refers to the power consumption generated by signal switching and logic operations during active circuit operation. The latter is highly dependent on circuit activity and operating frequency. Reducing dynamic energy means optimizing designs to minimize signal switching, and applying techniques such as dynamic power management. The effective management of both types of energy is crucial to achieving overall low power consumption levels in systems-on-chip.

2.2.2 Evaluation tools

2.2.2.1 Simulators

A NoC simulator is a modeling tool that simulates the behavior of a communication system integrated within a SoC. It is designed to test and evaluate the performance and efficiency of communication architectures prior to their hardware implementation. Figure 2.2 describes the general overview of a simulator that is located at the center of various input and output streams. Simulator inputs consist of the configuration parameters that

define the basic characteristics of the network, such as topology, number of nodes, router buffer size, etc., traffic patterns (synthetic traffic or embedded application traces), traffic parameters, router parameters (such as routing algorithms) and environmental parameters. On the other hand, the simulator's outputs generally include the performance metrics such as the network throughput, its latency, and power.

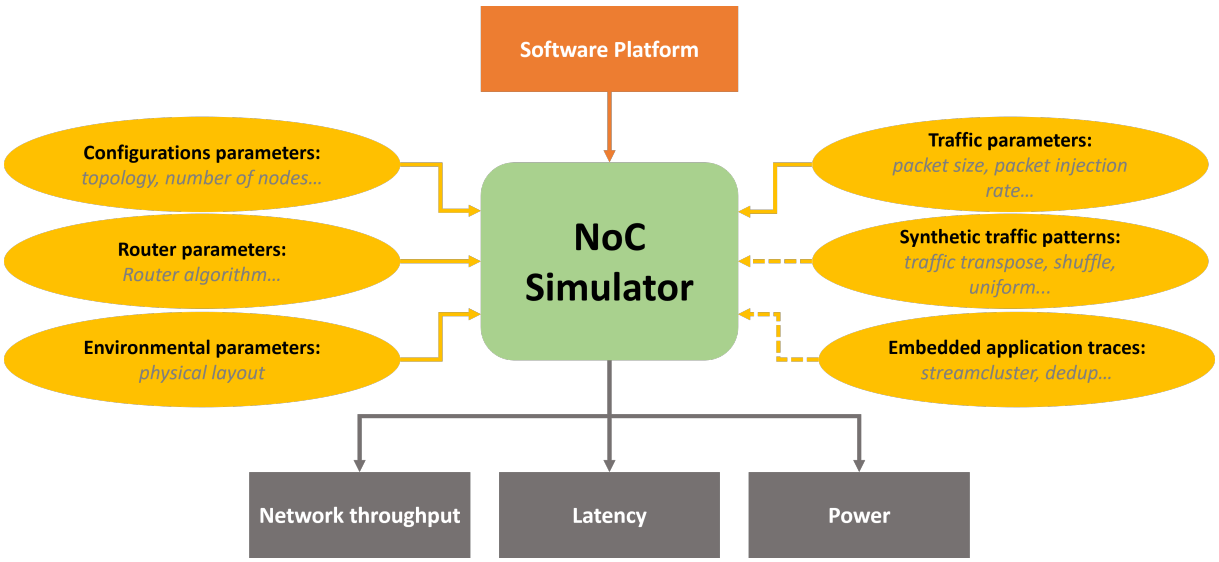


Figure 2.2: Overview of NoC simulator.

As previously mentioned, NoC simulators are essential for modeling and evaluating communication mechanisms within complex systems-on-chip, to optimize their performance and energy efficiency. They are indispensable tools for researchers and engineers in the field of system-on-chip. These tools make it possible to model, simulate, and analyze various NoC aspects, such as topology, routing, congestion management, and arbitration strategies, in a controlled environment. By using NoC simulators, it is possible to explore a wide range of design scenarios and optimize NoCs for specific applications. In simulation-based methods, several simulators have been developed [AS19]. There are several types of simulators, each offering a level of precision: Cycle-accurate simulators (Noxim [Cat+16]), enable detailed evaluation by taking every clock cycle into account, thus offering maximum accuracy. Interval simulators (Sniper [CHE11]) evaluate the performance over predefined time intervals, which can speed up simulation while maintaining good accuracy. Full-system simulators (QEMU [Bel05]), reproduce the entire system under study, including not only the NoC but also processors, memories and other components.

Table 2.1: Comparison of NoC simulators [Kha+18].

Simulator	**NoCTweak**	**Noxim**	**Nirgam**	**Nostrum**	**BookSim**	**WormSim**	**NOCMAP**	**ORION**
language	system C	system C	system C	system C	C++	C++	C++	C++
topology	2D mesh	2D mesh	2D mesh, torus	2D mesh, torus	wide range	2D mesh, torus	2D mesh	no
traffic pattern	synthetic, embedded	synthetic	synthetic, embedded	synthetic	synthetic	synthetic, embedded	synthetic, embedded	no
switching mechanism	wormhole virtual channel, Roshaq, bufferless, circuits switched	wormhole with virtual channel	wormhole with virtual channel	wormhole with virtual channel	wormhole with virtual channel	wormhole with virtual channel	wormhole with virtual channel	user design
buffer depth option	yes	yes	yes	yes	yes	yes	yes	yes
routing algorithm	XY, negative first, west first, northlast, OE, lookup table	XY, negative first, west first, northlast, OE, dyad, fully adaptive, lookup table	source routing, XY, OE	XY, deflection routing	all	XY, OE, dyad	XY, OE, west first, dyad	no
performance parameters	power/energy consumption, throughput,latency	energy, throughput, communicationdelay	power, latency, throughput	latency, throughput, linkutilisation	latency, throughput	energy	energy, reliability	router power, link power, router-area
energy model	CMOS standard cell library model	ORION model	ORION model	no	no	ORION/bit energy model	bit energy model	ORION model
input parameters interface	command line	command line	log file	command line	log file	command line	command line	command line
hotspot option	yes	yes	no	no	no	yes	no	no
mapping	NMAP, random	no	manual	manual	no	no	BB, SA	no

Table 2.1 resumes an effective comparison of various NoC simulators. These simulators differ in many aspects, including supported language, topology, traffic model, switching mechanism, buffer management, routing algorithm, performance parameters, energy model, input interfaces, hotspot traffic management, and mapping strategy. For example, some simulators use SystemC or C++ languages, offer synthetic or real workloads, and can support wormhole or virtual channel switching. Buffer support is commonly available, and routing algorithms vary from XY to dyadic, while performance parameters cover energy consumption, latency, and throughput. The energy model can be standard or based on a cell library, and input interfaces range from command-line to configuration files. Some simulate traffic hotspots, and the mapping strategy can be manual or random.

Some of the most popular NoC simulators:

1. Noxim [Cat+16]: Noxim is an open source NoC simulator based on SystemC. It allows users to test various network and traffic configurations, and provides information on throughput, latency, and energy consumption.

2. Booksim [Jia+13]: Booksim is a highly configurable NoC simulator developed by Stanford University. It allows the simulation of various network interconnection models, including irregular topologies.

It's also worth mentioning the existence of full-system simulators, which provide integrated modeling encompassing processor cores, interconnects, memory and other peripheral components, enabling exhaustive analysis of the behavior of the system as a whole. Some of the most popular full system simulators:

1. gem5 [Bin+11]: gem5 is a highly flexible open-source system simulator that offers modeling capabilities at both the architectural level (for simulating processors, caches, and other system components) and the system level (for simulating complete systems, including peripherals and networks).

2. Sniper [CHE11]: Sniper is a high-performance parallel simulator for multiprocessor systems. It offers modelling capabilities for a variety of processor technologies, including homogeneous and heterogeneous multicore systems. Its extensible architecture allows new models or modifications to be added. Although it is not a specific NoC simulator, it can be used to model the performance of high-performance computing systems, which typically include NoC.

[AS16] provides a detailed comparison of full-system simulators, offering an in-depth analysis of their functionality.

2.2.2.2 Analytical model

In this section we will discuss the latency analysis in networks on chips based mainly on queueing theory. This latency is calculated on an accumulative basis and includes the delays for every link in the network that the packet traverses. Each delay on a link is itself made up of four elements: first the processing time, which is the interval between the successful receipt of the packet at the entry node of the link and the assignment of the packet to a queue for transmission. Then the queueing time, which is the time between the packet being allocated to a queue and the start of its transmission. During this interval, the packet is queued, while other packets in the queue are transmitted. Then there is the transmission time, which is the time between transmission of the first bit of the packet and transmission of its last bit. Finally, the propagation time, which is the delay between the transmission of the last bit at the link's source node and its reception at the output node.

The queueing theory is an effective technique to analyze the performance of NoCs. Generally speaking, it is a mathematical study of how waiting lines originate, work, and become congested. This study is usually presented using Kendall's notation [Ken53], represented by three parameters $P_1/P_2/P_3$:

- P_1: is the nature of the arrival process. For example, M stands for the Poisson process, G for general distribution of inter-arrival times, and D for deterministic inter-arrival times. In NoCs, it represents the packets inter-arrival times that depend on the packet injection rate. These time intervals can follow different distributions. For instance, if they are equal, the distribution is deterministic.

- P_2: is the nature of the probability distribution of the service times. G for instance, stands for a general distribution, D for a deterministic distribution, and M for exponential distributions. In NoCs, it is how the packet size varies. If the packet size is fixed, it follows a deterministic distribution, but otherwise packet size can follow such as the General one.

- P_3: is the number of servers. In NoCs, it is how many outputs route a given packet.

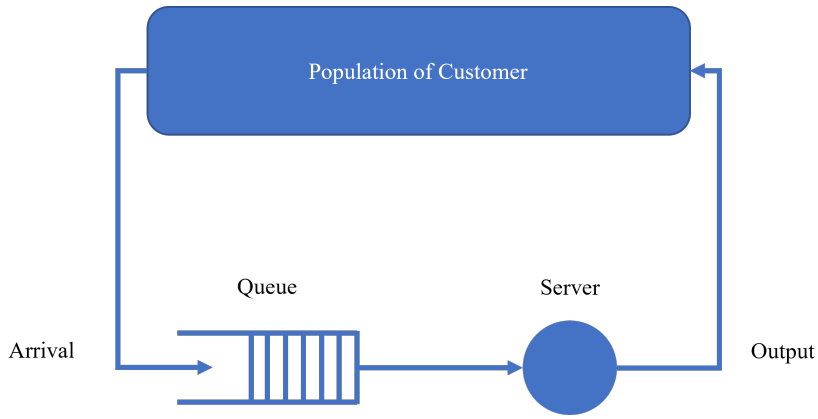


Figure 2.3: Queueing theory.

Different queueing models have been proposed allowing different NoC features. An analytical model was proposed by Ogras et al. [OBM10] for wormhole-switched NoCs. This proposal is based on a $M/G/1$ queueing model with a relative error of around 9% for 1000 random mappings, at 0.16 packets/cycle injection rate. In [Zha+10], Zhang et al. proposed an analytical model for the evaluation of the performance of a NoC with constant service time system, based on a $M/D/1$ queueing model with a mean error of 7.87% using XY deterministic routing and a 5×5, 2D-mesh network, and uniform traffic patterns. Kiasari et al. proposed a flexible $G/G/1$ queueing model for estimating the average packet latency with less than 10% error in NoCs with arbitrary network topology and deterministic routing [KJL13]. More recently, in order to represent packet flow in a NoC as an open feedforward queueing network, Bhaskar and Venkatesh [Bha21] proposed a mathematical model, with 10 and 15% errors for NoCs with 36 and 64 routers, respectively. Weighted Round-Robin arbitration has been also proposed for priority based NoCs in [Man+21b], with less than 5% error while executing real applications and 10% error under challenging synthetic traffic.

These models only address ENoCs, and therefore are not adapted for hybrid NoCs with WiNoCs using token-passing channel access. To the best of our knowledge, our work is the first attempt to propose an analytical model for hybrid interconnects based on both electrical and wireless interconnects.

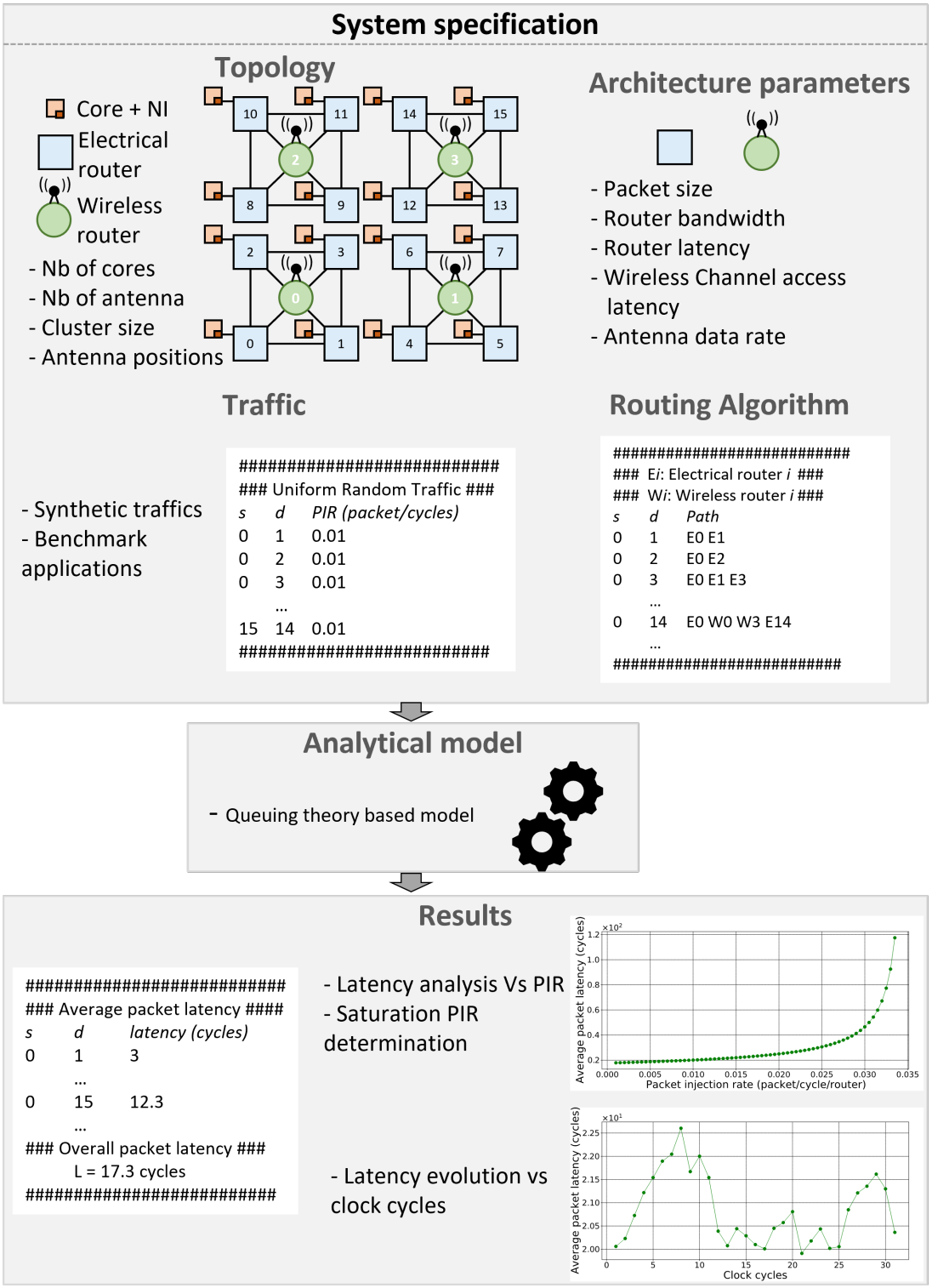


Figure 2.4: Overview of the analysis flow based on AMHNOC.

Table 2.2: Summary comparison of different analytical models.

	[OBM10]	[ZWG13]	[KJL13]	[Bha21]	[Man21]
Queue	M/G/1	M/D/1	G/G/1	G/G/1	General Geometric distribution
Arbitration	Round Robin	Round Robin	Fixed Priority	Round Robin	Weighted Round Robin
Implementation	C++	C++	C++	Matlab	C++
Validation tools	Wormsim	OPENET	-	Booksim	-

2.3 Hybrid interconnection models

In this section, AMHNOC, the proposed analytical model to estimate the average packet latency for a hybrid interconnection containing wired and wireless interconnections is thoroughly illustrated. Starting first by a global overview of the analysis flow presented in Section 2.3.1, followed by the a comprehensive description of our analytical model detailed in Section 2.3.2.

2.3.1 Overview of the analysis flow

The analysis flow to evaluate the communication latency of a hybrid wireless and electric NoC is depicted in Figure 2.4. According to the system specifications, the flow takes as inputs the interconnect architecture, the routing paths from sources to destinations (i.e. routing algorithm), and the traffic between cores. The architecture is described by the number of cores, routers, and antennas presented in the network and the way they are interconnected. The design flow relies on architectural parameters since they impact the communications performance. For example, packet size, router bandwidth, antenna data rate, and wireless channel access latency are considered. The routing algorithm on the other hand, specifies the path selected between any pair of source s and destination d, and with respect to the architecture. Furthermore, the latency of communications within a NoC is affected by the injected traffic. Our design flow takes as inputs synthetic traffic, such as uniform random or transpose communication patterns, or traffic from real application benchmarks, such as the Parsec Benchmark suite [BKL08]. Generally, the purpose of the flow is to evaluate communication latency. Our analytical model, based on the queueing theory, is in charge of this computation and is further detailed in the following Section 2.3.2. This allowed us to obtain the average latency for each couple of source and destination along with the overall latency. From this, it is easy to draw the classic latency versus PIR plot which represents the core metric to evaluate a NoC, see Section 2.4.2. Statistics are also obtained, such as the amount of packets using antennas

that will be further highlighted in Section 2.4.2. It is also possible, based on traces analysis of applications, to plot the communication latency during the execution time; see Section 2.4.5.

2.3.2 Analytical Model

We propose AMHNOC, an analytical model based on the $M/G/1$ queueing model and consider the wormhole routing. The Poisson distribution of packet interarrival M was selected in this study as it represents a simple model to handle compared to other more complicated ones such as General or Geometric distributions. It is good to mention at this point that our second approach was dedicated to develop a method of using Poisson distribution for traces coming from real applications and is thoroughly explained in chapter 3 of this manuscript. Furthermore, the second parameter was selected to be G, a general distribution for service time in order to evaluate packets of different sizes.

The model is generic and compatible with any kind of hybrid interconnectThis model is based on a set of parameters defined in Table 2.3.

In these hybrid NoCs, different types of communication paths exist between a source core s and a destination core d.

The path can be i) only with electrical routers, ii) only with wireless routers, and iii) a mix of both electrical and wireless routers.

To compute the average latency between a source s and a destination d, the proposed model first calculates the latency to cross each routing element. Then, with respect of injected traffic pattern, it calculates the probability of congestion which increases the overall latency. A routing element can be an electrical or a wireless router. An electrical router has the same amount of input and output ports, with respect to number of connected neighbors. A wireless router has one or several electrical inputs and outputs, with respect to the amount of connected neighbors, and one input and output connected to an antenna. In the following, we first introduce the model by illustrating how packets go through a simple one-input one-output routing element, hence without possible congestion, then we describe the complete model which can be applied to any routing elements with any number of inputs and outputs.

2.3.2.1 A simple one-input one-output routing element

The Figure 2.5 presents the model used to evaluate the traffic bandwidth for a service based on a single pair of input and output of a routing element.

Table 2.3: List of the parameters used in our model

Notation	Definition
m, n	The number of input and output ports
N_c	The number of cores
N_a	The number of antennas (equal to the number of wireless routers)
N_f, ρ	The number of flits in a packet with ρ (bits) each flit
a_i, A	a_i denotes the average number of packets at input buffer i, and A the vector of average number of packets for the entire routing element ($packets$)
T, $\overline{T}$, $\overline{T^2}$	T denotes the packet service time, $\overline{T}$ and $\overline{T^2}$ denotes the first and the second order moments of service time respectively ($cycle$)
λ_i, Λ	λ_i denotes the arrival rate at input buffer i, and Λ the arrival rates to the entire routing element ($packet/cycle$)
f_{ij}	Forwarding probability: probability that a packet arrives at input i and leaves the routing element through output j
γ_{ij}	Arrival rate at input i and routed toward the output j ($packet/cycle$)
β	Time needed by the token to pass between consecutive antennas ($cycle$)
σ	The average delay required to access the wireless channel ($cycle$)
q	Waiting time in queue, q^w and q^e for wireless and electrical routers respectively ($cycle$)
r_i, R	r_i denotes the residual service time at input buffer of input i and R the residual time for the entire routing element ($cycle$)
δ_{ij}, Δ	δ_{ij} denotes the contention probability between input i and input j and Δ the contention matrix for the entire routing element
S^e, S^w	Electrical and wireless routers service time ($cycle$)
B	Electrical router bandwidth (bit/s)
D	Antenna data rate (bit/s)
θ	Frequency ($Hz => cycles/s$)
x_{sd}	Rate of the traffic transmitted from source s to destination d ($packets/cycle$)
L_{sd}	Average latency for any packet from source s to destination d ($cycle$)
L	Overall average packet latency ($cycle$)
$PIR, SPIR$	PIR denotes the packet injection rate and SPIR denotes the saturation packet injection rate ($packet/cycle/router$)
Th	Network throughput ($packet/cycle/router$)

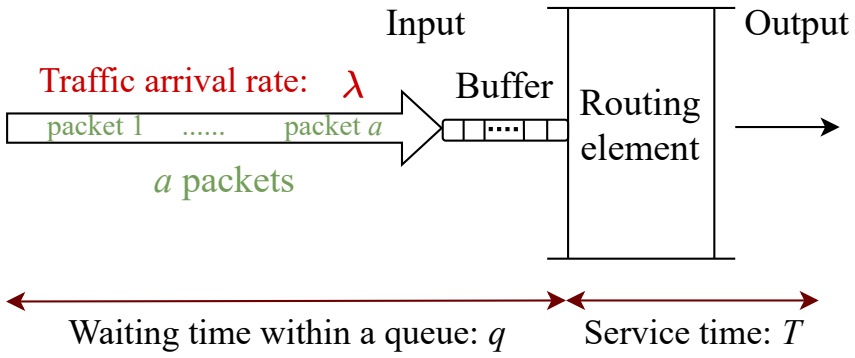


Figure 2.5: Example of a single queue routing element.

In this case, using the *Little's Theorem*, the average number of packets a in the system is computed from the waiting time q within a queue and the traffic arrival rate λ of packets into the system:

$$a = \lambda q \tag{2.4}$$

On the other hand, we can calculate the waiting time q of each packet by the *Pollaczek-Khinchin* (P-K) formula depending on the queueing model. For the $M/G/1$ model, the waiting time q is given by:

$$q = \frac{r}{(1 - \lambda T)} \tag{2.5}$$

where r is the *residual service time*, the time a new arriving packet needs to wait until the service time of already present packet is complete:

$$r = (\lambda \overline{T^2})/2 \tag{2.6}$$

T, $\overline{T}$ and $\overline{T^2}$ are respectively *the packet service time*, the first and the second order moments of service time without the queueing delay respectively. T is given by:

$$T = \begin{cases} S^e + \frac{N_f \times \rho \times \theta}{B} & \textit{Electrical router} \\ S^w + \frac{N_f \times \rho \times \theta}{D} + \sigma & \textit{Wireless router} \end{cases} \tag{2.7}$$

where S^e, S^w are electrical and wireless router service times respectively, N_f is the number of flits in the packets, ρ is the size of flits, B and D the electrical router bandwidth and the antenna data rate, and σ is the *Average delay token*. We assume a round-robin token circulation among N_a antennas, where σ is the average delay corresponding to the number of cycles required to access the wireless channel. In the best case, an antenna requiring

to send a packet already posses the token and will directly send it without any delay. On the other hand, the worse case is to wait a full round to receive the token. If the time needed by the token to pass between consecutive antennas is β *cycles*, the average delay σ can be stated as:

$$\sigma = \beta \times N_a/2 \tag{2.8}$$

2.3.2.2 General case

In a NoC, there are m inputs and n outputs for a routing element as shown in Figure 2.6. Hence, the equations 2.5 and 2.6 have to be computed as matrices. For instance, for a

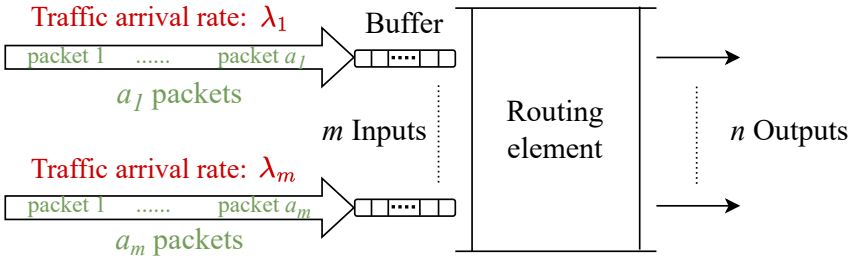


Figure 2.6: General case of service managing m inputs and n outputs.

routing element y, the average number of packets, the arrival rates and the residual service times are respectively defined by:

$$A^y = \begin{bmatrix} a_1^y & a_2^y & \dots & a_m^y \end{bmatrix}^T \tag{2.9}$$

$$\Lambda^y = diag(\lambda_1^y, \lambda_2^y, ..., \lambda_m^y) \tag{2.10}$$

$$R^y = \begin{bmatrix} r_1^y & r_2^y \dots r_m^y \end{bmatrix}^T \tag{2.11}$$

where the average number of packets for a routing element y is done by the equilibrium condition [OBM10]:

$$A^y = (I - \Lambda^y \Delta^y)^{-1} \Lambda R^y \tag{2.12}$$

Contention probability: Δ is the contention matrix, where δ_{ij} is contention probability between input i and input j:

$$\Delta^y = \begin{bmatrix} 1 & \delta^y_{12} & ... & \delta^y_{1m} \\ \delta^y_{21} & 1 & ... & \delta^y_{2m} \\ ... & ... & \ddots & ... \\ \delta^y_{m1} & \delta^y_{m2} & ... & 1 \end{bmatrix}_{m \times m}$$

with

$$\delta^y_{ij} = \begin{cases} \sum_{k=1}^{m} f^y_{ik} f^y_{jk} & i \neq j \\ 1 & i = j \end{cases} \tag{2.13}$$

Where f_{ij} is the forwarding probability, the probability that a packet arrives at input i and leaves the routing element through output j:

$$F^y = \begin{bmatrix} 0 & f^y_{12} & ... & f^y_{1m} \\ f^y_{21} & 0 & ... & f^y_{2m} \\ ... & ... & \ddots & ... \\ f^y_{m1} & f^y_{m2} & ... & 0 \end{bmatrix}_{m \times m}$$

$$f^y_{ij} = \begin{cases} \frac{\gamma^y_{ij}}{\sum_{k=1}^{m} \gamma^y_{ik}} & i \neq j \\ 0 & i = j \end{cases} \tag{2.14}$$

γ^y_{ij} is the arrival rate at the input i and is routed toward the output j. The traffic arrival rate at input j at the routing element y:

$$\lambda^y_j = \sum_{\forall s} \sum_{\forall d} x_{sd} \Re(s, d, y, j) \tag{2.15}$$

Where $\Re$ is the routing function:

$$\Re(s, d, y, j) = \begin{cases} 1 & \textit{if } (y, j) \textit{ in } \Pi_{sd} \\ 0 & \textit{otherwise} \end{cases} \tag{2.16}$$

Where Π_{sd} is the set of routers pairs traversed by the packet. Then, the average packet latency is done by:

$$L_{sd} = L_e + L_w \tag{2.17}$$

Where L_e and L_w are given by:

$$L_e = \begin{cases} 0 & \textit{if path fully wireless} \\ \sum_{(i,j) \in \pi_{sd}} (q^e_{ij} + S^e) + \frac{N_f \times \rho \times \theta}{B} & \textit{otherwise} \end{cases} \tag{2.18}$$

$$L_w = \begin{cases} 0 & \textit{if path fully electric} \\ q^w_{src} + q^w_{dst} + 2S^w + \frac{N_f \times \rho \times \theta}{D} + \sigma & \textit{otherwise} \end{cases} \tag{2.19}$$

Finally the overall average packet latency L is given by:

$$L = \frac{\sum_{\forall s,d}(x_{sd} \times L_{sd})}{\sum_{\forall s,d} x_{sd}} \tag{2.20}$$

2.4 Experimental Evaluations

In Section 2.4.1, we present the experimental setup considered for our study. Model validation is performed for synthetic traffic and application benchmarks in Sections 2.4.2 and 2.4.5, respectively. The network throughput validation is carried out in Section 2.4.3. Additionally, we analyze the execution time and evaluation error in comparison to the Noxim cycle-accurate simulator, and the results are presented in Sections 2.4.4 and 2.4.6, respectively.

2.4.1 Experimental Setup

To validate our model, we compare our results with the Noxim cycle-accurate simulator [Cat+16]. Noxim incorporates a hybrid NoC that combines electrical and wireless routers. In order to limit methodological bias, and ensure repeatability of results, all the results are obtained by respecting the Noxim hybrid topology which corresponds to Figure 2.7.

Furthermore, the primary objective of this work is not to determine the optimal topology for hybrid on-chip interconnect or to compare ENoCs and WiNoCs, but to propose a method that enables rapid performance analysis and exploration of such architectures. For comparative studies, please refer to [Dai+15; BJS19].

The N_c core architecture is divided in clusters of equal size, in accordance with the number of implemented antennas N_a. Each cluster is organized in a 2-D mesh ENoC with a wireless router shared among all cores within the same cluster. The wireless router is then connected to each electrical router from the cluster, and can be accessed by a core after crossing the associated electrical router. Intra-cluster communications are handled by ENoC using the XY routing algorithm, while inter-cluster communications are achieved by using WiNoC with a token-passing channel access scheme. For instance, in

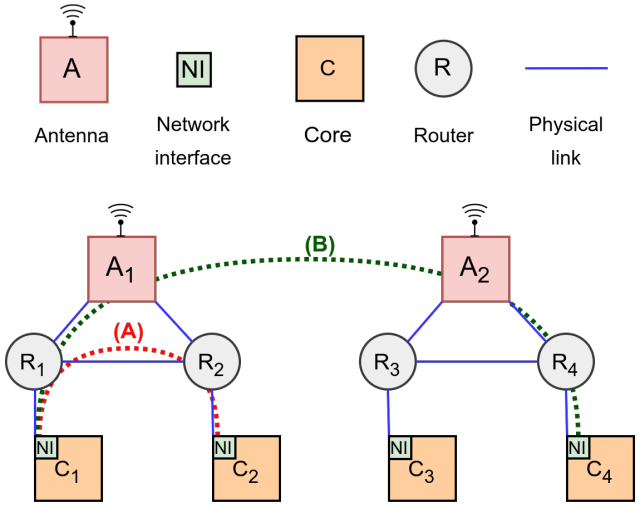


Figure 2.7: Overview of a cluster-based hybrid NoC. Intra-cluster communication are done through ENoC (A), while inter-cluster communications are handled with wireless communications (B).

a configuration with $N_c = 16$ cores and $N_a = 4$ antennas, the architecture is decomposed into 4 clusters that are not electrically interconnected. This architecture may be likened as a chiplet-based one where inter- and intra-chiplet communications are using a different interconnect.

We consider routers without virtual channels and a router bandwidth B set to 32 Gbit/s (32-bit flits at 1GHz). Service times S^e and S^w are set to 2 clock cycles, while the token passing latency β is set to 1 clock cycle. The simulations were conducted on an Ubuntu 18.04.5 LTS Linux distribution, executed on a 48-cores Intel© Xeon(R) Silver 4214 CPU @ 2.20GHz. Furthermore, the proposed analytical model is implemented using Python, and the results are compared with those obtained from the Noxim cycle-accurate simulator NoC [Cat+16].

It should be noted that any analytical method is deterministic in nature, which guarantees consistency in the obtained results without requiring multiple iterations. Simulation-based approaches, on the other hand, can exhibit a degree of variability. Nevertheless, to guarantee correctness on the analysis, we defined the simulation time that allows latency convergence toward a stable value. This value depends on the PIR value. Indeed, a high PIR value requires more clock cycles to be simulated than a lower PIR.

2.4.2 Parameters exploration with synthetic traffics

In this section, we explore the correctness of our model by exploring several parameters. We vary the traffic pattern, packet size, wireless datarate, and number of antennas. Each

parameter value is summarized within the figures. The purpose is to provide a model to efficiently determine the latency of the hybrid NoCs, not to identify the optimal architecture. The results are presented in a graph plotting the average packet latency versus the PIR (packet per core per cycle). For each result, we vary the PIR starting from 0.001 in increments of 0.001, until crossing the SPIR (defined as 10× the zero-load latency). Moreover, we simulate 100,000 clock cycles into Noxim for each PIR value, while our analytical model only requires a single computation as it only depends on the PIR.

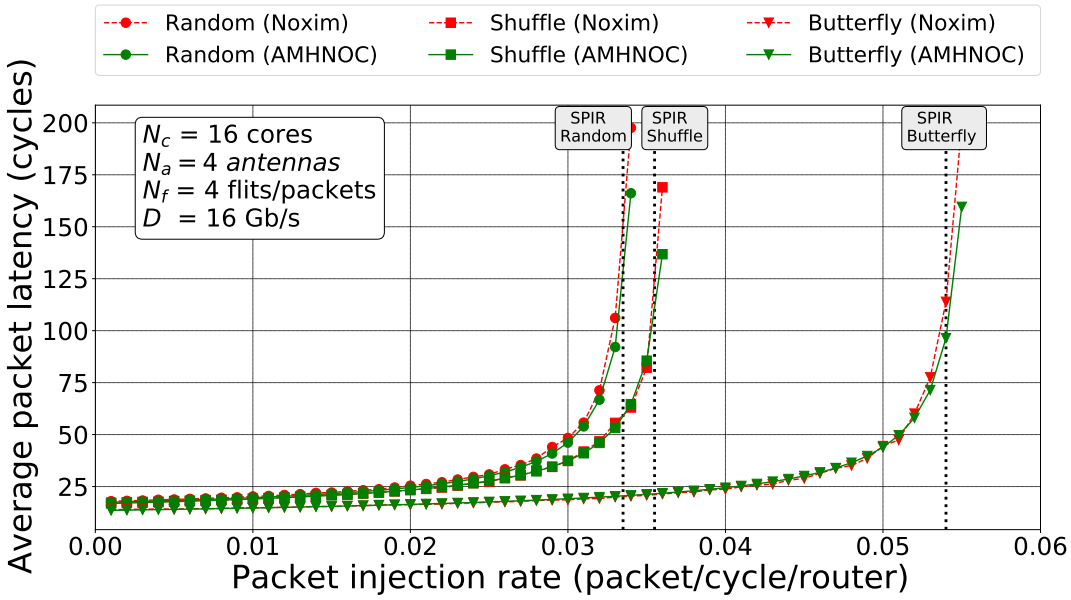
2.4.2.1 Traffic patterns exploration

We start by exploring different traffic patterns: random, shuffle and butterfly with a 16 cores architecture size, forming 4 clusters with 4 cores and one antenna per cluster, 4 flits per packet, and with an antenna datarate 16 Gb/s.

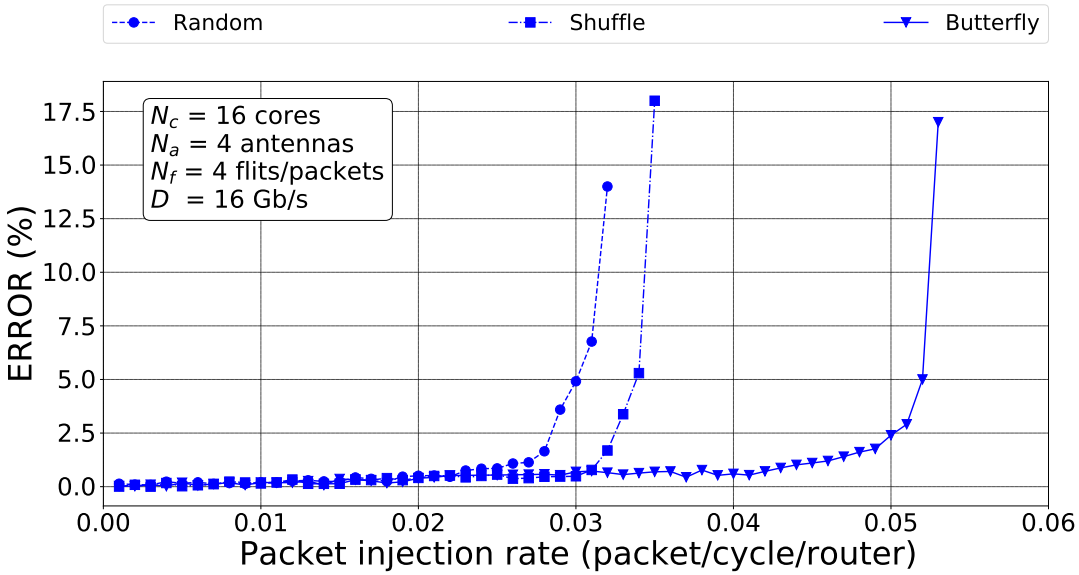
The same NoC behaviour is observed from the results obtained using both the analytical model and the simulator (Figure 2.8a). At low PIR, the average latency is low (around 17 clock cycles). The NoC starts reaching the saturation when the PIR increases. By performing these analysis on the three different traffic patterns, it was observed that the NoC saturates faster for the uniform random traffic pattern than shuffle than butterfly, thus creating more congestion.

To better understand the alignment between the analytical model and the simulator's outcomes, figure 2.8b was plotted illustrating the latency difference (error in %) for each PIR of Figure 2.8a. From this plot, it is clearly noticed that the error remains below 2% on the zero load part, and starts to gradually increase until saturation while remaining acceptable (7%). Beyond saturation (refer to section 2.2.1 for saturation determination), the error continues to increase reaching towards 17%. It is important to note that the increase in error when the network becomes highly congested is common to all analytical models [OBM10; Zha+10; KLJ13; Bha21; Man+21b]. This is the trade-off for extremely short computation times compared to very long simulation times (see Section 2.4.4).

Furthermore, the proposed model computes the routing latency for hybrid electrical-wireless paths as well as fully electrical paths. For instance, on the results of Figure 2.8a with the execution of traffic random, 76% of the packets used a hybrid path, while 24 % used a fully electrical path. Compared to Noxim cycle-accurate simulator, the latency evaluation accuracy for the two types of paths is satisfactory. If we focus on determining the SPIR value, these values are, respectively for the analytical model and Noxim: 0.0541 and 0.0514 for a butterfly traffic, 0.0346 and 0.0331 for a shuffle traffic, 0.0325 and 0.0317



(a) Traffic patterns exploration



(b) Analytical model Vs Noxim error

Figure 2.8: Traffic patterns exploration (a) and analytical model Vs Noxim error (b).

for a uniform random traffic. On average the difference is 4.3% in determining the SPIR.

2.4.2.2 Architecture size exploration

Afterwards we explored different sizes of architecture. By exploring the size of the architecture, the analytical model can be tested in a variety of configurations, ensuring complete and accurate validation of its ability to predict latency. NoCs of 36, 64 and 100 cores architecture size were tested, with 4 clusters (9, 16 and 25 cores per cluster respectively) and one antenna per cluster, 4 flits per packet, and with an antenna datarate of 16 Gb/s. The following results are performed with a uniform traffic pattern.

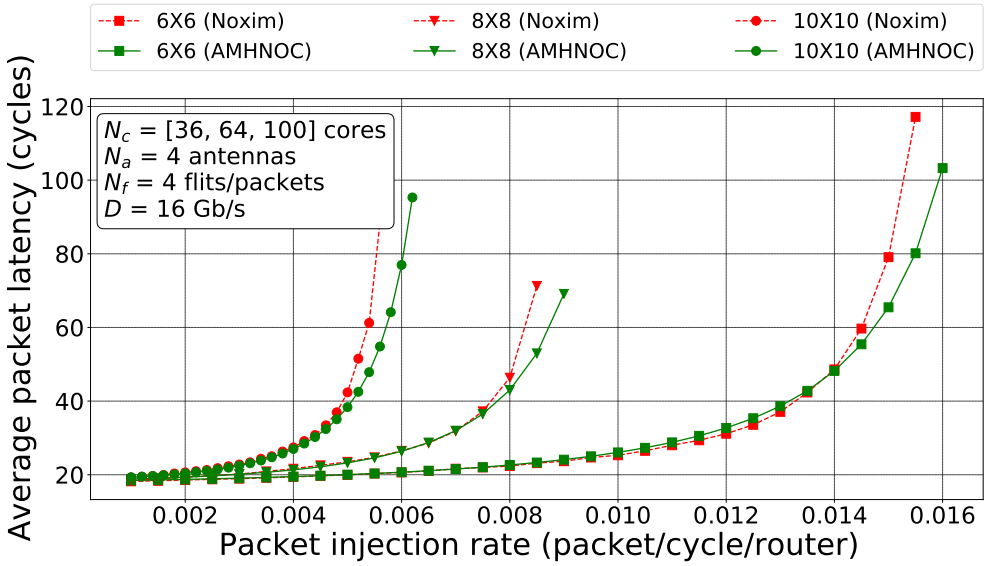


Figure 2.9: Architecture size exploration.

A similar NoCs behavior was also observed in this case for both the analytical model and the simulator and are illustrated in Figure 2.9 highlighting the impact of architecture sizes. It is observed that the 100 cores architecture get congested at a lower PIR compared to the 64 and 36 cores architectures. For instance, the SPIR decrease from 0.00159 to 0.0087 to 0.006 packet/cycle/router, for NoCs with 100, 64 and 36 cores, respectively. This can be attributed to the fact that more cores are sharing a wireless router, which increases the congestion in using wireless communication as it is the only way to communicate between clusters in this type of topology. In addition, we find that the error error of results obtained by analytical model and Noxim remains below 2% until saturation, where we find a maximum error of 16%.

2.4.2.3 Packet size exploration

Exploring packet size is also considered as a fundamental step in validating our proposed analytical model. In that respect we explored three different cases with 8, 16 and 32 flits per packet, for 16 cores architecture size, with 4 clusters (4 cores per cluster) and one antenna per cluster, and with an antenna datarate of 16 Gb/s. This exploration tests the model's ability to adapt to a variety of traffic conditions, which is essential to guarantee its reliability in different usage scenarios.

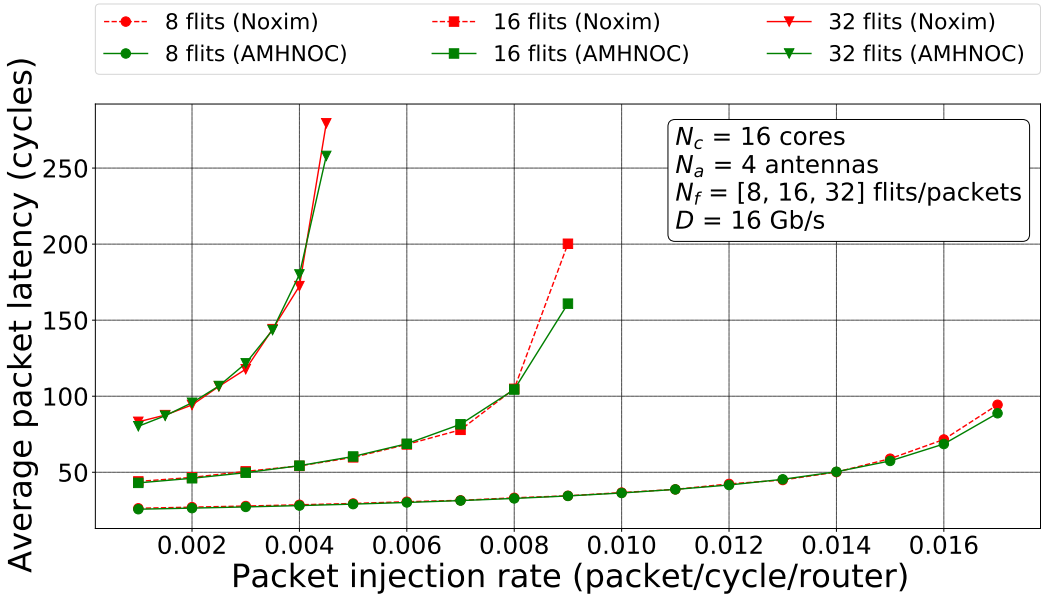


Figure 2.10: Packet size exploration.

Similarly, changing the size of the packet resulted in the same NoC behavior between the simulator and the analytical model as illustrated in Figure 2.10. It is not surprising that increasing the number of flits per packet (8, 16 and 32 flits) leads to higher latency (21, 40 and 85 cycles, respectively at zero load latency) and a reduction in the saturation PIR value (0.0045, 0.009 and 0.017 packet/cycle/router, respectively). This is due to the increased congestion accompanied with larger packets.

Likewise the error remains below 2% until saturation, where we find a maximum error of 13%, which means that the model is always accurate with different packet sizes.

2.4.2.4 Wireless datarate exploration

After that, different Wireless datarates were explored. This is also crucial for validating our analytical model, especially in the contexts where our model is proposed for hybrid communication. Fro that, we explore datarates of 8, 16 and 32 Gb/s for a 16 cores architecture size, with 4 clusters (4 cores per cluster) and one antenna per cluster, and with a packet size of 4 flits.

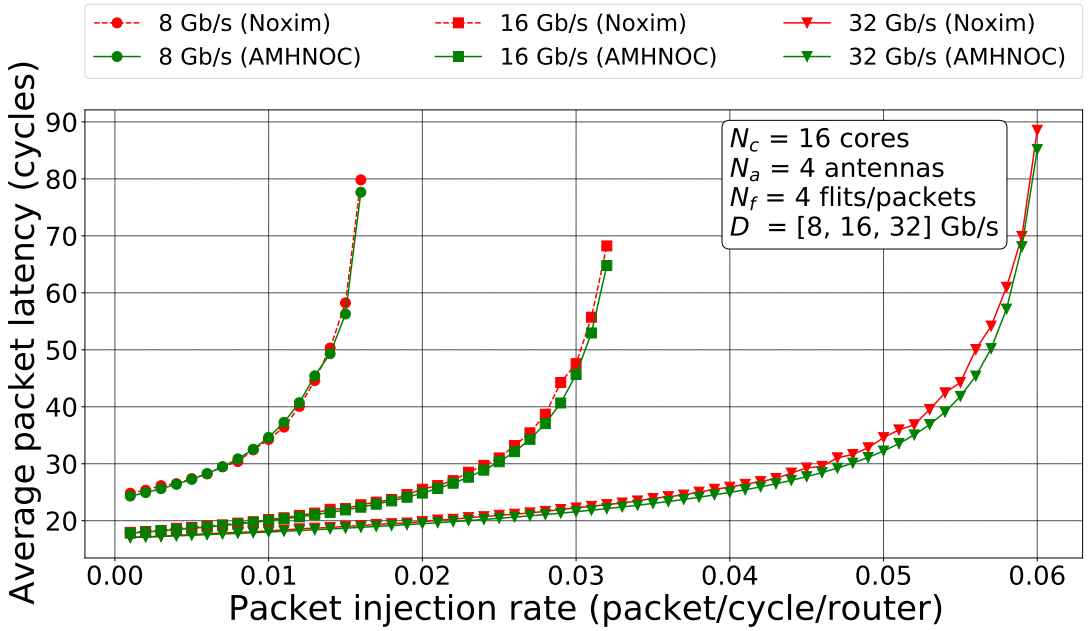


Figure 2.11: Wireless datarate exploration.

Contrarily, increasing the antenna datarate allows for an increase in the saturation PIR, as shown in Figure 2.11. For instance the SPIR increase from 0.016, to 0.032 to 0.06 packet/cycle/router, in corresponding to the increase of wireless datarate form 8 to 16 to 32 Gb/s, respectively. Likewise, the error remains below 2% until saturation, where a maximum error of 10% was then found. This assures that the model is also always accurate with different wireless datarates.

2.4.2.5 Exploration of number of antennas

Then we explored different number of antennas in the NoC. Such exploration is also essential in the validation of our analytical model especially as it is proposed for hybrid communications. Again in this case we explored three different cases with 4, 8 and 16 antennas, each with a 64 cores architecture size, 4 clusters (4, 8 and 16 cores per cluster), a packet size of 4 flits, and with 16 Gb/s wireless datarate.

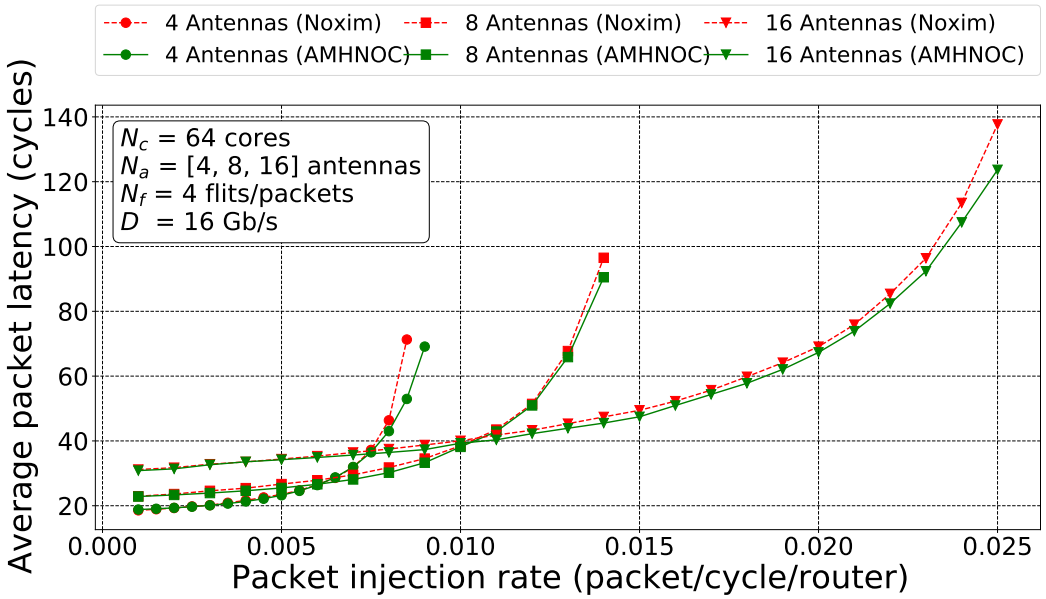


Figure 2.12: Exploration of number of antennas.

Figure 2.12 shows the impact on the number of antennas within a 64 cores architecture. Surprisingly, increasing the amount of antenna increases the latency at low PIR. This is as a result of the token passing channel access. Indeed, with 16 antennas and 64 cores (i.e. cluster of 4 cores with 1 antenna), each communication needs to get the token before being able to communicate, then the latency is around 35 clock cycles. However, the NoC saturates with a higher PIR as the number of hops between any source and destination is kept low thanks to a wireless communication.

Furthermore, it is worth-noting that, for different traffic patterns, by increasing the number of antennas, the percentage of the utilization of wireless interconnects also increases as represented in Figure 2.13. For instance, for random traffic with a 64 cores architecture size and 4 antennas, 78% of packets communicate via wireless interconnects, while 22% with electrical ones. However, with 8 antennas, the percentage of packets communicating via wireless interconnects increased to 89% with 11% left communicating with electrical ones.

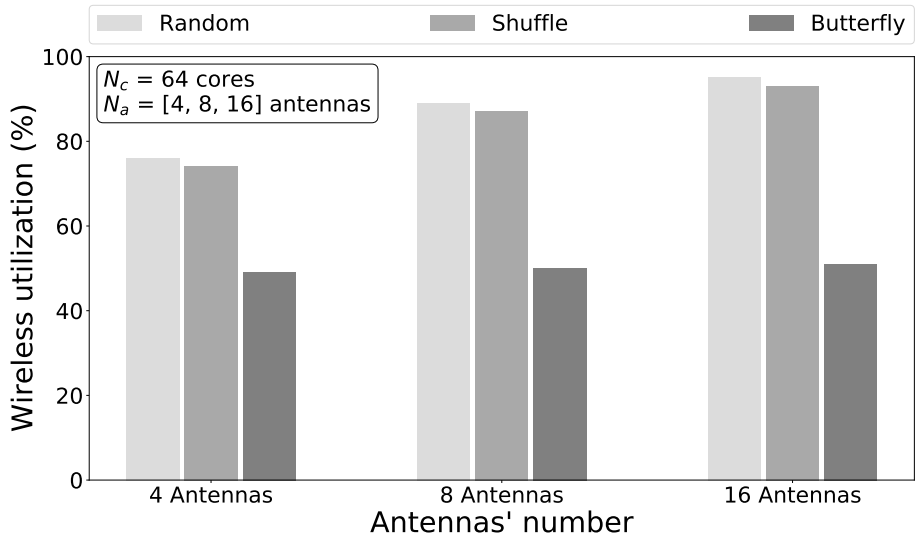


Figure 2.13: Wireless utilization rate.

2.4.3 Network throughput

It is straightforward to evaluate the throughput Th (number of packets per cycle) of an architecture with the proposed analytical model, as it may be directly derived from the latency evaluation. We proceed as follows: i) compute the latency Vs PIR, ii) determine SPIR (10× the zero-load latency for a given traffic), iii) we apply the Equation 2.2. In the Noxim simulator, the throughput is directly given in the output simulation results.

Figure 2.14 shows the network throughput Vs PIR of a 16 cores architecture forming 4 clusters with 4 cores and one antenna per cluster, 4 flits per packet, and with an antenna datarate 16 Gb/s for each of the traffic patterns, random, shuffle and butterfly. From this plot, it is clearly observed that before saturation (detailed in Section 2.4.2.1, the network throughput is proportional to the PIR, then becomes constant after reaching the SPIR. This validates our approach to compute both SPIR and throughput.

2.4.4 Execution time

Figure 2.15 shows the execution times of our analytical model and Noxim versus the number of cores in the architecture. It is evident that the speed up increases with respect to the architecture size. For instance, for a 16×16 architecture size, our analytical model requires only 1.22 seconds versus 603 seconds for a Noxim simulation, leading around

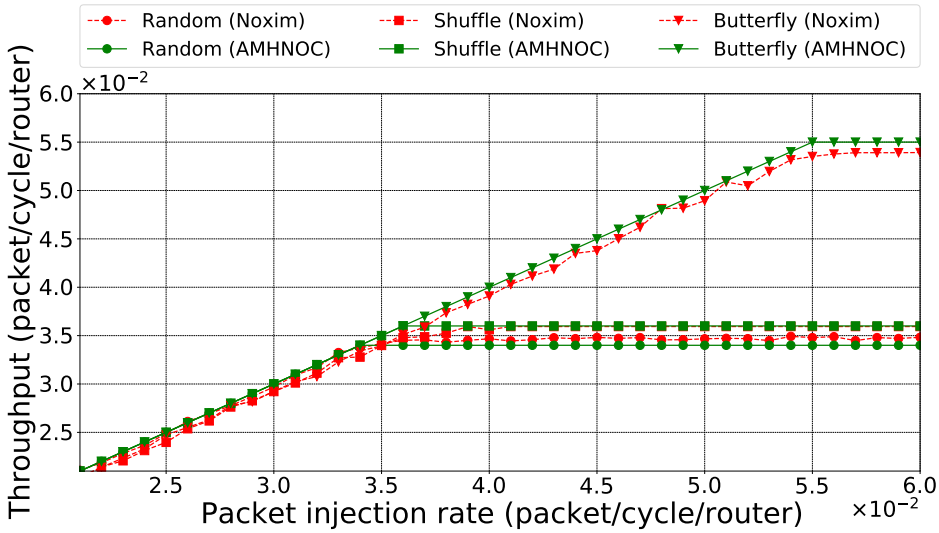


Figure 2.14: Throughput Vs. PIR

500× speed up. This validates the benefits of using an analytical model instead of a simulator to explore a plethora of parameters and architecture topologies. Indeed, 603 seconds with Noxim is only for one simulation. For instance with a 16×16 architecture, if we want to explore the following design space:

- $N_a = [4, 8, 16]$
- $D = [8, 16, 32]$ Gb/s
- $N_f = [8, 16, 32]$ flits/packets
- Traffic = [Random, Shuffle, Butterfly]

The number of simulations required is $N_a \times D \times N_f \times$ number of traffics = 81 simulations. With Noxim simulations, 13 hours are required, while our analytical model only requires 1 minute and 37 seconds. Obviously, both Noxim and our analytical model can be executed with multiple instances (i.e. in parallel) for larger design spaces. For instance evaluating the impact of antenna positions in a 16×16 architecture with 4 antennas requires $C\binom{4}{256}$ $= 174.8 \times 10^6$ simulations. With 10 instances running in parallel, Noxim would require more than 5 years of simulation time, while our analytical model only take 4 days.

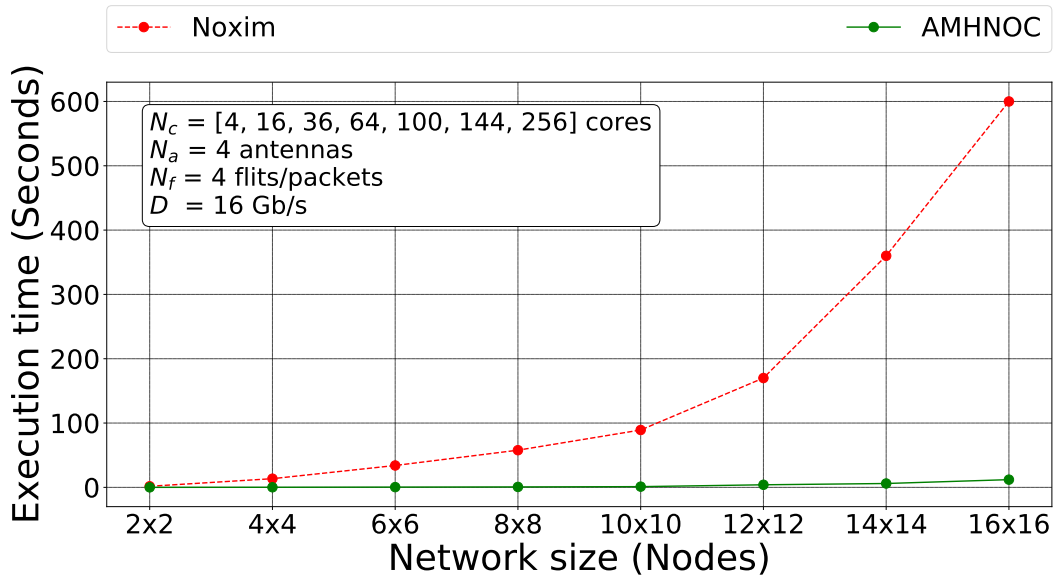


Figure 2.15: Execution time evaluation.

2.4.5 Benchmark applications

To validate the accuracy of our model with heterogeneous traffic patterns, we adopt the PARSEC [BKL08] benchmark suite, which is commonly used to study the performance of NoCs. We have chosen five representative applications, based on their parallelization level, working set, and data usage. We simulate architectures with 16, 32 and 64 cores, along with 4 antennas (i.e cluster of 4, 8 and 16 cores respectively). The memory organization follows a distributed shared memory model, and the cache coherency protocol is MESI. Other parameters are set to default values.

These PIR values were then used as inputs to the analytical model, which applies queueing theory to simulate contention. Hence, packets are injected following a Poisson distribution, as we use a M/G/1 queueing theory model.

Figure 2.16 shows the average latency versus the execution time of the dedup application. Each point plotted represents the average latency of the hybrid NoC within the time window. It shows the evolution of latency during execution with respect to application needs. It also shows the application speed-up when additional cores are utilized, at the cost of increased communications, leading to higher latency. Indeed, the more cores used, the faster the execution becomes, thanks to increased computing parallelism. However, more communications occur in a shorter time, resulting in increased congestion.

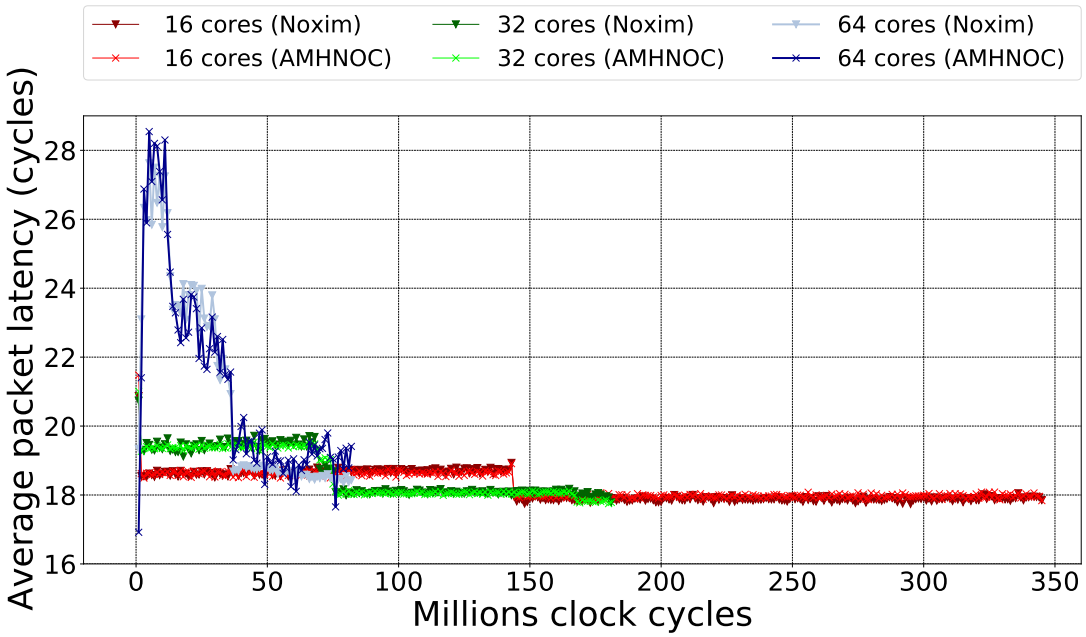


Figure 2.16: Validation of the analytical model using the dedup benchmark application

This is clearly visible with both 64 cores curves, which finish at 80 millions clock cycles versus almost 350 millions clock cycles for a 16-core architecture. In terms of communication latency, there is a latency peak around 28*cycles* for 64 cores, while kept around 19*cycles* for 16 cores. Both results from Noxim and our model exhibit similar behaviours validating the efficiency while considering heterogeneous traffic patterns [BKL08], which is particularly important for non-homogeneous system-on-chip such as accelerator-based architectures.

Finally, Table 2.4 summarizes the average latency percentage error of our analytical model compared to Noxim for the different applications and architecture sizes. We can notice that the error of our analytical model is less than 5.5%, with an average of 3.87%.

It has to be noticed that using application traces for on-chip interconnect performance analysis is a classic technique. It offers faster exploration than full system simulation. As limitation of this technique, increasing or reducing latency of communications will not modify the system level behavior, i.e. will not impact the execution time of the application as only traces are injected. On the other hand, it provides i) a quick coarse-grained analysis to explore interconnect parameters, and ii) to know whether the interconnect is congested or not regarding heterogeneous traffic, hence whether the application will be slowed down or accelerated. This analysis is useful to verify whether hybrid NoC satisfies

the communication needs of real applications or suffers from congestion.

Table 2.4: Average latency error between the analytical model compared to Noxim for different benchmark applications.

Applications	Average latency error (%)		
	16 cores	32 cores	64 cores
blackscholes	2.24	4.56	1.9
dedup	3.75	3.67	1.99
raytrace	4.02	3.17	3.93
streamcluster	4.56	5.5	2.4
*x*264	2.8	3.95	2.55

2.4.6 Relative Error

Analytical models are based on queueing theory which mathematically computes the packet latency. It determines the behavior of queues (here, electrical and wireless routers) and predicts the time it takes for packets to be processed and transmitted while considering congestion. In analytical models, packets are considered to define NoC traffic, and this traffic is modeled by an injection rate. On the other hand, the cycle-accurate simulator will simulate the behavior of NoC at each clock cycle while considering each packet injected into the network. Most of the error appears when the network becomes highly congested (see Section 2.4.2). In fact, the increasing latency due to congestion in each router will modify the arrival time in the following routers along the path, which is more complex to accurately predict in analytical models. This error is common to all analytical models, and this is the trade-off for shorter computation times compared to simulators (see Section 2.4.4).

During the extensive experimentation conducted in Section 2.4, we performed more than 1,000 simulations with the different parameters highlighted. The relative error between the analytical model and Noxim simulator was computed using the equation 2.21 and was found to be around 4%.

$$E = \frac{1}{N_s} \times \sum_{i=0}^{N_s} \frac{|L_A[i] - L_S[i]|}{L_S[i]} \tag{2.21}$$

Where N_s is the number of simulations ($N_s = 1000$), $L_A[i]$ is the latency computed by our model for simulation i, and $L_S[i]$ is the latency computed by Noxim for the same simulation i.

2.5 Conclusion

NoC performance analysis is an important and evolving field of study, which seeks to address the challenges posed by the increasing complexity of SoC. In this chapter, we have examined two major approaches to analyze the performance of NoCs, namely simulators and analytical models based on queueing theory.

Simulators play an indisputable role in the evaluation of NoC performance, providing a high degree of accuracy and fidelity in modeling real NoC behavior. They allow complex interactions between components to be reproduced in detail and provide empirical information on performance metrics such as latency, throughput, and energy consumption. However, analytical models, particularly those based on queueing theory, provide a complementary perspective. These models enable faster and more computationally efficient analysis, offer a deeper theoretical understanding of the factors that influence performance, and are often more flexible and scalable.

It is important to recognize that each approach has its own strengths and weaknesses, and the choice between them depends on the specific objectives of the analysis. In many cases, the joint use of simulators and analytical models may be the most effective strategy, as it allows the accuracy of the simulations to be combined with the theoretical insights provided by the analytical models. Ultimately, as technology continues to evolve and SoC become increasingly complex and integrated, the importance of state-of-the-art tools for analysing NoC performance cannot be underestimated. Future progress in this area will require continued efforts to develop more sophisticated, accurate, and efficient analysis methods, which will be essential to support innovation and optimize performance in the next generation of SoC.

This chapter proposes an analytical model to evaluate communication latency and network throughput of manycore architectures using a hybrid NoC based on both ENoCs and WiNoCs. The proposed model is validated with Noxim simulation experiments. We validated the correctness with several parameter explorations on synthetic traffics. The packet latency error of our model is less than 5% compared to the cycle-accurate Noxim simulator with Parsec application traffics. Compared to Noxim, our model requires up to 500 times less time to perform the results on a 16×16 architecture, hence enabling design space exploration of such complex multi-parameter architectures.

Finally, it is important to mention that this contribution has been published in IEEE Design & Test journal [Kra+23], and was presented at the international NOCS conferences

in September 2023.

Chapter 3

Windowing of application traces for fast NoC performance analysis

3.1 Introduction

The evaluation of the NoC performance in the early stages of architecture design, named design space exploration, is a key step for SoC designers. Full-system simulations, which consider the whole architecture, i.e. cores, memories, interconnects, and application execution, is the most accurate way to evaluate the interconnect performance and the impact of this latter on the application execution. Additionally, it is the only possible way to measure application speed-up. However, such simulation is very time-consuming. Firstly, the complexity to modify such simulator is an obstacle for designers to implement emerging on-chip interconnects such as, for example, in the well-known Gem5 full-system simulator [Bin+11]. Secondly, the time required for simulation dramatically limits the design space exploration. For instance, to simulate the execution of a 64-core architecture running the x264 application with a 13-second video from the PARSEC benchmark suite [BKL08], it requires around 5 days of simulation with a Dual Intel Xeon 4214 processor (2x12 cores at 3.5GHz) and 64GB of memory (2.4GHz), hence clearly limiting the exploration if we target several parameters evaluation.

On the other hand, analytical models and NoC simulator provide a good trade-off to evaluate NoC performance. However, they are not capable of evaluating the application speed-up as only communication traces are considered and applications are not executed along the NoC. Nevertheless, their uses are favored, and designers have considered for years generic traffic patterns to evaluate the behavior of NoCs, such as random or transpose traffics. In addition, the use of more realistic traffics which mimic real-life applications, is also considered. To this end, a first execution of an application is usually performed on a full-system simulator and all communication traces are saved in a data-base. Then, these traces can be injected into a NoC performance analysis tool. The

injected traces can be either i) packet by packet at a specific timestamp, or ii) statistic injection traffic based on the PIR. The first possibility may be time-consuming with respect to the number of packets to inject from the application execution traces. On the other hand, for the second possibility, the computation of the PIR may directly impact the fidelity of the trace usage.

The injection of the traces into a NoC performance analysis tool is widely studied in the literature. These application traffic traces were used either to validate their proposed work, as it is also the case in our study, or to compare and characterize these benchmark applications [BKL08; BFM09]. For example, in [AT23; CL20], there is one simulation for the whole application duration, which smooths out variations and clearly lacks accuracy to evaluate interconnect performance. On the other hand, other researchers, such as [Man+19; Man+21a], split the traces into multiple fixed-size windows to illustrate the evolution of latency as a function of clock cycles. Nevertheless, these studies were performed by choosing an arbitrary size of window without any analysis of its impact on the performance evaluation accuracy. Our approach on the contrary, aims at answering this problem by precisely determining how to correctly split the traces and study the influence of window sizes, while using the Poisson distribution.

This chapter aims at addressing these gaps in the existing literature by proposing a comprehensive framework to analyze application traces and automate their usage for NoC performance evaluation, either into a NoC simulator or for analytical models. The motivation behind incorporating windows into the analysis is presented, highlighting their importance in capturing the dynamic behavior of latency over time. The framework not only advocates for the use of windows but also provides insight into optimal window sizes and emphasizes the advantages of considering window usage effects. The proposed trace analysis framework provides a detailed perspective on the evaluation of benchmark applications. By delving into the details of window usage and its implications on the obtained results, researchers can improve the robustness of their studies and construct more accurate comparisons between different NoC architectures. This chapter thus contributes to the broader understanding of NoC performance evaluation and encourages a more comprehensive approach to benchmark application analysis in the field of computer architectures.

As a fundamental step, the distribution of inter-arrival time for various traces in parsec benchmark applications was first studied and is detailed in Section 3.3. These traces were studied using different techniques such as Q-Q plot and Square Coefficient of Variation

(SCV) calculations and were performed to check whether the distribution tracked in the benchmark application gathers Poisson distributions or any other specific distribution.

With the aim of establishing a better way to simulate traces in a faster and more efficient manner, we proposed a new framework, Section 3.4, that focuses on the precise determination of the optimal size of windows to be applied. After that, the validation of this proposed framework was performed by running different simulations, which are fully illustrated in Section 3.5.

3.2 Benchmark Application Traffic Latency Analysis

As previously mentioned, various analysis methods are available and used in the pursuit of analyzing application performance. With the aim of validating our proposed analytical model, as illustrated in Chapter 2 of this manuscript, we characterized the applications by adhering to Poisson distribution principles. This involved calculating injection rates by averaging the number of packets at each given time t.

However, while the study overall validates our proposed analytical model based on the use of Poisson's law, a critical question remains: to what extent does the practical behavior of real-life applications align with the assumptions of Poisson's distribution? To answer this question, the potential errors introduced when applying the Poisson distribution for application performance analysis must first be quantified. In other words, we aim at investigating whether the applications under consideration truly follow Poisson's distribution. In the case of any deviation, we seek to quantify the extent of error employed when applying Poisson's distribution as a model for analysis. By addressing this question, we hope to provide a comprehensive evaluation of the analytical method's reliability and highlight any differences that can occur between the assumed distribution and the actual characteristics of the applications. Ultimately, this study would allow us to better understand the limitations and accuracy of the analytical approach in the field of application performance analysis.

For comparison purposes, Noxim base-line that follow Poisson distribution (*Poisson*) has been extended to take into account the cycle-by-cycle simulation (*Accurate*) and is named as Noxim-Cycle. The results obtained with Noxim-Cycle are compared with those obtained following the Poisson distribution.

3.2.1 Noxim-Cycle

Noxim-Cycle is a modified version of Noxim base-line, adapted to use Packet injection time (PIT) instead of the packet injection rate (PIR). It was developed in collaboration with the Labsticc at the University of South Brittany in Lorient. This modification allows the realization of accurate results, but still in a very slow manner. As Noxim input, we use a table-based traffic pattern and a text file containing information about the source, destination, and PIT (Table 3.1.b) instead of PIR (Table 3.1.a).

Furthermore, the new input text file based on PIT is generally much larger than that based on PIR as it contains all the packets rather than the average injection time for each pair (source, destination). This is considered the first limitation that is encountered with this version of Noxim. Furthermore, while the Noxim output remains consistent, benchmark accuracy requires a higher level of precision. This is particularly important as we simulate packets cycle by cycle, insisting on the need for great attention to details in our analysis.

Table 3.1: Example of table-based input of Noxim: a) using PIR and b) using PIT.

Source	Destination	PIR (packet/cycle)
0	1	0.066
1	5	0.033
5	3	0.066
8	3	0.033

a) Using PIR

Source	Destination	PIT (cycle)
0	1	10
5	3	15
1	5	20
0	1	23
8	3	27
5	3	30

b) Using PIT

3.2.2 Results Comparison

Table 3.2: Parameters for latency evaluation.

Parameter	Value
N_C	16 cores
Packet size	4 flits
Flit size	32 bits
Buffer size	8 flits
Number of virtual channels	1

Taking into consideration the parameters shown in table 3.2, if we simulate two million clock cycles according to the Poisson distribution using Noxim base-line with an input of the PIR packet injection rate for each source destination, we obtain a latency $l_{poisson} =$ 13.49 cycles. However, if we simulate this latency using Noxim-Cycle with an input of the PIT packet injection time for each source destination, we obtain a latency $l_{accurate} =$ 30.01 cycles. Therefore, compared to the accurate simulator, an error of 55% is present. To further assure the high error values, these simulations were repeated on different applications such as streamcluster and x264. Similarly, high errors (more than 45%) in most of these portions were obtained. It is worth noting that the error can remain negligible unless the injection rate is very low throughout the application, as on blackscholes applications, for instance. Based on these preliminary results, we can conclude that the use of Poisson's distribution cannot be generalized when applied on real-life applications.

3.3 Benchmark Application Traffic Injection Distribution

In spite of the high error values observed when employing the Poisson distribution for real-life application performance analysis, attempts have been undertaken to elucidate the origin of this error. In other words, our objective is to demonstrate why benchmark applications do not follow the Poisson distribution. This has been done using three different metrics: i- the inter-arrival time which is the time between the arrival of successive packets, ii- the SCV calculation, and iii- the Quantile-Quantile (Q-Q) plot.

3.3.1 Inter-arrival times

In order to compare the inter-arrival times (which refer to the period of time that elapses between the arrival of consecutive packets) in a real application that follows the Poisson distribution, we first calculated the injection rate of the real application, router by router, and then globally considering all routers. Figure 3.1 illustrates the packet injection rates per source for the dedup application. For example, source 0 has a $PIR_0 = 9.416 \times 10^{-4}$ packet/cycle whereas source 6 has a $PIR_6 = 6.513 \times 10^{-3}$ packet/cycle. Overall, the average packet injection rate per source is $\overline{PIR} = 2.0146 \times 10^{-3}$ packet/cycle.

Then, the inter-arrival times for different benchmark applications were calculated. Figure 3.2a showcases the frequency of inter-arrival times of successive packets (in cycles)

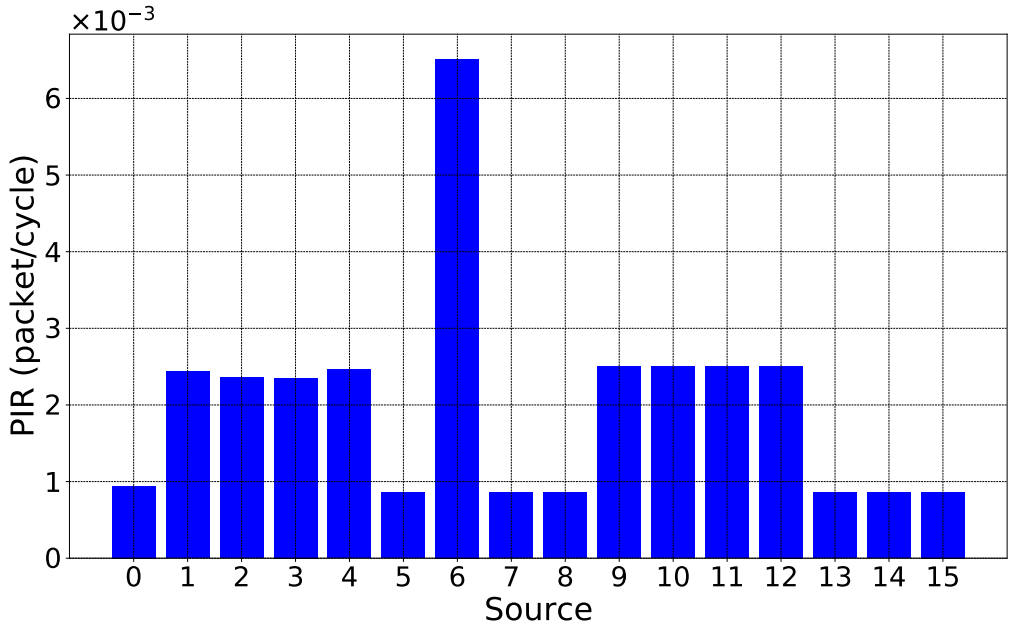
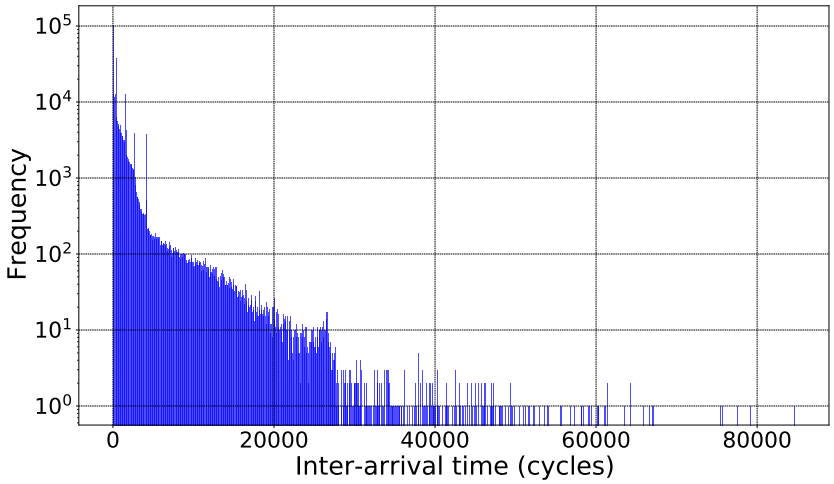


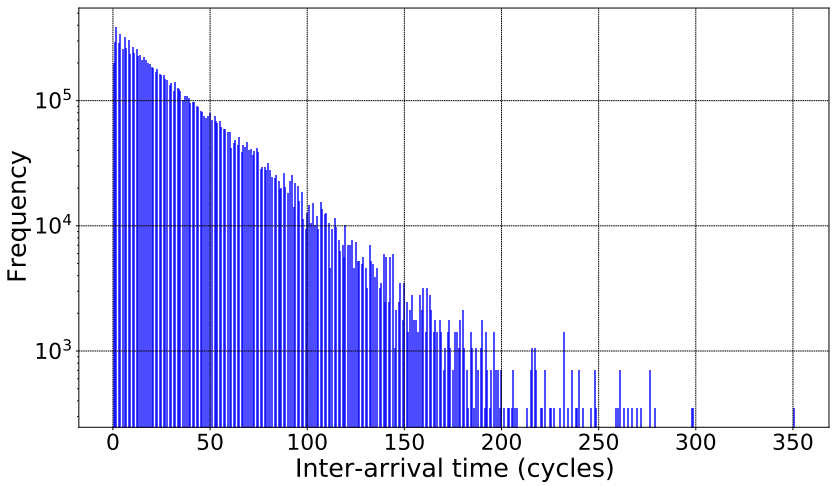
Figure 3.1: Packet injection rate per source for dedup application.

for dedup application on a logarithmic scale. The majority of data is concentrated in the inter-arrival cycle range below 1000, where frequency is highest. After this range, the frequency decreases rapidly for most inter-arrival times, although a few isolated peaks at specific points are observed. On the other hand, by applying Poisson distribution with the same packet injection rate over the same duration, different inter-arrival times were obtained and are shown in Figure 3.2b.

By comparing these results, a very huge difference in terms of inter-arrival time can be noticed. More precisely, following the Poisson law, the inter-arrival time is up to 350 cycles, which is greatly smaller than that obtained in real applications that is up to 80000 cycles. Moreover, considering the part with high frequency, the inter-arrival time was found to be up to 250 cycles when following the Poisson law which is again less than that observed using the accurate simulator with a value up to 1000 cycles. On the basis of these comparisons, we can conclude that the employment of the Poisson law over the entire execution time in the study of benchmark application will lead to erroneous results.



(a) Distribution of inter-arrival times for dedup application.



(b) Distribution of inter-arrival times for dedup application by applying Poisson distribution.

Figure 3.2: Distribution of inter-arrival times for dedup application.

3.3.2 Squared Coefficient of Variation and Q-Q Plot

Furthermore, using the inter-arrival time, the SCV can be calculated and the Q-Q plot is plotted. The SCV illustrates the dispersion of a variable, independent of the measurement unit. The SCV can be defined by:

$$scv = \frac{\sigma^2}{(E[(x)])^2} \tag{3.1}$$

where σ is the standard deviation of packet inter-arrival times and $E[(x)]$ is the mean of packet inter-arrival times.

Then, the SCV for the injection times for different benchmark applications was calculated. The obtained results are summarized in Table 3.3. The SCV values were found to be ranging from 19 to 312 for different benchmark applications. As the SCV of inter-arrival times must be equal to 1 for applications that follow the Poisson distribution, this confirms that, in the view of the obtained results, the benchmark applications do not follow the Poisson distribution.

Table 3.3: Mean, standard deviation and Square Coefficient of Variation (SCV) of the Packet Inter-Arrival Times for PARSEC Benchmarks.

Benchmark	Mean	Standard deviation	SCV
blackscholes	1506.911	26651.49	312.8
dedup	31.023	212.105	46.742
x264	9.326	70.743	57.533
streamcluster	4.878	21.361	19.17

Furthermore, the Q-Q plot (Quantile-Quantile plot) is a graphical tool for comparing the distribution of two data sets. It is mainly used to check the normality of a data set. In a Q-Q plot, the quantiles of a data set are plotted against the quantiles of a theoretical distribution (such as normal, exponential, etc.). If the data follows the chosen theoretical distribution, the points on the Q-Q plot should align approximately linearly on a straight line. This is a crucial condition to assure that a data set follows a specific distribution. It is important to mention that the time between successive events (inter-arrival times) for the Poisson distribution follows an exponential distribution. Hence, if a linear plot is obtained by simulating the data with the exponential distribution, we can conclude that the application follows the Poisson distribution.

In order to test whether our data set follows the Poisson distribution, we plotted the

Q-Q plot versus the exponential distribution. As shown in Figure 3.3-a, the plot obtained does not meet the required condition. Thus, it does not follow the Poisson distribution. Additionally, the Q-Q plot of these data sets was also plotted versus three other widely known distributions (normal, logistic, and generalized Pareto distribution GPD). The plots are illustrated in Figure 3.3-b, c and d. Similarly, the condition in these plots is not respected. Hence, the distribution of inter-arrival times does not follow any given distribution.

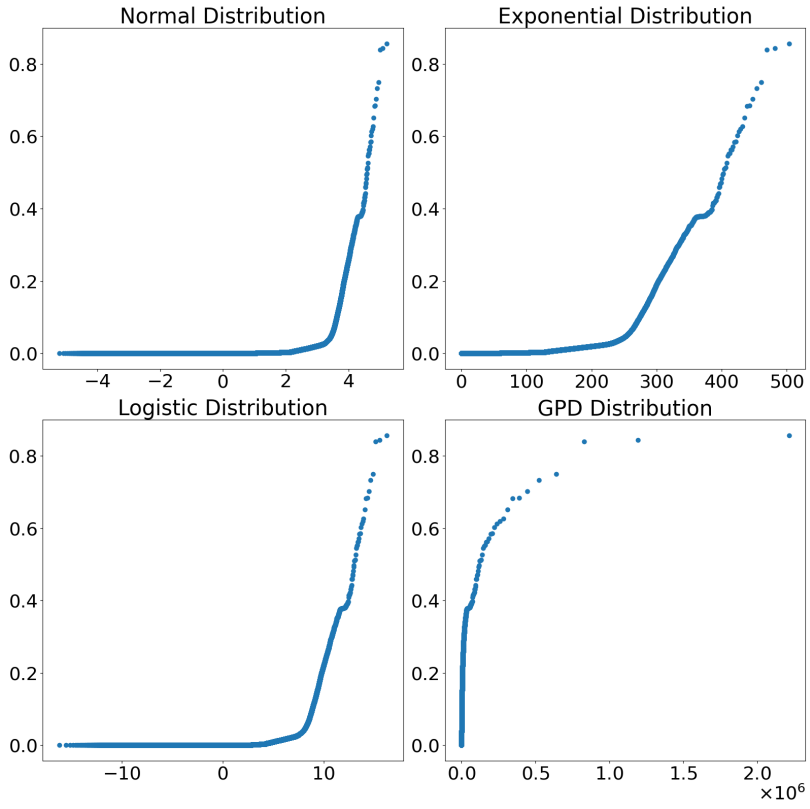


Figure 3.3: Q-Q plots of inter-arrival times of dedup benchmark against well known distributions

Therefore, it is revealed that the Poisson distribution or any other distribution are not applicable when modeling these traces as they do not follow the general profile of these distributions, more importantly for us, the Poisson one. In order to adapt the modeling of the traces with Poisson distribution, we proposed a new framework that consists of windowing the communication traces of the NoC.

3.4 Proposed Framework

3.4.1 Overview

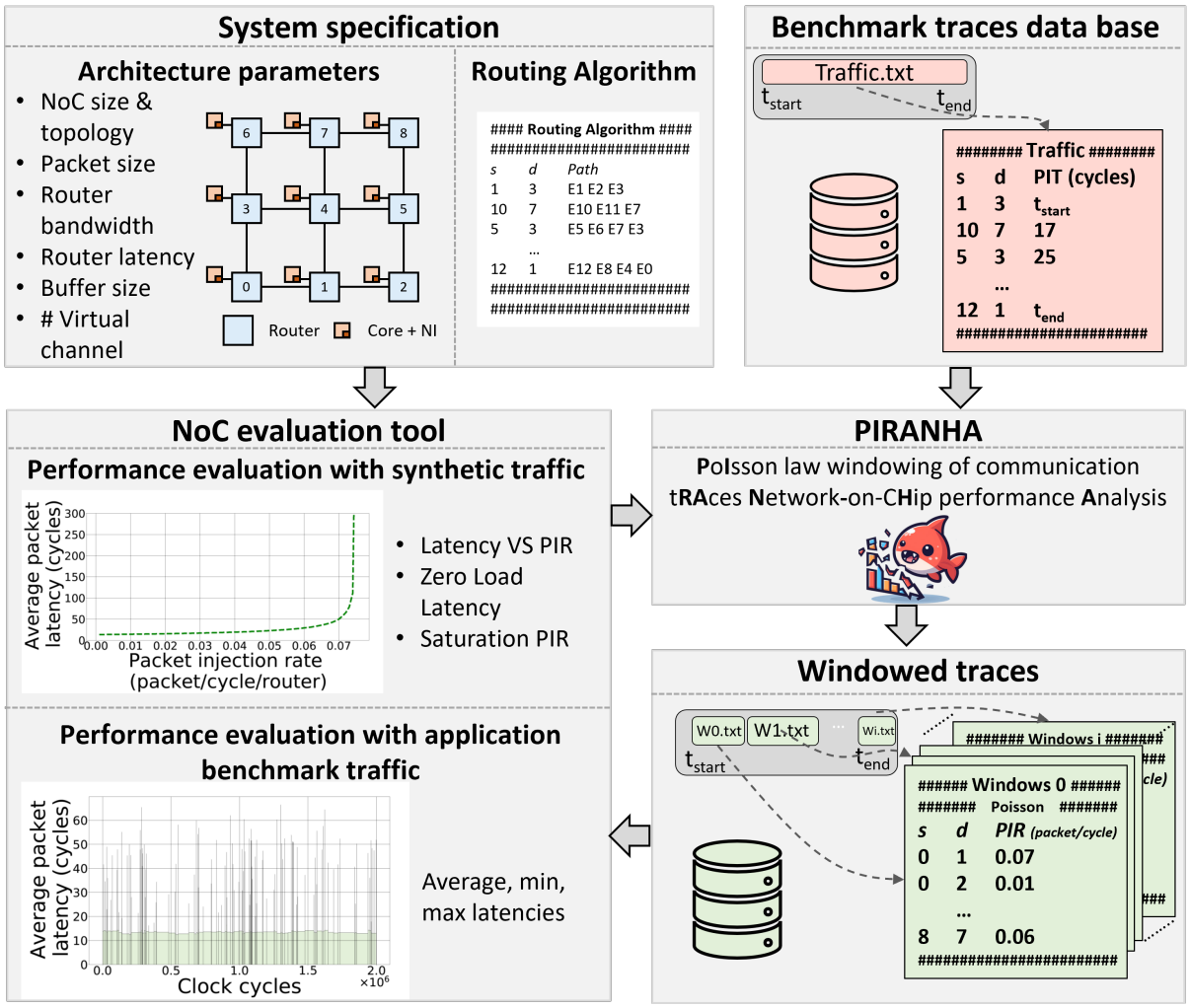


Figure 3.4: Overview of the NoC performance evaluation flow integrating the proposed PIRANHA method.

Figure 3.4 illustrates an overview of the NoC performance evaluation flow integrating our proposed method, named PoIsson law windowing of communication tRAces Network-on-CHip performance Analysis (PIRANHA). This method will be used to segment the traces of a benchmark application while minimizing error due to this segmentation. The resulting windows can be thus used to i) accelerate the performance analysis through a NoC simulator or analytical models and ii) evaluate the NoC performance for a real-life application benchmark. The evaluation flow takes as input the system specification through the interconnect architecture (described by the number of cores, routers, and how they are interconnected) and the routing paths from sources to destinations (i.e. routing algorithm). From these architectural parameters, a NoC evaluation tool can be used to evaluate the performance of the interconnect for synthetic traffic. This evaluation tool can either be a NoC simulator or an analytical model. From this first classic NoC performance evaluation, the packet latency vs PIR for random traffic patterns must be performed. For instance, it could be done with any cycle-accurate NoC simulator, such as Noxim, which integrates this feature inherently.

The PIRANHA naturally takes as input a traffic file, which is a list of all communications detailing, at least, the source, destination and injection time. This kind of application trace is done with a full-system simulator such as Sniper or Gem5 [Bin+11; HCE12]. In PIRANHA, the user can set as parameters a targeted maximum error latency evaluation. Without going into details, which will be presented further in the manuscript, PIRANHA will: i) extract key metrics of the NoC performance, such as the zero load latency and the saturation PIR from the latency vs PIR plot; ii) segment the traffic traces into small windows based on window size error analysis; iii) compute the PIR for each source and destination pair; and finally iv) iteratively merge windows with respect to a window merging error analysis.

These files can be directly used as input for NoC performance analysis tools, such as analytical models, and into cycle-accurate simulators with PIR inputs, such as NoXim simulator. As a result, a text file for every window is obtained containing the mean PIR for each source and destination pair in this time window. Therefore, from the PIRANHA method, it is possible to have some windows that do not meet the used error constraints. These files are highlighted by PIRANHA in which we will further describe the reason for this limitation and a complementary mitigation technique to limit its effect.

3.4.2 Impact of windows on errors

Understanding the impact of using windows on the overall performance analysis of a NoC is crucial to efficiently segment traces and limit susceptible errors. Different effects are associated with the use of the windowing method, as illustrated in Figure 3.5.

First and foremost, cutting traces into two parts will affect the simulation of the second window, which will start with empty buffers. This might be a source of error, especially in the case of congested NoCs. The second source of errors is due to the packets injected at the end of a window, which will not reach their destination before the end of the simulation.

It is worth noting that the percentage of the two errors (represented in red in Figure 3.5) is intertwined with the size of the windows. As window size increases relative to the size of the two erroneous parts, the overall error rate decreases. Therefore, using a heterogeneous window size to segment application traces will allow the formation of larger windows thus decreasing the error.

However, larger windows lead to the hiding of PIR variations, which is one of the most interesting aspects of application-based traces. The PIRANHA method considers these sources of errors to perform an efficient windowing of communication traces.

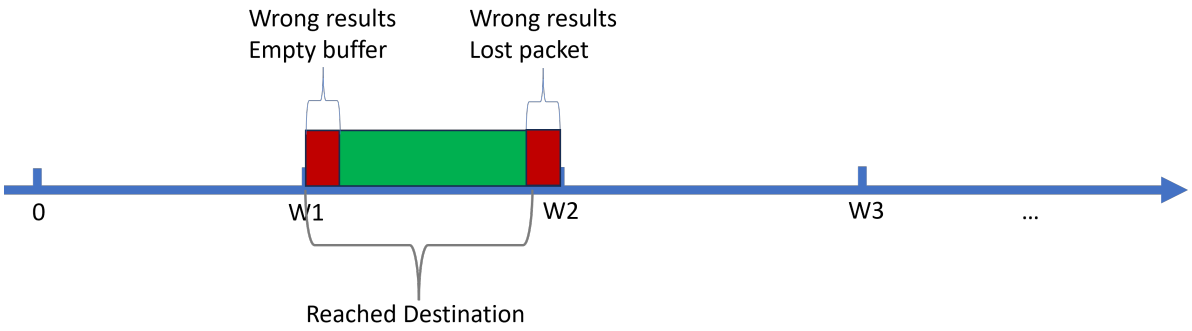


Figure 3.5: Lost/reach packets.

3.4.3 PIRANHA methodology

The PIRANHA method is depicted in Fig. 3.6. As mentioned in the previous section, it takes as inputs: i) the latency vs. PIR plot from a NoC evaluation tool; ii) the user constraints; and iii) the traces to windows. Regarding the user constraints (*user_constraint*)

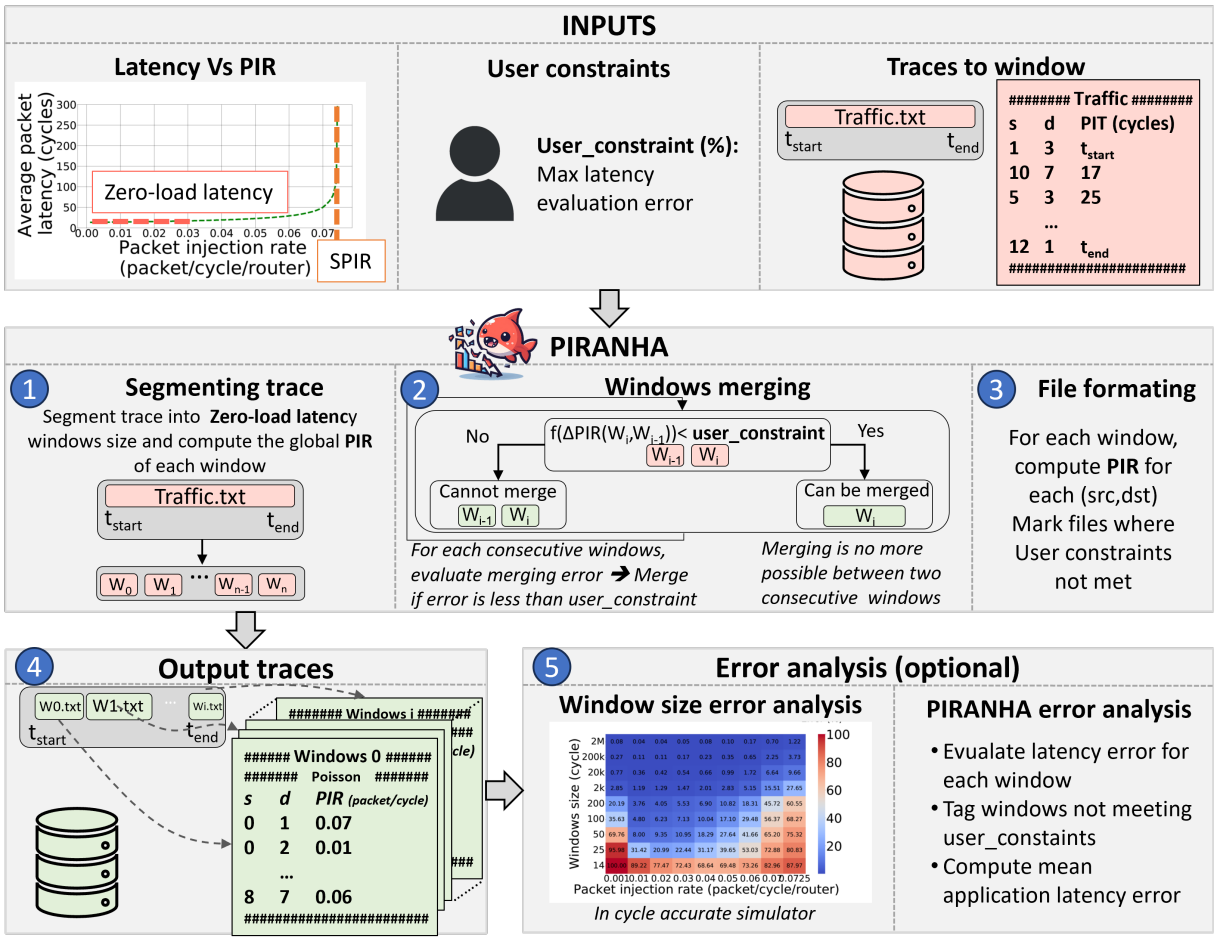


Figure 3.6: PIRANHA method.

in %), it allows the user to define the acceptable error percentage of latency due to trace segmentation compared to a cycle-accurate simulation with packets injected cycle by cycle.

The proposed method consist in the following main steps:

- **Segmenting traces** (*Mark 1*):
  The traces are segmented in several files of $zero - load_{latency}$ clock cycles. For each file, the average PIR is computed. This $zero - load_{latency}$ is extracted from the input. Thus resulting in a large number of windows containing consecutive windows with similar PIR.

- **Windows merging** (*Mark 2*):
  For each couple of two consecutive windows $i-1$ and i, the method analyzes the

possibility of merging. For this purpose, PIRANHA computes the merging effect as follows: The average latency without merging is extracted from the Latency vs. PIR input plot (L_{W_i} and $L_{W_{i-1}}$). Then the average latency on the two windows $L_{(W_{i-1},W_i)}$ is computed as follows:

$$L_{(W_{i-1},W_i)} = \frac{L_{W_i} \times S_{W_i} + L_{W_{i-1}} \times S_{i-1}}{S_{W_i} + S_{i-1}} \tag{3.2}$$

After that, the average PIR is computed in the case where the two windows are merged as follows:

$$PIR_{(W_{i-1},W_i)} = \frac{PIR_{W_i} \times S_{W_i} + PIR_{W_{i-1}} \times S_{i-1}}{S_{W_i} + S_{i-1}} \tag{3.3}$$

From this new PIR, the associated latency $L_{merging}$ is determined from the input plot. Finally, the error of latency between $L_{(W_{i-1},W_i)}$ and $L_{merging}$ is computed by:

$$Error = \frac{|L_{merging} - L_{(W_{i-1},W_i)}|}{L_{merging}} \times 100 \tag{3.4}$$

and the result is compared to the user constraint. In the case where the error is smaller, merging is applied; elsewhere, the window is kept in two files. We repeat this process iteratively until no more windows can be merged. Naturally, the higher the user constraint value, the more merging of windows can be performed.

- **File formating** (*Mark 3*):
  The PIR is computed for each couple of source and destination, and traces with injection cycles are erased.

- **Output results** (*Mark 4*):
  As a result, PIRANHA outputs a set of heterogeneous text files. Each file contains the PIR for each source and destination couple, where the windows size have been adapted to have windows as large as possible without the loss of PIR variations.

- **Error analysis (optional)** (*Mark 5*):
  An optional part of the PIRANHA method concerns the evaluation of windowing traces. This analysis is done in two phases: i) Window size error analysis; and ii) PIRANHA error analysis.

  The first part requires a cycle-accurate simulator that can consider the injection of

packets with PIT. In our case, we use a modified version of NoXim that allows this feature. First, we generate for a fixed value, PIR, a trace file with synthetic random traffic with a number of cycles similar to the application. Then we compute the associated file with PIR following Poisson's law. Both files are then injected to the simulator and the latency results are obtained and compared. The file with PIT is the baseline to compute the analysis error.

After this, the traces are segmented into a fixed number of n files, and the PIR is computed and simulated. The average packet latency of these n files is compared with the average latency of the baseline. This process is repeated to create window-size error analysis matrices. Each cell of this matrix provides the error for a given PIR vs. window size.

The second part of this optional step consists of comparing each PIRANHA windows with this matrix. From this analysis, the method is able to determine the average latency error, compute the mean error for the whole application, and tag windows where the user constraint is not met.

In the following, we estimate the efficiency of the method with real-life applications.

3.5 Experimental Results

3.5.1 Experimental Setup

In order to validate the proposed framework, we need to validate the error of merging two windows (to be less than that indicated by the user) and the minimum window size. To analyze the latency with Noxim-Cycle, which is very slow, with real applications that are very large (158 516 584 cycles for blackscholes, 344 638 994 cycles for streamcluster, etc.), it can take several days or even weeks of simulation. This is why we propose to simulate portions of 2 million cycles, for example. We have therefore applied this approach to a large number of portions, each of 2 million cycles and from different benchmark applications. This approach was first applied with an example of 2 million clock cycles in the dedup benchmark application. This required first the analysis of the random traffic with the same system specifications as for the benchmark applications (Table 3.4).

Table 3.4: Parameters for experimental setup.

Parameter	Value
N_C	16 cores
Packet size	4 flits
Flit size	32 bits
Buffer size	8 flits
Number of virtual channels	1 then 4 VC

3.5.2 Random Traffic Analysis

The random traffic is used to determine the average latency error for windows of different sizes. By knowing the user's restriction maximum latency error, the possible window sizes can be determined along with the specification of the windows that could be simulated either in *Poisson* or *Accurate*.

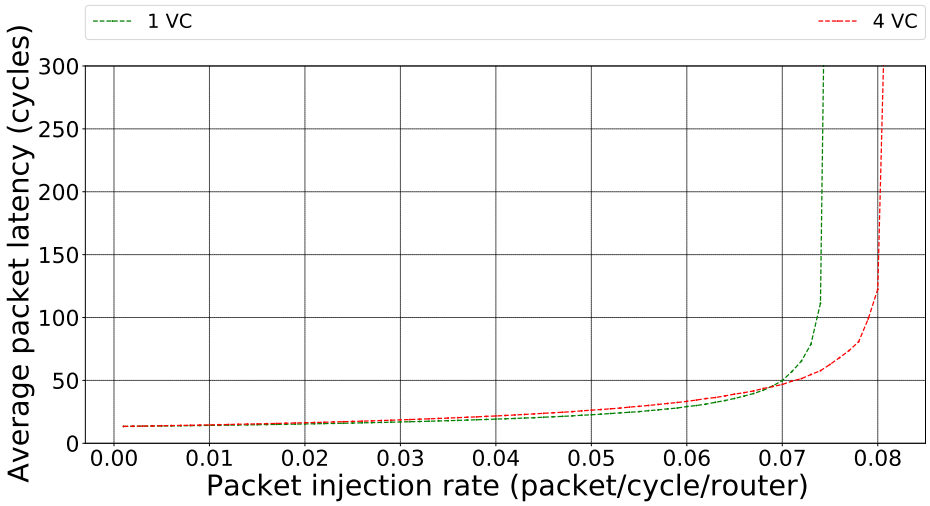


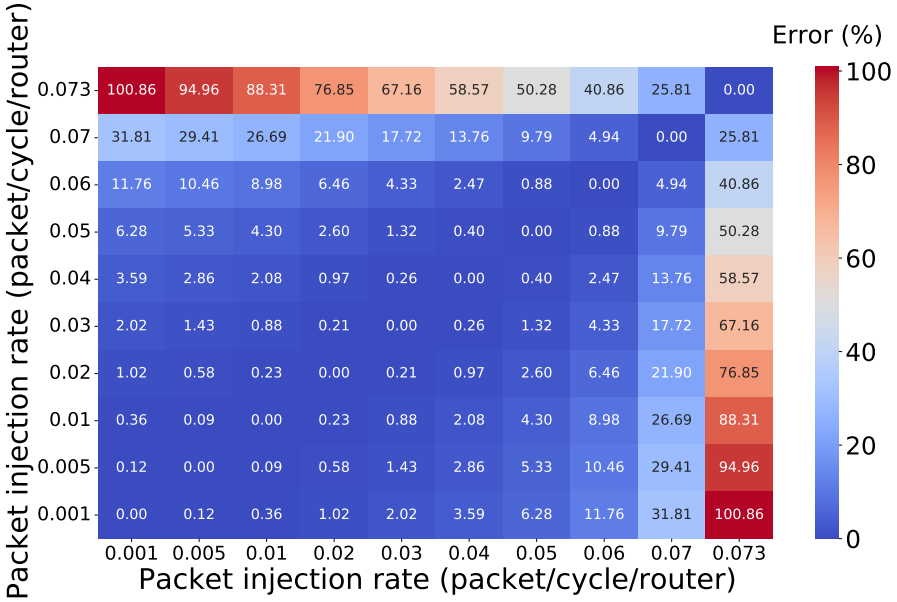
Figure 3.7: Average packet latency Vs. PIR for 1 and 4 virtual channel.

Figure 3.7 shows the evolution of the average packet latency versus the packet injection rate using 1 and 4 virtual channels. Using this plot, the zero-load latency and the SPIR can be determined. The zero-load latency (latency without congestion) is 13.8 cycles for 1 and 4 virtual channels, and the SPIR is 0.073 and 0.08 packet/cycle/router for 1 and 4 virtual channels, respectively. These values are used in the determination of the matrices detailed in the following section.

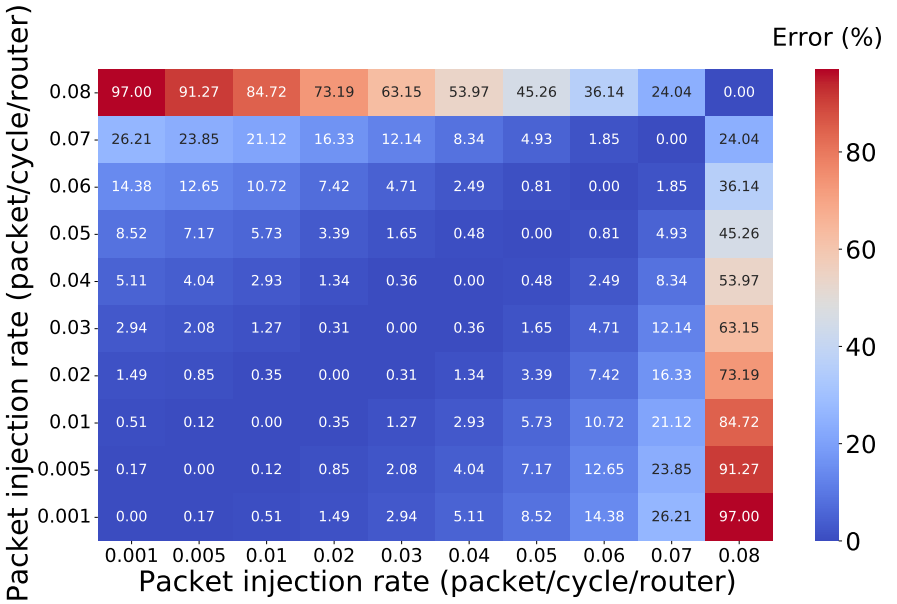
3.5.3 Error Matrices

Matrix merge: Figure 3.8a illustrates the error rate obtained for a two-window merging experiment made by Noxim for random traffic using 1 virtual channel. The injection rates of these two windows to be merged, represented by the axes, are combined, and the result of this combination is the latency errors that are represented in the cells of this heat map. These errors are expressed as a percentage and are colored gradually from blue to red, indicating an increase in error from 0% to 100%. The error values range from 0.001 (zero-load latency) to 0.073 packets per cycle per router (SPIR). The increase in error values suggests that the injection rate difference between the two windows is large. On the other hand, merging two windows with similar injection rates will lead to a decrease in the value of the error obtained. Likewise, simulations with 4 virtual channels show similar behavior as for 1 virtual channel (Figure 3.8b).

Matrix size: Figure 3.9a shows the different obtained latency errors by simulating the different PIR with different window sizes (from zero load latency to the application size) using Noxim with a random traffic pattern for a single virtual channel. The vertical axis corresponds to the window size expressed in cycles, with values ranging from 14 cycles (zero load latency) to 2 million cycles (application size). The horizontal axis, on the other hand, corresponds to the packet injection rate expressed in packets per cycle per router, with values ranging from 0.001 (zero load latency) to 0.0725 (SPIR). Similarly, each cell of the matrix represents the percentage of error associated with a specific combination of window size and packet injection rate. Again, with this matrix, the error percentages are color-coded, ranging from blue for low errors to red for high errors, as indicated by the color legend on the right side, which ranges from 0 to 100 %. The numerical values within the matrix suggest that, for small window sizes and any packet injection rate, the error is high (close to 100 %), while for larger window sizes and any packet injection rate, the error is much lower. Likewise, simulations with 4 virtual channels show similar behavior as for 1 virtual channel (Figure 3.9b).

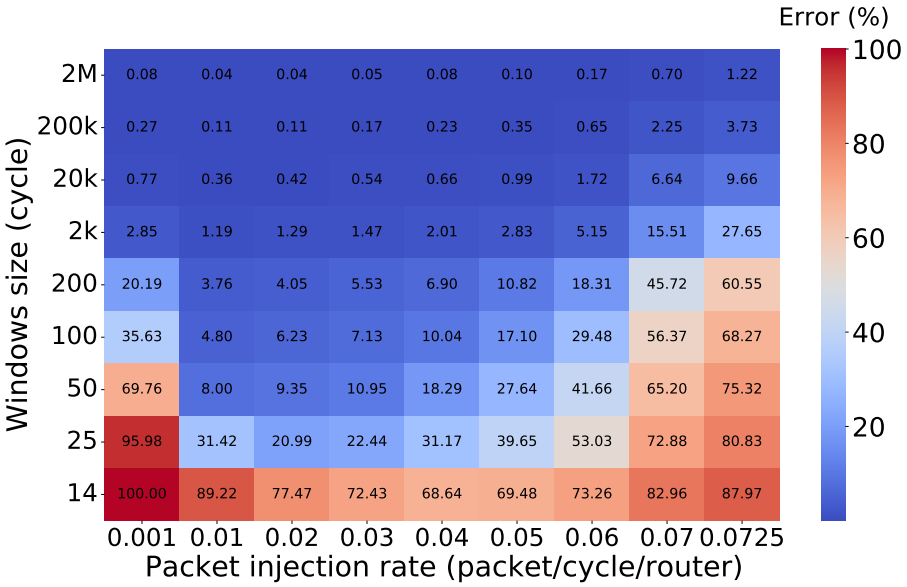


(a) Error matrix for 1 virtual channel.

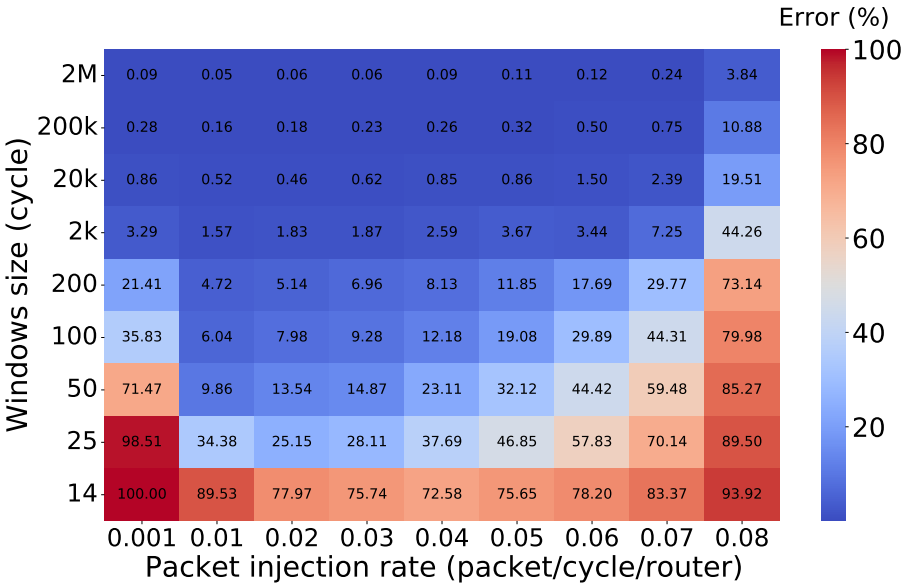


(b) Error matrix for 4 virtual channels.

Figure 3.8: Error matrix obtained when merging the two windows.



(a) Error matrix for 1 virtual channel.



(b) Error matrix for 4 virtual channels.

Figure 3.9: Error matrices obtained versus size window.

3.5.4 PIRANHA Validation

For windowing just with Poisson distribution, the application has to be segmented according to the zero-load latency (14 cycles). Then windows with similar PIR are merged in accordance to the value of the user's restriction maximum latency error which is indicated to be 5 % in this study. Consecutive windows that result with an error smaller than that of the error indicated by the user are merged.

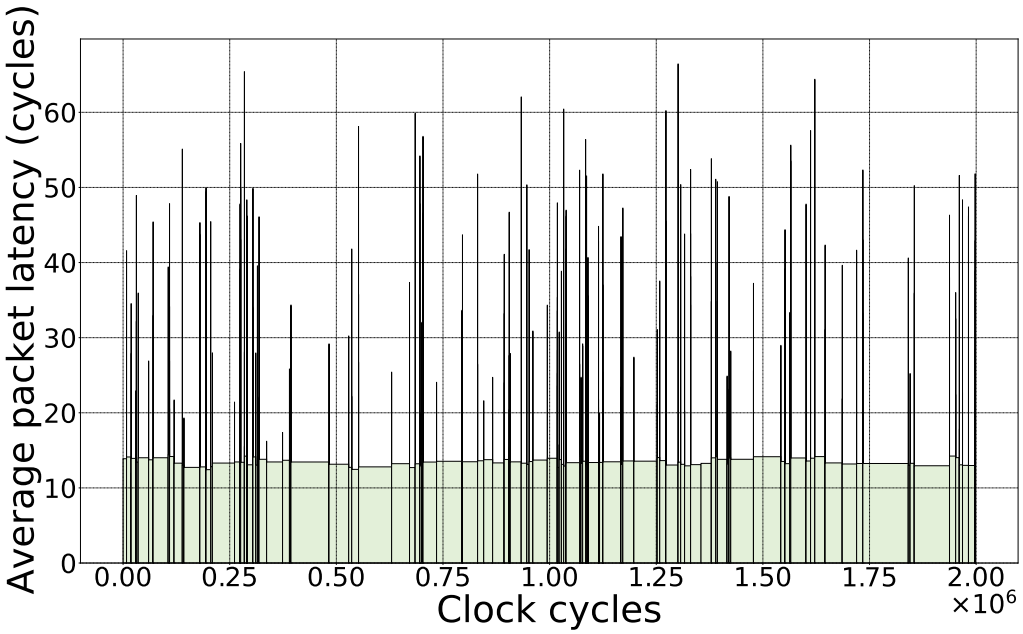


Figure 3.10: Validation of PIRANHA using the dedup benchmark application with *Poisson* method for traces windowing.

Figure 3.10 illustrates the results of 2 million clock cycles of the dedup application, simulated with the initial version of Noxim base-line following a Poisson distribution. A variety of window sizes, with very large and very small windows can be observed where the width of the bar indicates the size of the window. It should be noted that the number of windows is related to how many can be fused together. Hence, the more errors that can be accepted, the more the final number of windows can be reduced. Finally, the average latency is calculated using the following formula:

$$Average\_latency = \frac{\sum_{i=0}^{n_w} L_{w_i}^P}{n_w} \tag{3.5}$$

Where nw is the total number of windows, pir_i the PIR of the window and L^P_{wi} the latency of the window obtained by Noxim base-line simulator using *Poisson* distribution.

Having calculated the average latency ($Average\_latency = 31.26 cycles$ and standard deviation $\sigma = 14.52$), the error was then evaluated and found to be 15.7 % which is higher than that accepted by the user (5 %). This is related to the fact that even after merging the windows, there are still windows with very small size, thus resulting in additional errors as discussed in section 3.5.3. Furthermore, this analysis was repeated on different benchmark applications. The average error was found to be around 10 % with a maximum error of 21 % and a minimum error of 2 %. The latter can be found in the blackscholes application and is linked to the very low injected packet number compared to other applications.

In order to solve the present error problem and try to decrease it, an alternative analysis of generated windows is suggested and consists of utilizing both *Accurate* and *Poisson* as will be discussed in the following section.

3.5.5 PIRANHA Extension : Windowing *Poisson* and *Accurate*

To address the current issue and attempt to reduce errors, an extension to the PIRANHA approach is proposed for analyzing generated windows. This approach involves employing both *Accurate* and *Poisson* in the NoC performance analysis. The PIRANHA extension is used when the user's restriction maximum latency error rate is rather lower than the minimum error rate that would be provided by following the Poisson distribution. As illustrated in Figure 3.11, windows yielding a high error rate when merged with the Poisson distribution will be rather merged using the *Accurate* one thus resulting in a lower error rate.

With this method, windows with sizes smaller than the accepted minimum window size for each PIR are no longer accepted. This is related to the fact that the error rate is greatly influenced by the window size in light of the results obtained with the established matrix size (Section 3.5.3). As these small sized windows result in high errors when merged and when simulated separately in *Poisson* without merging, the PIRANHA extension involves the simulation of these windows in *Accurate* after merging using the modified Noxim version, Noxim-cycle (refer to section 3.2.1).

Figure 3.12 illustrates the results of 2 million clock cycles of the dedup application, simulated both with the Noxim base-line following a *Poisson* distribution (results shown in red bar) and Noxim-Cycle with *Accurate* (results shown in blue bar) for two different user restriction maximum latency errors 5 % and 20 % (Figures 3.12b and 3.12a, respectively).

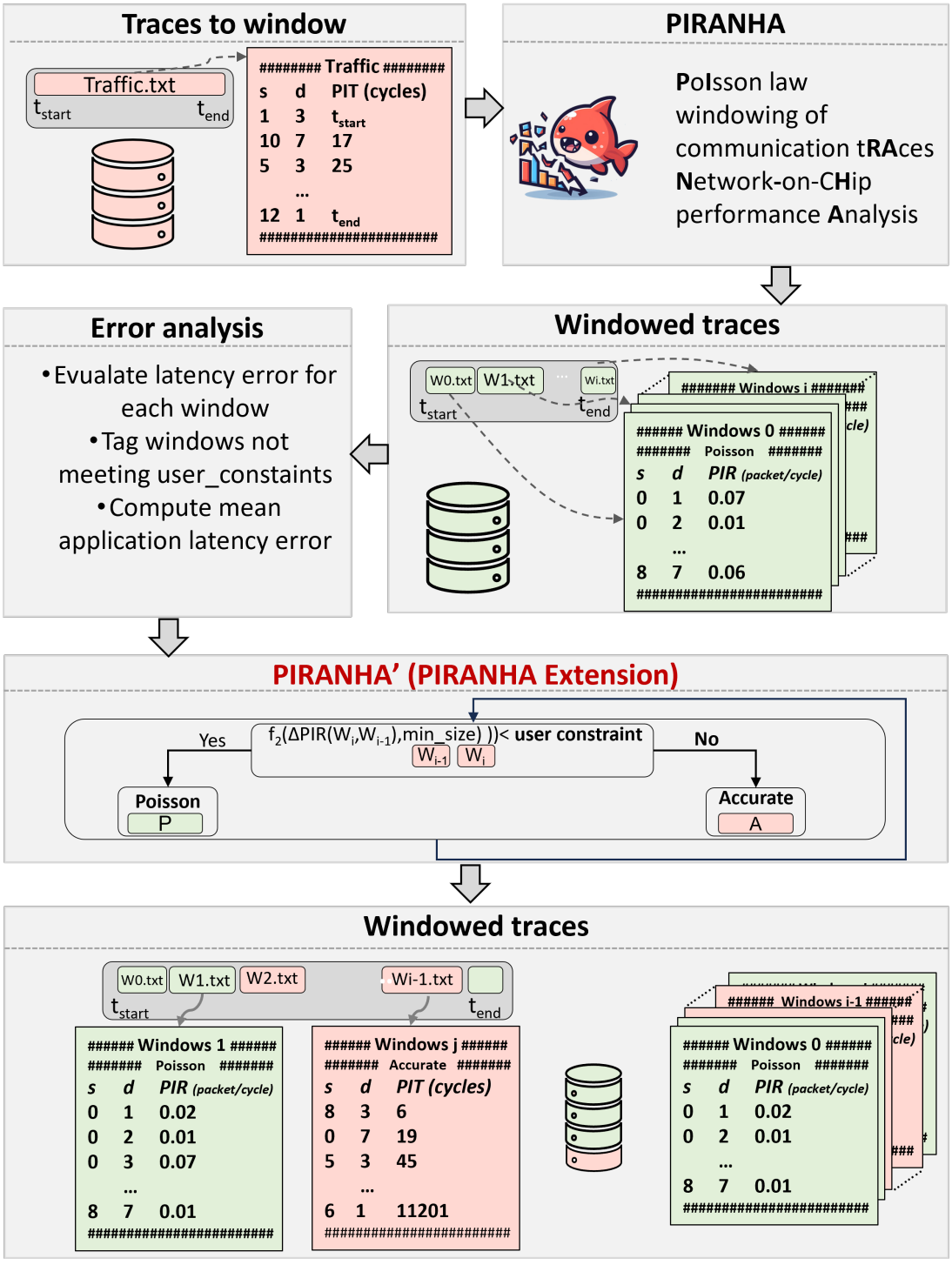
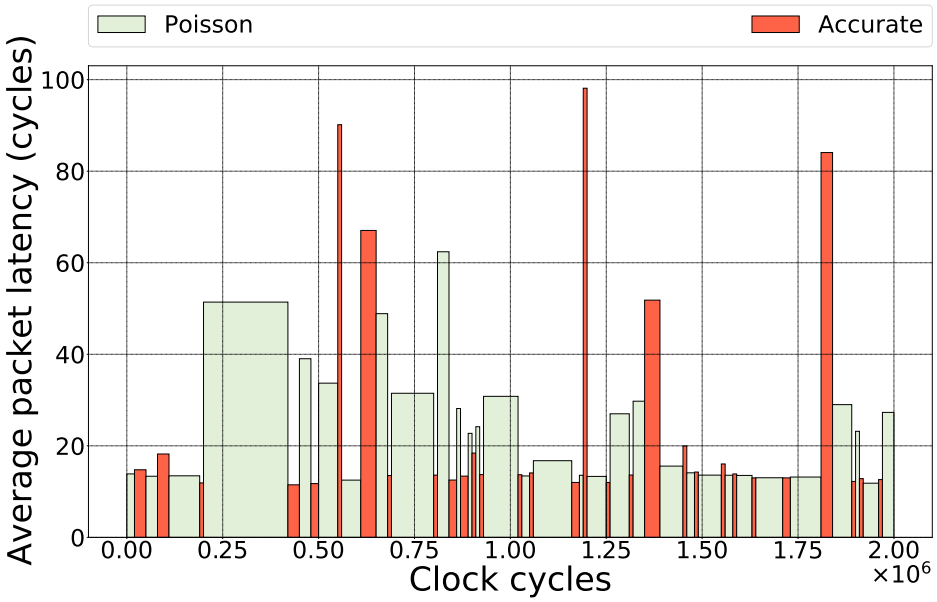
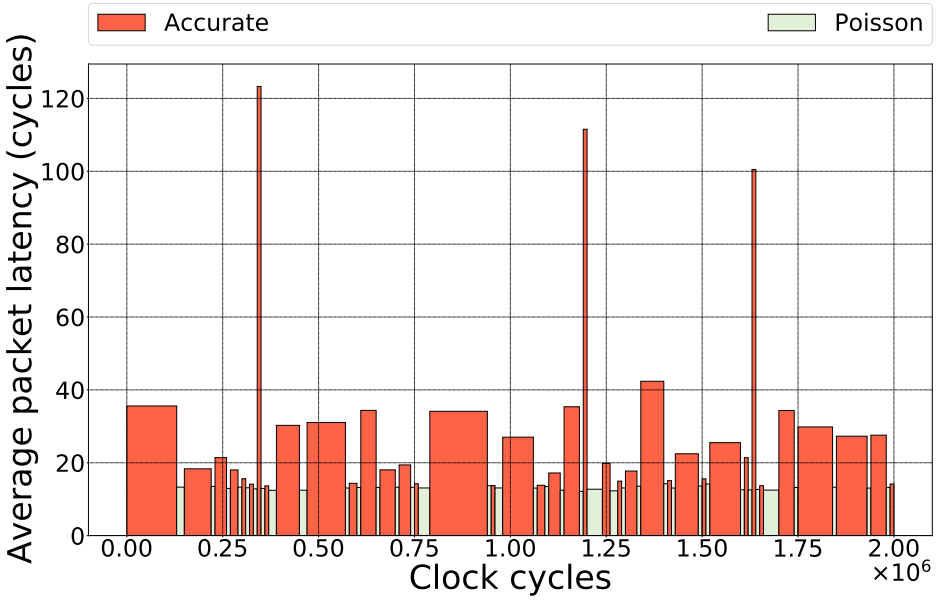


Figure 3.11: PIRANHA Extension method.



(a) User restriction maximum latency error = 20%



(b) User restriction maximum latency error = 5%

Figure 3.12: Validation of PIRANHA using the dedup benchmark application with *Poisson* & *Accurate* method for traces windowing.

Based on that, when the user's restriction maximum latency error increases, the more the *Poisson* simulations can be used. For a 5 % user restriction maximum latency error, 25 % of the application can be performed with *Poisson* and the rest with *accurate*. However, for a 20 % user restriction maximum latency error, 75 % of the application can be performed with *Poisson* and the rest with *Accurate*.

3.5.6 Discussion

3.5.6.1 *Poisson* and *Accurate* utilization

Figure 3.13 shows a comparison between the accuracy of two methods and the error associated with each, in terms of latency and percentage of use of each method. The blue bars represent the percentage of *Poisson* usage, while the red bars indicate the percentage of *Accurate* usage. These values are calculated versus the user's restriction maximum latency error (%). The right vertical axis corresponds to black line that shows the percentage of error. For instance, for a very low user restriction error (around 0 %),

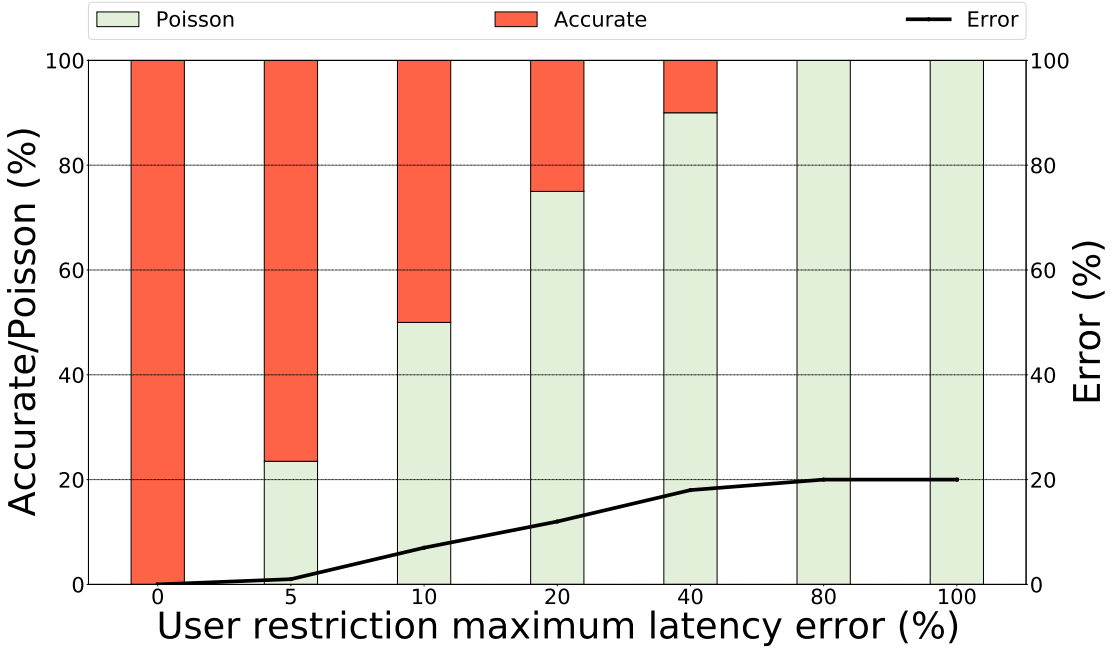


Figure 3.13: Validation of PIRANHA using the dedup benchmark application with *Poisson* & *Accurate*.

the framework parses to a total use of the accurate method, giving a zero error. However, for 5 % user restriction error, the framework was found to use the *Poisson* method for

23 % of the cycles within the application with an error of less than 1 %, which is related to the fact that our initial studies were done on a random traffic, i.e. the worst case. Additionally, for a user restriction error above 60 %, a total use of the *Poisson* method is achieved with an error tending towards 20 %.

3.5.6.2 Input file size

A different aspect in this study is the input file size that differs between *Poisson* and *Accurate*. It was found that the input file size of the *Poisson* (e.g., 6.4 KB for dedup) is much smaller than for *Accurate* (e.g., 149 MB). This offers a further advantage of employing the *Poisson* approach as it helps in conserving valuable storage space.

Table 3.5: Input size for *Accurate* and *Poisson* for different benchmark applications.

Application	*Accurate* input size	*Poisson* input size
blackscholes	1.49 MB	6.6 KB
dedup	149 MB	6.4 KB
streamcluster	350 MB	6.5 KB
x264	348 MB	6.4 KB
raytrace	334 MB	6.4 KB

3.5.6.3 Execution time

Figure 3.14 shows the average speedup of the simulation using Noxim base-line (*Poisson*) versus Noxim-Cycle (*Accurate*) according to the windows size for different applications. For instance, for a 2 million clock cycle window size, Noxim-Cycle requires 7759 s of simulation time, whereas Noxim base-line requires only 47.4 s. Thus Noxim base-line is 160 $\times$ faster than Noxim-cycle. This explains the need for the hybrid use of both Noxim-cycle for *Accurate* simulations and Noxim base-line for *Poisson* simulations to facilitate the analysis for a part of the application while maintaining an acceptable error rate that relies within the user's restriction maximum latency error. This is basically why PIRANHA is established.

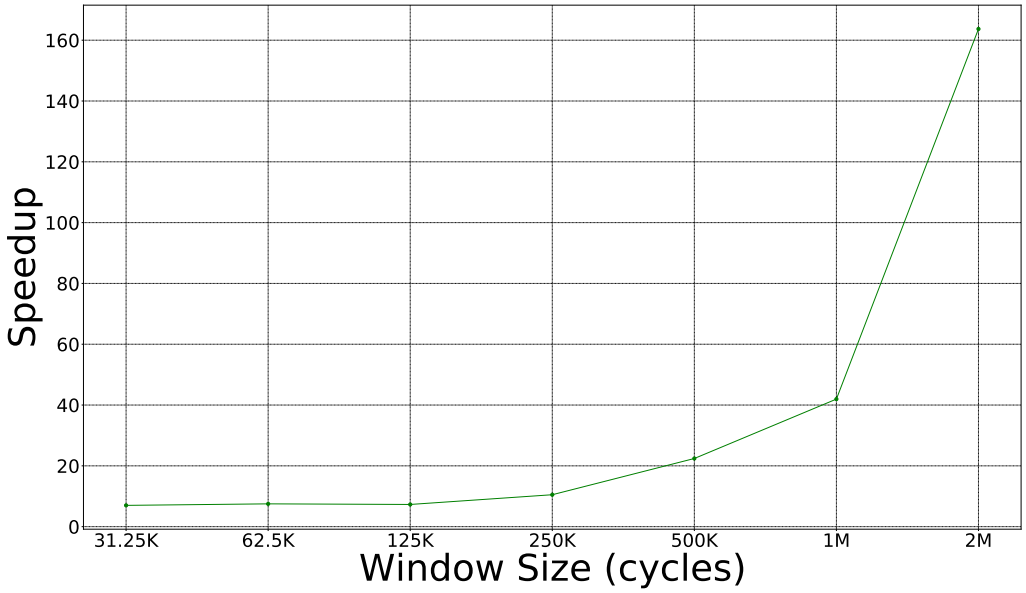


Figure 3.14: Speedup evaluation.

3.5.6.4 Number of windows

Another aspect that could be important to analyze is the number of windows and its effect on the overall NoC performance analysis. Ultimately, the final number of windows depends on two initial parameters. The first parameter is the size of the base window (taken in our study at zero load latency). However, this will give a very large number of starting windows, as for example, for 2 million cycles, there are 142857 windows.

The second parameter that also plays a role in determining the final number of windows is the merging of windows according to an indicated acceptable merging error. The more windows we are able to merge, even with a larger PIR difference, the more the total number of windows decreases. Introducing windows of variable sizes can help analyze the latency evolution by determining the peak and trough values. Using our proposed framework, PIRANHA, 230 windows were obtained with an error of around 20 %. Then, by using the PIRANHA extension, and by applying the *Accurate* in a part of the application, we could obtain 75 and 55 windows for a user acceptance error of 5 % and 10 %, respectively.

Furthermore, it should also be noted that the number of windows can indirectly affect simulation time, making it possible to simulate several windows in parallel, depending on the number of processors in the simulation tool.

3.6 Conclusion

In conclusion, although full system simulation offer the highest accuracy in assessing interconnect performance and its impact on applications, they remain extremely time-consuming. NoC simulators and analytical models, represent a faster alternative and are widely used. However, the current approach of injecting traces into NoC performance analysis tools requires particular attention to guarantee the accuracy and fidelity of results, underlining the need for a standardized methodology and more in-depth analyses to optimize these evaluations.

In this chapter, we explored the detailed process of windowing application traces for rapid analysis of NoC performance. We addressed the latency analysis of benchmark application traffic, introducing methods such as Noxim-Cycle and assessed the impact of variations in traffic injection distribution. Our proposed framework highlighted the importance of understanding the error due to windowing and presented a window splitting and merging technique to optimize the analysis.

Experimental results revealed that our approach can effectively capture random traffic characteristics while minimizing errors in performance predictions. In particular, the study of *Poisson* and mixed *Accurate* and *Poisson* demonstrated the ability of the proposed model to balance accuracy and computational complexity.

Finally, the discussion and analysis of execution time reinforces the viability of our method in real application scenarios, offering a window into its potential application for the design and optimization of NoC systems. This chapter paves the way for future research to further refine these methods and extend their applicability to systems of greater scale and complexity.

Chapter 4

Conclusion and perspectives

4.1 Conclusion

In this manuscript, we delved into the study of NoC architectures, an essential component of modern microelectronics. Our analysis began with a historical and technical exploration of microprocessor developments, highlighting how NoCs have become a crucial element in embedded system designs.

We examined the different topologies of NoCs, their data transfer formats, and the key components that make them up. Emphasis was placed on routing algorithms, flow control and arbitration methods, as well as the advantages and disadvantages of conventional electrical NoCs. We then illustrated the emerging technologies such as optical NoCs, RF-Interconnect, surface wave interconnects, and wireless NoCs.

The study of NoC traffic models revealed the importance of selecting traffic patterns, whether synthetic or derived from reference applications, to realistically assess NoC performance.

In the second chapter of this manuscript, we presented our proposed analytical model, AMHNOC. It is an analytical model for hybrid NoCs with antennas for long-distance communication, offering a comprehensive performance assessment through various evaluation and simulation tools. The experiments conducted explored various parameters, such as traffic patterns, architecture size, packet size, wireless data rate, and number of antennas.

In the third chapter, we represented our study of application trace windows for rapid analysis of NoC performance. This approach enabled us to reduce errors and optimize processes in terms of traffic latency and traffic injection distribution. For the dedup application, for example, we were able to reduce the error from 55% to 15% simply by using the Poisson distribution, and to less than 5% by combining accurate methods with the Poisson distribution.

This study not only highlighted the complexities and challenges associated with NoC

architectures, but also paved the way for new avenues of research and optimization in this constantly evolving field. NoCs continue to play a pivotal role in system-on-chip efficiency and performance, and a thorough understanding of them is essential for future advances in microprocessor and embedded system design.

4.2 Perspectives

One of the main contributions of this thesis is the AMHNOC analytical model, designed to assess the performance of hybrid NoC, particularly those incorporating wireless technologies. In future perspectives, it is crucial to extend and update this model so that it can also analyze other emerging on-chip interconnects, such as guided RF or photonic on silicon where the channel accesses differ. For example, for photonic on silicon, the optical link may be shared with different optical network interfaces, and the bandwidth can be adapted on demand thanks to the optical wavelength allocations [Luo18]. This enhancement would make the AMHNOC model more universal and applicable to a wider range of NoC technologies, taking into account the latest advances in this field. In addition, the application of this updated model to hybrid chiplets represents a promising line of research, offering prospects for significant improvements in performance and energy efficiency in high-performance computing systems.

The second major contribution of this research is the development of methods for windowing application traces to accelerate NoC performance analysis. Future work could focus on improving and expanding this framework, testing it with different distribution laws beyond the traditional Poisson distribution. This exploration could reveal unexpected network behaviors or performance patterns under different traffic conditions, contributing to a deeper understanding of NoCs. In addition, a promising prospect would be the adaptation and optimization of Noxim-Cycle, the modified NoC simulator, as discussed in chapter 3 is very slow. By developing a faster version of Noxim, such as using parallelization in simulation, we could not only improve the efficiency of NoC simulations, but also make the process more practical and accessible for researchers and engineers. This improved version of Noxim, coupled with our advanced windowing method, could open up new possibilities for performance analysis in complex NoC architectures. The aim would be to contribute to the creation of robust and flexible NoC performance analysis tools, adapted to the challenges of modern computing architectures.

Another interesting point is the integration of Deep Neural Network (DNN) with

broadcasting functionalities in chiplet-based multicore architectures represents a promising breakthrough for theses on NoCs and emerging technologies, offering an effective combination of neural signal processing and communication optimization [Sil+23].

In conclusion, these two major contributions, AMHNOC and application trace windowing, with their possible extensions, notably the adaptation of Noxim and the integration of various traffic distribution models, form the basis of a powerful analysis framework. By taking energy analysis into account, this set of tools and methods could be transformed into a robust and comprehensive solution for evaluating and optimizing the performance and energy efficiency of networks-on-chip. The use of an analytical method for analyzing energy efficiency in NoC focuses on the application of mathematical models to understand and optimize energy consumption. For example, in [Mar+14], the authors present a method for estimating the energy and power of NoCs and processors based on a Register-Transfer Level (RTL) description. The paper provides a set of results, highlighting the energy consumption attributable to the applications, to each element of the system, as well as the impact of two low-energy strategies. This approach involves detailed modeling of the energy consumption of different NoC components, such as routers and communication links, taking into account factors such as operating frequency, and traffic patterns. This integrated framework would represent a significant advance in the field of embedded system architectures, and pave the way for future developments in the optimal management of computing resources.

Scientific Contributions

Publications

[P1] **Ibrahim Krayem**, Romain Mercier, Cédric Killian, Angeliki Kritikakou, and Daniel Chillet. 2023. Data and Fault Aware Routing Algorithm for NoC Based Approximate Computing. In Proceedings of the 17th ACM International Symposium on Nanoscale Architectures (NANOARCH '22). Association for Computing Machinery, New York, NY, USA, Article 17, 1–6. https://doi.org/10.1145/3565478.3572327.

[P2] **Ibrahim Krayem**, Joel Ortiz Sosa, Cédric Killian and Daniel Chillet, "Analytical Model for Performance Evaluation of Token-Passing-Based WiNoCs," in IEEE Design & Test, vol. 40, no. 6, pp. 136-148, Dec. 2023, doi: 10.1109/MDAT.2023.3309730.

Communication

[C1] **Ibrahim Krayem**, Cédric Killian, and Daniel Chillet. An Analytical Model for Hybrid Network on Chip Latency Analysis. Colloque National du GDR SOC^2, June 2022.

Posters

[1] PhD Students' Day 2021 - Rennes.

[2] Lannion PhD Students' Day 2022 - Lannion.

[3] D3 PhD Days' 2022 - Rennes.

[4] AML Summer School 2022- Lorient.

Acronyms

Bibliography

[AAK10] Akram Ben Ahmed, Abderazek Ben Abdallah, and Kenichi Kuroda, *Architecture and Design of Efficient 3D Network-on-Chip (3D NoC) for Custom Multicore SoC*, *in*: *2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, Nov. 2010, pp. 67–73, DOI: 10.1109/BWCCA.2010.50.

[Ack+00] B. Ackland et al., *A single-chip, 1.6-billion, 16-b MAC/s multiprocessor DSP*, *in*: *IEEE Journal of Solid-State Circuits* 35.*3* (Mar. 2000), Conference Name: IEEE Journal of Solid-State Circuits, pp. 412–424, ISSN: 1558-173X, DOI: 10.1109/4.826824.

[AI] AI AI, *What Is Moore's Law and How Does It Impact AI ? - Unite.AI*, URL: https://www.unite.ai/moores-law/ (visited on 09/29/2023).

[App] Apple, *Apple unveils M2 Pro and M2 Max: next-generation chips for next-level workflows*, en-US, URL: https://www.apple.com/newsroom/2023/01/apple-unveils-m2-pro-and-m2-max-next-generation-chips-for-next-level-workflows/ (visited on 05/11/2023).

[AS09] Ankur Agarwal and Ravi Shankar, *Survey of Network on Chip (NoC) Architectures & Contributions*, *in*: *Journal of Engineering, Computing and Architecture* 3 (Jan. 2009).

[AS16] Ayaz Akram and Lina Sawalha, *×86 computer architecture simulators: A comparative study*, *in*: *2016 IEEE 34th International Conference on Computer Design (ICCD)*, Oct. 2016, pp. 638–645, DOI: 10.1109/ICCD.2016.7753351, URL: https://ieeexplore.ieee.org/abstract/document/7753351 (visited on 02/25/2024).

[AS19] Ayaz Akram and Lina Sawalha, *A Survey of Computer Architecture Simulation Techniques and Tools*, *in*: *IEEE Access* 7 (2019), Conference Name: IEEE Access, pp. 78120–78145, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2019.2917698.

[AT23] Vijaya Bhaskar Adusumilli and Venkatesh TG, *Traffic Characterization Based Stochastic Modelling of Network-on-Chip*, *in*: *IEEE Transactions on Computers* 72.*4* (Apr. 2023), Conference Name: IEEE Transactions on Computers, pp. 1215–1222, ISSN: 1557-9956, DOI: 10.1109/TC.2022.3191965.

[BBB17] Ahmed Ben Achballah, Slim Ben Othman, and Slim Ben Saoud, *An extensive review of emerging technology networks on chip proposals*, *in*: *Global Journal of Research in Engineering: F Electrical and Electronic Engineering* 17 (Oct. 2017), pp. 17–40.

[BD02] L. Benini and G. De Micheli, *Networks on chips: a new SoC paradigm*, *in*: *Computer* 35.*1* (Jan. 2002), Conference Name: Computer, pp. 70–78, ISSN: 1558-0814, DOI: 10.1109/2.976921.

[Bec+12] Daniel Ulf Becker et al., *Efficient microarchitecture for network-on-chip routers*, Medium: electronic resource, PhD thesis, 2012, URL: http://purl.stanford.edu/wr368td5072 (visited on 09/14/2023).

[Bel05] Fabrice Bellard, *QEMU, a fast and portable dynamic translator*, *in*: *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, USA: USENIX Association, 2005, p. 41, (visited on 09/19/2023).

[BFM09] Nick Barrow-Williams, Christian Fensch, and Simon Moore, *A communication characterisation of Splash-2 and Parsec*, *in*: *2009 IEEE International Symposium on Workload Characterization (IISWC)*, Oct. 2009, pp. 86–97, DOI: 10.1109/IISWC.2009.5306792.

[BG20] Avik Bose and Prasun Ghosal, *A low latency energy efficient BFT based 3D NoC design with zone based routing strategy*, *in*: *Journal of Systems Architecture* 108 (Sept. 2020), p. 101738, ISSN: 1383-7621, DOI: 10.1016/j.sysarc.2020.101738, URL: https://www.sciencedirect.com/science/article/pii/S1383762120300321 (visited on 09/13/2023).

[Bha21] Vijaya Bhaskar, *A Study of Network-on-Chip Performance*, en, *in*: *2021 Thirteenth International Conference on Contemporary Computing (IC3-2021)*, Noida India: ACM, Aug. 2021, pp. 37–42, ISBN: 978-1-4503-8920-4, DOI: 10.1145/3474124.3474129, URL: https://dl.acm.org/doi/10.1145/3474124.3474129 (visited on 10/29/2022).

[Bha22] Adusumilli Vijaya Bhaskar, *Estimation of Power Consumption in a Network-on-Chip Router*, *in*: *2022 IEEE Delhi Section Conference (DELCON)*, Feb. 2022, pp. 1–7, DOI: 10.1109/DELCON54057.2022.9753477.

[Bie+08] Christian Bienia et al., *The PARSEC benchmark suite: characterization and architectural implications*, *in*: *Proceedings of the 17th international conference on Parallel architectures and compilation techniques*, PACT '08, New York, NY, USA: Association for Computing Machinery, Oct. 2008, pp. 72–81, ISBN: 978-1-60558-282-5, DOI: 10.1145/1454115.1454128, URL: https://doi.org/10.1145/1454115.1454128 (visited on 11/20/2022).

[Bin+11] Nathan Binkert et al., *The gem5 simulator*, *in*: *ACM SIGARCH Computer Architecture News* 39.*2* (Aug. 2011), pp. 1–7, ISSN: 0163-5964, DOI: 10.1145/2024716.2024718, URL: https://doi.org/10.1145/2024716.2024718 (visited on 10/28/2022).

[BJS19] Bahareh Bahrami, Mohammad Ali Jabraeil Jamali, and Shahram Saeidi, *A hierarchical architecture based on traveling salesman problem for hybrid wireless network-on-chip*, en, *in*: *Wireless Networks* 25.*5* (July 2019), pp. 2187–2200, ISSN: 1572-8196, DOI: 10.1007/s11276-017-1641-8, URL: https://doi.org/10.1007/s11276-017-1641-8 (visited on 03/06/2023).

[BKL08] Christian Bienia, Sanjeev Kumar, and Kai Li, *PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on Chip-Multiprocessors*, *in*: *2008 IEEE International Symposium on Workload Characterization*, Sept. 2008, pp. 47–56, DOI: 10.1109/IISWC.2008.4636090.

[BS73] Fischer Black and Myron Scholes, *The Pricing of Options and Corporate Liabilities*, *in*: *Journal of Political Economy* 81.*3* (May 1973), Publisher: The University of Chicago Press, pp. 637–654, ISSN: 0022-3808, DOI: 10.1086/260062, URL: https://www.journals.uchicago.edu/doi/10.1086/260062 (visited on 06/05/2023).

[BV15] Adusumilli Vijaya Bhaskar and Tiruchirai Gopalakrishnan Venkatesh, *A study of the effect of virtual channels on the performance of Network-on-Chip*, *in*: *2015 IEEE Student Conference on Research and Development (SCOReD)*, Dec. 2015, pp. 255–260, DOI: 10.1109/SCORED.2015.7449335.

[Car+11] Aaron Carpenter et al., *A design space exploration of transmission-line links for on-chip interconnect*, *in*: *IEEE/ACM International Symposium on Low Power Electronics and Design*, Aug. 2011, pp. 265–270, DOI: 10.1109/ISLPED.2011.5993647.

[Car+12] Aaron Carpenter et al., *Using Transmission Lines for Global On-Chip Communication*, *in*: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 2.*2* (June 2012), Conference Name: IEEE Journal on Emerging and Selected Topics in Circuits and Systems, pp. 183–193, ISSN: 2156-3365, DOI: 10.1109/JETCAS.2012.2193519.

[Cat+16] Vincenzo Catania et al., *Cycle-Accurate Network on Chip Simulation with Noxim*, *in*: *ACM Trans. on Modeling and Computer Simulation* 27.*1* (Aug. 2016), 4:1–4:25, ISSN: 1049-3301, DOI: 10.1145/2953878, URL: https://doi.org/10.1145/2953878 (visited on 10/28/2022).

[Cat+17] Vincenzo Catania et al., *Improving Energy Efficiency in Wireless Network-on-Chip Architectures*, *in*: *ACM Journal on Emerging Technologies in Computing Systems* 14.*1* (Nov. 2017), 9:1–9:24, ISSN: 1550-4832, DOI: 10.1145/3138807, URL: https://doi.org/10.1145/3138807 (visited on 10/29/2022).

[CGL11] Jie Chen, P. Gillard, and Cheng Li, *Network-on-Chip (NoC) Topologies and Performance: A Review*, *in*: Nov. 2011, URL: https://www.semanticscholar.org/paper/Network-on-Chip-(NoC)-Topologies-and-Performance%3A-A-Chen-Gillard/814e37b6ca6eb8698972894e2639418b0025e922 (visited on 05/12/2023).

[CGP12] Shubhangi D. Chawade, Mahendra A. Gaikwad, and Rajendra M. Patrikar, *Review of XY routing algorithm for network-on-chip architecture*, *in*: *International Journal of Computer Applications* 43.*21* (2012), Publisher: International Journal of Computer Applications, 244 5 th Avenue,# 1526, New . . ., pp. 975–8887.

[CHE11] Trevor E. Carlson, Wim Heirman, and Lieven Eeckhout, *Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation*, *in*: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, New York, NY, USA: Association for Computing Machinery, Nov. 2011, pp. 1–12, ISBN: 978-1-

4503-0771-0, DOI: 10.1145/2063384.2063454, URL: https://doi.org/10.1145/2063384.2063454 (visited on 11/20/2022).

[CL14] Jenhui Chen and Yen-Han Lai, *A study of CSMA-based and token-based wireless interconnects network-on-chip*, *in*: *2014 IEEE International Conference on Communiction Problem-solving*, Dec. 2014, pp. 205–209, DOI: 10.1109/ICCPS.2014.7062254.

[CL20] Yuechen Chen and Ahmed Louri, *An Approximate Communication Framework for Network-on-Chips*, *in*: *IEEE Transactions on Parallel and Distributed Systems* 31.*6* (June 2020), Conference Name: IEEE Transactions on Parallel and Distributed Systems, pp. 1434–1446, ISSN: 1558-2183, DOI: 10.1109/TPDS.2020.2968068.

[CML12] Érika Cota, Alexandre de Morais Amory, and Marcelo Soares Lubaszewski, *Reliability, Availability and Serviceability of Networks-on-Chip*, en, *in*: *Reliability, Availability and Serviceability of Networks-on-Chip*, ed. by Érika Cota, Alexandre de Morais Amory, and Marcelo Soares Lubaszewski, Boston, MA: Springer US, 2012, pp. 1–9, ISBN: 978-1-4614-0791-1, DOI: 10.1007/978-1-4614-0791-1_1, URL: https://doi.org/10.1007/978-1-4614-0791-1_1 (visited on 05/12/2023).

[Cop+04] M. Coppola et al., *Spidergon: a novel on-chip communication network*, *in*: *2004 International Symposium on System-on-Chip, 2004. Proceedings.* Nov. 2004, pp. 15–, DOI: 10.1109/ISSOC.2004.1411133.

[Dai+15] Peng Dai et al., *A study of a wire–wireless hybrid NoC architecture with an energy-proportional multicast scheme for energy efficiency*, en, *in*: *Computers & Electrical Engineering* 45 (July 2015), pp. 402–416, ISSN: 0045-7906, DOI: 10.1016/j.compeleceng.2015.06.005, URL: https://www.sciencedirect.com/science/article/pii/S0045790615002098 (visited on 03/06/2023).

[Deb+13] Sujay Deb et al., *Design of an Energy-Efficient CMOS-Compatible NoC Architecture with Millimeter-Wave Wireless Interconnects*, *in*: *IEEE Transactions on Computers* 62.*12* (Dec. 2013), Conference Name: IEEE Transactions on Computers, pp. 2382–2396, ISSN: 1557-9956, DOI: 10.1109/TC.2012.224.

[DM14] Armen Dzhagaryan and Aleksandar Milenkovic, *Impact of Thread and Frequency Scaling on Performance and Energy in Modern Multicores: A Measurement-based Study*, *in*: Mar. 2014, DOI: 10.1145/2638404.2638473.

[DT04] William James Dally and Brian Patrick Towles, *Principles and Practices of Interconnection Networks*, en, Google-Books-ID: Rz7pCY8gIq0C, Elsevier, Mar. 2004, ISBN: 978-0-08-049780-8.

[DYN03] José Duato, Sudhakar Yalamanchili, and Lionel M. Ni, *Interconnection networks: an engineering approach*, en, Rev. printing, San Francisco, CA: Morgan Kaufmann, 2003, ISBN: 978-1-55860-852-8.

[Edg30] Lilienfeld Julius Edgar, *Method and apparatus for controlling electric currents*, US1745175A, Jan. 1930, URL: https://patents.google.com/patent/US1745175A/en (visited on 05/31/2023).

[Fas+21] Ayodeji Ireti Fasiku et al., *A Centralized Token-based Medium Access Control Mechanism for Wireless Network-on-Chip*, *in*: *2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST)*, June 2021, pp. 102–107, DOI: 10.1109/ICAICST53116.2021.9497802.

[Fra+21] Antonio Franques et al., *WiDir: A Wireless-Enabled Directory Cache Coherence Protocol*, *in*: *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2021, pp. 304–317, DOI: 10.1109/HPCA51647.2021.00034.

[GD21] Sri Harsha Gade and Sujay Deb, *A Novel Hybrid Cache Coherence with Global Snooping for Many-core Architectures*, *in*: *ACM Transactions on Design Automation of Electronic Systems* 27.*1* (Sept. 2021), 2:1–2:31, ISSN: 1084-4309, DOI: 10.1145/3462775, URL: https://doi.org/10.1145/3462775 (visited on 10/28/2022).

[GGM23] B. Gomatheeshwari, K. Gopi, and Ajisha Mathias, *Low-complex resource mapping heuristics for mobile and iot workloads on NoC-HMPSoC architecture*, *in*: *Microprocessors and Microsystems* 98 (Apr. 2023), p. 104802, ISSN: 0141-9331, DOI: 10.1016/j.micpro.2023.104802, URL: https://www.sciencedirect.com/science/article/pii/S0141933123000480 (visited on 09/10/2023).

[GNR08] M.H. Ghadiry, M. Nadi, and D. Rahmati, *New approach to calculate energy on NoC*, *in*: *2008 International Conference on Computer and Communication Engineering*, May 2008, pp. 1098–1104, DOI: 10.1109/ICCCE.2008.4580777, URL: https://ieeexplore.ieee.org/abstract/document/4580777 (visited on 02/21/2024).

[Ham+19] Mohamad Hamieh et al., *A new interconnect method for radio frequency intra-chip communications using transistors-based distributed access*, en, *in*: *Microwave and Optical Technology Letters* 61.*2* (2019), _eprint: https://onlinelibrary.wi pp. 297–302, ISSN: 1098-2760, DOI: 10.1002/mop.31590, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mop.31590 (visited on 02/21/2024).

[HCE12] Wim Heirman, Trevor Carlson, and Lieven Eeckhout, *Sniper: scalable and accurate parallel multi-core simulation*, eng, *in*: *8th International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems, Abstracts*, High-Performance, Embedded Architecture, and Compilation Network of Excellence (HiPEAC), 2012, pp. 91–94, ISBN: 978-90-382-1987-5, URL: http://hdl.handle.net/1854/LU-2968322 (visited on 11/27/2023).

[HUA] HUAWEI HUAWEI, *HUAWEI P60 Pro - HUAWEI Global*, en, URL: https://consumer.huawei.com/en/phones/p60-pro/ (visited on 09/10/2023).

[Int] Intel Intel, *Intel® Core™ i9 Processor - Features, Benefits and FAQs*, en, URL: https://www.intel.com/content/www/us/en/products/details/processors/core/i9.html (visited on 09/12/2023).

[Jai+15] Kunj Jain et al., *Problems encountered in various arbitration techniques used in NOC router: A survey*, *in*: *2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)*, Jan. 2015, pp. 62–67, DOI: 10.1109/EDCAV.2015.7060540.

[Jia+13] Nan Jiang et al., *A detailed and flexible cycle-accurate Network-on-Chip simulator*, *in*: *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Apr. 2013, pp. 86–96, DOI: 10.1109/ISPASS.2013.6557149.

[Kar+12] Ammar Karkar et al., *Surface wave communication system for on-chip and off-chip interconnects*, *in*: *Proceedings of the Fifth International Workshop on Network on Chip Architectures*, NoCArc '12, New York, NY, USA: Association for Computing Machinery, 2012, pp. 11–16, ISBN: 978-1-4503-1540-1, DOI: 10.1145/2401716.2401720, URL: https://dl.acm.org/doi/10.1145/2401716.2401720 (visited on 06/08/2023).

[Kar+16] Ammar Karkar et al., *A Survey of Emerging Interconnects for On-Chip Efficient Multicast and Broadcast in Many-Cores*, *in*: *IEEE Circuits and Systems Magazine* 16.*1* (2016), Conference Name: IEEE Circuits and Systems Magazine, pp. 58–72, ISSN: 1558-0830, DOI: 10.1109/MCAS.2015.2510199.

[Ken53] David G. Kendall, *Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain*, *in*: *The Annals of Mathematical Statistics* 24.*3* (Sept. 1953), Publisher: Institute of Mathematical Statistics, pp. 338–354, ISSN: 0003-4851, 2168-8990, DOI: 10.1214/aoms/1177728975, URL: https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-24/issue-3/Stochastic-Processes-Occurring-in-the-Theory-of-Queues-and-their/10.1214/aoms/1177728975.full (visited on 08/31/2023).

[Kha+18] Sarzamin Khan et al., *Comparative analysis of network-on-chip simulation tools*, en, *in*: *IET Computers & Digital Techniques* 12.*1* (2018), _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/iet-cdt.2017.0068, pp. 30–38, ISSN: 1751-861X, DOI: 10.1049/iet-cdt.2017.0068, URL: https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-cdt.2017.0068 (visited on 10/28/2022).

[KJL13] Abbas Eslami Kiasari, Axel Jantsch, and Zhonghai Lu, *Mathematical formalisms for performance evaluation of networks-on-chip*, *in*: *ACM Computing Surveys* 45.*3* (July 2013), 38:1–38:41, ISSN: 0360-0300, DOI: 10.1145/2480741.2480755, URL: https://doi.org/10.1145/2480741.2480755 (visited on 10/29/2022).

[KLJ13] Abbas Eslami Kiasari, Zhonghai Lu, and Axel Jantsch, *An Analytical Latency Model for Networks-on-Chip*, *in*: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21.*1* (Jan. 2013), Conference Name: IEEE

Transactions on Very Large Scale Integration (VLSI) Systems, pp. 113–123, ISSN: 1557-9999, DOI: 10.1109/TVLSI.2011.2178620.

[KPJ08] Amit Kumar, Li-Shiuan Peh, and Niraj K. Jha, *Token flow control*, *in*: *2008 41st IEEE/ACM International Symposium on Microarchitecture*, ISSN: 2379-3155, Nov. 2008, pp. 342–353, DOI: 10.1109/MICRO.2008.4771803.

[Kra+23] Ibrahim Krayem et al., *Analytical Model for Performance Evaluation of Token-Passing Based WiNoCs*, *in*: *IEEE Design & Test* (2023), Conference Name: IEEE Design & Test, pp. 1–1, ISSN: 2168-2364, DOI: 10.1109/MDAT.2023.3309730.

[Kum+02] S. Kumar et al., *A network on chip architecture and design methodology*, *in*: *Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design. ISVLSI 2002*, Apr. 2002, pp. 117–124, DOI: 10.1109/ISVLSI.2002.1016885.

[Le+17] Tung Thanh Le et al., *Optimizing the heterogeneous network on-chip design in manycore architectures*, *in*: *2017 30th IEEE International System-on-Chip Conference (SOCC)*, ISSN: 2164-1706, Sept. 2017, pp. 184–189, DOI: 10.1109/SOCC.2017.8226033.

[Lee+13] Junghee Lee et al., *Do we need wide flits in Networks-on-Chip?*, *in*: *2013 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, ISSN: 2159-3477, Aug. 2013, pp. 2–7, DOI: 10.1109/ISVLSI.2013.6654614.

[Lee22] Jaechul Lee, *Approximate communication techniques exploration for efficient nano-photonics interconnects*, en, PhD thesis, Université Rennes 1, Dec. 2022, URL: https://theses.hal.science/tel-03958287 (visited on 05/09/2023).

[Liu+11] Weichen Liu et al., *A NoC Traffic Suite Based on Real Applications*, *in*: *2011 IEEE Computer Society Annual Symposium on VLSI*, ISSN: 2159-3477, July 2011, pp. 66–71, DOI: 10.1109/ISVLSI.2011.49.

[LJL13] Yanhua Liu, Jie Jin, and Zongsheng Lai, *A dynamic adaptive arbiter for Network-on-Chip*, *in*: *Informacije MIDEM* 43 (Jan. 2013), pp. 111–118.

[Luo18] Jiating Luo, *Architectural and Protocol Exploration for 3D Optical Network-on-Chip*, en, PhD thesis, Université de Rennes 1 [UR1], July 2018, URL: https://hal.inria.fr/tel-01956255 (visited on 10/28/2022).

[Man+19] Sumit K. Mandal et al., *Analytical Performance Models for NoCs with Multiple Priority Traffic Classes*, *in*: *ACM Transactions on Embedded Computing Systems* 18.*5s* (Oct. 2019), 52:1–52:21, ISSN: 1539-9087, DOI: 10.1145/3358176, URL: https://doi.org/10.1145/3358176 (visited on 10/29/2022).

[Man+21a] Sumit K. Mandal et al., *Analytical Performance Modeling of NoCs under Priority Arbitration and Bursty Traffic*, *in*: *IEEE Embedded Systems Letters* 13.*3* (Sept. 2021), Conference Name: IEEE Embedded Systems Letters, pp. 98–101, ISSN: 1943-0671, DOI: 10.1109/LES.2020.3013003.

[Man+21b] Sumit K. Mandal et al., *Theoretical Analysis and Evaluation of NoCs with Weighted Round-Robin Arbitration*, *in*: *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, ISSN: 1558-2434, Nov. 2021, pp. 1–9, DOI: 10.1109/ICCAD51958.2021.9643448.

[Man+23] Sumit K. Mandal et al., *Fast Performance Analysis for NoCs With Weighted Round-Robin Arbitration and Finite Buffers*, *in*: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 31.*5* (May 2023), Conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 670–683, ISSN: 1557-9999, DOI: 10.1109/TVLSI.2023.3250662, URL: https://ieeexplore.ieee.org/abstract/document/10072009 (visited on 02/12/2024).

[Man21] Sumit Mandal, *Network-on-Chip (NoC) Performance Analysis and Optimization for Deep Learning Applications*, en_US, Accepted: 2021-12-08T20:59:34Z, PhD thesis, July 2021, URL: https://minds.wisconsin.edu/handle/1793/82502 (visited on 10/29/2022).

[Mar+14] André L. M. Martins et al., *A method for NoC-based MPSoC energy consumption estimation*, *in*: *2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec. 2014, pp. 427–430, DOI: 10.1109/ICECS.2014.7050013, URL: https://ieeexplore.ieee.org/abstract/document/7050013 (visited on 12/14/2023).

[Mat21] Oumaima Matoussi, *NoC Performance Model for Efficient Network Latency Estimation*, *in*: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, ISSN: 1558-1101, Feb. 2021, pp. 994–999, DOI: 10.23919/DATE51398.2021.9474101.

[Mea66] C.A. Mead, *Schottky barrier gate field effect transistor*, *in*: *Proceedings of the IEEE* 54.*2* (Feb. 1966), Conference Name: Proceedings of the IEEE, pp. 307–308, ISSN: 1558-2256, DOI: 10.1109/PROC.1966.4661.

[Mer+22] Romain Mercier et al., *BiSuT: A NoC-Based Bit-Shuffling Technique for Multiple Permanent Faults Mitigation*, *in*: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41.*7* (July 2022), Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 2276–2289, ISSN: 1937-4151, DOI: 10.1109/TCAD.2021.3101406.

[MIG15] Naseef Mansoor, Pratheep Joe Sullivai Iruthayaraj, and Amlan Ganguly, *Design Methodology for a Robust and Energy-Efficient Millimeter-Wave Wireless Network-on-Chip*, *in*: *IEEE Transactions on Multi-Scale Computing Systems* 1.*1* (Jan. 2015), Conference Name: IEEE Transactions on Multi-Scale Computing Systems, pp. 33–45, ISSN: 2332-7766, DOI: 10.1109/TMSCS.2015.2478439.

[MMH21] Misbah Manzoor, Roohie Naaz Mir, and Najeeb-ud-Din Hakim, *A Review of Design Approaches for Enhancing the Performance of NoCs at Communication Centric Level*, en, *in*: *Scalable Computing: Practice and Experience* 22.*3* (Nov. 2021), Number: 3, pp. 347–364, ISSN: 1895-1767, DOI: 10.12694/scpe.v22i3.1896, URL: //www.scpe.org/index.php/scpe/article/view/1896 (visited on 09/20/2023).

[MSP23] Ashish Mulajkar, Sanjeet K. Sinha, and Govind Singh Patel, *Emerging trends in network on chip design for low latency and enhanced throughput applications*, *in*: *AIP Conference Proceedings* 2800.*1* (Sept. 2023), p. 020120, ISSN: 0094-243X, DOI: 10.1063/5.0162952, URL: https://doi.org/10.1063/5.0162952 (visited on 09/12/2023).

[MV23] Loren Merritt and Rahul Vanam, *X264: A HIGH PERFORMANCE H.264/AVC ENCODER*, *in*: (Oct. 2023).

[OBM10] Umit Y. Ogras, Paul Bogdan, and Radu Marculescu, *An Analytical Approach for Network-on-Chip Performance Analysis*, *in*: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29.*12* (Dec. 2010), Conference Name: IEEE Transactions on Computer-Aided Design

of Integrated Circuits and Systems, pp. 2001–2013, ISSN: 1937-4151, DOI: 10.1109/TCAD.2010.2061613.

[OCa+02] Liadan O'Callaghan et al., *High-Performance Clustering of Streams and Large Data Sets*, *in*: (Jan. 2002).

[Oka94] Sōgo Okamura, *History of Electron Tubes*, en, Google-Books-ID: VHFyngmO95YC, IOS Press, 1994, ISBN: 978-90-5199-145-1.

[Ort+19] Joel Ortiz et al., *Multi-carrier spread-spectrum transceiver for WiNoC*, *in*: *Proceedings of the 13th IEEE/ACM International Symposium on Networks-on-Chip*, NOCS '19, New York, NY, USA: Association for Computing Machinery, Oct. 2019, pp. 1–2, ISBN: 978-1-4503-6700-4, DOI: 10.1145/3313231.3352373, URL: https://doi.org/10.1145/3313231.3352373 (visited on 10/29/2022).

[Ort09] John W. Orton, *Semiconductors and the Information Revolution: Magic Crystals that made IT Happen*, en, Google-Books-ID: 6YLL9197NfMC, Academic Press, June 2009, ISBN: 978-0-08-096390-7.

[OSR19] Joel Ortiz Sosa, Olivier Sentieys, and Christian Roland, *Adaptive Transceiver for Wireless NoC to Enhance Multicast/Unicast Communication Scenarios*, *in*: *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, ISSN: 2159-3477, July 2019, pp. 592–597, DOI: 10.1109/ISVLSI.2019.00111.

[Pan+05] Partha Pratim Pande et al., *Performance evaluation and design trade-offs for network-on-chip interconnect architectures*, *in*: *IEEE Transactions on Computers* 54.*8* (Aug. 2005), Conference Name: IEEE Transactions on Computers, pp. 1025–1040, ISSN: 1557-9956, DOI: 10.1109/TC.2005.134.

[Pan+09] Partha Pratim Pande et al., *Hybrid wireless Network on Chip: a new paradigm in multi-core design*, *in*: *Proceedings of the 2nd International Workshop on Network on Chip Architectures*, NoCArc '09, New York, NY, USA: Association for Computing Machinery, 2009, pp. 71–76, ISBN: 978-1-60558-774-5, DOI: 10.1145/1645213.1645230, URL: https://dl.acm.org/doi/10.1145/1645213.1645230 (visited on 09/09/2023).

[PF07] Vasilis F. Pavlidis and Eby G. Friedman, *3-D Topologies for Networks-on-Chip*, *in*: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 15.*10* (Oct. 2007), Conference Name: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 1081–1090, ISSN: 1557-9999, DOI: 10.1109/TVLSI.2007.893649.

[Pha18] Van Dung Pham, *Architectural exploration of network Interface for energy efficient 3D optical network-on-chip*, These de doctorat, Rennes 1, Dec. 2018, URL: https://www.theses.fr/2018REN1S076 (visited on 11/16/2023).

[PJD22] Sudeep Pasricha, John Jose, and Sujay Deb, *Electronic, Wireless, and Photonic Network-on-Chip Security: Challenges and Countermeasures*, *in*: *IEEE Design & Test* 39.*6* (Dec. 2022), Conference Name: IEEE Design & Test, pp. 90–98, ISSN: 2168-2364, DOI: 10.1109/MDAT.2022.3203017.

[QD02] Sean Quinlan and Sean Dorward, *Venti: A New Approach to Archival Data Storage*, en, *in*: 2002, URL: https://www.usenix.org/conference/fast-02/venti-new-approach-archival-data-storage (visited on 10/27/2023).

[Rup22] Kari Rupp, *microprocessor-trend-data/50yrs*, en, 2022, URL: https://github.com/karlrupp/microprocessor-trend-data (visited on 10/30/2022).

[Sam+23] Werner Florian Samayoa et al., *A Survey on FPGA-Based Heterogeneous Clusters Architectures*, *in*: *IEEE Access* 11 (2023), Conference Name: IEEE Access, pp. 67679–67706, ISSN: 2169-3536, DOI: 10.1109/ACCESS.2023.3288431.

[Sil+23] Cristina Silvano et al., *A Survey on Deep Learning Hardware Accelerators for Heterogeneous HPC Platforms*, arXiv:2306.15552 [cs], June 2023, DOI: 10.48550/arXiv.2306.15552, URL: http://arxiv.org/abs/2306.15552 (visited on 12/14/2023).

[SPE] SPEC, *SPEC Benchmarks and Tools*, URL: https://www.spec.org/benchmarks.html (visited on 02/25/2024).

[SW00] Neil T. Spring and David Wetherall, *A protocol-independent technique for eliminating redundant network traffic*, *in*: *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '00, New York, NY, USA: Association for Comput-

ing Machinery, 2000, pp. 87–95, ISBN: 978-1-58113-223-6, DOI: 10.1145/347059.347408, URL: https://dl.acm.org/doi/10.1145/347059.347408 (visited on 10/27/2023).

[Tau23] Yuan Taur, *Evolution of the MOSFET*, *in*: *75th Anniversary of the Transistor*, Conference Name: 75th Anniversary of the Transistor, IEEE, 2023, pp. 101–114, ISBN: 978-1-394-20245-4, DOI: 10.1002/9781394202478.ch10, URL: https://ieeexplore.ieee.org/document/10194391 (visited on 09/15/2023).

[Tim+18] Xavier Timoneda et al., *Engineer the Channel and Adapt to it: Enabling Wireless Intra-Chip Communication*, Dec. 2018.

[TSJ] Konstantinos Tatas, Kostas Siozios Dimitrios Soudris, and Axel Jantsch, *Designing 2D and 3D Network-on-Chip Architectures*, en, URL: https://link.springer.com/book/10.1007/978-1-4614-4274-5 (visited on 10/29/2022).

[TSM] TSMC, *Logic Technology - Taiwan Semiconductor Manufacturing Company Limited*, en, URL: http://www.tsmc.com/english/dedicatedFoundry/technology/logic (visited on 09/29/2023).

[Vid+12] Anuroop Vidapalapati et al., *NoC architectures with adaptive Code Division Multiple Access based wireless links*, *in*: *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, ISSN: 2158-1525, May 2012, pp. 636–639, DOI: 10.1109/ISCAS.2012.6272112.

[Vij+14] Vineeth Vijayakumaran et al., *CDMA Enabled Wireless Network-on-Chip*, *in*: *ACM Journal on Emerging Technologies in Computing Systems* 10.*4* (2014), 28:1–28:20, ISSN: 1550-4832, DOI: 10.1145/2536778, URL: https://dl.acm.org/doi/10.1145/2536778 (visited on 09/17/2023).

[Viv21] Vivet, E. Guthmuller, Y. Thonnart et al., *IntAct: A 96-Core Processor With Six Chiplets 3D-Stacked on an Active Interposer With Distributed Interconnects and Integrated Power Management*, *in*: *IEEE Journal of Solid-State Circuits* 56.*1* (Jan. 2021), Conference Name: IEEE Journal of Solid-State Circuits, pp. 79–97, ISSN: 1558-173X, DOI: 10.1109/JSSC.2020.3036341.

[Whi05] Turner Whitted, *An improved illumination model for shaded display*, *in*: *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA: Association for Computing Machinery, 2005, 4–es, ISBN: 978-1-4503-7833-8, DOI: 10.1145/1198555.1198743, URL: https://dl.acm.org/doi/10.1145/1198555.1198743 (visited on 10/27/2023).

[WJ14] Shuai Wang and Tao Jin, *Wireless network-on-chip: a survey*, *in*: *The Journal of Engineering* 2014 (Apr. 2014), DOI: 10.1049/joe.2013.0209.

[WNL17] Sebastian Werner, Javier Navaridas, and Mikel Luján, *A Survey on Optical Network-on-Chip Architectures*, *in*: *ACM Computing Surveys* 50.*6* (2017), 89:1–89:37, ISSN: 0360-0300, DOI: 10.1145/3131346, URL: https://dl.acm.org/doi/10.1145/3131346 (visited on 06/08/2023).

[Woo+95] Steven Cameron Woo et al., *The SPLASH-2 programs: characterization and methodological considerations*, *in*: *ACM SIGARCH Computer Architecture News* 23.*2* (1995), pp. 24–36, ISSN: 0163-5964, DOI: 10.1145/225830.223990, URL: https://dl.acm.org/doi/10.1145/225830.223990 (visited on 05/29/2023).

[Yua+23] Jun Yuan et al., *High Performance 5G Mobile SOC Productization with 4nm EUV Fin-FET Technology*, *in*: *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, ISSN: 2158-9682, June 2023, pp. 1–2, DOI: 10.23919/VLSITechnologyandCir57934.2023.10185435.

[Zha+10] Youhui Zhang et al., *A Performance Analytical Approach Based on Queuing Model for Network-on-Chip*, *in*: *2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming*, ISSN: 2168-3042, Dec. 2010, pp. 354–359, DOI: 10.1109/PAAP.2010.46.

[Zha+11] Dan Zhao et al., *Design of multi-channel wireless NoC to improve on-chip communication capacity*, *in*: *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '11, New York, NY, USA: Association for Computing Machinery, 2011, pp. 177–184, ISBN: 978-1-4503-0720-8, DOI: 10.1145/1999946.1999975, URL: https://dl.acm.org/doi/10.1145/1999946.1999975 (visited on 09/18/2023).

[ZW08] Dan Zhao and Yi Wang, *SD-MAC: Design and Synthesis of a Hardware-Efficient Collision-Free QoS-Aware MAC Protocol for Wireless Network-on-Chip*, *in*: *Computers, IEEE Transactions on* 57 (Oct. 2008), pp. 1230–1245, DOI: 10.1109/TC.2008.86.

[ZWG13] Ying Zhang, Ning Wu, and Fen Ge, *The Co-Design of Test Structure and Test Data Transfer Mode for 2D-Mesh NoC*, en, *in*: *IAENG Transactions on Engineering Technologies: Special Edition of the World Congress on Engineering and Computer Science 2011*, ed. by Haeng Kon Kim, Sio-Iong Ao, and Burghard B. Rieger, Lecture Notes in Electrical Engineering, Dordrecht: Springer Netherlands, 2013, pp. 171–184, ISBN: 978-94-007-4786-9, DOI: 10.1007/978-94-007-4786-9_14, URL: https://doi.org/10.1007/978-94-007-4786-9_14 (visited on 09/13/2023).

COLLEGE DOCTORAL BRETAGNE
MATHS, TELECOMS
INFORMATIQUE, SIGNAL
SYSTEMES, ELECTRONIQUE

Université de Rennes

Titre : Méthodes pour l'exploration rapide d'architectures multicœurs basées sur les réseaux sur puce avec des technologies émergentes

Mot clés : Réseau sur puce sans fil, Analyse de performance, Latence, Théorie des files d'attente, Application benchmark

Résumé : Les progrès récents en matière d'intégration technologique ont introduit de nouvelles interconnexions sur puce, telles que les réseaux sur puce sans fil (WiNoCs), ce qui rend l'espace de conception trop vaste pour être exploré efficacement à l'aide de simulateurs standard qui prennent beaucoup de temps. Nous proposons un modèle analytique basé sur la théorie des files d'attente pour évaluer la latence des interconnexions de l'architecture manycore. Nous considérons une interconnexion hybride qui utilise des NoC électriques et sans fil pour les communications intra- et inter-clusters. Les résultats démontrent que le modèle proposé réduit de manière significative le temps d'exécution de la simulation jusqu'à 500× tout en maintenant un taux d'erreur inférieur à 5 % par rapport au simulateur Noxim précis au niveau du cycle. Ensuite, nous proposons une méthode pour accélérer l'analyse des performances du NoC en utilisant le fenêtrage des traces d'application. Cette technique de fractionnement et de fusion des fenêtres améliore la précision des prévisions de performance tout en réduisant la complexité des calculs. Les résultats expérimentaux confirment l'efficacité de cette méthode avec différents types de trafic, réduisant de manière significative les erreurs et améliorant le temps d'exécution.

Title: Methods for fast exploration of manycore architectures based Network-on-Chip with Emerging Technologies

Keywords: Wireless Network-on-Chip, Performance Analysis, Latency, Queuing theory, Benchmark application

Abstract: Recent advances in technology integration have introduced new on-chip interconnects, such as wireless Network-on Chips (WiNoCs), making the design space too large to be efficiently explored with time-consuming standard simulators. We propose an analytical model based on queuing theory to evaluate the latency of manycore architecture interconnects. We consider a hybrid interconnection that utilizes electrical and wireless NoCs for both intra- and inter-cluster communications. The results demonstrate that our proposed model significantly reduces the simulation execution time by up to 500× while maintaining an error rate of less than 5% compared to the Noxim cycle-accurate simulator. Next, we propose a method for accelerating NoC performance analysis using application trace windowing. This window splitting and merging technique improves the accuracy of performance predictions while reducing computational complexity. Experimental results confirm the effectiveness of this method with different types of traffic, significantly reducing errors and improving execution time.